REVISITING IMPLICIT FINITE DIFFERENCE SCHEMES FOR 3-D ROOM ACOUSTICS SIMULATIONS ON GPU

Brian Hamilton,[*] *Stefan Bilbao,*
Acoustics and Audio Group,
University of Edinburgh
`first.lastname@ed.ac.uk`

Craig J. Webb,
Oxford e-Research Centre
University of Oxford
`craig.webb@oerc.ox.ac.uk`

ABSTRACT

Implicit finite difference schemes for the 3-D wave equation using a 27-point stencil on the cubic grid are presented, for use in room acoustics modelling and artificial reverberation. The system of equations that arises from the implicit formulation is solved using the Jacobi iterative method. Numerical dispersion is analysed and computational efficiency is compared to second-order accurate 27-point explicit schemes. Timing results from GPU implementations demonstrate that the proposed algorithms scale over their explicit counterparts as expected: by a factor of $M + 2$, where M is a fixed number of Jacobi iterations (eight can be sufficient in single precision). Thus, the accuracy of the approximation can be improved over explicit counterparts with only a linear increase in computational costs, rather than the quartic (in operations) and cubic (in memory) increases incurred when oversampling the grid. These implicit schemes are advantageous in situations where less than 1% dispersion error is desired.

1. INTRODUCTION

Room acoustics simulations are important for the purposes of auralization and artificial reverberation. There are many models and techniques used in room acoustics simulations; see [1, 2] for a review. One popular starting point for room acoustics modelling is the second-order scalar wave equation with impedance boundary conditions [3]. This model problem can be discretised with finite difference (FD) operators on regular spatial grids, and solutions can be approximated through explicit (leapfrog) time integration [4] at a sample rate of choice (e.g. 44.1 kHz). Explicit time-stepping FD methods have been used extensively in the literature for simulating room acoustics [5, 6, 7] in various equivalent formulations [8, 9, 10, 11].

FD methods can be computationally expensive for large 3-D spaces due to the fact that the solution is approximated for the entire domain at each time-step. Furthermore, numerical dispersion affects the approximation, to a large degree, in high frequencies. This may require that the spatial grid be oversampled, which incurs cubic increases in memory usage and quartic increases in the operation count. Explicit schemes are well-suited to implementation on graphics processing units (GPU), allowing for real-time low-frequency [12] and offline full-bandwidth applications [13].

Numerical dispersion can be improved by employing implicit generalisations of explicit schemes [14, 15], however, implicit schemes require a linear system to be solved at each time-step. This extra burden at each time-step can be alleviated somewhat when the implicit scheme allows for an alternating direction implicit (ADI) decomposition [16, 15], since the overall system in 3-D ADI schemes can be decomposed into three tridiagonal systems that can be solved efficiently with the Thomas algorithm [16]. However, the Thomas algorithm is serial in nature, so it is not easily parallelised. Furthermore, the formulation of impedance boundary conditions that are compatible with the ADI decomposition and the Thomas algorithm seems to be an open problem [7]. On the other hand, simple iterative methods [17] can be employed to tackle the implicit system, free from ADI constraints. The Jacobi method is a simple iterative method whose iterations reduce to sparse matrix-vector multiplications (SpMV) that are easily parallelised on a GPU. The purpose of this paper is then to revisit implicit schemes in the context of the Jacobi method and identify schemes that are suitable for room acoustics applications and GPU implementations.

This paper is laid out as follows. The model problem is introduced in Section 2, followed by the implicit finite difference schemes in Section 3 and conditions for stability in Section 4. The Jacobi method is described in Section 5, and optimal parameters for the implicit schemes are chosen in Section 6. Numerical dispersion and computational efficiency are analysed in Section 7. In Section 8, numerical experiments are conducted in order to validate the implicit schemes, check convergence rates for the Jacobi solve, and test the stability of the proposed schemes in finite precision arithmetic. Section 9 presents timing results from CUDA implementations of the implicit schemes on an Nvidia Tesla K20 GPU card, followed by conclusions and future work in Section 10.

2. MODEL PROBLEM

2.1. Initial and boundary value problem

The 3-D wave equation can be written as

$$\Box\Psi := \partial_t^2\Psi - c^2\Delta\Psi = 0\,. \tag{1}$$

Here, t is time and $t \in \mathbb{R}^+$, $\boldsymbol{x} := (x, y, z) \in \mathbb{R}^3$, c is the wave speed, assumed to be a constant, and Δ is the 3-D Laplacian operator, $\Delta := \partial_x^2 + \partial_y^2 + \partial_z^2$. The box symbol ($\Box$) represents the d'Alembert operator and the scalar field $\Psi = \Psi(t, \boldsymbol{x})$ represents the acoustic velocity potential [3]. Two initial conditions, $\Psi(0, \boldsymbol{x})$ and $\partial_t\Psi(0, \boldsymbol{x})$, are required to complete the initial value problem (IVP).

For the boundary value problem (BVP), let $\mathcal{V}$ denote a closed 3-D volume and $\partial\mathcal{V}$ its boundary. Frequency-independent lossy boundaries can be written as

$$\boldsymbol{n}\cdot\nabla\Psi = (\gamma/c)\partial_t\Psi\,, \quad \boldsymbol{x} \in \partial\mathcal{V}\,, \tag{2}$$

where γ represents the *specific acoustic admittance* with $\gamma \in \mathbb{R}$, $\gamma \geq 0$ and where $\boldsymbol{n}$ is the outward normal vector at $\boldsymbol{x} \in \partial\mathcal{V}$.

[*] This work was supported by the European Research Council, under grant StG-2011-279068-NESS, and by the Natural Sciences and Engineering Research Council of Canada.

These become first-order absorbing boundary conditions of the Engquist-Majda type for $\gamma = 1$. When $\gamma = 0$ the condition (2) is a homogeneous Neumann (lossless) boundary condition.

3. AN IMPLICIT FINITE DIFFERENCE SCHEME

3.1. Discretising time and space

Time can be discretised by restricting t to the grid of points $\mathbb{T} := \{nk, n \in \mathbb{Z}^+\}$, where k is the time-step, and the spatial domain can be discretised using a cubic grid: $\mathbb{G} := h\mathbb{Z}^3$. The finite spatial grid to consider can then be written as $\overline{\mathbb{G}} := \mathbb{G} \cap \mathcal{V}$. For the purposes of this paper, the closed volume of interest will be a box-shaped room. Furthermore, it will be assumed, for convenience and comparison with published literature [7], that the "boundary nodes" of the grid are precisely on the boundary $\partial\mathcal{V}$.

