APPLIED PHYSICAL MODELING FOR SOUND SYNTHESIS: THE YAYBAHAR

Pelle J. CHRISTENSEN[1], **Silvin WILLEMSEN**[1], and **Stefania SERAFIN**[1]

[1]*Multisensory Experience Lab, CREATE*, **Aalborg University**, Copenhagen, Denmark

ABSTRACT

In this paper, finite-difference time-domain methods are adopted to model a specific instrument, the Yaybahar, invented by Turkish artist Görkem Şen. Each part of the instrument is simulated independently and its physical behavior is explained in an intuitive yet accurate manner. The models are implemented in C++ to form an interactive, real-time application. Code and sound samples are available online.

1. INTRODUCTION

The Yaybahar is a novel acoustical instrument invented by Turkish artist Görkem Şen. It consists of a string connected to two drum heads via a couple of springs as seen in Figure 1. It is played by bowing the string and striking the springs and drums using a mallet. The combination of the springs and the drums create a reverberant sound that is unusual for acoustical instruments. A video of Görkem Şen playing the Yaybahar can be viewed on YouTube [1] .

This paper presents a real-time sound synthesis algorithm that mimics the Yaybahar by modeling its physical behaviour. To accomplish this we employ finite-difference time-domain (FDTD) methods in the form of finite difference schemes (FDSs).

The basic concept of FDTD is to describe our system as a set of differential equations, which are then descritized to form a FDS, which may then be implemented. In the descritization we use various *finite difference operators* [1, Chapter 2.2]. For example, the difference operators

$$\delta_{t-}u^n = \frac{1}{k}(u^n - u^{n-1}) \quad \delta_{t+}u^n = \frac{1}{k}(u^{n+1} - u^n) \quad (1)$$

where k is the sample period, compute an approximation of the first order time derivative of u, and

$$\delta_{tt}u^n = \delta_{t+}\delta_{t-} = \frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) \quad (2)$$

approximates the second order time derivative.

To exemplify FTDT methods we take the simple harmonic oscillator, described by the differential equation

$$u_{tt} = -\omega_0^2 u \quad (3)$$

[1] https://youtu.be/_aY6TxC1ojA

Figure 1. A screenshot of a video[1] of Görkem Şen playing the Yaybahar by bowing the string.

and descritize it by replacing the derivative with a suitable difference operator to get the FDS

$$\delta_{tt}u = -\omega_0^2 u. \quad (4)$$

If we expand the difference operator

$$\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) = -\omega_0^2 u \quad (5)$$

we can isolate u^{n+1} to get our *update rule*

$$u^{n+1} = -k^2\omega_0^2 u + 2u^n - u^{n-1} \quad (6)$$

which we can use to compute the state of the system at the next time step from the current and previous state.

The application of finite difference schemes (FDSs) to sound synthesis is quite mature, see e.g. *Numerical Methods for Sound Synthesis* [1] for an in-depth examination of the topic. We assume the reader is familiar with the basic concepts and notation found in [1].

Despite their maturity, FDSs have not been readily applied to real-time synthesis because they are computationally demanding. This is especially true for non-linear systems. One example of a real-time FDS implementation can be found in [2], which also covers expressive real-time interaction with the model. While this paper does not expand upon the theory of FDSs, it does show how the theory may be applied in the hope that it may be used as a pedagogical resource or as a starting point for newcomers to the topic.

In Section 2, we examine all the elements of the instrument and discuss their physics in an intuitive manner, and show how they may be discretized and connected to one

another. In Section 2.6 we touch upon the important topic of stability. Then, in Section 3 we link to the C++ application and discuss the optimizations that were needed to get everything running in real-time. Lastly, we conclude the paper in Section 4.

2. THE ELEMENTS OF THE YAYBAHAR

In this section we will go through each element of the Yaybahar individually: the string, springs, drum heads, the bow, and the connections between those, and cover their physics and how to simulate them. A block diagram of the Yaybahar model can be viewed in Figure 2. An overview of all parameters (and their units) used for the models presented in this section is shown in Table 1.

