A FDTD Model of a 2D Kirchhoff Thin Plate Model and Coupled Strings

Matthew Hamilton

A final project dissertation submitted in partial fulfilment
of the requirements for the degree of

Master of Science (MSc)

Acoustics and Music Technology

Acoustics and Audio Group

Edinburgh College of Art
University *of* Edinburgh

August 2017

Supervisor 1: Dr Stefan Bilbao

Abstract

An exploration of finite difference time domain implementations of the Kirschhoff Thin Plate coupled with string models under different conditions. Primarily focussing on a matrix operation based approach, particularly in the MATLAB programming environment. Also focuses on methods and strategies to optimise performance, improve understanding of the core concepts and identifying common problems that may be encountered.

Declaration

I do hereby declare that this dissertation was composed by myself and that the work described within is my own, except where explicitly stated otherwise.

Matthew Hamilton
August 2017

Acknowledgements

For my wife Helen, without whose help and support I would not have achieved any of the following.

Contents

Abstract i

Declaration iii

Acknowledgements v

Contents vii

List of figures xi

1 Introduction 1
- 1.1 FDTD Theory . . . 2
- 1.2 Notation . . . 3
- 1.3 FDTD Matrices . . . 3

2 Kirschoff Thin Plate 7
- 2.1 Thin Plate Equation . . . 7
- 2.2 Thin Plate Finite Difference Model . . . 8
  - 2.2.1 Boundary Conditions . . . 9
  - 2.2.2 Constructing the Biharmonic . . . 10
  - 2.2.3 Energy Analysis . . . 12
  - 2.2.4 Unit Checking . . . 13
- 2.3 Loss . . . 14
- 2.4 Applying Force . . . 15
  - 2.4.1 Modal Analysis . . . 16

3 Coupling Plate and Strings 21
- 3.1 1D Wave and Plate . . . 22
  - 3.1.1 Coupling conditions . . . 22
  - 3.1.2 1D Wave /w Loss . . . 25
  - 3.1.3 1D Wave /w Force . . . 26
  - 3.1.4 Joining vectors . . . 27
  - 3.1.5 Interpolation . . . 28
- 3.2 Multiple Strings . . . 29
- 3.3 The Stiff String . . . 30
  - 3.3.1 Coupling with Stiff String . . . 31
  - 3.3.2 Frequency Dependant Loss . . . 32
- 3.4 Scheme Output . . . 33

4 Extensions to the FD Plate and Coupled String **35**
4.1 Optimisation . . . 35
4.2 Calibration . . . 36
4.3 Reverberation . . . 37
4.4 Further additions . . . 38
4.5 Reflections . . . 38

A Full Derivation of Formulas **39**
A.1 The Thin Plate . . . 39
A.1.1 Lossless Thin Plate . . . 39
A.1.2 Thin Plate w/ Generic Loss . . . 39
A.1.3 Thin Plate w/ Frequency Dependant Loss . . . 40
A.1.4 Energy: Continuous . . . 41
A.1.5 Energy: Discrete . . . 41
A.1.6 Modal Analysis: Simply Supported . . . 42
A.1.7 Lossless w/ Force . . . 42
A.1.8 Frequency Dependant Loss w/ Force . . . 43
A.2 Coupling String and Plate . . . 43
A.2.1 FDTD 1D Wave Lossless . . . 43
A.2.2 FDTD 1D Wave Generic Loss . . . 44
A.2.3 Coupling Conditions: Energy Analysis . . . 44
A.2.4 Plate Centre Difference . . . 44
A.2.5 1D Wave Centre Difference . . . 45
A.2.6 1D Wave/Plate Coupling Force . . . 45
A.2.7 Thin Plate w/ Coupling Force . . . 45
A.2.8 1D Wave FD Scheme w/ Coupling Force . . . 46
A.2.9 1D Wave/Plate Coupling Force . . . 46
A.2.10 Stiff String FD Scheme . . . 46
A.2.11 Stiff String and Plate w/ Frequency Dependent Loss Coupling Force . . . 47

B MATLAB Code **49**
B.1 Building the Laplacian and Bi-Harmonic . . . 49
B.1.1 biharm.m . . . 50
B.1.2 laplace.m . . . 51
B.1.3 fidimat.m . . . 52
B.2 The Kirschoff Thin Plate . . . 57
B.3 The Coupled Plate and String . . . 60

C Project Archive **67**
C.1 Audio . . . 67
C.2 File Structure . . . 67
C.3 Naming Convention . . . 68
C.4 Force . . . 68
C.5 Stability . . . 69
C.6 Calibration . . . 69
C.7 File Breakdown . . . 69
C.8 File List . . . 72

Bibliography **73**

List of Figures

1.1 The transformation of a 2D grid to a linear indexed vector. The grid co-ordinates are aligned with their corresponding linear index value. N_y is the number of grid points in the y-axis. 4

2.1 Laplacian (D_Δ) and Biharmonic ($D_{\Delta\Delta}$) Matrix Patterns 11
2.2 Biharmonic Stencil With linear index offsets for x and y axis. P varies with number and type of boundaries it is adjacent to. N_y represents the number of grid points in the y-axis 11
2.3 Motion of a thin plate FD model with frequency dependant loss under clamped conditions and initial conditions of a raised cosine.) 12
2.4 Rounding Error Pattern in change in energy($\frac{dE}{dt}$) 13
2.5 The dirac delta in a) continuous space and b) discrete space. 15
2.6 Force function f(n) as a raised cosine 16
2.7 Simply Supported Modal Frequencies at a) no oversampling and b) 8 times oversampling 17
2.8 Clamped Modal Frequencies 18
2.9 Thin Plate FD Modal Frequency Response and Approximated Modal Frequencies under clamped conditions 18
2.10 $f_{3,3}$ Modal frequency 19
2.11 Modal frequency patterns under simply supported boundary conditions [1, Figure 11.3] 19

3.1 The string connected to the plate. The red line illustrates the point of connection between the two systems 25
3.2 Force Vector a) overwritten b) summation: What should be an imperceptible double strike (b) can result in an unpleasant pluck (a). 27
3.3 The Matrix Pattern of a joined plate and string system. Coupled points have been circled and the sections demarcated to reflect which system they refer. 28
3.4 The cross represents the coupling point lieing between grid points. The normalised distance (or percentage) that the coupling point lies towards the next grid point on the x and y axis is represented by α_x and α_y respectively. 29

4.1 Overwound Piano String 36
4.2 The Palme Diffuseur for the Ondes Martenot [2] 37

Chapter 1

Introduction

With ever-increasing computer processing power and its availability to a home user, the feasibility of implementing more involved and processor-intensive computation increases. With respect to computer music, the potential for realising new and interesting ideas is far greater than it was even 10 years ago. This means the ability to create much more complex and intricate ways of processing and interacting with sound. One such way of generating sound is through physical modelling synthesis. In the past, physical modelling has been less feasible as the methods involved tended to be computationally intensive. Now these same methods are not only viable but they are ever more capable of being achieved in real-time.

Physical-modelling in this sense is the description of the movement of some physical system to a digital domain. The equations of motion which are used to predict the behaviour of musical systems, such as membranes and plates strings &c.. are used to generate sound. The benefits are that the sounds tend to be richer than other modes of sound synthesis as well as the fact that they are more closely informed by the behaviour of real instruments. One of the ways of physical modelling of musical systems is through use of finite difference time domain method (FDTD). FDTD is a method of translating the differential operators of partial difference equations (PDEs) used in equations of motion. As such, they are a little more tricky to get to grips with than traditional methods of sound synthesis. Should one wish to begin creating FDTD systems to make instruments or audio effects it can help to understand some of the underlying theory involved in FDTD.

The purpose of this paper is to explore the means for creating more complex FDTD instruments by coupling, in particular coupling a plate model with string models. This will allow the creation of chordophone-like instruments that, depending on the configuration, resemble either a piano, a harp or a guitar.

This chapter will deal with basics of FTDT, the operations involved and some methods for beginning to tackle this kind of problem. The second chapter will tackle the thin plate and and its respective FTDT model. It will also discuss making the

model more complex, what this involves and how to go about deriving the correct FDTD scheme. Chapter 3 will concern coupling the FDTD plate model with, firstly, a 1D wave and then a stiff string model. Finally Chapter 4 concerns the additional measure that can be taken to produce more interesting sounds as well other ways of interacting with the model and extensions that can be pursued beyond what is covered in this paper.

If unacquainted with the type of algebra contained within this paper, explicit derivations have been provided in Appendix A and are referenced throughout. Appendix B provides some example function and FDTD schemes which should aid in simplifying the coding process.

1.1 FDTD Theory

First, it is worthwhile giving an overview of FDTD operators and how they work. FDTD operators represent an approximation of differential operators in continuous time and space. This centres around a difference between two points in space or time divided by the meaasured distance between. Since FDTD is in the time domain the goal is to find an unknown displacement that is forward in time.

$$\begin{aligned} \delta_{x+} &\triangleq \frac{1}{h}(e_{x+} - 1) & \delta_{t+} &\triangleq \frac{1}{k}(e_{t+} - 1) \\ \delta_{x-} &\triangleq \frac{1}{h}(1 - e_{x-}) & \delta_{t-} &\triangleq \frac{1}{k}(1 - e_{t-}) \\ \delta_{x\cdot} &\triangleq \frac{1}{2h}(e_{x+} - e_{x-}) & \delta_{t\cdot} &\triangleq \frac{1}{2k}(e_{t+} - e_{t-}) \end{aligned}$$

Operators behave the same way in space as they do in time and involve the difference between two points which are shifted and divided by the spacing that separates them. In terms of differential term $\frac{dy}{dt}$ the $\frac{1}{dt}$ can be considered the continuous form of 1/k. For the time operators, δ_t, the variable k is the period of time between samples (i.e. 1/sampling rate). The smaller k is, the more accurate a representation it can be considered of the continuous case. Operators can be combined to create approximations of higher order difference.

$$\begin{aligned} \delta_{xx} &= \frac{1}{h}(\delta_{x+} - \delta_{x-}) = \frac{2}{h}(\delta_{x\cdot} - \delta_{x-}) \\ \delta_{xxxx} &= (\delta_{xx})(\delta_{xx}) \end{aligned}$$

It is hopefully apparent how each operator functions; if not, [1] is an excellent source, especially with its perspective of FDTD methods for musical applications. More complex operators such as the laplacian, Δ and the biharmonic $\Delta\Delta$ will be discussed later. It will hopefully show that spatial difference can be mostly automated and temporal difference will be restricted to concerning three time steps.

1.2 Notation

The notation for this paper will be in terms of dot and primes for temporal and spatial derivatives respectively. Given that the plate is two-dimenional, there is little overlap in the notation between systems. This should hopefully assist in minimising any confusion.

For a 2D FDTD scheme, of greater concern are the laplacian and biharmonic operators.

$$\Delta = \left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right], \quad \Delta\Delta = \left[\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial^2 x \partial^2 y} + \frac{\partial^4}{\partial y^4}\right] = \left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right]^2 \tag{1.1}$$

The laplacian is a partial second order term in each of the spatial dimensions and the biharmonic is simply the laplacian squared. The implications this has on a FDTD scheme will be discussed later. It is worth familiarising oneself with the terms in equation 1.1. If new to these terms, the notation can do more to obfuscate the meaning than make it clearer. In other texts ∇^2 may be used instead of Δ, the latter will be used in this paper.

The FDTD systems that will be considered are the plate and the string. These will always be represented by 'u' for the plate and 'w' for the string. These terms both represent a vector of displacements/amplitudes. The finer details will be discussed later. The superscript for these terms refers to time step, 'n' and the subscript the spatial index. For instance, u_{n+1} refers to the state of the system one sample ahead of the current time step. Given the approach that will be taken to the discussed schemes, the spatial index will rarely be used as systems for the plate and string will contained in the vector form.

1.3 FDTD Matrices

As had been previously stated, the end goal for a FDTD is to compute an unknown value, forward in time, in terms of previously computed values. In the confines of this paper there are three time steps that are considered. The forward step that needs computed, u^{n+1}, the current time step u^n and previous step u^{n-1}. The FDTD will be organised in vectors with linear indexing. Even though it might seem the obvious first choice for a 2D model to be represented as a grid, a vector is more readily operated on by matrices. The organisation of linear indexing for the plate is given in Figure 1.1. The spatial differences can be contained within $N \times N$ matrices, where N is the number of grid points in the system. The update at each time step to find the next state will always be in the form

$$Au^{n+1} = Bu^n + Cu^{n-1} \tag{1.2}$$

In equation 1.2, A, B and C refer to coefficient matrices consiting of the spatial FDTD operations that relate to each time step. Building these matrices will be the main task for creating the schemes discussed here. The benefits of sticking to this form are that very little change should actually be required when altering and adding greater complexity to the schemes, which will hopefully be conveyed throughout the remainder of the paper.

The same thinking can be applied to spatial difference operators. As an example, take the definition of the forward difference operator for the 1D wave $\delta_{\eta+}$.

$$\delta_{\eta+} w = \frac{1}{h^2}(u_{\eta+1} - u_{\eta}) \tag{1.3}$$

Carrying out this operation across the whole domain of w can be represented as the following matrix operation:

$$\frac{1}{h^2} D_{\eta+} \begin{bmatrix} w_1 \\ \vdots \\ w_{N-1} \\ w_N \end{bmatrix} = \frac{1}{h^2} \begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_{N-1} \\ w_N \end{bmatrix} \tag{1.4}$$

Building matrices in this manner simplifies the problem of higher order FD spatial operators. As another example for 1D wave, $\delta_{\eta\eta} = \frac{1}{h}(\delta_{\eta+} - \delta_{\eta-})$

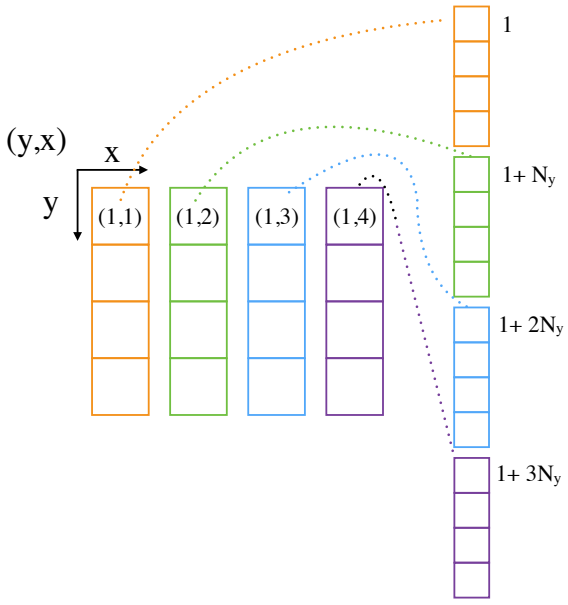

Figure 1.1: The transformation of a 2D grid to a linear indexed vector. The grid coordinates are aligned with their corresponding linear index value. N_y is the number of grid points in the y-axis.

$$
\begin{aligned}
\delta_{\eta\eta} &= \frac{1}{h^2}(D_{\eta+} - D_{\eta-}) \\
&= \frac{1}{h^2}\begin{bmatrix} -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{bmatrix}\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \\
&= \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \end{bmatrix} = \frac{1}{h^2}D_{\eta\eta}
\end{aligned}
$$

This approach turns the construction into a trivial problem that requires only a simple matrix multiplied by the correct coefficient. Approaching the problem in this manner is a much more attractive method than the linear equivalent. This modular approach allows for much greater separation of the various elements in the scheme in comparison to coding it explicitly. The only extra consideration is the modifications to the matrix coefficients as a result of boundary conditions. This will be covered in section 2.2.1 though the functions in B.1 provide one approach to automating the solution. It is recommended to create your own FDTD matrix functions; this will be a valuable aid in streamlining any code and will provide a handy tool that can be reused. Though API and toolkits are readily available, coding by hand should bring to light any uncertainties or misconceptions. Functions also help to reduce any needless repetition and improve code readability.

Chapter 2

Kirschoff Thin Plate

The first stage of coupling two FD schemes together is having something to couple to. It is best to begin with the thin plate model as it is the more tricky of the two schemes. It should also help to clarify the processes involved in making an FDTD scheme in general. The plate model that will be considered is the Kirschhoff plate model. The model and accompanying finite difference scheme will be stated strictly in cartesian coordinates. It is of course possible to create a FDTD scheme in radial co-ordinates and in fact, this is probably the first step to creating a circular membrane or shell model such as for drums and cymbals. For more on radial coordinates and plates see [1, Section 12.6]. The derivation of the these equations of motion will not be discussed in any great detail in this paper. This has been covered succinctly elsewhere, [3] has been very helpful for getting to grips with the physical analysis. The FDTD model of the plate that will be dealt with here is that in [1] and by extension the physical model from [4].

2.1 Thin Plate Equation

The Kirschhoff plate model, equation 2.1, is in essence an extension of the 1D bar equation [3].

$$\rho H \ddot{u} = -D\Delta\Delta u, \quad D = \frac{EH^3}{12(1-\nu)} \tag{2.1}$$

On the left hand side the displacement of the plate in this case is u, denisty ρ, plate thickness H.On the right is the fourth order term, the biharmonic operating on u and the constant D containing the Young's Modulus of the material E and the poisson ratio ν. Bringing over the density and thickness gives us a more simplistic equation 2.2 with one constant κ which will be easier to implement in a FDTD scheme. It is advisable, when adding extra terms, to begin afresh from 2.1, especially when including a force term. It is easy to forget the inclusion of the ρH term and the scheme will be incorrect

without it.

$$\ddot{u} = -\kappa^2 \Delta\Delta u, \quad \kappa = \sqrt{\frac{EH^2}{12\rho(1-\nu)}} \tag{2.2}$$

In this model the plate is isotropic, which is to say that wave velocity is equal in all directions. Anisotropy is encountered in materials like wood and will be discussed in Chapter 4.

