Physical Modelling Of Spring Reverberation

Project Report

Marius George Onofrei

Aalborg University
IT and Design

IT and Design
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Physical Modelling Of Spring Reverberation

Theme:
Physical Models for Sound Synthesis

Project Period:
Spring Semester 2020

Project Group:
Group 12

Participant(s):
Marius George Onofrei

Supervisor(s):
Silvin Willemsen

Copies: 1

Page Numbers: 56

Date of Completion:
May 28, 2020

Abstract:

The sound of spring reverberation is one that is deeply ingrained in popular culture. This is mainly due to the compact size and affordability of spring reverb units, which have been included in many guitar amplifiers throughout the late 20st century. Hence a lot of music was "colored" throughout recent history with this distinct sound. While analog spring reverb units are still quite popular, they somewhat plagued by the inflexibility, as a unit will typically have only one type of response. One cannot easily change the spring in a unit nor change its physical properties.
Therefore, this project aims to implement a numerical scheme capable of producing realistic helical spring impulse responses, which can then be used to add the reverberation effect to an input sound by means of convolution. The main method used throughout the project is the finite difference method (FDM), which is employed to model a range of increasingly complex acoustic systems, forming a building block for the eventual helical spring implementation.

Contents

Preface

Aalborg University, May 28, 2020

Marius George Onofrei
<monofr11@student.aau.dk>

Chapter 1

Introduction

Ever since the inception of digital sound synthesis there has been an aim for reproducing the sounds made by classic analog instruments and effects. Purists of the analog sound will undoubtedly ask "Why do this when you already have the originals?", but one may argue that it is within human nature to be curious and explore the limits of new technology. So a better question is: "Why not?"..

1.1 Spring Reverberation

Spring reverbs have been around since the 1940s, [8] and have been developed as cheap, compact devices which give the illusion of room-reverberation. Due to their small size, spring reverbs have been included in classic guitar amplifiers throughout the late 20^{th} century and therefore this sound is deeply ingrained in popular culture, whether we are aware of it or not.

However, the sound produced by spring reverberation is distinctly different than that of an acoustic space, which it originally intended to replicate. The quality of echoes produced by springs are more prominent and repetitive and the sounds seems more "spread out". This is due the dispersive nature of wave propagation through helical springs, typically used in spring reverb units. This report will put a considerable focus on this dispersive quality, giving an overview of how it affects the response of various acoustic systems of increasing complexity, shown in Chapter 3, leading up to the model of a helical spring in Chapter 4.

Figure 1.1 shows a spring reverberation unit, together with an associated schematic for it. The springs are suspended in the casing between two wires. Between these wires and the casing lie some magnetic beads which will be excited by an adjacent electromagnetic coil through which the electrical input signal,i.e., the sound, is passed. The motion of these magnets will vibrate the springs, which will in turn displace another pair of magnetic beads at the other end of the springs. Finally these magnetic beads will generate some current in the "output" electromagnetic

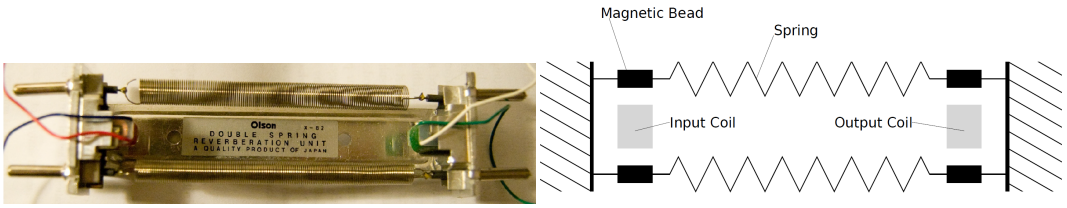


Figure 1.1: (a) a typical spring reverberation unit [15] and (b) a sketch of this unit, illustrating its crucial components, [16].

coil, which will then be passed to an amplifier and eventually an output source, or speakers.

1.2 Physical Modelling

The idea behind physical modelling is to describe a target system, in this case the spring reverb unit, which is an analog electronic device, via a physical model. That is a system of mathematical equations which describe the input ⇒ system dynamics ⇒ output chain, cf. [5]. Starting from this set of equations, there are a number of directions one can go towards for "solving" this problem, i.e. calculating the output state as a response to some input state. Perhaps the most direct approach is doing a discrete time simulation of this physical system. One method for doing this is the finite difference method (FDM), which has an entire chapter dedicated to it in this report: Chapter 2, albeit with a focus on the application of finite difference schemes (FDSs) to sound synthesis, even though it can be used in a much more general context.

Other approaches starting from the mathematical description of the physical system are modal synthesis and digital waveguides. In the first, the modes of vibration of the system can be computed and the response of a system to some excitation will then be a combination of these modes, with different weight factors. The second makes use of decomposing the solution to the equations into travelling waves which can then be modelled as digital delay lines. Although more efficient in terms of implementations than the FDM, these two approaches are limited to more simple systems. For modal synthesis is only possible if a closed-form solution is available for the mode shapes of the system, otherwise numerical methods for calculating the modes would strip the method of its efficiency. Similarly, digital waveguides are difficult to implement for systems with complex dispersion relationships, where waves of the same frequency can have multiple propagation velocities, as is the case for the helical spring, further described in Chapter 4.

1.3 State Of The Art

There are limitless spring reverberation plugins and digital devices available on the market. However, it is the author's personal opinion that many are spring reverbs only in name, using various tricks to replicate to some extent the original sound. One plugin that is based on a physical model of helical springs is the PA4 Dual Spring Reverb released by Physical Audio [13]. It is in fact based on a paper which is more advanced than the implementation used in this report, that is Bilbao's REF (Bilbao Numerical Simulation Of Spring Reverberation). How they tackle some of the issues which will be discussed in this project is unfortunately unknown, but all good products need to have a "secret sauce".

1.4 Project Goals

The main goal of this project is to implement a FDS for simulating the response of a helical spring. The work is mainly based on [2] and [4]. This implementation is described in Chapter 4. In order to reach this goal, the author goes through FDS implementations for a number of different acoustic systems of increasing complexity, which are presented in Chapter 3. Potential uses of the helical spring numerical model are discussed in Chapter 5. Finally, a conclusion and notes on future work are given in Chapter 6.

Chapter 2

Finite Difference Method

Many physical phenomena can be mathematically described by describing the rates of change of certain variables of the considered process such as time or spatial position for instance. These mathematical formulations are represented by ordinary differential equations (ODEs) and partial differential equations (PDEs), which are equations that relate one or more functions and their derivatives with the difference between them being that ODEs contain single-variable functions while PDEs contain multi-variable functions. For example the displacement of a vibrating string, u, is dependent on the time variable, denoted as t and the position along the string, denoted as x. This means that $u = u(x, t)$ is a multi-variable function.

Often these systems of differential equations which describe the considered physical phenomena do not have a closed form solution meaning they cannot be solved analytically. However, there are certain techniques that can convert these ODEs or PDEs to systems of equations which can be solved by linear algebra techniques, which are furthermore well suited for computer processing. These are called numerical analysis techniques, one of which is the Finite Difference Method, where discretization techniques are employed in order to approximate continuous systems into grid points and thus compute an approximate numerical solution to the differential equations.

This chapter will give an overview of the building blocks of the finite difference method, and present finite difference schemes (FDSs) with respect to their application to physical modeling sound synthesis.

2.1 Terminology and Notation

The notation which is used throughout this report is taken from [4] and is somewhat of a hybrid between traditional notations used in the general science field of physical simulation and the notation used by audio and electrical engineers, with respect to the signal processing side.

Continuous time and space partial differential equations (of 1st and 2nd order as exemplified here - higher orders follow the same pattern) will be written in the following subscript notation:

$$\frac{\partial u}{\partial t} = u_t, \quad \frac{\partial^2 u}{\partial t^2} = u_{tt}, \quad \frac{\partial u}{\partial x} = u_x, \quad \frac{\partial^2 u}{\partial x^2} = u_{xx}, \quad \frac{\partial^2 u}{\partial x \partial t} = u_{xt}, \tag{2.1}$$

As an example, one can rewrite the traditional damped mass-spring equation (derived from the famous Newton's 2nd law) in the following way:

$$m\frac{d^2u}{dt^2} = -Ku - R\frac{du}{dt}, as \tag{2.2a}$$

$$mu_{tt} = -Ku - Ru_t, \tag{2.2b}$$

where u represents the displacement of a mass, m, attached to a spring with a stiffness K and with a system damping coefficient of R, as illustrated in Figure 2.1.

In the case of the mass-spring system, the 1-D displacement of the mass, u is only a function of time t, meaning that Equation 2.2b is an ODE, but FDSs as previously stated are more flexible and can tackle PDEs as well as is the case for the 1-D wave equation, which will be described in more detail in the next chapter. By introducing the concept of grid functions, which allow the discretization of the continuous time-space domain, a multi-variable function $u = u(t, x)$ can be approximated as u_l^n across a time-space grid of $x = lh$ and $t = nk$, where h is the spatial step and k is the time step, with l and n being integers counting the number of steps in space and time, i.e. $l \in [0, N_x]$ and $n \in [0, N_t]$. This is illustrated in Figure 2.2, where it can be seen that u_l^{n+1} is the value of the function at the "next" time step compared to the "current" time step u_l^n and u_l^{n-1} is the value at the "previous" time step. Similarly u_{l+1}^n and u_{l-1}^n are the values of the function at the "next" and "previous" spatial point. Of course, with this notation one can go as

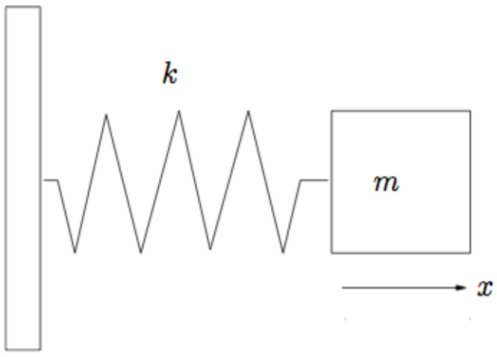


Figure 2.1: Sketch of a simple mass-spring system. [18]

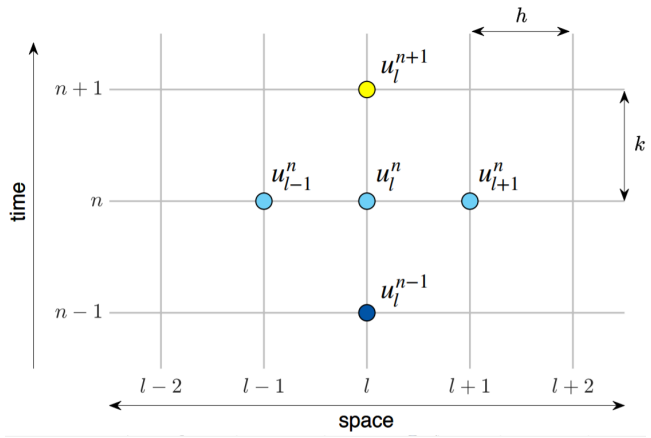


Figure 2.2: Illustration of the discretization of a continuous function $u = u(t, x)$ as a grid function, u_l^n. [18]

forward and backward as one pleases. Typically for physical simulations for sound synthesis, k is chosen as to obtain a desired sampling frequency f_s, thus $k = 1/f_s$.

Furthermore, based on such a grid function configuration, the idea of operators can be introduced, which denote actions which can be applied to the discrete function u_l^n. The most fundamental of these actions are shifts, which can be forward or backward, or stationary (described as the identity operation "1"):

$$\begin{aligned} e_{t+}u_l^n &= u_l^{n+1} \quad e_{t-}u_l^n = u_l^{n-1} \quad 1u_l^n = u_l^n \\ e_{x+}u_l^n &= u_{l+1}^n \quad e_{x-}u_l^n = u_{l-1}^n \quad 1u_l^n = u_l^n \end{aligned} \tag{2.3}$$

These shift operators now form the building blocks for the very useful difference operators, shown in equations 2.4 which allow for the formulation of discrete approximations to continuous derivatives as well as the averaging operators, shown in equations 2.5 which are highly useful in constructing more complex and accurate FDSs.

$$\delta_{t+} \triangleq \frac{1}{k}(e_{t+} - 1) \approx \frac{d}{dt} \tag{2.4a}$$

$$\delta_{t-} \triangleq \frac{1}{k}(1 - e_{t-}) \approx \frac{d}{dt} \tag{2.4b}$$

$$\delta_{t\cdot} \triangleq \frac{1}{2k}(e_{t+} - e_{t-}) \approx \frac{d}{dt} \tag{2.4c}$$

$$\mu_{t+} = \frac{1}{2}\left(e_{t+} + 1\right) \approx 1 \tag{2.5a}$$

$$\mu_{t-} = \frac{1}{2}\left(1 + e_{t-}\right) \approx 1 \tag{2.5b}$$

$$\mu_{t\cdot} = \frac{1}{2}\left(e_{t+} + e_{t-}\right) \approx 1 \tag{2.5c}$$

Note that for the case of spatial difference operators the considered step to be used in the denominator will be h, i.e. the discrete spatial grid step, as opposed to k, which is the time step.

Using the difference operators, it is possible to describe the partial derivatives of the continuous function $u = u\left(t, x\right)$ in the following way:

$$\frac{\partial u}{\partial t} = u_t \approx \begin{cases} \delta_{t+} u_l^n = \frac{1}{k}\left(u_l^{n+1} - u_l^n\right) & \text{Forward time difference} \\ \delta_{t-} u_l^n = \frac{1}{k}\left(u_l^n - u_l^{n-1}\right) & \text{Backward time difference} \\ \delta_{t\cdot} u_l^n = \frac{1}{2k}\left(u_l^{n+1} - u_l^{n-1}\right) & \text{Centered time difference} \end{cases} \tag{2.6}$$

$$\frac{\partial u}{\partial x} = u_x \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h}\left(u_{l+1}^n - u_l^n\right) & \text{Forward difference in space} \\ \delta_{x-} u_l^n = \frac{1}{h}\left(u_l^n - u_{l-1}^n\right) & \text{Backward difference in space} \\ \delta_{x\cdot} u_l^n = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right) & \text{Centered difference in space} \end{cases} \tag{2.7}$$

The simple difference and averaging operators shown in 2.4 and 2.5 can be combined to form limitless types of approximations or difference schemes for continuous ODEs. For example, the 2nd order derivative can be obtained by applying the operators δ_{t+} and δ_{t-}. The order in which one applies them to the grid function u_l^n is not important.

$$\delta_{tt} = \delta_{t+}\delta_{t-} = \delta_{t-}\delta_{t+} = \frac{1}{k^2}\left(e_{t+} - 2 + e_{t-}\right) \approx \frac{d^2}{dt^2} \tag{2.8}$$

The constant "2" should be thought as twice the identity operation, see Equation 2.3. It is perhaps better understood when actually applying the δ_{tt} operator to the grid function u_l^n:

$$\frac{\partial^2 u}{\partial t^2} = u_{tt} \approx \delta_{tt} u_l^n = \delta_{t+}\delta_{t-} u_l^n = \frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) \tag{2.9a}$$

$$\frac{\partial^2 u}{\partial x^2} = u_{xx} \approx \delta_{xx} u_l^n = \delta_{x+}\delta_{x-} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \tag{2.9b}$$

The number of dimensions of the grid function can of course be extended, for example when there is an aim to describe the sound pressure propagation

throughout a 3-D room. The notation of the grid function will be the following: $u^n_{l,m,p}$ where the x, y and z directions are discretized as $x = lh_x$, $y = mh_y$, $z = ph_z$. The spatial step in the 3 spatial directions needs not necessarily be the same, hence the different subscripts for the h parameters.

2.2 Damped Mass Spring FDS

This section revisits Equation 2.2b describing the 1-D motion of a mass, m, attached to a linear-elastic spring, with a stiffness K and a system damping R, and expands it in order to get an update equation for calculating the 1-D displacement u, at a future time step, i.e., solve for u^{n+1}_l.

