Real-time Implementation of a Physical Model of the Persian Santur

Simon Rostami Mosen (srosta17@student.aau.dk)[1], **Oddur Ingi Kristjansson** (okrist17@student.aau.dk)[2], **Ali Adjorlu** (adj@create.aau.dk)[3], and **Stefania Serafin** (sts@create.aau.dk)[4]

[1,2,3,4]*Department of Architecture, Design and Media Technology*, **Aalborg University Copenhagen**, A. C. Meyers Vænge 15, Copenhagen SV, 2450 DK

ABSTRACT

The Santur is an old Persian instrument that falls under the dulcimer family. It consists of 18 notes, each with 4 strings, alternating between sets of brass and steel. The strings are excited with wooden hammers called Mezràb, resulting in a vibrant sound. This paper presents a real-time implementation of a physical model of the Santur. The result is a Virtual Studio Technology (VST) which can be used for audio production. Using finite-difference time-domain (FDTD) methods, the physical model is implemented. Although the results show that the synthesized samples are perceived to be less realistic than the real recordings, several participants stated that it was difficult to tell the synthetic renderings from the recorded samples.

1. INTRODUCTION

The Persian Santur is a hammered trapezoidal dulcimer [1]. It originates from Iran and has later inspired a wide range of similar instruments such as the Indian 'Santoor', the Chinese 'Yangqin', and the Thai 'Khim' [1]. It is a common assumption that these instruments have been traded along the ancient Asian trade route, the Silk Road, and thereby have been spread across countries and cultures [2].

As seen in figure 1, the Santur consists of 18 sets of strings, alternating between steel and brass strings. With four strings of similar tuning per set, it results in a total of 72 strings [1]. Each string is mounted on a string holder on one side of the instrument, a tuning peg on the other side of the instrument and rests on a movable bridge on the soundboard [1]. Each note [1] is excited using a pair of delicate wooden hammers called the Mezràb (figure 2) [1].

In this paper, we present a real-time implementation of a physical model of the Persian Santur. As digital emulations of acoustic instruments are no new practice, different types of synthesis have been used in the process of achieving realistic sound emulations [3], such as frequency modulation (FM), wavetable synthesis, and digital waveguide synthesis. FM synthesis dominated the digital emulations

[1] https://github.com/sibilu/Santur/tree/main/Santur%20Tuning

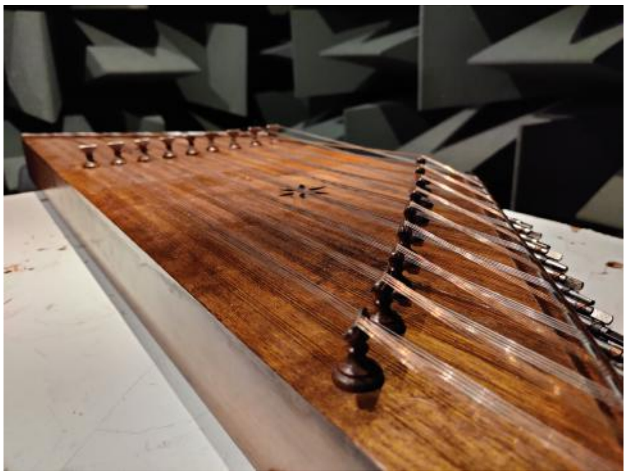

Figure 1: *Persian Santur.*

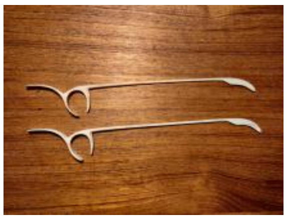

Figure 2: A set of hammers (Mezràb).

