Implementing physical models in real-time using partitioned convolution: An adjustable spring reverb

Onofrei, Marius Gerorge; Willemsen, Silvin; Serafin, Stefania

Published in:
Proceedings of the 18th Sound and Music Computing Conference

DOI (link to publication from Publisher):
10.5281/zenodo.5113511

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Onofrei, M. G., Willemsen, S., & Serafin, S. (2021). Implementing physical models in real-time using partitioned convolution: An adjustable spring reverb. In D. A. Mauro, S. Spagnol, & A. Valle (Eds.), *Proceedings of the 18th Sound and Music Computing Conference* (pp. 108-114). Axea sas/SMC Network. Proceedings of the Sound and Music Computing Conference https://doi.org/10.5281/zenodo.5113511

IMPLEMENTING PHYSICAL MODELS IN REAL-TIME USING PARTITIONED CONVOLUTION: AN ADJUSTABLE SPRING REVERB

Marius G. ONOFREI (monofr11@student.aau.dk)[1], **Silvin WILLEMSEN** (sil@create.aau.dk)[2], and **Stefania SERAFIN** (sts@create.aau.dk)[2]

[1]*Dept. of Architecture, Design & Media Technology*, **Aalborg University Copenhagen**, Denmark
[2]*Multisensory Experience Lab, CREATE*, **Aalborg University Copenhagen**, Denmark

ABSTRACT

This paper presents a detailed description of the development of a real-time spring reverb effect interface which is built based on the solution of a complex physical modelling implementation of helical springs. Impulse responses for various helical springs with different physical parameters are computed offline using an implicit finite difference scheme. These are then used to manipulate a real-time input sound by means of a partitioned convolution implementation, which allows for the use of long impulse responses with low latency. Furthermore, a smooth transition is carried out when changing from one impulse response to another, thus providing the means for real-time manipulation of the physical parameters of the spring and consequently the effect's quality.

1. INTRODUCTION

Spring reverbs have been around since the 1940s [1], and have been developed as cheap, compact devices which give the illusion of room-reverberation [2]. However, due to their highly dispersive nature, they could never really reproduce the sound of a real room. Even so, their sound had its unique appeal and they became very popular, particularly due to their affordability and compact size which allowed them to be included in classic guitar amplifiers throughout the late 20th century. Spring reverb simulations have been implemented for example using combinations of allpass filters [3] or other combinations of filters and delays [4], [5], as well as virtual analogue simulations [6].

In [7], Bilbao and Parker suggest a spring reverb simulation using finite difference schemes (FDSs), based on a helical spring model where the pitch angle is assumed to be small, described as a system of two variables: the transverse and longitudinal vibration. The model is a reduced version of the twelve variable system described by Wittrick [8]. Starting from the same model, Bilbao describes a more complex interleved FDS which includes the effect of the pitch angle in [9]. Van Walstijn presents an alternative FDS which uses ghost nodes in order achieve higher order spatial accuracy in [10], but needs to be evaluated a a very high sample rate of 1 MHz in order to limit numerical dispersion.

The implicit method proposed by [7] can be run at a typical sampling frequency of 44100 Hz and capture the complex dispersive properties of the spring and could potentially be implemented as-is in real-time for a fixed set of spring parameters. This, however, requires a large dimensional matrix inversion for solving the update equation. Additionally if one desires to change the spring's physical parameters dynamically, an optimization procedure is necessary in order to calculate values for the free parameters of the scheme as to minimize the difference between the numerical dispersion and the dispersion of the original system. This computationally demanding fact results in this solution being impractical for a real-time implementation.

The main aim in our paper is to make use of Bilbao and Parker's helical spring physical model in a real-time spring reverb application regardless of the limitation described in the above paragraph. Using this model gives us the flexibility of easily simulating springs of different physical properties. We digitally reproduce a spring reverb and embed it in a physical interface aiming to extend its possibilities compared to the typical uses, with a focus being on the real-time manipulation of the spring's physical parameters. Our approach consists of computing a database of helical spring impulse responses, where the physical parameters of the springs are varied in a consistent way with a focus on the parameters found to be most critical in terms of audio perception. These impulse responses can then be convolved with a dry input sound in order to add the wet spring reverb quality. This is achieved by implementing a partitioned convolution algorithm which allows for the use of long duration impulse responses with minimal latency, as described in [11].

Using this convolution approach one could be tempted to skip the physical model part and use measured impulse responses of mechanical spring reverb units instead. This, however, would reduce the flexibility of changing the spring physical parameters as desired and achieving a database with smoothly varying parameters.

Furthermore, an additional feature is developed in order to achieve a smooth transition of sound when changing between the various impulse responses. This is an addition compared to what the first author found as readily available implementations of partitioned convolution, such as the one available in the SuperCollider platform [12], where one cannot switch from one impulse response to another without stopping the sound.

