ROOM ACOUSTICS MODELLING USING GPU-ACCELERATED FINITE DIFFERENCE AND FINITE VOLUME METHODS ON A FACE-CENTERED CUBIC GRID

Brian Hamilton, [*]
Acoustics and Audio Group
University of Edinburgh
b.hamilton-2@sms.ed.ac.uk

Craig J. Webb,
Acoustics and Audio Group
University of Edinburgh
c.j.webb-2@sms.ed.ac.uk

ABSTRACT

In this paper, a room acoustics simulation using a finite difference approximation on a face-centered cubic (FCC) grid with finite volume impedance boundary conditions is presented. The finite difference scheme is accelerated on an Nvidia Tesla K20 graphics processing unit (GPU) using the CUDA programming language. A performance comparison is made between 27-point finite difference schemes on a cubic grid and the 13-point scheme on the FCC grid. It is shown that the FCC scheme runs faster on the Tesla K20 GPU and has less numerical dispersion than best 27-point schemes on the cubic grid. Implementation details are discussed.

1. INTRODUCTION

Room acoustics simulations are important for architectural acoustics, auralising virtual spaces, and artificial reverberation. Traditionally, these simulations have been carried out using image-source [1] and ray-tracing methods [2], requiring simplifying assumptions about room geometry or wave behaviour. Finite element and boundary element methods [3, 4] have the potential to simulate full wave behaviour for complex room geometries, but these methods are not easily parallelisable and thus are less suited to implementation on graphics processing units (GPUs). Other methods offering full 3-D acoustical simulations such as the finite difference (FD) [5, 6] and finite volume (FV) methods [7] can be computationally heavy, but are well-suited for GPU programming due to their explicit formulations [8, 9]. Computation times can be long for large spaces at audio rates like 44.1 kHz, even with state of the art GPU cards [10], so computational efficiency in the numerical scheme is critical.

Within FD methods there are many choices for FD operators and grids on which to approximate solutions to the 3-D wave equation [11]. The computational efficiencies of FD schemes differ and determining the most suitable candidate for room acoustics simulations has been the subject of many studies [11–14]. The *interpolated wideband scheme* (IWB) scheme, employing a 27-point stencil on the cubic grid, has been shown to be computationally efficient at minimising wave speed error [13] and has been used in GPU-accelerated simulations [8, 15, 16]. Another candidate, and the focus of this paper, is a 13-point scheme which can be used on the cubic grid (the *close-cubic packed* (CCP) scheme) [13] or on the face-centered cubic grid (FCC) [12, 14, 17, 18]. The 13-point FD scheme on the FCC grid has recently been shown to be more computationally efficient than 27-point schemes on the cubic grid [14], but it has yet to be implemented on a GPU for large-scale room acoustics simulations.

Frequency-dependent impedance boundary conditions are necessary for realistic room acoustic simulations and they must be coupled with the FD scheme on the interior such that the entire scheme remains numerically stable. Passive locally-reacting boundary conditions for the IWB scheme have been formulated [13] and implemented in large-scale simulations [8, 15]. Recently, FV methods have been employed to model locally-reacting frequency-dependent impedance boundaries on unstructured grids for complex geometries [19]. FV methods reduce to FD methods on regular grids and the impedance boundary conditions can be applied to certain FD schemes employing nearest neighbouring points [19], such as the 13-point scheme on the FCC grid. It is the purpose of this paper to couple these FV boundary conditions to the 13-point FD scheme on the FCC grid for a large-scale room acoustics simulation using GPU acceleration.

The structure of this paper is as follows: in Section 2, we present the room model and in Section 3, we introduce the FD schemes. In Section 4, we discuss the discretisation of time and space on grids of points and in Section 5, we present a practical way to map the FCC grid to a cubic grid for room acoustics simulations. The FV impedance boundary conditions are presented in Section 6, GPU implementation details are discussed in Section 7 and practical comparisons carried out on the Tesla K20 GPU are presented in Section 8. Conclusions are given in Section 9.

2. ROOM ACOUSTICS MODEL

2.1. 3-D Wave Equation

Let us abbreviate $\partial_w^i = \frac{\partial^i}{\partial w^i}$ for some variable $w \in \mathbb{R}$ and $i \in \mathbb{N}$ (positive integers). We begin with the 3-D wave equation:

$$\left(\partial_t^2 - c^2\Delta\right) u = 0\,, \quad \Delta = \partial_x^2 + \partial_y^2 + \partial_z^2\,, \tag{1}$$

where c is the wave speed, Δ is the 3-D Laplacian, t is time, and $u = u(t, \boldsymbol{x})$ is a solution to be approximated for $\boldsymbol{x} \in \mathbb{R}^3$, $\boldsymbol{x} = (x, y, z)$. In our model, $u(t, \boldsymbol{x})$ represents a *velocity potential* [20] and the pressure p and velocity field $\boldsymbol{\nu}$ are given by:

$$p = \rho\, \partial_t u \tag{2}$$

$$\boldsymbol{\nu} = -\nabla u \tag{3}$$

This part of the model will be approximated with FD methods. To later incorporate finite volume methods at the boundaries we note that we can also split the second-order wave equation into

[*] 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.

two first-order vector equations representing conservation of momentum and mass [20]:

$$\frac{1}{\rho c^2}\partial_t p = -\nabla \cdot \boldsymbol{\nu}\,, \tag{4}$$

$$\rho \partial_t \boldsymbol{\nu} = -\nabla p\,. \tag{5}$$

2.2. Impedance Boundary Conditions

A room is not complete without walls, and modelling of frequency-dependent absorption at the walls is necessary for realistic room acoustics simulations. Over a volume $\mathcal{V}$ with boundary $\partial\mathcal{V}$ we use the impedance boundary condition:

$$\nu_n = Y_0\left(A\partial_t p + Bp + Cg\right)\,, \quad \partial_t g = p\,, \tag{6}$$

where $Y_0 = 1/\rho c$ is the characteristic admittance of air, ν_n is the normal velocity component at the boundary, and $A = A(\boldsymbol{x}), B = B(\boldsymbol{x}), C = C(\boldsymbol{x})$ are parallel stiffness, inertance, and resistance parameters respectively [19]. The variable $g = g(t,\boldsymbol{x})$ is part of the stored energy at the boundary [19]. This part of the model will be approximated with finite volume methods.

