The Emulated Ensemble

Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods

Ph.D. Dissertation

Silvin Willemsen

Dissertation submitted June 27, 2021

Thesis submitted: June 27, 2021

PhD Supervisor: Prof. Stefania Serafin
Aalborg University

PhD Committee: Prof. X, Y University
Prof. X, Y University
Prof. X, Y University

PhD Series: Technical Faculty of IT and Design, Aalborg University

Department: Department of Architecture, Design and Media Technology

ISSN: xxxx-xxxx
ISBN: xxx-xx-xxxx-xxx-x

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2021

Curriculum Vitae

Silvin Willemsen

Here is the CV text.

Acknowledgements

I would like to thank my mom..

Both Stefan Bilbao and his seminal work *Numerical Sound Systhesis* have been invaluable to the result of this project...

Finally, I would like to wholeheartedly thank my supervisor Stefania Serafin for essentially my entire PhD project. I quickly found out that this project was a "golden ticket" to do whatever as long as it is publishable. You provided this opportunity and let me be free in my decisions throughout my project and I am extremely grateful for that.

List of Publications

Listed below are the publications that I (co)authored during the Ph.D. project. These are grouped by: the main publications, which are also included in Part IV, papers where I had a supervisory role, and finally, other publications from various collaborative efforts.

Main Publications

[A] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," in *Proceedings of the 16th Sound and Music Computing (SMC) Conference*, 2019, pp. 275–280.

[B] S. Willemsen, S. Bilbao, N. Andersson, and S. Serafin, "Physical models and real-time control with the sensel morph," in *Proceedings of the 16th Sound and Music Computing (SMC) Conference*, 2019, pp. 95–96.

[C] S. Willemsen, S. Bilbao, and S. Serafin, "Real-time implementation of an elasto-plastic friction model applied to stiff strings using finite difference schemes," in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019, pp. 40–46.

[D] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 161–168.

[E] S. Willemsen, R. Paisa, and S. Serafin, "Resurrecting the tromba marina: A bowed virtual reality instrument using haptic feedback and accurate physical modelling," in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 300–307.

[F] S. Willemsen, A.-S. Horvath, and M. Nascimben, "Digidrum: A haptic-based virtual reality musical instrument and a case study," in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 292–299.

[G] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "Dynamic grids for finite-difference schemes in musical instrument simulations," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21)*, 2021.

[H] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "A physical model of the trombone using dynamic grids for finite-difference schemes," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21)*, 2021.

Publications with a Supervisory Role

[S1] R. S. Alecu, S. Serafin, S. Willemsen, E. Parravicini, and S. Lucato, "Embouchure interaction model for brass instruments," in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 153–160.

[S2] T. Lasickas, S. Willemsen, and S. Serafin, "Real-time implementation of the shamisen using finite difference schemes," in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021.

[S3] M. G. Onofrei, S. Willemsen, and S. Serafin, "Implementing complex physical models in real-time using partitioned convolution: An adjustable spring reverb," in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021.

[S4] M. G. Onofrei, S. Willemsen, and S. Serafin, "Real-time implementation of an elasto-plastic friction drum using finite difference schemes," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21)*, 2021.

Other Publications

[O1] J. M. Hjerrild, S. Willemsen, and M. G. Christensen, "Physical models for fast estimation of guitar string, fret and plucking position," *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 155–159, 2019.

[O2] K. Prawda, S. Willemsen, S. Serafin, and V. Välimäki, "Flexible real-time reverberation synthesis with accurate parameter control," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, 2020.

[O3] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, "Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation," *Journal of the Acoustical Society of America (JASA)*, vol. 149, pp. 3502–3516, 2021.

Abstract

Physical modelling is a technique /approach to synthesise sound. It is widely accepted that physical modelling is the best way to realistically and naturally simulate real-world musical instruments [1, 2, 3]. As this technique simulates the instrument based on its physics rather than using pre-recorded samples, it is more flexible to player-interaction and thus more realistic when synthesising sound in performance. Although physical models could potentially sound indistinguishable from the instrument that they are simulating, it has been impossible, until recently, to make highly physically accurate physical models 'playable' in real-time [4]. With the computational power we currently possess, we can run the simulations in real-time and make them available for musicians in latency-less applications. These applications include digital instrument plug-ins that can be used by music producers, but also resurrect old or rare instruments that can not be played anymore due to damage, or because they are too valuable.

directly copy-pasted, also contains citations, etc.

Abstract

Resumé

Danish Abstract

Contents

FULL DOC SWEEP: check capitalisation of headings throughout document

FULL DOC SWEEP: check hyphen in titles

Todo list

Preface

Starting this Ph.D. project, I did not have a background in mathematics, physics or computer science, which were three equally crucial components in creating the result of this project. After the initial steep learning curve of notation and terminology, I was surprised to find that the methods used for physical modelling are actually quite straightforward!

Of course it should take a bit of time to learn these things, but

Many concepts that seemed impossible at the beginning

I feel that the literature lacks a lot of the intuition needed for readers without a background in any of these topics. Rather, much of the literature I came across assumes that the reader has a degree in at least one of the aforementioned topics. Stefan Bilbao's seminal work *Numerical Sound Synthesis*, which is the most complete work to date describing how to physically model musical instruments using finite-difference time-domain methods says that "A strong background in digital signal processing, physics, and computer programming is essential." Even though some basic calculus knowledge is assumed to understand the concepts used in this work, a degree in any of the aforementioned topics is (hopefully) unnecessary. Furthermore, some experience with MATLAB and C++ is assumed for the code examples

Some working titles: *Physical Modelling for Dummies Physical Modelling for the faint-hearted*

Also, I came across a lot of "it can be shown that's without derivations. This is why I decided to write this work a bit more pedagogical, and perhaps more elaborate than what could be expected.

I believe that anyone with some basic skills in mathematics and programming is able to create a simulation based on physics within a short amount of time, given the right tools, which I hope that this dissertation could be.

The knowledge dissemination of this dissertation is thus not only limited to the research done and publications made over the course of the project, but also its pedagogical nature hopefully allowing future (or past) students to benefit from.

As with a musical instrument itself, a low entry level, a gentle learning curve along with a high virtuosity level is desired. Take a piano, for instance.

Most will be able to learn a simple melody — such as "Frère Jacques" — in minutes, but to become virtuous requires years of practice.

This is the way I wanted to write this dissertation: easy to understand the basic concepts, but many different aspects touched upon to allow for virtuosity. Hopefully by the end, the reader will at least grasp some highly complex concepts in the fields of mathematics, physics and computer science (which will hopefully take less time than it takes to become virtuous at playing the piano).

As Smith states in his work *Physical Audio Signal Processing* [**?**] "All we need is Newton", and indeed, all Newton's laws of motion will make their appearance in this document.

I wanted to show my learning process and (hopefully) explain topics such as *Energy Analysis*, *Stability Analysis*, etc. in a way that others lacking the same knowledge will be able to understand.

Make physical modelling more accessible to the non-physicist.

Interested in physically impossible manipulations of now-virtual instruments.

Silvin Willemsen
Aalborg University, June 27, 2021

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

Digital sound synthesis an increased interest in the last few decades. In the 1960s, efficient sinusoidal-based sound synthesis techniques such as additive synthesis [**?**], or FM (frequency modulation) synthesis [5] were invented. The latter became widely popular through the Yamaha DX7 synthesiser created in 1983 that synthesised sounds based solely on this technique [6]. Through a simple change of variables the same formula could generate sounds ranging from brass instruments to drums.

As computing power increased, so did the popularity of using physics-based simulations of musical instruments. Most likely the very first example of a physically modelled sound example is the "Bicycle Built for Two" by Kelly, Lochbaum, and Matthews in 1961[1]. It uses what later got known as the Kelly-Lochbaum vocal-tract model to generate a voice and was published the year thereafter [7].

to more to more computationally demanding techniques in the

The simulation[2] of traditional musical instruments has seen. From

Stefania, I need you here for the proper references :)

The interest in creating virtual or digital musical instruments

Virtualisation or digitisation

A simulation of a musical instrument is defined as an implementation of

where the sound is generated on the spot. Sample-based sound synthesis, where the sound of a real instrument has been recorded and is played back through a digital device is not considered a simulation.

The umbrella term would be digital musical instrument,

[1] http://ccrma.stanford.edu/ jos/wav/daisy-klm.wav

[2] The term *emulated* is only used in the title of this work (because of the alliteration), but is synonymous to *simulated* in this context.

1.1 Exciter-Resonator Approach

Nearly any musical instrument can be subdivided into a resonator component, an exciter component, and the interaction between them. This modular approach to musical instruments was first introduced by Borin, De Poli and Sarti in [8] and is used to structure this work . Examples or resonator-exciter combinations are the violin and the bow, or the trumpet and the lips of the player. Note that Part **??** does not include the interactions between the resonator and exciter, but rather the interactions between different parts of the resonator itself, for example, the interactions between the string and the body of a violin, which both are resonators.

interactions in Borin1989 are not the same as the interactions "Part" here..

reread

A resonator is a passive system, in this work mostly assumed to be linear, that does not emit sound unless triggered by an external source. Exciters can be seen as these external sources, and generally have a nonlinear element. Exciters insert energy into a resonator and cause it to vibrate and emit sound.

In the real world, the interaction between the exciter and the resonator is bi-directional, hence called an interaction. In other words, the exciter not only affects the state of the resonator, but the resonator affects the exciter as well. For the most part, this is also what is attempted to model in this work.

1.2 Physical Modelling Techniques

The time-evolution of dynamic systems, including that of musical instruments, can be well described by partial differential equations (PDEs) [9, 3]. Examples of a dynamic systems are a guitar string, a drum-membrane, or air propagation in a concert hall; three very different concepts, but all based on the same types of equations of motion. Many of these equations and other knowledge currently available on the physics of musical instruments have been collected by Fletcher and Rossing in [9]. Though these equations are very powerful, only few have a closed-form solution, and in order for them to be implemented, they need to be approximated. In the past decades, much research has been done on implementing these PDEs to model and simulate different musical instruments. Great overviews of implementation techniques are given by, for example, Vesa Välimäki et al. in [1] and Julius O. Smith in [4, 2].

The most popular physical modelling techniques that are described in this literature can be found below:

Modal Synthesis decomposes a system into a series of uncoupled 'modes of vibration' and can be seen as a physically-based additive synthesis technique. First used in a musical context by Morrison and Adrien in [10], it is a technique that is still used today due to its computational efficiency, especially when

simulating higher dimensional systems such as (two-dimensional) plates or (three-dimensional) rooms. It is especially effective when used to describe a linear system with a small number of long-resonating modes [11, 4]. When used to describe nonlinear systems, however, the modes become 'coupled' and the system will quickly become more computationally expensive. Recent developments using the FAUST programming language allow a 3D-mesh model of any three-dimensional object to directly be decomposed into its modes of vibration [12].

Finite-Difference Time Domain methods (FDTD) aim to solve PDEs by approximating them with difference equations, discretising a continuous system into grid-points in space and time. In a musical context, this technique was first used for the case of string vibration in [13, 14, 15] and later in [16, 17]. Stefan Bilbao extensively describes this method in [3, 11]. Although computationally expensive, especially when working with higher-dimensional systems, this technique can accurately model any system, whether it is linear or nonlinear, time-invariant or time-variant.

Digital Waveguide Modelling (or Digital Waveguides (DWGs)) is a modelling technique that discretises wave propagation and scattering. The technique was first presented in [18], and is mostly used for one-dimensional systems, such as strings and acoustic tubes and decomposes their system into travelling wave components. This technique has also been used in higher-dimensional systems, but is superior in efficiency when used in the one-dimensional case [1]. Some authors have combined DWGs with FD schemes (such as in [19, 20]) to accurately model nonlinear behaviour while maintaining high-speed implementation.

Mass-spring networks can be similar in nature to FDTD methods, but treat each grid point as an individual mass connected to other masses through springs in a network. Pioneered in a musical context by Cadoz in [21, 22, 23] it is currently being further developed by Leonard and Villeneuve in a real-time, interactive environment [24, 25].

Other techniques include Functional Transformation Method [26], state-space modelling [27], wave-domain modelling and energy-based port-Hamiltonian

should I include this

This work focuses on physical modelling using FDTD methods. The main advantage of these methods is that they are extremely general and flexible in terms of the types and amount of systems they can model. They allow any set of PDEs can be directly numerically simulated without making any assumptions regarding travelling wave solutions or modes. DWGs, for example, assume a travelling wave solution, which makes dispersive effects, let alone nonlinear effects, extremely hard to model using this technique. To use modal

synthesis to model a PDE, it requires the system to have closed-form or analytical solution. If this is not available, (finite-element) analysis of the system could be performed to obtain the modal shapes and frequencies of the system. This in itself is very computationally expensive and requires a lot of storage if the modal data needs to be saved.

check all of this

Moreover, FDTD methods allow for various PDEs, fx. a violin body and four strings, to be connected in a fairly straightforward manner.

The main drawback of FDTD methods is the fact that working with these methods requires great attention to numerical stability of the solution. For a wrong choice of parameters, the implemented system could become unstable and "explode"[3]. Stability analysis as well as energy analysis techniques are invaluable in the process of ensuring a stable implementation and much attention to this will be given throughout this work.

A final drawback of using FDTD methods is that – especially for higher-dimensional systems – they require much more computationally heavy than other methods, such as DWGs or modal synthesis techniques. The bright side, if one believes in Moore's law [28], is that it can be assumed that computing power will continue to increase and that within several years, running high-quality simulations of musical instruments based on FDTD methods in real time, would not be an issue. More information on real-time implementation is given below.

1.3 Real-Time Implementation

Although many techniques to digitally simulate musical instruments exist

proving that we have only recently reached the computing power in personal computers to make real-time playability of these models an option. The biggest challenge in real-time audio applications as opposed to those only involving graphics, is that the sample rate is extremely high. As Nyquist's sampling theory tells us, a sampling rate of at least 40 kHz is necessary to produce frequencies up to the human hearing limit of 20 kHz **[Nyquist]**. Visuals

Even though physical modelling has been a popular research field in the past few decades, relatively little research has been done on making the models work in real-time, i.e., 'playable' **[?]**. Several virtual string instruments and different electric pianos have been made real-time by Pfeifle and Bader in **[?, ?, ?]**. They used field programmable gate arrays (FPGAs) for implementing models based on FDTD methods. Furthermore, Roland's V-series use COSM (Composite Object Sound Modelling) technology **[?]** that implement real-time physical models in hardware instruments. In the NESS project [29, 30], Stefan

[3] I learned the hard way that one should always implement a limiter when working with real-time physical modelling using FDTD methods.

Bilbao and his team focused on implementing systems using FDTD methods in real-time.

Real-time: no noticeable latency

1.4 Why?

So why would we go through all this hassle of modelling musical instruments? Could we not use a recording of the original and play that back at the right moment? Or taking another step back, why not buy a real instrument and learn to play that instead? "I'm a musician. Will I be out of a job if you keep making physical models?"

exactly as in [3]

Physical modelling is not here to replace the original instruments and the musicians playing them. Instead, it can be used as a tool to understand the physics of existing instruments and possibly go beyond. Simulated instruments are not restricted by physics anymore and could provide new ways of expression for the musician.

1.4.1 Samples vs. Physical Modelling

Digital musical instruments based on recordings of an actual instrument, referred to as *samples*, have an advantage of having an optimally realistic sound. As the output of the digitised instrument is exactly that of the original instrument, the digital version should sound indistinguishable from the original. Furthermore, using samples is extremely computationally efficient as it simply requires loading the data and playing this back to the player.

That said, these are the only advantages of using samples over physical models for digitising a musical instrument. Samples are static and unable to adapt to changes in performance; the recording is made by one player, using a specific microphone and location. Moreover, capturing the the entire interaction space of an instrument is nearly impossible. Imagine recording a violin with every single combination of bowing force, velocity, position, duration and other aspects such as vibrato, pizzicato. Even if a complete sample library could be created, this would contain an immense amount of data. Using physical models to simulate the musical instrument, on the other hand, allows the sound to be generated on the spot based on physical parameters that the user can interact with. The

Trade off between storage and speed, or hard-disk and processing power.

1.4.2 Resurrect Old or Rare Instruments

Why not use the real instrument in the first place? In some cases, recording samples of an instrument is not even possible as they are too old, too rare

or too vulnerable to be played. The physics of the instrument, including the geometry and material properties, are available, and could potentially be modelled digitally.

Even popular instruments require maintenance and might need to be replaced after years of usage.

FULL DOC SWEEP: plug-ins or plug-ins

Entire orchestras come from plug-ins

1.4.3 Go beyond what is physically possible

check wording

As a virtual instrument is not restricted to the laws of physics in the real world, this opens up a world of possibilities.

Musical instrument simulations make it possible for parameters like shape, size, material properties, etc. to be dynamically changed, which is physically impossible or very hard to do. Furthermore, different instrument components can be combined to create hybrid instruments. For example, one could bow the air in a trumpet, or lip-excite a string (similar to what Smith states in [4]). This could potentially resulting in unique sounds that can only be created using physical models.

1.5 Project Objectives and Main Contributions

Over the past few decades, much work has been done on the accurate modelling of physical phenomena. In the field of sound and musical instruments..

From [9] to [30]

The main objective of this thesis is to implement existing physical models simulated using FDTD methods in real time. Many of the physical models and methods presented in this thesis are taken from the literature and are thus not novel.

Secondly, to combine the existing physical models to get complete instruments and be able to control them in real time.

As FDTD methods are quite rigid, changing parameters on the fly, i.e., while the instrument simulation is running, is a challenge. Other techniques, such as modal synthesis, are much more suitable for this, but come with the drawbacks mentioned in Section 1.2. Therefore, a novel method was devised to smoothly change parameters over time, introducing this to FDTD methods.

Put this work into perspective of the literature (higher level)

1.6 Thesis Outline

The dissertation is divided into several parts which in their turn are divided in chapters.

Part I: Introduction introduced the field of physical modelling for musical instruments in this chapter by giving a brief history of the field and provides and background for the project. Furthermore, the project objectives and contributions to the field have been detailed. Chapter 2 will provide a thorough introduction to finite-difference time-domain methods using simple sound-generating systems as examples, after which Chapter 3 will introduce several analysis techniques in a tutorial-like fashion. This part has – as much as possible – been written in layman's terms and perhaps more pedagogical than could be expected of a dissertation.

Part II: Resonators introduces various physical models in isolation that have been used extensively throughout the project.

Part ??: Exciters shows two different ways that the resonators introduced in Part II can be excited.

Part ??: Interactions shows two different ways that the resonators can interact with each other:

Introduction to finite-difference methods and analysis techniques

Less focus on continuous equations and mathematical substantiation, but more on the practical and implementation side of things.

Despite the "collection of papers" format that I chose to use for this work, the style of

Models used over the course of the project divided into resonators in part II, exciters in part **??** and the interactions between them in **??**.

Focus on real-time implementation and control of the models in part **??**

Chapter 2

Introduction to Finite-Difference Time-Domain Methods

"Since Newton, mankind has come to realize that the laws of physics are always expressed in the language of differential equations."
- Steven Strogatz

This chapter introduces some important concepts needed to understand finite-difference time-domain (FDTD) methods. These techniques are what the implementation of the physical models presented later on in this document are based on. By means of a simple mass-spring system and the 1D wave equation, the notation and terminology used throughout this document will be explained. Unless denoted otherwise, the notation has been taken from [3], most of which originates from [31].

2.1 Differential Equations

Differential equations are used to describe the motion of dynamic systems, including vibrations in musical instruments. In this work, these equations are used to describe, among others, the movement of a string, an instrument body and the air pressure in an acoustic tube.

A characteristic feature of these equations is that, rather than an absolute value or *state* of a system, the time derivative of its state – its velocity – or the second time derivative – its acceleration – is described. From this, the absolute state of the system can then be computed. This state is usually described by

the variable u which is always a function of time, i.e., $u = u(t)$. If the system is distributed in space, u also becomes a function of space, i.e., $u = u(x, t)$, or with two spatial dimensions, $u = u(x, y, t)$, etc. Though this work only describes systems of up to two spatial dimensions, one can easily extend to three dimensions [32] and potentially higher-dimensional systems! See Section 2.1.1 for more information on dimensions.

If u is univariate, and only a function of time, the differential equation that describes the motion of this system is called an *ordinary differential equation* (ODE). Various ways to describe the second derivative in time of u, or the acceleration of u are

$$\begin{aligned} \frac{d^2u}{dt^2} &\quad \text{(Leibniz's notation)}, \\ \ddot{u} &\quad \text{(Newton's notation)}, \\ D_t^2 u &\quad \text{(Euler's notation)}. \end{aligned}$$

Leibniz's notation could be considered the most standard notation but is not necessarily compact. Newton's notation on the other hand allows for an ultra compact notation using a dot above the function to denote a time-derivative. In this work, for ODEs in isolation, Newton's notation will be used for this reason. The drawback of this notation is that it only be used for univariate functions. Finally, Euler's notation indicates a derivative using an operator which can be applied to a function.

If u is also a function of at least one spatial dimension, the equation of motion is a called a *partial differential equation* (PDE). The literature uses different types of notation for taking (continuous-time) partial derivatives. Applied to a state variable u these can look like

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &\quad \text{(Leibniz's notation)}, \\ u_{tt} &\quad \text{(subscript notation)}, \\ \partial_t^2 u &\quad \text{(Euler's notation)}, \end{aligned}$$

where the subscript notation could be seen as the partial derivative counterpart to Newton's notation due to its compactness. In the remainder of this document, Euler's notation will be used for PDEs, due to their similarity to operators in discrete time (introduced in Section 2.2.2) and as it allows for creation of bigger operators for more compactness when working with multiple (connected) systems (see e.g. Chapter **??**). Also, state-of-the-art literature in the field of FDTD methods sound synthesis use this notation [11].

2.1.1 Dimensions and Degrees of Freedom

All objects in the physical world are three-dimensional (3D) as they have a non-zero width, length and depth. Moreover, these objects can move in these

three dimensions and thus have three translational *degrees of freedom (DoF)* (the three rotational DoF are ignored here). To reduce the complexity of the model as well as computational complexity (computational cost), simplifications can be made to reduce both the dimensionality of the spatial distribution of a physical object as well as that of the translational DoF.

Generally, the spatial distribution of an object can be simplified if one (or more) of the dimensions are small relative to the wavelengths of interest. A guitar string, for instance, has much greater length than its width or depth and can therefore be reduced to a one-dimensional (1D) system. If a 3D description were to be kept, the relative displacement between two locations on one cross-section along the length of the string would be taken into account. One could imagine that this displacement will always be orders of magnitude smaller than the relative displacement of two points along the string length and is thus negligible. Similarly, the thickness of a drum membrane is much smaller than its length and width and can therefore be simplified to a two-dimensional (2D) system.

The translational DoF, on the other hand, describe now many "coordinates" a state variable includes. In much of the literature on FDTD methods in the field of musical acoustics, the state variable only has one coordinate. In most string models, for example, only the transverse displacement in one polarisation is considered (see Chapter 4) and the other polarisation as well as the longitudinal motion of the string (motion along the string length) is ignored. In other words, every point along the string can only move up and down, not side-to-side and not forward and back. Although this greatly simplifies the system at hand and reduces computational complexity, this is not what happens in reality, and non-linear effects such as phantom partials and pitch glides due to tension modulation are not present in the simplified model.

check whether this needs to be per point along a system

Work has been done on strings with dual (transverse) polarisation by Desvages [33] and Desvages and Bilbao [34] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are couples can be found in [3, 35]. In [24], Villeneuve and Leonard present a mass-spring network where the state of every individual mass has three translational DoF. Due to these additional DoF, these networks do capture the aforementioned effects, but greatly increase the computational complexity of the models.

Although the dimensionality reduction ignores some of the physical processes, surprisingly realistic sounding models can be made despite these simplifications. Due to computational considerations, all models used in this work thus only have 1 translational DoF.

Notation

When describing the state of a system, the spatial dimensions it is distributed over appears in the argument of the state variable. For example, the state of a 2D system, with 1 translational DoF is written as $u(x, y, t)$.

The translational DoF, on the other hand, determines the amount of coordinates that the state variable describes. A 1D system with 3 translational DoF can thus be written as $\mathbf{u}(x, t)$ where $\mathbf{u}$ is a vector containing the coordinates for all three translational DoF.

2.1.2 Ranges of Definition and Domains

When modelling physical systems, one needs to provide a *range of definition* over which they are defined. For a 1D system $u = u(x, t)$, ranges of definition must be given for x and t. Usually, the temporal range $t \geq 0$, meaning that the system is defined for non-negative time.

In space, the range of definition is usually referred to as a (spatial) *domain*, denoted by the symbol $\mathcal{D}$. Using the example above, x may be defined over $\mathcal{D}$, which is written as $x \in \mathcal{D}$. For analysis purposes, infinite domains ($\mathcal{D} = \mathbb{R} = (-\infty, \infty)$) or semi-infinite domains ($\mathcal{D} = \mathbb{R}^+ = [0, \infty)$) may be used, but for implementation purposes, a finite domain needs to be established. For higher dimensional systems, one needs to define higher dimensional domains. A 2D system $u = u(x, y, t)$, for simplicity assumed to be rectangular, may be defined over 'horizontal domain' $\mathcal{D}_x$ and 'vertical domain' $\mathcal{D}_y$, which are both 1D domains. The system is then defined for $(x, y) \in \mathcal{D}$ where $\mathcal{D} = \mathcal{D}_x \times \mathcal{D}_y$.

2.2 Discretisation using FDTD methods

Differential equations are powerful tools to describe the motion of physical systems. Despite this, only few of these have a closed-form, or analytical, solution. More complex systems require methods that do not perfectly solve, but rather *approximate* the solutions to these equations. FDTD methods are the most straightforward approach to numerically approximate differential equations. These methods are considered of the most general and flexible techniques in terms of the systems they can model, and frankly, relatively simple to understand once some familiarity with them is obtained. The main concern with these methods is the numerical stability of the eventual approximation. Conditions for stability can be mathematically derived and will be introduced in Section 3.3.

FDTD methods essentially subdivide a continuous differential equation into discrete points in time and space, a process called *discretisation*. Once an ODE or PDE is discretised using these methods it is now called a *finite-difference (FD) scheme* which approximates the original differential equation. In

the following, for generality and ease of explanation, a 1D system will be used. Unless denoted otherwise, the equations and theory used in this chapter has been taken from [3].

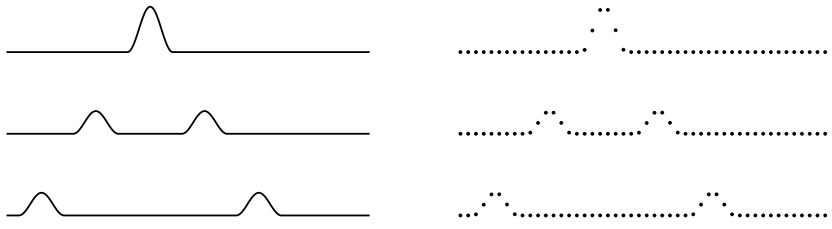


Fig. 2.1: A continuous PDE is discretised...

Figure and caption are not done yet

2.2.1 Grid Functions

FULL DOC SWEEP: check capitalisation of headings throughout document

The first step to approximate continuous PDEs, is to define a discrete *grid* over time and space. A system described by state $u = u(x, t)$ defined over time t and one spatial dimension x, can be discretised to a *grid function* u_l^n. Here, integers l and n describe the spatial and temporal indices respectively and arise from the discretisation of the continuous variables x and t according to $x = lh$ and $t = nk$. The spatial step h, also called the *grid spacing* describes the distance (in m) between two neighbouring *grid points*, and is closely related to the stability of the FD scheme. The temporal step k, or *time step* is the time (in s) between two consecutive temporal indices and can be calculated $k = 1/f_s$ for a sample rate f_s (in Hz). In many audio applications $f_s = 44100$ Hz which will be used in this work (unless denoted otherwise).

grid figure

As mentioned in Section 2.1.2, a 1D system needs to be defined over a temporal range of definition and one spatial domain. In discrete time, $t \geq 0$ is discretised to $n \in \mathbb{N}^0$.[1] The spatial domain $\mathcal{D}$ can be subdivided into N equal sections, or intervals, of length h (see Figure 2.2). The grid points describing the state of the system are placed at the edge of each interval, including the end points. The spatial range of interest then becomes $l \in \{0, \ldots, N\}$ and the total number of grid points is $N + 1$, which is one more than the number of intervals.

To summarise, for a 1D system

this might be unnecessary, but I thought that it might be nice to have this in an equation for clarity

$$u(x, t) \approxeq u_l^n \quad \text{with} \quad x = lh \quad \text{and} \quad t = nk,$$
$$l \in \{0, \ldots, N\} \quad \text{and} \quad n \in \mathbb{N}^0.$$

[1] In this work, $\mathbb{N}^0$ is used to denote the set of non-negative integers ($\mathbb{N}^0 = 0, 1, 2, \ldots$).

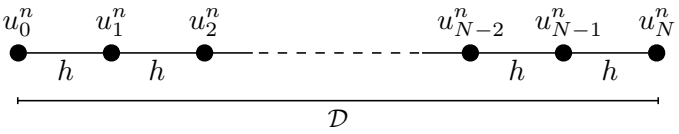


Fig. 2.2: When a 1D system $u(x,t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n, the spatial domain $\mathcal{D}$ is divided into N intervals of length h and spatial range of interest $l = \{0, \ldots, N\}$.

2.2.2 Finite-Difference Operators

Now that the state variable has a discrete counterpart, this leaves the derivatives to be discretised, or approximated. We start by introducing shift operators that can be applied to a grid function and 'shifts' its indexing, either temporally or spatially. Forward and backward shifts in time, together with the identity operation are

$$e_{t+}u_l^n = u_l^{n+1}, \quad e_{t-}u_l^n = u_l^{n-1}, \quad \text{and} \quad 1u_l^n = u_l^n. \tag{2.1}$$

Similarly, forward and backward shifts in space are

$$e_{x+}u_l^n = u_{l+1}^n, \quad \text{and} \quad e_{x-}u_l^n = u_{l-1}^n. \tag{2.2}$$

many figures for shift and FD operators

These shift operators are rarely used in isolation, though they do appear in energy analysis techniques detailed in Section 3.4. The operators do, however, form the basis of commonly used *finite-difference (FD) operators*. The first-order derivative in time can be discretised three different ways. The forward, backward and centred difference operators are

FULL DOC SWEEP: check centred instead of centered

these spacings are different in overleaf...

$$\partial_t \approx \begin{cases} \delta_{t+} \triangleq \frac{1}{k}\left(e_{t+} - 1\right), & \text{(2.3a)} \\ \delta_{t-} \triangleq \frac{1}{k}\left(1 - e_{t-}\right), & \text{(2.3b)} \\ \delta_{t\cdot} \triangleq \frac{1}{2k}\left(e_{t+} - e_{t-}\right), & \text{(2.3c)} \end{cases}$$

where "$\triangleq$" means "equal to by definition". These operators can then be applied to grid function u_l^n to get

$$\partial_t u \approx \begin{cases} \delta_{t+}u_l^n = \frac{1}{k}\left(u_l^{n+1} - u_l^n\right), & \text{(2.4a)} \\ \delta_{t-}u_l^n = \frac{1}{k}\left(u_l^n - u_l^{n-1}\right), & \text{(2.4b)} \\ \delta_{t\cdot}u_l^n = \frac{1}{2k}\left(u_l^{n+1} - u_l^{n-1}\right), & \text{(2.4c)} \end{cases}$$

and all approximate the first-order time derivative of u. Note that the centred difference has a division by $2k$ as the time difference between $n+1$ and $n-1$ is, indeed, twice the time step.

figure here visualising operators (with reference to grid figure)

Similar operators exist for a first-order derivative in space, where the for-

ward, backward and centred difference are

$$\partial_x \approx \begin{cases} \delta_{x+} \triangleq \frac{1}{h}\left(e_{x+} - 1\right), & \text{(2.5a)} \\ \delta_{x-} \triangleq \frac{1}{h}\left(1 - e_{x-}\right), & \text{(2.5b)} \\ \delta_{x\cdot} \triangleq \frac{1}{2h}\left(e_{x+} - e_{x-}\right), & \text{(2.5c)} \end{cases}$$

and when applied to u_l^n are

$$\partial_x u \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h}\left(u_{l+1}^n - u_l^n\right), & \text{(2.6a)} \\ \delta_{x-} u_l^n = \frac{1}{h}\left(u_l^n - u_{l-1}^n\right), & \text{(2.6b)} \\ \delta_{x\cdot} u_l^n = \frac{1}{2h}\left(u_{l+1}^n - u_{l-1}^n\right). & \text{(2.6c)} \end{cases}$$

Higher order differences can be approximated through a composition of first-order difference operators where their definitions are multiplied. The second-order difference in time may be approximated using

$$\partial_t^2 \approx \delta_{t+}\delta_{t-} = \delta_{tt} \triangleq \frac{1}{k^2}\left(e_{t+} - 2 + e_{t-}\right), \tag{2.7}$$

where "2" is the identity operator applied twice. This can similarly be done for the second-order difference in space

$$\partial_x^2 \approx \delta_{x+}\delta_{x-} = \delta_{xx} \triangleq \frac{1}{h^2}\left(e_{x+} - 2 + e_{x-}\right), \tag{2.8}$$

both of which can be applied to a grid function u_l^n in a similar fashion. Figure 2.3 shows the *stencils* of the operators introduced above. A stencil shows the grid points needed to perform the operation of a FD operator.

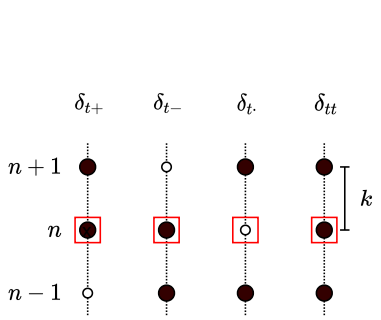
Also useful are averaging operators, all of which approximate the identity operation. The temporal forward, backward and centred averaging operators are

in energy analysis, interleaved grids, etc.

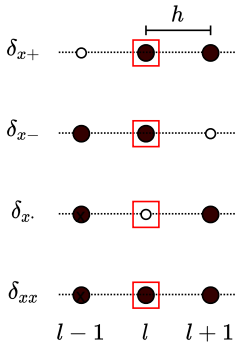
$$1 \approx \begin{cases} \mu_{t+} \triangleq \frac{1}{2}\left(e_{t+} + 1\right), & \text{(2.9a)} \\ \mu_{t-} \triangleq \frac{1}{2}\left(1 + e_{t-}\right), & \text{(2.9b)} \\ \mu_{t\cdot} \triangleq \frac{1}{2}\left(e_{t+} + e_{t-}\right). & \text{(2.9c)} \end{cases}$$

Notice how these definitions are different than the difference operators in (2.3): the terms in the parentheses are added rather than subtracted, and rather than a division by the time step k there is a division by 2. Finally, the centred averaging operator does not have an extra division by 2 as in (2.3c). Applied to u_l^n, Eqs. (2.9) become

$$u_l^n \approx \begin{cases} \mu_{t+} u_l^n = \frac{1}{2}\left(u_l^{n+1} + u_l^n\right), & \text{(2.10a)} \\ \mu_{t-} u_l^n = \frac{1}{2}\left(u_l^n + u_l^{n-1}\right), & \text{(2.10b)} \\ \mu_{t\cdot} u_l^n = \frac{1}{2}\left(u_l^{n+1} + u_l^{n-1}\right). & \text{(2.10c)} \end{cases}$$



(a) Temporal operators.



(b) Spatial operators.

Fig. 2.3: The stencils of various FD operators applied to the grid point highlighted with a red square. Black grid points are used in the calculation, and white grid points are not. The averaging operators follow the same pattern.

Similarly, spatial averaging operators are

$$1 \approx \begin{cases} \mu_{x+} \triangleq \frac{1}{2}\left(e_{x+} + 1\right), & \text{(2.11a)} \\ \mu_{x-} \triangleq \frac{1}{2}\left(1 + e_{x-}\right), & \text{(2.11b)} \\ \mu_{x\cdot} \triangleq \frac{1}{2}\left(e_{x+} + e_{x-}\right), & \text{(2.11c)} \end{cases}$$

and when applied to u_l^n

$$u_l^n \approx \begin{cases} \mu_{x+} u_l^n = \frac{1}{2}\left(u_{l+1}^n + u_l^n\right), & \text{(2.12a)} \\ \mu_{x-} u_l^n = \frac{1}{2}\left(u_l^n + u_{l-1}^n\right), & \text{(2.12b)} \\ \mu_{x\cdot} u_l^n = \frac{1}{2}\left(u_{l+1}^n + u_{l-1}^n\right). & \text{(2.12c)} \end{cases}$$

Finally, using forward and backward averaging operators, second-order temporal and spatial averaging operators can be created according to

$$1 \approx \mu_{tt} = \mu_{t+}\mu_{t-} \triangleq \frac{1}{4}\left(e_{t+} + 2 + e_{t-}\right), \tag{2.13}$$

and

$$1 \approx \mu_{xx} = \mu_{x+}\mu_{x-} \triangleq \frac{1}{4}\left(e_{x+} + 2 + e_{x-}\right). \tag{2.14}$$

Operators and derivatives in 2D will be discussed in Chapter 6.

Accuracy

As FDTD methods approximate continuous systems, the resulting solution is rarely 100% accurate. To determine the accuracy of the FD operators above,

one can perform a *Taylor series analysis*. The Taylor series is an infinite sum and its expansion of a function f about a point a is defined as

$$f(x) = \sum_{n=0}^{\infty} \frac{(x-a)^n}{n!} f^{(n)}(a) \tag{2.15}$$

where superscript (n) denotes the n^{th} derivative of f with respect to x. The analysis will be performed on the temporal operators in this section, but also apply to the spatial operators presented.

Using continuous function $u = u(t)$ and following Bilbao's "slight abuse of notation" in [3], one may apply FD operators to continuous functions according to

$$\delta_{t+} u(t) = \frac{u(t+k) - u(t)}{k} \,. \tag{2.16}$$

Assuming that u is infinitely differentiable, $u(t+k)$, i.e., u at the next time step (in continuous time), can be approximated using a Taylor series expansion of u about t according to

$$u(t+k) = u(t) + k\dot{u} + \frac{k^2}{2}\ddot{u} + \frac{k^3}{6}\dddot{u} + \mathcal{O}(k^4). \tag{2.17}$$

Here, (following Newton's notation introduced in Section 2.1) the dot describes a single temporal derivative and $\mathcal{O}$ includes additional terms in the expansion. The power of k in the argument of $\mathcal{O}$ describes the order of accuracy, the higher the power of k the more accurate the approximation. Equation (2.17) can be rewritten to

$$\frac{u(t+k) - u(t)}{k} = \dot{u} + \frac{k}{2}\ddot{u} + \frac{k^2}{6}\dddot{u} + \mathcal{O}(k^3),$$

and using Eq. (2.16) can be written to

$$\delta_{t+} u(t) = \dot{u} + \mathcal{O}(k). \tag{2.18}$$

This says that the forward difference operator approximates the continuous first order derivative with an additional error term that depends on k. As the power of k in $\mathcal{O}$'s argument is 1, the forward operator is first-order accurate. One can also observe that, as expected, the error gets smaller as the time step k gets smaller and indicates that higher sample rates result in more accurate simulations (through $k = 1/f_s$). confirming our intuition

One can arrive at a similar result for the backward operator. Applying Eq. (2.3b) to $u(t)$ yields

$$\delta_{t-} u(t) = \frac{u(t) - u(t-k)}{k}. \tag{2.19}$$

One can then approximate $u(t-k)$ by performing a Taylor series expansion of u about t according to

$$u(t-k) = u(t) + (-k)\dot{u} + \frac{(-k)^2}{2}\ddot{u} + \frac{(-k)^3}{6}\dddot{u} + \mathcal{O}(k^4), \tag{2.20}$$

$$\frac{u(t-k)-u(t)}{k} = -\dot{u} + \frac{k}{2}\ddot{u} - \frac{k^2}{6}\dddot{u} + \mathcal{O}(k^3),$$

$$\delta_{t-}u(t) = \dot{u} + \mathcal{O}(k). \tag{2.21}$$

Notice that the sign of $\mathcal{O}$ does not matter.

Applying the centred operator in Eq. (2.3c) to $u(t)$ yields

$$\delta_{t\cdot}u(t) = \frac{u(t+k)-u(t-k)}{2k}, \tag{2.22}$$

indicating that to find the order of accuracy for this operator, both Eqs. (2.17) and (2.20) are needed. Subtracting these and substituting their definitions yields

$$u(t+k) - u(t-k) = 2k\dot{u} - \frac{2k^3}{6}\dddot{u} + 2\mathcal{O}(k^5),$$

$$\frac{u(t+k)-u(t-k)}{2k} = \dot{u} + \mathcal{O}(k^2),$$

$$\delta_{t\cdot}u(t) = \dot{u} + \mathcal{O}(k^2), \tag{2.23}$$

and shows that the centred difference operator is second-order accurate.

As a first-order derivative indicates the *slope* of a function, the differences in accuracy between the above operators can be visualised as in Figure 2.4. It can be observed that the derivative approximation – the slope – of the centred operator much more closely matches the the true derivative of u at t.

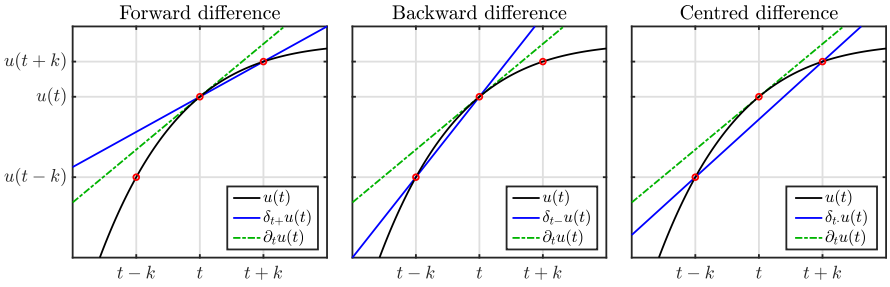


Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. One can observe that the centred difference operator much more closely approximates the derivative, or the slope, of u at t than the forward and backward difference operators.