2. HELICAL SPRING

2.1 Continuous Model

A simple model for the vibration of a helical structure, which can reasonably simulate the behavior in the human auditory range is given by Bilbao and Parker in [7] which follows their previous work in [13]. They propose a coupled model where the transverse displacement u, and longitudinal displacement, ζ, along the arclength $x \in [0, L]$, for some unwound spring length L, are coupled. This is described in continuous time by the following system of PDEs, where the subscripts t and x denote a derivative with respect to time and space respectively:

$$u_{tt} = \frac{-Er^2}{4\rho}\left(u_{xxxx} + 2\epsilon^2 u_{xx} + \epsilon^4 u\right) + \frac{E\epsilon}{\rho}\left(\zeta_x - \epsilon u\right) - 2\sigma_t u_t, \tag{1a}$$

$$\zeta_{tt} = \frac{E}{\rho}\left(\zeta_{xx} - \epsilon u_x\right) - 2\sigma_l \zeta_t, \tag{1b}$$

where r is the radius of the wire, which is of circular cross-section and the parameter $\epsilon \approx 1/R$ is a measure of the curvature of the spring, with R being the coil radius. Furthermore, parameters E and ρ are related to the material of the spring, the first being the Young's modulus of elasticity and the second being the material density. The parameters σ_l and σ_t model loss in the longitudinal and transverse direction respectively. The number of physical parameters of the spring can be reduced by rewriting the system in Equation (1) in a scaled form. This can be done by introducing the non-dimensional variables $x' = x/L$, $u' = \epsilon u$ and $\zeta' = \zeta/L$. Furthermore, one can write $\kappa = (r\sqrt{E/\rho})/(2L^2)$ as a measure of the stiffness of the spring and $\gamma = \sqrt{E/\rho}/L$ as the longitudinal wave velocity, both measured in s^{-1}. Then the curvature is normalized as the dimensionless parameter $q = \epsilon L$. This results in the following system (the $'$ superscript that indicates the 'scaled' parameter was removed for brevity):

$$u_{tt} = -\kappa^2\left(u_{xxxx} + 2q^2 u_{xx} + q^4 u\right) + q^2\gamma^2\left(\zeta_x - u\right) - 2\sigma_t u_t, \tag{2a}$$

$$\zeta_{tt} = \gamma^2\left(\zeta_{xx} - u_x\right) - 2\sigma_l \zeta_t. \tag{2b}$$

One can investigate the typical dispersive characteristic of the helical spring by deriving the dispersion relationship of the system given in Equation (2) in the lossless case. This can be done by introducing solutions of the form $u(x,t) = Ue^{j(\omega t + \beta x)}$ and $\zeta(x,t) = Ze^{j(\omega t + \beta x)}$, where U and Z are some constants. Solving for the nontrivial solutions of the resulting system of equations the following relationship governing the dispersion is obtained:

$$(\omega^2 - \gamma^2\beta^2)(\omega^2 - \kappa^2(\beta^2 - q^2)^2 - \gamma^2 q^2) - \gamma^4 q^2 \beta^2 = 0. \tag{3}$$

This results in two separate dispersion relationships, i.e. pairs of (ω, β) functions that satisfy Equation (3), one of

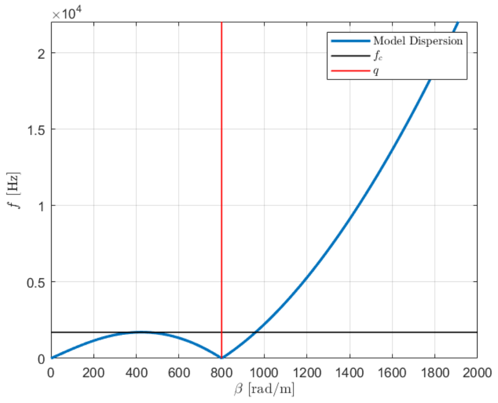


Figure 1. Auditory range solution to the dispersion relationship for a helical spring system, with $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$.

which lies above the limit of human hearing. The other solution, which lies in the audiable range, is illustrated in Figure 1 for a spring with the parameters as given in the caption, where $f = \omega/2\pi$. This figure is highly illustrative of the the interesting behavior of helical springs. It can be seen that dispersion relationship is not monotonic, meaning that components of different wave lengths can have the same temporal frequency. Furthermore, it can be seen that the dispersion relationship shown has a zero at $\beta = q$ (vertical line in the Figure 1) and has a maximum in the lower frequency range at approximately $\beta = q/2$. This is the wave length which corresponds to the frequency f_c given by:

$$f_c = \frac{3\kappa q^2}{8\pi\sqrt{5}}, \tag{4}$$

which is the transition frequency where the dispersion regime changes, and is of perceptual interest for reverberation purposes.

