Embedded physical modelling modular synthesizer based on FDTD methods

Benjamin Fullerton Støier
Aalborg University, Copenhagen
`bstoie21@student.aau.dk`

ABSTRACT

This paper describes the implementation and features of a modular synth that utilises physical modelling synthesis. The physical models are based on finite-difference time domain (FDTD) methods and consist of a damped stiff string and an acoustic tube. Several exciters including a bow, strike, pluck and lip reed have also been implemented. Furthermore, a dynamic grid has been utilised, which allows for smooth variations of the defining physical properties of the models.

1. INTRODUCTION

In recent decades, significant progress has been made in emulating real-world musical instruments with various physical modeling techniques. As a result, many physical modeling techniques exist today, such as modal synthesis, digital wave guides and mass-spring systems. This project utilises the finite-difference time domain (FDTD) methods technique first used in a musical context by Ruiz in [1]. FDTD methods are numerical techniques used to solve partial differential equations (PDEs). They involve approximating the PDEs with difference equations and discretising the continuous system into a grid points in space and time. FDTD methods are by no means the most efficient form of physical modelling. The methods are however extremely general and flexible in modeling various systems. The methods allow for direct numerical simulation of any set of PDE's without making assumptions of modes or traveling wave solutions [2]. Another benefit of FDTD methods is that you gain direct access to the physical properties of the model. In recent years real-time applications utilising FDTD methods have gained prominence due to higher computational power being available to the general public [2]. However, still to this day the vast majority of the real-time applications utilising FDTD are limited to plugins running in a DAW, and all commercially available embedded physical modelling based synthesizers still utilise less demanding methods such as modal synthesis and digital wave guides. As far as the author is concerned, this is the first (including state of the art) embedded modular synthesizer utilising FDTD methods.

One of the issues with the FDTD methods is that the method lacks the capability of handling smooth parameter changes while retaining stability and optimal simulation quality. Recently, a method known as "the dynamic grid" has been introduced in order to mitigate this problem. The dynamic grid was first introduced by Willemsen et al. in [3] where it was applied to the 1D wave equation. Later, the method has been utilised to model a trombone in [4] and for more complex higher-order systems in [5,6].

One of the interesting aspects of physical modelling synthesis is that you can exploit the virtual aspect of the simulation and create sounds that would not be physical possible in the real word. With the dynamic grid it is possible to change material properties and the geometry of a physical resonator in real time, thus allowing the user to create sounds that would otherwise be impossible. As the physical models are embedded in the form factor of a modular synth, it takes this aspect of experimenting with non-realistic sounds even further, since it makes it easy to modulate multiple physical properties. The modular aspect of the synthesizer allows it to be patched together with other modular gear that the user might posses, thereby making it possible to be integrated into a larger system.

This paper will start out with presenting the necessary theory for implementing the psychical models. Section 2 introduces the FDTD methods. Section 3 presents the damped stiff string model. Section 4 presents the acoustic tube model. Section 5 presents models related to the exciters. Section 6 presents the dynamic grid method. Section 7 presents how the models have been implemented in real-time, as well as the design considerations and the additional features of the final synth. Section 8 discusses the outcome of this project and potential improvements. Lastly, Section 9 gives concluding remarks on the project.

2. FDTD METHODS

FDTD methods was utilised for this project in order to approximate continuous physical systems by subdividing them in to discrete points in time and space. The theory in this section is based on [7, Ch. 2] unless otherwise noted.

Consider a PDE that describes a continuous 1D system with state variable, $u(x,t)$. The system has a static length L (in m), a spacial domain is defined for $x \in [0, L]$, and a temporal range defined for $t \geq 0$. To discretise the system, we can represent $u(x,t)$ as a grid function u_l^n with spacial index $l \in \{0, ..., N\}$ and temporal index $n \in \mathbb{N}^0$. When performing the discretisation, the spacial part of the continuous system is divided into $N = \text{floor}(L/h)$ grid points with grid spacing h (in meters). Similarly, discretisation of the temporal part of the system is achieved by setting $t = nk$, where $k = 1/fs$ is the time step (in seconds) and f_s is the sampling frequency (in Hz). Shift operators, denoted by e_s of type s, can be used to shift the temporal or spatial index of a grid function. The forward and backward temporal shift operators are defined as

$$e_{t+}u_l^n = u_l^{n+1}, \quad \text{and} \quad e_{t-}u_l^n = u_l^{n-1} \tag{1}$$

and the spacial shift operators are

$$e_{x+}u_l^n = u_{l+1}^n, \quad \text{and} \quad e_{x-}u_l^n = u_{l-1}^n \tag{2}$$

The shift operators are instrumental in defining the finite difference (FD) operators, which can be used for discretising first order derivatives in time and space. The FD operators can be used in three different ways, either as forward, backwards or centered operators. The FD operators used for discretising first order derivatives in time are defined as follows

$$\partial_t \approxeq \begin{cases} \delta_{t+} \triangleq \frac{1}{k}(e_{t+} - 1) \\ \delta_{t-} \triangleq \frac{1}{k}(1 - e_{t-}) \\ \delta_{t\cdot} \triangleq \frac{1}{2k}(e_{t+} - e_{t-}) \end{cases} \tag{3}$$

and the FD operators used for discretising first order derivatives in space are

$$\partial_x \approxeq \begin{cases} \delta_{x+} \triangleq \frac{1}{h}(e_{x+} - 1) \\ \delta_{x-} \triangleq \frac{1}{h}(1 - e_{x-}) \\ \delta_{x\cdot} \triangleq \frac{1}{2h}(e_{x+} - e_{x-}) \end{cases} \tag{4}$$

Multiplying first order FD operators can yield approximations for higher order differences. Second order derivatives in time can be approximated as

$$\partial_t^2 \approxeq \delta_{t+}\delta_{t-} = \delta_{tt} \triangleq \frac{1}{k^2}(e_{t+} - 2 + e_{t-}) \tag{5}$$

and similarly second-order derivatives in space are approximated using the following operator

$$\partial_x^2 \approxeq \delta_{x+}\delta_{x-} = \delta_{xx} \triangleq \frac{1}{h^2}(e_{x+} - 2 + e_{x-}) \tag{6}$$

3. DAMPED STIFF STRING

This section describes how a damped stiff string can be modelled using FDTD methods. The theory in section is based on [7] unless otherwise noted.

A PDE governing the motion of a damped stiff string can be expressed as follows

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa\partial_x^4 u - 2\sigma_0\partial_{t\cdot}u + 2\sigma_1\partial_{t-}\partial_x^2 u \tag{7}$$

with wave speed $c = \sqrt{T/\rho A}$ (in m/s), tension T (in N), material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), stiffness coefficient $\kappa = \sqrt{EI/\rho A}$ (in m^2/s), Young's modulus E (in Pa), area moment of inertia $I = \pi r^4/4$ (in m^4), frequency-independent damping σ_0 (in s^{-1}) and frequency-dependent damping σ_1 (in m^2/s).