Higher-order differences, such as the second-order difference in time operator in Eq. (2.7) can also be applied to $u(t)$ to get

$$\delta_{tt} u(t) = \frac{u(t+k) - 2u(t) + u(t-k)}{k^2}, \tag{2.24}$$

and can be proven to be second-order accurate by adding Eqs. (2.17) and (2.20):

$$\begin{aligned} u(t+k) + u(t-k) &= 2u(t) + k^2 \ddot{u} + \mathcal{O}(k^4), \\ \frac{u(t+k) - 2u(t) + u(t-k)}{k^2} &= \ddot{u} + \mathcal{O}(k^2), \\ \delta_{tt} u(t) &= \ddot{u} + \mathcal{O}(k^2). \end{aligned} \tag{2.25}$$

The accuracy of averaging operators can be found in the same way and follow a similar pattern.

$$\begin{aligned} \mu_{t+} u(t) &= u(t) + \mathcal{O}(k), \quad \mu_{t-} u(t) = u(t) + \mathcal{O}(k), \\ \mu_{t\cdot} u(t) &= u(t) + \mathcal{O}(k), \quad \mu_{tt} u(t) = u(t) + \mathcal{O}(k^2). \end{aligned} \tag{2.26}$$

2.2.3 Identities

For working with FD schemes, either for implementation or analysis, it can be extremely useful to rewrite the operators presented above to equivalent versions of themselves. These are called *identities* and for future reference, some useful ones are listed below:

$$\delta_{tt} = \frac{2}{k} \left(\delta_{t\cdot} - \delta_{t-} \right), \tag{2.27a}$$

$$\delta_{t\cdot} = \delta_{t+} \mu_{t-} = \delta_{t-} \mu_{t+}, \tag{2.27b}$$

$$\mu_{t+} = \frac{k}{2} \delta_{t+} + 1. \tag{2.27c}$$

That these equalities hold can easily be proven by expanding the operators defined in Section 2.2.2. Naturally, these identities also hold for spatial operators by simply substituting the 't' subscripts for 'x'.

see whether the negative version of identity (2.27c) is also used later on

2.3 The Mass-Spring System

Though a complete physical modelling field on their own (see Chapter 1), mass-spring systems are also sound-generating systems and lend themselves well to illustrating and explaining FDTD methods in practice. Starting with the continuous-time ODE, this section continues to discretise it to an FD scheme using the operators described in Section 2.2.2. Finally, the scheme is rewritten to an update equation that can be implemented and the output of the system is shown.

2.3.1 Continuous-time

Using dots to indicate a temporal derivative, the ODE of a simple mass-spring system is defined as

$$M\ddot{u} = -Ku, \tag{2.28}$$

where $u = u(t)$ is the distance from the equilibrium position (in m), $M > 0$ is the mass of the mass (in kg) and $K \geq 0$ is the spring constant (in N/m). Equation (2.28) can be written as

$$\ddot{u} = -\omega_0^2 u, \tag{2.29}$$

with angular frequency (in rad/s)

$$\omega_0 = \sqrt{K/M}. \tag{2.30}$$

This way of writing the mass-spring ODE is more compact and can more directly be related to the fundamental frequency $f_0 = \omega_0/2\pi$ (in Hz) of the system.

Apart from the choices of K and M, the behaviour of the mass-spring system is determined by its *initial conditions*, being $u(0)$ and $\partial_t u(0)$, i.e., the displacement and velocity of the mass at $t = 0$. If the initial conditions are non-zero, the path that the displacement of the mass follows over time is sinusoidal (see Figure 2.5), which is also why the mass-spring system is often referred to as the *simple harmonic oscillator*. The amplitude of the sinusoid is determined by the initial conditions, whereas the frequency is determined by M and K.

Intuition

move section up (stefan's comment)

The behaviour of the mass-spring system in Eq. (2.28) arises from two basic laws of physics: *Newton's second law* and *Hooke's law*.

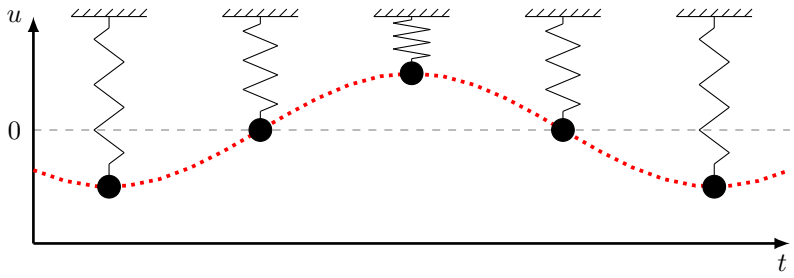


Fig. 2.5: Mass spring system over time. The system follows a harmonic (sinusoidal) motion.

Starting with Newton's second law – *force equals mass times acceleration* – and relating this to the variables used in Eq. (2.28) yields an expression for force

$$F = M\ddot{u}. \tag{2.31}$$

This equation in isolation can be used to, for example, calculate the force necessary to accelerate a mass of M kg to $\ddot{u}$ m/s^2. Next, the force generated by the spring follows Hooke's law:

$$F = -Ku, \tag{2.32}$$

which simply states that the force generated by a spring with stiffness K is negatively proportional to the value of u. In other words, the further the spring is extended (from the equilibrium $u = 0$), the more force will be generated in the opposite direction. Finally, as the sole force acting on the mass is the one generated by the spring, the two expressions for the force F can be set equal to each other and yields the equation for the mass-spring system in (2.28).

The sinusoidal behaviour of the mass-spring system, or a least the fact that the mass "gets pulled back" to the equilibrium, is apparent from the minus-sign in Eq. (2.32). The frequency of the sinusoid, depends on the value of K as the "pull" happens to a higher degree for a higher spring stiffness. That the frequency of the system is also dependent on the mass M can be explained by the fact that a lighter object is more easily moved and vice versa, which is apparent from Eq. (2.31). In other words, the pull of the spring has a greater effect on the acceleration of a lighter object than a heavier one.

Finally, if $u = 0$ there is no spring force present and the acceleration remains unchanged. This is exactly what Newton's first law states: if the net force acting on an object is zero, its velocity will be constant. If the mass is not in motion, this means that it remains stationary. If it is, at the exact moment that $u = 0$, the velocity is unchanged.

2.3.2 Discrete-time

Following the discretisation process introduced in Section 2.2, one can approximate the PDE in Eq. (2.28). The displacement of the mass is approximated using

$$u(t) \approx u^n, \tag{2.33}$$

with time $t = nk$, time step $k = 1/f_\mathrm{s}$, sample rate f_s and temporal index and $n \in \mathbb{N}^0$. Note that the "grid function" does not have a subscript l as u is not distributed in space and is now simply called a *time series*.

Using the operators found in Section 2.2.2, Eq. (2.28) can be discretised as follows:

$$M\delta_{tt}u^n = -Ku^n, \tag{2.34}$$

which is the first appearance of a FD scheme in this work. Expanding the δ_{tt} operator yields

$$\frac{M}{k^2}\left(u^{n+1} - 2u^n + u^{n-1}\right) = -Ku^n,$$

and solving for u^{n+1} results in the following recursion or *update equation*:

$$u^{n+1} = \left(2 - \frac{Kk^2}{M}\right)u^n - u^{n-1}, \tag{2.35}$$

which can be implemented in a programming language such as `MATLAB`.

2.3.3 Implementation and Output

A simple `MATLAB` script implementing the mass-spring system described in this section is shown in Appendix C.1. The most important part of the algorithm happens in a for-loop recursion, where update equation (2.35) is implemented. At the end of each loop, the system states are updated and prepared for the next iteration.

To be able to start the simulation of the scheme, the initial conditions given in Section 2.3.1 must be discretised at $n = 0$. As n is only defined for values greater than zero, the forward difference operator is used. A simple way to obtain a sinusoidal motion with an amplitude of 1, is to set the initial conditions as follows:

$$u^0 = 1 \quad \text{and} \quad \delta_{t+}u^0 = 0. \tag{2.36}$$

The latter equality can be solved for u^1 to obtain its definition:

$$\begin{aligned} &\frac{1}{k}\left(u^1 - u^0\right) = 0, \\ \overset{u^0=1}{\Longleftrightarrow} \quad &u^1 - 1 = 0, \\ &u^1 = 1. \end{aligned}$$

In short, setting $u^0 = u^1 \neq 0$ yields an oscillatory behaviour with an amplitude of 1. Note that any other non-zero initial condition will also yield oscillatory behaviour, but likely with a different amplitude.

The values for K and M are restricted by a stability condition

$$k < 2\sqrt{\frac{M}{K}}, \tag{2.37}$$

which will be elaborated on in Section 3.3. If this condition is not satisfied, the system will exhibit (exponential) growth and is *unstable*.

The output of the system can be obtained by 'recording' the displacement of the mass and listening to this at the given sample rate f_s. An example of this can be found in Figure 2.6 where the frequency of oscillation $f_0 = 440$ Hz.

2.4 The 1D Wave Equation

Arguably the most important PDE in the field of physical modelling for sound synthesis is the 1D wave equation. It can be used to describe transverse vibration in an ideal string, longitudinal vibration in an ideal bar or the pressure in an acoustic tube (see Chapter 5). Although the behaviour of this equation alone does not appear in the real world as such – as no physical system is ideal – it is extremely useful as a test case and a basis for more complicated models.

2.4.1 Continuous time

FULL DOC SWEEP: check hyphen in titles

The 1D wave equation is a PDE that describes the motion of a system distributed in one dimension of space. Consider the state of a 1D system $u = u(x,t)$ of length L (in m) defined for time $t \geq 0$ and $x \in \mathcal{D}$ with $\mathcal{D} = [0, L]$.

unit?

The PDE describing its motion is

$$\partial_t^2 u = c^2 \partial_x^2 u, \tag{2.38}$$

where c is the wave speed of the system (in m/s). Figure 2.7 shows the wave propagation of the 1D wave equation excited using a raised cosine

Intuition

As with the mass-spring system in Section 2.3 the working of the PDE in (2.38) arises from Newton's second law, even though this connection might be less apparent.

The 1D wave equation in (2.38) states that the acceleration of $u(x,t)$ at location x is determined by the second-order spatial derivative of u at that same location (scaled by a constant c^2). In the case that u describes the transverse displacement of an ideal string, this second-order derivative denotes the *curvature* of this string. As c^2 is always positive, the sign (or direction) of the acceleration is fully determined by the sign of the curvature. In other words,

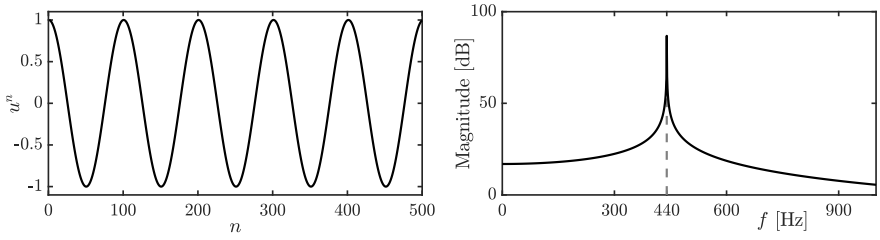
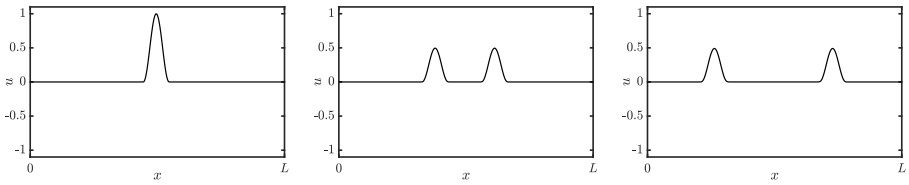


Fig. 2.6: The time-domain and frequency-domain output of a mass-spring system with $f_0 = 440$ Hz.



(a) $t = 0$ ms. **(b)** $t = 1$ ms. **(c)** $t = 2$ ms.

Fig. 2.7: Wave propagation in the 1D wave equation in Eq. (2.38) wtih $c \approx 127$ m/s.

a 'positive' curvature at location x along the ideal string yields a 'positive' or upwards acceleration at that same location.

What a 'positive' or 'negative' curvature implies is more easily seen when we take a simple function describing a parabola, $y(x) = x^2$, and take its second derivative to get $y''(x) = 2$. The answer is a positive number which means that y has a positive curvature.

So, what does this mean for the 1D wave equation? As a positive curvature implies a positive or upwards acceleration as per Eq. (2.38), u with a positive curvature at a location x will start to move upwards and vice versa. Of course, the state of a physical system such as u will rarely have a perfect parabolic shape, but the argument still applies. See Figure 2.8 for a visualisation of the forces acting on u due to curvature.

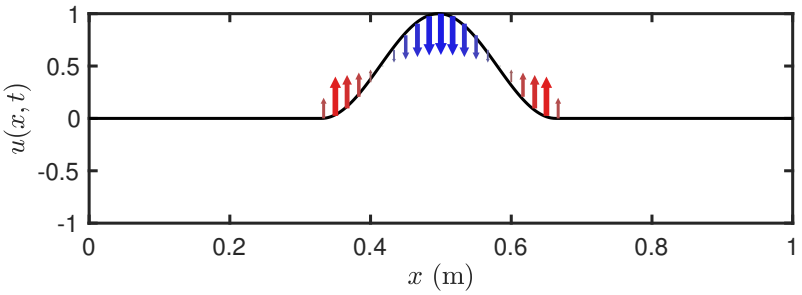
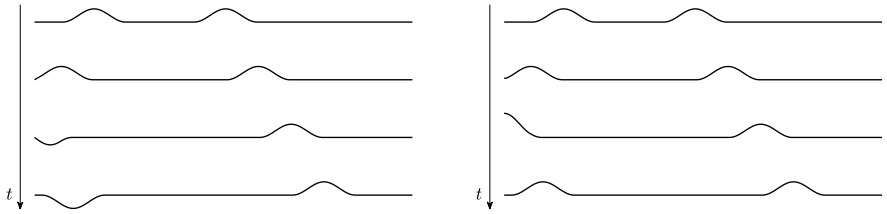


Fig. 2.8: The forces acting on the 1D wave equation described by $u(x, t)$ due to curvature. The arrows indicate the direction and magnitude of the force, and simultaneously the acceleration as these are connected through Eq. (2.38).

different wording in caption

Boundary Conditions

When a system is distributed in space, *boundary conditions* must be determined. Recalling that x is defined over domain $\mathcal{D} = [0, L]$, the boundaries, or end



(a) The Dirichlet boundary condition in Eq. (2.39a) fixes the boundary, which causes the incoming waves to invert.

(b) The Neumann or free boundary condition in Eq. (2.39b) fixes the slope at the boundary, causing the incoming waves to not invert.

Fig. 2.9: The behaviour of the 1D wave equation with (a) Dirichlet or (b) Neumann boundary conditions.

points of the system are located at $x = 0$ and $x = L$. Two often-used alternatives for the boundary conditions are

$$u(0,t) = u(L,t) = 0 \quad \text{(Dirichlet, fixed)}, \tag{2.39a}$$

$$\partial_x u(0,t) = \partial_x u(L,t) = 0 \quad \text{(Neumann, free)}. \tag{2.39b}$$

The Dirichlet boundary condition says that at the end points of the system, the state is 0 at all times. The Neumann condition on the other hand, says that rather the slope of these points needs to be 0, but that the end points are free to move transversely. In the former case, incoming waves to invert after reaching the boundary whereas in the latter incoming waves are reflected un-inverted. See Figure 2.9.

add why this is relevant?

If both boundaries of the 1D wave equation share the same condition, the fundamental frequency of the simulation can be calculated using

$$f_0 = \frac{c}{2L}\,. \tag{2.40}$$

Scaling

As this work follows much of Bilbao's *Numerical Sound Synthesis* [3], it might be good to talk about a major discrepancy between the PDEs and FD schemes that appear there and those used here. Non-dimensionalisation, or *scaling*, is extensively used in [3] and much of the literature published around that time (fx. [36, 35]) and can be useful to reduce the amount of parameters used to describe a system.

Scaling techniques normalise the domain $x \in [0, L]$ to $x' \in [0, 1]$ with $x' = x/L$. The 1D wave equation in (2.38) can then be rewritten to

$$\partial_t^2 u = \gamma^2 \partial_{x'x'} u, \tag{2.41}$$

where scaled wave speed $\gamma = c/L$ has units of frequency. The scaling has removed the necessity for both c and L and simply specifying the scaled wave speed γ is enough to parameterise the behaviour of the system. The parameter reduction gets more apparent for more complex systems and could greatly simplify of the models used, at least in notation and parameter control.

Although this parameter reduction might be useful for resonators in isolation, when multiple resonators interact with each other (see Part **??**), it is better to keep the systems dimensional. As a big part of this work includes interaction between multiple resonators, only dimensional systems will appear here.

check whether still correct

2.4.2 Discrete time

FULL DOC SWEEP: check straightforward or straightforward

Coming back to the PDE presented in Eq. (2.38), we continue by finding a discrete-time approximation for it. The most straightforward discretisation of Eq. (2.38) is the following FD scheme

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n, \tag{2.42}$$

with $l \in \{0, \ldots, N\}$ and number of grid points $N+1$. Other schemes exist (see e.g. [3]), but are excluded as they have not been used in this work. Expanding the operators using the definitions given in Section 2.2.2 yields

$$\frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) = \frac{c^2}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right). \tag{2.43}$$

and solving for u_l^{n+1} yields

$$u_l^{n+1} = \left(2 - 2\lambda^2\right) u_l^n + \lambda^2 \left(u_{l+1}^n + u_{l-1}^n\right) - u_l^{n-1}. \tag{2.44}$$

Here,

$$\lambda = \frac{ck}{h} \tag{2.45}$$

is called the *Courant number* and plays a big role in stability and quality of the FD scheme. More specifically, λ needs to abide the (famous) Courant-Friedrichs-Lewy or *CFL condition* for short [37]

$$\lambda \leq 1, \tag{2.46}$$

which acts as a stability condition for scheme (2.42). More details on this are given in Section 2.4.4.

As c, k and h are interdependent due to the CFL condition, it is useful to rewrite Eq. (2.46) in terms of known variables. As the time step k is based on the sample rate and thus (usually) fixed, and c is a user-defined wave speed, the CFL condition can be rewritten in terms of the grid spacing h:

$$h \geq ck, \tag{2.47}$$

which, in implementation, is used as a stability condition for the scheme. See Section 3.3 for more information on how to derive a stability condition from a FD scheme.

Stencil

As was done for several FD operators in Figure 2.3, it can be useful to visualise the *stencil*, or region of operation, of a FD scheme. A stencil of a scheme visualises what grid values are necessary to calculate the state at the next time step u_l^{n+1}. Figure 2.10 shows the stencil for scheme (2.42) and – in essence – visualises the various shifts of the grid function in (2.44).

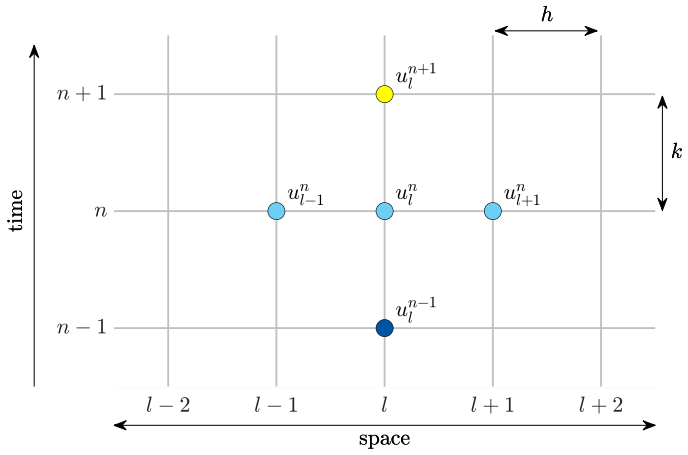


Fig. 2.10: The stencil, or region of operation, for the FD scheme in (2.42).

Boundary Conditions and Virtual Grid Points

The end points of the discrete domain are located at $l = 0$ and $l = N$. Substituting these locations into Eq. (2.44) seemingly shows that grid points outside of the defined domain are needed, namely u_{-1}^n and u_{N+1}^n. These can be referred to as *virtual grid points* and can be accounted for by discretising the boundary conditions in Eq. (2.39). Discretising these (using the most accurate centred spatial difference operator for the Neumann condition) yields

$$u_0^n = u_N^n = 0, \quad \text{(Dirichlet)} \tag{2.48a}$$

$$\delta_x . u_0^n = \delta_x . u_N^n = 0. \quad \text{(Neumann)} \tag{2.48b}$$

If Dirichlet boundary conditions are used, the states of the boundary points will always be zero and can therefore can be excluded from the calculations.

The range of calculation then simply becomes $l \in \{1, \ldots, N-1\}$ and no virtual grid points are needed when performing the update.

If, on the other hand, Neumann conditions are used, the range of calculation remains $l \in \{0, \ldots, N\}$ and definitions for the virtual grid points need to be found. Expanding the operators in Eq. (2.48b) and solving for u_{-1}^n and u_{N+1}^n provides the definitions for these virtual grid points based on values inside the defined domain:

$$\begin{aligned} &\frac{1}{2h}\left(u_1^n - u_{-1}^n\right) = 0, \\ &u_1^n - u_{-1}^n = 0, \\ &u_{-1}^n = u_1^n. \end{aligned} \qquad \begin{aligned} &\frac{1}{2h}\left(u_{N+1}^n - u_{N-1}^n\right) = 0, \\ &u_{N+1}^n - u_{N-1}^n = 0, \\ &u_{N+1}^n = u_{N-1}^n. \end{aligned}$$

At the boundaries, the update equation in (2.44) will then have the the above definitions for the virtual grid points substituted and will become

$$u_0^{n+1} = \left(2 - 2\lambda^2\right) u_0^n + 2\lambda^2 u_1^n - u_0^{n-1}, \tag{2.49}$$

and

$$u_N^{n+1} = \left(2 - 2\lambda^2\right) u_N^n + 2\lambda^2 u_{N-1}^n - u_N^{n-1}, \tag{2.50}$$

at the left and right boundary respectively.

2.4.3 Implementation: Excitation and Output

See Appendix C.2 for a `MATLAB` implementation of the 1D wave equation.

A simple way to excite the system is to initialise the state using a raised cosine, or Hann window. More information on excitations will be given in Part **??**, but for completeness, the formula for a discrete raised cosine will be given here.

The discrete raised cosine can be parametrised by its center location l_c and width w from which the start index l_s and end index l_e can be calculated, according to

$$l_\mathrm{s} = l_\mathrm{c} - \lfloor w/2 \rfloor \quad \text{and} \quad l_\mathrm{e} = l_\mathrm{c} + \lfloor w/2 \rfloor, \tag{2.51}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation and needs to be used as all the above variables are integers. Furthermore, both l_s and l_e must fall into the defined spatial range of calculation. Then, a raised cosine with an amplitude of 1 can be calculated and used as an initial condition for the system according to

$$u_l^1 = u_l^0 = \begin{cases} 0.5 - 0.5\cos\left(\frac{2\pi(l - l_\mathrm{s})}{w-1}\right), & l_\mathrm{s} \leq l < l_\mathrm{e}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.52}$$

As done for the implementation of the mass-spring system in Section 2.3.3, both u_l^0 and u_l^1 are initialised with the same state, as to only have an initial displacement, and not an initial velocity.

In `MATLAB`, an easier way to obtain a raised cosine is to use the `hann(w)` function which returns a raised cosine (or Hann window) of width `w`.

Output and Modes

After the system is excited, one can retrieve the output of the system by selecting a grid point l_{out} and listening to that at at the given sample rate f_{s}. An example using the parameters in Table 2.1 and Dirichlet boundary conditions is shown in Figure 2.11.

Name	Symbol (unit)	Value
User-defined parameters		
Length	L (m)	1
Wave speed	c (m/s)	1470
Sample rate	f_{s} (Hz)	44100
Derived parameters		
Fundamental frequency	f_0 (Hz)	735
No. of intervals	N (-)	30
Time step	k (s)	$\approx 2.27 \cdot 10^{-5}$
Grid spacing	h (m)	≈ 0.033
Courant number	λ (-)	1
Excitation and output		
Center location	l_{c} (-)	$0.2N$
Width	w (-)	4
Output location	l_{out}	3

Table 2.1: Parameters used for 1D wave equation example used in this section. The user-defined parameters have been chosen such that $\lambda = 1$.

As can be seen from Figure 2.11, the output of the 1D wave equation contains many peaks in the frequency spectrum on top of the fundamental frequency. These are called *harmonic partials* or *harmonics* for short and arise from the various modes of vibration present in the system (see Figure 2.12). Although the PDE has not been discretised using modal synthesis (another physical modelling technique discussed in Chapter 1), the system can still be decomposed into different modes of vibration, each corresponding to a harmonic frequency. These modes are assumed to vibrate independently, and their weighted sum yields the eventual behaviour of the system.

The amount of modes present in the continuous PDE of the 1D wave equation is theoretically infinite. The amount present in the discrete FD scheme, however, is determined by the number of moving points in the system. If Dirichlet boundary conditions are used, this means that there are $N-1$ modes, and $N+1$ modes for Neumann boundary conditions. If the CFL condition is

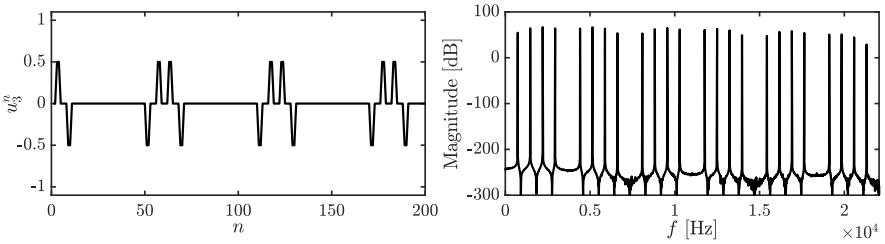


Fig. 2.11: The time-domain and frequency-domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_s = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_c = 0.2N$) and $w = 4$ and the output is retrieved at $l_{out} = 3$.

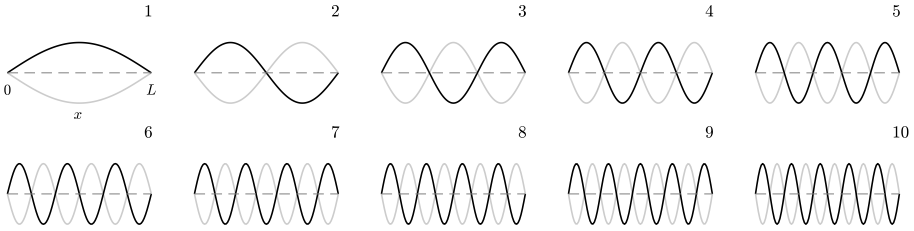
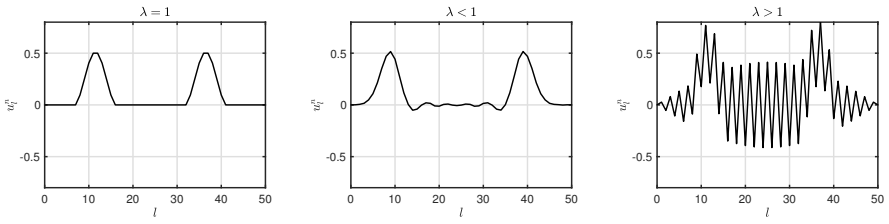


Fig. 2.12: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions defined for $x \in [0, L]$ (only shown for mode 1). The modes are normalised to have the same amplitude and vibrate at their respective modal frequencies with the extremes indicated by the black and the grey plot. The number of the shape can be determined by the amount of antinodes present in the shape.

satisfied with equality, the frequencies of these modes are integer multiples of the fundamental: $f_m = mf_0$ for mode number $m \in \{1, \ldots, N-1\}$ for Dirichlet and $m \in \{0, \ldots, N\}$ for Neumann boundary conditions. The frequency of the harmonics – and even the modal shapes – can be analytically derived using modal analysis as will be explained in Section 3.5.

The amplitude of the different modes depends on the excitation location (and type) and the output location. Figure 2.11, for example, seemingly shows that the system only exhibits 24 modes, rather than the 29 ($N - 1$) predicted. As the system is excited at $0.2N$, or in other words, $1/5^{\text{th}}$ of the length of the system, this means that that every 5^{th} mode will be attenuated. To understand how and/or why this happens, one can refer to Figure 2.12 and see that every 5^{th} modal shape has a node at $1/5^{\text{th}}$ its length. If the system is excited exactly there, this modal shape will not obtain any energy and will thus not resonate. Similarly, if the system is excited exactly in the middle, every 2^{nd} modal frequency will be attenuated as there is a node present in the corresponding modal shape. The output would then only contain odd-numbered modes.



(a) $\lambda = 1$ **(b)** $\lambda = 0.9$ **(c)** $\lambda = 1.001$

Fig. 2.13: Grid function u_l^n visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (2.46) is not satisfied and the system is unstable.

2.4.4 Stability and Simulation Quality

As shown in Eq. (2.46), the Courant number needs to abide the CFL condition in order for the scheme to be stable. A system is regarded *unstable* if it exhibits (exponential) unbounded growth. If Neumann boundary conditions (free) are used, it is possible that the system drifts off over time. This does not mean that the system is unstable, and is actually entirely physically possible![2]

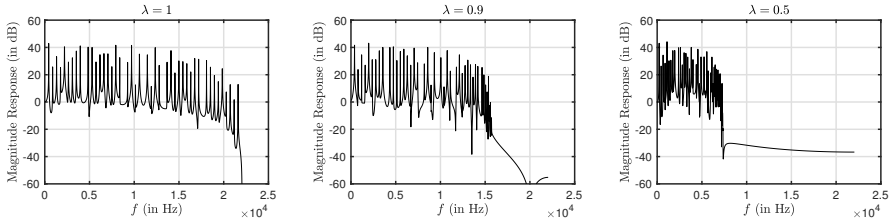
Besides stability, the value of λ is closely related to the quality of the simulation. If $\lambda = 1$, Eq. (2.42) is actually an exact solution to Eq. (2.38), which is quite uncommon in the realm of differential equations! See Figure 2.13a. Identically, if Eq. (2.47) is satisfied with equality, the FD scheme is an exact solution to the PDE, and if h deviates from this condition, the quality of the simulation decreases.

If $\lambda < 1$, the quality of the simulation quality decreases in an effect called *numerical dispersion*. Dispersion is a phenomenon where some frequencies travel faster through a medium than others, which is desired in some models (see fx. Chapter 4). Numerical dispersion, however, which is due to numerical inaccuracy, never is! Figure 2.13b shows an example when $\lambda = 0.9$, and one can observe that the wave propagation does not match the ideal case as Figure 2.13a shows. Moreover, bandlimiting effects occur, meaning that the highest frequency that the system can generate decreases. See Figure 2.14. Higher modes get 'squished' together and are not exact multiples of the fundamental anymore. Section 3.5 elaborates on how to calculate the exact modal frequencies of a FD implementation of the 1D wave equation.

Finally, if $\lambda > 1$ the system becomes unstable. An example is shown in Figure 2.13c. Unstable behaviour usually comes in the form of high frequencies (around the Nyquist frequency of $f_\mathrm{s}/2$) growing without bounds.

So in what situation would the stability condition not be satisfied with

[2]Imagine a 'free' guitar string where the ends are not connected to the nut and bridge of a guitar. The string can be taken far away from the guitar without it breaking or exploding.



(a) $\lambda = 1$ **(b)** $\lambda = 0.9$ **(c)** $\lambda = 1.001$

Fig. 2.14: Frequency spectra of the simulation output. The Courant number is set to (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$. One can observe that for lower values of λ the bandwidth of the output decreases drastically.

equality? As mentioned in Section 2.2.1, a continuous domain $\mathcal{D} = [0, L]$ for a system of length L needs to be divided into N equal sections of length h in the discretisation process. A logical step to calculate N would be to divide L by h calculated using Eq. (2.47) satisfied with equality to get the highest possible simulation quality. However, this calculation might not result in an integer value, which N should be! To stay as close to the stability condition as possible, the following calculations are performed in order:

$$h := ck, \quad N := \left\lfloor \frac{L}{h} \right\rfloor, \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}. \tag{2.53}$$

In other words, Eq. (2.47) is satisfied with equality and used to calculate integer N. After this, h is recalculated based on N and used to calculate the Courant number using Eq. (2.45). This process assures that N is an integer and that the CFL condition is satisfied, though not necessarily with equality.

To understand why h needs to be recalculated, consider the following example. Consider the 1D wave equation defined over domain $\mathcal{D} = [0, L]$ where $L = 1$. Furthermore, we say that the system should produce a fundamental frequency of $f_0 = 750$ Hz which requires a wave speed of $c = 1500$ m/s according to Eq. (2.40). If we use the commonly-used sample rate of $f_\mathrm{s} = 44100$ Hz, and recalling that $k = 1/f_\mathrm{s}$, these values can be filled into (2.47) satisfied with equality and yields $h \approx 0.034$. If we divide the length by the grid spacing, we get $L/h = 29.4$, meaning that exactly 29.4 intervals of size h fit in the domain $\mathcal{D}$. However, the number of intervals needs to be an integer and – using Eq. (2.53) – we get $N = 29$. If h is not recalculated according to (2.53), the total length will be 29 times the grid spacing h. This results in $L \approx 0.986$ and is slightly less than the original length of 1. Although the CFL condition will be satisfied with equality, the fundamental frequency will be slightly higher than desired: $f_0 \approx 760.34$ Hz. If h is recalculated based on N, then L and f_0 will be unchanged, and the system will have the correct fundamental frequency. The Courant number $\lambda \approx 0.986$ is still very close to satisfying condition (2.46), and

the decrease in quality will be perceptually irrelevant – or at the very least, less perceptually relevant than the change in f_0 if h is not recalculated.

more intuition on stability and gridspacing here

Chapter 3

Analysis Techniques

This chapter provides some useful techniques to analyse FD schemes. Techniques to analyse PDEs also exist, but the focus here is of a practical nature and will especially revolve around the discrete schemes. This chapter can be seen as a 'tutorial' on how to use these techniques. Starting off with some necessary theory on matrices and other mathematical tools, this chapter continues to introduce

- *Frequency domain analysis*, which can be used to determine stability conditions of (linear and time-invariant) FD schemes,
- *Energy analysis*, which can both used to debug implementations of FD schemes, as well as determine stability conditions in a more general fashion, and
- *Modal analysis* which can be used to determine the modal frequencies (and damping per mode) that a FD scheme exhibits.

perhaps also dispersion analysis?

3.1 Matrices

For several purposes, such as implementation in `MATLAB` and several analysis techniques described shortly, is useful to write a FD scheme in *matrix form*. A matrix is a rectangular array with numerical elements and its dimensions are denoted using "$row \times column$". A 3×5 matrix, for example, thus has 3 rows and 5 columns (see Figure 3.1a). Along those lines, a *row vector* is a matrix with 1 row and more than 1 column and a *column vector* is a matrix with 1 column and more than 1 row. If a matrix has only 1 row and 1 column, it can be used as a *scalar*.

In this document, matrices and vectors are written using bold symbols. A matrix is denoted by a capital letter – such as $\mathbf{A}$ – whereas vectors are decapitalised – such as $\mathbf{u}$. An element in a matrix is denoted with a non-bold, decapitalised variable, where the subscripts indicate the indices of the row and column. For example, the element in the 2nd row and the 4th column of a matrix $\mathbf{A}$ is denoted as a_{24}. An element in a vector only has one subscript, regardless of whether it is a row or a column vector.

3.1.1 Operations

Multiplying and dividing a matrix by a scalar (a single number) is valid and happens on an element-by-element basis. For a 2×2 matrix $\mathbf{A}$ and scalar p the following operations hold

$$p\mathbf{A} = \mathbf{A}p = \begin{bmatrix} p \cdot a_{11} & p \cdot a_{12} \\ p \cdot a_{21} & p \cdot a_{22} \end{bmatrix}, \quad \text{and} \quad \mathbf{A}/p = \begin{bmatrix} a_{11}/p & a_{12}/p \\ a_{21}/p & a_{22}/p \end{bmatrix}.$$

Notice that although a matrix can be divided by a scalar, a scalar can not necessarily be divided by a matrix. See Section 3.1.3 for more information.

Matrix transpose

A matrix or vector can be *transposed*, and is indicated with the T operator. Transposing a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. means that the elements in the i^{th} row and the j^{th} column of the original matrix become the elements in the j^{th} row and the i^{th} column of the transposed matrix. Essentially the row and column indices of the elements inside the matrix get switched according to

$$a_{ij} = a_{ji}. \tag{3.1}$$

Also see Figure 3.1. For a row vector, the transpose operation simply changes it to a column vector and vice versa. Another way of seeing a transpose is that all the elements get flipped over the *main diagonal* of the matrix. The main diagonal comprises the elements a_{ij} where $i = j$ and a transpose does not affect the location of these elements.

Matrix Multiplication

Matrix multiplication (this includes matrix-vector multiplication) is different from regular multiplication in that it needs to abide several extra rules. In order for matrix multiplication to be valid, the number of columns of the first matrix needs to be equal to the number of rows in the second matrix. The result will then be a matrix with a number of rows equal to that of the first matrix and a number of columns equal to that of the second matrix. See Figure 3.2 for reference.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{bmatrix} \qquad \mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \\ a_{14} & a_{24} & a_{34} \\ a_{15} & a_{25} & a_{35} \end{bmatrix}$$

(a) A 3×5 matrix $\mathbf{A}$. **(b)** A transposed matrix $\mathbf{A}^T$ of size 5×3.

Fig. 3.1: A matrix $\mathbf{A}$ and its transpose $\mathbf{A}^T$. The elements get flipped along the main diagonal of the matrix according to Eq. (3.1).

As an example, consider the $L \times M$ matrix $\mathbf{A}$ and a $M \times N$ matrix $\mathbf{B}$ with $L \neq N$. The multiplication $\mathbf{AB}$ is defined as the number of columns of matrix $\mathbf{A}$ (M) is equal to the number of rows of matrix $\mathbf{B}$ (also M). The result, $\mathbf{C}$, is a $L \times N$ matrix. The multiplication $\mathbf{BA}$ is undefined as the number of columns of the first matrix does not match the number of rows in the second matrix. A valid multiplication of two matrices written in their dimensions is

$$\overbrace{(L \times M)}^{\mathbf{A}} \cdot \overbrace{(M \times N)}^{\mathbf{B}} = \overbrace{(L \times N)}^{\mathbf{C}}. \tag{3.2}$$

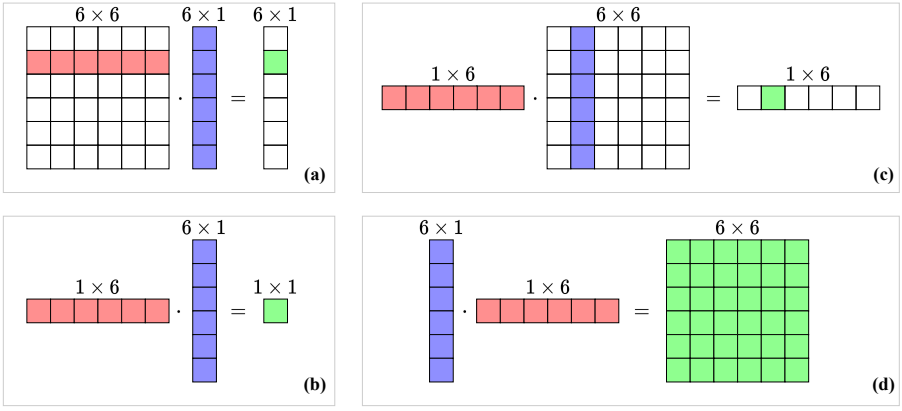


Fig. 3.2: Visualisation of valid matrix multiplications. The "inner" dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of "outer" dimensions (rows of the left matrix and columns of the right).

3.1.2 In a FDTD context

Matrix multiplication when working with FDTD methods usually involves multiplying a square matrix (with equal rows and columns) onto a column vector (see Figure 3.2a). Consider a $(N+1) \times (N+1)$ square matrix $\mathbf{A}$ and a $(N+1) \times 1$ column vector $\mathbf{u}$. Multiplying these results in a $(N+1) \times 1$ column vector $\mathbf{w}$:

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \tag{3.3}$$

Expanding this operation results in

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} a_{00}u_0 + a_{01}u_1 + \ldots + a_{0N}u_N \\ a_{10}u_0 + a_{11}u_1 + \ldots + a_{1N}u_N \\ \vdots \\ a_{N0}u_0 + a_{N1}u_1 + \ldots + a_{NN}u_N \end{bmatrix}}_{\mathbf{w}} \tag{3.4}$$

where the indexing of the matrix elements starts at 0 rather than 1 here, as it relates better to a FDTD context.

Operators in Matrix Form

FD operators approximating spatial derivatives and averages introduced in Section 2.2.2 can be written in matrix form and applied to a column vector $\mathbf{u}^n$ containing the state of the system at time index n. These matrices are square and their sizes depend on the number of grid points the system is described for and the boundary conditions. Not assuming a specific size for now, the FD operators in (2.6) can be written in matrix form according to

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

$$\mathbf{D}_{x\cdot} = \frac{1}{2h}\begin{bmatrix} \ddots & \ddots & & & \mathbf{0} \\ \ddots & 0 & 1 & & \\ & -1 & 0 & 1 & \\ & & -1 & 0 & 1 \\ & & & -1 & 0 \quad \ddots \\ \mathbf{0} & & & & \ddots \quad \ddots \end{bmatrix}$$

where the diagonal dots denote that the values on the respective diagonals continue until the top-left and bottom-right corners of the matrix. The **0**s indicate that the rest of the values in the matrix are zeros.

is this how you explain it?

Averaging operators μ_{x+}, μ_{x-} and $\mu_{x\cdot}$ are defined in a similar way:

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

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

It is important to notice that only spatial operators are written in this matrix form and then applied to state vectors at different time steps ($\mathbf{u}^{n+1}$, $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$).

Finally, the identity matrix is a matrix with only 1s on the diagonal and 0s elsewhere as

$$\mathbf{I} = \begin{bmatrix} \ddots & & & & \mathbf{0} \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 & \\ \mathbf{0} & & & & & \ddots \end{bmatrix}$$

Schemes and Update Equations in Matrix Form

With the spatial operators in matrix form presented above, the FD scheme of the 1D wave equation in Eq. (2.42) can be written in matrix form.

If the Dirichlet boundary conditions in (2.48a) are used, the end points of the system do not have to be included in the calculation. The values of the grid function u_l^n for $l \in \{1, \ldots, N-1\}$ can then be stored in a column vector according to $\mathbf{u}^n = [u_1^n, \ldots, u_{N-1}^n]^T$. Furthermore, $(N-1) \times (N-1)$ matrix

$\mathbf{D}_{xx}$ is defined as

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 1 & -2 \end{bmatrix}. \tag{3.5}$$

If instead, Neumann boundary conditions in Eq. (2.48a) are used, the values of u_l^n for the full range $l \in \{0, \ldots, N\}$ need be stored as $\mathbf{u}^n = [u_0^n, \ldots, u_N^n]^T$ and the $(N+1) \times (N+1)$ matrix $\mathbf{D}_{xx}$ will be

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} -2 & 2 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 2 & -2 \end{bmatrix}, \tag{3.6}$$

where the 2s in the top and bottom row correspond to the multiplication by 2 with u_1^n and u_{N-1}^n in update equations (2.49) and (2.50) respectively.

Regardless of the boundary conditions, the FD scheme in (2.42) can be written in matrix form as

$$\frac{1}{k^2}\left(\mathbf{u}^{n+1} - 2\mathbf{u} + \mathbf{u}^{n-1}\right) = c^2\mathbf{D}_{xx}\mathbf{u}^n, \tag{3.7}$$

and rewritten to a matrix form of the update equation analogous to Eq. (2.44)

$$\mathbf{u}^{n+1} = (2\mathbf{I} + c^2k^2\mathbf{D}_{xx})\mathbf{u}^n - \mathbf{u}^{n-1}. \tag{3.8}$$

The identity matrix is necessary here for correct matrix addition.