3. FINITE DIFFERENCE METHOD

3.1. Difference Operators

Let $\hat{u} = \hat{u}(t,\boldsymbol{x})$ denote the approximation to $u(t,\boldsymbol{x})$. We introduce the time and space first-order difference operators:

$$\delta_{t\pm}\hat{u} = \frac{\pm 1}{k}\left(\hat{u}(t\pm k,\boldsymbol{x}) - \hat{u}(t,\boldsymbol{x})\right)\,, \tag{7}$$

$$\delta_{\boldsymbol{v}\pm}\hat{u} = \frac{\pm 1}{h}\left(\hat{u}(t,\boldsymbol{x}\pm \boldsymbol{v}h) - \hat{u}(t,\boldsymbol{x})\right)\,, \tag{8}$$

where k is the time-step, usually chosen to be $1/F_s$ where F_s is an audio sampling rate like 44.1 kHz, h is the spatial step, and $\boldsymbol{v} \in \mathbb{R}^3$.

The standard approximation to ∂_t^2 in explicit FD schemes for the wave equation is the FD operator:

$$\delta_{tt}\hat{u} = \delta_{t+}\delta_{t-}\hat{u} = \frac{1}{k^2}\left(\hat{u}(t+k,\boldsymbol{x}) - 2\hat{u}(t,\boldsymbol{x}) + \hat{u}(t-k,\boldsymbol{x})\right). \tag{9}$$

Let $\Omega \subset \mathbb{R}^3$ be a finite set of equal-norm vectors and let $|\Omega|$ denote its cardinality. We can build approximations to the Laplacian using the following FD operator:

$$\delta_{\Delta,\Omega}\hat{u} = \kappa\sum_{i=1}^{|\Omega|}\delta_{\boldsymbol{v}_i+}\delta_{\boldsymbol{v}_i-}\hat{u}\,, \tag{10}$$

where $\boldsymbol{v}_i \in \Omega$, and κ will be chosen according to consistency conditions. We call this a $(2|\Omega|+1)$-point *discrete Laplacian* or *stencil*. The 13-point stencil uses the six vectors:
$\Omega_F = \{\hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_z, \hat{\boldsymbol{e}}_x \pm \hat{\boldsymbol{e}}_z\}/\sqrt{2}$ and $\kappa_F = 1/2$. A 27-point stencil can be built using the three discrete Laplacians:

$$\delta_{\boldsymbol{\Delta},\boldsymbol{\alpha},\boldsymbol{\Omega}}\hat{u} = \left(\alpha_1\delta_{\Delta,\Omega_1} + \alpha_2\delta_{\Delta,\Omega_2} + \alpha_3\delta_{\Delta,\Omega_3}\right)\hat{u}\,, \tag{11}$$

where Ω_1 consists of the standard unit vectors $\{\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y, \hat{\boldsymbol{e}}_z\}$ with $\kappa_1 = 1$, $\Omega_2 = \sqrt{2}\Omega_F$ with $\kappa_2 = 1/4$, and where Ω_3 consists of the four vectors $\{\hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_x \pm \hat{\boldsymbol{e}}_z, \hat{\boldsymbol{e}}_y \mp \hat{\boldsymbol{e}}_x \pm \hat{\boldsymbol{e}}_z\}$ with $\kappa_3 = 1/4$. We require $\sum \alpha_i = 1$ for consistency. Note that the 13-point stencil is a special case of the 27-point stencil with $\boldsymbol{\alpha} = (0,1,0)$ and a scaled spatial step $h' = \sqrt{2}h$. These stencils are featured in Fig. 1.

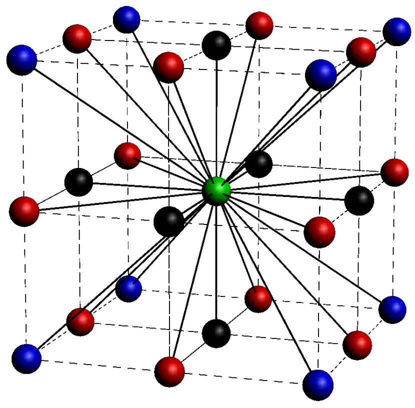


Figure 1: *Vectors (points) used in 27-point stencil $\delta_{\boldsymbol{\Delta},\boldsymbol{\alpha},\boldsymbol{\Omega}}$. Points employed in δ_{Δ,Ω_1}, δ_{Δ,Ω_2}, δ_{Δ,Ω_3} are coloured black, red, and blue respectively. The center point (green) is shared among the three stencils. The 13-point stencil δ_{Δ,Ω_F} is simply a scaled version of δ_{Δ,Ω_2}.*

3.2. Finite Difference Scheme for 3-D Wave Equation

Combining these operators we get a FD scheme for the 3-D wave equation:

$$(\delta_{tt} - c^2\delta_{\boldsymbol{\Delta},\boldsymbol{\alpha},\boldsymbol{\Omega}})\hat{u} = 0\,, \tag{12}$$

using the 27-point stencil, or the 13-point stencil as the special case mentioned previously. The approximated solution $\hat{u}(t,\boldsymbol{x})$ can be updated in time with the explicit recursion:

$$\hat{u}(t+k,\boldsymbol{x}) = (c^2k^2\delta_{\boldsymbol{\Delta},\boldsymbol{\alpha},\boldsymbol{\Omega}} + 2)\hat{u}(t,\boldsymbol{x}) - \hat{u}(t-k,\boldsymbol{x})\,, \tag{13}$$

given some initial conditions.

4. DISCRETISING TIME AND SPACE

The FD scheme will have a certain *cutoff frequency* in time and space, due to the FD operators, which limits the temporal and spatial bandwidth of the approximated solution $\hat{u}(t,\boldsymbol{x})$. Thus, we can discretise time and space and reconstruct $\hat{u}(t,\boldsymbol{x})$ from a set of values $\{\hat{u}(t,\boldsymbol{x}) : t \in \boldsymbol{T}_k, \boldsymbol{x} \in \boldsymbol{G}_h\}$ on temporal and spatial grids $\boldsymbol{T}_k$ and $\boldsymbol{G}_h$. The minimum number of points required can be determined using multidimensional sampling theory [21]. For time we use the integer lattice (grid) $\mathbb{Z}$ scaled by the time-step k. For space, we can use the cubic grid for the 27-point stencil, and we have the choice between the cubic and FCC grids for the 13-point stencil.

