DYNAMIC GRIDS FOR FINITE-DIFFERENCE SCHEMES: CHANGING PARAMETERS OF MUSICAL INSTRUMENT SIMULATIONS IN REAL TIME

Silvin Willemsen

Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark
sil@create.aau.dk

Stefan Bilbao, Michele Ducceschi

Acoustics and Audio Group
University of Edinburgh
Edinburgh, UK

Stefania Serafin

Multisensory Experience Lab
Aalborg University Copenhagen
Copenhagen, Denmark

ABSTRACT

Many techniques of modelling musical instruments exist of which time-stepping methods (e.g., finite-difference time-domain (FDTD) methods) are considered the most flexible and generalisable in terms of the type of systems they can model, both linear and non-linear. These methods do, however, lack the capability of handling smooth parameter changes while retaining optimal simulation quality, something other techniques are better suited for. Not sure this is completely true either...the reviewers will definitely point out examples from the massive FDTD literature (like CFD e.g., where they use immersed boundary techniques for fluid structure interaction with moving boundaries). This article proposes a method to dynamically alter the grids of simulations based on FDTD methods by smoothly adding and removing grid points from the system. This allows for dynamic parameter changes in physical models of musical instruments which are based on this technique. An instrument such as the trombone can now be modelled using FDTD methods, as well as physically impossible instruments.

1. INTRODUCTION

Simulating musical instruments using physical modelling is a well-established field. Among the reasons of why one would simulate an instrument rather than sample an existing one, is that one could extend the capabilities of this instrument beyond what is physically possible, such as changing material properties or size of the instrument dynamically. "on the fly" is too informal. Maybe "under dynamic playing conditions".

Much more intro here obviously

One of the incentives of simulating the physics of musical instruments rather than sampling their real-world counterparts, is that the virtual instrument can be manipulated in physically impossible ways. OK, it's good to have this in here, but this is immediately kind of a red flag for JASA, where they'd expect you to be attacking a problem from the real world first. Otherwise, they might well say that this belongs in CMJ. So: suggesting referring to real world examples first, then bring up the idea of imaginary instruments... Will do! Examples of this could be to change material properties, or even the shape of the instrument. An example of a real-world instrument that requires these manipulations is the trombone, where tube length is dynamically controlled by the player.

Existing physical modelling techniques include mass-spring systems [REF] and digital waveguides [REF]. SOTA on trombone / variable parameters with waveguides here!

Modal synthesis [REF], though requiring some assumptions and simplifications for most systems, does allow for easy and smooth parameter changes – as seen in [REF] and [REF] – and could thus be a good candidate for implementing the aforementioned manipulations. In the case of the trombone, and due to its non-homogenous geometry, there is no closed-form solution available and the modes would need to be calculated for every single slide configuration. Finite-difference time domain (FDTD) methods on the other hand, though generally more computationally expensive than other techniques, are more flexible and generalisable and do not need as many simplifications as modal synthesis does.OK, I think reviewers may not like this at all. Particularly in the case of the trombone, where people like Perry Cook were doing things with variable delay lines (I think) a long time ago. if a waveguide person gets this for review, they will probably get annoyed. These methods subdivide continuous partial differential equations (PDEs) that describe the physics of the system at hand into a grid of discrete points in space and time. The distance between two discrete points in space (the grid spacing) and the time step between two discrete moments in time are closely connected through a *stability condition*. This condition dictates the maximum number of points allowed to describe system before it gets unstable and the simulation "explodes". The closer the grid spacing is to the stability condition, the higher the quality of the simulation. If the condition is satisfied with equality, the quality of the simulation is at a maximum. OK, this whole argument above seems out of place at little. Plus there are no references, or an indication of what quality means. As in: reviewers probably will not grasp why this is an important problem to solve. I suggest removing. Better to focus attention here away from the detailed numerical details (like grid spacings and numerical stability conditions and stuff) as the reviewers will probably see these as just small technical details. The important thing is the dynamic behaviour... Yes, I completely agree, the introduction mainly consists of notes at this point, some that were left from when I started writing this paper long ago :) Furthermore, the stability condition depends on the parameters of the model, such as material properties or size of the system, i.e., parameters that could be changed on the fly As above.

This article proposes a method to smoothly add and subtract points from a system in real time so that the stability condition is always satisfied with equality. This allows for a simulation where the parameters can be changed dynamically without running into either stability or quality issues. One can consider this work an introduction to implementing the trombone using FDTD methods.

2. CONTINUOUS SYSTEMS

The physics of dynamic systems is commonly described using partial differential equations (PDEs) operating in continuous time. To aid the illustration of the proposed method, the 1D wave equation will be used. This does not mean that the method is limited to this, and could be extended to other (linear) systems, possibly even higher dimensional ones.

The 1D wave equation may be written

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}\,, \tag{1}$$

and is defined over spatial domain $x \in [0, L]$, for length L (in m) and time $t \geq 0$ (in s). c (in m/s) is the wave speed. The dependent variable $u(x,t)$ in Eq. (1) may be interpreted as the transverse displacement of an ideal string, or the acoustic pressure in the case of a cylindrical tube. Two possible choices of boundary conditions are

$$u(0,t) = u(L,t) = 0 \quad \text{(Dirichlet)}, \tag{2a}$$

$$\frac{\partial}{\partial x}u(0,t) = \frac{\partial}{\partial x}u(L,t) = 0 \quad \text{(Neumann)}, \tag{2b}$$

which describe a 'fixed' and 'free' boundary respectively in the case of an ideal string, and 'open' and 'closed' conditions respectively in the case of a cylindrical acoustic tube.

2.1. Dynamic parameters

In the case of the 1D wave equation, only the wave speed c and length L can be altered. If Dirichlet-type boundary conditions – as in Eq. (2a) – are used, and under static conditions the fundamental frequency f_0 of the 1D wave equation can be calculated according to

$$f_0 = \frac{c}{2L}. \tag{3}$$

In the dynamic case, and under slow (sub-audiorate) variations of c or L, Eq. (3) still approximately holds. From Eq. (3), one can easily conclude that in terms of fundamental frequency, halving the length of Eq. (1) is identical to doubling the wave speed and vice versa. Looking at Eq. (1) in isolation, f_0 is the only behaviour that can be changed. One can thus leave L fixed and make c dynamic (or time-varying), i.e., $c = c(t)$, which will prove easier to work with in the following section. OK: here, need to refer back to the two primary cases here: the trombone, and the string under variable tensioning, and justify the use of a time-varying c only in these cases. Otherwise this becomes too abstract.

