REAL-TIME IMPLEMENTATION OF THE DYNAMIC STIFF STRING USING FINITE-DIFFERENCE TIME-DOMAIN METHODS AND THE DYNAMIC GRID

Silvin Willemsen and Stefania Serafin

Multisensory Experience Lab
Aalborg University Copenhagen
Denmark
`sil@create.aau.dk | sts@create.aau.dk`

ABSTRACT

Digital musical instruments based on physical modelling have gained increased popularity over the past years. This is partly due to recent advances in computational power, which allow for their real-time implementation. One of the great potentials for digital musical instruments based on physical models, is that one can go beyond what is physically possible and change properties of the instruments which are static in real life. This paper presents a real-time implementation of the dynamic stiff string using finite-difference time-domain (FDTD) methods. The defining parameters of the string can be varied in real time and change the underlying grid that these methods rely on based on the recently developed dynamic grid method. For most settings, parameter changes are nearly instantaneous and do not cause noticeable artefacts due to changes in the grid. A reliable way to prevent artefacts for all settings is under development.

1. INTRODUCTION

Physical models of strings have been known since D'Alembert's formulations in the 18th century, but recent computational developments have allowed for their real-time implementation. One of the first real-time physical models, due to Karplus and Strong, was that of a (damped) ideal string and used digital waveguides (DWGs) [1]. Jaffe and Smith used allpass filters to extend the algorithm to include inharmonicity due to stiffness in [2]. Other techniques, such as modal synthesis [3], have been used to model stiff strings in e.g. [4]. Both these techniques, however, rely on assumptions that the system can be decomposed into either travelling waves (for DWGs) or uncoupled modes (for modal synthesis) [5].

Finite-difference time-domain (FDTD) methods, on the other hand, only assume that a continuous partial differential equation (PDE) can be described over a grid, which is by no means restrictive [5]. These methods are therefore very general and allow for physical parameters to be directly modelled and controlled. FDTD methods were first used to model strings by Ruiz in [6] and Hiller and Ruiz in [7, 8], and appear in many recent publications [9, 10, 11, 12].

According to the authors, one of the most interesting potentials of physical modelling is to exploit the virtual nature of the simulation, and to go beyond that what is physically possible. One could, for example, change material properties and geometry of the string while the simulation is running, to create sounds which are impossible in the physical world. Furthermore, using time-varying parameters could help greatly in tuning the models in real time, as opposed to needing to change the parameters at the start of the simulation.

An issue with FDTD methods is that the parameters describing the underlying system are closely tied to the discrete grid that these methods rely on. Previously mentioned methods have been shown to allow for smooth parameter changes, see e.g., [13] for DWGs and [4, 14] for modal synthesis, but come with their own aforementioned drawbacks.

Recently, the authors of the current work developed a method to smoothly change grid configurations of FDTD-based musical instrument simulations referred to as 'the dynamic grid'. Using this method, the defining parameters of the physical model can be varied while smoothly adapting the underlying grid to the changes in parameter values. As opposed to, e.g., adaptive mesh refinement [15], this method uses the same time step for the entire grid, and thus does not require complex interfacing between different parts of the grid. The dynamic grid method first appeared in [16], and presented its application to the 1D wave equation. Simultaneously, a real-time physical model of the trombone was presented in [17] using this method for implementing the time-varying length of the instrument in real time. A more in-depth description of the method can be found in [12, Ch. 12].

More recently, [18] extended the dynamic grid method to more complex systems, including the damped stiff string, by using a matrix-vector formulation, which will be used as the foundation of this work. This paper presents the work on the real-time dynamic stiff string and provides more details on the mathematical formulation as well as its (real-time) implementation.

The rest of this paper is structured as follows: Section 2 presents the non-dynamic stiff string in continuous time. Section 3 introduces FDTD methods and discretises the (non-dynamic) stiff string. The dynamic grid is presented in Section 4 and applied to the stiff string. The real-time application, together with implementation details, are presented in Section 5, and results are presented and discussed in Section 6. Finally, concluding remarks appear in Section 7.

2. PHYSICAL MODEL

The model of the stiff string is well covered in the literature [19, 5], but will be recalled here. Consider the transverse displacement of a stiff string of length L (in m) described by $u = u(x, t)$ (in m) with time $t \geq 0$ (in s) and space $x \in \mathcal{D}$ (in m), where domain $\mathcal{D} = [0, L]$. Using ∂_t and ∂_x to denote partial derivatives in time

t and space x respectively, the dynamics of the damped stiff string can be described by the following PDE [20]:

$$\rho A\partial_t^2 u = T\partial_x^2 u - EI\partial_x^4 u - 2\sigma_0\rho A\partial_t u + 2\sigma_1\rho A\partial_t\partial_x^2 u. \quad (1)$$

Parameters are material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), tension T (in N), Young's modulus E (in Pa), area moment of inertia $I = \pi r^4/4$ (in m^4) and frequency-independent and frequency-dependent loss coefficients $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s) respectively. Notice that the parameters in Eq. (1) are not time-varying.

For any system distributed in space, boundary conditions need to be defined [5]. In this work, simply supported boundary conditions are chosen such that:

$$u = \partial_x^2 u = 0, \quad \text{at} \quad x = 0, L. \quad (2)$$

3. NUMERICAL METHODS

Following the notation in [5], this section introduces FDTD methods and applies it to the stiff string presented in the previous section.

Applying FDTD methods to discretise a continuous-time PDE such as Eq. (1) starts by defining a discrete grid in time and space [5]. Time can be discretised using $t = nk$, with temporal index $n = 0, 1, \ldots$ and time step $k = 1/f_\text{s}$ (in s) where f_s is the sample rate of the simulation (in Hz). The continuous spatial domain $\mathcal{D}$ can be subdivided into N equal intervals according to $x = lh$. Here h is the grid spacing (in m) and spatial index $l \in \{0, \ldots, N\}$, resulting in $N+1$ grid points. Using these definitions, the continuous state variable $u = u(x,t)$ can be discretised to grid function u_l^n, which describes the transverse displacement of the discrete string at spatial index l and temporal index n.