The FCC grid is shown in Fig. 2 in two unit cell orientations. Either unit cell can be used to tile space and build the FCC grid. Notice that the union of the cells in Figs. 2a and 2b makes the cubic grid in Fig. 2c. It is then not hard to see that the 13-point scheme ($\boldsymbol{\alpha} = (0,1,0)$) on the cubic grid operates on two disjoint sets of points, decoupling into two subschemes on FCC subgrids.[1] As such, to completely reconstruct $\hat{u}(t,\boldsymbol{x})$ in space at any given time t we only need $\{\hat{u}(t,\boldsymbol{x}) : \boldsymbol{x} \in \boldsymbol{G}_h\}$ where $\boldsymbol{G}_h$ is one FCC subgrid, since the other subscheme carries no extra information (bandwidth). This also means that the computational efficiency of the 13-point scheme, reported in [13] on the cubic grid, can be

[1]The same observation has been made in the context of lattice Boltzmann simulations [22].

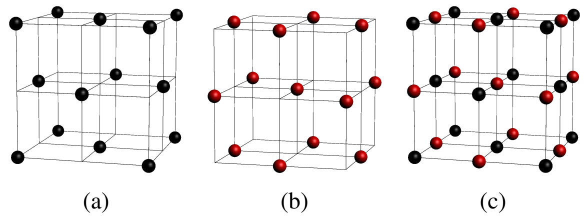


Figure 2: *Two orientations of a unit cell in FCC grid. (a) face-centered: vertices and centers of cube faces. (b) edge-centered: center of cube and centers of edges. (c) Cubic grid with checkerboard colouring.*

increased by a factor of two when employed on the FCC grid.[2] Considering this feature, the 13-point scheme becomes more efficient than 27-point schemes on the cubic grid when less than 8% wave speed error is deemed critical, according to the data reported in [13] (reproduced in Table 2). When simulating large acoustical spaces it is important to keep this error low so that artifacts (smearing of transients) remain inaudible, since they will accumulate over time and space. See [14] for further analysis on the numerical dispersion of these schemes.

If we fix the time-step k, there is a lower bound on how finely we can discretise space for a given FD scheme such that numerical stability is ensured [24]. The condition on the spatial step for numerical stability is $h \geq ck/\lambda_{\max,\boldsymbol{\alpha}}$ where $\lambda_{\max,\boldsymbol{\alpha}}$ is the maximum Courant number of the scheme. While computational costs will increase with smaller h, simulated bandwidth is highest when h is kept to the minimum, as well as computational efficiency in terms of minimising numerical dispersion [11,13,14]. The need to satisfy both of these constraints is something particular to audio.

For the 27-point scheme, $\lambda_{\max,\boldsymbol{\alpha}}$ is [11]:

$$\lambda_{\max,\boldsymbol{\alpha}} = \min\left(1, \frac{1}{\sqrt{2\alpha_1+\alpha_2}}, \frac{1}{\sqrt{2\alpha_1-\alpha_2+1}}\right), \quad (14)$$

as well as extra conditions on $\boldsymbol{\alpha}$ [11]. Considering the scaled spatial step h', the 13-point scheme on the FCC grid has $\lambda_{\max} = ck/h' = \sqrt{1/2}$. Two other 27-point schemes of interest are the IWB scheme, with $\boldsymbol{\alpha} = (1/4, 1/2, 1/4)$, $\lambda = 1$ and an interpolated isotropic scheme (IISO2) with $\boldsymbol{\alpha} = (5/12, 1/2, 1/12)$, $\lambda = \sqrt{3/4}$ [13]. We will also compare these schemes in terms of computation times to the "standard leapfrog" (SLF) 7-point scheme with $\boldsymbol{\alpha} = (1,0,0)$, $\lambda = \sqrt{1/3}$ [25].

We note that the optimal Courant number in the 13-point FD scheme on the FCC grid is greater than the Courant number employed in its *digital waveguide mesh* implementation ($\lambda = \sqrt{1/3}$) [12, 17, 18], which means that numerical dispersion will be better with a FD implementation.

5. MAPPING TO A CUBIC GRID

The points on the FCC grid and vectors of the 13-point stencil must be mapped to $\mathbb{Z}^3$ so that they can further be mapped to computer memory. The simplest method is to employ alternating points on the cubic grid in a checkerboard fashion with one stencil orientation, as in Fig. 2c, but this is wasteful in memory use. Another way to build the grid is:

$$\boldsymbol{G}_h = \{\boldsymbol{x}_{\boldsymbol{m},h} = \boldsymbol{V}\boldsymbol{m}h \in \mathbb{R}^3 : \boldsymbol{m} \in \mathbb{Z}^3\}, \quad (15)$$

where $\boldsymbol{V}$ is a *generator matrix* [26] made up of any three vectors chosen from Ω_F. In this way, the grid can be indexed using $\boldsymbol{m}$ and the stencil vectors remapped to $\mathbb{Z}^3$ can be found with $\boldsymbol{V}^{-1}$. However, generator matrices for the FCC grid [26] tend to generate points filling a parallelpiped volume of space [27]. This is impractical for room acoustics because memory will be wasted when modelling a box-shaped room.

Another construction uses four cubic grids shifted by the vectors $\{\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y, \hat{\boldsymbol{e}}_z, \hat{\boldsymbol{e}}_x+\hat{\boldsymbol{e}}_y+\hat{\boldsymbol{e}}_z\}$, as shown in Fig. 3. This better fits a rectangular volume and provides a straightforward way to update the scheme since each subgrid has one set of stencil vectors with respect to the other subgrids. The memory space could be partitioned into four using this construction. In this paper, we construct the FCC grid in a slightly different manner, similar to that in [22]. We use the following construction:

$$\boldsymbol{G}_h = \left\{\boldsymbol{x}_{\boldsymbol{m},h} = (\boldsymbol{m} + m_y\hat{\boldsymbol{e}}_y + ((m_x+m_z) \bmod 2)\hat{\boldsymbol{e}}_y)\, h/\sqrt{2}\right\}, \quad (16)$$

where $\boldsymbol{m} \in \mathbb{Z}^3$. In other words, we take a cubic grid and scale one dimension by two, then we shift in the same direction by the modulo-two sum of the other two integer coordinates. Finally, we scale the grid by $h/\sqrt{2}$ so that the grid spacing is h. Referring to Fig. 3, this results in the black and green points having one stencil orientation, and the blue and red another. In this way, we do not need to partition the memory space and we have only two orientations of stencils. However, some extra care must be taken at walls and edges of a rectangular domain, and one half face of points is discarded at one wall.