3.2. Difference operators

Let $u(t, \boldsymbol{x})$, which will be restricted to $\mathbb{T} \times \mathbb{G}$ or $\mathbb{T} \times \overline{\mathbb{G}}$, represent an approximation to the solution of interest $\Psi(t, \boldsymbol{x})$. A time-shift operator may be defined as

$$e_{t\pm} u := u(t \pm k, \boldsymbol{x}) \,, \tag{3}$$

and the following abbreviation will be employed throughout this paper: $u^{\pm} := u(t \pm k, \boldsymbol{x})$. Centered time-difference operators can be written as

$$\delta_{t\cdot} := (e_{t+} - e_{t-})/(2k) = \partial_t + O(k^2) \,, \tag{4a}$$

$$\delta_{tt} := (e_{t+} - 2 + e_{t-})/k^2 = \partial_t^2 + O(k^2) \,. \tag{4b}$$

A parameterised 27-point discrete Laplacian (stencil) can be defined on the cubic grid as

$$\delta_{\Delta} u := \sum_q \frac{6\alpha_q}{|\Omega_q| q h^2} \sum_{\boldsymbol{v} \in \Omega_q} (u(t, \boldsymbol{x}+\boldsymbol{v}h) - u(t, \boldsymbol{x})) = \Delta u + O(h^2) \,, \tag{5}$$

where $q \in \{1, 2, 3\}$, $\Omega_q := \{\boldsymbol{x} \in \mathbb{Z}^3 : \|\boldsymbol{x}\|^2 = q\}$, where $|\Omega_q|$ denotes the cardinality of the set Ω_q, and $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{R}^3$. $\sum_q \alpha_q = 1$ is required for consistency. The 27-point stencil vectors are displayed in Fig. 1.

3.3. Difference scheme for IVP

An implicit finite difference scheme for (1) can now be written as

$$\delta_{\Box} u := \left(1 + \beta h^2 \delta_{\Delta}\right) \delta_{tt} u - c^2 \delta_{\Delta} u = 0 \,, \quad (t, \boldsymbol{x}) \in \mathbb{T} \times \mathbb{G} \,, \tag{6}$$

where $\beta \in \mathbb{R}$ is a free parameter. The scheme is consistent since $\delta_{\Box} u \to \Box u$ as $h \to 0$ for a fixed Courant number $\lambda := ck/h$. Starting from the two known (or approximated) values $u(0, \boldsymbol{x})$ and $u(k, \boldsymbol{x})$ determined by the initial conditions, the unknown variable u^+ is related to the two previous states by

$$(1 + \beta \delta_{\Delta}^h) u^+ = ((\lambda^2 + 2\beta)\delta_{\Delta}^h + 2) u - (1 + \beta \delta_{\Delta}^h) u^- \,, \tag{7}$$

where $\delta_{\Delta}^h := h^2 \delta_{\Delta}$. The unknown variable cannot be isolated algebraically unless $\beta = 0$, in which case the scheme is explicit. For $\beta \neq 0$ the scheme is implicit, and a linear system of equations must be solved at each time-step. This family of implicit schemes generalises the 27-point compact explicit schemes analysed in [7]. The operator δ_{Δ}^h expressed in a similar notation to that used in [7] can be found in the Appendix.

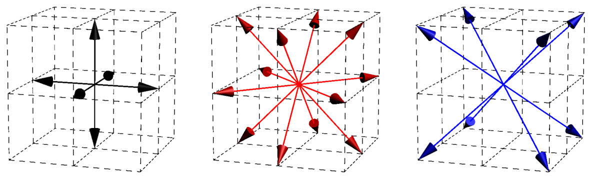


Figure 1: Stencil vectors for δ_{Δ}: Ω_1 (black), Ω_2 (red), Ω_3 (blue)

3.4. Matrix update for BVP

Imposing the boundary condition (2) reduces (7) to a finite system of equations which can be written as a matrix update. The approximation for the BVP at a particular time t can be written as the $N \times 1$ vector $\mathbf{u}$ with the values of u for $\boldsymbol{x} \in \overline{\mathbb{G}}$ ($N = |\overline{\mathbb{G}}|$). Similarly, $\mathbf{u}^{\pm}$ is a vector of $u^{\pm}$ values. The operator ∂_t in the lossy case (2) can be discretised with $\delta_{t\cdot}$ and the spatial derivatives are approximated with centered spatial differences, following [7]. The update equation in matrix form becomes

$$(\gamma\lambda\mathbf{Q} + \mathbf{I} + \beta\mathbf{L})\mathbf{u}^+ = ((\lambda^2 + 2\beta)\mathbf{L} + 2\mathbf{I})\mathbf{u} + (\gamma\lambda\mathbf{Q} - \mathbf{I} - \beta\mathbf{L})\mathbf{u}^- \,, \tag{8}$$

where $\mathbf{L}$ is the $N \times N$ Laplacian matrix corresponding to δ_{Δ}^h with discretised Neumann conditions, $\mathbf{I}$ is the $N \times N$ identity matrix, and $\mathbf{Q}$ is a non-negative diagonal matrix. Constructions for the matrices $\mathbf{L}$ and $\mathbf{Q}$ are given in the Appendix. This matrix update encapsulates the point-wise explicit updates presented in [7] for interior, wall, edge, and corner nodes, in the special case of frequency-independent boundaries.

4. NUMERICAL STABILITY

4.1. Stability for the IVP

First we consider stability conditions for the initial value problem. The recursion in (7) must be numerically stable for $\|u - \Psi\|_h \to 0$ as $h \to 0$ (for λ fixed) by the Lax-Richtmyer theorem, where $\|f\|_h$ denotes the spatial L^2-norm of $f(\boldsymbol{x})$ on $\mathbb{G}$ or $\overline{\mathbb{G}}$. Stability conditions for (7) can be found by taking the Z-transform in time and the Fourier transform in space [18]. After some cancellation we obtain the following quadratic in $z \in \mathbb{C}$:

$$(1 - 4\beta\Lambda)z + 4\Lambda(\lambda^2 + 2\beta) - 2 + (1 - 4\beta\Lambda)z^{-1} = 0 \,, \tag{9}$$

where Λ is the Fourier symbol of the operator $-\frac{1}{4}\delta_{\Delta}^h$. Solving for the roots of the quadratic (9) it can be shown [8] that $|z| \leq 1$ as long as

$$\left| \frac{4\Lambda(\lambda^2 + 2\beta) - 2}{1 - 4\beta\Lambda} \right| \leq 2 \,, \tag{10}$$

given that Λ is non-negative, which is satisfied when

$$-2\alpha_1 \leq \alpha_2 \leq 2\alpha_1 + 1 \,. \tag{11}$$

Condition (10) then simplifies to the following

$$\beta < \frac{1}{4\Lambda_{\max}} \,, \quad \lambda \leq \lambda_{\max} := \sqrt{\frac{1}{\Lambda_{\max}} - 4\beta} \,, \tag{12}$$

where $\Lambda_{\max} := \max_{\boldsymbol{\xi}} \Lambda$ for the spatial frequencies $\boldsymbol{\xi} \in \mathbb{R}^3$. We can extract $\Lambda_{\max}$ from previous studies [8] since this must reduce to the explicit case when $\beta = 0$. We have then

$$\Lambda_{\max} = \max(1, 2\alpha_1 + \alpha_2, 2\alpha_1 - \alpha_2 + 1) \,. \tag{13}$$

Note, the stability conditions allow linear growth in the solution, but this is valid since linear growth is permitted in the underlying system [18].