Discretisng the PDE in equation (7) yields

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n \tag{8}$$

and by expanding the FD operators one can derive at the following update equation

$$\begin{aligned}(1+\sigma_0 k)u_l^{n+1} = &\left(2 - 2\lambda^2 - 6\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u_l^n \\ &+ \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)(u_{l+1}^n + u_{l-1}^n) \\ &- \mu^2(u_{l+2}^n + u_{l-2}^n) \\ &+ \left(-1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}\right)u_l^{n-1} \\ &- \frac{2\sigma_1 k}{h^2}(u_{l+1}^{n-1} + u_{l-1}^{n-1})\end{aligned} \tag{9}$$

where

$$\lambda = \frac{ck}{h} \tag{10}$$

and

$$\mu = \frac{\kappa k}{h^2} \tag{11}$$

The stability condition for the scheme can be derived through von Neumann stability analysis as done in [2, Ch. 4] resulting in

$$h \geq h_{\text{min}} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}} \tag{12}$$

´ A simply supported boundary condition was used for the stiff string in this project. This boundary condition is defined as

$$u = \partial_x^2 u = 0, \quad \text{at} \quad x = 0, L \tag{13}$$

and can be discretised to

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

which is implemented by reducing the range of operation to $l \in \{1, ..., N-1\}$ and by applying the following virtual grid points

$$u_{-1}^n = -u_{-1}^n, \quad \text{and} \quad u_{N+1}^n = -u_{N-1}^n \tag{15}$$

which are needed for calculating Equation (9) at $l = 1$ and $l = N-1$, respectively

4. ACOUSTIC TUBE

This section describes how an acoustic tube can be implemented using FDTD methods. The theory in this section is based on [8] unless otherwise noted. Webster's horn equation [9] is a model that can be used to describe the lossless, small-amplitude propagation of air in a tube as

$$S\partial_t^2\Psi = c^2\partial_x(S\partial_x\Psi) \tag{16}$$

with the cross sectional area along the tube S (in m^2), acoustic potential Ψ (in m$^2/s$) and the speed of sound c (in m/s). In order to mitigate rounding errors, Webster's equation is reformulated as a system of two coupled first-order instead of the original second-order form

$$\frac{S}{\rho_0 c^2}\partial_t p = -\partial_x(Sv) \tag{17}$$

and

$$\rho_0 \partial_t v = -\partial_x p \tag{18}$$

with particle velocity $v(x,t)$ (in m/s) and air pressure $p(x,t)$ (in Pa). Both of these parameters are related to the acoustic potential in the following manner

$$p = \rho_0 \partial_t \Psi, \quad \text{and} \quad v = -\partial_x \Psi \tag{19}$$

with air density ρ_0 (in kg/m^3).

It can be useful to introduce an interleaved grid when performing the discretisation as done in [2] in order to ensure second-order accuracy for the forward and backward FD operators. Accordingly, the cross section is placed on the interleaved grid in space, while the velocity is placed on the interleaved grid in both space and time Therefore, the cross section $S(x)$ at location $x = lh$ can be approximated in the following manner

$$S_{l-1/2} = \mu_{x-}S(x=lh), \text{ and } S_{l+1/2} = \mu_{x+}S(x=lh) \tag{20}$$

$\bar{S}_l$ is another valuable proberty that can be defined as the average of $S_{l-1/2}$ and $S_{l+1/2}$

$$\begin{aligned}\bar{S}_l &= \mu_{x+}S_{l-1/2} = \mu_{x-}S_{l+1/2} \\ &= \mu_{xx}S(x=lh) = \frac{1}{2}(S_{l+1/2} + S_{l-1/2})\end{aligned} \tag{21}$$

With the interleaved grid in mind, equations 17 and 18 are discretised as

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p_l^n = -\delta_{x-}(S_{l+1/2}v_{l+1/2}^{n+1/2}) \tag{22}$$

$$\rho_0 \delta_{t-} v_{l+1/2}^{n+1/2} = -\delta_{x+}p_l^n \tag{23}$$

By expanding the the FD operators, we can obtain the update equation for pressure and velocity as

$$p_l^{n+1} = p_l^n - \frac{\rho_0 c\lambda}{\bar{S}_l}(S_{l+1/2}v_{l+1/2}^{n+1/2} - S_{l-1/2}v_{l-1/2}^{n+1/2}) \tag{24}$$

$$v_{l+1/2}^{n+1/2} = v_{l+1/2}^{n-1/2} - \frac{\lambda}{\rho_0 c}(p_{l+1}^n - p_l^n) \tag{25}$$

with Courant number $\lambda = ck/h$ [10].

The Levine and Schwinger radiation model [11] is employed to simulate energy loss caused by radiation. This model characterizes the radiation of plane waves in an unflanged circular pipe and leads to the following update equation for the right boundary

$$\begin{aligned}p_N^{n+1} &= \frac{1-\rho_0 c\lambda\zeta_3}{1+\rho_0 c\lambda\zeta_3}p_N^n \\ &- \frac{2\rho_0 c\lambda}{1+\rho_0 c\lambda\zeta_3}\left(v_{(1)}^n + \zeta_4 p_{(1)}^n - \frac{S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right)\end{aligned} \tag{26}$$

with

$$\begin{aligned}\zeta_1 &= \frac{2R_2 k}{2R_1R_2C_r + k(R_1+R_2)} \\ \zeta_2 &= \frac{2R_1R_2C_r - k(R_1+R_2)}{2R_1R_2C_r + k(R_1+R_2)} \\ \zeta_3 &= \frac{k}{2L_r} + \frac{\zeta_1}{2R_2} + \frac{C_r\zeta_1}{k} \\ \zeta_4 &= \frac{\zeta_2+1}{2R_2} + \frac{C_r\zeta_2 - C_r}{k}\end{aligned} \tag{27}$$

The various parameters are $R_1 = \rho_0 c$, $R_2 = 0.505\rho_0 c$, $L_r = 0.613\rho_0\sqrt{\bar{S}_N/\pi}$, and $C_r = 1.111\frac{\sqrt{\bar{S}_N}}{\rho_0 c^2\sqrt{\pi}}$. Once p_N^{n+1} is determined, we can compute the internal states $p_{(1)}$ and $v_{(1)}$ in the following manner

$$v_{(1)}^{n+1} = v_{(1)}^n + \frac{k}{L_r}\mu_{t+}p_N^n \tag{28}$$

and

$$p_{(1)}^{n+1} = \zeta_1\mu_{t+}p_N^n + \zeta_2 p_{(1)}^n \tag{29}$$

5. EXCITATION

Different types of exciters are available dependent on which resonator is in use. Bowing, strike and pluck excitation is available when user choose the damped stiff string as a resonator. Lip reed excitation is used when the user select the acoustic tube as a resonator. This section describes how these excitation types where implemented.

5.1 Bow

The following subsection is based on [2, Ch. 8], and describes how a the bowing excitation can be implemented using a static friction model. In order to simulate bowing excitation a nonlinear friction model dependent on the relative velocity v_{Rel} between the bow and the string is required. This project utilises the following static friction model

$$\Phi(v_{rel}) = \sqrt{2a}v_{rel}e^{-av_{rel}^2+1/2} \tag{30}$$

with friction characteristic Φ and friction parameter a.