It was seen that u_t can be approximated using one of the difference equations in 2.6. For this example it is first chosen to use the forward time difference, $\frac{1}{k}\left(u^{n+1}_l - u^n_l\right)$. Furthermore, $u_{tt} = \frac{1}{k^2}\left(u^{n+1}_l - 2u^n_l + u^{n-1}_l\right)$ as per Equation 2.9a. Using this we can rewrite Equation 2.2b as:

$$m\frac{1}{k^2}\left(u^{n+1}_l - 2u^n_l + u^{n-1}_l\right) = -Ku^n_l - R\frac{1}{k}\left(u^{n+1}_l - u^n_l\right), \tag{2.10}$$

and from this extract the update equation for u^{n+1}_l:

$$u^{n+1}_l = \frac{-Kk^2u^n_l + Rku^n_l + 2mu^n_l - mu^{n-1}_l}{Rk + m} \tag{2.11}$$

One may choose to use another time difference approximation from the ones given in Equation 2.2b, for instance using the centered difference approximation, the update equation becomes:

$$u^{n+1}_l = \frac{-2Kk^2u^n_l + Rku^{n-1}_l + 4mu^n_l - 2mu^{n-1}_l}{Rk + 2m} \tag{2.12}$$

The choice influences the degree of accuracy of the approximation of the mass-spring PDE solution, with the centered difference approximation being 2^{nd} order accurate compared to the 1^{st} order accuracy of the forward and backward difference approximations. The accuracy of the difference operators gives an incomplete picture of the accuracy of the finite difference scheme, since although each operator has its own degree of accuracy, their combination when used in a FDS may result in another accuracy level of the PDE solution. A detailed description is given in [4] in Section 2.2.3 and furthermore in Section 3.3.2.

An example of a FDS solution for a mass-spring system with $K = 5E6$ N/m, $m = 30$ kg without any damping, $R = 0$ kg/s will be shown in the following as compared with the analytical, closed-form solution given by:

$$u(t) = A\cos(\omega_0 t) + B\sin(\omega_0 t), \tag{2.13}$$

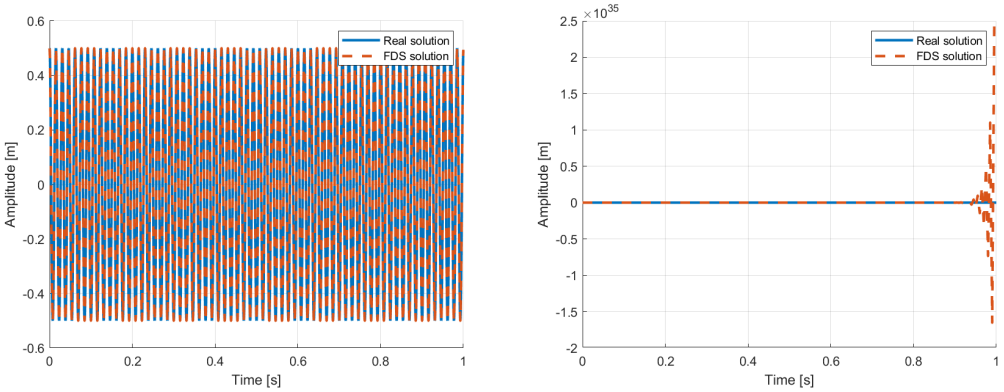
where the parameters A and B are related to the initial displacement u_0, initial velocity v_0 and the angular frequency of oscillation, ω_0,

$$A = u_0, \quad B = v_0/\omega_0 \tag{2.14}$$

Figure 2.3 shows comparisons of the real solution of the oscillator's displacement compared to the FDS solution based on the update equation shown in Equation 2.12. The time step, k, considered in the simulation shown in the left of the figure is based on a desired sampling frequency, f_s, of 44100 Hz, thus $k = 1/f_s = 2.2676 \cdot 10^{-5}$ s. In the simulation shown in the right side of the figure, the time step $k = 0.005$ s. It can be seen that when a very small time step is used the FDS solution is highly accurate, but in the second case (to the right), the numerical solution becomes unstable and explodes.

In fact, stability analysis represents a vital part of all numerical methods. Typically used for such analysis are frequency domain methods and energy methods, which will not be described in detail here. However, the reader is invited to the exhaustive overview given in [4]. Using such methods for the case of the undamped mass-spring system, one finds that the condition on k for numerical stability is: $k < \frac{2}{\omega_0}$.

Figure 2.4 shows a FDS solution to the same system but with a damping coefficient $R = 100$ kg/s simulated for a total of 5 seconds with an f_s = 44100 Hz.



(a) $k = 2.2676E - 5$. **(b)** $k = 0.005$.

Figure 2.3: Comparison of FDS solution for the displacement u of mass-spring system with closed-form solution.

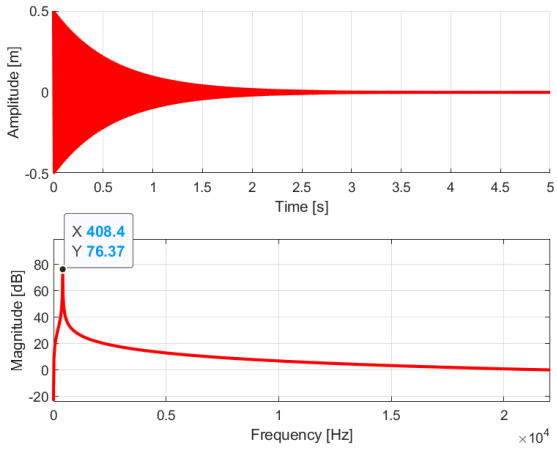


Figure 2.4: An FDS solution to damped mass spring system.

2.3 FDS Identities

The operators described in this section can be combined and rewritten in various ways. Below are some examples which may be used throughout this report:

$$\begin{aligned}
\delta_{tt} &= \frac{2}{k}(\delta_{t\cdot} - \delta_t) \\
\delta_{tt} &= \frac{1}{k^2}(e_{t+} - 2 + e_{t-}) \\
\delta_{t\cdot} &= \frac{1}{2k}(e_{t+} - e_{t-}) \\
\mu_{t,\alpha} &= \alpha + \frac{(1-\alpha)}{2}(e_{t+} + e_{t-}) \\
\delta_{xx} &= \delta_{x+}\delta_{x-} \\
\delta_{xxxx} &= \delta_{xx}\delta_{xx},
\end{aligned} \tag{2.15}$$

2.4 Matrix Operators

A more compact form of these operators which is needed in the case of implicit FDSs, is the matrix form. The values of a grid function u_l^n of infinite length can be written as a column vector of the form: $\mathbf{u}^n = [..., u_{-1}, u_0, u_1, ...]^T$, where T represents the transpose operations. Then the operators δ_{x-}, δ_{x+}, δ_{xx} and δ_{xxxx} can be written in matrix form as:

$$\mathbf{D}_{x-} = \frac{1}{h}\begin{bmatrix} \ddots & & & & & \\ \ddots & 1 & & & \mathbf{0} & \\ & -1 & 1 & & & \\ & & -1 & 1 & & \\ & \mathbf{0} & & -1 & 1 & \\ & & & & \ddots & \ddots \end{bmatrix} \quad \mathbf{D}_{x+} = \frac{1}{h}\begin{bmatrix} \ddots & \ddots & & & & \\ & -1 & 1 & & \mathbf{0} & \\ & & -1 & 1 & & \\ & & & -1 & 1 & \\ & \mathbf{0} & & & -1 & \ddots \\ & & & & & \ddots \end{bmatrix}$$

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} \ddots & \ddots & & & & \\ \ddots & -2 & 1 & & \mathbf{0} & \\ & 1 & -2 & 1 & & \\ & & 1 & -2 & 1 & \\ & \mathbf{0} & & 1 & -2 & \ddots \\ & & & & \ddots & \ddots \end{bmatrix} \quad \mathbf{D}_{xxxx} = \frac{1}{h^4}\begin{bmatrix} \ddots & \ddots & \ddots & & & \mathbf{0} \\ \ddots & 6 & -4 & 1 & & \\ \ddots & -4 & 6 & -4 & 1 & \\ & 1 & -4 & 6 & -4 & \ddots \\ & & 1 & -4 & 6 & \ddots \\ \mathbf{0} & & & \ddots & \ddots & \ddots \end{bmatrix} \tag{2.16}$$

Furthermore the averaging operators μ_{x-}, μ_{x+} and $\mu_{x\cdot}$ can be written as:

$$\mathbf{M}_{x-} = \frac{1}{2}\begin{bmatrix} \ddots & & & & \\ \ddots & 1 & & \mathbf{0} & \\ & 1 & 1 & & \\ & & 1 & 1 & \\ & \mathbf{0} & & 1 & \ddots \\ & & & \ddots & \ddots \end{bmatrix} \quad \mathbf{M}_{x+} = \frac{1}{2}\begin{bmatrix} \ddots & \ddots & & & \\ & 1 & 1 & \mathbf{0} & \\ & & 1 & 1 & \\ & & & 1 & \ddots \\ & \mathbf{0} & & & \ddots \end{bmatrix}$$

$$\mathbf{M}_{x\cdot} = \frac{1}{2}\begin{bmatrix} \ddots & \ddots & & & \\ \ddots & 0 & 1 & \mathbf{0} & \\ & 1 & 0 & 1 & \\ & & 1 & 0 & \ddots \\ & \mathbf{0} & & 1 & \ddots \\ & & & \ddots & \ddots \end{bmatrix} \tag{2.17}$$

It is important to keep in mind that for real world problems such infinite representations of the matrices is not appropriate, as the simulations will be carried out on a finite number of grid points. In such a case, boundary conditions will come into play, usually affecting the above matrices in the extreme rows and columns.

Chapter 3

Numerical Simulations of 1-D Acoustic Systems

This chapter will describe implementations of numerical simulations via the finite difference method of different acoustic physical systems. Each section will focus on a system of increasing degree of complexity, as a prelude to the numerical model of the helical spring, presented in detail in the next chapter and which is the foundation for the physical modelling of spring reverberation.

The first section focusing on the 1-D wave equation will be described in more detail, as it introduces important concepts related to finite difference methods, such as stability analysis, energy analysis, dispersion of the physical system and numerical dispersion. Furthermore, a distinction between explicit and implicit finite difference schemes will be shown in Section 3.2 and their different advantages and disadvantages will be discussed. The last section will move from the idealized cases illustrated in the first two sections and show something of a real-life application of the ideal bar model: a slinky. This is basically a very flexible and large helical spring (compared to the smaller versions used in spring reverberation units) and is modelled here as an ideal bar with a very small stiffness and additional loss parameters which control the loss characteristics of the system.

Most of the theory presented in this chapter is based on [4].

Furthermore, scripts with FDS implementations of each of the systems described here can be found in [12], together with sound output examples.

3.1 The 1-D Wave Equation

The 1-D wave equation is a traditional test problem in physics, one of the reasons being that it is the rare example of a PDE that admits an exact numerical solution. Furthermore it is particularly important in musical acoustics as it is can be used to model the behavior of strings or blowing tubes, albeit only idealized versions of

them. Using the notation presented in the previous chapter, it is defined as:

$$u_{tt} = c^2 u_{xx} \tag{3.1}$$

where $u = u(x,t)$ is the transverse displacement along of the string in time, t, along its length, x and thus Equation 3.1 is a 2^{nd} order PDE.

For the case where it is used to model an idealized string under low-amplitude conditions, the parameter $c = \sqrt{T_0/\rho A}$, where T_0 is the string tension, ρ is the string's material density and A is its cross-sectional area. Furthermore, when it models a wind instrument, the parameter c is defined as a combination of different physical parameters, particularly $c = \sqrt{B/\rho}$, where B and ρ are the bulk modulus and the material density of the air in the considered tube.

(Add reference to derivation ?? meh)

An important method which aims to simplify a physical system is scaling, which ac hives a reduction of the number of parameters needed to describe the system. For the case of the 1-D wave equation, the dimensionless coordinate $x' = x/L$ is introduced, where L can represent the string's length. Thus the domain of analysis turns into $x \in [0,1]$. Equation 3.1 now becomes:

$$u_{tt} = \gamma^2 u_{x'x'} \quad \text{with} \quad \gamma = c/L \tag{3.2}$$

3.1.1 Finite Difference Scheme

Using the operators described in Chapter 2 one can rewrite Equation 3.2 as:

$$\delta_{tt} u_l^n = \gamma^2 \delta_{xx} u_l^n \tag{3.3}$$

where u_l^n is a grid function representing the approximate solution for $u(x,t)$ at $x = lh$ and $t = nk$, with h being the spatial step and k being the time step of the discretization, see Figure 2.2.

Equation 3.3 can now be expanded, again using the operator definitions in Chapter 2, to give the update equation for calculating u_l^{n+1}, i.e. the transverse displacement at the "next" time step:

$$u_l^{n+1} = \frac{-2\gamma^2 k^2 u_l^n + \gamma^2 k^2 u_{l-1}^n + \gamma^2 k^2 u_{l+1}^n + 2h^2 u_l^n - h^2 u_l^{n-1}}{h^2} \tag{3.4}$$

Rearranging Equation 3.4 and introducing the important dimensionless parameter $\lambda = \gamma k/h$, reffered to as the Courant number, one arrives to:

$$u_l^{n+1} = 2\left(1 - \lambda^2\right) u_l^n + \lambda^2 \left(u_{l-1}^n + u_{l+1}^n\right) - u_l^{n-1} \tag{3.5}$$

3.1.2 Von Neumann Analysis - Stability

The stability of the solution given in Equation 3.5 can be investigated by applying a Von Neumann frequency domain analysis. This implies introducing a test solution of the form:

$$u_l^n = z^n e^{jl\beta h} \tag{3.6}$$

where $z = e^{j\omega k}$. The test solution thus behaves as a single wave-like component of angular frequency ω and wavenumber β. j is the imaginary unit number, $\sqrt{-1}$.

Substituting the test solution in Equation 3.5 results in:

$$z^{n+1}e^{jl\beta h} = 2\left(1-\lambda^2\right)z^n e^{jl\beta h} + \lambda^2\left(z^n e^{j(l-1)\beta h} + z^n e^{j(l+1)\beta h}\right) - z^{n-1}e^{jl\beta h}, \tag{3.7}$$

which after some rearrangement and simplification (division by z^n), results in the characteristic equation:

$$z + 2(2\lambda^2 \sin^2(\beta h/2) - 1) + z^{-1} = 0 \tag{3.8}$$

here the following identities were used:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \quad \Rightarrow \quad \sin^2(x) = \frac{e^{j2x} + e^{-j2x}}{-4} + \frac{1}{2} \tag{3.9}$$

Equation 3.8 has the following roots, as per the solution of a 2^{nd} order polynomial equation:

$$z_\pm = 1 - 2\lambda^2 \sin^2(\beta h/2) \pm \sqrt{\left(1 - 2\lambda^2 \sin^2(\beta h/2)\right)^2 - 1} \tag{3.10}$$

The finite difference scheme will be stable when the roots are bounded by unity for all wave numbers β, i.e. $|z| \leq 1$ and furthermore, it can be shown that this is true for a 2^{nd} order polynomial of the form $z^2 + a_1 z + a_2$, when:

$$|a_1| - 1 \leq a_2 \leq 1 \tag{3.11}$$

Making use of the above condition and applying it to the characteristic Equation 3.7, one can arrive after some manipulation to:

$$\lambda^2 \sin^2(\beta h/2) \leq 1 \tag{3.12}$$

which results in the stability condition, known as the Courant-Friedrichs-Lewy (CFL) condition:

$$\lambda \leq 1 \tag{3.13}$$

This condition is applied in practice when implementing the finite difference scheme as a minimum bound on the grid spacing:

$$h \geq h_{min} = ck/L = \gamma k \tag{3.14}$$

One may interpret the minimum spacing condition h_{min} as the smallest grid spacing needed to capture the shortest wavelength.

3.1.3 Dispersion - Model vs Numerical

The dispersion of a system is described via a *dispersion relationship*, which relates the wavelength of a particular wave to its temporal frequency, i.e. $\omega = f(\beta)$. As the ratio of the angular frequency to the wave number represents the phase velocity of the wave, $v_\phi = \omega/\beta$, which describes the speed of propagation of the wave crest, it follows that the dispersion relationship gives an indication of how different frequencies propagate with respect to each other. Also of interest is the group velocity which describes the propagation velocity of the overall envelope of the wave, given by $v_g = d\omega/d\beta$. A linear dispersion relationship indicates that the phase velocity is constant for all combinations of temporal and spatial frequency (ω and β), while a non-linear dispersion relationship indicates that some frequencies will travel faster or slower than others.

