Dynamic Grids for Finite-Difference Schemes: Changing Parameters of Musical Instrument Simulations in Real Time

Silvin Willemsen,[1, a] Stefania Serafin,[1] Stefan Bilbao,[2] and Michele Ducceschi[2]
[1] *Multisensory Experience Lab, CREATE, Aalborg University, Copenhagen, Denmark*
[2] *Acoustics and Audio Group, University of Edinburgh, Edinburgh, Scotland*

(Dated: 26 January 2021)

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 on the fly. Many modelling techniques exist of which 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. 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, which allows for dynamic parameter changes in physical models of musical instruments which are based on this technique. Furthermore, this technique allows the stability condition that the schemes using FDTD methods are based on, to always be satisfied with equality and thus have the highest simulation quality possible.

©*2021 Acoustical Society of America.* [https://doi.org(DOI number)]

[XYZ] Pages: 1–10

I. INTRODUCTION

Simulation of musical instruments through physical modelling is a well established field... 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. Examples of this could be to change material properties, or even the shape of the instrument on the fly. An example of a real-world instrument that requires these manipulations is the trombone, where tube-length needs to be dynamic in order to play the instrument.

Existing physical modelling techniques include mass-spring systems [REF] and digital waveguides [REF]. 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. 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. 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.

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.

This article is structured as follows: *NOTES:*

- Section 2: Continuous systems
- Section 3: Numerical methods
- Section 4: Dynamic grid

[a] sil@create.aau.dk

- Section 5: Results
- Section 6: Discussion
- Section 7: Conclusions and Future work

II. 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, even higher dimensional ones.

Consider a 1-dimensional system with length L (in m) described by state variable $u = u(x,t)$ defined over spatial domain $x \in [0, L]$ (in m) and time $t \geq 0$ (in s). The partial differential equation (PDE) of the 1D wave equation is then described as

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

parameterised by wave speed c (in m/s). As x in Eq. (1) is only defined over a finite region in space, boundary conditions need to be provided. Two possible 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.

A. 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, the fundamental frequency f_0 of the 1D wave equation can be calculated according to

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

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 system (1) in isolation, f_0 is the only behaviour of the system 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.

III. 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.

A. 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 $l \in [0, \ldots, N]$, grid spacing (distance between two consecutive grid points) h (in m) and total number of points $N+1$ (including the boundaries) where

$$N = \text{floor}(L/h), \tag{4}$$

describes the total number of intervals between the grid points. 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 state u at spatial index l and time index n". The total state at time index n is then denoted as vector $\mathbf{u}^n$ with size $N+1$.

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++`. Here, $\lambda = ck/h$ 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 III B and III C.

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

$$u_0^n = u_N^n = 0 \quad \text{(Dirichlet)} \tag{8a}$$

$$\delta_{x\cdot} u_0^n = \delta_{x\cdot} u_N^n = 0 \quad \text{(Neumann)} \tag{8b}$$

where

$$\frac{\partial u}{\partial x} \approx \delta_{x\cdot} u_l^n = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right) \tag{9}$$

is a second-order accurate approximation of the first-order spatial derivative. The Dirichlet condition in (8a) says that the states 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. (8b) can then be expanded to yield defnitions for these virtual grid points

$$u^n_{-1} = u^n_1 \quad \text{and} \quad u^n_{N+1} = u^n_{N-1}\,. \tag{10}$$

Now that the full system is described the output sound can be retrieved by following the state u^n_l in Eq. (7) at $0 < l < N$ (when using fixed boundary conditions , unless silence is your thing) and listening to that at the given sample rate f_s.

B. Stability

Discretising continuous equations using numerical methods place 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 stability analysis [REF] – that the system is stable if

$$\lambda \leq 1, \tag{11}$$

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 III C) and if $\lambda = 1$, Eq. (7) provides an exact solution to Eq. (1) [REF] (see Figures 1(A) and 1(B)). If $\lambda > 1$ the system will become unstable (see Figure 1(C)). One can rewrite Eq. (11) in terms of grid spacing h to get

$$h \geq ck. \tag{12}$$

This shows that the CFL condition in (11) puts a lower bound on the grid spacing, calculated from the sample rate and wave speed. Usually, the following steps are taken to calculate λ

$$h := ck, \quad N := \mathrm{floor}\left(\frac{L}{h}\right), \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}. \tag{13}$$

In other words, condition (12) 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. (13) can be compactly rewritten as

$$\lambda = \frac{ck}{L} \cdot \mathrm{floor}\left(\frac{L}{ck}\right). \tag{14}$$

The flooring operation causes the CFL condition in (11) to not always be satisfied with equality and results in a reduced simulation quality described in the following section.

C. 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 (Bilbao, 2009, Chap. 6)

$$f_\mathrm{max} = \frac{f_\mathrm{s}}{\pi} \sin^{-1}(\lambda), \tag{15}$$

shown in Figure 3. Note that only a small deviation of λ from condition (11) already has a profound effect on the bandwidth of the output.

Secondly, choosing $\lambda < 1$ causes numerical dispersion. See Figures 1(B) and 2. Harmonic partials get closer together at higher frequencies (i.e. get more inharmonic) as λ decreases, which is generally undesirable.

Apart from the recalculation of λ due to the flooring operation in Eq. (13), 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.

