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

(In the initial submission, omit all the following author information to ensure anonymity during peer review. On final submission please make sure that the author address is a complete, functioning postal address. Post will be sent to that address.)

Firstname Lastname

Sound Computing Group Full Address
University of Anywhere
1234 Anywhere Street
Somecity, Somestate 012345 USA
email@email.com

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, 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.Too technical for the abstract: can say "Stability analysis follows directly as in the static case" Does it? I haven't tried to perform stability analysis on the method yet..I don't know!

Introduction

TODO: Decide on modelling or modeling

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 on the flySee comment in abstract. 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]. 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,

This article is structured as follows: *NOTES:*

- Section 2: Continuous systems
- Section 3: Numerical methods
- Section 4: Dynamic grid
- Section 5: Results
- Section 6: Discussion
- Section 7: Conclusions and Future work

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 equation (1) may be interpreted as the 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.

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

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

OK, but this notion of a frequency is only true under static conditions...kind of confusing Is it? It should be true for the dynamic case as well right?Well, I guess under slow variations, you can sort of say that there are still modes...but in general once you have time varying behaviour, you lose the ability to do Fourier analysis easily, and the modal analysis is not strictly correct Hmm.. but if this is true for any static combination of c and

L (which it is right?) wouldn't it be true for the dynamic case as well?No way. It's approximately true for sub-audio rate variation in c. But once the variation gets higher, you will start to see AM effects (sidebands!).All modal analysis requires LTI behaviour. You can say that if the rate of variation of c is sufficiently slow, then the modal frequencies above are approximately valid... Ah of course, gotcha!

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 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.

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.

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 $k = 1/f_\mathrm{s}$ (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 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,

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

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 and .

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^n_{-1} and u^n_{N+1}. 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

$$u^n_0 = u^n_N = 0 \quad \text{(Dirichlet)} \tag{9a}$$

$$\delta_{x\cdot} u^n_0 = \delta_{x\cdot} u^n_N = 0 \quad \text{(Neumann)} \tag{9b}$$

where

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

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

$$u^n_{-1} = u^n_1 \quad \text{and} \quad u^n_{N+1} = u^n_{N-1}\,. \tag{11}$$

Now that the full system is described, the output sound can be retrieved by 'recording' the state u^n_l 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.

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 – 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. (7) provides an exact solution to Eq. (1) (see Figures 1a and 1b). If $\lambda > 1$ the system will become unstable (see Figure 1c).

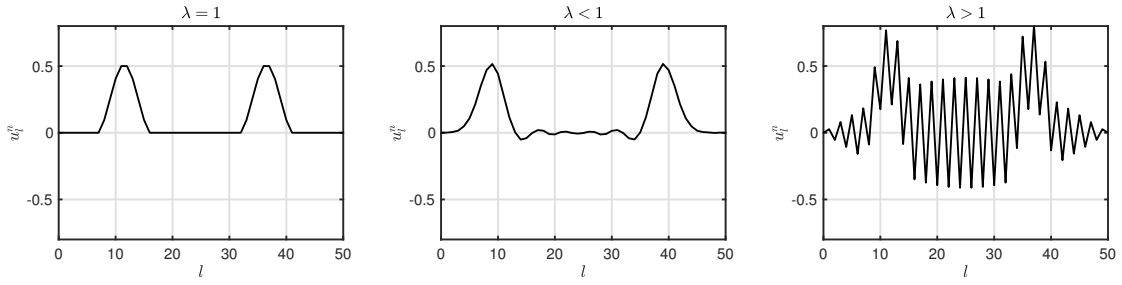


Figure 1. State u_l^n with $N = 50$ and $f_s = 44100$ visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (12) *is not satisfied and the system is unstable.*

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

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

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.

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.