3. NUMERICAL METHODS

This section will give a brief introduction of physical modelling using FDTD methods, including details on stability and quality of the simulations based on these methods.

3.1. Discretisation

Using FDTD methods, the continuous 1D wave equation in Eq. (1) can be discretised into points in space and time. The spatial variable can be discretised using $x_l = lh$ (read: x at location l) with integer $l \in [0, \ldots, N]$, distance between two consecutive grid points h (in m), also referred to as the grid spacing, and total number of points $N+1$ (including the boundaries) where the total number of intervals between the grid points is described as

$$N = \lfloor L/h \rfloor, \tag{4}$$

with $\lfloor \cdot \rfloor$ denoting the flooring operation. The temporal variable can be discretised using $t_n = nk$ with positive integer n, time step $k = 1/f_\text{s}$ (in s) and sample rate f_s (in Hz). The state variable u can then be approximated using $u(x,t) \approx u_l^n$, where grid function u_l^n is the displacement of u at spatial index l and time index n.

The following operators can then be applied to u_l^n to get the following approximations to the derivatives in Eq. (1)

$$\delta_{tt} u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) \approx \frac{\partial^2 u}{\partial t^2}\,, \tag{5a}$$

$$\delta_{xx} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \approx \frac{\partial^2 u}{\partial x^2}\,. \tag{5b}$$

Substituting these definitions into Eq. (1) yields the following finite-difference scheme (FDS)

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n. \tag{6}$$

Expanding the operators as in (5) and solving for u_l^{n+1} (which is the only unknown) yields the following update equation

$$u_l^{n+1} = 2u_l^n - u_l^{n-1} + \lambda^2\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right), \tag{7}$$

saying that u at the next time index $(n+1)$ can be calculated using only values of u at the current (n) and previous time indices $(n-1)$. This update can be implemented in software, such as `MATLAB` or `C++`. In Eq. (7),

$$\lambda = \frac{ck}{h} \tag{8}$$

is referred to as the Courant number and determines whether the system is stable, as well as the quality and behaviour of the simulation. This will be described in detail in Sections 3.2 and 3.3.

In the FDS described in Eq. (6), the boundary locations are at $l = 0$ and $l = N$. Substituting these locations into Eq. (7) seemingly shows that grid points outside of the defined domain are needed, namely u_{-1}^n and u_{N+1}^n. These can be referred to as *virtual grid points* and can be accounted for by using the boundary conditions in Eq. (2). Discretising these yields

$$u_0^n = u_N^n = 0 \quad \text{(Dirichlet)} \tag{9a}$$

$$\delta_{x\cdot} u_0^n = \delta_{x\cdot} u_N^n = 0 \quad \text{(Neumann)} \tag{9b}$$

where

$$\delta_{x\cdot} u_l^n = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right) \approx \frac{\partial u}{\partial x} \tag{10}$$

is a second-order accurate approximation of the first-order spatial derivative. The Dirichlet condition in (9a) says that the displacement of u at the boundary locations are always 0. In practice, this means that these grid points do not need to be updated and the spatial range of calculation for Eq. (7) then becomes $l = [1, \ldots, N-1]$. If the Neumann condition is used, the boundary points do need to be updated as these are not necessarily 0; rather, their 'slope' is 0. Eq. (9b) can then be expanded to yield defnitions for these virtual grid points

$$u_{-1}^n = u_1^n \quad \text{and} \quad u_{N+1}^n = u_{N-1}^n\,. \tag{11}$$

Now that the full system is described, the output sound can be

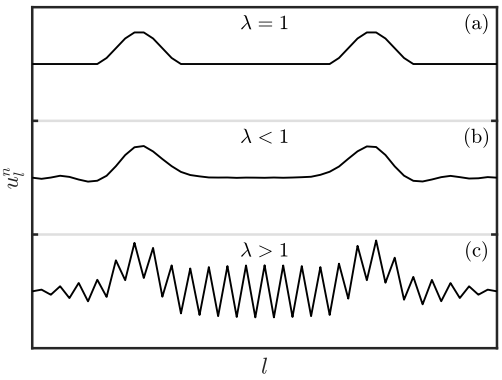

Figure 1: State u_l^n with $N = 50$ and $f_s = 44100$ visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (12) is not satisfied and the system is unstable.

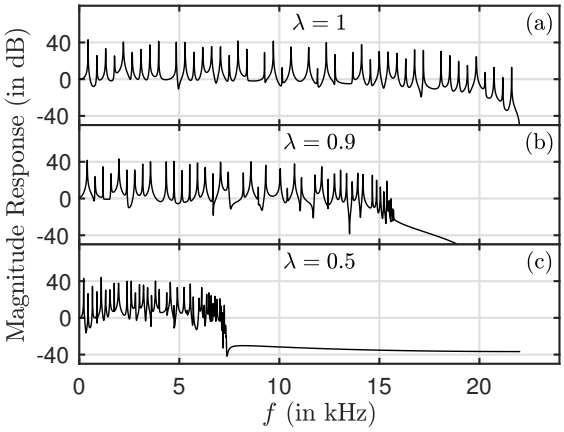

Figure 2: Bandwidths of the simulation output with $f_s = 44100$ Hz and (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$.

retrieved by 'recording' the state u_l^n in Eq. (7) at $0 < l < N$ (when using fixed boundary conditions) and listening to that at the given sample rate f_s. Though the output location does not change the behaviour of the system, it will determine the amplitude of different harmonic partials in the system output.

3.2. Stability

Discretising continuous equations using numerical methods places limits on the parameters describing it. A wrong choice of parameters could render the system unstable and make it "explode". In the case of the update in Eq. (7) it can be shown – using Von Neumann analysis [1] – that the system is stable if

$$\lambda \leq 1, \tag{12}$$

which is referred to as the Courant-Friedrichs-Lewy (CFL) stability condition. The closer λ is to this condition, the higher the quality of the simulation (see Section 3.3) and if $\lambda = 1$, Eq. (7) provides an exact solution to Eq. (1) (see Figures 1a and 1b). If $\lambda > 1$ the system will become unstable (see Figure 1c).