3.1.3 Matrix Inverse

If a matrix has the same number of rows as columns, it is called a *square matrix*. Square matrices have special properties, one of which is that it (usually) can be *inverted*. A square matrix $\mathbf{A}$ is invertable if there exists a matrix $\mathbf{B}$ such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \tag{3.9}$$

This matrix $\mathbf{B}$ is then called the *inverse* of $\mathbf{A}$ and can be written as $\mathbf{A}^{-1}$. Not all square matrices have an inverse, in which case it is called *singular*. Rather than going through manually inverting a matrix, or determining whether it is singular, the following function in `MATLAB` will provide the inverse of a matrix `A`:

```
A_inverted = inv(A);
```

The inverse of a *diagonal matrix* (a matrix with non-zero elements on its main diagonal and the rest zeros) is obtained by replacing the diagonal elements by their reciprocal. So for a diagonal 3×3 matrix, the following holds:

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{12} & 0 \\ 0 & 0 & a_{33} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & 0 \\ 0 & \frac{1}{a_{12}} & 0 \\ 0 & 0 & \frac{1}{a_{33}} \end{bmatrix}.$$

3.1.4 Systems of Linear Equations

Matrices can be conveniently used to solve *systems of linear equations*, a set of linear equations containing the same set of variables.

For example, take the system of linear equations

$$\begin{aligned} x + z &= 6, \\ z - 3y &= 7, \\ 2x + y + 3z &= 15, \end{aligned}$$

with independent variables x, y and z. The goal is to find a solution for these variables that satisfy all three equations. This system could be solved by hand using algebraic methods, but alternatively, the system can be written in matrix form:

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \tag{3.10}$$

Here, column vector $\mathbf{u}$ contains the independent variables x, y, and z, matrix $\mathbf{A}$ contains the coefficients multiplied onto these variables and $\mathbf{w}$ contains the right-hand side, i.e., the coefficients not multiplied onto any of the variables:

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & -3 & 1 \\ 2 & 1 & 3 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} 6 \\ 7 \\ 15 \end{bmatrix}}_{\mathbf{w}}$$

We can then solve for $\mathbf{u}$ by taking the inverse of $\mathbf{A}$ (see Section 3.1.3) and multiplying this onto $\mathbf{w}$

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \tag{3.11}$$

Generally, if X unknowns are described by X equations, the unknowns can be solved for using this method.

Solving a system of linear equations can be implemented in MATLAB by using the code given in Section 3.1.3 and multiplying this onto a vector `w`

```
u = inv(A) * w;
```

or more compactly, by using the '\' operator:

```
u = A\w;
```

3.1.5 Eigenvalue Problems

A square matrix $\mathbf{A}$ is characterised by its *eigenvalues* and corresponding *eigenvectors*. In a FDTD context, these are usually associated with the modes of a system, where the eigenvalues relate to the modal frequencies and the eigenvectors to the modal shapes. Section 3.5 will provide more information on this.

ask stefan

To find these characteristic values for a $p \times p$ matrix $\mathbf{A}$, an equation of the following form must be solved

$$\mathbf{A}\phi = \lambda\phi. \tag{3.12}$$

This is called is an *eigenvalue problem* and has p solutions (corresponding to the dimensions of $\mathbf{A}$). These are the p^{th} eigenvector ϕ_p and the corresponding eigenvalue λ_p which is calculated using

$$\lambda_p = \text{eig}_p(\mathbf{A}), \tag{3.13}$$

where $\text{eig}_p(\cdot)$ denotes the p^{th} eigenvalue of. Instead of delving too deep into eigenvalue problems and the process of how to solve them, an easy way to obtain the solutions using `MATLAB` is provided here:

```
[phi, lambda] = eig(A, 'vector');
```

The p^{th} eigenvector appears in the p^{th} column of $p \times p$ matrix `phi` and the corresponsding eigenvalues are given in a $p \times 1$ column vector `lambda`.

3.2 Mathematical Tools and Product Identities

Some useful mathematical tools used for the energy analysis techniques presented in Section 3.4 will be shown here. The tools shown here can be applied to 1D systems. These will be extended to 2D systems in Chapter 6.

3.2.1 Inner product

For two functions $f(x,t)$ and $g(x,t)$ defined for $x \in \mathcal{D}$ where $\mathcal{D} = [0, L]$, their l_2 inner product and l_2 norm are defined as

$$\langle f, g\rangle_{\mathcal{D}} = \int_{\mathcal{D}} fg dx \quad \text{and} \quad \|f\|_{\mathcal{D}} = \sqrt{\langle f, f\rangle_{\mathcal{D}}}. \tag{3.14}$$

These functions do not have to be time-dependent (i.e., they can also simply be $f(x)$ and $g(x)$), but as all functions used in this work are in fact time-dependent, this is left for coherence. It is also important to note that these functions do not have to be 'isolated' state variables per se (such as $u(x,t)$ used in the previous

chapter), but could also be a state variable with a derivative applied to it (such as $\partial_t u(x,t)$).

The discrete inner product of any two (1D) functions f_l^n and g_l^n defined for $l \in d$, with discrete domain $d = \{0, \ldots, N\}$, is

$$\langle f_l^n, g_l^n \rangle_d = \sum_{l=0}^{N} h f_l^n g_l^n, \tag{3.15}$$

where the multiplication by h is the discrete counterpart of dx in the continuous definition in (3.14). Also useful are the primed inner product

$$\langle f_l^n, g_l^n \rangle_d' = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{h}{2} f_0^n g_0^n + \frac{h}{2} f_N^n g_N^n, \tag{3.16}$$

and the more general weighted inner product

$$\langle f_l^n, g_l^n \rangle_d^{\epsilon_\mathrm{l}, \epsilon_\mathrm{r}} = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{\epsilon_\mathrm{l}}{2} h f_0^n g_0^n + \frac{\epsilon_\mathrm{r}}{2} h f_N^n g_N^n, \tag{3.17}$$

which scale the boundary points of the regular inner product. Naturally, if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 1$, Eq. (3.17) reduces to Eq. (3.16), and if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 2$, (3.17) reduces to (3.15).

3.2.2 Summation by Parts

Extremely useful when performing energy analysis on distributed systems is *summation by parts*, which is the discrete counterpart of integration by parts. Although its application will be only be apparent when actually performing an energy analysis (see fx. Sections 3.4.3 and 4.4) some definitions will be presented here for future reference.

Here, the same functions as in the previous section, $f(x,t)$ and $g(x,t)$ and domain $\mathcal{D}$, will be used. Applying a spatial derivative to g, and using Eq. (3.14), integration by parts is defined as

$$\langle f, \partial_x g \rangle_\mathcal{D} = -\langle \partial_x f, g \rangle_\mathcal{D} + fg|_0^L \tag{3.18}$$

where $fg|_0^L$ describes the boundary terms that appeared in the process. One can observe that the spatial derivative switched function and is now applied to f rather than g.

In discrete time, we use the same two (1D) functions as before f_l^n and g_l and are defined for $l \in d$ with discrete domain $d = \{0, \ldots, N\}$. Then, using the discrete inner product in Eq. (3.15), two variants of summation by parts are defined as

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_d + f_{N+1}^n g_N^n - f_0^n g_{-1}^n, \tag{3.19a}$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_d + f_N^n g_{N+1}^n - f_{-1}^n g_0^n. \tag{3.19b}$$

A derivation of Eq. (3.19a) is given below. As in the case of integration by parts in Eq. (3.18), the process of summation by parts causes the derivative to be applied to the other function and the sign of the resulting inner product changes. Important to note, is that the sign (forward / backward) of the derivative operator has also changed. Lastly, discrete boundary terms have appeared and it can be seen that values outside of the defined domain are needed, i.e., g^n_{N+1} and f^n_{-1}. These can be accounted for by the boundary conditions imposed on the system (see Section 2.4.2 as an example).

One could also choose to work with reduced domains after summation by parts. Domains that have one fewer point at the boundaries are defined as $\underline{d} = \{0, \dots, N-1\}$, $\overline{d} = \{1, \dots, N\}$ and $\underline{\overline{d}} = \{1, \dots, N-1\}$. The following identities can be shown to hold

$$\langle f^n_l, \delta_{x-} g^n_l \rangle_d = -\langle \delta_{x+} f^n_l, g^n_l \rangle_{\underline{d}} + f^n_N g^n_N - f^n_0 g^n_{-1}, \tag{3.20a}$$
$$\langle f^n_l, \delta_{x+} g^n_l \rangle_d = -\langle \delta_{x-} f^n_l, g^n_l \rangle_{\overline{d}} + f^n_N g^n_{N+1} - f^n_0 g^n_0, \tag{3.20b}$$

and, using the primed inner product in Eq. (3.16),

$$\langle f^n_l, \delta_{x-} g^n_l \rangle'_d = -\langle \delta_{x+} f^n_l, g^n_l \rangle_{\underline{d}} + f^n_N \mu_{x-} g^n_N - f^n_0 \mu_{x-} g^n_0, \tag{3.21a}$$
$$\langle f^n_l, \delta_{x+} g^n_l \rangle'_d = -\langle \delta_{x-} f^n_l, g^n_l \rangle_{\overline{d}} + f^n_N \mu_{x+} g^n_N - f^n_0 \mu_{x+} g^n_0, \tag{3.21b}$$

all of which will prove useful in energy analysis techniques later on. A derivation of (3.20a) is given below.

Finally, recalling that $\delta_{xx} = \delta_{x+}\delta_{x-}$, one can apply summation by parts twice to get the following identities

$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \tag{3.22a}$$
$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_{\underline{\overline{d}}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0, \tag{3.22b}$$
$$\langle f, \delta_{xx} g \rangle'_d = \langle \delta_{xx} f, g \rangle'_d + f_N \delta_{x\cdot} g_N - g_N \delta_{x\cdot} f_N - f_0 \delta_{x\cdot} g_0 + g_0 \delta_{x\cdot} f_0 \tag{3.22c}$$

Derivations

To see why the above identities hold true, it is useful to briefly go through a derivation. As an example, we go through Eqs. (3.19a) and (3.20a) as they have the same inner product as a starting point, but yield different results. In the following, $d = \{0, \dots, N\}$ and $N = 2$ are used.

Starting with Eq. (3.19a), suppressing the n superscript for brevity, and

using the definition for the discrete inner product in Eq. (3.15), we get

$$\begin{aligned}
\langle f_l, \delta_{x-} g_l \rangle_d &= \sum_{l=0}^{2} h f_l \frac{1}{h} \left(g_l - g_{l-1} \right), \\
&= f_0 g_0 - f_0 g_{-1} + f_1 g_1 - f_1 g_0 + f_2 g_2 - f_2 g_1, \\
&= g_0 (f_0 - f_1) - f_0 g_{-1} + g_1 (f_1 - f_2) + g_2 (f_2 - f_3) + f_3 g_2, \\
&= -g_0 (f_1 - f_0) - g_1 (f_2 - f_1) - g_2 (f_3 - f_2) + f_3 g_2 - f_0 g_{-1}, \\
&= -\sum_{l=0}^{2} h g_l \frac{1}{h} \left(f_{l+1} - f_l \right) + f_3 g_2 - f_0 g_{-1}, \\
&= -\langle \delta_{x+} f_l, g_l \rangle_d + f_3 g_2 - f_0 g_{-1}.
\end{aligned}$$

As $N = 2$, the result is identical to Eq. (3.19a).

Similarly, identity (3.20a) can be proven to hold:

$$\begin{aligned}
\langle f_l, \delta_{x-} g_l \rangle_d &= \sum_{l=0}^{2} h f_l \frac{1}{h} \left(g_l - g_{l-1} \right), \\
&= f_0 g_0 - f_0 g_{-1} + f_1 g_1 - f_1 g_0 + f_2 g_2 - f_2 g_1, \\
&= -f_0 g_{-1} + g_0 (f_0 - f_1) + g_1 (f_1 - f_2) + f_2 g_2, \\
&= -g_0 (f_1 - f_0) - g_1 (f_2 - f_1) + f_2 g_2 - f_0 g_{-1}, \\
&= \sum_{l=0}^{1} h g_l \frac{1}{h} (f_{l+1} - f_l) + f_2 g_2 - f_0 g_{-1}, \\
&= -\langle \delta_{x+} f_l, g_l \rangle_{\underline{d}} + f_2 g_2 - f_0 g_{-1},
\end{aligned}$$

where the resulting inner product has a reduced domain of $\underline{d} = \{0, \ldots, N-1\}$. Similar processes can be used to prove the other identities presented in this section.

3.2.3 Product identities

Some useful identities used in this work are

$$(\delta_{t\cdot} u_l^n)(\delta_{tt} u_l^n) = \delta_{t+}\left(\frac{1}{2}(\delta_{t-} u_l^n)^2\right), \tag{3.23a}$$

$$(\delta_{t\cdot} u_l^n) u_l^n = \delta_{t+}\left(\frac{1}{2} u_l^n e_{t-} u_l^n\right), \tag{3.23b}$$

$$(\delta_{t+} u_l^n)(\mu_{t+} u_l^n) = \delta_{t+}\left(\frac{1}{2}(u_l^n)^2\right), \tag{3.23c}$$

$$(\delta_{t\cdot} u_l^n)(\mu_{t\cdot} u_l^n) = \delta_{t\cdot}\left(\frac{1}{2}(u_l^n)^2\right), \tag{3.23d}$$

$$u_l^n e_{t-} u_l^n = (\mu_{t-} u_l^n)^2 - \frac{k^2}{4}(\delta_{t-} u_l^n)^2 \tag{3.23e}$$

These identities can be used for spatial derivatives as well by substituting the 't' subscripts for 'x'.

When an operator is applied to a product of two grid functions, the discrete counterpart of the product rule needs to be used according to

$$\delta_{t+}(u_l^n w_l^n) = (\delta_{t+} u_l^n)(\mu_{t+} w_l^n) + (\mu_{t+} u_l^n)(\delta_{t+} w_l^n). \tag{3.24}$$

3.3 Frequency Domain Analysis

Frequency domain analysis, also called Fourier analysis, is a way to determine various properties of a FD scheme, including conditions for stability. The process is similar to finding stability for digital filters. In essence, a FD scheme can be seen as a complex filter of which its coefficients are defined by physical parameters. This section will explain how to obtain a frequency-domain representation of a scheme and will mainly follow [3], albeit in a slightly more practical manner.

Frequency-domain representation and Ansatz

Frequency-domain analysis of FD schemes starts by performing a *z-transform* on the scheme. The z-transform converts a discrete signal into a frequency-domain representation, and is extensively used in the field of digital signal processing (DSP) to analyse the behaviour and especially stability of digital filters. To not go too much into detail here, the interested reader is referred to the very comprehensive explanation on the z-transform given in [38, Ch. 5].

If a system is distributed in space, one can perform a spatial Fourier transform on a grid function. Frequency-domain analysis in the distributed case is called *von Neumann analysis* which first appeared in [39] co-authored by John

von Neumann. Later, this technique got a more general treatment in [31] and is heavily used in [3]. The discrete-time z-transform and discrete spatial Fourier transform performed on a 1D grid function are defined as [3]

$$\hat{u} = \sum_{n=-\infty}^{\infty} u_l^n z^{-n} \quad \text{and} \quad \tilde{u} = \sum_{l=-\infty}^{\infty} u_l^n e^{-jl\beta h} \tag{3.25}$$

with complex number $z = e^{sk}$, complex frequency $s = j\omega + \sigma$ (more elaborated on in 3.5) and real wavenumber β. Frequency-domain analysis in 2D will be elaborated on in Section 6.1.

check if I should not refer to a subsection

A shortcut to performing a full frequency-domain analysis is to use a test solution, or *ansatz*, and replace the grid functions by their transforms. The grid function for a 1D system can be replaced by an ansatz of the form (1D) [31]

check reference

$$u_l^n \overset{\mathcal{A}}{\Longrightarrow} z^n e^{jl\beta h} \tag{3.26}$$

where "$\overset{\mathcal{A}}{\Longrightarrow}$" indicates to replace the grid function with the ansatz (the shortcut to taking the full z-transform and spatial Fourier transform).

Like in the DSP realm, the power of z indicates a temporal shift, i.e., z^{-1} is a one-sample delay. In a FDTD context, this corresponds to a time shift as seen in Section 2.2.2. For spatially distributed systems, a shift in l can be interpreted as a phase shift of a frequency with wavenumber β. See Table 3.1 for the frequency-domain representation of of grid functions with their temporal and spatial indices shifted in different ways.

check

Grid function	Ansatz	Result
u_l^n	$z^0 e^{j0\beta h}$	1
u_l^{n+1}	$z^1 e^{j0\beta h}$	z
u_l^{n-1}	$z^{-1} e^{j0\beta h}$	z^{-1}
u_{l+1}^n	$z^0 e^{j1\beta h}$	$e^{j\beta h}$
u_{l-1}^n	$z^0 e^{j(-1)\beta h}$	$e^{-j\beta h}$
u_{l+2}^n	$z^0 e^{j2\beta h}$	$e^{j2\beta h}$
u_{l-2}^n	$z^0 e^{j(-2)\beta h}$	$e^{-j2\beta h}$
u_{l+1}^{n-1}	$z^{-1} e^{j1\beta h}$	$z^{-1} e^{j\beta h}$
u_{l-1}^{n-1}	$z^{-1} e^{j(-1)\beta h}$	$z^{-1} e^{-j\beta h}$

Table 3.1: Frequency-domain representation of a grid function using ansatz (3.26) with frequently appearing temporal and spatial shifts.

Using these definitions, the effect of various operators on a grid function can be written in their frequency-domain representation. For systems distributed

in space, the following trigonometric identities are extremely useful when performing the analyses [40, p. 71]:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \quad \Rightarrow \quad \sin^2(x) = \frac{e^{j2x} + e^{-j2x}}{-4} + \frac{1}{2}, \tag{3.27a}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \quad \Rightarrow \quad \cos^2(x) = \frac{e^{j2x} + e^{-j2x}}{4} + \frac{1}{2}. \tag{3.27b}$$

Take for example

$$\delta_{xx} u_l^n = \frac{1}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \overset{\mathcal{A}}{\Longrightarrow} \frac{1}{h^2}\left(e^{j\beta h} - 2 + e^{-j\beta h}\right).$$

Then, using $x = \beta h/2$, identity (3.27a) can be rewritten to

$$e^{j\beta h} - 2 + e^{-j\beta h} = -4\sin^2(\beta h/2),$$

and substituted into the above to get

$$\delta_{xx} u_l^n \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2}\sin^2(\beta h/2).$$

Examples of various temporal FD operators applied to grid functions in their frequency-domain representation are

$$\begin{aligned} \delta_{t+} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(z - 1\right), & \delta_{t-} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(1 - z^{-1}\right), \\ \delta_{t\cdot} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{2k}\left(z - z^{-1}\right), & \delta_{tt} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k^2}\left(z - 2 + z^{-1}\right) \end{aligned} \tag{3.28}$$

and for spatial operators identity (3.27a) can be used to obtain

$$\delta_{xx} u_l^n \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2}\sin^2(\beta h/2), \tag{3.29a}$$

$$\delta_{xxxx} u_l^n \overset{\mathcal{A}}{\Longrightarrow} \frac{16}{h^4}\sin^4(\beta h/2). \tag{3.29b}$$

Frequency-domain analysis only works on linear and time-invariant (LTI) systems and assumes systems with infinite domains. Energy analysis allows for nonlinear systems to be analysed (see Section 3.4) as well as handling boundary conditions.

FULL DOC SWEEP: nonlinear, non-linear or non linear

not talking about nonlinear systems though

Proving stability

Similar to digital filters, the system is stable when the roots of the characteristic polynomial in z are bounded by 1 (unity)

only the denominator of the transfer function

$$|z| \leq 1. \tag{3.30}$$

In the FDTD context, the frequency-domain representation of a FD scheme results in a *characteristic equation* – which is usually a second-order polynomial – in z and needs to satisfy condition (3.30) for all wave numbers β. It can be shown that for a polynomial of the form

$$z^2 + a^{(1)}z + a^{(2)} \tag{3.31}$$

its roots satisfy condition (3.30) when it abides the following condition [3]

$$|a^{(1)}| - 1 \leq a^{(2)} \leq 1. \tag{3.32}$$

If $a^{(2)} = 1$, the simpler condition

$$|a^{(1)}| \leq 2, \tag{3.33}$$

suffices.

3.3.1 Mass-Spring System

Recalling the FD scheme of the mass-spring system in Eq. (2.35)

$$M\delta_{tt}u^n = -Ku^n$$

a frequency-domain representation can be obtained using the ansatz in (3.26) with $l = 0$. Using Table 3.1 and Eqs. (3.28) as a reference and substituting the definitions yields

$$\frac{M}{k^2}\left(z - 2 + z^{-1}\right) = -K.$$

Gathering the terms and moving all to the left-hand side, the characteristic equation for the mass-spring system can be obtained:

$$z - \left(2 - \frac{Kk^2}{M}\right) + z^{-1} = 0. \tag{3.34}$$

To begin to prove stability, this equation needs to be written in the form found in (3.31). Multiplying all the terms by z, and noticing that $a^{(2)} = 1$, we could continue with condition (3.33). However, the scheme used here is a special case where the roots of the characteristic equation can not be identical [3]. When this happens, the output of the system will grow linearly and is called "marginally unstable". This means that $|a^{(1)}| \neq 1$ and the condition in (3.33) becomes $|a^{(1)}| < 2$. Continuing with this conditions yields

$$\begin{aligned} \left|-2 + \frac{Kk^2}{M}\right| &< 2, \\ -2 < -2 + \frac{Kk^2}{M} &< 2, \\ 0 < \frac{Kk^2}{M} &< 4. \end{aligned}$$

If only non-zero values are chosen for K, k and M they are positive and the first condition is always satisfied. The second condition is then easily solved for k by

$$k < 2\sqrt{\frac{M}{K}}. \tag{3.35}$$

Recalling that $\omega_0 = \sqrt{K/M}$, Eq (3.35) can be more compactly written as

$$k < \frac{2}{\omega_0}. \tag{3.36}$$

3.3.2 1D Wave Equation

This section will derive the stability condition for the 1D wave equation presented in Section 2.4 using von Neumann analysis.

Recalling the FD scheme in (2.42):

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n,$$

its frequency-domain representation can be obtained using the definitions in Eqs. (3.28) and (3.29a):

$$\frac{1}{k^2}\left(z - 2 + z^{-1}\right) = -\frac{4c^2}{h^2}\sin^2\left(\beta h/2\right). \tag{3.37}$$

Also recalling that

$$\lambda = \frac{ck}{h},$$

the characteristic equation of the 1D wave equation is

$$z + \left(4\lambda^2 \sin^2(\beta h/2) - 2\right) + z^{-1} = 0. \tag{3.38}$$

The scheme is then stable if the roots satisfy condition (3.30). As the characteristic equation is of the form in (3.31) (after multiplication with z) with $a^{(2)} = 1$, stability is shown by abiding condition (3.33) for all β and when applied to the characteristic equation (3.38), it can be seen that

$$\begin{aligned}
|4\lambda^2 \sin^2(\beta h/2) - 2| &\leq 2, \\
|2\lambda^2 \sin^2(\beta h/2) - 1| &\leq 1, \\
-1 \leq 2\lambda^2 \sin^2(\beta h/2) - 1 &\leq 1, \\
0 \leq 2\lambda^2 \sin^2(\beta h/2) &\leq 2, \\
0 \leq \lambda^2 \sin^2(\beta h/2) &\leq 1.
\end{aligned}$$

Observing that all terms in $\lambda^2 \sin^2(\beta h/2)$ are squared, this term will always be non-negative and therefore always satisfy the first condition. Continuing with

the second condition, and knowing that the $\sin^2(\beta h/2)$-term is bounded by 1 for all β, we arrive at the following stability condition:

$$\lambda \leq 1.$$

This is the CFL condition given in Eq. (2.46). To obtain the stability condition in terms of the grid spacing, the definition for λ is substituted and written in terms of the grid spacing

$$h \geq ck, \tag{3.39}$$

which is the stability condition given in Eq. (2.47).

3.4 Energy Analysis

Of all analysis techniques described in this chapter, energy analysis is without a doubt the most important when working with FDTD methods. First of all, from a practical point of view, it is essential for debugging implementations of FD schemes. Especially when trying to model more complex systems, programming errors are unavoidable, and energy analysis can be extremely helpful in pinpointing where the error lies. Secondly, energy analysis techniques can be used to obtain stability conditions in a much more general sense than the frequency-domain analysis techniques presented in Section 3.3. Where frequency-domain analysis is restricted to LTI systems with infinite domains (for distributed systems), energy analysis can be applied to nonlinear systems and boundary conditions [3].

Gustafsson et al. in (the first edition of) [41] worked with energy to find stability conditions for FD schemes. This, they referred to as 'the energy method' and it effectively circumvented the need of a frequency domain representation to find stability conditions (as presented in Section 3.3). Later, energy, or more specifically 'energy as a conserved quantity', was used to determine stability and passivity of systems. Bilbao gives an extensive overview in [3] where this has been extensively used to show stability of the FD schemes used.

One of the main goals when performing energy analysis is to find an expression for the total energy present in the system. This is referred to as the *Hamiltonian* and denoted by $\mathfrak{H}$ in continuous time and $\mathfrak{h}$ in discrete time. In this work, the focus of the energy analysis will be in discrete time.

In this section, four steps are presented and can be followed to perform a full energy analysis of a FD scheme and implement it afterwards. Then, the analysis will be performed on the mass-spring system and the 1D wave equation presented in Chapter 2. Finally, it will be shown how to obtain stability conditions through the techniques presented in this section.

3.4.1 Energy Analysis: A 4-Step Tutorial

Step 1: Obtain the rate of change of the total energy $\delta_{t+}\mathfrak{h}$.

The first step to energy analysis is to take the appropriate *norm* of the scheme (see Eq. (3.14)), which yields and expression for the rate of change of the energy of the system: $\delta_{t+}\mathfrak{h}$. Usually, this means to take the inner product of the scheme with $(\delta_{t\cdot}u_l^n)$. See Section 3.2.1 for more details on the inner product. Note that the forward time difference δ_{t+} is used (and not the backwards or centred) because of convention and preference [Bilbao, verbally].

For the units of the resulting energy balance to add up (also see Step 3), it is useful to perform the analysis on a scheme with all physical parameters written out (so the discretised version of Eq. (2.28) rather than Eq. (2.29)).

Step 2: Identify different types of energy and obtain the total energy $\mathfrak{h}$ by isolating δ_{t+}.

The energy of a FD scheme can generally be divided into three different types: the total energy contained within the system, or Hamiltonian $\mathfrak{h}$, energy losses through damping $\mathfrak{q}$ and energy input through external forces or excitations $\mathfrak{p}$. For distributed systems, an additional boundary term $\mathfrak{b}$ appears, but vanishes under 'regular' (lossless and not energy-storing) boundary conditions. Nearly any energy balance is thus of the form

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q} - \mathfrak{p}. \tag{3.40}$$

This equation essentially says that the total energy present in the system changes due to losses and inputs. For a lossless system without externally supplied energy over the course of the simulation (so initial conditions excluded), the energy should remain unchanged over the course of the simulation,

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.41}$$

As the eventual interest lies in the total energy of the system $\mathfrak{h}$ and not its rate of change, δ_{t+} must be isolated in the definition of $\delta_{t+}\mathfrak{h}$. In this step, the identities in Section 3.2.3 will come in handy, as well as summation by parts described in Section 3.2.2 for distributed systems.

The Hamiltonian itself can usually be further subdivided into kinetic energy and potential energy, denoted by the symbols $\mathfrak{t}$ and $\mathfrak{v}$ respectively:

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \tag{3.42}$$

As a rule of thumb, the definition for kinetic energy contains 'velocity squared' (as in the classical-mechanics definition $E_{\text{kin}} = \frac{1}{2}M\dot{u}$) and the potential energy includes the restoring forces of the system.

Step 3: Check the units in the expression for $\mathfrak{h}$.

To know that the previous steps have been carried out correctly, it is good to check whether the units of the resulting expression for $\mathfrak{h}$ is indeed in Joules, or $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$. The other quantities such as energy losses $\mathfrak{q}$ and inputs $\mathfrak{p}$, should be in Joules per second or in SI units: $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$. Some information about operators and grid functions and how they 'add' units to the equation will be given below.

An (1D) inner product (or norm) will 'add' one 'm' unit due to the h in its definition in (3.14). A first-order temporal difference operator will 'add' one 's^{-1}'-unit (because of the $1/k$) and a first-order spatial difference operator will 'add' one 'm^{-1}'-unit ($1/h$). Along these lines, a second-order time or difference operator will 'add' a 's^{-2}' ($1/k^2$) or 'm^{-2}'-unit ($1/h^2$) respectively. It is important to note that the time shift operator (e_{t-}) does not influence the units. Finally, the appearance of a grid function u_l^n 'adds' whatever it describes. Usually, as u_l^n describes a displacement in m, it will 'add' this to the equation. If it describes anything else, it will 'add' that.

Step 4: Implement the definitions for energy and debug the FD scheme.

In the end, the definition for the energy can be implemented and used as a check for whether the FD scheme has been implemented correctly. Usually, the energy of the system is calculated for every iteration in the for loop and plotted after the simulation. For a system without losses or energy inputs, the energy should be unchanged according to Eq. (3.41) and can be plotted according

$$\mathfrak{h}_\text{e}^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0, \tag{3.43}$$

where $\mathfrak{h}_\text{e}^n$ can be seen as the normalised energy and shows the error variation. Although this equation should always return 0 (as $\mathfrak{h}^n = \mathfrak{h}^0$), in a finite precision simulation, ultra slight fluctuations of the energy should be visible due to rounding errors. Plotting the Hamiltonian should show fluctuations within *machine precision*, which is usually in the range of 10^{-15}. Over time, the fluctuations can add up, and possibly end up out of this range, but generally, any fluctuations less than in the 10^{-10} range indicate that there is no programming error. See fx. Figures 3.3 and 3.4.

For a system with losses or energy inputs, a discrete integration, or summed form can be used (as done in fx. [42]):

$$\mathfrak{h}_\text{e}^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0 + k\sum_{m=0}^{n-1}(\mathfrak{q}^m + \mathfrak{p}^m)}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0. \tag{3.44}$$

check if the sum should indeed go until $n-1$ and why

3.4.2 Mass-spring system

Recalling the FD scheme for the simple mass-spring system in Eq. (2.34)

$$M\delta_{tt}u^n = -Ku^n$$

we can start to perform an energy analysis using the five steps described above.

Step 1: Obtain $\delta_{t+}\mathfrak{h}$

The energy balance of the simple mass-spring system presented in Section 2.3 can be obtained by first taking the product of scheme (2.34) with $(\delta_{t\cdot}u^n)$:

$$\delta_{t+}\mathfrak{h} = M(\delta_{t\cdot}u^n)(\delta_{tt}u^n) + K(\delta_{t\cdot}u^n)(u^n) = 0. \tag{3.45}$$

Note that the inner product is not necessary as the system is not distributed.

Step 2: Identify energy types and isolate δ_{t+}

As there are no losses or externally supplied energy present in the system, all terms are part of the Hamiltonian $\mathfrak{h}$. To isolate δ_{t+} from (3.45), one can use identities (3.23a) and (3.23b) to get the following:

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n\right) = 0, \tag{3.46}$$

and the following definition for $\mathfrak{h}$ can be obtained

$$\mathfrak{h} = \frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n = 0. \tag{3.47}$$

This can be rewritten in terms of the kinetic energy $\mathfrak{t}$ and potential energy $\mathfrak{v}$ according to

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-}u^n)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}u^n e_{t-}u^n. \tag{3.48}$$

Step 3: Check units

As mentioned above, the energy $\mathfrak{h}$ needs to be in Joules, or $\text{kg}\cdot\text{m}^2\cdot\text{s}^{-2}$. Taking the terms in Eq. (3.48) one-by-one and writing them in their units results in

$$\mathfrak{t} = \frac{M}{2}(\delta_{t-}u^n)^2 \xrightarrow{\text{in units}} \quad \text{kg}\cdot(\text{s}^{-1}\cdot\text{m})^2 = \text{kg}\cdot\text{m}^2\cdot\text{s}^{-2},$$
$$\mathfrak{v} = \frac{K}{2}u^n e_{t-}u^n \xrightarrow{\text{in units}} \quad \text{N}\cdot\text{m}^{-1}\cdot\text{m}\cdot\text{m} = \text{kg}\cdot\text{m}^2\cdot\text{s}^{-2},$$

and indeed have the correct units.

Step 4: Implementation

Equation (3.53) can then be implemented in same for-loop recursion where the update is calculated.

```
%% Calculate the energy using Eq. (3.48)

% Kinetic energy
kinEnergy(n) = M / 2 * (1/k * (u - uPrev))^2;

% Potential energy
potEnergy(n) = K / 2 * u * uPrev;

% Total energy (Hamiltonian)
totEnergy(n) = kinEnergy(n) + potEnergy(n);
```

Figure 3.3 shows the normalised energy (according to Eq. (3.43)) of the mass-spring system and shows that the deviation is indeed within machine precision.

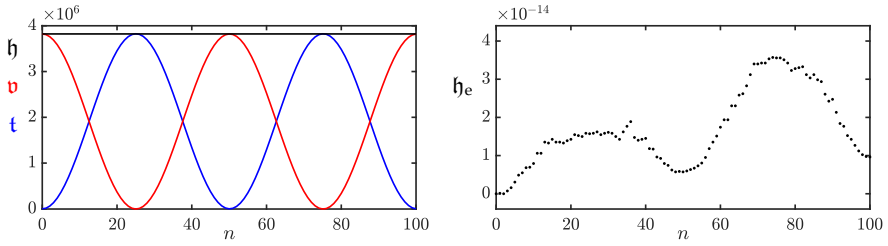


Fig. 3.3: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.43)). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within machine precision.

3.4.3 1D Wave Equation

Energy analysis could be directly performed on the FD scheme in (2.42). However, in order for the units of the scheme to add up to energy in Joules, it is useful to write out all physical parameters. Taking the definition for the wave speed for the ideal string $c = \sqrt{T/\rho A}$ and multiplying both sides of Eq. (2.42) by ρA yields

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n, \tag{3.49}$$

where $l \in d$ with discrete domain $d \in \{0, \ldots, N\}$ and number of grid points $N + 1$. Furthermore, Dirichlet boundary conditions as given in Eq. (2.48a) are used. A note on using Neumann boundary conditions is given at the end of this section.

Step 1: Obtain $\delta_{t+}\mathfrak{h}$

Taking an inner product using Eq. (3.49) with $(\delta_{t\cdot}u_l^n)$ and moving all terms to the left-hand side yields the definition for the rate of change of the Hamiltonian:

$$\delta_{t+}\mathfrak{h} = \rho A\langle \delta_{t\cdot}u_l^n, \delta_{tt}u_l^n\rangle_d - T\langle \delta_{t\cdot}u_l^n, \delta_{xx}u_l^n\rangle_d = 0. \tag{3.50}$$

Step 2: Identify energy types and isolate δ_{t+}

As in the case of the mass-spring system in the previous section, there are no losses or externally supplied energy present in the system, and all terms are part of the Hamiltonian $\mathfrak{h}$.

To isolate δ_{t+} in Eq. (3.50), the terms have to be rewritten in a way that fits the product identities in Section 3.2.3. Summation by parts as described in Section 3.2.2 can be used. Using identity (3.20a) with $f_l^n \triangleq \delta_{t\cdot}u_l^n$ and $g_l^n \triangleq \delta_{x+}u_l^n$, the second term can be rewritten to

$$-T\langle \delta_{t\cdot}u_l^n, \delta_{xx}u_l^n\rangle_d = T\langle \delta_{x+}(\delta_{t\cdot}u_l^n), \delta_{x+}u_l^n\rangle_{\underline{d}} - \mathfrak{b},$$

where the boundary term

$$\mathfrak{b} = T(\delta_{t\cdot}u_N^n)(\delta_{x+}u_N^n) - T(\delta_{t\cdot}u_0^n)(\delta_{x+}u_{-1}^n),$$

and reduced domain $\underline{d} = \{0, \ldots, N-1\}$. As Dirichlet boundary conditions are used, the boundary term vanishes as

$$u_0^n = u_N^n = 0 \quad \Longrightarrow \quad \delta_{t\cdot}u_0^n = \delta_{t\cdot}u_N^n = 0.$$

In other words, if the states of the system at the boundaries are zero, their velocity will also be zero. Then, using the discrete inner product in Eq. (3.15), Eq. (3.50) can be expanded to

$$\delta_{t+}\mathfrak{h} = \rho A\sum_{l=0}^{N} h(\delta_{t\cdot}u_l^n)(\delta_{tt}u_l^n) + T\sum_{l=0}^{N} h(\delta_{t\cdot}\delta_{x+}u_l^n)(\delta_{x+}u_l^n) \tag{3.51}$$

Then, using identities (3.23a) and (3.23b), δ_{t+} can be isolated

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}\right), \tag{3.52}$$

and the definition for the Hamiltonian and the kinetic and potential energy can be found:

$$\begin{gathered}\mathfrak{h} = \mathfrak{t} + \mathfrak{v},\\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad \text{and} \quad \mathfrak{v} = \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}.\end{gathered} \tag{3.53}$$

Step 3: Check units

Writing out the definitions for kinetic and potential energy in Eq. (3.53) respectively, yields

$$\begin{aligned}\mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 &\xrightarrow{\text{in units}} \mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2\cdot\mathrm{m}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})^2 \\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2}, \\ \mathfrak{v} = \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}} &\xrightarrow{\text{in units}} \mathrm{N}\cdot\mathrm{m}\cdot(\mathrm{m}^{-1}\cdot\mathrm{m}\cdot\mathrm{m}^{-1}\cdot\mathrm{m}^{-1}\mathrm{m}) \\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},\end{aligned}$$

and are indeed in Joules. Notice that an extra 'm' unit appears due to the norm and inner product.

Step 4: Implementation

The energy balance in Eq. (3.53) can be implemented with the following code in the for-loop recursion:

```
%% Calculate the energy using Eq. (3.53)

% Kinetic energy
kinEnergy(n) = rho * A / 2 * h * sum((1/k * (u-uPrev)).^2);

% Potential energy
potEnergy(n) = T/(2*h) * sum(([u; 0] - [0; u]) ...
             .* ([uPrev; 0] - [0; uPrev]));

% Total energy (Hamiltonian)
totEnergy(n) = kinEnergy(n) + potEnergy(n);
```

Here, `u` is the vector $\mathbf{u} = [u_1^n, \ldots, u_{N-1}^n]^T$ (as Dirichlet boundary conditions are used) and need to be concatenated with 0 in the calculation of the potential energy as the boundaries need to be included in the calculation (despite them being 0!). Figure 3.4 shows the plot of the normalised energy according to Eq. (3.43) and shows that the deviation of $\mathfrak{h}^n$ is within machine precision.

Neumann boundary conditions

If Neumann boundary conditions – as per Eq. (2.48b) – are used instead, the primed inner product in Eq. (3.16) needs to be used in Step 1. Using the identity in (3.21a), summation by parts of the second term results in

$$-T\langle\delta_{t\cdot}u_l^n, \delta_{xx}u_l^n\rangle_d' = T\langle\delta_{x+}(\delta_{t\cdot}u_l^n), \delta_{x+}u_l^n\rangle_{\underline{d}} - \mathfrak{b},$$

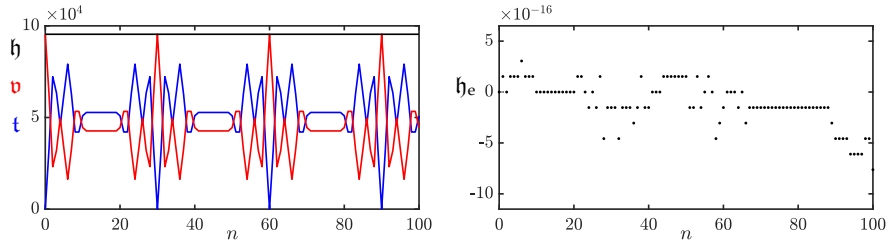


Fig. 3.4: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.43)) and shows that the deviation of the energy is within machine precision.

where the boundary term

$$\begin{aligned} \mathfrak{b} &= T(\delta_{t.}u_N^n)(\mu_{x-}\delta_{x+}u_N^n) - T(\delta_{t.}u_0^n)(\mu_{x-}\delta_{x+}u_0^n), \\ \overset{\text{Eq. (2.27b)}}{\Longleftrightarrow} \quad &= T(\delta_{t.}u_N^n)(\delta_{x.}u_N^n) - T(\delta_{t.}u_0^n)(\delta_{x.}u_0^n). \end{aligned}$$

As the Neumann boundary condition states that

$$\delta_{x.}u_0^n = \delta_{x.}u_N^n = 0$$

the boundary term vanishes and the energy balance results in

$$\begin{gathered} \mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\left(\|\delta_{t-}u_l^n\|_d'\right)^2, \quad \text{and} \quad \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}. \end{gathered} \tag{3.54}$$

Using u for the vector $\mathbf{u} = [u_0^n, \ldots, u_N^n]^T$, this is then implemented as

```
%% Calculate the energy using Eq. (3.54)

% Scaling of the boundaries through weighted inner product
scaling = [0.5; ones(N-1, 1); 0.5];

% Kinetic energy
kinEnergy(n) = rho * A / 2 * h * sum(scaling .* (1/k * (u-uPrev)).^2);

% Potential energy
potEnergy(n) = T/(2*h) * sum(u(2:end) - u(1:end-1) ...
            .* (uPrev(2:end) - uPrev(1:end-1)));

% Total energy (Hamiltonian)
totEnergy(n) = kinEnergy(n) + potEnergy(n);
```

3.4.4 Stability Analysis using Energy Analysis Techniques

Section 3.3 showed how to obtain a stability condition of a FD scheme using a frequency-domain representation. Although not operating in the frequency domain, the energy analysis techniques presented here may also be used to obtain stability conditions of FD schemes. Stability analysis using the energy method might even be considered more powerful than the frequency domain approach, as it can also be used to analyse nonlinear systems! check

To arrive at a stability condition, the energy must be *non-negative* ($\mathfrak{h} \geq 0$) or, in some cases *positive definite* ($\mathfrak{h} > 0$). Below, the mass-spring system and the 1D wave equation will be used as a test case.

Mass-spring system

Section 3.3.1 mentions that the mass-spring system is a special case in that the roots of its characteristic equation can not be identical. When proving stability using energy analysis, this means that the energy of the system needs to be positive definite. It can be shown that an equation of the form

$$x^2 + y^2 + 2axy \tag{3.55}$$

is positive definite if $|a| < 1$.