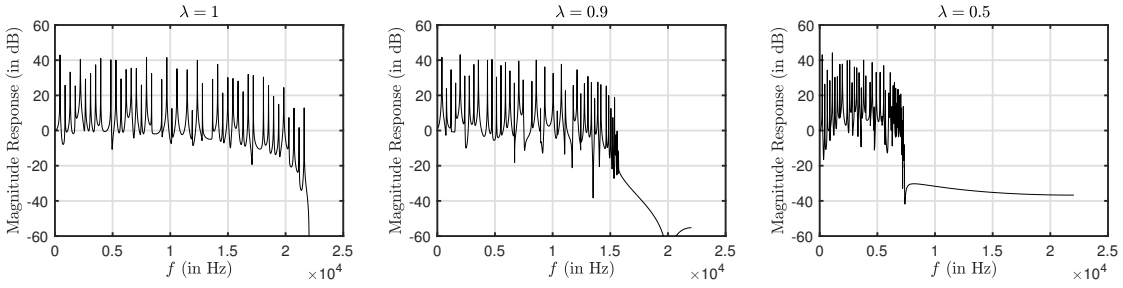


Figure 2. Bandwidths of the simulation output with $f_s = 44100$ Hz and (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$.

By analysing the scheme in Eq. (7), it can be shown that the maximum frequency

produced by the system can be calculated using (**?**, Chap. 6)

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

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.

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.

Dynamic Parameters

This section describes how parameters could be made dynamic using FDTD methods and what this means for the simulation quality detailed in Section . 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 , 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 elaborated on in Section , 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 in a discrete setting. 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.

A solution to the latter is 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 (12), 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. (12) 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 ← elaborate on the fact that this means “number of grid points"? which the following section will describe.

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 .

First, this section will list the requirements of a method that dynamically changes FDTD grid configurations. Then, the iterations done over the course of this project will briefly be described, the details of each can be found in Appendix **??**. (I think I might remove the iterations and appendix actually..) Finally, the proposed method will be described in detail and summarised in the end.

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 grid configurations so that no auditory artefacts 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 $\lfloor N \rfloor - 1$ modes corresponding to the number of moving points of the system ($p = [1, \ldots, \lfloor N \rfloor - 1]$),

5. work in real time.

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

Iterations (optional section)

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 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 (**?**, 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 at this location. 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 does allow for a "fractional" N as described in Section and has a fundamental frequency calculated using (17). 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).

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, the location is chosen to be the center of the system. 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.

System Setup

Consider a grid function, u_l^n with integer $M_u = \lceil 0.5L/ck \rceil$ (or simply M below for brevity) ($\lceil \cdot \rceil$ 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 (see Figure 3a). The following boundary conditions are then imposed:

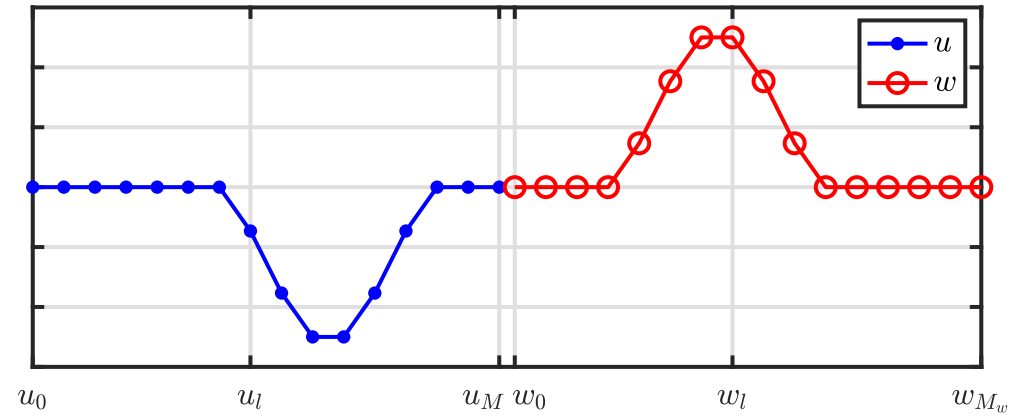
$$u_0^n = w_{M_w}^n = 0, \quad \text{(Dirichlet)} \tag{18a}$$