4.2. Stability for the BVP

Stability conditions for the lossless boundary value problem are straightforward to obtain using the matrix method [18]. Using the ansatz $\mathbf{u} = z\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is an eigenvector of $\mathbf{L}$, we get, analogous to (9), the following quadratic in z with matrix coefficients:

$$z(\mathbf{I}+\beta\mathbf{L})\boldsymbol{\phi} - ((\lambda^2+2\beta)\mathbf{L}+2\mathbf{I})\boldsymbol{\phi} + z^{-1}(\mathbf{I}+\beta\mathbf{L})\boldsymbol{\phi} = 0\,. \quad (14)$$

This can be reduced to a set of decoupled scalar equations, and thus, we can obtain a sufficient condition for stability in terms of the spectrum of $\mathbf{L}$. Comparing with (9) we can see that (12) is sufficient for stability as long as $\mathbf{L} \leq 0$ (negative semi-definite) and $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$, where $\rho(\mathbf{L})$ denotes the spectral radius of $\mathbf{L}$. The first condition on $\mathbf{L}$ is easily verified using Gerschgorin's theorem [17]. That the condition $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$ is satisfied, with $\mathbf{L}$ as defined in the Appendix, follows from stability in the explicit case [7].

A matrix-type stability analysis becomes more difficult after including the additional matrix $\mathbf{Q}$, with $\gamma > 0$ for lossy boundaries, because the matrix coefficients of the resulting quadratic equation no longer commute. It is possible to show, through the use of energy techniques [19, 10] or by investigating reflection coefficients at the boundaries [7], that the lossy case is stable as long as the lossless case is stable and the boundaries remain passive ($\gamma \geq 0$). A detailed proof is left out for brevity.

5. SOLVING THE LINEAR SYSTEM

5.1. Jacobi method

To solve the linear system of equations with the Jacobi method we first write (8) in the form $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = (\gamma\lambda\mathbf{Q} + \mathbf{I} + \beta\mathbf{L})\,, \quad \mathbf{x} = \mathbf{u}^+\,, \quad (15a)$$

$$\mathbf{b} = ((\lambda^2 + 2\beta)\mathbf{L} + 2\mathbf{I})\mathbf{u} + (\gamma\lambda\mathbf{Q} - \mathbf{I} - \beta\mathbf{L})\mathbf{u}^-\,. \quad (15b)$$

Next, we use the matrix splitting $\mathbf{A} = \mathbf{D} - \mathbf{N}$ where $\mathbf{D}$ is a diagonal matrix with just the main diagonal of $\mathbf{A}$. Starting from any initial guess $\mathbf{x}^0$ (a good choice is $\mathbf{x}^0 = \mathbf{u}$), the Jacobi iterative solve proceeds with

$$\mathbf{x}^{n+1} = \mathbf{Hx}^n + \mathbf{b}'\,, \quad (16)$$

where $\mathbf{H} = \mathbf{D}^{-1}\mathbf{N}$ is the iteration matrix (sparse), $\mathbf{b}' = \mathbf{D}^{-1}\mathbf{b}$ and where the superscript n on $\mathbf{x}^n$ denotes the nth iteration ($n \geq 0$). Note that $\mathbf{b}'$ only needs to be computed once per time-step. The entire iterative solve can be accomplished with only four states stored in memory since the space in memory that is used to store $\mathbf{u}^-$ can be overwritten after $\mathbf{b}'$ has been calculated. This Jacobi solve requires two SpMVs to compute $\mathbf{b}'$ and M subsequent SpMVs, where M is the number of iterations. Thus, the increase in operations over the explicit case is a factor of $M + 2$. The memory increase over the explicit case is a factor of two.

The iterative solve can be halted when the following condition on the relative error is satisfied:

$$\frac{\|\mathbf{b} - \mathbf{Ax}^{n+1}\|_h}{\|\mathbf{b}\|_h} \leq E\,, \quad \|\mathbf{b}\|_h > 0\,, \quad (17)$$

where E is some threshold, such as IEEE 754 single precision machine epsilon, $\varepsilon_s \approx 1.2 \times 10^{-7}$, or double precision machine epsilon, $\varepsilon_d \approx 2.2 \times 10^{-16}$. Calculating the relative residual requires one additional SpMV per iteration, as well as the calculation of two discrete norms.

It is worth pointing out that while we use a matrix representation to illustrate the iterative method, a practical implementation does not require construction or storage of the matrices involved. For practical implementations, one can 'unroll' each SpMV into a (parallelisable) for-loop, as in the explicit case [13]. In fact, the explicit case is expressed by a single Jacobi iteration ($\beta = 0 \Rightarrow \mathbf{H} = 0$). The matrices involved are sparse and have entries that are mostly constant or zero along the diagonals, and the non-zero entries change only for boundary nodes. The storage of these constants is negligible. Point-wise updates can be extracted from the matrices in the Appendix, or derived from the explicit case in [7], so they are left out for brevity.

5.2. Convergence of the Jacobi method

The Jacobi iterations will converge from any initial guess $\mathbf{x}^0$ as long as the matrix $\mathbf{A}$ is *diagonally dominant* [17]. For a diagonally dominant $\mathbf{A}$, in the lossless case, we require that

$$\left|1 - 6\beta\sum_q \frac{\alpha_q}{q}\right| \geq 6\sum_q \frac{|\beta\alpha_q|}{q}\,. \quad (18)$$

If we assume that $\alpha_q \geq 0$ then this reduces to

$$|\beta| < \frac{1}{12}\,, \quad (19)$$

and in the general case $|\beta|$ has to be sufficiently small. By examining $\mathbf{L}$ it can be seen that the rows pertaining to boundary nodes will not change (19). This is also left out for brevity.

It can be shown that with each iteration the residual decreases by a factor of approximately $1/\rho(\mathbf{H})$ [17], and using Gerschgorin's theorem the following bound on $\rho(\mathbf{H})$ can be obtained:

$$\rho(\mathbf{H}) \leq \Upsilon\,, \quad \Upsilon := \frac{6\sum_q \frac{1}{q}|\beta\alpha_q|}{\left|1 - 6\beta\sum_q \frac{1}{q}\alpha_q\right|}\,. \quad (20)$$

Thus, we can neglect the residual calculation and fix the number of iterations to $M = \lceil -\log_{10}(E)/\eta \rceil$ or $M = \lfloor -\log_{10}(E)/\eta \rfloor$, where $\eta := -\log_{10}(\Upsilon)$. The parameter η represents, approximately, the number of additional digits of relative accuracy obtained with each iteration.