2.2 Finite Difference Scheme

The FDS used for solving this system is an implicit scheme proposed by Bilbao and Parker in [7]. First, the continuous time-space domain over which the PDEs are defined is discretized across a time-space grid of $t = nk$ and $x = lh$, such that the grid function u_l^n denotes a discretized version of $u(x,t)$. The same applies for ζ where $\zeta(x,t) \approx \zeta_l^n$. Then h is the spatial step of the discretization and k is the time step, which results from a desired sampling frequency of the model from $k = 1/f_s$. Furthermore, l and n are integers indexing space and time respectively. As a means of approximating the derivatives in the continuous-time PDEs given in Equation (2), the following finite difference operators are introduced, as per [14]:

$$u_t \approx \delta_{t\cdot} u_l^n = \frac{1}{2k}\left(u_l^{n+1} - u_l^{n-1}\right), \tag{5a}$$

$$u_{tt} \approx \delta_{tt} u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right), \tag{5b}$$

$$u_x \approx \delta_{x-} u_l^n = \frac{1}{h}\left(u_l^n - u_{l-1}^n\right), \tag{5c}$$

$$u_x \approx \delta_{x+} u_l^n = \frac{1}{h}\left(u_{l+1}^n - u_l^n\right), \tag{5d}$$

$$u_{xx} \approx \delta_{xx} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right), \tag{5e}$$

$$\begin{aligned} u_{xxxx} \approx \delta_{xxxx} u_l^n = \frac{1}{h^4}(&u_{l+2}^n - 4u_{l+1}^n + 6u_l^n \\ &- 4u_{l-1}^n + u_{l-2}^n), \end{aligned} \tag{5f}$$

$$u \approx \mu_{t\cdot} u_l^n = \frac{1}{2}\left(u_l^{n+1} + u_l^{n-1}\right). \tag{5g}$$

With this framework, a discretization of the system of equations given in Equation (2) can be written in the following way [7]:

$$\begin{aligned} (1+\eta\kappa k\delta_{xx})\delta_{tt}u = &-\kappa^2\left(\delta_{xxxx}u + 2\overline{q}^2\delta_{xx}u + \overline{q}^4 u\right) \\ &+ \gamma^2 q^2(\alpha + (1-\alpha)\mu_{t\cdot})\left(\delta_{x-}\zeta - u\right) \\ &- 2\sigma_t\delta_{t\cdot}u, \end{aligned} \tag{6a}$$

$$\begin{aligned} (1+\theta\gamma^2 k^2\delta_{xx})\delta_{tt}\zeta = &\gamma^2(\alpha + (1-\alpha)\mu_{t\cdot})\left(\delta_{xx}\zeta - \delta_{x+}u\right) \\ &- 2\sigma_l\delta_{t\cdot}\zeta. \end{aligned} \tag{6b}$$

For explicit numerical schemes one can calculate an update grid function value at a location $x = lh$, i.e. u_l^{n+1}, knowing the values of the grid function u at current (u_l^n) and previous time steps (u_l^{n-1}). Such methods however need to be run at very high sample rates in order to accurately model the dispersion of the system within an audible bandwidth [14]. This can be alleviated by the use of an implicit scheme instead, where the update solution needs to be computed at multiple locations along the grid functions. Expanding the difference operators in Equation (6), it is found that in order to compute the future value at the update point u_l^{n+1}, future values at its neighboring points: u_{l+1}^{n+1} and u_{l-1}^{n+1} are needed, hence a linear coupling among the unknown values of the grid function is introduced. This is due to the introduction of a number of the difference operators δ_{xx} to the left side of the equations and $\mu_{t\cdot}$ to the coupling terms (between u and ζ). The "weights" of these operators are controlled by a number of free parameters in the numerical scheme: η, θ, α. For example if these three parameters are all taken as 0, then the scheme in Equation (6) reduces to an explicit scheme. These parameters can be tuned to provide a more accurate solution in terms of dispersion for a given set of physical spring parameters.

Following the suggestion in [14], $\alpha = 1/2$ and an additional parameter $\overline{q} = (2/h)\sin(qh/2)$ is introduced as an approximation to q. The remaining parameters η and θ are free to change.

Similar to how the dispersion relationship was computed for the continuous model in Equation (3), the numerical dispersion can be derived by inserting the discretized form of the test solutions, i.e. $u_l^n = Ue^{j(\omega kn+\beta lh)}$ and $\zeta_l^n = Ze^{j(\omega kn+\beta lh)}$ in Equation (6). Figure 2 shows a comparison of the numerical dispersion with the model dispersion for different choices of the free parameters. The effect of the choice of $\overline{q}$ is illustrated going from (a) to (b), while the effect of tuning the η and θ parameters is seen going from (b) to (c). The optimal η and θ values differ for each possible combination of spring parameters, hence an optimization procedure is used to compute these optimal values by means of introducing a mean squared error loss function, $\mathcal{L}$ between the values of ω_{model}, resulting from the continuous model dispersion relationship and ω_{FDS}, which results from the dispersion relationship of the FDS, as given in Equation (7):

$$\begin{aligned} &\mathcal{L} = \frac{1}{M}\sum_{i=1}^{M}(\omega_{\text{model},i} - \omega_{\text{FDS},i})^2, \\ &\omega_{\text{model},M} < \pi f_s, \end{aligned} \tag{7}$$

where i is an index of the ω values and $M+1$ is the index at which ω_{model} is bigger than the range of interest. This loss is minimized with respect to the η and θ parameters via a Nedler-Mean simplex algorithm as described in [15]. If one would desire to change the physical parameters of the spring in real-time this optimization needs to be recomputed.