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

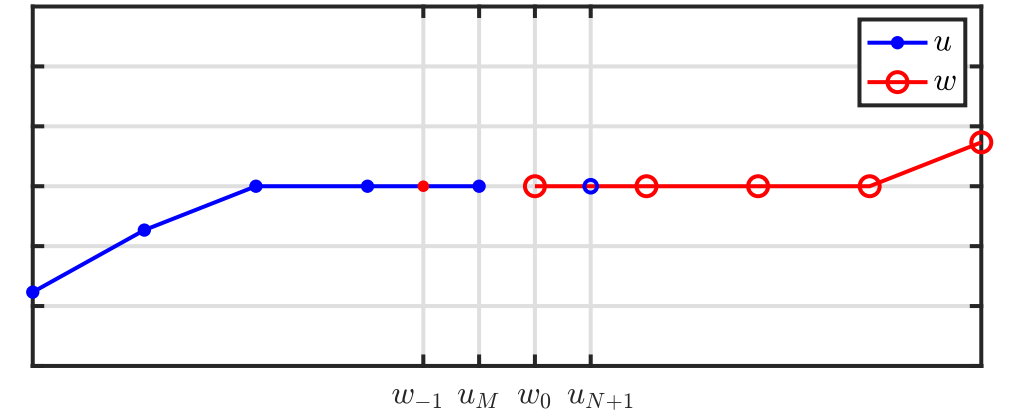
i.e., the outer boundaries are fixed and the inner boundaries are free.

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

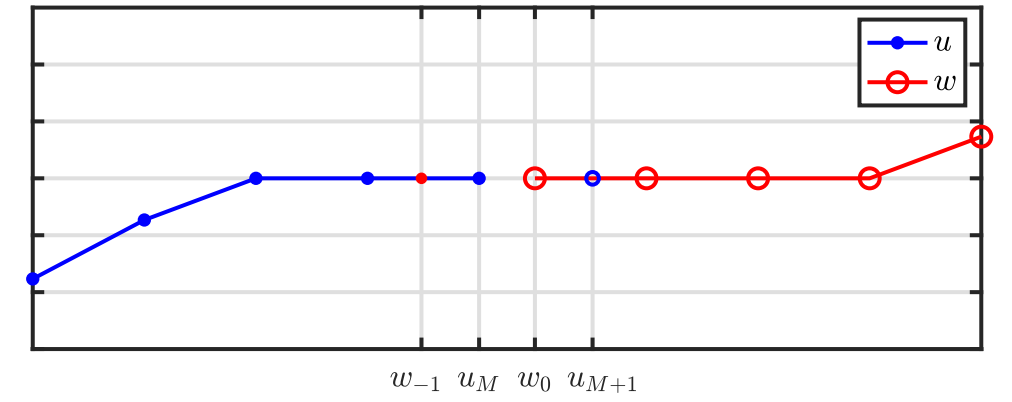
$$u_M^n = w_0^n, \quad \forall n. \tag{19}$$



(a)



(b)



(c)

Figure 3. Illustration of the proposed method. In all figures, the x-axis shows the location (in m) of the respective grid points (fx. $x_{u_l^n}$), but the x and n are omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundary ($N = 30$, $x^n_{u_M} = x^n_{w_0}$). (b) When c – and consequently h – are decreased and the positions of the grid points change ($N = 30.5$, $x^n_{u_M} \neq x^n_{w_0}$). (c) Figure 3a zoomed-in around $x^n_{u_M}$ and $x^n_{w_0}$. The states at the inner boundaries u^n_M and w^n_0 are shown together with virtual grid points u^n_{M+1} and w^n_{-1}.

