Fast computation of eikonal and transport equations on graphics processing units computer architectures

Marcus Noack¹ and Tor Gillberg²

**ABSTRACT**

Eikonal models have been widely used for traveltime computations in the field of seismic imaging, but they are often criticized for having low accuracy and poor resolution of the output image. Including amplitude information can provide higher model resolution and accuracy of the images. We have developed a new approach for computing eikonal traveltimes and amplitudes, and we implemented it for multicore computer processing unit and graphics processing unit computer architectures. Traveltimes and amplitudes are computed simultaneously in iterations of the 3D velocity model. This is achieved by extending a fast sweeping method by computing amplitudes directly after the traveltimes with upwind numerical difference schemes. By performing the extra computations simultaneously with the traveltimes, the additional cost for the amplitude and raypaths is low. We tested our method on synthetic 3D data sets to compute traveltimes, amplitudes, and raypaths, from which the Helmholtz Green’s function was assembled. Using a grid of 124³ nodes, the computations were performed in less than 1 s. The proposed method could work as a feasible alternative to full waveform modeling in seismic applications, which suffers from demanding computations because it requires several orders of magnitudes shorter computing times.

**INTRODUCTION**

Wave propagation phenomena play a central role in the fields of physics, environmental research, and medical imaging. The fast and accurate numerical modeling of these phenomena is hence an important task. In the field of geophysics, the availability of an accurate and fast wave modeling method can result in higher resolved images of the subsurface in shorter computing time, therefore allowing for lower costs and higher confidence for oil and gas exploration and new possibilities in geological research. Furthermore, a faster and more accurate wave modeling method can improve risk assessment methodologies for earthquakes and volcanoes.

Computing the direct solution of the wave equation is a difficult and time-consuming process (Mehra et al., 2012). Nonreflecting boundary conditions and imposed time stepping further increase the required computational time. Taking the Helmholtz equation as the static part of the wave equation is a frequently used simplification (Haber and MacLachlan, 2011). However, using the Helmholtz equation does not sufficiently decrease the computational time, and the solution process remains complicated (Haber and MacLachlan, 2011). A widely used approach to avoid these drawbacks is to use the Helmholtz Green’s function for a point source (Luo et al., 2012). First-arrival traveltimes and their amplitudes are needed to estimate the Helmholtz Green’s function. These traveltimes and amplitudes are computed as solutions of the eikonal and transport equations, respectively. We introduce methods that solve both equations fast and accurately.

**BACKGROUND**

Several finite-difference schemes have been introduced in past years to solve the eikonal equation. Vidale (1988) proposes the expanding square method in 2D and extends it later for applications in 3D with the expanding box method (Vidale, 1990). In this method, traveltimes are updated on the surface of a box around the source. When all nodes in this structure are updated, the adjacent nodes are used to build a larger box. The expanding box method is still widely used, but it sometimes suffers in accuracy for complex velocity models due to the fact that the box shape does not in general resemble a wavefront (Rawlinson and Sambridge, 2005).
The fast marching method (Sethian and Popovici, 1999) instead tracks a general wavefront as it moves through the domain. The fast marching method is more stable, more accurate, and faster than the expanding box method. To track a general wave shape, nodes are divided into three groups: alive, close, and far (Rawlinson and Sambridge, 2005). All nodes in the wavefront, called close nodes, are sorted depending on their traveltimes. The close node with the smallest traveltime is relabeled as alive, and thereby evolving the position of the front. Traveltimes of nodes in the vicinity of the relabeled node that are not alive are recomputed and labeled as close. The procedure continues until all nodes are alive. The method handles sharp velocity contrasts well, and the accuracy is sufficient for most applications (Sethian and Popovici, 1999). The computational costs are of the order $O(N \log N$, where the $N \log N$ term is due to the ordering. Therefore, the feasibility of on-the-fly modeling decreases with data sets of increasing size. Even though there have been some attempts of parallelization (Herrmann, 2003), the method remains sequential by nature because the front passes only one node at a time. This is a drawback because processing units are becoming cheaper rather than faster (Sutter, 2005).