Having a fixed time step k, then h needs to be computed such that the numerical solution remains stable. In [7], an energy-based stability analysis is presented which results in the following stability conditions for h:

$$h \geq 2\gamma k\sqrt{\theta^+}, \tag{8a}$$

$$h \geq \sqrt{\kappa k\left(2\eta^+ + \sqrt{4(\eta^+)^2 + (1+|\cos(qh)|)^2}\right)}, \tag{8b}$$

where $\eta^+ = (\eta + |\eta|)/2$ and $\theta^+ = (\theta + |\theta|)/2$, describe the positive parts of η and θ respectively. An h that satisfies the stability conditions as close to equality as possible, will result in a more accurate numerical solution. Once an h value is chosen, the maximum number of grid intervals can be calculated as $N = 1/h$ (since the length of the spring is normalized to 1 in the scaled system).

Simply supported boundary conditions are considered for the model at the edges, i.e. $u = u_{xx} = \zeta = \zeta_{xx} = 0$ at $l = 0$ and $l = N$, where N is the number of discretized segments of the scaled helical spring. This means that the domain of calculation will be $l \in [1, 2, ..., N-2, N-1]$, as values at the edges will always be 0.

For the actual implementation of the solver, the finite difference scheme in Equation (6) is rewritten in matrix form. Hence, the finite-length column vectors $\mathbf{u}^n = [u_1^n, ..., u_{N-1}^n]^T$ and $\boldsymbol{\zeta}^n = [\zeta_1^n, ..., \zeta_{N-1}^n]^T$ are introduced over the spatial domain (with $l \in [1, \ldots, N-1]$), where T denotes the transpose. The resulting matrix equations are then factorized with respect to $\mathbf{u}^{n+1}$, $\mathbf{u}^n$, $\mathbf{u}^{n-1}$, $\boldsymbol{\zeta}^{n+1}$, $\boldsymbol{\zeta}^n$ and $\boldsymbol{\zeta}^{n-1}$ resulting in a system of the form

$$\begin{aligned} &\mathbf{A}_1\mathbf{u}^{n+1} + \mathbf{B}_1\mathbf{u}^n + \mathbf{C}_1\mathbf{u}^{n-1} \\ &\quad + \mathbf{D}_1\boldsymbol{\zeta}^{n+1} + \mathbf{E}_1\boldsymbol{\zeta}^n + \mathbf{F}_1\boldsymbol{\zeta}^{n-1} = 0, \end{aligned} \tag{9a}$$

$$\begin{aligned} &\mathbf{A}_2\mathbf{u}^{n+1} + \mathbf{B}_2\mathbf{u}^n + \mathbf{C}_2\mathbf{u}^{n-1} \\ &\quad + \mathbf{D}_2\boldsymbol{\zeta}^{n+1} + \mathbf{E}_2\boldsymbol{\zeta}^n + \mathbf{F}_2\boldsymbol{\zeta}^{n-1} = 0, \end{aligned} \tag{9b}$$

for Equations (6a) and (6b) respectively. This can be further merged by introducing a state vector which concatenates $\mathbf{u}^n$ and $\boldsymbol{\zeta}^n$ as: $\mathbf{w}^n = [u_1^n, ..., u_{N-1}^n, \zeta_1^n, ..., \zeta_{N-1}^n]^T$.

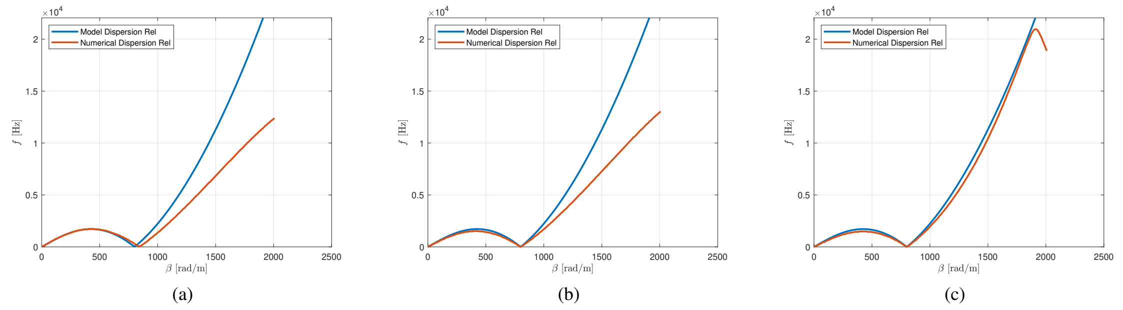


Figure 2. Comparisons of model dispersion of a helical spring with physical parameters $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$ with numerical simulations using different sets of free parameters. (a) $\overline{q} = q$, $\eta = \theta = 0$. (b) $\overline{q} = \frac{2}{h}\sin(qh/2)$, $\eta = \theta = 0$. (c) $\overline{q} = \frac{2}{h}\sin(qh/2)$, $\eta = 0.4313, \theta = 0.000327$.