2.2 Thin Plate Finite Difference Model

A FDTD model for thin plate is given in 2.3. It is a direct translation of the operators in 2.1.

$$\rho H \delta_{tt} u = -D \delta_{\Delta\Delta} w \quad \rightarrow \quad \delta_{tt} u = -\kappa^2 \delta_{\Delta\Delta} w \tag{2.3}$$

Deriving an update that can be used in a scheme is shown in equation 2.4.

$$\begin{aligned}
\rho H \delta_{tt} u &= -D \delta_{\Delta\Delta} u \\
\delta_{tt} u &= -\kappa^2 \delta_{\Delta\Delta} u \\
\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) &= -\kappa^2 \delta_{\Delta\Delta} u \\
u^{n+1} &= -k^2\kappa^2 \delta_{\Delta\Delta} u + 2u^n - u^{n-1} \\
&= -\frac{k^2\kappa^2}{h^4} D_{\Delta\Delta} u + 2u^n - u^{n-1} \\
&= -\mu^2 D_{\Delta\Delta} u + 2u^n - u^{n-1}
\end{aligned} \tag{2.4}$$

The addition here is the scheme variable μ^2 which is defined as:-

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

Here, k is the time step and h is the grid spacing which is to the power 4, given that $\Delta\Delta$ consists of fourth order terms. To set the plate in motion, some initial conditions, in the form of a raised cosine, at some point on the plate can be used (see Appendix B.2 lines 98-102 and 118-119).

When beginning to solve for a FD update the benefits of the matrix approach become more apparent. Take for instance the update for the the biharmonic written out explicitly.

$$\begin{aligned}\delta_{\Delta\Delta} &= \frac{1}{h^4}(e_{x+1} - 2 + e_{x-1}e_{y+1} - 2 + e_{y-1})^2 \\ &= \frac{1}{h^4}(20 - 8(e_{x+} + e_{x-} + e_{y+} + e_{y-}) + ... \\ &\quad (e_{x+,y+} + e_{x+,y-} + e_{x-,y+} + e_{x-,y-}) + ... \\ &\quad (e_{x+2} + e_{x-2} + e_{y+2} + e_{y-2}))\end{aligned}$$

This term would need to be applied to every point on the plate, not having taken into account boundary conditions, which is a little unwieldy. In matrix form the aesthetics are a little less distressing. Moving forward in the form suggested in 1.2, 2.4 can be broken down into three matrix operations.

$$\underbrace{(I)}_{A}\, u^{n+1} = \underbrace{(-\mu 2 D_{\Delta\Delta} + 2I)}_{B} u^n - \underbrace{(I)}_{C}\, u^{n-1} \tag{2.5}$$

The three matrices, A, B and C operate on the vector u, that is the thin plate in this case. The matrices 'A' and 'C' may seem trivial in this form but, as greater complexity is added these become correspondingly more complex. It is worth noting at this point that the '$-$' sign in front of the C matrix can be included, $Au^{n+1} = Bu^n + Cu^{n-1}$ or kept outside $Au^{n+1} = Bu^n - Cu^{n-1}$. For this paper the first approach will be taken.

2.2.1 Boundary Conditions

This paper will primarily consider simply supported and clamped boundary conditions. In particular, clamped conditions, as these should closely relate to the behaviour of a soundboard of a string instrument as suggested in [5]. Since the plate equation is forth order in both the x-axis and y-axis, there are two conditions in each set. Both sets have a Dirichlet condition where the boundary points are fixed at zero. The Clamped set has a Neumann condition that the gradient is equal to zero. Simply Supported has a condition that the curvature is fixed at 0. In FDTD this can be represented as:

$$u_0 = u_N = 0 \tag{2.6a}$$

$$\delta_{x.} u_0 = \delta_{x.} u_N = 0 \tag{2.6b}$$

$$\delta_{xx} u_0 = \delta_{xx} u_N = 0 \tag{2.6c}$$

The implications of these conditions are not immediately apparent. In fact, in second order equations 2.6b and 2.6c are not required. In fourth order, however, we now have to take into account a 'ghost' point, u_{-1}, when considering locations neighbouring a boundary. As an example, take the following forth order operation on a point u_1

where the domain is between 0 and N

$$\delta_{xxxx} u_0 = \frac{1}{h^2}\left(u_3 - 4u_2 + 6u_1 - 4u_0 + u_{-1}\right) \tag{2.7}$$

The point u_{-1} is non-existent, but, using the other condition we can derive an expression in terms of known points.

$$u_{-1} = -u_1 \quad \text{[Clamped]} \tag{2.8a}$$

$$u_{-1} = u_1 \quad \text{[Simply Supported]} \tag{2.8b}$$

For derivation see Appendix A, Equation A.1.3

2.2.2 Constructing the Biharmonic

As illustrated in equation 1.1, the biharmonic is simply the laplacian operator squared. The same can be extended to the equivalent FDTD operators. There are a number of ways to approach this problem. Hard coding will impose greater limitations and not recommended although, it can be a useful learning exercise if the matrices are is kept small. The best means of constructing the biharmonic is procedurally.

$$\delta_{\Delta\Delta} = \delta_\Delta \delta_\Delta = \left(\delta_{xx} + \delta_{yy}\right)^2 \tag{2.9}$$

$$\frac{1}{h^4} D_{\Delta\Delta} = \left(\frac{1}{h^2} D_\Delta\right)\frac{1}{h^2} D_\Delta = \left(\frac{1}{h^2} D_{xx} + \frac{1}{h^2} D_{yy}\right)^2 \tag{2.10}$$

Constructing the biharmonic, beginning from the D_{xx} and D_{yy} matrices, allows for the easy inclusion of boundary conditions on the plate. For clamped and simply supported conditions, this is just a case of altering two points in the second order matrices.

$$D_{yy} = \frac{1}{h^2}\begin{bmatrix} -2 & b & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & b & -2 \end{bmatrix} \tag{2.11}$$

In 3.30, changing the constant b to 0 will implement simply supported conditions, and -2, clamped conditions. Both of these alterations are determined by 2.8a and 2.8b. Since the plate grid is indexed linearly, there can be technical and conceptual hurdles to overcome in constructing the D_{xx} matrix. In the MATLAB environment tools, such as the `kron()` function, prove very useful. In the case of the function in Appendix B.1.1 that creates the biharmonic, this is the approach that has been taken. Creating

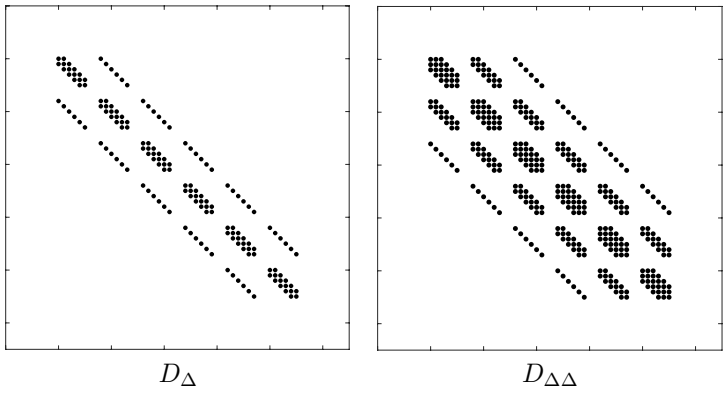

Figure 2.1: Laplacian (D_Δ) and Biharmonic ($D_{\Delta\Delta}$) Matrix Patterns

these matrices individually can also be useful for implementing any FDTD scheme (see Appendix B.1.3)

An excellent resource and further reading can be found in [6], which has some very helpful illustrations for beginning to understand the vectorisation of grids and the construction of the matrices for FDTD plates.

Figure 2.1 shows the pattern that should be obtained from a biharmonic and laplacian matrix. In the case of Figure 2.1 the boundary points that are fixed at 0 have been left. In the case of sparse matrix this should not add any extra strain

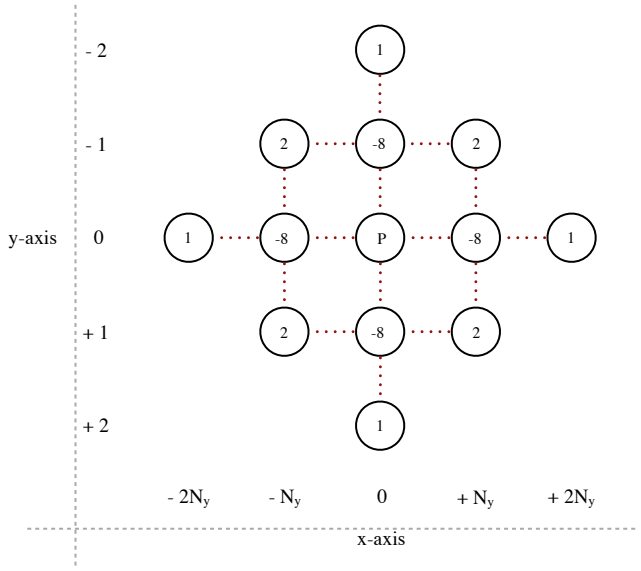

Figure 2.2: Biharmonic Stencil With linear index offsets for x and y axis. P varies with number and type of boundaries it is adjacent to. N_y represents the number of grid points in the y-axis

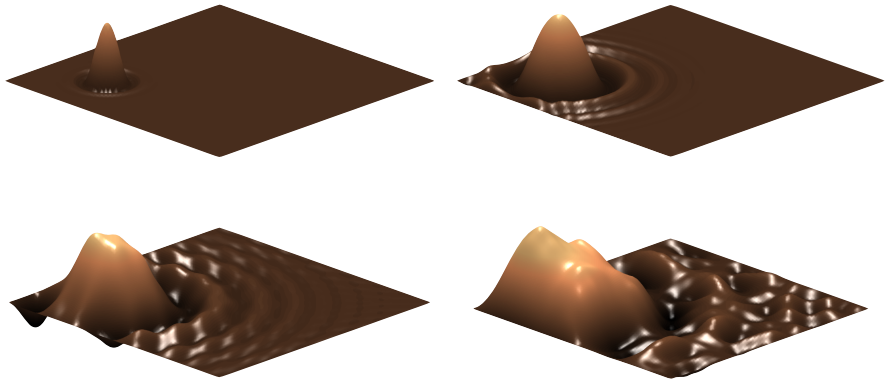

Figure 2.3: Motion of a thin plate FD model with frequency dependant loss under clamped conditions and initial conditions of a raised cosine.)

to computation, but it would be advisable to remove these calculations in other implementations. Using linear indexing, the outer diagonals are the coefficients for differences in the x-axis, while the y-axis differences are those points adjacent to the diagonals.

Figure 2.2 illustrates the stencil for the biharmonic matrix. The value of the central point P depends on the number and type of boundaries that are adjacent to it. All other points either keep the same value or are zero, depending on grid point index. Correctly zeroing out these points is given in Appendix B.1.1 lines 29-30.

For linear indexing the correct index for each point needs to be modified by the given values on the x and y axis. For simply supported conditions P is 20 for internal grid points, 19 when adjacent to a single boundary and 18 when in a corner. The pattern for these points in clamped conditions is 20, 21 and 22 respectively. There is not too much effort to be had in mixing boundary conditions but for the purposes of this paper it has be restricted to being unanimous across all boundaries. The alteration in 3.30 should hopefully show that enforcing arbitrary conditions for each boundary is fairly trivial.

2.2.3 Energy Analysis

Energy Analysis is a good means of sanity checking and proof that a FD model is showing stable behaviour. This can be very help full when fault-finding during coding, particularly with the addition of greater complexity. For example, should a model be stable when lossless and unstable when loss conditions are added, it is likely that the instability stems from the implementation of loss conditions.

Deriving energy follows the form of multiplying by the velocity and integrating over the domain. This will provide the kinetic and potential energy terms along with the boundary conditions $\mathcal{B}$ that need to be satisfied. In a lossless system, the change in

energy $\frac{dE}{dt}$ should equal 0. For the thin plate model, the change in energy is defined in equation 2.12. The total energy E has been marked. The derivation for this can be found in Appendix A A.1.4

$$\frac{\partial}{\partial t}\underbrace{\left[\frac{1}{2}\|\dot{u}\|_{\mathcal{D}}^2+\frac{\kappa^2}{2}\|\Delta u\|_{\mathcal{D}}^2\right]}_{E}=0 \tag{2.12}$$

For the discrete case we sum over the domain as there are a limited number of points on the grid. In this case, there is no longer an infinitesimally small distance being integrated over but rather a measurable quantity in the form of the grid spacing. The energy in the discrete case is given by equation 2.13 (also see Appendix A.1.5). In this instance the grid spacing is for x and y are the same (i.e. $h_x = h_y = h$) There is a slight alteration that needs to made when deriving this in discrete time. The change stems from summation by parts(see [1, Section 5.2.10]).

$$\delta_{t+}\left[\frac{h^2}{2k^2}(u^n-u^{n-1})^T(u^n-u^{n-1})+\frac{\kappa^2}{2h^2}(D_\Delta u^n)^T(D_\Delta u^{n-1})\right]=0 \tag{2.13}$$

When implementing this in a FDTD scheme, only a very short time period need be considered, which should save computation. The result should be something similar to that found in Figure 2.4, the scattering pattern is created by rounding error as the difference in values is near machine epsilon.

2.2.4 Unit Checking

As the FDTD schemes increase in complexity, especially when coupling two systems, it is easy to get lost in a jumble of coefficients. For the most part, it tends to get uglier

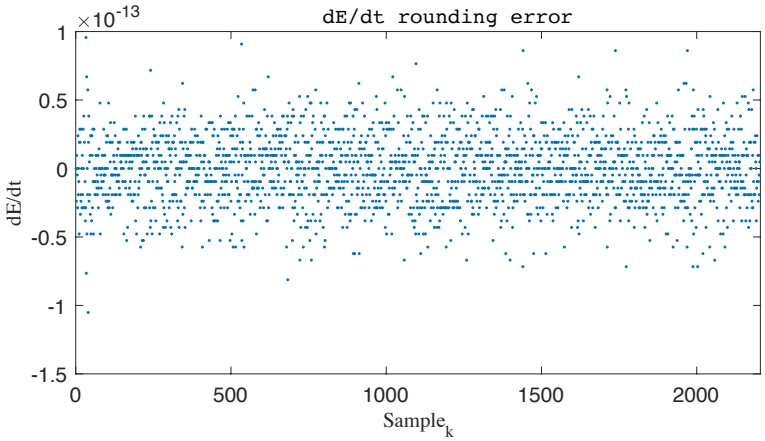

Figure 2.4: Rounding Error Pattern in change in energy($\frac{dE}{dt}$)

before it gets any better. Unit checking is always a good sanity check for any fault finding. No units in these schemes are scaled and the units of each term should be equal.

$$\rho H \frac{\partial^2}{\partial t^2}[u] = -D\Delta\Delta u, \quad D = \frac{EH^3}{12(1-\nu)}$$

Considering just the units on the LHS

$$\rho = kgm^{-3}, \quad H = m, \quad \frac{\partial^2}{\partial t^2} = s^{-2}, \quad w = m$$

All terms on the LHS combine into a single term in units of $kgm^{-1}s^{-2}$. So long as the RHS reflects the same units then we know that we can proceed without issue. For the term D, the poisson ratio is without units. We only consider the Young's modulus and the height cubed. The displacement u in m and the height H in m^3 cancel out the biharmonic which is units of m^{-4}. All that is left is E which is units of $kgm^{-1}s^{-2}$ which matches the units on the LHS. This is a very useful tool for sanity checking, especially when force is introduced, as the equations can start to become very cluttered.

2.3 Loss

So far, the system does not have very much musical function as it will continue moving forever. By adding loss, a decay in the energy will be introduced, creating a sound of limited duration. In the first instance a generic loss term can be added to 2.1, gives equation 2.14.

$$\begin{aligned} \rho H \ddot{u} &= -D\Delta\Delta u - 2\rho H \sigma_0 \dot{u} \\ \ddot{u} &= -\kappa^2 \Delta\Delta u - 2\sigma_0 \dot{u} \end{aligned} \tag{2.14}$$

Below is a FDTD scheme and its corresponding update for 2.14 (see Appenix A.1.2).

$$\begin{aligned} \rho H \delta_{tt} u &= -D\delta_{\Delta\Delta} u - 2\rho H \sigma_0 \delta_{t\cdot} u \\ \underbrace{(1 + k^2\sigma_0)}_{A} u^{n+1} &= \underbrace{(-\mu^2 D_{\Delta\Delta} + 2I)}_{B} u^n - \underbrace{(1 - k^2\sigma_0)}_{C} u^{n-1} \end{aligned} \tag{2.15}$$

The A and C matrices are no longer empty. These matrices will primarily deal with loss terms in the following FDTD schemes. Since A is a diagonal matrix, its values can be precomputed and integrated into the B and C matrices.

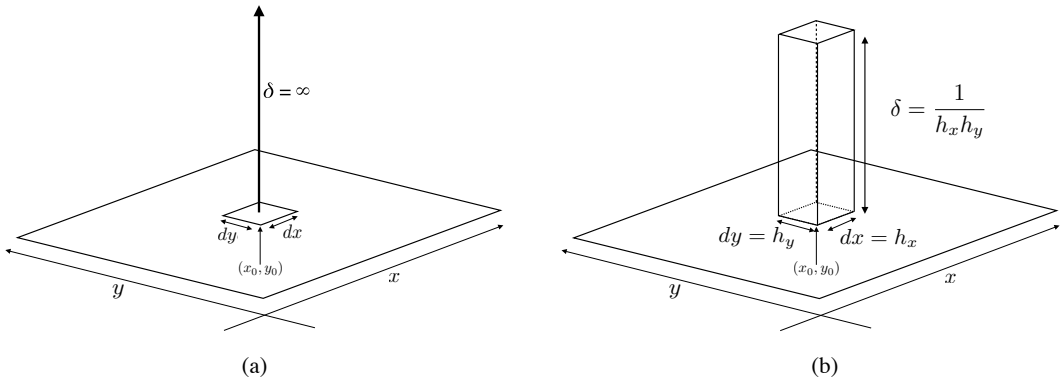

Figure 2.5: The dirac delta in a) continuous space and b) discrete space.

2.4 Applying Force

Rather than applying an initial displacement, a force can be added to the plate allowing for multiple excitations. This allows for interaction with the model and the ability to start using it like a musical instrument.

$$\rho H \ddot{u} = -D\Delta\Delta u - 2\rho H \sigma_0 \dot{u} + \delta(x - x_0, y - y_0) f(t) \tag{2.16}$$

In 2.16 the force is applied with a dirac delta. The dirac delta is defined as having a magnitude of infinity at an infinitesimally small point (x_0, y_0). The dirac spreads the scalar force at a time t in the function $f(t)$ to the appropriate point on the plate. The condition for the dirac is given below:

$$\delta(x - x_0, y - y_0) = \begin{cases} \infty, & dS = 0 \\ 0, & dS \neq 0 \end{cases}, \quad \int_D \delta(x - x_0, y - y_0) = 1 \tag{2.17}$$

In discrete space, there is no longer an infinitesimally small point to be considered. There is now a minimum grid spacing in the form of h. In order for the dirac to have the same behaviour, its magnitude needs to be changed to $\frac{1}{h^2}$. This is shown in Figure 2.5, where if $h_x = h_y = h$ then $h_x h_y = h^2$. In the FD scheme this is implemented with a spreading function J. For the time being in this case J is a flooring function 2.18. Truncating the force co-ordinates to the nearest grid point on the plate.

