A fast method for linear waves based on geometrical optics

Christiaan C. Stolk

July 31, 2007

Abstract

We develop a fast method for solving the 1-D wave equation based on geometrical optics. From geometrical optics (e.g. Fourier integral operator theory or WKB approximation) it is known that high-frequency waves split into forward and backward propagating parts, each propagating with the wave speed, with amplitude that is slowly changing depending on the medium coefficients, under the assumption that the medium coefficients vary slowly compared to the wavelength. Based on this we construct a method of optimal, $O(N)$ complexity, with basically the following steps: 1. Decouple the wavefield into approximately forward and a backward propagating parts; 2. propagate each component explicitly along the characteristics over a time step that is small compared to the medium scale, but can be large compared to the wavelength; 3. apply a correction to account for the errors in the explicit propagation; repeat steps 2 and 3 over the necessary amount of time steps; and 4. reconstruct the full field by adding forward and backward propagating components again. Due to step 3 the method accurately computes the full wavefield. A variant of the method was implemented and outperformed a standard order (4,4) finite difference method by a substantial factor. The general principle is applicable also in higher dimensions, but requires efficient implementations of Fourier integral operators which are currently not available.

Keywords: Wave equation, numerical method, multiscale method, geometrical optics, integrating factor
1 Introduction

Consider waves propagating in an inhomogeneous medium which varies slowly on the scale of the wavelength. In this case, waves behave much like in the constant coefficient case. For example, in 1-D an initial pulse approximately splits into a forward propagating pulse and a backward propagating pulse, each propagating with the wave speed, and with slowly varying amplitude. Indeed for small times, the wave “sees” only a small, approximately constant part of the medium. This can be made precise using WKB, or geometrical optics theory, or the more general and advanced theory of Fourier integral operators. One finds that the above picture is true in the limit for high-frequency waves, these have the just described relatively simple interaction with the medium. For the low-frequency part the interaction with the medium is of course more complicated, e.g. reflections occur.

Simulating high-frequency waves using finite differences or finite elements is notoriously expensive, especially in 3 dimensions. One reason for this is the large number of time steps that is generally needed, since in conventional methods the time step is bounded by the space discretization length. In 1-D this leads to cost at least $O(N^2)$ if $N$ is the number of space discretization points. This on one the hand is quite understandable: The wavefield is computed over a finite part of the $(x,t)$-plane with resolution $1/N$ in both the $x$ and the $t$ direction. On the other hand, if we are only interested in the map from initial to final values one can argue that there is room for improvement: The high frequencies are well described by translation and scaling over quantities that follow from the smoothly varying medium. The low frequencies still need to be computed by some discretization, but with a coarse grid. In this paper we will show that in fact we can device a scheme that follows this pattern, and is of complexity $O(N)$, i.e. optimal.

The observation about the high cost of simulating high-frequency waves is not new, and various authors have sought to deal with this, e.g. [12] in 1-D, [2, 9] in higher dimensions. The paper [12] uses the observation that the matrix that describes the propagator $P(t)$ (the operator exponent $e^{tM}$ in the notation below, that maps initial values at time 0 to values at a later time $t$, assuming time independent coefficients) can be compressed by
wavelet compression. High-frequency signals in the propagator are concentrated around the characteristics. Low frequencies signals are not. Due to the separation in space and scale that is obtained using wavelets, this leads to many small entries that, if omitted, give only a small error to the matrix. The matrix is compressed in this way, and it becomes possible to store it. The operator exponent is then first computed for a small time $\tau$, and subsequently for longer times by repeated squaring $P(2\tau) = P(\tau)^2$, $P(4\tau) = U(2\tau)^2$, etc.

Unlike our method this idea is restricted to time-independent coefficients. Curvelet frames [15, 4] have been proposed to extend this idea to multiple dimensions.

In this paper we introduce a new, different concept to reduce computational cost. We explicitly separate forward and backward propagating parts of the waves, as made possible by high-frequency asymptotic theory, and propagate these explicitly. No matrix compression is used. Roughly speaking the method involves the following steps, that are repeated over a number of time steps to obtain the final result:

1. Decouple the wave field into a forward and a backward propagating part, like for the constant coefficient medium where we can find two functions $F$ and $B$ such that the solution is given by $U_1(x, t) = B(x + ct) + F(x - ct)$.

2. Propagate each component explicitly over a time step that is small compared to the medium scale, but large compared to the wavelength.

3. Apply a correction to account for the errors in the explicit propagation

4. Reconstruct the full field by adding forward and backward propagating components again.

For higher dimensions one could perhaps devise a similar scheme, however at this point in time it is not clear how to efficiently compute the Fourier integral operators needed in step 2.

Two methods according to this outline will be described. First we derive a relatively straightforward method, that is implemented numerically and tested. The goal of this is to get a first impression of what kind of numerical results can be obtained. Compared with an order (4,4) finite difference method we find improvements in speed of factors up to 20, depending on the smoothness of the medium.
A second method is derived using several more innovations (see section 6 and thereafter). For this method we study error estimates and the complexity, and we show that it has optimal $O(N)$ complexity. The $O(N)$ complexity is better than that in [12], but we also have another improvement compared to the repeated squaring method, namely that our method is also applicable in media with time-dependent coefficients.

Let us discuss in more mathematical terms the ideas behind the method. We consider the one-dimensional acoustic wave equation

\begin{equation}
(a(x, t)\partial_t^2 - \partial_x \circ b(x, t)\partial_x)U_1(x, t) = 0,
\end{equation}

with domain given by a circle $\Omega$ of integer length $L$. It will be convenient to write this as a first order system, let

\begin{equation}
U_2 = a\partial_t U_1, \quad U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}, \quad M = \begin{pmatrix} 0 & a^{-1} \\ \partial_x \circ b\partial_x & 0 \end{pmatrix},
\end{equation}

then (1) becomes

\begin{equation}
\frac{d}{dt}U = MU.
\end{equation}

We view this as an ODE with values in a functions space, which explains the notation $\frac{d}{dt}$ in this equation.

We are interested in the initial value problem where $U(t_0) = U_0$ is given and $U(t_1)$ is to be determined. The natural space to consider the equation is $U(t) \in H^1 \times L^2$, where $H^s = H^s(\Omega)$ denotes the Sobolev space of order $s$. With coefficients that are $C^{k,1}$ in space, and with time-derivative that is also $C^{k,1}$ in space, there is existence, uniqueness and stable dependence on initial values for $U_0 \in H^{s+1} \times H^s$, with

\begin{equation}
U(t) \in C([t_0, t_1], H^{s+1} \times H^s),
\end{equation}

for $-k - 1 \leq s \leq k$ [14, 16].

Let’s consider now where there is room for improvement in standard finite difference or finite element methods. Suppose $U_1, U_2$ are discretized on $\Omega$ by finite differences, using a regular grid with grid distance $h$ and $N = L/h$ grid points. Then the operator $M$ is discretized, and the time evolution is computed with some time-stepping procedure. The
operator $M$ behaves like a first order operator, mapping $H^{s+1} \times H^s$ to $H^s \times H^{s-1}$. Its norm is proportional to $h^{-1}$. Accuracy and stability of a discrete approximation now require that the time step is of order $h$, $\Delta t \lesssim h/c(x,t)$, with $c = \sqrt{b/a}$ the velocity (the Courant-Friedrichs-Lewy condition). The cost for given $N$ is therefore at least $O((\# \text{ of time steps}) \cdot N) = O(N^2)$.

To have lower cost, we will attack the number of time steps, by using larger time steps. An idea that has been used for this purpose is operator splitting with an integrating factor method. Suppose $M$ is of the form

$$M = A + B.$$  \hfill (4)

Operator splitting is the idea that the matrix exponential $e^{\Delta t(A+B)}$ is approximated by products of factors $e^{\Delta t_j A}$ and $e^{\Delta t_k B}$. One way to derive an operator splitting method is the integrating factor method. Let $E(t,t_0)$ be a solution operator for $U' = AU$, i.e. an operator that maps $U(t_0)$ to the solution $U(t)$ of $U' = AU$. For the time-independent case $E(t,t_0) = e^{(t-t_0)A}$. Then we can define

$$V = E(t,t_0)^{-1}U.$$ \hfill (5)

The term $E(t,t_0)^{-1}$ is then an integrating factor. Differentiating the equivalent equation $E(t,t_0)V = U$ gives that

$$(A + B)U = \frac{dU}{dt} = AE(t,t_0)V + E(t,t_0)\frac{dV}{dt}.$$  

Therefore, solving for $\frac{dV}{dt}$,

$$\frac{dV}{dt} = E(t,t_0)^{-1}BE(t,t_0)V.$$ \hfill (6)

To apply this usefully, the operator on the r.h.s. must have smaller norm than the original operator $M$, so that time-stepping can be performed with larger time steps. This is applied in some nonlinear equations with a diffusive part, for which the time evolution can be computed efficiently in the Fourier domain [20]. Because of this use of an integrating factor, we call our method a geometrical optics integrating factor method.

A similar idea is used in the Egorov theorem of microlocal analysis. In this theory, a Fourier integral operator (FIO) $E(t,t_0)$ is constructed [11, 10, 19, 21], such that the field
\[ V(t) = E(t, t_0)^{-1}U(t) \] satisfies

\[ \frac{\partial}{\partial t} V(t) = R(t, t_0) V(t), \quad (7) \]

where the operator \( R \) is smoothing, in the sense that it maps \( H^{s+1} \times H^s \to H^{s+1+K} \times H^{s+K} \) for any \( K \) desired (the order \( K \) depends on the amount of terms in the asymptotic series for the amplitude in the FIO \( E(t, t_0) \)). The fact that \( R \) is bounded, means that a properly discretized version can be bounded independent of \( h \). By the above reasoning the stepsize requirement would become independent of \( h \) (of course an estimate of the time-discretization error is needed to establish this). For small \( h \), as the number of time steps would become large due to the CFL condition, one might expect to have a gain in computation speed for the transformed differential equation (7).