Figure 3: Tuning pegs on the right side of the Santur.

in the '80s where synthesizers such as the Yamaha DX7 defined the sound of pop, country, and R&B music of the decade [4]. Wavetable synthesis, in contrast, is a form of sampling, where a short sample is looped and depending on the sample, different instrument-like sounds can be generated [5]. The Karplus-Strong algorithm, a digital waveguide synthesis method, was published in 1983 and also produced impressive, string-like sounds using only a noise burst and a low pass filter [6]. While the aforementioned methods were more realistic compared to other synthesis methods, these techniques focused on modeling the audible sound (spectrum, waveform) rather than the sound source itself [7]. Physical modeling, on the other hand, is a synthesis technique focused on digitally imitating the physical properties of the sound source, such as the resonators, and is often a set of partial differential equations (PDE) [3] although it consists of other techniques as well, such as mass-spring networks, waveguide synthesis, and more [7]. For the PDE, the aim is to solve these equations using a numerical approximation to yield a waveform [3].

2. MODELS

This section will go into the process of physically modeling a string. Starting from a simple 1D wave equation, to a plucked damped stiff string, the equations will be presented and described how they are used in the Santur implementation.

2.1 1D Wave Equation

The one-dimensional wave equation (eq. 1), is the foundation used to describe behaviors in strings [3].

$$u_{tt} = \gamma^2 u_{xx} \tag{1}$$

The equation above is a second-order PDE in the dependent variable $u = u(x,t)$, where x represents distance, t represents time, and γ is the wave speed [3]. In order to keep the system numerically stable, λ, often referred to as the Courant condition, must be ≤ 1 [8]. The wave equation itself is mainly useful as a test problem as it builds upon the following physical assumptions [9]:

1. The mass of the string being homogeneous. The string is perfectly elastic and does not offer any resistance to bending.
2. The tension of the string is large enough that gravitational force on the string can be neglected.
3. The string performs small transverse motions in a vertical plane.

Frequency-domain analysis of a vibrating Santur string would reveal that it is not entirely harmonic. As the 1D wave equation describes a frequency spectrum without inharmonicity [10], the wave equation itself is not a precise model to describe Santur strings.

2.2 Ideal Bar

As the wave equation describes an ideal string, it does not take the string's stiffness into account [3]. Stiffness is given by the material's resistance to deformation and is what introduces inharmonicity to the model. To introduce stiffness, and thereby inharmonicity, the equation of an ideal bar is introduced

$$u_{tt} = -k^2 u_{xxxx} \tag{2}$$

where the stiffness parameter, κ, is defined by:

$$\kappa = \sqrt{\frac{EI}{\rho A L^4}} \tag{3}$$

where ρ is the material density of the medium, A is the cross-sectional area, E is Young's modulus, I is a parameter often referred to as the bar moment of inertia and L is the length of the string [3]. Combining the wave equation (eq. 1) and that of the ideal bar (eq. 2) yields the equation of a stiff string (eq. 4).

$$u_{tt} = \gamma^2 u_{xx} - \kappa^2 u_{xxxx} \tag{4}$$

2.3 Damped stiff string

As a result of radiation and internal losses, the vibrations of a real string decay over time [3]. This can be modeled by adding loss/damping to the equation [3]. In real strings, not all components decay at the same rate, as loss is generally increasing with frequency [3]. Therefore, both frequency-independent and frequency-dependent damping is added to the stiff string model as follows:

$$u_{tt} = \gamma^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx} \tag{5}$$

At this point, the stability condition (section 2.1) has expanded (eq. 8 in table 1) [11].

2.4 Body

In this project, the body of the Santur is not mathematically modeled. On the assumption that the body of a Persian Santur is a linear system, an impulse response (IR) of the body was recorded to be applied to the signal using convolution. The IR was recorded in an anechoic chamber using a Shure SM57 microphone and the body was excited by hammering on one of the bridges of the Santur. The IR was low-cut at a cutoff frequency of 80 Hz with a 24 dB roll-off as the information below 80 Hz caused noticeable audio disruption. The final IR can be found here [2].

3. DISCRETISATION

The damped stiff string equation (eq. 5) is discretized using finite-difference time-domain (FDTD) methods [3]. The operators for discretizing PDEs are described in the following section.

3.1 Operators

To denote discrete-time approximations to continuous-time, the following operators are used.