With this coordinate system for the FCC grid, we have the following update recursion ($\lambda = ck/h = \sqrt{1/2}$):

$$\hat{u}(t+k, \boldsymbol{x}) = -\hat{u}(t, \boldsymbol{x}) - \hat{u}(t-k, \boldsymbol{x}) + \frac{1}{4}\sum_{i=1}^{12}\hat{u}(t, \boldsymbol{x}+\boldsymbol{v}'_i), \quad (17)$$

where $\boldsymbol{v}'_i = \boldsymbol{x}_{\boldsymbol{m}'_i,h}$ denotes the nearest neighbouring points on $\boldsymbol{G}_h$ and the set $\{\boldsymbol{m}'_i : i = 1, \ldots 12\}$ denotes the neighbouring points

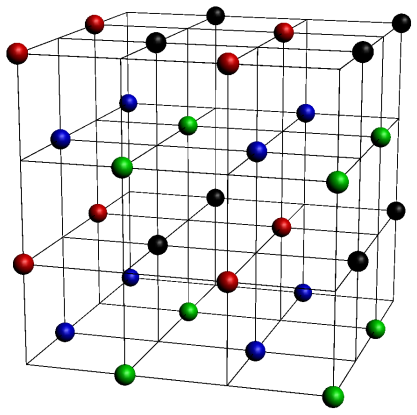


Figure 3: *FCC grid constructed using four cubic grids*

[2]The same reasoning can be applied to the 9-point scheme $\boldsymbol{\alpha} = (0,0,1)$, which decouples into four subschemes on scaled body-centered cubic (BCC) grids. While the computational efficiency reported in [13] can be improved by a factor of four, the 9-point scheme on the BCC grid is still not a good candidate for our problem, even if arguments can be made about the BCC grid being optimal for sampling of space [23]. This is further discussed in [14].

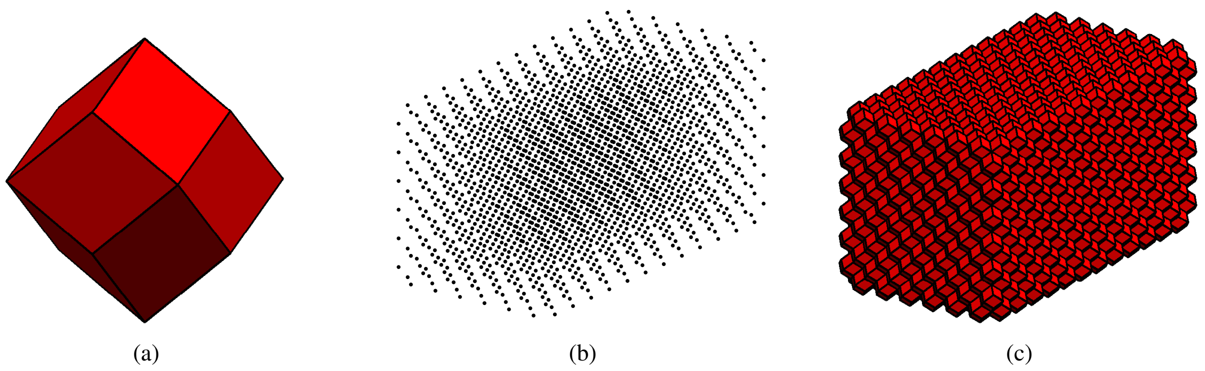


Figure 4: *(a) Voronoi cell of FCC grid: rhombic dodecahedron. (b) FCC grid of points. (c) FCC grid of cells.*

indexed by the $\mathbb{Z}^3$ grid, which depends on whether $\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{m},h}$ has $(m_x + m_z)$ odd or even. For $i \in \{1, \dots, 8\}$ we have $\boldsymbol{m}'_i \in \{\pm\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_z \pm \hat{\boldsymbol{e}}_x, -\hat{\boldsymbol{e}}_z \pm \hat{\boldsymbol{e}}_x, \pm\hat{\boldsymbol{e}}_z\}$ in common between the two orientations. For $i \in \{9, \dots, 12\}$ we have $\boldsymbol{m}'_i \in \{\hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_z\}$ for $(m_x + m_z)$ even and $\boldsymbol{m}'_i \in \{-\hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_x, -\hat{\boldsymbol{e}}_y \pm \hat{\boldsymbol{e}}_z\}$ for $(m_x + m_z)$ odd.

6. FINITE VOLUME IMPEDANCE BOUNDARIES

At the boundaries, we consider the FD scheme to be a special case of a finite volume (FV) scheme on a regular grid of cells encompassing a volume $\mathcal{V}$ with boundary $\partial\mathcal{V}$ [19]. The grid of cells is the Voronoi tessellation of the spatial grid of points (lattice). One of the key differences with this approach is that there are no "ghost points" [13] to consider, as the boundary is composed of the faces of the boundary grid cells. The FCC grid has a Voronoi cell which is the rhombic dodecahedron, as shown in Fig. 4a. This shape permits a space-filling tessellation, as shown in Fig. 4c. For the "staircase" approximation to a box-shaped room, the boundary $\partial\mathcal{V}$ is the jagged outer surface in Fig. 4c. The FV method described in [19] allows for fitted boundary cells that can conform to straight walls, but for our purposes a staircase approximation is sufficient since the simulated room volumes will be large in comparison to the grid cells.

For the remainder of this section, it is assumed that $\boldsymbol{x} \in \boldsymbol{G}_h$ where $\boldsymbol{G}_h$ is the FCC grid with boundary $\partial\mathcal{V}$. We have the following values for the volume of a cell, V, and the surface area of one rhombic side, S, in terms of the grid spacing h (the *inradius* of the cell is $h/2$) [26]:

$$V = \frac{h^3}{\sqrt{2}}, \quad S = \frac{h^2}{2\sqrt{2}}. \tag{18}$$

We can use the following FD approximations to (2) and (5):

$$\hat{p} = \rho\delta_{t-}\hat{u} \tag{19}$$
$$\rho\delta_{t-}\hat{\nu}_i = -\delta_{\boldsymbol{v}_i+}\hat{p}, \tag{20}$$

where $\hat{\nu}_i = \hat{\nu}_i(t, \boldsymbol{x})$ is an approximation to $\boldsymbol{\nu}(t, \boldsymbol{x})\cdot\boldsymbol{v}_i$ for $\boldsymbol{v}_i \in \Upsilon$, and $\Upsilon = \{-\Omega_F \cup \Omega_F\}$. After integrating both sides of (4) over a grid cell and applying the divergence theorem, the following FV approximation to (4) can be obtained [19]:

$$\frac{V}{\rho c^2}\delta_{t+}\hat{p} = -S\sum_{i=1}^{12} q_i\hat{\nu}_i - S_{(b)}q_{(b)}\hat{\nu}_{n,(b)}, \tag{21}$$

where $q_i = q_i(\boldsymbol{x})$ is an *indicator function* taking on the value of 1 when an interior cell is adjacent along the vector $\boldsymbol{v}_i \in \Upsilon$ and 0 otherwise, $S_{(b)} = S_{(b)}(\boldsymbol{x})$ is the total surface area of the boundary cell on $\partial\mathcal{V}$, $q_{(b)} = q_{(b)}(\boldsymbol{x})$ is an indicator function taking on the value of 1 when $\boldsymbol{x}$ is a boundary cell and 0 otherwise, and $\hat{\nu}_{n,(b)}$ is an approximation to the velocity component of $\boldsymbol{\nu}$ normal to the boundary. Combining (18)-(21), the FV scheme reduces to the 13-point FD scheme for the wave equation on the interior cells of the FCC grid (cells where $q_i = 1$, $q_{(b)} = 0$) [19].

The boundary velocity component $\hat{\nu}_{n,(b)}$ is given by a passive discretisation of the impedance boundary condition (6) [19]:

$$\hat{\nu}_{n,(b)} = Y_0\left(A\delta_{t+}\hat{p} + B\mu_{t+}\hat{p} + C\mu_{t+}\hat{g}\right), \tag{22}$$
$$\delta_{t+}\hat{g} = \mu_{t+}\hat{p}, \tag{23}$$

with A, B, C non-negative for passivity [19] and where μ_{t+} is the following averaging operator:

$$\mu_{t+}\hat{p} = \frac{\hat{p}(t+k, \boldsymbol{x}) + \hat{p}(t, \boldsymbol{x})}{2}. \tag{24}$$

Combining (18)-(24) we get the following update at the boundary cells in terms of $\hat{u}$ (the extra storage $\hat{g}$ can be eliminated):

$$\hat{u}(t+k, \boldsymbol{x}) = \frac{1}{\gamma}\left(2\beta + cS\sum_{i=1}^{12} q_i\delta_{\boldsymbol{v}_i+}\right)\hat{u}(t, \boldsymbol{x}) - \frac{\phi}{\gamma}\hat{u}(t-k, \boldsymbol{x}), \tag{25}$$

where:

$$\phi = \frac{V}{ck^2} + S_{(b)}\left(\frac{A}{k^2} - \frac{B}{2k} + \frac{C}{4}\right), \tag{26}$$
$$\beta = \frac{V}{ck^2} + S_{(b)}\left(\frac{A}{k^2} - \frac{C}{4}\right), \tag{27}$$
$$\gamma = \frac{V}{ck^2} + S_{(b)}\left(\frac{A}{k^2} + \frac{B}{2k} + \frac{C}{4}\right). \tag{28}$$

Finally, we have $S_{(b)} = 4S$ at a wall node and $S_{(b)} = 7S$ at an edge. It is possible to not have "corner" cells, as in Fig. 4c, where each corner of the approximated box is shared by three edge cells.

6.1. Numerical Energy and Room Modes

A useful debugging tool for such numerical schemes is to measure the total numerical energy of the system [19]. A measure of the stored numerical energy on the interior, $\mathfrak{h}_i = \mathfrak{h}_i(t)$, is given by [19]:

$$\mathfrak{h}_i = \sum_{\boldsymbol{x}\in\boldsymbol{G}_h} \left(\frac{V}{2\rho c^2}\hat{p}^2 + \frac{\rho S h}{4} \sum_{i=1}^{12} q_i \hat{\nu}_i s_{t-} \hat{\nu}_i \right) , \qquad (29)$$

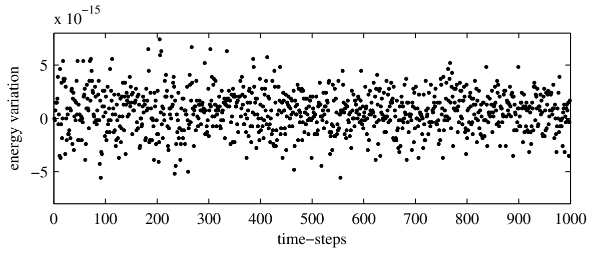
where s_{t-} is the time-shift operator: $s_{t-}\hat{\nu}_i(t,\boldsymbol{x}) = \hat{\nu}_i(t-k,\boldsymbol{x})$. The stored energy at the boundary, $\mathfrak{h}_b = \mathfrak{h}_b(t)$, is given by [19]:

$$\mathfrak{h}_b = \frac{Y_0}{2} \sum_{\boldsymbol{x}\in\boldsymbol{G}_h} S_{(b)} q_{(b)} \left(A\hat{p}^2 + C\hat{g}^2 \right) . \qquad (30)$$

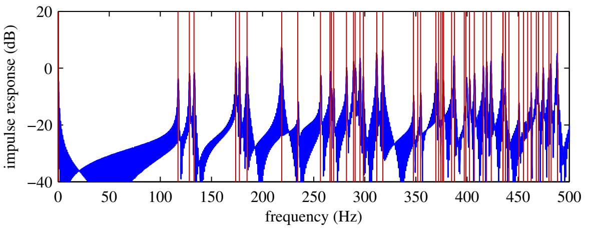
The total energy $\mathfrak{h} = \mathfrak{h}_i + \mathfrak{h}_b$ should be non-increasing:

$$0 \le \mathfrak{h}(t) \le \mathfrak{h}(0) , \quad t > 0 , \qquad (31)$$

and in the lossless case ($B = 0$), energy should be conserved, $\mathfrak{h}(t) = \mathfrak{h}(0)$ for all t, to numerical precision. To verify this, a simulation was conducted for a 0.1 m × 0.1 m × 0.1 m box with $c =$ 344 m/s, a Kronecker delta initial condition, $A = 2, B = 0, C =$ 4, and with F_s = 44.1 kHz. As seen in Fig. 5a, the normalised variation in the total energy is negligible.