Continuing this line of reasoning, the time-step could become independent of the number of space discretization points \( N \), assuming the desired accuracy stays fixed. For example, having initial conditions double in frequency, with the same medium and accuracy, one can conjecture that the time step could stay the same.

While Fourier integral operator theory has been developed for any space dimension, for dimension 2 or higher it is not clear how to efficiently obtain numerical approximations of Fourier integral operators (see for work in this direction e.g. the recent paper [3]). Here we treat therefore the 1-D case.

In this case, it is convenient not to work with the field \( U \), or with \( V \) in (7) directly, but instead work with forward and backward propagating components. These will be denoted by \( u_1 \) and \( u_2 \). An operator \( Q \) and its inverse will be constructed such that \( u = (u_1, u_2)^T = Q^{-1}U \) (this gives step 1 and 3). We will show that in terms of these variables the differential equation (3) becomes

\[ \frac{d}{dt} u = (T + R)u \quad (8) \]

with

\[ T = \begin{pmatrix} \sqrt{b/a}\partial_x + f_1 & 0 \\ 0 & -\sqrt{b/a}\partial_x + f_2 \end{pmatrix}, \]

\( f_1, f_2 \) functions given below, and \( R \) a remainder operator, that is explicitly derived and is continuous \( H^{s+1} \times H^{s+1} \to H^{s+2} \times H^{s+2} \) (for time-independent coefficients \( f_1 = \)
\( f_2 = 0 \). Versions of \( R \) with off-diagonal terms that are even more smoothing can also be constructed, see further on in the paper.

Equation (8) will be used for operator splitting. The equation \( u' = Tu \) corresponds to two transport equations (step 2 in the outline above). These are solved using the method of characteristics. This yields a geometrical optics approximation of the propagator. The term \( R \) then yields the correction mentioned in step 3 of the 4 points above.

Computing with the characteristics is cheaper than computing directly on the wave field, e.g. using a discretization of the transport equation. The explanation for this is that the time steps in an ODE solver needed for solving for the characteristics depend on the medium smoothness, and not on the smoothness of the wavefield, and can therefore be longer than the time steps in a discretization of the transport equation. Similarly not for each grid point is it necessary to compute a characteristic because interpolation can be used. After computing the characteristics, applying the flow along the characteristics becomes a standard interpolation problem.

The computation of flow along characteristics is related to the use of moving grids in scalar conservation laws. Originally the reason to have the grid moving with the singularities of solutions, was that an adapted (locally refined) grid would stay adapted to the singularities. But it was also observed that this could lead to larger time steps [13].

As mentioned we have both numerical and theoretical results. First we derive a relatively simple method following the above ideas. This method has been implemented, and compared with a standard order (4,4) finite-difference method described in [6]. Factors of order 10 to 20 of improvement in the computation speed were obtained in examples.

In the second part of the paper we study error estimates and complexity. It turns out that the method described in sections 2 to 4 does not yet have the best possible complexity. With several enhancements we construct a method (or a class of methods) with optimal complexity \( O(N) \) to solve the initial value problem. These additional features are the use of higher-order decoupling, and of a multiscale decomposition where each scale has its own time step (multiscale time-stepping). They will be further introduced in section 6.

The remainder of the paper will be organized as follows. In section 2 we describe the separation of the forward and backward propagating parts of the wavefield (decoupling). The differential equation is then transformed into one to which operator splitting and the
integrating factor method can be applied. This is discussed in section 3. We then describe a simple space discretization and the resulting algorithm in section 4. Section 5 contains some numerical results. Section 6 introduces the main additional ideas behind the method for which we establish \( O(N) \) complexity. These are further worked out and proved in sections 7, 8 and 9. We end with a short discussion of the results.

2 Decoupling the equation

The splitting in equations (4)-(6) is not directly applied to \( M \), first the equation is transformed to new variables as announced in (8). We define new variables by

\[
U(t) = Q(t)u(t),
\]

with \( Q \) an invertible matrix operator. The operator \( Q \) is independent of \( t \) if \( M \) is independent of \( t \), and may otherwise depend on \( t \). The equation for \( u \) is then (' denoting time differentiation)

\[
u' = (Q^{-1}MQ - Q^{-1}Q' )u.
\]

The purpose of this section is to find a suitable operator \( Q \), such that the resulting differential equation is of the form

\[
\frac{d}{dt} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{b}{a}} \partial_x + f_1 & 0 \\ 0 & -\sqrt{\frac{b}{a}} \partial_x + f_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} + R \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},
\]

with \( Q \), \( f_1 \) and \( f_2 \) and the remainder operator \( R \) to be determined. In fact, we will derive an explicit expression for

\[
QR = MQ - Q \begin{pmatrix} \sqrt{\frac{b}{a}} \partial_x + f_1 & 0 \\ 0 & -\sqrt{\frac{b}{a}} \partial_x + f_2 \end{pmatrix} - Q'.
\]

The notations \( A, B, C \) will be used below in evaluating the product. Note that \( R \) is not given directly, but has to be computed as the product of \( Q^{-1} \) and \( QR \) which are given, the reason for this is that we want to minimize the use of inverse differential operators, and
here the only place where those occur is in $Q^{-1}$. We will find that the operator $R$ belongs to a class of pseudodifferential operators of order $-1$.

In the remainder of the section the actual computation is done. We treat separately the cases where $a, b$ are time-independent, resp. the general case with time-dependent $a, b$. For convenience we collect the results in the following lemma.

**Lemma 2.1.** For the time-independent case, with $Q$ given by (12), and $f_1 = f_2 = 0$, $QR$ is given by (13) and (14). For the time-dependent case, with $Q, f_1, f_2$ given by (15), (16) and (19), $QR$ is given by (17), (18), (20) and (21).

**Computation for the time-independent case** In this case we will take $Q$ independent of $t$, so that $C = 0$, and such that $f_1$ and $f_2$ vanish. Consider first the following choice for $Q$

$$Q^{(0)} = \begin{pmatrix} 1 & 1 \\ \sqrt{ab} \partial_x & -\sqrt{ab} \partial_x \end{pmatrix}$$

A quick computation shows that

$$Q^{(0)}R^{(0)} = MQ^{(0)} - Q^{(0)} \begin{pmatrix} \sqrt{b/a} \partial_x & 0 \\ 0 & -\sqrt{b/a} \partial_x \end{pmatrix} = \begin{pmatrix} \text{order}(0) & \text{order}(0) \\ \text{order}(1) & \text{order}(1) \end{pmatrix} \tag{11}$$

so to highest order this is a good choice.

Next we modify $Q$ so that (1) it is invertible, and (2) the components of $QR$ vanish to one order lower. The operator $Q$ becomes invertible when the derivative is replaced by a regularized derivative, which will be denoted by $\tilde{\partial}_x$, defined in the Fourier domain by multiplication with $ik + \frac{\alpha}{\beta k^2 + 1}$, with $\alpha, \beta$ suitable positive, real constants that remain to be chosen. To eliminate the order 0 and order 1 terms in (11), the columns of $Q$ will be normalized by a weight function, we will try

$$Q = \begin{pmatrix} f(x) & f(x) \\ f(x)\sqrt{ab}\tilde{\partial}_x & -f(x)\sqrt{ab}\tilde{\partial}_x \end{pmatrix}, \quad Q^{-1} = \frac{1}{2} \begin{pmatrix} f^{-1} & f^{-1} f^{-1} \frac{1}{\sqrt{ab}} \\ f^{-1} & -f^{-1} f^{-1} \frac{1}{\sqrt{ab}} \end{pmatrix}, \tag{12}$$

with $f$ given by $f = a^{-1/4}b^{-1/4}$.

For contribution $A$ we then find

$$A_{11} = -A_{21} = f(x)\sqrt{\frac{b}{a}}\partial_x + f(x)\sqrt{\frac{b}{a}}(\tilde{\partial}_x - \partial_x) = a^{-3/4}b^{1/4}\partial_x + a^{-3/4}b^{1/4}(\tilde{\partial}_x - \partial_x),$$
and

$$A_{21} = A_{22} = \partial_x b \partial_x f$$

$$= a^{1/4} b^{1/4} \partial_x a^{-1/2} b^{1/2} \partial_x + R_1$$

with

$$R_1 = a^{-1/4} b^{3/4} \left[ \left( \frac{1}{4} \partial_x \log a - \frac{3}{4} \partial_x \log b \right) \left( \frac{1}{4} \partial_x \log a + \frac{1}{4} \partial_x \log b \right) - \left( \frac{1}{4} \partial_x^2 \log a + \frac{1}{4} \partial_x^2 \log b \right) \right].$$

Contribution $B$ is given by

$$B_{11} = -B_{12} = a^{-3/4} b^{1/4} \partial_x.$$ 

and

$$B_{21} = B_{22} = a^{1/4} b^{1/4} \partial_x a^{-1/2} b^{1/2} \partial_x + a^{1/4} b^{1/4} (\tilde{\partial}_x - \partial_x) a^{-1/2} b^{1/2} \partial_x$$

We thus find the following for $QR$

$$(QR)_{11} = -(QR)_{12} = a^{-3/4} b^{1/4} (\tilde{\partial}_x - \partial_x), \quad (13)$$

and

$$(QR)_{21} = (QR)_{22} = R_1 - a^{1/4} b^{1/4} (\tilde{\partial}_x - \partial_x) a^{-1/2} b^{1/2} \partial_x \quad (14)$$

The time-dependent case In this case we try

$$Q = \begin{pmatrix}
  f(x) & f(x) \\
  f(x) \sqrt{ab} \tilde{\partial}_x + c_1 & -f(x) \sqrt{ab} \tilde{\partial}_x + c_2 
\end{pmatrix}, \quad (15)$$

with $f$ as above, and $f_1, f_2, c_1, c_2$ to be determined. The inverse of $Q$ will be discussed below. We find

$$A_{11} = a^{-3/4} b^{1/4} \partial_x + a^{-3/4} b^{1/4} (\tilde{\partial}_x - \partial_x) + a^{-1} c_1$$

$$A_{12} = -a^{-3/4} b^{1/4} \partial_x - a^{-3/4} b^{1/4} (\tilde{\partial}_x - \partial_x) + a^{-1} c_2,$$