The fast sweeping method (Zhao, 2004) sweeps through the 3D domain in eight alternating directions. When sweeping through the grid in one direction, traveltimes are computed for wavefronts traveling through the grid in that direction. The fast sweeping method is an iterative method that sweeps through the domain until the solution converges. The method has a computational cost of order $O(N)$ and can be parallelized to some degree, which makes it appropriate for large data sets (Zhao, 2007). Using the fast sweeping method for the amplitudes amounts to sweeping separately for the traveltimes and amplitudes until convergence (Luo et al., 2012).

A recently developed method, called the 3D parallel marching method (3DPMM) (Gillberg et al., 2012b) is a parallel sweeping method, extending the parallel marching method to three dimensions (Weber et al., 2008). In 3DPMM, a stencil shaped like a tetrahedron is used instead of the standard Godunov “box” stencil. The sweeps in 3DPMM are in different directions compared with traditional fast sweeping method sweeps. The result is a significant gain in parallelization opportunity compared with the traditional fast sweeping method. On a grid with $N^3$ nodes, $N^2$ nodes can be computed in parallel. If the parallel computing opportunities are used, the computational time can be decreased by several orders of magnitude. By making use of diagonal stencils, the accuracy increases compared with the “box stencils” (Jeong and Whitaker, 2008; Gillberg et al., 2012a), independent of the algorithm used. Therefore, the 3DPMM method offers a very fast and accurate solution of the eikonal equation.

Some methods try to combine aspects of sweeping and front tracking algorithms. This marriage of algorithms creates very efficient, but also more complicated algorithms allowing for parallel implementations. For instance, the semiordered list of active subdomains (SOLAS) method (Gillberg et al., 2014) uses the 3DPMM method on selected parts of the full domain. We expect the methodology introduced in this paper to be applicable to algorithms other than pure 3DPMM.

A NEW METHOD FOR TRAVELTIME AND AMPLITUDE COMPUTATION

We propose a solver for the Helmholtz Green’s function that takes full advantage of parallel computer architectures by extending the 3DPMM method. The main idea is to solve the transport equation simultaneously with the eikonal equation. Our key observation is that for any given node, the upwind amplitude data are in the vicinity of the upwind traveltime data. The data used to compute new traveltimes can be reused immediately to compute amplitudes. In our novel approach, when the traveltime of one node is updated, we immediately estimate the amplitude for the same node. Therefore both amplitude and time values are updated in the same sweep through the domain. Because we only use data from the last update planes, the computations of all nodes in the current update plane are independent. In this way, entire planes can be updated in parallel. To estimate new traveltimes, the entry location of the ray yielding the new traveltime is computed directly. The result is a method that calculates first-arrival traveltimes, amplitudes, and all rays simultaneously and efficiently on multiprocessor architectures.

The remainder of the paper is organized as follows: The theory section gives an overview of the basic methods and the main principles of the algorithm, including a summary of the 3DPMM method and a section detailing the computation of raypaths and entrance locations. This is followed by a section describing two stencils for the amplitude estimation. The proposed method was applied to three synthetic examples to show the functionality of the novel solver for the Helmholtz Green’s function as described in the results section.

THEORY

3DPMM was proposed by Gillberg et al. (2012b) and is used to model geologic folds. A folded layer is modeled as the isosurface of the solution $T(x)$ to the following static Hamilton-Jacobi equation:

$$F[\nabla T(x)] + \psi(a, \nabla T(x)) = 1, \quad T(x) = 0 \text{ on } \Gamma,$$

where $\Gamma$ is a reference horizon, $\| \cdot \|$ is the Euclidean norm, $\langle \cdot, \cdot \rangle$ is the inner product of the Euclidean space, $F$ and $\Psi$ are scalars, and $a$ is a unit vector pointing in the symmetry line of the fold. The eikonal equation, commonly used to estimate first-arrival traveltimes of seismic waves (Crandall and Lions, 1983), is formulated as