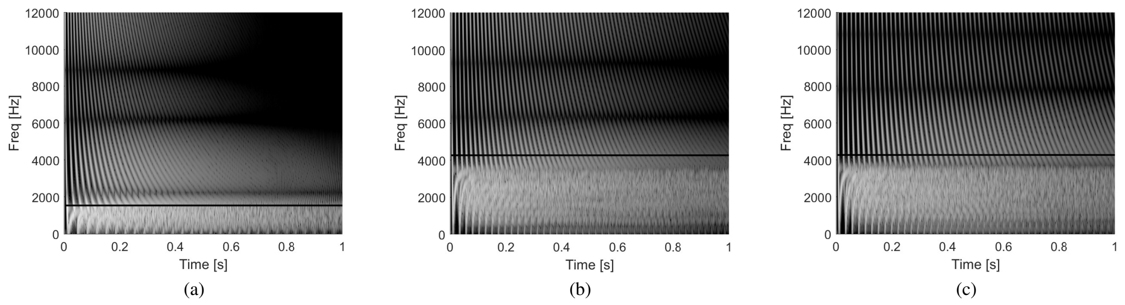


Figure 3. Spectrogram of the impulse response for three helical springs. Black lines represent the model transition frequency f_c. (a) $\kappa = 0.08$, $q = 600$, $\gamma = 1800$. (b) $\kappa = 0.08$, $q = 1000$, $\gamma = 1800$. (c) $\kappa = 0.08$, $q = 1000$, $\gamma = 1000$.

This results in

$$\mathbf{A}\mathbf{w}^{n+1}+\mathbf{B}\mathbf{w}^{n}+\mathbf{C}\mathbf{w}^{n-1}=0, \quad \text{where,}$$
$$\mathbf{A}=\begin{bmatrix}\mathbf{A}_1 & \mathbf{D}_1\\ \mathbf{A}_2 & \mathbf{D}_2\end{bmatrix}$$
$$\mathbf{B}=\begin{bmatrix}\mathbf{B}_1 & \mathbf{E}_1\\ \mathbf{B}_2 & \mathbf{E}_2\end{bmatrix} \tag{10}$$
$$\mathbf{C}=\begin{bmatrix}\mathbf{C}_1 & \mathbf{F}_1\\ \mathbf{C}_2 & \mathbf{F}_2\end{bmatrix}$$

from which an update equation to the state vector can be computed as

$$\mathbf{w}^{n+1}=\mathbf{A}^{-1}\left(-\mathbf{B}\mathbf{w}^{n}-\mathbf{C}\mathbf{w}^{n-1}\right). \tag{11}$$

As can be seen, a large matrix inversion is necessary for solving the system, which along with the optimization of the scheme's free parameters necessary to minimize the numerical dispersion, causes the solution to be impractical with regards to a direct real-time implementation.

3. IMPLEMENTATION

This section details the implementation of the helical spring presented in the previous section and the development of the spring reverb interface. A demonstrative video can be found at [16].

3.1 Impulse Response Database

Impulse responses are generated by exciting the helical spring at one end (at the first free point, i.e. $l = 1$) with a sine sweep covering the human auditory perception range, 20 Hz to 20 kHz and measuring the output at the other end of the spring (the last free point, i.e. $l = N - 1$). Both the excitation and pickup are carried out in terms of the transverse displacement of the spring u, and is consistent with [10]. The initial sine sweep is then deconvolved from this output leaving only the impulse response of the system. For the loss parameters, values suggested in [14] are used: $\sigma_l = \sigma_t = 1.65$.

A database of impulse responses is generated with the aim that there is a distinct difference between each impulse response. This is achieved by systematically changing the parameters of the model in a well-informed manner. Looking again at the dispersion relationship of the continuous model, it is clear that a highly important parameter is q, as it governs both the location where there is a zero, as well as having a big weight in the transition frequency f_c, as it is squared in the numerator in Equation (4). Furthermore, the γ parameter is a scaled version of the longitudinal wave velocity and therefore plays a role in the density of the echoes of the spring.

This can be seen in Figure 3 where two distinct dispersion zones appear: for frequencies above the transition f_c

the behavior is essentially bar-like with higher frequencies travelling faster, while below f_c solutions of differing wave-numbers are possible, [14]. Comparing (a) to (b) one can see that a higher q value raises the transition frequency, while comparing (b) to (c) one can see that a higher γ leads to greater echo density.