Bowing force can be added to the discretised PDE of the stiff string from Equation (8) according to

$$\begin{aligned}\delta_{tt}u_l^n &= c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n \\ &+ 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_{l,p}(x_B^n)F_B^n\Phi(v_{rel}^n)\end{aligned} \tag{31}$$

with F_B being the bowing force divided by the total plate mass, x_B the bowing location and spreading operator $J_{l,p}(x_B^n)$ or order p. In this project the 0th-order spreading function was utilised, which is defined as

$$J_{l,0} = \frac{1}{h}\begin{cases}1, & \text{if } l = l_i \\ 0, & \text{otherwise,}\end{cases} \tag{32}$$

with discrete excitation position l_i. The relative velocity in the discrete domain is given as

$$v_{rel} = I_{l,p}(x_B^n)\partial_{t\cdot}u_l^n - v_B^n \tag{33}$$

with interpolation operator $I_{l,p}(x_B^n)$ of order p. The 0th-order interpolation operator, which has been used in this project is defined as

$$I_{l,0} = \begin{cases} 1, & \text{if } l = l_i \\ 0, & \text{otherwise,} \end{cases} \tag{34}$$

In order to obtain a solution for u_l^{n+1} at the bowing location, an inner product of the scheme in Equation (31) is calculated as

$$\begin{aligned} I_{l,p}(x_B^n)\delta_{tt}u_l^n &= c^2 I_{l,p}(x_B^n)\delta_{xx}u_l^n - \kappa^2 I_{l,p}(x_B^n)\delta_{xxxx}u_l^n \\ &\quad - 2\sigma_0 I_{l,p}(x_B^n)\delta_{t\cdot}u_l^n + 2\sigma_1 I_{l,p}(x_B^n)\delta_{t-}\delta_{xx}u_l^n \\ &\quad - ||J_{l,p}(x_B^n)||_d^n F_B^n \Phi(v_{rel}^n) \end{aligned} \tag{35}$$

By utilising the following identity

$$\partial_{tt} = \frac{2}{k}(\partial_t \cdot - \partial_{t-}), \tag{36}$$

Equation (35) can be assigned to a function $g(v_{rel}^n)$ in the following manner

$$\begin{aligned} g(v_{rel}^n) &= \left(\frac{2}{k} + 2\sigma_0\right) v_{rel}^n \\ &\quad + ||J_{l,p}(x_B^n)||_d^n F_B^n \Phi(v_{rel}^n) + b^n = 0 \end{aligned} \tag{37}$$

where

$$\begin{aligned} b^n &= -\frac{2}{k}\delta_{t-}u_l + \kappa^2 \delta_\Delta \delta_\Delta u_l^n \\ &\quad + \left(\frac{2}{k} + 2\sigma_0\right) v_B^n - 2\sigma_1 \delta_{t-}\delta_\Delta u_l^n. \end{aligned} \tag{38}$$

The Newton-Raphson method is then used to iteratively solve for v_{rel}

$$(v_{rel}^n)_{i+1} = (v_{rel}^n)_i - \frac{g((v_{rel}^n)_i}{g'((v_{rel}^n)_i} \tag{39}$$

where

$$g'((v_{rel}^n) = \frac{2}{k} + 2\sigma_0 + ||J_{l,p}(x_B^n)||_d^n F_B^n \Phi'(v_{rel}^n) \tag{40}$$

and

$$\Phi'(v_{rel}^n) = \sqrt{2a}(1 - 2a(v_{rel})^2)e^{-a(v_{rel}^n)^2 + 1/2} \tag{41}$$

5.2 Strike and pluck

In order to simulate strike and pluck excitation a raised cosine was utilised, which is defined as

$$F = F_{exc}(t) = \begin{cases} \frac{F_{max}}{2}\left(1 - \cos\left(\frac{q\pi(t-t_0)}{T_{exc}}\right)\right), & t_0 \le t \le t_0 + T_{exc} \\ 0, & \text{otherwise} \end{cases} \tag{42}$$

with maximum force F_{max} , the duration of the excitation T_{exc} and the point in time in which the excitation occurs t_0. Parameter q is altering the exitation type to be a pluck when $q = 1$ and a strike when $q = 2$ [7]. The force from the raise cosine can be applied to the discretised PDE of the stiff string from Equation (8) as follows

$$\begin{aligned} \delta_{tt}u_l^n &= c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n \\ &\quad + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_l(x_i)F \end{aligned} \tag{43}$$

5.3 Lip reed

The lip reed, which is an exciter used in brass instruments, is used as the excitation method when the tube resonator is selected. The theory in this subsection is based on [8, Ch. 5] unless otherwise noted. A single one-DoF-mass-spring-damper system can be employed to model the lip reed excitation. In this project, the outward sticking door model described in [12] has been utilized, excluding the collision component.

$$M_r\ddot{y} = -M_r\omega_r^2 y - M_r\sigma_r\dot{y} + S_r\Delta p \tag{44}$$

with the displacement of the lip reed from the equilibrium position $y = y(t)$ (in m), mass of the reed M_r (in kg), angular frequency of the reed ω_r (in rad/s), loss parameter σ_r (in s^{-1}) and surface area of the lip S_r (in m^2). Δp is the difference between the externally supplied mouth pressure P_m and the pressure at the mouth piece (in Pa), thus

$$\Delta p = P_m - p(0,t) \tag{45}$$

According to the Bernoulli equation, the pressure difference can be related to the volume flow velocity in the mouthpiece, denoted as U_B (in m^3/s), as follows

$$U_B = w[y + H_0]_+ \text{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}} \tag{46}$$

with width of the reed w (in m) and static equilibrium separation H_0 (in m). The notation $[\cdot]_+$ represents the positive part of a quantity and is defined as

$$[\cdot]_+ = \frac{\cdot + |\cdot|}{2} \tag{47}$$

Therefore, when the lips are closed, the flow velocity becomes zero. The motion of the reed leads to a secondary volume flow velocity, denoted as U_r (in m^3/s), which can be expressed as follows

$$U_r = S_r\dot{y} \tag{48}$$

Assuming conservation of volume velocity, the total volume of air injected at the left boundaries can be expressed as

$$S(0)v(0,t) = U_B(t) + U_r(t) \tag{49}$$

Parameters, U_b, U_r, y and Δp are all discretised to the interleaved temporal grid time as follows

$$\delta_{tt} y^{n+1/2} = -\omega_0^2 \mu_{t\cdot} y^{n+1/2} + \frac{S_r}{M}\Delta p^{n+1/2} \tag{50}$$

$$\Delta p^{n+1/2} = P_m - \mu_{t+} p_0^n \tag{51}$$

$$U_B^{n+1/2} = w[y^{n+1/2} + H_0]_+ \text{sgn}(\Delta p^{n+1/2})\sqrt{\frac{2|\Delta p^{n+1/2}|}{\rho_0}} \tag{52}$$

$$U_r^{n+1/2} = S_r \partial_{t\cdot} y^{n+1/2} \tag{53}$$

$$\mu_{x-}(S_{1/2} v_{1/2}^{n+1/2}) = U_B^{n+1/2} + U_r^{n+1/2} \tag{54}$$

Expanding the FD operators and solving for $y^{n+3/2}$ in Equation (50) leads to the following update equation

$$y^{n+3/2} = \frac{4y^{n+1/2} + \beta_r y^{n-1/2} + \xi_r \Delta p^{n+1/2}}{\alpha_r} \tag{55}$$

where

$$\alpha_r = 2 + \omega_0^2 k^2 + \sigma_r k, \quad \beta_r = \sigma_r k - 2 - \omega_0^2 k^2,$$
$$\text{and} \quad \xi_r = \frac{2 S_r k^2}{M} \tag{56}$$