$$\|\nabla T(x)\| = s(x), \quad T(x) = \Psi(x) \forall x \in \Omega.$$

The isotropic eikonal equation is a special case of the static Hamilton-Jacobi equation. The 3DPMM method can therefore be used unmodified to estimate solutions to the eikonal equation. For the description of 3DPMM, consider a regular box grid with nodal values $T_{ijk}$ being approximations of first-arrival traveltimes. We assume that some nodes are given initially and all other nodes are set to infinity. Because we are solving for first arrivals, we solve for the minimum obtained value of $T$. Nodes are updated with a “pyramid-shaped” stencil (Gillberg et al., 2012b), as illustrated in Figure 1a, in which the top node $T_{ijk}$ is the one being updated. The stencil contains 8 tetrahedron three-point stencils, 16 two-node stencils, and 9 one-node stencils, as further detailed in Gillberg (2013). The smallest estimation for the update node of these 33 stencils is used according to Huygen’s principle (for an explanation of the systematic application of Huygen’s principle in the finite-difference approximation, see Podvin and Lecomte, 1991). Appropriate upwind conditions can be used to decrease this number and increase performance. Entire planar surfaces of nodes can be computed in parallel because nodes on the same surface do not depend
on each other, as shown in Figure 1b. The 3DPMM sweeps through
the domain in the axial directions, by shifting the planar surface of
the nodes in the direction of the “pyramid” tops. If a new estimate
$\tau_{\text{new}}$ is smaller than the previous estimate $\tau_{\text{old}}$, the node will receive a
new estimate. In our novel method, amplitude estimates are com-
puted for a node after the traveltimes only if $\tau_{\text{new}} \leq \tau_{\text{old}}$. This con-
dition ensures that amplitude and time data in the upwind direction
are used for the amplitude computation. The transport equation for
computing the amplitude $A(x)$ is as follows (Červený, 2005):

$$\langle \nabla T(x), \nabla A(x) \rangle + \frac{1}{2} A(x) \nabla^2 T(x) = 0. \quad (3)$$

The gradient of the wavefront $\nabla T(x)$ is estimated during the trav-
elt ime computation. Given the gradient, the entrance location $x_E = (x_E, y_E)$ of the ray into the stencil bottom layer can be calculated.
Knowing the location at which the ray enters the stencil, we can interpo-
late for the traveltime (Zhang et al., 2011) and the amplitude
values at this point. Let $A_E, T_E$ be the amplitude and traveltime
values at the location, where the ray enters the stencil element, and
$A_N, T_N$ are the corresponding new values of the node being up-
dated. Using the transformed transport equation, we can approxi-
mate $A_N$ by

$$A_N = \frac{A_E(T_N - T_E)}{T_N - T_E + \frac{1}{2} \|x\|^2 \nabla^2 T}, \quad (4)$$

where $\|x\|$ is the Euclidean distance between node $N$ and the loca-
tion $E$. The additional work required to compute these values is
small because most traveltime data have recently been used in com-
putations and the gradient of the wavefront is estimated in the trav-
elt ime computation. The accuracy of the method highly depends on
the accuracy of the upwind traveltime field. It is necessary to avoid
inaccurate traveltimes around the source by special, very accurate
techniques such as refined meshes, wavefront construction methods
(Vinje et al., 1993; Rawlinson et al., 2008), or ray-tracing methods
(Julian and Gubbins, 1977; Rawlinson et al., 2008) to compute the
traveltimes around the source. The estimation of the Laplacian
of the traveltime field is the crucial and most difficult part of the cal-
culation. Using a traditional three-point stencil for the second deriv-
aves is often not possible because not all needed values are
necessarily already assigned values. Two different ways to solve
this problem are described in the following sections after character-
istic curves for the eikonal equation are discussed. After obtaining
the traveltime and the amplitude field, we want to estimate the
Green’s function $G(x, \omega)$ for the Helmholtz equation. The Helm-
holtz Green’s function as presented in Luo et al. (2012), is formu-
lated as

$$\nabla^2 G(x, \omega) + \frac{\omega^2}{v(x)^2} G(x, \omega) = -\delta(x - x_0), \quad (5)$$

where $\delta(x - x_0)$ is the Dirac delta function at the position $x_0$, $\omega$ is
the frequency, and $v(x)$ is the wave velocity at $x$. The solution in 3D
is given by (Leung et al., 2007)

$$G(x, \omega) = A(x) e^{i \omega T(x)}, \quad (6)$$

where $A(x)$ is the amplitude field and $T(x)$ is the traveltime field.
The Helmholtz Green’s function can be used directly for estimating
the amplitude field for different frequencies.

