A Model of the Shamisen based on Finite Difference Schemes

Master's Thesis

Titas Lasickas

Aalborg University
Architecture, Design and Media Technology

Architecture, Design and Media Technology
Aalborg University
http://www.aau.dk

Title:
A Model of the Shamisen based on Finite Difference Schemes

Theme:
Master's Thesis
Modelling Physical Systems

Project Period:
Fall Semester 2020

Participant(s):
Titas Lasickas

Supervisor(s):
Silvin Willemsen
Stefania Serafin

Copies: 1

Page Numbers: 56

Date of Completion:
December 18, 2020

Abstract:

Synthesising musical instruments and using their digital versions in music production and performance without the need of owning and learning the real instrument is a highly popular demand by the musicians and producers alike. Currently, more and more synthesizers are being developed, decreasing the need for going through a tedious recording process to sample an instrument. As more common and simpler instruments have synthesizers that mimic the real instrument quite well, less known and more complex instruments are waiting in line for a synthesizer that can replace the sample-based instrument reproduction. This work focuses on modelling Shamisen using Finite Difference Schemes. The performance of the model is evaluated in perceptual quality and verity tests. While the model does not synthesize a realistic Shamisen sound, the results of the work show the real potential of digitizing the Shamisen using Finite Difference Time Domain methods.

Contents

Chapter 1

Introduction

1.1 The Sound of the Shamisen

Shamisen is a Japanese three-stringed lute, which has originated in China. Shamisen is a chordophone that has a sound-box covered with a membrane made from an animal skin which lengthens the produced sound. The instrument is played using a large plectrum called bachi while using the fingers of the other hand to fret the stings producing different pitches[1]. The timbre of the Shamisen has a very distinct buzzing which is associated with a low nut which lets vibrating string come in the contact with the neck [1] and the specific playing technique that increases the percussiveness of the instrument by hitting the body with bachi during the plucking of the strings [2].

Figure 1.1: An image of a Shamisen manufactured by Komatsuya, designed to appeal to more people and boost the popularity of the instrument [3].

[1] Live Shamisen performance: https://youtu.be/nerhpPdKImA

1.2 Modeling Shamisen

Physical modeling will be used to synthesise the sound of the Shamisen as it provides the flexibility of adjusting different parameters as opposed to using samples when recreating shamisen sound digitally. Physical modeling synthesis refers to a synthesis technique where the sound produced is computed using mathematical equations which model the behavior of the sound source. The ultimate aim of the Shamisen model is to synthesize sound that is indistinguishable from a real instrument. Sound of which could be adjusted by the user with a logical connection to the real instrument. Thus adjustments have to reflect the available adjustments of a real Shamisen. Ideally, user could also adjust the parameters, which are not possible to adjust if they used a real instrument. The intended use of such model is a plugin in a Digital Audio Workstation (DAW) environment to make the Shamisen sound more available to the producers and musicians thus the model should work in real-time.
Creating such model would require to model a couple of strings, bridge, membrane and neck together while also having a bachi model for the excitation and have it running in real-time. Figure 1.2 shows the main parts of the Shamisen and a bachi. The requirements for this work are going to be reduced down to modeling and coupling the strings, the bridge and the membrane to study a simpler model with user adjustable parameters. When tuned correctly, such simpler model should yield similar sound to the real instrument, although the model lacks the buzzing and percussive elements of a real instrument.

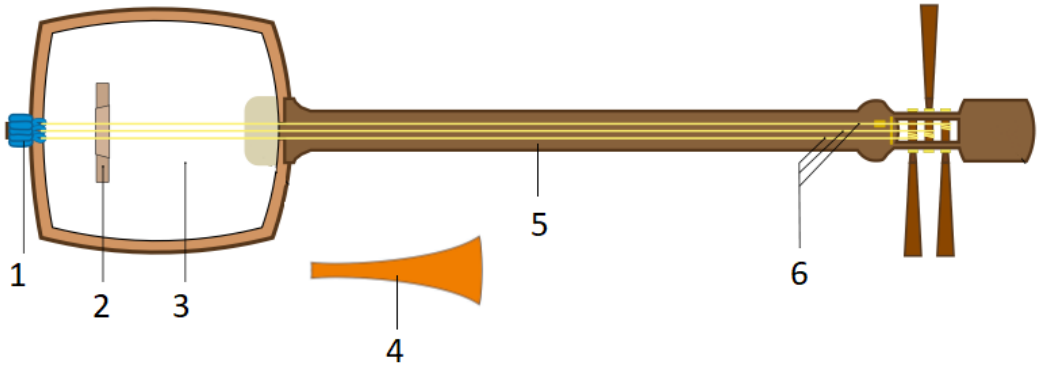


Figure 1.2: Diagram of the Shamisen [4][5].The parts marked with numbers are as follows: 1. neo - the silk knot for mounting the strings; 2. koma - the bridge; 3. kawa - the membrane; 4. bachi - the plectrum for playing Shamisen; 5. hatomune - the neck of the instrument; 6. ito - the strings.

1.3 Different Physical Modeling Techniques

As one of the goals of this work require the logical parameter adjustment, where user could intuitively adjust the sound by changing the parameters which they would change in the real instrument such as increasing the tension on the strings and changing the strings to a different gauge and/or material as well as selecting a different bridge which could differ in size and/or material, the physical modelling must be used to create such model.
One of the most efficient ways of modelling physical systems for sound production is using digital waveguides (DWG) [6]. DWG discretizes wave equations using bi-directional delay lines, commuting losses and phase inversions at a single point. Another physical modelling approach is to use Finite Difference Time Domain (FDTD) methods, which require developing a full mathematical description of the modelled system. Developing such description uses differential equations which are then discretized using finite difference methods, yielding Finite Difference Scheme (FDS). Considering these two approaches, the FDTD method provides better spatial accuracy when the model includes frequency-dependent damping and dispersion. Besides, the FDTD methods are more flexible as there are no assumptions being made about the linearity of the travelling wave solution [7]. These advantages come with a solution, which is more computationally expensive and prone to numerical dispersion and inaccuracies than DWG approach [8]. Despite that, real-time implementations of similar models have been achieved as it will be shown in the next chapter, thus FDTD approach has been chosen to model Shamisen.

Chapter 2

State of the Art

This section will describe the state of the art regarding digital synthesis of string instruments. First, some of the earlier attempts to digitize Shamisen for music production will be presented. After that the proposed physical models of similar instruments will be shown.

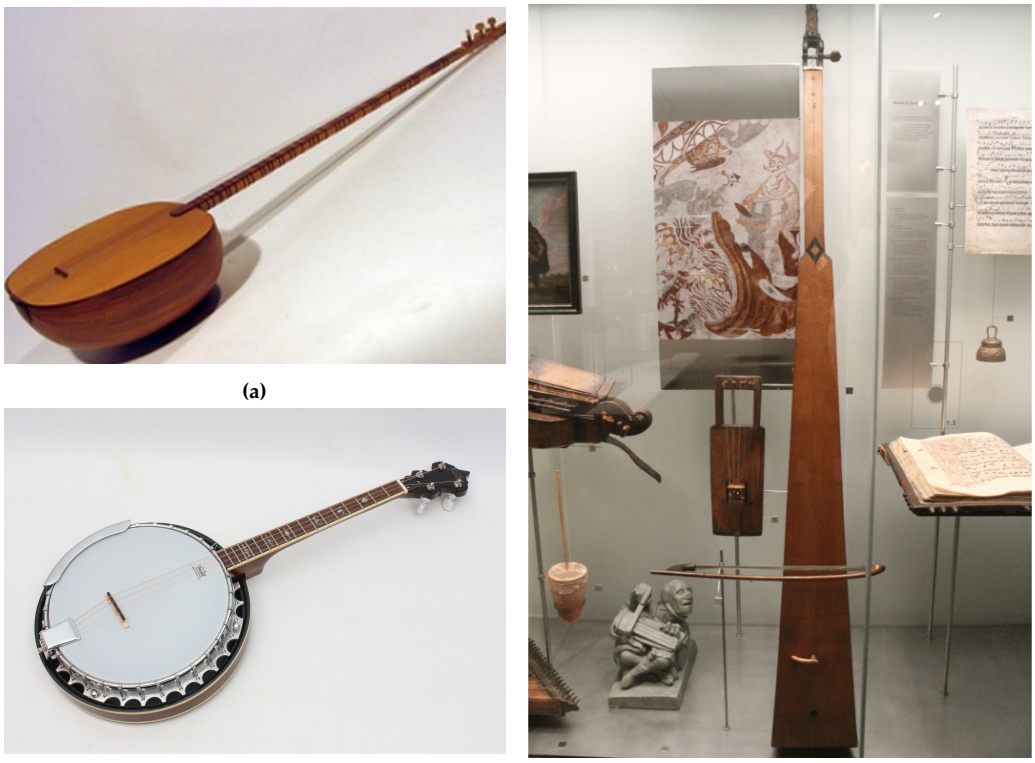
2.1 Shamisen plugins

The only plugin that focuses on Shamisen that could be found is called *ShamiKoto* [9] (see figure 2.1). The plugin authors do not specify if the plugin is sample based or if it is synthesizing the sound algorithmically. The plugin has ADSR envelope generator, pitch bend with direction controls, LFO for modulation with rate and depth controls, reverb and low-pass, high-pass and cutoff filters. The lack of any information regarding the technical aspects of the plugin limit the comparison to just mentioning the existence of the plugin.

Figure 2.1: ShamiKoto interface [9]. The text box in the center is a drop down menu with different presets.

2.2 Similar Instrument Models

A physical model of the Shamisen could not be found, thus a similar instrument models have to be looked at. This section will present and discuss models of the Banjo, the Tanbur and the Tromba Marina.



(a) (b) (c)

Figure 2.2: Images of the similar instruments. 2.2a is a picture of the Tanbur [10]; 2.2b is a picture of the Banjo [11]; 2.2c is a picture of the Tromba Marina [7]

2.2.1 Physical model of the Tanbur

Tanbur is a seven stringed lute made entirely out of wood. The body of the instrument resembles a shell with a thin wooden resonating plate on the face of the instrument (See figure 2.2a). The instrument is traditionally excited using tortoise shell that has been cut to an asymmetrical V shape [12]. The authors of [**tanbur**] developed two models for synthesising the Tanbur. One model was linear and the other one was non-linear. The linear model consists of a delay line, a fractional delay filter and a loop filter. The delay filter implements interpolation between samples and together with a delay line determine the fundamental frequency of the synthetic tone by adding to the loop delay. The loop filter is there to attenuate

the waveform of the tone from one frequency to the other smoothly and naturally. The non-linear model has a more complex structure, featuring two delay lines with fractional delay filter varying the delay. String terminations are modelled using reflection filters. As expected the linear model is fast and efficient, but it does not model the more interesting features such as varying fundamental frequency. The non-linear model is more computationally heavy, at the expense, modelling the desired nonlinear features.

2.2.2 Physical Model of the Banjo

Some might refer to a Shamisen as a "Japanese Banjo"[13] as it features a similar structural design of stringed instrument with a membrane stretched over the face of the instrument's body as it can be seen in the figure 2.2b. A real-time implementation of a physical model of the Banjo, proposed by Florian Pfeiffle and Rolf Bader uses FDTD methods when modeling the instrument [14]. The model couples a string, bridge, membrane and the air inside the body to create a full model of the Banjo. The authors state that the model is producing indistinguishable sound, compared to the sound from a real banjo, but comments are made that the model needs a tremendous amount of computational power and there has not been done any evaluation to confirm the statement about the sound reproduction. As the focus of that work shifts to FPGA technology that makes running such models in real-time possible the authors do not discuss the model of Banjo more in detail.

2.2.3 Physical model of the Tromba Marina

Tromba marina is a large bowed monochord instrument which dates back to the medieval ages the image of it can be seen in the figure 2.2c. The interesting feature of the instrument is that it has a rattling bridge which when a string is bowed produces a trumpet-like sound by vibrating its loose end against the wooden face of the instrument. In [7] a real-time model of a Tromba marina is modelled using FDTD techniques. The authors couple string, bridge, body and collision models to create the model of Tromba marina. Since the instrument has a single string, the bridge is modelled as a mass-spring-damper. The body is simplified to a 2D stiff plate. Similar model parts make this work very relevant when creating a Shamisen model based on Finite Difference Schemes.

Chapter 3

Development

In this chapter, related theory of Finite Difference Schemes will be presented as well as a test case that shows all the steps needed to obtain a working model, followed by PDE models of separate parts of the Shamisen that have the most influence to sound, along with the discretization, that has been performed and the coupling of said models into one complete system. Subsequently, the techniques for creating a model of the Shamisen will be presented. Finally, the more practical implementation choices that were made will be illustrated as well as parameter calculation and acquisition will be shown. Most of the theory and the equations is based on the "Numerical Sound Synthesis", written by Stefan Bilbao[15] unless specified otherwise.

3.1 Finite Difference Operators

To implement an FDS the first step is to define a sampling interval for the continuous variables of the system. It is equitable to sample time $t = nk$ and space $x = lh$ in discrete series, where $n = 0, 1, 2, ...$ and $l \in \mathbb{Z}$. k is the time step and h is the grid spacing. In the signal processing field it is common to define the sampling frequency as $f_s = 1/k$. These values are not independent, meaning, the choice of one value will influence the other through a stability condition, which will be described more in detail bellow. Once the grid has been defined, a function $u(x,t)$ can be approximated as a grid function u_l^n, where the l is the spatial term and n is the temporal term, which both correspond to the values presented earlier. It is important to mention that the solution of an PDE, approximated with finite differences, is not a sampled version of the continuous function $u(x,t)$, but rather an approximation to it.
Finite Difference operators which can be applied to a series are shifts, thus it is possible to define the *shift operators* as:

$$e_{t+}u_l^n = u_l^{n+1} \quad e_{t-}u_l^n = u_l^{n-1} \quad e_{x+}u_l^n = u_{l+1}^n \quad e_{x-}u_l^n = u_{l-1}^n \tag{3.1}$$

Using these shift operators, more complex operators can be derived. For instance approximations of the first-order derivative operator. In the time domain, approximation of the first-order derivative would look like this:

$$\delta_{t+} := \frac{1}{k}(e_{t+} - 1) \approx \frac{\partial}{\partial t} \quad \delta_{t-} := \frac{1}{k}(1 - e_{t-}) \approx \frac{\partial}{\partial t} \quad \delta_{t\cdot} := \frac{1}{2k}(e_{t+} - e_{t-}) \approx \frac{\partial}{\partial t} \tag{3.2}$$

These are the *forward*, *backward* and *center* difference approximations respectively, which, when applied to the series u_l^n would yield:

$$\delta_{t+} u_l^n = \frac{u_l^{n+1} - u_l^n}{k} \tag{3.3}$$

$$\delta_{t-} u_l^n = \frac{u_l^n - u_l^{n-1}}{k} \tag{3.4}$$

$$\delta_{t\cdot} u_l^n = \frac{u_l^{n+1} - u_l^{n-1}}{2k} \tag{3.5}$$

These are three different approximations for the first time derivative. Forward and backward difference approximations are first order accurate, while center difference is second order accurate as proven by expanding these operators in Taylor series about t. Combining these we can attain second-order time difference operator:

$$\delta_{tt} u_l^n := \delta_{t+}\delta_{t-} u_l^n = \frac{u_l^{n+1} - 2u_l^n + u_l^{n-1}}{k^2} \tag{3.6}$$

Obtaining spacial derivatives follows the same rules, yielding:

$$\delta_{x+} u_l^n = \frac{u_{l+1}^n - u_l^n}{h} \approx \frac{\partial}{\partial x} u(x,t) \tag{3.7}$$

$$\delta_{x-} u_l^n = \frac{u_l^n - u_{l-1}^n}{h} \approx \frac{\partial}{\partial x} u(x,t) \tag{3.8}$$

$$\delta_{x\cdot} u_l^n = \frac{u_{l+1}^n - u_{l-1}^n}{2h} \approx \frac{\partial}{\partial x} u(x,t) \tag{3.9}$$

$$\delta_{xx} u_l^n := \delta_{x+}\delta_{x-} u_l^n = \frac{u_{l+1}^n - 2u_l^n + u_{l-1}^n}{h^2} \tag{3.10}$$

See the figure 3.1 for visualisation of the stencils used for the above mentioned operators.