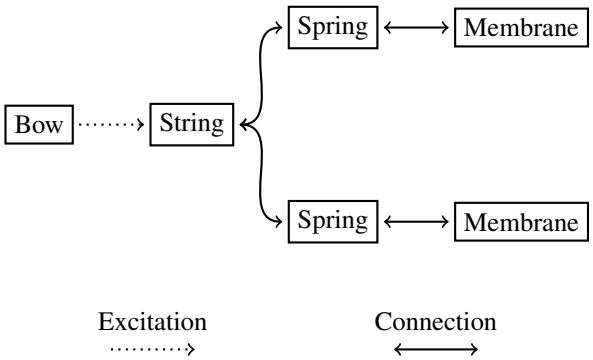

Figure 2. Block diagram of the Yaybahar. A bowed string is connected to two springs that connect to two drum heads.

2.1 String

Parameter	Description	Unit
u	Displacement	m
ρ	Material density	kg·m^{-3}
T_0	Tension	N
E	Young's modulus	Pa
A	Area	m^2
L	Length	m
I	Area moment of inertia	m^4
σ_0	Freq. independent damping	s^{-1}
σ_1	Freq. dependent damping	m^2·s^{-1}
κ	Spring dispersion parameter	m^4·s^{-2}
M	Mass	kg
μ	Linear density	kg·m^{-1}
H	Membrane thickness	m
v_b	Bow speed	m·s^{-1}
K	Spring constant	kg·s^{-2}

Table 1. Table of the parameters of the elements of the Yaybahar.

To model the string of the Yaybahar we will use the equations found in [1, Chapters 6,7, and 8], [3], and [4]. A nonlinear damped stiff string with externally supplied forces is defined by the partial differential equation

$$u_{tt} = \frac{1}{\rho A}\left(T_0 + \frac{EA}{2L}\int_0^L u_x^2 dx\right)u_{xx} - \frac{EI}{\rho A}u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx} + \frac{1}{\rho A}\sum_f \delta(x - x_f)F_f\ , \tag{7}$$

where $u = u(x,t)$ is the displacement of the string at point x and time t. See Table 1 for a description of all the parameters. The last term is the sum of external forces F_f acting on the string, localized at positions x_f along the string via the Dirac delta function $\delta(x - x_f)$.

The first term of (7) is the force arising due to the tension of the string. We see that this force is determined by u_{xx}, the curvature of the string, a result that is arrived at by analysing the tension forces acting on an infinitesimal section of the string, see e.g. [5, Chapter 2] or [1, Chapter 6].

We can understand what this means by looking at Figure 3, where the solid line is the displacement of the string and u_{xx} is indicated with the dashed line. Notice that u_{xx} is somewhat opposite of u, meaning that the string will be pulled towards the middle (as one would expect).

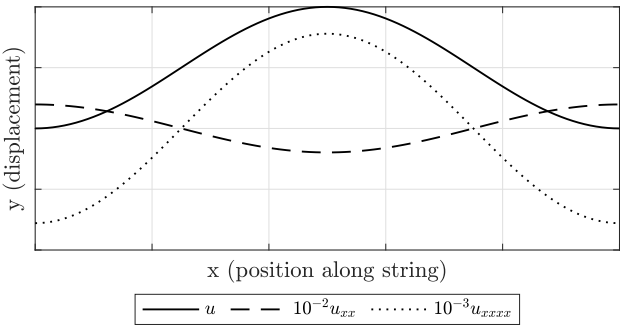

Figure 3. Plot of a part of a string $u = u(x,t)$ (solid line) , its second derivative u_{xx} (dashed line), which determines the force due to tension, and u_{xxxx} (dotted line), which determines the force due to stiffness. u_{xx} and u_{xxxx} has been scaled for them to fit in the plot.

The curvature is scaled by $T_0 + \frac{EA}{2L}\int_0^L u_x^2 dx$. The inclusion of T_0 means that a higher tension will result in a stronger restoring force and thus higher pitch. The integral can be interpreted as the increase in tension that occurs at large displacements [5, Chapter 5]. This give rises to phenomena such as downwards pitch glides which can be experienced by strongly plucking e.g. a guitar string.