6. ISOTROPIC AND FOURTH-ORDER SCHEMES

To reduce the space of free parameters let us introduce some additional constraints. In the interest of isotropic error we can impose the following constraint

$$\alpha_2 = 4/3 - 2\alpha_1\,, \quad (21)$$

with which we get

$$\delta_{\Delta} u = \Delta u + \frac{h^2}{12}\Delta^2 u + O(h^4)\,. \quad (22)$$

The error will be isotropic (direction-independent) up to the $O(h^4)$ term since the (isotropic) biharmonic operator Δ^2 appears in the

$O(h^2)$ term. Through the use of modified equation methods [14], it is straightforward to arrive at the condition

$$\lambda^2 = 1 - 12\beta\,, \tag{23}$$

to have a fourth-order local truncation error for the IVP

$$\delta_\Box u = \Box u + O(h^4)\,. \tag{24}$$

Under the isotropy constraint (21) the stability condition (12) reduces to

$$\lambda_{\max,\beta,\alpha_1} = \begin{cases} \sqrt{3/4 - 4\beta} & 1/12 \le \alpha_1 \le 5/12 \\ \sqrt{3/(12\alpha_1 - 1) - 4\beta} & 5/12 < \alpha_1 \end{cases}, \tag{25}$$

and the constraints (11) and (19) reduce the parameter space of stable fourth-order schemes to the following

$$1/12 \le \alpha_1 \le 5/12\,, \tag{26}$$

with $\lambda = \sqrt{5/8} \approx 0.79$ and $\beta = 1/32$. Finally, we can optimise α_1 with respect to η. Using (20) it can be shown that

$$\eta \in [\log_{10}(19/3), \log_{10}(7)] \approx [0.802, 0.845]\,,$$

for the region defined in (26). The optimal value, $\eta = 0.845$, is given by $\alpha_1 = 1/3$, which corresponds to a scheme with a 19-point stencil ($\alpha_3 = 0$).

7. NUMERICAL DISPERSION

At this point, we can analyse the numerical dispersion of the schemes that are suitable candidates for the Jacobi iterative solve. To further reduce the space of free parameters, we will restrict our attention to two cases: $\alpha_1 = 1/3$ and $\alpha_1 = 5/12$. The former leads to an isotropic 19-point stencil, and the latter is an isotropic 27-point stencil. The resulting finite difference schemes are implicit generalisations of the "IISO1" and "IISO2" (interpolated isotropic) explicit schemes [15, 7].

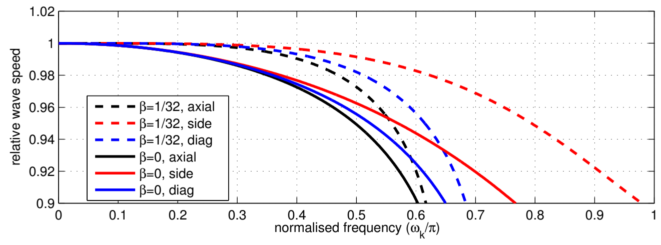
In order to analyse dispersion it helps to define a normalised spatial frequency $\boldsymbol{\xi}_h := \boldsymbol{\xi} h$ and a normalised temporal frequency $\omega_k := \omega k$. We can then write $\Lambda(\boldsymbol{\xi}_h)$ as

$$\Lambda(\boldsymbol{\xi}_h) = \sum_q \frac{3\alpha_q}{|\Omega_q| d} \sum_{\boldsymbol{v}\in\Omega_q} \sin^2(\boldsymbol{\xi}_h \cdot \boldsymbol{v}/2)\,, \tag{27}$$

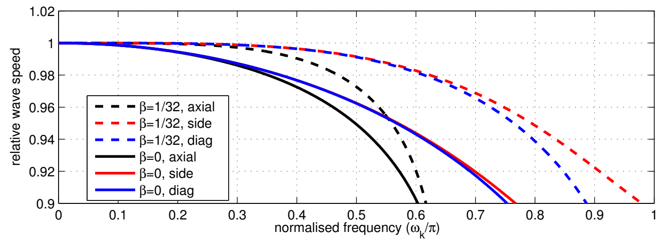
and the relative numerical wave speed (ideally unity), also known as the *dispersion coefficient*, is defined as

$$\nu(\boldsymbol{\xi}_h) := \frac{\omega_k(\boldsymbol{\xi}_h)}{\lambda|\boldsymbol{\xi}_h|}\,, \quad \omega_k(\boldsymbol{\xi}_h) := 2\arcsin\left(\lambda(\Lambda^{-1} - 4\beta)^{-\frac{1}{2}}\right)\,, \tag{28}$$

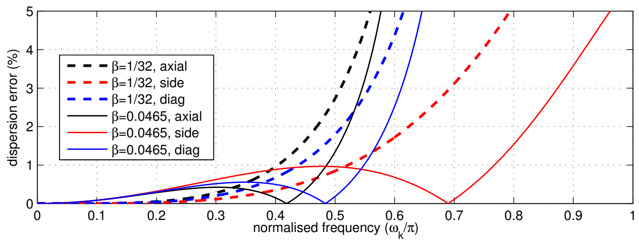
for $\omega_k \in (0, \pi]$ and $\boldsymbol{\xi}_h \in \mathbb{B}$, where $\mathbb{B}$ is the *wavenumber cell* of the grid, which is a cube centered at zero with sides of length 2π. Furthermore, by inverting the dispersion relation (in the region where it is surjective) we can plot the numerical wave speed as a function of spherical coordinates, where the radial coordinate represents the temporal frequency ω_k and where the two polar angles represent a plane-wave direction of propagation in $\mathbb{R}^3$ [15]. The wave speed errors can be seen in Fig. 2a for the schemes $\alpha_1 = 1/3$ and $\beta \in \{0, 1/32\}$ along the axial (center to face-center of $\mathbb{B}$), side diagonal (center to edge-center), and diagonal (center to vertex) directions; these are the directions in which the extreme cases are generally found [7]. Fig. 2b shows the dispersion coefficient along the same directions for the scheme with $\alpha_1 = 5/12$ and $\beta \in \{0, 1/32\}$. It can be seen from these figures that the fourth-order implicit schemes give improvements over their second-order explicit counterparts in each direction.



(a) Relative wave speed for $\alpha_1 = 1/3$ and $\beta \in \{0, 1/32\}$



(b) Relative wave speed for $\alpha_1 = 5/12$ and $\beta \in \{0, 1/32\}$



(c) Dispersion error for $\alpha_1 = 1/3$ and $\beta \in \{1/32, 0.0465\}$

Figure 2: Numerical dispersion for various schemes

The fourth-order condition (23) can also be ignored in order to find a scheme optimised for some fixed amount of dispersion error that can be tolerated, where *dispersion error* is defined as $|1-\nu| \times 100\%$. For example, the parameter $\beta = 0.0465$ is a good choice for a 1% dispersion error tolerance. The dispersion errors for $\alpha_1 = 1/3$ and $\beta \in \{1/32, 0.0465\}$ are shown in Fig. 2c. More optimised parameters will be given shortly. Note, the relative wave speeds are plotted only up to a 5% or 10% dispersion error for the purposes of showing detail. The minimum directional cutoff frequencies, above which the modal density will be incorrect, are not seen in the figures, but they are listed in Table 1 ($\omega_{k,\mathrm{cutoff}}$). The cutoff frequencies for the implicit schemes are near to $\omega_{k,\mathrm{cutoff}}$ for the IISO1 (or IISO2) explicit scheme, which is $(2/3)\pi$ rad/sample [7].