3.1. Finite-Difference Operators

To approximate the continuous-time derivatives in Eq. (1), shift operators must be introduced. These can be applied to a grid function to shift it with one step either in space or time. The forward and backward shift in time, as well as the identity operator are defined as follows:

$$e_{t+}u_l^n = u_l^{n+1}, \quad e_{t-}u_l^n = u_l^{n-1}, \quad \text{and} \quad 1u_l^n = u_l^n. \quad (3)$$

Shift operators can be used to create finite-difference (FD) operators that approximate derivatives. The forward, backward and centred difference in time operators are defined respectively as

$$\delta_{t+} \triangleq \frac{e_{t+} - 1}{k}, \quad \delta_{t-} \triangleq \frac{1 - e_{t-}}{k}, \quad \delta_{t\cdot} \triangleq \frac{e_{t+} - e_{t-}}{2k} \quad (4)$$

and all approximate a first-order temporal derivative ∂_t. The centred difference can be shown to be second-order accurate as opposed to the first-order accurate forward and backwards difference operators [5].

Similarly, forward and backward shifts in space are

$$e_{x+}u_l^n = u_{l+1}^n, \quad \text{and} \quad e_{x-}u_l^n = u_{l-1}^n, \quad (5)$$

and can be used to create the forward, backward and centred difference in space operators:

$$\delta_{x+} \triangleq \frac{e_{x+} - 1}{h}, \quad \delta_{x-} \triangleq \frac{1 - e_{x-}}{h}, \quad \delta_{x\cdot} \triangleq \frac{e_{x+} - e_{x-}}{2h}, \quad (6)$$

all approximating ∂_x.

Combinations of these FD operators can be used to approximate higher-order derivatives:

$$\delta_{tt} = \delta_{t+}\delta_{t-} \triangleq \frac{e_{t+} - 2 + e_{t-}}{k^2}, \quad (7)$$

$$\delta_{xx} = \delta_{x+}\delta_{x-} \triangleq \frac{e_{x+} - 2 + e_{x-}}{h^2} \quad (8)$$

$$\delta_{xxxx} = \delta_{xx}\delta_{xx} \triangleq \frac{e_{x+}^2 - 4e_{x+} + 6 - 4e_{x-} + e_{x-}^2}{h^4}, \quad (9)$$

which approximate ∂_t^2, ∂_x^2 and ∂_x^4 respectively. A squared shift operator means to apply the operator twice.

3.2. Discrete Damped Stiff String

Using the operators defined in Section 3.1, the damped stiff string in Eq. (1) can be discretised to the following most commonly used FDTD scheme [5] (notice the division by ρA):

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n, \quad (10)$$

with wave speed $c = \sqrt{T/\rho A}$ (in m/s), stiffness coefficient $\kappa = \sqrt{EI/\rho A}$ (in m^2/s). The operators approximating the first-order temporal derivatives are chosen to yield the highest accuracy while keeping the system explicit.

To implement Eq. (10), the operators must be expanded and the equation solved for u_l^{n+1} (the only unknown). This results in the following update equation (before division by A):

$$\begin{aligned} Au_l^{n+1} &= B_0u_l^n + B_1\left(u_{l+1}^n + u_{l-1}^n\right) + B_2\left(u_{l+2}^n + u_{l-2}^n\right) \\ &\quad + C_0u_l^{n-1} + C_1\left(u_{l+1}^{n-1} + u_{l-1}^{n-1}\right), \end{aligned} \quad (11)$$

with

$$\begin{aligned} A &= 1 + \sigma_0 k, & B_2 &= -\mu^2, \\ B_0 &= 2 - 2\lambda^2 - 6\mu^2 - 2S, & C_0 &= \sigma_0 k + 2S - 1, \\ B_1 &= \lambda^2 + 4\mu^2 + S, & C_1 &= -S, \end{aligned} \quad (12)$$

and

$$\lambda = \frac{ck}{h}, \quad \mu = \frac{\kappa k}{h^2}, \quad \text{and} \quad S = \frac{2\sigma_1 k}{h^2}. \quad (13)$$

The boundary condition in Eq. (2) can be discretised to:

$$u_l^n = \delta_{xx}u_l^n = 0, \quad \text{at} \quad l = 0, N. \quad (14)$$

This is implemented by reducing the range of calculation to $l = \{1, \ldots, N-1\}$ (as the boundary points are 0 at all times), and by applying the following definitions for the *virtual grid points*, which are needed to calculate Eq. (11) at $l = 1$ and $l = N-1$ respectively:

$$u_{-1}^n = -u_1^n, \quad \text{and} \quad u_{N+1}^n = -u_{N-1}^n. \quad (15)$$

Equation (10), like all explicit FDTD schemes, needs to abide a stability condition [5]. Usually this is written in terms of the grid spacing h, which in the case of the damped stiff string can be shown to be (using von Neumann analysis [21])

$$h \geq h_\text{min} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2k^2}}{2}}. \quad (16)$$

See [12, Ch. 4] for a derivation. The closer h is to the minimum stable grid spacing h_{min} (in m) for a given time step k, the higher the simulation quality and bandwidth.[1] In implementation, this is achieved by performing the following operations in order:

$$N := \left\lfloor \frac{L}{h_{\text{min}}} \right\rfloor, \quad h := \frac{L}{N}. \tag{17}$$

Here, $\lfloor\cdot\rfloor$ denotes the flooring operation, and is necessary as the number of intervals needs to be an integer value. This causes many combinations of parameters to not yield the optimal simulation quality, though usually not noticable for a large N.