Recalling (8) can rewrite Eq. (12) in terms of grid spacing h to get

$$h \geq ck. \tag{13}$$

This shows that the CFL condition in (12) puts a lower bound on the grid spacing, determined by the sample rate and wave speed. Usually, the following steps are taken to calculate λ

$$h := ck, \quad N := \left\lfloor \frac{L}{h} \right\rfloor, \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}. \tag{14}$$

In other words, condition (13) is first satisfied with equality and used to calculate integer N according to Eq. (4). Thereafter, h is recalculated based on N and used to calculate λ. The calculation of λ in Eq. (14) can be compactly rewritten as

$$\lambda = \frac{ck}{L} \cdot \left\lfloor \frac{L}{ck} \right\rfloor. \tag{15}$$

The flooring operation causes the CFL condition in (12) to not always be satisfied with equality and results in a reduced simulation quality described in the following section.

3.3. Simulation Quality

As mentioned above, the Courant number λ decides the quality of the simulation. Choosing $\lambda < 1$ will decrease this quality in two ways. Firstly, it will decrease the maximum frequency that the simulation is able to produce, i.e., it will decrease the bandwidth of the output sound of the system. See Figure 2.

By analysing the scheme in Eq. (7), it can be shown that the maximum frequency produced by the system can be calculated using [2, Chap. 6]

$$f_{\text{max}} = \frac{f_s}{\pi} \sin^{-1}(\lambda). \tag{16}$$

Note that only a small deviation of λ from condition (12) already has a profound effect on the bandwidth of the output. Secondly, choosing $\lambda < 1$ causes numerical dispersion. See Figures 1b and 2. Harmonic partials get closer together at higher frequencies (i.e. get more inharmonic) as λ decreases, which is generally undesirable.

can cut this paragraph → Apart from the recalculation of λ due to the flooring operation in Eq. (14), a reason that one would choose $\lambda < 1$ could be to decrease the total number of grid points used in the simulation by increasing h. This makes the simulation less computationally expensive, while keeping a desired wave speed c and time step k. For 1-dimensional systems such as the 1D wave equation, this is rarely necessary.

4. DYNAMIC PARAMETERS

This section describes how parameters could be made dynamic using FDTD methods using the state of the art. To clarify, a *dynamic* parameter refers to one that is time-varying while the simulation is running.

Usually when simulating musical instruments, the parameters of individual FDSs are fixed for the entire simulation. For the 1D wave equation used in Section 2, only c and L can be made dynamic, but as elaborated on in Section 2.1, these manipulations are equivalent when looking the fundamental frequency f_0 of the system. We continue by looking at a dynamic wave speed c while the length L is fixed.

In a discrete setting, several things need to be taken into account as c changes. First of all, a change in c causes a change in

λ according to Eq. (15) affecting the simulation quality and bandwidth. Secondly, and more importantly, a change in c could result in a change in N through Eq. (4). As N directly relates to the number of grid points, this raises questions as to *where* and especially *how* one would add and remove points to the grid according to the now-dynamic wave speed.

One method that could be used to go from one grid configuration to the next is full-grid interpolation as described in [2, Chap. 5]. However, this method essentially has a lowpassing effect on the entire system state every time N is changed and can cause artefacts in the output sound due to the interpolation. A (much) higher sample rate could be used to avoid these issues, but this would render this method impossible to work in real time.

Another solution could be to set and fix N at the beginning of the simulation and tune c away from the stability condition by decreasing it, such as done in [3]. This would avoid problems with stability, as decreasing c would continue to satisfy condition (12), but the simulation would end up with lower quality, exhibiting dispersive and bandlimiting effects as discussed in Section 3.3. In essence, decreasing the value of c immediately translates to decreasing the value of λ as h and k are left unchanged. On top of this, c is limited by Eq. (13) and increasing it beyond a certain value would render the system unstable.

The only way to circumvent the aforementioned undesirable effects such as dispersion and bandlimiting, is to somehow allow N to be fractional or non-integer. This will remove the necessity of the flooring operation in Eq. (4) and Eq. (15), and will consequently satisfy the CFL condition in (12) with equality at all times. Eq. (3) can then be rewritten in terms of N by substituting Eq. (4) into Eq. (3) (using Eq. (13) satisfied with equality) yielding

$$f_0 = \frac{1}{2Nk}\,. \tag{17}$$

This shows that if $\lambda = 1$, N, which is now not necessarily an integer, solely decides the fundamental frequency of the simulation.

Even if a fractional N would be possible, this still leaves the question of where and how to add and remove points to and from the grid. Furthermore, this change in grid size needs to happen in a smooth fashion so as not to create audible artefacts. This paper proposes a method that allows for a fractional N and smoothly changes grid configurations, i.e, the number of grid points.

5. THE DYNAMIC GRID

By now, it is hopefully clear to the reader why dynamic parameters would make an interesting case in the field of physical modelling, and why dynamically changing grid configurations would be a good solution to undesirable behaviour such as a decrease in bandwidth and increase in numerical dispersion.

First, this section will list the requirements of a method that dynamically changes FDTD grid configurations. Then, the proposed method will be described in detail and summarised, and finally, some details on its implementation are given.

5.1. Method requirements

Ideally, a method that dynamically changes the grid size of finite-difference schemes should

1. generate an output with a fundamental frequency f_0 which is linearly proportional to wave speed c ($f_0 \propto c$),
2. allow for a fractional N to smoothly transition between different grid configurations
3. generate an output containing $\lfloor N \rfloor - 1$ modes which are integer multiples of the fundamental ($f_p = f_0 p$ with integer p),
4. work in real time.

The last requirement is an optional one, but in order for the final implementation to be "playable", this is necessary.

5.2. Proposed Method

This section introduces the proposed method to dynamically and smoothly change the grid of FDSs to account for dynamic parameter changes. In the following, the location of a grid point (in m from the left boundary) i (such as $i = u_0$) at time index n is denoted by x_i^n.

5.2.1. System Setup

Consider a grid function, u_l^n with integer $M = \lceil 0.5L/ck \rceil$ with $\lceil \cdot \rceil$ denoting the ceiling operation and w_l^n with integer $M_w = \lfloor 0.5L/ck \rfloor$, i.e., half the number of points allowed by the stability condition, plus one for overlap. The grid functions can then be placed in line with each other with u_M^n and w_0^n (defined as the inner boundaries) overlapping, i.e., $x_{u_M}^n = x_{w_0}^n$, and u_0^n and $w_{M_w}^n$ (defined as the outer boundaries) located at $x_{u_0} = 0$ and $x_{w_{M_w}} = L$. See Figure 3a. The following boundary conditions are then imposed:

$$u_0^n = w_{M_w}^n = 0, \quad \text{(Dirichlet)} \tag{18a}$$

$$\delta_{x\cdot} u_M^n = \delta_{x\cdot} w_0^n = 0. \quad \text{(Neumann)}, \tag{18b}$$

i.e., the outer boundaries are fixed and the inner boundaries are free. The systems can then be connected at the inner boundaries using a rigid connection

$$u_M^n = w_0^n, \quad \forall n. \tag{19}$$

To sum up, a grid function with $N = \lfloor L/h \rfloor$ (as per Eq. (4)) is divided into two separate subsystems connected at their respective inner boundaries.