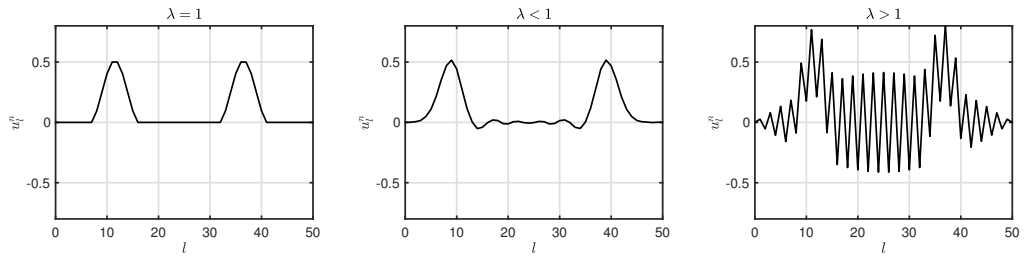
IV. DYNAMIC PARAMETERS

This section describes how parameters could be made dynamic using the state of the art and what this means for the simulation quality detailed in Section III C. To clarify, a *dynamic* parameter refers to one that is time-varying while the simulation is running.

Usually when simulating instruments, the parameters that describe the system are fixed for the entire simulation. For the 1D wave equation used in Section II, these are the wave speed c and time step k (calculated from the sample rate f_s) from which the grid spacing h and Courant number λ are calculated. On top of this, the length L can be changed, but as shown in Eq. (3), this can be directly translated to a change in c through the fundamental frequency f_0. As f_s is rarely changed due to all sorts of issues, the sole parameter that could be interesting to make time-varying is c.

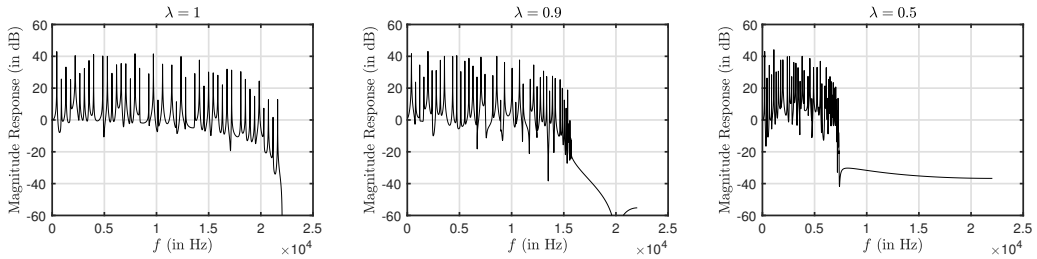
As c changes, several things need to be taken into account. First of all, a change in c causes a change in λ according to Eq. (14) affecting the simulation quality and bandwidth. Secondly, and more importantly, a change in c could result in a change in N through Eq. (4). As N directly relates to the 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.

A solution to the latter is to set and fix N and tune c away from the stability condition by decreasing it, such as done in [REF]. This would avoid problems with stability,



(A) (B) (C)

FIG. 1. State u_l^n with $N = 50$ and $f_s = 44100$ visualised. (A) If $\lambda = 1$, the solution is exact. (B) If $\lambda < 1$ dispersive behaviour shows. (C) If $\lambda > 1$ the CFL condition in Eq. (11) is not satisfied and the system "explodes".



(A) (B) (C)

FIG. 2. Bandwidths of the simulation output with $f_s = 44100$ Hz and (A) $\lambda = 1$, (B) $\lambda = 0.9$ and (C) $\lambda = 0.5$.

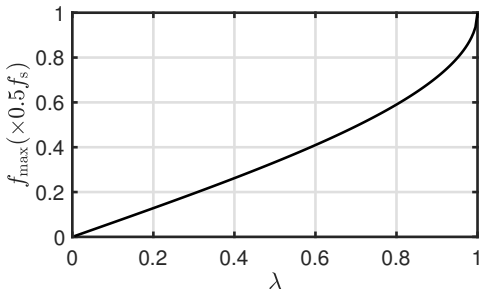


FIG. 3. The effect of the Courant number λ on the output bandwidth. This figure should probably be much more compact, if not removed altogether.

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 III C. 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. (11) 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 (i.e., non-integer). This will remove the flooring operation in Eq. (4) and Eq. (14), and will consequently satisfy the CFL condition in (11) with equality at all times. Eq. (3) can then be rewritten in terms of N by substituting Eq. (4) into Eq. (3) (using Eq. (12) 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.

As this still leaves the question of where and how to add and remove points to and from the grid a method needs to be devised to dynamically and smoothly change the number of grid points. This paper proposes a method to do this and is described in the following section.

V. 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 dynamic grids would be a good solution to undesirable behaviour such as a decrease in bandwidth and increase in numerical dispersion discussed in Section III C.

In this section, the requirements of a method that dynamically changes the grid will be explained. After this the iterations done over the course of this project will briefly be described, the details of each can be found in Appendix A. Finally, the proposed method will be described in detail and summarised in the end.

A. 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 c ($f_0 \propto c$),
2. allow for a fractional N to smoothly transition between different numbers of grid points so that no artefacts (auditory clicks) are present in the output sound,

3. generate an output containing harmonic partials – or modes – which are integer multiples of the fundamental ($f_p = f_0 p$ with integer p),

4. generate an output with floor(N) − 1 modes corresponding to the number of moving points of the system ($p = [1, \ldots, \text{floor}(N) - 1]$),

5. work in real time.