**COMPUTING RAYPATHS AND ENTRANCE LOCATIONS**

The characteristic curves of the eikonal equation are often re-
ferred to as rays in seismology. Characteristic curves of the isotropic
eikonal equation show the fastest path between two points given the
velocity (Červený, 2005). Characteristic curves are often defined in
terms of the Hamiltonian $H$ of the equation. For the eikonal equa-
tion 2, the Hamiltonian is defined as

$$H(x, p) = \frac{1}{s(x)} |p| - 1, \quad (7)$$

where $p = \nabla T(x)$. Let $x(s)$ be a parametrization of a characteristic
curve, $x(s) = (x(s), y(s), z(s))^T$, where $s$ is the parametrization
parameter. If the Hamiltonian $H$ is convex in $p$, the raypath can
be found from the following relation:

---

**Figure 1.** (a) The pyramid-shaped stencil used in 3DPMM. Notice the inclusion of the diagonal nodes. The node represented as an empty circle shows the position of a neighbor of the update node and is not used for the computation of $T_{ijk}$. (b) A layer of independent, pyramid shaped stencils. The entire layer of stencils can be computed in parallel. Image modified from Gillberg et al. (2012b).
\[ \frac{dx}{ds} = \nabla p H(x, p), \]  
(8)

as defined in Červený (2005). After differentiating \( H \) with respect to \( p \) and integration with respect to \( s \), we obtain the characteristic curve for the eikonal equation as

\[ x(s) = v(x) \frac{\nabla T(x)}{\| \nabla T(x) \|}, \]
(9)

\[ x(0) = x_0, \]
(10)

where \( \| \nabla T(x) \| = \sqrt{(\frac{\partial T(x)}{\partial x})^2 + (\frac{\partial T(x)}{\partial y})^2 + (\frac{\partial T(x)}{\partial z})^2} \), and \( x_0 \) is the starting point of the curve. Because the pyramid base is a distance \( dz \) from the node being updated, the entrance location of the ray follows as:

\[ x_E = -dz \frac{\partial T(x)}{\partial x}, \]
(11)

\[ y_E = -dz \frac{\partial T(x)}{\partial y}, \]
(12)

The entrance location (see Figure 2) ensures that only rays traveling through the bottom of the stencil are used to estimate a new time value. If the entrance locations are outside the stencil boundaries, the shortest path along the boundary is used to compute the new amplitude estimate. To compute the time and amplitude values at the entrance point of the ray into the stencil, we use linear interpolation in case of a two-node stencil and bilinear interpolation in case of a three-node stencil.

### Estimating the Laplacian for slowly changing velocity fields

The first method of estimating the Laplacian uses a wide area upwind of the stencil, and it is therefore applicable only to slowly changing velocity fields. Only one assigned node in the stencil contributes to the update node, and the fastest sweep manages to create a new time estimate. The amplitude calculation needs at least seven nodes with traveltime values to estimate the Laplacian. Therefore propose to use two more stencil planes in the upwind direction, as shown in Figure 3a. Already estimated nodes in a 147-point cuboid behind the stencil are used to approximate the necessary second derivatives for the Laplacian. The pseudocode for the algorithm using this stencil extension for the estimation of the Laplacian can be found in Algorithm 1. Provided a sufficiently large source with respect to the spatial dimensions, this stencil extension manages to create an amplitude estimate even during the first sub sweep, when most of the nodes within the stencil do not yet have a noninfinite time estimate (see Figure 3b). The method lacks accuracy because the estimation of the Laplacian is not sufficiently local when a large set of nodes is used. The fraction of nodes not being upwind of the update node can be high, especially for fast varying velocity fields, which leads to poor accuracy.

### A local approach for the estimation of the Laplacian

Provided a sufficiently large source, the method introduced in the previous section results in a new amplitude estimate for all numerical experiments we tested. This reduces the number of iterations needed for convergence, but the computational cost of performing an update is high. Two key points must be addressed: First, the number of sweeps can only be reduced by one with the stencil described in the previous section. Because the fast sweeping approach is an iterative method, and therefore needs many sweeps to converge, a reduction of the needed sweeps by one is only a minor improvement. Second, the computational cost of finding the Laplacian in a 147-point cuboid is high considering that it has to be done for every node in the grid.