3.2 1-D Wave Equation

The 1-D Wave Equation will be used as a test case for showing the necessary steps for creating a physical model using the Finite Difference Schemes, mainly because it is probably one of the most important, if not the most important PDE not only

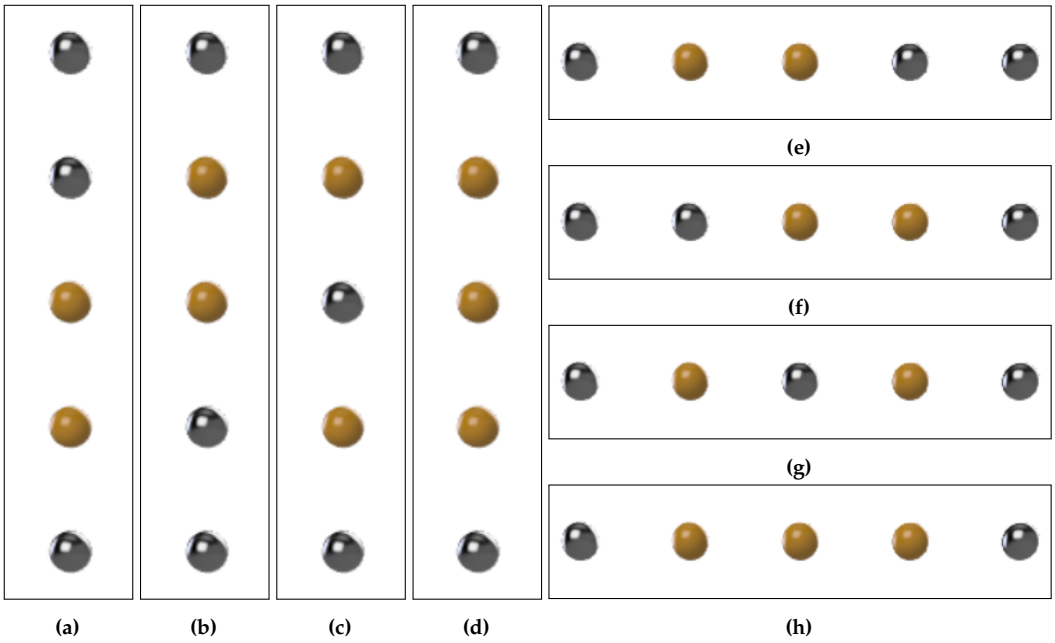


Figure 3.1: Simple temporal or spatial difference operator stencils. All the points represent samples in the grid. Points in space are mapped to the horizontal axis and points in time are mapped to the vertical axis. The yellow points in the figures represent the samples which are used to calculate the respective approximations: figures 3.1a, 3.1b, 3.1c and 3.1d for equations 3.4, 3.3, 3.5 and 3.6 respectively; while figures 3.1e, 3.1f, 3.1g and 3.1h for equations 3.8, 3.7, 3.9 and 3.10 respectively.

in musical acoustics but in physics as well. Even though, is mostly used as a test case, variants of it act as building blocks that describe the approximated behaviour such as strings, air movement in a closed acoustic tube and longitudinal motion of a uniform bar. The 1-D wave equation is defined as:

$$u_{tt} = c^2 u_{xx} \tag{3.11}$$

Dependent variable u in a 1-D wave PDE model (eq. 3.11) is equal to $u(x,t)$ where x represents distance and t represents time. Subscripts t and x here refer to a derivative in time or in space respectively. This equation is a second-order PDE, where x is the distance variable and t is the variable for time. This equation is defined over $x \in \mathcal{D}$, where $\mathcal{D} = [0, L]$ and L is the length of our system. is a subset of $\mathbb{R}$ and $t \geq 0$. c in this equation is a constant, generally it is called *wave speed* and its definition depends on what is being modelled. $c = \sqrt{T_0/\rho A}$ for modeling string, T_0 is tension, ρ and A is material density and cross-sectional area respectively.
Discretizing 1-D wave PDE (eq. 3.11) gives an equation 3.12

$$\delta_{tt} u = c^2 \delta_{xx} u \tag{3.12}$$

Now, using the rules, presented in section 3.1 we can create an expanded update equation (eq. 3.13) using finite difference operators.

$$u_l^{n+1} = \frac{c^2k^2}{h^2}(u_{l+1}^n - 2u_l^n + u_{l-1}^n) + 2u_l^n - u_l^{n-1} \tag{3.13}$$

l here is the spatial sample and n is the time sample. To make sure, that the update equation does not go unstable, Courant-Friedrichs-Lewy (CFL) condition has to satisfied. CFL condition for this simple 1-D wave equation:

$$\lambda \leq 1 \tag{3.14}$$

where

$$\lambda = ck/h \tag{3.15}$$

c is selected for the desired frequency, k is calculated from the sampling frequency $f_s = 44100Hz$: $k = 1/f_s$. The variable which is free for the manipulation is the grid spacing h (equation 3.16).

$$h \geq h_{min} = ck \tag{3.16}$$

Minimising the grid spacing gives the most accurate solution while still keeping the equation stable[1]. Having solved for the grid spacing the total number of intervals N can be calculated using:

$$N = \text{floor}(L/h) \tag{3.17}$$

Since the 1-D wave equation has second-order differentiation in space it requires a single boundary condition at the endpoints of the spatial domain. Typically boundary is set to 0 at the endpoints of the domain.

$$u(0,t) = u(L,t) = 0 \qquad \text{,Dirichlet boundary condition} \tag{3.18}$$

This type of condition is called *Dirichlet* type. In a case for string, this condition represents a stiff termination. Another widely used boundary condition is called *Neumann* condition:

$$u(0,t)_x = u(L,t)_x = 0 \qquad \text{,Neumann boundary condition} \tag{3.19}$$

Such boundary condition can be interpreted as stiff in longitudinal direction, but free to move in transverse direction. Both of these conditions are lossless, lossy boundary conditions are out of the scope for this project.

[1]The stability conditions shown in this thesis have been arrived at using von Neumann stability analysis. It was out of the scope of this project to perform the steps myself.

3.3 Strings

As in the case of the 1D wave equation, the dependent variable u in a damped stiff string PDE model (eq. 3.20) is equal to $u(x,t)$ where x represents distance and t represents time. $x \in \mathcal{D}_S$ where $\mathcal{D}_S = [0, L_S]$ with L_S being the string length,

$$\rho_S A_S u_{tt} = T_S u_{xx} - E_S I_S u_{xxxx} - 2\rho_S A_S \sigma_{0,S} u_t + 2\rho_S A_S \sigma_{1,S} u_{txx} \tag{3.20}$$

In equation 3.20 ρ_S is the density and A_S is the cross sectional area of the string, T_S is the tension applied to the string, E_S and I_S being the Young's modulus and Inertia (described by eq. 3.21 [7]) of the string respectively.

$$I_S = \frac{\pi {r_S}^4}{4} \tag{3.21}$$

$\sigma_{0,S}$ and $\sigma_{1,S}$ are the damping coefficients for the string. Frequency-dependent damping models more realistic strings, where higher frequencies will decay faster than lower ones, creating a synthesized sound which has a wide range of frequencies at the attack and in the end of the a decay that sound will only have a few harmonics audible. σ_0 gives a rise to the bulk frequency-independent loss while σ_1 is the means of modeling frequency-dependent loss.
Fourth-order differentiation in space introduces dispersive behaviour, which means, that the components that have short wavelength travel faster than long wavelength components. c_S is the wave speed that depends on the tension of the String and κ_S is the stiffness parameter. c_S and κ_S are described in equations 3.23 and 3.24 respectively.

$$u_{tt} = {c_S}^2 u_{xx} - {\kappa_S}^2 u_{xxxx} - 2\sigma_{0,S} u_t + 2\sigma_{1,S} u_{txx} \tag{3.22}$$

$$c_S = \sqrt{\frac{T_S}{\rho_S A_S}} \tag{3.23}$$

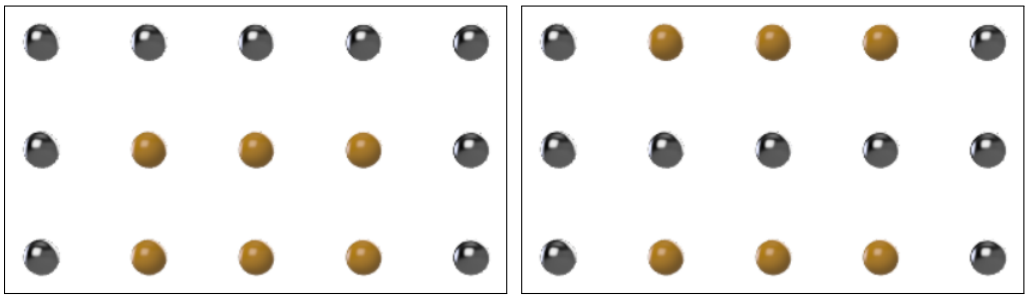
$$\kappa_S = \sqrt{\frac{E_S I_S}{\rho_S A_S}} \tag{3.24}$$

The continuous damped stiff string PDE (eq. 3.22) in discrete time and space is represented by the equation 3.25.

$$\delta_{tt} u = {c_S}^2 \delta_{xx} u - {\kappa_S}^2 \delta_{xxxx} u - 2\sigma_{0,S} \delta_t u + 2\sigma_{1,S} \delta_{txx} u \tag{3.25}$$

For the $\sigma_{0,S}$ term, center difference operator is chosen due to it being second-order accurate as mentioned in section 3.1, where as the forward and backward difference operators are first-order accurate. For the mixed spatial-temporal difference operator, backward time difference operator is chosen since it keeps the scheme explicit as it does not include future neighbouring samples in its solution. Same

is true with the discretization decisions for the Bridge and the Membrane. See the figure 3.2 for the visualisation of mixed difference operator stencils.



(a) (b)

Figure 3.2: Mixed spatial-temporal difference operator stencils. All the points represent samples in the grid. Points in space are mapped to the horizontal axis and points in time are mapped to the vertical axis. The yellow points in the figures represent the samples which are used to calculate the respective approximations: Figure 3.2a is the stencil for the mixed backward time difference operator and figure 3.2b is the stencil for mixed center time difference operator.

The string grid spacing h_S is calculated for the reasons given in the end of the section 3.2 although the CFL condition for the stiff string is different from the one presented in the aforementioned section:

$$\lambda_S^2 + 4\mu_S^2 + \frac{4\sigma_{1,S}k}{h_S^2} \leq 1 \tag{3.26}$$

where $\mu_S = \kappa_S k/h_S^2$ and $\lambda_S = c_S k/h_S$. Counting in the damping terms, the grid spacing equation 3.27 for the damped stiff string is equal to:

$$h_S \geq h_{S,\min} = \sqrt{\frac{{c_S}^2k^2 + 4\sigma_{1,S}k + \sqrt{({c_S}^2k^2 + 4\sigma_{1,S}k)^2 + 16{\kappa_S}^2k^2}}{2}} \tag{3.27}$$

Having solved for the grid spacing, the total number of intervals N_S can be calculated by:

$$N_S = \text{floor}(L_S/h_S) \tag{3.28}$$

3.4 Bridge

The PDE for modelling the bridge is modelled as a linear damped bar, which has the same PDE as the damped stiff string (eq. 3.20), but without the tension term. Dependent variable v in a damped bar PDE model (eq. 3.29) is equal to $v(x,t)$ where x represents distance and t represents time. $x \in \mathcal{D}_B$ where $\mathcal{D}_B = [0, L_B]$ with L_B being the bridge length.

$$\rho_B A_B v_{tt} = -E_B I_B v_{xxxx} - 2\rho_B A_B \sigma_{0,B} v_t + 2\rho_B A_B \sigma_{1,B} v_{txx} \tag{3.29}$$

In equation 3.29 just like in string equations in section 3.3, ρ_B is the density and A_B is the cross-sectional area of the linear bridge, E_B and I_B are Young's modulus and inertia of the bridge respectively. Cross-sectional area and inertia are described by equations 3.30 and 3.31 where H_B is the thickness and b is the width of a bridge. $\sigma_{0,B}$ and $\sigma_{1,B}$ are the damping coefficients for the bridge.

$$A_B = bH_B \tag{3.30}$$

$$I_B = \frac{1}{12} bH_B{}^3 \tag{3.31}$$

κ_B is the stiffness parameter of a bridge.

$$v_{tt} = -\kappa_B{}^2 v_{xxxx} - 2\sigma_{0,B} v_t + 2\sigma_{1,B} v_{txx} \tag{3.32}$$

$$\kappa_B = \sqrt{\frac{E_B I_B}{\rho_B A_B}} \tag{3.33}$$

Using definitions of A_B and I_B we can simplify the definition of the stiffness parameter κ_B (eq. 3.33) from the one we used for the string in section 3.3 to equation 3.34. Where knowing the cross-sectional area and the width of a bar is not necessary to compute stiffness of the bar.

$$\kappa_B = \sqrt{\frac{E_B H_B{}^2}{12\rho_B}} \tag{3.34}$$

The continuous bridge PDE (eq. 3.32) in discrete time and space is represented by the equation 3.35.

$$\delta_{tt} v = -\kappa_B \delta_{xxxx} v - 2\sigma_{0,B} \delta_{t\cdot} v + 2\sigma_{1,B} \delta_{t-} \delta_{xx} v \tag{3.35}$$

The grid spacing h_B for the bridge model, is calculated similarly to the h_S that was used in the damped stiff string, excluding all the *wave speed* terms, which are omitted:

$$h_B \geq h_{B,\min} = \sqrt{\frac{4\sigma_{1,B} k + \sqrt{4\sigma_{1,B} k^2 + 16\kappa_B{}^2 k^2}}{2}} \tag{3.36}$$

The grid spacing h_B, determines the total number of intervals N_B:

$$N_B = \text{floor}(L_B / h_B) \tag{3.37}$$

3.5 Membrane

Dependent variable w in a damped stiff membrane PDE model (eq. 3.38) is equal to $w(x, y, t)$ where (x, y) represents distance on a 2D surface and t represents time. $(x, y) \in \mathcal{D}_M$ where, $\mathcal{D}_M = [0, L_x] \times [0, L_y]$. Arriving at the damped stiff membrane

PDE is a matter of combining a tension term from the 2D membrane model with a stiffness term from the Kirchhoff thin plate model and adding the damping.

$$\rho_M H_M w_{tt} = T_M \Delta w - D_M \Delta\Delta w - 2\rho_M H_M \sigma_{0,S} w_t + 2\rho_M H_M \sigma_{1,S} \Delta w_t \tag{3.38}$$

In equation 3.38 ρ_M is the density and H_M is the thickness of the membrane, T_M is the tension applied to the membrane, D_M is the membrane flexural rigidity. D_M term is the taken from the Kirchhoff thin plate and acts as a stiffness parameter, which depends on Young's modulus E_M, thickness of the membrane and the Poisson's ratio ν (eq. 3.39).

$$D_M = \frac{E_M {H_M}^3}{12(1-\nu^2)} \tag{3.39}$$

$\sigma_{0,M}$ and $\sigma_{1,M}$ are the damping coefficients for the membrane in equation 3.38 [16]. c_M is the wave speed, which depends on the tension parameter and κ_M is the stiffness parameter of a stiff membrane. c_M and κ_M are described in equations 3.41 and 3.42 respectively.