The next term is the one involving u_{xxxx}, which is an expression of the force due to the stiffness of the string material. A derivation of this can be found in [5, Chapter 2]. Looking at Figure 3 we see that u_{xxxx} points in the same direction as u. The minus sign in front of the term in equation (7) means that the stiffness will somewhat counteract the displacement of the string trying to make it straight. Stiffness causes dispersion which gives rise to inharmonic overtones, which can be observed in e.g. a low frequency piano string or metal rods.

The term $-2\sigma_0 u_t$ causes damping of the system that, greatly simplified, occurs due to viscous air flow, internal

friction, and radiated energy [5, Chapter 2]. u_t is the velocity of u, and is a scalar pointing in the direction that u is currently moving. Thus, adding a force opposite of this will cause damping. Higher velocities will cause a stronger damping force which results in a exponential decrease in volume over time.

The term $2\sigma_1 u_{txx}$ causes frequency dependent damping of the string. Intuitively we expect u_{xx} to be larger when the string is vibrating at higher frequencies and thus cause a stronger damping. The physical justification for this term is weak, but it causes the correct perceptual phenomena of higher frequencies dying out more quickly [4].

Proceeding with the discretization of Equation (7), we replace the derivative operators with suitable finite difference operators to get

$$\delta_{tt} u_l^n = \frac{1}{\rho A}\left(T_0 + \sum_{i=0}^{N-1} (\delta_{x+} u_i^n)^2\right) \delta_{xx} u_l^n - \frac{EI}{\rho A}\delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n + \frac{1}{\rho A}\sum_f J_l(x_f) F_f. \tag{8}$$

In addition to time we have also descritized space by introducing a number of grid points N with grid spacing h, where l is the grid point under consideration. The function $J_l(x_f)$, the discrete counterpart of $\delta(x - x_f)$, is a spreading operator scaling F_f by $1/h$ whenever $\lfloor x_f/h \rfloor = l$ and by 0 otherwise [1, Chapter 5].

We now derive the update rule for the spring by expanding all operators that contain u_l^{n+1} and then proceed isolate u_l^{n+1} to get

$$u_l^{n+1} = \frac{1}{1 + k\sigma_0}(F_\mathrm{t} + F_\mathrm{s} + F_\mathrm{d} + F_\mathrm{e} + 2u_l^n - u_l^{n-1}) \tag{9}$$

where

$$F_\mathrm{t} = \frac{k^2}{\rho A}\left(T_0 + \sum_{i=0}^{N-1} h(\delta_{x+} u_i^n)^2\right)\delta_{xx} u_l^n \tag{10}$$

$$F_\mathrm{s} = -\frac{k^2 EI}{\rho A}\delta_{xxxx} u_l^n \tag{11}$$

$$F_\mathrm{d} = k\sigma_0 u_l^{n-1} + 2k\sigma_1(\delta_{xx} u_l^n - \delta_{xx} u_l^{n-1}) \tag{12}$$

$$F_\mathrm{e} = \frac{k^2}{\rho A}\sum_f J_l(x_f) F_f. \tag{13}$$

Notice that we did not expand the spatial difference operators because they have no influence on the isolation of u_l^{n+1}.

For implementation purposes to bound the domain of our string by choosing an integer N to be the number of points on our string, evenly spaced by h. Assuming that the length of our string is $L = 1$ we have $N = 1/h$.

Special care needs to be taken at the boundaries — the points $l = 0$ and $l = N - 1$. There are multiple ways of doing this depending on what behavior we want. We will use a clamped boundary condition, which means that the points at each end must satisfy [1, Chapter 7]

$$u = u_x = 0, \tag{14}$$

which can be implemented by introducing *virtual points* at positions -1 and N.

To implement $u = 0$ in Equation (14) we set the virtual points

$$u_0 = u_{N-1} = 0. \tag{15}$$

To implement $u_x = 0$ on the left end of the string we apply $\delta_{t\cdot}$ at $l = 0$ to get

$$\delta_{t\cdot} u_0^n = \frac{1}{2h}(u_1^n - u_{-1}^n) = 0, \tag{16}$$

which means that setting the virtual point

$$u_{-1}^n = u_1^n, \tag{17}$$

will satisfy the condition. The same is done for the right end which gives us $u_N^n = u_{N-2}^n$.

