DYNAMIC GRIDS FOR FINITE-DIFFERENCE SCHEMES IN MUSICAL INSTRUMENT SIMULATIONS

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 nonlinear. 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. Finite difference time domain methods (FDTD) lack the capability of handling smooth parameter changes while retaining optimal simulation quality and stability. In this paper we propose a method to smoothly add and subtract points from a FDTD system in real time so that the stability condition is always satisfied with equality. We present a case study to the simulation of a trombone.

1. INTRODUCTION

something before this Any musical instrument can be subdivided into an exciter and a resonator component [**?**]. Examples of exciter-resonator combinations are the bow and violin and the lips and trumpet. In nearly all instruments, the variables describing the exciter are 'modified' to play the instrument, i.e, the bow-velocity and pressure for the violin, and lip pressure and frequency for the trumpet. Naturally, the resonator is altered by fingering the strings of the violin or pressing valves on the trumpet to change its pitch. The variables describing the physics of the resonator itself are not changed though; the string-length stays the same and the total tube length remains unchanged, only the parts that resonate are shortened or lengthened.

In some cases, though, the parameters describing the resonator are modified. A prime example of this is the trombone, where the tube length is dynamically changed in order to generate different pitches. The slide whistle is another example in this category. Guitar strings are another category where the tension can be smoothly modulated during performance using the fretting finger, a whammy bar or even the tuning pegs directly (see [**?**]) to create smooth pitch glides. The same kind of tension modulation is used for the membranes of timpani or "hourglass drums" to change the pitch. It is these direct parameter modifications of the resonators that we are interested in to simulate.

Other than simulating existing instruments, one can simulate instruments that can be manipulated in physically impossible ways. Examples of this could be to dynamically change material properties such as density or stiffness, or even the geometry and size of the instrument.

Simulating musical instruments using physical modelling is a well-established field. Existing physical modelling techniques that have shown their ability to implement dynamic parameter changes of resonators include digital waveguides (DWGs) [**?**] and modal synthesis [**?**]. Literature using DWGs for time-varying parameters include Michon's BladeAxe [**?**, **?**] and Cook's ... SOTA on trombone / variable parameters with waveguides here!

Modal synthesis, though requiring some assumptions and simplifications for most systems, does allow for easy and smooth parameter changes. Resonant frequencies of the different modes can be dynamically modified without many issues as long as modal frequencies exceeding the stability condition are excluded as parameters are changed. Examples can be found in [**?**] and [**?**, **?**, **?**] ← maybe not all of these van Walstijn papers. Modal synthesis could thus be a good candidate for implementing the aforementioned dynamic parameters. In the case of the trombone, however, 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. 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. slightly different wording here → The challenge when trying to implement systems with dynamic parameters using FDTD methods, is that the number of grid points used in the simulation is closely tied to the parameters themselves through a stability condition. Adding and removing points from the grid is nontrivial and could cause audible artefacts or instability of the simulation. Full-grid interpolation [**?**, Ch. 5] could be a possible solution here, but extremely high sample rates are necessary to avoid audible artefacts and low-passing effects when changing between grid configurations.

This article proposes an efficient method to smoothly add and remove points from 1D finite-difference grids to allow for dynamic parameter changes. We are interested in 'slowly' varying parameters (sub-audio rate) so that modal and energetic analysis techniques are still meaningful to a degree. One can consider this work an introduction to implementing the trombone using FDTD methods.

Only if there is space → This paper is structured as follows:

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 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. (**??**) 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. (**??**) – 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. (**??**) still approximately holds. From Eq. (**??**), one can easily conclude that in terms of fundamental frequency, halving the length of Eq. (**??**) is identical to doubling the wave speed and vice versa. Looking at Eq. (**??**) 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. I feel like I need to talk about scaling here and how this confirms this statement ($f_0 = \gamma/2$ with $\gamma = c/L$ and scaled space $x \in [0, 1]$).

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. (**??**) can be discretised into points in space and time. The spatial variable can be discretised using $x_l = lh$ (read: x at spatial index 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 $k = 1/f_s$ (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. (**??**)

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

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

Substituting these definitions into Eq. (**??**) 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 (**??**) and solving for u_l^{n+1}, the only unknown, yields the following update equation

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

This update can then be implemented in software, such as `MATLAB` or `C++`. In Eq. (**??**),

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

is referred to as the Courant number and determines stability, as well as the quality and behaviour of the simulation. This will be described in detail in Sections **??** and **??**.