3.1.1 First-order difference operators

Time difference operator:

$$u_t \approx \begin{cases} \delta_{t+} u_l^n = \frac{1}{k}\left(u_l^{n+1} - u_l^n\right) & \text{Forw. time diff.} \\ \delta_{t-} u_l^n = \frac{1}{k}\left(u_l^n - u_l^{n-1}\right) & \text{Backw. time diff.} \\ \delta_{t\cdot} u_l^n = \frac{1}{2k}\left(u_l^{n+1} - u_l^{n-1}\right) & \text{Cent. time diff.} \end{cases} \tag{6}$$

Space difference operator:

$$u_x \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h}\left(u_{l+1}^n - u_l^n\right) & \text{Forw. space diff.} \\ \delta_{x-} u_l^n = \frac{1}{h}\left(u_l^n - u_{l-1}^n\right) & \text{Backw. space diff.} \\ \delta_{x\cdot} u_l^n = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right) & \text{Cent. space diff.} \end{cases} \tag{7}$$

3.1.2 Second-order difference operators

Time difference operator:

$$u_{tt} \approx \delta_{tt} u_l^n = \delta_{t+}\delta_{t-} u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) \tag{8}$$

[2] https://github.com/sibilu/Santur/tree/main/Plot%20of%20Impulse%20Reponse

Space difference operator:

$$u_{xx} \approx \delta_{xx} u_l^n = \delta_{x+}\delta_{x-} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \tag{9}$$

3.2 Discrete Models

Using the operators in section 3.1, the equation of a damped stiff string (eq. 4) can be discretized as follows:

$$\delta_{tt} u_l^n = \gamma^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xx}\delta_{xx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n \tag{10}$$

To implement the discretized damped stiff string (eq. 10), the equation should use the expanded form of the operators in section 3.1 before solving for u_l^{n+1}.

3.3 Boundary Conditions

There are different boundary conditions available such as lossy conditions or the lossless *Dirichlet* and *Neumann* conditions [3]. However, in this project, the lossless *Clamped* boundary condition was chosen to reduce CPU load, as mentioned in section 4.2. For the clamped boundary condition, the first two and last two indices in the implemented array are left untouched at the value of 0 where points are only calculated from $l = [2, ..., N-2]$.

4. IMPLEMENTATION

The real-time implementation of the physical model has been implemented in C++ using the JUCE framework [3]. The code structure is set up in 3 main files. The `PluginProcessor` handles the setup of the plugin and audio buffer handling. From there, the `SanturString` class is called. There are 18 instances of this class as it replicates a single note of the Santur. From there, the `DampedString` class is called. This class handles the damped string calculations and implements 4 strings for each note. For a clearer understanding of each variable and the equation associated with it, table 1 lays out all relevant variables and equations used.

4.1 Structure of Implementation in JUCE

This section will describe the implementation for each class individually. The full implementation is found here [4].

4.1.1 PluginProcessor Class

The PluginProcessor class is where each note, a total of 18, is initialized as a unique pointer. Furthermore, individual vectors for string tension values (double), string length values (double), MIDI note values (int), and played notes (bool) are created. Here, values such as damping coefficients, pluck position, and output position (of the string) are initialized. For the damping coefficients, a value of 1.14 is used for σ_0, and a value of 0.0006 for σ_1 is used.