(a) *Variation in total energy,* $(\mathfrak{h}(t) - \mathfrak{h}(0))/\mathfrak{h}(0)$



(b) *Impulse response in low frequencies*

Figure 5: *Numerical energy and room modes*

We can also check that the impulse response of our staircase-approximated box-shaped room agrees with the analytical modes of a box with reflective (phase-preserving) boundaries. To this end, another simulation was run on a 1.45 m × 1.28 m × 1.32 m box with $A = B = C = 0$, and with a Kronecker delta initial condition. As seen in Fig. 5b, the room modes from the simulation agree with the low-frequency analytical modes (vertical lines). A series of snapshots in time from another small simulation is displayed in Fig. 8 to give an idea of the wave propagation in the combined FD-FV scheme.

7. IMPLEMENTATION AND GPU ACCELERATION

This section details the implementation of the FCC 13-point scheme with impedance boundaries, firstly in serial C code, and then using Nvidia's CUDA language to run large-scale simulations on graphics processing units (GPUs).

7.1. Serial C code

To implement the scheme in serial C code, the 3-D grid is mapped to linear memory in a standard *row-major* arrangement for each Z layer. Only two state data grids are required, as a read then overwrite procedure can be used to access the center point from two time-steps ago. For both the interior and the boundaries, the grid is calculated using separate updates for consecutive points in linear memory. Fig. 6 shows the grid points used for updating the interior of the scheme.

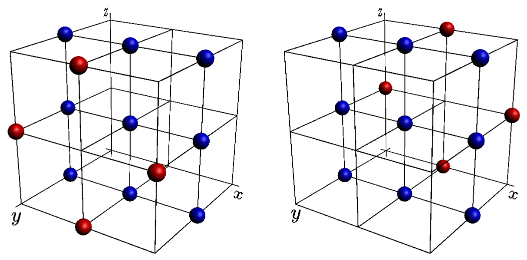


Figure 6: *Grid points (FCC remapped to cubic grid) used by even (left) and odd (right) updates.*

There are nine points which are common to both (marked blue), and four which differ. Similarly at the boundaries, we apply separate updates to odd then even data points. Extra conditional statements are required to identify the faces and edges.

7.2. GPU kernel design

At each time-step, updating the grid points is data independent, and so can clearly benefit from parallel execution. The main concern in terms of GPU efficiency is ensuring *memory coalescing*. When consecutive CUDA threads access consecutive memory locations, data transfers are coalesced and occur much faster. For computations which are limited by memory bandwidth, such as room acoustics simulations with FD methods, this is the most important performance aspect.

In terms of the FCC scheme here, achieving memory coalescing requires some careful attention to the design of the CUDA kernel. Ideally we want to update the grid with a single kernel launch, with enough threads to cover the entire grid. However, as previously detailed, consecutive points in memory use a different update, and access different points. Simply applying a conditional statement in the kernel would result in non-coalesced memory access, and non-optimal performance.

To ensure fully coalesced access, the kernel is designed to pick up all of the points required by both updates, the blue and both sets of red points in Fig. 6. The eight red points are accessed, and then a conditional statement calculates a partial sum based on the points required, as shown in line 27 of the kernel code. The remaining nine common points are then summed using a single update equation at line 30. The kernel also contains the boundary updates,

```
__global__ void UpDate( ... )
{
    // get X,Y,Z from thread and block Id's
    int X = blockIdx.x * Bx + threadIdx.x;
    int Y = blockIdx.y * By + threadIdx.y;
    int Z = blockIdx.z * Bz + threadIdx.z;

    double ps,s1,s2,s3,s4,s5,s6,s7,s8;

    // Interior Points
    if( ... ){

       // get linear position
       int cp = Z*area+(Y*Nx+X);

       // Load differing sum parts
       s1 = u1[cp+1+Nx];
       s2 = u1[cp-1+Nx];
       s3 = u1[cp+Nx-area];
       s4 = u1[cp+Nx+area];

       s5 = u1[cp+1-Nx];
       s6 = u1[cp-1-Nx];
       s7 = u1[cp-Nx-area];
       s8 = u1[cp-Nx+area];

       if ((X+Z)%2==0) ps = s1 + s2 + s3 + s4;
       else            ps = s5 + s6 + s7 + s8;

       u[cp] = 0.25*(u1[cp+1]       + u1[cp-1]
                    +u1[cp-1-area] + u1[cp+1-area]
                    +u1[cp-1+area] + u1[cp+1+area]
                    +u1[cp-area]   + u1[cp+area]
                    +ps)
                    -u1[cp]-u[cp];
    }

    // Update boundaries
    ...
}
```

Figure 7: *CUDA kernel for FCC scheme.*

which are implemented using separate conditional statements as per the serial C code. These are left out for brevity.

In terms of GPU memory design, rather than using a complicated shared memory approach, the kernel accesses data using the read-only data cache available on Kepler cards. In the parameters of the kernel declaration, the data pointer is declared as:

```
const double * __restrict__ u1
```

No additional code elements are required to implement this cache functionality, and it operates correctly even though the data pointers are swapped at each time iteration. This provides a significant performance gain of around 15% over directly reading from global memory.

7.3. Performance

In order to benchmark the system, a standard simulation was computed using both the serial C version and the CUDA version. This consisted of a grid size of 800x520x470 (195,520,000 points) computed for 44,100 time-steps, using double precision floating-point arithmetic. Serial C code was tested on an Intel Xeon E5-2620 with -O3 compiler optimisation and the CUDA code was tested on an Nvidia Tesla K20. The resulting times were: serial C code - 30 hours, CUDA - 38.5 minutes, a speed-up of 46x. A simulation of this size was not fully run in MATLAB since after 100 time-steps it was estimated that 44,100 time-steps would take 30 days, even with vectorised code.

8. PRACTICAL COMPARISONS

8.1. Computation times

In this section, we conduct some experiments using GPU acceleration to compare the computation times for the FCC 13-point scheme, the IWB scheme and IISO2 27-point schemes, and the SLF 7-point scheme. The purpose of these tests is to see if the time taken to compute a certain grid size and number of time-steps aligns with some measure of computational costs of the schemes. Analysing computational cost usually begins with the *computational density* of a scheme, which is the number of points, or updates, per unit space and time. The densities of specific operations (additions, multiplications, memory reads) are then obtained by multiplying the computational density by the number of specific operations carried out at each point-wise update. For an N-point stencil using M shells of points, the number of multiplies is $M+1$, the number of additions is N, and the number of memory reads is $N+1$, at each point-wise update. However sometimes multiplications can be swapped for additions (subtractions), depending on the values of $\boldsymbol{\alpha}$ and λ, as in (17). The density of the memory reads is also known as the *memory bandwidth*.