3.3. Matrix Form

To be able to apply the dynamic grid to the stiff string later on, the update equation in Eq. (11) needs to be written in matrix form. One can store the state of u_l^n for $l = \{1, \dots, N-1\}$ (as $u_0^n = u_N^n = 0$ due to the simply supported boundary condition) in the $(N-1)\times 1$ column vector

$$\mathbf{u}^n = [u_1^n, \dots, u_{N-1}^n]^T, \tag{18}$$

where T denotes the transpose operation. Using Eq. (18), at different time indices, Eq. (11) can then be written in matrix form as:

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{19}$$

where A is as defined in Eq. (12), and

$$\begin{aligned}\mathbf{B} &= 2\mathbf{I}_{N-1} + \lambda^2\mathbf{D}_{xx} - \mu^2\mathbf{D}_{xxxx} + S\mathbf{D}_{xx}, \text{ and}\\ \mathbf{C} &= -(1-\sigma_0 k)\mathbf{I}_{N-1} - S\mathbf{D}_{xx},\end{aligned} \tag{20}$$

where the $(N-1)\times(N-1)$ matrix

$$\mathbf{D}_{xx} = \begin{bmatrix} \ddots & \ddots & & & \mathbf{0} \\ \ddots & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & \ddots \\ \mathbf{0} & & & \ddots & \ddots \end{bmatrix}, \tag{21}$$

is the matrix form of the δ_{xx} operator in Eq. (8) (scaled by h^2) and $\mathbf{I}_{N-1}$ is the $(N-1)\times(N-1)$ identity matrix. Finally, for simply supported boundary conditions,

$$\mathbf{D}_{xxxx} = \mathbf{D}_{xx}\mathbf{D}_{xx} = \begin{bmatrix} 5 & -4 & 1 & & \mathbf{0} \\ -4 & 6 & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & 6 & -4 \\ \mathbf{0} & & 1 & -4 & 5 \end{bmatrix}, \tag{22}$$

and is the matrix form of the δ_{xxxx} operator in Eq. (9) (scaled by h^4).

[1]For a smaller time step k, a smaller minimum grid spacing h_{min} can be chosen. See [12, Sec. 2.4.4] for additional intuition.

4. THE DYNAMIC STIFF STRING

One could imagine that varying the defining parameters of the damped stiff string might cause various issues in the above mentioned framework. Large enough changes in parameters causes a change in the number of intervals N as per Eq. (17), which, due to the flooring operation, causes sudden changes in the number of grid points defining the system. As one might imagine, this could cause audible artefacts and – in the worst case – an unstable simulation.

The formulation of the dynamic grid therefore starts by introducing a *fractional number of intervals* $\mathcal{N}$, where $\lfloor\mathcal{N}\rfloor = N$. Apart from allowing for smooth transitions between grid configurations, substituting N for $\mathcal{N}$ in Eq. (17) removes the flooring operation and allows for $h = h_{\text{min}}$ at all times, yielding an optimal simulation quality.

In the following, all variables that are made time varying will get the superscript n. These are all physical parameters from Eq. (1): L^n, ρ^n, A^n, r^n, T^n, E^n, I^n, σ_0^n, σ_1^n, and all derived parameters used in Section 3.2: h^n, N^n c^n, κ^n, λ^n, and S^n. Notice that the sample rate f_{s} and therefore the time step k remain fixed. The fractional number of intervals can then be calculated according to

$$\mathcal{N}^n = \frac{L^n}{h^n}. \tag{23}$$

Finally, the (time-varying) location of grid point u_l (in meters from the left boundary) at time n will be denoted by $x_{u_l}^n$.

4.1. System Setup

As done in [16, 18], to implement the fractional number of intervals, the original system u_l^n first needs to be split into two subsystems $v_{l_v}^n$ and $w_{l_w}^n$ with $l_v \in \{0, \dots, M_v^n\}$ and $l_w \in \{0, \dots, M_w^n\}$. Here,

$$M_v^n = N^n - M_w^n \quad \text{and} \quad 0 < M_w^n < N^n \tag{24}$$

are the number of intervals of the left and right subsystems respectively. Notice that the total number of grid points $(M_v^n + M_w^n + 2)$ is one more than the original system $(N^n + 1)$, and that the lower limit of the number grid points on a subsystem is 2 grid points (including the boundary). The FDTD scheme in (10) can then be split into the following schemes:

$$\begin{aligned}\delta_{tt}v_{l_v}^n &= (c^n)^2\delta_{xx}v_{l_v}^n - (\kappa^n)^2\delta_{xxxx}v_{l_v}^n \\ &\quad - 2\sigma_0^n\delta_{t\cdot}v_{l_v}^n + 2\sigma_1^n\delta_{t-}\delta_{xx}v_{l_v}^n,\end{aligned} \tag{25a}$$

$$\begin{aligned}\delta_{tt}w_{l_w}^n &= (c^n)^2\delta_{xx}w_{l_w}^n - (\kappa^n)^2\delta_{xxxx}w_{l_w}^n \\ &\quad - 2\sigma_0^n\delta_{t\cdot}w_{l_w}^n + 2\sigma_1^n\delta_{t-}\delta_{xx}w_{l_w}^n,\end{aligned} \tag{25b}$$

with wave speed $c^n = \sqrt{T^n/\rho^n A^n}$ and stiffness coefficient $\kappa^n = \sqrt{E^n I^n/\rho^n A^n}$, both made time-varying. Here, the time-varying cross-sectional area and moment of inertia are $A^n = \pi(r^n)^2$ and $I^n = \pi(r^n)^4/4$ respectively, and the stability condition in Eq. (16) is modified to

$$h^n = \sqrt{\frac{(c^n)^2k^2 + 4\sigma_1^n k + \sqrt{\left((c^n)^2k^2 + 4\sigma_1^n k\right)^2 + 16(\kappa^n)^2k^2}}{2}}, \tag{26}$$

which, as mentioned before, can now always be satisfied with equality.

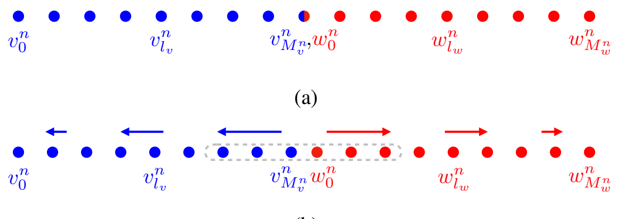


Figure 1: *Visualisation of the system setup for the dynamic grid method. (a) The split systems in Eq.* (25)*. The inner boundaries are overlapping. (b) Grid points move according to Eq.* (27) *due to parameter variation (in this case h^n is decreased). The boundaries no longer overlap. The dashed grey box is used as a reference for Figures 2 and 3.*

The subsystems in (25) are placed adjecent on the same domain x with the locations of the grid points defined as

$$x_{v_{l_v}}^n = l_v h^n \quad \text{and} \quad x_{w_{l_w}}^n = L^n - (M_w^n - l_w)h^n, \tag{27}$$

for subsystems $v_{l_v}^n$ and $w_{l_w}^n$ respectively. See Figure 1a. Notice that the locations $l_v = 0$ and $l_w = M_w^n$, which will be referred to as the *outer boundaries*, are fixed along the x-axis at the edges of the domain $x = 0$ and $x = L^n$. Furthermore, the outer boundaries will have the same boundary conditions as the original system, i.e., the simply supported condition as defined in Eq. (14). The locations $l_v = M_v^n$ and $l_w = 0$, referred to as the *inner boundaries* will, however, need to be calculated in a different fashion.