[3] https://juce.com/

[4] https://github.com/sibilu/Santur/tree/main/Juce%20Implementation

	Var	Descr.	Equation
1	L	String Length	
2	t	String Tension	
3	r	String Radius	$\frac{4^{1/4}B^{1/4}t^{1/4}L^{1/2}}{\pi^{1/4}E^{1/4}}$
4	a	Cross-Sectional Area	πr^2
5	i	Moment of Inertia	$\frac{\pi r^4}{4}$
6	c	Wave speed	$\sqrt{\frac{t}{pA}}$
7	κ	Stiffness	$\sqrt{\frac{EI}{pA}}$
8	h	Stability Condition	$\sqrt{\frac{c^2k^2+4\sigma_1 k+\sqrt{(c^2k^2+4\sigma_1 k)^2+16\kappa^2k^2}}{2}}$
9	λ^2	Courant Number	$\frac{c^2k^2}{h^2}$
10	μ^2	Constant Number	$\frac{\kappa^2k^2}{h^4}$
11	N	Gridpoints	$\frac{L}{h}$
12	B_1	Constant Number	$\sigma_0 k$
13	B_2	Constant Number	$\frac{2\sigma_1 k}{h^2}$
14	A_1	u_l^n	$2 - 2\lambda^2 - 6\mu^2 - 2B_2$
15	A_2	u_{l+-1}^n	$\lambda^2 + 4\mu^2 + B_2$
16	A_3	u_{l+-2}^n	$-\mu^2$
17	A_4	u_l^{n-1}	$B_1 - 1 + 2B_2$
18	A_5	u_{l+-1}^{n-1}	$-B_2$
19	D	u_l^{n+1}	$\frac{1}{1+\sigma_0 k}$

Table 1: *Variables and equations used.*

Concerning decay time, these values were carefully selected by comparing the synthesized sound with a recording of the real instrument. As mentioned in section 1, the Santur strings are both of brass and steel material. The Young's Modulus values for both the steel [5] and brass were gathered from [12] and converted from GPa to Pa. The material density is also initialized here, with a value of 7700 kg/m^3 for steel [13] and 8400 kg/m^3 for brass [14]. For the inharmonicity factor β, a value of 0.00031 is used [15]. Having the aforementioned values, it was possible to calculate the remaining physical values of the instruments. The *r*, *a*, and *i* values for each string were calculated with equations 3-5 in table 1.

Each unique pointer note is then initialized with the appropriate values and moved into JUCE's `OwnedArray`. This allows for accessing each note through a for-loop for an easier understanding of the code.

When a MIDI note is triggered, a for-loop checks whether the MIDI value matches the value in the previously initialized MIDI vector, and if it does, booleans within the *playNote* and *triggerProcess* are switched to true and a timer of 10 seconds is started. The purpose of the timer will be further explained in section 4.2. Lastly, the *processAndUpdateStrings()* and *outputSound()* functions are called before the sound is outputted. These last two functions cycle through all 18 notes and call functions within the `SanturString` class to set the pluck location, excite them with the appropriate excitation window and retrieve the sound.

4.1.2 SanturString Class

As mentioned, this class represents each note on the Santur. In the constructor, the parameter used to initialize each note is mapped to four different `NamedValueSet`, a class holding a set of variable objects, which are then used for each string of the note. The reason for this approach is

[5] Here, 'Steel, stainless AISI 302' was used

to have the chance to change individual parameters within the note and in this case, it allows for relative de-tuning of each string of the note. Each function called from the `PluginProcessor` goes through this class and is forwarded to the `DampedString` class from there.

4.1.3 DampedString Class

This class is the core of the string implementation. Here, each instance represents a single string on the instrument and all calculations concerning it take place. In here, the calculations for wave-speed, κ, etc., take place. Furthermore, the three state vectors *u*, *uNext*, and *uPrev* are initialized and resized to a size of $N + 1$. To get the output position relevant to L, the outPos variable is used as follows:

$$\textit{uNext} = round(N + 1) * outPos \tag{11}$$

To simulate a plucking action, the model must be initialized to a certain shape [3]. For this model, a triangle window [16] is used to excite the model. Here, the width of the excitation window was set to a fixed size of 10 gridpoints.

The discretized damped stiff string (eq. 10) can be found in the *DampedString.cpp* file. This implementation is inspired by Willemsen's implementation [17] as mentioned in section 4.2.

4.2 Optimizing CPU Usage

After having successfully implemented all 18 notes of the Santur, it was experienced that the CPU was running at a constant 70-100%+ [6]. Therefore, it was important to figure out a way to optimize the CPU usage. Inspired by Willemsen's stiff string implementation [17], the three state vectors *u*, *uNext*, and *uPrev* were changed into pointer vectors instead. Furthermore, as the boundaries were clamped, initializing both the first and last two indices each loop iteration was removed. Lastly, repetitive calculations of non-changing values were removed and made into their static variables [17] (equations 12-19 in table 1).