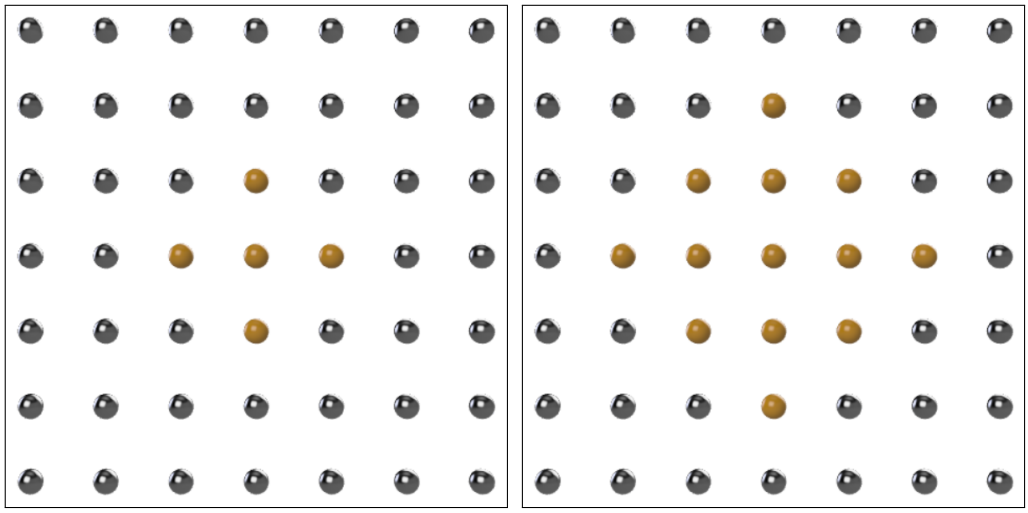
$$w_{tt} = {c_M}^2 \Delta w - {\kappa_M}^2 \Delta\Delta w - 2\sigma_{0,M} w_t + 2\sigma_{1,M} \Delta w_t \tag{3.40}$$

$$c_M = \sqrt{\frac{T_M}{\rho_M H_M}} \tag{3.41}$$

$$\kappa_M = \sqrt{\frac{D_M}{\rho_M H_M}} \tag{3.42}$$

The continuous damped stiff membrane PDE (eq. 3.40) in discrete time and space is represented by the equation 3.43. See the figure 3.3 for some of the stencils used.

$$\delta_{tt} w = c_M \delta_\Delta w - \kappa_M \delta_\Delta \delta_\Delta w - 2\sigma_{0,S} \delta_{t\cdot} w + 2\sigma_{1,S} \delta_{t-} \delta_\Delta w \tag{3.43}$$



(a) (b)

Figure 3.3: 2-D spatial difference operator stencils for the current time index. All the points represent samples in the grid. Points in x-axis are mapped to the horizontal axis and points in y-axis are mapped to the vertical axis. The yellow points in the figures represent the samples which are used to calculate the respective approximations: Figure 3.3a is the stencil for $\delta_\Delta w$ and figure 3.3b is the stencil for $\delta_\Delta \delta_\Delta w$.

The grid spacing h_M is equal in both directions thus using von Neumann stability analysis it is determined by the equation 3.44.

$$h_M \geq h_{M,\min} = \sqrt{{c_M}^2 k^2 + 4\sigma_{1,M} k + \sqrt{({c_M}^2 k^2 + 4\sigma_{1,M} k)^2 + 16{\kappa_M}^2 k^2}} \tag{3.44}$$

With the grid spacing h_M, the total number of intervals N_{Mx} and N_{My} can be calculated by:

$$\begin{aligned} N_{Mx} &= \text{floor}(L_{Mx}/h_M) \\ N_{My} &= \text{floor}(L_{My}/h_M) \end{aligned} \tag{3.45}$$

3.6 Coupling

The entire system consists of 3 strings, a bridge and a membrane working together. For coupling all the parts together, rigid connections are assumed at the localizer points:

Localizer points

Strings	$x_{c,S1}$	$x_{c,S2}$	$x_{c,S3}$
Bridge	$x_{c,B1}$	$x_{c,B2}$	$x_{c,B3}$
	$x_{c,BL}$	$x_{c,BR}$	
Membrane	$x_{c,ML}$, $y_{c,ML}$	$x_{c,MR}$, $y_{c,MR}$	

Table 3.1: All the connection points in this system. Subscripts S1, S2 and S3 indicate that the connection points are on different strings, B1, B2 and B3 are the corresponding connection points on the bridge for these strings. BL and BR subscripts indicate connection points on the left side and the right side of the bridge, in turn, ML and MR subscripts are the same points on the membrane.

Spreading operator $J(x_c)$ in the equations above is defined by:

$$J(x_c) = \begin{cases} \frac{1}{h^d}, & l = l_c = \text{round}(x_c/h) \\ 0, & \text{otherwise} \end{cases} \tag{3.46}$$

Where d is a number of the dimensions on which x_c is defined over a domain $\mathcal{D}$. In this case, $d = 1$ for the strings and the bridge and $d = 2$ for the membrane.
From a physical perspective, strings will exert some force F_t on the bridge to the locations where strings touch the Top of the bridge, which will exert force F_b to the membrane at the mounting points at the Bottom of the bridge. Adding or subtracting respective forces multiplied by a spreading operator $J(x_c)$, which spreads force at the location x_c to or from models (equations 3.25, 3.35, 3.43) will give us starting equations for calculating connection forces:

$$\delta_{tt}u_i = {c_{Si}}^2\delta_{xx}u_i - {\kappa_{Si}}^2\delta_{xxxx}u_i - 2\sigma_{0,Si}\delta_{t\cdot}u_i + 2\sigma_{1,Si}\delta_{t-}\delta_{xx}u_i - J(x_{c,Si})\frac{F_{ti}}{\rho_S i A_{Si}} \tag{3.47}$$

i in the subscript denotes which string the equation 3.47 is describing. Since shamisen has 3 strings, $i = [1, 2, 3]$.

$$\begin{aligned} \delta_{tt}v = {\kappa_B}^2\delta_{xxxx}v - 2\sigma_{0,B}\delta_{t\cdot}v + 2\sigma_{1,B}\delta_{t-}\delta_{xx}v \\ + J(x_{c,B1})\frac{F_{t1}}{\rho_B A_B} + J(x_{c,B2})\frac{F_{t2}}{\rho_B A_B} + J(x_{c,B3})\frac{F_{t3}}{\rho_B A_B} \\ - J(x_{c,BL})\frac{F_{bL}}{\rho_B A_B} - J(x_{c,BR})\frac{F_{bR}}{\rho_B A_B} \end{aligned} \tag{3.48}$$

F_t with a number 1, 2 or 3 denotes, which string is exerting the force on the bridge, while F_b with letters L or R denote force exerted from either left or right side of the bridge to the membrane.

$$\begin{aligned} \delta_{tt}w = {c_M}^2\delta_{\Delta}w - {\kappa_M}^2\delta_{\Delta}\delta_{\Delta}w - 2\sigma_{0,M}\delta_{t\cdot}w + 2\sigma_{1,M}\delta_{t-}\delta_{\Delta}w \\ + J(x_{c,ML}, y_{c,ML})\frac{F_{bL}}{\rho_M H_M} + J(x_{c,MR}, y_{c,MR})\frac{F_{bR}}{\rho_M H_M} \end{aligned} \tag{3.49}$$

3.7 Implementation

After obtaining discrete equations for strings, bridge, membrane and the coupling forces, update equations can be described. To reach final expanded update equations, operators presented in section 3.1 will be used alongside with operators, that will be derived here, using the rules that were presented in the aforementioned section.
In the equations bellow, u is used on the left side of the equality, in all these cases, $u = u_l^n$. In other cases just the superscript could be missing, there the n is omitted for brevity and the current time should be assumed.
Here, u is chosen as a general notation meaning that these rules also apply to other models and should not be confused with *dependent variable u* that is used in String model.
The fourth-order spatial derivative operator is obtained by applying the second-order derivative operator twice:

$$\delta_{xxxx}u = \delta_{xx}\delta_{xx}u_l^n = \frac{1}{h^4}(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n) \tag{3.50}$$

Mixed temporal-spatial derivative operator:

$$\delta_{t-}\delta_{xx}u = \frac{u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}}{kh^2} \tag{3.51}$$

Finite difference operators in 2D are defined as:

$$\delta_{\Delta\boxplus}u = \delta_{xx}u + \delta_{yy}u = \frac{u_{l-1,m}^n + u_{l+1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n - 4u_{l,m}^n}{h^2} \tag{3.52}$$

Notice the subscript $\boxplus$ in equation 3.52, this indicates the shape of the stencil - the spatial samples are taken 1 position from current strictly horizontally and vertically (both sides).
$\delta_{\Delta\boxplus}\delta_{\Delta\boxplus}$ operator, using previously mentioned stencil is equal to:

$$\delta_{\Delta\boxplus}\delta_{\Delta\boxplus}u = \delta_{xxxx}u + 2\delta_{xxyy}u + \delta_{yyyy}u \tag{3.53}$$

where

$$\delta_{xxyy}u = \frac{1}{h^4}(u_{l\pm1\&m\pm1}^n - 2u_{l\pm1\oplus m\pm1}^n + 4u_{l,m}^n) \tag{3.54}$$

and

$$\begin{aligned} u_{l\pm1\otimes m\pm1}^n &= u_{l+1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m+1}^n + u_{l-1,m-1}^n \\ u_{l\pm1\oplus m\pm1}^n &= u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n \end{aligned} \tag{3.55}$$

The mixed temporal-spatial derivative operator in 2D is determined by the same logic as in 1D mixed derivative operator (eq. 3.51):

$$\delta_{t-}\delta_{\Delta}u = \frac{u_{l\pm1\oplus m\pm1}^n - 4u_{l,m}^n - u_{l\pm1\oplus m\pm1}^{n-1} + 4u_{l,m}^{n-1}}{kh^2} \tag{3.56}$$

Lowering the computation time for the update equation, each *dependent variable* has been assigned to a single constant multiplier, which is calculated before the iterations start and *dependent variable* terms are calculated only once per iteration.

3.7.1 Strings

The *dependent variable* of the string is u, thus the next time step term u^{n+1} has a multiplier a, current time step terms u^n have multiplier b and the previous time step u^{n-1} terms have multiplier c. Expanding equation 3.25 yields the intermediate update equation for strings 3.63 with:

$$a_u = k\sigma_{0,S} + 1 \tag{3.57}$$

$$b_u = \left(\frac{(-2{c_S}^2)}{{h_S}^2} - \frac{6{\kappa_S}^2}{{h_S}^4} - \frac{4\sigma_{1,S}}{k{h_S}^2}\right)k^2 + 2 \tag{3.58}$$

$$b_{u\pm1} = \left(\frac{{c_S}^2}{{h_S}^2} + \frac{4{\kappa_S}^2}{{h_S}^4} + \frac{2\sigma_{1,S}}{k{h_S}^2}\right)k^2 \tag{3.59}$$

$$b_{u\pm2} = \frac{-{\kappa_S}^2}{{h_S}^4}k^2 \tag{3.60}$$

$$c_u = \frac{4\sigma_{1,S}k^2}{k{h_S}^2} + \sigma_{0,S}k - 1 \tag{3.61}$$

$$c_{u\pm1} = \frac{-2\sigma_{1,S}k^2}{k{h_S}^2} \tag{3.62}$$

$b_{u\pm1}$, $b_{u\pm2}$ and $c_{u\pm1}$ has a $\pm$ sign which indicates that the multiplier is the same, whether it is applied to the left spatial sample or the right. u in the subscript of each multiplier assigns the multiplier to the string. Yielding the intermediate update equation for strings:

$$\begin{aligned} a_u u_l^I = {} & b_u u_l^n + b_{u\pm1}(u_{l-1}^n + u_{l+1}^n) + b_{u\pm2}(u_{l-2}^n + u_{l+2}^n) \\ & + c_u u_l^{n-1} + c_{u\pm1}(u_{l-1}^{n-1} + u_{l+1}^{n-1}) \end{aligned} \tag{3.63}$$

Note that the u on the left side of the equation has a superscript I (Intermediate), this denotes u_l^{n+1} without the coupling force.

3.7.2 Bridge

The *dependent variable* of the bridge is v, using the the same naming logic that was used for the stings the multipliers for bridge update equation are defined. Expanding equation 3.35 yields the intermediate update equation for the bridge 3.70 with:

$$a_v = {h_B}^4(k\sigma_{1,B} + 1) \tag{3.64}$$

$$b_v = 2{h_B}^4 - 4{h_B}^2 k\sigma_{1,B} - 6k^2{\kappa_B}^2 \tag{3.65}$$

$$b_{v\pm1} = 2{h_B}^2 k\sigma_{1,B} + 4k^2{\kappa_B}^2 \tag{3.66}$$

$$b_{v\pm2} = -k^2{\kappa_B}^2 \tag{3.67}$$

$$c_v = -{h_B}^4 + 4{h_B}^2 k\sigma_{1,B} + {h_B}^4 k\sigma_{0,B} \tag{3.68}$$

$$c_{v\pm1} = -2k\sigma_{1,B} \tag{3.69}$$

These help to define the intermediate update equation for the bridge:

$$\begin{aligned} a_v v_l^I = {} & b_v v_l^n + b_{v\pm1}(v_{l-1}^n + v_{l+1}^n) + b_{v\pm2}(v_{l-2}^n + v_{l+2}^n) \\ & + c_v v_l^{n-1} + c_{v\pm1}(v_{l-1}^{n-1} + v_{l+1}^{n-1}) \end{aligned} \tag{3.70}$$

Note that the *dependent variable* v on the left side of the equation has a superscript I, this denotes v_l^{n+1} without the coupling force.