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

With these 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. The system of FDSs will now be

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

with spreading operator

$$J(x_i) = \begin{cases} \frac{1}{h}, & l = l_i = \lfloor x_i/h \rfloor \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

and the effect of the connection ("connection force", but not really as it isn't in N) F (in $\mathrm{m}^2/\mathrm{s}^2$). 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 $\delta_{tt} u_M^n = \delta_{tt} w_0^n$, and F can be calculated by setting the right side of the equations in (22) equal to each other:

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

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. 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.

Changing the Grid

The previous section describes the case in which the stability condition is satisfied with equality, i.e, when $1/ck$ 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 $h = ck$, 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. (12) with equality, as is the case with the previous iteration described in .

As mentioned above, the state of $\mathbf{w}^n$ (from (20)) can be used to calculate the virtual grid point needed at the right boundary of $\mathbf{u}^n$ and vice versa. If the inner boundaries are not overlapping (i.e., $x^n_{u_M} \neq x^n_{w_0}$) a Lagrangian interpolator $I(x_i)$ at location x_i from the left boundary (in m) can be used to calculate the value of this virtual grid point. The interpolator I is a row-vector the size of $\boldsymbol{\mathcal{U}}^n$ (from Eq. (21)) its values depending on what interpolation order is used. In the following, the distance between the inner boundaries is defined as

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

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

$$u^n_{M+1} = I^{\leftarrow}(x^n_{u_{M+1}})\boldsymbol{\mathcal{U}}^n \tag{27a}$$

$$w^n_{-1} = I(x^n_{w_{-1}})\boldsymbol{\mathcal{U}}^n, \tag{27b}$$

where $I^{\leftarrow}$ is a flipped and shifted version of I. ← further explanation needed here.. 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 $m_i^{\leftarrow} = \lfloor x_i/h + (1-\alpha) \rfloor$. The shift in the latter is necessary to transform the location to the right indices of $\boldsymbol{\mathcal{U}}^n$. ← more explanation here Substituting (28) into (27) yields

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

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

Also see Figure **??**. 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 $f_\mathrm{s}/4$ 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, using a cubic Lagrange interpolator instead. Though 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 to modes higher than $f_\mathrm{s}/4$.

Much better behaviour is observed when points of both $\mathbf{u}^n$ and $\mathbf{w}^n$ 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 and when applied to Eq. (21) yields

$$\begin{aligned} u^n_{M+1} &= I_2^{\leftarrow}(x^n_{u_{M+1}})\boldsymbol{\mathcal{U}}^n \\ &= \frac{\alpha-1}{\alpha+1}u^n_M + w^n_0 - \frac{\alpha-1}{\alpha+1}w^n_1 \end{aligned} \tag{30a}$$

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

As will be shown in Section , quadratic interpolation yields the expected fundamental frequency at all times. One can show that when N is an integer, and thus $\alpha = 0$, Eqs. (30a) and (30b) can be substituted as w^n_1 and u^n_{M-1} into Eqs. (25a) and (25b) respectively (as these acted as virtual grid points u^n_{M+1} and w^n_{-1}). Then recalling Eq. (19) it can be seen that the system is reduced to (25).

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

$$\begin{cases} \mathbf{u}^n = [\mathbf{u}^n, I_3(x^n_{u_{M+1}})\mathbf{v}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is odd}, \\ \mathbf{w}^n = [I_3^{\leftarrow}(x^n_{w_{-1}})\mathbf{v}^n, \mathbf{w}^n]^T & \text{if } \lfloor N^n \rfloor \text{ is even}, \end{cases} \tag{31}$$

where

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

and cubic Lagrangian interpolator

$$I_3 = \left[-\frac{\alpha'(\alpha'+1)}{(\alpha'+2)(\alpha'+3)} \quad \frac{2\alpha'}{\alpha'+2} \quad \frac{2}{\alpha'+2} \quad -\frac{2\alpha'}{(\alpha'+3)(\alpha'+2)} \right], \tag{32}$$

with

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

See Figure 4.