With these considerations in mind, we have designed a method that uses data from a smaller area behind the stencil. The Laplacian

**Algorithm 1. Pseudocode for a subsweep in the positive x-direction (index \( i \)) for a slowly changing velocity field.** The indexing is adapted to the size of the stencil to ensure that all nodes in the stencil are inside the domain.

```plaintext
for i = 4, ..., n_x do
  for j = 4, ..., n_y - 3 do
    for k = 4, ..., n_z - 3 do
      \( t_{\text{old}} = T_{ijk} \)
      for all of the eight tetrahedron stencils do
        Compute a \( t_{\text{new}} \) estimate of \( T_{ijk} \) using values on the tetrahedron base;
        Compute \( \nabla T(x) \);
        Estimate entrance location \( x_E = (x_E, y_E) \)
        \( T_{ijk} = \min \{ T_{ijk}, \text{all } t_{\text{new}} \text{ estimates} \}; \)
        \( T_{ijk} = t_{\text{new}}; \)
        estimate \( \nabla^2 T(x) \) using
        \( T_{i-a,j+b,k+c} \in [0,1.23], b \in [0,1.23], c \in [0,1.23]; \)
        Create new amplitude estimate \( a_{\text{new}}; \)
        \( A_{ijk} = a_{\text{new}}; \)
        Select angle of incidence;
        Update the previous estimates.
```

---

Figure 2. Illustration of the entrance location \( (x_E, y_E) \) at the bottom of the tetrahedron stencil.
is estimated faster and more accurately at the expense of an increased number of sweeps needed for convergence. The idea is to create a method that does not always create an amplitude estimate to nodes at which the time value has been updated. Rather, the algorithm assigns an amplitude value if it is possible to calculate a Laplacian using local data. Different ways of estimating the Laplacian are possible. It may be tempting to directly use the seven-point stencil (see Figure 4) behind the pyramid stencil for the calculation of the Laplacian; however, the accuracy is poor in this case. The key to higher accuracy is to vary the position of three-point stencils for the second derivatives, depending on the entry of the ray into the pyramid base. The three-point stencils for the second derivatives are closest to the raypath are used for the estimation of the Laplacian. In cases in which the rays are outside the stencils, the shortest path along the stencil boundary is used (see Figure 4).

Amplitudes and traveltimes have to converge independently, therefore, at least three sweeps are necessary. The algorithm using the local estimation of the Laplacian is presented in Algorithm 2. Due to the special stencil shape, entire 2D planes can be updated in parallel. The simultaneous computation of amplitudes and traveltimes and the abundant parallelization make the method very efficient.

IMPLEMENTATION DETAILS

The current implementation, using the local approach for the estimation of the Laplacian, was written in C++ and was compiled with the CUDA code compiler nvcc. Because of the special stencil shape, some of the T-nodes were placed in the shared memory. Shared memory is a user-managed cache memory on graphics processing units (GPUs). Because the computation of one node is quite demanding, the use of CUDA shared memory only leads to a speed up of approximately 10% compared with the use of global memory. Our code was also parallelized for multicore CPU structures using OpenMP. The code was optimized with full optimization instructions using the -O3 flag. The numerical experiments were performed on two different machines.

The first machine is equipped with 32 GB of DDR3 RAM, a four-core Intel Core i7-4800MQ CPU @ 2.70 GHz CPU, and a GeForce GTX 770M GPU. The second machine is a Linux server with 64 GB of DDR3 RAM, an eight-core dual Intel E5-2670 (2.6 GHz) with a total of 16 physical compute cores, and a Tesla K20x GPU. When computing times are presented, they are averages from four repeated measurements. However, the variation in computing time is negligible between different runs.