The pressure difference across the reed can be determined by combining Equations (50) - (54) and manipulating the discrete leading to

$$\Delta p^{n+1/2} = \text{sgn}(c_3^n)\left(\frac{-c_1^n + \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2}\right)^2 \tag{57}$$

with

$$\begin{aligned} c_1^n &= w[y^{n+1/2} + H_0]_+ \sqrt{\frac{2}{\rho_0}} \geq 0, \\ c_2 &= b_2 + \frac{a_2 S_R}{a_1} \geq 0, \\ \text{and} \quad c_3^n &= b_1^n - \frac{\sqrt{a_3^n S_r}}{a_1} \end{aligned} \tag{58}$$

where

$$\begin{aligned} a_1 &= \frac{2}{k} + \omega_0^n k + \sigma_r \geq 0, \\ a_2 &= \frac{S_r}{M} \geq 0, \\ a_3^n &= \left(\frac{2}{k}\delta_{t-} - \omega_0^2 e_{t-}\right) y^{n+1/2}, \\ b_1^n &= S_{1/2} v_{1/2}^{n+1/2} + \frac{\bar{S}_0 h}{\rho_0 c^2 k}(P_m - p_0^n), \\ \text{and} \quad b_2 &= \frac{\bar{S}_0 h}{\rho_0 c^2 k} \geq 0 \end{aligned} \tag{59}$$

The lip reed is coupled to the tube by rearranging the equation at the left boundary $l = 0$ in the following manner

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0}(-2\mu_{x-}(S_{1/2} v_{1/2}^{n+1/2}) + 2S_{1/2} v_{1/2}^{n+1/2}) \tag{60}$$

By substituting Equation (49) into Equation (60) it is possible to derive the following update equation for the left boundary

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0}(-2(U_B + U_r) v_{1/2}^{n+1/2}) + 2S_{1/2} v_{1/2}^{n+1/2}) \tag{61}$$

6. DYNAMIC GRID

For both models simulation quality and bandwidth increase as the grid spacing h approaches the minimum stable grid spacing h_{min}, which for the damped stiff string is given by Equation (12) and for the tube is given by Courant–Friedrichs–Lewy condition stating that $\lambda \leq 1$ [10]. This is commonly implemented by the following operations in order [7]

$$N = \text{floor}(L/h); \quad h = L/N; \tag{62}$$

Sudden changes in parameters can lead to variations in the number of intervals N according to Equation (62). Because of the flooring operation, these changes can result in sudden variations in the number of grid points defining the system. These changes can introduce audible artifacts and, in the worst case, lead to an unstable simulation. in order to mitigate these issues a dynamic grid is introduced, which allows for smooth changes to a non-integer number of intervals between grid configurations. In order to utilise the dynamic grid it is necessary to introduce the fractional number of intervals $\mathcal{N}$, where $\lfloor \mathcal{N} \rfloor = N$. Substituting N for $\lfloor \mathcal{N} \rfloor = N$ in Equation (62) allows for smooth transitions between grid configurations and removes the need for the flooring operation. As a result, the stability condition is always satisfied with equality leading to optimal simulation quality [3].

In order to implement the fractional number of intervals for the original system u^n of the damped stiff string it is necessary to split it into two subsystems as done in [6] and [5]. Now the left part of the system is $u_{l_u}^n$ defined over $l_u \in \{0, ..., M_u^n\}$ and the right system $o_{l_o}^n$ defined over $l_o \in \{0, ..., M_o^n\}$. Where,

$$M_u = N^n - M_o^n \quad \text{and} \quad 0 < M_o^n < N^n \tag{63}$$

are the numbers of intervals for the left and right system, respectively. The resulting FD schemes are

$$\begin{aligned} \delta_{tt} u_{l_u}^n &= (c^n)^2 \delta_{xx} u_{l_u}^n - (\kappa^n)^2 \delta_{xxxx} u_{l_u}^n \\ &\quad - 2\sigma_0^n \delta_{t\cdot} u_{l_u}^n + 2\sigma_1^n \delta_{t-}\delta_{xx} u_{l_u}^n \end{aligned} \tag{64a}$$

$$\begin{aligned} \delta_{tt} o_{l_o}^n &= (c^n)^2 \delta_{xx} o_{l_o}^n - (\kappa^n)^2 \delta_{xxxx} w_{l_w}^n \\ &\quad - 2\sigma_0^n \delta_{t\cdot} o_{l_o}^n + 2\sigma_1^n \delta_{t-}\delta_{xx} o_{l_o}^n \end{aligned} \tag{64b}$$

Notice that parameters c^n, κ^n, σ_0^n and σ_1^n have all been made time-varying. The subsystems in Equation (64) are positioned adjacent to each other on the same domain x, where the locations of the grid points are defined as follows

$$x^n_{u_{l_u}} = l_u h^n \quad \text{and} \quad x^n_{o_{l_o}} = L^n - (M^n_o - l_o)h^n \tag{65}$$

for subsystems $u^n_{l_u}$ and $o^n_{l_o}$ respectively [5, 6]. A similar process can be done for the tube splitting the scheme of p^n and v^n into two-sets of first order as shown in [4]. In this case the pressure and velocity of the left system $p^n_{l_p}$ and $v^{n+1/2}_{l_p+1/2}$ are both defined over $l_p \in \{0, ..., M^n_p\}$ and the right $q^n_{l_q}$ and $w^{n+1/2}_{l_q-1/2}$ system are $l_q \in \{0, ..., M^n_q\}$, with

$$M_p = N^n - M^n_q \quad \text{and} \quad 0 < M^n_q < N^n. \tag{66}$$

Now the FD schemes becomes

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p^n_{l_p} = -\delta_{x-}(S_{l+1/2}v^{n+1/2}_{l_p+1/2}) \tag{67a}$$

$$\rho_0\delta_{t-}v^{n+1/2}_{l_p+1/2} = -\delta_{x+}p^n_{l_p} \tag{67b}$$

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}q^n_{l_q} = -\delta_{x+}(S_{l+1/2}w^{n+1/2}_{l_q+1/2}) \tag{67c}$$

$$\rho_0\delta_{t-}w^{n+1/2}_{l_q+1/2} = -\delta_{x-}q^n_{l_q} \tag{67d}$$

The two pairs of first order systems are placed on the same domain adjacent to each other in a similar fashion to the stiff string case, such that

$$x^n_{p_{l_p}} = l_p h \quad \text{and} \quad x^n_{q_{l_q}} = L^n - (M^n_q - l_q)h \tag{68}$$

describes the locations of the left and right system, respectively [4].

In both the damped stiff string and tube case, the total number of grid points are one more than their original systems. Furthermore, the outer boundaries of both original systems will be the same for the outer boundaries of the modified systems.

6.1 Connection of inner boundaries

If $\mathcal{N}^n = N^n$, the inner boundaries will perfectly overlap, and thus impose a rigid connection on the inner boundaries resulting in

$$u^n_{M^n_u+1} = o^n_0, \quad \text{if} \quad x^n_{u_{M^n_u}} = x^n_{o_0} \tag{69}$$

for the damped stiff string [6] and

$$p^n_{M^n_p} = q^n_0, \quad \text{if} \quad x^n_{p_{M^n_p}} = x^n_{q_0} \tag{70}$$

for the tube [4]. In the tube case, the domains of v and w are extended at the inner boundaries to include $v^{n+1/2}_{M^n_p+1/2}$ and $w^{n+1/2}_{-1/2}$ in order to calculate $p^{n+1}_{M_p}$ and q^{n+1}_0. However, this requires two virtual grid points $p^n_{M^n_p+1}$ and q^n_{-1}. When the inner boundaries are perfectly overlapping these virtual grid points can be calculated as

$$p^n_{M_p+1} = q^n_1, \quad \text{and} \quad q^n_{-1} = p^n_{M_p-1} \tag{71}$$

When a physical parameter is changed, the location of the grid points will change in accordance with Equations (65) and (68). As a result the inner boundaries will no longer overlap and new definitions are required for the virtual grid points. Utilising quadratic Lagrangian interpolation these new definitions are found as

$$p^n_{M^n_p+1} = \mathcal{I}^n p^n_{M^n_p} + q^n_0 - \mathcal{I}^n q^n_1 \tag{72a}$$

$$q^n_{-1} = -\mathcal{I}^n p^n_{M^n_p-1} + p^n_{M^n_p} - \mathcal{I}^n q^n_0 \tag{72b}$$

where

$$\mathcal{I}^n = \frac{\alpha^n - 1}{\alpha^n + 1} \tag{73}$$

and

$$\alpha^n = \mathcal{N}^n - N^n \tag{74}$$

is the fractional part of $\mathcal{N}^n$ [4].