3.7.3 Membrane

The *dependent variable* of the membrane is w, using the the same naming logic that was used for the stings and the bridge, the multipliers for membrane update equation are defined, although in this case we have more w terms, thus, more multipliers for which naming scheme will slightly differ. Expanding equation 3.43 yields the intermediate update equation for the membrane 3.78 with:

$$a_w = k\sigma_{0,M} + 1 \tag{3.71}$$

$$b_w = \left(\frac{(-2{c_M}^2)}{{h_M}^2} - \frac{20{\kappa_M}^2}{{h_M}^4} - \frac{8\sigma_{1,M}}{k{h_M}^2}\right)k^2 + 2 \tag{3.72}$$

$$b_{w\oplus\pm1} = \left(\frac{{c_M}^2}{{h_M}^2} + \frac{8{\kappa_M}^2}{{h_M}^4} + \frac{2\sigma_{1,M}}{k{h_M}^2}\right)k^2 \tag{3.73}$$

$$b_{w\otimes\pm1} = \frac{-2{\kappa_M}^2}{{h_M}^4}k^2 \tag{3.74}$$

$$b_{w\oplus\pm2} = \frac{-{\kappa_M}^2}{{h_M}^4}k^2 \tag{3.75}$$

$$c_w = \frac{8\sigma_{1,M}k^2}{k{h_M}^2} + \sigma_{0,M}k - 1 \tag{3.76}$$

$$c_{w\oplus\pm1} = \frac{-2\sigma_{1,M}k^2}{k{h_M}^2} \tag{3.77}$$

Multipliers with a subscript $\oplus$ are multiplied with either one of the dimension not at the current spatial sample, while $\otimes$ indicates that sample at both dimensions is not at the current spatial sample. Otherwise the naming scheme is the same as it was used in the string and the bridge update equations.

$$\begin{aligned} a_w w^l_{l,m} = {} & b_w w^n_{l,m} + b_{w\oplus\pm1} w^n_{l\pm1\oplus m\pm1} + b_{w\otimes\pm1} w^n_{l\pm1\otimes m\pm1} \\ & + b_{w\oplus\pm2} w^n_{l\pm2\oplus m\pm2} + c_w w^{n-1}_{l,m} + c_{w\oplus\pm1} w^{n-1}_{l\pm1\oplus m\pm1} \end{aligned} \tag{3.78}$$

Note that the w on the left side of the equation has a superscript I, this denotes $w^{n+1}_{l,m}$ without the coupling forces. The notation of the $w^n_{l\pm1\oplus m\pm1}$, $w^n_{l\pm1\otimes m\pm1}$, $w^n_{l\pm2\oplus m\pm2}$ and $w^{n-1}_{l\pm1\oplus m\pm1}$ was adapted from the equations 3.55 for compactness of the update equation.

3.7.4 Boundary Conditions

Recalling equations 3.28 and 3.45 for the number of intervals N_S and N_{Mx}, N_{My}, string and membrane have their boundaries set as clamped:

$$\begin{aligned} u_0 = 0 \qquad & \text{and} \qquad u_{N_S} = 0 \\ \delta_{x+}u_0 = 0 \qquad & \text{and} \qquad \delta_{x-}u_{N_S} = 0 \end{aligned} \tag{3.79}$$

$$\begin{aligned} w_{0,m} = 0; \quad w_{l,0} = 0; \qquad & w_{N_{Mx},m} = 0; \quad w_{l,N_{My}} = 0 \\ \delta_{x+}w_{0,m} = 0; \quad \delta_{y+}w_{l,0} = 0; \quad & \delta_{x-}w_{N_{Mx},m} = 0; \quad \delta_{y-}w_{l,N_{My}} = 0 \end{aligned} \tag{3.80}$$

The boundary conditions are met by setting first and last two samples from the boundary to zero and when updating equations, these points are left at zero.
The bridge has a free boundary condition:

$$\begin{aligned} \delta_{xx}v_0 = 0 \qquad & \text{and} \qquad \delta_{xx}\delta_{x\cdot}v_0 = 0 \\ \delta_{xx}v_{N_B} = 0 \qquad & \text{and} \qquad \delta_{xx}\delta_{x\cdot}v_{N_B} = 0 \end{aligned} \tag{3.81}$$

While the first two spatial sample points at the boundaries are set to zero for the string and the membrane models, having a free boundary in the bridge, these points must be calculated separately:

$$\begin{aligned} a_v v_0^{n+1} = {} & b_v v_0^n + b_{v\pm1}(v_{-1}^n + v_1^n) + b_{v\pm2}(v_{-2}^n + v_2^n) \\ & + c_v v_0^{n-1} + c_{v\pm1}(v_{-1}^{n-1} + v_1^{n-1}) \end{aligned} \tag{3.82}$$

$$\begin{aligned} a_v v_1^{n+1} = {} & b_v v_1^n + b_{v\pm1}(v_0^n + v_2^n) + b_{v\pm2}(v_{-1}^n + v_3^n) \\ & + c_v v_1^{n-1} + c_{v\pm1}(v_0^{n-1} + v_2^{n-1}) \end{aligned} \tag{3.83}$$

$$\begin{aligned} a_v v_{N_B-1}^{n+1} = {} & b_v v_{N_B-1}^n + b_{v\pm1}(v_{N_B-2}^n + v_{N_B}^n) + b_{v\pm2}(v_{N_B-3}^n + v_{N_B+1}^n) \\ & + c_v v_{N_B-1}^{n-1} + c_{v\pm1}(v_{N_B-2}^{n-1} + v_{N_B}^{n-1}) \end{aligned} \tag{3.84}$$

$$\begin{aligned} a_v v_{N_B}^{n+1} = {} & b_v v_{N_B}^n + b_{v\pm1}(v_{N_B-1}^n + v_{N_B+1}^n) + b_{v\pm2}(v_{N_B-2}^n + v_{N_B+2}^n) \\ & + c_v v_{N_B}^{n-1} + c_{v\pm1}(v_{N_B-1}^{n-1} + v_{N_B+1}^{n-1}) \end{aligned} \tag{3.85}$$

Mind the points outside the boundaries v_{-1}, v_{-2}, v_{N_B+1} and v_{N_B+2}, these sample points are called *virtual points* as they are only used to calculate equation at the boundaries and they do not belong in the model itself. These *virtual points* are calculated using the equations below, which are derived from the equations 3.81.

$$v_{-1}^n = 2v_0^n - v_1^n \tag{3.86}$$

$$v_{-2}^n = 2(v_{-1}^n - v_1^n) + v_2^n \tag{3.87}$$

$$v_{N_B+1}^n = 2v_{N_B}^n - v_{N_B-1}^n \tag{3.88}$$

$$v_{N_B+2}^n = 2(v_{N_B+1}^n - v_{N_B}^n) + v_{N_B-2}^n \tag{3.89}$$

$$v_{-1}^{n-1} = 2v_0^{n-1} - v_1^{n-1} \tag{3.90}$$

$$v_{N_B+1}^{n-1} = 2v_{N_B}^{n-1} - v_{N_B-1}^{n-1} \tag{3.91}$$

3.7.5 Coupling

The full system can be visualised by the figure 3.4. Figure shows all the coupling points used.

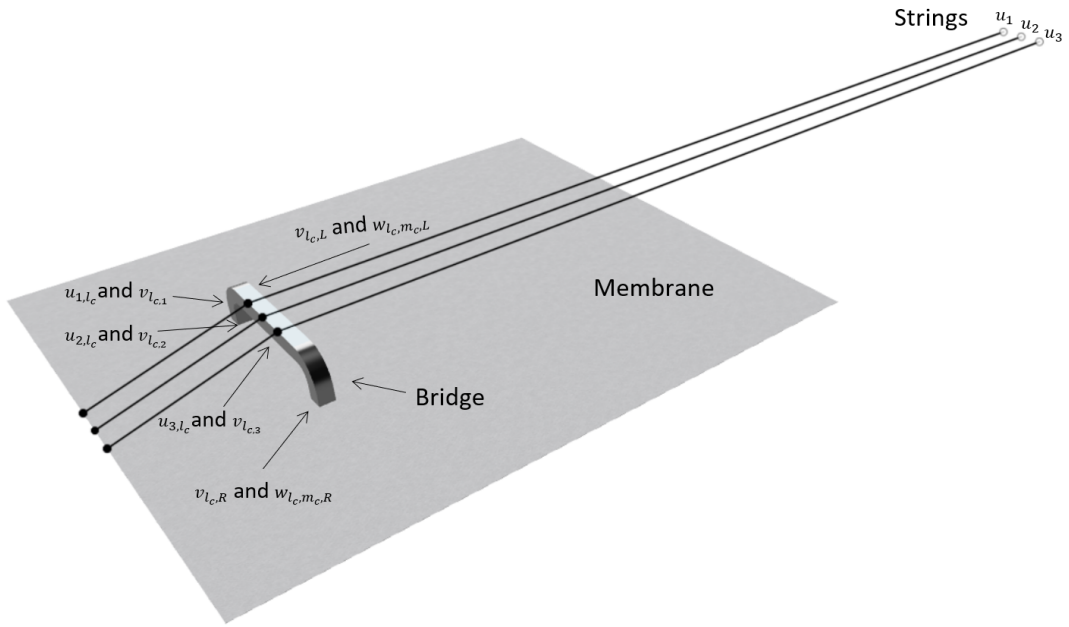


Figure 3.4: Coupled system

The string equation at the connection points $l_{c,Si}$ can be found by taking an inner product with J, which is described in section 3.6, and only considering the connection points.

$$\delta_{tt}u_{i,l_{c,Si}} = {c_{Si}}^2\delta_{xx}u_{i,l_{c,Si}} - {\kappa_{Si}}^2\delta_{xxxx}u_{i,l_{c,Si}} - 2\sigma_{0,Si}\delta_{t\cdot}u_{i,l_{c,Si}} + 2\sigma_{1,Si}\delta_{t-}\delta_{xx}u_{i,l_{c,Si}} - \frac{F_{ti}}{h_{Si}\rho_{Si}A_{Si}} \tag{3.92}$$

The i in the subscript has the same meaning as it did in the section above. The sampled strings would have a different number of intervals N_S, depending on the thickness and the tension of the individual strings (as shown by equation 3.27), thus the connection point for each string is different, denoted by $l_{c,Si}$.
The bridge model has the highest amount of coupling points, 3 of which are for connecting the strings to the bridge:

$$\delta_{tt}v_{l_{c,Bi}} = -{\kappa_B}^2\delta_{xxxx}v_{l_{c,Bi}} - 2\sigma_{0,B}\delta_{t\cdot}v_{l_{c,Bi}} + 2\sigma_{1,B}\delta_{t-}\delta_{xx}v_{l_{c,Bi}} + \frac{F_{ti}}{h_B\rho_B A_B} \tag{3.93}$$

i in the subscript denotes the connection point to the corresponding string.
Two coupling points are for connecting the bridge to the membrane. These points

are named $l_{c,BL}$ and $l_{c,BR}$ for left and right mounting points (figure 3.4).

$$\begin{aligned}\delta_{tt}v_{l_{c,BL}} &= -{\kappa_B}^2\delta_{xxxx}v_{l_{c,BL}} - 2\sigma_{0,B}\delta_{t\cdot}v_{l_{c,BL}} + 2\sigma_{1,B}\delta_{t-}\delta_{xx}v_{l_{c,BL}} - \frac{F_{bL}}{h_B\rho_B A_B}\\ \delta_{tt}v_{l_{c,BR}} &= -{\kappa_B}^2\delta_{xxxx}v_{l_{c,BR}} - 2\sigma_{0,B}\delta_{t\cdot}v_{l_{c,BR}} + 2\sigma_{1,B}\delta_{t-}\delta_{xx}v_{l_{c,BR}} - \frac{F_{bR}}{h_B\rho_B A_B}\end{aligned} \tag{3.94}$$

It is important that the connection points of the bridge do not overlap as such scenario would directly couple the string to the membrane instead of letting the energy precipitate from the string through the bridge to the membrane.

Membrane is coupled with the bridge in two locations $l_{c,ML}, m_{c,ML}$ and $l_{c,MR}, m_{c,MR}$, each on the opposite sides of the bridge via forces F_{bL} and F_{bR}.

$$\begin{aligned}\delta_{tt}w_{l_{c,ML},m_{c,ML}} &= {c_M}^2\delta_\Delta w_{l_{c,ML},m_{c,ML}} - {\kappa_M}^2\delta_\Delta\delta_\Delta w_{l_{c,ML},m_{c,ML}}\\ &\quad - 2\sigma_{0,M}\delta_{t\cdot}w_{l_{c,ML},m_{c,ML}} + 2\sigma_{1,M}\delta_{t-}\delta_\Delta w_{l_{c,ML},m_{c,ML}} + \frac{F_{bL}}{{h_M}^2\rho_M H_M}\\ \delta_{tt}w_{l_{c,MR},m_{c,MR}} &= {c_M}^2\delta_\Delta w_{l_{c,MR},m_{c,MR}} - {\kappa_M}^2\delta_\Delta\delta_\Delta w_{l_{c,MR},m_{c,MR}}\\ &\quad - 2\sigma_{0,M}\delta_{t\cdot}w_{l_{c,MR},m_{c,MR}} + 2\sigma_{1,M}\delta_{t-}\delta_\Delta w_{l_{c,MR},m_{c,MR}} + \frac{F_{bR}}{{h_M}^2\rho_M H_M}\end{aligned} \tag{3.95}$$

Since force only acts in these specific locations:

Discrete Connection points

Strings	$l_{c,S1}$	$l_{c,S2}$	$l_{c,S3}$
Bridge	$l_{c,B1}$ $l_{c,BL}$	$l_{c,B2}$ $l_{c,BR}$	$l_{c,B3}$
Membrane	$l_{c,ML}$, $m_{c,ML}$	$l_{c,MR}$, $m_{c,MR}$	

Table 3.2: All the connection points in this discrete system. Subscripts S1, S2 and S3 indicate that the connection points are on different strings, B1, B2 and B3 are the corresponding connection points on the bridge for these strings. BL and BR subscripts indicate connection points on the left side and the right side of the bridge, in turn, ML and MR subscripts are the same points on the membrane.

We only need to update models with force at those specific points using equations 3.96 and 3.97 and equations 3.63, 3.70 and 3.78 at the connection points.

$$u^I_{i,l_{c,Si}} - \frac{F_{ti}}{h_{Si}\rho_{Si}A_{Si}(\sigma_{1,Si}+1)} = v^I_{l_{c,Bi}} + \frac{F_{ti}}{h_i\rho_B A_B(\sigma_{1,B}+1)} \tag{3.96}$$

$$\begin{aligned}v^I_{l_{c,BL}} - \frac{F_{bL}}{h_B\rho_B A_B(\sigma_{0,B}+1)} &= w^I_{l_{c,ML},m_{c,ML}} + \frac{F_{bL}}{{h_M}^2\rho_M H_M(\sigma_{0,M}+1)}\\ v^I_{l_{c,BR}} - \frac{F_{bR}}{h_B\rho_B A_B(\sigma_{0,B}+1)} &= w^I_{l_{c,MR},m_{c,MR}} + \frac{F_{bR}}{{h_M}^2\rho_M H_M(\sigma_{0,M}+1)}\end{aligned} \tag{3.97}$$

Updating all the other spatial model points equations 3.63, 3.70, 3.78 are used. Doing so decreases computational complexity of the system. Lossless coupling means

that the net of the forces at the same location is equal to zero. Since there are only two forces acting at any given coupling location, forces are equal in magnitude, but opposite in direction. Solving for forces in equations 3.96, 3.97, expanded update equations for the forces are obtained in equation 3.98.

$$\begin{aligned} F_{ti} &= \frac{1}{c_{F_{tSi}} + c_{F_{tB}}}(u^I_{i,l_{c,Si}} - v^I_{l_{c,Bi}}) \\ F_{bL} &= \frac{1}{c_{F_{bB}} + c_{F_{bM}}}(w^I_{l_{c,MR},m_{c,MR}} - v^I_{l_{c,BL}}) \\ F_{bR} &= \frac{1}{c_{F_{bB}} + c_{F_{bM}}}(w^I_{l_{c,MR},m_{c,MR}} - v^I_{l_{c,BR}}) \end{aligned} \tag{3.98}$$

The updated forces are used when calculating equations 3.92, 3.93, 3.94 and 3.95. The multipliers $c_{F_{tSi}}$, $c_{F_{tB}}$, $c_{F_{bB}}$ and $c_{F_{bM}}$ are defined as:

$$\begin{aligned} c_{F_{tSi}} &= \frac{1}{\rho_{Si} A_{Si} h_{Si}(\sigma_{0,Si} + 1)} \\ c_{F_{tB}} &= \frac{1}{\rho_B A_B h_B(\sigma_{0,B} + 1)} \\ c_{F_{bB}} &= \frac{-1}{\rho_B A_B h_B(\sigma_{0,B} + 1)} \\ c_{F_{bM}} &= \frac{-1}{\rho_M H_M h_M^2(\sigma_{0,M} + 1)} \end{aligned} \tag{3.99}$$

3.8 Performance

In this section, a method of producing sound will be explained. Sound production is made up by the excitation, fretting for playing different notes and the output position and gain.