2.2 Spring

The springs that connect the string to the drum membranes are modeled using [6]

$$u_{tt} = -\kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx} + \frac{1}{\mu}\sum_f \delta(x - x_f) F_f\ . \tag{18}$$

In this model κ is a constant that determines the dispersive behaviour of the spring [3] [6]. $\mu = L/M$ is the linear density of the spring where L is the length and M the total mass of the spring.

Equation (18) actually describes the dynamics of a linear stiff bar [3] [1, Chapter 7]. However, it was used by Parker, Penttinen, and Bilbao to model wave propagation on a slinky — a long, large diameter spring — so it should be suitable for modeling the springs of the Yaybahar as well [6]. More accurate models of springs exist, see e.g. [7], but the one used here has the benefit of simplicity and ease of implementation for real-time purposes.

The terms of Equation (18) should look familiar as they are also found in Equation (7) and explained in Section 2.1. Note there is no force due to tension — the only restoring force in the spring is the material stiffness.

We discretize Equation (18) by applying the suitable difference operators to get

$$\delta_{tt} u_l^n = -\kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n + \frac{1}{\mu}\sum_f J_l(x_f) F_f\ . \tag{19}$$

Then, expanding the operators and isolating u_l^{n+1} we get

$$u_l^{n+1} = \frac{1}{1 + k\sigma_0}(F_\mathrm{s} + F_\mathrm{d} + F_\mathrm{e} + 2u_l^n - u_l^{n-1}) \tag{20}$$

where

$$F_\mathrm{s} = -k^2\kappa^2 \delta_{xxxx} u_l^n \tag{21}$$

$$F_\mathrm{d} = k\sigma_0 u_l^{n-1} + 2k\sigma_1(\delta_{xx} u_l^n - \delta_{xx} u_l^{n-1}) \tag{22}$$

$$F_\mathrm{e} = \frac{k^2}{\mu}\sum_f J_l(x_f) F_f. \tag{23}$$

At each end, a spring is attached either to the string or to a drum head. We will model this by using free boundary conditions and the connections covered in Section 2.5. Free boundary conditions mean that at either end of the spring we must satisfy [1]

$$u_{xx} = u_{xxx} = 0. \tag{24}$$

This is implemented by first discretizing Equation (24) using finite difference operators to get

$$\delta_{xx} u_l^n = \delta_{x-}\delta_{xx} u_l^n = 0. \tag{25}$$

Expanding Equation (25) for $l = 0$ we get

$$\frac{1}{h^2}(u_{-1}^n - 2u_0^n + u^1) = 0 \tag{26}$$

$$\frac{1}{h^3}(-u_{-2}^n + 3u_{l-1}^n - 3u_0^n - u_1^n) = 0. \tag{27}$$

we then take Equation (26) and solve for the first virtual point u_{-1}^n to get

$$u_{-1}^n = 2u_0^n - u_1^n. \tag{28}$$

This is inserted into Equation (27) to solve for u_{-2}^n

$$u_{-2}^n = 3u_0^n - 2u_1^n. \tag{29}$$

Doing the same for the other end of the spring we get

$$u_N^n = 2u_{N-1}^n - u_{N-2}^n \tag{30}$$

$$u_{N+1}^n = 3u_{N-1}^n - 2u_{N-2}^n, \tag{31}$$

2.3 Membrane

The drum heads of the Yaybahar can be modeled using a damped 2D wave equation with a non-linearity in the tension term, based on the Berger plate model [1, Chapter 11 and 13]

$$\begin{aligned} u_{tt} = &\frac{1}{\rho H}\left(T_0 + \frac{6D}{AH^2}\iint_{\mathcal{D}} (\nabla u)^2 dxdy\right)\Delta u \\ &- 2\sigma_0 u_t + 2\sigma_1 \Delta u_t \\ &+ \frac{1}{\rho H}\sum_f \delta(x - x_f, y - y_f) F_f \ . \end{aligned} \tag{32}$$

where $u = u(x, y, t)$ is the displacement of point (x, y) of the membrane at time t. See table 1 for a description of the parameters. $\delta(x - x_f, y - y_f)$ is the (2D) spatial Dirac delta function localizing force F_f at coordinate (x_f, y_f). Furthermore, $\mathcal{D} \in [0, L_x] \times [0, L_y]$ denotes the domain of the membrane with membrane width L_x and height L_y. The parameter D is defined by

$$D = \frac{EH^3}{12(1 - \nu^2)}, \tag{33}$$

which is the *flextural regidity* of the membrane.