7.1. Relative computational efficiency

One can achieve any level of accuracy in the dispersion error up to any desired temporal frequency (in Hz) with any (convergent) scheme simply by reducing the spatial-step (for a fixed Courant number), due to consistency with the model equation. Of course, oversampling the grid incurs *cubic* increases in memory usage and *quartic* increases in the operation count, so this quickly becomes an impossible strategy for simulating large spaces. Nevertheless,

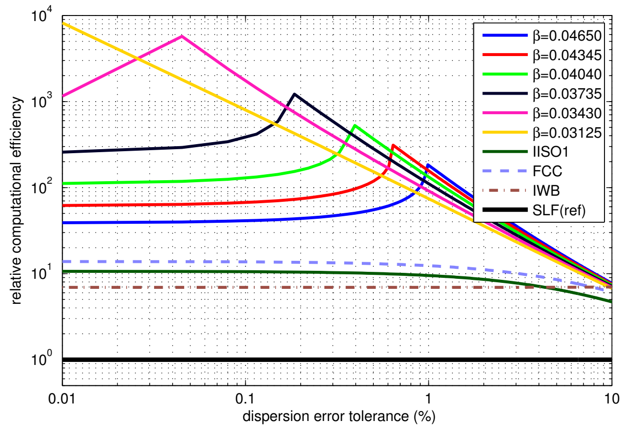


Figure 3: Relative computational efficiencies for various implicit schemes with $\alpha \in \{1/3, 5/12\}$ and explicit schemes, with the simplest scheme (SLF) as reference. Table 1 lists relative comparisons where IISO1 and FCC explicit schemes are reference schemes, also taking into account Jacobi iterations.

this strategy exists, so we must take into account some measures of computational costs in order to determine whether these implicit schemes are more or less effective than their simpler explicit counterparts with oversampled grids. First we will consider the spatiotemporal density of points required to achieve a certain dispersion error globally, and then we will include additional costs from the iterative solve.

As in [20, 7, 21], we start by investigating the relative computational efficiency (RCE), which is defined as the spatiotemporal density of points ($\mathbb{T} \times \mathbb{G}$) necessary to keep the dispersion error below some tolerance level, relative to that required by some reference scheme [20]. Thus, the RCE of a scheme, with some chosen reference scheme, is dimensionless and is a function of the dispersion error tolerance. As in [15, 7] we use the simplest explicit scheme [4], also known as the "standard leapfrog" (SLF), as a reference. The RCEs for the cases $\alpha_1 \in \{1/3, 5/12\}$ with various choices of β are shown in Fig. 3 for a 0.01-10% dispersion error tolerance. Along the axial directions, the schemes with $\alpha_1 \in \{1/3, 5/12\}$ have the same dispersion (worst-case), so the RCEs for the implicit schemes with $\alpha_1 = 5/12$ are the same as those with $\alpha_1 = 1/3$. The explicit IISO2 ($\alpha_1 = 5/12, \beta = 0$) scheme is also equivalent to IISO1 ($\alpha_1 = 1/3, \beta = 0$) in terms of its RCE.

Also included in Fig. 3 are the 13-point face-centered cubic (FCC) explicit scheme ($\alpha_1 = 0, \alpha_2 = 1$) (on its native grid [21]) and the 27-point "interpolated wideband" (IWB) explicit scheme ($\alpha_1 = 1/4, \alpha_2 = 1/2$), for comparison with existing literature [7, 21].[1] As can be seen in Fig. 3, the implicit schemes have higher RCEs than their explicit counterparts and, in particular, the fourth-order scheme ($\beta = 0.03125$) becomes exponentially (linear on a log scale) more efficient, relative to the second-order schemes, as the dispersion error tolerance approaches zero.

Now taking into account the additional iterations that are necessary for the Jacobi solve, the implicit schemes should be advantageous if the RCE for some desired dispersion error tolerance is greater than $\lceil -\log_{10}(E)/\eta \rceil + 2$ (the residual check is neglected). Table 1 lists the dispersion error tolerances below which the implicit schemes with $\alpha = 1/3$ are more efficient than the FCC and IISO1 (or IISO2) explicit schemes, in terms of point-wise updates required for the iterative solve to converge in single and double precision. As can be seen in the table, one can choose β to give an implicit scheme that is more efficient than the FCC explicit scheme for any dispersion error tolerance that is less than 1.1%. In double precision, the implicit schemes become more favourable when the dispersion error tolerance is less than 0.56%.

Table 1: Dispersion error tolerance levels where implicit schemes are more computationally efficient than the FCC and IISO1 (or IISO2) explicit schemes, taking into account Jacobi solve with $M = \lceil -\log_{10}(E)/\eta \rceil$. Also shown are the minimum directional cutoff frequencies, $\omega_{k,\text{cutoff}}$ in rad/sample.

$\alpha_1 \in \{1/3, 5/12\}$			more eff. than FCC		more eff. than IISO1	
β	η	$\omega_{k,\text{cutoff}}$	$E = \varepsilon_\text{s}$	$E = \varepsilon_\text{d}$	$E = \varepsilon_\text{s}$	$E = \varepsilon_\text{d}$
0.04650	0.641	0.626π	<1.1%	–	<1.3%	–
0.04345	0.677	0.629π	<0.98%	–	<1.2%	<0.73%
0.04040	0.715	0.632π	<0.92%	<0.56%	<1.1%	<0.67%
0.03735	0.755	0.635π	<0.79%	<0.48%	<0.98%	<0.58%
0.03430	0.799	0.638π	<0.70%	<0.37%	<0.88%	<0.48%
0.03125	0.845	0.641π	<0.53%	<0.28%	<0.70%	<0.36%

Using the same techniques, we could compare the schemes in terms of the spatial grid densities, leading to memory costs required for some dispersion error tolerance level. This relative comparison is similar to what appears in Fig. 3, but the vertical axis would represent the relative efficiency in terms of spatial grid density, and it would be scaled by a factor of 3/4 (on a log scale) to reflect the cubic increase in grid density versus the quartic increase in operations with oversampling of the grid. As such, we simply summarise the main result. In terms of the extra memory storage required for the Jacobi solve (two extra states), the implicit schemes become more efficient than the explicit FCC and IISO1 schemes when the dispersion error tolerance is <3.8% (vs. FCC) or <5.3% (vs. IISO1).

8. NUMERICAL EXPERIMENTS

8.1. Modal frequencies of cubic domain

The known analytical modes of a cubic-shaped room with lossless boundaries provide a simple validation test that can also illustrate some advantages of the implicit schemes. To this end, the low-frequency response of a cubic domain with $\gamma = 0$ and with dimensions (11 m)^3 was simulated using the scheme with $\alpha_1 = 1/3$ in explicit ($\beta = 0$) and implicit forms ($\beta = 1/32$). The first two time-steps, $u(0, \boldsymbol{x})$ and $u(k, \boldsymbol{x})$, were set to a spatial Gaussian with mean (1 m, 2 m, 3 m) (the domain is centered about the origin) and variance 1 m^2. The Courant number was set to λ_max respectively for both schemes and $c = 340$ m/s. To normalise for computational costs (total number of operations), the implicit scheme with $M = 5$ iterations used a coarse grid of size 12x12x12, whereas the explicit scheme used a finer grid of size 21x21x21. The outputs were read at the grid points (4,6,3) and (8,12,6) for the implicit and explicit schemes respectively (counting from one). Spectra of the outputs from these simulations are shown in Fig. 4. As can be seen, the implicit scheme results in a better agreement with the analytical modal frequencies, despite having a coarser spatial grid.