Typically, one can determine the dispersion relationship of a physical system, such as the 1-D wave equation described in Equation 3.2, by inserting a traveling wave solution of the form $Ue^{j(\omega t+\beta x)}$, where U is a constant. Applying the derivations to this solution and making use of the chain rule, knowing that $\partial\left(e^x\right)/\partial x = e^x$ yields:

$$U\beta^2\gamma^2 e^{j(\beta x+\omega t)} - U\omega^2 e^{j(\beta x+\omega t)} = 0 \quad \Rightarrow \quad \omega = \pm\beta\gamma \tag{3.15}$$

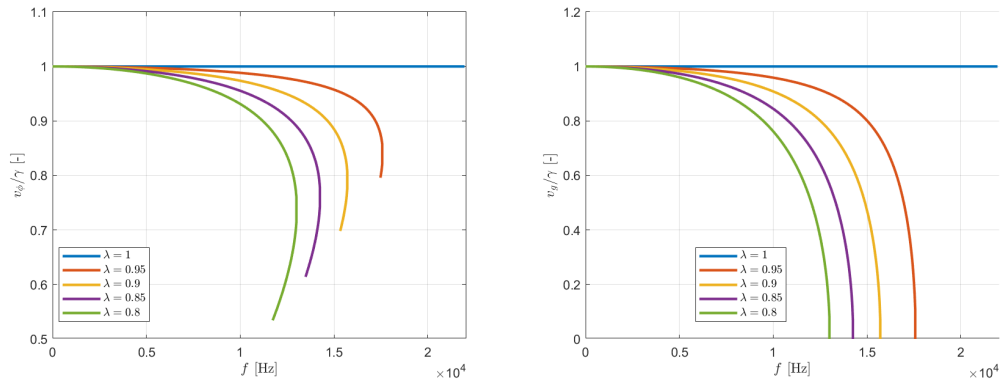
Furthermore, the dispersion relationship of the numerical FDS (numerical dispersion) can also be derived by using $z = e^{j\omega k}$ in the characteristic Equation 3.8 and again making use of the identities in Equation 3.9. This results in:

$$-4\sin^2(\omega k/2) = -4\lambda^2\sin^2(\beta h/2) \quad \Rightarrow \quad \omega = \pm\frac{2}{k}\sin^{-1}(\lambda\sin(\beta h/2)) \tag{3.16}$$

In the case that $\lambda = 1$ the numerical dispersion reduces to the same relationship seen for the physical system:

$$\omega = \pm\frac{2}{k}\sin^{-1}(\sin(\beta h/2)) = \frac{\beta h}{k} = \gamma\beta \tag{3.17}$$

This implies that the numerical solution is exact in this particular case. Figure 3.1 shows the phase velocity and the group velocity (normalized with the γ parameter) for different values of λ and for a FDS scheme where the time step is chosen



(a) **(b)**

Figure 3.1: Phase velocity (left) and group velocity (right) of the numerical scheme normalized with the model velocity, γ, as a function of frequency.

as $k = 1/f_s$, with $f_s = 44100$ Hz. It can be seen than when $\lambda < 1$ the numerical scheme is essentially not able to fill the entire audio bandwidth, from 0 to $f_s/2$, i.e. the scheme cannot produce frequencies larger than a certain values. As a reminder the relationship between angular frequency and the ordinary frequency is: $\omega = 2\pi f$.

The Von Neumann analysis presented in the previous section as a method for evaluating the stability of the FDS, i.e. introducing a test solution of the form $u_l^n = z^n e^{jl\beta h} = e^{j(n\omega k + l\beta h)}$ discretized scheme, is seen to be also the basis for arriving at a dispersion relationship of the form $\omega = \omega(\beta)$.

A more in depth description of the dispersion phenomenon can be seen in [15].

3.1.4 Boundary Conditions

Stability of the FDS shown in Equation 3.5 is not only dependent on the choice of the λ parameter as described in Section 3.1.2 but also on the choice of boundary conditions, which describe the behavior of the string (or tube) at its edges. Considering the string to be spatially discretized into $N + 1$ points, the range of calculation will be l = [0,1,...,N-1,N], see Figure 2.2. Therefore the points of interest regarding boundary conditions are $l = 0$ and $l =$ N. Here one may choose the following different behaviors:

- Clamped: $u = u_x = 0$
  - Displacement is 0
  - Rotation is 0
- Simply Supported: $u = u_{xx} = 0$

  - Displacement is 0
  - Curvature is 0

- Free: $u_{xx} = u_{xxx} = 0$
  - Curvature is 0
  - Rate of change of curvature is 0

Normally the behaviour at the boundary conditions and their discretization in the FDS can be investigated via energy analysis. For the case of a lossless system, such as the 1-D wave equation, one aims at choosing boundary conditions which show a conservation of total energy in the system. The kinetic and potential energies may vary with time but their sum should be constant. Such an energy analysis is beyond the scope of this project and will not be described here but the reader is referred to [4].

3.1.5 Output of the FDS

The FDS presented in Equation 3.5 is implemented for a simply supported string where the tension is chosen to produce a string vibration with a fundamental frequency at 110 Hz. This is done by adjusting the string tension and therefore the γ parameter. The resulting displacement of the 1-D wave at various time steps is illustrated in Figure 3.3. A raised cosine is used for the initial excitation, i.e. $u(x, 0)$ which is illustrated in Figure 3.3 as u at time step 0. One can observe the left and right travelling waves along the length as well as the inversion when reflected at the boundaries. Finally, Figure 3.4 shows the time series at the output point, chosen at a random location along the string (thin grey line in Figure 3.3) and the resulting spectrum of this output.

3.2 Ideal Bar

Another interesting 1-D system whose transverse vibrations have interesting acoustic properties is the ideal bar. This physical system can be described by the following equation:

$$\rho A u_{tt} = -EIu_{xxxx}, \tag{3.18}$$

where ρ is the material density of the bar, A is its cross-sectional area, E is Young's modulus (which gives a measure of the stiffness of the material) and I is the bar's moment of inertia resulting from the geometry of the bar's cross-section. These physical properties are assumed to be constant over the entire bar length.

Equation 3.18 can be scaled and thus x is replaced by x/L, where L is the length of the bar. Furthermore, the physical properties described in the previous

paragraph can be grouped and isolated into a single parameter $\kappa = \sqrt{\frac{EI}{\rho AL^4}}$ which represents the stiffness of the bar. The resulting equation describes the scaled ideal bar physical system:

$$u_{tt} = -\kappa^2 u_{xxxx}, \tag{3.19}$$

3.2.1 Dispersion

The dispersion relationship of the physical system described in Equation 3.19 can be computed by inserting a solution of the form $Ue^{j(\beta x+\omega t)}$, resulting in the following:

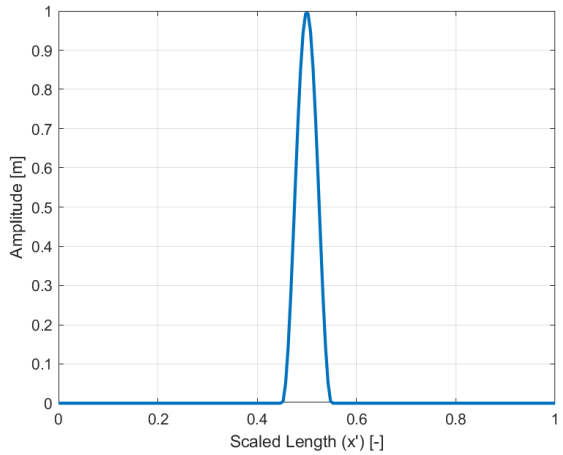


Figure 3.2: Raised cosine displacement shape which is the initial excitation of the string.

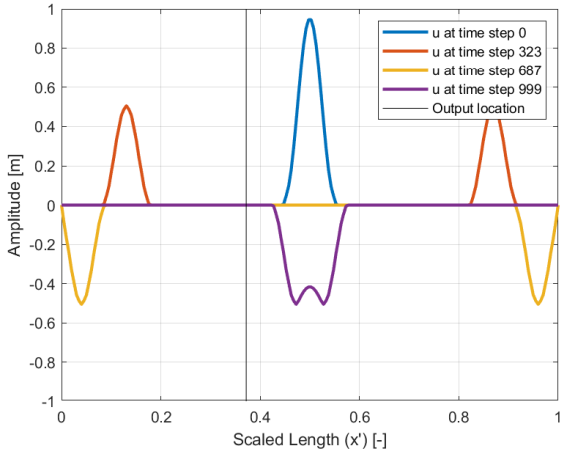
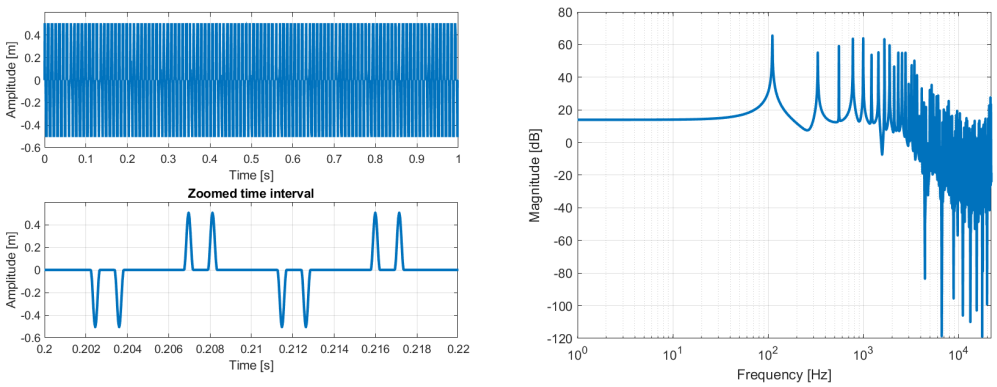


Figure 3.3: Displacement of the string at various time steps in the numerical simulation.



(a) **(b)**

Figure 3.4: Time series of the amplitude at the output point along the string (left) and the magnitude of the frequency spectrum of this output signal (right).

$$\omega = \pm\kappa\beta^2, \tag{3.20}$$

This results in the following phase and group velocity:

$$v_\phi(\omega) = \frac{\omega}{\beta} = \sqrt{\kappa\omega}, \quad v_g(\omega) = \frac{d\omega}{d\beta} = 2\sqrt{\kappa\omega}, \tag{3.21}$$

indicating that higher frequencies travel faster than lower frequencies.

3.2.2 Explicit vs Implicit Finite Difference Scheme

The physical model of the ideal bar is an illustrative system for introducing the difference between explicit and implicit numerical schemes. Firstly, a scheme similar to that presented in the previous section is chosen:

$$\delta_{tt}u = -\kappa^2\delta_{xxxx}u, \tag{3.22}$$

which results in the following update equation:

$$u_l^{n+1} = (2 - 6\mu^2)u_l^n + 4\mu^2(u_{l+1}^n + u_{l-1}^n) - \mu^2(u_{l-2}^n + u_{l+2}^n) - u_l^{n-1}, \tag{3.23}$$

Here $\mu \triangleq \kappa k/h^2$ is a similar parameter to the Courant number λ from the previous section and is found as an important parameter for stability of the scheme. Employing Von Neumann analysis, i.e. introducing a test function of the form $u_l^n = z^n e^{j\beta lh}$ in the FDS, as done in Section 3.1.2 one may arrive at the following stability condition:

$$\mu \leq \frac{1}{2}, \quad k \leq \frac{h^2}{2\kappa}, \tag{3.24}$$

An important observation regarding the scheme in Equation 3.23 is that one can solve directly for u_l^{n+1} in terms of u_l^n and u_l^{n-1}, which are known at each time step. Only present and past information is needed to solve for the next time step. This is termed an explicit update scheme.

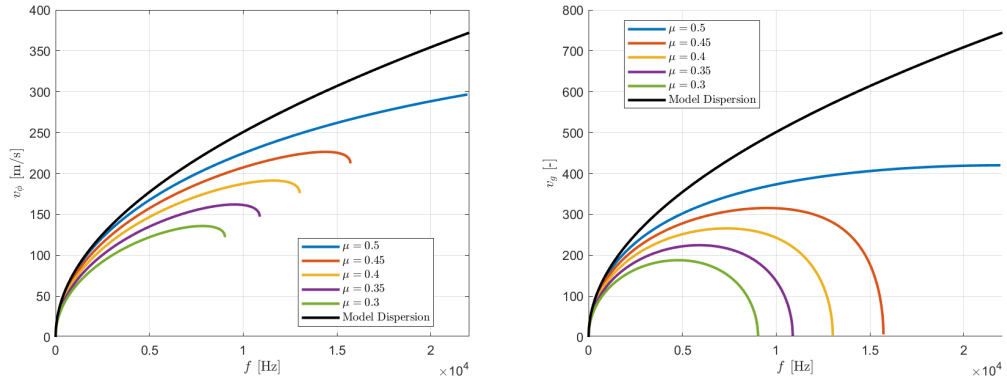
When evaluating the numerical dispersion of this scheme, which can be found, via Von Neumann analysis, to be:

$$\sin(\omega k/2) = \pm 2\mu \sin^2(\beta h/2), \tag{3.25}$$

it can be observed that the phase and group velocities will be different from the dispersion of the physical model, for any choice of μ. This is illustrated in Figure 3.5.

This observation implies that an exact solution cannot be achieved with this numerical scheme. Furthermore, when operating away from the stability condition, i.e. $\mu < \frac{1}{2}$, there is also a loss in bandwidth, as the FDS cannot produce frequencies above a certain cutoff. As μ is smaller, this difference increases.

It turns out that a more accurate FDS can be chosen for discretizing the continuous model in Equation 3.19, by introducing an averaging operator for the time derivative part, controlled by a free parameter θ. This scheme is shown in Equation 3.26.



(a) **(b)**

Figure 3.5: Comparison of phase velocity (a) and group velocity (b) of the numerical scheme with the continuous model for different values of μ as a function of frequency. An ideal bar with $\kappa = 1$ was simulated, with a time step $k = 1/44100$ and spatial step h resulting from the stability condition in Equation 3.24 satisfied with equality.

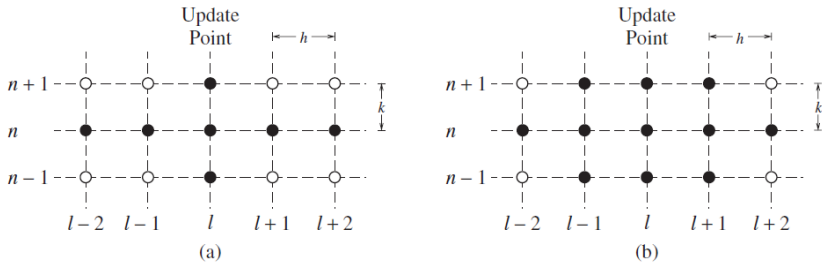


Figure 3.6: Computational stencil for the explicit scheme 3.22 and for the implicit scheme 3.26.

$$(\theta + (1-\theta)\mu_{x\cdot})\delta_{tt}u = -\kappa^2\delta_{xxxx}u, \tag{3.26}$$

When the free parameter $\theta = 1$ this scheme reduces to previous one shown in Equation 3.22. Expanding this scheme results in:

$$\begin{aligned} &\theta\frac{1}{k^2}(u_l^{n+1} - 2u_l^n + u_l^{n-1}) \\ +(1-\theta)\frac{1}{2}(&\frac{1}{k^2}(u_{l+1}^{n+1} - 2u_{l+1}^n + u_{l+1}^{n-1}) + \frac{1}{k^2}(u_{l-1}^{n+1} - 2u_{l-1}^n + u_{l-1}^{n-1})) \\ &= -\kappa^2\frac{1}{h^4}(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n), \end{aligned} \tag{3.27}$$

The reason for expanding the equation in this rather ugly manner is to illustrate the presence of the u_{l-1}^{n+1} and u_{l+1}^{n+1} terms in addition to the previously used update point u_l^{n+1}. This means that in order to compute values at a "future" time step at a location l one needs to know values which have not yet been calculated. This is referred to as an implicit scheme and the system will become a linear system of equations which can be solved by a matrix-division. An illustration of the necessary stencil update for the implicit scheme compared with the explicit is shown in Figure 3.6.