Notice that Equation (32) is similar to (7) except now in two dimensions. In fact, the forces in both equation arise due to the same phenomena. Notice also that our membrane has no stiffness, because we assume that it is made from some non-stiff material such as hide or plastic.

Because we are now working with a 2D system we need to be aware of spatial derivatives in two directions. This is why we have the gradient, defined as

$$\nabla u = u_x + u_y, \tag{34}$$

and the Laplacian, defined as

$$\Delta u = u_{xx} + u_{yy}. \tag{35}$$

At this point we are assuming cartesian coordinates, however, since the drum heads of the Yaybahar are round, we could want to make the domain of our model circular. However, FDSs using radial coordinates, especially the efficient, *explicit* ones, have a tendency to misbehave [1, Chapter 12]. The main problem in our case is that the sound will be severely band limited, which will kill the tone of our assembled Yaybahar model, so we will keep using cartesian coordinates and square drum heads.

Equation (32) can be discretized like

$$\begin{aligned} \delta_{tt} u_{l,m}^n = &\frac{1}{\rho H}\left(T_0 + \frac{6D}{|\mathcal{D}|H^2}\sum_{l=0}^{N_x-1}\sum_{m=0}^{N_y-1} h_x h_y (\delta_\nabla u_{l,m}^n)^2\right) \\ &\cdot \delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \delta_\Delta \delta_{t-} u_{l,m}^n \\ &+ \frac{1}{\rho H}\sum_f J_{l,m}(x_f, y_f) F_f, \end{aligned} \tag{36}$$

where $J_{l,m}(x_f, y_f)$ is a (2D) spreading operator that scales F_f by $1/(h_x h_y)$ whenever $\lfloor x_f/h_x \rfloor = l \wedge \lfloor y_f/h_y \rfloor = m$ and by 0 otherwise. $N_x = 1/h_x$ and $N_y = 1/h_y$ are the number of horizontal and vertical points of the discrete membrane respectively. Lastly, we define the following [3, Chapter 11]

$$\delta_\Delta u_{l,m}^n = \delta_{xx} u_{l,m}^n + \delta_{yy} u_{l,m}^n, \tag{37}$$

$$\delta_\nabla u_{l,m}^n = \delta_{x+} u_{l,m}^n + \delta_{y+} u_{l,m}^n. \tag{38}$$

Expanding the operators in Equation (36) and solving for $u_{l,m}^{n+1}$ we get

$$u_{l,m}^{n+1} = \frac{1}{1 + k\sigma_0}(F_\mathrm{s} + F_\mathrm{d} + F_\mathrm{e} + 2u_{l,m}^n - u_{l,m}^{n-1}) \tag{39}$$

where

$$\begin{aligned} F_\mathrm{s} = &\frac{k^2}{\rho H}\left(T_0 + \frac{6D}{|\mathcal{D}|H^2}\sum_{l=0}^{N_x-1}\sum_{m=0}^{N_y-1} h_x h_y (\delta_\nabla u_{l,m}^n)^2\right) \\ &\cdot \delta_\Delta u_{l,m}^n \end{aligned} \tag{40}$$

$$F_\mathrm{d} = k\sigma_0 u_{l,m}^{n-1} + 2k\sigma_1(\delta_\Delta u_{l,m}^n - \delta_\Delta u_{l,m}^{n-1}) \tag{41}$$

$$F_\mathrm{e} = \frac{k^2}{\rho H}\sum_f J_{l,m}(x_\mathrm{f}, y_\mathrm{f}) F_f. \tag{42}$$

As for boundary conditions, we now have the four edges around our membrane we need to take care of [2] . Like the

[2] Even though we are using cartesian coordinates the shape of our membrane does not need to be rectangular, but for simplicity's sake we keep it that way

string we use a clamped boundary conditions which are now defined by

$$u = \nabla u = 0, \tag{43}$$

when $l = 0$, $l = N_x - 1$, $m = 0$, or $m = N_y - 1$. Implementation is analogous to the 1D case.