[1] It is worth pointing out that the implicit generalisations of the FCC and IWB explicit schemes were investigated, but they did not offer significant improvements over the explicit cases. This can be traced to the lack of an isotropic error term in their discrete Laplacians.

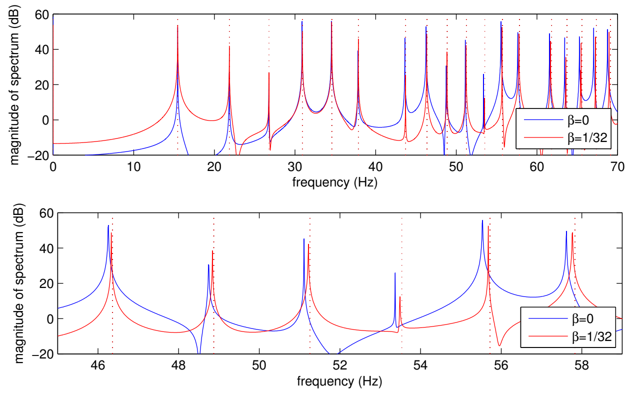


Figure 4: Comparison of low frequency responses for cubic room using IISO1 explicit scheme ($\alpha_1 = 1/3, \beta = 0$) with grid size 21x21x21, and fourth-order implicit scheme ($\alpha_1 = 1/3, \beta = 1/32$) with grid size 12x12x12 and $M = 5$. Dotted lines denote theoretical modal frequencies.

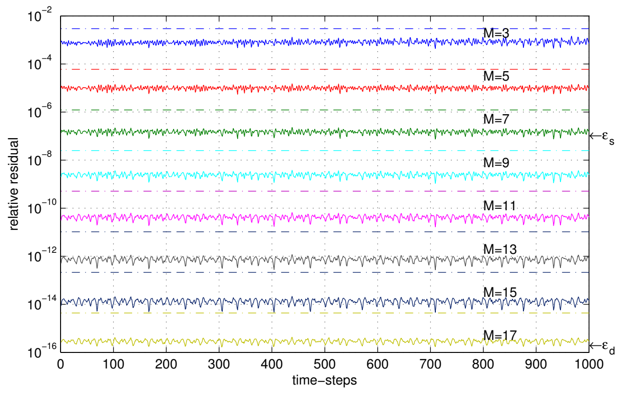


Figure 5: Relative residual from implicit scheme ($\alpha_1 = 1/3, \beta = 1/32$) after M iterations for simulation of cubic domain. Dashed lines denote expected residual, $10^{-0.845M}$. Jagged lines are measured relative residuals. Machine epsilon for single and double precision are marked by arrows.

8.2. Relative residual with fixed number of iterations

It is also worth investigating the relative residual over time using a fixed number of Jacobi iterations. Using the same test case, the relative residuals obtained from conducting simulations with various choices of M are plotted in Fig. 5. As can be seen, the relative residuals (jagged lines) remain smaller in magnitude than the expected residuals with magnitude $10^{-0.845M}$ (dashed lines). In this test case, the limits of single and double precision are effectively reached with seven and 17 iterations respectively.

8.3. Stability in finite precision arithmetic

The stability conditions derived in Section 4 may not be sufficient in practical situations due to unavoidable finite precision effects (round-off error). Single precision may be preferred to double precision since GPU cards tend to have a better peak performance for single precision arithmetic than double, and single precision variables use half of the memory space on the GPU card. However, round-off error in single precision has been known to cause late-time instabilities (after $\mathcal{O}(10^4)$ time-steps) with even the simplest of explicit schemes (SLF) [22], while such instabilities are rarely seen in double precision. A typical room impulse response at 44.1 kHz will require $\mathcal{O}(10^5)$ time-steps to be calculated, so it is important to ensure the long-time stability of these schemes in single precision. These round-off effects have been analysed using the spectral properties of the one-step recursion (state space) matrix in the explicit 27-point schemes [23]. Here, we consider the usual two-step recursion, which does not necessarily encapsulate all round-off errors, but focuses on the spectrum of the Laplacian matrix.

As described in Section 4.2, the two conditions: $\rho(\mathbf{L}) \leq 4\Lambda_{\max}$ and $\mathbf{L} \leq 0$, along with (12), lead to stability of the explicit/implicit schemes. In practice, it is possible that these conditions will not hold in the presence of round-off errors. However, measures can be taken to protect against any consequent instabilities (exponential growth). Linear growth is possible at the stability limit, but such growth is undesirable for room impulse responses. Setting the Courant number slightly below its limit: $\lambda = (1-\mu)\lambda_{\max}$, for $0 < \mu \lll 1$ prevents such growth (μ = 1e-4 is a good choice), as well as any exponential growth near the Nyquist caused by round-off errors [23]. To buffer against a violation of the second condition, we can replace $\mathbf{L}$ with $\mathbf{L} - \sigma\mathbf{I}$, for $0 < \sigma \lll 1$, since it follows from Gerschgorin's theorem that $\mathbf{L} - \sigma\mathbf{I} \leq 0$ for σ sufficiently large. This also causes a shift of modal frequencies, but the effect is negligible for $\sigma \ll \omega/c$, and σ should be on the order of 10^{-7}.

To test these counter-measures, the IISO1 explicit scheme ($\alpha_1 = 1/3$) and its fourth-order implicit counterpart ($\beta = 1/32$, $M = 8$) were excited with a Kronecker delta (in space and time) on a grid of size 26x10x10, and run for 10^6 time-steps. The excitation was also DC-filtered [24] to eliminate any unwanted, yet valid, linear drift in the solution.

In Fig. 6a, an exponential drift (DC instability [23]) can be seen; this is caused by round-off error in single precision and is to be corrected through the use of the σ parameter. Fig. 6b shows the effect using a small σ, approximately $2\varepsilon_s$, to correct such an instability (note the scaling on the horizontal axes in Figs. 6a and 6b). The use of $\sigma > 0$ is not necessary in double precision (at least for $\mathcal{O}(10^6)$ time-steps), as seen in Fig. 6c with $\sigma = 0$. Figs. 6d-6f show the fourth-order implicit counterparts using eight iterations. In double precision the implicit scheme is stable for 10^6 time-steps with $M = 8$ and $\sigma = 0$ (Fig. 6f). Lossy boundaries (γ = 1e-5) are employed in Figs. 6g-6h, which results in a decay in the responses.

It is important to point out a low-frequency amplitude modulation in Figs. 6b, 6e, and 6g. This is due to the DC mode ($\omega = 0$) being shifted by the effect of σ non-zero. A similar effect arises when a so-called "hard source" is used as an excitation [25]. Here, the oscillation has a normalised frequency of approximately $\sqrt{\sigma}\pi$ rad/sample. The value of σ that will be required in single precision should scale with the duration of the simulation. Thus, for a typical room impulse response, σ should scale with the sample rate, and this low-frequency oscillation should remain inaudible. If desired, the artefact can be removed by applying another DC blocking filter [24] to the output, as seen in Fig. 6h.