3.8.1 Excitation

System is excited by plucking one of the strings. Simplified plucking is modeled as a hann window at the start of the simulation. Since Shamisen uses the sharp edge of the plectrum to excite the stings [2], the width of the hann window was chosen to be $\lfloor N_{Si}/10 \rfloor$. The position of the pluck can change the sound significantly. If the pluck position is set to one of the node locations of a string mode, this mode will be fully attenuated. Figure 3.5 is an example of a string vibrating at mode 4, where string length is L and one of the node locations is at the point A. String at the node locations has to be stationary for this mode to exist, exciting string at one of the node locations would attenuate all the modes that have the node in that location, to avoid that, pluck is placed at $\lfloor (N_{Si}5)/(6\pi) \rfloor$. Irrational multiplier minimizes the

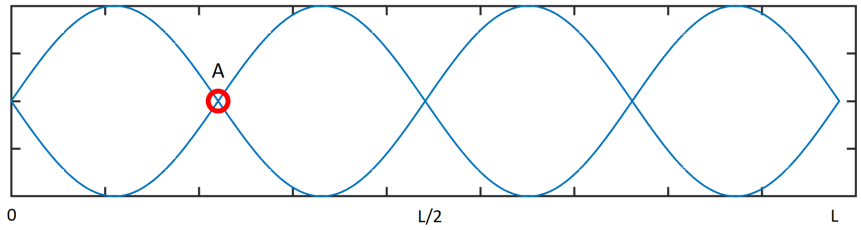


Figure 3.5: String vibration at 4th mode. Node location $A = L/4$

probability of plucking the string at the node location, at the same time, together with other multipliers places the pluck somewhere on the first half of the string.

3.8.2 Fretting

To achieve different pitches without changing the tension of the strings a string player pinches a string against the neck, thus restricting the spread of vibrations further down the sting, essentially shortening the string length. The simplified approach to achieve similar result would be to limit the length of the string that is being updated. It only works, when all the strings start with initial displacement at zero except the excitation area. Since the system is discrete, fretting positions have to be tuned manually. Each string has a different fretting position due to the stings having different sized grids as it can be shown in table 3.3.

Fretting stings at different locations will affect the string as seen in the figure 3.6.

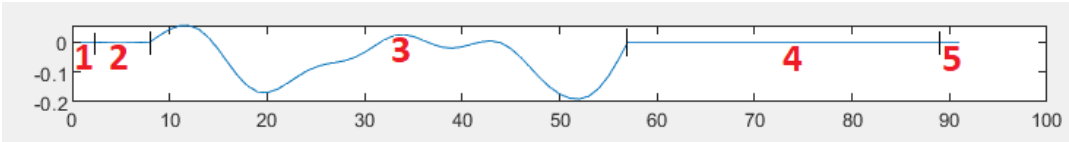


Figure 3.6: Excited fretted first string at time sample n where x axis plots spatial samples l_{S1} and y axis plots the string displacement in meters. Different numbers on the graph are denoting different parts of the string divided by black perpendicular lines: 1 is the left boundary; 2 is the part between the bridge connection and the left boundary; 3 is the active vibrating string; 4 is the part where the string does not vibrate because the fretting has restricted transfer of vibrations; 5 is the right boundary

3.8.3 Output

It is important to decide how the sound will be extracted from the parts of the model. Output of the string is taken from a single point, position of which is set to be $\lfloor (2N_{Si})/(7\pi) \rfloor + 4$ which is the location near the bridge. It is important to keep the output at a location described by an irrational number for the reason described

First String			Second String			Third String		
Position	Note	Error	Position	Note	Error	Position	Note	Error
0	4C	0	0	4G	0	0	5C	0
4	4Db	0	3	4Ab	–4	2	5Db	–16
8	4D	0	6	4A	0	5	5D	+6
13	4Eb	0	9	4Bb	+4	7	5Eb	+3
17	4E	0	12	4B	+7	9	5E	+10
21	4F	0	14	5C	0	11	5F	+17
24	4Gb	0	16	5Db	–16	12	5Gb	–19
27	4G	0	19	5D	+3	14	5G	0
30	4Ab	0	21	5Eb	+5	16	5Ab	+24
33	4A	–8	23	5E	+3	17	5A	–6
36	4Bb	0	25	5F	+10	18	5Bb	–31
39	4B	+10	26	5Gb	+23	20	5B	+12
41	5C	0	28	5G	0	21	6C	0
43	5Db	–13	30	5Ab	+22	22	6Db	–16
45	5D	–21	31	5A	–8	23	6D	–27
47	5Eb	–20	32	5Bb	–27	24	6Eb	–43
49	5E	–22	34	5B	+10	25	6E	–35
51	5F	–18	35	6C	–5	26	6F	–29
53	5Gb	0	36	6Db	–18	27	6Gb	–23
55	5G	+15	37	6D	–29	28	6G	+1
56	5Ab	–11	38	6Eb	–40	29	6Ab	+24
58	5A	+15	39	6E	–39			
59	5Bb	0	40	6F	–34			
60	5B	–18	41	6Gb	–23			
61	6C	–37	42	6G	–2			
63	6Db	+27	43	6Ab	+21			
64	6D	+20						
65	6Eb	+19						

Table 3.3: Discrete fretting positions for each of the strings. *Position* columns show how many additional points from the far end of the string are set to zero to produce a note from the *Note* column. The *Error* columns show by how many cents the fretted string is off from the equal-tempered tuning of A4 = 440Hz.

in the section 3.8.1, while also giving enough room for the fretting, in addition, the output taken from the left side of the bridge will have a low amplitude as the string vibrations are subdued there by the bridge. The output location for the bridge is not very important as it has a minimal effect on the sound by itself, more importantly it transfers energy from the string to the membrane, either way the output position for the bridge is chosen to be $\lfloor (2N_B)/(\pi) \rfloor$. The output of the

membrane is the sum of all the grid points for the full sized grid. For smaller grid sizes, membrane output must be multiplied to match the full sized grid output. The multiplication factor m_{out} is calculated using equation 3.100.

$$m_{out} = \frac{(N_{Mx,F} - 4) \cdot (N_{My,F} - 4))}{(N_{Mx,C} - 4) \cdot (N_{My,C} - 4))} \tag{3.100}$$

Where F in the subscript denotes the full sized grid size and C in the subscript denotes the current grid size. The boundary conditions make the first two and the last two grid points to be zero, thus 4 is subtracted from grid width and height to obtain an accurate multiplication factor.

3.9 Parameter acquisition

This model is uses quite a few parameters that shape the sound. The table 3.4 has all of the parameters listed with the values or the equations (if the parameter can be described by the values of the other parameters).

3.9.1 Strings

Strings have a circular cross section and they can be made from a variety of different materials, the more traditional material is silk, while the material which is becoming more common is nylon as the strings made from nylon take longer to wear out. For this model, wearing out strings is not an issue thus silk strings are chosen. The silk density of is not traditionally measured in kg/m^3, instead denier is used as it is a fiber. Having string density measured in denier does not translate directly to the specific mass measured in kg/m^3, thus separate calculations are required. The silk moth silk (latin Bombyx mori) is coming out in stands which have triangular cross sections with a known average thickness [17] [18]. Packing as many triangles in a circular area and using denier to find mass, a rough approximation of the string specific mass can be calculated, the calculations of which are beyond the scope of this project. Young's modulus of Bombyx mori silk has been measured and published [18]. Radii of the strings has been set while tuning the instrument as the exact radii of the Shamisen strings could not be found. The main parameter when tuning the strings is the tension, values for which were found manually. Strings are connected to the bridge near the mounting knots which are called Neo.

3.9.2 Bridge

The cross sectional area of the bridge (Koma) is a rectangle, with an assumed width of around 2.69 mm as the regular Shamisen bridges are usually thin pieces made from wood, ivory or plastic. Bridge used in this model is chosen to be made from

acrylic, which gives the Young's modulus [19] and specific mass [20] for the bridge. The thickness of the bridge is an average height of the Komas being sold [21]. The bridge connects to the membrane at the ends, while the strings are equally spaced in between the connection points to the membrane and themselves.

3.9.3 Membrane

The membrane of shamisen is traditionally made from dog or cat skin [22], although from 2000s [23] replacing natural skin by nylon membrane became more and more popular [24]. Due to the availability of the parameters, membrane has been chosen to be made from nylon, giving Young's modulus [19] and specific mass [20]. Both, frequency independent and dependent damping parameters were set manually. Grid aspect ratio was set to 1.3, following rough proportions of the Shamisen body [25]. Poisson's ratio has to be lower than $\frac{1}{2}$ thus a ratio of 0.4 has been chosen to satisfy this condition. The thickness of the membrane is set to an average thickness of a single ply drum membrane [26] as the thickness of the shamisen membrane could not be obtained. The tension on the membrane is manually tuned. The bridge connects to the membrane at two points in a horizontal axis.

Notation	Value	Unit	Parameter
A_B	$b \cdot H_B$	m^2	Cross sectional area of the bridge
A_{S1}	$\pi \cdot r_{S1}^2$	m^2	Cross sectional area of the first string
A_{S2}	$\pi \cdot r_{S2}^2$	m^2	Cross sectional area of the second string
A_{S3}	$\pi \cdot r_{S3}^2$	m^2	Cross sectional area of the third string
b	$2.69333 \cdot 10^{-3}$	m	Width of the bridge
E_B	$3.2 \cdot 10^9$	Pa	Young's Modulus of the bridge
E_M	$3 \cdot 10^9$	Pa	Young's Modulus of the membrane
E_S	$9.9 \cdot 10^9$	Pa	Young's Modulus of the strings
H_B	0.0075	m	Thickness of the bridge
H_M	0.0002	m	Thickness of the membrane
I_{S1}	$(\pi \cdot r_{S1}^4)/4$	$kg \cdot m^2$	Moment of inertia of the first string
I_{S2}	$(\pi \cdot r_{S2}^4)/4$	$kg \cdot m^2$	Moment of inertia of the second string
I_{S3}	$(\pi \cdot r_{S3}^4)/4$	$kg \cdot m^2$	Moment of inertia of the third string
$l_{c,S1}$	$\lfloor (2N_{S1})(\pi 7) \rfloor$		CP of the first string to the bridge
$l_{c,S2}$	$\lfloor (2N_{S2})(\pi 7) \rfloor$		CP of the second string to the bridge
$l_{c,S3}$	$\lfloor (2N_{S3})(\pi 7) \rfloor$		CP of the third string to the bridge
$l_{c,B1}$	$\left\lfloor (\frac{N_B}{2} - 1)/2 \right\rfloor$		CP of the bridge to the first string
$l_{c,B2}$	$\lfloor N_B/2 \rfloor$		CP of the bridge to the second string
$l_{c,B3}$	$\left\lceil \frac{\frac{N_B}{2}-1}{2} \right\rceil + l_{c,B2}$		CP of the bridge to the third string
$l_{c,BL}$	1		CP of the left side of the bridge to the membrane
$l_{c,BR}$	N_B		CP of the right side of the bridge to the membrane
$l_{c,ML}$, $m_{c,ML}$	$N_{Mx} - \lfloor 3N_{Mx}/5 \rfloor$, $N_{My} - \lfloor N_{My}/4 \rfloor$		CP of the membrane to the left side of the bridge
$l_{c,MR}$, $m_{c,MR}$	$N_{Mx} - \lfloor 2N_{Mx}/5 \rfloor$, $N_{My} - \lfloor N_{My}/4 \rfloor$		CP of the membrane to the right side of the bridge
ν	0.4		Poisson's ratio
r	1.3		Grid aspect ratio
r_{S1}	$4.15088 \cdot 10^{-4}$	m	Radius of the first string
r_{S2}	$2.83487 \cdot 10^{-4}$	m	Radius of the second string
r_{S3}	$2.10984 \cdot 10^{-4}$	m	Radius of the third string
ρ_B	1190	kg/m^3	Specific mass of the bridge
ρ_M	1150	kg/m^3	Specific mass of the membrane
ρ_S	1156.481	kg/m^3	Specific mass of the strings
$\sigma_{0,S}$	1.37803		Frequency independent damping of the string
$\sigma_{0,B}$	1.343		Frequency independent damping of the bridge
$\sigma_{0,M}$	1.37806		Frequency independent damping of the membrane
$\sigma_{1,S}$	$3.57021 \cdot 10^{-3}$		Frequency dependent damping of the string
$\sigma_{1,B}$	$4.59 \cdot 10^{-3}$		Frequency dependent damping of the bridge
$\sigma_{1,M}$	$9.60559 \cdot 10^{-2}$		Frequency dependent damping of the membrane
T_M	400	N/m	Tension on the membrane
T_{S1}	138.67	N	Tension on the first string
T_{S2}	145.53	N	Tension on the second string
T_{S3}	140.73	N	Tension on the third string

Table 3.4: List of parameter values used when simulating Shamisen. CP in the Parameter column is used to shorten the description, which stands for "Connection point".

Chapter 4

Evaluation and Discussion

In this chapter, two evaluation methods will be presented and employed to test the Shamisen model developed in this work. The evaluation begins with a more quantitative evaluation, where the quality of the sound is compared between the models with different sized membranes. After that a realism evaluation is conducted, where the sound of the biggest model is compared to the sound of a real Shamisen. In the first one, comparing models with different sized membranes is necessary since the membrane has the highest count of samples, which accounts to a longer computational time. Finding a smaller grid size for the membrane which still keeps a comparable sound quality to the full sized model is beneficial when thinking about real-time implementation in the future. In the second part of evaluation, subjects were asked to listen to a melody which was produced using a sample of a real Shamisen and a melody which was produced using samples of the Shamisen model developed in this work, then the open-ended question followed asking for the comments about the sound characteristics of the Shamisen model.

It is important to mention that during the sampling of the model for the test, the bridge by accident had its thickness set to be 10 times as large, which reduced the number of spatial samples from 78 to 22. After an informal listening test it was determined that this mistake did not produce any audible difference, thus it was decided to keep and discuss the results of the survey as if the mistake was not made due to the time constraints restricting running the test again with corrected samples.