Equation (3.55) can be used to prove stability for the mass spring system using the energy balance in Eq. (3.48). One can easily conclude that $\mathfrak{t}$ is non-negative due to the fact that $M > 0$ and $(\delta_{t-}u^n)$ is squared. The potential energy $\mathfrak{v}$, however, is of indefinite sign. Expanding the operators in Eq. (3.48) yields

$$\begin{aligned}\mathfrak{h} &= \frac{M}{2k^2}\left((u^n)^2 - 2u^n u^{n-1} + (u^{n-1})^2\right) + \frac{K}{2}u^n u^{n-1},\\ &= \frac{M}{2k^2}\left((u^n)^2 + (u^{n-1})^2\right) + \left(\frac{K}{2} - \frac{M}{k^2}\right)u^n u^{n-1}.\end{aligned}$$

Dividing all terms by $M/2k^2$ this equation is of the form in Eq. (3.55):

$$\mathfrak{h} = (u^n)^2 + (u^{n-1})^2 + \left(\frac{Kk^2}{M} - 2\right)u^n u^{n-1}.$$

For $\mathfrak{h}$ to be positive definite, the following condition must hold

$$\left|\frac{Kk^2}{2M} - 1\right| < 1.$$

This can then be written as

$$\begin{aligned}-1 < \frac{Kk^2}{2M} - 1 < 1\\ 0 < \frac{Kk^2}{2M} < 2\end{aligned}$$

where, as long as K and k are non-zero, the first inequality is always satisfied. Then the condition solved for k can easily be shown to be

$$k < 2\sqrt{\frac{M}{K}} \tag{3.56}$$

which is identical to the definition in Eq. (3.35).

1D wave equation

For the 1D wave equation, the energy must be proven to be non-negative. One can take the energy balance in Eq. (3.53) and conclude that $\mathfrak{t}$ is non-negative due to the non-negativity of the parameters and $(\delta_{t-}u_l^n)$ being squared. The potential energy, however, is of indefinite sign. One can rewrite $\mathfrak{v}$ using identity (3.23e) as

$$\begin{aligned}
\mathfrak{v} &= \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}, \\
&= \frac{T}{2}\sum_{l=0}^{N-1} h(\delta_{x+}u_l^n)(e_{t-}\delta_{x+}u_l^n), \\
&= \frac{T}{2}\sum_{l=0}^{N-1} h\left((\mu_{t-}\delta_{x+}u_l^n)^2 - \frac{k^2}{4}(\delta_{t-}\delta_{x+}u_l^n)^2\right), \\
&= \frac{T}{2}\left(\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2 - \frac{k^2}{4}\|\delta_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2\right).
\end{aligned}$$

One can then use the following bound for spatial differences [3]

$$\|\delta_{x+}u_l^n\|_{\underline{d}} \leq \frac{2}{h}\|u_l^n\|_d' \leq \frac{2}{h}\|u_l^n\|_d, \tag{3.57}$$

to put a condition on $\mathfrak{v}$

$$\begin{aligned}
\mathfrak{v} &\geq \frac{T}{2}\left(\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2 - \frac{k^2}{4}\left(\frac{2}{h}\|\delta_{t-}u_l^n\|_d\right)^2\right), \\
&\geq \frac{T}{2}\left(\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2 - \frac{k^2}{h^2}\|\delta_{t-}u_l^n\|_d^2\right),
\end{aligned}$$

Substituting this condition into the energy balance in Eq. (3.53) yields

$$\begin{aligned}
\mathfrak{h} = \mathfrak{t} + \mathfrak{v} &\geq \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\left(\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2 - \frac{k^2}{h^2}\|\delta_{t-}u_l^n\|_d^2\right), \\
&\geq \left(\frac{\rho A}{2} - \frac{Tk^2}{2h^2}\right)\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2.
\end{aligned}$$

Recalling that $c = \sqrt{T/\rho A}$ and $\lambda = ck/h$, all terms can be divided by ρA which yields

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \geq \frac{1}{2}\left(1 - \lambda^2\right) \|\delta_{t-} u_l^n\|_d^2 + \frac{c^2}{2} \|\mu_{t-}\delta_{x+} u_l^n\|_{\underline{d}}^2, \tag{3.58}$$

and is non-negative for

$$\begin{aligned} 1 - \lambda^2 &\geq 0, \\ \lambda &\leq 1. \end{aligned}$$

This is the same (CFL) condition obtained through von Neumann analysis in Section 3.3.2.

3.5 Modal Analysis

Modes are the resonant frequencies of a system. The amount of modes that a discrete system contains depends on the number of moving points. A mass-spring system thus has one resonating mode, but – as briefly touched upon in Section 2.4.3 – a FD scheme of the 1D wave equation with $N = 30$ and Dirichlet boundary conditions will have 29 modes. This section will show how to numerically obtain the modal frequencies of an FD implementation using the 1D wave equation as a test case.

We start by using the matrix form of the 1D wave equation from Eq. (3.7)

$$\frac{1}{k^2}\left(\mathbf{u}^{n+1} - 2\mathbf{u}^n + \mathbf{u}^{n-1}\right) = c^2 \mathbf{D}_{xx}\mathbf{u}^n.$$

Following [3] we can assume a test solution of the form $\mathbf{u}^n = z^n \boldsymbol{\phi}$. Substituting this into the above equation yields the characteristic equation

more explanation, perhaps refer to von neumann analysis in 3.3

$$(z - 2 + z^{-1})\boldsymbol{\phi} = c^2 k^2 \mathbf{D}_{xx}\boldsymbol{\phi}. \tag{3.59}$$

This is an eigenvalue problem (see Section 3.1.5) where the p^{th} solution $\boldsymbol{\phi}_p$ may be interpreted as the modal shape of mode p. The corresponding modal frequencies are the solutions to the following equations:

$$\begin{aligned} z_p - 2 + z_p^{-1} &= c^2 k^2 \text{eig}_p(\mathbf{D}_{xx}), \\ z_p + \Big(- 2 - c^2 k^2 \text{eig}_p(\mathbf{D}_{xx})\Big) + z_p^{-1} &= 0. \end{aligned} \tag{3.60}$$

Furthermore, we can substitute a test solution $z_p = e^{s_p k}$ with complex frequency $s_p = j\omega_p + \sigma_p$ which contains the (angular) frequency ω_p and damping $\sigma_p \leq 0$ of the p^{th} mode. As there is no damping present in the system, the test solution reduces to $z_p = e^{j\omega_p k}$ which can be substituted into Eq (3.5) to get

check with Stefan

$$\begin{aligned} e^{j\omega_p k} + e^{-j\omega_p k} - 2 - c^2 k^2 \text{eig}_p(\mathbf{D}_{xx}) &= 0, \\ \frac{e^{j\omega_p k} + e^{-j\omega_p k}}{-4} + \frac{1}{2} + \frac{c^2 k^2}{4} \text{eig}_p(\mathbf{D}_{xx}) &= 0. \end{aligned}$$

Finally, using the trigonometric identity in Eq. (3.27a) we get

$$\begin{aligned}\sin^2(\omega_p k/2) + \frac{c^2k^2}{4}\text{eig}_p(\mathbf{D}_{xx}) &= 0,\\ \sin(\omega_p k/2) &= \frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})},\\ \omega_p &= \frac{2}{k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right).\end{aligned} \tag{3.61}$$

and can be rewritten to

$$f_p = \frac{1}{\pi k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right) \tag{3.62}$$

to get the modal frequency of the p^{th} mode in Hz.

3.5.1 One-Step Form

For more complicated systems, specifically those containing damping terms, it is useful to rewrite the update in *one-step form* (also referred to as a state-space representation). The damping terms cause the coefficients of z and z^{-1} in the characteristic equation to not be identical and the trigonometric identities in (3.27) can not be used directly. Although the eigenfrequency calculation needs to be done on a larger matrix, it allows for a more general and direct way to calculate the modal frequencies and damping coefficients per mode.

If matrix $\mathbf{A}$ has an inverse, any scheme of the form

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{3.63}$$

can be rewritten to

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^n\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{A}^{-1}\mathbf{B} & \mathbf{A}^{-1}\mathbf{C}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}}\underbrace{\begin{bmatrix}\mathbf{u}^{n}\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^{n}} \tag{3.64}$$

which relates the unknown state of the system to the known state through matrix $\mathbf{Q}$ which encompases the scheme. The sizes of the identity matrix $\mathbf{I}$ and zero matrix $\mathbf{0}$ are the same size as $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$.

Again, solutions of the form $\mathbf{w}^n = z^n\boldsymbol{\phi}$ can be assumed (where $\boldsymbol{\phi}$ is now less-trivially connected to the modal shapes)

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{3.65}$$

which can be solved for the pth eigenvalue as

$$z_p = \text{eig}_p(\mathbf{Q}). \tag{3.66}$$

As the scheme could exhibit damping, the test solution $z_p = e^{s_p k}$ is used. Substituting this yields

$$\begin{aligned} e^{s_p k} &= \text{eig}_p(\mathbf{Q}), \\ s_p &= \frac{1}{k} \ln\left(\text{eig}_p(\mathbf{Q})\right). \end{aligned} \tag{3.67}$$

Solutions for the frequency and damping for the pth eigenvalue can then be obtained through

$$\omega_p = \Im(s_p) \quad \text{and} \quad \sigma_p = \Re(s_p), \tag{3.68}$$

where $\Im(\cdot)$ and $\Re(\cdot)$ denote the "imaginary part of" and "real part of" respectively.

As the elements of $\mathbf{Q}$ are real-valued, the solutions s_p in Eq. (3.67) come in complex conjugates (pairs of numbers of which the imaginary part has an opposite sign). For analysis, only the $\Im(s_p) \geq 0$ should be considered as these correspond to non-negative frequencies.

3.6 Dispersion analysis

not sure whether to include..

Part II

Resonators

Resonators

Although the physical models described in the previous part – the simple mass-spring system and the 1D wave equation – are also considered resonators, they are *ideal* cases. In other words, you would not be able to find these "in the wild" as they do not include effects such as losses or frequency dispersion.

This part presents the different resonators used over the course of the project that are more true to the physical world and is structured as follows: Chapter 4 introduces the stiff string, an extension of the 1D wave equation and is the single most-used model in this project. Chapter 5 talks about brass instruments, or more generally, 1D systems of varying geometry along their spatial dimension. Finally, Chapter 6 will introduces 2D systems which, in this project, have been used to simulate (simplified) instrument bodies. The analysis techniques introduced in the previous section will be applied to all models and explained in detail.

Chapter 4

Stiff string

In earlier chapters, the case of the ideal string was presented modelled using the 1D wave equation. As shown, if the CFL condition is satisfied with equality, the model generates an output with harmonic partials which are integer multiples of the fundamental frequency. In the real world, however, strings exhibit a phenomenon called *frequency dispersion* due to stiffness in the material, hence the name *stiff string*. This phenomenon causes another effect known as *inharmonicity*: the "harmonic" partials get exponentially further apart the higher their frequency is. The stiffness in a string is dependent on its material properties and geometry and will be elaborated on in this chapter. The stiff string played a prominent part in the following papers: [A], [B], [C], [D] and [E].

This chapter presents the PDE of the stiff string in continuous time, and goes through the discretisation process. The analysis techniques presented in Chapter 3 will then be applied to the resulting FD scheme and derived in detail. Unless denoted otherwise, this chapter follows [3].

4.1 Continuous time

Consider a lossless stiff string of length L and a circular cross-section. Its transverse displacement is described by $u = u(x,t)$ (in m) defined for $x \in \mathcal{D}$ with domain $\mathcal{D} = [0, L]$ and time $t \geq 0$. The PDE describing its motion is

should I even include the lossless one? It's just so that we can slowly build up to the damped model...

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u \tag{4.1}$$

parametrised by material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), tension T (in N), Young's modulus E (in Pa) and area moment of inertia $I = \pi r^4/4$ (in m^4). If $E = 0$, Eq (4.1) reduces to the 1D wave equation in Eq. (2.38) where $c = \sqrt{T/\rho A}$. If instead $T = 0$, Eq. (4.1) reduces

to the *ideal bar* equation. A more compact way to write Eq. (4.1) is

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u \tag{4.2}$$

with wave speed $c = \sqrt{T/\rho A}$ (in m/s) and stiffness coefficient $\kappa = \sqrt{EI/\rho A}$.

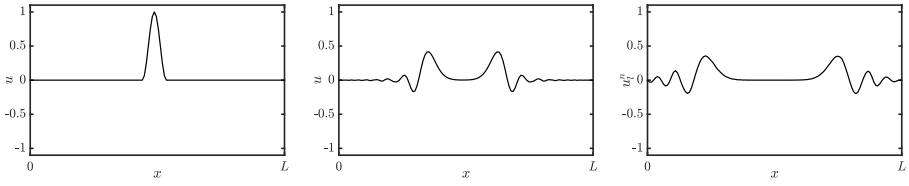
The difference between the ideal string (1D wave equation) and the stiff string is the presence of the 4$^{\text{th}}$-order spatial derivative in the stiffness term which causes frequency dispersion. As opposed to unwanted numerical dispersion due to numerical error (see Section 2.4.4) this type of dispersion is physical and thus something desired in the model. This phenomenon causes higher frequencies to travel faster through a medium than lower frequencies. See Figure 4.1. Furthermore, frequency dispersion is closely tied to *inharmonicity*, an effect where 'harmonic' partials get exponentially further apart as frequency increases. For low values of κ, the frequency of these partials can be expressed in terms of the fundamental frequency $f_0 = c/2L$ (as in Eq. (2.40)) and frequency of partial p (in Hz) is defined as

$$f_p = f_0 p \sqrt{1 + Bp^2}, \tag{4.3}$$

with inharmonicity coefficient

$$B = \frac{\kappa^2 \pi^2}{c^2}.$$

Frequency dispersion and inharmonicity will be further discussed in Section 4.2.3.



(a) $t = 0$ ms. **(b)** $t = 1$ ms. **(c)** $t = 2$ ms.

Fig. 4.1: Dispersion in a stiff string due to stiffness.

4.1.1 Adding Losses

Before moving on to the discretisation of the PDE in Eq. (4.1), losses can be added to the system. In the physical world, strings lose energy through fx. air viscosity and thermoelastic effects. All frequencies lose energy and die out (damp) over time, but higher frequencies do so at a much faster rate. This phenomenon is called *frequency-dependent damping* and can be modelled

using a mixed derivative $\partial_t \partial_x^2$. This way of frequency-dependent damping first appeared in [43] and has been used extensively in the literature since. A damped stiff string can be modelled as

citations here?

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \tag{4.4}$$

where the non-negative loss coefficients σ_0 (in s^{-1}) and σ_1 (in m^2/s) determine the frequency-independent and frequency-dependent losses respectively.

A more compact way to write Eq. (4.4), and as is also found often in the literature [3], is to divide both sides by ρA to get

etc.

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u - 2\sigma_0 \partial_t u + 2\sigma_1 \partial_t \partial_x^2 u \tag{4.5}$$

where $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) as in the 1D wave equation in (2.38) and $\kappa = \sqrt{EI/\rho A}$ is referred to as the stiffness coefficient (in m^2/s).

check wavespeed or wave speed (entire document)

Boundary Conditions

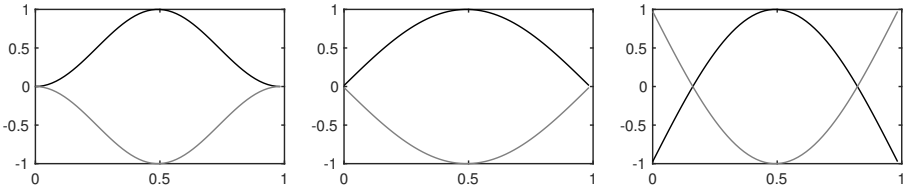
Section 2.4 presents two types of boundary conditions for the 1D wave equation in Eq. (2.39). In the case of the stiff string, these can be extended to

$$u = \partial_x u = 0 \quad \text{(clamped)} \tag{4.6a}$$

$$u = \partial_x^2 u = 0 \quad \text{(simply supported)} \tag{4.6b}$$

$$\partial_x^2 u = \partial_x^3 u = 0 \quad \text{(free)} \tag{4.6c}$$

at $x = 0, L$. See Figure 4.2 for plots of the first modal shape for each respective boundary condition.



(a) Clamped. **(b)** Simply supported. **(c)** Free.

Fig. 4.2: Plots of the first (normalised) modal shape for the three boundary conditions in Eqs. (4.6). The extremes are indicated with black and grey lines respectively.

4.2 Discrete Time

For the sake of compactness, we continue with Eq. (4.5), rather than Eq. (4.4). Naturally, same process can be followed for the latter, the only difference being a multiplication by ρA of all terms.

Equation (4.5) can be discretised as

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n, \tag{4.7}$$

and is defined for domain $l \in \{0, \ldots, N\}$ and number of grid points $N+1$. The δ_{xxxx} operator is defined as the second-order spatial difference in Eq. (2.8) applied to itself:

$$\delta_{xxxx} = \delta_{xx}\delta_{xx} = \frac{1}{h^4}\left(e_{x+}^2 - 4e_{x+} + 6 - 4e_{x-} + e_{x-}^2\right). \tag{4.8}$$

A multiplication of two shift operators applied to a grid function simply means to apply each shift individually. The δ_{xxxx} operator applied to u_l^n thus becomes

$$\delta_{xxxx}u_l^n = \frac{1}{h^4}\left(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\right). \tag{4.9}$$

A definition for the mixed-derivative operator can similarly be found. Recalling the definitions for δ_{t-} in Eq. (2.3b) and δ_{xx} Eq. (2.8), their combination results in

$$\begin{aligned}\delta_{t-}\delta_{xx} &= \frac{1}{k}\left(1 - e_{t-}\right)\frac{1}{h^2}\left(e_{x+} - 2 + e_{x-}\right), \\ &= \frac{1}{kh^2}\left(e_{x+} - 2 + e_{x-} - e_{t-}(e_{x+} - 2 + e_{x-})\right).\end{aligned} \tag{4.10}$$

Two different shift operators multiplied together still simply means to apply each of them to the grid function individually. The $\delta_{t-}\delta_{xx}$ operator applied to u_l^n thus yields

$$\delta_{t-}\delta_{xx}u_l^n = \frac{1}{hk^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}\right). \tag{4.11}$$

The reason a backwards difference is used here is to keep the system explicit. An example of an implicit scheme using the centred operator instead can be found in Section 4.6.

With these definitions, the operators in scheme (4.7) can be expanded to get

$$\begin{aligned}\frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) = \frac{c^2}{h^2}&\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \\ &- \frac{\kappa^2}{h^4}\left(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\right) \\ &- \frac{\sigma_0}{k}\left(u_l^{n+1} - u_l^{n-1}\right) \\ &+ \frac{2\sigma_1}{kh^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} + u_{l-1}^{n-1}\right),\end{aligned} \tag{4.12}$$

and after multiplication by k^2 and collecting the terms yields

$$\begin{aligned}(1+\sigma_0 k)u_l^{n+1} = &\left(2-2\lambda^2-6\mu^2-\frac{4\sigma_1 k}{h^2}\right)u_l^n \\ &+\left(\lambda^2+4\mu^2+\frac{2\sigma_1 k}{h^2}\right)(u_{l+1}^n+u_{l-1}^n) \\ &-\mu^2(u_{l+2}^n+u_{l-2}^n)+\left(-1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\right)u_l^{n-1} \\ &-\frac{2\sigma_1 k}{h^2}(u_{l+1}^{n-1}+u_{l-1}^{n-1}),\end{aligned} \tag{4.13}$$

with

$$\lambda = \frac{ck}{h} \quad \text{and} \quad \mu = \frac{\kappa k}{h^2}. \tag{4.14}$$

The update equation follows by dividing both sides by $(1+\sigma_0 k)$.

The stability condition for the FD scheme in (4.7) is defined as

$$h \geq \sqrt{\frac{c^2k^2+4\sigma_1 k+\sqrt{(c^2k^2+4\sigma_1 k)^2+16\kappa^2k^2}}{2}}, \tag{4.15}$$

and will be derived in Section 4.3 using von Neumann analysis. This condition can then be used to calculate the number of intervals N in a similar fashion as for the 1D wave equation shown in Eq. (2.53). First, Eq. (4.15) should be satisfied with equality, after which

$$N := \left\lfloor \frac{L}{h} \right\rfloor, \quad \text{and} \quad h := \frac{L}{N}$$

which can then be used to calculate λ and μ in (4.14).

Stencil

As done in Section 2.4.2, a stencil for the FD scheme implementing the damped stiff string can be created. This is shown in Figure 4.3. In order to calculate u_l^{n+1}, 5 points at the current time step are needed due to the 4$^{\text{th}}$-order spatial derivative. Due to the mixed derivative in the frequency-dependent damping term neighbouring points at the previous time step are also required.

4.2.1 Boundary conditions

Due to the 4$^{\text{th}}$-order spatial derivative, two virtual grid points need to be accounted for at the boundaries of the system. Discretising the boundary

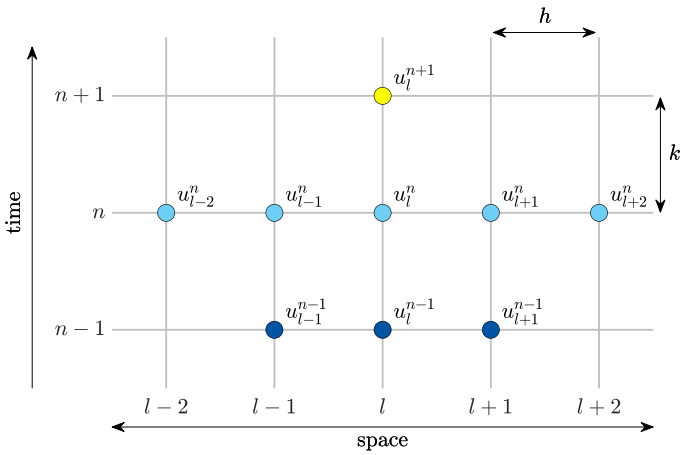


Fig. 4.3: The stencil for the damped stiff string scheme in Eq. (4.7).

conditions in (4.6) yields

$$u_l^n = \delta_{x\pm} u_l^n = 0 \quad \text{(clamped)} \tag{4.16a}$$

$$u_l^n = \delta_{xx} u_l^n = 0 \quad \text{(simply supported)} \tag{4.16b}$$

$$\delta_{xx} u_l^n = \delta_{x.} \delta_{xx} u_l^n = 0 \quad \text{(free)} \tag{4.16c}$$

at $l = 0, N$. The operator in the clamped condition uses the δ_{x+} operator at the left boundary ($l = 0$) and δ_{x-} at the right ($l = N$). Note that for the free boundary condition in Eq. (4.6c), to discretise ∂_x^3 the more accurate $\delta_{x.}\delta_{xx}$ operator has been chosen over the less accurate $\delta_{x-}\delta_{xx}$ and $\delta_{x+}\delta_{xx}$ operators for the left and right boundary respectively.

Below, the boundary conditions are expanded to get definitions for the virtual grid points.

insert figure showing virtual grid points somewhere in this section

Clamped

Expanding the operators for the clamped condition yields

$$u_0^n = u_1^n = 0 \quad \text{and} \quad u_{N-1}^n = u_N^n = 0. \tag{4.17}$$

This can be simplified by reducing the range of calculation to $l \in \{2, \ldots, N-2\}$.

Simply supported

As the states of the end points of a system with simply supported boundary conditions are 0 at all times, the range of calculation can be reduced to $l \in$

$\{1, \ldots, N-1\}$. At $l = 1$ and $l = N-1$, definitions for the virtual grid points u_{-1}^n and u_{N+1}^n are needed. A definition for u_{-1}^n can be found by expanding Eq. (4.16b) at $l = 0$:

$$\begin{aligned} & \frac{1}{h^2}\left(u_1^n - 2u_0^n + u_{-1}^n\right) = 0, \\ \overset{u_0^n=0}{\Longleftrightarrow} \quad & u_1^n + u_{-1}^n = 0, \\ & u_{-1}^n = -u_1^n, \end{aligned} \tag{4.18}$$

and similarly for u_{N+1}^n by expanding the condition at $l = N$:

$$u_{N+1}^n = -u_{N-1}^n.$$

Filling the first definition into the expanded scheme at $l = 1$ in (4.12) and collecting the terms yields

$$\begin{aligned} (1+\sigma_0 k)u_1^{n+1} = & \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2}\right) u_1^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right) u_2^n \\ & - \mu^2 u_3^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2}\right) u_1^{n-1} - \frac{2\sigma_1 k}{h^2} u_2^{n-1}. \end{aligned} \tag{4.19}$$

Doing the same for $l = N-1$, we get

$$\begin{aligned} (1+\sigma_0 k)u_{N-1}^{n+1} = & \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2}\right) u_{N-1}^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right) u_{N-2}^n \\ & - \mu^2 u_{N-3}^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2}\right) u_{N-1}^{n-1} - \frac{2\sigma_1 k}{h^2} u_{N-2}^{n-1}. \end{aligned} \tag{4.20}$$

Free

Finally, the free boundary condition requires all points to be calculated and the range of calculation is $l \in \{0, \ldots, N\}$. At each respective boundary, two virtual grid points are needed: u_{-1}^n and u_{-2}^n at the left and u_{N+1}^n and u_{N+2}^n at the right boundary respectively. The combined operator in Eq. (4.16c) is defined as:

$$\begin{aligned} \delta_x.\delta_{xx} &= \frac{1}{2h^3}\left(e_{x+} - e_{x-}\right)\left(e_{x+} - 2 + e_{x-}\right), \\ &= \frac{1}{2h^3}\left(e_{x+}^2 - 2e_{x+} + 1 - \left(1 - 2e_{x-} + e_{x-}^2\right)\right), \\ &= \frac{1}{2h^3}\left(e_{x+}^2 - 2e_{x+} + 2e_{x-} - e_{x-}^2\right), \end{aligned} \tag{4.21}$$

and can be used to solve for u_{-2}^n at $l = 0$:

$$\begin{aligned} & \frac{1}{2h^3}\left(u_2^n - 2u_1^n + 2u_{-1}^n - u_{-2}^n\right) = 0, \\ & u_{-2}^n = u_2^n - 2u_1^n + 2u_{-1}^n. \end{aligned}$$

As u_0^n is not necessarily 0 at all times, solving the first part of the boundary condition yields a different result than in the simply supported case:

$$\begin{aligned}\frac{1}{h^2}\left(u_1^n - 2u_0^n + u_{-1}^n\right) &= 0,\\ u_{-1}^n &= 2u_0^n - u_1^n.\end{aligned}$$

The same can be done at $l = N$ to get the following definitions for the virtual grid points

$$u_{N+2}^n = u_{N-2}^n - 2u_{N-1}^n + 2u_{N+1}^n \quad \text{and} \quad u_{N+1}^n = 2u_N^n - u_{N-1}^n.$$

The update equations for the boundary points will not be given here. Instead the matrix form of the FD scheme with free boundaries will be provided below.

In practice, the simply supported boundary condition is mostly chosen as this reflects reality the most. The clamped condition could be chosen for simplicity as this does not require an alternative update at the boundaries. The free boundary condition is more often used to model a (damped) ideal bar, (Eq. (4.4) with $T = 0$).

4.2.2 Implementation and Matrix Form

When using `MATLAB`, for a more compact implementation, it is useful to write the scheme in matrix form. The FD scheme of the stiff string in (4.7) can be written as

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{4.22}$$

where

$$A = (1 + \sigma_0 k), \quad \mathbf{B} = c^2k^2\mathbf{D}_{xx} - \kappa^2k^2\mathbf{D}_{xxxx} + 2\sigma_1 k\mathbf{D}_{xx}$$
$$\text{and} \quad \mathbf{C} = -(1 - \sigma_0 k)\mathbf{I} - 2\sigma_1 k\mathbf{D}_{xx}.$$

Notice that A is a scalar rather than a matrix.

The size of the state vectors and the matrix-form operators depend on the boundary conditions. For clamped conditions, the state vectors ($\mathbf{u}^{n+1}$, $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$) and matrices will be of size $(N-3)\times 1$ and $(N-3)\times(N-3)$ respectively. The $\mathbf{D}_{xx}$ matrix will be of the form given in (3.5) and the matrix form of the δ_{xxxx} operator is

$$\mathbf{D}_{xxxx} = \frac{1}{h^4}\begin{bmatrix} 6 & -4 & 1 & & \mathbf{0} \\ -4 & 6 & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & 6 & -4 \\ \mathbf{0} & & 1 & -4 & 6 \end{bmatrix}. \tag{4.23}$$

For simply supported conditions, the state vectors and matrices will be of size $(N-1)\times 1$ and $(N-1)\times(N-1)$ respectively. Again, $\mathbf{D}_{xx}$ is as defined in (3.5) and $\mathbf{D}_{xxxx}$ can be obtained by multiplying two $\mathbf{D}_{xx}$ matrices according to

$$\mathbf{D}_{xxxx} = \mathbf{D}_{xx}\mathbf{D}_{xx} = \frac{1}{h^4}\begin{bmatrix} 5 & -4 & 1 & & & \mathbf{0} & \\ -4 & 6 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & -4 & 1 & & \\ & \ddots & -4 & 6 & -4 & \ddots & \\ & & 1 & -4 & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & 6 & -4 \\ & \mathbf{0} & & & 1 & -4 & 5 \end{bmatrix}. \tag{4.24}$$

Finally for free boundary conditions as given in (4.16c), the state vectors and matrices are $(N+1)\times 1$ and $(N+1)\times(N+1)$ respectively. Now, the $\mathbf{D}_{xx}$ matrix is of the form in (3.6) instead, and

$$\mathbf{D}_{xxxx} = \frac{1}{h^4}\begin{bmatrix} 2 & -4 & 2 & & & \mathbf{0} & \\ -2 & 5 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 5 & -2 \\ & \mathbf{0} & & & 2 & -4 & 2 \end{bmatrix}. \tag{4.25}$$

4.2.3 Parameters and output

The values of the parameters naturally determine the properties of the output sound. Where in the 1D wave equation, only the fundamental frequency f_0 could be affected, the stiff string has much more aspects that can be changed. See Table 4.1 for parameters most commonly used in this project. There exists a formula to calculate the loss coefficients σ_0 and σ_1 from T_{60} values at different frequencies (see [3, Eq (7.29)]). During this project, however, these values have been tuned by ear and are usually set to be approximately those found in Table 4.1.

Output

Figure 4.4 shows the time-domain and frequency-domain output (retrieved at $l = 3$) of an implementation of the stiff string excited using a raised-cosine. The parameters used can be found in Table 4.1 with $T = 1129$. Furthermore,

Name	Symbol (unit)	Value
Length	L (m)	1
Material density	ρ (kg/m^3)	7850
Radius	r (m)	$5 \cdot 10^{-4}$
Tension	T (N)	$100 \leq T \leq 1.5 \cdot 10^4$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq.-independent damping	σ_0 (s^{-1})	1
Freq.-dependent damping	σ_1 (m^2/s)	0.005

Table 4.1: Parameters and their values most commonly used over the course of this project.

$E = 7 \cdot 10^{11}$ to highlight dispersive effects. Finally, simply supported boundary conditions are chosen. From the left panel, one can observe that over time, dispersive effects show where higher-frequency components in the excitation travel faster through the medium than lower-frequency components. In the frequency domain (the right panel in Figure 4.4) this shows in the partials not being perfect integer multiples of the fundamental. Notice that the partials are closer to each other for lower frequencies and further apart as their frequency increases. Finally, the frequency-dependent damping term causes higher frequencies to have a lower amplitude than lower frequencies.

Apart from the obvious material properties such as density, stiffness and geometry, perceptual qualities of the sound are surprisingly much determined by σ_1, and for lower values the output can become extremely metallic.

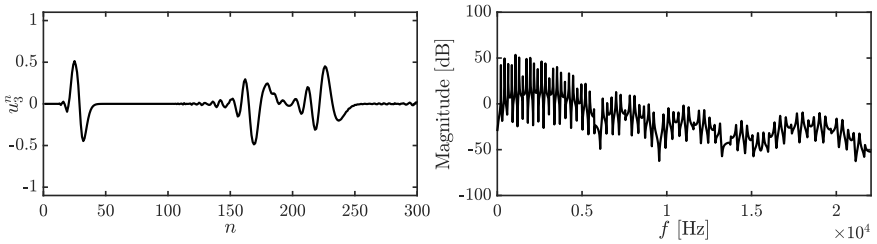


Fig. 4.4: The time-domain and frequency-domain output of the stiff string. The parameters are set as in Table 4.1 with $E = 7 \cdot 10^{11}$ to highlight dispersive effects and $T = 1129$. From the left panel, one can observe that over time, dispersive effects show due to the stiffness in the string. In frequency domain (right panel) this shows through the partials not being perfect integer multiples of the fundamental. Lastly, higher frequency components have a lower amplitude due to the frequency-dependent damping.

4.3 von Neumann Analysis and Stability Condition

In order to obtain the stability condition for the damped stiff string, one can perform von Neumann analysis as presented in Section 3.3 on the FD scheme in Eq. (4.7).

Using the definitions found in Eq. (3.28) for the temporal operators and Eqs. (3.29a) and (3.29b) for the spatial operators, the frequency-domain representation of Eq. (4.7) can be obtained:

$$\begin{aligned}\frac{1}{k^2}\left(z-2+z^{-1}\right) = &-\frac{4c^2}{h^2}\sin^2(\beta h/2) - \frac{16\kappa^2}{h^4}\sin^4(\beta h/2) - \frac{\sigma_0}{k}z + \frac{\sigma_0}{k}z^{-1} \\ &- \frac{8\sigma_1}{kh^2}\sin^2(\beta h/2) + \frac{8\sigma_1}{kh^2}\sin^2(\beta h/2)z^{-1},\end{aligned}$$

and after collecting the terms, the characteristic equation follows:

$$\begin{aligned}(1+\sigma_0 k)z + \left(16\mu^2\sin^4(\beta h/2) + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\sin^2(\beta h/2) - 2\right) \\ + \left(1-\sigma_0 k - \frac{8\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z^{-1} = 0.\end{aligned} \tag{4.26}$$

Rewriting this to the form in Eq. (3.31), and using $\mathcal{S} = \sin^2(\beta h/2)$ for brevity, yields

$$z^2 + \left(\frac{16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\mathcal{S} - 2}{1+\sigma_0 k}\right)z + \frac{1-\sigma_0 k - \frac{8\sigma_1 k}{h^2}\mathcal{S}}{1+\sigma_0 k} = 0.$$

Stability of the system can then be proven using condition (3.32) and substituting the coefficients into this condition yields

$$\begin{aligned}\left|\frac{16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\mathcal{S} - 2}{1+\sigma_0 k}\right| - 1 \leq \frac{1-\sigma_0 k - \frac{8\sigma_1 k}{h^2}\mathcal{S}}{1+\sigma_0 k} \leq 1, \\ \left|16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\mathcal{S} - 2\right| - (1+\sigma_0 k) \leq 1-\sigma_0 k - \frac{8\sigma_1 k}{h^2}\mathcal{S} \leq 1+\sigma_0 k, \\ \left|16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\mathcal{S} - 2\right| \leq 2 - \frac{8\sigma_1 k}{h^2}\mathcal{S} \leq 2 + 2\sigma_0 k.\end{aligned}$$

The second condition is always true due to the fact that $\sigma_0, \sigma_1 \geq 0$. Continuing with the first condition:

$$\begin{aligned}-2 + \frac{8\sigma_1 k}{h^2}\mathcal{S} \leq 16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\mathcal{S} - 2 \leq 2 - \frac{8\sigma_1 k}{h^2}\mathcal{S}, \\ 0 \leq 16\mu^2\mathcal{S}^2 + 4\lambda^2\mathcal{S} \leq 4 - \frac{16\sigma_1 k}{h^2}\mathcal{S}.\end{aligned}$$

As $16\mu^2\mathcal{S}^2 + 4\lambda^2\mathcal{S}$ is non-negative, the first condition is always satisfied. Continuing with the second condition:

$$16\mu^2\mathcal{S}^2 + \left(4\lambda^2 + \frac{16\sigma_1 k}{h^2}\right)\mathcal{S} \leq 4,$$
$$4\mu^2\mathcal{S}^2 + \left(\lambda^2 + \frac{4\sigma_1 k}{h^2}\right)\mathcal{S} \leq 1.$$

different wording

As $\mathcal{S}$ is bounded by 1, this can be substituted as it challenges the condition the most. Continuing with the substituted definitions for λ and μ from Eq. (4.14) yields

$$\frac{4\kappa^2 k^2}{h^4} + \frac{c^2k^2 + 4\sigma_1 k}{h^2} \leq 1,$$
$$4\kappa^2 k^2 + (c^2k^2 + 4\sigma_1 k)h^2 \leq h^4,$$
$$h^4 - (c^2k^2 + 4\sigma_1 k)h^2 - 4\kappa^2 k^2 \geq 0,$$

which is a quadratic equation in h^2. The grid spacing h is then bounded by

$$h \geq \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}, \tag{4.27}$$

which is the stability condition for the damped stiff string also shown in Eq. (4.15).

4.4 Energy Analysis

As mentioned in Section 3.4, it is useful to perform the energy analysis on the scheme with all physical parameters written out. Discretising the PDE in (4.4) yields

$$\rho A \delta_{tt} u_l^n = T\delta_{xx}u_l^n - EI\delta_{xxxx}u_l^n - 2\sigma_0\rho A\delta_{t\cdot}u_l^n + 2\sigma_1\rho A\delta_{t-}\delta_{xx}u_l^n, \tag{4.28}$$

defined for $l \in d$ with discrete domain $d = \{0, \ldots, N\}$. This section will follow the 4 steps described in Section 3.4.

Step 1: Obtain $\delta_{t+}\mathfrak{h}$

The first step is to take the inner product (see Eq. (3.15)) of the scheme with $(\delta_{t\cdot}u_l^n)$ over discrete domain d:

$$\begin{aligned}\delta_{t+}\mathfrak{h} = \rho A\langle\delta_{t\cdot}u_l^n, \delta_{tt}u_l^n\rangle_d - T\langle\delta_{t\cdot}u_l^n, \delta_{xx}u_l^n\rangle_d + EI\langle\delta_{t\cdot}u_l^n, \delta_{xxxx}u_l^n\rangle_d \\ + 2\sigma_0\rho A\langle\delta_{t\cdot}u_l^n, \delta_{t\cdot}u_l^n\rangle_d - 2\sigma_1\rho A\langle\delta_{t\cdot}u_l^n, \delta_{t-}\delta_{xx}u_l^n\rangle_d = 0.\end{aligned} \tag{4.29}$$

Step 2: Identify energy types and isolate δ_{t+}

As there is damping present in the system, and the system is distributed, the damping term $\mathfrak{q}$ and boundary term $\mathfrak{b}$ appear and the energy balance will be of the form

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q}. \tag{4.30}$$

The damping term is defined as

$$\mathfrak{q} = 2\sigma_0\rho A\|\delta_{t\cdot}u_l^n\|_d^2 - 2\sigma_1\rho A\langle\delta_{t\cdot}u_l^n, \delta_{t-}\delta_{xx}u_l^n\rangle_d, \tag{4.31}$$

and the boundary term $\mathfrak{b}$ appears after rewriting Equation (4.29) using summation by parts (see Section 3.2.2). Specifically, using Eq. (3.20a) for the second term and Eq. (3.22b) for the third, we get

$$\begin{aligned}\delta_{t+}\mathfrak{h} &= \rho A\langle\delta_{t\cdot}u_l^n, \delta_{tt}u_l^n\rangle_d + T\langle\delta_{t\cdot}\delta_{x+}u_l^n, \delta_{x+}u_l^n\rangle_{\underline{d}} + EI\langle\delta_{t\cdot}\delta_{xx}u_l^n, \delta_{xx}u_l^n\rangle_{\underline{\overline{d}}} \\ &= \mathfrak{b} - \mathfrak{q}\end{aligned}$$

where the boundary term becomes

$$\begin{aligned}\mathfrak{b} = T\Big((\delta_{t\cdot}u_N^n)(\delta_{x+}u_N^n) - (\delta_{t\cdot}u_0^n)(\delta_{x+}u_{-1}^n)\Big) \\ + EI\Big((\delta_{t\cdot}u_N^n)(\delta_{x+}\delta_{xx}u_N^n) - (\delta_{xx}u_N^n)(\delta_{x-}\delta_{t\cdot}u_N^n)\Big) \\ + EI\Big(-(\delta_{t\cdot}u_0^n)(\delta_{x-}\delta_{xx}u_0^n) + (\delta_{xx}u_0^n)(\delta_{x+}\delta_{t\cdot}u_0^n)\Big).\end{aligned}$$

For the clamped and simply supported boundary conditions in (4.16a) and (4.16b) it can easily be shown that $\mathfrak{b} = 0$. If free conditions as in Eq. (4.16c) are used, the boundary conditions will vanish when the primed inner product in Eq. (3.16) is used in Step 1 and identity (3.22c) is used when performing summation by parts. Here, we continue with the clamped / simply supported case.

Isolating δ_{t+} to obtain the total energy $\mathfrak{h}$ in the definition for $\delta_{t+}\mathfrak{h}$ above, requires identities (3.23a) and (3.23b) and yields

$$\begin{aligned}\delta_{t+}\mathfrak{h} &= \delta_{t+}\left(\frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}} + \frac{EI}{2}\langle\delta_{xx}u_l^n, e_{t-}\delta_{xx}u_l^n\rangle_{\underline{\overline{d}}}\right) \\ &= -\mathfrak{q}.\end{aligned}$$

From this, the definition for the Hamiltonian $\mathfrak{h}$, the kinetic energy $\mathfrak{t}$ and potential energy $\mathfrak{v}$ can be found:

$$\begin{aligned}\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \text{and} \\ \mathfrak{v} = \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}} + \frac{EI}{2}\langle\delta_{xx}u_l^n, e_{t-}\delta_{xx}u_l^n\rangle_{\underline{\overline{d}}}.\end{aligned} \tag{4.32}$$

Step 3: Check units

Comparing the acquired energy balance in Eq. (4.32) to the energy balance for the 1D wave equation in Eq. (3.53), one can observe that the balances are nearly identical, the only difference being the second term in the definition for $\mathfrak{v}$ in Eq. (4.32). Writing this term out in units, and recalling that Pa (the unit for E) in SI units is $\mathrm{kg}{\cdot}\mathrm{m}^{-1}{\cdot}\mathrm{s}^{-2}$, yields

$$\frac{EI}{2}\langle \delta_{xx} u_l^n, e_{t-}\delta_{xx} u_l^n \rangle_{\underline{d}} \xrightarrow{\text{in units}} \mathrm{Pa}\cdot\mathrm{m}^4\cdot\mathrm{m}\cdot(\mathrm{m}^{-2}\cdot\mathrm{m}\cdot\mathrm{m}^{-2}\cdot\mathrm{m}) = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},$$

and indeed has the correct units.

The damping terms in $\mathfrak{q}$ need to be in Joules per second, or $\mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3}$. Writing the terms in Eq. (4.31) out in their units yields

$$2\sigma_0\rho A\|\delta_{t\cdot} u_l^n\|_d^2 \xrightarrow{\text{in units}} \mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2\cdot\mathrm{m}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})^2 = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},$$

$$-2\sigma_1\rho A\langle \delta_{t\cdot} u_l^n, \delta_{t-}\delta_{xx} u_l^n \rangle_d \xrightarrow{\text{in units}} \mathrm{m}^2\cdot\mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2 \cdot\mathrm{m}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})(\mathrm{s}^{-1}\cdot\mathrm{m}^{-2}\cdot\mathrm{m}), = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},$$

and also have the correct units.

Step 4: Implementation

Add to appendix or refer to a gist

An implementation of the energy for the stiff string is given in Appendix **??** . Figure 4.5 shows that the deviation of the total energy calculated using Eq. (3.44) is within machine precision.