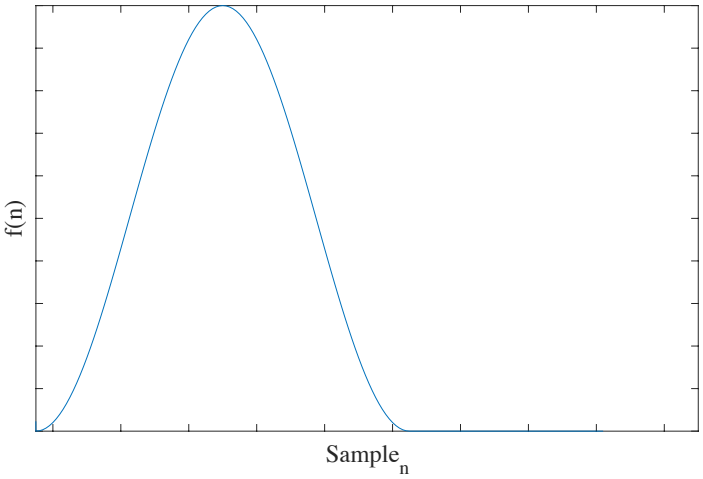

Figure 2.6: Force function f(n) as a raised cosine

$$J_0 = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad (x_0, y_0) \tag{2.18}$$

J can also be an interpolation function of some variety but this will be discussed further in section 3.1.5. For clarity the $1/h^2$ is kept outside of J however, it can be included, along with other coefficients when coded (see Appendix lines 201-251).

$$\delta_{tt} u = -\kappa^2 \delta_{\Delta\Delta} u^n - 2\sigma_0 \delta_{t.} u + \frac{1}{\rho H h^2} J f(n)$$

The force is now some discrete function of sample index n. A rough modelling of a mallet would be a raised cosine over a period of time (Figure 2.6). The force does not need to simply be a simple function, it can be any stream of numbers. Exchanging the raised cosine for a `.wav` file will produce a plate reverb effect. Choosing two readout points on the plate will produce a stereo plate reverb which can then be mixed with the original audio. The plate model as instrument can then be repurposed as a digital audio effect

2.4.1 Modal Analysis

A second sanity check to ensure that the plate scheme is adhering to the behaviour of the physical model is through modal analysis. If the modes of vibration of FD model

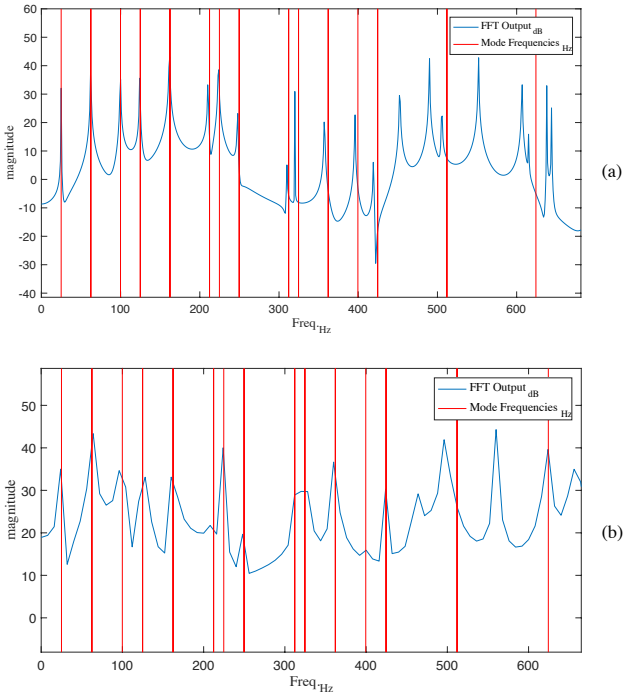

Figure 2.7: Simply Supported Modal Frequencies at a) no oversampling and b) 8 times oversampling

reflect the predicted modes we can safely assume there are no inherent errors in the scheme. Under simply supported conditions, prediction of the modes of vibration is reasonably simplistic.

Firstly an ansatz of the plate is predicted as $u(x, y, t) = e^{ik_x x} e^{ik_y y} e^{i\omega t}$ which is is separable to $u(x, y, t) = T(t)X(x)Y(y)$. Since the boundary points can only ever be equal to zero and the movement of the plate is to be oscillatory, it can be assumed that behaviour across x and y can be represented as sine waves.

$$f_{p,q} = \frac{\pi\kappa^2}{2}\left(\frac{p^2}{L_x} + \frac{q^2}{L_y}\right) \tag{2.19}$$

In 2.19, f is a frequency in Hz and p, q is the mode number (see Appendix A.1.6). To check that the scheme adheres to these mode numbers, the frequency output can be compared to the pre-calculated modes. Figure 2.7 shows the frequency response of the scheme under two sampling rates. What we find is a close match in lower frequencies. As we go up the frequency spectrum, there is a drift between the solution and scheme output as a result of numerical dispersion. This is rectified partially with increasing the sample rate, Figure 2.7 (b), though computational expense is increased greatly. Adjusting the grid spacing to take into account the oversampling will undo any benefits from oversampling in the first place. If using the plate model for musical

TABLE 4.24.—*Approximate Frequencies for a C–C–C–C Square Plate*

m	n	$\omega a^2\sqrt{\rho/D}$
1	1	35. 10
2	1	72. 90
2	2	107. 47
3	1	131. 63
3	2	164. 39
4	1	210. 35
3	3	219. 32
4	2	242. 20
4	3	295. 69
4	4	370. 66

Figure 2.8: Clamped Modal Frequencies

purposes, the benefits of oversampling will be minimal, especially if trying to keep within real time computation. It is worth noting that the modes present will depend on both the read-in and readout position of the plate; should either lie on an anti-nodal point of frequency it will not be present in the output.

Under clamped conditions it becomes a little more difficult to analyse the modal behaviour. Using [7] as a reference we can compare the predicted modes. The modes for a square plate from [7, Table 4.24, Page 61] have been reproduced in figure 2.8. Figure 2.9 shows the output of the thin plate FTDT model with equal length and the predicted modal frequencies from 2.8.

When driven with a sine wave matching a modal frequency, the plate exhibits the checker board pattern that we would expect from clamped and fixed boundary

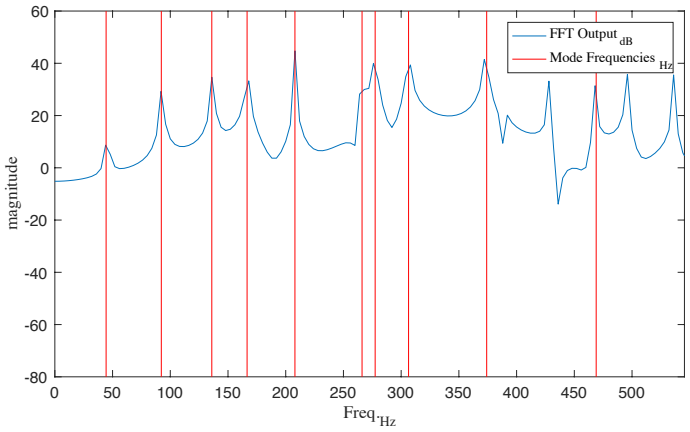

Figure 2.9: Thin Plate FD Modal Frequency Response and Approximated Modal Frequencies under clamped conditions

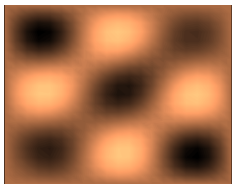

Figure 2.10: $f_{3,3}$ Modal frequency

conditions. Figure 2.10 shows the 3-3 mode of the plate. Further mode patterns are illustrated in Figure 2.11 replicated from [1, Figure 11.3, page 309]. Which all in all, further demonstrates that at the very least, the FTDT plate model is within the bounds of expected behaviour.

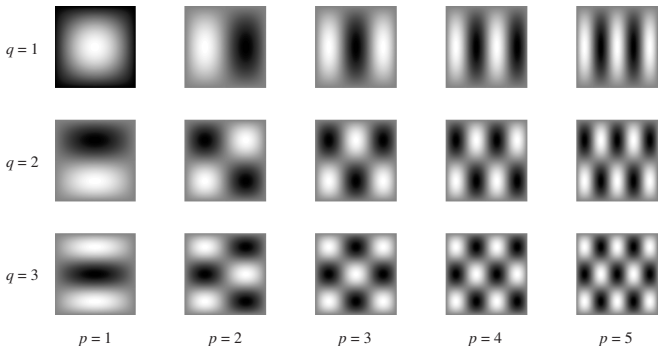

Figure 2.11: Modal frequency patterns under simply supported boundary conditions [1, Figure 11.3]

Chapter 3

Coupling Plate and Strings

This chapter will concern the problem of coupling the FDTD plate scheme discussed in Chapter 2. Also, it will look into the concepts in used order to achieve this goal; first deriving coupling conditions through energy analysis and then implementing them within the FDTD scheme. There are a number of constants/coefficients that are common between the plate and the string. For this chapter, and the remainder of this paper, all terms relating to just the plate will have a subscript $_p$ and all those relating to the string, a subscript s. For instance, the plate density ρ_p and string denisty ρ_s.

For coupling a plate and a string an assumption is made that the end of a string is rigidly connected to some point on a plate. Any two systems can be coupled together. This chapter should hopefully provide the groundwork for how to approach the coupling of two FDTD systems. For deriving the force that a single string exerts on a plate, we begin with equation 3.1. Little alteration needs to be made for coupling multiple strings, though this will be discussed in greater depth in 3.2.

$$\ddot{u} \quad = \quad -\kappa^2 \Delta\Delta u + \frac{1}{\rho_p H}\delta(x - x_e, y - y_e) f \tag{3.1}$$

The model of applying force in 3.1 is, again, with a dirac delta. It is identical to the application of force to the plate discussed in section 2.4. The difference here is that the means for the string driving the plate must be found. It is no longer a function of sample index f(n), but a scalar force exerted on the plate by the string.

3.1 1D Wave and Plate

The first string model that will be coupled to the plate is the 1D wave model

$$\rho_s A \ddot{w} = T w'' \tag{3.2a}$$

$$\ddot{w} = c^2 w'', \quad c = \sqrt{\frac{T}{\rho_s A}} \tag{3.2b}$$

Here, w repsents tanversal displacement of the string, ρ_s is the density of the string, A the cross sectional area and T is the tension. The constant c is the wave speed in the string. Like the plate it is advisable to return to 3.2b as greater complexity is added. A translation to a FDTD model of 3.2b is:-

$$\delta_{tt} w = c^2 \delta_{\eta\eta} w. \tag{3.3}$$

In this instance, η refers to the spatial dimension of the string. The spatial dimensions of the string and the plate are completely uncorrelated. This can make for some interesting combinations that would be impossible to realise physically. A ten meter long string where the ends are both coupled a mm apart on one meter squared plate would be very easy to mock up in a FD scheme. To couple the plate and string together we turn to energy analysis.

3.1.1 Coupling conditions

Rather than 'boundary conditions' there are 'coupling conditions', in this case a rigid coupling between the plate and the string. Assuming all energy is contained in both the string and the plate, equation 3.4, the change in energy being equal to the sum of the energy of the string and the plate leaves us with a boundary term for the string and a force term for the plate. Equating the units (speed of force, speed of boundary) to each other gives the first interpretation of how to couple the two systems.

$$\frac{dE}{dt} = \mathcal{B}_p + \mathcal{B}_s + \int_{\mathcal{D}_s} J f \dot{u} \tag{3.4}$$

In this instance we know that the plate boundaries $\mathcal{B}_p$ equal 0 and we are only connecting one end of the string to the plate at w_0. Given this is a lossy system the change in energy is zero, therefore all of this reduces to a term of the force and a boundary condition for the string. For the 1D Wave Equation, the boundary term is:

$$\mathcal{B}_s = T[\dot{w} w']_0^{N_s} = \dot{w}_{N_s} w'_{N_s} - \dot{w}_0 w'^0 \tag{3.5}$$

If we aim to only connect one end of the string to the plate then in equation 3.5, N_s is the unconnected end of the string (.i.e $\dot{w}_{N_s} w'_{N_s} = 0$). As such, it is free to have typical simply supported or clamped boundary conditions applied to it. The benefit of only connecting one end is that there is control over the boundary conditions and as such, some aesthetic possibilities are left open. Taking all this into account we can express the remaining terms as in equation 3.6.

$$T\dot{w}_{N_s} w'_{N_s} - T\dot{w}_0 w'_0 + \int_{\mathcal{D}_s} \delta f \dot{u} = 0 \tag{3.6}$$

Converting this to FDTD, 3.6 transforms to:-

$$\begin{aligned} h_p^2 \sum_{\mathcal{D}_p} \frac{1}{h_p^2} J f \delta_{t\cdot} u &= T \delta_{t\cdot} w_0 \delta_{\eta-} w_0 \\ f J^T[\delta_{t\cdot} u] &= \delta_{t\cdot} w_0 \delta_{\eta-} w_0 \end{aligned} \tag{3.7}$$

The sum of the spreading vector J leaves just the scalar force f and coupling point of the plate $J^T[\delta_{t\cdot} u]$. Equating the units of the terms that are remaining we can impose conditions to satisfy coupling the two systems. both f and $T\delta_{\eta-} w_0$ are in Newtons and the two velocity terms, $\delta_{t\cdot} w_0$ and $J^T[\delta_{t\cdot} u]$ are of course in ms^{-1}. One interpretation of these coupling conditions is given in 3.8a along with a FDTD equivalent 3.8b.

$$\dot{w}_0 = J^T[\dot{u}], \quad f = T w'_0 \tag{3.8a}$$

$$\delta_{t\cdot} w_0 = J^T[\delta_{t\cdot} u], \quad f = T \delta_{\eta-} w_0 \tag{3.8b}$$

The conditions in 3.8b provide the starting point from which we can couple the two systems. The schemes for both plate and the string need to be expressed in terms of a centre time difference at the relevant grid point. These can then be equated which will provide a term for the force. The centre time difference schemes for the plate and string are shown in equations 3.9a and 3.9b.

$$J^T[\delta_{t\cdot} u] = J^T[-\frac{k\kappa^2}{2} \delta_{\Delta\Delta} u + \delta_{t-} u + \frac{k}{2\rho_p H h_p^2} J f] \tag{3.9a}$$

$$\delta_{t\cdot} w_0 = \frac{kc^2}{2h_s} (\delta_{x+} w_0 - \frac{1}{T} f) + \delta_{t-} w_0 \tag{3.9b}$$

Using 3.8b we can equate both these terms and express just in terms of the force.

$$f = \mathcal{M}(\frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - J^T[-\kappa^2\delta_{\Delta\Delta}u + \frac{2}{k}\delta_{t-}u]) \tag{3.10}$$

Where $\mathcal{M}$ can be though of as the mass ratio between the plate and the string and is defined as:

$$\mathcal{M} = \frac{\rho_s A h_s \rho_p H h_p^2}{\rho_s A h_s + \rho_p H h_p^2}$$

This definition of the force in 3.10 can then be fed back into the string and the plate. From this, an update of the coupled scheme can be derived just in terms of previously calculated points.

$$w_0^{n+1} = \lambda^2 D_{x+}w_0 + 2w - w^{n-1} - \frac{k^2}{\rho A h_s}f \tag{3.11}$$

$$J^T[u^{n+1}] = J^T[-\mu^2 D_{\Delta\Delta}u + 2u - u^{n-1}] + \frac{k^2}{\rho_p H h_p}f \tag{3.12}$$

At this point, given the differences between the terms for force, plate update and the string update, it is easy to get lost in the coefficient chaos, especially when deriving this for the first time. What 3.11 and 3.12 have in common is the k^2 term in front of the force term f. Taking the k^2 and integrating it into the force term means that we can express the force purely in terms of the coefficient matrix and a slightly modified version of the matrix C.

$$\begin{aligned} F &= \mathcal{M}(\lambda^2 D_{x+}w_0 + 2w_0 - 2w_0^{n-1} - J^T[-\mu^2 D_{\Delta^2}u + 2u - 2u^{n-1}]) \\ F &= \mathcal{M}(B_{s(0,:)}w - J^T[B_p u] + 2C_{s(0,:)}w^{n-1} - J^T[2Cu^{n-1}]) \end{aligned} \tag{3.13}$$

The (0,:) subscript in 3.17 is an elegant way of cherry-picking the first row of the string matrices. We will see in a later section, that combining the two schemes together will simplify this problem. To a degree, 3.17 remains true for the inclusion of loss and stiffness of the string, both of which will be discussed in future sections. Explicitly the update for the string and the plate can be expressed as follows.