The case of the damped stiff string is more complex because the expansion of the δ_{xxxx} operator in Equation (64) at $u_{M^n_u}$ and o^n_0 requires the definition of two virtual grid points for each inner boundary. Additionally, expansion of the operator at $u_{M^n_u-1}$ and o^n_1 requires the definition of a single virtual grid point. In order to find these definitions it is necessary to introduce a matrix form of the system from Equation (64) as shown in [6] and [5]. The state vectors placed in the following $N^n \times 1$ column vector

$$\mathbf{U}^n = \begin{bmatrix}\mathbf{u}^n \\ \mathbf{o}^n\end{bmatrix} \tag{75}$$

where $\mathbf{u}^n = [u^n_1, ..., u^n_{M_u}]^T$ and $\mathbf{o}^n = [o^n_0, ..., o^n_{M_o} - 1]^T$. The outer boundaries are excluded due to their states being 0 at all times. The system from Equation (64) can then be rewritten as

$$A^n\mathbf{U}^{n+1} = \mathcal{B}^n\mathbf{U}^n + \mathcal{C}^n\mathbf{U}^{n-1} \tag{76}$$

with $A^n = 1 + \sigma^n_0 k$ and

$$\begin{aligned}\mathcal{B}^n &= 2\mathbf{I}_{N^n} + (\lambda^n)^2\mathcal{D}^n_{xx} - (\mu^n)^2\mathcal{D}^n_{xxxx} + S^n\mathcal{D}^n_{xx}, \\ \mathcal{C}^n &= -(1 - \sigma^n_0 k)\mathbf{I}_{N^n} - S^n\mathcal{D}^n_{xx},\end{aligned} \tag{77}$$

where $\lambda^n = c^n k/h^n$, $\mu^n = \kappa^n k/(h^n)^2$ and $S^n = 2\sigma^n_1 k/(h^n)^2$. The $N^n \times N^n$ matrix

$$\mathcal{D}^n_{xx} = \begin{bmatrix} \ddots & \ddots & & & & \mathbf{0} \\ \ddots & -2 & 1 & & & \\ & 1 & \mathcal{I}^n - 2 & 1 & -\mathcal{I}^n & \\ & -\mathcal{I}^n & 1 & \mathcal{I}^n - 2 & 1 & \\ & & & 1 & -2 & \ddots \\ \mathbf{0} & & & & \ddots & \ddots \end{bmatrix} \tag{78}$$

is the matrix form of the δ_{xx} operator including the quadratic interpolation at the inner boundaries and $\mathbf{I}_{N^n}$ is the identity matrix. Furthermore,

$$\mathcal{D}^n_{xxxx} = \mathcal{D}^n_{xx}\mathcal{D}^n_{xx} \tag{79}$$

is the matrix form of the δ_{xxxx} operator [5, 6].

6.2 Adding and removing grid points

If the case of parameter change resulting in $N^n \neq N^{n-1}$, grid points has to be added to or removed from the system. Similar to [5] and [6], this work only considers changes in grid points to affect the left side of each system.

In the damped stiff string case, if $N^n > N^{n-1}$ grid points are added to $\mathbf{u}$ and $\mathbf{u^{n-1}}$ as follows

$$\begin{aligned} \mathbf{u}^n &:= \left[(\mathbf{u}^n)^T \quad I_3^n \mathbf{z}_u^n\right]^T, \\ \mathbf{u}^{n-1} &:= \left[(\mathbf{u}^{n-1})^T \quad I_3^n \mathbf{z}_u^{n-1}\right]^T, \end{aligned} \quad \text{if } N^n > N^{n-1} \tag{80}$$

with

$$\begin{aligned} \mathbf{z}_u &= \left[u^n_{M_u^{n-1}-1} \quad u^n_{M_u^{n-1}} \quad o_0^n \quad o_1^n\right]^T, \\ \mathbf{z}_u^{n-1} &= \left[u^{n-1}_{M_u^{n-1}-1} \quad u^{n-1}_{M_u^{n-1}} \quad o_0^{n-1} \quad o_1^{n-1}\right]^T, \end{aligned} \tag{81}$$

and cubic interpolator

$$I_3^n = \left[-\frac{\alpha^n(\alpha^n+1)}{(\alpha^n+2)(\alpha^n+3)} \quad \frac{2\alpha^n}{\alpha^n+2} \quad \frac{2}{\alpha^n+2} \quad -\frac{2\alpha^n}{(\alpha^n+3)(\alpha^n+2)}\right] \tag{82}$$

If $N^n < N^{n-1}$ grid points are simply removed at the left inner boundary in the following manner [6]

$$\begin{aligned} \mathbf{u}^n &:= \left[u_1^n \quad \cdots \quad u^n_{M_u^{n-1}-1}\right]^T, \\ \mathbf{u}^{n-1} &:= \left[u_1^{n-1} \quad \cdots \quad u^{n-1}_{M_u^{n-1}-1}\right]^T, \end{aligned} \quad \text{if } N^n < N^{n-1} \tag{83}$$

In the tube case grid points are added to $\mathbf{p}$ and $\mathbf{v^{n-1/2}}$ according to

$$\begin{aligned} \mathbf{p}^n &:= \left[(\mathbf{p}^n)^T \quad I_3^n r^n\right]^T, \\ \mathbf{v}^{n-1/2} &:= \left[(\mathbf{v}^{n-1/2})^T \quad I_3^n \mathbf{z}_v^{n-1/2}\right]^T, \end{aligned} \quad \text{if } N^n > N^{n-1} \tag{84}$$

where

$$\begin{aligned} \mathbf{p}^n &= \left[p_0^n \quad \cdots \quad p^n_{M_p^n}\right]^T, \\ \mathbf{v}^{n-1/2} &= \left[v_{1/2}^{n-1/2} \quad \cdots \quad v^{n-1/2}_{M_p^n+1/2}\right]^T, \\ \mathbf{r}^n &= \left[p_{M_p^n-1} \quad p_{M_p^n} \quad q_0^n \quad q_1^n\right]^T \quad \text{and} \\ \mathbf{z}_v^{n-1/2} &= \left[v^{n-1/2}_{M_p^n-1/2} \quad v^{n-1/2}_{M_p^n+1/2} \quad w^{n-1/2}_{1/2} \quad w^{n-1/2}_{3/2}\right]^T - \eta \end{aligned} \tag{85}$$

with

$$\eta = \eta^{n-1/2} = \left(w^{n-1/2}_{-1/2} - v^{n-1/2}_{M_p^n+1/2}\right) \cdot \left[0 \quad 0 \quad 1 \quad 1\right]^T. \tag{86}$$

Grid points can be removed from the system according to [4]

$$\begin{aligned} \mathbf{p}^n &:= \left[p_0^n \quad \cdots \quad p^n_{M_p^n-1}\right]^T, \\ \mathbf{v}^{n-1/2} &= \left[v_{1/2}^{n-1/2} \quad \cdots \quad v^{n-1/2}_{M_p^n-1/2}\right]^T. \end{aligned} \quad \text{if } N^n < N^{n-1} \tag{87}$$

7. IMPLEMENTATION

A real time implementation of the physical models was implemented on the Bela platform. Bela is an open-source embedded computing platform designed for developing responsive, real-time interactive systems with audio and sensors. The Bela system is based on the BeagleBone Black single-board computer, which uses 1GHz ARM Cortex-A8 processors, has 512MB of DDR RAM and features a programmable real-time unit consisting of a 200MHz, 32-bit processor that enables ultra-low latency processing. Bela operates on a custom audio processing environment that utilizes the real-time Linux extensions of Xenomai [13]. This section will go through how the theory from the previous sections have been implemented in real time as well as the design considerations and the additional features that have been implemented.