4.2. Connecting the Inner Boundaries

Here, the (damped) 1D wave equation will be taken as a starting point, which is done by setting the stiffness coefficient $\kappa^n = 0$ in Eq. (25) such that the δ_{xxxx} operator can be ignored for now. One can observe that expanding the δ_{xx} operator (see Eq. (8)) at the inner boundaries, i.e., Eq. (25a) at $l_v = M_v^n$ and Eq. (25b) $l_w = 0$, shows that the system requires definitions for two virtual grid points: $v_{M_v^n+1}^n$ and w_{-1}^n. If $\mathcal{N}^n = N^n$ and thus $\mathcal{N}^n$ is an integer, the inner boundaries perfectly overlap as per Eq. (27), and a rigid connection is imposed on the inner boundaries as

$$v_{M_v^n}^n = w_0^n, \quad \text{if } x_{v_{M_v^n}}^n = x_{w_0}^n. \tag{28}$$

As shown in [16], calculating the virtual grid points according to

$$v_{M_v^n+1}^n = w_1^n, \quad \text{and} \quad w_{-1}^n = v_{M_v^n-1}^n \tag{29}$$

yields identical behaviour between the split system and the original system.

If any physical parameter is changed, the locations of the grid points will change according to Eq. (27). See Figure 1b. If the length L^n is changed, only the grid points of the right subsystem will move. Changes in any other parameter cause a change in the grid spacing h^n through Eq. (26). The latter results in the grid points of the two subsystems to move away from or towards their respective outer boundaries according to Eq. (27). In either case, as the inner boundaries will no longer overlap, Eq. (29) can no longer be used and alternative definitions for the virtual grid points need to be found.

Following earlier work in [16, 18] quadratic Lagrangian interpolation[2] can be used, yielding the following definitions for the virtual grid points:

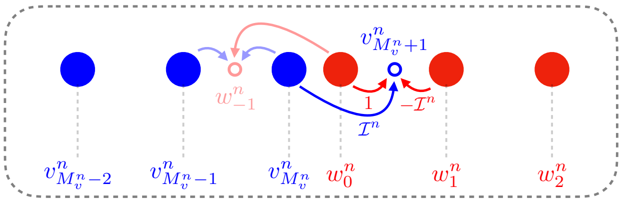


Figure 2: *Figure 1b zoomed in. Virtual grid points are calculated from surrounding grid point values using quadratic interpolation according to Eq.* (30)*. The calculation of $v_{M_v^n+1}^n$ is highlighted.*

$$v_{M_v^n+1}^n = \mathcal{I}^n v_{M_v^n}^n + w_0^n - \mathcal{I}^n w_1^n, \tag{30a}$$
$$w_{-1}^n = -\mathcal{I}^n v_{M_v^n-1}^n + v_{M_v^n}^n + \mathcal{I}^n w_0^n, \tag{30b}$$

where

$$\mathcal{I}^n = \frac{\alpha^n - 1}{\alpha^n + 1}, \tag{31}$$

and

$$\alpha^n = \mathcal{N}^n - N^n \tag{32}$$

is the fractional part of $\mathcal{N}^n$ (and thus $0 \leq \alpha^n < 1$). See Figure 2 for a visualisation of Eq. (30).

As can be seen from Eq. (9), expanding the δ_{xxxx} operator at $v_{M_v^n}^n$ and w_0^n requires two virtual grid points to be calculated for each of these inner boundaries. Similarly, at $v_{M_v^n-1}^n$ and w_1^n the definition of a single virtual grid point is required. To find these definitions, a matrix form needs to be used.

4.3. Matrix Form

To write system (25) in matrix form, the state vectors can be stacked to the following $N^n \times 1$ column vector:

$$\boldsymbol{u}^n = \begin{bmatrix} \mathbf{v}^n \\ \mathbf{w}^n \end{bmatrix} \tag{33}$$

with $\mathbf{v}^n = [v_1^n, \ldots, v_{M_v^n}^n]^T$ and $\mathbf{w}^n = [w_0^n, \ldots, w_{M_w^n-1}^n]^T$. Notice that the outer boundaries, v_0^n and $w_{M_w^n}^n$, are not included as their states are 0 at all times.

Similar to Eq. (19), system (25) can be written in matrix form as

$$A^n \boldsymbol{u}^{n+1} = \boldsymbol{\mathcal{B}}^n \boldsymbol{u}^n + \boldsymbol{\mathcal{C}}^n \boldsymbol{u}^{n-1} \tag{34}$$

where $A^n = 1 + \sigma_0^n k$ and

$$\begin{aligned} \boldsymbol{\mathcal{B}}^n &= 2\mathbf{I}_{N^n} + (\lambda^n)^2 \boldsymbol{\mathcal{D}}_{xx}^n - (\mu^n)^2 \boldsymbol{\mathcal{D}}_{xxxx}^n + S^n \boldsymbol{\mathcal{D}}_{xx}^n, \\ \boldsymbol{\mathcal{C}}^n &= -(1 - \sigma_0^n k)\mathbf{I}_{N^n} - S^n \boldsymbol{\mathcal{D}}_{xx}^n, \end{aligned} \tag{35}$$

with $\lambda^n = c^n k/h^n$, $\mu^n = \kappa^n k/(h^n)^2$ and $S^n = 2\sigma_1^n k/(h^n)^2$.

[2] See [12, Ch. 12.3] for experiments using other interpolators.