$A_{21}$ and $A_{22}$ remain unchanged. For $B$ we find

$$B_{11} = a^{-3/4} b^{1/4} \partial_x + (ab)^{-1/4} f_1$$

$$B_{12} = -a^{-3/4} b^{1/4} \partial_x + (ab)^{-1/4} f_2.$$
and

\[ B_{21} = a^{1/4}b^{1/4} \partial_x a^{-1/2}b^{1/2} \partial_x + a^{1/4}b^{1/4}(\tilde{\partial}_x - \partial_x)a^{-1/2}b^{1/2} \partial_x + c_1 \sqrt{b/a} \partial x + (ab)^{1/4} \tilde{\partial}_x f_1 + c_1 f_1, \]

and

\[ B_{22} = a^{1/4}b^{1/4} \partial_x a^{-1/2}b^{1/2} \partial_x + a^{1/4}b^{1/4}(\tilde{\partial}_x - \partial_x)a^{-1/2}b^{1/2} \partial_x - c_2 \sqrt{b/a} \partial x - (ab)^{1/4} \tilde{\partial}_x f_2 + c_2 f_2. \]

For \( C \) we have

\[ C_{11} = \partial_t (ab)^{-1/4}, \]

\[ C_{12} = \partial_t (ab)^{-1/4}, \]

and

\[ C_{21} = \partial_t (ab)^{1/4} \tilde{\partial}_x + \partial_t c_1, \]

\[ C_{22} = -\partial_t (ab)^{1/4} \tilde{\partial}_x + \partial_t c_2, \]

Adding all the contributions we find that

\[ (QR)_{11} = +a^{-3/4}b^{1/4}(\tilde{\partial}_x - \partial_x) + a^{-1} c_1 - (ab)^{-1/4} f_1 - \partial_t (ab)^{-1/4} \]

and

\[ (QR)_{21} = R_1 - a^{1/4}b^{1/4}(\tilde{\partial}_x - \partial_x)a^{-1/2}b^{1/2} \partial_x - c_1 \sqrt{b/a} \partial x - (ab)^{1/4} \tilde{\partial}_x f_1 - c_1 f_1 - \partial_t (ab)^{1/4} \tilde{\partial}_x - \partial_t c_1. \]

The lower order terms vanishes if

\[ c_1 = -a^{3/4}b^{-1/4}((ab)^{-1/4} \partial_t (ab)^{1/4}), \]

\[ f_1 = 0. \]

What results is

\[ (QR)_{11} = a^{-3/4}b^{1/4}(\tilde{\partial}_x - \partial_x), \]

and

\[ (QR)_{21} = R_1 - a^{1/4}b^{1/4}(\tilde{\partial}_x - \partial_x)a^{-1/2}b^{1/2} \partial_x - c_1 \sqrt{b/a} \partial x - \partial_t (ab)^{1/4} \tilde{\partial}_x - \partial_t c_1 \]

\[ = R_1 - a^{1/4}b^{1/4}(\tilde{\partial}_x - \partial_x)a^{-1/2}b^{1/2} \partial_x - \partial_t (ab)^{1/4} (\tilde{\partial}_x - \partial_x) + \partial_t (\sqrt{a/b} \partial_t (ab)^{1/4}) \]
Similarly we have

\[(QR)_{12} = -a^{-3/4}b^{1/4} \left( \tilde{\partial}_x - \partial_x \right) + a^{-1}c_2 - (ab)^{-1/4}f_2 - \partial_t(ab)^{-1/4} \]

and

\[(QR)_{22} = R_1 - a^{1/4}b^{1/4} \left( \tilde{\partial}_x - \partial_x \right) a^{-1/2}b^{1/2} \partial_x + c_2 \sqrt{b/a} \partial_x + (ab)^{1/4} \tilde{\partial}_xf_2
- c_2f_2 + \partial_t(ab)^{1/4} \tilde{\partial}_x - \partial_t c_2, \]

with lower order terms vanishing if

\[c_2 = -a^{3/4}b^{-1/4}((ab)^{-1/4}\partial_t(ab)^{1/4}), \]

\[f_2 = 0. \]

The result for \((QR)_{12}\) and \((QR)_{22}\) is

\[(QR)_{12} = -a^{-3/4}b^{1/4} \left( \tilde{\partial}_x - \partial_x \right), \]

and

\[(QR)_{22} = R_1 - a^{1/4}b^{1/4} \left( \tilde{\partial}_x - \partial_x \right) a^{-1/2}b^{1/2} \partial_x + c_2 \sqrt{b/a} \partial_x + \partial_t(ab)^{1/4} \tilde{\partial}_x - \partial_t c_2
= R_1 - a^{1/4}b^{1/4} \left( \tilde{\partial}_x - \partial_x \right) a^{-1/2}b^{1/2} \partial_x + \partial_t(ab)^{1/4} \left( \tilde{\partial}_x - \partial_x \right) + \partial_t(\sqrt{a/b} \partial_t(ab)^{1/4}) \]

This completes the time-dependent case, except for the inverse of \(Q\).

For the inversion, rewrite \(Q\) as

\[Q = \left( \begin{array}{cc} f(x) & f(x) \\ f(x) \sqrt{ab} (\tilde{\partial}_x + \bar{c}_1) & -f(x) \sqrt{ab} (\tilde{\partial}_x + \bar{c}_2) \end{array} \right), \]

with \(\bar{c}_j = \frac{c_j}{f(\sqrt{ab})}\). The operator \(\partial_x + \bar{c}_1\) can be written as

\[\partial_x + \bar{c}_1 = e^{-f(\tilde{\partial}_1 - \langle \tilde{c}_1 \rangle) dx} (\partial_x + \langle \tilde{c}_1 \rangle) e^{f(\tilde{\partial}_1 - \langle \tilde{c}_1 \rangle) dx} \]

and similarly for \(\partial_x + \bar{c}_2\). When \(\partial_x\) is replaced by \(\tilde{\partial}_x\) this becomes an invertible operator, if \(\alpha\) in the definition of \(\tilde{\partial}_x\) is chosen large enough, depending on \(\langle \tilde{c}_1 \rangle\). This does change \(Q\) somewhat, but only to lower order, we will therefore define \(Q\) as

\[Q = \left( \begin{array}{cc} f(x) & f(x) \\ G_1 & G_2 \end{array} \right) \]
with \( G_1 = f(x)\sqrt{ab}e^{-\int (\hat{c}_1 - \langle \hat{c}_1 \rangle) \, dx} (\tilde{\partial}_x + \langle \tilde{c}_1 \rangle)e^{\int (\hat{c}_1 - \langle \hat{c}_1 \rangle) \, dx} \) and \( G_2 = f(x)\sqrt{ab}e^{\int (\hat{c}_2 - \langle \hat{c}_2 \rangle) \, dx} (\tilde{\partial}_x - \langle \tilde{c}_2 \rangle)e^{-\int (\hat{c}_2 - \langle \hat{c}_2 \rangle) \, dx} \). For the inverse of \( Q \) we find
\[
Q^{-1} = \begin{pmatrix}
-(G_1 - G_2)^{-1} G_2 f^{-1} & (G_1 - G_2)^{-1} \\
(G_1 - G_2)^{-1} G_1 f^{-1} & -(G_1 - G_2)^{-1}
\end{pmatrix}.
\]
This means also that \( \tilde{\partial}_x - \partial_x \) must be replaced by \( e^{-\int (\hat{c}_1 - \langle \hat{c}_1 \rangle) \, dx} (\tilde{\partial}_x - \partial_x)e^{\int (\hat{c}_1 - \langle \hat{c}_1 \rangle) \, dx} \) in (17) and (18) and by \( e^{\int (\hat{c}_2 - \langle \hat{c}_2 \rangle) \, dx} (\tilde{\partial}_x - \partial_x)e^{-\int (\hat{c}_2 - \langle \hat{c}_2 \rangle) \, dx} \) in (20) and (21).

3 Operator splitting and time-stepping

The equation for the decoupled wavefields \( u \) is now
\[
\frac{d}{dt}u = (T + R)u
\]
with \( R \) as derived in the previous section and \( T \) given by
\[
T = \begin{pmatrix}
\sqrt{b/a} \partial_x & 0 \\
0 & -\sqrt{b/a} \partial_x
\end{pmatrix}.
\]

The integrating factor will be \( E(t, t_0)^{-1} \), where \( E(t, t_0) \) solves \( \frac{d}{dt}E(t, t_0) = TE(t, t_0) \), \( E(t_0, t_0) = \text{Id} \), and we will define a field \( v \) by
\[
v(t, t_0) = E(t, t_0)^{-1}u(t),
\]
which satisfies the differential equation
\[
\frac{dv}{dt} = E(t, t_0)^{-1}RE(t, t_0)v.
\]

Applying Euler forward time-stepping for this equation, gives
\[
v(t + \Delta t, t) \approx (1 + \Delta t E(t + \Delta t, t)^{-1}RE(t + \Delta t, t))u(t),
\]
using that \( v(t, t) = u(t) \). Hence
\[
u(t + \Delta t) \approx (1 + \Delta t R)E(t + \Delta t, t)u(t).
\]
A symmetric form of splitting (cf. Strang splitting [17]), leads to the following time-stepping, expressed in time-stepping for $u$

$$u(t + \Delta t) \approx (1 + \frac{1}{2}\Delta t R) E(t + \Delta t, t)(1 + \frac{1}{2}\Delta t R)u(t). \quad (25)$$

Let us now explain in more detail the computation of $E(t, t_0)$. This is a diagonal $2 \times 2$ matrix operator. We take the forward propagating component (the $E_{2,2}$ components, which acts on the $u_2$ field), the backward propagating component is done similarly. The characteristic equations is

$$\frac{dx}{dt} = c(x, t). \quad (26)$$

For the time-independent case, we can solve this ODE for $x(t)$ with initial value $x(t_0) = x_0$ by separating the variables, which yields the equation

$$\int_{x_0}^{x} c(\xi)^{-1} d\xi = t - t_0,$$

so the computation can be done from a primitive $\int c(x) dx$. For the time-dependent case (26) is solved directly. Let $X(x_0, t, t_0)$ denotes the solution $x(t)$ with initial values $x(t_0) = x_0$. Then we have

$$E_{2,2}(t, t_0)u_2 = u_2(t_0, X(x, t_0, t)) \quad (27)$$

(the characteristic is computed backward.) If $\Phi_2(t, t_0)$ denotes the characteristic flow mapping $x_0$ to $X(x_0, t, t_0)$, this can also be written as a pullback

$$E_{2,2}(t, t_0)u_2(t_0) = \Phi_2(t_0, t)^*u_2(t_0).$$

### 4 Numerical implementation

For a numerical implementation, it remains to perform the space discretization. We chose to work with finite-differences, which is easy to implement. The following operators were discretized:

1. $\partial_x$. This operator was discretized using central differences.