The values of the grid function u_l^n to be computed can be arranged in a finite-length column vector $\mathbf{u}^n = [u_0^n, ..., u_N^n]^T$, where T denotes the transpose operation. Furthermore, the spatial finite difference operators can be written in matrix form as described in Section 2.4. Then, expanding the time difference operator, results in the following FDS, where bold capital symbols represent matrices:

$$(\theta + (1-\theta)\mathbf{M}_{x\cdot})\frac{1}{k^2}(\mathbf{u}^{n+1} - 2\mathbf{u}^n + \mathbf{u}^{n-1}) = -\kappa\mathbf{D}_{xxxx}\mathbf{u}^n, \tag{3.28}$$

Rearranging Equation 3.26, one can arrive at the following form, with $\mathbf{I}$ being the identity matrix:

$$\begin{aligned} \mathbf{A}\mathbf{u}^{n+1} + \mathbf{B}\mathbf{u}^{n} + \mathbf{C}\mathbf{u}^{n-1} = 0, \quad \text{with,} \\ \mathbf{A} = \frac{1}{k^2}(-\mathbf{M}_{x\cdot}\theta + \mathbf{M}_{x\cdot} + \theta\mathbf{I}) \\ \mathbf{B} = \frac{1}{k^2}(\mathbf{D}_{xxxx}\kappa^2 k^2 + 2\mathbf{M}_{x\cdot}\theta - 2\mathbf{M}_{x\cdot} - 2\theta\mathbf{I}) \\ \mathbf{C} = \frac{1}{k^2}(-\mathbf{M}_{x\cdot}\theta + \mathbf{M}_{x\cdot} + \theta\mathbf{I}) \end{aligned} \tag{3.29}$$

From here an update equation for $\mathbf{u}^{n+1}$ can be written which involves the inversion of the matrix $\mathbf{A}$,

$$\mathbf{u}^{n+1} = \mathbf{A}^{-1}(\mathbf{B}\mathbf{u}^{n} + \mathbf{C}\mathbf{u}^{n-1}), \tag{3.30}$$

As it can be seen, using an implicit scheme is computationally more costly, as it requires significantly more memory, due to the matrix form (one solves the entire spatial bar at once, per each time step) as well as the fact that the matrix inversion is a complex computation. Luckily the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are sparse and thus using special techniques for dealing with sparse computations, the computational speed can be reduced. Such techniques are fortunately already implemented in Matlab and available to use [10].

The Von Neumann energy methods yield the following stability conditions:

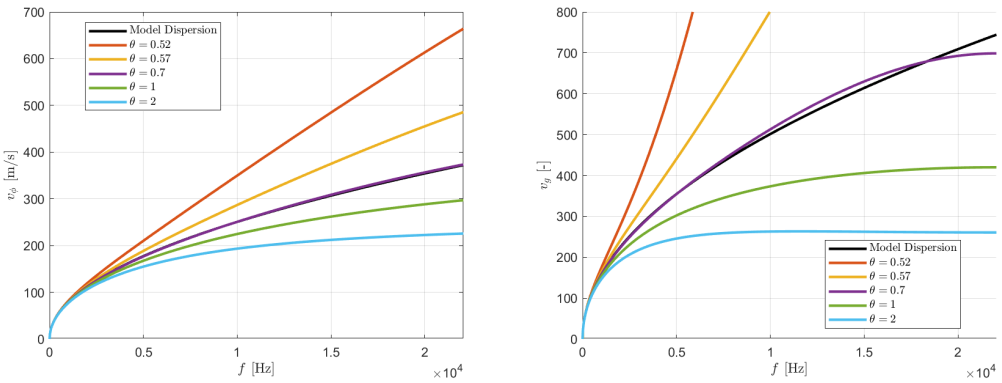
$$\theta \geq \frac{1}{2} \qquad \mu \leq \frac{\sqrt{2\theta - 1}}{2}, \tag{3.31}$$

As a reminder, the $\mathbf{D}_{xxxx}$ operator contains the spatial step h in its formulation, see Section 2.4, hence the parameter $\mu \triangleq \kappa k/h^2$ is present in Equation 3.36.

The numerical dispersion relationship of the FDS in Equation 3.26 can be computed by introducing a solution of the form $u_l^n = e^{j(\omega nk + \beta lh)}$. It is more convenient to introduce it in the non-matrix formulation shown in Equation 3.27. This will result in a quite complicated dispersion relationship (should I add it ?). One can again compute the phase velocity and group velocity of the implicit FDS for various values of the parameter θ. This is illustrated in Figure 3.7.

(Maybe make a github and reference to that instead of all the code.)

It can be seen that the implicit scheme avoids the loss of bandwidth problem, observed for the explit scheme for values wher $\mu < 1/2$ as shown in Figure 3.5. Additionally, the parameter θ can be tweaked such that the numerical solution is much more accurate in terms of dispersion as compared to the continuous model, than was previously observed for the explicit scheme. It is still not an exact solution, but it a considerable improvement.



(a) **(b)**

Figure 3.7: Comparison of phase velocity (a) and group velocity (b) of the implicit numerical scheme 3.26 with the continuous model for different values of the free parameter θ as a function of frequency. An ideal bar with $\kappa = 1$ was simulated, with a time step $k = 1/44100$ and spatial step h resulting from the stability condition in Equation 3.31 satisfied with equality.

3.2.3 Output of the FDSs

The implicit FDS described in Equation 3.26 has been implemented using the update matrix equation described in Equation 3.36. Similar to the previous sections, Figure 3.8 shows the displacement of an ideal bar with a scaled stiffness $\kappa = 3$, at different time steps in a simulation where the free parameter $\theta = 0.7$, the value that was shown to give a really accurate result in terms of model vs numerical dispersion comparison. A time step $k = 1/44100$ was chosen and the spatial step was computed as to satisfy the stability conditions with equality. The shape at time step 0 is the initial shape, i.e. the excitation.

Furthermore, the simulation was rerun with the free parameter $\theta = 1$ in which case the implicit scheme practically reduces to the explicit scheme given in Equation 3.22. A comparison of the time series at the same output point along the string for the two solutions is illustrated in Figure 3.9 both in time and frequency domain. It can be seen that the two time-series start to oscillate quite in sync, but quickly begin to drift apart. This is due to the fact that the high frequency content of the system propagates at different velocities in the two cases, as was shown in Figure 3.7. This can be observed from a different angle also in the frequency response, where the first few resonances in the low frequency are almost identical in the two cases, but in the upper part of the bandwidth there is more discrepancy.

It must be noted that the boundary condition chosen in this analysis was the clamped case. This has a straightforward implementation where the clamped condition at $l = 0$ and $l = N$ is the following:

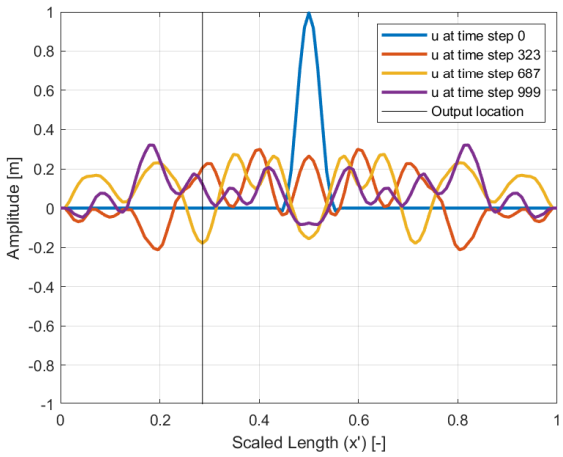
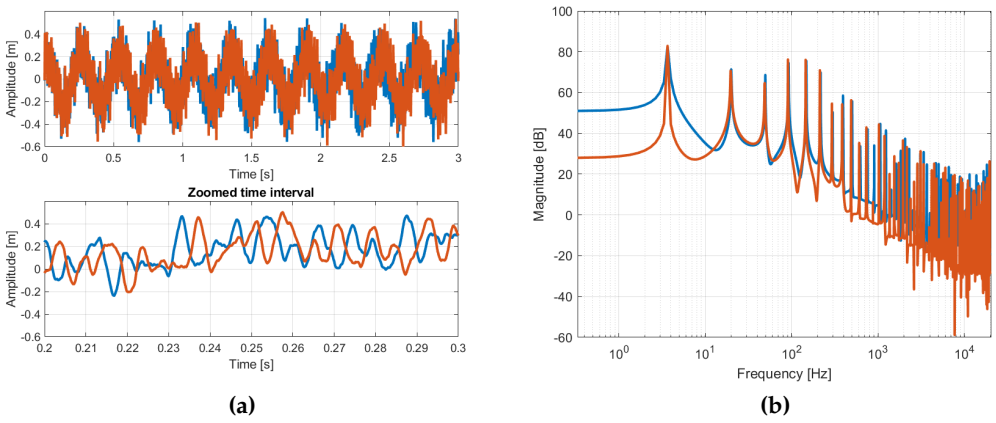


Figure 3.8: Displacement of an ideal bar with stiffness $\kappa = 3$ at various time steps in a numerical simulation using an implicit scheme with $\theta = 0.7$. The shape at time step 1 corresponds to the initial displacement (excitation).



(a) **(b)**

Figure 3.9: Numerical simulation result of (a) time series of the amplitude at the output point along the ideal bar for the case where an implicit scheme with $\theta = 0.7$ - blue line and $\theta = 1$ - red line (which reduces to the explicit solution) and (b) the magnitude of the frequency spectrum of these output signals.

$$u_0 = u_1 = 0 \qquad u_{N-1} = u_N = 0 \tag{3.32}$$

Basically at each time step after computing the update vector $\mathbf{u}^n$, the entries corresponding to the edge points described above are set to 0.

3.3 Slinky Spring

Building on the ideal cases presented in the previous two sections, a new 1-D acoustical system is described here. This new system models a slinky; a toy which has been around since the 1940s and is basically an oversized and highly flexible helical spring. It turns out that this toy can produce interesting sonic behaviors, having been used even by John Cage in an avant garde piece in 1959, [7]. Parker et. al. [17] described the physical model of the slinky as well as the FDS implementation presented here. Its behavior can be described using the Euler-Bernoulli ideal bar equation, which in dimensionless form is given by:

$$u_{tt} = -\kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{xxt} \tag{3.33}$$

where $u = u(x,t)$ is the transverse displacement of the slinky, $x \in [0,1]$ is a dimensionless coordinate running along the slinky, κ is the dimensionless constant which models the overall stiffness. For the case of the ideal bar κ can be calculated precisely knowing the material density, Young's modulus and so on, see Section 3.2, but for the slinky due to its rough approximation as a bar (which it is not) this is not as straight-forward. In this case κ can be calibrated based on measurements of the impulse response of the slinky, this will be discussed in an upcoming section.

It can be seen that the only addition of the model compared to the ideal bar presented in the previous section is the last two terms in Equation 3.33. σ_0 is a parameter that describes the damping of the system, a loss which is constant for all frequencies. In real life systems however, loss generally increases with frequency, resulting in the wide band characteristic of the initial sound decaying into only a few harmonics. Hence, the parameter σ_1 is introduced, which models the frequency dependent damping.

3.3.1 Finite Difference Scheme

Equation 3.33 can be discretized using the following operators, as per [17]:

$$\delta_{tt} u = -\kappa \delta_{xxxx} u - 2\sigma_0 \delta_{t\cdot} u + 2\sigma_1 \delta_{xx} \delta_{t-} u \tag{3.34}$$

This is an explicit scheme, as one only needs information from the "current" and "previous" time steps to calculate the "next" time step, u_l^{n+1}.

For this current implementation however, it is chosen to build on the implicit scheme in the previous section and introduce the same type of scheme with the free parameter θ as in Equation 3.26. Thus the following scheme will be used:

$$(\theta + (1-\theta)\mu_{x\cdot})\delta_{tt}u = -\kappa\delta_{xxxx}u - 2\sigma_0\delta_{t\cdot}u + 2\sigma_1\delta_{xx}\delta_{t-}u \tag{3.35}$$

This scheme can be written in matrix form in the following way:

$$\begin{aligned} \mathbf{A}\mathbf{u}^{n+1} + \mathbf{B}\mathbf{u}^{n} + \mathbf{C}\mathbf{u}^{n-1} = 0, \quad \text{with,} \\ \mathbf{A} = \frac{1}{k^2}(-\mathbf{M}_{x\cdot}\theta + \mathbf{M}_{x\cdot} + 2k\sigma_0\mathbf{I} + \theta\mathbf{I}) \\ \mathbf{B} = \frac{1}{k^2}(-2\mathbf{D}_{xx}k\sigma_1 + \mathbf{D}_{xxxx}\kappa^2k^2 + 2\mathbf{M}_{x\cdot}\theta - 2\mathbf{M}_{x\cdot} - 2\theta\mathbf{I}) \\ \mathbf{C} = \frac{1}{k^2}(-\mathbf{M}_{x\cdot}\theta + \mathbf{M}_{x\cdot} + 2k\mathbf{D}_{xx}\sigma_1 - 2k\sigma_0\mathbf{I} + \theta\mathbf{I}) \end{aligned} \tag{3.36}$$

Simply supported boundary conditions are implemented, meaning that at the boundaries the following conditions must be satisfied at $l = 0$ and $l = N$:

$$u = u_{xx} = 0 \tag{3.37}$$

Applying these conditions at point $l = 0$ and $l = N$ one arrives at the equations $u_{-1} = -u_1$ and $u_{N+1} = -u_{N-1}$. This results from the definition of δ_{xx}:

$$\begin{aligned} \delta_{xx}u_0 = \frac{1}{h^2}(u_1 - 2u_0 + u_{-1}) = 0, \quad \text{where} \quad u_0 = 0 \quad \Rightarrow \quad u_1 + u_{-1} = 0 \\ \delta_{xx}u_N = \frac{1}{h^2}(u_{N+1} - 2u_N + u_{N-1}) = 0, \quad \text{where} \quad u_N = 0 \quad \Rightarrow \quad u_{N-1} + u_{N+1} = 0 \end{aligned} \tag{3.38}$$

The domain of interest for calculation will be $l \in [1, 2, ..., N-2, N-1]$, as at points $l = 0$ and $l = N$ the u will always be 0. It can now be seen that the equations presented above come in handy when applying the δ_{xxxx} at the first and last points in the domain of calculation, as u_{-1} and u_{N+1} can now be replaced accordingly.

$$\delta_{xxxx}u_1 = \frac{1}{h^4}(u_3 - 4u_2 + 6u_1 - 4u_0 + u_{-1}) = \frac{1}{h^4}(u_3 - 4u_2 + 5u_1) \tag{3.39}$$

These boundary conditions need to be introduced in the spatial matrix operators $\mathbf{D}_{xxxx}$, $\mathbf{D}_{xx}$ and $\mathbf{M}_{x\cdot}$. Following Equation 3.39, the first row of the $\mathbf{D}_{xxxx}$ operator will be $[5, -4, 1]$ followed by $N-4$ zeros, while the last row will be $N-4$ zeros followed by $[1, -4, 5]$. The other matrix form spatial operators $\mathbf{D}_{xx}$ and $\mathbf{M}_{x\cdot}$ will have a perfect sparse Toeplitz matrix form in the domain of interest, see 2.4.

The same approach described for the boundary condition at $l = 0$ is applied for $l = N$.

The model of the slinky in the case of lossless conditions ($\sigma_0 = \sigma_1 = 0$) is identical to that of the ideal bar presented in Section 3.2. Therefore, the model vs. numerical dispersion comparison between the continuous and discretized case will be the same as previously shown, albeit with a different stiffness parameter κ.

Regarding stability conditions, one can again look back at the ideal bar model. Given a time step $k = 1/44100$, driven by a desired sampling frequency $f_s = 44100$ Hz, one can find the optimum h for this explicit scheme by satisfying the condition in Equation 3.31 with equality. (Say that you use the stability conditions for the lossless case, if you don't find the right equation that incudes $sigma_1$).

3.3.2 FDS Results - Impulse response of the Slinky

(Talk about impulse response briefly ? More ?)

The impulse response of a system is the output of that system when excited with an impulse, generally a Dirac function ([14]), but can be any function that contains all frequencies, and therefore it describes the response of a linear time-invariant system for all frequencies.

In [17], the authors set up an experiment where they measure the impulse response of an actual slinky toy and calibrate the κ parameter in the FDS based on those measurements. Particularly they look at the dispersive echoes observed in the spectrogram of a measured slinky impulse response, knowing that the time it takes to perform an end-to-end traversal of the lossless system at a particular frequency is given by:

$$T_d = \frac{1}{2\sqrt{\kappa 2\pi f}} \tag{3.40}$$

where 1 is basically the length of the slinky, as $x \in [0, 1]$ in the scaled case. One can see that the denominator of Equation 3.40 is in fact the group velocity, v_g given in 3.21.

Finding the time between the echoes observed in the measured spectrum and knowing 3.40, κ was estimated to be 0.06. The same value was used for the implementation used here.