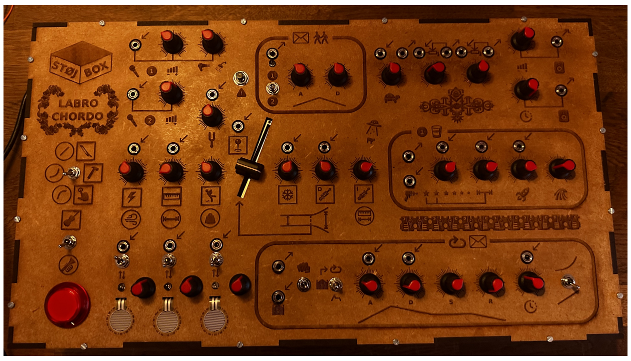

Figure 1. The implemented modular synth

7.1 Real time implementation of the damped stiff string and acoustic tube

Bela provides a browser-based IDE and supports both low-level languages such as C/C++ as well as computer music programming languages like Pure Data, SuperCollider, and Csound [13]. This project was implemented using C++. The physical models were implemented with the dynamic grid, utilising the theory provided in the previous sections. An important consideration when implementing FD schemes in real time is how to minimise the required computational power. In order to update the states systems in an efficient manner a pointer switch is employed, which reduces the number of copy operations significantly. The pointer switch is performed by assigning a temporary pointer to the memory location of time step to u^{n-1} and o^{n-1} for the damped string and to p^n, $v^{n-1/2}$, q^n and $w^{n-1/2}$ for the acoustic tube. The temporary pointer is later overwritten when calculating the scheme. The pointer switch method reduces the number of operations from $2(N+$

1) down to 8 in the damped string case and down to 10 operations in the tube case. To prevent extra computations in the FD scheme, it was decided to calculate as many of the non time varying coefficients as possible. The implementation of the dynamic grid has utilised the same method as [6], where $M_o^n = 1 \forall n$ and $M_q^n = 1 \forall n$, such that M_u^n and M_p^n are dynamically changed according to Equations (63) and (66). As a result the right system of both the damped stiff string and the tube only has a single moving grid point. The implemented code can be found as an attachment on digital exam.

7.2 Audio I/O

The Bela features two audio inputs and two audio outputs running at 44.1 kHz, and they are all utilised in this project. The two audio inputs are able to act as an alternative to the exciters described in Section 5. Both audio inputs have a level control, and the first audio input also features a dry/wet control allowing some of the original input to pass through unaffected to the output. The output from the selected resonator goes to both audio outputs. The first audio output features a volume control and the second audio output features a delay control. The delay control allows the user to delay the second output up to a total of 2 seconds using a potentiometer. The delay control allows the user to create an echo effect by patching the second output to an input and the level control of that input will as a result act as feedback control. It also gives the user a lot of flexibility, since the user e.g. can patch different types of external effect processors into the feedback loop.

7.3 Controls

The Bela features 8 16-bit 22.05khz ADC's (excluding the two separate audio outputs) that can be used by e.g. potentiometers and CV inputs for controlling parameters. This project did however require a total of 26 analog inputs. In order to satisfy this need a 16 channel multiplexer and a 4 channel multiplexer were utilised. As a result 20 channels can be read using only two of the ADC's. However, this process does sacrifice 6 digital pins (out of 16 available) and reduces the sampling speed for the multiplexer channels.

7.3.1 CV inputs

Control Voltage (CV) is a standardized method used by synthesizers to exchange control signals and functions among different modules. It serves as a common language for communication between various components within a modular synthesizer system. CV signals are typically transmitted between modules in a modular synthesizer system using patch cables. In the case of the Eurorack format, these patch cables are commonly 3.5 mm mono jack plugs. The same 3.5 mm mono jack connections have been used for this project. There is no standard for the voltage ranges of the CV that eurorack and other modular gear outputs. The most commonly used ranges are bipolar $-5\text{V} - 5\text{V}$ or unipolar $0 - 10\text{V}$. However, the analog inputs on the Bela only have an usable input range of $0 - 4.049\text{V}$. Furthermore, negative signals and signal above 5V can damage the board. In order to mitigate these issues it has been decided to scale a bipolar $-5\text{V} - 5\text{V}$ signal down to a range of $0 - 3.3\text{V}$ using differential amplifiers. Each CV input, that is going to the Bela, is going through differential amplifier. The differential amplifiers are build using MCP602 rail-to-rail op-amps in a configuration such that a 5V signal becomes a 0V signal, and a -5V signal becomes a 3.3V signal. The outputs from the differential amplifiers goes to the ADC's of the Bela and are inverted and mapped in code as required by the respective parameters that the CV inputs represents.

7.3.2 Potentiometers

The signals from all the potentiometers have been filtered using 1592 Hz passive RC low-pass filters in order to avoid high frequency jitter and noise. It was decided that all of the potentiometers should be connected to the 16 channel multiplexer because the reduction in sampling speed would be practically unnoticeable for these, since it is not possible to turn the potentiometers at such high speeds.

7.3.3 Controllable parameters

The user has control whether a damped stiff string model or a acoustic tube model is in use via a switch, since the synth can only run one model at a time. The controls related to the physical parameters and excitation types will affect different parameters dependent on, which mode is selected. If the damped stiff string is selected, the user can control the stiffness coefficient, the length of the string, the frequency dependent and independent damping and the excitation position. Furthermore, the user can change the excitation type of the string from either a bow, strike or a pluck using a 3-way switch. When bowing excitation is selected the user can control the bowing force/velocity (bowing force and velocity are linked together) and the friction of the string. When pluck or strike excitation is selected the user have control of excitation force and excitation time.