The densities of the spatial grids are σ/h^3, where $\sigma = 1$ for the cubic grid and $\sigma = \sqrt{2}$ for the FCC grid. The computational density of a scheme on a specific grid is then $\sigma(h^3k)^{-1}$. Fixing the Courant number λ and the wave speed c, we can write the computational density as $(c\sigma/\lambda)h^{-4}$. Thus, to put schemes on an equal footing we can use the grid spacing $h = \sqrt[4]{\sigma/\lambda}h'$ with a common spatial step h' so that each scheme has the density ch'^{-4}. Courant numbers are chosen at the stability limit for the reasons stated in Section 4.

We set the computational densities of the four schemes to be equal and compare the times taken to simulate the wave propagation in a 512 m^3 room for 0.2 s using 10^{12} updates ($h' = 13.7$ mm, $c = 344$ m/s) in time and space using double floating-point precision. As a point of reference, the IWB is run with $F_s = 25.1$ kHz. We also simulate the same sized space for 0.1s (0.5x10^{12} updates) and a 256 m^3 room for 0.1s (0.25x10^{12} updates). Since the interior updates are the bulk of the computational load (for boundary conditions which are not too complex) we employ Dirichlet boundary conditions: $\hat{u}(t, \boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \partial\mathcal{V}$. We use a spatially raised cosine initial condition with zero initial velocity. The floating-point operations per second (FLOPS) and memory bandwidth is different for each scheme (IWB and IISO2 are equivalent in this respect), which should translate to different computation times.

For the comparison tests, each of the four schemes was implemented with only an interior update with a non-updated halo at the boundaries (fixed boundaries). Each CUDA kernel was designed in the same way, using a 3D thread block of size 32x4x2 and issuing enough threads to cover the entire data grid. Data was accessed using the read-only data cache as described in the previous section. The simulations were performed on a single Nvidia Tesla K20 card. We list the computation times for these simulations in Table 1.

We see that the 7-point SLF scheme and the 13-point FCC scheme respectively take (roughly) 80% and 90% of the time taken for the 27-point IWB and IISO2 schemes. The number of additions and multiplications carried out at each point is not sufficient to create a bottleneck in the GPU execution because the theoretical maximum FLOP rate of the Tesla K20 card is 1.17 teraFLOPS at double-precision [28]. Counting the number of addi-

Table 1: *Computation times (in seconds) of schemes*

# updates	SLF	FCC	IISO2	IWB
1.00×10^{12}	256.6	282.0	321.6	321.1
0.50×10^{12}	128.3	141.0	160.8	160.6
0.25×10^{12}	64.0	70.4	80.3	80.2

tion and multiplication operations performed in the GPU kernel, the 27-point schemes are using less than 20% of the maximum FLOP rate. Thus, we can attribute these differences to memory bandwidth. The fact that the times do not scale linearly with the relative memory bandwidths indicates that the GPU cache is also having an effect. For large problems, such as room acoustics simulations, the computation time for a given scheme on a GPU scales linearly with the number of grid points and number of time-steps computed. This is confirmed in Table 1, so we can conclude that for the same computational density the execution time is lower for schemes with lower memory bandwidth (generally, for schemes which employ fewer points). Now we must relate this to some measure of accuracy in the sound produced.

The accuracy of a FD approximation is generally defined in terms of Taylor series expansions [24], but these schemes are all second-order accurate, and for audio we are more interested in minimising the wave speed error across a wide range of spatial or temporal frequencies. Previous studies have compared the computational efficiencies of different schemes in terms of minimising wave speed error across a range of wavenumbers [12–14]. We will refer to the relative computational efficiency measure devised in [13] for the interior FD scheme. This measure gives the computational density in the SLF scheme required to keep the wave speed error below some threshold up to some critical frequency (in Hz) in all directions, relative to the computational density required to achieve the same in another scheme. These numbers [13] are listed in Table 2 and we have taken the liberty to increase the efficiency for the FCC scheme by a factor of two for the reasons stated in Section 4, but this is confirmed in another study [14].

Table 2: *Relative computational efficiencies of schemes [13]*

	SLF	FCC	IISO2	IWB
2% error	1.00	11.26	8.67	7.01
4% error	1.00	9.30	7.14	7.07
8% error	1.00	6.92	5.31	6.99

Table 2 tells us that for the same computational density, the FCC scheme will have a lower worst-case directional wave speed error than the IWB and IISO2 schemes, until 8% wave speed error. We have yet to see any tests conducted to relate this measure to actual perception, but we suspect that 8% wave speed error is too optimistic for large-scale room acoustics. It is true that the IWB scheme has no wave speed error for plane waves travelling along grid axes [13], but this is also the case for the 13-point scheme along the same directions (relative to the cubic grid) [13,14]. Similarly, the SLF scheme has no wave speed error for plane waves travelling along grid diagonals [11, 13] (the *space diagonals* of a cubic cell, as opposed to the side diagonals). In any case, the wave propagation should be uniform in all directions, as described by spherical wave solutions to (1) [29], so we consider the worst-case direction to be critical. With the relative computational efficiencies in Table 2, and the data in Table 1, we can conclude that less dispersion is achieved for less computation time with the FCC scheme than the other schemes, when less than 8% wave speed error is desired in every direction.

8.2. Use of finite memory

The discussion above does not put a constraint on memory use, but we are actually limited to 5GB per card, which translates to roughly 300 million grid points at double-precision (two states per point), or 600 million at single-precision. We can make another comparison if time is not an issue and we only want the best possible sound from a given sized virtual room. Since the grid density is σ/h^3, we can use $h = h'/\sqrt[3]{\sigma}$ to equalise the schemes for number of points in a given sized space. The computational density then becomes $(\sigma^{7/3}/\lambda)ch'^{-4}$, which gives $\sqrt{3} \approx 1.73$, $2^{2/3} \approx 1.59$, $\sqrt{4/3} \approx 1.33$, and 1 for the SLF, FCC, IISO2, and IWB schemes respectively. As the computational density is higher for the FCC scheme than the 27-point schemes, the dispersion will certainly be less (below 8% wave speed error), since we know it is already better for equal computational density. With equal memory use, the SLF has roughly 1.09 times the computational density of the FCC scheme, but the FCC scheme is nearly seven times more efficient for an 8% threshold in the wave speed error, as seen in Table 2. Thus, we can conclude that for equal memory use, the FCC scheme has less numerical dispersion than the other FD schemes on the cubic grid.