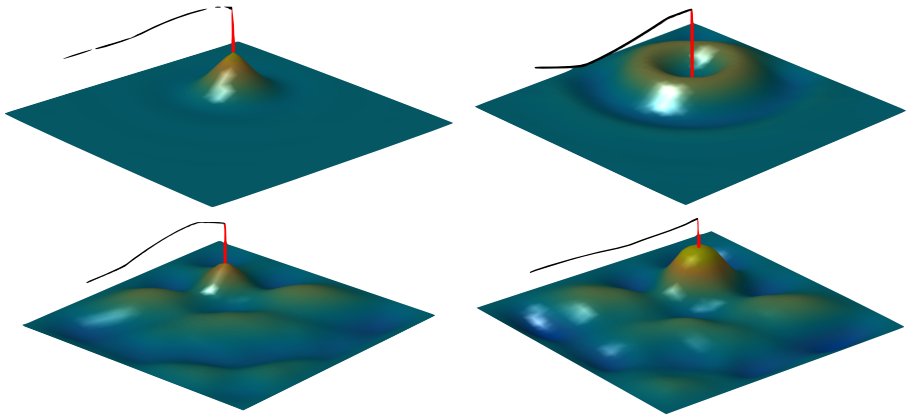

Figure 3.1: The string connected to the plate. The red line illustrates the point of connection between the two systems

$$\begin{aligned} w_0^{n+1} &= \lambda^2 D_{x+} w_0 + 2w - w^{n-1} - \frac{1}{\rho_s A h_s} F \\ J^T[u^{n+1}] &= J^T[-\mu^2 D_{\Delta\Delta} u + 2u - u^{n-1}] + \frac{1}{\rho_p H h_p} F \end{aligned} \tag{3.14}$$

3.1.2 1D Wave /w Loss

The 1D wave equation is limited to just a generic loss term. This is more useful than just a lossless system and is a good place to start generating some output with discernible notes. It can also provide an opportunity to begin planning on how to start expanding a model without rewriting the underlying infrastructure.

$$(1 + k\sigma_0) w^{n+1} = (\lambda^2 D_{\eta\eta} + I) w^n - (1 - k\sigma_0) w^{n-1} \tag{3.15}$$

The inclusion of loss in 3.15 has some implication on the derivation of the force term. The following should hopefully convey that these changes are fairly trivial with alteration to existing code. The derivation for 3.15 can be found in section A.15 which is worth consulting to understand how to manipulate schemes that include loss.

$$\left(\frac{1}{\rho_s A h_s (1 + k\sigma_{0s})} + \frac{1}{\rho_p H h_p^2 (1 + k\sigma_{0p})} \right) f = \frac{\frac{c^2}{h_s} \delta_{x+} w_0 + \frac{2}{k} \delta_{t-} w_0}{(1 + k\sigma_{0s})} - J^T \left[\frac{-\kappa^2 \delta_{\Delta\Delta} u + \frac{2}{k} \delta_{t-} u}{(1 + k\sigma_{0p})} \right] \tag{3.16}$$

The generic loss is a centre time difference so can be brought over to the LHS and factorised out (See Appendix derivation A.23). The structure of the operation is mostly unchanged. The mass ratio now contains the extra loss terms added by the plate and the string

$$\mathcal{M} = \left(\frac{1}{\rho_s A h_s (1 + k\sigma_{0s})} + \frac{1}{\rho_p H h_p^2 (1 + k\sigma_{0p})} \right)^{-1}$$

Following the same approach as the lossless case, the coupling force is also the same as before for the current time step matrix B and a little alteration must be made for the C matrix.

$$\begin{aligned} F &= \mathcal{M} \left(\frac{\lambda D_{x+} w_0 + 2w_0^n - 2w_0^{n-1}}{(1 + k\sigma_{0s})} - J^T [\frac{-\kappa^2 \delta_{\Delta\Delta} u + \frac{2}{k} \delta_{t-} u}{(1 + k\sigma_{0p})}] \right) \\ F &= \mathcal{M}(B_{s(0,:)} w - J^T [B_p u] + (C_s - I_s)_{(0,:)} w^{n-1} - J^T [(C_p - I_p) u^{n-1}]) \end{aligned} \tag{3.17}$$

It is worth noting at this point that all loss modelled in this paper will be implemented explicitly. That is to say, the loss matrix for both the string and the plate, A, is strictly diagonal and does not include any spatial differences. As a result the inverse of A can be precomputed and stacked on top of the B and C matrices to minimise computational costs.

3.1.3 1D Wave /w Force

The exact same approach to force discussed in section 2.4 can be applied to the string.

$$\delta_{tt} w = c^2 \delta_{\eta\eta} w - 2\sigma_0 \delta t \cdot w + \frac{1}{\rho_s A h} \mathcal{J}_f f(n) \tag{3.18}$$

The operator $\mathcal{J}_f$ just locates the force at right grid point for the vector w. It can be either a raised cosine for strikes or half raised cosine for plucks. The important point to remember when applying any force is the $\frac{k^2}{\rho_s A h}$ coefficient that is in front of any force term for FDTD string model. Also, there is nothing to stop force from being applied to both plate and string for percussive accompaniment. Imagination is the limit on this front. The string force can also be interpreted as a series of vectors for each string, containing raised cosines at the required excitation points. One possible fault in this approach is the situation where two excitations on the same string overlap. It is important that the excitation is not overwritten as (figure 3.2 a.), as this will create a sudden pluck. This will result in a very prominent note playing. An unusual case but, worth being aware of especially if excitation is generated by some data source such as a MIDI file. Figure 3.2 illustrates the product of both approaches.

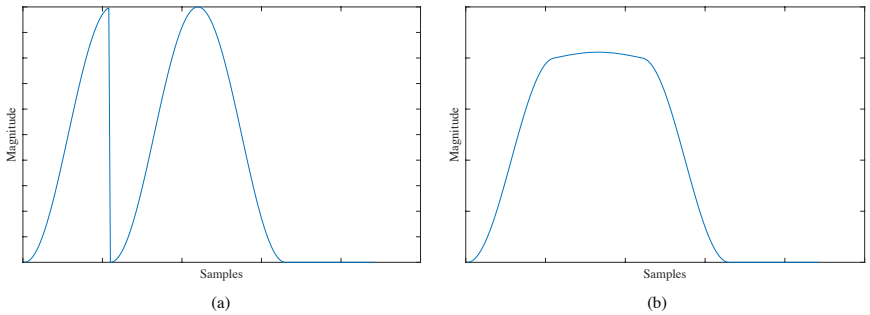

Figure 3.2: Force Vector a) overwritten b) summation: What should be an imperceptible double strike (b) can result in an unpleasant pluck (a).

3.1.4 Joining vectors

What can be beneficial for computation is to amalgamate the update for the string and the plate into one neat operation. With a force acting on the string and the string driving the plate it is possible to confine all this into one operation $Au\vec{w} = Bu\vec{w} + Cu\vec{w} + J_f\mathcal{F}_n$. where J_f spreads a force across the correct point on the string. $\mathcal{F}$ is a pre-computed vector which is accessed at sample index 'n'. For the case of multiple string I is a matrix $N \times N_s$ and $\mathcal{F}$ is a matrix $N_f \times N_s$. N is the total number of grid points across the plate and all strings, N_s is the number of strings and N_f is the number of samples to be computed. The case of multiple strings will be discussed in section 3.2.

The spreading operator, J, can be thought of as a concatenation of two spreading vectors. One that spread a force onto the plate J_p and another that connects to the end of the string J_s. The equation for the coupling force can be restated as:-

$$\begin{aligned} F &= \mathcal{M}(\underbrace{(J_s^T[B_s] - J_p^T[B_p])}_{F^n} u\vec{w}^n + \underbrace{(J_p^T[(C_p - I_p)] + J_s^T(C_s - I_s))}_{F^{n-1}} u\vec{w}^{n-1}]) \\ F^n &= \mathcal{M}(J_s^T[B_s] - J_p^T[B_p]) \\ F^{n-1} &= \mathcal{M}(J_p^T[C_p - I_p] + J_s^T[C_s - I_s]) \end{aligned} \tag{3.19}$$

If the plate and the string are joined in a single vector $u\vec{w}$, 3.19 can be integrated into the coefficient matrices for the plate and string. In this case, F^n and F^{n-1} are the force coefficients of the B and C matrices for coupling the plate and string. Remembering that the concatenated vectors are now operated on by concatenated matrices. If this is difficult to follow, see Appendix B.3 lines 250 - 251. For spreading the force back out, the coefficients in front of the force term in 3.14 can be integrated into the new spreading vector J (see Appenix B B.3 lines 216 - 238).

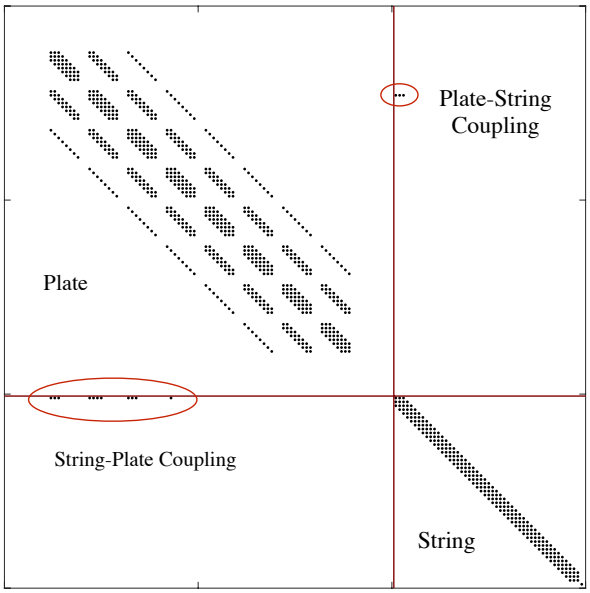

Figure 3.3: The Matrix Pattern of a joined plate and string system. Coupled points have been circled and the sections demarcated to reflect which system they refer.

$$u\vec{w}^{n+1} = \underbrace{(B_1 + JF^n)}_{\text{B}} u\vec{w}^n - \underbrace{(C_1 + JF^{n-1})}_{\text{C}} u\vec{w}^{n-1} + J_f\mathcal{F}_n \tag{3.20}$$

The Matrices B_1 and C_1 refer to the combined coefficient matrices without the coupling force included. What 3.20 implies is that the full Matrices B and C can be precomputed along with the loss that comes from the A matrix. It also allows for easy expansion to multiple strings as well as beginning to include a more sophisticated interpolation for connecting to the plate.

Once the force is integrated into the coefficient matrix the coupling points should be visible. This is illustrated in figure 3.3, which shows the region of the matrix that relates to the plate and the string and sparse areas that lie to the side indicate that a point of one system is coupled to another.

3.1.5 Interpolation

It is worth mentioning a different approach to the J operator at this stage. A more flexible way of coupling the strings is by a simple linear interpolation model. When selecting the desired coordinates of the string coupling point, it may not lie directly on a grid point. Since nothing technically exists between grid points, a method for finding a likely value for four grid points must be found. The force must be spread amongst 4 grid points proportionally (see Figure 3.4). J is no longer just a 1 at a

chosen co-ordinate but reflects the normalised distance the coupling point lies between each grid point, 3.21.

$$J = \begin{bmatrix} \vdots & \\ (1-\alpha_x)(1-\alpha_y) & (x_0, y_0) \\ (1-\alpha_x)\alpha_y & (x_0, y_1) \\ \vdots & \\ \alpha_x(1-\alpha_y) & (x_1, y_0) \\ \alpha_x\alpha_y & (x_1, y_1) \\ \vdots & \end{bmatrix} \tag{3.21}$$

3.2 Multiple Strings

Just one string coupled to a plate has limited musical application. Multiple strings allows for a more diverse set of tonal possibilities. Adding extra strings is also a quick win in that it gives a sympathetic vibration effect to the output which can be put to further creative use (see 4.3).

When considering multiple strings the coupling force F is no longer a scalar but a vector. The spreading operator J becomes a matrix and the string coefficients also become a matrix. Sticking with the flooring operator, (i.e. no interpolation), some issues can arise issue, particularly if it is preferable to avoid involved dealings with

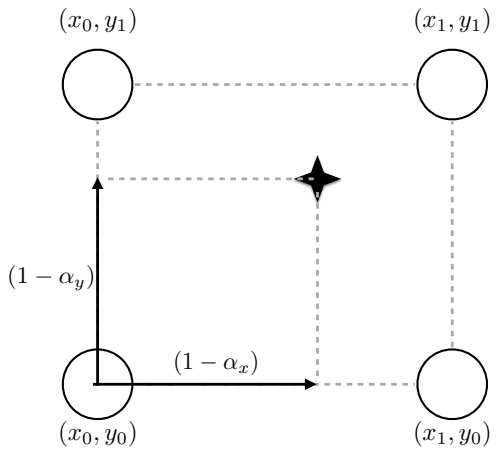

Figure 3.4: The cross represents the coupling point lieing between grid points. The normalised distance (or percentage) that the coupling point lies towards the next grid point on the x and y axis is represented by α_x and α_y respectively.

interpolation. Any two strings attached to the same coupling point will result in an unstable model. Extra error checking to ensure stability would need to be implemented. This would mean limiting the number of strings, ensuring all coupling points are unique or extending the number of grid points to accommodate all the strings (This is the approach taken in `thin_plate_stiff_string_loss.m` of the digital archive.). At which stage, it is beneficial to just to implement interpolation (discussed in section 3.1.5).

$$\underbrace{(\frac{1}{\rho_s A h_s(1+k\sigma_{0s})}I + \frac{1}{\rho_p H h_p^2(1+k\sigma_{0p})}J_p^T J_p)}_{\mathcal{M}^{-1}} f \quad = \quad \frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - ... \\ J^T[-\kappa^2\delta_{\Delta\Delta}u + \frac{2}{k}\delta_{t-}u] \tag{3.22}$$

There is not a great deal of change between 3.16 and 3.22. The main difference here is that the multiplication of $J_p^T J_p$ is not a scalar. Should all connecting strings share no grid points in common then $J_p^T J_p$ is an identity matrix for f. Otherwise, $J_p^T J_p$ is a more complex matrix that will need to be inverted. The trem $\frac{1}{\rho_s A h_s(1+k\sigma_{0s})}$ is also a diagonal matrix made up of the corresponding parameters for each string. Should all strings share the same $\frac{1}{\rho_s A h_s(1+k\sigma_{0s})}$ it can be considered a scalar. The mass ratio term $\mathcal{M}$ is now an inverse matrix from $\mathcal{M}^{-1}$. There are a number of means for determining this. MATLAB's 'backlash operator', or inverse matrix method or a Jacobi method approach. The luxury in this case is that it is static, so it does not need to be recomputed every sample. If this were not the case, it would be worth considering the Jacobi method approach. Other than the construction of $\mathcal{M}$, the Force is identical to 3.19. The greatest difficulty at this point is indexing, which is more a question of tedium rather than any great mental challenge. Careful collation of coupling indices should allow the construction of the spreading operators, and $\mathcal{M}$, to be a fairly trivial procedure.

3.3 The Stiff String

Developing the scheme further, the 1D wave can be substituted for the a stiff string model

$$\rho_s A\ddot{w} = Tw'' - E_s I w''''\ddot{w} = c^2 w'' - \kappa_s^2 w'''' \tag{3.23}$$

The stiffness of the string brings with it a forth order term along with Young's

Modulus for the string E_s and the string moment of inertia I_s.

$$w^{n+1} = c^2k^2\delta_{\eta\eta}w - \kappa^2k^2\delta_{\eta\eta\eta\eta} - 2\sigma_0\delta_{t\cdot}u + 2w - w^{n-1} + \frac{hk^2}{\rho A}Jf(n) \tag{3.24}$$

The update for the stiff string FD scheme is 3.32. With the stiffness comes a new bound on the grid spacing.

$$h_\eta \geq \sqrt{\frac{c^2k^2 + \sqrt{c^4k^4 + 16\kappa_s^2k^2}}{2}} \tag{3.25}$$

Splitting equation 3.32 into coefficient matrices gives the following:-

$$\underbrace{(I + k\sigma_0 I)}_{A_s} w^{n+1} = \underbrace{(\lambda^2 D_{\eta\eta} + \frac{2k\sigma_1}{h^2}D_{\eta\eta} + 2I - \mu^2 D_{\eta\eta\eta\eta})}_{B_s} w^n - ... \tag{3.26}$$

$$\underbrace{(\frac{2k\sigma_1}{h^2}D_{\eta\eta} + (1 - k\sigma_0)I)}_{C_s} w^{n-1} + \frac{k^2}{\rho Ah}Jf(t) \tag{3.27}$$

Only this section would need to be altered in a FD script. The force term for the concatenated system 3.20 goes unchanged, making this a reasonably easy extension to implement.

3.3.1 Coupling with Stiff String

With energy analysis a new set of coupling conditions are derived for the stiff string. The change in energy of the stiff string, $\frac{d\mathcal{E}}{dt}$, is defined as:-

$$\frac{d\mathcal{E}}{dt} = T[\dot{w}w']_0^{N_\eta} - E_sI[\dot{w}w'']_0^{N_\eta} + E_sI[\dot{w}'w'']_0^{N_\eta} \tag{3.28}$$

Comparing the units of force term from the plate energy, the following FDTD coupling conditions are derived.

$$\delta t\cdot w_0 = J^T[\delta_{t\cdot}u] \tag{3.29a}$$

$$\delta_{\eta\eta}w_0 = 0 \tag{3.29b}$$

$$f = T\delta_{\eta-}w_0 - EI\delta_{\eta-}\delta_{\eta\eta}w_0 \tag{3.29c}$$

In order to maintain the same $A = B + C$ structure of the finite difference update, construction of the matrices for the string need to be altered. In order for the scheme

to comply to 3.29a then the $D_{\eta\eta}$ should have the first row as all zero.

$$D_{\eta\eta} = \frac{1}{h^2}\begin{bmatrix} 0 & 0 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & b & -2 \end{bmatrix} \tag{3.30}$$

The fourth order matrix should be constructed by squaring the second order. This should help to keep the construction of coefficient matrices automated.

$$D_{\eta\eta\eta\eta} = D_{\eta\eta}D_{\eta\eta}$$

The force term also changes but not by a great degree.

$$\left(\frac{1}{\rho_s A h_s(1+k\sigma_{0s})}\bar{I} + \frac{1}{\rho_p H h_p^2(1+k\sigma_{0p})}J^tJ\right) f = (\frac{\frac{c^2}{h}\delta_{\eta+}w_0 - \frac{\kappa^2}{h_s^2}\delta_{\eta\eta}w_1 + \frac{k}{2}w_0}{1+k\sigma_{0s}} - ... \\ J^T\left[\frac{-\kappa^2\delta_{\Delta\Delta}u + 2\sigma\delta_{t-}\delta_{\Delta}u}{1+k\sigma_{0p}}\right]) \tag{3.31}$$

Should the string matrices be prepared correctly then 3.19 and 3.20 should still hold true. This means that again, very little intervention is required to actually implement the stiff string (see Appendix B.3 lines 180-200).