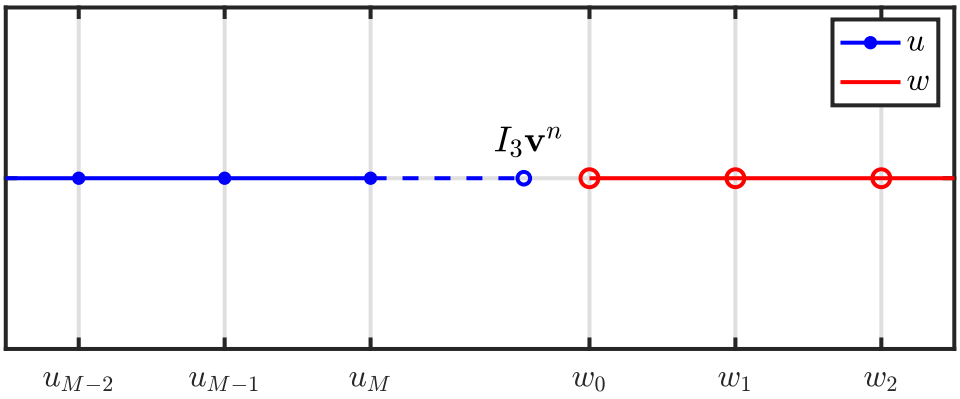


Figure 4. The moment when a point is added to $\mathbf{u}$ *at location* $x_{u_M} + h$ *in Eq.* (31)*. This figure shows an extreme case where this location is far from* x_{w_0}*, i.e.,* $\alpha' \not\approx 0$ *in Eq.* (32)*.*

Except in the case of extremely quick parameter variations, α' in Eq. (32) is expected to be close to zero, i.e., $x^n_{u_M} + h \approx x^n_{w_0}$, meaning that $I_3 \approx [0, 0, 1, 0]$. This makes sense by looking at Figure **??**, as exactly when the boundary points u^n_M and w^n_0 surpass the virtual points w^n_{-1} and u^n_{M+1}, these are going to be close to overlapping.

Removing grid points happens when c, and consequently h, is increased and $x^n_{u_M} \geq x^n_{w_0}$ (or $\lfloor N^n \rfloor < \lfloor N^{n-1} \rfloor$). Compared to adding grid points, removing these is slightly more straightforward as points are simply removed from $\mathbf{u}$ and $\mathbf{w}$ (again for both

n and $n-1$) in an alternating fashion

$$\begin{cases} \mathbf{u}^n = [u_0^n, u_1^n ..., u_{M-1}^n]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w_1^n, w_2^n ..., w_{M_w}^n]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{33}$$

A problem that arises from simply removing a grid points, is that it is possible that $u_M^n \not\approx w_0^n$ at the time of removal. 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 of displacement correction 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:

$$\begin{aligned} u_M^n &:= u_M^n + \beta(w_0^n - u_M^n), \\ w_0^n &:= w_0^n - \beta(w_0^n - u_M^n), \end{aligned} \tag{34}$$

with

$$\beta(\epsilon) = \frac{(1-\alpha)^\epsilon}{2} \tag{35}$$

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

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. Furthermore, one does not have to add and remove points from $\mathbf{u}$ and $\mathbf{w}$ in an alternating fashion as in (31), 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 $\mathbf{w}$ with only one moving grid point, w_0^n) the method

still works.

Summary

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

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

$$u_l^{n+1} = u_{l+1}^n + u_{l-1}^n - u_l^{n-1}, \tag{36a}$$

$$w_l^{n+1} = w_{l+1}^n + w_{l-1}^n - w_l^{n-1}, \tag{36b}$$

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. (36a) and (36b) then become $l = [1, \ldots, M]$ and $l = [0, \ldots, M_w - 1]$ respectively.