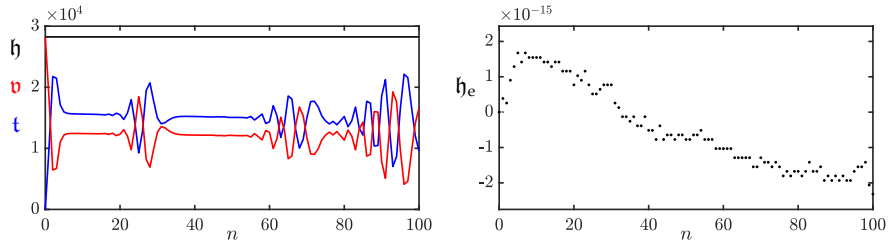


Fig. 4.5: The kinetic (blue), potential (red), and total (black) energy of an implementation of the stiff string are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.44)) and shows that the deviation of the energy is within machine precision.

4.5 Modal Analysis

To be able to perform a modal analysis on the FD scheme in (4.7), it must be written in one-step form – introduced in Section 3.5.1 – due to the damping. Using the matrix form of the damped stiff string in Eq. (4.22), the one-step form can be written as

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^{n}\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{B}/A & \mathbf{C}/A\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix}\mathbf{u}^{n}\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^{n}} \tag{4.33}$$

where the definitions for $\mathbf{B}$, $\mathbf{C}$ and A can be found in Section 4.2.2. The definitions for $\mathbf{D}_{xx}$ and $\mathbf{D}_{xxxx}$ those for simply supported boundary conditions as these are used most often in the case of strings.

Assuming test solutions of the form $\mathbf{w}^n = z^n\boldsymbol{\phi}$, and recalling that $z = e^{sk}$ and complex frequency $s = j\omega + \sigma$, we get the following eigenvalue problem (see Section 3.1.5)

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{4.34}$$

which has the following solutions

$$s_p = \frac{1}{k}\ln\left(\mathrm{eig}_p(\mathbf{Q})\right). \tag{4.35}$$

The (angular) frequency of the p^{th} mode can then be obtained using $\Im(s_p)$ and the damping per mode as $\Re(s_p)$. Only selecting the non-negative frequencies obtained from $\Im(s_p)$, these can be plotted and are shown in Figure 4.6. The parameters used are the ones found in Table 4.1 with $T = 1885$ N, and $E = 2 \cdot 10^{14}$ to highlight inharmonic behaviour. The left panel shows that the system is indeed inharmonic, i.e., modal frequencies increase more as the modal number increases. The right panel shows that higher modes exhibit a higher amount of damping. This is due to the frequency-dependent damping term. If $\sigma_1 = 0$ in (4.7), it can be shown that $\sigma_p = \sigma_0$ for every mode p (in this case -1).

4.6 Implicit Scheme

Although not used in the published work of this project, it is useful to touch upon an example of an implicit scheme. Consider a discretisation of Eq. (4.5) where the (more accurate) centred operator is used for the frequency-dependent damping term:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t\cdot}\delta_{xx}u_l^n. \tag{4.36}$$

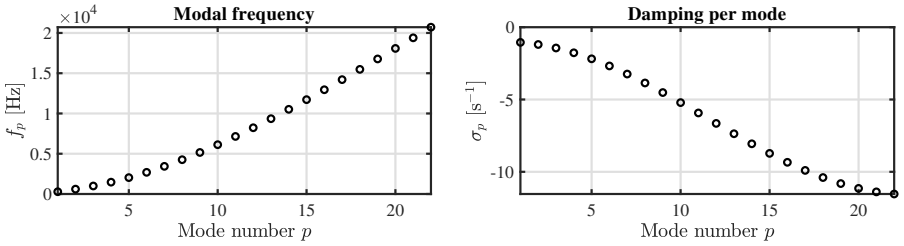


Fig. 4.6: The modal frequencies and damping per mode for the stiff string using the values in 4.1 and $T = 1885$ and $E = 2 \cdot 10^{14}$ to highlight effects of stiffness. Notice form the left panel that the frequency increases exponentially with the mode number. The right panel shows that higher modes exhibit a greater amount of damping due to the frequency-dependent damping term.

Using the centred operator in the mixed-spatial-temporal operator renders the system *implicit*, meaning that a definition for u_l^{n+1} can not explicitly be found from known values. The stencil in Figure 4.7 also shows this: in order to calculate u_l^{n+1}, neighbouring points at the next time step u_{l+1}^{n+1} and u_{l-1}^{n+1} are needed. The issue is that these values are unknown at the time of calculation.

Luckily, as the scheme is linear, it can be treated as a system of linear equations and solved following the technique described in Section 3.1.4. The drawback is that this requires one matrix inversion per iteration which can be extremely costly (see Section **??**). However, both von Neumann and modal analysis (below) show that using the centred instead of the backwards operator has a positive effect on the stability and the modal behaviour of the scheme.

Taking simply supported boundary conditions such that $l \in \{1, \dots, N-1\}$, the sytem will have $N-1$ unknowns (u_l^{n+1} for $l \in \{1, \dots, N-1\}$) that can be calculated using $N-1$ (updat) equations. Writing this in matrix form using column vector $\mathbf{u}^n = [u_1^n, u_2^n, \dots, u_{N-1}^n]$ yields

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{4.37}$$

where

$$\mathbf{A} = (1 + \sigma_0 k)\mathbf{I} - \sigma_1 k \mathbf{D}_{xx}, \quad \mathbf{B} = c^2 k^2 \mathbf{D}_{xx} - \kappa^2 k^2 \mathbf{D}_{xxxx}$$
$$\text{and} \quad \mathbf{C} = -(1 - \sigma_0 k)\mathbf{I} - \sigma_1 k \mathbf{D}_{xx}.$$

4.6.1 von Neumann analysis

Using the same process as in Section 4.3, the definitions in Section 3.3 can be used to obtain a frequency-domain representation of the FD scheme in Eq.

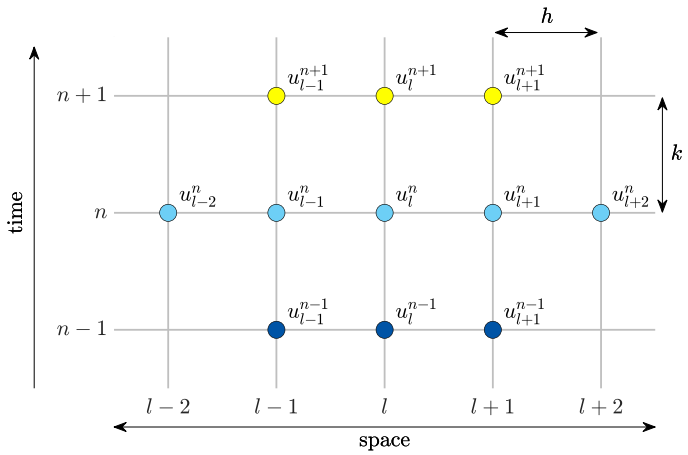


Fig. 4.7: The stencil for the damped stiff string scheme in (4.36).

(4.36) can be obtained:

$$\frac{1}{k^2}\left(z-2+z^{-1}\right) = -\frac{4c^2}{h^2}\sin^2(\beta h/2) - \frac{16\kappa^2}{h^4}\sin^4(\beta h/2) - \frac{\sigma_0}{k}z + \frac{\sigma_0}{k}z^{-1} - \frac{4\sigma_1}{kh^2}\sin^2(\beta h/2)z + \frac{4\sigma_1}{kh^2}\sin^2(\beta h/2)z^{-1}, \tag{4.38}$$

and collecting the terms, yields the following characteristic equation:

$$\left(1+\sigma_0 k + \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z + \left(16\mu^2\sin^4(\beta h/2) + 4\lambda^2\sin^2(\beta h/2) - 2\right) + \left(1-\sigma_0 k - \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z^{-1} = 0. \tag{4.39}$$

Rewriting this to the form found in Eq. (3.31) and, again, using $\mathcal{S} = \sin^2(\beta h/2)$ yields:

$$z^2 + \frac{16\mu^2\mathcal{S}^2 + 4\lambda^2\mathcal{S} - 2}{1+\sigma_0 k + \frac{4\sigma_1 k}{h^2}\mathcal{S}}z + \frac{1-\sigma_0 k - \frac{4\sigma_1 k}{h^2}\mathcal{S}}{1+\sigma_0 k + \frac{4\sigma_1 k}{h^2}\mathcal{S}} = 0.$$

Stability of the system can then be proven using condition (3.32), and after substitution of the coefficients yields

$$\left|\frac{16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}-2}{1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\mathcal{S}}\right|-1\leq\frac{1-\sigma_0 k-\frac{4\sigma_1 k}{h^2}\mathcal{S}}{1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\mathcal{S}}\leq 1,$$

$$\begin{aligned}\left|16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}-2\right|-\left(1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\mathcal{S}\right)&\leq 1-\sigma_0 k-\frac{4\sigma_1 k}{h^2}\mathcal{S}\\&\leq 1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\mathcal{S},\end{aligned}$$

$$\left|16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}-2\right|\leq 2\leq 2+2\sigma_0 k+\frac{8\sigma_1 k}{h^2}\mathcal{S}.$$

Because $\sigma_0, \sigma_1, k, \mathcal{S}$ and h are all non-negative, the last condition is always satisfied. Continuing with the first condition:

$$\begin{aligned}-2\leq 16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}-2\leq 2,\\0\leq 16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}\leq 4.\end{aligned}$$

Again, the first condition is always satisfied due to the non-negativity of all coefficients. Continuing with the second condition

$$4\mu^2\mathcal{S}^2+\lambda^2\mathcal{S}\leq 1,$$

and knowing that $\mathcal{S}$ is bounded by 1 for all β, the process can be finalised:

$$\begin{aligned}4\mu^2+\lambda^2&\leq 1,\\\frac{4\kappa^2k^2}{h^4}+\frac{c^2k^2}{h^2}&\leq 1,\\h^4-c^2k^2h^2-4\kappa^2k^2&\geq 0,\end{aligned}$$

and yields the following stability condition:

$$h\geq\sqrt{\frac{c^2k^2+\sqrt{c^4k^4+16\kappa^2k^2}}{2}}. \tag{4.40}$$

Comparing this to the stability condition for the explicit scheme in Eq. (4.15), one can observe that the terms containing σ_1 have vanished. It can thus be concluded that if the centred (rather than the backwards) difference is used to discretise the temporal derivative in the frequency-dependent damping term, σ_1 no longer influences the stability of the scheme. What this means in terms of behaviour of the scheme will be elaborated on in the following section.

4.6.2 Modal analysis

As the matrix form of the implicit FD scheme in Eq. (4.37) matches the form in Eq. (3.63), one can perform a modal analysis by writing the scheme in one-step form as explained in Section 3.5.1. The results of the analysis are shown in Figure 4.8. To highlight the difference between using the backwards and centred difference for the frequency-dependent damping term, σ_1 has been set to 1.

One can observe that higher values of σ_1 start to influence the frequencies of especially higher-frequency modes in the explicit scheme. Looking at the expression

As σ_1 increases, h increases due to Eq. (4.15), causing λ^2 and μ^2 to decrease

The modal frequencies

As the values for λ and μ

the best approximation to the frequencies of these partials is obtained when $\sigma_1 = 0$

More specifically, higher modes decrease in frequency due to the numerical dispersion introduced by the inaccuracy of the backwards time difference operator in the frequency-dependent damping term. Furthermore, due to the absence of σ_1 in the the stability condition for the implicit scheme in Eq. (4.40), the bound on the grid spacing h relaxes and it allows for one more grid point. As the frequency-dependent damping term no longer affects the stability condition for the implicit scheme, and a more accurate simulation can be obtained with fewer numerically dispersive effects.

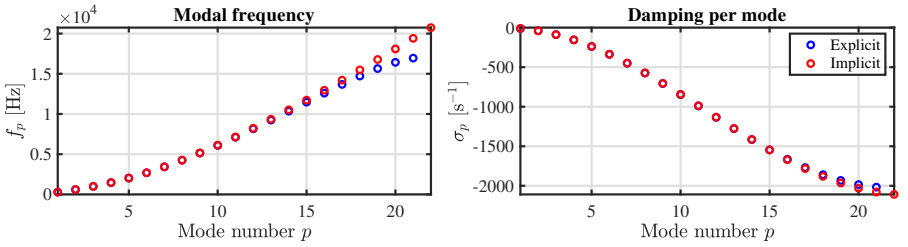


Fig. 4.8: A comparison between the modal frequencies and damping per mode of the explicit (blue) and implicit (red) scheme. Here, $T = 1885$, $E = 2 \cdot 10^{14}$ and $\sigma_1 = 1$ to highlight differences between the two schemes. One can observe that the modes of the implicit scheme follow the expected exponential pattern for the stiff string, where the explicit scheme shows numerically dispersive effects. Furthermore, due to the absence of σ_1 in the stability condition in Eq. (4.40) and allows for one more grid point

4.6.3 Conclusion

This section presented an implicit discretisation of the stiff string where the centred operator has been used to discretise the temporal derivative in the

frequency-dependent damping term. By means of stability analysis and modal analysis some advantages that the implicit scheme has over its explicit counterpart (presented in Section 4.2) have been shown.

As these advantages only show for higher values of σ_1, much higher than the ones used in this project, it has been chosen to use the explicit scheme for all further implementation in this project. The decrease in accuracy is negligible for lower values of σ_1 and the calculation of the scheme becomes orders of magnitude more computationally expensive if the implicit scheme is used.

Chapter 5

Brass

terms I could use: bore, tube, cylinder

In this work, the wave propagation in tubes is approximated using a 1D system

5.1 Webster's Equation

The main difference between the 1D brass PDE and the 1D wave equation is the presence of a variable cross-section. For low-amplitude vibrations in an (axially symmetric) acoustic tube and for which the wavelengths are much larger than the radius of this tube, one can describe its dynamics using *Webster*'s equation [44]

$$S\partial_t^2\Psi = c^2\partial_x(S\partial_x\Psi), \tag{5.1}$$

with *acoustic potential* $\Psi = \Psi(x,t)$ (in m^2/s), $S = S(x)$ is the cross sectional area (in m^2) and the speed of sound in air c (in m/s). If $S(x) = 1$ for all values of x, Eq. (5.1) reduces to the 1D wave equation in Eq. (2.38). The acoustic potential can be related to pressure $p = p(x,t)$ (in Pa) and particle velocity $v = v(x,t)$ (in m/s) according to

$$p = \rho\partial_t\Psi, \quad \text{and} \quad v = -\partial_x\Psi. \tag{5.2}$$

5.1.1 Discretisation

Introducing interleaved gridpoints at $n - 1/2$ and $n + 1/2$ for S, a we can discretise Eq. (5.1) (following [11]) to

$$\bar{S}\delta_{tt}\Psi_l^n = c^2\delta_{x+}(S_{l-1/2}(\delta_{x-}\Psi_l^n)), \tag{5.3}$$

where

$$\bar{S} = \mu_{x+}S_{l-1/2} = \frac{S_{l+1/2} + S_{l-1/2}}{2}. \tag{5.4}$$

The right side of the equation in (5.3) contains an operator applied to two grid functions (S and Ψ) multiplied onto each other. In order to expand this, we need to use the product rule (Eq. (2.23) in [3]) which is

$$\delta_{x+}(u_l w_l) = (\delta_{x+} u_l)(\mu_{x+} w_l) + (\mu_{x+} u_l)(\delta_{x+} w_l). \tag{5.5}$$

In the case of (5.3), $u_l \triangleq S_{l-1/2}$ and $w_l \triangleq \delta_{x-}\Psi_l^n$. Expanding (retaining the notation for $\bar{S}$) and solving for Ψ_l^{n+1} yields (Appendix **??**)

$$\Psi_l^{n+1} = 2(1-\lambda^2)\Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}}\Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}}\Psi_{l-1}^n, \tag{5.6}$$

which is identical to Eq. (19.51) in [11]. Also see Figure 5.1

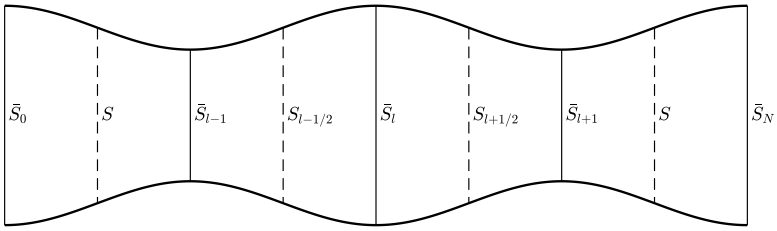


Fig. 5.1: Approximations to $S(x)$..left off here!

5.1.2 Boundary Conditions

The choices for boundary conditions in an acoustic tube are open and closed, defined as [11]

$$\begin{aligned} \partial_t \Psi &= 0 \text{ (open, Dirichlet)} \\ \partial_x \Psi &= 0 \text{ (closed, Neumann),} \end{aligned} \tag{5.7}$$

at the ends of the tube. This might be slightly counter-intuitive as in the case of a string “closed" might imply the “clamped" or Dirichlet boundary condition. The opposite can be intuitively shown imagining a wave front with a positive acoustic potential moving through a tube and hitting a closed end. What comes back is also a wave front with a positive acoustic potential, i.e., the sign of the potential does not flip, which also happens using the free or Neumann condition for the string.

In this case we follow [3, Chapter 9] and use the following

$$\partial_x \Psi(0,t) = 0 \quad \text{and} \quad \partial_t \Psi(L,t) = 0 \tag{5.8}$$

i.e. closed at the left end and open at the right end. In discrete time we have two choices for the closed condition

$$\begin{aligned} \delta_{x\cdot}\Psi_0^n = 0 \;\Rightarrow\; \Psi_{-1}^n = \Psi_1^n \quad \text{(centered)} \\ \delta_{x-}\Psi_0^n = 0 \;\Rightarrow\; \Psi_{-1}^n = \Psi_0^n \quad \text{(non-centered)} \end{aligned} \tag{5.9}$$

At the left boundary we can now solve Eq. (5.6) for the centered case:

$$\begin{aligned} \Psi_0^{n+1} &= 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\Psi_{-1}^n \\ \Psi_0^{n+1} &= 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 (S_{1/2}+S_{-1/2})}{\bar{S}_0}\Psi_1^n \\ \Psi_0^{n+1} &= 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + 2\lambda^2\Psi_1^n, \end{aligned}$$

and the non-centered case

$$\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\Psi_0^n. \tag{5.10}$$

As can be seen from the equations above, we need undefined points $\bar{S}_0$ and $S_{-1/2}$. At the left boundary, we set $\bar{S}_0 = S_0$ from which, we can calculate $S_{-1/2}$:

$$S_0 = \frac{1}{2}(S_{1/2} + S_{-1/2}) \;\Rightarrow\; S_{-1/2} = 2S_0 - S_{1/2} \tag{5.11}$$

The same can be done for the right boundary ($\bar{S}_N = S_N$) if this is chosen to be anything else but open (e.g., closed or radiating – see Section 5.1.2):

$$S_N = \frac{1}{2}(S_{N+1/2} + S_{N-1/2}) \;\Rightarrow\; S_{N+1/2} = 2S_N - S_{N-1/2}. \tag{5.12}$$

For now though, we follow the conditions given in (5.8) and we can simply set the right boundary to its initial state

$$\Psi_N^n = \Psi_N^0 \tag{5.13}$$

which is normally 0. A more realistic open end is a radiating one, which can be found below.

Radiating end

We can change the condition presented in Eq. (5.8) to a radiating end,

$$\partial_x \Psi(L,t) = -a_1 \partial_t \Psi(L,t) - a_2 \Psi(L,t) \tag{5.14}$$

where [3]

$$a_1 = \frac{1}{2(0.8216)^2 c} \quad \text{and} \quad a_2 = \frac{L}{0.8216\sqrt{S_0 S(1)/\pi}}. \tag{5.15}$$

taken from [**?**] and are valid for a tube terminating on an infinite plane. The terms in Eq. (5.14) are a damping and an inertia term where a_1 is a loss coefficient (in s/m) and a_2 is the **inertia coefficient** (in m^{-1}). The centered and non-centered case are defined as

$$\begin{aligned} \delta_{x\cdot}\Psi_N^n = 0 \;\Rightarrow\; \Psi_{N+1}^n = \Psi_{N-1}^n \quad \text{(centered)} \\ \delta_{x+}\Psi_N^n = 0 \;\Rightarrow\; \Psi_{N+1}^n = \Psi_N^n \quad \text{(non-centered)} \end{aligned} \tag{5.16}$$

First, we solve Eq. (5.14) for the centered (Eq. (9.16) in [3])

$$\delta_{x\cdot}\Psi_N^n = -a_1\delta_{t\cdot}\Psi_N^n - a_2\mu_{t\cdot}\Psi_N^n \tag{5.17}$$

which can be expanded and solved for Ψ_{N+1}^n according to

$$\begin{aligned} \frac{1}{2h}(\Psi_{N+1}^n - \Psi_{N-1}^n) &= -\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1}) \\ \Psi_{N+1}^n &= h\left(-\frac{a_1}{k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - a_2(\Psi_N^{n+1} + \Psi_N^{n-1})\right) + \Psi_{N-1}^n, \end{aligned} \tag{5.18}$$

which can be substituted into Eq. (5.6) (Appendix **??**)

$$\Psi_N^{n+1} = \frac{2(1-\lambda^2)\Psi_N^n - \Psi_N^{n-1} + \frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\left(\frac{a_1}{k} - a_2\right)\Psi_N^{n-1} + 2\lambda^2\Psi_{N-1}^n}{\left(1 + \left(\frac{a_1}{k} + a_2\right)\frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\right)}. \tag{5.19}$$

The same can be done for the non-centered case (Eq. (9.15) in [3])

$$\delta_{x+}\Psi_N^n = -a_1\delta_{t\cdot}\Psi_N^n - a_2\mu_{t\cdot}\Psi_N^n \tag{5.20}$$

which when solved for Ψ_{N+1}^n yields

$$\begin{aligned} \frac{1}{h}(\Psi_{N+1}^n - \Psi_N^n) &= -\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1}) \\ \Psi_{N+1}^n &= h\left(-\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1})\right) + \Psi_N^n. \end{aligned} \tag{5.21}$$

Substituted into Eq. (5.6) yields (Appendix **??**)

$$\Psi_N^{n+1} = \frac{2(1-\lambda^2)\Psi_N^n - \Psi_N^{n-1} + \frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\left(\frac{a_1}{2k} - \frac{a_2}{2}\right)\Psi_N^{n-1} + \frac{\lambda^2 S_{N+1/2}}{\bar{S}_N}\Psi_N^n + \frac{\lambda^2 S_{N-1/2}}{\bar{S}_N}\Psi_{N-1}^n}{\left(1 + \left(\frac{a_1}{2k} + \frac{a_2}{2}\right)\frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\right)}. \tag{5.22}$$

5.2 First-order system

This will be the first appearance of a first-order system.

Chapter 6

2D Systems

In this work, it is mainly used to model a simplified body in papers...

Rectangular system described by state variable $u = u(x, y, t)$ where $t \geq 0$ and $(x, y) \in \mathcal{D}$ where $\mathcal{D}$ is 2-dimensional. The state variable can then be discretised according to $u(x, y, t) \approx u_{l,m}^n$ with space $x = lh$ and $y = mh$ and time $t = nk$ and $k = 1/f_\mathrm{s}$. For simplicity the grid spacing in both the x and y directions are set to be the same but could be different.

Circular or elliptical systems can be modelled using a staircase approximation or radial coordinates [3]

In continuous time the operators:

$$\Delta = \partial_x^2 + \partial_y^2 \tag{6.1}$$

The same shift operators as defined in Chapter 2 can be applied to grid function $u_{l,m}^n$. Additional ones are

$$e_{y+}u_{l,m}^n = u_{l,m+1}^n, \quad \text{and} \quad e_{y-}u_{l,m}^n = u_{l,m-1}^n. \tag{6.2}$$

6.1 Analysis Techniques in 2D

Here, some of the differences between the analysis techniques presented in Chapter 3 and those in 2D will be elaborated on. Although, modal analysis remains the same

6.1.1 Frequency Domain Analysis

$p_x + p_y$

ansatz:

$$u_{l,m}^n = z^n e^{jh(l\beta_x + m\beta_y)} \tag{6.3}$$

6.1.2 Energy Analysis

Squared domain $d \in \{0, \ldots, N_x\} \times \{0, \ldots, N_y\}$

$$\langle f^n, g^n \rangle_d = \sum_{l=0}^{N_x} \sum_{m=0}^{N_y} h^2 f^n_{l,m} g^n_{l,m} \tag{6.4}$$

6.1.3 Modal Analysis

Stacked matrix form

6.2 2D Wave Equation

The 2D wave equation be used to model an ideal membrane such as done in

It has identical behaviour to the 2D waveguide mesh presented by van Duyne and Smith [45].

Consider a square ideal membrane with side lengths L_x and L_y (both in m) and its transverse displacement described by $u = u(x, y, t)$. The membrane is defined over $(x, y) \in \mathcal{D}$ with domain $\mathcal{D} = [0, L_x] \times [0, L_y]$ and its motion is described by the following PDE

$$\partial_t^2 u = c^2 \Delta u \tag{6.5}$$

with wave speed $c = \sqrt{T/\rho H}$ (in m/s), tension per unit length (applied to the boundary)T (in N/m), material density ρ (in kg/m^3) and thickness H.

check whether this is right..

6.3 Thin plate

Used in [A], [B], [D] and [E] biharmonic operator, Laplacian in (6.1) applied to itself.

$$\rho H \partial_t^2 u = -D \Delta \Delta u \tag{6.6}$$

where $D = EH^3/12(1 - \nu^2)$

6.4 Stiff membrane

Combination between Eqs. (6.7) and (6.6)

$$\rho H \partial_t^2 u = T \Delta u - D \Delta \Delta u \tag{6.7}$$

[F]

check whether all references are used

References

[1] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Institute of Physics Publishing*, 2006.

[2] J. O. Smith, "Physical audio signal processing (online book)," 2010. [Online]. Available: http://ccrma.stanford.edu/~jos/pasp/

[3] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley & Sons, 2009.

[4] J. O. Smith, "Virtual acoustic musical instruments: Review and update," *Center for computer research in music and acoustics (CCRMA)*, 2010.

[5] J. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society*, vol. 21, no. 7, 526–534.

[6] M. L. Lavengood, "What makes it sound '80s?: The yamaha DX7 electric piano sound," *Journal of Popular Music Studies*, vol. 31, pp. 73–94, 2019.

[7] J. Kelly and C. Lochbaum., "Speech synthesis," in *Proceedings of the Fourth International Congress on Acoustics*, 1962, pp. 1–4.

[8] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitator-resonator interaction in physical models syntheses," *Proceedings of the International Computer Music Conference*, 1989.

[9] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1998.

[10] J. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[11] S. Bilbao, B. Hamilton, R. L. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial." *Springer handbook of systematic musicology*, pp. 349–384, 2018.

[12] R. Michon, S. Martin, and J. O. Smith, "MESH2FAUST: a modal physical model generator for the faust programming language - application to bell modeling," in *Proceedings of the 2017 International Computer Music Conference, ICMC*, 2017.

[13] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[14] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 6, pp. 462–470, 1971.

[15] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 7, pp. 542–550, 1971.

[16] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[17] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[18] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[19] C. Erkut and M. Karjalainen, "Finite difference method vs. digital waveguide method in string instrument modeling and synthesis," *International Symposium on Musical Acoustics*, 2002.

[20] E. Maestre, C. Spa, and J. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC|SMC|2014*, pp. 1305–1311, 2014.

[21] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," 1979, thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France.

[22] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.

[23] ——, "CORDIS-ANIMA: a modeling and simulation system for sound and image synthesis: the general formalism," *Computer Music Journal*, vol. 17, no. 1, pp. 19–29, 1993.

[24] J. Villeneuve and J. Leonard, "Mass-interaction physical models for sound and multi-sensory creation: Starting anew," in *Proceedings of the 16th Sound and Music Computing Conference*, 2019.

[25] J. Leonard and J. Villeneuve, "MI-GEN~: An efficient and accessible mass interaction sound synthesis toolbox," *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, 2019.

[26] L. Trautmann and R. Rabenstein, *Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method*. Springer-Science + Business Media LLC, 2003.

[27] D. Matignon, P. Depalle, and X. Rodet, "State space models for wind-instrument synthesis," *ICMC*, 1992.

[28] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114—117, 1965.

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

[30] S. Bilbao, J. Perry, P. Graham, A. Gray, K. Kavoussanakis, G. Delap, T. Mudd, G. Sassoon, T. Wishart, and S. Young, "Large-scale physical modeling synthesis, parallel computing, and musical experimentation: The ness project in practice," *Computer Music Journal*, vol. 43, no. 2-3, pp. 31–47, 2019.

[31] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, California: Wadsworth and Brooks/Cole Advanced Books and Software, 1989.

[32] B. Hamilton, "Finite difference and finite volume methods for wave-based modelling of room acoustics," Ph.D. dissertation, The University of Edinburgh, 2016.

[33] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2018.

[34] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Applied Sciences*, vol. 6, no. 5, 2016.

[35] S. Bilbao and J. Parker, "A virtual model of spring reverberation," *IEEE transactions on audio, speech, and language processing*, vol. 18, pp. 799–808, 2009.

[36] S. Bilbao, "A modular percussion synthesis environment," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.

[37] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen de mathematischen physik," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928.

[38] T. H. Park, *Introduction To Digital Signal Processing: Computer Musically Speaking*. World Scientific Publishing Co. Pte. Ltd, 2010.

[39] J. G. Charney, R. Fjörtoft, and J. V. Neumann, "Numerical integration of the barotropic vorticity equation," *Tellus*, vol. 2, no. 4, 1950.

[40] R. Zucker, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards Applied Mathematics Series 55, 1972, ch. 4: Elementary Transcendental Functions, pp. 65–226, Tenth Printing.

[41] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time-Dependent Problems and Difference Methods*, 2nd ed. John Wiley & Sons, 2013.

[42] R. L. Harrison-Harsley, "Physical modelling of brass instruments using finite-difference time-domain methods," Ph.D. dissertation, University of Edinburgh, 2018.

[43] J. Bensa, S. Bilbao, R. Kronland-Martinet, and J. O. Smith, "The simulation of piano string vibration: From physical models to finite difference schemes and digital waveguides," *Journal of the Acoustical Society of America (JASA)*, vol. 114, no. 2, pp. 1095–1107, 2003.

[44] A. Webster, "Acoustical impedance, and the theory of horns and of the phonograph," in *Proceedings of the National Academy of Sciences of the United States of America*, vol. 5, no. 7, 1919, pp. 275–282.

[45] S. A. van Duyne and J. O. Smith, "Physical modeling with the 2-D digital waveguide mesh," in *ICMC Proceedings*, 1993.

check whether to sort references or not

Part III

Appendix

Appendix A

List of Symbols

The list of symbols found below contains often-used symbols in the thesis in the context that they are normally used. Depending on the context they might carry a different meaning (y being displacement of the lip-reed in Chapter **??** but the vertical spatial coordinate for 2D systems in fx. Chapter 6). Some might also be accompanied by a subscript in the main document

Symbol	Description	Unit
α	Fractional part of	
A	Cross-sectional area of string	m^2
c	Wave speed	$\mathrm{m/s}$
$\frac{d^n}{dt^n}$	n^{th} order derivative with respect to t	-
∂_t^n	n^{th} order partial derivative with respect to t	-
δ_{t+}, δ_{t-}, $\delta_{t\cdot}$	Forward, backward and centred difference in time operator	-
δ_{x+}, δ_{x-}, $\delta_{x\cdot}$	Forward, backward and centred difference in space operator	-
δ_{tt}	Second order difference in time operator	-
δ_{xx}	Second order difference in space operator	-
δ_{xxxx}	Fourth order difference in space operator	-
μ_{t+}, μ_{t-}, $\mu_{t\cdot}$	Forward, backward and centred average in time operator	-
μ_{x+}, μ_{x-}, $\mu_{x\cdot}$	Forward, backward and centred average in space operator	-
μ_{tt}	Second order average in time operator	-
μ_{xx}	Second order average in space operator	-

Symbol	Description	Unit
E	Young's Modulus	Pa ($\mathrm{kg{\cdot}m^{-1}{\cdot}s^{-2}}$)
f	Force	N
f	Frequency	Hz
f_s	Sample rate	Hz
F	Scaled force	depends on system
h	Grid spacing	m
H	Membrane / Plate thickness	m
I	Area moment of inertia	m^4
l	Spatial index to grid function	-
L	Length	m
k	Time step ($= 1/f_\mathrm{s}$)	s
K	Spring coefficient	$\mathrm{N/m}$
κ	Stiffness coefficient	$\mathrm{m^2/s}$ (1D) $\mathrm{m^4{\cdot}s^{-2}}$ (2D)
n	Sample index to grid function	-
N	Number of points string	-
$\mathbb{N}^0$	Set of non-negative integers	-
p	Pressure	Pa
r	Radius	m
S	Cross-sectional area (brass)	m^2
t	Time	s
T	Tension	N (1D) $\mathrm{N/m}$ (2D)
u	State variable	m
v	Particle velocity	$\mathrm{m/s}$
x	Spatial dimension (horizontal for 2D systems)	m
y	Vertical spatial dimension	m
γ	Scaled wave speed	s^{-1}
λ	Courant number for 1D wave eq. ($= ck/h$)	-
μ	Stiffness free parameter	-
ν	Poisson's ratio	-
η	Relative displacement spring	m
ρ	Material density	$\mathrm{kg{\cdot}m^{-3}}$

Operations

Symbol	Description	Unit
$\mathfrak{I}(\cdot)$	Imaginary part of	
$\mathfrak{R}(\cdot)$	Real part of	
$\lfloor \cdot \rfloor$	Flooring operation	
$\lceil \cdot \rceil$	Ceiling operation	
Subscripts		
c	Connection	
p	Plate	
s	String	

Appendix A. List of Symbols

Appendix B

List of Abbreviations

Abbreviation	Definition
1D	one-dimensional
2D	two-dimensional
3D	three-dimensional
DoF	Degrees of freedom
DSP	Digital signal processing
Eq.	Equation
Eqs.	Equations
FD	Finite-difference
FDTD	Finite-difference time-domain
LTI	Linear time-invariant
ODE	Ordinary differential equation
PDE	Partial differential equation

Appendix C

Code Snippets

C.1 Mass-Spring System (Section 2.3)

```
%% Initialise variables
fs = 44100;              % sample rate [Hz]
k = 1 / fs;              % time step [s]
lengthSound = fs;        % length of the simulation (1 second) [samples]

f0 = 440;                % fundamental frequency [Hz]
omega0 = 2 * pi * f0;    % angular (fundamental) frequency [Hz]
M = 1;                   % mass [kg]
K = omega0^2 * M;        % spring constant [N/m]

%% initial conditions (u0 = 1, d/dt u0 = 0)
u = 1;
uPrev = 1;

% initialise output vector
out = zeros(lengthSound, 1);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.35)
    uNext = (2 - K * k^2 / M) * u - uPrev;

    out(n) = u;

    % Update system states
    uPrev = u;
    u = uNext;
end
```

C.2 1D Wave Equation (Section 2.4)

```
%% Initialise variables
fs = 44100;          % Sample rate [Hz]
k = 1 / fs;          % Time step [s]
lengthSound = fs;    % Length of the simulation (1 second) [samples]

c = 300;             % Wave speed [m/s]
L = 1;               % Length [m]
h = c * k;           % Grid spacing [m] (from CFL condition)
N = floor(L/h);      % Number of intervals between grid points
h = L / N;           % Recalculation of grid spacing based on integer N

lambdaSq = c^2 * k^2 / h^2; % Courant number squared

% Boundary conditions ([D]irichlet or [N]eumann)
bcLeft = "D";
bcRight = "D";

%% Initialise state vectors (one more grid point than the number of
    intervals)
uNext = zeros(N+1, 1);
u = zeros(N+1, 1);

%% Initial conditions (raised cosine)
loc = round(0.8 * N);        % Center location
halfWidth = round(N/10);     % Half-width of raised cosine
width = 2 * halfWidth;       % Full width
rcX = 0:width;               % x-locations for raised cosine

rc = 0.5 - 0.5 * cos(2 * pi * rcX / width); % raised cosine
u(loc-halfWidth : loc+halfWidth) = rc; % initialise current state

% Set initial velocity to zero
uPrev = u;

% Range of calculation
range = 2:N;

% Output location
outLoc = round(0.3 * N);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.44)
    uNext(range) = (2 - 2 * lambdaSq) * u(range) ...
        + lambdaSq * (u(range+1) + u(range-1)) - uPrev(range);

    % boundary updates Eq. (2.49)
    if bcLeft == "N"
        uNext(1) = (2 - 2 * lambdaSq) * u(1) - uPrev(1) ...
        + 2 * lambdaSq * u(2);
```

```
    end

    % Eq. (2.50)
    if bcRight == "N"
        uNext(N+1) = (2 - 2 * lambdaSq) * u(N+1) - uPrev(N+1) ...
        + 2 * lambdaSq * u(N);
    end

    out(n) = u(outLoc);

    % Update system states
    uPrev = u;
    u = uNext;
end
```

Part IV

Papers

Paper Errata

Here, some errors in the published papers will be listed:

Real-Time Tromba [D]

- The minus sign in Eq. (28) (and thus Eqs. (31) and (35)) should be a plus sign.
- $\sigma_{1,\mathrm{s}}$ in Eq. (21) should obviously be $\sigma_{1,\mathrm{p}}$
- the unit of the spatial Dirac delta function δ should be m^{-1}

DigiDrum [F]

- σ_0 and σ_1 should be multiplied by ρH in order for the stability condition to hold.
- stability condition is wrong. Should be:

$$h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}} \tag{1}$$

- Unit for membrane tension is $\mathrm{N/m}$.

Dynamic grids [G]

- Reference in intro for 'recently gained popularity' should go to [29]
  Note: not really an error, but should be changed before resubmission

Paper A

Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Silvin Willemsen, Nikolaj Andersson, Stefania Serafin
and Stefan Bilbao

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp.
275–280, 2019.

Abstract

In this paper, implementation, instrument design and control issues surrounding a modular physical modelling synthesis environment are described. The environment is constructed as a network of stiff strings and a resonant plate, accompanied by user-defined connections and excitation models. The bow, in particular, is a novel feature in this setting. The system as a whole is simulated using finite difference (FD) methods. The mathematical formulation of these models is presented, alongside several new instrument designs, together with a real-time implementation in JUCE using FD methods. Control is through the Sensel Morph.

1 Introduction

Physical models for sound synthesis have been researched for several decades to mathematically simulate the sonic behaviour of musical instruments and everyday sounds. Various techniques and methodologies have developed, ranging from mass-spring models [1, 2, 3] to modal synthesis [4] and waveguide based models [5]. The latter two techniques may be viewed as numerical simulation techniques applied to the systems of partial differential equations (PDEs). These equations define the dynamics of a musical instrument, either real or imagined.

Mainstream time-domain simulation techniques, such as finite difference (FD) methods, were first applied to the case of string vibration by Ruiz [6] and Hiller and Ruiz [7, 8], and then later by other authors [9] including, most notably Chaigne [10] and Chaigne and Askenfelt [11]. The general use of finite-difference schemes (FDSs) in sound synthesis is described in [12]. Modularized physical modelling sound synthesis, whereby the user may construct a virtual instrument using basic canonical components dates back to the work of Cadoz and collaborators [1, 2, 3]. It has been also used as a design principle in the context of FD methods [13, 14, 15], where the canonical elements are strings and plates, with a non-linear connection mechanism. Though computational cost of such methods is high, standard computing power is now approaching a level suitable for real-time performance for simpler systems.

We are interested in bridging the gap between large-scale modular physical modelling synthesis and sonic interaction design [16], to be able to play with such simulations in real-time. Specifically, we are interested in using the expressivity of the Sensel Morph [17] to control our simulations, using both percussive and bowing excitations. Our ultimate goal is to create models that are both mathematically accurate and efficient. This goal is nowadays possible thanks to improvements in hardware and software technologies for sound synthesis, yet it has rarely been achieved. The ultimate goal is to provide a modular efficient synthesizer based on accurate simulations, where real-

time expressivity can also be achieved. This synthesizer has already been informally evaluated by composers and sound designers, who appreciated the current sonic palette.

This paper is structured as follows: Section 2 describes the physical models used in the implementation and Section 3 shows a general description of the FD methods used to digitally implement these models. Furthermore, Section 4 elaborates on the real-time implementation, Section 5 shows several different configurations of the physical models inspired by real musical instruments, Section 6 will present the results on CPU usage and evaluation and discuss this and finally, in Section 7, some concluding remarks appear.

2 Models

In this section, the PDEs for the damped stiff string and plate will be presented. The notation used will be the one found in [12] where the subscript for state variable u denotes a single derivative with respect to time t or space x respectively. Furthermore, to simplify the presented physical models, non-dimensionalization (or scaling) will be used [12].

2.1 Stiff string

A basic model of the linear transverse motion of a string of circular cross section may be described in terms of several parameters: the total length L (in m), the material density ρ (in kg·m^{-3}), string radius r (in m), Young's modulus E (in Pa), tension T (in N), and two loss parameters σ_0 and σ_1. The PDE for a damped stiff string may be written as [12]

$$u_{tt} = \gamma^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \tag{1}$$

In this representation, spatial scaling has been employed using a length L, so the solution $u = u(x,t)$ is defined for $t \geq 0$ and for dimensionless coordinate $x \in [0,1]$. Furthermore, parameters $\gamma = \sqrt{T/\rho\pi r^2 L^2}$ and $\kappa = \sqrt{Er^2/4\rho L^4}$ and have units s^{-1}.

In this work, the string is assumed clamped at both ends, so that

$$u = u_x = 0 \quad \text{where} \quad x = \{0,1\}. \tag{2}$$

A model of a bowed string [12] may be incorporated into (1) as

$$u_{tt} = \ldots - \delta(x - x_{\text{B}}) F_{\text{B}} \phi(v_{\text{rel}}), \quad \text{with} \tag{3a}$$

$$v_{\text{rel}} = u_t|_{(x=x_{\text{B}})} - v_{\text{B}}, \tag{3b}$$

where $F_{\text{B}} = f_{\text{B}}/M_{\text{s}}$ is the excitation function (in m/s^2) with externally-supplied bowing force $f_{\text{B}} = f_{\text{B}}(t)$ (in N) and total string mass $M_{\text{s}} = \rho\pi r^2 L$ (in kg).

The relative velocity v_{rel} is defined as the difference between the velocity of the string at bowing point x_{B} and the externally-supplied bowing velocity $v_{\text{B}} = v_{\text{B}}(t)$ (in m/s) and ϕ is a dimensionless friction characteristic, chosen here as [12]

$$\phi(v_{\text{rel}}) = \sqrt{2a} v_{\text{rel}} e^{-a v_{\text{rel}}^2 + 1/2}. \tag{4}$$

Furthermore, $\delta(x - x_{\text{B}})$ is a spatial Dirac delta function selecting the bowing location $x = x_{\text{B}}$. The single bowing point can be extended to a bowing area [12]. More detailed models of string dynamics, again in a bowed string context, have been proposed by Desvages [18].