B. Iterations

One method that could be used to go from one grid configuration to the next is full-grid interpolation as described in (Bilbao, 2009, Chap. 5). However, this method essentially has a lowpassing effect on the system state and can cause 'clicks' 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 method is to add and remove points at the boundary using an interpolated boundary condition, the possibility of which has been briefly mentioned in (Bilbao, 2009, p. 145). If the boundary is fixed through Eq. (2a), the state at this location will always be 0 and potentially allows for smooth entry and exit of grid points. This method can be seen analogous to tuning a guitar string where string-material enters and leaves the playable part of the string at the nut, the boundary. The interpolated nature of the boundary allows for a "fractional" number of points, removing the flooring operation in Eq. (14) and always satisfying the CFL condition with equality. This has the added feature that L/h in Eq. (4) is an integer and the flooring operation can be ignored. Substituting Eq. (4) into Eq. (3) (using Eq. (12) satisfied with equality) yields

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

which shows that if $\lambda = 1$, N, which is now not necessarily an integer, solely decides the fundamental frequency of the simulation. Although informal testing shows that adding points to the grid can happen smoothly, removing points smoothly is more challenging. This is due to the fact that the grid point at the boundary will be moving right before it is removed and its displacement needs to (somehow) smoothly be reduced to 0 to satisfy the fixed boundary condition in Eq. (2a).

C. Proposed Method

This section introduces the proposed method of dynamically and smoothly changing the grid to account for dynamic parameter changes. To avoid the issues of adding and removing points at the boundary due to boundary conditions, they can be added or removed along the grid instead. For the sake of simplicity in explanation, the location is chosen to be the center of the system. At the end of this section, the location exhibiting the best behaviour will be shown. In the following, the location of a grid point (in m from the left boundary) i (such as $i = u_0$) is denoted by x_i.

1. System Setup

Consider a grid function, u_l^n with integer $M_u = \text{ceil}(0.5L/ck)$ (or simply M below for brevity) and w_l^n with integer $M_w = \text{floor}(0.5L/ck)$ points, i.e., half the number of points allowed by the stability condition, plus one for overlap (see Figure 4(A)). In the following, state (column) vectors $\mathbf{u}^n = [u_0^n, u_1^n, \ldots, u_M^n]^T$ and $\mathbf{w}^n = [w_0^n, w_1^n, \ldots, w_{M_w}^n]^T$ (with T denoting the transpose operation) and have $M+1$ and M_w+1 points respectively (including the boundary). The vector concatenating these is then defined as

$$\boldsymbol{\mathcal{U}}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix} \tag{18}$$

The following boundary conditions are then imposed:

$$u_0^n = w_{M_w}^n = 0, \quad \text{(Dirichlet)} \tag{19a}$$

$$\delta_{x\cdot} u_M^n = \delta_{x\cdot} w_0^n = 0. \quad \text{(Neumann)}, \tag{19b}$$

i.e., the outer boundaries are fixed and the inner boundaries are free. The systems can then be connected at the inner boundaries (u_M and w_0) using a rigid connection, i.e. (only valid if $x_{u_M} = x_{w_0}$),

$$u_M^n = w_0^n, \quad \forall n. \tag{20}$$

Essentially, the complete system is divided into two separate systems connected at the inner boundary. The system will now be

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

with spreading operator

$$J(x_i) = \begin{cases} \frac{1}{h}, & l = l_i \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

and the effect of the connection ("connection force", but not really as it isn't in N) F (in m^2/s^2). Expanding the spatial operators in system (21) at inner boundaries u_M^n and w_0^n, recalling the conditions in (19) and the definition for the virtual grid points needed for the Neumann condition in Eq. (10) 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. (20), it is also true that $\delta_{tt} u_M^n = \delta_{tt} w_0^n$, and F can be calculated by setting the right side of the equations in (21) equal to each other:

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

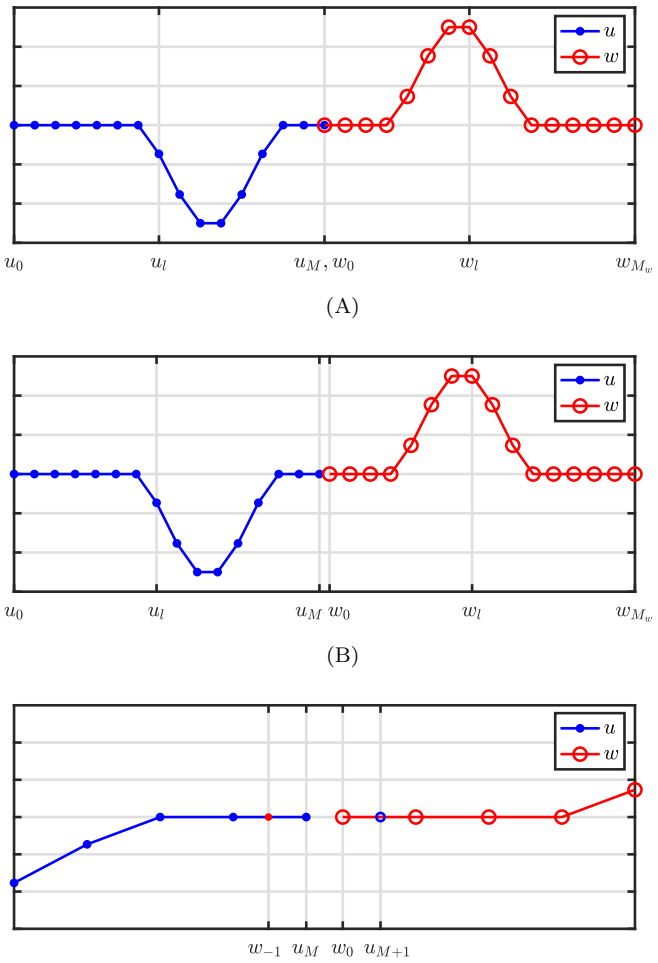


FIG. 4. Illustration of the proposed method. (A) Locations of the states of two (1D wave) systems connected at the inner boundary ($N = 30$, $x_{u_M} = x_{w_0}$). (B) When c – and consequently h – are decreased and the positions of the grid points change ($N = 30.5$, $x_{u_M} \neq x_{w_0}$). (C) Figure 4(B) zoomed-in around x_{u_M} and x_{w_0}. The states at the inner boundaries u_M and w_0 are shown together with virtual grid points u_{M+1} and w_{-1}. In all figures, the x-axis shows the location (in m) of the respective grid points (fx. x_{u_l}), but the x is omitted for brevity.

Substituting this into system (23) after expansion of the second-time derivative yields

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

which, (again, recalling Eq. (20)) are indeed equivalent expressions for the connected point. Here, w_1^n in the first expression acts as virtual grid point u_{M+1}^n and u_{M-1}^n as virtual grid point w_{-1}^n, essentially connecting the two systems using the state of one in the update of the other.

2. Changing the Grid

Section V C 1 describes the case in which the stability condition is satisfied with equality, i.e, when $1/ck$ is an integer and $x_{u_M} = x_{w_0}$ (read as: the locations of grid points u_M and w_0 are identical). The locations of the outer boundaries x_{u_0} and $x_{w_{M_w}}$ are fixed, i.e.

$$x_{u_0}^n = x_{u_0}^0 \quad \text{and} \quad x_{w_{M_w}}^n = x_{w_{M_w}}^0 \quad \forall n. \tag{26}$$

If the wave speed c is then decreased, and consequently the grid spacing h according to $h = ck$, all other points move towards their respective outer boundary (see Figure 4(B)). Calculating h this way allows this method to always satisfy the CFL condition in Eq. (11) with equality, as is the case with the previous iteration described in V B.

As mentioned above, the state of $\mathbf{w}$ can be used to calculate the virtual grid point needed at the right boundary of $\mathbf{u}$ and vice versa. If $x_{u_{M+1}}^n \neq x_{w_1}^n$ (and thus $x_{w_{-1}}^n \neq x_{u_{M-1}}^n$) ← I think this is clearer than simply saying $x_{w_0} \neq x_{u_M}$ as these are used in the Eqs. in (27) below an interpolator $I(x_i)$ at location x_i (in m) can be used to calculate the value of this virtual grid point

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

where I^* interpolates right to left (uses $\alpha = (1 - \alpha)$). If linear interpolator I_1 is used, the Eqs. in (27) can easily be calculated from known values according to

$$u_{M+1}^n = I_1^*(x_{u_{M+1}}^n)\mathbf{w}^n = (1-\alpha)w_1^n + \alpha w_0^n \tag{28a}$$
$$w_{-1}^n = I_1(x_{w_{-1}}^n)\mathbf{u}^n = (1-\alpha)u_{M-1}^n + \alpha u_M^n \tag{28b}$$

where

$$\alpha = \alpha^n = \frac{x_{w_0}^n - x_{u_M}^n}{h}. \tag{29}$$

Also see Figure 4(C). 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 wavespeed, opposite of what is expected.

One could extend the range of interpolation by one point to each side, and using a cubic interpolator. Though this would require w_{-1} to calculate u_{M+1} 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 to modes higher than $f_s/2$.

Much better behaviour is observed when points of both u and w are used, i.e., using u_M to calculate u_{M+1} and w_0 for w_{-1}. Now, the grid points used in the interpolation are not equidistant and a custom interpolator needs to be created. The lowest order interpolator that

can be used here is the quadratic Lagrangian interpolator I_2 and when applied to Eq. (18) yields

$$\begin{aligned} u^n_{M+1} &= I^*_2(x^n_{u_{M+1}})\boldsymbol{\mathcal{U}}^n \\ &= \frac{\alpha-1}{\alpha+1}u^n_M + w^n_0 - \frac{\alpha-1}{\alpha+1}w^n_1 \end{aligned} \tag{30a}$$

$$\begin{aligned} w^n_{-1} &= I_2(x^n_{w_{-1}})\boldsymbol{\mathcal{U}}^n \\ &= -\frac{\alpha-1}{\alpha+1}u^n_{M-1} + u^n_M + \frac{\alpha-1}{\alpha+1}w^n_0. \end{aligned} \tag{30b}$$

As will be shown in Section VI, quadratic interpolation yields the expected fundamental frequency at all times. It can be seen that when N is an integer, and thus $\alpha = 0$, the system is reduced to (25) (recalling Eq. (20)).