With the above boundary conditions imposed, the following state vectors can be defined:

$$\begin{aligned} \mathbf{u}^n &= [u_1^n, \ldots, u_M^n]^T, \\ \mathbf{w}^n &= [w_0^n, \ldots, w_{M_w-1}^n]^T, \end{aligned} \tag{20}$$

with T denoting the transpose operation, and have M and M_w points respectively. Note that the outer boundaries are excluded as they are 0 at all times. The vector concatenating (20) is then defined as

$$\boldsymbol{\mathcal{U}}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix}. \tag{21}$$

Note that the vectors in (20) and (21) also exist at the next $(n+1)$ and previous $(n-1)$ time indices.

Even though the new system has an extra (overlapping) grid point, the behaviour of the new system should be identical to that of the original system. That this holds will be shown below.

Using u_l^n and w_l^n in the context of the 1D wave equation, a

system of FDSs can be defined as

$$\begin{cases} \delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n + J_u(x_{u_M}^n) F \\ \delta_{tt} w_l^n = c^2 \delta_{xx} w_l^n - J_w(x_{w_0}^n) F \end{cases} \tag{22}$$

with spreading operators

$$J_u(x_i) = \begin{cases} \frac{1}{h}, & l = l_i = \lfloor x_i/h \rfloor \\ 0, & \text{otherwise} \end{cases} \quad \text{and}$$
$$J_w(x_i) = \begin{cases} \frac{1}{h}, & l = l_i = \lfloor x_i/h \rfloor - M \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

applying the effect of the connection F (in $\mathrm{m}^2/\mathrm{s}^2$) to grid points u_M^n and w_0^n respectively. Expanding the spatial operators in system (22) at inner boundaries u_M^n and w_0^n, recalling the Neumann condition in (18b) and the definition for the virtual grid points needed for this condition in Eq. (11) yields

$$\begin{cases} \delta_{tt} u_M^n = \frac{c^2}{h^2}(2u_{M-1}^n - 2u_M^n) + \frac{1}{h} F \\ \delta_{tt} w_0^n = \frac{c^2}{h^2}(2w_1^n - 2w_0^n) - \frac{1}{h} F. \end{cases} \tag{24}$$

Because of Eq. (19), it is also true that $\delta_{tt} u_M^n = \delta_{tt} w_0^n$, $\forall n$, and F can be calculated by setting the right side of the equations in (24) equal to each other:

$$\frac{c^2}{h^2}(2u_{M-1}^n - 2u_M^n) + \frac{1}{h} F = \frac{c^2}{h^2}(2w_1^n - 2w_0^n) - \frac{1}{h} F$$
$$F = h \frac{c^2}{h^2}(w_1^n - u_{M-1}^n).$$

Substituting this into system (24) after expansion of the second-time derivative yields the update of the inner boundaries

$$\begin{cases} u_M^{n+1} = 2u_M^n - u_M^{n-1} + \lambda^2(u_{M-1}^n - 2u_M^n + w_1^n) & \text{(25a)} \\ w_0^{n+1} = 2w_0^n - w_0^{n-1} + \lambda^2(u_{M-1}^n - 2w_0^n + w_1^n) & \text{(25b)} \end{cases}$$

which, (again, recalling Eq. (19)) are indeed equivalent expressions for the connected point and is necessary to satisfy the rigid connection. Here, w_1^n in Eq. (25a) acts as virtual grid point u_{M+1}^n and u_{M-1}^n in (25b) as virtual grid point w_{-1}^n, essentially connecting the two systems using the state of one in the update of the other.

5.2.2. *Changing the Grid*

The previous section describes the case in which the stability condition is satisfied with equality, i.e, when L/ck is an integer and $x_{u_M}^n = x_{w_0}^n$. The locations of the outer boundaries $x_{u_0}^n$ and $x_{w_{M_w}}^n$ are fixed, i.e.

$$x_{u_0}^n = x_{u_0}^0 \quad \text{and} \quad x_{w_{M_w}}^n = x_{w_{M_w}}^0 \quad \forall n.$$

If the wave speed c is then decreased, and consequently the grid spacing h according to Eq. (13) (with equality), all other points move towards their respective outer boundary (see Figure 3b). Calculating h this way allows this method to always satisfy the CFL condition in Eq. (12) with equality.

As mentioned above, the state of w_l^n can be used to calculate virtual grid point needed at the right boundary of u_l^n and vice versa. If the inner boundaries are not overlapping (i.e., $x_{u_M}^n \neq x_{w_0}^n$) a Lagrangian interpolator $I(x_i)$ at location x_i from the left

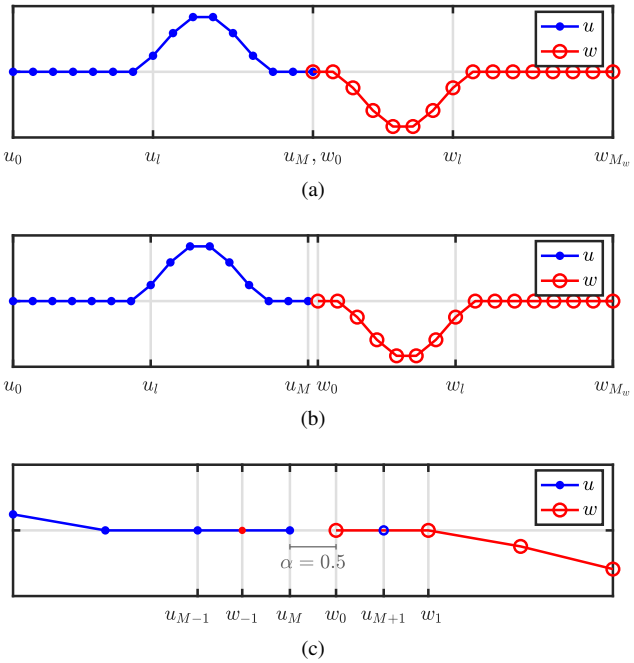

Figure 3: Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but the x and n are omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($N = 30$, $x_{u_M}^n = x_{w_0}^n$). (b) When c – and consequently h – are decreased and the positions of the grid points change ($N = 30.5$, $x_{u_M}^n \neq x_{w_0}^n$). (c) Figure 3b zoomed-in around the inner boundaries. The virtual grid points u_{M+1}^n and w_{-1}^n are shown together with the distance between them expressed using α in Eq. (26).