For the use in the spring reverb interface, it was decided to keep $\kappa = 0.08$ as a constant value and only vary the other two parameters. Since variations in q are heuristically found to be more distinct perceptually, a denser variation for this parameter is chosen. In total, 27 values are used ranging from 200 to 600 in increments of 25, then from 600 to 1000 in increments of 50 and from 1000 to 1200 in increments of 100. This was done to somewhat mimic the fact that the relationship between frequency and pitch is exponential. This holds when remembering that f_c is directly proportional to q^2. Furthermore two values of γ are considered, 1000 and 1800. These are found to provide a good distinction in the sound of the resulting impulse responses. A smaller difference in these values is not very noticeable, while for significantly higher values of γ, the accuracy of the numerical solution in terms of numerical dispersion suffers. As a result of these choices, this leads to a total of 54 impulse responses. This chosen variation in the spring physical parameters directly translates to a variation in f_c, the dispersion regime transition frequency, by means of Equation (4). Furthermore, a measure of the delay time of the springs in the low frequency dispersive region, T_d, can be derived based on the spring parameters using a relationship given by Parker and Bilbao in [13], $\mathrm{T}_d \approx 4LR/(r\sqrt{E/\rho})$. The average T_{40} decay time of these impulse responses is calculated to be of 2.12 seconds, using Schroeder's backward integration method [17].

3.2 Impulse Responses - Convolution

Since the helical spring system is linear time invariant (LTI) it follows that if one knows the impulse response of this system one can determine its output via the theory of convolution [18]. In time domain this is expressed as

$$y_c[n] = x_c[n] * h_c[n] = \sum_{m=-\infty}^{+\infty} (x_c[m]h_c[n-m]), \quad (12)$$

where x_c is the input to a system which is described by the impulse response h_c and y_c is its output. The subscript c (for convolution) is used to avoid confusion with the variables used in the previous section.

However, convolution is better implemented in frequency domain where Equation (12) transforms into simple multiplication, after which a conversion back to time domain can be performed:

$$\begin{aligned} Y(\omega) &= X(\omega)H(\omega), \\ y_c[n] &= \mathrm{IDFT}(\mathrm{DFT}(x_c[n])\mathrm{DFT}(h_c[n])), \end{aligned} \quad (13)$$

with X being the frequency response of the input, H the frequency response of the impulse response and finally Y is the frequency response of the output, while DFT and IDFT are abbreviations for the discrete Fourier transform and its inverse.

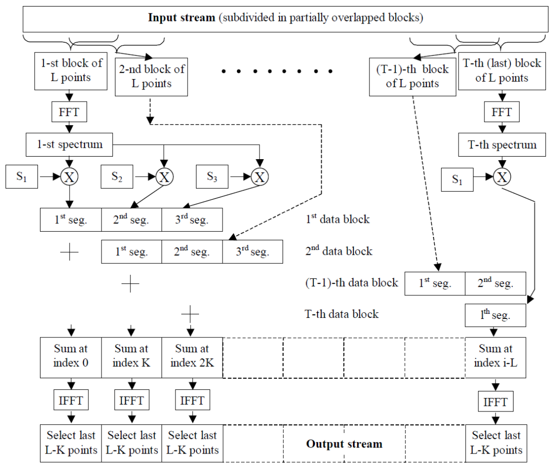


Figure 4. Partitioned convolution process overview, from [11].

Using this method, the length of the impulse response will immediately translate to the latency of the system (excluding other overheads). This is a straightforward and efficient method when the impulse response is of a small length, but the impulse responses of the springs are 2 seconds, or 88200 samples long. A better implementation of convolution for the current task is partitioned convolution, described in detail in [11]. It consists of partitioning the impulse response, h_c, into equally sized blocks of length K. Furthermore, these blocks are treated as separate impulse responses and are convolved using a standard overlap-and-save method [11]. That is, each block is zero-padded to a size $L = 2K$, after which it is transformed to frequency domain via FFT, obtaining a collection of frequency domain filters S. These are then applied to chunks of size L of the input signal while overlapping the results of the latest P input blocks. A diagram of the algorithm is shown in Figure 4, taken from [11]. This is what is used for the current project and the details of the implementation are given in the next section.

A major advantage of the partitioned convolution method is the reduced latency which is only of L rather than 88200 samples.

3.3 Interface Development

The concept behind the interface design is to keep a standard box like design, familiar to most users, while the design of the parameter controls of the interface went hand in hand with the implementation of the algorithm and the various I/O controls. Figure 5 shows the final result.