3. Adding and removing Grid Points

When c, and consequently h, is decreased and the boundary points surpass the virtual points (i.e. $x^n_{u_M} \leq x^n_{w_{-1}}$ and $x^n_{u_{M+1}} \leq x^n_{w_0}$) and $M^n > M^{n-1}$ (calculated using (33)), a point approximating u^n_{M+1} is added at the right boundary of $\mathbf{u}^n$ (for both time indices n and $n-1$)

$$\mathbf{u}^n = \begin{bmatrix} \mathbf{u}^n \\ I'_3(x^n_{u_{M+1}})\mathbf{v}^n \end{bmatrix}, \tag{31}$$

where

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

and

$$I'_3 = \left[-\tfrac{\alpha'(\alpha'+1)}{(\alpha'+2)(\alpha'+3)} \quad \tfrac{2\alpha'}{\alpha'+2} \quad \tfrac{2}{\alpha'+2} \quad -\tfrac{2\alpha'}{(\alpha'+3)(\alpha'+2)}\right], \tag{32}$$

(created using 4-point Lagrange interpolation) with

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

See Figure 5.

The following applies to odd-ordered Lagrange interpolators → The location at where points are added and removed greatly influences the behaviour of the system, especially in the higher frequencies (see Section VI). The best behaviour is obtained when the location is as close to a boundary as possible. Using a fractional N, consider grid functions u^n_l and w^n_l as described before, but now with

$$M = \text{floor}(N) - 1 \quad \text{and} \quad M_w = 1. \tag{33}$$

This results in grid function w^n_l only having one moving point w^n_0, the boundary at w^n_1 and virtual grid point w^n_2 (see Figure 5). Condition (19a) can be extended at the right boundary to be simply supported

$$w^n_{M_w} = \delta_{xx} w^n_{M_w} = 0, \quad \text{(simply supported)} \tag{34}$$

which after expansion and knowing $M_w = 1$ can be solved to (similarly to Eq. (10))

$$w_1 = 0 \quad \text{and} \quad w^n_2 = -w^n_0. \tag{35}$$

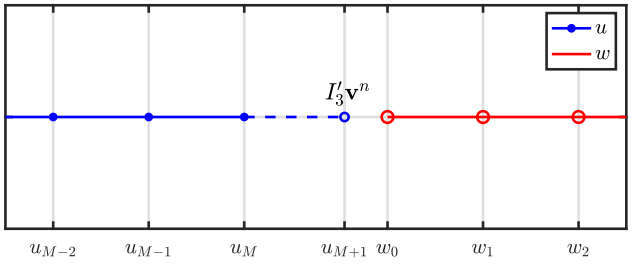


FIG. 5. The moment when a point is added to $\mathbf{u}$ at location $x_{u_M} + h$ in Eq. (31). This figure shows an extreme case where this location is far from x_{w_0}, i.e., $\alpha' \not\approx 0$ in Eq. (32).

This simplifies Eq. (30a) to

$$u^n_{M+1} = (\gamma_\text{I} - \alpha_\text{I})w^n_0 + \delta_\text{I} w^n_{-1}, \tag{36}$$

and can be inserted in Eq. (??).

As α' in Eq. (32) is expected to be close to zero maybe explain that at low parameter variations this happens?, the third term in $\mathbf{v}^n$ is expected to have the biggest contribution to the newly added point. This makes sense by looking at Figure 4(C), as exactly when boundary point w^n_0 surpasses virtual point u^n_{M+1}, these are going to be close to overlapping.

Removing grid points happens when c, and consequently h, is increased and the boundary points surpass the virtual points (i.e. $x_{u_M} \geq x_{w_{-1}}$ and $x_{u_{M+1}} \geq x_{w_0}$) and $N^n < N^{n-1}$. Compared to adding grid points, removing them is slightly easier where points are simply removed from the end of $\mathbf{u}$

$$\mathbf{u}^n = [u^n_0, u^n_1 ..., u^n_{M-1}]^T. \tag{37}$$

D. Summary (just for clarity now, but maybe actually nice to include)

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

The complete system consists of two grid functions u^n_l and w^n_l of size of size M and M_w defined by (33). Knowing that $\lambda = 1$ $\forall n$, Eq. (7), written for both grid functions, becomes

$$u^{n+1}_l = u^n_{l+1} + u^n_{l-1} - u^{n-1}_l, \tag{38}$$

$$w^{n+1}_l = w^n_{l+1} + w^n_{l-1} - w^{n-1}_l, \tag{39}$$

Due to the Dirichlet boundary condition in (8a) imposed on the outer boundaries of the system, u^n_0 and $w^n_{M_w}$ are 0 at all times and are not included in the calculation. The range of calculation for Eq. (38) then becomes $l = [1, \ldots, M]$ and because $M_w = 1$, the range of calculation for Eq. (39) is the single point $l = 0$.

The inner boundaries are calculated using

$$u^{n+1}_M = u^n_{M+1} + u^n_{M-1} - u^{n-1}_M, \tag{40}$$

$$w^{n+1}_0 = w^n_{-1} - w^{n-1}_0. \tag{41}$$

where virtual grid points u^n_{M+1} and w^n_{-1} can be calculated using Eq. (??) with Eq. (36) instead of Eq. (30a) and yields

$$\begin{bmatrix} u^n_{M+1} \\ w^n_{-1} \end{bmatrix} = \boldsymbol{\mathcal{A}} \begin{bmatrix} (\gamma_\mathrm{I} - \alpha_\mathrm{I}) w^n_0 \\ \alpha_\mathrm{I} u^n_{M-2} + \beta_\mathrm{I} u^n_{M-1} + \gamma_\mathrm{I} u^n_M \end{bmatrix}, \tag{42}$$

where definitions for α_I, β_I, γ_I (, δ_I) and $\boldsymbol{\mathcal{A}}$ can be found in (??) and (??).