Another, and more simple way to excite the string is by extending Equation (1) to

$$u_{tt} = \ldots + E_{\text{e}} F_{\text{e}} \tag{5}$$

using an externally-supplied distribution function $E_{\text{e}} = E_{\text{e}}(x)$ and excitation function $F_{\text{e}} = F_{\text{e}}(t)$. In this case, the excitation region is allowed to be of finite width.

2.2 Plate

Under linear conditions, a rectangular plate of dimensions L_x and L_y may be parameterized in terms of density ρ (in kg$\cdot$ m^{-3}), thickness H (in m), Young's modulus E (in Pa) and a dimensionless Poisson's ratio ν, as well as two loss parameters σ_0 and σ_1.

In terms of dimensionless spatial coordinates x and y scaled by $\sqrt{L_x L_y}$, the equation of motion of a damped plate is a variant of the Kirchhoff model [19]

$$u_{tt} = -\kappa^2 \Delta\Delta u - 2\sigma_0 u_t + 2\sigma_1 \Delta u_t. \tag{6}$$

Here, $u(x, y, t)$ is the transverse displacement of the plate as a function of dimensionless coordinates $x \in [0, \sqrt{a}\,]$, $y \in [0, 1/\sqrt{a}\,]$, where $a = L_x/L_y$ is the plate aspect ratio, as well as time t. Furthermore, Δ represents the 2D Laplacian [12]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{7}$$

The stiffness parameter κ, with dimensions of s^{-1}, is defined by $\kappa = \sqrt{D/\rho H L_x^2 L_y^2}$ where $D = EH^3/12\left(1 - \nu^2\right)$. As in the case of the stiff string, we chose to use clamped boundary conditions:

$$u = \mathbf{n} \cdot \nabla u = 0 \tag{8}$$

over any plate edge with outward normal direction $\mathbf{n}$ and where ∇u is the gradient of u.

2.3 Connections

Adding connections between different physical models, further referred to as elements, adds another term to Equation (3a), (5) or (6). Assuming that element α is a stiff string and β is a plate, the following terms are added to the aforementioned equations:

$$u_{tt} = ... + E_{\mathrm{c},\alpha} F_\alpha, \tag{9a}$$

$$u_{tt} = ... + E_{\mathrm{c},\beta} F_\beta, \tag{9b}$$

with force-functions $F_\alpha = F_\alpha(t)$ and $F_\beta = F_\beta(t)$ (in $\mathrm{m/s^2}$) and distribution functions $E_{\mathrm{c},\alpha}$ and $E_{\mathrm{c},\beta}$ which have chosen to be highly localised in our application and reduce to $\delta(x - x_{\mathrm{c},\alpha})$ and $\delta(x - x_{\mathrm{c},\beta}, y - y_{\mathrm{c},\beta})$ respectively, but can be extended to be connection areas [13]. We use the implementation as presented in [13] where the connection between two elements is a non-linear spring. The forces it imposes on the elements it connects are defined as

$$F_\alpha = -\omega_0^2 \eta - \omega_1^4 \eta^3 - 2\sigma_\times \dot{\eta}, \tag{10a}$$

$$F_\beta = -\mathcal{M} F_\alpha, \tag{10b}$$

where ω_0 and ω_1 are the linear (in $\mathrm{s^{-1}}$) and non-linear (in $(\mathrm{m{\cdot}s})^{-1/2}$) frequencies of oscillation respectively, $\sigma_\times$ is a damping factor (in $\mathrm{s^{-1}}$), $\mathcal{M}$ is the mass ratio between the two elements and η is the relative displacement between the connected elements at the point of connection (in m). Lastly, the dot above η denotes a derivative with respect to time.

3 Finite-Difference Schemes

To be able to digitally implement the continuous equations shown in the previous section, they need to be approximated. In this section, a high-level review of a finite difference approximation to a connected system of strings and plates is presented. For more technical details, see [13].

In the case of the stiff string, state variable $u(x,t)$ can be discretised at times $t = nk$, where $n \in \mathbb{N}$ and $k = 1/f_\mathrm{s}$ is the time step (at sample-rate f_s) and locations $x = lh$, with $l \in [0, \ldots, N]$ where the total number of points is $N + 1$ and grid spacing h. We can now write the discretised state variable as u_l^n, representing an approximation to $u(x,t)$.

In the case of the plate, $u(x,y,t)$ is discretised to $u_{(l,m)}^n$ using $x = lh$ where $l \in [0, \ldots, N_x]$ with $N_x + 1$ being the total horizontal number of points and $y = mh$ where $m \in [0, \ldots, N_y]$ with $N_y + 1$ being the total vertical number of points.

In a general sense, when discretising PDEs as presented in Equations (1) and (6), we will need to solve for $\mathbf{u}^{n+1}$, i.e., $\mathbf{u}$ at the next time step, where $\mathbf{u}$ is a

vector of size $N-1$ containing values of $u_l \; \forall l$ for a string and $(N_x-1)(N_y-1)$ containing values of $u_{(l,m)} \; \forall (l,m)$ for a plate. Note that the vector sizes are smaller than the total number of grid points as we do not include the values at the boundaries (which are always 0). For a PDE expressed as a function of u_{tt}, its FDS will be of the form

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}^n, \tag{11}$$

where

$$K = \frac{k^2}{1+\sigma_0 k}, \tag{12}$$

and $\boldsymbol{\mathcal{F}}^n$ is a combination of the discretised PDE (excluding terms containing u^{n+1}) together with connection and excitation terms.

3.1 Stiff String

In the case of the stiff string, $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (1) $\mathbf{f}_\alpha^n$, connection term (9a) and bowing (3a)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha} F_\alpha^n - \mathbf{J}(x_\mathrm{B}^n) F_\mathrm{B}^n \phi(v_\mathrm{rel}), \tag{13a}$$

or excitation (5) term

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha} F_\alpha^n + \mathbf{E}_\mathrm{e} F_\mathrm{e}^n, \tag{13b}$$

where $\mathbf{E}_{\mathrm{c},\alpha}$ contains the discretised distribution function for the connection ($1/h$ at connection index $l_{\mathrm{c},\alpha}$, rest 0's [12]), $\mathbf{E}_\mathrm{e}$ contains the discretised distribution function for the excitation (which will be presented in Equation (25) in the next section) and $\mathbf{J}(x_\mathrm{B}^n)$ is a spreading operator containing the discretised bowing distribution ($1/h$ at time-varying bowing position x_B). If x_B is between grid points, cubic interpolation is used to spread the bow-force over neighbouring grid points [12]. All vectors are columns of size $N-1$.

It can be useful to talk about the *region of operation* of a FDS in terms of a 'stencil'. A stencil describes the number of grid points needed to calculate a single point at the next time step. The stiff string FDS has a stencil of 5 grid points. In other words, two grid points at either side of l – and l itself – are necessary to calculate u_l^{n+1}. See Figure 1 for a visualisation of this.

In order for the scheme to be stable, the grid spacing needs to abide the following condition [12]

$$h \geq \sqrt{\frac{\gamma^2 k^2 + 4\sigma_1 k + \sqrt{(\gamma^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}. \tag{14}$$

The closer h is to this limit, the higher the quality of the implementation. The number of points N can then be calculated using

$$N = \mathrm{floor}\left(\frac{1}{h}\right). \tag{15}$$

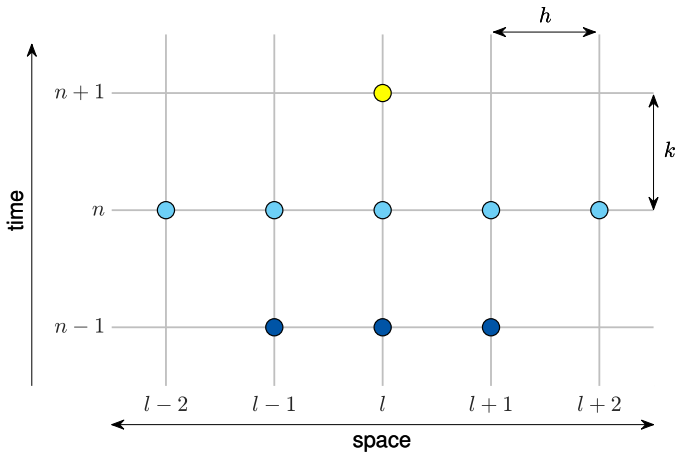


Fig. 1: Stencil for a stiff string FDS with grid spacing h and time step k. The point l at the next time step (yellow) is calculated using 5 points at the current time step (blue) and 3 at the previous time step (dark blue).

3.2 Plate

In the case of the plate, $\mathbf{u}$ is a column vector of concatenated vertical 'strips' of the plate state as in [13] of size $(N_x - 1)(N_y - 1)$ and $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (6) $\mathbf{f}_\beta^n$ and connection term (9b)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\beta^n + \mathbf{E}_{\mathrm{c},\beta} F_\beta^n. \tag{16}$$

Here, $\mathbf{E}_{\mathrm{c},\beta}$ contains the discretised distribution function for the connection ($1/h^2$ at connection index ($l_{\mathrm{c},\beta}$, $m_{\mathrm{c},\beta}$), rest 0's [13]) and is a column vector of size $(N_x - 1)(N_y - 1)$. For the plate, the stencil will consist of 13 grid points as can be seen in Figure 2.
The grid spacing needs to abide the following condition [13]

$$h \geq 2\sqrt{k\left(\sigma_1^2 + \sqrt{\kappa^2 + \sigma_1^2}\right)}, \tag{17}$$

(again, the closer h is to this limit the better) from which N_x and N_y can be derived using

$$N_x = \mathrm{floor}\left(\frac{\sqrt{a}}{h}\right) \quad \text{and} \quad N_y = \mathrm{floor}\left(\frac{1}{h\sqrt{a}}\right). \tag{18}$$

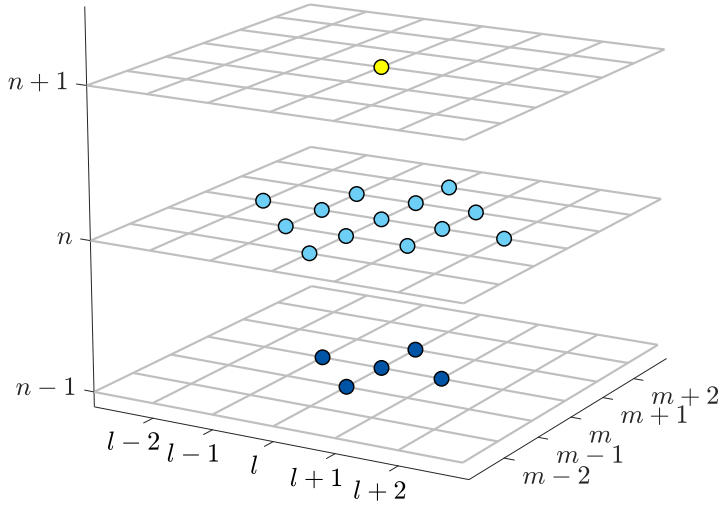


Fig. 2: Stencil for a plate FDS. The point (l, m) at the next time step (yellow) is calculated using 13 points at the current time step (blue) and 5 at the previous time step (dark blue).

3.3 Connections

In the following, we discretise the equations in (10) as shown in [13]. However, as these equations are not expressed as a function of u_{tt}, their FDS counterpart will be different. Moreover, instead of solving for $\mathbf{u}^{n+1}$, we need to solve for η^{n+1}, i.e., the relative displacement at the next time step, which will be in the form of

$$\eta^{n+1} = p^n F_\alpha^n + r^n \eta^{n-1}, \tag{19}$$

where $p^n = p(\eta^n)$ and $r^n = r(\eta^n)$ are functions of the relative displacement η if $\omega_1 \neq 0$ and constants if $\omega_1 = 0$. Again, assuming that element α is a stiff string and β is a plate, η can be calculated using

$$\eta^n = h_\alpha u^n_{\alpha, l_{\mathrm{c},\alpha}} - h_\beta^2 u^n_{\beta,(l_{\mathrm{c},\beta}, m_{\mathrm{c},\beta})}. \tag{20}$$

In other words, this is the difference between the state of element α at $l_{\mathrm{c},\alpha}$ and the state of element β at $(l_{\mathrm{c},\beta}, m_{\mathrm{c},\beta})$ scaled by their respective (for plates, squared) grid spacings h_α and h_β. The next step is to obtain F_α^n, which can be used to easily calculate F_β^n. We first obtain values for $\mathbf{u}^{n+1}$ by solving (11) using (13a), (13b) or (16) (without the connection term!) for a string or plate respectively. As, at this point, no connection forces have been added yet, this state will be referred to as an intermediate state $\mathbf{u}^{\mathrm{I}}$. This intermediate state can

be used to obtain η^{n+1} using (20)

$$\eta^{n+1} = h_\alpha(u^{\mathrm{I}}_{\alpha,l_{\mathrm{c},\alpha}} + K_\alpha F^n_\alpha) - \Big[h^2_\beta(u^{\mathrm{I}}_{\beta,(l_{\mathrm{c},\beta},m_{\mathrm{c},\beta})} + K_\beta F^n_\beta)\Big], \tag{21}$$

where K_α and K_β are as described in (12) using the damping coefficient σ_0 of their respective element. This can then be set equal to (19). Using Equation (10b), solving for F_α yields

$$F^n_\alpha = \frac{r^n\eta^{n-1} - (h_\alpha u^{\mathrm{I}}_{\alpha,l_{\mathrm{c},\alpha}} - h^2_\beta u^{\mathrm{I}}_{\beta,(l_{\mathrm{c},\beta},m_{\mathrm{c},\beta})})}{h_\alpha K_\alpha + \mathcal{M}h^2_\beta K_\beta - p^n}. \tag{22}$$

4 Implementation

In this section, we elaborate more on the chosen values for the parameters described in the previous two sections and present the system architecture of the real-time application. The values for most parameters have been arbitrarily chosen and can – as long as they satisfy the conditions in Equations (14) and (17) – be changed. We used C++ along with the JUCE framework [20] for implementing the physical models and connections in real-time. The main hardware used was a MacBook Pro with a 2.2 GHz Intel Core i7 processor.

4.1 Stiff String

As many string properties stay constant, we chose to set the following parameters directly, rather than calculating them from their physical properties: $\kappa = 2$, $\sigma_0 = 1$, $\sigma_1 = 0.005$. An interesting parameter to make dynamic is the fundamental frequency f_0 (in s^{-1}) of the string. According to [12], the fundamental frequency can be approximately calculated using

$$f_0 \approx \frac{\gamma}{2}. \tag{23}$$

However, as the grid spacing h is dependent on the wave speed γ according to the condition found in (14), we must put a lower limit on the number of points N if we plan to dynamically increase γ.

Another way to change frequency is to add damping to the model at specific points acting as a (simplified) fretting finger. The advantage of this is that the condition (14) will never be violated. On top of this, a tapping sound will be introduced when fretting the string making it more realistic than changing the wave speed. If the string is fretted at single location $x_\mathrm{f} \in [0, 1]$ and $l_\mathrm{f} = \mathrm{floor}(x_\mathrm{f}/h)$ we use

$$u^n_l = \begin{cases} 0, & l = l_\mathrm{f} - 1 \vee l = l_\mathrm{f} \\ (1 - \alpha^\epsilon_\mathrm{f})u^n_l, & l = l_\mathrm{f} + 1 \\ u^n_l, & \text{otherwise} \end{cases} \tag{24}$$

where $\alpha_{\mathrm{f}} = x_{\mathrm{f}}/h - l_{\mathrm{f}}$ describes the fractional location of x_{f} between two grid points. Note that the grid point at the finger location and the grid point before are set to 0 to (recalling the stencil) prevent the states at either side of the finger to influence each other. The disadvantage of using this technique over regular linear interpolation, is that the effect of damping between grid points does not linearly scale to pitch. We thus added $\epsilon = 7$ as a heuristic value to more properly map finger position to pitch.

In some cases, N is fixed to a certain value (as opposed to calculating it from Equations (14) and (15)) for multiple strings of different pitches. Even though some bandwidth will be lost (in the higher frequency range), this will allow the strings to be perfectly tuned to each other.

4.1.1 Bowed String

Parameters for the bowed strings abide the following conditions: $|v_{\mathrm{B}}| \leq 1$ m/s and $0 \leq F_{\mathrm{B}} \leq 100$ N. It was chosen to discretise Equation (3b) implicitly making it necessary to use an iterative root-finding method such as Newton-Raphson [21].

4.1.2 Excited string

If simply excited, we set the distribution function to a raised cosine with width w_{e} (in grid points)

$$E_{\mathrm{e}}(l) = \begin{cases} \frac{1-\cos(\frac{2\pi(l-(l_{\mathrm{e}}-w_{\mathrm{e}}/2))}{w_{\mathrm{e}}})}{2}, & l_{\mathrm{e}} - \frac{w_{\mathrm{e}}}{2} < l < l_{\mathrm{e}} + \frac{w_{\mathrm{e}}}{2} \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

scaled by the excitation function over time with excitation duration d_{e} (in samples)

$$F_{\mathrm{e}}(n) = \begin{cases} \frac{1-\cos(\frac{\pi(n-n_{\mathrm{e}})}{d_{\mathrm{e}}})}{2}, & n_{\mathrm{e}} \leq n < n_{\mathrm{e}} + d_{\mathrm{e}} \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

A visualisation of this can be found in Figure 3.

4.2 Plate

For the plate, the damping coefficients have been decided to be $\sigma_0 = 0.1$ and $\sigma_1 = 0.005$ and the aspect ratio is set to $a = 2$. The plate stiffness κ has been left as a user parameter to be changed dynamically and will be between the following bounds: $0.1 \leq \kappa \leq 50$ s^{-1}. In Equation (17), the grid spacing is calculated using the maximum value of κ to prevent stability issues. Using a sample rate of 44,100 Hz results in a plate with dimensions $N_x = 20$ and $N_y = 10$ (in grid points).

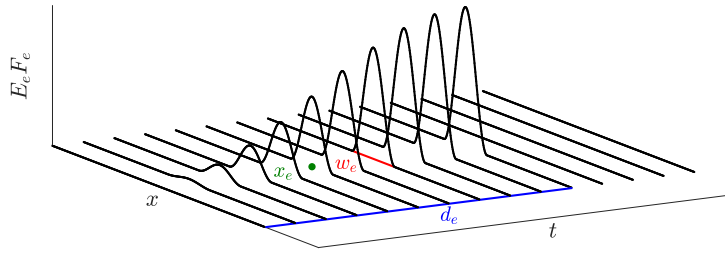


Fig. 3: A visualisation of the excitation used in our implementation presented in Equation (5). The location of excitation x_e is shown in green, excitation width w_e in red and excitation duration d_e in blue (also see Equations (25) and (26)).

4.3 Connections

Increasing $\omega_1 \gtrsim 100,000\ (\mathrm{m{\cdot}s})^{-1/2}$ while keeping $0 < \omega_0 \lesssim 100\ \mathrm{s}^{-1}$ will cause audible non-linear behaviour, such as pitch-glides and rattling sounds. These effects will be more dominant when the plate stiffness is higher. In our implementation we set $\omega_0 = 100\ \mathrm{s}^{-1}$ and $\omega_1 = 100,000\ (\mathrm{m{\cdot}s})^{-1/2}$. The spring-damping $\sigma_\times = 1\ \mathrm{s}^{-1}$ is kept to a minimum ($0 \leq \sigma_\times \leq 10\ \mathrm{s}^{-1}$).

4.4 System Architecture

The system architecture can be seen in Figure 4. The top box denotes the Sensel Morph (described in more detail in the next section) controlling the application, and the white boxes show the different classes or components of the application. The black arrows indicate instructions that one class can give to another and the hollow arrows show data flows between classes. All arrows are accompanied by a coloured box indicating which thread the instruction / dataflow is associated with and at what rate this thread runs.

The lowest priority thread, the graphics-thread, is shown by green boxes and runs at 15 Hz. This draws the states of the strings, connections and the plate on the screen.

Checking and retrieving the Sensel state happens at a rate of 150 Hz and is denoted by blue boxes. The parameters that the user controls by means of the Sensel, such as bowing position, force and velocity, will be updated in the models at this rate as well.

The highest priority thread is the audio-thread and runs at commonly-used sample rate 44,100 Hz. The main application gives an 'update' (u) instruction to the instrument, which in turn updates the FDSs in its strings and plate. After the FDS update is done, the intermediate state at the connection points $u^\mathrm{I}_{x_\mathrm{c}}$ (where $x_\mathrm{c} = l_{\mathrm{c},\alpha}$ for the string or $x_\mathrm{c} = (l_{\mathrm{c},\beta}, m_{\mathrm{c},\beta})$ for the plate) are sent to

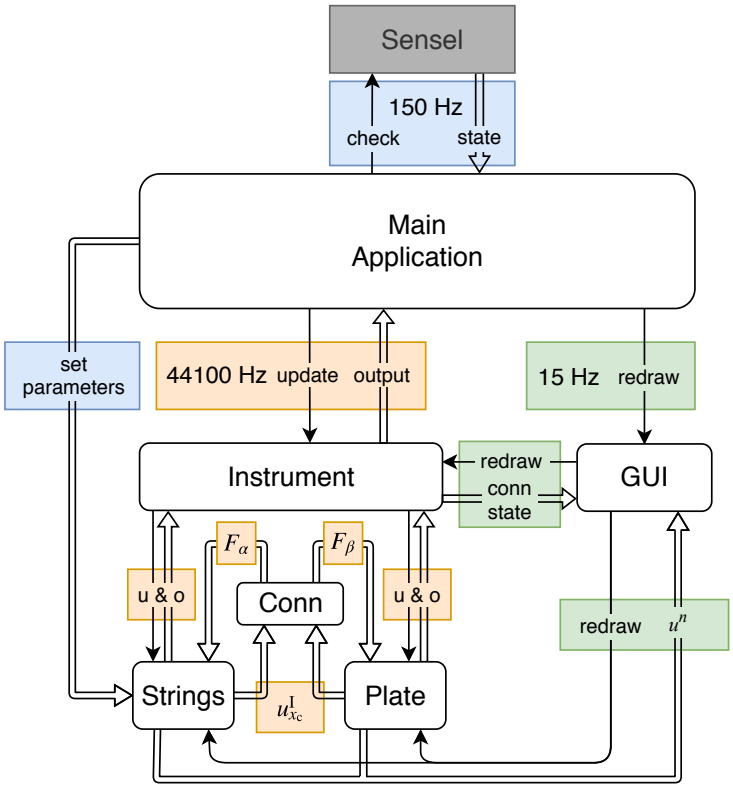


Fig. 4: System architecture flowchart. See Section 4.4 for a thorough explanation.

the connection (Conn) class which calculates the force-functions F_α and F_β. These values are then sent back to the string and plate classes and added to their respective states after which their outputs (o) (at arbitrary points) are sent back to the main application. See Algorithm 1 for this 'order of calculation'.

5 Instruments and User Interaction

In this section, the Sensel Morph (or simply Sensel) and user interface will be described in more detail. Furthermore, several configurations of strings, plates and connections that are inspired by real-life instruments will be presented. A demonstration of one of the instruments can be found in [22].

5.1 Sensel Morph

The Sensel Morph is a highly accurate touch controller that senses position and force of objects [17] (see Figure 5). We use the Sensel as an expressive interface

while *application runs* **do**
  for *all elements* **do**
    calculate intermediate state $\mathbf{u}^{\mathrm{I}}$ using previous state values (as in Equation (11))
      $\mathbf{u}^{\mathrm{I}}_{\mathrm{s}} = 2\mathbf{u}^{n} - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}$
  end
  if *element is excited/bowed* **then**
    calculate excitation term $\mathbf{E}$ and add to intermediate state of the element
      $\mathbf{u}^{\mathrm{I}}_{\mathrm{s+e}} = \mathbf{u}^{\mathrm{I}}_{\mathrm{s}} + \mathbf{E}$
  end
  for *all connections* **do**
    calculate connection forces and add connection term $\mathbf{C}$ to elements to obtain the state at the next time step
      $\mathbf{u}^{n+1}_{\mathrm{s+e+c}} = \mathbf{u}^{\mathrm{I}}_{\mathrm{s+e}} + \mathbf{C}$
  end
  update state vectors
    $\mathbf{u}^{n-1} = \mathbf{u}^{n}$
    $\mathbf{u}^{n} = \mathbf{u}^{n+1}_{\mathrm{s+e+c}}$
  increment time step
    n++
end

Algorithm 1: Pseudocode showing the correct order of calculation. The subscripts for state vector $\mathbf{u}$ shows what it consists of ('s' for previous state, 'e' for excitation and 'c' for connection).

for interacting with the instrument configurations. Right above the touch-sensitive area, the Sensel contains an array of 24 LEDs that can be controlled from the application.

5.2 User interface

Strings are shown as coloured paths (see Figure 6 for a descriptive visualisation). The state $\mathbf{u}^{n}$ of the string is visualised using the vertical displacement of the paths. Bowed strings are shown in cyan on the top left. The bow is shown as a yellow rectangle and moves on interaction. The fretting position is shown as a yellow circle. Plucked strings are shown in purple in the top right, underneath which the sympathetic strings are shown in light green. The plate is shown in the bottom using a grid of rectangles (clamped grid points are not shown). Its state is visualised using a grey-scale. Furthermore, connections are shown using orange circles/squares for the points of connection and dotted

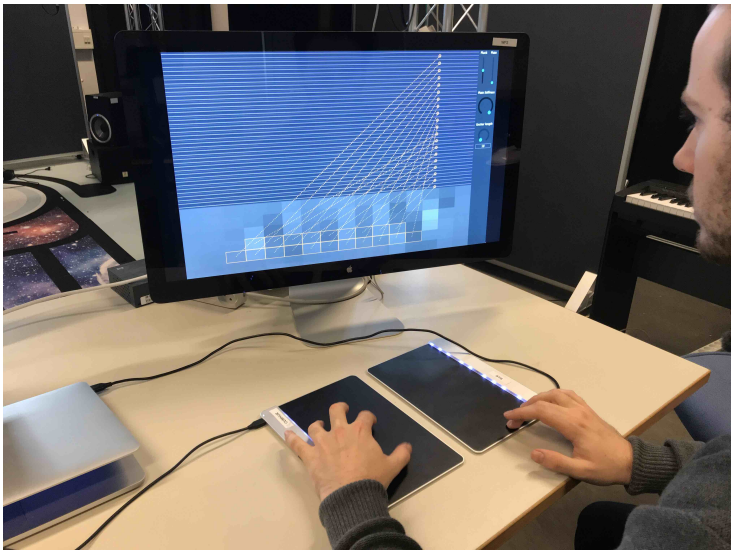


Fig. 5: Player using the Sensel Morphs to interact with one of the instruments.

lines between these points. Lastly, all parameters that are controlled by the mouse such as output-level and plate-stiffness are located in a column on the right side of the application.

5.3 Instruments

We subdivide string-elements into three types: bowed, plucked and sympathetic strings. All strings will be connected to one plate acting as an instrument body of which the user can control the plate-stiffness. Furthermore, the user can change the output-level of each element type. Apart from these parameters, which are controlled by the mouse, the instruments are fully controlled by two Sensels. The instruments we have chosen as our inspiration are the sitar, the hammered dulcimer and the hurdy gurdy.

5.3.1 Bowed Sitar

The sitar is originally an Indian string instrument that has both fretted strings and sympathetic strings. Instead of plucking the fretted strings, we extended the model to bow them. Our implementation consists of 2 bowed strings (tuned to A3 and E4), 13 sympathetic strings (tuned according to [23]) and 5 plucked strings (tuned A3-E4 following an A-major scale) as it is also possible to strum the sympathetic strings. See Figure 6 for a visual of the implementation. One Sensel is vertically subdivided into two sections; one for each bowed string. The first finger registered by the Sensel is mapped to a bow and the second

is mapped to a fretting finger controlling pitch. The horizontal position of both fingers is visualised using the Sensel's LED array. The frets are not implemented as such (the pitch is continuous), but they are visualised for reference. The horizontal position of the first finger is mapped to the bowing position on the string, the vertical velocity to the bow velocity v_{B} and the finger force is linked to the excitation function F_{B} (both in Equation (3a)). The other Sensel is subdivided into 5 sections mapped to the plucked strings. These sections are visualised by the LED array for reference.

The mass ratio for the bowed/plucked string to plate connections has been set to $\mathcal{M} = 2$ and ratio for the sympathetic string to plate connections has been set to $\mathcal{M} = 0.5$ to increase the effect that the playable strings have on the sympathetic strings.

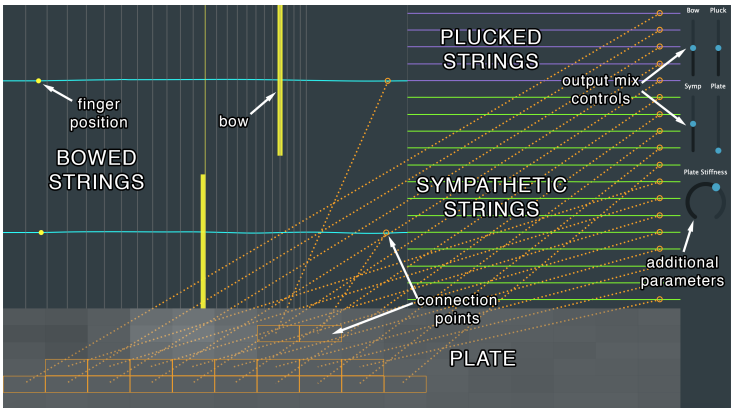


Fig. 6: The bowed sitar application. The descriptions of the different elements and other objects are shown in the image, but will (naturally) not be visible in the application.

5.3.2 Hammered Dulcimer

The hammered dulcimer is an instrument that can be seen as an 'open piano' where the musician has the hammers in their hand. Just like the piano, the strings are grouped in pairs or triplets that are played simultaneously. In our implementation, we have 20 pairs of plucked strings. Even though most hammered dulcimers have more strings, we decided that this configuration has the highest number of strings while maintaining playability. One of each pair is connected to the plate which slightly detunes it, creating a desired 'chorusing' effect. See Figure 7 for a visual of the implementation. In order for the excitation to more resemble a strike of a hammer than a pluck, the contents of the cosine in (26) will be multiplied by 2 for the excitation to have a less abrupt ending, something desired for a hammered interaction. Moreover, the excitation-length can be changed to simulate short and long hammer-times.

The Sensels are placed vertically next to each other (see Figure 5). The pair with the lowest frequency will then be located in the bottom right and the highest in the top left, as in the real instrument. As with the plucked strings of the bowed sitar, the LED array is used to visualise the way that the Sensel is subdivided, which is especially useful here as one Sensel controls 10 string-pairs.

The mass ratio is set relatively high ($\mathcal{M} = 100$) to amplify the non-linear interaction between the strings and the detuning of the strings connected to the plate.

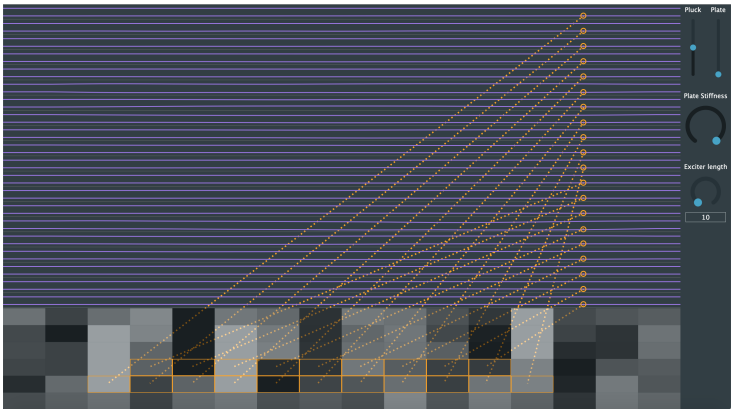


Fig. 7: The hammered dulcimer application.

5.3.3 Hurdy Gurdy

The hurdy gurdy is an instrument that consists of bowed and sympathetic strings. The bowing happens through a rosined wheel attached to a crank and bows these strings as the crank is turned. It is possible to change the pitch of a few bowed strings - the melody strings - using buttons that press tangent pins on the strings at different positions. The other strings, referred to as drone strings, are mostly tuned lower than the melody strings and provide the bass frequencies of the instrument. The musician can place the bowed strings on rests that keep the wheel from interacting with it.

Our implementation consists of 5 bowed strings subdivided into 2 drone strings tuned to A2, E3 and 3 melody strings tuned to A3, E4 and A4 and 13 sympathetic strings tuned the same way as the sympathetic strings in bowed sitar. Furthermore, the mass ratios have been set the same as in the bowed sitar application. See Figure 8 for a visual of the implementation.

The Sensel is vertically subdivided into 5 rows that control whether the strings are placed on the wheel. The bowing velocity is mapped to the average pressure of the fingers. The fundamental frequency (in the model $\gamma/2$) of the

melody-strings is changed by a Sensel with a piano-overlay acting as a midi controller. A demonstration of this instrument can be found in [22]. It is interesting to note here that the sympathetic strings that are in tune with the harmonics of the bowed strings resonate most, which is expected to happen in the real world as well.

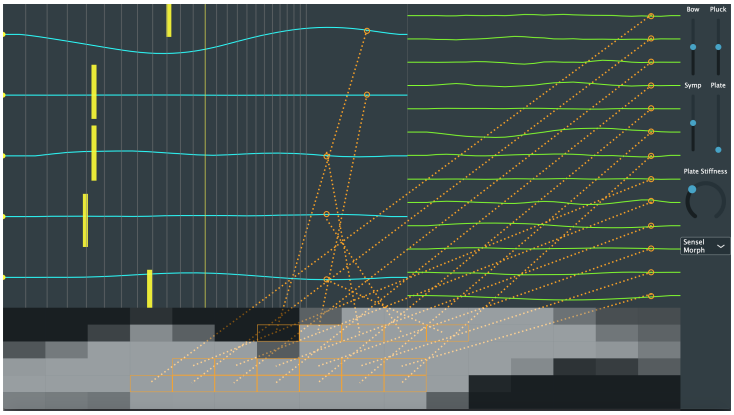


Fig. 8: The hurdy gurdy application.

6 Results and Discussion

Table 1 shows the CPU usage (on the same MacBook Pro 2.2 GHz i7 as described before) for the three instruments presented in the previous section. As the Sensel thread contributes a negligible amount to the CPU usage, this is not shown in the table.

Application	No Sound	No Graphics	Total
Bowed Sitar	32	63	85
Dulcimer	30	66	85
Hurdy Gurdy	28	58	78

Table 1: CPU usage (in %) for the instruments found in Section 5. Values show usage of one (virtual) thread and have been taken as an average (with a margin of ~5%) over a short period of time. The two middle columns show usage when the sound or graphics thread has been turned off.

As can be seen from the table, all instruments use about the same amount of CPU and none of them have audible dropouts (CPU $< 100\%$). It can be observed that the graphics use about 20% of the CPU, indicating that there is still much room to increase the complexity of the instrument-configurations before dropouts will occur. On the other hand, should the instruments be used

in parallel with other audio applications or plug-ins, the CPU usage has to be greatly reduced. The first step towards this would be to vectorise the FDSs using AVX instructions.

While our instruments have been not formally evaluated yet, we have performed some qualitative evaluations with sound and music computing experts. The goals of the evaluations were to explore the playability of the instrument, sonic quality and intuitiveness of control. These evaluations showed that especially the bowing interaction feels intuitive and creates a natural sound. The overall sound of the instruments was generally judged to be interesting, but not "sounding like a real-life instrument". This makes sense, as we did not seek to perfectly model each instrument, but rather used them as an inspiration for the configurations of the physical models. The next step for sound quality would be to replace the thin plate with a more realistic element, such as a wooden instrument body.

7 Conclusion

In this paper, a real-time modular physical modelling synthesis environment structured as a network of connected strings and plates has been presented. Several instruments have been created in the context of this environment which can be played by a pair of Sensel Morphs allowing for highly expressive control of these instruments. Informal evaluations with professional musicians have confirmed that the interaction is found natural and the output sound interesting. Further steps to improve this project are to optimise the algorithm and to replace the plate with a more realistic instrument body.

8 Acknowledgments

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

[1] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," 1979, thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France.

[2] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.

[3] C. Cadoz, A. Luciani, and J. L. Florens, "Cordis-anima: a modeling and simulation system for sound and image synthesis: the general formalism," *Computer music journal*, vol. 17, no. 1, pp. 19–29, 1993.

[4] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[5] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[6] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[7] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[8] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[9] R. Bacon and J. Bowsher, "A discrete model of a struck string," *Acustica*, vol. 41, pp. 21–27, 1978.

[10] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[11] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[12] S. Bilbao, *Numerical Sound Synthesis, Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley and Sons, Ltd, 2009.

[13] ——, "A modular percussion synthesis environment," *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, 2009.

[14] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis on gpu," in *Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference*, 2013.

[15] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," *Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx-15)*, 2015.

[16] K. Franinović and S. Serafin, *Sonic interaction design*. Mit Press, 2013.

[17] Sensel Inc. (2018) Sensel morph. [Online]. Available: https://sensel.com/

[18] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Applied Sciences*, 2016.

[19] K. Graff, *Wave Motion in Elastic Solids*. New York, New York: Dover, 1975.

[20] JUCE ROLI. (2019) JUCE. [Online]. Available: https://juce.com/

[21] J. Wallis, *A treatise of algebra, both historical and practical*. London, 1685.

[22] S. Willemsen. (2019) Hurdy gurdy demo. [Online]. Available: https://www.youtube.com/watch?v=BkxLji2ap1w

[23] Joe Rizzo. (2015) How to tune a sitar. [Online]. Available: http://www.joerizzo.com/sitar/

Paper B

Physical Models and Real-Time Control with the Sensel Morph

Silvin Willemsen, Stefan Bilbao, Nikolaj Andersson
and Stefania Serafin

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp.
95–96, 2019.

1 References

Paper C

Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao and Stefania Serafin

The paper has been published in the
Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), pp. 40–46, 2019.

Abstract

The simulation of a bowed string is challenging due to the strongly non-linear relationship between the bow and the string. This relationship can be described through a model of friction. Several friction models in the literature have been proposed, from simple velocity dependent to more accurate ones. Similarly, a highly accurate technique to simulate a stiff string is the use of finite-difference time-domain (FDTD) methods. As these models are generally computationally heavy, implementation in real-time is challenging. This paper presents a real-time implementation of the combination of a complex friction model, namely the elasto-plastic friction model, and a stiff string simulated using FDTD methods. We show that it is possible to keep the CPU usage of a single bowed string below 6 percent. For real-time control of the bowed string, the Sensel Morph is used.

1 Introduction

In physical modelling sound synthesis applications, the simulation of a bowed string is a challenging endeavour. This is mainly due to the strongly non-linear relationship between the bow and the string, through a model of friction. Such friction models can be categorised as static or dynamic; models of the latter type have only recently seen a major effort. As opposed to static friction models, where friction depends only on the relative velocity of the two bodies in contact, dynamic models describe the friction force through a differential equation.

A recently popular dynamic model is the elasto-plastic model, first proposed in [1]. The model assumes that the friction between the two objects in contact is caused by a large ensemble of bristles, each of which contributes to the total friction force. The average bristle deflection is used as an extra independent variable for calculating the friction force. As shown in [2], the elasto-plastic model can be applied to a bowed string simulation and it exhibits a hysteresis loop in the force versus velocity plane due to this multivariable dependency. This is consistent with measurements performed using a bowing machine in [3]. The elasto-plastic model has been thoroughly investigated in a musical context by Serafin et al. in [2, 4, 5].

Regarding bowed string simulations, the first musical non-linear systems, including bowed strings, were presented by McIntyre, et al. in [6]. Smith published the first real-time implementation of the bowed string using a digital waveguide (DW) for the string and a look-up table for the friction model in [7]. Simultaneously, Florens, et al. published a real-time implementation using mass-spring systems for the string and a static friction model for the bow in [8].

The dynamics of musical instruments are generally described by systems

of partial differential equations (PDEs). Specialised synthesis methods such as DWs [9] and modal synthesis [10] are derived from particular solutions. Mainstream time-stepping methods such as finite-difference time-domain (FDTD) methods were first proposed in [11, 12, 13], and developed subsequently [14, 15]. In [16] the authors adapted the thermal model proposed by Woodhouse in [3] for real-time applications using a DW for the string implementation and a combination of the DW and FDTD methods for the bowing interaction. In [17, 18], Desvages used FDTD methods for the implementation of the string in two polarizations and a static double exponential friction model introduced in [19]. This was, however, not implemented in real-time. To the best of the authors' knowledge, the only known real-time implementation of any bow model applied to complete FDTD strings was presented in [20] where the soft exponential friction function presented in [14] was used. The current work can be considered an extension of this work.

We are interested in bridging the gap between highly accurate physical models and efficient implementations so that these models can be played in real-time. In this work, we present an implementation of the elasto-plastic friction model in conjunction with a finite-difference implementation of the damped stiff string. Furthermore, we show that it is possible to play the string in real-time using the Sensel Morph controller [21].

This paper is structured as follows. In Section 2, the elasto-plastic bow model in conjunction with a PDE model for a stiff string is described. Discretisation is covered in Section 3, and implementation details appear in Section 4. In Section 5, simulated results are presented and discussed. Some concluding remarks appear in Section 6.

2 Elasto-Plastic Bow Model

Consider a linear model of transverse string vibration in a single polarization, where $u(x,t)$ represents string displacement as a function of time $t \geq 0$, in s, and coordinate $x \in [0,\, L]$ (in m) for some string length L (in m). Using the subscripts t and x to denote differentiation with respect to time and space respectively, a partial differential equation describing the dynamics of the damped stiff string is [14]

$$u_{tt} = c^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \tag{1}$$

Here, $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) with tension T (in N), material density ρ (in kg·m^{-3}) and cross-sectional area A (in m^2). Furthermore, $\kappa = \sqrt{EI/\rho A}$ is the stiffness coefficient (in m^2/s) with Young's Modulus E (in Pa) and area moment of inertia I (in m^4). For a string of circular cross section we have radius r (in m), cross-sectional area $A = \pi r^2$ and area moment of inertia

$I = \pi r^4/4$. Lastly, $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s) are coefficients allowing for frequency-independent and frequency-dependent damping respectively.