In order to have the best match between numerical and model dispersion, the free parameter θ was chosen to be 0.7, see Section 3.2. The system was excited with a Dirac function, $\delta(t)$, where $\delta(0) = 1$ and 0 otherwise, at location $l = 1$ (first free point after the boundary condition) and was let to vibrate freely for 5 seconds. During this time the output was measured chosen at the closest free point on the other end of the slinky, i.e., $l = N - 1$. Furthermore, the loss parameters were chosen as $\sigma_0 = 0.1$ and $\sigma_1 = 0.00001$ as to produce a gentle decay in the duration of the simulation.

The displacement of the slinky at various time steps, including the initial condition at time step 0, i.e. the Dirac function, is shown in Figure 3.10. Since the

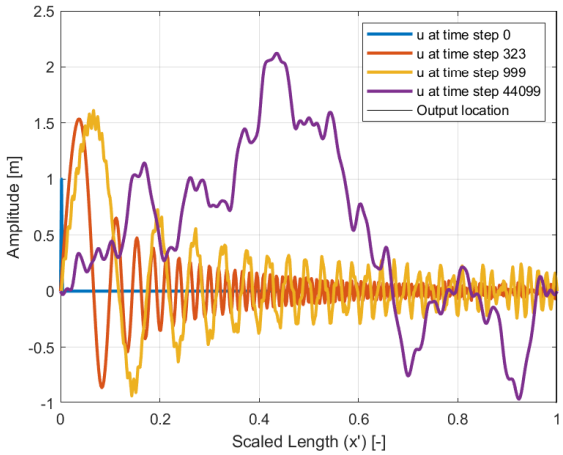


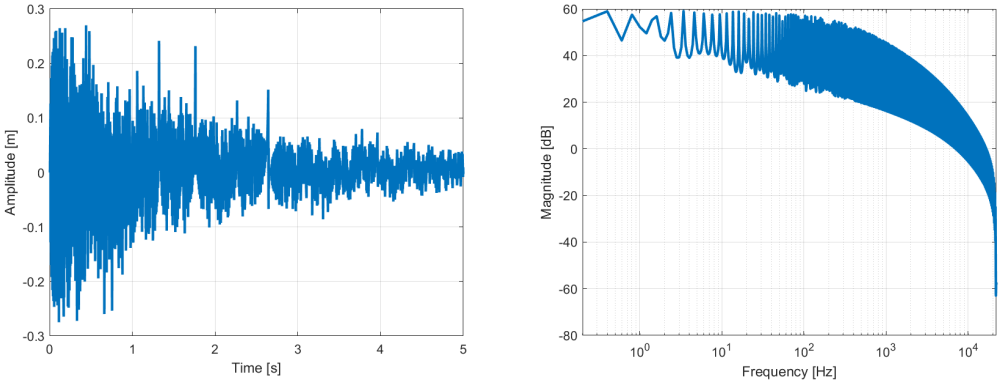
Figure 3.10: Displacement of the slinky with stiffness $\kappa = 0.06$ at various time steps in a numerical simulation using an implicit scheme with $\theta = 0.7$. The shape at time step 0 corresponds to the initial displacement (excitation).

system is excited with the Dirac function, all frequencies, having the same amplitude are sent at the excitation point at the same time (the frequency response of the Dirac function is a flat line). One can see that at the smaller time steps, i.e. closer to the initial excitation, only short waves are present, i.e. waves with a large wave number, β, which travel faster. After some time, longer waves, arrive as well, which is seen at time step 44099.

The recorded output represents the impulse response of the slinky and its time series as well as the magnitude of its frequency response is shown in Figure 3.11.

In order view the variation of the spectrum of frequencies as it varies with time one can calculate the spectrogram of the signal. Figure 3.12 shows this for the slinky impulse response, where a Hann window of 256 samples with 50% overlap was used.

One can verify that the numerical model produces an accurate solution compared to the continuous model, and by association even to the slinky measured by Parker et al. in [17], by plotting the dispersive echoes on top of the spectrogram of the impulse response. As Equation 3.40 gives the time it takes for different frequencies to traverse the length of the slinky based on the group velocity of the continuous model, we can check if our numerical solution follows these curves. This is shown in Figure 3.13, where the first 3 echoes are plotted on top of the spectrogram. Very good agreement is observed.



(a) **(b)**

Figure 3.11: Numerical simulation result of a. time series of the amplitude at the output point of the slinky for the case where an implicit scheme with $\theta = 0.7$ and b. the magnitude of the frequency spectrum of this output signals.

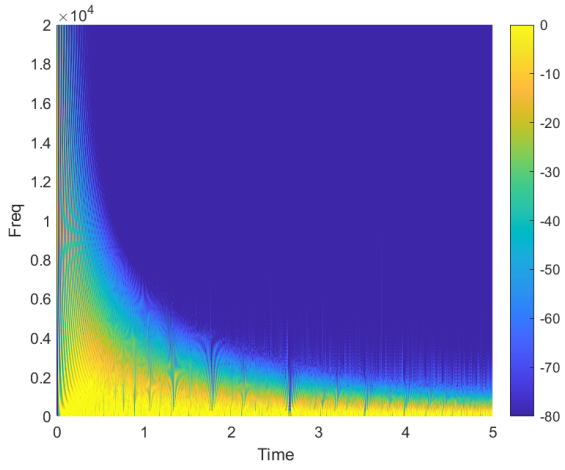


Figure 3.12: Spectrogram of the impulse response of a slinky with stiffness $\kappa = 0.06$ and loss parameters $\sigma_0 = 0.1$, $\sigma_1 = 0.00001$.

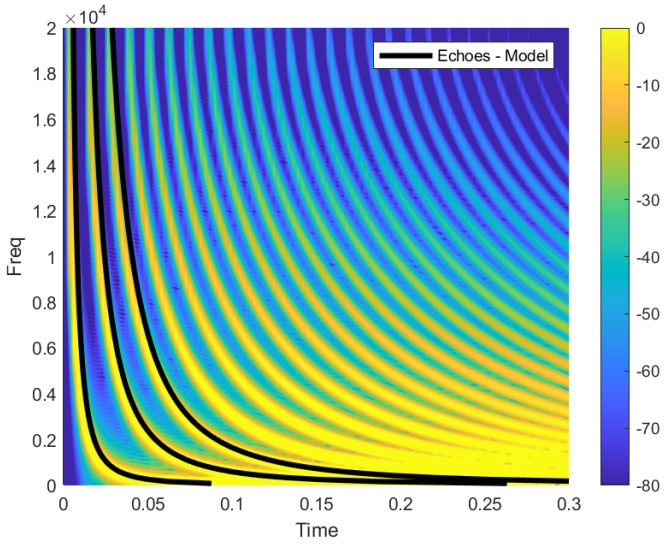


Figure 3.13: Spectrogram of the impulse response of a slinky with stiffness $\kappa = 0.06$ and loss parameters $\sigma_0 = 0.1$, $\sigma_1 = 0.00001$, together with dispersive echoes calculated based on the group velocity of the continuous model.

Chapter 4

Helical Spring

This chapter will describe the model of a helical spring, with dimensions typical to those used in spring reverberation units. Furthermore a FDS implementation of the model will be shown, with a focus on the necessity of an implicit scheme as opposed to an explicit one. A subsection will then be dedicated to the development of an optimization procedure for calculating the optimal FDS free parameters, as to best match the dispersion characteristics of the continuous model.

4.1 Continuous Model

The helical spring model presented in this report is based on the work of Bilbao and Parker in [2]. They propose a coupled model where the displacement in transverse, u, and longitudinal direction, ζ, along the arclength $x \in [0, L]$, for some unwound spring length L, are interdependent. This immediately highlights a major difference of this acoustic system as opposed to the ones presented in Chapter 3, the state of the system has 2 degrees of freedom opposed to the 1 degree of freedom, with the two displacements being coupled. Such a model is described in continuous time by the following equations:

$$u_{tt} = \frac{-Er^2}{4\rho}\left(u_{xxxx} + 2\epsilon^2 u_{xx} + \epsilon^4 u\right) + \frac{E\epsilon}{\rho}\left(\zeta_x - \epsilon u\right) - 2\sigma_t u_t, \tag{4.1a}$$

$$\zeta_{tt} = \frac{E}{\rho}\left(\zeta_{xx} - \epsilon u_x\right) - 2\sigma_l \zeta_t, \tag{4.1b}$$

where the spatial parameters of the spring are described in Figure 4.1. r is the radius of the wire, which is of circular cross-section and the parameter $\epsilon \approx 1/R$ is a measure of the curvature of the spring, with R being the coil radius. Furthermore, parameters E and ρ are related to the material of the spring, the first being the Young's modulus of elasticity and the second being the material density. The

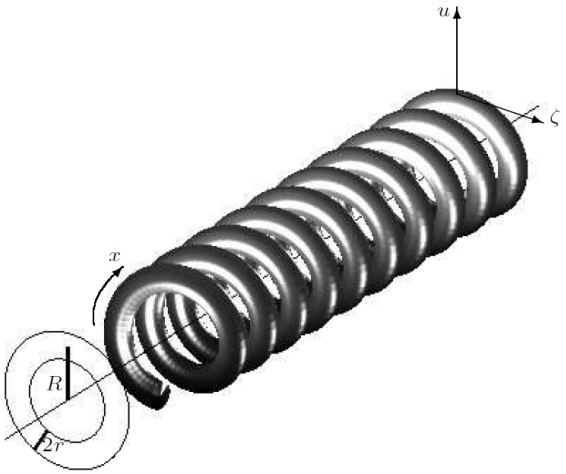


Figure 4.1: The spatial system of a helical spring as characterized by the model in Equation 4.1. The spring has a coil of radius R and a wire of radius r, while u describes the displacement in transverse direction and ζ describes the displacement in longitudinal direction. [2]

parameters σ_l and σ_t model loss in the longitudinal and transverse direction. There are a certain number of assumptions included in this model, particularly that the helix angle of the spring is small and that the wire is thin, under minimal tension.

As was done with the systems presented in Chapter 3, it is convenient to re-write the system in Equation 4.1 in a scaled form, thus reducing the number of parameters and simplifying the equations to some extent. This is done by introducing the non-dimensional variables $x' = x/L$, $u' = \epsilon u$ and $\zeta' = \zeta/L$. Furthermore, one can write $\kappa = \sqrt{\frac{E}{\rho}}\frac{r}{2L^2}$ as a measure of the stiffness of the spring, $\gamma = \frac{1}{L}\sqrt{\frac{E}{\rho}}$ as the longitudinal wave velocity and $q = \epsilon L$ as the curvature. This results in the following scaled system:

$$u_{tt} = -\kappa^2\left(u_{xxxx} + 2q^2u_{xx} + q^4u\right) + q^2\gamma^2\left(\zeta_x - \epsilon u\right) - 2\sigma_t u_t, \tag{4.2a}$$

$$\zeta_{tt} = \gamma^2\left(\zeta_{xx} - \epsilon u_x\right) - 2\sigma_l\zeta_t, \tag{4.2b}$$

It should be noted that for simplicity of notation, the $'$ superscript that indicates the "scaled" parameter was removed, and all subsequent notation regarding spatial variables refers to the "scaled" versions.

4.2 Dispersion - Continuous Model

In order to investigate/highlight the characteristics of the system presented in Equation 4.2, one may derive its dispersion relation in the lossless case, i.e., when

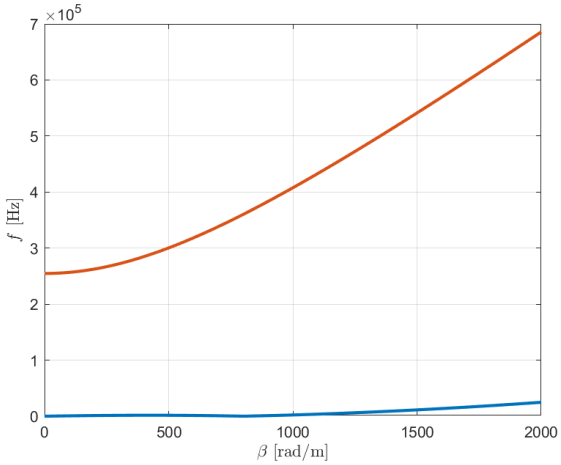


Figure 4.2: Positive solutions to the dispersion relationship for a helical spring system, with $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$.

$\sigma_l = \sigma_t = 0$. This is done by introducing solutions of the form $u(x,t) = Ue^{j(\omega t+\beta x)}$ and $\zeta(x,t) = Ze^{j(\omega t+\beta x)}$, where U and Z are some constants. After applying the derivatives to these solutions and factorizing both Equation 4.2a and Equation 4.2b with respect to U and Z, one can rewrite the system of equation in matrix form:

$$\begin{bmatrix} \omega^2 - \kappa^2(\beta^2 - q^2)^2 - q^2\gamma^2 & jq^2\gamma^2\beta \\ -j\gamma^2\beta & \omega^2 - \gamma^2\beta^2 \end{bmatrix} \begin{bmatrix} U \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.3}$$

An infinite number of nontrivial solutions to Equation 4.3 occur when the determinant of the matrix is zero. This would mean that there are an infinite number of combinations of (ω, β) that satisfy that equation, which results in a dispersion relationship. If the determinant of this matrix would be different than zero, then Equation 4.3 would have one single nontrivial solution. Another possible solution is the trivial solution, which is independent of this matrix, and would occur if $U = Z = 0$. This results in:

$$(\omega^2 - \gamma^2\beta^2)(\omega^2 - \kappa^2(\beta^2 - q^2)^2 - \gamma^2q^2) - \gamma^4q^2\beta^2 = 0 \tag{4.4}$$

Solving for ω in Equation 4.5, one finds 4 solutions grouped in two pairs (two $\pm$ solutions). Figure 4.2 shows the positive solutions for a case where $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$.

One can see that one of the solutions gives very large frequencies which in fact lie outside of the human auditory range. In fact one can calculate the minimum of the upper curve to be:

$$f_{\text{u}} = \frac{\gamma q}{2\pi} = \frac{\sqrt{E/\rho}}{2\pi R} \gg 20000\text{Hz} \tag{4.5}$$

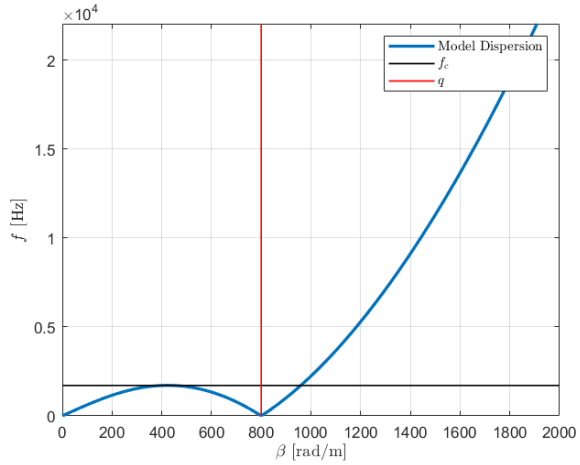
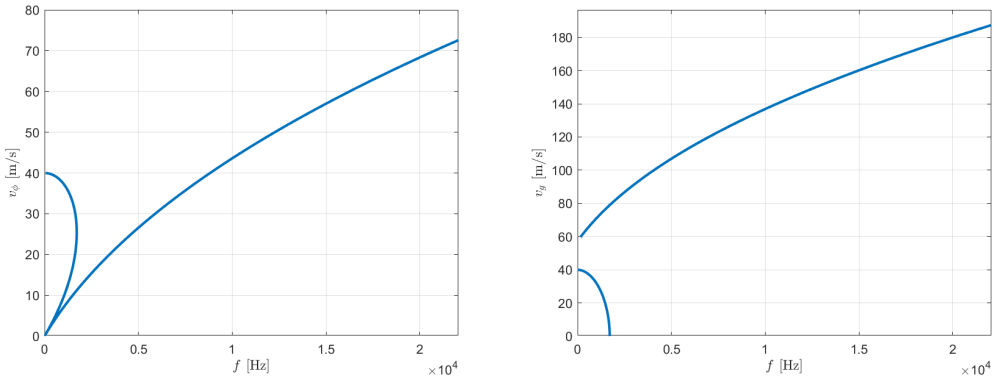


Figure 4.3: Auditory range solution to the dispersion relationship for a helical spring system, with $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$.



(a) **(b)**

Figure 4.4: a.) phase velocity and b) group velocity of a helical spring model with $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$.

This leads to focusing on the solution which gives rise to frequencies lying in the auditory range. Figure 4.3 shows this dispersion relationship. This highlights yet again another major difference of the helical spring system as opposed to the ones presented in Chapter 3: its dispersion relationship is not monotonic, meaning that components of different wave lengths can have the same temporal frequency. Or rephrased in terms of phase and group velocity, it means that certain frequency components can travel at multiple velocities, see Figure 4.4.