Algorithm 2. Pseudocode for a subsweep in the positive x-direction (index i) for the local approach. The indexing is adapted to the size of the stencil and to ensure that all nodes in the stencil are inside the domain.

```plaintext
for i = 3, . . . , nx do
  for j = 3, . . . , ny − 2 do
    for k = 3, . . . , nz − 2 do
      for all of the eight tetrahedron stencils do
        calculate VT(x)
        compute Tijk using T_i−1,j,k; b, a ∈ {0, 1}, and b ∈ {0, 1};
        calculate corresponding entrance location xe = (xe, ye);
        calculate V2T(x) using T_i−a,j−b,k−c, a ∈ {0, 1, 2}, b ∈ {0, 1, 2}, c ∈ {0, 1, 2};
        compute Aijk using A_i−1,j−a,k−b, a ∈ {0, 1}, b ∈ {0, 1};
        if amplitude computation possible and tnew ≤ told
          select dnew;
        else if amplitude computation possible and tnew > told
          and amplitude is unassigned
          select dnew;
        else if tnew > told
          and amplitude is unassigned
          select dnew;
```
RESULTS

In this section, we present computational times needed for three different velocity models. All results were computed with a spherical source of 1% of the volume of the domain, which was computed by a first-order ray tracing. Different source sizes can be used depending on the demanded accuracy and the complexity of the velocity model. For every experiment, computing times for all required sweeps until convergence are presented. Convergence is reached when no amplitude or traveltime estimate changes by more than 0.1% compared with the value after the previous sweep. For each experiment, CPU computing times for a sequential computation, as well as using multiple cores, are presented. The computational times needed when using the mentioned GPUs are also presented. The CPU and GPU measurements use data stored with float accuracy.

The runs were performed on two different grid sizes, consisting of $124^3$ and $248^3$. The grid sizes were chosen to satisfy the constraints of the specific GPU that was used for the experiments. Other grid sizes are possible as long as they do not exceed the GPU’s bounded memory limitations.

Homogeneous velocity field

The simplest case, using a homogeneous velocity field, is a special challenge for the method. This is due to straight rays traveling along the stencil boundaries. Our estimation of the Laplacian suffers slightly in accuracy in these regions, as seen in Figure 5. This error does not emerge when rays are bent. Convergence was reached after three iterations. Computing times are presented in Table 1.

Two homogeneous half-spaces

The method was also applied to two homogeneous half-spaces with a smooth transition zone (see Figure 6). Once again, we see the inaccuracies in the $45^\circ$ regions that are present due to the inaccuracy of the stencils along boundaries of the tetrahedrons. We can see a rise of the amplitude, where rays collide (Figure 6); a natural behavior of linear systems known as superposition or, in the case of wave phenomena, interference. By reducing the width of the transition zone, we can model interfaces in the domain. The algorithm also works in the case of noncontinuous interfaces; however, the accuracy is improved if treated as such by creating new wavefronts along an interface. Convergence was reached after three iterations. The computing times are presented in Table 2.

The random velocity field

The third example is a random velocity field (see Figure 7) containing velocities in the range of 1000–1500 m/s. It was created by assigning uniformly distributed nodes random values in the range of 1000–1500 m/s and subsequent linear interpolation. The highest

![Figure 5.](image-url)
velocity gradients are 50 s\(^{-1}\). Once again, we can see the superposition principle of waves as amplitudes rise, where rays collide. In this example, we see that rays avoid slow velocity areas. Notice that we have full ray coverage also in the slow velocity areas, and any “shadow zone” problems are avoided using the suggested method (Červený, 2005). Convergence is reached after four iterations. The computing times are presented in Table 3.

**Comments on performance**

The GPUs consistently outperform the tested CPUs. This is expected because the 3DPMM allows for so many nodes (a minimum of \(124^3 = 15376\) nodes in the presented examples) to be updated in parallel at any time. The CPU implementation scales well up to four cores, but it flattens out thereafter. A significant amount of data is needed to update a node, which seems to decrease performance on the multicore implementation.

**DISCUSSION AND PERSPECTIVES**

The introduced method is able to compute the Helmholtz Green’s function in a stable, fast, and accurate manner; however, it suffers from some restrictions. As mentioned earlier, the current implementation is built for acoustic waves in isotropic media. The adjustment for elastic waves is straightforward, although it requires some modifications. Because the entrance location of the ray into the stencil is...
already used, the algorithm exhibits a natural ability for anisotropy. Adapting the algorithm to anisotropic problems is possible but not trivial because the gradient of the wavefront no longer coincides with the raypath. However, raypaths can be found in the time estimation process (Gillberg et al., 2012b), and our approach of estimating the Laplacian of the time field close to the ray origin can be

### Table 2. Computing time until convergence in seconds for different grid sizes and computer architectures for the second experiment. The type of processing unit and the number of used cores are indicated.