2.4 Bow

The bow using to play the Yaybahar excites the instrument through the frictional interaction between the bow hairs and the string. At first, the bow pulls the string along, but at some point, the restoring force in the string becomes too large and the string bounce back only to be caught by the bow again [5, Chapter 2].

We model this by using the simple simulation from [1, Chapter 4], where we apply a force F to our system in Equation (7) that is determined by

$$F = -F_\text{b}\phi(v_\text{rel}), \tag{44}$$

where

$$v_\text{rel} = u_t - v_\text{b}. \tag{45}$$

Here $F_\text{b} = F_\text{b}(t)$ is the force in Newtons at which the bow is pressed down onto the string, $v_\text{rel} = v_\text{rel}(t)$ is the relative velocity between the bow and the string at the bowing point with $v_\text{b} = v_\text{b}(t)$ being the bow velocity in m/s.

There are many ways of defining the dimensionless friction characteristic ϕ, but we will be using the definition [1, Chapter 4.3]

$$\phi(v_\text{rel}) = \sqrt{2a}v_\text{rel}e^{-av_\text{rel}^2+1/2}, \tag{46}$$

where a is a dimensionless constant that determines the shape of the function.

To implement our bow model we must discretize Equation (44) and thus v_rel. We do this using the scheme

$$v_\text{rel}^n = \delta_{t\cdot}u_{l_\text{b}}^n - v_\text{b}^n, \tag{47}$$

where l_b is the index of our bowing point.

If we expand Equation (47) we get

$$v_\text{rel}^n = \frac{1}{2k}\left(u_{l_\text{b}}^{n+1} - u_{l_\text{b}}^{n-1}\right) - v_\text{b}^n. \tag{48}$$

Notice that v_rel^n depends on $u_{l_\text{b}}^{n+1}$, but to compute $u_{l_\text{b}}^{n+1}$ using Equation (7) combined with Equation (47) we need v_rel^n. This cyclic dependency results in what we call an *implicit* scheme which must solve using *iterative methods*. One such method is the Newton-Raphson algorithm which states that given a function $f(x) = 0$, then we can find x by iteratively calculating

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}, \tag{49}$$

with iteration number i and $f'(x) = df(x)/dx$, where we keep iterating until the change between x^{i+1} and x^i is sufficiently small.

We apply this by first putting Equation (48) into the correct from

$$f(v_\text{rel}^n) = \frac{1}{2k}\left(u_{l_\text{b}}^{n+1} - u_{l_\text{b}}^{n-1}\right) - v_\text{b} - v_\text{rel}^n = 0. \tag{50}$$

Then we substitute $u_{l_\text{b}}^{n+1}$ using Equation (9) and take the derivative of it all with respect to v_rel^n to get

$$f'(v_\text{rel}^n) = -\frac{1}{2k}k^2F_\text{b}\phi'(v_\text{rel}^n) - 1. \tag{51}$$

where

$$\phi'(v_\text{rel}^n) = \sqrt{2a}\left(e^{-a(v_\text{rel}^n)^2+1/2} - 2a(v_\text{rel}^n)^2e^{-a(v_\text{rel}^n)^2+1/2}\right). \tag{52}$$

And thus, we have everything we need to iteratively compute v_rel^n.

2.5 Connections

To connect the elements of our Yaybahar we use a simple spring-like [3] connection. If $u_{l_\text{uc}}^n$ is the connecting point on our string and $w_{l_\text{wc}}^n$ is the connecting point on one of the springs, the force caused by the connection between them is defined by

$$F_\text{c} = K\left(u_{l_\text{uc}}^n - w_{l_\text{wc}}^n\right), \tag{53}$$

where K is a positive constant determining the strength of the connection. This connection scheme represents Hooke's law, and is similar the one found in [3], where they also include nonlinear terms.

After computing the connection force F_c we simply apply it to our string and spring, but in opposite directions, so if F_c is applied to the string, then $-F_\text{c}$ is applied to the spring and added to their respective F (external forces) in Equations (7) and (18).