Here, $N^n \times N^n$ matrix

$$\boldsymbol{\mathcal{D}}_{xx}^n = \left[\begin{array}{ccc|ccc} \ddots & \ddots & & & & \mathbf{0} \\ \ddots & -2 & 1 & & & \\ & 1 & \mathcal{I}^n - 2 & 1 & -\mathcal{I}^n & \\ \hline & -\mathcal{I}^n & 1 & \mathcal{I}^n - 2 & 1 & \\ & & & 1 & -2 & \ddots \\ \mathbf{0} & & & & \ddots & \ddots \end{array}\right] \qquad (36)$$

is an adapted version of Eq. (21) including the quadratic interpolation at the inner boundaries in Eq. (30).

Furthermore, $\boldsymbol{\mathcal{D}}_{xxxx}^n$ is a matrix form of the δ_{xxxx} operator in Eq. (9) (scaled by h^4) which includes the dynamic grid method. It is created by substituting $\mathbf{D}_{xx}$ in Eq. (22) for $\boldsymbol{\mathcal{D}}_{xx}^n$ in Eq. (36), and performing the matrix product $\boldsymbol{\mathcal{D}}_{xx}^n \boldsymbol{\mathcal{D}}_{xx}^n$. This yields the following $N^n \times N^n$ matrix[3]

$$\boldsymbol{\mathcal{D}}_{xxxx}^n = \left[\begin{array}{cccc|cccc} 5 & \ddots & \ddots & & & & & \mathbf{0} \\ \ddots & \ddots & -4 & 1 & & & & \\ \ddots & -4 & 6 & \mathfrak{J}_1^n & 1 & \mathfrak{J}_3^n & & \\ & 1 & -4 & \mathfrak{J}_0^n & \mathfrak{J}_1^n & \mathfrak{J}_2^n & \mathfrak{J}_3^n & \\ \hline & \mathfrak{J}_3^n & \mathfrak{J}_2^n & \mathfrak{J}_1^n & \mathfrak{J}_0^n & -4 & 1 & \\ & & \mathfrak{J}_3^n & 1 & \mathfrak{J}_1^n & 6 & -4 & \ddots \\ & & & & 1 & -4 & \ddots & \ddots \\ \mathbf{0} & & & & & \ddots & \ddots & 5 \end{array}\right], \qquad (37)$$

with

$$\begin{aligned} \mathfrak{J}_0^n &= (\mathcal{I}^n)^2 - 4\mathcal{I}^n + 6, & \mathfrak{J}_1^n &= \mathcal{I}^n - 4, \\ \mathfrak{J}_2^n &= -(\mathcal{I}^n)^2 + 4\mathcal{I}^n + 1, & \mathfrak{J}_3^n &= -\mathcal{I}^n. \end{aligned} \qquad (38)$$

Figure 3 visualises the calculation of the virtual grid points required to calculate the inner boundaries $v_{M_v^n}^n$ and w_0^n. Those required at $v_{M_v^n-1}^n$ and w_1^n can be visualised by Figure 2.

Although the matrix in Eq. (37) looks quite different from $\mathbf{D}_{xxxx}$ in Eq. (22), Eq. (34) exhibits identical behaviour (within machine precision) to the original system in Eq. (19) with the same parameters for a value of $\alpha^n = 0$ in Eq. (31) (and thus $\mathcal{I}^n = -1$ in Eq. (38)). Furthermore, if $\alpha^n \to 1$ in Eq. (31) (and thus $\mathcal{I}^n \to 0$), the matrix in (37) reduces to $\mathbf{D}_{xxxx}$ as defined in Eq. (22).

In the extreme case that the right system only has one moving grid point (i.e., $M_w^n = 1\ \forall n$ in Eq. (24)), the simply supported boundary condition in Eq. (14) can be used. In this case, the lower-right quadrant of matrix (36) will only have one element, and $\boldsymbol{\mathcal{D}}_{xx}^n \boldsymbol{\mathcal{D}}_{xx}^n$ results in:

$$\boldsymbol{\mathcal{D}}_{xxxx}^n = \left[\begin{array}{ccccc|c} 5 & -4 & 1 & & \mathbf{0} & \mathbf{0} \\ -4 & \ddots & \ddots & \ddots & & \\ 1 & \ddots & 6 & -4 & 1 & \\ & \ddots & -4 & 6 & \mathfrak{J}_1^n & 1 \\ \mathbf{0} & & 1 & -4 & \mathfrak{J}_0^n, & \mathfrak{J}_1^n - \mathfrak{J}_3^n \\ \hline \mathbf{0} & & \mathfrak{J}_3^n & \mathfrak{J}_2^n & \mathfrak{J}_1^n & \mathfrak{J}_0^n - 1 \end{array}\right]. \qquad (39)$$

[3] Note that only the first element and the last element of the main diagonal are 5, the rest are 6 unless denoted otherwise.

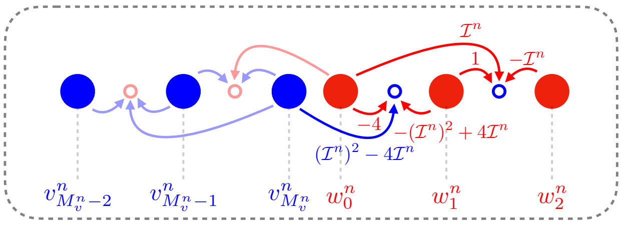


Figure 3: *Figure 1b zoomed in. Visualisation of the calculation of the virtual grid points needed to calculate the inner boundaries $v_{M_v^n}^{n+1}$ and w_0^{n+1}. The calculations of $v_{M_v^n+1}^n$ and $v_{M_v^n+2}^n$ are highlighted.*

4.4. Adding and Removing Grid Points

If parameters are changed such that $N^n \neq N^{n-1}$, grid points must be added to or removed from the system in a smooth fashion. As done in [18], this work only considers changes in grid configurations to affect the left system. If the right system is to be affected, or points are added or removed from both systems in an alternating fashion (as done in [16, 17]), the same principles apply.