4.1 Quality test

4.1.1 Setup

The largest membrane had an average computation time of 58.6 seconds in MATLAB on Lenovo IdealPad 5 with AMD Ryzen 7 4700U 2-4.1GHz processor which meant it would not work in a real-time implementation. The table 4.1 has the average computation times for models with different membrane grid sizes. The

grid size	average computation time
123×73	58.6 seconds
61×46	15.2 seconds
30×23	8.7 seconds
15×11	8.1 seconds
10×7	7.2 seconds
7×5	7.4 seconds

Table 4.1: Computation time comparison on different membrane grid sizes.

average computation times for different grid sizes flattens out at around 7.5. That can be explained by the fact that the lowest grid size has only 7×5 sized grid which gives total of 35 points in a grid, also the second smallest grid has 10×7 sized grid totaling with 70 points in a grid which when compared to the string grid sizes of 91×1, 62×1 and 47×1 is quite close in size thus the speed of computing the model does not depend on the membrane size more than the other parts of the model. The increase of the computation time for the lowest grid may happen due to the inner optimisation algorithms changing when presented with less demanding calculations. To determine the grid size suitable for the real-time implementation the output of the models with a different sized membrane grid have to be compared, thus a perceptual quality test was performed using MUSHRA methodology [27]. For testing, 4 musical phrases were used, two of which were popular Japanese pentatonic scales - Yo and Akebono and the other two were short excerpts from Japanese folk instrumental music - "Kayo Kami" and "Shashari"[28]. The audio files were created using MIDI notation and a sampler, where each note was mapped into the individual MIDI key[1]. Six audio files per musical phrase were given for the rating, additionally one audio file was given as a reference. The six rated audio files were created using different sized grid for the membrane in the Shamisen model. In between the rated files a reference was also included which is referred to as a hidden reference and instructions were given to rate the audio files which do not have any audible difference from the reference with a 100. The model with a membrane which has the most samples was used as a reference. The

[1]The audio files are available at https://github.com/titas2001/FDS_Shamisen_audio

order of all the rated audio files was randomised between the musical phrases.

4.1.2 Results

The survey had 19 participants, 8 of which miss-understood the rating system and their quantitative ratings were dismissed. Understanding of the rating system was checked with the hidden reference: if the person consistently rated hidden reference at values bellow 90, this meant that the instructions were not understood properly. Table 4.2 has the average quality scores of different membrane grid sizes as well as how many times on average through 4 musical phrases the audio file was indistinguishable from the reference.

grid size	Average quality score	People who heard it as the reference
123×73	94.59	57.2%
61×46	93.27	40.9%
30×23	93.00	45.5%
15×11	59.16	0%
10×7	66.36	0%
7×5	56.93	0%

Table 4.2: Quantitative results of the questionnaire. Average quality score is the mean value of all the accepted ratings. The value in third column is calculated by counting how many ratings of a 100 each grid size had over all 4 musical phrases and then it was divided by 4 and converted to the %.

The table 4.2 shows that the quality of the sound drops significantly with the grid size 15×11 and bellow. The largest grid had the highest average and the most people rated it being the same as the reference, although it was the hidden reference and ideally, it would have all the participants recognise it as one. The second largest grid size had the highest average score, but it also had less people rating it as indistinguishable from the reference than the 30×23 grid had. Matching the results from the quality test with the table 4.1 we can determine that using a membrane with a grid of 30×23 would be the best option, as it has a computation time almost 7 times faster than the full sized grid while also having a comparable audio quality.

4.2 Verity test

4.2.1 Setup

Realism evaluation is conducted right after the MUSHRA test. During this evaluation, participants are given two audio files with an excerpt from "Shigin-Cho" by Nanagon Zeku[28]. As before, when creating melodies midi files were used with a sampler, but this time one of the audio files was a using a sample of a

real Shamisen playing C4 while the other audio file was playing the melody using sampled Shamisen model from this work. Participants responded by writing open-ended answers on how they felt about the modeled Shamisen sound.

4.2.2 Results

The main consensus was that the modeled sound lacked the distinctive percussive attack of the instrument, which is created by following the plectrum motion after it had excited the string until it hits the edge of the instrument [2] and that it did not buzz as the real Shamisen does, which is related to the short nut allowing stings to vibrate against the neck of the instrument[1]. These comments were expected as the percussive attack of the bachi and the string rattling were not modeled. Other detail that many participants pointed out was that the model has too long of a decay time and the membrane resonates at a lower frequency than in the real Shamisen. Both of these miss-matches could possibly be rectified by tuning the parameters of the system, if the tuning still could not match the decay and the pitch of the real instrument, modeling the buzzing of the strings and the collision of the bachi might prove to be necessary. In addition, participants noticed that the model produces strong high harmonics, where in a real shamisen these are more subdued. Solving this problem might not be so straightforward as the other ones but the tuning the parameters and adding the percussive bachi hit model might just solve it. Overall the comments about the sound leave more work on tuning and modeling to be desired.

Chapter 5

Future Work

5.1 The Real-Time Implementation

The state of the current model is not ideal, but it would still be beneficial to implement the model in a real-time application in a form of a plugin. The quality test in section 4.1 reveals the optimal sized grid for the membrane, which performance could be tested in the plugin. At the same time the real-time suitability of the current grid sizes of the strings and the bridge would be tested. Moreover the user interface (UI) could be designed and passed through the testing phases.
The first iteration of the plugin could be designed using MATLAB Audio Plugin libraries [29] as it would not require converting the code from the MATLAB code to some other programming language, although some alterations are needed when creating the plugin using MATLAB.
The further steps in creating a more developed audio plugin would require rewriting the code to C++ and using a framework specifically designed for creating audio plugins such as JUCE [30]. Moving to using a dedicated framework would add additional control to the design elements of the plugin and a plugin written in C++ could potentially further improve the computational speed of the plugin.

5.2 Model Improvements

In this section future improvements regarding the modeling aspect of the work will be discussed. The improvements are presented in the order of how much the addition of the model could potentially improve the sound verity.

5.2.1 Collisions Model

As discussed in section 4.2 the sound produced by the model lacks the rattling that the real shamisen is distinguished by. Creating the rattling requires a complex

collisions modelling. The collisions could be modeled as described in [31], but instead of rattling the bridge as it is done by paper authors, the strings would be rattling.

5.2.2 Excitation model

Coming back to the section 3.8.1 the excitation is a simple hann window at the first time step. To create a more realistic excitation model the pluck can be modeled by increasing the amplitude of the hann window over time and releasing the string at the highest point. Achieving that is a matter of creating an excitation which follows sinusoidal curve from 0 to 1 and as it reaches 1, the amplitude of hann window would go to zero [32]. Changing the speed and the amplitude of the pluck would add another layer of control - the volume of the instrument. In a case, where the rattling of the string does not produce a convincing timbre of the instrument, the bachi could be modeled as a stiff plate [33] with 3 free boundaries where a mallet like hit on the edge of it could simulate the bachi hitting the edge of the instrument adding to the percussive attack sound.

5.2.3 Fretting model

In order to further develop the accuracy of the Shamisen model a simplified solution of fretting described in section 3.8.2 must be replaced by a String/Fretboard interaction model proposed in [34]. The authors of the paper describe a method of fretting either fretted or fretless instrument strings using FDS. The proposed method involves modeling a rigid barrier bellow the strings which would simulate the fretboard acting as a collision model similar to the ones modeling the striking action. Another part of the fretting model would be the model of a fretting finger, which would press the string down on the fretboard. The finger is modeled as an another collision model but this time the model would include the damping effects and a mass. While the fretboard is modeled as an immovable object, the finger has to move in order to press down on the strings. Incorporating this model would attend to one of the concerns pointed out in the section 4.2 - the longer sustain of the model when compared to the real instrument.

Chapter 6

Conclusion

A properly working Shamisen model had been implemented in MATLAB using FDS. The models of the strings, the bridge and the membrane had been coupled and are behaving as expected. The goal of implementing a real-time Shamisen plugin was not reached due to the time constraints although the section 5.1 discusses methods of reaching the set-out goal. The evaluation had provided valuable data for the development of the real-time plugin. Also, it pointed out that the model timbre does not yet match the real Shamisen, thus the simplified model of the Shamisen proved to be non-sufficient. Improvements discussed in the section 5.2 when implemented could possibly synthesise a more realistic Shamisen sound and become a viable alternative for the currently used recorded samples.

Bibliography

[1] Brussels Musical Instruments Museum. *Shamisen*. http://www.mim.be/shamisen. (Accessed on 12/16/2020).

[2] By Ono. *How to play a shamisen/Way to play a shamisen is explained here in the picture and moving image.* https://www.shamisen.info/ehikikatasan/kotonosoho.html. 2009.

[3] Dale Roll. *Beautiful shamisen featuring original illustration by Final Fantasy designer now on sale | SoraNews24 -Japan News-*. https://soranews24.com/2018/11/29/beautiful-shamisen-featuring-original-illustration-by-final-fantasy-designer-now-on-sale/. (Accessed on 12/16/2020). Nov. 2018.

[4] Azyri. *Shamisen_diagram.jpg (49643512)*. https://upload.wikimedia.org/wikipedia/commons/7/76/Shamisen_diagram.jpg. Aug. 2018.

[5] Le Loi. *Shamisen diagram*. https://en.wikipedia.org/wiki/Shamisen#/media/File:Shamisen_diagram.svg. Oct. 2018.

[6] Julius O. Smith. "Physical Modeling Using Digital Waveguides". In: *Computer Music Journal*. Vol. 16. 1992, p. 74.

[7] Silvin Willemsen, Stefania Serafin, Stefan Bilbao, and Michele Ducceschi. "Real-Time Implementation of a Physical Model of the Tromba Marina". In: *Proceedings of the 17th Sound and Music Computing Conference*. June 2020.

[8] Cumhur Erkut and Matti Karjalainen. "Finite difference method vs. digital waveguide method in string instrument modeling and synthesis". In: *Proceedings of the International Symposium on Musical Acoustics (ISMA-02), Mexico City, 2002*. Sept. 2002.

[9] *ShamiKoto Virtual Koto and Shamisen VST VST3 Audio Unit Plugin. Traditional Japanese Musical Instruments*. https://syntheway.com/ShamiKoto.htm. (Accessed on 12/16/2020).

[10] *Tanbur*. http://www.turkosfer.com/tanbur/. (Accessed on 12/17/2020).

[11] *Banjo 17 fret - Metal Rim Trad instrument*. https://www.freyaguitars.ie/product/4-string-banjo-17-fret-koda-with-gigbag/. (Accessed on 12/17/2020).

[12] *Tambur*. https://www.ktb.gov.tr/EN-98665/tambur.html. (Accessed on 12/17/2020).

[13] A. C. Yu. *Shamisen (a three-stringed Japanese banjo) - Japanese Wiki Corpus*. https://japanese-wiki-corpus.github.io/culture/Shamisen%20(a%20three-stringed%20Japanese%20banjo).html. (Accessed on 12/14/2020).

[14] Florian Pfeiffle and Rolf Bader. *Real-Time Physical Modelling of a complete Banjo geometry using FPGA hardware technology*. http://systmuwi.de/Pdf/Papers/Bader%20papers/Physical%20Modeling/PhysicalModeling_Banjo/Pfeifle,Bader_04FPGA_Format.pdf. (Accessed on 12/14/2020).

[15] Stefan Bilbao. *Numerical Sound Synthesis*. John Wiley & Sons, Ltd, 2009. ISBN: 9780470749012. DOI: 10.1002/9780470749012. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470749012. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470749012.

[16] Silvin Willemsen, Anca-Simona Horvath, and Mauro Nascimben. "DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study". English. In: *Proceedings of the 17th Sound and Music Computing Conference*. Ed. by Simone Spagnol and Andrea Valle. 2020, pp. 292–299.

[17] Menachem Lewin. "Handbook of Fiber Chemistry". English. In: Marcel Dekker, 1998, 438–441. ISBN: 0-8247-9471-0.

[18] Frank Ko, Sueo Kawabata, Mari Inoue, Masako Niwa, Stephen Fossey, and John Song. "Engineering Properties of Spider Silk Fibers". In: *MRS Proceedings*. Vol. 702. Jan. 2001. DOI: 10.1557/PROC-702-U1.4.1.

[19] *Young's Modulus - Tensile and Yield Strength for common Materials*. https://www.engineeringtoolbox.com/young-modulus-d_417.html. 2003.

[20] *Density of Selected Solids*. https://www.engineeringtoolbox.com/density-solids-d_1265.html. 2009.

[21] *Tsugaru Shamisen Koma (Bridge) — Bachido Store*. https://bachido.com/store/. (Accessed on 12/12/2020).

[22] *Tugaru Jamisen*. http://www.shamisen-katoh.com/e/AXIS.htm. Jan. 2001.

[23] Dave Hueston. *Shamisen faces crisis as cat skins fall from favor*. https://www.japantimes.co.jp/culture/2016/12/29/music/shamisen-faces-crisis-cat-skins-fall-favor/.

[24] Miki Minoru and Philip Flavin. *Composing for Japanese Instruments*. University of Rochester Press, 2008.

[25] Viktorija Reuta. *Vector Illustration Traditional Japanese Sting Instrument Shamisen. Royalty Free Cliparts, Vectors, And Stock Illustration. Image 85576071.* https://www.123rf.com/photo_85576071_stock-vector-vector-illustration-traditional-japanese-sting-instrument-shamisen-.html. (Accessed on 12/12/2020).

[26] *What You Need to Know About...Drumheads*. https://www.moderndrummer.com/2011/10/what-you-need-to-know-about-drumheads/. Dec. 2010.

[27] *RECOMMENDATION ITU-R BS.1534-1 - Method for the subjective assessment of intermediate quality level of coding systems*. https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.1534-1-200301-S!!PDF-E.pdf. 2001-2003.

[28] Tom Potter. *Daisyfield.com Archive of Japanese Traditional Music*. http://www.daisyfield.com/music/htm/-genres/japan.htm. (Accessed on 12/13/2020).

[29] *Audio Plugin Creation and Hosting - MATLAB & Simulink*. https://www.mathworks.com/help/audio/audio-plugin-creation-and-hosting.html. (Accessed on 12/17/2020).

[30] *JUCE*. https://juce.com/. (Accessed on 12/17/2020).

[31] Michele Ducceschi and Stefan Bilbao. "Non-iterative solvers for nonlinear problems: the case of collisions". In: *Proceedings of the 22nd Conference of Digital Audio Effects (DAFx-19)*. Sept. 2019.

[32] Silvin Willemsen, Stefania Serafin, Stefan Bilbao, and Nikolaj Andersson. "Real-Time Control of Large-Scale Modular Physical Models Using the Sensel Morph". In: *Proceedings of the 16th Sound and Music Computing Conference*. July 2019.

[33] Stefan Bilbao. "Linear plate vibration". In: *Numerical Sound Synthesis*. John Wiley & Sons, Ltd, 2009. Chap. 12, pp. 330–359. ISBN: 9780470749012. DOI: 10.1002/9780470749012.ch12. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470749012.ch12. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470749012.ch12.

[34] Stefan Bilbao and Alberto Torin. "Numerical Modeling and Sound Synthesis for Articulated String/Fretboard Interactions". In: *Journal of the Audio Engineering Society*. Vol. 63. June 2015. DOI: 10.17743/jaes.2015.0023.

Chapter 7

Appendix

	Yo scale			Akebono scale			"Kayo Kami"			"Shashari"		
grid size	$\overline{X}$	100	> 90	$\overline{X}$	100	> 90	$\overline{X}$	100	> 90	$\overline{X}$	100	> 90
123×73	97.64	7	10	96.18	7	9	93.27	5	8	91.27	6	8
61×46	95.36	5	10	93.00	3	9	96.18	5	11	88.55	5	7
30×23	90.18	4	8	97.00	7	11	91.45	4	7	93.36	5	10
15×11	55.00	0	1	59.64	0	1	59.27	0	2	62.73	0	1
10×7	63.64	0	2	55.55	0	1	69.91	1	2	76.36	2	3
7×5	52.45	0	0	53.36	0	0	58.82	0	1	63.09	0	0

Table 7.1: Quantitative results of the questionnaire. Columns marked with $\overline{X}$ are the mean values of the quality ratings; Columns with a 100 show how many people rated the sound to be indistinguishable from the reference; Columns with > 90 show how many people rated the audio file with a score above 90, which might be useful, when considering cases, where the slider accidentally slipped off the 100 value.