3.3.2 Frequency Dependant Loss

With the stiffness term in the string, frequency dependent loss term can also be added. A new term appears in 3.23, $2\sigma_1\dot{w}''$ which requires a different approach in the definition of σ_0.

$$w^{n+1} = c^2k^2\delta_{\eta\eta}w - \kappa^2k^2\delta_{\eta\eta\eta\eta} - 2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_{t-}\delta_{\eta\eta} + 2w - w^{n-1} + \frac{hk^2}{\rho A}Jf(n) \tag{3.32}$$

The calculation for σ_0 now changes. Taking the same approach as [1, page 178] where σ is a function of frequency. We first derive a characteristic equation for the string using an ansatz:

$$w = e^{st}e^{j\beta\eta}, \quad s = \sigma + j\omega$$

from which we derive a function of frequency

$$\sigma(\omega) = -\sigma_0 - \sigma_1 \xi(\omega), \quad \xi(\omega) = \frac{-c^2 + \sqrt{c^4 + \kappa^2 \omega^2}}{2\kappa^2}$$

Setting a T_60 for 2 frequency to the same as what is given in equation (7.29) of [1, page 178].

$$\sigma_0 = \frac{6\ln(10)}{\xi(\omega_2) - \xi(\omega_1)} \left(\frac{\xi(\omega_2)}{T_{60}(\omega_1)} - \frac{\xi(\omega_2)}{T_{60}(\omega 2)} \right), \quad \sigma_1 = \frac{6\ln(10)}{\xi(\omega_2) - \xi(\omega_1)} \left(-\frac{1}{T_{60}(\omega_1)} + \frac{1}{T_{60}(\omega 2)} \right) \tag{3.33}$$

For the plate, the exact same apporach can be taken except, there is no c^2 term in the plate. As such, the ξ function reduces to

$$\xi(\omega) = \frac{\omega}{\kappa}$$

$$u^{n+1} \underbrace{(I + k\sigma_0 I)}_{\text{A}} = \underbrace{(-\mu_p^2 D_{\Delta\Delta} + 2k\sigma_{1p} D_\Delta + 2I)}_{\text{B}} u^n - \underbrace{((1 - k\sigma_0)I + 2k\sigma_{1p} D_\Delta)}_{\text{C}} u^{n-1} \tag{3.34}$$

If $\sigma_1 = 0$, the scheme reduces down to the terms of the generic loss case. A conditional statement for loss should allow for easy switching between the two methods of loss. Example code is available at Appendix B.3 lines 100-130.

Alteration needs to take place to the stability condition of the plate and string when loss is introduced. For plate stability conditions with frequency dependant loss, consult [6]. For the plate and string the new minimum grid spacing have been replicated below.

$$h_p \geq \sqrt{4\sigma_{1p}k + \sqrt{(4\sigma_{1p}k)^2 + 12\kappa_p^2 k^2}} \tag{3.35a}$$

$$h_s \geq \sqrt{c^2 k^2 + 4\sigma_{1s}k + \sqrt{(c^2 k^2 + 4\sigma_{1s}k)^2 + 16\kappa_s^2 k^2}} \tag{3.35b}$$

3.4 Scheme Output

Selecting a point on the plate and assigning it to an output vector will suffice to produce an output. Amplitude output such as this will be very low passed. Amplitude output tends to exhibit a DC-like off-set. It can cause difficulties in avoiding clipping on output.

It is advisable to use velocity as the output; this helps to suppress the low frequency bias of amplitude output. Of course, this system is not radiating in a room, coupled to air, so any output directly from the plate will just be an approximation.

Should an excitation be close to a coupling point, there can be very large spikes in

output vector. This will very much depend on grid spacing, magnitude of force and plate dimensions. It is worth keeping in mind, perhaps built into error checking, to accommodate or anticipate the proximity of the excitation to the output.

Consideration should also be given to the frequency content of the output, which will depend on where the output 'pickup' is placed. If the read-out point lies directly at a nodal or anti-nodal point then frequencies can be entirely absent or overly prominent respectively. Treating the read-out point as a normalised co-ordinate, careful coding can help to avoid this kind of problem, regardless of plate dimensions.

Simply selecting two opposing points on a plate will produce a stereo output of fairly good quality. Experimenting with multiple points that are spread across the stereo field with equal power panning can also produce some interesting audio, see [8, 457-461].

Chapter 4

Extensions to the FD Plate and Coupled String

Though many of the main issues concerning implementation have been covered in the previous chapters, there are still some extensions and methods of approach to these models that will improve the quality of the output and code. This chapter will first deal with methods for optimisation as well as any helpful tips for those unfamiliar with programming. With regard to extending the schemes, some extensions have been carried out and are contained in the digital archive accompanying this paper. This includes generating notes by parsing MIDI data which was briefly touched on in section 3.1.3. MIDI offers the easiest, ready-made solution but, for a greater degree of nuance, a music notation based markup language may also be considered. There are also some further ideas which have not been tackled, which are suggested in section 4.4.

4.1 Optimisation

For those taking on a coding project such as this for the first time it can be beneficial to take into consideration a few helpful tips before coding. It is best to attempt to compartmentalise each section of the scheme; isolating each part in this way should allow for easy alteration without having to tread over old ground. This is obvious for seasoned coders but still worth remembering, even for a fairly short script. Of course there are always times when optimising will require amalgamating or discarding entire sections and there is no real solution but to rebuild the infrastructure of a script.

Taking advantage of some file versioning system is highly recommended. Not keeping track of changes to code can have disastrous consequences. Simply creating new files for every edit will also lead to a cluttered library of scripts fairly quickly, though is better than no versioning at all. Creating bespoke functions will greatly reduce space and code duplication; some 2D FDTD functions have been suggested in Appendix B.1. A little tweaking should allow for an arbitrary number of dimensions

as well as boundary conditions. The best means of optimising code in MATLAB is to keep coefficient matrices in sparse matrix format. Careful linear indexing and a little thought can render it a fairly simple problem (see section 1.3). This is the approach used for all FDTD schemes in the digital archive. Appendix B.3 presents a simple single string/plate model. From experience, it is tricky to initially get to grips with this kind of logic. Starting with small and theoretical examples should help familiarisation.

Joining systems together will greatly improve computation speed. This has been discussed in the previous chapter, though it is worth reiterating that the real difficulty here is careful indexing. The key goal in any optimisation is to scrutinise every aspect of the model and avoid duplicating any computation.

4.2 Calibration

An easy method for producing timbres that are less synthetic involves calibrating the models against some real recordings. This has been carried the case of the Piano Models in the digital archive. Individual strings of a piano were recorded while all others were damped with fabric. The frequency spectrum of the piano strings was compared to the output of isolated FDTD stiff string. The stiff string parameters, such as Young's Modulus, length, radius, loss were altered until the frequency response of the FDTD model closely resembled that of the recording. Changes to Young's modulus accommodate in part for changes in string behaviour from overwinding the solid core with another material (see Figure 4.2).

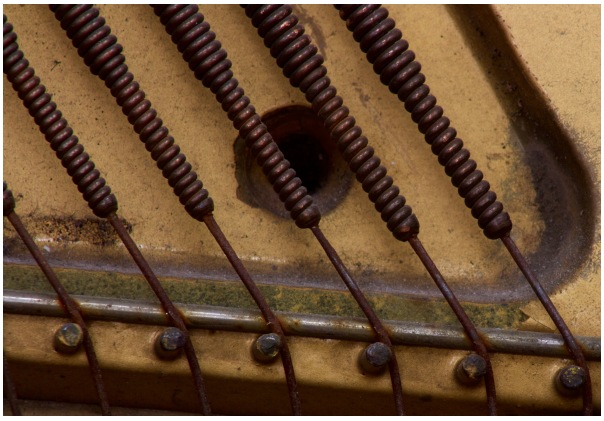

Figure 4.1: Overwound Piano String

It is advisable to construct an environment that will allow for easy comparison between a string model and sound file. With sensible file naming, a single changed variable should be all that is required to compare different strings. The data can either be stored in a separate file or supplant the current string parameters. One warning when undertaking this is that the stability condition for grid spacing should be changed,

Figure 4.2: The Palme Diffuseur for the Ondes Martenot [2]

as in 3.35a and 3.35b. Without this change it is likely that the model will be unstable unless under very particular conditions.

With regard to the plate properties, [9, Chapter 5] has been a valuable reference of poisson ratios and densities of wood as well as [10] which is a useful source of shear coefficients.

4.3 Reverberation

The sympathetic vibration of multiple strings on a plate can lead to some interesting reverberation effects. The effect of a piano with a sustain pedal held down during recording was a favourite of Frank Zappa [11, 79] and driving the plate/string model with audio is one approach to mimicking the behaviour of systems such as the Ondes Martenot's Palme Diffuseur (Figure 4.2).

The FDTD scheme suggested may not be constrained to behaving as musical instrument but may also be used as a digital audio effect. In this paper, just one

plate and one set of strings has been coupled. Their is no real limit to the number of coupled systems and a whole realm of physically impossible instruments and effects is available.

4.4 Further additions

Aesthetically the output of the previously discussed schemes is still noticeably synthetic. This is not to say that there are not musical possibilities however, if the goal of the reader is to create sounds approaching a 'real' instrument there are a number of ways forward. For fretted instruments there are the interactions between the string and the fretboard [12]. Damping, mallet and other force modelling is covered well in [1], as well as nonlinear models of both the string and plate. A variation in the string model will provide the greatest aesthetic change, so may be the first place to consider investigating.

For the plate model, there is the addition of anisotropy. The difficulty here is the decision of the shear modulus for a material, of which there is no consensus. This becomes a more intense problem to solve and may not be for the casual physical modeller. Coupling with air and irregular geometries will require a step towards finite element or finite volume approaches both on which are beyond the scope of this paper.

4.5 Reflections

The initial goal of this project was to create a full irregularly-shaped air-coupled instrument model. Under the constraints of a three month working period, this has proven to be a little too ambitious. With what has been created instead, there is still potential for musical applications. Throughout the project the main division has been between analysis and implementation. With a limited time scope, to implement the most diverse and interesting model requires that it be checked for stability, and validity, analytically. To maximise interesting results, a healthy balance has to be found between the two. Hopefully this paper will have provided some use to those undertaking a similar project, and helped to illuminate the more difficult concepts more clearly.

Appendix A

Full Derivation of Formulas

Each section of the Appendix relates to the equivalent Chapter

A.1 The Thin Plate

A.1.1 Lossless Thin Plate

$$\begin{aligned}
\rho H \delta_{tt} u &= -D \delta_{\Delta\Delta} u \\
\delta_{tt} u &= -\kappa^2 \delta_{\Delta\Delta} u \\
\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) &= -\kappa^2 \delta_{\Delta\Delta} u \\
u^{n+1} &= -k^2\kappa^2 \delta_{\Delta\Delta} u + 2u^n - u^{n-1} \\
&= -\mu^2 D_{\Delta\Delta} u + 2u^n - u^{n-1}
\end{aligned} \tag{A.1}$$

A.1.2 Thin Plate w/ Generic Loss

$$\begin{aligned}
\rho H \delta_{tt} u &= -D \delta_{\Delta\Delta} u - 2\rho H \sigma_0 \delta_{t.} u \\
\delta_{tt} u &= -\kappa^2 \delta_{\Delta\Delta} u^n - 2\sigma_0 \delta_{t.} u \\
\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) &= -\kappa^2 \delta_{\Delta\Delta} u^n - \tfrac{2\sigma_0}{2k} u^{n+1} + \tfrac{2\sigma_0}{2k} u^{n-1} \\
u^{n+1} &= -k^2\kappa^2 \delta_{\Delta\Delta} u^n + k^2 \sigma_0 u^{n-1} + 2u^n - u^{n-1} \\
u^{n+1} + k^2 \sigma_0 u^{n+1} &= -k^2\kappa^2 \delta_{\Delta\Delta} u^n + k^2 \sigma_0 u^{n-1} + 2u^n - u^{n-1} \\
\underbrace{(1 + k^2\sigma_0)}_{A} u^{n+1} &= \underbrace{(-\mu^2 D_{\Delta\Delta} + 2I)}_{B} u^n - \underbrace{(1 - k^2\sigma_0)}_{C} u^{n-1}
\end{aligned} \tag{A.2}$$

A.1.3 Thin Plate w/ Frequency Dependant Loss

$$
\begin{aligned}
\rho H\ddot{u} &= -D\Delta\Delta u - 2\rho H\sigma_0\dot{u} + 2\rho H\sigma_1\Delta\dot{u} \\
\ddot{u} &= -\kappa^2\Delta\Delta u - 2\sigma_0\dot{u} + 2\sigma_1\Delta\dot{u} \\
\delta_{tt}u &= -\kappa^2\delta_{\Delta\Delta}u + -2\sigma_0\delta_{t.}u + 2\sigma_1\delta_{\Delta}\delta_{t-}u \\
\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) &= -\kappa^2\delta_{\Delta\Delta}u - 2\sigma_0\delta_{t.}u + 2\sigma_1\delta_{\Delta}\delta_{t-}u \\
u^{n+1} &= k^2(-\kappa^2\delta_{\Delta\Delta}u - 2\sigma_0\delta_{t.}u + 2\sigma_1\delta_{\Delta}\delta_{t-}u + \frac{1}{\rho_p H h_p^2}Jf) + 2u^n - u^{n-1} \\
u^{n+1} &= -\kappa^2k^2\delta_{\Delta\Delta}u - \frac{2\sigma_0k^2}{2k}(u^{n+1} - u^{n+1}) + ... \\
&\quad 2\sigma_1k^2\delta_{\Delta}\delta_{t-}u \qquad \text{(A.3)} \\
u^{n+1} + \sigma_0ku^{n+1} &= -\kappa^2k^2\delta_{\Delta\Delta}u + 2u^n + 2\sigma_1k\delta_{\Delta}u + ... \\
&\quad \sigma_0ku^{n-1} - u^{n-1} - 2\sigma_1k\delta_{\Delta}u^{n-1} \\
\underbrace{(I + k\sigma_0I)}_{\text{A}}u^{n+1} &= \underbrace{(-\mu^2D_{\Delta\Delta} + 2\sigma_1D_{\Delta} + 2I)}_{\text{B}}u^n - ... \\
&\quad \underbrace{((1 - k\sigma_0)I + 2\sigma_1D_{\Delta})}_{\text{C}}u^{n-1} \qquad \text{(A.4)}
\end{aligned}
$$

The Bihamonic Operator

$$
\begin{aligned}
\delta_{\Delta\Delta} &= (\delta_{\Delta})^2 \\
(\delta_{\Delta})^2 &= (e_{x+1} - 2 + e_{x-1}e_{y+1} - 2 + e_{y-1})^2 \\
&= (20 - 8(e_{x+} + e_{x-} + e_{y+} + e_{y-}) + ... \\
&\quad (e_{x+,y+} + e_{x+,y-} + e_{x-,y+} + e_{x-,y-}) + ... \\
&\quad (e_{x+2} + e_{x-2} + e_{y+2} + e_{y-2}) \qquad \text{(A.5)}
\end{aligned}
$$

Simply Supported:

$$
\begin{aligned}
\delta_{xx}u_0 &= 0 \\
\delta_{xx}u_0 &= \frac{1}{h^2}(u_1 - 2u_0 + u_{-1}) \\
\frac{1}{h^2}(u_1 - 2u_0 + u_{-1}) &= 0 \\
u_1 - 2u_0 + u_{-1} &= 0 \\
u_{-1} &= 2u_0 - u_1, \quad (u_0 = 0) \\
u_{-1} &= -u_1 \qquad \text{(A.6)}
\end{aligned}
$$

Clamped:

$$\begin{aligned} \delta_{x\cdot} u_0 &= 0 \\ \delta_{x\cdot} u_0 &= \frac{1}{2h}(u_1 - u_{-1}) \\ \frac{1}{2h}(u_1 - u_{-1}) &= 0 \\ u_1 - u_{-1} &= 0 \\ u_{-1} &= u_1 \end{aligned} \tag{A.7}$$

A.1.4 Energy: Continuous

Boundary conditions = $\mathcal{B}$

$$\begin{aligned} \ddot{u} &= -\kappa^2 \Delta\Delta u \quad (\times \dot{u}, \int_D) \\ \int_{\mathcal{D}} \dot{u}\ddot{u} &= -\kappa^2 \int_{\mathcal{D}} \dot{u}\Delta\Delta u \\ \int_{\mathcal{D}} \frac{\partial}{\partial t}[\frac{1}{2}\dot{u}^2] = -\kappa^2 \langle \dot{u}, \Delta\Delta u\rangle_{\mathcal{D}} & \\ \frac{\partial}{\partial t}\left[\frac{1}{2}\int_{\mathcal{D}} \dot{u}^2\right] &= \mathcal{B} - \frac{\partial}{\partial t}\left[\frac{\kappa^2}{2}\langle \Delta u, \Delta u\rangle_{\mathcal{D}}\right] \\ \frac{\partial}{\partial t}\underbrace{\left[\frac{1}{2}\|\dot{u}\|^2_{\mathcal{D}} + \frac{\kappa^2}{2}\|\Delta u\|^2_{\mathcal{D}}\right]}_{E} &= \mathcal{B} \end{aligned} \tag{A.8}$$

A.1.5 Energy: Discrete

$$\begin{aligned} \delta_{tt} u &= -\kappa^2 \delta_{\Delta\Delta} u \\ h^2 \sum_{\mathcal{D}} \delta_{t\cdot} u \delta_{tt} u &= -\kappa^2 h^2 \sum_{\mathcal{D}} \delta_{t\cdot} u \Delta\Delta u \\ h^2 \sum_{\mathcal{D}} \frac{1}{2}\delta_{t+}[\delta_{t-} u]^2 &= -\kappa^2 h^2 \langle \delta_{t\cdot} u, \Delta\Delta u\rangle_{\mathcal{D}} \\ \frac{h^2}{2}\delta_{t+}\|\delta_{t-} u\|^2 &= -\kappa^2 h^2 \langle \delta_{t\cdot} u, \Delta\Delta u\rangle_{\mathcal{D}} \\ \frac{h^2}{2}\delta_{t+}\|\delta_{t-} u\|^2 &= \mathcal{B} - \delta_{t+}\left[\frac{\kappa^2 h^2}{2h^4}\langle D_\Delta u^n, D_\Delta u^{n-1}\rangle_{\mathcal{D}}\right] \end{aligned} \tag{A.9}$$

Energy: Discrete Solution

$$\delta_{t+}\left[\frac{h^2}{2k^2}(u^n - u^{n-1})^T(u^n - u^{n-1}) + \frac{\kappa^2}{2h^2}(D_\Delta u^n)^T(D_\Delta u^{n-1})\right] = 0 \tag{A.10}$$