Another step taken was reducing the number of simultaneously active notes. Initially, all 18 notes were being processed constantly, resulting in 72 damped stiff string calculations at all times. Therefore, it was decided to limit the number of active notes to 6. This was done by implementing a circular buffer that stored notes that have been played. Once the buffer becomes full, the 'oldest' note played is removed. Conditional statements were included to make sure that triggering the same notes again would not affect the buffer, but only when more than 6 different notes are played does the 'oldest' note get removed. An additional step was taken by adding a timer of 10 seconds (section 4.1.1) restarting each time a note is played. That way, it was made sure that no notes are processed if none have been triggered after 10 seconds.

[6] Macbook Pro (15-inch 2016) 2.6GHz Quad-Core Intel Core i7

These steps reduced the CPU usage to 30-45% when 6 notes are being processed simultaneously.

4.3 Convolution

As mentioned in section 2.4, an IR was recorded to convolve the synthesized signal with the Santur body. Therefore, for the evaluation (section 5), each used note was exported and convolved offline (non-real-time) separately with the impulse response at a convolved-to-dry ratio of 20% convolved signal to 80% dry signal. A video demonstration of the real-time implementation of the Santur can be found here [7] , where both sound examples and GUI are found.

5. EVALUATION

The quality of the implemented physical model was evaluated in a listening experiment. Inspired by the procedure of [18], 18 participants were asked to describe the realism of audio renderings from the implemented physical modal and recordings from the Santur. Here, 58% had 7+ years of experience, 15.8% had 3-6 years of experience, and 26.3% had 0-2 years of experience with playing instruments/working with music. The evaluation was conducted online using Google Forms and as it was browser-based, no extra software had to be installed. Participants were encouraged to use headphones, and adjust the playback volume to a comfortable level.

6 single notes from the Santur were randomly selected: D3 - F3 - F4 - G4 - C4 - C5, consult table here [8] for more information. Audio recordings of the Santur were pooled with 6 renderings (of matching notes) from the physical model as well as 6 identical renderings from the physical model with -60dB of white noise added to match the noise level in the recordings of the real Santur. The recordings were conducted in an anechoic room and recorded using a Shure SM57 microphone through a TC Electronic Impact Twin audio interface into Ableton Live 10. Both the recordings and the renderings had a sampling rate of 44100 and were normalized to -35dB RMS. An image of the recording setup can be seen here [9] .

The participants were informed that they participated in an experiment about the perceived realism of Santur sounds but did not have any information about the origin of the samples they were listening to.

The evaluation was two-fold:

A Participants were asked to rate how realistic they perceived each sample, on an 11 point Likert scale labeled from "very unrealistic" (0) to "very realistic" (10).

B Using the same 18 stimuli as in test A, participants were asked to decide if they thought the sample was digitally made (synthesized) or recordings of an acoustic instrument.

[7] https://github.com/sibilu/Santur/tree/main/Video%20Demonstration
[8] https://github.com/sibilu/Santur/tree/main/Santur%20Tuning
[9] https://github.com/sibilu/Santur/tree/main/Plots

The order of the samples was randomized through both parts of the experiment to minimize response bias.

6. RESULTS AND DISCUSSION

As seen in table 2, part A of the evaluation showed that the recorded stimuli on average was perceived 1.69 points more realistic than the synthesized stimuli and 1.40 points more realistic than synthesized stimuli with added noise.

Cond.	Mean	Median	St. Dev.
Recorded	6.46	7	2.97
Synthesized	4.77	5	2.49
Synth. + -60dB noise	5.06	5	2.50

Table 2: *Descriptive statistics of part A.*

The minimum, first quartile, median, third quartile, and maximum for each type of stimuli is illustrated with a boxplot in figure 4.

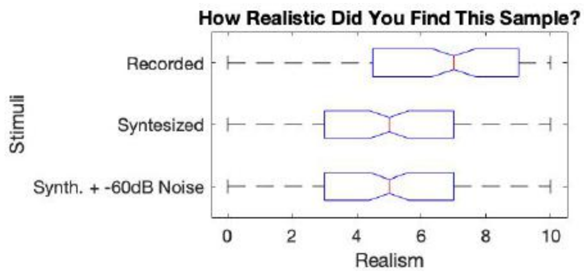

Figure 4: *Summarized results from evaluation part A.*

From the boxplot, it is also seen that the two synthesized conditions yielded almost identical results.