boundary (in m) can be used to calculate the value of these virtual grid points (also see Figure 3c for reference). The interpolator I is a row-vector the same length as $\boldsymbol{u}^n$ (from Eq. (21)) its values depending on what interpolation order is used. it's possible to skip until quadratic interpolation straight away from here, skipping the iterations In the following, the distance between the inner boundaries normalised with h is defined as

$$\alpha = \alpha^n = \frac{x_{w_0}^n - x_{u_M}^n}{h}, \tag{26}$$

and for clarity, I and $\boldsymbol{u}^n$ are indexed by m. Applying the interpolator to $\boldsymbol{u}^n$ yields

$$u_{M+1}^n = I^{\leftarrow}(x_{u_{M+1}}^n)\boldsymbol{u}^n \tag{27a}$$
$$w_{-1}^n = I(x_{w_{-1}}^n)\boldsymbol{u}^n, \tag{27b}$$

where $I^{\leftarrow}$ is a flipped and shifted version of I. If linear interpolation is used,

$$I_1(x_i) = \begin{cases} (1-\alpha), & m = m_i \\ \alpha, & m = m_i + 1 \\ 0, & \text{otherwise} \end{cases} \tag{28a}$$

and

$$I_1^{\leftarrow}(x_i) = \begin{cases} \alpha, & m = m_i^{\leftarrow} \\ (1-\alpha), & m = m_i^{\leftarrow} + 1 \\ 0 & \text{otherwise} \end{cases} \tag{28b}$$

with $m_i = \lfloor x_i/h \rfloor$ and $m_i^{\leftarrow} = \lfloor x_i/h + (1-\alpha) \rfloor$, where the shift in the latter is necessary to transform the location x_i to the right indices of $\boldsymbol{\mathcal{U}}^n$. Substituting (28) into (27) and expanding yields

$$u_{M+1}^n = I_1^{\leftarrow}(x_{u_{M+1}}^n)\boldsymbol{\mathcal{U}}^n = \alpha w_0^n + (1-\alpha) w_1^n, \tag{29a}$$

$$w_{-1}^n = I_1(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n = (1-\alpha) u_{M-1}^n + \alpha u_M^n. \tag{29b}$$

Using I_1, analysis of the output shows that the expected fundamental frequency f_0 is slightly higher when interpolation needs to happen than the one expected when using Eq. (17). Furthermore, modes higher than $f_s/4$ would follow an odd pattern up when decreasing the wave speed, opposite of what is expected. ← didn't know where (or whether) to include this.

One could extend the range of interpolation by one point to each side, using a cubic Lagrange interpolator instead. Although this would require w_{-1}^n to calculate u_{M+1}^n and vice versa, it is possible to solve this by treating the interpolation equations as a system of linear equations. Analysis of this method, though yielding a correct f_0 at all times, shows similar behaviour to the linear interpolation, with odd behaviour regarding modes higher than $f_s/4$.

Much better behaviour is observed when points of both u_l^n and w_l^n are used to calculate the values of the virtual grid points. This means to also use u_M^n to calculate u_{M+1}^n and w_0^n for w_{-1}^n. Now, the locations of the grid points used in the interpolation are not equidistant and a custom Lagrangian interpolator needs to be created. The lowest order interpolator that can be used here is the quadratic interpolator I_2

$$I_2(x_i) = \begin{cases} -(\alpha-1)/(\alpha+1), & m = m_i - 1 \\ 1 & m = m_i \\ (\alpha-1)/(\alpha+1), & m = m_i + 1 \\ 0, & \text{otherwise} \end{cases} \tag{30}$$

(with the flipped version $I_2^{\leftarrow}(x_i)$ defined similarly to (28b)) and when applied to Eq. (21) yields

$$u_{M+1}^n = I_2^{\leftarrow}(x_{u_{M+1}}^n)\boldsymbol{\mathcal{U}}^n = \frac{\alpha-1}{\alpha+1} u_M^n + w_0^n - \frac{\alpha-1}{\alpha+1} w_1^n \tag{31a}$$

$$w_{-1}^n = I_2(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n = -\frac{\alpha-1}{\alpha+1} u_{M-1}^n + u_M^n + \frac{\alpha-1}{\alpha+1} w_0^n. \tag{31b}$$

As will be shown in Section 6, quadratic interpolation yields the expected fundamental frequency at all times. One can show that when N is an integer, and thus $\alpha = 0$, Eqs. (31a) and (31b) can be substituted as w_1^n and u_{M-1}^n into Eqs. (25a) and (25b) respectively (as these acted as virtual grid points u_{M+1}^n and w_{-1}^n). Then recalling Eq. (19) it can be seen that the system is reduced to (25) and exhibits the same exact behaviour as the normal case.

5.2.3. *Adding and removing Grid Points*

When c, and consequently h, is decreased and the inner boundary points surpass the virtual points (i.e. $x_{u_M}^n \leq x_{w_{-1}}^n$ and $x_{w_0}^n \geq x_{u_{M+1}}^n$ or $\alpha \geq 1$) and $\lfloor N^n \rfloor > \lfloor N^{n-1} \rfloor$, a point is added to the right boundary of u_l^n and the left boundary of w_l^n (for both time indices n and $n-1$) in an alternating fashion:

$$\begin{cases} \mathbf{u}^n = [\mathbf{u}^n, I_3 \mathbf{v}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is odd,} \\ \mathbf{w}^n = [I_3^{\leftarrow} \mathbf{v}^n, \mathbf{w}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is even,} \end{cases} \tag{32}$$

where

$$\mathbf{v}^n = [u_{M-1}^n, u_M^n, w_0^n, w_1^n]^T,$$

and cubic Lagrangian interpolator

$$I_3 = \left[-\frac{\alpha'(\alpha'+1)}{(\alpha'+2)(\alpha'+3)} \quad \frac{2\alpha'}{\alpha'+2} \quad \frac{2}{\alpha'+2} \quad -\frac{2\alpha'}{(\alpha'+3)(\alpha'+2)} \right], \tag{33}$$

($I_3^{\leftarrow}$ being just a flipped, not shifted, version of (33)) with

$$\alpha' = \frac{x_{w_0}^n - (x_{u_M}^n + h)}{h}.$$

See Figure 4.