In the FDS described in Eq. (**??**), the boundary locations are at $l = 0$ and $l = N$. Substituting these locations into Eq. (**??**) 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 using the boundary conditions in Eq. (**??**). 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 (**??**) says that the dis-

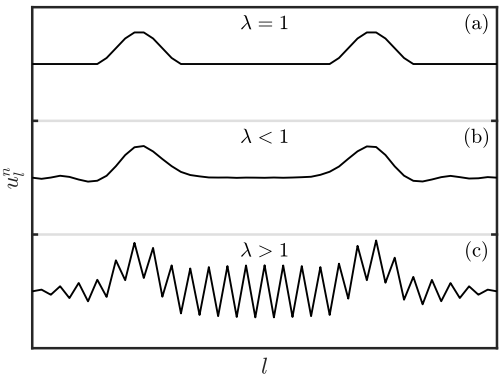

Figure 1: State u_l^n with $N = 50$ and $f_s = 44100$ visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (**??**) is not satisfied and the system is unstable.

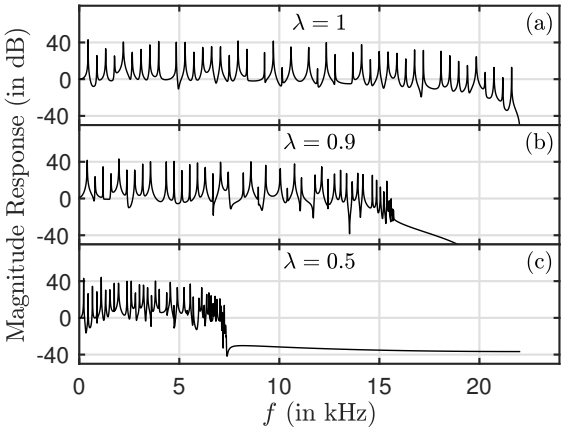

Figure 2: Bandwidths of the simulation output with $f_s = 44100$ Hz and (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$.

placement 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. (**??**) 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. (**??**) 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 retrieved by 'recording' the state u_l^n in Eq. (**??**) at $0 < l < N$ (when using fixed boundary conditions) and listening to that at the given sample rate f_s.

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. (**??**) it can be shown – using Von Neumann analysis [**?**] – 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 **??**) and if $\lambda = 1$, Eq. (**??**) provides an exact solution to Eq. (**??**) (see Figures **??**a and **??**b). If $\lambda > 1$ the system will become unstable (see Figure **??**c).

Recalling (**??**) can rewrite Eq. (**??**) in terms of grid spacing h to get

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

This shows that the CFL condition in (**??**) 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

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

In other words, condition (**??**) is first satisfied with equality and used to calculate integer N according to Eq. (**??**). Thereafter, h is recalculated based on N and used to calculate λ. The calculation of λ in Eq. (**??**) 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 (**??**) to not always be satisfied with equality and results in a reduced simulation quality described in the following section.

3.3. Simulation Quality

Choosing $\lambda < 1$ in Eq. (**??**) will decrease the simulation 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 **??**.

By analysing the scheme in Eq. (**??**), it can be shown that the maximum frequency produced by the system can be calculated using $f_{\text{max}} = f_s \sin^{-1}(\lambda)/\pi$ [**?**, Chap. 6]. Note that only a small deviation of λ from condition (**??**) already has a profound effect on the bandwidth of the output. Secondly, choosing $\lambda < 1$ causes numerical dispersion. See Figures **??**b and **??**. Harmonic partials get closer together at higher frequencies (i.e. get more inharmonic) as λ decreases, which is generally undesirable.

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 **??**, only c and L can be made dynamic, but as elaborated on in Section **??**, 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. (**??**) affecting the simulation quality and bandwidth. Secondly, and more importantly, a change in c could result in a change in N through Eq. (**??**). 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 [**?**, 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 [**?**]. This would avoid problems with stability, as decreasing c would continue to satisfy condition (**??**), but the simulation would end up with lower quality, exhibiting dispersive and bandlimiting effects as discussed in Section **??**. 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. (**??**) and increasing it beyond a certain value would render the system unstable.