The unaltered open-ended responses about the modeled sound when compared to the real instrument sound are listed bellow. Some responses uses "the first" and "the second" when talking about the audio files, which refer to the melody created from a real instrument sample and the melody created from the modeled sound respectively.

- Seems like the specific "buzzing" of the original instrument is not well produced on the virtual one.
- Virtual Shamisen seems to sound overwhelming, more strong and more bass, which I do not prefer. Real Shamisen sounds more light, more airy and therefore I prefer it more.
- Virtual Shamisen sounds a lot more synthetic and high pitched compared to real. Also real has a lot more depth vs virtual sounds flat.

- I feel the virtual sound is too "smooth", in the sense of less dirty, less random, more "perfect" (like a synth), and that is not how the sampled shamisen sounds. Also, noise decays before in the Samples Shamisen.... I'm missing the "hit" sound in the virtual. (I know there is, but needs more force).
- It is super hard to compare these 2 last files, since the first audio was made fully naturally, most likely with the live instrument, while the second audio was made within computer. However the melody seems to be fully the same. Just the instrument is totally different.
- The virtual Shamisen sounds as it is played on different musical instruments than the original one. The sounds are longer and last more in the virtual one.
- It seems that first audio has a purer sound
- Apart from the melody the audios are quite different. The real Shamisen has quite vibrant sounds, melody sounds bit more organic than the second audio. The second audio has a sound somewhat similar to the sound of a guitar, but it is clear that the sound is generated by a computer.
- The real shamisen has a more "wooden" sound and thus more real sound. Also, there is much less high-frequency content than in the simulated one.
- The first one is much more better than the second one. More realistic sounds and in the second one there is more electronic sounds.
- The virtual Shamisen sounds more like a keyboard, less echo and more base.
- i really like how you can hear the resonance of a body in the virtual Shamisen. It makes the sound way more realistic even that the real Shamisen samples do not have that sound. Overall I think it's a great physical model.
- Virtualaus instrumento tembras atrodo aukštesnis, garsas labiau aidi, yra skardesnis. Fone girdisi žemo tembro beat'as, kurio nėra originaliame įraše.
- Virtual Shamisen has a low almost kick like sounds for each note (maybe increase attack slightly?) Virtual Shamisen sounds much more bright than the real one. I think this would benefit from more egg, who doesn't love egg?
- The vibrating membrane seems in a lower tune with respect to the real one. The attack sound appears to be lost in the virtual instrument.
- They sound different, you can clearly identify that its the same melody but it sounds like the second one needs a bit more of practice.
- oh no, I couldn't hear this last piece I don't know what happened. But super cool project, sounds great congrats :))) Ramon

- Sound 2 was clearer compared go sound 1.