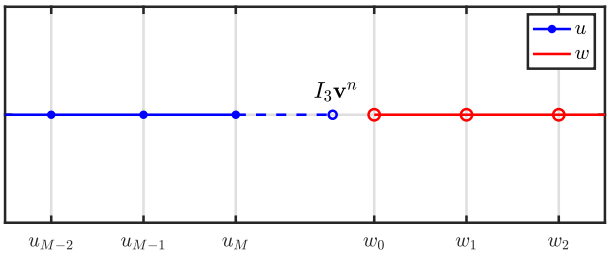

Figure 4: The moment when a point is added to $\mathbf{u}$ at location $x_{u_M} + h$ in Eq. (32). This figure shows an extreme case where this location is far from x_{w_0}, i.e., $\alpha' \not\approx 0$ in Eq. (33).

Except in the case of extremely quick parameter variations, α' in Eq. (33) is expected to be close to zero, i.e., $x_{u_M}^n + h \approx x_{w_0}^n$, meaning that $I_3 \approx [0, 0, 1, 0]$. This makes sense by looking at Figure 3c, as exactly when the boundary points u_M^n and w_0^n surpass the virtual points w_{-1}^n and u_{M+1}^n, these are going to be close to overlapping.

Removing grid points happens when c, and consequently h, is increased and $x_{u_M}^n \geq x_{w_0}^n$ (or $\lfloor N^n \rfloor < \lfloor N^{n-1} \rfloor$). Grid points are simply removed from $\mathbf{u}$ and $\mathbf{w}$ (again for both n and $n-1$) in an alternating fashion according to

$$\begin{cases} \mathbf{u}^n = [u_0^n, u_1^n ..., u_{M-1}^n]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w_1^n, w_2^n ..., w_{M_w}^n]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{34}$$

Until now, only adding and removing points in the center of the system has been considered. This location could be moved anywhere along the grid, the limit being one point from the boundary. In other words, both u_l^n and w_l^n need to have at least one point (excluding the outer boundaries). Furthermore, one does not have to add and remove points from $\mathbf{u}$ and $\mathbf{w}$ in an alternating fashion as in (32), but can just add and remove from (fx.) $\mathbf{u}$ leaving $\mathbf{w}$ the same size throughout the simulation. In the extreme case where $M = \lfloor N \rfloor - 1$ and $M_w = 1$ (leaving w_l^n with only one moving grid point, w_0^n) the method still works.

5.2.4. Displacement correction

A problem that arises from increasing h, is that it is possible that $u_M^n \not\approx w_0^n$ at the time when a grid point needs to be removed. As $x_{u_M}^n \approx x_{w_0}^n$ at the time of removal (except for extremely quick parameter variations), this violates the rigid connection in (19) and causes audible artefacts. A method is proposed that decreases the relative displacement of the inner boundaries the closer their grid-locations are together, i.e., the closer α in (26) is to 0:

$$\begin{aligned} u_M^n &:= u_M^n + \beta(w_0^n - u_M^n), \\ w_0^n &:= w_0^n - \beta(w_0^n - u_M^n), \end{aligned} \tag{35}$$

with

$$\beta(\epsilon) = \frac{(1-\alpha)^\epsilon}{2} \tag{36}$$

where $\epsilon \geq 1$ is a tunable parameter. A lower ϵ decreases the chance of artifacts but will have a greater filtering effect on the system. For low-speed changes, using ~ 1000 samples for removing one grid point, $\epsilon \approx 30$ ensures that $u_M^n \approx w_0^n$ at the time of removal and already suffices to remove artefacts.

5.3. Summary

Here, Section 5.2 is summarised and describes the final version of the proposed method.

The method subdivides a grid function with $N+1$ grid points into two grid functions u_l^n and w_l^n with $M+1$ and M_w+1 grid points respectively for a total of $N+2$ grid points. Knowing that $\lambda = 1\ \forall n$, Eq. (7), written for both grid functions, becomes

$$u_l^{n+1} = u_{l+1}^n + u_{l-1}^n - u_l^{n-1}, \tag{37a}$$

$$w_l^{n+1} = w_{l+1}^n + w_{l-1}^n - w_l^{n-1}, \tag{37b}$$

Due to the Dirichlet boundary condition in (9a) imposed on the outer boundaries of the system, u_0^n and $w_{M_w}^n$ are 0 at all times and are not included in the calculation. The range of calculation for Eq. (37a) and (37b) then become $l = [1, \ldots, M]$ and $l = [0, \ldots, M_w - 1]$ respectively.

The inner boundaries are calculated using

$$u_M^{n+1} = u_{M+1}^n + u_{M-1}^n - u_M^{n-1}, \tag{38a}$$

$$w_0^{n+1} = w_{-1}^n + w_2^n - w_0^{n-1}. \tag{38b}$$

where virtual grid points u_{M+1}^n and w_{-1}^n can be calculated using Eq. (31).

Then, when $\lfloor N^n \rfloor > \lfloor N^{n-1} \rfloor$ a point is added to $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ (or $\mathbf{w}^n$ and $\mathbf{w}^{n-1}$) using Eq. (32), and when $\lfloor N^n \rfloor < \lfloor N^{n-1} \rfloor$ a point is removed from the same vectors using Eq. (34). In order to prevent audible artefacts when increasing c (and thus decreasing N), a method in (35) is proposed to ensure that the inner boundaries have a similar displacement when one of them is removed.

Finally, using $\boldsymbol{\mathcal{U}}$ from Eq. (21) the total system can be compactly written in matrix form as

$$\boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}\boldsymbol{\mathcal{U}}^n - \boldsymbol{\mathcal{U}}^{n-1} \tag{39}$$

with $\lfloor N \rfloor \times \lfloor N \rfloor$ matrix

$$\mathbf{B} = \left[\begin{array}{ccc|ccc} \ddots & \ddots & & & & 0 \\ 1 & 0 & 1 & & & \\ & 1 & \frac{\alpha-1}{\alpha+1} - \beta & 1+\beta & -\frac{\alpha-1}{\alpha+1} & \\ \hline & -\frac{\alpha-1}{\alpha+1} & 1+\beta & \frac{\alpha-1}{\alpha+1} - \beta & 1 & \\ & & & 1 & 0 & 1 \\ 0 & & & & \ddots & \ddots \end{array}\right] \tag{40}$$

Notice that when α approaches 1, $\mathbf{B}$ will reduce to a matrix with ones on the diagonals next to the main diagonal and zeros elsewhere, which translates directly to the usual situation in Eq. (7) with $N = M + M_w + 1$.