The inner boundaries are calculated using

$$u_M^{n+1} = u_{M+1}^n + u_{M-1}^n - u_M^{n-1}, \tag{37a}$$

$$w_0^{n+1} = w_{-1}^n + w_2^n - w_0^{n-1}. \tag{37b}$$

where virtual grid points u_{M+1}^n and w_{-1}^n can be calculated using Eq. (30).

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. (31), and when $\lfloor N^n \rfloor < \lfloor N^{n-1} \rfloor$ a point is removed from the same vectors using Eq. (33). In order to prevent audible artefacts when increasing c (and thus decreasing N), the displacement correction in (34) 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 then be compactly written in matrix form as

$$\boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}\boldsymbol{\mathcal{U}}^{n} - \boldsymbol{\mathcal{U}}^{n-1} \tag{38}$$

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

$$\mathbf{B} = \left[\begin{array}{ccc|cc|c} \ddots & \ddots & & & & 0 \\ 1 & 0 & 1 & & & \\ & 1 & \frac{\alpha-1}{\alpha+1} - \beta & 1+\beta & -\frac{\alpha-1}{\alpha+1} & \\ \hline & -\frac{\alpha-1}{\alpha+1} & 1+\beta & \frac{\alpha-1}{\alpha+1} - \beta & 1 & \\ & & & 1 & 0 & 1 \\ 0 & & & & \ddots & \ddots \end{array}\right] \tag{39}$$

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 $N = M + M_w + 1$.

Implementation

When implementing this method, it is important not to change the grid as the scheme is calculated. I.e., the same α and β need to be used to calculate u_l^n and w_l^n at time index n for all l.

going to include some pseudocode here to rather than the list below

- Wavespeed is changed
- Check whether $\lfloor N^n \rfloor \neq \lfloor N^{n-1} \rfloor$
  - If so add/remove point using (31) or (33)
- Calculate α and β in (39) using (26) and (35).

- Calculate scheme (38)
- Retrieve output
- Update states

Analysis and Results

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

Frequency

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 (39)) 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 5.

The output of the system at integer $N = 31$ contains partials at perfect integer multiples of f_0 (Eq. (17)). When this is compared to the output of the system with $N = 15.5$, one can observe that the lower partials are nearly perfectly overlapping, whereas higher partials exhibit slight downwards deviations, exponentially increasing with frequency.

For a more detailed look at the behaviour of the system, a modal analysis can be performed on system (38). The modal frequency of the p'th mode can be retrieved as

$$f_p = \frac{1}{2\pi k} \cos^{-1}\left(\frac{1}{2}\mathrm{eig}_p(\mathbf{B})\right). \tag{40}$$

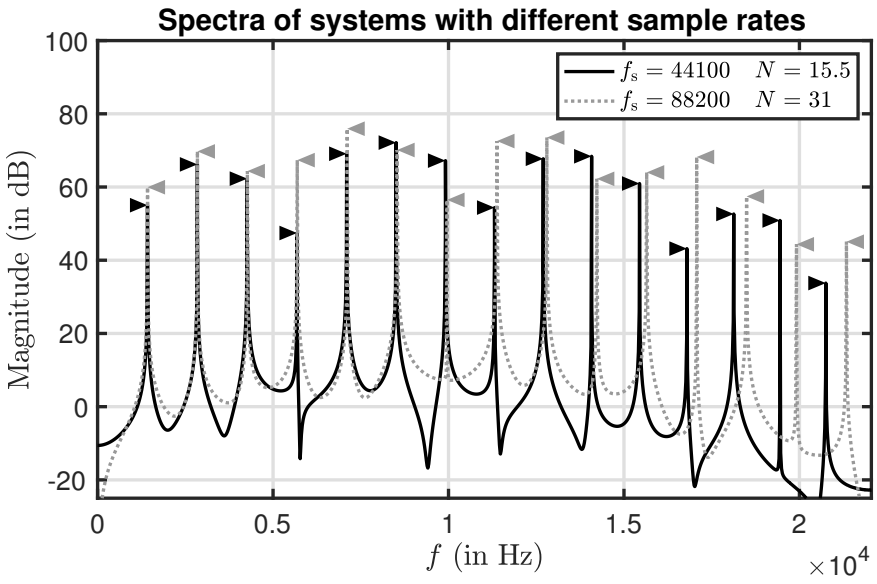


Figure 5. A system with $N = 15.5$ at $f_s = 44100$ compared to a system with $N = 31$ at $f_s = 88200$. Both are supposed to have the same fundamental frequency f_0 according to Eq. (17).