Defining state vector $\boldsymbol{\mathcal{U}}^n$ as a concatenation of $\mathbf{u}^n$ and $\mathbf{w}^n$ excluding the boundaries

$$\boldsymbol{\mathcal{U}}^n = [u^n_1, u^n_2, \dots u^n_M, w^n_0]^T, \tag{43}$$

the total system can then be written in matrix form as

$$\boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}\boldsymbol{\mathcal{U}}^n - \boldsymbol{\mathcal{U}}^{n-1} \tag{44}$$

with $N \times N$ matrix

$$\mathbf{B} = \left[\begin{array}{ccccccc|c} 0 & 1 & & & & & & \\ 1 & 0 & 1 & & & \mathbf{0} & & \mathbf{0} \\ & \ddots & \ddots & \ddots & & & & \\ & & 1 & 0 & 1 & & & \\ \mathbf{0} & & & 1 & 0 & 1 & & \\ & & & \boldsymbol{\mathcal{I}}_{1,1} & \boldsymbol{\mathcal{I}}_{1,2}+1 & \boldsymbol{\mathcal{I}}_{1,3} & \boldsymbol{\mathcal{I}}_{1,4} \\ \hline \mathbf{0} & & & \boldsymbol{\mathcal{I}}_{2,1} & \boldsymbol{\mathcal{I}}_{2,2} & \boldsymbol{\mathcal{I}}_{2,3} & \boldsymbol{\mathcal{I}}_{2,4} \end{array}\right] \tag{45}$$

and interpolation matrix

$$\boldsymbol{\mathcal{I}} = \begin{bmatrix} \boldsymbol{\mathcal{A}}_{1,2}\alpha_\mathrm{I} & \boldsymbol{\mathcal{A}}_{1,2}\beta_\mathrm{I} & \boldsymbol{\mathcal{A}}_{1,2}\gamma_\mathrm{I} & \boldsymbol{\mathcal{A}}_{1,1}(\gamma_\mathrm{I} - \alpha_\mathrm{I}) \\ \boldsymbol{\mathcal{A}}_{2,2}\alpha_\mathrm{I} & \boldsymbol{\mathcal{A}}_{2,2}\beta_\mathrm{I} & \boldsymbol{\mathcal{A}}_{2,2}\gamma_\mathrm{I} & \boldsymbol{\mathcal{A}}_{2,1}(\gamma_\mathrm{I} - \alpha_\mathrm{I}) \end{bmatrix}.$$

One can see that if $\alpha \lesssim 1$ in (??), then $\gamma_\mathrm{I} \approx 1$ and $\alpha_\mathrm{I} \approx \beta_\mathrm{I} \approx \delta_\mathrm{I} \approx 0$. This will reduce $\mathbf{B}$ to a matrix with ones on the diagonals next to the main diagonal and zeros elsewhere.

Finally, when $M^n > M^{n-1}$ a point is added from $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ using Eq. (31), and when $M^n < M^{n-1}$ a point is removed from the same vectors using Eq. (37).

VI. RESULTS

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

A. Static

In order to determine whether the proposed method yields an output with the correct frequency content, a spectrum is taken of the system's output. The system with $N = 15.5$ (so $\alpha = 0.5$ in (??)) at a sample rate of $f_\mathrm{s} = 44100$ Hz is compared to the same system with $f_\mathrm{s} = 88200$ Hz resulting in $N = 31$ ($\alpha = 0$) for the same f_0 according to Eq. (17). See Figure 6.

Discussion: As can be seen from the figure, the first 8 harmonic partials of both systems line up exactly ($< 0.2\%$ of their respetcive frequency, $< 1.3\%$ of the fundamental f_0).

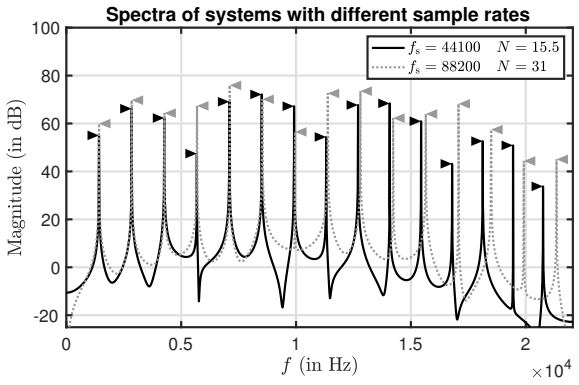


FIG. 6. A system with $N = 15.5$ at $f_\mathrm{s} = 44100$ compared to a system with $N = 31$ at $f_\mathrm{s} = 88200$ which are supposed to have the same fundamental frequency f_0 according to Eq. (17).