5.4. Implementation

Algorithm 1 shows the order of calculation to implement the method presented in this paper. Important is to only retrieve a change in c at time index n before all other, so that u_l^n and w_l^n are calculated with the same α and β time index n for all l.

while *application is running* **do**
- Retrieve new c
- Calc. h (Eq. (13) with equality)
- Calc. $\lfloor N^n \rfloor$ (Eq. (4))
- **if** $\lfloor N^n \rfloor \neq \lfloor N^{n-1} \rfloor$ **then**
  - Add or remove point (Eq. (32) or (34))
  - Update M and M_w
- **end**
- Calc. α and β (Eqs. (26) and (36))
- Update $\mathbf{B}$ (Eq. (40))
- Calc. scheme (Eq. (39))
- Retrieve output
- Update states ($\boldsymbol{\mathcal{U}}^{n-1} = \boldsymbol{\mathcal{U}}^n, \boldsymbol{\mathcal{U}}^n = \boldsymbol{\mathcal{U}}^{n+1}$)
- Update N ($N^{n-1} = N^n$)
- Increment n

end

Algorithm 1: Pseudocode showing the order of calculations.

6. ANALYSIS AND RESULTS

This section shows the analysis of the system presented in the previous section and its behaviour.

6.1. Frequency

A modal analysis can be performed on system (39). For static parameters, the modal frequency of the p'th mode can be retrieved as

$$f_p = \frac{1}{2\pi k} \cos^{-1}\left(\frac{1}{2}\mathrm{eig}_p(\mathbf{B})\right), \tag{41}$$

and still holds for slow variations of c. As a test case, the wave speed of a system running at $f_s = 44100$ Hz is dynamically varied

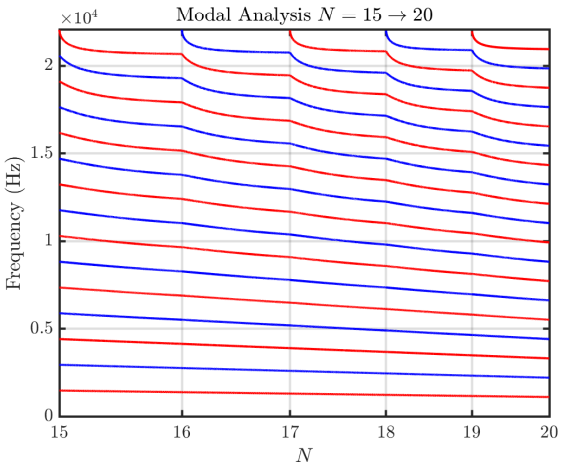

Figure 5: Modal analysis of system (39) using (41) running at $f_s = 44100$ Hz. The wave speed is reduced from $c = 2940$ ($N = 15$) to $c = 2205$ ($N = 20$).

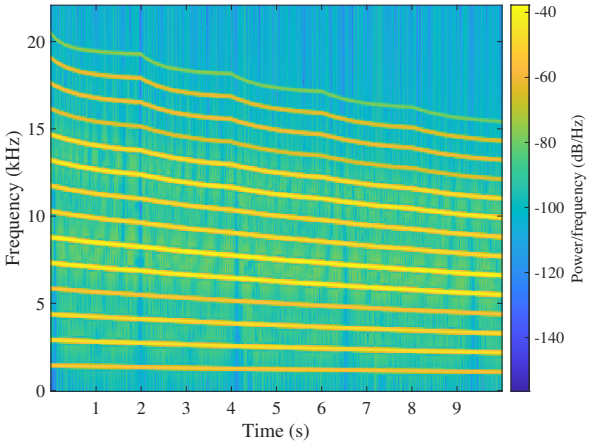

Figure 6: Output of the system excited with a narrow raised cosine at $n = 0$ retrieved at u_1^n. The sound output follows the same pattern as predicted by the analysis shown in Figure 5.

from $c = 2940$ ($N = 15$) to $c = 2205$ ($N = 20$), changing $\mathbf{B}$ and thus the modal frequencies over time. The displacement correction presented in Section. 5.2.4 is not included in this analysis ($\epsilon \gg 1$), but will be elaborated on in Section 6.2. The results of the analysis are shown in Figure 5. Figure 6 shows the resulting spectrogram of the system excited at $n = 0$ with a narrow raised cosine and the output retrieved at u_1^n.

In the following, the lowest mode generated by the analysis is referred to as f_1 and should ideally be equal to f_0 calculated using Eq. (3). To measure how much a mode deviates from a perfect harmonic, cents will be used according to $\varepsilon = 1200 \cdot \log_2(f_p/pf_0)$.

The first thing one can observe from Figure 5 is that the frequencies of the modes decrease as c decreases. The lower the mode, the more linear this decrease happens. Between $N = 15$ and $N = 16$, f_1 maximally deviates by -0.15 cents. In this same interval f_{15} maximally deviates by -67 cents. This deviation gets semi-linearly less as N increases.

Experiments with higher even-ordered Lagrange interpolators shows that these frequency deviations become smaller, but not by a substantial amount. The quadratic interpolator has thus been chosen for being simpler and more flexible while not being substantially worse than higher order interpolators.

Another observation from Figure 5 is that there are always $\lfloor N \rfloor$ modes present, corresponding to the number of moving points of the system. As can be seen from the spectrum in Figure 6 the highest mode is not excited. If the system is excited when N is not an integer, the highest mode will also be excited. When an implementation of the system using this method with integer N (static) is compared to a normal implementation of the 1D wave equation (shown in Section 3) with the same N, identical outputs are observed, even though the latter has $N - 1$ moving points.

Using the quadratic interpolation from (31), or any other even-ordered Lagrange interpolator for that matter, the modal behaviour of the system does not change based on location. Experiments done with odd-ordered Lagrange interpolators showed that better behaviour is observed when points are added / removed closer to the boundaries.