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

$$f_0 = \frac{1}{2Nk}\,. \tag{16}$$

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 **??**. The following boundary conditions are then imposed:

$$u_0^n = w_{M_w}^n = 0, \quad \text{(Dirichlet)} \tag{17a}$$

$$\delta_{x\cdot} u_M^n = \delta_{x\cdot} w_0^n = 0. \quad \text{(Neumann)}, \tag{17b}$$

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

$$u_M^n = w_0^n, \quad \forall n. \tag{18}$$

To sum up, a grid function with $N = \lfloor L/h \rfloor$ (as per Eq. (**??**)) 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:

$$\begin{aligned} \mathbf{u}^n &= [u_1^n, \ldots, u_M^n]^T, \\ \mathbf{w}^n &= [w_0^n, \ldots, w_{M_w-1}^n]^T, \end{aligned} \tag{19}$$

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 (**??**) is then defined as

$$\boldsymbol{\mathcal{U}}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix}. \tag{20}$$

Note that the vectors in (**??**) and (**??**) 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

$$\begin{cases} \delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n + J_u(x_{u_M}^n) F \\ \delta_{tt} w_l^n = c^2 \delta_{xx} w_l^n - J_w(x_{w_0}^n) F \end{cases} \tag{21}$$

with spreading operators

$$J_u(x_i) = \begin{cases} \frac{1}{h}, & l = l_i = \lfloor x_i/h \rfloor \\ 0, & \text{otherwise} \end{cases} \quad \text{and}$$
$$J_w(x_i) = \begin{cases} \frac{1}{h}, & l = l_i = \lfloor x_i/h \rfloor - M \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

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 (**??**) at inner boundaries u_M^n and w_0^n, recalling the Neumann condition in (**??**) and the definition for the virtual grid points needed for this condition in Eq. (**??**) yields

$$\begin{cases} \delta_{tt} u_M^n = \frac{c^2}{h^2}(2u_{M-1}^n - 2u_M^n) + \frac{1}{h} F \\ \delta_{tt} w_0^n = \frac{c^2}{h^2}(2w_1^n - 2w_0^n) - \frac{1}{h} F. \end{cases} \tag{23}$$

Because of Eq. (**??**), 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 (**??**) 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 (**??**) after expansion of the second-time derivative yields the update of the inner boundaries

$$\begin{cases} u_M^{n+1} = 2u_M^n - u_M^{n-1} + \lambda^2(u_{M-1}^n - 2u_M^n + w_1^n) & \text{(24a)} \\ w_0^{n+1} = 2w_0^n - w_0^{n-1} + \lambda^2(u_{M-1}^n - 2w_0^n + w_1^n) & \text{(24b)} \end{cases}$$

which, (again, recalling Eq. (**??**)) are indeed equivalent expressions for the connected point and is necessary to satisfy the rigid connection. This system can be shown to exhibit behaviour identical to that of the original system. In (**??**), w_1^n in Eq. (**??**) acts as virtual grid point u_{M+1}^n and u_{M-1}^n in (**??**) as virtual grid point w_{-1}^n. This important fact is what the method relies on and will be extensively used in the following.

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. The locations of the outer boundaries $x_{u_0}^n$ and $x_{w_{M_w}}^n$ are fixed, i.e.

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

If the wave speed c is then decreased, and consequently the grid spacing h according to Eq. (**??**) (with equality), all other points move towards their respective outer boundary (see Figure **??**). Calculating h this way allows this method to always satisfy the CFL condition in Eq. (**??**) 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 boundary (in m) can be used to calculate the value of these virtual grid points (also see Figure **??** for reference). The interpolator I is a row-vector the same length as $\boldsymbol{u}^n$ (from Eq. (**??**)) its values depending on what interpolation order is used. In the following the distance between the inner boundaries normalised by h is defined as

$$\alpha = \alpha^n = \frac{x_{w_0}^n - x_{u_M}^n}{h}, \tag{25}$$

and for clarity, I and $\boldsymbol{u}^n$ are indexed by m.

The best behaviour is observed when I is even-ordered. Although higher orders yield slightly better behaviour this improvement is negligible and the lowest even-ordered, quadratic interpolation, already yields good results. The quadratic interpolator I_2 is defined as

$$I_2(x_i) = \begin{cases} -(\alpha - 1)/(\alpha + 1), & m = m_i - 1 \\ 1 & m = m_i \\ (\alpha - 1)/(\alpha + 1), & m = m_i + 1 \\ 0, & \text{otherwise} \end{cases} \tag{26a}$$

and its flipped version as

$$I_2^{\leftarrow}(x_i) = \begin{cases} (\alpha - 1)/(\alpha + 1), & m = m_i^{\leftarrow} - 1 \\ 1 & m = m_i^{\leftarrow} \\ -(\alpha - 1)/(\alpha + 1), & m = m_i^{\leftarrow} + 1 \\ 0, & \text{otherwise} \end{cases} \tag{26b}$$