B. Dynamic

A modal analysis can be performed on system (44). The modal frequency of the p'th mode can be retrieved as

$$f_p = \frac{1}{2\pi k} \cos^{-1}\left(\frac{1}{2}\mathrm{eig}_p(\mathbf{B})\right). \tag{46}$$

As a test case, the wave speed is dynamically varied from $c = 2940$ ($N = 15$) to $c = 2205$ ($N = 20$), changing $\mathbf{B}$ and thus the modal frequencies over time, and the results of the analysis are shown in Figure 7. Figure 8 shows the resulting spectrogram of the system excited at $n = 0$ with a narrow raised cosine and the output retrieved at u^n_1.

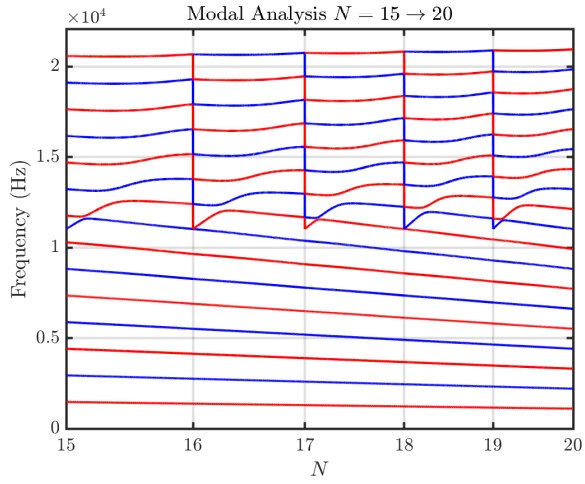


FIG. 7. Modal analysis

From Figure 7 one can observe several things.

- At every integer N there are N modes, $N - 1$ of which are integer multiples of each other cor-

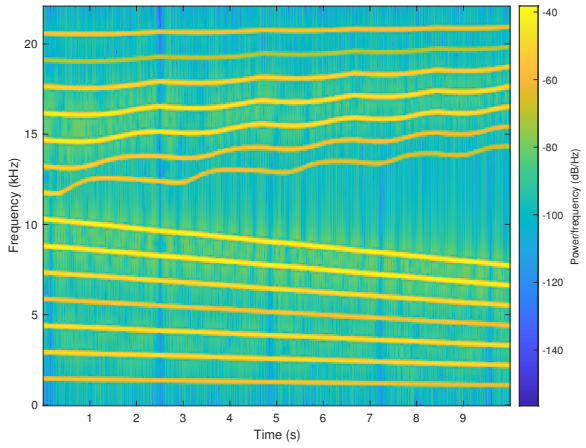


FIG. 8. System output

responding to the amount of moving points, plus one extra due to the overlap.

As can be seen from the spectrum in Figure 8

C. Limit on speed of change

The method presented in this paper can only add one point per sample using Eq. (31). The speed of decreasing f_0 by increasing the number of points is thus limited by the following condition

$$N^n - N^{n-1} \leq 1 \tag{47}$$

Notes: In the case that $\alpha = 0$ in Eq. (44) (and thus $x_{u_M} = x_{w_0}$), one could go up to adding two points at a time by adding u^n_{M+1} according to Eq. (31) and adding an extra point u^n_{M+2} using (42) (in this case $u^n_{M+2} = w^n_0$). This would, however, not work if $\alpha \neq 0$, as a point added to $\mathbf{u}^n$ with $x_i < w_0$ needs interpolator (32) and α' would be greater than 1. As Eq. (32) only works if $0 \leq \alpha' \leq 1$, it is best to abide condition (47) to be safe.

No speed limit if

$$\text{floor}(N^n) = \text{floor}(N^{n-1}) \Rightarrow M^n = M^{n-1}, \tag{48}$$

i.e., if no points are added or removed from the system.

VII. DISCUSSION

Applications could be non-linear systems where parameters are modulated based on the state of the system.

The proposed method does 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.

frequency domain, the locations of the partials comparing the discrete 1D wave with $N = 30.5$ and $f_s = 44100$ (interpolation needs to happen) with $N = 61$ and f

If the amount that a parameter changes within a small enough period of time (give values here, hopefully referring back to the results) the fact that points are added at a specific location (rather than distributed over the grid) will not matter as...

Notes (warning, very pretentious, and probably untrue..): The aforementioned drawbacks of the method are eclipsed by the advantages it brings in terms of dynamic parameter changes and quality improvements.

APPENDIX A:

In this appendix, some iterations done over the course of this project will be shown in more detail. In the following, the 1D wave equation with a wave speed of $c = 1470$ m/s, a length of $L = 1$ m, Dirichlet boundary conditions and a sample rate of $f_s = 44100$ Hz is considered, and – through Eq. (14) – satisfies the CFL condition with equality. These values result in $N = 30$, or a grid of 31 points including the boundaries. Then, the wave speed is decreased to $c \approx 1422.6$ m/s, i.e., the wave speed that results in $N = 31$ and satisfies the stability condition with equality again.

1. Full-Grid Interpolation

One way to go from one grid to another is performing a full-grid interpolation (Bilbao, 2009, Chap. 5). If the number of points changes according to Eq. (13), i.e., if $N^n \neq N^{n-1}$ the full state of the system $(u^n_l, u^{n-1}_l \; \forall l)$ can be interpolated to the new state. See Figure 9.