Using the full 5GB of memory on the Tesla K20 card, a simulation was run with the FCC scheme for a 540 m^3 room with $F_s = 44.1$ kHz. It took 7.35 hours to simulate a four second room impulse response, and there was a 70x speed-up over serial C code (on a single core). At double precision, 5GB allowed for a 270 m^3 room with $F_s = 44.1$ kHz, and this simulation took 3.78 hours, with a 53x speed-up over serial C code (on a single core). Audio examples of these simulations are available at `http://www2.ph.ed.ac.uk/~s0956654/Site/VirtualRoomAcoustics.html`.

9. CONCLUSIONS

In this paper, a room acoustics model with frequency-dependent absorption at the walls was implemented. The interior sound field was approximated with a finite difference scheme on a face-centered cubic (FCC) grid and the boundary conditions were derived from finite volume methods. The computation of the scheme was accelerated using CUDA programming on an Nvidia Tesla K20 GPU card and there were significant speed-ups over serial C code run on a single core. The discretisation of the 13-point finite difference scheme was discussed and it was shown that the FCC grid is more efficient (in theory) at minimising numerical dispersion, below an 8% threshold. Practical tests on the GPU showed that the 13-point scheme on the FCC grid runs faster than 27-point schemes on a cubic grid for equal computational densities. This was attributed to the lower memory bandwidth of the FCC scheme. Considering the practical efficiency of the FCC scheme and the additional theoretical efficiency, it can be concluded that the 13-point scheme on the FCC grid is better-suited to large-scale room acoustics simulations than 27-point schemes on cubic grids. Future work will include the use of fitted boundary cells to simulate complex geometries, the addition of air viscosity to the room model, and investigating the use of multiple GPU cards in parallel.

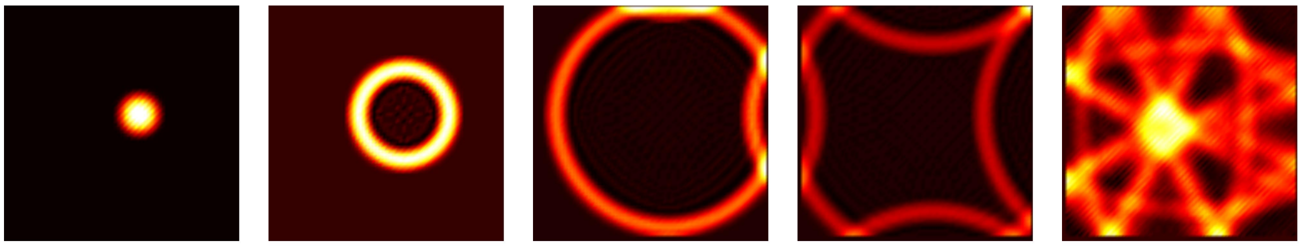


Figure 8: *Snapshots of $\hat{u}(t, \boldsymbol{x})$ on $x = 0$ plane for 0.5 m $\times$ 0.5 m $\times$ 0.5 m box with $A = 1$, $B = 10$, $C = 2$ at 44.1 kHz and with $c =$ 344 m/s after 10, 20, 60, 80, and 180 time-steps. Input is a raised cosine point source.*

10. REFERENCES

[1] J. B. Allen and D. A. Berkley, “Image method for efficiently simulating small-room acoustics,” *J. Acoustical Society of America*, vol. 65, p. 943, 1979.

[2] M. Vorländer, “Simulation of the transient and steady-state sound propagation in rooms using a new combined ray-tracing/image-source algorithm,” *J. Acoustical Society of America*, vol. 86, p. 172, 1989.

[3] F. Ihlenburg, *Finite element analysis of acoustic scattering*. Springer, 1998, vol. 132.

[4] R. D. Ciskowski and C. A. Brebbia, *Boundary element methods in acoustics*. Computational Mechanics Publications Southampton, Boston, 1991.

[5] 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.

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

[7] D. Botteldooren, “Acoustical finite-difference time-domain simulation in a quasi-Cartesian grid,” *J. Acoustical Society of America*, vol. 95, p. 2313, 1994.

[8] 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.

[9] 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.

[10] C. J. Webb, “Computing virtual acoustics using the 3D finite difference time domain method and Kepler architecture GPUs,” in *Proc. Stockholm Musical Acoustics Conf. (SMAC)*, Stockholm, Sweden, 2013.

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

[12] G. R. Campos and D. M. Howard, “On the computational efficiency of different waveguide mesh topologies for room acoustic simulation,” *IEEE Trans. Speech and Audio Processing*, vol. 13, no. 5, pp. 1063–1072, 2005.

[13] 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.

[14] 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.

[15] J. Sheaffer, B. M. Fazenda, D. T. Murphy, and J. A. S. Angus, “A simple multiband approach for solving frequency dependent problems in numerical time domain methods,” in *Proceedings of Forum Acusticum*, 2011, pp. 269–274.

[16] T. Tsuchiya, Y. Iwaya, and M. Otani, “Large-scale sound field rendering with graphics processing unit cluster for three-dimensional audio with loudspeaker array,” in *Proc. Int. Cong. Acoustics (ICA)*, Montréal, Canada, 2013.

[17] J. A. Laird, “The physical modelling of drums using digital waveguides,” Ph.D. thesis, University of Bristol, 2001.

[18] M.-L. Aird, “Musical instrument modelling using digital waveguides,” Ph.D. thesis, University of Bath, 2002.

[19] 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.

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

[21] D. P. Petersen and D. Middleton, “Sampling and reconstruction of wave-number-limited functions in N-dimensional Euclidean spaces,” *Information and control*, vol. 5, no. 4, pp. 279–323, 1962.

[22] K. Petkov, F. Qiu, Z. Fan, A. E. Kaufman, and K. Mueller, “Efficient LBM visual simulation on face-centered cubic lattices,” *IEEE Trans. Visualization and Computer Graphics*, vol. 15, no. 5, pp. 802–814, 2009.

[23] N. Röber, M. Spindler, and M. Masuch, “Waveguide-based room acoustics through graphics hardware,” in *Proc. Int. Computer Music Conf. (ICMC)*, 2006.

[24] J. Strikwerda, *Finite difference schemes and partial differential equations*. Society for Industrial Mathematics, 2004.

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

[26] J. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*. Springer-Verlag, 1988.

[27] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*. Saunders College, Philadelphia, 1976.

[28] “Nvidia Tesla K-series datasheet,” Nvidia Corp., Santa Clara, California, 2012.

[29] S. W. Rienstra and A. Hirschberg, “An introduction to acoustics,” *Eindhoven University of Technology*, 2013.