The results of part B of the evaluation are illustrated below in figure 5. The results show that the origin of the recorded stimuli was correctly determined in 69% of the cases, while the origin of the synthesized renderings was correctly determined in 79% of the cases without noise added and 70% with added noise, meaning that the majority was not in doubt whether a sample was recorded or synthesized.

If the stimuli of each condition are divided into the three lowest and three highest notes, results indicate that the lower notes were perceived to be more synthetic than the higher notes in the synthetic condition (low: 83.3% synthetic, high: 74.1% synthetic). This is in contrast to the recorded condition where the higher notes were perceived to be more synthetic than the lower notes (low: 24.1%, high: 37.0% synthetic). The raw results can be found here [10] .

Several participants mentioned that it was difficult to determine if the stimuli were synthetic or recorded. A participant stated that he was: "Surprisingly uncertain regarding the answers" and that he "expected it to be easier to hear which ones were digitally made". Another participant also stated that: "I feel like I’ve just only listened to ’real’

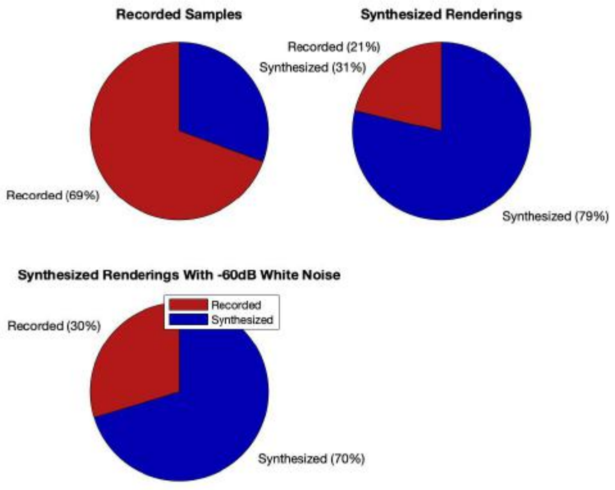

Figure 5: *Summarized results from evaluation part B.*

sound". Another participant expressed that it was difficult to rate the realism of the stimuli when you did not know how the real acoustic instrument would normally sound.

As the results show, the rendered audio from the physical model was perceived to be less realistic than the recordings of the instrument in both parts of the evaluation. Before regarding these as definite findings, several things should be taken into consideration. Firstly, the recordings of the Santur were made using a microphone without a flat frequency response [11] . This was not compensated for in the synthetic renderings and could have caused unnecessary coloring of the recorded samples.

The IR used in this physical model was recorded in an anechoic chamber by hammering the Santur at one of the bridges and recording the response. This yielded a relatively low-quality IR where much high-end was lost, why the convolved signal was mixed at a 20/80 ratio with the dry signal, as mentioned in section 4.3. Having a cleaner IR would allow for increasing the ratio of convolved signal to dry signal, which could enhance the realism of the overall model.

Furthermore, the recorded samples used in the evaluation were of a ’hammered’ string with the Mezrab compared to the triangle window excitation of the synthesized samples. This, naturally, results in a different timbre and sound output which introduces dissimilarity between the samples, specifically as the Santur is usually not played without the Mezrab.

7. CONCLUSION

In this paper, a real-time physical model of the Persian Santur has been presented. The physical model was evaluated against its real counterpart and it was found that recordings of the real Santur were perceived to be more

[10] github.com/sibilu/Santur/tree/main/Evaluation%20Results

[11] https://pubs.shure.com/guide/SM57/en-US

realistic than renderings of the physical model, although several participants stated that it was difficult to tell the synthetic renderings from the recorded samples. These findings should not be taken as definite as aspects regarding the recording of the Santur samples could have affected the perceived realism of the samples.