Listing 7.1: The MATLAB code used to model and sample the Shamisen

```
clear all;
close all;
clc

scale = 7;
shamisenString = 0;
fs = 44100;                % sampling freq
k = 1/fs;                  % time step
TS1 = 14.15*9.8;             % applied string tension https://
    mk0larsenstringsti68.kinstacdn.com/wp-content/uploads/2018/12/
    Larsen-String-Tension-Charts-18.pdf
TS2 = 14.85*9.8;             % applied string tension
TS3 = 14.36*9.8;             % applied string tension
TP = 400;               % applied plate tension
rhoS = 1156.48151991993;% material density of the string
                           % "Handbook of Fiber Chemistry", Menachem
                               Lewin, Editor, 2nd ed.,1998, Marcel Dekker,
                                pp. 438 441 , ISBN 0-8247-9471-0
                           % "ENGINEERING PROPERTIES OF SPIDER SILK"
                               http://web.mit.edu/course/3/3.064/www/
                               slides/Ko_spider_silk.pdf
rhoP = 1150;               % nylon https://www.engineeringtoolbox.com/
    engineering-materials-properties-d_1225.html
HP = 0.0002;                % plate thickness
EP = 3e+9;                 % nylon https://www.engineeringtoolbox.com/
    engineering-materials-properties-d_1225.html
nu = 0.4;                  % Poisson s ratio nu < 0.5
r = 1.3;                   % grid aspect ratio
Lx = r*0.4;                % length of plate in x direction
Ly = (1/r)*0.4;            % length of plate in y direction
LS = 1;                  % lenght of the string
LB = 1;                    % lenght of the bridge
durration = 3;             % synthesised sound lenght in seconds
dur = fs*durration;        % synthesised sound lenght in samples
rhoB = 1190;                % material density
AreaS1 = 0.00000054129; % string cross sectional area
AreaS2 = 2.52473376e-7; % string cross sectional area
AreaS3 = 1.398451e-7; % string cross sectional area
AreaB = 2.02e-4;           % bridge cross sectional area
EB = 3.2e+9;               % Young's modulus acrylic https://www.
    engineeringtoolbox.com/young-modulus-d_417.html
HB = 0.0075;                % thickness
rS1 = sqrt(AreaS1/pi);  % string1 radius
rS2 = sqrt(AreaS2/pi);  % string2 radius
rS3 = sqrt(AreaS3/pi);  % string2 radius
ES =  9.9e+9;               % Young modulus "ENGINEERING PROPERTIES OF
    SPIDER SILK"  http://web.mit.edu/course/3/3.064/www/slides/
```

```
    Ko_spider_silk.pdf
38
39
40 gammaS1 = sqrt(TS1/(rhoS*AreaS1*LS^2));         % String tension
41 gammaS2 = sqrt(TS2/(rhoS*AreaS2*LS^2));         % String tension
42 gammaS3 = sqrt(TS3/(rhoS*AreaS3*LS^2));         % String tension
43 gammaP = sqrt(TP/(rhoP*HP*Lx*Ly));
44
45
46
47 IS1 = (pi*rS1^4)/4;      % string1 inertia
48 IS2 = (pi*rS2^4)/4;      % string1 inertia
49 IS3 = (pi*rS3^4)/4;      % string1 inertia
50 kappaB=sqrt((EB*HB^2)/(12*rhoB*LB^4)); % eq. 7.70 pg. 210
51
52 %
53 kappaS1=sqrt((ES*IS1)/(rhoS*AreaS1*LS^4));
54 kappaS2=sqrt((ES*IS2)/(rhoS*AreaS2*LS^4));
55 kappaS3=sqrt((ES*IS3)/(rhoS*AreaS3*LS^4));
56 D = EP*HP^3 / (12 * (1 - nu^2)); % plate flexural rigidity pg.331
57 kappaPsq = D / (rhoP * HP * Lx^2 * Ly^2); % pg.332 eq.12.3 kappa^2
58
59
60 sigmaS0 = 1.378027748373650;
61 sigmaS1 = 3.570213734102943e-03;
62 sigmaP0 = 1.378062296963499*2;
63 sigmaP1 = 0.096055930949692*2;
64 sigmaB0 =  1.343;
65 sigmaB1 =  0.00459;
66
67 hB = sqrt((4*sigmaB1*k+sqrt((4*sigmaB1*k)^2+16*kappaB^2*k^2))/2); %
    same method as for the string in pg. 176 but without gamma term
68 hS1 = sqrt((gammaS1^2 * k^2 + 4*sigmaS1*k + sqrt((gammaS1^2 * k^2 + 4*
    sigmaS1*k)^2 + 16*(kappaS1^2)*k^2))/2); % set grid spacing for
    String eq.7.24-25 pg.176
69 hS2 = sqrt((gammaS2^2 * k^2 + 4*sigmaS1*k + sqrt((gammaS2^2 * k^2 + 4*
    sigmaS1*k)^2 + 16*(kappaS2^2)*k^2))/2); % set grid spacing for
    String eq.7.24-25 pg.176
70 hS3 = sqrt((gammaS3^2 * k^2 + 4*sigmaS1*k + sqrt((gammaS3^2 * k^2 + 4*
    sigmaS1*k)^2 + 16*(kappaS3^2)*k^2))/2); % set grid spacing for
    String eq.7.24-25 pg.176
71 hP = sqrt((gammaP^2 * k^2 + 4*sigmaP1*k + sqrt((gammaP^2 * k^2 + 4*
    sigmaP1*k)^2 + 16*kappaPsq*k^2))); % set grid spacing for Plate
    tromba marina paper eq. 20
72
73
74
75 NS1 = floor(1/hS1);            % string spatial subdivisions
76 NS2 = floor(1/hS2);            % string spatial subdivisions
77 NS3 = floor(1/hS3);            % string spatial subdivisions
78 NB = floor(1/hB);              % bar spatial subdivisions
```

```
79  Nx = floor(sqrt(r)/hP);         % number of x-subdivisions of spatial
        domain
80  Ny = floor(1/(sqrt(r)*hP));     % number of y-subdivisions of spatial
        domain
81
82  if Nx > scale
83      Nx = scale;
84  end
85
86
87  hP = sqrt(r)/min(Nx, Ny);       % reset grid spacing for Plate
88  hS1 = 1/NS1;                    % reset grid spacing for String1
89  hS2 = 1/NS2;                    % reset grid spacing for String2
90  hS3 = 1/NS3;                    % reset grid spacing for String3
91  hB = 1/NB;                      % reset grid spacing for Bar
92  Ny = floor(1/(sqrt(r)*hP));     % number of y-subdivisions of spatial
        domain
93
94
95
96  %% Connection points
97  lBc1 = floor((NB/2 - 1)/2);        % bar connection to the 1st string
98  lBc2 = floor(NB/2);        % bar connection to the 2nd string
99  lBc3 = ceil((NB/2 - 1)/2)+lBc2;       % bar connection to the 3rd
        string
100 lBcl = 1;        % bar left side connection to the plate
101 lBcr = NB;       % bar right side connection to the plate
102
103 lS1c = floor((2*NS1)/(pi*7)); % 1st string connection to the bar
104 lS2c = floor((2*NS2)/(pi*7)); % 2nd string connection to the bar
105 lS3c = floor((2*NS3)/(pi*7)); % 3rd string connection to the bar
106
107 lPcl = Nx - floor(3*Nx/5); % Plate connection to the bar on the left
        side x coordinate
108 lPcr = Nx - floor(2*Nx/5); % Plate connection to the bar on the right
        side x coordinate
109 mPcl = Ny - floor(Ny/4);   % Plate connection to the bar on the left
        side y coordinate
110 mPcr = Ny - floor(Ny/4);   % Plate connection to the bar on the right
        side y coordinate
111
112 %% Multipliers
113
114 % Strings
115 uS1lMult = (((-2*gammaS1^2)/hS1^2 - 6*kappaS1^2/hS1^4 - 4*sigmaS1/(k*
        hS1^2))*k^2 + 2)/(k*sigmaS0 + 1);
116 uS1l1Mult = (gammaS1^2/hS1^2 + 4*kappaS1^2/hS1^4 + 2*sigmaS1/(k*hS1^2)
        )*k^2/(k*sigmaS0 + 1);
117 uS1l2Mult = ((-1*k^2*kappaS1^2)/(hS1^4))/(k*sigmaS0 + 1);
118 uS1PrevlMult = ((4*sigmaS1*k^2)/(k*hS1^2) + k*sigmaS0 - 1)/(k*sigmaS0
        + 1);
```

```
119 uS1Prevl1Mult = ((-2*sigmaS1*k^2)/(k*hS1^2))/(k*sigmaS0 + 1);
120
121 uS2lMult = (((-2*gammaS2^2)/hS2^2 - 6*kappaS2^2/hS2^4 - 4*sigmaS1/(k*
        hS2^2))*k^2 + 2)/(k*sigmaS0 + 1);
122 uS2l1Mult = (gammaS2^2/hS2^2 + 4*kappaS2^2/hS2^4 + 2*sigmaS1/(k*hS2^2)
        )*k^2/(k*sigmaS0 + 1);
123 uS2l2Mult = ((-1*k^2*kappaS2^2)/(hS2^4))/(k*sigmaS0 + 1);
124 uS2PrevlMult = ((4*sigmaS1*k^2)/(k*hS2^2) + k*sigmaS0 - 1)/(k*sigmaS0
        + 1);
125 uS2Prevl1Mult = ((-2*sigmaS1*k^2)/(k*hS2^2))/(k*sigmaS0 + 1);
126
127 uS3lMult = (((-2*gammaS3^2)/hS3^2 - 6*kappaS3^2/hS3^4 - 4*sigmaS1/(k*
        hS3^2))*k^2 + 2)/(k*sigmaS0 + 1);
128 uS3l1Mult = (gammaS3^2/hS3^2 + 4*kappaS3^2/hS3^4 + 2*sigmaS1/(k*hS3^2)
        )*k^2/(k*sigmaS0 + 1);
129 uS3l2Mult = ((-1*k^2*kappaS3^2)/(hS3^4))/(k*sigmaS0 + 1);
130 uS3PrevlMult = ((4*sigmaS1*k^2)/(k*hS3^2) + k*sigmaS0 - 1)/(k*sigmaS0
        + 1);
131 uS3Prevl1Mult = ((-2*sigmaS1*k^2)/(k*hS3^2))/(k*sigmaS0 + 1);
132
133 % Bar
134 uBlMult = (2*hB^4 - 4*hB^2*k*sigmaB1 - 6*k^2*kappaB^2)/(hB^4*(k*
        sigmaB0 + 1));
135 uBl1Mult = (2*hB^2*k*sigmaB1 + 4*k^2*kappaB^2)/(hB^4*(k*sigmaB0 + 1));
136 uBl2Mult = (-1*k^2*kappaB^2)/(hB^4*(k*sigmaB0 + 1));
137 uBPrevlMult = (hB^4*k*sigmaB0 - hB^4 + 4*hB^2*k*sigmaB1)/(hB^4*(k*
        sigmaB0 + 1));
138 uBPrevl1Mult = (-2*sigmaB1*k)/(hB^2*(k*sigmaB0 + 1));
139
140 % Plate
141 uPlMult = ((-20*kappaPsq/hP^4 - 4*gammaP^2/hP^2 - 8*sigmaP1/(k*hP^2))*
        k^2 + 2)/(k*sigmaP0 + 1);
142 uPl1Mult = (8*kappaPsq/hP^4 + gammaP^2/hP^2 + 2*sigmaP1/(k*hP^2))*k
        ^2/(k*sigmaP0 + 1);
143 uPl1dMult = (-2*kappaPsq*k^2)/(hP^4)/(k*sigmaP0 + 1);
144 uPl2Mult = (-1*kappaPsq*k^2)/(hP^4)/(k*sigmaP0 + 1);
145 uPPrevlMult = ((8*sigmaP1*k^2)/(k*hP^2) + k*sigmaP0 - 1)/(k*sigmaP0 +
        1);
146 uPPrevl1Mult = ((-2*sigmaP1*k^2)/(k*hP^2))/(k*sigmaP0 + 1);
147
148 % Forces
149 Fs1bMult = 1/(1/(rhoB*AreaB*hB * (sigmaB0 + 1)) + 1/(rhoS*AreaS1*hS1 *
         (sigmaS0 + 1)));
150 Fs2bMult = 1/(1/(rhoB*AreaB*hB * (sigmaB0 + 1)) + 1/(rhoS*AreaS2*hS2 *
         (sigmaS0 + 1)));
151 Fs3bMult = 1/(1/(rhoB*AreaB*hB * (sigmaB0 + 1)) + 1/(rhoS*AreaS3*hS3 *
         (sigmaS0 + 1)));
152 FbpMult = 1/(-1/(rhoB*AreaB*hB * (sigmaB0 + 1)) - 1/(rhoP*HP*hP^2 * (
        sigmaP0 + 1)));
153 if shamisenString == 1
154     frettingpos =
```

```
        [0,4,8,13,17,21,24,27,30,33,36,39,41,43,45,47,49,51,53,55,56,58,59,60,61,63,64,65];
           %1st string
155     noteName = ["C4","Db4","D4","Eb4","E4","F4","Gb4","G4","Ab4","A4
        ","Bb4","B4","C5","Db5","D5"...
156     ,"Eb5","E5","F5","Gb5","G5","Ab5","A5","Bb5","B5","C6","Db6","D6
        ","Eb6"]; %1st string
157 end
158 if shamisenString == 2
159     frettingpos =
        [0,3,6,9,12,14,16,19,21,23,25,26,28,30,31,32,34,35,36,37,38,39,40,41,42,43];
                %2nd string
160     noteName = ["G4","Ab4","A4","Bb4","B4","C5","Db5","D5","Eb5","E5
        ","F5","Gb5","G5","Ab5","A5"...          %2nd string
161     ,"Bb5","B5","C6","Db6","D6","Eb6","E6","F6","Gb6","G6","Ab6"];
162 end
163 if shamisenString == 3
164     frettingpos =
        [0,2,5,7,9,11,12,14,16,17,18,20,21,22,23,24,25,26,27,28,29];
                                %3rd string
165     noteName = ["C5","Db5","D5","Eb5","E5","F5","Gb5","G5","Ab5","A5
        ","Bb5","B5","C6","Db6","D6"...          %3rd string
166     ,"Eb6","E6","F6","Gb6","G6","Ab6"];
167 end
168 if shamisenString == 0
169     frettingpos = [0];
170     noteName = ["C4"];
171     shamisenString =1;
172 end
173
174 for i=1:length(frettingpos)
175 %% Intialise states of the system
176
177 % Strings
178 noteName(i)
179 frettingpos(i)
180 uS1Next = zeros(NS1,1);
181 uS1 = zeros(NS1,1);
182 if shamisenString == 1
183     width = round(NS1/10);
184     excitationRange = 1:width;
185     uS1(excitationRange + floor((NS1*5)/(pi*6))) = hann(width);
186     lS1 = 3+(0):NS1-2-(frettingpos(i));
187     lS2 = 3+(0):NS2-2-(0);
188     lS3 = 3+(0):NS3-2-(0);
189 end
190 uS1Prev = uS1;
191 uS2Next = zeros(NS2,1);
192 uS2 = zeros(NS2,1);
193 if shamisenString == 2
194     width = round(NS2/10);
195     excitationRange = 1:width;
```

```
196     uS2(excitationRange + floor((NS2*5)/(pi*6))) = hann(width);
197     lS1 = 3+(0):NS1-2-(0);
198     lS2 = 3+(0):NS2-2-(frettingpos(i));
199     lS3 = 3+(0):NS3-2-(0);
200 end
201 uS2Prev = uS2;
202 uS3Next = zeros(NS3,1);
203 uS3 = zeros(NS3,1);
204 if shamisenString == 3
205     width = round(NS3/10);
206     excitationRange = 1:width;
207     uS3(excitationRange + floor((NS3*5)/(pi*6))) = hann(width);
208     lS1 = 3+(0):NS1-2-(0);
209     lS2 = 3+(0):NS2-2-(0);
210     lS3 = 3+(0):NS3-2-(frettingpos(i));
211 end
212 uS3Prev = uS3;
213
214
215 % Plate
216 uPNext = zeros(Nx,Ny);
217 uP = zeros(Nx,Ny);
218 uPPrev = zeros(Nx,Ny);
219 % Bar
220 uBNext = zeros(NB,1);
221 uB = zeros(NB,1);
222 uBPrev = zeros(NB,1);
223
224 % Output
225 outPosB = floor(2*NB/pi);
226 outPosS1 =floor((2*NS1)/(pi*7))+4;
227 outPosS2 =floor((2*NS2)/(pi*7))+4;
228 outPosS3 =floor((2*NS3)/(pi*7))+4;
229 out = zeros(dur,1);
230 %% Intialise l for update equations
231 lP = 3:Nx-2;
232 mP = 3:Ny-2;
233
234 lB = 3:NB-2;
235
236 %%
237 tic
238 for n = 1:dur
239     % Calculate virtual grid points
240     uB0 = 2*uB(1)-uB(2);                    % uB(0)
241     uBm1 = 2*(uB0-uB(2))+uB(3);             % uB(-1)
242     uBPrev0 = 2*uBPrev(1)-uBPrev(2);        % uBPrev(0)
243     uBNp1 = 2*uB(NB) - uB(NB-1);            % uB(N+1)
244     uBNp2 = 2*(uBNp1 - uB(NB-1)) + uB(NB-2);% uB(N+2)
245     uBPrevNp1 = 2*uBPrev(NB) - uBPrev(NB-1);% uBPrev(N+1)
246
```

```
247
248
249 %     ____UPDATE_EQUATIONS_____________________________________________________________________
250
251     %% Update equation for the Strings
252     uS1Next(lS1) = uS1(lS1)*uS1lMult + (uS1(lS1-1) + uS1(lS1+1))*
            uS1l1Mult + (uS1(lS1-2) + uS1(lS1+2))*uS1l2Mult + ...
253          uS1Prev(lS1)*uS1PrevlMult + (uS1Prev(lS1-1)+ uS1Prev(lS1+1))*
                 uS1Prevl1Mult;
254     uS2Next(lS2) = uS2(lS2)*uS2lMult + (uS2(lS2-1) + uS2(lS2+1))*
            uS2l1Mult + (uS2(lS2-2) + uS2(lS2+2))*uS2l2Mult + ...
255          uS2Prev(lS2)*uS2PrevlMult + (uS2Prev(lS2-1)+ uS2Prev(lS2+1))*
                 uS2Prevl1Mult;
256     uS3Next(lS3) = uS3(lS3)*uS3lMult + (uS3(lS3-1) + uS3(lS3+1))*
            uS3l1Mult + (uS3(lS3-2) + uS3(lS3+2))*uS3l2Mult + ...
257          uS3Prev(lS3)*uS3PrevlMult + (uS3Prev(lS3-1)+ uS3Prev(lS3+1))*
                 uS3Prevl1Mult;
258
259     %% Update equation of the Bar (bridge)
260     uBNext(lB) = uB(lB)*uBlMult + (uB(lB-1) + uB(lB+1))*uBl1Mult + (uB
            (lB-2) + uB(lB+2))*uBl2Mult + ...
261          uBPrev(lB)*uBPrevlMult + (uBPrev(lB-1)+ uBPrev(lB+1))*
                 uBPrevl1Mult;
262     % solve for uBNext at points 1, 2, N-1, N
263     uBNext(2) = uB(2)*uBlMult + (uB(2-1) + uB(2+1))*uBl1Mult + (uB0 +
            uB(2+2))*uBl2Mult + ...
264          uBPrev(2)*uBPrevlMult + (uBPrev(2-1) + uBPrev(2+1))*
                 uBPrevl1Mult;
265     uBNext(1) = uB(1)*uBlMult + (uB0 + uB(1+1))*uBl1Mult + (uBm1 + uB
            (1+2))*uBl2Mult + ...
266          uBPrev(1)*uBPrevlMult + (uBPrev0+ uBPrev(1+1))*uBPrevl1Mult;
267     uBNext(NB-1) = uB(NB-1)*uBlMult + (uB(NB-1-1) + uB(NB-1+1))*
            uBl1Mult + (uB(NB-1-2) + uBNp1)*uBl2Mult + ...
268          uBPrev(NB-1)*uBPrevlMult + (uBPrev(NB-1-1) + uBPrev(NB-1+1))*
                 uBPrevl1Mult;
269     uBNext(NB) = uB(NB)*uBlMult + (uB(NB-1) + uBNp1)*uBl1Mult + (uB(NB
            -2) + uBNp2)*uBl2Mult + ...
270          uBPrev(NB)*uBPrevlMult + (uBPrev(NB-1) + uBPrevNp1)*
                 uBPrevl1Mult;
271
272
273     %% Update equation of the Plate
274     uPNext(lP,mP) = uP(lP,mP)*uPlMult + (uP(lP-1,mP) + uP(lP+1,mP) +
            uP(lP,mP+1) + uP(lP,mP-1))*uPl1Mult + ...
275          (uP(lP-1,mP-1) + uP(lP+1,mP-1) + uP(lP-1,mP+1) + uP(lP+1,mP+1)
                 )*uPl1dMult + ...
276          (uP(lP-2,mP) + uP(lP+2,mP) + uP(lP,mP-2) + uP(lP,mP+2))*
                 uPl2Mult + ...
277          uPPrev(lP,mP)*uPPrevlMult + (uPPrev(lP+1,mP) + uPPrev(lP-1,mP)
```

```
            + ...
278         uPPrev(lP,mP+1) + uPPrev(lP,mP-1))*uPPrevl1Mult;
279
280 %% Calculate the forces
281     % Force from the Strings to the bridge
282     Fs1b = Fs1bMult * (-uBNext(lBc1) + uS1Next(lS1c));
283
284     Fs2b = Fs2bMult * (-uBNext(lBc2) + uS2Next(lS2c));
285
286     Fs3b = Fs3bMult * (-uBNext(lBc3) + uS3Next(lS3c));
287
288     % Force from bridge' left and right mounting points to the plate
289     Fbpl = FbpMult*(-uBNext(lBcl) + uPNext(lPcl, mPcl));
290
291     Fbpr = FbpMult*(-uBNext(lBcr) + uPNext(lPcr, mPcr));
292
293
294 %% Update equations at localizer points
295     % Update Strings equation at the localizer point with Fsm (Force
            loss to the bridge)
296     uS1Next(lS1c) = uS1Next(lS1c) - Fs1b/(rhoS * AreaS1 *hS1 * (
            sigmaS0 + 1));
297     uS2Next(lS2c) = uS2Next(lS2c) - Fs2b/(rhoS * AreaS2 *hS2 * (
            sigmaS0 + 1));
298     uS3Next(lS3c) = uS3Next(lS3c) - Fs3b/(rhoS * AreaS3 *hS3 * (
            sigmaS0 + 1));
299
300
301     % Update Bar function at the localizer point with Fs1b (Force gain
             from the string 1)
302     uBNext(lBc1) = uBNext(lBc1) + Fs1b/(rhoB * AreaB *hB * (sigmaB0 +
            1));
303     % Update Bar function at the localizer point with Fs2b (Force gain
             from the string 2)
304     uBNext(lBc2) = uBNext(lBc2) + Fs2b/(rhoB * AreaB *hB * (sigmaB0 +
            1));
305     % Update Bar function at the localizer point with Fs3b (Force gain
             from the string 3)
306     uBNext(lBc3) = uBNext(lBc3) + Fs3b/(rhoB * AreaB *hB * (sigmaB0 +
            1));
307     % Update Bar function at the localizer point lBl with Fbp (Force
            gain from the string 1)
308     uBNext(lBcl) = uBNext(lBcl) - Fbpl/(rhoB * AreaB *hB * (sigmaB0 +
            1));
309     % Update Bar function at the localizer point lBr with Fbp (Force
            gain from the string 1)
310     uBNext(lBcr) = uBNext(lBcr) - Fbpr/(rhoB * AreaB *hB * (sigmaB0 +
            1));
311
312
313     % Update Plate function at the localizer points with Fbp (Force
```

```
          gain from the bridge)
314       uPNext(lPcl,mPcl) = uPNext(lPcl,mPcl) + Fbpl/(rhoP*HP*hP^2*(
              sigmaP0 + 1));
315       uPNext(lPcr,mPcr) = uPNext(lPcr,mPcr) + Fbpr/(rhoP*HP*hP^2*(
              sigmaP0 + 1));
316    %% plot
317    %     variable = uPNext;
318    %     subplot(9,1,1);
319    %     plot(uS1Next);
320    %     subplot(9,1,2);
321    %     plot(uS2Next);
322    %     subplot(9,1,3);
323    %     plot(uS3Next);
324    %     subplot(6,1,3);
325    %     plot(uBNext);
326    %     subplot(3,1,3);
327    %     mesh(variable);
328    % %     imagesc(variable);
329    %     drawnow;
330    %
331       %% Output
332    %     outP(n) = uPNext(outPosP(1),outPosP(2));                    %
          plate
333       outP(n) = sum(uPNext, 'all');
334       outB(n) = uBNext(outPosB);                                        %
              bridge
335       outS(n) = uS1Next(outPosS1) + uS2Next(outPosS2) +uS3Next(outPosS2)
              ;  % string
336
337       %% Update the state variables
338       uS1Prev  = uS1;
339       uS1 = uS1Next;
340    %     uS2Prev  = uS2;
341    %     uS2 = uS2Next;
342    %     uS3Prev  = uS3;
343    %     uS3 = uS3Next;
344       uPPrev  = uP;
345       uP = uPNext;
346       uBPrev  = uB;
347       uB = uBNext;
348    %     if n == floor(dur/3)
349    %        uexS2 = zeros(NS2,1);
350    %        uexS2((1:floor(NS2/10)) + floor(NS2/5)) = hann(floor(NS2/10)
          );
351    %        uS2((1:floor(NS2/10)) + floor(NS2/5)) = uS2((1:floor(NS2/10)
          ) + floor(NS2/5)) + uexS2((1:floor(NS2/10)) + floor(NS2/5));
352    % %        uP(ceil(Nx/2-Nx/8):floor(Nx/2+Nx/8),ceil(Ny/2-Ny/8):floor(
          Ny/2+Ny/8)) = ...
353    % %            uP(ceil(Nx/2-Nx/8):floor(Nx/2+Nx/8),ceil(Ny/2-Ny/8):
          floor(Ny/2+Ny/8)) + ... here plate is excited by velocity
354    % %            k*vP0*hamming_3d(length(ceil(Nx/2-Nx/8):floor(Nx/2+Nx
```

```
    /8)),length(ceil(Ny/2-Ny/8):floor(Ny/2+Ny/8)),1);
%       end
%       if n ==floor(dur*2/3)
%          uexS3 = zeros(NS3,1);
%          uexS3((1:floor(NS3/10)) + floor(NS3/5)) = hann(floor(NS3/10)
    );
%          uS3((1:floor(NS3/10)) + floor(NS3/5)) = uS3((1:floor(NS3/10)
    ) + floor(NS3/5)) + uexS3((1:floor(NS3/10)) + floor(NS3/5));
%       end
     uS2Prev  = uS2;
     uS2 = uS2Next;
     uS3Prev  = uS3;
     uS3 = uS3Next;

end
toc
out = (2737*outP)+outB+outS;
write(shamisenString,out,Nx,Ny,noteName(i),i);
end

plot(out);
```

Listing 7.2: The function used write samples of the Shamisen

```
function write(shamisenString, out,Nx,Ny,note,i)
M = max(abs(out));
outN = out/M;
filename = strcat("C:\University\SMC\SMC10\multiString\
    MultiStringFDS_withRealTimeImplementation\audio\samples\",string(i)
    ,"#","string",string(shamisenString),note,"_Grid",string(Nx),"x",
    string(Ny),".wav");
audiowrite(filename, outN, 44100);
end
```