If the acoustic tube is selected the user has control over the length of the tube and several parameters related to the shape of the tube. The tube consists of two parts: a cylindrical section followed by a bell-shaped section. The user can control the radius of the cylindrical part and the radius of the bell boundary. Furthermore, the user can determine whether the bell grows linearly, exponentially or logarithmic using the 3-way switch used for selecting excitation type in the stiff string mode. If the bell growth curve is set to linear, the growth rate is calculated as follows

$$r_{growth} = \frac{B_r - C_r}{N_B} \tag{88}$$

with growth rate of the radius r_{growth}, the radius of bell end point B_r , the cylinder radius C_r and the number of grid points for the bell part N_B. Afterwards, the radius of each grid point in the bell section is calculated in a for loop iterating over the number of grid points for the bell part using $r = r + r_{growth}$. For exponential growth, a similar process is used, but the growth rate is calculated as

$$r_{growth} = e^{\log(B_r/C_r)/N_B} \tag{89}$$

In this case the radius of each grid point is calculated as $r = C_r r^i_{growth}$, where i is an index representing the current grid point of the bell. If the bell is logarithmic, the growth rate is calculated using

$$r_{growth} = \frac{B_r - C_r}{\log(N_B)} \tag{90}$$

and the the radius of each grid point is calculated as $r = C_r + r_{growth}\log(i)$. The growth rates from Equation (88), (89) and (90) were derived using the definitions for linear, exponential and logarithmic growth, respectively. The cross section is the calculated from the radius using $S = \pi r^2$. The user can also specify the length of the cylinder compared to the total length using a "cylinder/bell ratio" parameter. The cylinder length is calculated as $c_L = L \cdot CB_{ratio}$, and the bell length is calculated as $b_L = L \cdot (1 - CB_{ratio})$, where CB_{ratio} represents the selected "cylinder/bell ratio". Besides the length and shape of the tube, the user can also change the mouth pressure as well as the width and mass of the reed used to excite the tube.

Of course the user also has control over the fundamental frequency of each resonator type. The method used for controlling the fundamental frequency is however different for each resonator. For the stiff string, the fundamental frequency can be controlled by changing wave speed, according to [7]

$$f_0 = \frac{c}{2L} \tag{91}$$

The length of the string is set to a constant value, thus the desired fundamental frequency can be applied by setting $c^n = 2L \cdot f_0^n$. For the acoustic tube, the fundamental frequency is determined by the frequency of the lip reed. The acoustic tube acts as an amplifier for certain resonant frequencies. Therefore it has been decided to match the lip frequency with a resonating mode determined by the length of the tube according to [2]

$$\omega_r^{n+1/2} = B_s \frac{2\pi c}{\rho_0 L^n} \tag{92}$$

Here B_s is a scalar multiplier, which scales the length of the tube. B_s is what is controlled by the length parameter. This ensures matching of the same resonating modes when playing different frequencies and allows the user to alter the length parameter to match different resonating modes.

A common standard for modular synths is 1V/octave, which means that every additional 1V of CV produces a doubling of the frequency. This standard has also been utilised in this project using the following equation

$$f_0 = f_{ref} 2^{\mathrm{V}} \tag{93}$$

with reference frequency f_{ref}, which the user can select using a potentiometer.

The chosen upper limits and lower limits for the accessible physical parameters can be seen in Table (1) for the damped stiff string mode and Table (2) for the acoustic tube mode.

CV inputs have been added to most of the available parameters, with the exception of frequency independent damping and friction for the string, and length scaling and reed mass for the tube. The CV inputs are added to or subtracted from the values selected by the potentiometers.

Table 1. Upper and lower parameter limits when in damped stiff string mode

Parameter name	Symbol	Lower limit	Upper limit
String length	L	0.1 m	4 m
Stiffness coefficient	κ	$0.01\text{m}^2/\text{s}$	$0.2\text{m}^2/\text{s}$
Freq-indep. damping	σ_0	0.1s^{-1}	10s^{-1}
Freq-dep. damping	σ_1	$0.0002\text{m}^2/\text{s}$	$0.05\text{m}^2/\text{s}$
Excitation force	F_b	0	4 N
Excitation velocity	v_b	0	0.4 m/s
Excitation position	l_i	floor($0.1N$)	floor($0.9N$)
Friction	a	$5 \cdot 10^{-11}$	$5 \cdot 10^{-7}$

Table 2. Upper and lower parameter limits when in acoustic tube mode

Parameter name	Symbol	Lower limit	Upper limit
Length scaling	B_s	0.1	0.8
Cylinder radius	C_r	1 cm	10 cm
Bell radius	B_r	$C_r + 0$cm	$C_r + 50$cm
Cylinder bell ratio	CB_{ratio}	0	0.9
Mouth pressure	P_M	0	20 Pa
Reed width	w_r	0.1 cm	2 cm
Reed mass	w_m	0.01 g	50 g

Beside the parameters mentioned here, additional parameters related to the modulation sources and CV utilities are also controllable with CV inputs and/or potentiometers. These will be discussed in the upcoming subsection.

7.4 Modulation sources and CV utilities

Most users that would be interested in a semi-modular physical modelled synth most likely already have access to modules that can provide various CV modulation signals and CV utilities. However, it still seems reasonable to provide a variety modulation sources and CV utilities to the synth, such that the synth also is usable on its own without external modules. This subsection present the various modulation sources and CV utilities implemented for the synth.

7.4.1 Loopable ADSR and One–shot event generator

Two modulation sources are provided in form of a loopable ADSR and a one-shot event generator. Both of these are based on programmable PIC microcontrollers developed by the Electric Druid company. These chips have been used in order to free up processing power, ADC's and DAC's on the Bela board. The loopable ADSR can work a regular ADSR envelopes, but it also provides a lot of features not available in most ADSR envelopes. These include the ability to change between linear and exponential curves, a time control that can modulate the length of the entire envelope, and an optional punch control adding an additional "punch" stage between the attack and decay. Furthermore, the chip allows for three different envelope modes - a normal ADSR envelope, a gated looping mode and a LFO looping mode [14]. The ADSR chip also features a level control, that modulates the output level of the envelope. This level control has been wired to the gate input of the chip, resulting in a dynamic gate input. The one-shot event generator is designed to create unique unipolar modulation waveforms when activated by a trigger signal. The user is able to alter the shape and speed of the modulation waveform, and also add decaying echos to it, with separate delay and repeat controls [15]. Both chips output a modulation signal from $0 - 5$V. The signal from loopable ADSR is ready as is, but the output from the one-shot-envelope generator is 2MHz pulse-density modulation (PDM) output signal, which has to be filtered first. The PDM signal has been filtered using a passive 340Hz 12dB/Oct RC filter. All controllable parameters of these chips are accessible via switches and potentiometers. Patchable CV inputs have also been added to some controls, including the attack, decay and time control of the loopeable ADSR, and the speed, repeats, and delay control of the one-shot generator. The CV inputs as well as the trigger/gate inputs are protected from negative voltage and over-voltage using a circuit consisting of a diode, transistor and two resistors.

7.4.2 *Envelope follower*

An envelope follower has been implemented using two digital one-pole low-pass filters, one for the attack stage and one for the release stage. The reasoning for having two filters is that you often want a different behavior for the attack and release stages (e.g. a quick attack and slow release). Thus, the two filters will have different time constants, τ_a for the attack stage and τ_b for the release stage, that can be selected by the user using two potentiometers. The filter coefficients are calculated as

$$b_a = \sqrt[f_s]{\mathrm{e}^{-1/\tau_a}} \quad \text{and} \quad b_r = \sqrt[f_s]{\mathrm{e}^{-1/\tau_r}} \tag{94}$$

We can calculate the filters and also choose which filter is active dependent on the level of input, $x[n]$, compared to the level of the output, $y[n]$, as shown in [16, Ch. 4] in the following manner

$$y[n] = \begin{cases} b_a y[n-1] + (1-b_a)x[n] & x[n] \geq y[n-1] \\ b_r y[n-1] + (1-b_r)x[n] & x[n] < y[n-1] \end{cases}. \tag{95}$$

The user can choose to let the envelope follower react to either the first or the second audio input using a switch.