2. $\tilde{\partial}_x, \tilde{\partial}_x^{-1}, \tilde{\partial}_x - \partial_x$. These are applied in the Fourier domain, with a regularized version of central differences. The computation involves an FFT and an inverse FFT,
which, due to the $O(N \log N)$ cost of this operation will form the bulk of the computations.


4. The translation operator $E(t, t_0)$ is computed for the time-independent case using the primitive $\int c(x)^{-1} \, dx$, mentioned above, and using a Runge-Kutta ODE solver otherwise. Then 3-th order Lagrange interpolation is applied. For the time-independent case a sparse matrix is precomputed, that performs the translation over a given time step $\Delta t$.

In this way a simple numerical implementation of the method given by (25) was made.

5 Numerical results

In the numerical results we concentrate on the method for the time-independent case. For this case comparisons of computation time were made. For the time-dependent case it was observed that solution are well approximated. But we feel the results for time-independent case give sufficient indication of the effectiveness of the method.

For this method, with the assumption of medium smoothness it is of course an important question just how smooth the medium coefficients need to be in order that the method demonstrates an improvement compared to more conventional methods. Therefore numerical results were computed for media with increasing smoothness. The media were chosen parameterized by B-splines of order 3, the coefficients $a$ of the media were randomly chosen, uniformly distributed between 0.4 and 1.6. The increasing smoothness was obtained by increasing the node distance, for which we took the values 1, 2, 4 and 8. The $b$ coefficient was chosen equal to 1. The initial value for $U_1$ was a pulse of approximately unit width, the initial value for $U_2$ was chosen equal to zero. In Figure 1 one such medium is displayed. In figure 2 the initial value for $U_1$ is plotted. The propagation was over approximately 100 wavelengths.

The results were compared with the result of a order (4,4) finite difference method, see [6]. Both methods were implemented in Matlab. For our method the main cost was in the
Figure 1: Medium coefficients with random B-splines with knot distance 4.

Fourier transform used for computing $\tilde{\partial}_x$ and its inverse. In the standard finite difference methods, for each time step a sparse matrix is applied and this constituted almost 100 % of the cost.

The first check was that the method actually approximates the solutions well. This was indeed the case. In Table 1 some numerical results are given, where computation time is compared. For the new method we required the error to be smaller in both the supremum and the $L^2$ sense, or at most 10 % larger in one of the two, but better when both are taken into account. As can be seen, knot distance 1 is not sufficient to obtain any gain, but from knot distance 2 considerable gain is obtained, up to a factor of about 20 for very smooth media.

As this is only a first implementation we feel this is a strong encouragement to further analyze geometrical optics based methods.
6 An optimal complexity method: overview

For the method introduced above there were no rigorous error estimates given. The complexity is however at least $O(N \log N)$ since the regularized derivative $\tilde{\partial}_x$ and its inverse were computed in the Fourier domain, and needed to be computed for each time step. In this section we present a more elaborate algorithm, for which we establish that the complexity is $O(N)$, where $N$ denotes the number of grid points in the space discretization.

So the task in the remaining sections is on the hand to control the error in a numerical method, and on the other hand control the cost. The discretization will be done for the differential equation

$$\frac{dv}{dt} = E(t, t_0)^{-1} RE(t, t_0)v,$$

that resulted from (23) after applying the integrating factor. It follows from the results in section 8 below that the transformation from the original equation (3) to this form and back can be done at cost $O(N)$ and with error satisfying bounds that are sufficient.

We will provide precise error estimates of classical type, i.e. we assume the input has a certain amount of additional regularity, we consider the discretization error in the result
Table 1: Comparison of the cost of the method of section 4 with an order (4,4) finite difference method. $h_{FD}$ is the space stepsize taken in the finite difference scheme for which the comparison is made.

given that the input has to be approximated in an $N$-dimensional space of (spline-) functions, and then show that the total error in the output is of the same order in $N$ as the discretization error. Evolution according to (28) maps initial values $v(t_0) = v_0$ in $H^1 \times H^1$ to final values $v(t_1)$ that are also in $H^1 \times H^1$. We will assume that $v_0$ is in $H^{1+\alpha} \times H^{1+\alpha}$, i.e. has $\alpha$ additional orders of regularity. The discretization error that results of putting $v_0$ in an $N$-dimensional spline space can then be estimated by $C N^{-\alpha}$. We will show that, for a method with cost that can be bounded by $C N$, the final result satisfies an estimate of the type

$$\|v_{\text{approx}}(t_1) - v(t_1)\|_{H^1 \times H^1} \leq C N^{-\alpha}$$

(the letter $C$ may mean a different constant in different equations).

A naive approach would be to simply take the differential equation (28), first apply a discretization in space, and then subsequently apply discretization in time. The time discretization should preferably be of higher order. There are two main problems with this approach, which will lead to additional special features of our method. These new features are the following:
1. **Higher order decoupling**

Control of the time-discretization error in higher-order time-stepping, say of order \( K \), requires bounds on the time-derivatives of the operator \( E(t, t_0)^{-1} RE(t, t_0) \) occurring on the right hand side of (28). The first time derivative contains a commutator \([R, T]\) (which is of order 0, and hence bounded), but higher time derivatives contain higher-order commutators, that are of positive order, and hence do not satisfy the required bounds. To address this issue we will introduce higher-order decoupling. In section 7 we will construct a new operator \( R \), with off-diagonal terms that are smoothing operators of order \( K \), and show that its time derivatives of order 0, \ldots, \( K \) are bounded on a sufficiently large range of Sobolev spaces. The higher-order decoupling is obtained by adapting an argument of Taylor [19, chapter 9] or [18].

2. **Multiscale time-stepping**

The second problem that needs to be addressed is that in our complexity estimates, with increasing \( N \), the error must decrease. This in turn means that the time step must decrease, which would lead to superlinear complexity. To address this issue we introduce **multiscale time-stepping**. The idea is that the coarse scales are propagated with a small time step. The coarse scales are parameterized with relatively few coefficients, but contain most of the energy. It is therefore affordable to use a smaller time step, and at the same time leads to a big improvement in the error. For the fine scales, that contain relatively little energy larger time steps are used. Incidentally this is very much in agreement with the philosophy of asymptotic methods, where the high frequencies are well approximated. Each time step amounts to a correction to the purely asymptotic approximation, so little of those are needed for the high frequencies. The idea of multiscale time-stepping is new to my knowledge.

Because of the multiscale time-stepping, we assume the use of a wavelet based multiscale discretization in space. We will use [5] as our main reference for wavelet discretization, see also [7].

In the next 3 sections we will work out the above issues in detail and prove the \( O(N) \) complexity result. Section 7 concerns the higher-order decoupling. Discretization and operator approximation will be discussed in section 8. Section 9 will contain the ideas on
multiscale time-stepping and the final parts of the proof that combine all the intermediate results.

7 Higher-order decoupling

By the transformation $u = Q^{-1}U$ in section 2, the original system (3) was transformed to

$$u' = (T + R)u,$$

where $T + R = Q^{-1}MQ - Q^{-1}Q'$. We had

$$T = \begin{pmatrix} \sqrt{b/a} \partial_x & 0 \\ 0 & -\sqrt{b/a} \partial_x \end{pmatrix}.$$

The operator $R$ is a matrix pseudodifferential operator, with components that are of order

$$R = \begin{pmatrix} \text{order}(-1) & \text{order}(-1) \\ \text{order}(-1) & \text{order}(-1) \end{pmatrix}.$$  

(29)

Here by order(-1) we mean that it is bounded $H^s \rightarrow H^{s+1}$ for a suitable range of $s$.

In this section we explain how to construct $Q$ such that $R$ has the property that

$$\frac{d^j}{dt^j}(E(t, t_0)^{-1}RE(t, t_0))$$

is bounded on $H^1 \times H^1$ for $j = 0, 1, \ldots, K$,  

(30)

with $K$ a positive integer indicating, as mentioned, the order of the time-stepping that is going to be used. In particular we will construct a modified operator $Q$, such that (with $T$ unchanged)

$$R = \begin{pmatrix} \text{order}(-1) & \text{order}(-K) \\ \text{order}(-K) & \text{order}(-1) \end{pmatrix}.$$  

(31)

The old operators $Q$ and $R$ will be referred to as $Q(-1)$ and $R(-1)$, because of (29). The new operators will be referred to as $Q(-K)$ and $R(-K)$. In the process we will also construct operators $Q(-2), \ldots, Q(-K+1)$, and $R(-2), \ldots, R(-K+1)$, with off-diagonal terms in $R(j)$ of order $j$.

We will use a variant of a construction of Taylor [19, chapter 9] or [18]. He showed that strictly hyperbolic pseudodifferential systems in any dimension can be decoupled such that
the off-diagonal terms are of arbitrary low order. His case is slightly different than ours. On
the one hand the problem at hand is easier because it is one-dimensional. On the other hand
we don’t assume \( C^\infty \) coefficients, and must therefore keep track of coefficient regularity.
Therefore we work in a special class of pseudodifferential operators, that we now discuss.