2.6 Stability

An important topic we have not touched upon in this paper is stability — a property that is not guaranteed for FDSs. To ensure stability, the choice of parameters for each model must satisfy a stability condition. The procedure is usually to choose one's desired parameters and then compute a suitable value for h. For the schemes for the string and membranes, the stability condition is not known exactly, however, they should at least satisfy the stability conditions for their linear counterparts. For the string this is [2]

$$h \geq \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2k^2}}{2}} \tag{54}$$

where $c = \sqrt{T_0/\rho A}$ and $\kappa = \sqrt{EI/\rho A}$. For the membrane the stability condition is [1, Chapter 11]

$$h \geq \sqrt{2}k\frac{T_0}{\rho H}, \tag{55}$$

assuming that $h = h_x = h_y$.

In both cases, it is practical to choose an h slightly larger (e.g. $1.05 \cdot h$) than the one given by the stability conditions to avoid instability due to the non-linearity, and in the case of the membrane, the missing σ_1 terms.

[3] Spring-like as in mass-spring system. Not like the model in 2.2

For the spring scheme the stability condition is exactly [3]

$$h \geq \sqrt{2k\left(\sigma_1^2 + \sqrt{\kappa^2 + \sigma_1^2}\right)}. \tag{56}$$

3. IMPLEMENTATION

The equations described in Section 2 are implemented in C++ and are available on GitHub [4] where you can also find a sound sample.

Due to the computational complexity of the models presented it is necessary to write performant code to accomplish real-time sound synthesis. See [8] for a discussion of the complexity and implementation strategies of FDS algorithms.

On of the most important implementation tricks is the use of reference swapping. A naive implementation of e.g. the string model would have three state arrays u, un, and up representing the current, next, and previous states. After computing un using the update rule it would copy the values of the arrays like so:

```
for (int i = 0; i < N; i++)
{
  up[i] = u[i];
  u[i] = un[i];
}
```

The actual implementation uses C++ references in a loop so memory coping is avoided. The three arrays u, un, and up are now references instead, so we can simply do

```
  auto &uswap = up;
  up = u;
  u = un;
  un = uswap;
```

which is nearly instant.

4. CONCLUSION

In this paper we have covered how to build a physical modeling sound synthesis algorithm for a specific instrument: the Yaybahar.

We went through each element of the instrument, a bowed string, springs, and drum membranes, discussed their physics, and showed to how implement numerical simulations of them using FDSs. After that, we showed how they can be connected using an equation reminiscent of Hooke's law.

Stability conditions were not given for the string and membrane models, but we argue that the conditions for the linear variants of the equations can be used by choosing h to be slightly larger.

For the implementation, we linked to the code hosted on GitHub and discussed the importance of using reference swapping in the implementation of the models.

[4] https://github.com/pellejuul/yaybahar

5. REFERENCES

[1] S. Bilbao, *Numerical Sound Synthesis*. Chichester, UK: John Wiley & Sons, Ltd, Oct. 2009. [Online]. Available: http://doi.wiley.com/10.1002/9780470749012

[2] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," in *Proceedings of the 16th Sound and Music Computing Conference*, Malaga, May 2019, pp. 151 – 158.

[3] S. Bilbao, "A Modular Percussion Synthesis Environment," *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, 2009.

[4] 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," *The Journal of the Acoustical Society of America*, vol. 114, no. 2, pp. 1095–1107, Aug. 2003. [Online]. Available: http://asa.scitation.org/doi/10.1121/1.1587146

[5] N. H. Fletcher and T. D. Rossing, *The physics of musical instruments*, 2nd ed. New York: Springer, 1998.

[6] J. Parker, H. Penttinen, S. Bilbao, J. S. Abel, and T. L. Crew, "Modeling methods for the highly dispersive Slinky spring: A novel musical toy," in *Proc. of the 18th Int. Conference on Digital Audio Effects*, 2010, p. 4.

[7] J. Parker and S. Bilbao, "Spring Reverberation: A Physical Perspective," in *Proc. of the 12th Int. Conference on Digital Audio Effects*, 2009, p. 6.

[8] C. J. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proc. of the 18th Int. Conference on Digital Audio Effects*, 2015, p. 9.