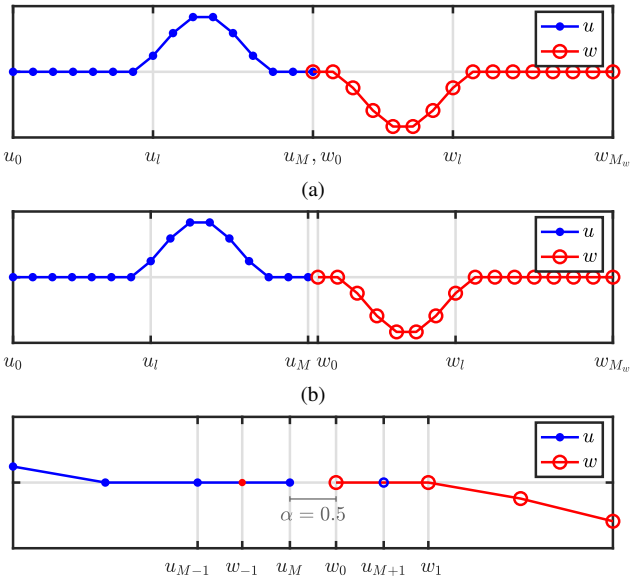

Figure 3: Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but the x and n are omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($N = 30$, $x_{u_M}^n = x_{w_0}^n$). (b) When c – 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 **??** 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. (**??**).

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 $\boldsymbol{\mathcal{U}}^n$. When applied to Eq. (**??**) this yields the definitions for the virtual grid points

$$u^n_{M+1} = I_2^{\leftarrow}(x^n_{u_{M+1}})\boldsymbol{\mathcal{U}}^n = \frac{\alpha-1}{\alpha+1}u^n_M + w^n_0 - \frac{\alpha-1}{\alpha+1}w^n_1 \tag{27a}$$

$$w^n_{-1} = I_2(x^n_{w_{-1}})\boldsymbol{\mathcal{U}}^n = -\frac{\alpha-1}{\alpha+1}u^n_{M-1} + u^n_M + \frac{\alpha-1}{\alpha+1}w^n_0. \tag{27b}$$

As will be shown in Section **??**, quadratic interpolation yields the expected fundamental frequency at all times. One can show that when N is an integer, and thus $\alpha = 0$, Eqs. (**??**) and (**??**) can be substituted as w^n_1 and u^n_{M-1} into Eqs. (**??**) and (**??**) respectively (as these acted as virtual grid points u^n_{M+1} and w^n_{-1}). Then recalling Eq. (**??**) it can be seen that the system reduces to (**??**) and exhibits the same exact behaviour as the normal case. Rewriting system (**??**) to include this yields

$$\begin{cases} \delta_{tt}u^n_l = c^2\delta_{xx}u^n_l + J_u(x^n_{u_M})\frac{2c^2}{h}\delta_{x.}u^n_M \\ \delta_{tt}w^n_l = c^2\delta_{xx}w^n_l - J_w(x^n_{w_0})\frac{2c^2}{h}\delta_{x.}w^n_0 \end{cases} \tag{28}$$

where the definitions for the virtual grid points needed after expansion of the last terms is found in Eqs. (**??**).

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. $x^n_{u_M} \leq x^n_{w_{-1}}$ and $x^n_{w_0} \geq x^n_{u_{M+1}}$ or $\alpha \geq 1$) and $\lfloor N^n \rfloor > \lfloor N^{n-1} \rfloor$, a point is added to the right boundary of u^n_l and the left boundary of w^n_l (for both time indices n and $n-1$) in an alternating fashion:

$$\begin{cases} \mathbf{u}^n = [\mathbf{u}^n, I_3\mathbf{v}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is odd,} \\ \mathbf{w}^n = [I_3^{\leftarrow}\mathbf{v}^n, \mathbf{w}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is even,} \end{cases} \tag{29}$$

where

$$\mathbf{v}^n = [u^n_{M-1}, u^n_M, w^n_0, w^n_1]^T,$$

and cubic Lagrangian interpolator

$$I_3 = \left[-\frac{\alpha'(\alpha'+1)}{(\alpha'+2)(\alpha'+3)} \quad \frac{2\alpha'}{\alpha'+2} \quad \frac{2}{\alpha'+2} \quad -\frac{2\alpha'}{(\alpha'+3)(\alpha'+2)} \right], \tag{30}$$

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

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

See Figure **??**.