Furthermore, it can be seen that the dispersion relationship shown in Figure 4.3 has a zero at $\beta = q$ (vertical line in the figure) and has a maximum in the lower

frequency range at approximately $\beta = q/2$. Inserting this value in Equation 4.5 one can find the frequency f_c after which waves do not travel at multiple phase velocities. This is highlighted as the horizontal line in the same figure.

$$f_c \approx \frac{3\kappa q^2}{8\pi\sqrt{5}} \tag{4.6}$$

This can be viewed as a transition frequency where the dispersion regime changes, and is of perceptual interest for reverberation purposes.

4.3 Finite Difference Scheme

A reasonable numerical simulation of a helical spring system needs to be able to capture the complex dispersion behavior highlighted in the previous section. One needs to aim to avoid problems associated with numerical dispersion. It has been shown in the previous chapter that explicit methods are less accurate at capturing the real dispersion as compared to implicit methods. As it is pointed out by Bilbao and Parker in [2], explicit methods can only give reasonable results for modelling the helical spring at extremely high sample rates, which makes them unpractical. Therefore, the following implicit scheme, based on [4] is used here:

$$\begin{aligned}(1+\eta\kappa k\delta_{xx})\delta_{tt}u = &-\kappa^2\left(\delta_{xxxx}u + 2\overline{q}^2\delta_{xx}u + \overline{q}^4 u\right)\\ &+\gamma^2 q^2(\alpha+(1-\alpha)\mu_{t\cdot})\left(\delta_{x-}\zeta - u\right) - 2\sigma_t\delta_{t\cdot}u,\end{aligned} \tag{4.7a}$$

$$(1+\theta\gamma^2k^2\delta_{xx})\delta_{tt}\zeta = \gamma^2(\alpha+(1-\alpha)\mu_{t\cdot})\left(\delta_{xx}\zeta - \delta_{x+}u\right) - 2\sigma_l\delta_{t\cdot}\zeta, \tag{4.7b}$$

Equation 4.7 represents a discretization of the continuous model shown in Equation 4.2b, where $u = u_l^n$ and ζ_l^n are grid functions that approximate the continuous functions $u = u(x,t)$ and $\zeta = \zeta(x,t)$, see Chapter 2. This implicit scheme can be tuned via 4 free parameters, α, $\overline{q}$, η and θ. The parameter $\overline{q}$ must be chosen as a discrete approximation of q. Furthermore, in [4] it is mentioned that "it is wise" to choose $\alpha = 1/2$, while $\overline{q}$ is chosen as:

$$\overline{q} = \frac{2}{h}\sin(qh/2) \tag{4.8}$$

This leaves the two other parameters to be tuned in order to get a most accurate fit between the model and numerical dispersion.

4.3.1 Stability Conditions

Before going into the details of this tuning, it is important to present the stability conditions of this scheme. These were calculated in [2] based on energy analysis methods to be:

$$h \geq 2\gamma k\sqrt{\theta^+} \tag{4.9a}$$

$$h \geq \sqrt{\kappa k\left(2\eta^+ + \sqrt{4(\eta^+)^2 + (1+|\cos(qh)|)^2}\right)}, \tag{4.9b}$$

where $\eta^+ = (\eta + |\eta|)/2$ and $\theta^+ = (\theta + |\theta|)/2$, describing the positive parts of η and θ respectively.

As was the case for the previous FDSs modelling other acoustic systems, the best accuracy is found when the stability conditions are satisfied with equality.

4.3.2 Numerical Dispersion

Similar to what was done in Section 3.2, the numerical dispersion can be evaluated in a similar manner as the dispersion relationship of the continuous model by discretizing the test solutions used in Section 4.2. This means: $u_l^n = Ue^{j(\omega kn+\beta lh)}$ and $\zeta_l^n = Ze^{j(\omega kn+\beta lh)}$, with U and Z being some constants. Furthermore, the difference and averaging operators in the FDS presented in Equation 4.7a and Equation 4.7b are expanded and factorized with respect to the terms U and Z. This allows for the system of equations to be written in matrix form as:

$$\mathbf{S}\begin{bmatrix} U \\ Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{4.10}$$

where the matrix $\mathbf{S}$ is a 2x2 matrix whose entries contain the terms ω, β and all the physical parameters of the system as well as the step parameters of the FDS, h and k. The actual matrix is tedious to write down here, but the reader is referred to [12], where the script "helical_spring_dispersion_analysis.py" implements a script based on the symbolic library SymPy ([11]) that calculates this matrix and its subsequent determinant. One can print it in the console of a Python IDE for inspection.

As was the case for the continuous model, when the determinant of the matrix $\mathbf{S}$ is 0, Equation 4.10 has an infinite number of nontrivial solutions. The form of this determinant is again very complex and the SymPy solver fails to converge on a symbolic solution $\omega = \omega(\beta)$ that satisfied $\det(\mathbf{S}) = 0$. However, one can find these solutions numerically. What can be done is to calculate the determinant of $\mathbf{A}$ for ω values going from $[0, 2\pi f_s/2]$ (the entire audio bandwidth) for each value of β. For this particular implementation, a step size of 252.27 rad/s is chosen. The ω values where $\det(\mathbf{S}) = 0$ (or at least closest to zero, to be discussed in the following) are the solutions for a given β. It is a brute-force approach basically.

In order to illustrate this approach, the numerical dispersion for a helical spring with the physical parameters $\kappa = 0.05$, $\gamma = 2000$, $q = 800$, will be compared to the model dispersion for different choices of the free parameters θ, η and $\overline{q}$. This

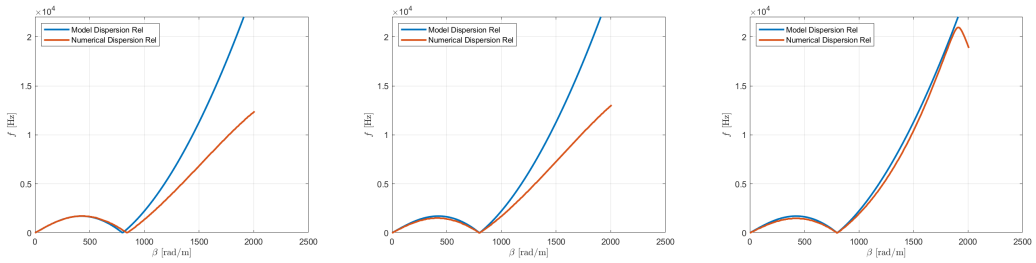


Figure 4.5: Comparisons of model dispersion of a helical spring with physical parameters $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$ with numerical simulations using different FDS free parameters. (a) $\overline{q} = q$, $\eta = \theta = 0$. (b) $\overline{q} = \frac{2}{h}\sin(qh/2)$, $\eta = \theta = 0$. (c) $\overline{q} = \frac{2}{h}\sin(qh/2)$, $\eta = 0.4313\theta = 0.000327$.

is illustrated in Figure 4.5 The step parameters of the FDS follow from the stability conditions in Equation 4.9.

It can be seen that using $\overline{q} = \frac{2}{h}\sin(qh/2)$ corrects the zero location of the numerical model at $\beta = q$, which is observed in the continuous model. Additionally, the η and θ parameters adjust the dispersive curve in the high frequency range in order to match the continuous model. Their values are in fact computed using an optimization procedure which will be described in the next section. It can be observed however that for this particular case the numerical model does not produce results in the entire audio bandwidth (up to $f = 22500$ Hz), as the numerical dispersion curve mirrors at around 21000 Hz about the β axis. However, this is beyond human perception range and this problem may also be solved with oversampling. All in all, there is a quite accurate agreement with the model dispersion.

Returning to how exactly the numerical dispersion is calculated, Figure 4.6 shows the value of $\det(\mathbf{S})$ for different ω and β values. This is illustrated for the system described in Figure 4.5, (c). One can see that for each considered β there is some ω which satisfies $\det(\omega) = 0$. In fact for some curves there seems to be a zero crossing close to $f_s/2$, and that could be expected as perhaps $\det(\omega) = 0$ may have multiple solutions for $\omega(\beta)$. For comparisons with the model dispersion, only the first solution is considered.

4.3.3 Optimization of FDS Free Parameters

When modelling a helical spring with new physical parameters, one needs to predict what the implicit FDS free parameter values of η and θ need to be. Being able to compare the numerical dispersion to the model dispersion for different choices of these parameters, as was illustrated in Figure 4.5, means that an optimization procedure can be introduced. A mean squared error loss function, $\mathcal{L}$, is introduced between the values of ω_{model}, resulting from the continuous model dispersion re-

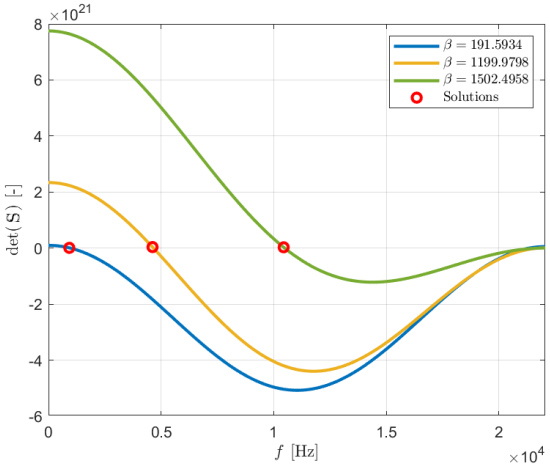


Figure 4.6: Resulting values for det($\mathbf{S}$) for various β values for the entire audio bandwidth. Zero crossings represent solutions to det($\mathbf{S}$) $= 0$, and form (ω,β) pairs in the numerical dispersion relationships.

lationship and ω_{FDS}, which results from the dispersion relationship of the FDS.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\omega_{\text{model},i}^2 - \omega_{\text{FDS},i}^2), \quad \text{where,} \qquad \omega_{\text{model},N} < 2\pi(\frac{f_s}{2} + 2000) \tag{4.11}$$

where i is an index of the ω values and $N+1$ is the index at which ω_{model} is bigger than the range of interest.

As can be seen in Equation 4.11, the loss is computed over a frequency interval from 0 to $f_s/2$ plus a 2 kHz bumper. This is done in order to "encourage" the optimization procedure to find optimal values of θ and η that give a good match over the entire bandwidth.

For the optimization algorithm, the aim is to minimize the loss function, i.e. the difference between the model and FDS dispersion relationship solutions. This is implemented via the built-in the Matlab ([10]) function "fminsearch" which uses the Nelder-Mead simplex algorithm as described in [9].

For the case presented in Figure 4.5, it took the iteration procedure 74 steps to converge. Figure 4.7 shows the progress of the numerical dispersion curve at different steps in the iteration.

4.3.4 Matrix Form

Being an implicit scheme, one needs to solve a system of equations in order to have an update equation for u_l^{n+1} and ζ_l^{n+1}. Hence we introduce the finite-length column

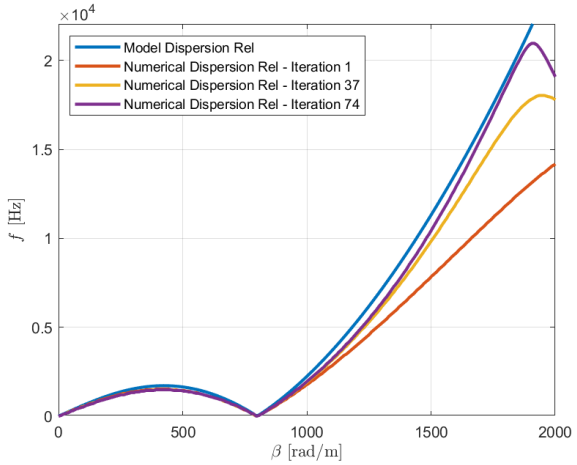


Figure 4.7: Model vs numerical dispersion relationship for a helical spring with $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$. The numerical dispersion curve resulting from η and θ parameters obtained at various steps in the optimization procedure.

vectors $\mathbf{u}^n = [u_0^n, ..., u_N^n]^T$ and $\boldsymbol{\zeta}^n = [\zeta_0^n, ..., \zeta_N^n]^T$, where T denotes the transpose. The scheme in Equation 4.7 can now be written in matrix form, after expanding the time difference operators as:

$$
\begin{aligned}
&(\mathbf{I} + \eta\kappa k\mathbf{D}_{xx})(1/k^2)(\mathbf{u}^{n+1} - 2\mathbf{u}^n + \mathbf{u}^{n-1}) + \kappa^2(\mathbf{D}_{xxxx}\mathbf{u}^n 2 + 2\overline{q}^2\mathbf{D}_{xx}\mathbf{u}^n + \mathbf{I}\overline{q}^4\mathbf{u}^n) \\
&- (q^2\gamma^2\alpha\mathbf{D}_{x-}\boldsymbol{\zeta}^n \\
&- q^2\gamma^2\alpha\mathbf{u}^n \\
&+ q^2\gamma^2(1-\alpha)\mathbf{D}_{x-}(1/2)(\boldsymbol{\zeta}^{n+1} + \boldsymbol{\zeta}^{n-1}) \\
&- q^2\gamma^2(1-\alpha)(1/2)(\mathbf{u}^{n+1} + \mathbf{u}^{n-1})) \\
&+ \sigma_t/k(\mathbf{u}^{n+1} + \mathbf{u}^{n-1}) = 0
\end{aligned}
\tag{4.12}
$$

$$
\begin{aligned}
&(\mathbf{I} + \theta\gamma^2k^2\mathbf{D}_{xx})(1/k^2)(\boldsymbol{\zeta}^{n+1} - 2\boldsymbol{\zeta}^n + \boldsymbol{\zeta}^{n-1}) \\
&- (+\gamma^2\alpha\mathbf{D}_{xx}\zeta^2 - \gamma^2\alpha\mathbf{D}_{x+}\mathbf{u}^n \\
&+ \gamma^2(1-\alpha)\mathbf{D}_{xx}(1/2)(\boldsymbol{\zeta}^{n+1} + \boldsymbol{\zeta}^{n-1}) \\
&- \gamma^2(1-alpha)\mathbf{D}_{x+}(1/2)(\mathbf{u}^{n+1} + \mathbf{u}^{n-1}) \\
&- (\sigma_l/k)(\boldsymbol{\zeta}^{n+1} + \boldsymbol{\zeta}^{n-1})) = 0
\end{aligned}
\tag{4.13}
$$

Again a Python script was developed, which can be found in [12]. It is used to factorize the two equations in Equation **??** with respect to u_l^{n+1},u_l^n, u_l^{n-1} and ζ_l^{n+1},ζ_l^n, ζ_l^{n-1}. This will result in the following system of equations in matrix form:

$$\begin{aligned}\mathbf{A_1}\mathbf{u}^{n+1} + \mathbf{B_1}\mathbf{u}^{n} + \mathbf{C_1}\mathbf{u}^{n-1} + \mathbf{D_1}\boldsymbol{\zeta}^{n+1} + \mathbf{E_1}\boldsymbol{\zeta}^{n} + \mathbf{F_1}\boldsymbol{\zeta}^{n-1} = 0,\\ \mathbf{A_2}\mathbf{u}^{n+1} + \mathbf{B_2}\mathbf{u}^{n} + \mathbf{C_2}\mathbf{u}^{n-1} + \mathbf{D_2}\boldsymbol{\zeta}^{n+1} + \mathbf{E_2}\boldsymbol{\zeta}^{n} + \mathbf{F_2}\boldsymbol{\zeta}^{n-1} = 0\end{aligned} \tag{4.14}$$

where $\mathbf{A_1}$ to $\mathbf{F_1}$ are the matrix resulting from factorization Equation **??**, while $\mathbf{A_2}$ to $\mathbf{F_2}$ result from the factorization of Equation **??**. These matrices will be the following, with $\mathbf{I}$ being the identity matrix:

$$\begin{aligned}\mathbf{A_1} &= (\mathbf{I} + \mathbf{I}0.5\gamma^2k^2q^2(1-\alpha) + k(\mathbf{D}_{xx}\kappa\eta + \mathbf{I}\sigma_t))/k^2\\ \mathbf{B_1} &= 2\mathbf{D}_{xx}\kappa^2\overline{q}^2 - 2\mathbf{D}_{xx}\kappa\eta/k + \mathbf{D}_{xxxx}\kappa^2 + \mathbf{I}\kappa^2\overline{q}^4 - 2\mathbf{I}/k^2 + \mathbf{I}\alpha\gamma^2q^2\\ \mathbf{C_1} &= \mathbf{I} + \mathbf{I}0.5\gamma^2k^2q^2(1-\alpha) + k(\mathbf{D}_{xx}\kappa\eta - \mathbf{I}\sigma_t))/k^2\\ \mathbf{D_1} &= 0.5\mathbf{D}_{x-}\gamma^2q^2(\alpha-1)\\ \mathbf{E_1} &= -\mathbf{D}_{x-}\alpha\gamma^2q^2\\ \mathbf{F_1} &= 0.5\mathbf{D}_{x-}\gamma^2q^2(\alpha-1)\\ \mathbf{A_2} &= 0.5\mathbf{D}_{x+}\gamma^2(1-\alpha)\\ \mathbf{B_2} &= \mathbf{D}_{x+}\alpha\gamma^2\\ \mathbf{C_2} &= 0.5\mathbf{D}_{x+}\gamma^2(1-\alpha)\\ \mathbf{D_2} &= (\mathbf{D}_{xx}\gamma^2k^2(0.5\alpha+\theta-0.5) + \mathbf{I} + \mathbf{I}k\sigma_l)/k^2\\ \mathbf{E_2} &= -(\mathbf{D}_{xx}\gamma^2k^2(\alpha+2\theta) + 2\mathbf{I})/k^2\\ \mathbf{F_2} &= (\mathbf{D}_{xx}\gamma^2k^2(0.5\alpha+\theta-0.5) + \mathbf{I} - \mathbf{I}k\sigma_l)/k^2\end{aligned} \tag{4.15}$$