In order to have the necessary computational power for the heavy partitioned convolution algorithm, as well as to have a standalone interface, it was decided to adopt the Bela platform. The Bela is a small, single-board Linux computer built on the BeagleBone Black that provides high quality audio at ultra-low latency, in addition with a large array of analog and digital I/O options [19]. For this project, the Bela Mini was chosen due to its reduced size

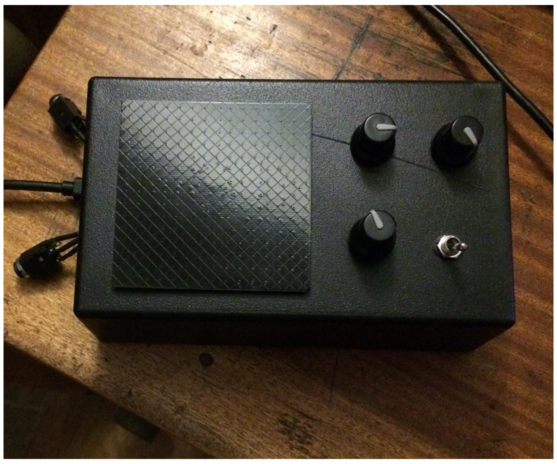


Figure 5. Spring reverb interface

and the fact that it provides all the needed features. Also from the Bela platform, the Bela Trill Square sensor was used, which is a capacitive touch sensor that supports I^{2}C communication. Details regarding the choice of the sensors are given in Section 3.4.

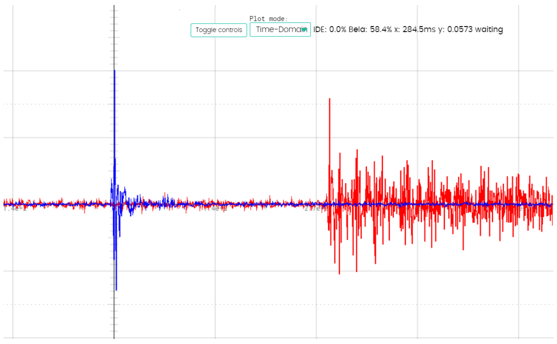
A standard ABS plastic enclosure was used for embedding the Bela and the various sensor/controls. The electrical connections between the various sensors and the Bela board were done on a standard prototype breadboard using jumper wire cables.

The physical model simulations of the spring impulse responses were carried out in Matlab [20], while the implementation of the partitioned convolution algorithm is carried out in the Bela in-browser IDE in C++. The audio sampling rate used is 44100 Hz, while the analog I/O sample rate is 22050 Hz.

The block size for the audio render is taken as 256 samples, while the block size used in the partitioned convolution algorithm (size L described in Section 3.2) is 8192 samples. This means that the convolution processing function is only run every time 8192 new audio samples are added in the audio input circular buffer. Furthermore, to avoid possible underruns, a multi-threaded implementation is carried out for the convolution processing function, making sure that the audio render function always has higher priority.

Moreover it was found that an initial lag between the global input buffer and the global output buffer of 3 times the hop size K is needed. This basically gives the I/O latency of the interface, which results in 279 ms. Figure 6 shows this latency as measured with the in-browser oscilloscope available in the Bela IDE. If one would reduce the block size L to half, i.e 4096 samples for instance, the resulting latency would also be halved.

However, it was found that the real-time capability of the implementation is limited by the maximum number of frequency domain filter blocks S (see Section 3.2), that is how many overlapped bins the impulse response is divided in. It was found that the program cannot run in real time when more than 22 blocks are used, regardless of the block size



IDE: 0.0% Bela: 58.4% x: 284.5ms y: 0.0573 waiting

Figure 6. Illustration of system latency as measured with the in-browser oscilloscope available in the Bela IDE. Blue line is the input signal and red line is the output. There is a lag of approximately 279 ms between the two.

L. So when reducing the size from 8192 to 4096 samples, one can only use impulse responses of length $2048 \cdot 23 = 47104$ samples (22 block sizes of length 4096 overlapped with a hop size of 2048), which is a bit more than 1 second at the audio sample rate used. There appears to be a trade-off between the latency of the implementation and the maximum length of the impulse responses.

3.4 Parameter Mapping

Since spring impulse responses from the available database differed with respect to two physical parameters: curvature parameter q and wave velocity γ, it was desired to have the option to quickly navigate through them with respect to both these dimensions. Hence, the Trill Square capacitive sensor from the Bela platform was used, and each impulse response was mapped to locations on the (x,y) grid position on the square. More specifically, the wave speed γ is mapped to the x-axis while curvature parameter q is mapped to the y-axis.

An important feature of the implementation is that a smooth transition from using one impulse response to another is achieved. This was done by doing all the processing related to all the impulse responses in the setup part of the audio algorithm. Then, when moving from one impulse response to another, two convolutions are in fact carried out for a brief transition time chosen as one audio processing bin, i.e., 8192 samples (size L in the partitioned convolution algorithm). A square-root fade-in/fade-out is then carried out between the two resulting outputs. Moreover, when taking the finger off of the sensor the impulse response used for convolution will be the one associated with the last read position.

Other parameter controls are carried out by means of three potentiometers. One is mapped to a global master volume, lineally mapped between -40 dB and +6 dB. Another controls the dry/wet mixture of the output signal, where the minimum value of the potentiometer gives a fully dry sound and the maximum value gives a fully wet sound. The last parameter mapped to the remaining potentiometer is the number of bins of the impulse response considered in the convolution algorithm.