Except in the case of extremely quick parameter variations, α' in Eq. (**??**) is expected to be close to zero, i.e., $x^n_{u_M} + h \approx x^n_{w_0}$, meaning that $I_3 \approx [0, 0, 1, 0]$.

Removing grid points happens when c, and consequently h, is increased and $x^n_{u_M} \geq x^n_{w_0}$ (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

$$\begin{cases} \mathbf{u}^n = [u^n_0, u^n_1 ..., u^n_{M-1}]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w^n_1, w^n_2 ..., w^n_{M_w}]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{31}$$

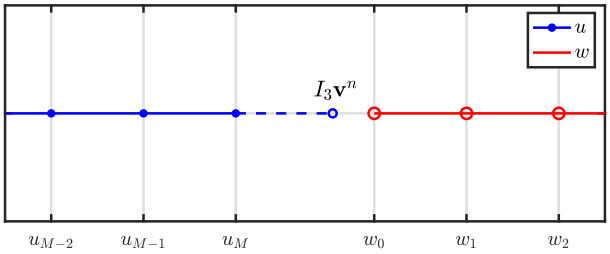

Figure 4: The moment when a point is added to $\mathbf{u}$ at location $x_{u_M} + h$ in Eq. (**??**). This figure shows an extreme case where this location is far from x_{w_0}, i.e., $\alpha' \not\approx 0$ in Eq. (**??**).

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^n_l and w^n_l 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 (**??**), 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^n_l with only one moving grid point, w^n_0) the method still works.

5.2.4. Displacement correction

A problem that arises from increasing h, is that it is possible that $u^n_M \not\approx w^n_0$ at the time when a grid point needs to be removed. As $x^n_{u_M} \approx x^n_{w_0}$ at the time of removal (except for extremely quick parameter variations), this violates the rigid connection in (**??**) 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 (**??**) is to 0. We thus extend system (**??**) regular case as

$$\begin{cases} \delta_{tt}u^n_l = c^2\delta_{xx}u^n_l + J_u(x^n_{u_M})\left(\frac{2c^2}{h}\delta_{x.}u^n_M + F_c\right) \\ \delta_{tt}w^n_l = c^2\delta_{xx}w^n_l - J_w(x^n_{w_0})\left(\frac{2c^2}{h}\delta_{x.}w^n_0 + F_c\right) \end{cases} \tag{32}$$

where J_u and J_w are as in (**??**). Furthermore, using centered temporal averaging and first-order difference operators

$$\mu_{t.}u^n_l = \frac{1}{2}\left(u^{n+1}_l + u^{n-1}_l\right) \tag{33}$$

$$\delta_{t.}u^n_l = \frac{1}{2k}\left(u^{n+1}_l - u^{n-1}_l\right) \tag{34}$$

the effect of the correction (in $\mathrm{m}^2/\mathrm{s}^2$) is defined as

$$F_c = \beta\left(\omega_0^2\mu_{t.}\eta^n + \sigma_0\delta_{t.}\eta^n\right). \tag{35}$$

with the difference in displacement between the inner boundaries

$$\eta^n \triangleq w^n_0 - u^n_M \tag{36}$$

(in m), (angular) frequency of the spring ω_0 (in s^{-1}) and damping coefficient σ_0 (in s^{-1}). Furthermore, $\beta = \beta(\alpha)$ scales the effect of the displacement correction and is defined as

$$\beta = \frac{1-\alpha}{\alpha+\epsilon}, \tag{37}$$

where $\epsilon \ll 1$ prevents a division by 0. Despite Eq. (**??**), it is possible to calculate the force explicitly (such as in [**?**] or [**?**]). Furthermore, it can be shown that this calculation is defined, even for $\alpha = \epsilon = 0$ which acts as a rigid connection such as Eq. (**??**).

5.3. Summary

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

The method subdivides a grid function with $\lfloor N \rfloor + 1$ 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 $\lfloor N \rfloor + 2$ grid points. Knowing that $\lambda = 1\ \forall n$, Eq. (**??**), written for both grid functions, becomes

$$u_l^{n+1} = u_{l+1}^n + u_{l-1}^n - u_l^{n-1}, \tag{38a}$$

$$w_l^{n+1} = w_{l+1}^n + w_{l-1}^n - w_l^{n-1}. \tag{38b}$$

Due to the Dirichlet boundary condition in (**??**) 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 ranges of calculation for Eq. (**??**) and (**??**) then become $l = [1, \dots, M]$ and $l = [0, \dots, M_w - 1]$ respectively.