Furthermore, the system of two matrix equations shown in Equation **??** can be in fact combined in one equation by introducing a state vector $\mathbf{w}^n = [u_0^n, ..., u_N^n, \zeta_0^n, ..., \zeta_N^n]^T$. We basically concatenate the two vectors $\mathbf{u}^n$ and $\boldsymbol{\zeta}^n$. Equation **??** becomes:

$$\begin{aligned}\mathbf{A}\mathbf{w}^{n+1} + \mathbf{B}\mathbf{w}^{n} + \mathbf{C}\mathbf{w}^{n-1} = 0, \quad \text{where,}\\ \mathbf{A} = \begin{bmatrix}\mathbf{A_1} & \mathbf{D_1}\\ \mathbf{A_2} & \mathbf{D_2}\end{bmatrix}\\ \mathbf{B} = \begin{bmatrix}\mathbf{B_1} & \mathbf{E_1}\\ \mathbf{B_2} & \mathbf{E_2}\end{bmatrix}\\ \mathbf{C} = \begin{bmatrix}\mathbf{C_1} & \mathbf{F_1}\\ \mathbf{C_2} & \mathbf{F_2}\end{bmatrix}\end{aligned} \tag{4.16}$$

Finally, the update equation for the state vector can be calculated using after inverting $\mathbf{A}$:

$$\mathbf{w}^{n+1} = \mathbf{A}^{-1}(-\mathbf{B}\mathbf{w}^{n} - \mathbf{C}\mathbf{w}^{n-1}) \tag{4.17}$$

Simply supported boundary conditions are considered for the model at the edges, i.e. $u = u_{xx} = \zeta = \zeta_{xx} = 0$ at $l = 0$ and $l = N$, where N is the number of discretized segments of the scaled helical spring. This means that the domain of

calculation will be $l \in [1, 2, ..., N-2, N-1]$, as values at the edges will always be 0. As a consequence, the $\mathbf{D}_{xxxx}$ operator needs to be adjusted in the first and last row as compared to the definition given in Section 2.4. This is the same approach that was done for the slinky spring in Chapter 3.

The loss parameters are chosen based on [4] as $\sigma_l = \sigma_t = 1.65$.

4.4 Output of the FDS

Similar to what was shown in Section 3.3, the aim is to produce an impulse response of the helical spring system. Once this is available, the "spring reverb" effect can be applied to some audio by convolving the input audio with the impulse response, a process named convolution reverb. See [14].

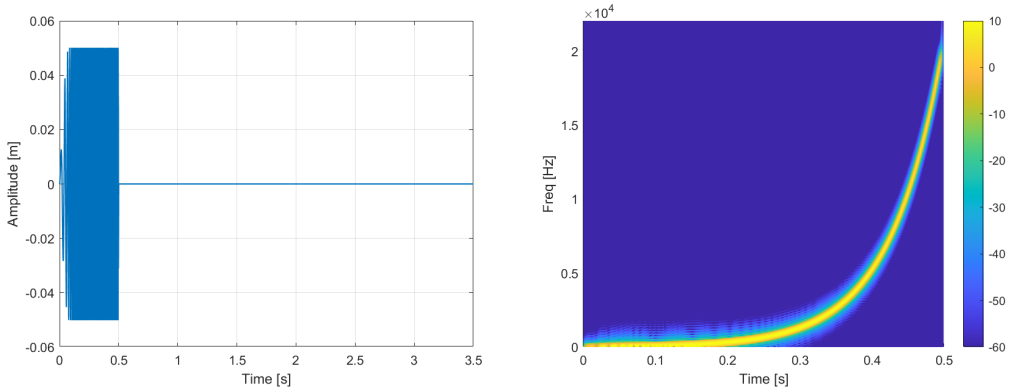
While for the slinky an ideal impulse was used for excitation, i.e., a Dirac function, for the case of the helical spring it is wiser to use a different method for extracting the impulse response. It is desired to have a broader range and higher quality impulse response, capable of adding some interesting texture to the processed sound. This can be achieved by using a longer sound to excite the helical spring, as opposed to the unrealistic ideal impulse. It is still desired to cover the entire audio bandwidth with the excitation signal, so an exponential sine sweep is chosen, covering the the human auditory perception range, 20 Hz to 20 kHz.

This is quite straight forward to implement in Matlab, using the function "sweeptone.m", where one specifies the duration of the sweep and the duration of the silence which follows the sweep. This duration will in fact be the duration of the impulse response, when using yet another Matlab function to process the output, "impzest.m", which estimates the impulse response of an audio system. The Matlab documentation references [6] as a source for these functions.

Figure 4.8, shows the time series and the spectrogram of such a sweep tone.

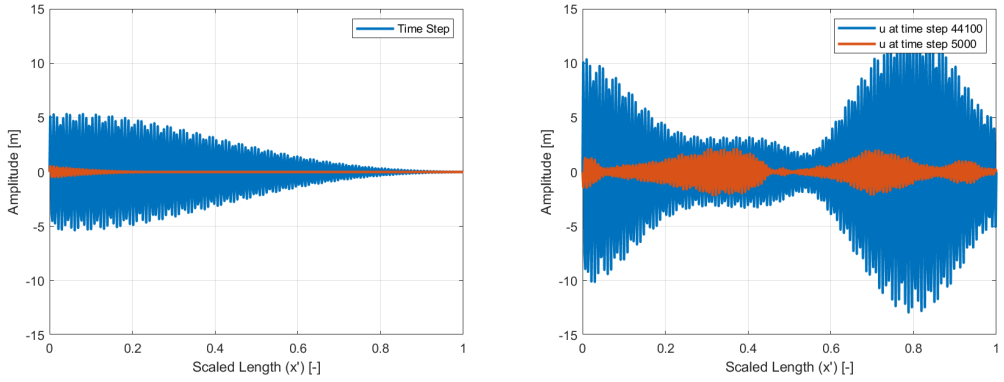
While Bilbao and Parker in [2], model a detailed excitation physical system, that more closely resembles the mechanism inside a traditional spring reverb unit, the work in the current project is limited at exciting the helical spring system as a forced displacement at the first free point. The output will then be "measured" at the other end of the spring, at the last free point. This is identical to what was shown in Section 3.3. Figure 4.9 shows the displacement along the helical spring with parameters $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$ at various time steps in the FDS. (a) shows two time steps close to the beginning of the excitation, where frequencies have yet to reach the opposite boundary and the dispersion effect is not yet very noticeable. (b) shows two later time steps where waves that travel at similar velocities have started to form wave groups. Furthermore, interference of incoming waves with reflected waves start to form as a result of the reflection from the right boundary.

It turns out that the longitudinal displacement, ζ at the output location is of



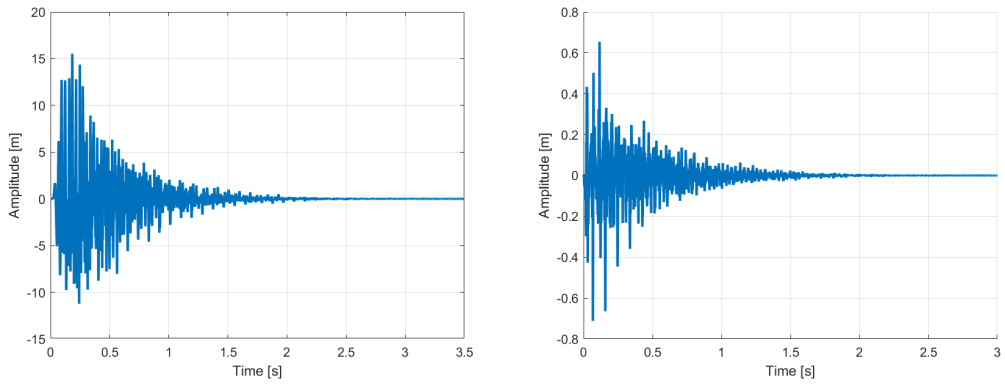
(a) **(b)**

Figure 4.8: (a) Time-series and (b) spectrogram of an exponential sweep tone ranging from 20 Hz to 20 kHz in 0.5 s, followed by 3 s of silence.



(a) **(b)**

Figure 4.9: Transverse displacement, u, during a numerical simulation of a helical spring excited at the left end with a sine sweep.



(a) **(b)**

Figure 4.10: (a) amplitude of the helical spring at the output location (right most free point), excited at the input location (left most free point) with an exponential sine sweep. (b) Impulse response of the helical spring, found by deconvolving the sine sweep signal from the output.

orders of magnitude lower than the transverse displacement, u, therefore the output will be taken solely with respect to u. Figure 4.10 (a) shows the amplitude of the output, while (b) shows the impulse response of the helical spring, obtained by deconvolving the sine sweep input from the output.

Finally we can visualize the effect of the interesting dispersion relationship of the helical spring system by looking at the spectrogram of the impulse response. This is illustrated in Figure 4.11 for two springs with different physical properties: (a) $\kappa = 0.05$, $\gamma = 2000$ and $q = 800$, which have been used so far in this section and (b) $\kappa = 0.0246$, $\gamma = 1905$ and $q = 1190$. Two distinct types of dispersion echoes are observed, one in the higher frequency range, above the transition frequency f_c, given in Equation 4.6, and one below. These reflect the group velocity curves of the helical spring model, an example of which was shown in Figure 4.4, from which the dispersive echoes can be estimated by knowing the time it takes to perform an end to end traversal of the lossless system: $T_d = 1/v_g(\omega)$, where $v_g(\omega)$ is the group velocity as a function of angular frequency.

We can see that there is some discrepancy between the dispersive echoes calculated based on the group velocity of the continuous model and what is observed in the spectrograms. This may actually be due to the fact that the echoes in the lower frequency range may be hidden due to the actual spectrogram algorithm, and may be improved with better windowing and perhaps zero padding of the bins. For the case where a higher f_c, Figure 4.11 (b), it can also be seen that the numerical model does not capture this transition frequency perfectly. This can be observed when looking at the model vs numerical dispersion comparison of this simulation, shown in Figure 4.12, where there is arguably not a great match. However, this

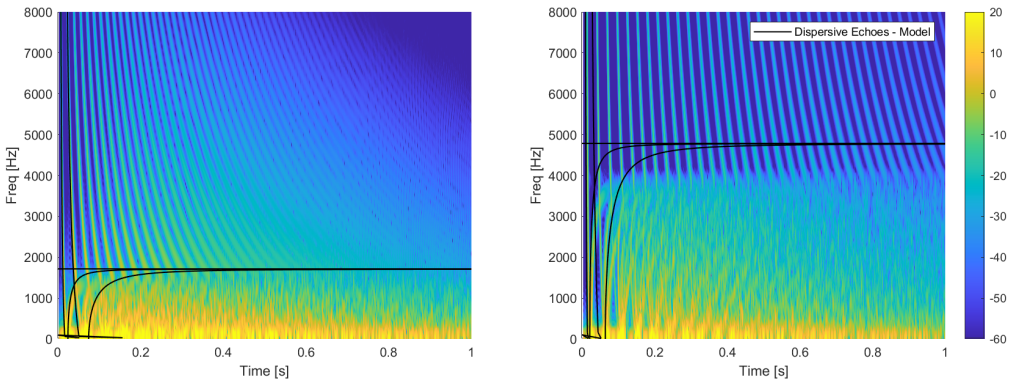


Figure 4.11: Spectrogram of the impulse response overlaid with the dispersive echoes as calculated based on the group velocity of the helical spring continuous model for a spring with the physical properties (a) $\kappa = 0.05$, $\gamma = 2000$, $q = 800$ and (b) $\kappa = 0.0246$, $\gamma = 1905$, $q = 1190$.

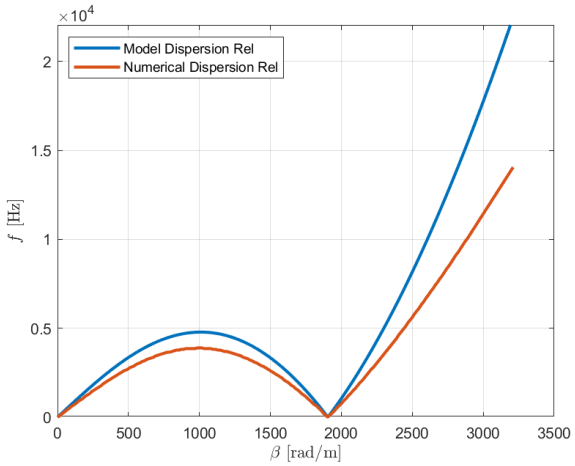


Figure 4.12: Model vs numerical dispersion relationship for a helical spring with $\kappa = 0.0246$, $\gamma = 1905$, $q = 1190$. The numerical dispersion curve resulting from η and θ parameters obtained via optimization procedure.

numerically generated impulse response still has a very interesting sonic texture, distinctly different from the model in Figure 4.11 (b). These can be found in [12] and the reader can judge for themselves.

Chapter 5

Outcome

The following chapter will discuss possible uses for the FDS implementation of the helical spring system presented in Chapter 4. It will try to touch upon what is realistic and what is not under the current conditions.

5.1 Database of Spring Impulse Responses

It turns out that very interesting numerical simulations of impulse responses of varying helical springs can be achieved using the FDS described in Chapter 4. However, it is clearly evident that the numerical dispersion is highly sensitive to the choice of the free parameters of the FDS. When one changes the physical parameters of the spring, with the purpose of changing its reverberation quality, these free parameters need to be recomputed. Similarly, the optimal spatial step of the scheme resulting from the stability conditions plays again a vital role in numerical dispersion. This means that changing the parameters of the spring reverb in real-time is unrealistic.

It is for this reason that it is decided to generate a database of impulse responses, where the physical properties of the helical springs considered are varied. It was seen in Chapter 4 that the shape of the dispersion relationship of the helical spring continuous model is dependent mainly on the normalized curvature of the spring q and its normalised stiffness measure, κ. The value of q gives the non-zero wave number β at which the dispersion relationship is 0 and κ (together with q) are used in Equation 4.6 to calculate f_c, the transition frequency above which the dispersion regime is monotonic. See Figure 4.3. The parameter γ has an effect on the density of echoes. It affects the group velocity of the physical system and thus affects the time it takes for the various frequencies to traverse the system. This will be further discussed shortly.

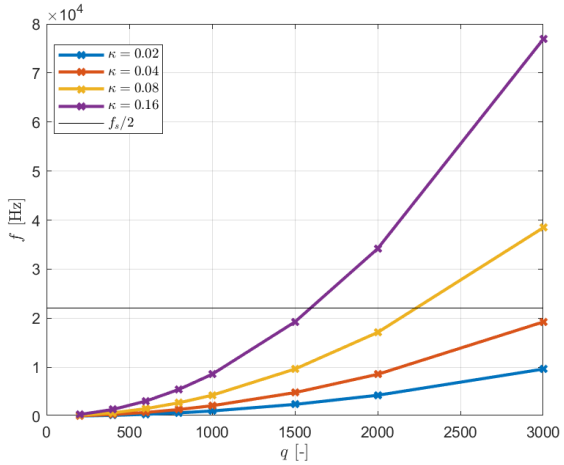


Figure 5.1: Resulting transition frequency f_c for various physical property combinations chosen for the database. As f_c is independent of γ, the curves will be the same for both considered cases.