If $N^n > N^{n-1}$, grid points are added to $\mathbf{v}^n$ and $\mathbf{v}^{n-1}$ using cubic interpolation according to

$$\begin{aligned} \mathbf{v}^n &:= \left[(\mathbf{v}^n)^T \quad I_3^n \mathbf{z}^n\right]^T, \\ \mathbf{v}^{n-1} &:= \left[(\mathbf{v}^{n-1})^T \quad I_3^n \mathbf{z}^{n-1}\right]^T, \end{aligned} \quad \text{if } N^n > N^{n-1}, \qquad (40)$$

where

$$\begin{aligned} \mathbf{z}^n &= \left[v_{M_v^{n-1}-1}^n \quad v_{M_v^{n-1}}^n \quad w_0^n \quad w_1^n\right]^T, \quad \text{and} \\ \mathbf{z}^{n-1} &= \left[v_{M_v^{n-1}-1}^{n-1} \quad v_{M_v^{n-1}}^{n-1} \quad w_0^{n-1} \quad w_1^{n-1}\right]^T, \end{aligned}$$

contain the values of the inner boundaries and their neighbours at n and $n-1$ respectively. Furthermore, cubic interpolator

$$I_3^n = \left[-\frac{\alpha^n(\alpha^n+1)}{(\alpha^n+2)(\alpha^n+3)} \quad \frac{2\alpha^n}{\alpha^n+2} \quad \frac{2}{\alpha^n+2} \quad -\frac{2\alpha^n}{(\alpha^n+3)(\alpha^n+2)}\right], \qquad (41)$$

with α^n as defined in Eq. (32). Notice that I_3^n is used for appending to both $\mathbf{v}^n$ and $\mathbf{v}^{n-1}$ and that M_v^{n-1} is used for indexing the left inner boundary for $\mathbf{z}^n$ and $\mathbf{z}^{n-1}$.

If $N^n < N^{n-1}$, grid points need to be removed from the system. This is done by simply removing the grid point at the left inner boundary as follows

$$\begin{aligned} \mathbf{v}^n &:= \left[v_1^n \quad \ldots \quad v_{M_v^{n-1}-1}^n\right]^T, \\ \mathbf{v}^{n-1} &:= \left[v_1^{n-1} \quad \ldots \quad v_{M_v^{n-1}-1}^{n-1}\right]^T, \end{aligned} \quad \text{if } N^n < N^{n-1}. \qquad (42)$$

An issue that occurs when removing grid points is that $v_{M_v^n}^n \not\approx w_0^n$ when $x_{v_{M_v^n}}^n \approxeq x_{w_0}^n$ at the time of removal. This violates the rigid connection in Eq. (28) and causes auditory artefacts. As mentioned in [18], if the system has damping terms, audible artefacts can be greatly reduced, as the difference between the inner boundaries is generally reduced too. As can be seen in the next section, a minimum value is placed on the frequency-dependent damping coefficient σ_1^n to reduce artefacts when removing grid points.

5. REAL-TIME IMPLEMENTATION

A real-time implementation of the dynamic stiff string has been created using C++ and the JUCE framework[4]. The application can be found online[5] as well as a video demonstration [22]. This section provides details on the implementation, including the update equations at the inner boundaries as well as the parameter ranges used, the graphical user interface (GUI) design, and the order in which the various equations from the previous sections are calculated.

5.1. Update Equations at the Inner Boundaries

Although one could potentially use a matrix library to implement Eq. (34) directly, it is generally more efficient to calculate the update equation in a loop. This requires Eq. (34) to be rewritten into its N^n separate update equations. If the right system has least two moving grid points, the update equations at the inner boundaries and their neighbouring points can be calculated as follows:

$$\begin{aligned}
A^n v_{M_v^n-1}^{n+1} &= \mathcal{B}_{0,\star}^n v_{M_v^n-1}^n + \mathcal{B}_{-1}^n v_{M_v^n-2}^n + \mathcal{B}_1^n v_{M_v^n}^n \\
&\quad - (\mu^n)^2 (v_{M_v^n-3}^n + w_0^n + \mathfrak{I}_3^n w_1^n) \\
&\quad + \mathcal{C}_{0,\star}^n v_{M_v^n-1}^{n-1} - S^n \left(v_{M_v^n}^{n-1} + v_{M_v^n-2}^{n-1}\right),
\end{aligned} \tag{43a}$$

$$\begin{aligned}
A^n v_{M_v^n}^{n+1} &= \mathcal{B}_0^n v_{M_v^n}^n + \mathcal{B}_{-1}^n v_{M_v^n-1}^n + \mathcal{B}_1^n w_0^n + \mathcal{B}_2^n w_1^n \\
&\quad - (\mu^n)^2 (v_{M_v^n-2}^n + \mathfrak{I}_3^n w_2^n) + \mathcal{C}_0^n v_{M_v^n}^{n-1} \\
&\quad - S^n (v_{M_v^n-1}^{n-1} + w_0^{n-1} + \mathfrak{I}_3^n w_1^{n-1}),
\end{aligned} \tag{43b}$$

$$\begin{aligned}
A^n w_0^{n+1} &= \mathcal{B}_0^n w_0^n + \mathcal{B}_{-1}^n w_1^n + \mathcal{B}_1^n v_{M_v^n}^n + \mathcal{B}_2^n v_{M_v^n-1}^n \\
&\quad - (\mu^n)^2 (w_2^n + \mathfrak{I}_3^n v_{M_v^n-2}^n) + \mathcal{C}_0^n w_0^{n-1} \\
&\quad - S^n (w_1^{n-1} + v_{M_v^n}^{n-1} + \mathfrak{I}_3^n v_{M_v^n-1}^{n-1}),
\end{aligned} \tag{43c}$$

$$\begin{aligned}
A^n w_1^{n+1} &= \mathcal{B}_{0,\star}^n w_1^n + \mathcal{B}_{-1}^n w_2^n + \mathcal{B}_1^n w_0^n \\
&\quad - (\mu^n)^2 (w_3^n + v_{M_v^n}^n + \mathfrak{I}_3^n v_{M_v^n-1}^n) \\
&\quad + \mathcal{C}_{0,\star}^n w_1^{n-1} - S^n \left(w_0^{n-1} + w_2^{n-1}\right),
\end{aligned} \tag{43d}$$