We write \( \tilde{\partial}_x = \partial_x + S \), where we assume that \( S \) is smoothing in the sense that it is
continuous \( H^s \to H^{s+K}, 1 - K \leq s \leq 1 \). We will assume that we can write the \( R^{(j)} \) in
the form of a polyhomogeneous pseudodifferential operators, with a remainder as follows

\[
A = \sum_{j=-m_2+1}^{m_1} a_j(t,x)(\partial_x + S)^j + \text{Rem}(A, m_2). \tag{32}
\]

The remainder \( \text{Rem}(A, m_2) \) is assumed to be an operator that is bounded from \( H^s \to H^{s+m_2} \), for \( 1 - m_2 \leq s \leq 1 \). It is assumed to be a product of coefficients that are in \( C^{m_2,1} \) and smoothing convolutional operators used above (like e.g. \( S \), \( (\partial_x + S)^{-j} \), or \( (\partial_x + S + \text{constant})^{-j} \) needed in the evaluation of \( Q \) in (22)), such that the latter are
together sufficiently smoothing. The coefficients \( a_j(t,x) \) will be assumed to have \( C^{2m_2+j,1} \)
regularity. We assume \( R^{(-1)} \) is of this form with \( m_1 = -1 \) and \( m_2 = K \).

The properties of the new \( R \), i.e. \( R^{(-K)} \), that we want to construct can be made precise
in terms of this class of pseudodifferential operators. We will use the following require-
ments for the diagonal and off-diagonal components, and for their time derivatives, where
the parameter \( j \) is equal to \( K \) for \( R^{(-K)} \): For the diagonal components there is the property

\[
R_{1,1}, R_{2,2} \text{ are of the form (32) with } m_1 = -1, m_2 = j, \tag{33}
\]

the property for the off-diagonal components is

\[
R_{1,2}, R_{2,1} \text{ are of the form (32) with } m_1 = -j, m_2 = j, \tag{34}
\]
i.e. only the remainder term is present, while for the time-derivatives there are similar
properties as follows:

\[
\frac{dR^l}{dt^l} \text{ satisfies (34) and (33) with } j - l \text{ instead of } j, l = 1, \ldots, j. \tag{35}
\]

Next we show that it is sufficient if \( R^{(-K)} \) satisfies these three properties with \( j = K \). Below we will construct \( R^{(-K)} \) that actually satisfy these properties.
**Lemma 7.1.** Properties (33), (34), (35) with $j = K$ imply (30).

**Proof.** Differentiating $E(t, t_0)^{-1} R^{(-K)} E(t, t_0)$ gives

$$\frac{d}{dt} (E(t, t_0)^{-1} R^{(-K)} E(t, t_0)) = E(t, t_0)^{-1} \left( [R^{(-K)}, T] + \frac{dR^{(-K)}}{dt} \right) E(t, t_0).$$

Based this on we can define operators $F_p$ such that $\frac{d}{dt} (E(t, t_0)^{-1} R^{(-K)} E(t, t_0)) = E(t, t_0)^{-1} F_p E(t, t_0)$.

Let $F_0 = R^{(-K)}$, and define $F_p$, $1 \leq p \leq K$ inductively by $F_p = [F_{p-1}, T] + F'_{p-1}$. The lemma follows if all the $F_p$, $0 \leq p \leq K$ satisfy the above requirement with $j = 0$. In fact we will argue that the $F_p$ satisfy the requirement with $j = K - p$.

So assume $F_p$ satisfies the above requirement with $j = K - p$. We will show that then $F_{p+1}$ satisfies the requirement with $j = K - p - 1$. Now $F_{p+1} = [F_p, T] + F'_p$. Clearly $F'_p$ satisfies the requirement with $j = K - p - 1$. The commutator $[F_p, T]$ equals

$$[F_p, T] = \left( \begin{array}{cc} [F_{p,1,1}, T_{1,1}] & F_{p,1,2}T_{2,2} - T_{1,1}F_{p,1,2} \\ F_{p,2,1}T_{1,1} - T_{2,2}F_{p,2,1} & [F_{p,2,2}, T_{2,2}] \end{array} \right).$$

The requirement for the off-diagonal terms follows from the fact that $T_{1,1}$ and $T_{2,2}$ are first order differential operators with $C^{2K+1,1}$ coefficients. Composing these with an order $-(K - p)$ remainder term leads to an remainder term of order $-(K - p - 1)$ as required. As for the diagonal components, for the remainder term it follows in the same way as for the off-diagonal part. For the other terms it is used that the commutator has order that is one lower than the sum of the orders of the two operators, with coefficients that are one order less smooth. \qed

As mentioned we will construct a sequence $R^{(-k)}$, $k = 2, \ldots, K$. The construction will be done inductively. For the $R^{(-k)}$ we will have the following requirements on diagonal and off-diagonal terms:

$$\frac{d^l R^{(-k)}_{1,1}}{dt^l}, \quad \frac{d^l R^{(-k)}_{2,2}}{dt^l} \text{ are of the form (32) with } m_1 = -1, m_2 = (K - l), \quad (36)$$

and

$$\frac{d^l R^{(-k)}_{1,2}}{dt^l}, \quad \frac{d^l R^{(-k)}_{2,1}}{dt^l} \text{ are of the form (32) with } m_1 = -k + l, m_2 = (K - l). \quad (37)$$

When $k = K$ this reduces to the properties (33), (34), (35).
To start the induction we show that, under suitable hypotheses on $a, b$, $R^{(-1)}$ satisfies the requirement with $k = 1$.

**Lemma 7.2.** The operator $R^{(-1)}$ can be written in the above polyhomogeneous form for $m_1 = -1$ and $m_2 = K$ if $a, b$ are $C^{2K+1,1}$ coefficients.

**Proof.** The operators given above are not directly in this standard form. The standard form has the multiplication with coefficients to the left of the convolutional operators. But the standard form can be obtained by commuting the convolutional parts in the operators to the right. With derivatives this is no problem, this simply yields an extra term that involves the derivative of the coefficient that is commuted with. Commuting with the smoothing part is not necessary, the smoothing part is so strongly regularizing, that it is part of the remainder. But we need to commute $(\partial_x + S)^{-1}$ with coefficients. For this we use that $A^{-1}[A, B] = [B, A^{-1}]A$, hence

$$[(\partial_x + S)^{-1}, B] = -(\partial_x + S)^{-1}[\partial_x + S, B](\partial_x + S)^{-1}. $$

The commutator on the right hand side is with $\partial_x$ and $S$, so poses no problem. But the right hand side is not of the form with convolutional part of operator to the right hand side, however it is of lower order. We can continue commuting the $(\partial_x + S)^{-1}$ and generating lower order terms until the terms are of low enough order to be part of the remainder term.

The operator $R^{(-1)}$ is given by the composition of the operators $Q^{-1}$ and $QR$ that were given in section 2. The terms $(Q^{-1})_{j_1,2}(QR)_{2, j_2}$ give the order $-1$ terms, the other terms contain smoothing operators. As we require more regularity for the order $-1$ terms than for the remainder that contains the smoothing operator, these restrict the regularity of the coefficients. As mentioned, the off-diagonal terms in $R$ are of order $-1$. For the coefficient of this order $-1$ terms $C^{2K-1,1}$ smoothness is required. As these coefficients contain at most second order derivatives of $a, b$, it is sufficient if $a, b$ are $C^{2K+1,1}$.

Finally we have to prove the similar estimates for the time derivatives of $R$. These follow in the same way, observing that a time differentiation of a coefficient means losing one space-derivative in regularity.  

Finally we construct $Q^{(-2)}, \ldots, Q^{(-K)}, R^{(-2)}, \ldots, R^{(-K)}$.  

23
Lemma 7.3. There exist $Q^{(-k)}$, $R^{(-k)}$, $k = 2, \ldots, k$ such that $R^{(-k)}$ satisfies (36), (37).

Proof. We follow a strategy of Taylor [19, 18], eliminating the off-diagonal terms order by order until the desired order of smoothing is reached. Let's assume we have $R^{(-k)}$, $Q^{(-k)}$ and we want to construct $R^{(-k-1)}$, $Q^{(-k-1)}$. In the previous lemma it was shown that we have a starting point for this induction. We set

$$Q^{(-k-1)} = Q^{(k)} \begin{pmatrix} 1 & E \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ F & 1 \end{pmatrix} = Q^{(k)} \begin{pmatrix} 1 + EF & E \\ F & 1 \end{pmatrix},$$

with $E, F$ of order $-k - 1$, and

$$(Q^{(-k-1)})^{-1} = \begin{pmatrix} 1 & 0 \\ -F & 1 \end{pmatrix} \begin{pmatrix} 1 & -E \\ 0 & 1 \end{pmatrix} (Q^{(-k)})^{-1} = \begin{pmatrix} 1 & -E \\ -F & 1 + FE \end{pmatrix} (Q^{(-k)})^{-1}. \quad (38)$$

What is important in this formula is that there is an explicit formula for the inverse of $Q^{(-k-1)}$. By this modification of $Q$ we get similarly as above

$$T + R^{(-k-1)} = \begin{pmatrix} 1 & E \\ F & 1 \end{pmatrix}^{-1} (T + R^{(-k)}) \begin{pmatrix} 1 & E \\ F & 1 \end{pmatrix} - \begin{pmatrix} 1 & E \\ F & 1 \end{pmatrix}^{-1} \begin{pmatrix} 1 & E' \\ F' & 1 \end{pmatrix} + \text{lower order.}$$

Denote

$$T + R^{(-k)} = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

we have of course $A = \sqrt{b/a} \partial_x + \text{order}(0)$, $D = -\sqrt{b/a} + \text{order}(0)$. Then

$$T + R^{(-k-1)} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} + \begin{pmatrix} BF - EC & AE - ED \\ DF - FA & CE - FB \end{pmatrix} + \text{lower order.} \quad (39)$$

Clearly $E, F$ can be chosen such that $AE - ED$ cancels the order $-k$ contribution to $B$, and $DF - FA$ cancels the order $-k$ contribution to $C$, simply let $E = -r^{(-k)}_{1,2} / \sqrt{b/a} (\partial_x + S)^{-k-1}$ and $F = r^{(-k)}_{2,1} / \sqrt{b/a} (\partial_x + S)^{-k-1}$. It follows then that $R^{(-k-1)}$ has order $-k - 1$ off-diagonal part.