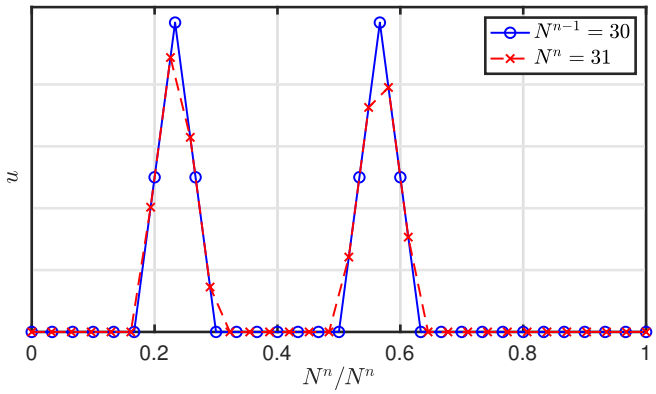


FIG. 9. Upsampling u (with an arbitrary state) using (linear) full-grid interpolation with $N^{n-1} = 30$ and $N^n = 31$. The horizontal axis is normalised with respect to N^n.

An issue that arises using this method is that the Courant number λ will slightly deviate from the CFL condition as c changes. Using Eq. (14) with L/ck

approaching 31 (from below), the minimum value of $\lambda \approx 30/31 \approx 0.9677$. This, employing Eq. (15), has a maximum frequency output of $f_{\max} \approx 18,475$ Hz. The Courant number will deviate more for higher values of c and thus lower values for N – for instance, if N approaches 11 (from below), $\lambda \approx 10/11 \approx 0.9091$ and $f_{\max} \approx 16,018$ Hz.

Another problem with full-grid interpolation, is that it has a low-passing effect on the system state, and thus on the output sound. Furthermore, this state-interpolation causes artefacts or 'clicks' in the output sound as the method causes sudden variations in the states.

All the aforementioned issues could be solved by using a (much) higher sample rate and thus more grid points, but this would render this method impossible to work in real time.

2. Adding and removing Points at the Boundary

To solve the issues exhibited by a full-grid interpolation, points can be added and removed at a single location and leave most points unaffected by the parameter changes. A good candidate for a location to do this is at a fixed (Dirichlet) boundary. The state u at this location is always 0 so points can be added smoothly.

As c decreases, h can be calculated according to Eq. (13) and decreases as well.

This has a physical analogy with tuning a guitar string. Material enters and exits the neck (playable part of the string) at the nut, which in discrete time means grid points appearing and disappearing at one boundary.

To yield smooth changes between grid configurations, an interpolated boundary has been developed, the possibility of which has been briefly mentioned in (Bilbao, 2009, p. 145). The Dirichlet condition in Eq. (2a) can be extended to be the simply supported boundary condition:

$$u(x,t) = \frac{\partial^2}{\partial x^2} u(x,t) = 0 \quad \text{where} \quad x = 0, L, \tag{A1}$$

or, when discretised,

$$u_l^n = \delta_{xx} u_l^n = 0, \quad \text{where} \quad l = 0, N. \tag{A2}$$

This means that on top of that the state of the boundary should be 0, the curvature around it should also be 0. One can again solve for the virtual grid points at the boundary locations, yielding

$$u_{-1}^n = -u_1^n \quad \text{and} \quad u_{N+1}^n = -u_{N-1}^n. \tag{A3}$$

This is visualised in Figure 10.

If the flooring operation in Eq. (4) is removed this introduces a fractional number of grid points.

The by-product of using a fractional N this is that the CFL condition in (11) can now always be satisfied with equality no matter what the wave speed is.

An issue with this method is that removing points is much harder than adding.

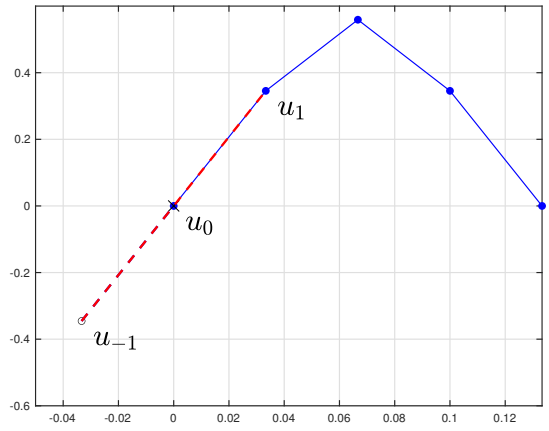


FIG. 10. The simply supported boundary condition: both the state and the curvature at the boundary – at $l = 0$ – should be 0.

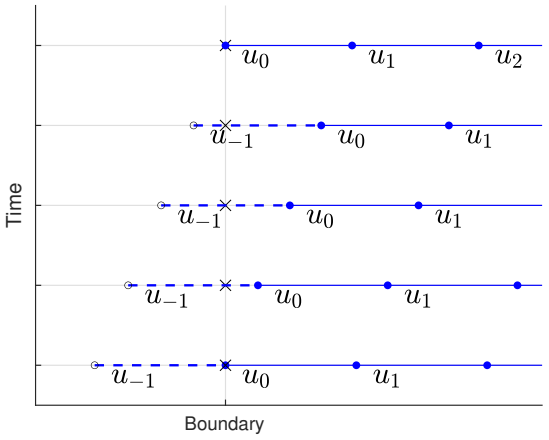


FIG. 11. The grid changing over time

their interactions change through a change in the grid spacing and wave speed. This interaction, though, is defined by λ which

Bilbao, S. (**2009**). *Numerical Sound Synthesis* (John Wiley & Sons, United Kingdom).