6.2. Displacement Correction

As mentioned in Section 5.2.4, the displacement correction works with low-speed parameter changes and indeed removes artefacts when grid points are removed. Analysis of a static system with non-integer N does, however, show that the correction has a low-pass comb-filtering effect on the system. The filtering is stronger for a lower values of ϵ and α. The location and amount of the notches is determined by the location of the inner boundaries along the grid, i.e., where along the grid the displacement correction is applied. The amount of notches is equal to $\min(M, M_w)$. For a dynamic system, the notches move around the frequency spectrum and overall yield a low-passing effect.

6.3. Limit on Speed of Change

The method presented in this paper can only add or remove a maximum one point per sample using Eqs. (32) and (34). The speed of decreasing f_0 according to (17) is thus limited by the following condition

$$|N^n - N^{n-1}| \leq 1. \tag{42}$$

Though this is the maximal limitation on speed, a much lower limitation needs to be placed to keep the system well-behaved. The usual stability and energy analyses performed on FDSs are not valid anymore in the time-varying case. Frozen coefficient analysis as in [1] could be applied here and hold for slowly varying coefficients, but is left for future work.

7. DISCUSSION

To decide whether the proposed method works satisfactory, the results presented in the previous section are compared to the method requirements listed in Section (5.1) (denoted by r# for short).

It can be argued that the frequency deviations of f_1 from f_0 are sufficiently small to say that the r1 is satisfied. As for r2, a fractional N has been introduced and smooth transitions are indeed observed from Figure 6, in the case when c is decreased and N is increased. When c is increased instead, transitions are not

smooth as artefacts are generated when grid points are removed. This problem is solved by the displacement correction presented in Section 5.2.4. However, the filtering effect that the displacement correction has on the system (mentioned in Section 6.2) is not ideal as it creates notches in the spectrum of the output sound, especially for a (relatively) static system. The least intrusive filtering happens when points are added and removed as close to the boundary as possible, i.e., when $M \vee M_w = 1$ where the notch only occurs in the higher end of the spectrum. As this is still not ideal, another method for reducing artefacts that less affects the frequency content of the system should be devised, if possible.

The modal analysis in Figure 5 shows that method generates $\lfloor N \rfloor$ rather than $\lfloor N \rfloor - 1$ modes as set by r3. However, the output does contain the correct number of modes as shown in Figure 6 due to the highest mode not being excited. This is a result of the rigid connection imposed on the inner boundaries, forcing them to have the same displacement and act as one point. As mentioned in the results, when the system is excited at a non-integer N, the highest mode is excited due to the fact that the inner boundaries can have different displacements in that case, but this is not an issue.

The latter part of r3, however, is not satisfied. The modes deviate from integer multiples of f_0, moreso for higher modes. Other interpolation techniques could be investigated to improve the behaviour and decrease this deviation.

Finally, the method only adds a few extra calculations for the inner boundaries so r4 is also easily satisfied.

Although the results bring forward some drawbacks of the method, such as frequency deviations, and low-passing effects, most of these happen affect the higher frequencies of the output. First of all, human frequency sensitivity becomes very limited above 3000 Hz [4] making high-frequency deviations much less important perceptually. Secondly, the physical systems one usually tries to model contain high-frequency losses, causing higher modes to usually not have very high amplitudes. Finally, N is usually much bigger in the systems that one tries to model, which decreases the deviations of higher frequencies from perfect harmonics.

8. CONCLUSIONS AND FUTURE WORK

This paper presents a method to change grid configurations of finite difference schemes to allow for dynamic parameters

The proposed method might not provide the exact solution to the problem, but does circumvent the need for upsampling and higher orders of computations necessary to approximate this solution. Even though interpolation needs to happen, the drawbacks of full-grid interpolation can be avoided by not 'listening' to the location where points are added but rather closer to the boundary. If one wants to listen to the center, the location where points are added or removed can easily be changed.

Though this method has only been presented using the 1D wave equation it could be applied to many other 1D FDSs.

Finally, an applications that will be investigated is the case of non-linear systems where parameters are modulated based on the state of the system.

Future work includes creating an adaptive version of the displacement correction that changes its effect depending on the speed at which the grid is changed.

Stability and energy analyses will have to be performed to test the limits on parameter / grid changes.

9. ACKNOWLEDGMENTS

Many thanks to the great number of anonymous reviewers!

10. REFERENCES

[1] John Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, California, 1989.

[2] Stefan Bilbao, *Numerical Sound Synthesis*, John Wiley & Sons, United Kingdom, 2009.

[3] Silvin Willemsen, Nikolaj Andersson, Stefania Serafin, and Stefan Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," 2019, pp. 275–280.

[4] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, Springer-Verlag, Berlin-Heidelberg, Germany, 1990.

[5] S. K. Mitra and J. F. Kaiser, Eds., *Handbook for Digital Signal Processing*, J. Wiley & Sons, New York, NY, USA, 1993.

[6] Simon Haykin, *Adaptive Filter Theory*, Prentice Hall, Englewood Cliffs, NJ, USA, second edition, 1991.

[7] X. Serra, *Musical Signal Processing*, chapter Musical Sound Modeling with Sinusoids plus Noise, pp. 91–122, G. D. Poli, A. Picialli, S. T. Pope, and C. Roads Eds. Swets & Zeitlinger, Lisse, Switzerland, 1996.

[8] James A. Moorer, "Audio in the new millennium," *J. Audio Eng. Soc.*, vol. 48, no. 5, pp. 490–498, May 2000.

[9] A. Nackaerts, B. De Moor, and R. Lauwereins, "Parameter estimation for dual-polarization plucked string models," in *Proc. Intl. Computer Music Conf.*, Havana, Cuba, Sept. 17-23, 2001, pp. 203–206.

[10] D. Arfib, "Different ways to write digital audio effects programs," in *Proc. Digital Audio Effects (DAFx-98)*, Barcelona, Spain, Nov. 19-21, 1998, pp. 188–91.

[11] A. Askenfelt, "Automatic notation of played music (status report)," Tech. Rep., STL-QPSR, Vol. 1, pp. 1–11, 1976.

[12] E. B. Egozy, "Deriving musical control features from a real-time timbre analysis of the clarinet," M.S. thesis, Massachusetts Institute of Technology, 1995.

[13] P. Dutilleux, *Vers la machine à sculpter le son, modification en temps-réel des caractéristiques fréquentielles et temporelles des sons*, Ph.D. thesis, University of Aix-Marseille II, 1991.

[14] K. Fitz and L. Haken, "Current Research in Real-time Sound Morphing," Available at http://www.cerlsoundgroup .org/RealTimeMorph/, accessed March 08, 2006.