9. SIMULATIONS ON GPU

In this section, we present timing results from a basic CUDA implementation of the implicit schemes on a single Nvidia Tesla K20 GPU card. The goal here is not to present speed-ups over single-thread CPU codes, since significant speed-ups have been reported for 27-point explicit schemes in various studies [12, 13, 26, 22]. The interest here is simply to compare GPU implementations of the

Table 2: Timing results from computation of 2000 time-steps on a Tesla K20 GPU card for explicit schemes and implicit schemes using M Jacobi iterations and the compute time increase (CTI) for each implicit scheme over its respective explicit counterpart. The CTIs are expected to be $(M+2)$ due to the additional SpMVs required by the implicit schemes.

			explicit	implicit, $M=8$		implicit, $M=12$	
δ_Δ	(N_x, N_y, N_z)	precision	time (s)	time (s)	CTI	time (s)	CTI
19-point	(640,480,480)	single	50	453	9.06	635	12.7
19-point	(960,640,480)	single	100	894	8.94	1254	12.5
27-point	(640,480,480)	single	75	608	8.11	846	11.3
27-point	(960,640,480)	single	148	1201	8.11	1676	11.3
19-point	(640,480,480)	double	79	801	10.1	1112	14.1
27-point	(640,480,480)	double	89	910	10.2	1242	14.0

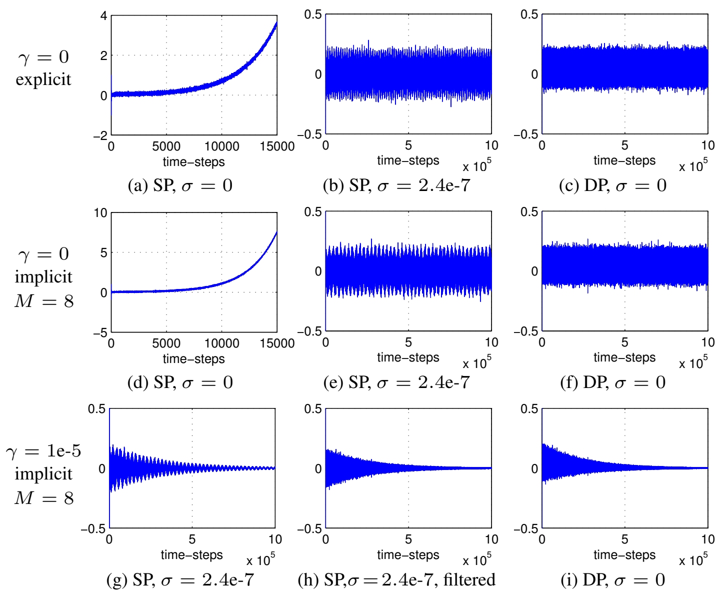


Figure 6: Responses from cubic box obtained using explicit/implicit schemes with $\alpha_1 = 1/3$ in single precision (SP) and double precision (DP), with $\lambda = 0.9999\lambda_{\max}$ in each case. Implicit schemes use $M = 8$ iterations. Note that σ is not used in double precision. A DC blocking filter was applied to the output in Fig. 6h.

explicit schemes and their implicit counterparts for a fixed number of Jacobi iterations. Specific details on the GPU implementation will be left out for brevity, but the implementation is similar in nature to those found in [26, 22]. However, it is important to note that the memory bandwidth was maximised by making use of the read-only data cache in the Nvidia Kepler GPU architecture.

Table 2 lists the timing results from computing 2000 time-steps for 19-point and 27-point explicit schemes (the choice of $\boldsymbol{\alpha}$ is not important here) and their implicit counterparts (the choice of $\beta > 0$ is not important) with a fixed number of iterations $M \in \{8, 12\}$. Two different grid sizes were used and the simulations were run in both single and double precision. Results for the larger grid size are only given in single precision due to memory limitations on the GPU card (5 GB).

We expect the implicit schemes to take $M + 2$ times as long as their explicit counterparts due to the extra SpMVs (not taking into account the extra memory bandwidth required). As can be seen in Table 2, the implicit schemes are 10-20% faster than expected in single precision. Meanwhile, in double precision they behave approximately as expected. These variations from the $M+2$ increase are due to cache effects and memory bandwidth bottlenecks.

10. CONCLUSIONS AND FUTURE WORK

In this study, we have presented 19- and 27-point fourth-order accurate and optimised implicit finite difference schemes for the 3-D wave equation with frequency-independent lossy boundaries on a box-shaped domain. These schemes can be solved using the Jacobi method with a convergence rate of nearly one digit of relative accuracy per iteration. Numerical dispersion was analysed and it was found that the implicit schemes, taking into account the iterative solve, become more computationally efficient than second-order explicit counterparts for situations where the amount of dispersion error that can be tolerated is less than 1%, and exponentially more efficient as this tolerance level approaches zero. These schemes were shown to be stable in finite precision arithmetic for as many as 10^6 time-steps in double precision, as well as in single precision at the cost of introducing inaudible artefacts. Timing results were presented from CUDA implementations run on an Nvidia Tesla K20 GPU card. It was found that the compute times for the implicit schemes scaled as expected with the additional SpMVs required.

Future work will investigate further generalisations for these implicit schemes. The first is to consider a more general form for the implicit scheme where different sets of $\boldsymbol{\alpha}$ parameters are used for the implicit and explicit discrete Laplacian operators, as in [14], providing more free parameters to optimise in order to further minimise numerical dispersion. Another generalisation is to include viscothermal effects, which are necessary for a more detailed model of sound propagation in air [3]. A third generalisation would be to consider these schemes in an unstructured finite volume framework (allowing for the modelling of irregular geometries) with more general (complex) impedance boundary conditions, as in the explicit case [10].

More advanced iterative techniques that are amenable to parallel implementations (Krylov subspace methods) could also be considered; many of which are known, for certain problems, to converge in fewer iterations than the Jacobi method [17]. However, preliminary tests with the system (15), using the myriad iterative methods provided in MATLAB, indicate that while such advanced techniques can converge in fewer iterations, they do not offer substantial improvements in compute times. Ultimately, this is because they require more computation within each iteration (additional SpMVs and residual checks), not to mention additional storage. Finally, an important area of research will be to determine the minimum number of Jacobi iterations required to simultaneously ensure that the residual is inaudible and that stability is maintained for the duration of a given simulation.

Comparative sound examples for the implicit and explicit counterpart schemes will be available for listening at:
`http://www2.ph.ed.ac.uk/~s1164563/dafx14.`

11. REFERENCES

[1] S. Siltanen, T. Lokki, and L. Savioja, "Rays or waves? understanding the strengths and weaknesses of computational room acoustics modeling techniques," in *Proc. Int. Symposium on Room Acoustics*, 2010.

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith III, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[3] P. M. Morse and K. U. Ingard, *Theoretical acoustics*. Princeton University Press, 1968.

[4] G. E. Forsythe and W. R. Wasow, *Finite-difference methods for partial differential equations*. New York: Wiley, 1960.

[5] D. Botteldooren, "Finite-difference time-domain simulation of low-frequency room acoustic problems," *J. Acoustical Society of America*, vol. 98, pp. 3302–3308, 1995.

[6] L. Savioja, T. J. Rinne, and T. Takala, "Simulation of room acoustics with a 3-D finite difference mesh," in *Proc. Int. Computer Music Conf. (ICMC)*, Danish Institute of Electroacoustic Music, Denmark, 1994, pp. 463–466.