3.5 From Effect to Instrument

Another interesting addition to the interface is the fact that the user can change between two different audio input sources with the use of a toggle switch. The first is an audio-in jack, which can be connected to any other sound producing device, in essence turning the interface into an effect processor. Furthermore, an electret microphone was embedded in a side panel of the box, for which a signal amplifier circuit is built on the breadboard. When switching to the mic input, the interface can be used similarly to an instrument. One can tap, scratch, whistle and speak into the microphone while changing the various real-time parameter controls and produce interesting sounds. Additional investigations regarding this potential use coupled with feedback are planned.

4. CONCLUSION

The work described in this paper focused on combining physical modelling sound synthesis techniques with convolution with the aim to supplement each other towards a real-time implementation of a spring reverb, whose physical parameters can be adjusted on the fly. While the physical model with adjustable parameters proved to be too computationally expensive for a real-time implementation, using partitioned convolution provided a way around this. An important addition to this approach is providing a way to have a smooth transition from the use of one impulse response to another, and the fact that the impulse responses are physically related to each other.

A physical interface was built which allows for an expressive manipulation of an input sound, via the provided parameter controls.

Future work will focus on a more streamlined design of the interface, by means of a CAD software together with laser cutting or 3-D printing manufacturing options. As for the electrical components, a more sturdy strip board implementation followed by a custom PCB design of the circuit is planned. Additional investigations regarding the software implementation can focus on pushing the limits of the model. For instance, how many impulse responses can be easily processed in the audio setup and mapped to the Trill square. Since the physical model for the helical spring is mainly dependent on 3 parameters, an additional dimension representing the κ parameter can be added to the mapping, perhaps using a pressure sensor or making use of the area of the touch.

While the interface was designed with the concept of a spring reverb unit in mind, what it is essentially is a real-time convolution effect processor, with an implemented use case as a spring reverb. One can easily use other impulse responses and completely change the nature of the interface. Such other use cases will be looked into. Lastly, an important observation is that this is a mono audio device, and expanding it to stereo, perhaps mapping different impulse responses to each output channel, would be a welcome feature.

5. REFERENCES

[1] L. Hammond, *"Electrical musical instrument," US Patent No. 2230836*, Feb 1941.

[2] V. Välimäki, J. D. Parker, L. Savioja, J. O. Smith, and J. S. Abel, "Fifty years of artificial reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1421–1448, 2012.

[3] J. S. Abel, D. P. Berners, S. Costello, and J. O. Smith, "Spring reverb emulation using dispersive allpass filters in a waveguide structure," in *Audio Engineering Society Convention 121*. Audio Engineering Society, 2006.

[4] J. S. Abel and E. K. Canfield-Dafilou, "Dispersive delay and comb filters using a modal structure," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1748–1752, 2019.

[5] V. Välimäki, J. Parker, and J. S. Abel, "Parametric spring reverberation effect," *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 547–562, 2010.

[6] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects*, 2011, pp. 473–522.

[7] S. Bilbao and J. Parker, "A virtual model of spring reverberation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 799–808, 2010.

[8] W. Wittrick, "On elastic wave propagation in helical springs," *International Journal of Mechanical Sciences*, vol. 8, pp. 25–47, 1966.

[9] S. Bilbao, "Numerical simulation of spring reverberation," in *DAFX: Digital Audio Effects*, 2013.

[10] M. Van Walstijn, "Numerical calculation of modal spring reverb parameters," in *Proceedings of the 23rd International Conference on Digital Audio Effects*, 2020, pp. 38–45.

[11] E. Armelloni, C. Giottoli, and A. Farina, "Implementation of real-time partitioned convolution on a dsp board," *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pp. 71–74, 2003.

[12] S. Wilson, D. Cottle, and N. Collins, *The SuperCollider Book*. The MIT Press, 2011.

[13] J. Parker and S. Bilbao, "Spring reverberation: A physical perspective," 2009.

[14] S. Bilbao, *Numerical Sound Synthesis*. John Wiley and Sons, Ltd, 2009.

[15] J. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on Optimization*, vol. 9, pp. 112–147, 1998.

[16] M. G. Onofrei. A physical modelling based adjustable spring reverb effect. Youtube. [Online]. Available: https://www.youtube.com/watch?v=HZRWU0b45vg

[17] M. R. Schroeder, "New method of measuring reverberation time," *The Journal of the Acoustical Society of America*, vol. 37, no. 6, pp. 1187–1188, 1965.

[18] T. H. Park, *Introduction To Digital Signal Processing: Computer Musically Speaking*. World Scientific Publishing Co Pte Ltd, 2008.

[19] A. McPherson and V. Zappi, "An environment for submillisecond-latency audio and sensor processing on beaglebone black," vol. 2, pp. 965–971, 2015.

[20] MATLAB, *version (R2020b)*. Natick, Massachusetts: The MathWorks Inc., 2019.