Finally the question is whether the new contributions to $R^{(-k-1)}$ have the appropriate form given by (32). From the induction assumption that $\frac{d}{dt} r^{(-k)}_{1,2}, \frac{d}{dt} r^{(-k)}_{2,1}$ have the appropriate smoothness (namely $C^{2(K-1)-k}$), and the smoothness of $\sqrt{b/a}$ this follows easily. This completes the proof. \qed
8 Discretization and operator approximation

The multiscale discretization will be done using wavelets. We follow the book of Cohen [5], which gives an excellent description of 1-D wavelet discretization theory, see also [7]. In a wavelet discretization functions in $L^2(\Omega)$ and $H^s(\Omega)$ are approximated by elements of increasingly large finite dimensional subspaces of $L^2(\Omega)$ given by a multiresolution analysis $V_j$, $j = 0, 1, 2, \ldots$. The spaces $V_j$ are spanned by translates and scalings of the scaling function $\phi$

$$\phi_{j,k} = 2^{j/2} \phi(2^j \cdot -k), \quad k \in \mathbb{Z}/(2^j L \mathbb{Z}).$$

The $V_j$ are assumed to form an increasing sequence $V_j \subset V_{j+1}$, $\bigcup_{j=0}^{\infty} V_j = L^2(\Omega)$.

In our case, where the domain is a circle of integer length $L$, the space $V_j$ has $L^{2^j}$ elements. We denote by $J$ the final level of discretization, so that $N = L^{2^J}$. Typically we will denote by $f_j$ an approximation of a function $f$ in $V_j$, and by $A_j$ the approximation of an operator $A$ on $V_j$.

The multiscale decomposition is obtained from the wavelet spaces. The wavelet space $W_j$ is such that $V_{j+1} = V_j \oplus W_j$. It is spanned by the translates and scalings $\psi_{j,k}$ of a mother wavelet $\psi$. This leads to the multiscale decomposition

$$V_j = V_0 \oplus W_0 \oplus \ldots \oplus W_{j-1}.$$

The scaling function can be chosen with compact support, and with any order $C^k$ smoothness. Together with the $V_j$, a dual multiresolution analysis $\tilde{V}_j$ can be constructed, spanned by translates and scalings of a dual scaling function $\tilde{\phi}$, such that the basis functions satisfy the biorthogonality property $\langle \tilde{\phi}_{j,k}, \phi_{j,k'} \rangle = \delta_{k,k'}$. One of $\phi, \tilde{\phi}$ can be chosen as a compactly supported spline, we assume $\phi$ is a spline, and $V$ is a splines space of a certain order. The space $V_j$ can be made to satisfy $V_j \subset H^s$ for any $s$ by choosing wavelets of sufficiently high order of smoothness. Throughout the analysis we will assume sufficient smoothness of the wavelets, without specifying this precisely.

The error estimates and assumptions on the smoothness of initial values are formulated in terms of regularity in $L^2$ based Sobolev spaces. That is natural and convenient for wave equations (where physical energy conservation holds). It is also easy to handle in wavelet discretizations, because of norm equivalences. The Sobolev norms $\| \cdot \|_{H^s}$ are equivalent
to weighted norms of the wavelet coefficients. If

\[ f = \sum_{k=0}^{K-1} c_{-1,k} \phi_{0,k} + \sum_{j=0}^{\infty} \sum_{k=0}^{2^j L} c_{j,k} \psi_{j,k}, \]

and the wavelets are sufficiently smooth, then there is the norm equivalence

\[ \|f\|_{H^\alpha(\Omega)}^2 \sim \sum_{j=-\infty}^{\infty} \sum_{k} |2^{\alpha j} c_{j,k}|^2. \]

From these norm equivalences one can easily derive an important approximation result.
Assume that \( f \) is in \( H^\alpha \), then the projections \( \Pi_{V_j} f \) of \( f \) to the \( V_j \) satisfy

\[ \|f - \Pi_{V_j} f\|_{L^2(\Omega)} \leq C 2^{-\alpha j} \|f\|_{H^\alpha(\Omega)}. \]

In our application we typically deal with products of operators that are applied after each other, in discrete form, to a discretized function. We first derive a criterion, that we call order \( k \) approximation operator, for each of the operators to satisfy, such that such products converge. After this we will argue that the operators in our application can be approximated such that the approximation indeed satisfies the approximation property.

Suppose \( A \) is some operator \( H^{s_1} \to H^{s_2} \), and \( A_j \) is a discrete approximation to \( A \). As pointed out, convergence estimates are done using additional regularity, say \( k \) additional orders of regularity. For our operator \( A \) from \( H^{s_1} \to H^{s_2} \) we therefore assume its argument, say \( f \) is in \( H^{s_1+k} \). The result \( Af \) may be the argument of another operator, so we will require \( Af \in H^{s_2+k} \), in other words we will assume

\[ A \text{ is continuous } H^{s_1+s} \to H^{s_2+s} \text{ for } 0 \leq s \leq k. \]

Next we discuss a property that ensures that \( A_j f_j \) approximates \( Af \) if \( f_j \) approximates \( f \).

**Definition** Let \( A \) be as just described, then we say \( A \) and \( A_j \) satisfy the order \( k \) approximation property if

\[ \|A - A_j\|_{H^{s_1+k} \to H^{s_2}} \leq C 2^{-jk}. \]

This also implies that \( A_j \) is continuous \( H^{s_1+k} \to H^{s_2+k} \), for \( 0 \leq s \leq k \). This implies that if a function \( f \in H^{s_1+k} \) is approximated in \( H^{s_1} \) by functions \( f_j \), with the convergence
as expected from the additional regularity, i.e. \( \| f - f_j \|_{H^{s_1}} \leq C 2^{-kj} \| f \|_{H^{s_1+k}} \), then \( A_j f_j \) approximates \( Af \) in the same way, since

\[
\| A f - A_j f_j \|_{H^{s_2}} \leq \| A_j (f - f_j) \|_{H^{s_2}} + \| (A - A_j) f \|_{H^{s_2}} \\
\leq C 2^{-jk} \| f \|_{H^{s_1+k}}.
\]

The basic operators needed here are partial differential operators, the operator \((-\partial_x^2 + 1)^{-1}\) or inverses of higher-order elliptic operator for the approximation of the operator \( S \) of section 7, and the pull back along the characteristic flow (which is a smooth coordinate transformation). Here we discuss partial differential operators and constant coefficient inverse partial differential operators, the pull back will be discussed in the last part of this section. We state the result on the approximation of \( R^{(-K)} \), \( Q^{(-K)} \) as a lemma.

**Lemma 8.1.** Assume the coefficients \( a, b \) are in \( C^{k+K+1,1} \). Then numerical approximations to the operators \( R^{(-K)} \) on \( H^1 \times H^1 \), \( Q^{(-K)} \) from \( H^1 \times H^1 \to H^1 \times L^2 \) and \( (Q^{(-K)})^{-1} \) from \( H^1 \times L^2 \to H^1 \times H^1 \) can be constructed that satisfy the order \( k \) approximation property.

**Proof.** Multiplication by polynomials and differentiation operators can be discretized using results of [8], see that reference or section 2.5 of [5]. They can be discretized at cost \( O(N) \), in such a way that the above order \( k \) approximation property is satisfied. For multiplication operators with other functions than polynomials, the coefficient is locally approximated by polynomials. As for the regularity requirement on the coefficients, for an approximate multiplication operator on \( H^{s_1} \) to have the order \( k \) approximation property, it sufficient to have \( C^{k+s_1-1,1} \) coefficients, since a \( C^{k-1,1} \) function can be approximated to error \( 2^{-jk} \) by polynomials on regions of size order \( 2^{-j} \).

In case of the approximation of \( R^{(-K)} \) on \( H^1 \times H^1 \), the coefficients in the remainder term need to be \( C^{k,1} \). It follows that the coefficients of the order -1 term in \( R \) need to be \( C^{k+K-1,1} \), and as these contain second derivatives of \( a, b \), it is sufficient if \( a \) and \( b \) are \( C^{k+K+1,1} \).

The operator \((-\partial_x^2 + 1)^{-1}\), can be computed in \( O(N) \) cost using a multigrid algorithm [1], a wavelet variant of this algorithm was given in [5]. To show that the approximation property holds, a slight change in the argument about multilevel preconditioning in example 4 in section 3.11 of [5] is needed, namely \( n_j \) is chosen such that \( \rho^{n_j} \leq 2^{-\nu j} \), with
\( t' > t \). Similar arguments work for higher-order inverse elliptic operators used in \( S \). This concludes the proof.

Next we will show a similar result for \( E(t, t_0) \). This operator was diagonal with \( E_{2,2} \) given by (see (27))

\[
E_{2,2}(t, t_0) u_2(x) = u_2(t_0, X(x, t_0, t)).
\] (40)
The 1,1 component of \( E(t, t_0) \) is given by a similar formula.

We will first discuss the approximation of \( X(x, t, t_0) \), then the next lemma will contain the result on \( E(t, t_0) \).

Let \( X_j(x, t, t_0) \) denote a numerical approximation used at level \( j \). This must be computed for a set of points \( x \). We require increasing accuracy as \( j \) increases, with error bounded by \( C2^{-j(k+1)} \). Also increasing cost, as \( 2^j \) is allowed. We find that for the time-independent case \( C^{k+1} \) smoothness of the coefficients is sufficient, while for the time-dependent case \( C^{2k+2} \) smoothness is sufficient for this computation, as we will now show.

For the time-independent case, the evaluation of (40) can be done by solving \( X = X(x, t, t_0) \) from

\[
\int_x^X c(\xi)^{-1} d\xi = t - t_0.
\] (41)
First the primitive \( \int_x^X c(\xi)^{-1} d\xi \) is computed for all \( x \) in the periodic grid with grid distance \( 2^{-j} \). Assuming that \( c \) is \( C^{k+1} \), this can be done at cost \( O(2^j) \), with error \( \leq C2^{-j(k+1)} \). Next the solution of (41) can be done for a set of \( 2^j \) points \( x \) using interpolation, which conserves the order of error, i.e. with error still bounded by \( C2^{-j(k+1)} \).