7.4.3 *Slew rate limiter*

A slew rate limiter has also been implemented. The slew rate limiter works by limiting the rate of change of a signal in a linear manner, thus allowing for a smooth transition from one voltage level to another, ensuring that the rate of change does not exceed a specified maximum rate. One common use case for a slew rate limiter is to introduce glides between notes. The implementation of the slew rate limiter involves tracking the previous input value and calculating the difference between the current input value and the previous input value. If that difference is larger than the allowed difference, the signal should be limited [17]. The user can change the maximum rate of change pr second using a potentiometer.

7.4.4 *FSR's and modulation looper*

Three force-sensing resistors (FSR's) have been added such that the user can modulate parameters by pressing the FSR's. Each FSR has an assigned potentiometer that can attenuate its output signal and there is also a switch that allows the user to invert the signal of the FSR. The signals from each FSR are read using the ADC's of the Bela, allowing the modulation signals to be recorded.

The user can record the signals from the FSR's into a buffer with three rows (one for each FSR) and a length of $10f_s$. The recording starts when the button is held and continues until the button is either released or the maximum recording length of 10 seconds have been reached. The recording will then loop until it is either cleared or overwritten. A recording can be cleared by double pressing the button. The rec/clear button features an LED that blinks while the modulations are being recorded, and is active while the loop is playing. If a loop is not playing, the direct signals from the FSR's are used as the output.

The outputs from the envelope follower, slew rate limiter and FSR's (or the looped buffer) are send to the DAC channels of the Bela. The output of each DAC is visualised by an LED and fed to a jack socket, which the user can patch to modulate any parameter with a CV input. The synth also contains two passive attenuators, that can reduce the amplitude of CV signals. These passive attenuators only consist of an input and output jack socket connected to a potentiometer.

7.5 Enclosure

The case and the front panel were designed using Inkscape [18], and build using 4mm HDF panels that were cut and engraved by a laser cutter.

8. DISCUSSION

The implementation is limited by the CPU performance of the Bela. Especially at low frequencies (since these requires more grid points) there are a risk of CPU overload and dropouts. Several initial ideas that the author had for the synth, such as polyphony and couplings between resonators, had to be scrapped, due to these CPU limitations. The models will still continue to run and be stable when experiencing CPU overload and dropouts, but it does result in a distorted sound output. It could be argued that the additional features like the envelope follower, slew rate limiter and modulation looper could be sacrificed for improved performance. However, the toll on the CPU performance from these features are negligible compared to how much processing power it requires to update the FD schemes.

There is a possibility that the grid points at the inner boundaries will have different values just before removing a grid point, thus violating the rigid condition from Equation (69) and (70). One might apply a displacement correction as suggested in [3, 4] by applying an artificial spring connection between the grid points at the inner boundary. However, as discussed in [5] this procedure of adding a localised damping might actually be more unnatural than not including this correction. As mentioned in [3, 5], the method of the dynamic grid only assumes sub-audio rate modulations of the physical properties. This is because the conventional stability and energy analyses performed on FDTD schemes are no longer applicable in the time varying case [3]. However, with modular synthesis there is nothing hindering the user to patch an audio rate signal to a CV input controlling one of the physical properties. While the author has successfully applied audio rate modulations to some of these properties, it has also lead to instability in some cases, especially when the amplitude of the audio rate modulation is large. So even though the author finds the sounds created by these audio rate modulations interesting and unique, there should be placed a limit on how fast grid points can be added to or removed from the system. More experimentation is however still required in order to find the right balance for this limit, such that stability is always ensured while still retaining the possibility to achieve some of the unique sounds that can result from fast modulation of physical properties. Currently, the damped stiff string model runs perfectly and the only situations where the author has experienced instability has been with the previously mentioned audio rate modulation of physical properties. However, while the acoustic tube does run well for the most part, the author has experienced some stability issues at certain settings. The author has yet to pinpoint what causes this instability.

9. CONCLUSION

This paper presented the implementation of a physically modelling based embedded modular synth utilising FDTD methods. Two types of resonators have been implemented, a damped stiff string and an acoustic tube. Furthermore various exciters in form of a bow, strike, pluck and lip reed have also been implemented. In order to allow for smooth parameter changes of physical properties a dynamic grid was applied to the physical models. The main obstacle with the implementation is currently CPU limitations, since dropouts and CPU overload will occur when the number of grid points becomes too large. Thus, as of today it is still best to limit these complex and CPU intensive physical models for use in plug-ins, since the average personal computer is much faster than the available embedded platforms. However, as embedded platforms inevitably will become faster in the future, it will open the possibility for embedding CPU intensive physical models. Future work will include finding an appropriate limit for how fast grid points can be added to or removed from the system, as well as solving the stability issues with the current implementation of the acoustic tube model.

10. REFERENCES

[1] P. Ruiz, “A technique for simulating the vibrations of strings with a digital computer,” Master’s thesis, University of Illinois, 1969.

[2] S. Willemsen, “The emulated ensemble: Real-time simulation of musical instruments using finite-difference time-domain methods,” Ph.D. dissertation, 2021, phD supervisor: Prof. Stefania Serafin, Aalborg University Co-Supervisor (external): Prof. Stefan Bilbao, University of Edinburgh.

[3] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, “Dynamic grids for finite-difference schemes in musical instrument simulations,” in *2021 24th International Conference on Digital Audio Effects (DAFx)*, 2021, pp. 144–151.

[4] ——, “A physical model of the trombone using dynamic grids for finite-difference schemes,” in *2021 24th International Conference on Digital Audio Effects (DAFx)*, 2021, pp. 152–159.

[5] “The dynamic grid: Time-varying parameters for musical instrument simulations based on finite-difference time-domain schemes,” *Journal of the Audio Engineering Society*, Sep. 2022.

[6] S. Willemsen and S. Serafin, “Real-time implementation of the dynamic stiff string using finite-difference time-domain methods and the dynamic grid,” in *Proceedings of the 25th International Conference on Digital Audio Effects (DAFx20in22),*, G. Evangelista and N. Holighaus, Eds., Sep. 2022, pp. 130–137.

[7] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*, 09 2009.

[8] R. L. Harrison-Harsley, “Physical modelling of brass instruments using finite-difference time-domain methods,” 2018.

[9] A. G. Webster, “Acoustical impedance and the theory of horns and of the phonograph,” *Proceedings of the National Academy of Sciences*, vol. 5, no. 7, pp. 275–282, 1919.

[10] R. Courant, K. Friedrichs, and H. Lewy, “Über die partiellen differenzengleichungen der mathematischen physik,” *Mathematische Annalen*, vol. 100, pp. 32–74.

[11] H. D. Levine and J. S. Schwinger, “On the radiation of sound from an unflanged circular pipe,” *Physical Review*, vol. 73, pp. 383–406, 1948.

[12] S. Bilbao, “Direct simulation of reed wind instruments,” *Computer Music Journal*, vol. 33, pp. 43–55, 12 2009.

[13] Bela. (Year unknown) Bela. [Online]. Available: https://bela.io

[14] E. Druid. (Year unknown) Envgen 8c. [Online]. Available: https://electricdruid.net/product/envgen8/

[15] ——. (Year unknown) Oneshot event generator. [Online]. Available: https://electricdruid.net/product/oneshot-event-generator/

[16] J. D. Reiss and A. P. McPherson, “Audio effects: Theory, implementation and application,” 2014.

[17] Weimich. (2021) Digital slew rate limiter filter and c implementation. [Online]. Available: https://www.dsp-weimich.com/digital-signal-processing/digital-slew-rate-limiter-filter-and-c-realization/

[18] (Year unknown) Inkscape. [Online]. Available: https://inkscape.org