with A^n and S^n as defined in Sec. 4.3 and

$$\begin{aligned}
\mathcal{B}_0^n &= 2 + (\mathcal{I}^n - 2)((\lambda^n)^2 + S^n) - \mathfrak{I}_0^n (\mu^n)^2, \\
\mathcal{B}_{0,\star}^n &= 2 - 2(\lambda^n)^2 - 6(\mu^n)^2 - 2S^n, \\
\mathcal{B}_{-1}^n &= (\lambda^n)^2 + 4(\mu^n)^2 + S^n, \\
\mathcal{B}_1^n &= (\lambda^n)^2 - \mathfrak{I}_1^n (\mu^n)^2 + S^n, \\
\mathcal{B}_2 &= \mathfrak{I}_3^n (\lambda^n)^2 - \mathfrak{I}_2^n (\mu^n)^2 + \mathfrak{I}_3^n S^n, \\
\mathcal{C}_0^n &= \sigma_0^n k - (\mathcal{I}^n - 2) S^n - 1, \\
\mathcal{C}_{0,\star}^n &= \sigma_0^n k + 2S^n - 1.
\end{aligned} \tag{44}$$

All other grid points can be calculated using the original update equation shown in Eq. (11) (albeit with the time-varying parameters rather than the static ones). If the right system only has one moving grid point, Eq. (39) will have to be used in Eq. (34).

[4]https://juce.com

[5]https://github.com/SilvinWillemsen/RealTimeDynamic/releases/

5.2. Parameter Ranges

Table 1 shows the initial values of all parameters and their lower and upper limits. Most parameters range from half to double their respective initial value (denoted by a subscript 'i'). For Young's modulus E^n and frequency-independent damping σ_0^n it is possible to reduce their values to 0.

As done in [17], a limit was put on the maximum change in $\mathcal{N}^n$ per sample, referred to as $\mathcal{N}^n_{\text{maxdiff}}$, to prevent audible artefacts due to too large rates of change in parameters. This was implemented by rewriting Eq. (26) in terms of its respective variable and calculating the amount of change it would take to reach a change of $\mathcal{N}_{\text{maxdiff}}$. Here, $\mathcal{N}_{\text{maxdiff}} = 1/20$, together with the lower limit placed on σ_1^n, was heuristically found to prevent most noticable artefacts, while still allowing for high rates of change in parameters. Note that it is assumed that only one parameter is changed at a time. A possible solution that could be investigated in the future is to interleave multiple simultaneous parameter changes sample by sample.

Finally, $M_w^n = 1 \; \forall n$ such that M_v^n is dynamically changed according to Eq. (24).[6] Notice that the right system only has one moving grid point, and update equations at the inner boundaries in Eq. (43) have been adapted using matrix (39) in Eq. (34).

Table 1: *Parameter values and ranges. Subscript* 'i' *denotes the initial value of that parameter, which is defined in the third column.*

Parameter name	Symbol	Init. value	Lower lim.	Upper lim.
Length	L^n	1	$0.5 \cdot L_\text{i}$	$2 \cdot L_\text{i}$
Material density	ρ^n	7850	$0.5 \cdot \rho_\text{i}$	$2 \cdot \rho_\text{i}$
Radius	r^n	$5 \cdot 10^{-4}$	$0.5 \cdot r_\text{i}$	$2 \cdot r_\text{i}$
Tension	T^n	300	$0.5 \cdot T_\text{i}$	$2 \cdot T_\text{i}$
Young's modulus	E^n	$2 \cdot 10^{11}$	0	$2 \cdot E_\text{i}$
Freq.-indep. loss	σ_0^n	1	0	$2 \cdot \sigma_{0,\text{i}}$
Freq.-dep. loss	σ_1^n	$5 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$2 \cdot \sigma_{1,\text{i}}$

5.3. Graphical User Interface

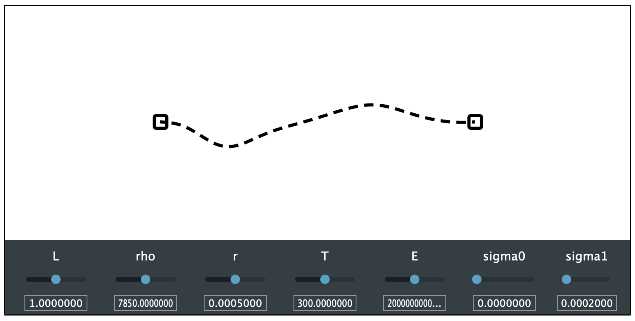
Figure 4 shows the GUI of the application. The bottom row contains sliders with which the user can change various parameters of the stiff string listed in Table 1. The top part shows the string using a dashed black line where the state of the system is visualised as a vertical displacement. The appearance of the string depends on the parameter values according to Table 2. Changes in loss coefficients are not directly visualised, but are implicitly shown by the time evolution of the string. The GUI is refreshed at a rate of 15 Hz. See [22] for a video demonstration.

The string can be excited by clicking the left mouse button in the top part, adding a raised cosine to $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ at the x-location of the mouse. The spacebar can also be used to trigger the excitation (using the last x-location of the mouse), so that the mouse can be used for slider interaction.

5.4. Order of Calculation

Algorithm 1 shows the order of calculation relating various instructions to the equations presented in previous sections. The output is retrieved at an arbitrary fixed location close to the left boundary ($l_v = 6$).

[6]It has been chosen to keep the n superscript in M_w^n to retain generality.



L	rho	r	T	E	sigma0	sigma1
1.0000000	7850.0000000	0.0005000	300.0000000	2000000000...	0.0000000	0.0002000

Figure 4: *The graphical user interface (GUI) of the application. The bottom part contains sliders that control various parameters (see Table 1). The top part shows the state of the string with the parameters visualised according to Table 2.*

Table 2: *Visualisation of parameters.*

Parameter name	Visual
Length	Length between boundaries
Material density	Length of dashes
Radius	Thickness of line
Tension	Rotation of boundaries
Young's modulus	Colour (blue - black - red)

6. RESULTS AND DISCUSSION

For the most part, the implementation of the dynamic stiff string works as it should. Parameters of the stiff can be changed in real time, and change the resulting timbre and pitch as expected. Informal testing by the authors confirms that for most settings, parameters can be changed from their minimum to their maximum value nearly instantaneously without producing noticable artefacts.