<table>
<thead>
<tr>
<th>Grid</th>
<th>i7,1</th>
<th>i7,4</th>
<th>E5,1</th>
<th>E5,4</th>
<th>E5,8</th>
<th>E5,16</th>
<th>770M</th>
<th>K20x</th>
</tr>
</thead>
<tbody>
<tr>
<td>124³</td>
<td>10.58</td>
<td>4.06</td>
<td>14.75</td>
<td>8.33</td>
<td>7.01</td>
<td>5.48</td>
<td>0.92</td>
<td>0.80</td>
</tr>
<tr>
<td>248³</td>
<td>94.37</td>
<td>29.90</td>
<td>144.81</td>
<td>62.79</td>
<td>56.09</td>
<td>50.57</td>
<td>6.23</td>
<td>4.08</td>
</tr>
</tbody>
</table>

---

### Table 3. Computing time until convergence in seconds for different grid sizes and computer architectures for the third experiment. The type of processing unit and the number of used cores are indicated.

<table>
<thead>
<tr>
<th>Grid</th>
<th>i7,1</th>
<th>i7,4</th>
<th>E5,1</th>
<th>E5,4</th>
<th>E5,8</th>
<th>E5,16</th>
<th>770M</th>
<th>K20x</th>
</tr>
</thead>
<tbody>
<tr>
<td>124³</td>
<td>14.56</td>
<td>5.95</td>
<td>20.14</td>
<td>11.22</td>
<td>9.22</td>
<td>7.68</td>
<td>1.21</td>
<td>1.01</td>
</tr>
<tr>
<td>248³</td>
<td>116.17</td>
<td>39.25</td>
<td>179.91</td>
<td>71.8</td>
<td>64.72</td>
<td>53.60</td>
<td>8.25</td>
<td>7.94</td>
</tr>
</tbody>
</table>

---

Figure 7. (a) A random velocity field created by linear interpolation. (b) The traveltime field rays avoid slow velocity zones and try to reach high-velocity zones. (c) The amplitude field shows where the linear interpolation causes roofs or valleys, and amplitudes are accordingly higher or lower. Sudden appearance of a slow velocity zone forces the wavefront to a concave bending, which leads to a superposition of amplitudes. (d) The corresponding Green’s function.
CONCLUSION

We have introduced a methodology to compute first-arrival traveltimes and amplitudes simultaneously by extending the 3DPMM method. The method was implemented for multicore CPUs and for GPUs, resulting in a fast and efficient solver for the Green’s functions of the Helmholtz equation. When solving for traveltimes, the direction of the ray is implicitly computed giving us the angle of incidence and raypaths. The proposed method was presented for first arrivals of acoustic waves, but extensions to anisotropic velocity fields are possible. The numerical experiments showed that the accuracy of the proposed method is highly dependent on the initialization step and the estimation of the Laplacian of the time field. The method may potentially perform optimally in conjunction with accurate ray tracing methods in the vicinity of the source. The presented algorithm takes only first arrivals into account, and it is therefore a high-frequency approximation. As shown in our paper, it is possible to estimate several wave phenomena components in combination with first-arrival traveltimes solvers. In fact, computing the amplitudes together with traveltimes increases the accuracy of the computed amplitude field when compared with postprocessing of the time field. Accurate amplitudes give rise to an additional source of information when imaging the subsurface. The methodology is well suited for parallel implementations, resulting in very fast solvers for Helmholtz Green’s functions. The proposed method can be applied to different components of a wave and presents the first step to a faster tool to model wave motion. An accurate and fast tool for forward modeling of wave phenomena could work as a viable alternative to full-waveform modeling in several seismic applications. Our methodology is especially appealing when taken into consideration that the computational time is in the order of seconds.

ACKNOWLEDGMENTS

The presented work was funded by Kalkulo AS and the Research Council of Norway under grant 238346. The work has been conducted at Kalkulo AS, a subsidiary of Simula Research Laboratory. We would like to thank S. Clark, A. Magnus Bruaset, and C. Tarrou for helpful comments and support.

REFERENCES