In our implementation we assume simply supported boundary conditions, which are defined as

$$u = u_{xx} = 0 \quad \text{where} \quad x = 0, L\,. \tag{2}$$

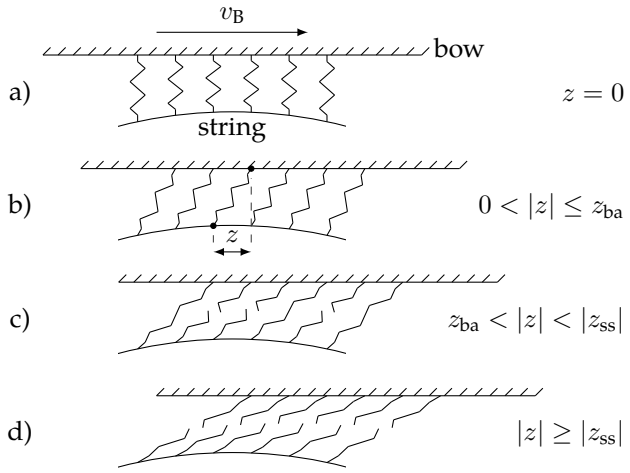


Fig. 1: *Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B. a) The initial state is where the average bristle displacement $z = 0$. b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (stick). c) After the break-away displacement z_{ba}, more and more bristles start to 'break'. This is defined as the elasto-plastic regime. d) After all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered.*

As mentioned in the introduction, the elasto-plastic bow model assumes that the friction between the bow and the string is due to a large ensemble of bristles, each of which contributes to the total friction force. See Figure 1 for a graphical representation of this. The bristles are assumed to be damped stiff springs and can 'break' after a given break-away displacement threshold. An extra term can be added to (1) to include the bowing interaction

$$u_{tt} = \ldots - \delta(x - x_\mathrm{B}) f(v, z)/\rho A. \tag{3}$$

Here, the spatial Dirac delta function $\delta(x - x_\mathrm{B})$ (in m^{-1}) allows for the pointwise application of the force f (in N) at externally supplied bowing position $x_\mathrm{B}(t)$ (in m).

In the following we will use the definitions found in [1]. The force f is defined in terms of the relative velocity v (in m/s) and average bristle displace-

ment z (in m) (see Figure 1) as

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{4}$$

where

$$v = u_t(x_{\mathrm{B}}) - v_{\mathrm{B}}, \tag{5}$$

where $v_{\mathrm{B}}(t)$ is an externally supplied bow velocity, s_0 is the bristle stiffness (in N/m), s_1 is the damping coefficient (in kg/s), s_2 is the viscous friction (in kg/s) and s_3 is a dimensionless noise coefficient multiplied onto pseudorandom function $w(t)$ (in N) as done in [4] and adds noise to the friction force. Here, $\dot{z}$ indicates a time derivative of z, and is related to v through

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_{\mathrm{ss}}(v)}\right], \tag{6}$$

where z_{ss} is the steady-state function

$$z_{\mathrm{ss}}(v) = \frac{\mathrm{sgn}(v)}{s_0}\left[f_{\mathrm{C}} + (f_{\mathrm{S}} - f_{\mathrm{C}})e^{-(v/v_{\mathrm{S}})^2}\right], \tag{7}$$

with Stribeck velocity v_{S} (in m/s), Coulomb force $f_{\mathrm{C}} = f_{\mathrm{N}}\mu_{\mathrm{C}}$ and stiction force $f_{\mathrm{S}} = f_{\mathrm{N}}\mu_{\mathrm{S}}$ (both in N). Here μ_{C} and μ_{S} are the dynamic and static friction coefficient respectively and $f_{\mathrm{N}}(t)$ is the normal force (in N) which is, like $v_{\mathrm{B}}(t)$, externally supplied. See Figure 2 for a plot of (7).

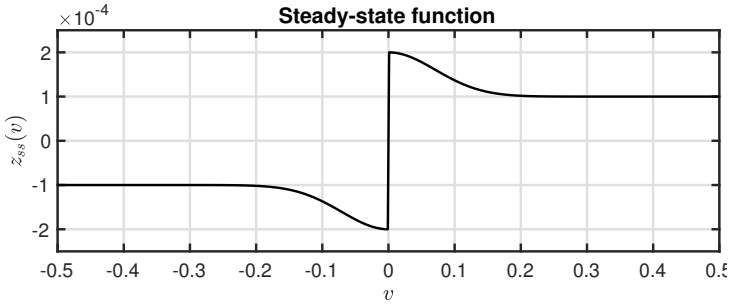


Fig. 2: *A plot of the steady-state function $z_{\mathrm{ss}}(v)$ with a force of 5 N.*

Furthermore, the adhesion map between the bow and the string is defined as

$$\alpha(v, z) = \begin{cases} \left.\begin{array}{ll} 0 & |z| \le z_{\mathrm{ba}} \\ \alpha_{\mathrm{m}}(v, z) & z_{\mathrm{ba}} < |z| < |z_{\mathrm{ss}}(v)| \\ 1 & |z| \ge |z_{\mathrm{ss}}(v)| \end{array}\right\} & \text{if } \mathrm{sgn}(v) = \mathrm{sgn}(z) \\ 0 & \text{if } \mathrm{sgn}(v) \neq \mathrm{sgn}(z), \end{cases} \tag{8}$$

where the transition between the elastic and plastic behaviour is defined as

$$\alpha_{\mathrm{m}} = \frac{1}{2}\left[1 + \mathrm{sgn}(z)\sin\left(\pi\frac{z - \mathrm{sgn}(z)\frac{1}{2}(|z_{\mathrm{ss}}(v)| + z_{\mathrm{ba}})}{|z_{\mathrm{ss}}(v)| - z_{\mathrm{ba}}}\right)\right], \tag{9}$$

with break-away displacement z_{ba}, i.e., where the bristles start to break (see Figure 1 c)). A plot of the adhesion map can be found in Figure 3.[1]

One of the difficulties in working with this model is that, due to the many approximations, the notion of an energy balance, relating the rate of stored energy in the system to power input and loss is not readily available. Such energy methods are used frequently in the context of physical modeling synthesis and virtual analog modeling as a means of arriving at numerical stability conditions for strongly nonlinear systems, as is the present case. See, e.g., [14]. This means that we do not have a means of ensuring numerical stability in the algorithm development that follows. This does not mean, however, that an energy balance is not available.

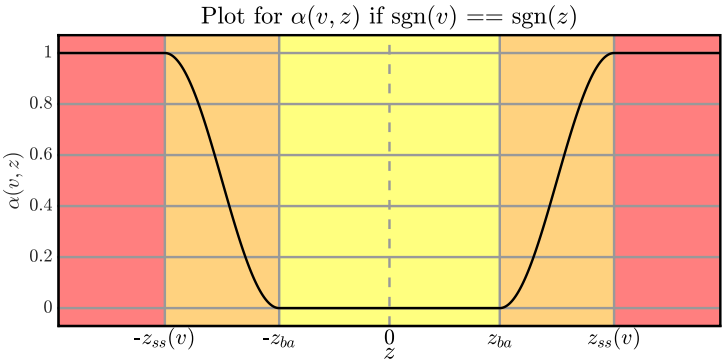


Fig. 3: *A plot of the adhesion map $\alpha(v,z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Figure 1 according to: yellow - a) & b), orange - c) and red - d).*

3 Discretisation

Finite-difference schemes for the stiff string in isolation are covered by various authors [13, 14].

Equation (1) can be discretised at times $t = nk$, with sample $n \in \mathbb{N}$ and time-step $k = 1/f_{\mathrm{s}}$ (in s) with sample-rate f_{s} (in Hz) and locations $x = lh$,

[1] It is interesting to note is that in the literature on this topic such as [1, 2, 4, 5], a few inaccuracies can be found in the definition of $\alpha(v,z)$: 1) all uses of z_{ss} in (8) and (9) lack the absolute value operator, 2) the multiplications with $\mathrm{sgn}(z)$ in (9) are excluded, 3) $\alpha(v,z)$ is undefined for $|z| = z_{\mathrm{ba}}$ and $|z| = |z_{\mathrm{ss}}(v)|$ (correct in the original paper by Dupont et al. [1]). It can be shown that only with the definitions presented here, is it possible to obtain the curve shown in Figure 3.

where grid spacing h (in m) needs to abide the following condition [14]

$$h \geq h_{\text{min}} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}} \tag{10}$$

and grid points $l \in [0, ..., N]$, where $N = \text{floor}(L/h)$ and $N + 1$ is the total number of grid points. It is important to note that the closer h is to h_{min}, the more accurate the scheme will be. Approximations for the derivatives found in (1) are described in the following way [14]:

$$u_t \approx \delta_{t\cdot} u_l^n = \frac{1}{2k}\big(u_l^{n+1} - u_l^{n-1}\big), \tag{11a}$$

$$u_{tt} \approx \delta_{tt} u_l^n = \frac{1}{k^2}\big(u_l^{n+1} - 2u_l^n + u_l^{n-1}\big), \tag{11b}$$

$$u_{xx} \approx \delta_{xx} u_l^n = \frac{1}{h^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n\big), \tag{11c}$$

$$u_{txx} \approx \delta_{t-}\delta_{xx} u_l^n = \frac{1}{hk^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}\big), \tag{11d}$$

$$u_{xxxx} \approx \delta_{xxxx} u_l^n = \frac{1}{h^4}\big(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\big), \tag{11e}$$

with grid function u_l^n denoting a discretised version of $u(x,t)$ at the nth time step and the lth point on the string. Note that in (11d), the backwards time difference operator is used to keep (12) explicit and thus computationally cheaper to update. Using the approximations shown in (11), (3) can be discretised to

$$\begin{aligned} \delta_{tt} u_l^n = {} & c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n \\ & + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n - J(x_{\text{B}}^n) f(v^n, z^n)/\rho A, \end{aligned} \tag{12}$$

where the relative velocity described in (5) can be discretised as

$$v^n = I(x_{\text{B}}^n)\delta_{t\cdot} u_l^n - v_{\text{B}}^n. \tag{13}$$

Here, $I(x_{\text{B}}^n)$ and $J(x_{\text{B}}^n)$ are weighting functions where the former interpolates the string displacement and velocity and the latter distributes the bowing term around time-varying bowing position x_{B}^n (see Figure 4 and [14] for more details on this). Furthermore,

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n \tag{14}$$

is the discrete counterpart of (4) where

$$r^n = r(v^n, z^n) = v^n \left[1 - \alpha(v^n, z^n)\frac{z^n}{z_{\text{ss}}(v^n)}\right] \tag{15}$$

is the discrete counterpart of (6).

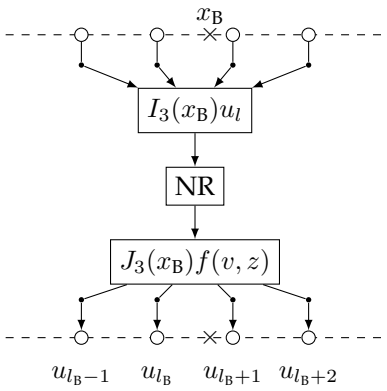


Fig. 4: *Cubic interpolation at bowing point x_B. The interpolator I retrieves the values of four grid points which are then used in the Newton-Raphson (NR) solver. This outputs the force function $f(v,z)$ that the spreading function J in turn distributes over the same four grid points. This process happens every single sample.*

At the bowing point we need to iteratively solve for two unknown variables: the relative velocity between the bow and the string v^n and the mean bristle displacement z^n of the bow at sample n. We can solve (12) at x_B^n using (13) and identity [14]

$$\delta_{tt} u_l^n = \frac{2}{k}\big(\delta_{t\cdot} u_l^n - \delta_{t-} u_l^n\big) \tag{16}$$

resulting in

$$I(x_\mathrm{B}^n) J(x_\mathrm{B}^n) f(v^n, z^n)/\rho A + \Big(\frac{2}{k} + 2\sigma_0\Big) v^n + b^n = 0, \tag{17}$$

where

$$\begin{aligned} b^n = {} & \frac{2}{k} v_\mathrm{B}^n - \frac{2}{k} I(x_\mathrm{B}^n)\delta_{t-} u_l^n - c^2 I(x_\mathrm{B}^n)\delta_{xx} u_l^n + \kappa^2 I(x_\mathrm{B}^n)\delta_{xxxx} u_l^n \\ & + 2\sigma_0 v_\mathrm{B}^n - 2\sigma_1 I(x_\mathrm{B}^n)\delta_{t-}\delta_{xx} u_l^n \end{aligned} \tag{18}$$

and can be pre-computed as its terms are not dependent on v^n or z^n. Recalling (4), this can be rewritten to

$$I(x_\mathrm{B}^n) J(x_\mathrm{B}^n) \left(\frac{s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n}{\rho A} \right) + \Big(\frac{2}{k} + 2\sigma_0\Big) v^n + b^n = 0. \tag{19}$$

To obtain the values of v^n and z^n, multivariate Newton-Raphson (NR) is

used. If (19) is defined to be $g_1 = g_1(v^n, z^n)$ and

$$g_2(v^n, z^n) = r^n - a^n = 0, \tag{20}$$

with

$$a^n = (\mu_{t-})^{-1}\delta_{t-}z^n \tag{21}$$

(where the operators applied to z^n denote the trapezoid rule [14]) we obtain the following iteration

$$\begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} = \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{22}$$

where i is the iteration number capped by 50 iterations, and the convergence threshold is set to 10^{-7}.

4 Implementation

In this section, we will elaborate on the implementation; the parameters used and the system architecture. The real-time implementation of the discrete-time model shown in the previous section has been done using C++ together with the JUCE framework [22]. The application is shown in Figure 5. The parameters we used can be found in Table 1, most of which are based on implementations by Serafin in [4]. These parameters will be static, i.e., are not user-controlled (except for z_{ba} and s_3 which rely on f_{N}). A demonstrative video can be found in [23]. We use the passivity condition proposed by

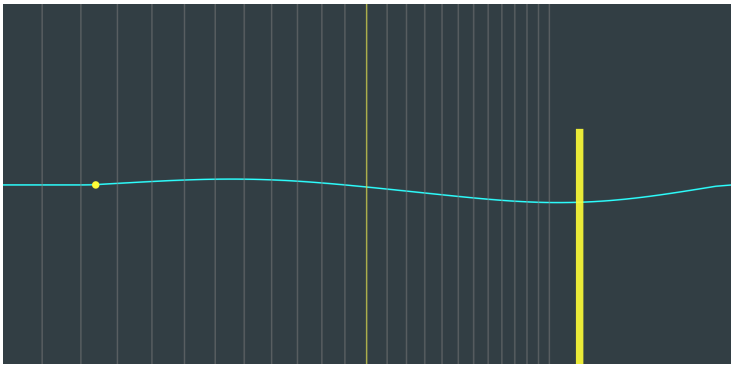


Fig. 5: *The elasto-plastic bowed string application. The bow is shown as a yellow rectangle, moves on interaction and its opacity depends on the finger force. The state* $\mathbf{u}^n$ *is visualised using the cyan curve and stopping-finger position is shown as a yellow circle. The grey lines show the 'frets' corresponding to semi-tones as a visual reference for the stopping position and do not influence the model.*

[24] for our choices of different parameter-values. As this condition applies to

Table 1: *Parameter values. Values for the fundamental frequency f_0 can be found in Section 5.*

Parameter	Symb. (unit)	Value (notes)
Material Density	ρ (kg·m^{-3})	7850
Radius	r (m)	$5 \cdot 10^{-4}$
String length	L (m)	1
Wave speed	c (m/s)	$2f_0/L$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq. indep. damping	σ_0 (s^{-1})	1
Freq. dep. damping	σ_1 (m^2/s)	$5 \cdot 10^{-3}$
Coulomb friction	μ_C (-)	0.3 ($< \mu_S$)
Static friction	μ_S (-)	0.8 ($> \mu_C$)
Normal force	f_N (N)	10
Bow velocity	v_B (m/s)	0.1
Bow position	x_B (m)	0.25
Stribeck velocity	v_S (m/s)	0.1
Bristle stiffness	s_0 (N/m)	10^4
Bristle damping	s_1 (kg/s)	$0.001\sqrt{s_0}$
Viscous friction	s_2 (kg/s)	0.4
Noise coefficient	s_3 (-)	$0.02f_N$
Pseudorandom func.	w (N)	$-1 < w < 1$
Break-away disp.	z_{ba} (m)	$0.7f_C/s_0$ ($< f_C/s_0$)
Sample rate	f_s (Hz)	44,100
Time step	k (s)	$1/f_s$

the LuGre model first proposed in [25, 26] from which the elasto-plastic model evolved, further investigation is required to conclude whether these conditions are identical for the elasto-plastic model.

4.1 Sensel Morph

As mentioned in Section 1, the Sensel Morph (or Sensel for short) is used as an interface to control the bowed string (see Figure 6). The Sensel is a highly sensitive touch controller containing ca. 20,000 pressure sensitive sensors that allow for expressive control of the implementation [21].

4.2 Interaction

The first finger the Sensel registers is linked to the following parameters: the normal force of the bow f_N (finger pressure), the bowing velocity v_B (vertical finger velocity) and bowing position x_B (horizontal finger position). The parameters are limited by the following conditions: $0 \leq f_N \leq 10$, $-0.3 \leq v_B \leq 0.3$ and $0 < x_B < L$. The second finger acts as a stopping finger on the string. As done in [20], for a string stopped at location $x_f \in [0, L]$ and $l_f = \text{floor}(x_f/h)$ we

Fig. 6: *The Sensel Morph: an expressive touch sensitive controller used for controlling the real-time elasto-plastic bowed string implementation.*

use

$$u_l^n = \begin{cases} 0, & l = l_\mathrm{f} - 1 \vee l = l_\mathrm{f} \\ (1 - \alpha_\mathrm{f}^\epsilon) u_l^n, & l = l_\mathrm{f} + 1 \\ u_l^n, & \text{otherwise} \end{cases} \tag{23}$$

where $\alpha_\mathrm{f} = x_\mathrm{f}/h - l_\mathrm{f}$ and $\epsilon = 7$ is a heuristic value that has been found to most linearly alter pitch between grid points.

4.3 System Architecture

Implementation of the scheme shown in (12) starts by expanding the operators shown in (11) and solving for the state at the next sample $\mathbf{u}^{n+1}$ where $\mathbf{u}$ is a vector containing the values for all grid points $l \in [0, ..., N]$.

An overview of the system architecture can be found in Figure 7. The three main components of the application are the Sensel controlling the application, the violin string class that performs the simulation and the main application class that moderates between these and the auditory and visual outputs. The black arrows indicate instructions that one of these components can give to another and the hollow arrows indicate data flows. Moreover, the arrows are accompanied by coloured boxes, depicting what thread the instruction or data flow is associated with and at what rate this runs.

The graphics thread has the lowest priority, is denoted by the green boxes and runs at 15 Hz. The redraw instruction merely retrieves the current string state $\mathbf{u}^n$ and bow and finger position and visualises this as shown in Figure 5.

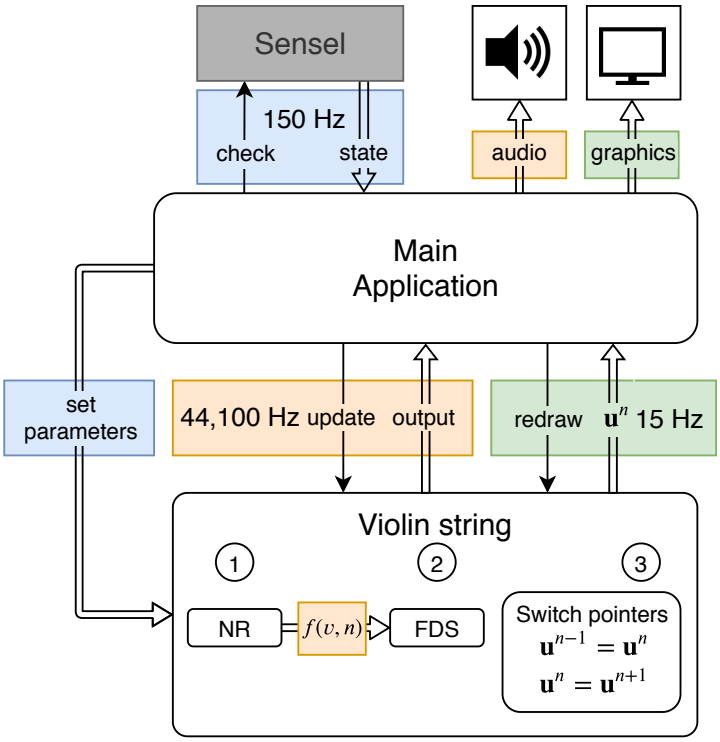


Fig. 7: *The system architecture. See Section 4.3 for a thorough explanation.*

The thread checking and receiving data from the Sensel runs at 150 Hz and is denoted by the blue boxes. The parameters that the user interacts with (bowing force, velocity and position) are also updated at this rate.

The highest priority thread is the audio thread denoted by the orange boxes and runs at 44,100 Hz. The violin string class gets updated at this rate and performs operations in the order shown in Algorithm 1.

5 Results and Discussion

Figure 8 shows the output waveforms for a string with $f_0 = 440$ Hz at different points along the string. The bowing parameters are $f_\mathrm{N} = 5\,\mathrm{N}$ and $v_\mathrm{B} = 0.1\,\mathrm{m/s}$. The figure shows the traditional Helmholtz motion, which is the characteristic motion of a bowed string.

To test whether the implementation exhibits a hysteresis loop, the force vs. relative velocity plane was visualised. In Figure 9, this plot can be found for which the same parameters have been used. The figure shows values for 500 samples around $t = 0.5 f_\mathrm{s}$. As can be seen from the figure, the hysteresis loop

for *t = 1:lengthSound* **do**
  calculate computable part b^n (Eq. (18))
  $\epsilon = 1$
  $i = 0$
  while $\epsilon < tol \wedge i < 50 \wedge f_C > 0$ **do**
    calculate..
    1. $z_{ss}(v^n_{(i)})$ (Eq. (7) in discrete-time)
    2. $\alpha(v^n_{(i)}, z^n_{(i)})$ (Eq. (8) in discrete-time)
    3. $r(v^n_{(i)}, z^n_{(i)})$ (Eq. (15))
    4. g_1, g_2 (Eqs. (19) and (20))
    5.–9. Compute derivatives of 1.–4. in the same order.
    10. Perform vector NR to obtain $v^n_{(i+1)}$ and $z^n_{(i+1)}$
    11. Calculate ϵ: $\epsilon = \left\| \begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} - \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} \right\|$
    12. Increment i: $i = i + 1$
  end
  Repeat 1.–3. using the values for v^n and z^n from the NR iteration.
  Calculate $f(v^n, z^n)$ (Eq. (14))
  Calculate $\mathbf{u}^{n+1}$ (Eq. (12) expanded)
  $\mathbf{u}^{n-1} = \mathbf{u}^n$
  $\mathbf{u}^n = \mathbf{u}^{n+1}$
end

Algorithm 1: Pseudocode showing the order of calculations.

is achieved and is similar to the one observed in [19]. The group of values around $v = 0$ are due to the sticking behaviour, and the others (the loop on the left) to the slipping behaviour.

For testing the speed of the algorithm, a MacBook Pro with a 2.2 GHz Intel Core i7 processor was used. The algorithm was tested using different frequencies according to the violin tuning of empty strings: $f_0 = 196.0$ (G3), 293.66 (D4), 440.0 (A4) and 659.26 (E5) Hz corresponding to $N = 95$, 71, 49, and 33 grid points respectively. The results can be seen in Table 2. When the total number of strings is smaller than 4, always the lowest frequency strings are used.

From Table 2 it can be observed that for one string, the CPU usage is $< 6\%$ with the graphics thread disabled. This is a great result, given the fact that both the bow and the string model are computationally complex. Empirical

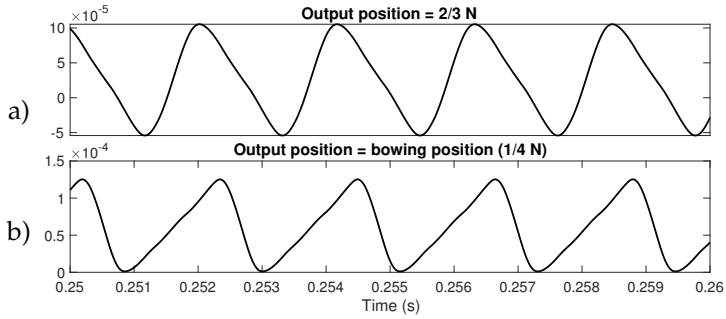


Fig. 8: *Output waveforms of the simulation at different positions along the string where N denotes the number of points of the string ($f_0 = 440$ Hz, $f_N = 5$ N and $v_B = 0.1$ m/s).*

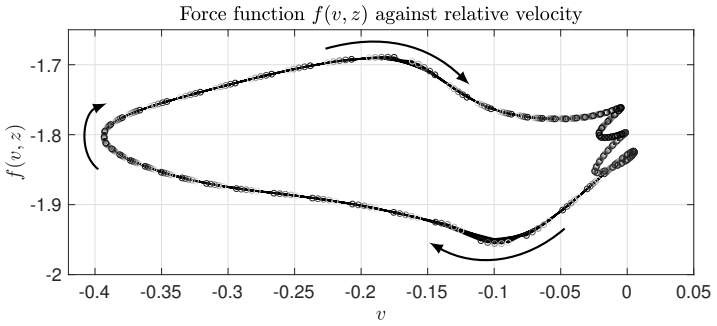


Fig. 9: *Hysteresis loop showing 500 values. The values around $v = 0$ are due to sticking behaviour and the loop on the left is due to slipping behaviour.*

investigation shows that the NR algorithm converges after ca. 3-4 iterations and the capping of 50 iterations never has to be used. A single string (but also more) could thus safely be used as an audio plugin in parallel to others without the user having to worry about auditory dropouts.

6 Conclusions

In this paper, we presented a real-time implementation of an elasto-plastic friction model with applications to a bow exciting a string, discretised using a finite-difference approach.

With a single string we are able to keep the CPU usage down to $< 6\%$ making for an efficient implementation that could be used in parallel with other virtual instruments or plugins.

Future work includes parameter design and including an instrument body for more realistic sounding results, as well as listening tests to verify the perceivable differences between simpler friction models versus the elasto-plastic

Table 2: *CPU usage for different amounts of strings. The values are averages over a 10 s period both for the enabled and disabled graphics thread. All strings are bowed simultaneously (polyphonically).*

Amount of strings	Graphics (%)	No graphics (%)
1	44.8	5.95
2	47.7	9.54
3	52.8	12.1
4	60.9	17.9

model.

7 Acknowledgments

Many thanks to the anonymous reviewers for giving their valuable input. This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8 References

[1] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter, "Single state elasto-plastic friction models," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 787–792, 2002.

[2] S. Serafin, F. Avanzini, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," *SMAC 03*, pp. 95–98, 2003.

[3] J. Woodhouse, "Bowed string simulation using a thermal friction model," *Acta Acustica united with Acustica*, vol. 89, pp. 355–368, 2003.

[4] S. Serafin, "The sound of friction: Real-time models, playability and musical applications," Ph.D. dissertation, CCRMA, 2004.

[5] F. Avanzini, S. Serafin, and D. Rocchesso, "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1073–1080, 2005.

[6] M. McIntyre, R. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments," *Journal of the Acoustical Society of America*, vol. 74, no. 5, pp. 1325–1345, 1983.

[7] J. Smith, "Efficient simulation of the reed bore and bow string mechanics," *ICMC 86*, pp. 275–280, 1986.

[8] J.-L. Florens, A. Razafindrakoto, A. Luciani, and C. Cadoz, "Optimized real time simulation of objects for musical synthesis and animated image synthesis," *ICMC 86*, pp. 65–70, 1986.

[9] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[10] J. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[11] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[12] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[13] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. a physical model for a struck string using finite difference methods," *JASA*, vol. 95, no. 2, pp. 1112–1118, 1994.

[14] S. Bilbao, *Numerical Sound Synthesis*. J. Wiley & Sons, 2009.

[15] S. Bilbao, R. Hamilton, B and. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," in *Springer Handbook of Systematic Musicology*. Springer, 2018, ch. 19, pp. 349–384.

[16] E. Maestre, C. Spa, and J. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC|SMC|2014*, pp. 1305–1311, 2014.

[17] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2017.

[18] C. Desvages and S. Bilbao, "Two-polarisation finite difference model of bowed strings with nonlinear contact and friction forces," *Proceedings of the International Conference on Digital Audio Effects*, 2015.

[19] J. Smith and J. Woodhouse, "The tribology of rosin," *Journal of the Mechanics and Physics of Solids*, vol. 48, pp. 1633–1681, 2000.

[20] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. of the 16th Sound and Music Computing Conference*, pp. 275–280, 2019.

[21] S. Inc., "Sensel Morph," accessed April 01, 2019, available at https://sensel.com/.

[22] J. ROLI, "JUCE," accessed April 04, 2019, available at https://juce.com/.

[23] S. Willemsen, "Elasto-Plastic Bow Model acting on a Finite-Difference Stiff String," accessed April 06, 2019, available at https://www.youtube.com/watch?v=-5bDebCW1Qg.

[24] K. J. Åström and C. Canudas de Wit, "Revisiting the lugre friction model," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 101–114, 2008.

[25] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "Dynamic friction models and control design," *Proceedings of the 1993 American Control Conference*, pp. 1920–1926, 1993.

[26] ——, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

Paper D

Real-time Implementation of a Physical Model of the Tromba Marina

Silvin Willemsen, Stefania Serafin, Stefan Bilbao and Michele Ducceschi

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 161–168, 2020.

Abstract

The tromba marina is a medieval bowed monochord instrument. The string of the instrument rests on a rattling bridge that, due to the collision with the body, creates a trumpet-like sound. This paper presents a real-time implementation of a physical model of the tromba marina. The goal of the simulation is to make the instrument accessible to a larger audience. The physical model is implemented using finite-difference time-domain (FDTD) methods and non-iterative collision methods. A real-time implementation of the instrument is also presented. The simulation exhibits brass-like qualities and sounds similar to a real tromba marina, but requires further testing to validate the realism.

1 Introduction

The tromba marina (see Figure 1) is a medieval bowed monochord instrument with a long quasi-trapezoidal body and a uniquely fashioned bridge (often called a shoe, because of its shape – see Figure 2). The name of the instrument derives from the fact that *tromba* means *trumpet* in Italian. A peculiarity of the instrument is that a foot of the bridge is free to rattle against the soundboard in sympathy with the vibrating string. This unusual bridge creates a trumpet-like sound. The frequency produced by the instrument is varied by placing the side of the knuckle of the non-dominant hand, lightly, at specific nodal points on the string, in order to select various harmonics of the open string. The dominant hand controls the bow, which is drawn across the string above the non-dominant hand [1].

In this paper, we present a real-time implementation of a physical model of the tromba marina. One of the ultimate goals is the emulation of an instrument that, due to its rarity, is not accessible to a large audience.

Physical modelling for sound synthesis has a long history. Various techniques have been developed to simulate real-world instruments, including mass-spring systems [2], digital waveguides [3] and modal synthesis [4]. Finite-difference time-domain (FDTD) methods were first used for sound synthesis by Hiller and Ruiz in [5, 6, 7], later by Chaigne et al. in [8, 9] and elaborated upon by Bilbao and colleagues in [10, 11]. Compared with other techniques, FDTD methods are more computationally expensive, but easily generalisable and flexible—no assumptions of linearity of travelling wave solutions are employed. Our goal is to implement these techniques in real time and thereby make the simulations playable for the users. For this purpose, we use the expressive Sensel Morph controller [12]. Other work in real-time control of FDTD methods using this controller includes [13].

The emulation of nonlinear collision interactions in musical instruments normally requires the use of iterative solvers (such as the Newton-Raphson

Fig. 1: The tromba marina from the Danish Music Museum in Copenhagen.

Fig. 2: The bridge of the tromba marina from the Danish Music Museum in Copenhagen. The right side is pressed against the body by the string while the left side is free and can rattle against the body.

algorithm) [14]. For the nonlinear collisions present in the instrument, a method recently proposed in the field of audio by Lopes and Falaize in [15, 16, 17] and later by Ducceschi and Bilbao in [18] allows such iterative methods to be sidestepped. It is thus suited to creating a real-time implementation of the tromba marina.

This paper is structured as follows: Section 2 presents the models used and Section 3 shows the discretisation of these. Section 4 provides information about implementation, parameter choices, the graphical user interface and control and mapping. Section 5 shows the results and discusses these. Concluding remarks and future work are presented in Section 6.

2 Models

The tromba marina can be subdivided into three main components: the string, the bridge and the body. In this section, the partial differential equations (PDEs) of the different components in isolation, under zero-input conditions, will be of the form

$$\mathcal{L}q = 0. \tag{1}$$

Here, $q = q(\boldsymbol{x}, t)$ represents the state of the component at time t and spatial coordinate $\boldsymbol{x} \in \mathcal{D}$, where the dimensions of domain $\mathcal{D}$ depend on the component at hand. Furthermore, $\mathcal{L}$ is a partial differential operator. (Subscripts 's', 'm' and 'p' used subsequently indicate that (1) applies to the string, bridge (mass) or body (plate), respectively.)

2.1 Bowed Stiff String

Consider a damped stiff string of length L (m), with domain $\mathcal{D} = \mathcal{D}_\mathrm{s} = [0, L]$ and state variable $q = u(\chi, t)$. With reference to (1), we define the operator $\mathcal{L} = \mathcal{L}_\mathrm{s}$ as [10]

$$\mathcal{L}_\mathrm{s} = \rho_\mathrm{s} A \partial_t^2 - T\partial_\chi^2 + E_\mathrm{s} I \partial_\chi^4 + 2\rho_\mathrm{s} A \sigma_{0,\mathrm{s}} \partial_t - 2\rho_\mathrm{s} A \sigma_{1,\mathrm{s}} \partial_t \partial_\chi^2. \tag{2}$$

Here, ∂_t and ∂_χ indicate partial differentiation with respect to t and χ. The various parameters appear as: material density ρ_s (kg·m^{-3}), cross-sectional area $A = \pi r^2$ (m^2), radius r (m), tension $T = (2f_{0,\mathrm{s}}L)^2 \rho_\mathrm{s} A$ (N),[1] fundamental frequency $f_{0,\mathrm{s}}$ (s^{-1}), Young's modulus E_s (Pa), area moment of inertia $I = \pi r^4/4$ (m^4), and loss coefficients $\sigma_{0,\mathrm{s}}$ (s^{-1}) and $\sigma_{1,\mathrm{s}}$ (m^2/s). We set the boundary

[1] Even though this definition for T from the fundamental frequency $f_{0,\mathrm{s}}$ is only valid for a simply supported string without stiffness, the effect of the stiffness eventually chosen for $f_{0,\mathrm{s}}$ is negligible.

conditions to be simply supported so that

$$u = \partial_\chi^2 u = 0 \quad \text{for} \quad \chi = 0, L. \tag{3}$$

As the string is excited using a bow, Equation (1) may be augmented as [10]

$$\mathcal{L}_\text{s} u = -\delta(\chi - \chi_\text{b}) F_\text{b} \Phi(v_\text{rel}), \tag{4}$$

with externally supplied downward bow force $F_\text{b} = F_\text{b}(t)$ (N), spatial Dirac delta function $\delta(\chi - \chi_\text{b})$ (m) selecting the bow position $\chi_\text{b} = \chi_\text{b}(t) \in \mathcal{D}_\text{s}$ (m) and dimensionless friction characteristic

$$\Phi(v_\text{rel}) = \sqrt{2a} v_\text{rel} e^{-a v_\text{rel}^2 + 1/2}, \tag{5}$$

with free parameter a. The relative velocity between the string at bow location χ_b and the externally supplied bow velocity $v_\text{b} = v_\text{b}(t)$ (m/s) is defined as

$$v_\text{rel} = \partial_t u(\chi_\text{b}, t) - v_\text{b}. \tag{6}$$

2.2 Bridge

The bridge is modelled as a simple mass-spring-damper system. As this system is point-like, or zero-dimensional, the state variable $q = w(t)$ and the definition of domain $\mathcal{D}$ is unnecessary. The operator $\mathcal{L} = \mathcal{L}_\text{m}$ is defined as

$$\mathcal{L}_\text{m} = M \frac{d^2}{dt^2} + M\omega_0^2 + MR\frac{d}{dt}, \tag{7}$$

with mass M (kg), linear angular frequency of oscillation $\omega_0 = 2\pi f_{0,\text{m}}$, ($\text{s}^{-1}$), fundamental frequency $f_{0,\text{m}}$ (s^{-1}) and damping coefficient R (s^{-1}).

2.3 Body

The body is simplified to a two-dimensional plate with side-lengths L_x and L_y, domain $\mathcal{D} = \mathcal{D}_\text{p} = [0, L_x] \times [0, L_y]$ and state variable $q = z(x, y, t)$. Using the 2D Laplacian

$$\Delta \triangleq \partial_x^2 + \partial_y^2, \tag{8}$$

the operator $\mathcal{L} = \mathcal{L}_\text{p}$ can be defined as [10]

$$\mathcal{L}_\text{p} = \rho_\text{p} H \partial_t^2 + D\Delta\Delta + 2\rho_\text{p} H \sigma_{0,\text{p}} \partial_t - 2\rho_\text{p} H \sigma_{1,\text{p}} \partial_t \Delta, \tag{9}$$

with material density ρ_p ($\text{kg}\cdot\text{m}^{-3}$), plate thickness H (m), stiffness coefficient $D = E_\text{p} H^3 / 12(1-\nu^2)$, Young's modulus E_p (Pa), dimensionless Poisson's ratio ν, and loss coefficients $\sigma_{0,\text{p}}$ (s^{-1}) and $\sigma_{1,\text{p}}$ (m^2/s). The boundary conditions of

the plate are set to be clamped so that

$$z = \mathbf{n} \cdot \nabla z = 0. \tag{10}$$

where ∇z is the gradient of z, and where $\mathbf{n}$ indicates a normal to the plate area at the boundary.

2.4 Collisions

It can be argued that the greatest contributor to the characteristic sound of the tromba marina is the rattling bridge colliding with the body. A diagram of the bridge with important parts highlighted can be found in Figure 3. A collision can be modelled by including a term to the PDEs mentioned above describing the potential energy of the system (further referred to as *the potential*) [18]. For the bridge-body (mass-plate) interaction this potential is defined as follows

$$\phi_{\text{mp}}(\eta_{\text{mp}}) = \frac{K_{\text{mp}}}{\alpha_{\text{mp}} + 1} [\eta_{\text{mp}}]_+^{\alpha_{\text{mp}}+1}, \tag{11}$$

$$K_{\text{mp}} > 0, \quad \alpha_{\text{mp}} \geq 1, \quad \eta_{\text{mp}} \triangleq z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t)$$

where K_{mp} is the collision stiffness (N/m if $\alpha_{\text{mp}} = 1$), α_{mp} is the dimensionless nonlinear collision coefficient, and $\eta_{\text{mp}} = \eta_{\text{mp}}(t)$ is the distance between the rattling part of the bridge and the body at the point of collision (m). Furthermore, $[\eta_{\text{mp}}]_+ = 0.5(\eta_{\text{mp}} + |\eta_{\text{mp}}|)$ is the positive part of η_{mp}. Note that penalty methods are employed here, where a positive η_{mp}, i.e., interpenetration of the colliding objects, is intended [19]. The term which can then be included in the PDEs is $\phi'_{\text{mp}} = d\phi_{\text{mp}}/d\eta_{\text{mp}}$. As described in [15, 16, 17, 18], using this form of the potential requires using iterative methods for solving its discrete

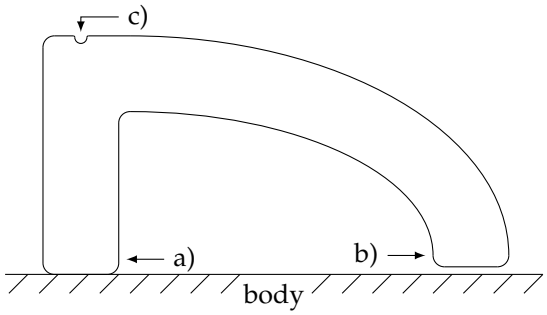


Fig. 3: Diagram of the bridge while rattling (view from top of the tromba marina). Indicated are: a) the pivoting point always in contact with the body, b) the rattling point colliding with the body (currently not colliding), and c) the string cavity straight above the middle of the pivoting point.

counterpart. In [18], the authors propose to rewrite the potential to

$$\psi = \sqrt{2\phi}, \tag{12}$$

and the term included in the PDEs to

$$\phi' = \psi\psi' = \psi\frac{d\psi}{d\eta} \xrightarrow{\text{chain rule}} \psi\frac{\dot{\psi}}{\dot{\eta}}\,, \tag{13}$$

where the dot above ψ and ϕ denotes a single time derivative. Equation (13), as can be seen in Section 3, leads to guaranteed stable and explicitly computable simulation algorithms without the need for iterative solvers.

As the string rests on the bridge, the interaction between these components needs to be modelled as well. Even though the bridge-body interaction is perpendicular to the string-bridge interaction, we can model them as being parallel, assuming that a “horizontal" movement of the string causes a “vertical" movement of the rattling part of the bridge. We can use an alternative version of the potential in Equation (11) described in [20] to make the collision two-sided acting as a connection:

$$\phi_{\text{sm}}(\eta_{\text{sm}}) = \frac{K_{\text{sm}}}{\alpha_{\text{sm}}+1}|\eta_{\text{sm}}|^{\alpha_{\text{sm}}+1}, \tag{14}$$

$$K_{\text{sm}} > 0, \quad \alpha_{\text{sm}} \geq 1, \quad \eta_{\text{sm}} \triangleq w(t) - u(\chi_{\text{sm}}, t)$$

where $\eta_{\text{sm}} = \eta_{\text{sm}}(t)$ is the distance between the string at the location of the bridge and the bridge itself.

2.5 Complete System

A complete system for the tromba marina may be written, in continuous-time as:

$$\begin{cases} \mathcal{L}_{\text{s}} u & = -\delta(\chi - \chi_{\text{b}})F_{\text{b}}\Phi(v_{\text{rel}}) \\ & \quad + \delta(\chi - \chi_{\text{sm}})\psi_{\text{sm}}\psi'_{\text{sm}} \\ \mathcal{L}_{\text{m}} w & = -\psi_{\text{sm}}\psi'_{\text{sm}} + \psi_{\text{mp}}\psi'_{\text{mp}}, \\ \mathcal{L}_{\text{p}} z & = -\delta(x - x_{\text{mp}}, y - y_{\text{mp}})\psi_{\text{mp}}\psi'_{\text{mp}}, \\ \eta_{\text{sm}} & = w(t) - u(\chi_{\text{sm}}, t), \\ \eta_{\text{mp}} & = z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t), \end{cases} \tag{15a–15e}$$

where $\chi_{\text{sm}} \in \mathcal{D}_{\text{s}}$ is the location of the bridge along the string and $(x_{\text{mp}}, y_{\text{mp}}) \in \mathcal{D}_{\text{p}}$ is the location on the body with which the bridge collides.

3 Discretisation

System (15) is discretised using FDTD methods. These methods subdivide the continuous system in grid points in space and samples in time. Before going into the discretisation of the models, and collision and connection terms in the system described in (15), some finite difference operators are introduced.

3.1 Operators

The identity and temporal shift operators are defined as

$$1\eta^n = \eta^n, \quad e_{t+}\eta^n = \eta^{n+1}, \quad e_{t-}\eta^n = \eta^{n-1}. \tag{16}$$

Using these, the operators for the forward, backward and centered time differences can be defined as

$$\delta_{t+} = \frac{e_{t+} - 1}{k}, \; \delta_{t-} = \frac{1 - e_{t-}}{k}, \; \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k}, \tag{17}$$

and are all approximations to a first-order time derivative. Furthermore, forwards and backwards averaging operators are defined as

$$\mu_{t+} = \frac{e_{t+} + 1}{2}, \quad \mu_{t-} = \frac{1 + e_{t-}}{2}. \tag{18}$$

and can be used to describe interleaved grid points $n + 1/2$ and $n - 1/2$ respectively.