For some parameter settings and system states, however, auditory artefacts due to grid point removal are not completely prevented by the measures described in Section 5.2, i.e., the lower limit on σ_1^n and the use of $\mathcal{N}_{\text{maxdiff}}$. Increasing these values seems to help, but does not prevent artefacts in all situations. As discussed in [16], artefacts seem to appear when the highest mode is undamped at the moment that a grid point is removed. To combat the aforementioned issues, one could therefore potentially reintroduce the method of *displacement correction* as presented in [16], adding an artificial (damped) spring between the inner boundaries that damps higher modes in the process. A drawback here is that the damping is local and therefore unnatural as discussed in [18]. One could thus investigate other forms of frequency-dependent damping that specifically affect higher-frequency modes, such as adding another damping term $-2\sigma_2\partial_t\partial_x^4 u$ to Eq. (1) as described in [5, p. 216].

Another issue with the current implementation relates to the use of $\mathcal{N}_{\text{maxdiff}}$, which causes discrepancies between speed of variation between different settings of the parameters. For example, changes in L^n for a setting with a low value for h^n is much slower than for a setting with a high value of h^n, as there is a much higher value for $\mathcal{N}^n$ in the former than in the latter.

Related to the variation of $\mathcal{N}^n$ are the dramatic differences in computational cost for different parameter settings. At a sample rate of $f_s = 44100$ Hz, for the ranges of parameters presented

```
while application is running do
    Retrieve new slider values
    Calc. $h^n$ (Eq. (26))
    Calc. $\mathcal{N}^n$ and $N^n$ (Eqs. (23) and (17))
    Calc. $\alpha^n$ (Eq. (32))
    if $N^n \neq N^{n-1}$ then
        Add or remove point (Eq. (40) or (42))
        Update $M_v^n$ and $M_w^n$ (Eq. (24))
    end
    Calc. $v_{l_v}^{n+1}$ and $w_{l_w}^{n+1}$ (Eqs. (11) and (43))
    Retrieve output
    Update states ($\mathbf{u}^{n-1} = \mathbf{u}^n$, $\mathbf{u}^n = \mathbf{u}^{n+1}$)
    Update $N^{n-1}$ ($N^{n-1} = N^n$)
    Increment $n$
end
```

Algorithm 1: *Pseudocode showing the order of calculations.*

in Table 1, the number of grid points that needs to be calculated can range between $24 \leq N^n \leq 1590$ and the number of operations scales with N^n. Even though the application still runs in real time (between 1.2% and 11.6% without graphics on a MacBook Pro with a 2,3 GHz Intel i9 processor), it might cause auditory dropouts for specific settings when used in parallel with other plugins.

7. CONCLUSION

This paper presents a real-time implementation of the dynamic stiff string. Parameters of the underlying model can be varied in real time, mostly in a smooth fashion, though for some settings, auditory artefacts can be heard.

Apart from being able to create sounds that go beyond what is physically possible using this method, the implementation allows users to tweak the parameters of the model in real time. This paves the way for a drastically more efficient way of parameter tuning of FDTD-based physical models of musical instruments.

Future work includes investigating a reliable way to reduce audible artefacts due to grid point removal, as well as performing listening tests to confirm their absence. Finally, creating real-time implementations of other systems presented in [18], such as the dynamic thin plate, will allow for entire instruments to be manipulated, rather than only components in isolation.

8. ACKNOWLEDGMENTS

This work has been funded by the European Art-Science-Technology Network for Digital Creativity (EASTN-DC), project number 883023.

9. REFERENCES

[1] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, pp. 43–55, 1983.

[2] D. A. Jaffe and J. O. Smith, "Extensions of the karplus-strong

plucked-string algorithm," *Computer Music Journal*, vol. 7, pp. 56–69, 1983.

[3] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, G. De Poli, A. Picalli, and C. Roads, Eds., pp. 269–298. MIT Press, 1991.

[4] S. Mehes, M. van Walstijn, and P. Stapleton, "Towards a virtual-acoustic string instrument," *Proceedings of the 13th Sound and Music Computing Conference (SMC)*, 2016.

[5] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, John Wiley & Sons, 2009.

[6] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," M.S. thesis, University of Illinois, 1969.

[7] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 6, pp. 462–470, 1971.

[8] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 7, pp. 542–550, 1971.

[9] C. G. M. Desvages, *Physical Modelling of the Bowed String and Applications to Sound Synthesis*, Ph.D. thesis, The University of Edinburgh, 2018.

[10] S. Bilbao, M. Ducceschi, and C. Webb, "Large-scale real-time modular physical modeling sound synthesis," in *Proceedings of the 22th International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[11] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 161–168.

[12] S. Willemsen, *The Emulated Ensemble: Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods*, Ph.D. thesis, Aalborg University Copenhagen, Jul. 2021.

[13] R. Michon and J.O. Smith, "A hybrid guitar physical model controller: The BladeAxe," in *Proceedings ICMC|SMC|2014*, 2014.

[14] S. Willemsen, S. Serafin, and J. R. Jensen, "Virtual analog simulation and extensions of plate reverberation," in *Proc. of the 14th Sound and Music Computing Conference*, 2017, pp. 314–319.

[15] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, no. 3, pp. 484–512, 1984.

[16] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "Dynamic grids for finite-difference schemes in musical instrument simulations," in *Proc. 24th Int. Conf. Digital Audio Effects (DAFx)*, Vienna, Austria, 2021, pp. 144–151.

[17] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "A physical model of the trombone using dynamic grids for finite-difference schemes," in *Proc. 24th Int. Conf. Digital Audio Effects (DAFx)*, Vienna, Austria, 2021, pp. 152–159.

[18] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "The dynamic grid: Time-varying parameters for musical instrument simulations based on finite-difference time-domain schemes," *The Journal of the Audio Engineering Society (JAES)*, 2022.

[19] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*, Springer, 1998.

[20] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *Journal of the Acoustical Society of America (JASA)*, vol. 114, no. 2, pp. 1095–1107, 2003.

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

[22] S. Willemsen, "Demo of the Real-Time Dynamic Stiff String - DAFx2022," Available at https://youtu.be/8elYYOTWH3I, accessed, April 5, 2022.