A.1.6 Modal Analysis: Simply Supported

$$
\begin{aligned}
u(x,y,t) &= e^{iwt}X(x)Y(y) \\
X(0) &= X(L_x) = 0, \quad Y(0) = Y(L_y) = 0 \\
X(x) &= \sin(k_x x), \quad Y(y) = \sin(k_y y) \\
k_x &= \frac{p\pi}{L_x}, \quad k_y = \frac{q\pi}{L_y}, \quad p,q = \mathbb{Z}^+ \\
\\
\ddot{u} &= -\kappa^2 \left[\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial^2 x \partial^2 y} + \frac{\partial^4}{\partial y^4} \right] u \\
-\omega^2 u &= -\kappa^2 \left(k_x^4 + 2k_x^2 k_y^2 + k_y^4 \right) u \\
\omega^2 u &= \kappa^2 \left(k_x^2 + k_y^2 \right)^2 u \\
\omega &= \kappa \left(k_x^2 + k_y^2 \right) \\
2\pi f_{p,q} &= \kappa \left(\frac{p\pi}{L_x}^2 + \frac{q\pi}{L_y}^2 \right) \\
2\pi f_{p,q} &= \pi^2 \kappa \left(\frac{p}{L_x}^2 + \frac{q}{L_y}^2 \right) \\
f_{p,q} &= \frac{\pi\kappa^2}{2} \left(\frac{p^2}{L_x} + \frac{q^2}{L_y} \right)
\end{aligned}
\tag{A.11}
$$

A.1.7 Lossless w/ Force

$$
\begin{aligned}
\ddot{u} &= -\kappa^2 \Delta\Delta u + \frac{1}{\rho_p H} \delta(x - x_e, y - y_e) f \\
\delta_{tt} u &= -\kappa^2 \delta_{\Delta\Delta} u + \frac{1}{\rho_p H h_p^2} J f \\
\frac{1}{k^2}(u^{n+1} - 2u + u^{n-1}) &= -\kappa^2 \delta_{\Delta\Delta} u + \frac{1}{\rho_p H h_p^2} J f \\
u^{n+1} &= -k^2\kappa^2 \delta_{\Delta\Delta} u + 2u - u^{n-1} + \frac{k^2}{\rho_p H h_p^2} J f \\
u^{n+1} &= -\mu^2 D_{\Delta\Delta} u + 2u - u^{n-1} + \frac{k^2}{\rho_p H h_p^2} J f
\end{aligned}
\tag{A.12}
$$

A.1.8 Frequency Dependant Loss w/ Force

$$
\begin{aligned}
\rho H\ddot{u} &= -D\Delta\Delta u - 2\rho H\sigma_0\dot{u} + 2\rho H\sigma_1\Delta\dot{u} + \delta(x-x_e, y-y_e)f(t) \\
\ddot{u} &= -\kappa^2\Delta\Delta u - 2\sigma_0\dot{u} + 2\sigma_1\Delta\dot{u} + \frac{1}{\rho_p H}\delta(x-x_e, y-y_e)f(t) \\
\delta_{tt}u &= -\kappa^2\delta_{\Delta\Delta}u + -2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_\Delta\delta_{t-}u + \frac{1}{\rho_p H h_p^2}Jf \\
\frac{1}{k^2}(u^{n+1} - 2u^n + u^{n-1}) &= -\kappa^2\delta_{\Delta\Delta}u - 2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_\Delta\delta_{t-}u + \frac{1}{\rho_p H h_p^2}Jf \\
u^{n+1} &= k^2(-\kappa^2\delta_{\Delta\Delta}u - 2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_\Delta\delta_{t-}u + \frac{1}{\rho_p H h_p^2}Jf) + 2u^n - u^{n-1} \\
u^{n+1} &= -\kappa^2 k^2\delta_{\Delta\Delta}u - \frac{2\sigma_0 k^2}{2k}(u^{n+1} - u^{n+1}) + ... \\
&\quad 2\sigma_1 k^2\delta_\Delta\delta_{t-}u + \frac{k^2}{\rho_p H h_p^2}Jf + 2u^n - u^{n-1} \\
u^{n+1} + \sigma_0 k u^{n+1} &= -\kappa^2 k^2\delta_{\Delta\Delta}u + 2u^n + 2\sigma_1 k\delta_\Delta u + ... \\
&\quad \sigma_0 k u^{n-1} - u^{n-1} - 2\sigma_1 k\delta_\Delta u^{n-1} + \frac{k^2}{\rho_p H h_p^2}Jf \\
\underbrace{(I + k\sigma_0 I)}_{\mathrm{A}}\, u^{n+1} &= \underbrace{(-\mu^2 D_{\Delta\Delta} + 2\sigma_1 D_\Delta + 2I)}_{\mathrm{B}}\, u^n - ... \\
&\quad \underbrace{((1-k\sigma_0)I + 2\sigma_1 D_\Delta)}_{\mathrm{C}}\, u^{n-1} + \frac{k^2}{\rho_p H h_p^2}Jf
\end{aligned}
\tag{A.13}
$$

A.2 Coupling String and Plate

A.2.1 FDTD 1D Wave Lossless

$$
\begin{aligned}
\ddot{w} &= c^2 w'' \\
\delta_{tt}w &= c^2\delta_{\eta\eta}w \\
\frac{1}{k^2}(w^{n+1} - 2w + w^{n-1}) &= c^2\delta_{\eta\eta}w \\
w^{n+1} &= \lambda^2 D_{\eta\eta}w + 2w - w^{n-1}
\end{aligned}
\tag{A.14}
$$

A.2.2 FDTD 1D Wave Generic Loss

$$\begin{aligned} \ddot{w} &= c^2 w'' - 2\sigma_0 \dot{w} \\ \delta_{tt} w &= c^2 \delta_{\eta\eta} w - 2\sigma_0 \delta t \cdot w \\ \frac{1}{k^2}(w^{n+1} - 2w + w^{n-1}) &= c^2 \delta_{\eta\eta} w \\ \underbrace{(1 + k\sigma_0)}_{\text{A}} w^{n+1} &= \underbrace{(\lambda^2 D_{\eta\eta} + I)}_{\text{B}} w^n - \underbrace{(1 - k\sigma_0)}_{\text{C}} w^{n-1} \end{aligned} \tag{A.15}$$

A.2.3 Coupling Conditions: Energy Analysis

1D Wave Energy

$$\begin{aligned} \int_{\mathcal{D}_s} \rho_s A \dot{w} \ddot{w} &= \int_{\mathcal{D}_s} T \dot{w} w'' \\ \int_{\mathcal{D}_s} \rho_s A \dot{w} \ddot{w} &= [\dot{w} w']_0^{N_s} - \int_{\mathcal{D}_s} T \dot{w}' w' \\ \frac{dE_s}{dt} &= [\dot{w} w']_0^{N_s} = \mathcal{B}_s \end{aligned} \tag{A.16}$$

Plate and String Energy

$$\begin{aligned} \frac{dE}{dt} &= \mathcal{B}_p + \mathcal{B}_s + \int_{\mathcal{D}_s} J f \dot{u} \\ \int_{\mathcal{D}_s} J f \dot{u} &= f J^T \dot{u} \end{aligned} \tag{A.17}$$

A.2.4 Plate Centre Difference

$$\begin{aligned} J^T \delta_{tt} u &= J^T[-\kappa^2 \delta_{\Delta\Delta} u + \frac{1}{\rho_p H h_p^2} J f] \\ J^T[\frac{2}{k}(\delta_{t\cdot} u - \delta_{t-} u)] &= J^T[-\kappa^2 \delta_{\Delta\Delta} u + \frac{1}{\rho_p H h_p^2} J f] \\ J^T[\delta_{t\cdot} u] &= J^T[-\frac{k\kappa^2}{2} \delta_{\Delta\Delta} u + \delta_{t-} u + \frac{k}{2\rho_p H h_p^2} J f] \end{aligned} \tag{A.18}$$

A.2.5 1D Wave Centre Difference

$$\begin{aligned}
\delta_{tt}w &= c^2\delta_{xx}w \\
\frac{2}{k}(\delta_{t\cdot}w_0 - \delta_{t-}w_0) &= c^2\delta_{xx}w_0 \\
\delta_{t\cdot}w_0 &= \frac{kc^2}{2}\delta_{xx}w_0 + \delta_{t-}w_0 \\
&= \frac{kc^2}{2h_s}(\delta_{x+}w_0 - \frac{1}{T}f) + \delta_{t-}w_0
\end{aligned} \tag{A.19}$$

A.2.6 1D Wave/Plate Coupling Force

$$\begin{aligned}
\frac{kc^2}{2h_s}(\delta_{x+}w_0 - \frac{1}{T}f) + \delta_{t-}w_0 &= J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u + \frac{k}{2\rho_p H h_p^2}Jf] \\
(\frac{k}{2\rho_p H h_p^2} + \frac{kc^2}{2Th_s})f &= \frac{kc^2}{2h_s}\delta_{x+}w_0 + \delta_{t-}w_0 - J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u] \\
\frac{k}{2}(\frac{1}{\rho_s A h_s} + \frac{1}{\rho_p H h_p^2})f &= \frac{kc^2}{2h_s}\delta_{x+}w_0 + \delta_{t-}w_0 - J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u] \\
(\frac{1}{\rho_s A h_s} + \frac{1}{\rho_p H h_p^2})f &= \frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - J^T[-\kappa^2\delta_{\Delta\Delta}u + \frac{2}{k}\delta_{t-}u] \\
f &= (\mathcal{M})(\frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - ... \\
&\quad J^T[-\kappa^2\delta_{\Delta^2}u + \frac{2}{k}\delta_{t-}u]))
\end{aligned} \tag{A.20}$$

A.2.7 Thin Plate w/ Coupling Force

$$\begin{aligned}
\ddot{u} &= -\kappa^2\Delta\Delta u + \frac{1}{\rho_p H}\delta(x - x_e, y - y_e)f \\
\delta_{tt}u &= -\kappa^2\delta_{\Delta\Delta}u + \frac{1}{\rho_p H h_p^2}Jf \\
\frac{1}{k^2}(u^{n+1} - 2u + u^{n-1}) &= -\kappa^2\delta_{\Delta\Delta}u + \frac{1}{\rho_p H h_p^2}Jf \\
u^{n+1} &= -k^2\kappa^2\delta_{\Delta\Delta}u + 2u - u^{n-1} + \frac{k^2}{\rho_p H h_p^2}Jf \\
&= -\mu^2 D_{\Delta\Delta}u + 2u - u^{n-1} + \frac{k^2}{\rho_p H h_p^2}Jf
\end{aligned} \tag{A.21}$$

A.2.8 1D Wave FD Scheme w/ Coupling Force

$$\begin{aligned} w_0^{n+1} &= \frac{k^2c^2}{h_s}(\delta_{x+}w_0 - \frac{1}{T}f) + 2w - w^{n-1} \\ &= \lambda^2 D_{x+}w_0 + 2w - w^{n-1} - \frac{k^2}{\rho A h_s}f \\ &= \lambda^2 D_{x+}w_0 + 2w - w^{n-1} - ... \\ &\quad \frac{\mathcal{M}}{\rho_s A h_s}(\lambda^2\delta_{x+}w_0 + 2w_0 - 2w_0^{n-1} - ... \\ &\quad J^T[-\mu^2 D_{\Delta^2}u + 2u - 2u^{n-1}]) \\ &= \lambda^2 D_{x+}w_0 + 2w - w^{n-1} - ... \\ &\quad \mathcal{M}_s(\lambda^2 D_{x+}w_0 + 2w_0 - 2w_0^{n-1} - ... \\ &\quad J^T[-\mu^2 D_{\Delta^2}u + 2u - 2u^{n-1}])) \end{aligned} \tag{A.22}$$

A.2.9 1D Wave/Plate Coupling Force

$$\begin{aligned} \frac{kc^2}{2h_s}(\delta_{x+}w_0 - \frac{1}{T}f) + \delta_{t-}w_0 &= J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u + \frac{k}{2\rho_p H h_p^2}Jf] \\ (\frac{k}{2\rho_p H h_p^2} + \frac{kc^2}{2Th_s})f &= \frac{kc^2}{2h_s}\delta_{x+}w_0 + \delta_{t-}w_0 - J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u] \\ \frac{k}{2}(\frac{1}{\rho_s A h_s} + \frac{1}{\rho_p H h_p^2})f &= \frac{kc^2}{2h_s}\delta_{x+}w_0 + \delta_{t-}w_0 - J^T[-\frac{k\kappa^2}{2}\delta_{\Delta\Delta}u + \delta_{t-}u] \\ (\frac{1}{\rho_s A h_s} + \frac{1}{\rho_p H h_p^2})f &= \frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - J^T[-\kappa^2\delta_{\Delta\Delta}u + \frac{2}{k}\delta_{t-}u] \\ f &= (\mathcal{M})(\frac{c^2}{h_s}\delta_{x+}w_0 + \frac{2}{k}\delta_{t-}w_0 - ... \\ &\quad J^T[-\kappa^2\delta_{\Delta^2}u + \frac{2}{k}\delta_{t-}u])) \end{aligned} \tag{A.23}$$

A.2.10 Stiff String FD Scheme

$$w^{n+1} = c^2k^2\delta_{\eta\eta}w - \kappa^2k^2\delta_{\eta\eta\eta\eta} - 2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_{t-}\delta_{\eta\eta} + 2w - w^{n-1} + \frac{hk^2}{\rho A}Jf \tag{A.24}$$

$$
\begin{aligned}
\delta_{tt}w &= c^2\delta_{\eta\eta}w - \kappa^2\delta_{\eta\eta\eta\eta}w - 2\sigma_0\delta_{t.}w + 2\sigma_1\delta_{t-}w\delta_{\eta\eta}w + 2w - w^{n-1} + \frac{hk^2}{\rho A}Jf(n) \\
w^{n+1} &= c^2k^2\delta_{\eta\eta}w - \kappa^2k^2\delta_{\eta\eta\eta\eta} - 2k^2\sigma_0\delta_{t.}w + 2k^2\sigma_1\delta_{t-}\delta_{\eta\eta}w + \frac{h}{\rho A}Jf(n) \\
(1+k\sigma_0)w^{n+1} &= \lambda^2 D_{\eta\eta}w - \mu^2 D_{\eta\eta\eta\eta} + 2k\sigma_1\delta_{\eta\eta}w + 2w - ... \\
&\quad (1-k\sigma_0)w^{n-1} - 2k\sigma_1\delta_{\eta\eta}w^{n-1} + \frac{h}{\rho A}Jf(n) \\
\underbrace{(1+k\sigma_0)}_{A}\, w^{n+1} &= \underbrace{\left(\lambda^2 D_{\eta\eta} - \mu^2 D_{\eta\eta\eta\eta} + \frac{2k\sigma_1}{h^2}D_{\eta\eta} + 2I\right)}_{B} - ... \\
&\quad \underbrace{\left((1-k\sigma_0)w^{n-1} + \frac{2k\sigma_1}{h^2}D_{\eta\eta}w^{n-1}\right)}_{C} + \frac{hk^2}{\rho A}J_f f(n)
\end{aligned}
\tag{A.25}
$$

A.2.11 Stiff String and Plate w/ Frequency Dependent Loss Coupling Force

$$
\left(\frac{1}{\rho_s A h_s(1+k\sigma_{0s})}\bar{I} + \frac{1}{\rho_p H h_p^2(1+k\sigma_{0p})}J^tJ\right) f = \left(\frac{\frac{c^2}{h}\delta_{\eta+}w_0 - \frac{\kappa^2}{h_s^2}\delta_{\eta\eta}w_1 + \frac{k}{2}w_0}{1+k\sigma_{0s}} - ... \right. \tag{A.26}
$$

$$
\left. J^T\left[\frac{-\kappa^2\delta_{\Delta\Delta}u + 2\sigma\delta_{t-}\delta_{\Delta}u}{1+k\sigma_{0p}}\right]\right) \tag{A.27}
$$

Appendix B

MATLAB Code

Included here is a series of MATLAB scripts from the material discussed above. The hope is that it will provide a useful framework to begin creating the desired instrument as well as crafting functions that will make coding a little easier. Relatively small schemes can generate quite large matrices; as a space saving measure it is advisable to use the spare matrix format in MATLAB. The coefficient matrices are largely compiled of zero values. A sparse matrix will simply declare zero cells as empty space rather than an explicit floating point zero. This format greatly increases computation speed and saves on memory.

B.1 Building the Laplacian and Bi-Harmonic

The biharmonic can be a little difficult to grasp the construction of the first few times round. It is worth noting the points that the boundary grid points are zeroed out on lines 29-30. For this and the laplacian the boundary points that are always have been left in. There will be some wasted computation but the effect should be negligible especially for larger matrices. An obvious further improvement would be to construct the coefficient matrices without including any zero boundary points.