For the time-dependent case we solve for the characteristics using a Runge-Kutta method of order \( 2k + 2 \). We require \( C^{2k+2} \) smoothness of \( c \), then we can take order \( 2^{j/2} \) points with distance between them of \( 2^{-j/2} \), and solve with time steps of order \( 2^{-j/2} \). The total error is then bounded by \( C2^{-j(k+1)} \).

Next we discuss how (40) can be computed numerically such that the order \( k \) approximation property is satisfied.

**Lemma 8.2.** Assume the coefficients \( a, b \) are in \( C^{k+1} \) for the time-independent case or in \( C^{2k+2} \) in the time-dependent case, and the wavelets are order \( k + 1 \) splines. Then a
numerical approximation to the operator $E(t, t_0)$ on $H^1 \times H^1$ can be constructed that satisfies the order $k$ approximation property.

Proof. We consider the approximation at level $J$ of $E_{2,2}(t, t_0)f$, with $f$ an element of $V_J$. We have that $E_{2,2}(t, t_0)f(x) = f(X(x, t_0, t))$. For brevity we will write $X(x)$ instead of $X(x, t_0, t)$. We propose the following method: We construct local polynomial approximations of $h(x) = f(X(x))$, denote one such an approximation by $p(x)$. We want to compute $c_{J,k} = \langle \phi_{J,k}, h \rangle$. However this is difficult, instead we compute the coefficient of a polynomial. For each $c_{J,k}$ a local polynomial approximation $p(x)$ is constructed that is accurate around the support $S_{J,k}$ of the dual scaling function $\tilde{\phi}_{J,k}$. The approximate coefficient of the scaling function $\phi_{J,k}$ is then $\tilde{c}_{J,k} = \langle \tilde{\phi}_{J,k}, p \rangle$, and is obtained according to the method of section 2.5 of [5].

Before going more into detail about the construction of the local approximating polynomials, let’s discuss the error bounds that are needed. A bound on the error in operator norm from $H^{1+k}$ to $H^1$ by $C2^{-Jk}$ is needed. Fortunately there are the norm equivalences, and it is sufficient to consider the error in the approximation of $E_{2,2}\psi_{J,k}$ in $L^2$, and show that it can be bounded by $C^2k^{j+1}$. So we set $f = \psi_{J,k}$.

The numerical approximation of $p$ can simply be done by polynomial interpolation with an order $k-1$ polynomial. A function $h$ in $C^{k-1,1}$ can be approximated by interpolation on a grid of size $2^{-J}$ up to an error bounded by

$$\sup_{x \in S_{J,k}} |h(x) - p(x)| \leq C2^{-kJ}\|h\|_{C^{k-1,1}(S_{J,k})}.$$ 

The function $h(x) = \psi_{J,k}(X(x))$ satisfies $\|\psi_{J,k}(X(\cdot))\|_{C^{k-1,1}(S_{J,k})} \leq C2^{j(k+1/2)}$. Thus the error with $p$ an exact interpolating polynomial is given by

$$|c_{J,k} - \tilde{c}_{J,k}| \leq \|\tilde{\phi}_{J,k}\|_{L^1} \sup_{x \in S_{J,k}} |h(x) - p(x)| \leq C2^{J(1/2) + j(k+1/2)} \leq C2^{j+1}.$$ 

This estimate is sufficient.

However, also the evaluation of $h(x)$ will contain errors, since a numerical approximation $X_J(x)$ is used instead of the exact value $X(x)$. For these errors we have

$$f(X_J(x)) - f(X(x)) = \int_{X_J(x)}^{X(x)} \frac{df}{dx}(s)ds.$$
Since $\frac{df}{ds}$ is bounded by $C 2^{3j/2}$, and $|X_J(x) - X(x)| < C 2^{-J(k+1)}$, we have that these errors satisfy

$$|f(X_J(x)) - f(X(x))| \leq C 2^{3j/2 - J(k+1)}.$$ 

These errors are hence smaller than the interpolation errors and do not change the result. 

9 Multiscale time-stepping and proof of the Theorem

In this section multiscale time-stepping is introduced to finally obtain an $O(N)$ algorithm. The results of section 7 enable the use of higher-order time-stepping methods and lead to estimates for the time discretization errors. The results of section 8 allow to estimate the errors due to space discretization. Here we will combine space and time-discretization, choose parameters, like the order of space and time discretization and establish the complexity of the algorithm by estimating error and cost of the algorithm.

We solve the equivalent of differential equation (24) with higher-order decoupling, after the application of the integrating factor, i.e. we solve

$$\frac{dv}{dt}(t) = S(t, t_0)v(t),$$

with

$$S(t, t_0) = E(t, t_0)^{-1}R(-K)E(t, t_0),$$

where $R(-K)$ is as constructed in section 7. We will approximate the solution $v(t_1)$ starting from $t_0$. The approximation is done in $H^1 \times H^1$. The initial values $v_0 = u_0$ also must be in $H^1 \times H^1$. We assume they have $\alpha$ additional orders of regularity, i.e. they are in fact in $H^{1+\alpha} \times H^{1+\alpha}$. It follows from the results of sections 7 and 8 that we can transform the values $U(t)$ of the original system (3) to those of the transformed system (42) and back with complexity $O(N)$.

Operators will be approximated with the order $k$ approximation property, with $k > \alpha$. A minimum value for $k$ is derived below. Regularity assumptions follow from these assumptions according to the previous sections. Note that this is different from the previous section, where the order $k$ corresponded to the order of additional regularity of functions that operators acted on, while here $k > \alpha$. 

30
In general in an integrating factor method it is common to frequently reset \( t_0 \), so that \( E(t, t_0) \) propagates only over small time intervals. We will refrain from doing so, as this is not needed in this context, and the frequent application of \( E(t, t_0) \) to the full signal (i.e. not only the addition made during a small time interval by a Runge-Kutta time step), may cause additional errors.

As motivated in section 6, we will make a multiscale decomposition of the signal and do time-stepping separately for each scale. The initial values are decomposed as follows:

\[
    u_0 = \sum_{j=0}^{J} w_{0,j},
\]

with \( w_{0,0} = \Pi_{V_0} u_0 \), and \( w_{0,j} = \Pi_{W_{j-1}} u_0 \), for \( j = 1, \ldots, J \). Here \( \Pi_{V_j}, \Pi_{W_j} \) denote the projection on \( V_j \times V_j \) and \( W_j \times W_j \) respectively. The field \( v(t) \) will also be decomposed. The \( j \)-th component, corresponding to initial values in \( W_{j-1} \times W_{j-1} \), will not be in \( V_j \times V_j \) however (nor in \( W_{j-1} \times W_{j-1} \)), but in a space \( V_{l(j)} \times V_{l(j)} \), \( j \leq l(j) \leq J \). To indicate this we write the components of the sum as \( v_{j,l(j)} \). We will show that \( v(t) \) can be approximated

\[
    v(t) \approx \sum_{j=0}^{J} v_{j,l(j)}.
\]

The motivation for doing this is simple: Large errors would result in the time propagation in \( V_j \times V_j \) of the \( w_{0,j} \), while large cost would result if we would work in the full space \( V_j \times V_j \). By working in an intermediate space both cost and errors can be controlled.

The final numerical approximation will be a sum of components \( w_{j,l(j)}, \Delta t_j \). The terms describe the discrete time propagation with time step \( \Delta t_j \), using the space discretized operators \( S_{l(j)}(t) \), applied to the initial values \( w_{0,j} \).

For purposes of error estimation we consider two sets of fields in addition to \( w_{j,l(j)}, \Delta t_j \). We assume the fields \( v_{j,l(j)} \) introduced above describe the continuous time propagation of the operator \( \Pi_{V_{l(j)}} S \Pi_{V_{l(j)}} \), and the field \( v_{j,l(j)}, \Delta t_j \) will describe the discrete time propagation of \( \Pi_{V_{l(j)}} S \Pi_{V_{l(j)}} \).

We first establish that \( v_J(t_1) \) can be approximated like

\[
    v_J(t_1) \approx \sum_{k=0}^{J} v_{j,l(j)}(t_1).
\]
Lemma 9.1. Suppose $l(j)$ is such that

$$k(l(j) - j) = \alpha(J - j).$$  \hfill (43)

Then

$$\left\| \sum_{k=0}^{j} v_{j,k}(t_1) - v(t_1) \right\|_{H^1 \times H^1} \leq C2^{-\alpha j} \left\| u_0 \right\|_{H^{1+\alpha} \times H^{1+\alpha}}.$$  \hfill (44)

Proof. Let $v_{j,\infty}$ denote the solution of the exact differential equation with initial value $w_{0,j}$. It satisfies $\frac{dv_{j,\infty}}{dt} = S v_{j,\infty}$. As $S$ is bounded on $H^{1+s} \times H^{1+s}$, $0 \leq s \leq k$ it follows that $v_{j,\infty}(t)$ satisfies the bound

$$\left\| v_{j,\infty}(t) \right\|_{H^{1+s} \times H^{1+s}} \leq C \left\| w_{0,j} \right\|_{H^{1+s} \times H^{1+s}}.$$

for $0 \leq s \leq k$, $t_0 \leq t \leq t_1$.