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. The displacement correction presented in Eq. (34) is not included in this analysis ($\epsilon \gg 1$), but will be elaborated on in Section . The results of the analysis are shown in Figure 6. Figure 7 shows the resulting spectrogram of the system excited at $n = 0$ with a narrow raised cosine and the output retrieved at u_1.

In the following came up with this measure myself.. it made sense to me for measuring harmonic deviation / inharmonicity. Please let me know if you know a better measure or something I can reference

$$\varepsilon_p = \frac{f_p - p f_0}{f_0} \quad [\%] \tag{41}$$

is used to calculate the harmonic deviation of mode p from an integer multiple of the fundamental. Furthermore, 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).

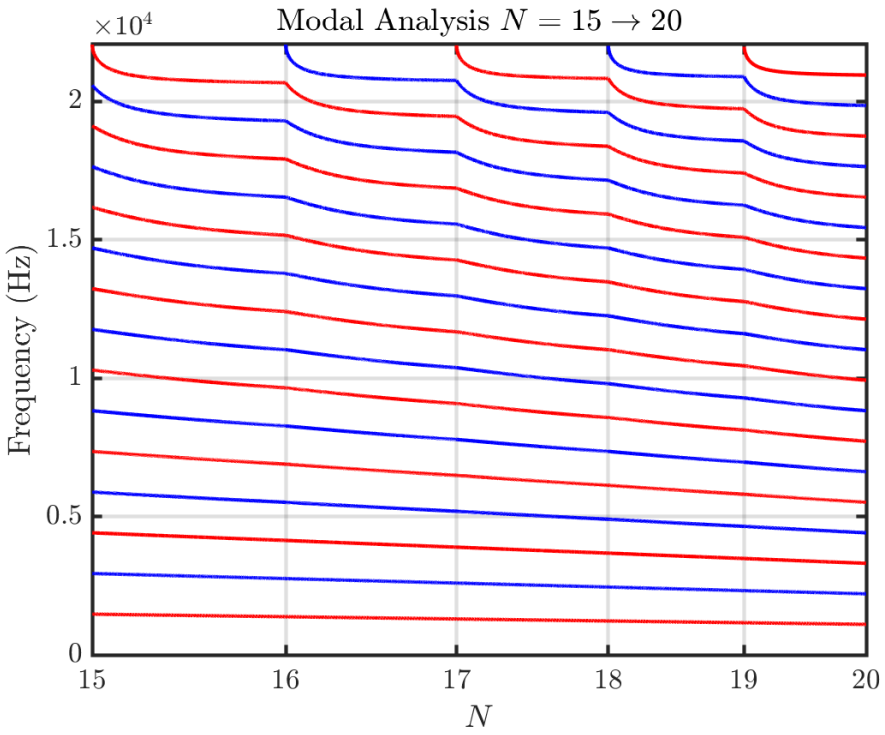


Figure 6. Modal analysis of system (38) *using* (40)*. The wave speed is reduced from* $c = 2940$ *(*$N = 15$*) to* $c = 2205$ *(*$N = 20$*).*

The first thing one can observe from Figure 6 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 -0.12 Hz or $\varepsilon_1 = -0.008\%$. In this same interval f_{15} maximally deviates by -827.15 Hz or $\varepsilon_{15} = -57\%$. This deviation for mode number p gets progressively less as N increases. The maximum deviation for the highest mode number $f_{\lfloor N \rfloor}$, however, gets larger with a larger N, with $\max(\varepsilon_{\lfloor N \rfloor})$ approaching -100% for extremely large values of N (> 1000). These deviations only happen between integer values of N where at integer N all modes are perfect integer multiples of f_0 and $\varepsilon_p = 0\%$ for all p.