B.1.1 biharm.m

```
function BH = biharm(l,m,bctype)
  % BIHARM2 Generate Biharmonic Matrix
  %   BIHARM2(l,m,bctype) A function to generate the biharmonic
      coefficients for
  %   2D given the number of ALL grid points given by l and m.
  %   bctype denotes boundary condition
  %
  %         Returns a Sparse matrix BH
  %
  %      bctype  % boundary condition 1: simply supported, 2: clamped
  %      l       % number of total grid points Y axis
  %      m       % number of total grid points X axis

  l = l;
  m = m;
  Iy = speye(l); % identity matrix for y axis
  Ix = speye(m); % identity matrix for x axis
  ss = l*m;
  %%%%%%%%%%%%%%%%%%%%%%%%
  %% Building Bi-Harmonic
  %%%%%%%%%%%%%%%%%%%%%%%%
  XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,Ix);
  XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
  YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,Iy);
  YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
  LA = kron(XX,Iy) + kron(Ix,YY);
  BH = LA*LA;

  % set the correct points to 0
  BH(:,[1:l+1, ss-l:ss]) = 0;BH([1:l+1, ss-l:ss],:) = 0;
  BH(:,[2*l:l:ss, (2*l)+1:l:ss]) = 0; BH([2*l:l:ss, (2*l)+1:l:ss],:) =
      0;

  %EOF
```

B.1.2 laplace.m

```
function LA = laplace2(l,m,bctype)
  % LAPLACE2 Generate Laplacian Matrix
  %   LAPLACE2(l,m,bctype) A function to generate the biharmonic
      coefficients for
  %   2D given the number of ALL grid points given by l and m.
  %   bctype denotes boundary condition
  %
  %         Returns a Sparse Matrix Laplacian
  %
  %  bctype  % boundary condition: 1: simply supported, 2: clamped
  %  l       % number of internal grid points Y axis
  %  m       % number of internal grid points X axis

  l = l;
  m = m;
  Iy = speye(l); % identity matrix for y axis
  Ix = speye(m); % identity matrix for x axis
  ss = l*m;

  %%%%%%%%%%%%%%%%%%%%%%%%%
  %% Building Laplacian
  %%%%%%%%%%%%%%%%%%%%%%%%%

  XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,Ix);
  XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
  YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,Iy);
  YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
  LA = kron(XX,Iy) + kron(Ix,YY);

  %Set the correct points to 0
  LA(:,[1:l+1, ss-l:ss]) = 0; LA([1:l+1, ss-l:ss],:) = 0;
  LA(:,[2*l:l:ss, (2*l)+1:l:ss]) = 0; LA([2*l:l:ss, (2*l)+1:l:ss],:) =
      0;

  %EOF
```

B.1.3 fidimat.m

A composite function allowing for arbitrary creation of 1D and 2D finite difference matrices

```
function outM = fidimat(arg1,arg2,arg3,arg4)
  % FIDIMAT Generate Finite Difference Spatial Sparcity Matrices
  %   FIDIMAT(l,m,ord,bctype) A function to generate the biharmonic
  %   coefficients for 2D given the number of ALL grid points
  %   given by l and m.
  %
  %   bctype denotes boundary condition
  %
  %         Returns a Sparse matrix BH
  %
  %         bctype  % boundary condition type:
  %                      1: simply supported
  %                      2: clamped
  %         l       % number of total grid points Y axis
  %         m       % number of total grid points X axis
  %         ord     % order of the matrix (string)
  %
  %                 % Valid order inputs
  %                 % 'x-', 'x+', 'xx', 'xxxx', 'laplace', 'biharm'

  % Arguement check
  if nargin<2
    error('Not enough input arguements')

  elseif nargin==2

    if ischar(arg2)
      ord = arg2;
      l = arg1;
      m = 1;
      bctype = 1;
    else
      bctype = arg2;
      ord = 'xx';
      l = arg1;
      m = 1;
    end

  elseif nargin==3

    if ischar(arg2)
      ord = arg2;
      l = arg1;
      m = 1;
```

```
45        bctype = arg3;
46      elseif ischar(arg3)
47        ord = arg3;
48        l = arg1;
49        m = arg2;
50        bctype = 1;
51      else
52        bctype = arg3;
53        ord = 'xx';
54        l = arg1;
55        m = arg2;
56      end
57
58    elseif nargin==4
59      l = arg1;
60      m = arg2;
61      ord = arg3;
62      bctype = arg4;
63
64    elseif nargin>1
65      error('Too many input arguements')
66
67    end
68
69    Iy = speye(l); % identity matrix for y axis
70    Ix = speye(m); % identity matrix for x axis
71    ss = l*m;
72
73    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74    %%%%% 1D Case
75    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76
77    %% if only one dimension is stipulated the function will return only
78    %% a 1D FD matrix. In this case input is by the letter x
79
80    if m == 1
81
82      switch ord
83
84      case 'x-'
85        outM = spdiags([-ones(l,1),ones(l,1)],-1:0,speye(l));
86
87      case 'x+'
88        outM = spdiags([-ones(l,1),ones(l,1)],0:1,speye(l));
89
90      case 'x.'
91        outM = spdiags([-ones(l,1),zeros(l,1),ones(l,1)],[-1:1],speye(l));
92
93      case 'xx'
```

```
94
95        XX = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
96        XX(1,2) = (bctype-1)*2; XX(l,l-1) = (bctype-1)*2;
97        % if bctype == 0
98        %   XX([1 end],:) = 0;
99        % end
100       outM = XX;
101
102     case 'xxxx'
103       XX = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
104       XX(1,2) = (bctype-1)*2; XX(l,l-1) = (bctype-1)*2;
105       outM = XX;
106       outM = XX^2;
107
108     case 'I'
109       I = speye(ss);
110       I([1 end],:) = 0;
111       outM = I;
112
113     otherwise
114       error('something went wrong, check your arguements');
115
116     end
117
118     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
119     %%%%% 2D Case
120     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
121   else
122     % if two dimension arguements are given then the matri will output
123     % the corresponding 2D Matrix.
124     %
125     % y is for columns of a matrix whilst x is for rows.
126     % x matrices in this case will contain 'off centre'
127     % diagonals corresponding to the number of elements in the row.
128
129     switch ord
130
131     case 'y-'
132       Y = spdiags([-ones(l,1),ones(l,1)],-1:0,speye(l));
133       outM = kron(Ix,Y);
134
135     case 'y+'
136       Y = spdiags([ones(l,1),-ones(l,1)],0:1,speye(l));
137       outM = kron(Ix,Y);
138
139     case 'y.'
140       Y = spdiags([-ones(l,1),zeros(l,1),ones(l,1)],[-1:1],speye(l));
141       outM = kron(Ix,Y);
142
```

```
143     case 'yy'
144       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
145       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
146       YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
147       outM = kron(Ix,Y);
148
149     case 'yyyy'
150       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
151       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
152       YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
153       outM = kron(Ix,YY)^2;
154
155     case 'x-'
156       X = spdiags([-ones(m,1),ones(m,1)],-1:0,speye(m));
157       outM = kron(X,Iy);
158
159     case 'x+'
160       X = spdiags([ones(m,1),-ones(m,1)],0:1,speye(m));
161       outM = kron(X,Iy);
162
163     case 'x.'
164       X = spdiags([-ones(m,1),zeros(m,1),ones(m,1)],[-1:1],speye(m));
165       outM = kron(X,Iy);
166
167     case 'xx'
168       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
169       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
170       XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
171       outM = kron(XX,Iy);
172
173     case 'xxxx'
174       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
175       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
176       XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
177       outM = kron(XX,Iy)^2;
178
179     case 'grad'
180       Y = spdiags([-ones(l,1),zeros(l,1),ones(l,1)],[-1:1],speye(l));
181       X = spdiags([-ones(m,1),zeros(m,1),ones(m,1)],[-1:1],speye(m));
182       outM = kron(X,Iy) + kron(Ix,Y);
183
184     case 'xxyy'
185       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
186       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,speye(l));
187       YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
188
189       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
190       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,speye(m));
191       XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
```

```
192
193       outM = kron(Ix,YY)*kron(XX,Iy);
194
195     case 'laplace'
196
197       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,Ix);
198       XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
199       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,Iy);
200       YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
201       LA = kron(XX,Iy) + kron(Ix,YY);
202       outM = LA;
203
204     case 'biharm'
205
206       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
207       %% Building Bi-Harmonic
208       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
209       XX = spdiags([ones(m,1),-2*ones(m,1),ones(m,1)],-1:1,Ix);
210       XX(1,2) = (bctype-1)*2; XX(m,m-1) = (bctype-1)*2;
211       YY = spdiags([ones(l,1),-2*ones(l,1),ones(l,1)],-1:1,Iy);
212       YY(1,2) = (bctype-1)*2; YY(l,l-1) = (bctype-1)*2;
213       LA = kron(XX,Iy) + kron(Ix,YY);
214       BH = LA*LA;
215       outM = BH;
216
217     case 'I'
218       I = speye(ss);
219       outM = I;
220
221     otherwise
222       error('something went wrong, check your arguements');
223
224     end
225
226     % set the correct points to 0
227     outM(:,[1:l+1, ss-l:ss]) = 0;outM([1:l+1, ss-l:ss],:) = 0;
228     outM(:,[2*l:l:ss, (2*l)+1:l:ss]) = 0; outM([2*l:l:ss, (2*l)+1:l:ss
          ],:) = 0;
229
230   end
```

B.2 The Kirschoff Thin Plate

KirschhoffPlate.m

```
%%%%% FDTD 2D Wave Model
%%%%% Matthew Hamilton s0674653
%%%%% Description:
%%%%%
%%%%% A Kirchhoff Thin Plate FDTD Model

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Flags
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Conditions
bctype = 2;   % boundary type: 1: simply supported, 2: clamped
losstype = 2; % loss type: 1: independant, 2: dependant

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% simulation
Tf = 1;                         % duration seconds
nu = .4;                        % Poisson Ratios (< .5)
ctr = [.25, .25];               % centre point of excitation as percentage
wid = .1;                       % width (m or %)?
u0 = 0; v0 = 1;                 % excitation displacement and velocity
rp = [.15, .85; .85, .15];    % readout position as percentage on grid.

% physical parameters
E = 2e11;                       % Young's modulus
rho = 7850;                     % density (kg/m^3)
H = .005;                       % thickness (m)
L = 1;                          % plate length (m)
Lx = 1.4;                       % plate length X axis (m)
Ly = 1;                         % plate length Y axis (m)
loss = [100, 5; 1000, 4];    % loss [freq.(Hz), T60;...]

% I/O
OSR = 1;                        % Oversampling ratio
SR = 44.1e3;                    % sample rate (Hz)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Derived Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% oversample
```

```
SR = SR*OSR;                    % redefine SR by OSR

%%% Motion Coefficients
D = (E*(H)^3)/(12*(1-(nu^2)));
kappa = sqrt(D / (rho*  H) );

%%%%% Scheme Spacing
k = 1/SR;                        % time step
hmin = 2*sqrt(k*kappa);          % NSS (equation 12.5)
Nx = floor((Lx)/hmin);           % number of segments
Ny = floor((Ly)/hmin);           % number of segments
h = sqrt(Lx*Ly/(Nx*Ny));         % adjusted grid spacing
Nf = floor(SR*Tf);               % number of time steps
Nx = Nx+1;                       % grid points x
Ny = Ny+1;                       % grid points y
ss = Nx*Ny;                      % total grid size.

mu = (kappa * k)/(h^2);          % scheme parameter

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% % Loss coefficients
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if losstype ==0 % lossless

  sigma0 = 0
  sigma1 = 0;
end

if losstype ==1 % frequency independant loss

  sigma0 = 6*log(10)/loss(1,2);
  sigma1 = 0;
end

if losstype == 2 % frequency dependant loss

  z1 = 2*kappa*(2*pi*loss(1,1))/(2*kappa.^2);
  z2 = 2*kappa*(2*pi*loss(2,1))/(2*kappa.^2);

  sigma0 = 6*log(10)*(-z2/loss(1,2) + z1/loss(2,2))./(z1-z2);
  sigma1 = 6*log(10)*(1/loss(1,2) - 1/loss(2,2))./(z1-z2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% % Read In/Out
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lo = rp.*[Nx Ny;Nx Ny];
lo = floor(sub2ind([Nx Ny],lo(1), lo(3)));
```

```
94
95  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
96  % Create Force Signal
97  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
98  [X,Y] = meshgrid([1:Nx]*h, [1:Ny]*h);          % Grid of point in value
        of meters
99  dist = sqrt((X-(ctr(1)*Lx)).^2 + (Y-(ctr(2)*Ly)).^2); % distance of
        points from excitation
100 ind = sign(max(-dist+(wid*0.5),0));           % displaced grid points (
        logical)
101 rc = .5*ind.*(1+cos(2*pi*dist/wid));          % displacement
102 rc = rc(:);                                   % 2D plane as vector
103
104 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105 % Coefficient Matrices
106 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
107 BH = biharm2(Ny,Nx,bctype); % biharmonic matrix
108 LA = laplace2(Ny,Nx,bctype); % Laplacian matrix
109
110 A = (1/(1+k*sigma0))*speye(ss);   %NOTE% Currently inverted
111 B = (-(mu^2)*BH + (2*sigma1*k/(h^2))*LA + 2*speye(ss)) * A;
112 C = (-(2*sigma1*k/(h^2))*LA - (1-sigma0*k)*speye(ss))  * A;
113 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
114 %%% Initialise I/O
115 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
116
117 %%%% initialise vectors
118 u2 = u0*rc;
119 u1 = (u0+(k*v0))*rc;
120 u  = u2;
121
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
123 %%% Main Calculation
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125   for n = 1:Nf
126
127     % Main Operation
128     u = B*u1 + C*u2;
129
130     % Velocity Ouput
131     y(n,:) = (SR*(u(lo)-u1(lo)));
132
133     % Shift State
134     u2 = u1; u1 = u;
135
136   end
137
138 % EOF
```

B.3 The Coupled Plate and String

CoupledPlateAndString.m

```
%%%%% FDTD 2D Wave Model
%%%%% Matthew Hamilton s0674653
%%%%% Description:
%%%%%
%%%%% A Coupled Kirchhoff Thin Plate and Stiff String FDTD Model

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Flags
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% boundary condition type: 1: simply supported, 2: clamped
pl_bctype = 2;  % plate boundary
st_bctype = 1;  % string boundary

losstype = 1;   % loss type: 1: independant, 2: dependant
itype = 2;      % type of input: 1: struck, 2: plucked
interpJ = 1;    % J Interpolation type 0: floor, 1: linear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Simulation Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% simulation
Tf = 1;                     % duration
rp = [.15 .15; .15 .85];  % readout position as percentage on grid.

% I/O
OSR = 1;                    % Oversampling ratio
SR = 44.1e3;                % sample rate (Hz)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Physical Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Plate

pl_E  = 11e9;                   % Young's modulus
pl_rho = 480;                   % density (kg/m^3)
nu = .4;                        % Poisson Ratio (< .5)
pl_H = .005;                    % thickness (m)
pl_Lx = .8;                     % plate length X (m)
pl_Ly = .5;                     % plate length Y (m)
pl_loss = [100, 1; 1000, .75]; % loss [freq.(Hz), T60;...]
pl_ctr = [.35 .35; .71 .78];   % string coupling [Xs1,Ys1; Xs2,Ys2];