[7] K. Kowalczyk and M. van Walstijn, "Room acoustics simulation using 3-D compact explicit FDTD schemes," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 34–46, 2011.

[8] S. Bilbao, "Wave and scattering methods for the numerical integration of partial differential equations," Ph.D. thesis, Stanford University, 2001.

[9] J. Botts and L. Savioja, "Integrating finite difference schemes for scalar and vector wave equations," in *Proc. IEEE ICASSP*, Vancouver, Canada, 2013, pp. 171–175.

[10] S. Bilbao, "Modeling of complex geometries and boundary conditions in finite difference/finite volume time domain room acoustics simulation," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 21, no. 7, pp. 1524–1533, Jul. 2013.

[11] B. Hamilton, "Finite volume perspectives on finite difference schemes and boundary formulations for wave simulation," in *Proc. Digital Audio Effects (DAFx)*, Erlangen, Germany, Sep. 2014.

[12] L. Savioja, "Real-time 3D finite-difference time-domain simulation of low-and mid-frequency room acoustics," in *Proc. Digital Audio Effects (DAFx)*, vol. 1, Graz, Austria, 2010, p. 75.

[13] C. J. Webb and S. Bilbao, "Computing room acoustics with CUDA - 3D FDTD schemes with boundary losses and viscosity," in *Proc. IEEE ICASSP*, Prague, Czech Republic, 2011, pp. 317–320.

[14] S. Bilbao, "Parameterized families of finite difference schemes for the wave equation," *Numerical Methods for PDEs*, vol. 20, no. 3, pp. 463–480, 2004.

[15] K. Kowalczyk and M. van Walstijn, "A comparison of nonstaggered compact FDTD schemes for the 3D wave equation," in *Proc. IEEE ICASSP*, Dallas, Texas, 2010, pp. 197–200.

[16] A. Gourlay and A. R. Mitchell, "A classification of split difference methods for hyperbolic equations in several space dimensions," *SIAM J. Numerical Analysis*, vol. 6, no. 1, pp. 62–71, 1969.

[17] C. Meyer, *Matrix analysis and applied linear algebra*. Philadelphia, PA: SIAM, 2000, vol. 2.

[18] J. Strikwerda, *Finite difference schemes and partial differential equations*. Philadelphia, PA: SIAM, 2004.

[19] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. Wiley, 2009.

[20] M. van Walstijn and K. Kowalczyk, "On the numerical solution of the 2D wave equation with compact FDTD schemes," in *Proc. Digital Audio Effects (DAFx)*, Espoo, Finland, 2008, pp. 205–212.

[21] B. Hamilton and S. Bilbao, "On finite difference schemes for the 3-D wave equation using non-Cartesian grids," in *Proc. Sound and Music Computing (SMC) Conf.*, Stockholm, Sweden, 2013, pp. 592–599.

[22] C. J. Webb and A. Gray, "Large-scale virtual acoustics simulation at audio rates using three dimensional finite difference time domain and multiple GPUs," in *Proc. Int. Cong. Acoustics (ICA)*, Montréal, Canada, 2013.

[23] J. Botts and L. Savioja, "Spectral and pseudospectral properties of finite difference models used in audio and room acoustics," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 9, pp. 1403–1412, Sept 2014.

[24] J. O. Smith III, *Introduction to digital filters: with audio applications*, 2007, vol. 2.

[25] J. Botts and L. Savioja, "Effects of sources on time-domain finite difference models," *J. Acoustical Society of America*, vol. 136, no. 1, pp. 242–247, 2014.

[26] B. Hamilton and C. J. Webb, "Room acoustics modelling using GPU-accelerated finite difference and finite volume methods on a face-centered cubic grid," in *Proc. Digital Audio Effects (DAFx)*, Maynooth, Ireland, Sep. 2013, pp. 336–343.

12. APPENDIX

12.1. 27-point discrete Laplacian

Using a notation similar to [7] we have:

$$\delta_\Delta^h = \delta_{xx}+\delta_{yy}+\delta_{zz}+a(\delta_{xx}\delta_{yy}+\delta_{xx}\delta_{zz}+\delta_{yy}\delta_{zz})+b(\delta_{xx}\delta_{yy}\delta_{zz})$$

where $a = (\alpha_2 + 2\alpha_3)/4$, $b = \alpha_3/4$, and

$$\delta_{xx}u := u(t, \boldsymbol{x} + \boldsymbol{e}_x h) - 2u(t, \boldsymbol{x}) + u(t, \boldsymbol{x} - \boldsymbol{e}_x h)$$

with the standard unit vector in the x-direction $\boldsymbol{e}_x$. The operators δ_{yy} and δ_{zz} are similarly defined.

12.2. 3-D Laplacian matrix with Neumann conditions

The Laplacian matrix corresponding to the centered Neumann conditions from [7] can be constructed as follows. The 1-D Laplacian matrix with centered Neumann conditions is:

$$\mathbf{D}_N = \begin{bmatrix} -2 & 2 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{bmatrix} .$$

Let $\mathbf{I}_N$ represent the $N \times N$ identity matrix. Consider a 3-D grid with dimensions $N_x \times N_y \times N_z$ and let it be decomposed into a vector, first into z-planes, then y-rows and x-columns. We construct the matrices

$$\begin{aligned} \mathbf{D}_{xx} &:= \mathbf{I}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{D}_{N_x} , \\ \mathbf{D}_{yy} &:= \mathbf{I}_{N_z} \otimes \mathbf{D}_{N_y} \otimes \mathbf{I}_{N_x} , \\ \mathbf{D}_{zz} &:= \mathbf{D}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{I}_{N_x} , \end{aligned}$$

where $\otimes$ denotes the Kronecker product. The Laplacian matrix of interest can then be written as

$$\begin{aligned} \mathbf{L} &= \mathbf{D}_{xx} + \mathbf{D}_{yy} + \mathbf{D}_{zz} \\ &+ a(\mathbf{D}_{xx}\mathbf{D}_{yy} + \mathbf{D}_{xx}\mathbf{D}_{zz} + \mathbf{D}_{yy}\mathbf{D}_{zz}) + b(\mathbf{D}_{xx}\mathbf{D}_{yy}\mathbf{D}_{zz}) . \end{aligned}$$

12.3. Loss matrix

The matrix $\mathbf{Q}$ can be constructed as follows. Let $\mathbf{q}_N$ be the vector: $(1, 0, \dots, 0, 1)^T$ of length N. We construct the matrices

$$\begin{aligned} \mathbf{Q}_x &:= \mathbf{I}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{q}_{N_x} , \\ \mathbf{Q}_y &:= \mathbf{I}_{N_z} \otimes \mathbf{q}_{N_y} \otimes \mathbf{I}_{N_x} , \\ \mathbf{Q}_z &:= \mathbf{q}_{N_z} \otimes \mathbf{I}_{N_y} \otimes \mathbf{I}_{N_x} . \end{aligned}$$

Then we have $\mathbf{Q} = \mathbf{Q}_x + \mathbf{Q}_y + \mathbf{Q}_z$.