Experiments with higher even-ordered Lagrange interpolators shows that ε_p becomes 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

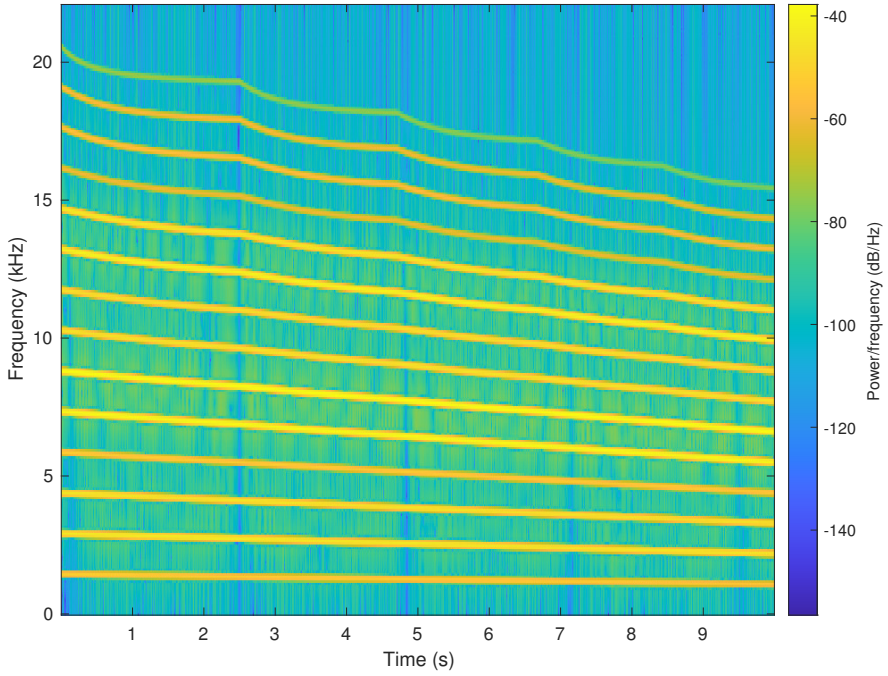


Figure 7. System output. The sound output follows the same pattern as predicted by the analysis shown in Figure 6.

interpolators.

Another observation from Figure 6 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 7 the highest mode is not excited. When an implementation of the system using this method, with wave speed $c = 2940$ m/s (static) is compared to a normal implementation of the 1D wave equation (shown in Section) with the same wavespeed, identical outputs are observed, even though the latter has $N - 1$ moving points. This proves that at integer N the method reduces to the normal case. If the system is excited when N is not an integer, the highest mode will also be excited.

Using the quadratic interpolation from (30), 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.

Displacement Correction

(FIR) Comb filtering effect where resonances appear depending on the location of correction. Middle has least

Limit on Speed of Change

The method presented in this paper can only add or remove a maximum one point per sample using Eqs. (31) and (33). 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

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 ().

The fact that he highest mode is not excited is probably due to the restriction imposed on the two moving but connected points through the rigid connection.

All issues with the proposed method happen in higher frequencies. Physical

processes usually lose the highest frequencies first

Due to high-frequency losses in physical systems.

Higher modes

As can be seen from the figure, the lower harmonic partials of both systems line up almost exactly before the partials interpolated start to deviate towards the lower end of the spectrum.

($< 0.2\%$ of their respective frequency, $< 1.3\%$ of the fundamental f_0).

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 $f_\mathrm{s} = 44100$ (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...

The

Conclusion and Future Work

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. Though this method has only been presented using the 1D wave equation it could potentially be applied to any kind of 1D FDS

References