```

```
%%%% String

st_f0 = 440;                  % frequency of note in Hz (see function at
    EOF)
st_r = 3e-4;                  % string radius (m)
st_L = .67;                    % length (m)
st_E = 2e11;                  % Young's modulus (Pa) (GPa = 1e9 Pa) of
    steel
st_rho = 7850;                % density (kg/m^3) of steel
st_loss = [100, 8; 1000 7];  % loss [freq.(Hz), T60;...]

% I/O string
st_xi = 0.75;                    % coordinate of excitation (normalised,
    0-1)
st_famp = 1;                     % excitation force (Newtons)
st_dur = 0.001;                  % duration of excitation (seconds)
st_exc_st = 0;                   % start time of excitation (seconds)
st_T = (((2*st_f0.*st_r).*st_L).^2)*pi*st_rho; % Tension in Newtons


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Plate Derived Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Motion Coefficients
pl_D = (pl_E*(pl_H)^3)/(12*(1-(nu^2)));
pl_kappa = sqrt(pl_D / (pl_rho*pl_H) );

%%%%% Scheme Spacing
k = 1/SR;                              % time step
pl_hmin = 2*sqrt(k*pl_kappa);          % NSS (equation 12.5)
pl_Nx = floor((pl_Lx)/pl_hmin);           % number of segments
pl_Ny = floor((pl_Ly)/pl_hmin);           % number of segments
pl_h = sqrt(pl_Lx*pl_Ly)/sqrt(pl_Nx*pl_Ny);    % adjusted grid spacing
pl_mu = (pl_kappa * k)/(pl_h^2);     % scheme parameter
Nf = floor(SR*Tf);                     % number of time steps
pl_Nx = pl_Nx +1;
pl_Ny = pl_Ny +1;
ss = pl_Nx*pl_Ny;                       % total grid size.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% String Derived Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% derived parameters
st_A = pi*st_r^2;                         % string cross-sectional area
st_I = 0.25*pi*st_r^4;                    % string moment of intertia
st_c = sqrt(st_T/(st_rho*st_A));          % wave speed
st_K = sqrt(st_E*st_I/(st_rho*st_A));    % stiffness constant
```

```
91
92  %%%%% grid
93  st_hmin = sqrt(0.5* (st_c.^2*k^2+sqrt(st_c.^4*k^4+16*st_K.^2.*k.^2)) );
94  st_N = floor(st_L/st_hmin);          % number of segments (N+1 is number
        of grid points)
95  st_h = st_L/st_N;                    % adjusted grid spacing
96  st_lambda = st_c*k/st_h;             % Courant number
97  st_mu = st_K*k/st_h^2;               % numerical stiffness constant
98  st_N = st_N+1;                       % change st_N to be number of grid
        points
99
100 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101 %%%% % String Loss coefficients
102 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103
104 if losstype ==0  %lossless
105   st_sigma0 = 0;  st_sigma1 = 0;
106   pl_sigma0 = 0;  pl_sigma1 = 0;
107 end
108
109 if losstype ==1
110   % frequency independant loss
111   st_sigma0 = 6*log(10)/st_loss(1,2);
112   st_sigma1 = 0;
113
114   pl_sigma0 = 6*log(10)/pl_loss(1,2);
115   pl_sigma1 = 0;
116 end
117
118 if losstype == 2
119   %%% String
120   st_z1 = (-st_c.^2 + sqrt(st_c.^4 + 4*st_K.^2.*(2*pi*st_loss(1,1))^2))
          ./(2*st_K.^2);
121   st_z2 = (-st_c.^2 + sqrt(st_c.^4 + 4*st_K.^2.*(2*pi*st_loss(2,1))^2))
          ./(2*st_K.^2);
122   st_sigma0 = 6*log(10)*(-st_z2/st_loss(1,2) + st_z1/st_loss(2,2))./(
          st_z1-st_z2);
123   st_sigma1 = 6*log(10)*(1/st_loss(1,2) - 1/st_loss(2,2))./(st_z1-st_z2)
          ;
124
125   %%% Plate
126   pl_z1 = 2*pl_kappa*(2*pi*pl_loss(1,1))/(2*pl_kappa.^2); %from 1D
127   pl_z2 = 2*pl_kappa*(2*pi*pl_loss(2,1))/(2*pl_kappa.^2);
128   pl_sigma0 = 6*log(10)*(-pl_z2/pl_loss(1,2) + pl_z1/pl_loss(2,2))./(
          pl_z1-pl_z2);
129   pl_sigma1 = 6*log(10)*(1/pl_loss(1,2) - 1/pl_loss(2,2))./(pl_z1-pl_z2)
          ;
130 end
131
```

```
132 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133 %%%% % Read In/Out
134 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
135
136 lo = rp.*[pl_Nx pl_Ny];
137 lo = [floor(sub2ind([pl_Nx pl_Ny],lo(1), lo(3))), floor(sub2ind([pl_Nx
        pl_Ny],lo(2), lo(4)))];
138
139 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
140 % IV Read In/Out Error Check
141 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
142
143 % Check that xo and xi are greater than h distance away from boundaries
144 if any([(st_xi*st_L)<st_h (st_L-(st_xi*st_L))<st_h])
145   warning([sprintf('xo is too close to string boundary\n'),...
146   sprintf('changing xo to equal h\n')]);
147   st_xi = abs(round(st_xi)-st_h);
148   if li<=2
149     li = 3;
150   elseif li>=st_N
151     li = st_N-1;
152   end
153 end
154
155 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
156 % Create Force Signal
157 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
158
159 st_f = zeros(Nf,1);                          % input force signal
160 durint = floor(st_dur*SR);                   % duration of force signal,
        in samples
161 exc_st_int = (floor(st_exc_st*SR))+1;        % start time index for
        excitation
162 durf = exc_st_int:exc_st_int+durint-1;       % sample values of force
163 st_d0 = (k^2/(st_h*st_rho*st_A));            % force foefficient
164
165 %%% reevaluate relevant points in force vector
166 st_f(durf) = st_famp*0.5*(1-cos((2/itype)*pi.*(durf/durint)))*st_d0;
167
168 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
169 % Plate Coefficient Matrices
170 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
171
172 LA = laplace2(pl_Ny,pl_Nx,pl_bctype); % Laplacian Matrix
173 BH = biharm2(pl_Ny,pl_Nx,pl_bctype); % biharmonic matrix
174 pl_mI = fidimat(pl_Ny,pl_Nx,'I');
175
176 pl_mA = (1/(1+k*pl_sigma0))*speye(ss);
177 pl_mB = (-(pl_mu^2)*BH + 2*pl_mI + (2*k*pl_sigma1/pl_h^2)*LA) * pl_mA;
```

```
pl_mC = (-(1-pl_sigma0*k)*pl_mI - (2*k*pl_sigma1/pl_h^2)*LA) * pl_mA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% String Coefficients Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% scheme coefficients
Dn = fidimat(st_N,'x+');
Dnn = fidimat(st_N,'xx');
Dnnnn = fidimat(st_N,'xxxx',st_bctype);
Dnnnn(2,1) = -2;
st_mI = speye(st_N);

%%% String sparsity matrix, uncoupled.
st_mA = (1/(1+k*st_sigma0));
st_mB = ((st_lambda^2)*Dnn - (st_mu^2)*Dnnnn + (2*st_mI) + ((2*st_sigma1
    *k/st_h^2)*Dnn)) * st_mA ;
st_mC = -((1-st_sigma0*k)*st_mI + ((2*st_sigma1*k/st_h^2)*Dnn) ) * st_mA
    ;

%%%%% Alter matrices for coupling string boundary
st_mB([1 end],:) = 0;st_mC([1 end],:) = 0;
st_mB(1,:) = ((st_lambda^2)*Dn(1,:) - (st_mu^2)*Dnn(2,:) + (2*st_mI(1,:)
    )) * st_mA;
st_mC(1,:) = -((1-st_sigma0*k)*st_mI(1,:));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Couple Matrices
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%Total Number of Grid points
Nt = ss+ st_N;

%%% floored coupling points
pl_w0 = sub2ind([pl_Nx, pl_Ny],floor(pl_ctr(1,1)*pl_Nx),floor(pl_ctr
    (1,2)*pl_Ny));
pl_wL = sub2ind([pl_Nx, pl_Ny],floor(pl_ctr(2,1)*pl_Nx),floor(pl_ctr
    (2,2)*pl_Ny));

%%% Mass Ratio Coefficients
pl_Mr = 1/((pl_rho*pl_H*pl_h^2)*(1+k*pl_sigma0));
st_Mr = 1/((st_rho*st_A*st_h)*(1+k*st_sigma0));

%%% Spreading vector
if interpJ == 0
    %%% Spreading operators
  J = sparse(zeros(Nt,1)); J([pl_w0 ss+1]) = [pl_Mr -st_Mr];
  pl_J = sparse(ss,1);pl_J(pl_w0) = 1;
  st_J = sparse(st_N,1); st_J(1) = 1;
```

```
end

if interpJ == 1
  %%% Interpolation off set
  pl_ax = (pl_ctr(1,1)*pl_Nx) - floor(pl_ctr(1,1)*pl_Ny);
  pl_ay = (pl_ctr(1,1)*pl_Nx) - floor(pl_ctr(1,2)*pl_Ny);

  %%% Linear interpolation indeces
  lii = [pl_w0, pl_w0+1, pl_w0+pl_Ny,pl_w0+pl_Ny+1];
  %%% Linear interpolation coeffcients
  lic = [(1-pl_ax)*(1-pl_ay),(1-pl_ax)*pl_ay,pl_ax*(1-pl_ay),pl_ax*pl_ay
      ];

  %%% Spreading operators
  pl_J = sparse(ss,1); pl_J(lii) = lic;
  st_J = sparse(st_N,1); st_J(1) = 1;
  J = sparse(Nt,1); J([lii, ss+1]) = [pl_Mr*lic, -st_Mr];
end

%%% Mass ratio
Mr = inv( 1/(st_rho*st_A*st_h*(1+k*st_sigma0)) +...
      (1/(pl_rho*pl_H*(pl_h^2)*(1+k*pl_sigma0)))*(pl_J'*pl_J));

%%% Coupling Vector
F = Mr*[-pl_J'*pl_mB,st_J'*st_mB];        % coupling vector for B matrix
F1 = Mr*[pl_J'*(2*pl_mI + (2*pl_sigma1*k/pl_h^2)*LA)/(1+k*pl_sigma0),...
        -2*st_J'*st_mI/(1+k*st_sigma0)];   % coupling vector for C
            matrix

%%% Main Sparisty Matrices
B = blkdiag(pl_mB,st_mB) + J*F;
C = blkdiag(pl_mC,st_mC) + J*F1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Initialise I/O
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% initialise output
u = zeros(ss,1);
w = zeros(st_N,1);

%%% Joined vectors
uw = [u;w];
uw1 = uw;
uw2 = uw;

%%%% input
fvect = [u;w];                % force vector
li = floor(st_xi*st_N) + ss;  % grid index of excitation
```

```
269
270 %%%% output
271 y = zeros(Nf,2);
272
273 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
274 %%% Main Calculation
275 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
276 disp('All variables initialised')
277
278 %%%%% Main loop
279 tic
280 if run
281
282   for n = 1:Nf
283
284     %%% update input forces
285     fvect(li) = st_f(n);
286
287     %%% main operation
288     uw = B*uw1 + C*uw2 + fvect;
289
290     % read output
291     if (outtype==1)
292       y(n,:) = uw(lo);
293
294     elseif (outtype==2)
295       y(n,:) = (SR*(uw(lo)-uw1(lo)));
296
297     end
298
299     %%% shift state
300     uw2 = uw1; uw1 = uw;
301
302   end
303
304 end
305 toc
306
307 % EOF
```

Appendix C

Project Archive

This appendix contains a list of the files included in the archive, how they are organised and operated as well as an explanation of any example files. It is worth noting that all files were created and tested in MATLAB r2016b running on macOS 10.11 El Capitan. Care has been taken to ensure that each script should run, without alteration on any machine. With changes in behaviour between versions of MATLAB it is difficult to state, outside of the configuration above, that the scripts will run without fault. A folder of older scripts that were created throughout this project have been included for completeness. They are located in the `Old Scripts` folder and should hopefully illustrate how the coding has progressed.

C.1 Audio

Some audio files have been provided to give an indication of the quality of output from each script. Each audio file is the MIDI file and the script that was used to generate the file. All audio files were created with schemes run at 44.1 kHz unless stated otherwise in the filename. MIDI files are also provided in the MIDI folder.

C.2 File Structure

Files relating to an FDTD scheme have the same structure. The first section refers to an 'instrument file', a file where all parameters relating to the string and the plate, are set. These instrument files also contain any flags relating to how the scheme is run, such as whether it is plotted, analysis is run, the output style and boundary conditions. Each of these should be clearly commented. There should be no reason to change any of the code contained within the main files. Each script has been set to run generating a mono or stereo audio vector y. Scripts generating output from MIDI data reference midi files contained within the folder `/midi`. All MIDI based scripts and the `stringVerb.m` script produce a sound file containing the script named and the file used to generate

audio. A progress has been implemented in the more process intensive scripts to give some indication of time left to compute.

To run any script the basic flow is:-

Set Parameters in Instrument File $\longrightarrow$ Run Main Script

By default the MIDI based script have been set to run using the `chord.mid` file. This is a single D major chord and provides the best example of model output quality for the shortest computation time. All files that are runnable are on the top level of the folder as are the relevant functions that are required.

C.3 Naming Convention

All files follow roughly the same naming convention as well as logical structure. Over the course of the project, naming of some sections has changed however, along with the code comments, it should be clear where these changes have taken place.

Variables relating to the string(s) are prefixed '`st_`' and plate variables with '`pl_`'. Other changes that have occurred are particular to the coupling force vector which was named as `cpl_v` and `cpl_vc` for current and previous time steps. With the introduction of Interpolation in later code, this has simply been changed to `F` and `F1` respectively.

C.4 Force

For the code that generates a force vector through MIDI files, a variable force system was implemented. Using [13] as a guide. The velocity data was scaled to between 1 and 40 newtons using a simple linear scaling function.

$$f(x) = \frac{(d-c)(x-a)}{b-a} + c, \quad a \leq x \leq b, \quad c \leq f(x) \leq d$$

The same process was taken to then create a linear relationship between a force in newtons $1 \leq F \leq 40$ and a duration in milliseconds $1 \leq t \leq 2.5$.

$$t = \frac{(2.5-1)(F-1)}{40-1} + 1$$

This appears to have the greatest effect on the strike model (raised cosine). The pluck (half cosine) on the other sounds best at over 4ms excitation duration. Any variation after that isn't really noticeable. This is the mode of generating force in all MIDI based scripts.

C.5 Stability

Stability conditions, in particular the grid spacing, for each scheme required altering with the addition of stiffness in the string. The addition of frequency independent loss was first altered without any change the grid spacing. This has been for the most part unproblematic. With the more intricate such as the Piano models, `multiStringGrandPiano.m` and `pianoModel_Interp.m`, the potential for instability to arise was a little more likely. Sadly, that addition of stability came quite late in the process and all credit has to go to [6] as covered in section 3.3.2.

C.6 Calibration

The results of calibrating the string model against piano recordings are contained in the files. `GrandPiano.m` and `multiStringGrandPiano.m`. The process is covered in section 4.2. Sadly, the strings were still attached to a sound board with no real means of disconnecting the two. It would be interesting to record a completely isolated set of piano strings but this was not feasible during this project. The values of the calibration files still off at least some greater resemblance to a physical instrument.

C.7 File Breakdown

This section is just a brief overview of each file, its functionality and how it fits into the greater picture of the whole project

accrue.m

There is no simple MATLAB function that will accumulate the values of a vector. When considering the number of grid points in each system and linear indexing, this would be a handy function to have. the `accrue.m` script does just that. It is only used in the `multiStringPianoModel` but it is integral for that script to function.

biharm2.m

The second incarnation of a 2D FDTD biharmonic sparse matrix generator. The code has also been included in Appendix B.1.1 as it is referenced throughout the paper. Should provide some details on use when typing into MATLAB command window:`help biharm2`

fidimat.m

The `fiditmat.m` function was created for more general FDTD use. It is sadly limited to only 1D and 2D and only clamped and simply supported boundary conditions. A

help file should be available when entering `help fidimat`

GrandPiano.m

The `GrandPiano.m` contains the parameters for the string derived from calibrating with piano string recordings. It is exclusively called in the `pianoModel.m` script.

instPiano.m

This is the instrument file for `pianoModel.m` and where the parameters for that script can be edited. Getting the right values of the MIDI data to relate to piano strings is somewhat inelegant. I would advise away from editing this area (line 40) though it does provide transposition for input.

laplace2.m

`laplace2.m` functions in exactly the same as `biharm2.m` except it creates a laplacian FDTD sparse matrix.

multiStringGrandPiano.m

This file is the `multiStringPianoModel.m` string calibration. Given the way that the tiering of multiple strings was achieved, the indexing for string parameters needed to altered.

multiStringPianoModel.m

This is a multi-strung version of the `PianoModel.m`. Rather than having one string per note it mimics the piano in having one string for the bass notes, 2 strings for the middle range and the remaining strings with 3 strings per note, giving a grand total 225 strings being modelled. The strings have been added by cell arrays given the uneven spread across the rest of the strings. This in retrospect has become more cumbersome then helpful. Upon creating this script again it would likely include all string variables in a single vector.This script was creating at a time when interpolation had yet to be implemented. As such it is limited in to a minimum size to accommodate the number of strings. Like the other MIDI based scripts it has been presented in state to play the `chord.mid` file. Typically 18000 grid points are calculated. Computation will likely take some time.

multiStringPianoVars.m

The variables used for the `multiStringPianoModel.m` script. follows the same rules as other instrument files though is sadly more cluttered at the expense of including extra

strings. Extra variables to consider are `TT` which changes the what the tension should be by the percentage given.

note2hz

This function takes a cell array of notes in scientific notation and converts it to a vector of values in Hz. This is used in the `stringVerb.m` script as well as the `plate_and_string_vars.m` instrument file.

PianoModel.m

The `PianoModel.m` seeks to imitate the make-up of a grand piano. It uses the `instPiano` instrument file to get parameters and also generates audio from a MIDI file. It will simulate 88 strings evenly spaced across the y-axis of a plate. The spacing is dictated by the size of the plate in the `pl_ctr`. The depth the strings are set into the x-axis is set by `pl_rx` The visualisation has been difficult to get correct and is not very intuitive. The plotting creates an image of the plate with markers indicating the coupling points of the string. Typical run times are 3 times slower than real-time. Plate size will reduce this but the strings tend to be the largest burden at 5000 grid points.

plate_analysis.m

This provides some plots of energy and modes against the scheme that was run. This was used fairly earl in the process to ensure the plate was behaving as intended.

plate_and_string_vars.m

The instrument file for the `thin_plate_stiff_string_loss.m` script

PlateStringMIDI

The instrument file for the `thin_plate_stiff_string_MIDI.m` script.

plateVerbVars.m

The instrument file for the `thin_plate_reverb.m` script. Contains reference to variables not used throughout the script but that were used in earlier implementations. These can be scene in older script folder provided in the digital archive.

read_midi_file.m

This function was written for a previous project as a means of importing MIDI data into a variable in MATLAB. To an extent it should be treated as a black box and should not be altered.

reverbParams.m

This is the instrument file for the `stringVerb.m` effect. It contains mostly the same parameters of other instrument files with the inclusion of choosing which notes to simulate.

reverbStringPlate.m

An implementation of the reverb effect discussed in 4.3. Audio is fed into the plate model and read out from the strings. Constant power panning is applied to all string running from left to right of the `tuning` variable. Like `multiStringPianoModel.m`, the coupling is achieved without interpolation. As such the number of strings that can be attached is limited to the number of grid points in the y-axis. An error check is in place to warn when this condition arises

thin_plate_loss_xy.m

A lossy Kirschhoff thin plate FDTD model. This script is self contained, with no instrument files. The primary purpose was to produce visualisations though a vector y still reads an output point on the plate. playback is not automatic.

thin_plate_reverb.m

A plate reverb effect, feed an audio file into a plate model and reads out at another position.

thin_plate_stiff_string_loss.m

A single string connected to a thin plate model. This was the early prototype for coupling and later used to generate visualisations. Plays output when the `play_on` flag is `true`.

thin_plate_stiff_string_MIDI.m

Multiple strings are connected to a plate. Unlike the piano model, only strings that play a note are simulated.

C.8 File List

`accrue.m`
`Audio`
`biharm2.m`
`fidimat.m`

```
GrandPiano.m
instPiano.m
laplace2.m
midi
multiStringGrandPiano.m
multiStringPianoModel.m
multiStringPianoVars.m
note2hz.m
OldSchemes
pianoModel.m
plate_analysis.m
plate_and_string_vars.m
plate_string_analysis.m
PlateStringMIDI.m
plateVerbVars.m
read_midi_file.m
reverbParams.m
reverbStringPlate.m
thin_plate_loss_xy.m
thin_plate_reverb.m
thin_plate_stiff_string_loss.m
thin_plate_stiff_string_MIDI.m
```

Bibliography

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

[2] C. de la Musique Philharmonie De Paris. (2017, August). [Online]. Available: http://collectionsdumusee.philharmoniedeparis.fr/image.ashx?q=http://www.mimo-db.eu/media/CM/IMAGE/CMIM000022692.jpg

[3] T. D. Rossing and N. H. Fletcher, "Principles of vibration and sound," 2004.

[4] P. M. Morse and K. U. Ingard, *Theoretical acoustics*. Princeton university press, 1968.

[5] C. Gough, "Violin plate modes," *The Journal of the Acoustical Society of America*, vol. 137, no. 1, January 2015.

[6] A. Torin, "Percussion instrument modelling in 3d: Sound synthesis through time domain numerical simulation," Ph.D. dissertation, The University of Edinburgh, 2015.

[7] A. W. Leissa, "Vibration of plates - nasa-sp-160," NASA, Tech. Rep., January 1969. [Online]. Available: http://hdl.handle.net/2060/19700009156

[8] C. Roads, *The computer music tutorial*. Cambridge, Mass.: MIT Press, 1995.

[9] F. P. Laboratory, *Wood handbook : wood as an engineering material.*, ser. General technical report FPL ; 113. Madison, Wis.: U.S. Dept. of Agriculture, Forest Service, Forest Products Laboratory, 1999.

[10] I. Brémaud, J. Gril, and B. Thibaut, "Anisotropy of wood vibrational properties: dependence on grain angle and review of literature data," *Wood Science and Technology*, vol. 45, no. 4, pp. 735–754, 2011.

[11] P. Carr, *Frank Zappa and the And*. Routledge, 2016.

[12] S. Bilbao and A. Torin, "Numerical modeling and sound synthesis for articulated string/fretboard interactions," *Journal of the Audio Engineering Society*, vol. 63, no. 5, pp. 336–347, 2015.

[13] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. ii. comparisons with measurements and systematic exploration of some hammer-string parameters," *The Journal of the Acoustical Society of America*, vol. 95, no. 3, pp. 1631–1640, 1994.