We have $\frac{dv_{j,\infty}}{dt} = \Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} v_{j,\infty}$, so the difference $v_{j,l(j)} - v_{j,\infty}$ satisfies

$$\frac{dv_{j,l(j)} - v_{j,\infty}}{dt} = \Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} (v_{j,l(j)} - v_{j,\infty}) + (\Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} - S) v_{j,\infty}. \hfill (45)$$

By standard estimates for ODEs we have that

$$\left\| v_{j,l(j)}(t) - v_{j,\infty}(t) \right\|_{H^{1+s} \times H^{1+s}} \leq C_1 \left\| v_{j,l(j)}(t_0) - v_{j,\infty}(t_0) \right\|_{H^{1+s} \times H^{1+s}}$$

$$+ C_2 \int_{t_0}^{t} \left\| (\Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} - S) v_{j,\infty}(s) \right\|_{H^{1+s} \times H^{1+s}} ds.$$

The first term on the right hand side is zero. For the second term we use that by the regularity assumptions we have

$$\left\| \Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} - S \right\|_{H^{1+k} \times H^{1+k} \rightarrow H^1 \times H^1} \leq C 2^{-k(l(j))}, \hfill (46)$$

The components of the initial values $w_{0,j}$ are bounded according to

$$\left\| w_{0,j} \right\|_{H^{1+k} \times H^{1+k}} \leq C 2^{j(k-\alpha)} \left\| w_{0,j} \right\|_{H^{1+\alpha}}, \hfill (47)$$

and the same is true for $v_{j,\infty}(t)$ for $t_0 < t < t_1$. The inhomogeneous term in (45) can therefore be bounded by

$$\left\| (\Pi_{V_{i(j)}} S \Pi_{V_{i(j)}} - S) v_{j,\infty}(t) \right\|_{H^1 \times H^1} \leq C 2^{-kl(j)+j(k-\alpha)} \left\| w_{0,j} \right\|_{H^{1+\alpha} \times H^{1+\alpha}}$$

$$= C 2^{-\alpha j} \left\| w_{0,j} \right\|_{H^{1+\alpha} \times H^{1+\alpha}}.$$
The error \( v_{j,l}(t_1) - v_{j,\infty}(t_1) \) therefore satisfies the bound
\[
\|v_{j,\infty}(t_1) - v_{j,l}(t_1)\|_{H^1 \times H^1} \leq C2^{-\alpha J}\|w_{0,j}\|_{H^{1+\alpha} \times H^{1+\alpha}}.
\] (48)

Adding the estimates for each \( j \) results in (44).

The second step in the estimation of the error is to estimate the time discretization error for the field \( v_{j,l}(j) \). We will argue that the fields \( v_{j,l}(j) \) can be sufficiently accurately approximated using Runge-Kutta time-discretization. By \( v_{j,l}(j), \Delta t_j \) we denote the time-discretized fields. We assume we use an order \( K \) Runge-Kutta method for the time-stepping.

**Lemma 9.2.** Suppose that the time step \( \Delta t_j \) satisfies the inequality
\[
\Delta t_j \leq C2^{-\alpha(J-j)/K},
\] (49)

then we have
\[
\left\| \sum_{j=0}^{J} v_{j,l}(j), \Delta t_j(t_1) - \sum_{j=0}^{J} v_{j,l}(j)(t_1) \right\|_{H^1 \times H^1} \leq C2^{-\alpha J}\|u_0\|_{H^{1+\alpha} \times H^{1+\alpha}}.
\] (50)

**Proof.** The error per time step in
\[
\|v_{j,l}(j) - v_{j,l}(j), \Delta t_j\|_{H^1 \times H^1}
\]
is bounded by
\[
(\Delta t_j)^{K+1} \sup_{\tau \in [t, t+\Delta t_j]} \left\| \frac{d^{K+1}v_j(\tau)}{dt^{K+1}} \right\|_{H^1 \times H^1}.
\]

In section 7 it was shown that the time derivatives \( \frac{d^j S}{dt^j} \Pi V_{l,(j)} S \Pi V_{l,(j)} \) were bounded, \( j = 0, \ldots, K \). The same is true for \( \frac{d^j S}{dt^j} \Pi V_{l,(j)} S \Pi V_{l,(j)} \).

It follows that the error at time \( t_1 \) satisfies
\[
\|v_{j,l}(j)(t_1) - v_{j,l}(j), \Delta t_j(t_1)\|_{H^1 \times H^1} \leq C(\Delta t_j)^K\|w_{0,j}\|_{H^1 \times H^1}.
\]

We have that
\[
\sum_{j=0}^{J} (2^{\alpha j}\|w_{0,j}\|_{H^1 \times H^1})^2
\]
is bounded. We therefore require that
\[
(\Delta t_j)^K \leq C2^{\alpha J}2^{-\alpha J},
\] (51)

then (50) follows. The conditions (49) and (51) are of course equivalent. \( \square \)
For the estimate of the time discretization error it turned out to be convenient to work with \( \Pi_{V_l(j)} S \Pi_{V_l(j)} \), an exact discretization that is not practical to compute, instead of \( S_{l(j)} \), the approximate discretization discussed in section 8. The reason is that the errors made in \( S_{l(j)} \) are not differentiable. So the next step is to take into account the difference between \( S_{l(j)} \) and \( \Pi_{V_l(j)} S \Pi_{V_l(j)} \).

**Lemma 9.3.** Assume still (43). We have the estimate

\[
\left\| \sum_{j=0}^{J} w_{j,l(j),\Delta t_j}(T) - \sum_{j=0}^{J} v_{j,l(j),\Delta t_j}(T) \right\|_{H^1 \times H^1} \leq C 2^{-\alpha J} \| u_0 \|_{H^{1+\alpha} \times H^{1+\alpha}}. \tag{52}
\]

**Proof.** The difference \( S_{l(j)} - \Pi_{V_l(j)} S \Pi_{V_l(j)} \) satisfies a similar estimate as the difference \( \Pi_{V_l(j)} S \Pi_{V_l(j)} - S \), which was considered in the proof of Lemma 9.1. The proof of (52) therefore proceeds similarly as the proof of Lemma 9.1, except that difference equations are used instead of differential equations. The difference \( w_{j,l(j),\Delta t_j} - v_{j,l(j),\Delta t_j} \) satisfies the linear inhomogeneous difference equation

\[ w_{j,l(j),\Delta t_j}(t + \Delta t) - v_{j,l(j),\Delta t_j}(t + \Delta t) = \Delta t \text{RKStep}(t, \Delta t, S_{l(j)})(w_{j,l(j),\Delta t_j}(t) - v_{j,l(j),\Delta t_j}(t)) + \Delta t(\text{RKStep}(t, \Delta t, S_{l(j)}) - \text{RKStep}(t, \Delta t, \Pi_{V_l(j)} S \Pi_{V_l(j)}))v_{j,l(j),\Delta t_j}(t), \]

where \( \Delta t \text{RKStep}(t, \Delta t, A)y \) denotes the Runge-Kutta step for the equation \( y' = Ay \), which is a linear map on \( y \). It follows that

\[
\left\| w_{j,l(j),\Delta t_j}(\hat{t}) - v_{j,l(j),\Delta t_j}(\hat{t}) \right\|_{H^1 \times H^1} \leq C \Delta t_j \sum_{t \text{-values } < \hat{t}} \left\| \left( \text{RKStep}(t, \Delta t, S_{l(j)}) - \text{RKStep}(t, \Delta t, \Pi_{V_l(j)} S \Pi_{V_l(j)}) \right) v_{j,l(j),\Delta t_j}(t) \right\|_{H^1 \times H^1}.
\]

The difference between the RKStep operators satisfies

\[
\left\| \text{RKStep}(t, \Delta t, S_{l(j)}) - \text{RKStep}(t, \Delta t, \Pi_{V_l(j)} S \Pi_{V_l(j)}) \right\|_{H^{1+k} \times H^{1+k} \to H^1 \times H^1} \leq C 2^{-kl(j)},
\]

while we have that

\[
\left\| v_{j,l(j),\Delta t_j}(t) \right\|_{H^{1+k} \times H^{1+k}} \leq 2^{j(k-\alpha)} \left\| w_{0,j} \right\|_{H^{1+\alpha}}.
\]

It follows that we can estimate

\[
\left\| w_{j,l(j),\Delta t_j}(t_1) - v_{j,l(j),\Delta t_j}(t_1) \right\| \leq C 2^{-kl(j)+j(k-\alpha)} \left\| w_{0,j} \right\|_{H^{1+\alpha} \times H^{1+\alpha}} = C 2^{-\alpha J} \left\| w_{0,j} \right\|_{H^{1+\alpha} \times H^{1+\alpha}}
\]

The estimate (52) trivially follows from this. \( \square \)
This ends our estimation of the error. The cost of this time-stepping is

\[ C(\Delta t_j)^{-1} 2^{l(j)} = C \sum_{j=0}^{J} 2^{\alpha(J-j)/K + \alpha(j-j)/k+j} \]

\[ = C 2^J \sum_{j=0}^{J} 2^{(-1+\alpha/K + \alpha/k)(J-j)} \]

The requirement is that the cost is bounded by \( CN \). hence that \(-1 + \alpha/K + \alpha/k < 0\). If we allow logarithmic cost \( O(N \log N) \), equality is also allowed. We hence have our final result:

**Theorem 9.4.** *The above described method has complexity \( O(N) \) if*

\[ 1/K + 1/k < 1/\alpha, \] (53)

*and complexity \( O(N \log N) \) if*

\[ 1/K + 1/k = 1/\alpha. \] (54)

**10 Discussion**

A numerical method for wave propagation in smooth media was developed. The numerical results in section 5 show that the method certainly has potential in applications with relatively smooth media. Further improvements might be possible to further improve computation speed, or weaken the requirements of medium smoothness. One step that could possibly give an improvement is a coordinate change that makes the wavespeed equal to unity. We refrained from doing this since it has no equivalent in higher dimensions, but it could reduce the error in the application of the operator \( T \).

The material of sections 6 to 9 not only leads to the \( O(N) \) complexity result, but also suggests ways to possibly improve the method.

The main question for future research is in my view about the generalization to higher-dimensional cases. For the multidimensional case, curvelets form a redundant basis (frame) with respect to which the solution operator can be made sparse [4]. Potentially it could be used for computations. However, one needs to be able to implement operators that give the
approximate effect of wave propagation, such as translation, rotation and deformation efficiently in a curvelet basis. Perhaps other fast implementations of Fourier integral operators could be used (cf. [3]) to compute the approximate wave propagation. In dimension two and higher the remainder operator $R$ becomes, at least in the continuous setting, a pseudodifferential operator, which is more challenging to implement. But a priori there is no reason why the principle of combining an approximate solution operator with lower order, exact “corrections” could not be extended to higher dimensions.

Acknowledgement

This research was supported by the Netherlands Organisation for Scientific Research through a VIDI grant no. 639.032.509.

References