The inner boundaries are calculated by expanding (**??**) (ignoring the displacement correction for now)

$$u_M^{n+1} = u_{M+1}^n + u_{M-1}^n - u_M^{n-1}, \tag{39a}$$

$$w_0^{n+1} = w_{-1}^n + w_1^n - w_0^{n-1}. \tag{39b}$$

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

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. (**??**), and when $\lfloor N^n \rfloor < \lfloor N^{n-1} \rfloor$ a point is removed from the same vectors using Eq. (**??**). In order to prevent audible artefacts when increasing c (and thus decreasing N), a method in (**??**) 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. (**??**) the total system can be compactly written in matrix form as

$$\mathbf{C}_+ \boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}\boldsymbol{\mathcal{U}}^n + \mathbf{C}_- \boldsymbol{\mathcal{U}}^{n-1} \tag{40}$$

with $\lfloor N \rfloor \times \lfloor N \rfloor$ matrices

$$\mathbf{B} = \left[\begin{array}{ccc|ccc} \ddots & \ddots & & & & 0 \\ 1 & 0 & 1 & & & \\ & 1 & \frac{\alpha-1}{\alpha+1} & 1 & -\frac{\alpha-1}{\alpha+1} & \\ \hline & -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1} & 1 & \\ & & & 1 & 0 & 1 \\ 0 & & & & \ddots & \ddots \end{array}\right] \tag{41}$$

containing the effect of the general method and

$$\mathbf{C}_\pm = \pm\left(\mathbf{I} - \frac{\beta k^2(\omega_0^2 \pm \sigma_0/k)}{2}\mathbf{J}\boldsymbol{\eta}\right) \tag{42}$$

containing the effect of the displacement correction where

$$\mathbf{J} = [\mathbf{0}_{M-1}, 1/h, -1/h, \mathbf{0}_{M_w-1}]^T \tag{43}$$

and

$$\boldsymbol{\eta} = [\mathbf{0}_{M-1}, -1, 1, \mathbf{0}_{M_w-1}] \tag{44}$$

are vectors of length $\lfloor N \rfloor$ where $\mathbf{0}_i$ is a zero-vector of length i. Finally $\mathbf{I}$ is the $\lfloor N \rfloor \times \lfloor N \rfloor$ identity matrix. Notice that as α approaches 1, $\mathbf{B}$ reduces to a matrix with ones on the diagonals next to the main diagonal and zeros elsewhere, which translates directly to the normal case in Eq. (**??**) with $N = M + M_w + 1$.

5.4. Implementation

Algorithm **??** 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. (**??**) with equality)
- Calc. $\lfloor N^n \rfloor$ (Eq. (**??**))
- **if** $\lfloor N^n \rfloor \neq \lfloor N^{n-1} \rfloor$ **then**
  - Add or remove point (Eq. (**??**) or (**??**))
  - Update M and M_w
- **end**
- Calc. α and β (Eqs. (**??**) and (**??**))
- Update $\mathbf{B}$ and $\mathbf{C}_\pm$ (Eqs. (**??**) and (**??**))
- Calc. scheme (Eq. (**??**))
- 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

Rewriting (**??**) in one-step form [**?**],

$$\begin{bmatrix} \boldsymbol{\mathcal{U}}^{n+1} \\ \boldsymbol{\mathcal{U}}^n \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_+^{-1}\mathbf{B} & \mathbf{C}_+^{-1}\mathbf{C}_- \\ \mathbf{I} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} \boldsymbol{\mathcal{U}}^n \\ \boldsymbol{\mathcal{U}}^{n-1} \end{bmatrix} \tag{45}$$

allows for a modal analysis to be performed. For static parameters, the complex frequency of the p'th mode can be retrieved as

$$s_p = \frac{1}{k}\ln\left(\mathrm{eig}_p(\mathbf{Q})\right), \tag{46}$$

where the frequency (in Hz) and damping (in s^{-1}) of the p'th mode can be retrieved using $f_p = \Im(s_p)/2\pi$ and $\sigma_p = \Re(s_p)$ respectively. Here, $\Im(\cdot)$ and $\Re(\cdot)$ denote the 'imaginary' and 'real part