8. FURTHER WORK

To improve the physical model proposed in this paper, several things could be investigated. As the Persian Santur is usually played using wooden hammers (mezrab), an appropriate hammer model could be implemented as opposed to the current triangle excitation used in this model.

As the current IR was recorded by hammering on one of the bridges of the Santur, recording a new IR where the body is excited using a sine sweep could potentially yield a cleaner IR. Additionally, the convolution process should be included in the real-time implementation. Further implementation could also consider the effect of sympathetic strings, as when one string is excited on the Santur, other non-excited strings may resonate, resulting in complex harmonies. This was not currently possible in this project due to CPU.

Concerning interaction, MIDI-keyboards are not optimal for interacting with string instruments as the keyboard-based interaction does not allow for note-by-note manipulation of aspects such as pluck position and damping of the strings. These aspects of the physical model, however, could be evaluated interactively by implementing the physical model in a Virtual Reality (VR) environment where the user could interact with a virtual model of a Santur, coupled with the physical model presented in this paper. Furthermore, controllers like the Sensel Morph [12] could allow for more expressive control of the model.

Acknowledgments

We thank Silvin Willemsen for his untiring helpfulness.

9. REFERENCES

[1] P. Heydarian and J. D. Reiss, "The persian music and the santur instrument." in *ISMIR*, 2005, pp. 524–527.

[2] M. Clark and B. Museum of Fine Arts, *Sounds of the Silk Road: Musical Instruments of Asia*. MFA Publications, 2005.

[3] S. Bilbao, *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. John Wiley & Sons, 2009.

[4] M. Lavengood, "What makes it sound'80s? the yamaha dx7 electric piano sound," *Journal of Popular Music Studies*, vol. 31, no. 3, pp. 73–94, 2019.

[5] D. C. Massie, "Wavetable sampling synthesis," in *Applications of Digital Signal Processing to Audio and Acoustics*. Springer, 2002, pp. 311–341.

[6] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43–55, 1983.

[7] V. Välimäki and T. Takala, "Virtual musical instruments—natural sound using physical models," *Organised Sound*, vol. 1, no. 2, pp. 75–86, 1996.

[8] S. Bilbao, C. Desvages, M. Ducceschi, B. Hamilton, R. Harrison-Harsley, A. Torin, and C. Webb, "Physical modeling, algorithms, and sound synthesis: The ness project," *Computer Music Journal*, vol. 43, no. 2-3, pp. 15–30, 2019.

[9] E. Kreyszig, *Advanced Engineering Mathematics 9th Edition*. Wiley India Pvt. Limited, 2011.

[10] S. Bilbao, B. Hamilton, R. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," *Springer Handbook of Systematic Musicology*, pp. 349–384, 2018.

[11] S. Willemsen, "Finite-difference schemes (pt. 2) - physical models for sound synthesis," March 2021.

[12] E. ToolBox. Young's modulus - tensile and yield strength for common materials. [Online]. Available: https://www.engineeringtoolbox.com/young-modulus-d_417.html

[13] T. W. Material. Weight density of stainless steel 304, 316, 316l 303 in lb/in3, g/cm3, lb/ft3, kg/m3. [Online]. Available: https://www.theworldmaterial.com/weight-density-of-stainless-steel/

[14] A. Calc. Density of brass (material). [Online]. Available: https://www.aqua-calc.com/page/density-table/substance/brass

[15] P. Heydarian and L. Jones, "Measurement and calculation of the parameters of santur," *Canadian Acoustics*, vol. 36, no. 3, pp. 86–87, 2008.

[16] Mathworks. sigwin.triang class. [Online]. Available: https://www.mathworks.com/help/signal/ref/sigwin.triang-class.html

[17] S. Willemsen. Simple string app. [Online]. Available: https://github.com/SilvinWillemsen/SimpleStringApp

[18] D. Moffat and J. D. Reiss, "Perceptual evaluation of synthesized sound effects," *ACM Transactions on Applied Perception (TAP)*, vol. 15, no. 2, pp. 1–19, 2018.

[12] https://morph.sensel.com