Hence impulse responses are generated for all combinations of:

$$\begin{aligned} \kappa &\in [0.02, 0.04, 0.08, 0.16] \\ q &\in [200, 400, 600, 800, 1000, 1500, 2000, 3000] \\ \gamma &\in [1000, 1800] \end{aligned} \tag{5.1}$$

giving a total of 64 possible impulse responses of different perceptual quality. However, in some cases, the transition frequency f_c will be larger than the audio bandwidth, as illustrated in Figure 5.1. These cases will be disregarded.

Furthermore, it is found that for cases with large f_c, the optimization procedure presented in Section 4.3.3 fails to find values for the free parameters of the FDS η and θ that provide good agreement between model and numerical dispersion. In fact it is likely the the FDS itself cannot accurately model springs with these types of physical parameters. Such a comparison is shown in Figure 5.2.

Consequently, only the cases where $f_c < 10000$ Hz will be considered in the runs. This leaves a total of 52 impulse responses.

This database can be found in [12], together with time series plots of the impulse responses and spectrograms of the 1$^{\text{st}}$ second. Figure 5.3 shows a comparison of the spectrograms of two cases with $\kappa = 0.08$, $q = 1000$ and different γ values: 1000 and 1800 respectively. One sees that with lower gamma values the density of the echoes is slightly smaller. This has an interesting perceptual effect, as the lower density case sounds slightly more like an echo/delay, while the denser case has a more reverb like, windy, tail. It is interesting to see the the comparison of the model and numerical dispersion in the two cases. This is shown in Figure 5.4, where it can be seen than neither numerical simulation fits perfectly, but they still show the expected behavior and provide an interesting sound.

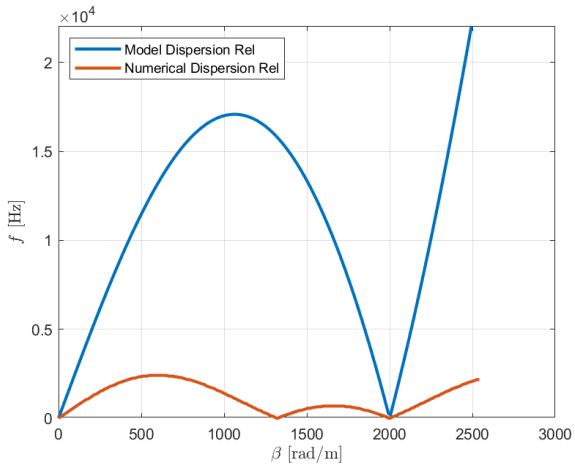
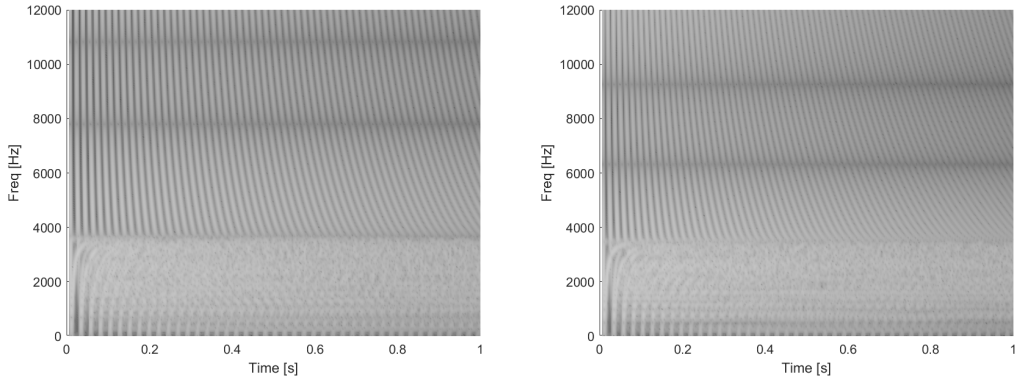


Figure 5.2: Model vs numerical dispersion relationship for a helical spring with physical parameters: $\kappa = 0.08$, $q = 2000$ and $\gamma = 1000$.



(a) **(b)**

Figure 5.3: Spectrogram of the 1st second of the impulse response for two helical spring systems: (a) $\kappa = 0.08$, $q = 1000$, $\gamma = 1000$ and (b) $\kappa = 0.08$, $q = 1000$, $\gamma = 1800$.

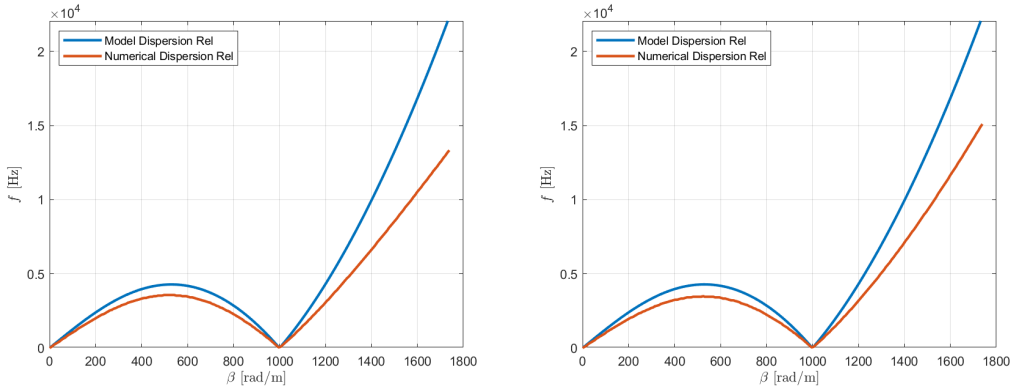


Figure 5.4: Model vs numerical dispersion for two helical spring systems: (a) $\kappa = 0.08$, $q = 1000$, $\gamma = 1000$ and (b) $\kappa = 0.08$, $q = 1000$, $\gamma = 1800$.

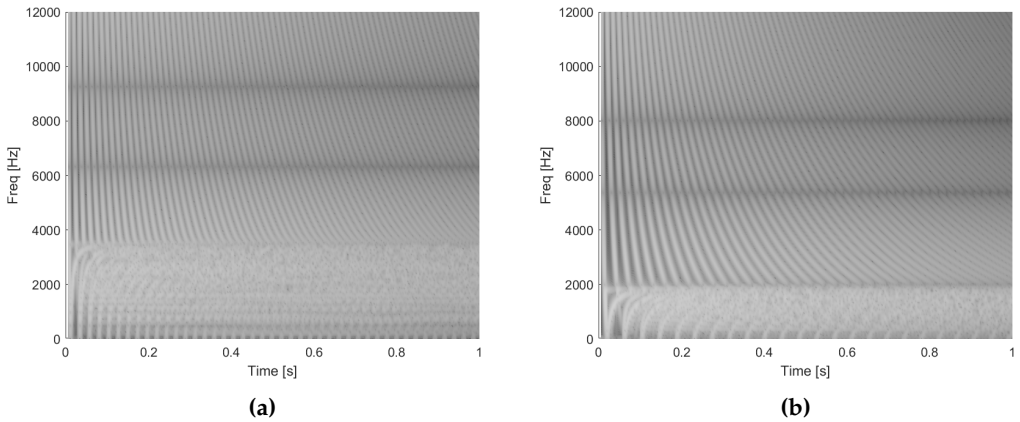


Figure 5.5: Model vs numerical dispersion for two helical spring systems: (a) $\kappa = 0.08$, $q = 1000$, $\gamma = 1800$ and (b) $\kappa = 0.04$, $q = 1000$, $\gamma = 1800$.

As yet another comparison, Figure 5.5 shows the spectrograms for the same system shown previously characterized by the physical parameters: $\kappa = 0.08$, $q = 1000$, $\gamma = 1800$, but now compared with a helical spring with a smaller stiffness: $\kappa = 0.04$, $q = 1000$, $\gamma = 1800$. Here we see that the response of the more flexible spring (b), has a lower transition frequency f_c and less dense echoes.

(nodes of modes ?!)

5.2 Application of Impulse Responses - Convolution

As mentioned previously, the way one typically uses impulse responses for reverberation is via convolution. This is a signal processing method, where one can determine the output of a linear time invariant system if one knows the impulse response of that system via the theory of convolution, [14]. In time domain this is expressed as:

$$y(n) = x(n) * h(n) = \sum_{m=-\infty}^{+\infty} (x[m]h[n-m]) \tag{5.2}$$

where x is the input to a system described by the impulse response h and y is its output.

In frequency domain convolution transforms into simple multiplication of values.

$$Y(\omega) = X(\omega)H(\omega) \tag{5.3}$$

where X is the frequency response of the input, H is the frequency response of the impulse response and finally Y is the frequency response of the output.

Traditionally convolution is better implemented in frequency domain as one can perform only two discrete Fourier transform (DFT) operations and one inverse disrete Fourier transform (IDFT) for computing the output. This can be done as such:

$$y = \text{IDFT}(\text{DFT}(x)\text{DFT}(h)) \tag{5.4}$$

However, the impulse responses generated from the helical spring systems presented in the previous section are very long, 132300 samples to be precise, sampled at 44100 Hz (3 seconds). This causes a lot of overhead and could add delays if one desires to process a stream of input audio in real time via the convolution process.

A better implementation of the convolution should be used, which is partitioned convolution. Such an implementation is described in [1], and consists of partitioning the impulse response, h, into equally sized blocks of equal reasonable length. Furthermore, these blocks are treated as separate impulse responses and are convolved by a standard overlap-and-save method, [14]. The input stream is also split into blocks, which allows for real time processing of say a microphone input. Figure 5.6 taken from [1] describes the process.

A prototype of such an implementation has been written in Matlab and can be found in [12], although not as a plugin.

In order to test that the idea works, a script has been developed in SuperCollider, which is a programming language, and IDE for sound synthesis and algorithmic composition. A partitioned convolution Unit Generator (UGen - basically a processing function) is already implemented there and is available to use. The

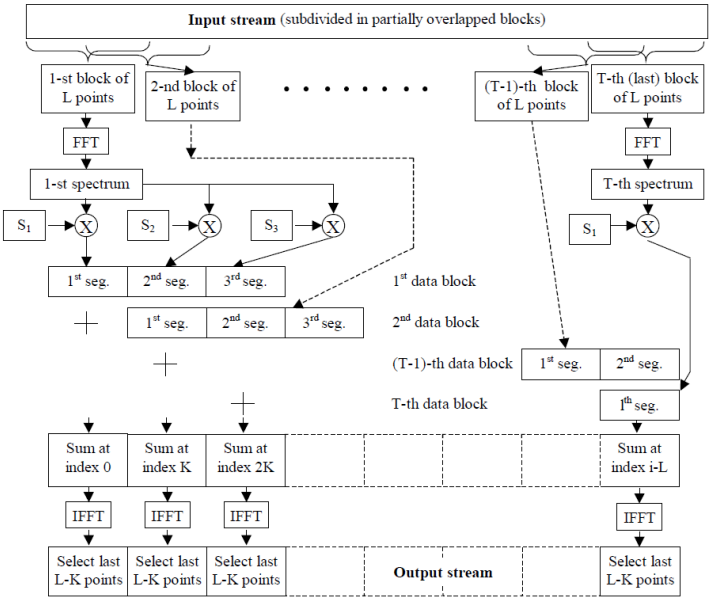


Figure 5.6: Partitioned convolution process overview, from [1].

script is again found in [12]. Unfortunately, one cannot freely change the impulse response used for convolution in real time, for example going from a more flexible to a stiffer spring response without stopping the audio. This could definitely be something to build on as future work.

A major advantage of the partitioned convolution method is the reduced latency as compared to standard convolution, driven by the size of the FFTs used. In the former case being driven by the chosen block size L, while in the latter case it will be $M + N - 1$, with M being the size of input x and N, the size of the impulse response h.

Chapter 6

Conclusion

6.1 Summary

The current report presents a physical modelling approach to implementing a digital spring reverberation simulation. This has been carried out by means of FDM, which have been described in Chapter 2.

Furthermore, in order to introduce the reader (and through this process, the author as well) to applications of FDSs, increasingly complex acoustic systems have been described and implemented in Chapter 3. Important general concepts related to FDM have been introduced here, such as stability analysis, model and numerical dispersion, choice of boundary conditions, and pros and cons of explicit and implicit schemes.

With this foundation, a detailed model describing the behavior of helical springs typically used in traditional spring reverberation units has been presented in Chapter 4, together with an implicit FDS aiming to accurately simulate such a system. The sensitivity of the model with respect to dispersion was highlighted, and an optimization scheme aiming to minimize the difference between the dispersion of the continuous model and the numerical simulation has been described. The outputs of these implementations have then been illustrated and discussed.

The application of the helical spring FDS is discussed in Chapter 5, where the computational cost is highlighted and an alternative to a real-time implementation is presented. A number of helical springs with different physical properties, carefully chosen as to produce different sonic qualities, are simulated and a database of impulse responses is created, which can be found at [12]. Lastly, a discussion of how these impulse response can be used to add the spring reverb quality to some live incoming audio, for example from a microphone, or some DAW.

6.2 Future Work

There are of course many directions in which future work related to this project may go. Possible improvements can be made to the helical spring FDS implementation, such as: a more realistic excitation model, which would include the magnetic beads fount in spring reverb units, similar to that used by Bilbao and Parker in [2] or an upgraded optimization technique for determining the best values for the free parameters of the implicit scheme. The implementation could be coded in a programming language which is better suited for audio processing such as C++.

Extensions of the model can be made by including additional options for boundary conditions or by introducing the possibility of coupling helical springs either in series or in parallel. Coupling could also be extended to other acoustic systems, such as the ones described in Chapter 3.

Some more creative directions could be to investigate possibilities of tuning the helical spring system and use it as a sound producing instrument (who needs strings?).

Lastly, one can take the implementation presented here one step further and (with the knowledge gained so far) build a new one based on a more state of the art helical spring model, such as the one in [3], where the helical spring is described in 3-D by a coupled system of 4 partial differential equations, accounting for 12 variables, respectively displacements, rotation angles, moments and forces in 3-D. Definitely a step up to the model shown here.

Bibliography

[1] Enrico Armelloni, Christian Giottoli, and Angelo Farina. "Implementation of real-time partitioned convolution on a DSP board". In: *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)* (2003), pp. 71–74.

[2] S. Bilbao and J. Parker. "A Virtual Model of Spring Reverberation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.4 (2010), pp. 799–808.

[3] Stefan Bilbao. "NUMERICAL SIMULATION OF SPRING REVERBERATION". In: Jan. 2013.

[4] Stefan Bilbao. *Numerical Sound Synthesis*. John Wiley and Sons, Ltd, 2009. ISBN: 978-0-470-51046-9.

[5] Stefan Bilbao et al. "The NESS Project: Physical modeling, algorithms and sound synthesis". English. In: *Computer Music Journal* (Nov. 2019). ISSN: 0148-9267.

[6] Angelo Farina. "Advancements in impulse response measurements by sine sweeps". In: (Sept. 4).

[7] William Fetterman. *John Cage's Theatre Pieces*. Routledge, 1996. ISBN: 3-7186-5642-6.

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

[9] Jeffrey Lagarias et al. "Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions". In: *SIAM Journal on Optimization* 9 (Dec. 1998), pp. 112–147. DOI: 10.1137/S1052623496303470.

[10] MATLAB. *version (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2019.

[11] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: 10.7717/peerj-cs.103. URL: https://doi.org/10.7717/peerj-cs.103.

[12] Marius Onofrei. *Physical-Modelling-Of-Spring-Reverberation*. https://github.com/mariusono/Physical-Modelling-of-Spring-Reverberation. 2020.

[13] *PA4 Dual Spring Reverb*. URL: https://physicalaudio.co.uk/.

[14] Tae Hong Park. *Introduction To Digital Signal Processing: Computer Musically Speaking*. World Scientific Publishing Co Pte Ltd, 2008. ISBN: 9789812790279.

[15] J. Parker. "Dispersive Systems in Musical Audio Signal Processing". PhD thesis. Aalto University, 2013.

[16] Julian Parker and Stefan Bilbao. "Spring reverberation: A physical perspective". In: (Sept. 2009).

[17] Julian Parker et al. "Modeling methods for the highly dispersive slinky spring: A novel musical toy". In: (Oct. 2).

[18] Silvin Willemsen. *Lecture notes in Physical Models for Sound Synthesis*. 2020.