3.2 Discrete Models

To approximate the state of a system in isolation we use

$$q(\boldsymbol{x}, t) \approx q_{\boldsymbol{l}}^n, \tag{19}$$

where grid function $q_{\boldsymbol{l}}^n$ is a discrete approximation to $q(\boldsymbol{x}, t)$ at $t = nk$ with time step k (s), time index $n \geq 0$ and grid location $\boldsymbol{l}$ that depends on domain $\mathcal{D}$ of the system at hand. In the case of the string, we use $\chi = lh_\mathrm{s}$ with grid spacing h_s (m), $\boldsymbol{l} = l \in [0, \ldots, N]$ and total number of grid points $N = L/h_\mathrm{s}$ to yield $u(\chi, t) \approx u_l^n$.

In the case of the body, we use $x = lh_\mathrm{p}$ and $y = mh_\mathrm{p}$ to get $z(x, y, t) \approx z_{(l,m)}^n$ where $\boldsymbol{l} = (l, m)$ with $l \in [0, \ldots, N_x]$ and $m \in [0, \ldots, N_y]$. Here, $N_x = L_x/h_\mathrm{p}$ and $N_y = L_y/h_\mathrm{p}$ are the horizontal and vertical number of grid points respectively with grid spacing h_p (m).

The discretisation of and expansion of operator $\mathcal{L} \approx \ell$ in the case of stiff strings, mass-spring systems and plates using FDTD methods are well covered

in the literature [10] and will not be described in detail in this paper. To obtain the highest accuracy possible while keeping the system explicit (except for the bow), centered differences – which are second-order accurate – have been chosen where possible.

For stability, grid spacings h_s and h_p should satisfy the conditions below. In the case of the damped stiff string,

$$h_\mathrm{s} \geq \sqrt{\frac{c^2k^2 + 4\sigma_{1,\mathrm{s}}k + \sqrt{(c^2k^2 + 4\sigma_{1,\mathrm{s}}k)^2 + 16\kappa_\mathrm{s}^2k^2}}{2}}, \tag{20}$$

with wave speed $c = \sqrt{T/\rho_\mathrm{s}A}$ and stiffness coefficient $\kappa_\mathrm{s} = \sqrt{E_\mathrm{s}I/\rho_\mathrm{s}A}$ and in the case of the plate,

$$h_\mathrm{p} \geq 2\sqrt{k\left(\sigma_{1,\mathrm{s}} + \sqrt{\kappa_\mathrm{p}^2 + \sigma_{1,\mathrm{s}}^2}\right)}, \tag{21}$$

with stiffness coefficient $\kappa_\mathrm{p} = \sqrt{D/\rho_\mathrm{p}H}$. The closer the grid spacings are to these conditions, the higher the accuracy of the approximation.

In order to discretise the Dirac delta functions found in system (15) we introduce a spreading operator $J(\boldsymbol{x}_\mathrm{c})$ that applies a force to coordinate $\boldsymbol{x}_\mathrm{c}$, which, in the simplest case, is defined as [10]

$$J(\boldsymbol{x}_\mathrm{c}) = \begin{cases} \frac{1}{h^d}, & \boldsymbol{l} = \boldsymbol{l}_\mathrm{c} = \mathrm{round}(\boldsymbol{x}_\mathrm{c}/h) \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

Here, d is the number of dimensions of domain $\mathcal{D}$ that $\boldsymbol{x}$ is defined for, i.e., $d = 0$ for the bridge, $d = 1$ for the string, and $d = 2$ for the plate. For finer control, a cubic spreading operator J_3 can be introduced [10]. This is used for the bowing term in Equation (4), which is discretised as follows

$$\ell_\mathrm{s} u_l^n = -J_3(\chi_\mathrm{b}) F_\mathrm{b}^n \phi(v_\mathrm{rel}^n) \tag{23}$$

where, using the centered difference operator from Equation (17),

$$v_\mathrm{rel}^n = \delta_{t\cdot} u_{l_\mathrm{b}}^n - v_\mathrm{b}^n, \tag{24}$$

with coordinate $l_\mathrm{b} = \chi_\mathrm{b}/h_\mathrm{s}$. Equation (24) needs to be calculated using iterative methods.

3.3 Collisions using Non-Iterative Methods

For the discrete-time definitions of the potential in (13) we can use

$$\psi \approx \mu_{t+}\psi^{n-1/2} \quad \text{and} \quad \psi' \approx \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}, \tag{25}$$

where ψ at interleaved grid point $n-1/2$ is defined as

$$\psi^{n-1/2} = \mu_{t-}\psi^n. \tag{26}$$

Note that applying a forward or backward difference operator to an interleaved grid – such as $\delta_{t+}\psi^{n-1/2}$ in Equation (25) – is second-order accurate.

For a system that has a single (upward) collision we get

$$\ell q_{\boldsymbol{l}}^n = J(\boldsymbol{x}_\mathrm{c})\left(\mu_{t+}\psi^{n-1/2}\right)\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}. \tag{27}$$

Here, we use the identity

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}\delta_{t+}\psi^{n-1/2} - \psi^{n-1/2} \tag{28}$$

and define

$$g^n = \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}, \tag{29}$$

which can be rewritten to

$$\delta_{t+}\psi^{n-1/2} = g^n\delta_{t\cdot}\eta^n. \tag{30}$$

Then, inserting (30) into (28) and this together with (29) into (27) we get

$$\ell q_{\boldsymbol{l}}^n = J(\boldsymbol{x}_\mathrm{c})\left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n - \psi^{n-1/2}\right)g^n \tag{31}$$

where g^n may be explicitly calculated using the analytic expressions for ψ and ϕ [18]:

$$g^n = \psi'\Big|_{\eta=\eta^n} = \frac{\phi'}{\sqrt{2\phi}}\Big|_{\eta=\eta^n}. \tag{32}$$

Numerical stability of this scheme is shown in [18]. When writing out (32) we can obtain definitions for g_sm^n using (14)

$$g_\mathrm{sm}^n = \mathrm{sgn}(\eta_\mathrm{sm}^n)\sqrt{\frac{K_\mathrm{sm}(\alpha_\mathrm{sm}+1)}{2}}|\eta_\mathrm{sm}^n|^{\frac{\alpha_\mathrm{sm}-1}{2}}, \tag{33}$$

and g^n_{mp} using (11)

$$g^n_{\text{mp}} = \sqrt{\frac{K_{\text{mp}}(\alpha_{\text{mp}}+1)}{2}}[\eta^n_{\text{mp}}]_+^{\frac{\alpha_{\text{mp}}-1}{2}} \ . \tag{34}$$

3.4 Complete Discrete System

Introducing for brevity,

$$\xi^n = \frac{k}{2} g^n \delta_{t\cdot}\eta^n - \psi^{n-1/2}, \tag{35}$$

the discrete counterpart of the complete system described in (15) will be

$$\begin{cases} \ell_{\text{s}} u^n_l & = -J_3(\chi^n_{\text{b}}) F_{\text{b}} \Phi(v^n_{\text{rel}}) + J(\chi_{\text{sm}}) \xi^n_{\text{sm}} g^n_{\text{sm}}, & \text{(36a)} \\ \ell_{\text{m}} w^n & = -\xi^n_{\text{sm}} g^n_{\text{sm}} + \xi^n_{\text{mp}} g^n_{\text{mp}}, & \text{(36b)} \\ \ell_{\text{p}} z^n_{(l,m)} & = -J(x_{\text{mp}}, y_{\text{mp}}) \xi^n_{\text{mp}} g^n_{\text{mp}}, & \text{(36c)} \\ \eta^n_{\text{sm}} & = w^n - u^n_{l_{\text{sm}}}, & \text{(36d)} \\ \eta^n_{\text{mp}} & = z^n_{(l_{\text{mp}}, m_{\text{mp}})} - w^n, & \text{(36e)} \end{cases}$$

where discrete counterparts of connection and collision locations in Equations (36d) and (36e) are described as $l_{\text{sm}} = \chi_{\text{sm}}/h_{\text{s}}$ and $(l_{\text{mp}}, m_{\text{mp}}) = (x_{\text{mp}}/h_{\text{p}}, y_{\text{mp}}/h_{\text{p}})$. This leaves us with two different types of update equations, one where $q^{n+1}_{\boldsymbol{l}}$ is calculated and one where $\psi^{n+1/2}$ is calculated.

One might think that due to the centered differences $\delta_{t\cdot}\eta^n$ still present in Equation (35), our system remains implicit, but as we can insert the definitions for Equations (36d) and (36e) evaluated at the next time index $n+1$, which are already present in $\ell_{\text{s}} u^n_l$, $\ell_{\text{m}} w^n$ and $\ell_{\text{p}} z^n_{(l,m)}$, the Equations in (36) reduce to a system of linear equations that can be solved by a single division.

4 Implementation

The real-time implementation of the system has been done in C++ using the JUCE framework [21] and will be controlled using the Sensel Morph (or simply Sensel) – an expressive touch controller. A demo of the application can be found in [22]. This section will first elaborate some important considerations regarding the setup of the system. Then, the algorithm together with the parameter design will be presented. Finally, the graphical user interface (GUI) will be detailed together with the Sensel and its mapping to the application.

4.1 Introducing an Offset

Firstly, for more realistic and expressive sounds, we model the bridge – and with that, the string – to rest slightly above the body. Expanding $\ell_{\mathrm{m}} w^n$ in (36b) and including the offset yields

$$\ell_{\mathrm{m}} w^n \Rightarrow M\delta_{tt} w^n + M\omega_0^2 (w^n - w_{\mathrm{off}}) + MR\delta_{t\cdot} w^n \tag{37}$$

where $w_{\mathrm{off}} \geq 0$ is a predefined offset between the body and the bridge. Furthermore, the second-order time derivative can be defined from the definitions in (17) as

$$\delta_{tt} = \delta_{t+}\delta_{t-} \ . \tag{38}$$

The boundary condition of the string defined in Equation (3) will also change depending on the bridge offset:

$$u = w_{\mathrm{off}} \quad \text{and} \quad \partial_\chi^2 u = 0. \tag{39}$$

4.2 Pitch Control

Secondly, as briefly mentioned in Section 1, the way that different pitches are played on the tromba marina, is to slightly rest a knuckle or finger on nodal points along the string to induce harmonics. Thus, a damping finger is implemented. Using the cubic interpolation operator I_3 [10], Equation (36a) can be extended to

$$\ell_{\mathrm{s}} u_l^n = \ldots - J_3(\chi_{\mathrm{f}}) I_3(\chi_{\mathrm{f}}) \sigma_{\mathrm{f}} (u_l^n - w_{\mathrm{off}}), \quad \text{where} \quad 0 \leq \sigma_{\mathrm{f}} \leq 1, \tag{40}$$

which essentially subtracts its own state at location $\chi_{\mathrm{f}} \in [0, 0.5\chi_{\mathrm{sm}}]$ according to the damping coefficient σ_{f} (kg$\cdot$ s^{-2}) applied. As done in [13], the fractional part used in the spreading operator ($\alpha_{\mathrm{i}} = \chi_{\mathrm{f}}/h - \mathrm{floor}(\chi_{\mathrm{f}}/h)$) is raised to the 7th power as it has been found to scale finger position to pitch more properly in the context of FDTD. As the string is bowed above the damping finger (at the other side of the rattling bridge) it is essential that the energy from the bow reaches the rattling bridge, which is still the case for lower values of σ_{f}. A more realistic approach that could be investigated is to model the finger as a mass colliding with the string, rather than imposing the damping directly to the state of the string as presented here.

A schematic plot of the full system, including the offset described in Equations (37) and (39) and the damping finger from Equation (40) can be found in Figure 4.

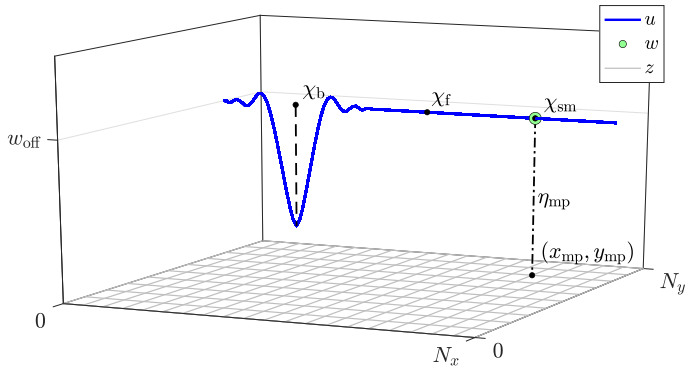


Fig. 4: The virtual system in (36) including the offset in Equation (37) and the damping finger in Equation (40), with different important coordinates highlighted. Note that η_{sm} (Equation (36d)) is not shown as it is close to 0 at all times.

4.3 Other Considerations

Realistic initialisation of both η_{sm} and η_{mp} is essential. In this case (at $n = 0$) $\eta_{sm}^0 = 0$ and $\eta_{mp}^0 \leq 0$ so that no collision is present at initialisation.

After h_p is calculated in Equation (21), we check whether it is smaller than a set value $h_{p,min} = 0.01$. This reduces the quality of the model, but increases the speed, ultimately allowing for real-time implementation.

4.4 Order of Calculation

The order of calculation is shown in the pseudocode in Algorithm 1. In theory, in order to iteratively calculate the bow force, the collision and connection forces should be included in this. However, as the string is practically never bowed at the bridge position χ_{sm}, these can be calculated independently.

4.5 Parameter Design

The list of parameters used in the implementation can be found in Table 1. As the authors had a real (recreated) tromba marina (presented in [23]) at their disposal, some parameters have been measured in accordance to the real instrument. The others have been tuned by ear by one of the authors.

Regarding the output of the system, through informal testing it was decided to retrieve the output from the state of the plate right at the point of collision $z_{out} = (l_{mp}, m_{mp})$ combined with the sound of the string at $u_{out} = L - \chi_{sm}$ at a lower volume. It can be argued that the loudest sound comes from the collision between the bridge and the body making it logical to select this point as the main sound source.

while *application is running* **do**

1. calculate schemes	(ℓq in Eqs. (36a-c))
2. apply bow to string	(Eq. (36a))
3. apply damping finger	(Eq. (40))
4. calculate g^n_{sm} and g^n_{mp}	(Eqs. (33) and (34))
5. calculate collision and connection forces and add to schemes	(Eqs. (36a-c))
6. Update states	$\boldsymbol{q}^{n-1} = \boldsymbol{q}^n$ $\boldsymbol{q}^n = \boldsymbol{q}^{n+1}$ $\boldsymbol{\psi}^{n-1/2} = \boldsymbol{\psi}^{n+1/2}$

end

Algorithm 1: Pseudocode showing the order of calculation after initialisation. Bold symbols denote the collection of states of the entire system ($\boldsymbol{q}$) and potentials ($\boldsymbol{\psi}$).

4.6 Graphical User Interface

A screenshot of the GUI is shown in Figure 5. The GUI is divided in four sections, three showing the states of the string, bridge and body respectively and one control section.

Firstly, the string section shows the state of the string u as a cyan-coloured path and the bow as a yellow rectangle with bow position χ_{b} and its opacity depending on the bow force F_{b}. Furthermore, the bridge state w^n is shown as a green circle at location (of the bridge along the string) χ_{sm}. Finally, the position of the damping finger χ_{f} is displayed as a yellow circle, the size of which depends on damping coefficient σ_{f}. The position of the finger triggers lines showing the locations of the closest nodes along the string according to the following equation

$$\chi^i_{\text{node}} = \frac{i \cdot \chi_{\text{sm}}}{n} \quad \text{for} \quad i = [1, \ldots, n-1], \tag{41}$$

where $n = \text{round}(\chi_{\text{sm}}/\chi_{\text{f}})$ is an integer closest to the ratio between the string length until the bridge location and the damping finger position. These lines are drawn to help the user place the damping finger at nodes along the string.

Secondly, the bridge section shows the displacement of the bridge w as a green circle, the state of the body at the collision location $z_{(l_{\text{sm}}, m_{\text{sm}})}$, both moving vertically according to their respective displacements and finally, a

Name	Symbol (unit)	Value
String		
Length	L (m)	1.90*
Material density	ρ_s (kg·m^{-3})	7850
Radius	r (m)	0.0005
Fundamental freq.	f_0 (s^{-1})	32*
Young's modulus	E_s (Pa)	$2 \cdot 10^{11}$
Freq. indep. loss	$\sigma_{0,\mathrm{s}}$ (s^{-1})	0.1
Freq. dep. loss	$\sigma_{1,\mathrm{s}}$ (m^2/s)	0.05
Bow		
Bow force	F_b (N)	$0 \leq F_\mathrm{b} \leq 0.1$
Bow velocity	v_b (m/s)	$-0.5 \leq v_\mathrm{b} \leq 0.5$
Free parameter	a (-)	100
Bridge		
Mass	M (kg)	0.001
Fundamental freq.	$f_{0,\mathrm{m}}$ (s^{-1})	500
Damping	R (s^{-1})	0.05
Body		
Length	L_x (m)	1.35*
Width	L_y (m)	0.18*
Material density	ρ_p (kg·m^{-3})	50
Thickness	H (m)	0.01
Young's modulus	E_p (Pa)	$2 \cdot 10^5$
Poisson's ratio	ν (-)	0.3
Freq. indep. loss	$\sigma_{0,\mathrm{p}}$ (s^{-1})	2
Freq. dep. loss	$\sigma_{1,\mathrm{p}}$ (m^2/s)	0.05
Min. grid spacing	$h_{\mathrm{p,min}}$ (m)	0.01
String-bridge connection		
Stiffness coefficient	K_sm (N/m)	$5 \cdot 10^6$
Nonlin. col. coeff.	α_sm (-)	1
Bridge location	χ_sm (m)	1.65*
Bridge-body collision		
Stiffness coefficient	K_mp (N/m)	$5 \cdot 10^8$
Nonlin. col. coeff.	α_mp (-)	1
Bridge location	$(x_\mathrm{mp}, y_\mathrm{mp})$ (m,m)	$(1.08, 0.135)$*
Other		
Offset	w_off (m)	$5 \cdot 10^{-6}$
Damp. finger coeff.	σ_f (kg·s^{-2})	$0 \leq \sigma_\mathrm{f} \leq 1$
Output loc. string	u_out (m)	$L - \chi_\mathrm{sm}$
Output loc. body	z_out (m, m)	$(x_\mathrm{mp}, y_\mathrm{mp})$

Table 1: List of parameter values used for the simulation. *These values have been taken from a real (recreated) tromba marina [23].

static grey horizontal line denoting the offset w_off, i.e., the resting position of the bridge.

Thirdly, the body section shows the state of the body z as a grid of rectangles

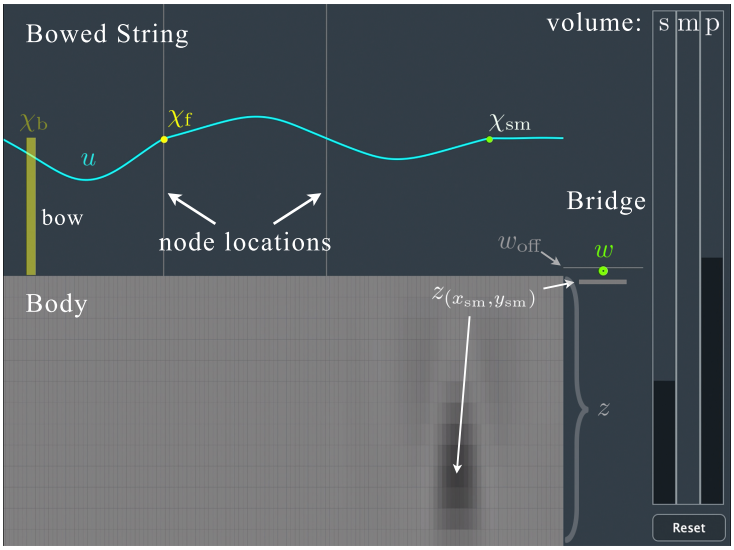


Fig. 5: The GUI showing the excited system with components highlighted. A more detailed description can be found in Section 4.6.

changing (grey-scale) colour according to their displacement.

Finally, the control section contains three sliders that control the volume-levels of the string (s), bridge (m) and body (p) respectively (for experimentation of volume ratios between the components) and a reset button to re-initialise the system.

4.7 Sensel Morph and Mapping

The Sensel is an expressive touch controller using ~20,000 pressure-sensitive sensors laid out in an hexagonal grid [12]. It retrieves x and y-positions and pressure at a rate of 150 Hz from which velocities and accelerations can be obtained.

The first finger registered by the Sensel is mapped to the bow: x-position is mapped to bow position χ_b, y-velocity to bow velocity v_b (y-position is shown in the GUI but does not influence the model directly) and pressure to bow force F_b. The second finger is mapped to the damping finger: x-position is mapped to finger location x_f and pressure to damping coefficient σ_f.

5 Results and Discussion

Informal listening by the authors has confirmed that the sound has brass-like qualities and comparison with the recreated tromba marina showed that the

sound exhibited similar qualities. Naturally, formal listening tests need to be conducted to verify this.

Disabling the graphics of the application, its CPU usage is 68.9% on a MacBook Pro with a 2.2 GHz Intel i7 processor, easily allowing it to work in real-time. As the heaviest part of the algorithm is the calculation of the body, the minimum grid spacing $h_{\text{p,min}}$ could be set to a higher value to decrease the CPU usage. However, as mentioned, this will decrease the quality of the output sound.

Through using the application, the authors found some odd behaviour, where the bridge 'gets stuck' behind the plate, i.e., values for ψ_{mp} would be negative for a short period of time (one to several samples). The explicit technique used in this work allows for this to happen (and can be proven to still be stable in this case [18]), but it is 'unphysical' to have a negative potential as this implies a 'pulling' collision. As can be seen from Table 1, the nonlinear collision coefficients α_{sm} and α_{mp} are set to 1. When increasing these values, this behaviour would arise much more often, and even occur for a prolonged period of time (several seconds to indefinitely). This is also the reason why the reset button presented in Section 4.6 has been implemented. As mentioned in [18], oversampling increases the accuracy of the explicit collision method, and could be a solution to this issue. However, in order for the application to run in real time, this solution can not be afforded without decreasing the quality of the implementation, e.g. increasing $h_{\text{p,min}}$. Further investigation will be necessary to solve this issue without oversampling.

Lastly, it has been found that when $|z^n_{(l,m)}| \lessapprox 10^{-306}$ (but non-zero) for any coordinate (l, m) (which happens when the body has not been collided with for a prolonged period of time), the CPU usage increases considerably. This could be explained by the fact that calculations with extremely small values are handled differently by the application. This is solved by implementing a limit to how small a value for $z^n_{(l,m)}$ can be. If the value of $z^n_{(l_{\text{sm}},m_{\text{sm}})}$ is lower than this limit, the total plate state is set to 0.

6 Conclusion and Future Work

In this paper, a real-time implementation of a simulation of the tromba marina has been presented. The output sound has been found natural and brass-like by the authors and exhibited similar qualities when compared to a real (recreated) tromba marina.

Future work includes a comparison between the non-iterative methods used in this paper and iterative methods (such as and Newton-Raphson) both regarding algorithm speed and sound quality.

Lastly, for a more physical implementation of the damping finger, it would be good to model it as another mass colliding with the string rather than

directly imposing damping onto the string state.

Acknowledgments

The authors would like to thank Peter Williams for his valuable feedback on our application.

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892. Ducceschi's work was supported by an Early Career Fellowship from the Leverhulme Trust.

7 References

[1] D. Munrow, *Instruments of the Middle Ages and Renaissance*. Oxford University Press, USA, 1976.

[2] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," Ph.D. dissertation, Grenoble INP, 1979.

[3] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

[4] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[5] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[6] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[7] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[8] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[9] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

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

[11] S. Bilbao, B. Hamilton, R. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial." *Springer handbook of systematic musicology*, 2018.

[12] Sensel Inc. (2020) Sensel morph. [Online]. Available: https://sensel.com/

[13] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," *Proc. of the 16th Sound and Music Computing (SMC) Conference*, pp. 275–280, 2019.

[14] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2015.

[15] N. Lopes, T. Hélie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," *Proc. 5th IFAC*, 2015.

[16] A. Falaize and T. Hélie, "Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach," *Applied Sciences*, vol. 6, pp. 273–273, 2016.

[17] A. Falaize, "Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation hamiltonienne à ports," Ph.D. dissertation, Université Pierre et Marie Curie, Paris, 2016.

[18] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: The case of collisions," *Proc. of the 22th Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[19] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2014.

[20] S. Bilbao and M. Ducceschi, "Large-scale real-time modular physical modeling sound synthesis," *Proc. of the 22th Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[21] JUCE ROLI. (2020) JUCE. [Online]. Available: https://juce.com/

[22] S. Willemsen. (2020) Virtual tromba marina - sensel morph. [Online]. Available: https://www.youtube.com/watch?v=x72Xh-nUoVc

[23] A. Baldwin, T. Hammer, E. Peciulis, P. Williams, D. Overholt, and S. Serafin, "Tromba moderna: A digitally augmented medieval instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, vol. 16, pp. 14–19, 2016.

Paper E

Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling

Silvin Willemsen, Razvan Paisa and Stefania Serafin

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 300–307, 2020.

Abstract

This paper proposes a multisensory simulation of a tromba marina – a bowed string instrument in virtual reality. The auditory feedback is generated by an accurate physical model, the haptic feedback is provided by the PHANTOM Omni, and the visual feedback is rendered through an Oculus Rift CV1 head-mounted display (HMD). Moreover, a user study exploring the experience of interacting with a virtual bowed string instrument is presented, as well as evaluating the playability of the system. The study comprises of both qualitative (observations, think aloud and interviews) and quantitative (survey) data collection methods. The results indicate that the implementation was successful, offering participants realistic feedback, as well as a satisfactory multisensory experience, allowing them to use the system as a musical instrument.

1 Introduction

Fig. 1: A tromba marina owned by *Nationalmuseet* in Copenhagen, Denmark.

The tromba marina is a bowed monochord from medieval Europe [1] (see Figure 1). The string rests on a loose bridge that rattles against the body. This rattling mechanism creates a sound with brass- or trumpet-like qualities. Unlike other bowed string instruments, different frequencies are created by slightly damping the string with a finger of the non-bowing hand as opposed to pressing the string fully against the neck. This interaction at different locations along the string triggers the different harmonics of the open string. Furthermore, the tromba marina is bowed closer to the nut, and the finger determining the frequency is closer to the bridge (below the bow). As the tromba marina is a rare instrument which can be merely found in museums, very few have the opportunity to play it and discover its interesting timbral possibilities. We wish to recreate the feeling of playing this instrument by using physics based multisensory simulations [2].

In the context of musical applications, physics based multisensory simulations have shown some interest in the sound and music computing community. As stated in [3], the combination of haptics and audio visual content has its own specific challenges worth investigating. Sile O'Modhrain is one of the pioneers that noticed the tight connection between auditory and haptic feedback and investigated how haptic feedback can improve the playability of virtual

instruments [4]. At the same time, Charles Nichols developed the vBow, a haptic human computer interface for bowing [5]. For several years, researchers from ACROE in Grenoble have developed multisensory instruments based on the mass-spring-system paradigm, with custom-made bowing interfaces [6, 7]. Such multisensory simulations have recently been made open source [8]. Haptic feedback has also been combined with digital waveguide models for simulating bowed string interactions [9].

Simulating the feeling of string-instrument vibrations is particularly important since it has been shown how vibrations' level can be strongly perceived [10]. We use democratized VR technologies controlled by a commercial device called the PHANTOM Omni (or simply Omni) by SenseAble Technologies (now 3D Systems) [11]. The Omni is a six-degrees-of-freedom system providing the tracking and haptic feedback (up to 3.3 N) in our application. Using the same device, Avanzini and Crosato tested the influence of haptic and auditory cues on perception of material stiffness [12]. Auditory stimuli were obtained using a physically-based audio model of impact, in which the colliding objects are described as modal resonators that interact through a non-linear impact force [13]. Auditory stiffness was varied while haptic stiffness was kept constant. Results show a significant interaction between auditory stiffness and haptic stiffness, the first affecting the perception of the second. Passalenti et. al's also used the Omni to simulate the act of plucking a virtual guitar string [14, 15, 16].

The goal of this project is to explore the experience of interacting with virtual bowed instrument by using physics based simulations and haptic feedback, together with a visual virtual reality (VR) experience. This effectively makes the implementation a virtual reality musical instrument (VRMI) [17]. The tromba marina is used solely as inspiration because it affords itself to being a solid starting point by having only one string. Besides that, the rarity of the instrument ensures that the participants do not have prior experience playing a tromba marina, nullifying possible comparisons between a real instrument and the virtual one. At no point the system was evaluated as an alternative to the real tromba marina. The system (and its evaluation) is targeted towards musicians in order to avoid discouragement frequently encountered when non-musicians interact with musical instruments. It is assumed that musicians acknowledge that mastering any instrument require extended study, therefore it is expected that they will not evaluate this system exclusively based on its difficulty to play.

We start by describing the implementation of the system, both from the hardware and software perspective in Section 2, followed by presenting a study that evaluates the setup in Section 3. Section 4 shows the results of the evaluation and Section 5 discusses these. Finally, concluding remarks appear in 6.

2 Implementation

The virtual tromba marina consists of three main components: auditory, visual and haptic feedback, all of which will be elaborated on in this section. For visuals, the Oculus Rift CV1 setup was used [18]. The setup consists of a head-mounted display (HMD) and a pair of of wireless controllers that provide tracking information and user input through several buttons and a joystick. A diagram showing the full setup of the system can be found in Figure 2. The controls, their mapping to the system and the final setup of the system will also be presented. A video showing the implementation can be found in [19].

2.1 Auditory Feedback

The audio is generated by a physical model of the tromba marina presented in a companion paper [20]. Some parameters of the model are exposed and can be controlled by the user. These are the velocity, force and position of the bow and the position of the finger inducing the harmonics. The algorithm will not be discussed in detail here, but the mapping to the various parameters of the model will be described in Section 2.4.

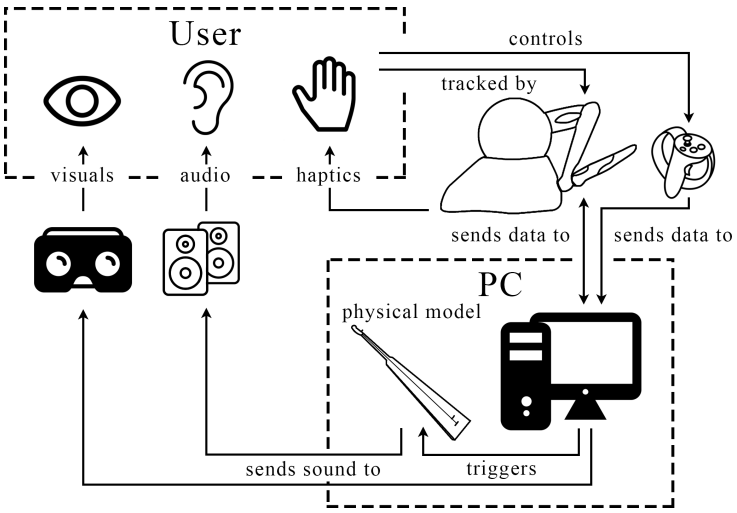


Fig. 2: Diagram showing the system layout of the application. The user interacts with the system using the Omni – which in turn provides haptic feedback – and the Oculus Touch controller. These trigger the physical model of the tromba marina. Auditory feedback then comes from speakers and visual feedback from the Oculus Rift headset. A detailed explanation can be found in Section 2.5.

2.2 Visual Feedback

The application was built using the cross-platform game engine Unity3D (or simply Unity) [21] which can be used to build VR applications. Even though the visual feedback is not the focus of the implementation and eventual evaluation, it was used to guide the users' movements and give them a sense of where the virtual instrument was located. Figure 3 shows a screenshot of the view from the HMD, depicting the virtual instrument, the bow and the damping finger indicator. A 3D model of the tromba marina was made inspired by a real-life instrument (presented in [22]) available to the authors. The overall environment resembled a medieval room, providing context to tromba marina's historical nature.

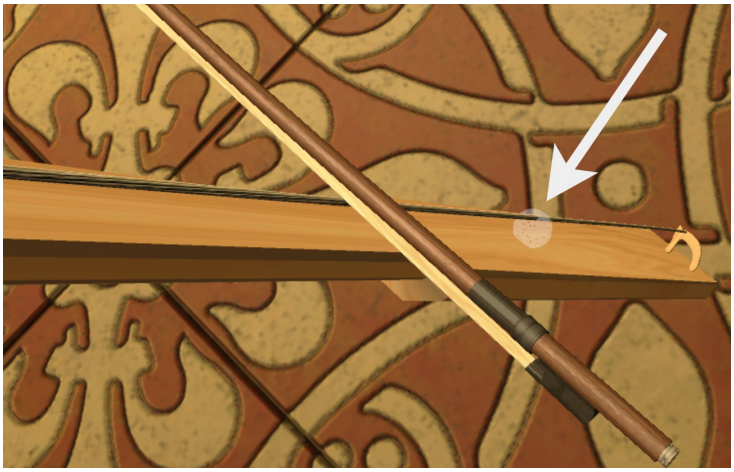


Fig. 3: The view from the head-mounted display (HMD). The damping finger is highlighted and shown as a transparent white sphere.

2.3 Haptic Feedback

The PHANTOM Omni (or simply Omni) is a six-degrees-of-freedom tracking and haptic system developed by SensAble Technologies (see Figure 4). The device has a pen-shaped arm that a user interacts with.

The raw data provided by the Omni are 1) the absolute position of pivot point B2 (three degrees of freedom), 2) the rotation (three degrees of freedom), and 3) the pressure (touching depth). The latter is calculated from the absolute euclidean distance between the virtual collision point of the object (in our case the bow) and the virtual position of the pen.

The axes are labelled as follows in relation to the virtual tromba marina (also see global coordinate system in Figure 5): x-axis (width): horizontally

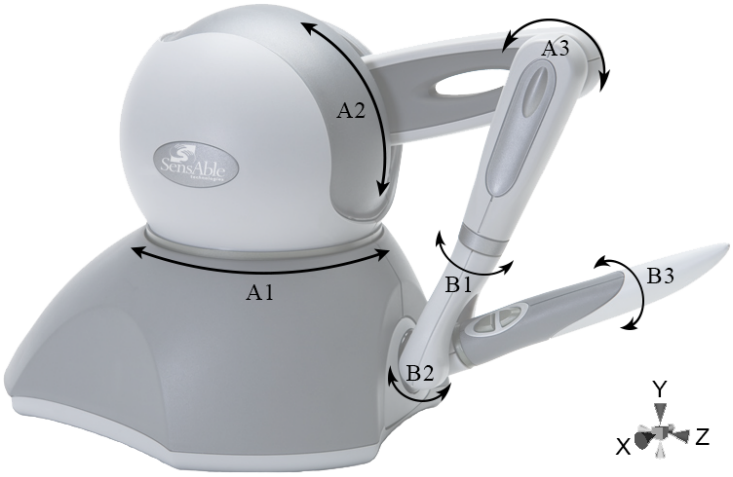


Fig. 4: The PHANTOM Omni has six axes of rotation, three of which provide force feedback (A1-3), and three only tracking position (B1-3). Together, these axes provide six degrees of freedom: x, y and z positions of B2 (according to the shown coordinate system) and rotations of the pen.

across the soundboard (the common interaction direction), y-axis (height): floor to ceiling, and z-axis (depth): perpendicular to the soundboard. The orientation of the Omni with respect to the aforementioned axis can be seen from the coordinate system in Figure 4.

The fact that pivot points B1-3 do not provide force feedback gives rise to an issue in our application. The virtual bow's frog (where it is held by the player) has been placed at the pivot point B2, whereas the interaction between the virtual bow and string happens at an offset as seen in Figure 5. To solve this issue, we created a separate game object with which the bow (pivot point B2 to be exact) will interact with in the virtual world Figure 5. This '(hidden) collision block' lives in a local coordinate system and its x and y-position exactly follow that of the Omni-pen. The y-rotation will change the rotation of the local coordinate system and uses the virtual string as the center point. If, for any reason, the bow ends up behind the string, the collision block will be offset to the left along the (local) x-axis so that no collision occurs when trying to return the bow to the normal playing area.

Through a list of pseudophysical parameters, the collision forces computed by Unity's physics engine are mapped to the haptic feedback produced by the Omni. Through empirical testing, the following pseudophysical parameters have been found: *Stiffness*: 0.003, *Damping*: 0.0071, *Static Friction*: 0, *Dynamic Friction*: 0.109, and *Pop-through*: 0. For more information, please refer to [23].

Throughout implementation, it was considered to actuate the Omni's pen with the output of the physical model used for the auditory feedback, in order to replicate the stick-slip interaction encountered in a real bowing scenario.

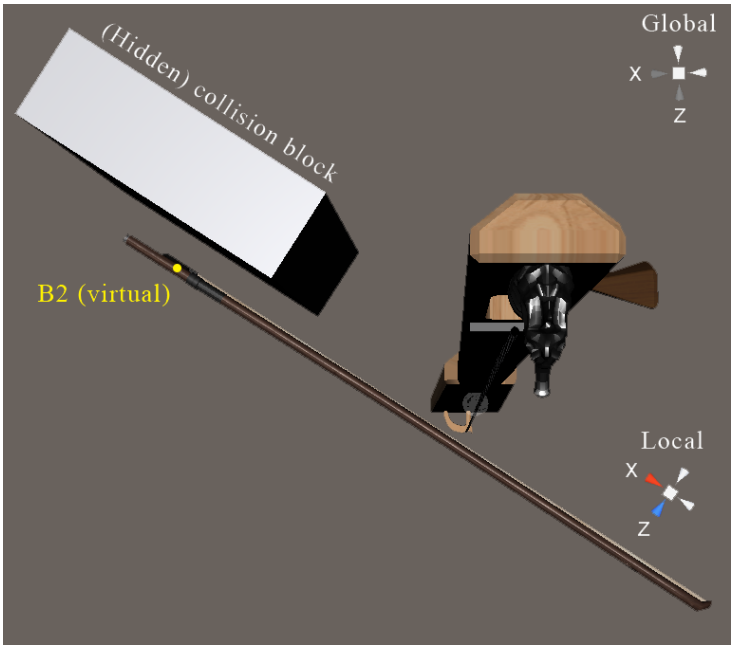


Fig. 5: Top-down view of the global and local coordinate system (x-z–plane). The rotation of the local coordinate system around the (global) y-axis is determined by the y-rotation of the bow. The (normally hidden) collision block lives in the local coordinate system. Its (local) x and y-position follows the (local) x and y-position of B2.

This was deemed unnecessary, as the Omni's internal gearing systems provide a similar, though uncorrelated, haptic feedback, which satisfied the authors.

2.4 Controls and Mapping

As most people are right-handed, it was chosen to also have the bow in the right hand in the application. The (now-local) x-velocity of the Omni is mapped to the bow velocity, pressure to bow force and y-position (including rotation around the local z-axis) to bow position. The left hand is used to control the pitch by changing the position of the damping finger along the string. This position is defined as

$$x_{\mathrm{f}} = L \cdot n^{-1}, \tag{1}$$

where L is the length of the string and $n \in [2, 8]$. If n is an integer, it is the number of the harmonic we want to induce. The lowest harmonic has been set at half the string length $L/2$, meaning that the string is never completely open. The highest harmonic (8 in this case) has been chosen to be the one that can still be (comfortably) reached. The location of the damping finger x_{f} is controlled using the 'X' and 'Y' buttons and the joystick on the left Oculus

Touch controller. The buttons are used for “discrete harmonic" control of the damping finger, i.e. integer values of n in Equation (1), where ‘Y’ increases n and ‘X’ decreases it. The joystick allows for fine pitch control, i.e., decimal values of n, and moves the damping finger up and down the string. The latter could potentially create pitch glides in the output sound of the application, but make it harder to ‘hit’ a perfect harmonic according to Equation (1). If a button is pressed while the current finger position is between two discrete points, the position will move to the next or previous discrete position, depending on the button pressed.

2.5 Physical Setup

The physical setup is shown in Figure 6. The Omni is mounted on a stand at ~125 cm to match the approximate bowing height of the real instrument. As can be seen in Figure 6, the right Oculus Touch controller is mounted right underneath the Omni. This is used to align the physical setup with the virtual tromba marina, both in the x-z–plane but also the height of the bow in the application. After the scene is initialised the controller is used for a tilting interaction so that the instrument can rest on the user’s body, as is done with the real instrument. The aforementioned alignment came with a drawback – as the center of the x-axis range of the Omni was aligned with the tromba marina and B2 was aligned with one end of the bow, only half of the range of the Omni could be used for bowing.

The setup shown in Figure 2 is implemented as follows: the user controls the application using the Omni (for tracking) and the left Oculus Touch controller which sends data to the computer running the application. The Omni produces haptic feedback based on Unity’s physics engine calculating the interaction force between the ‘(hidden) collision block’ and the virtual bow as shown in Figure 5. This data simultaneously triggers the physical model which sends its output to a pair of speakers. The user wears a HMD that gives visual information about the location of the tromba marina (and medieval scene). The user’s position in the VR environment is controlled by the HMD, but this dataflow is not visualised in the diagram. Lastly, the right Oculus Touch controller is attached to the stand the Omni is attached to, and sends position and tilting data to the application.

3 Evaluation

The goal of the study was to (1) evaluate the general experience of bowing in a VR environment using haptic feedback and accurate physical modelling and (2) to evaluate the playability of a VR monochord instrument. This was done by exploring the quality of the software, the acoustic model, the interface and

Fig. 6: User interacting with the physical setup. The Omni is mounted on a ~125cm stand together with the right Oculus Touch controller used for location and tilting information.

the mapping, as proposed by [24] and implemented previously in a similar study [25]. To meet this aim, an investigative study was performed through which feedback on the virtual instrument was collected. In order to ensure a high level of validity and reliability, a triangulation of methods has been used: think aloud protocol [26] throughout the interaction, observation and post-study self report through a modified Usability Metric for User Experience survey [27]. The study concluded with an semi-structured interview based on the observed actions, noted comments and questions loosely revolving around *goals, operators, methods and selection* method [28].

3.1 Participants

A total of 14 people (12 male, 2 female), 23-48 years old (M=29.5, SD=7.65) participated in the study. All participants were students or staff at Aalborg University Copenhagen. The selection of participants was based on the single criterion that one had to have experience playing a musical instrument. Over 70% of the participants have been playing an instrument for more than 5 years, guitar being the most common occurrence (25%). There was only one participant experienced in playing bowed instruments (violin). The same participant mentioned playing the tromba marina briefly before, but the majority of the

other participants had never heard (of) it. All but one participant have had tried VR experiences before joining the study.

3.2 Procedure and Task

The experiment started with the participant reading an introduction about the experiment and completing a questionnaire covering several demographic questions (age, gender, musical experience, familiarity with the tromba marina and VR experience). They were then introduced to the setup and task, and controls were explained. The participants were informed that the study is exploring the experience of bowing in a VR environment. It was emphasised that the most important part of the experiment was for the participant to talk aloud with the phrase: "anything positive, negative, basically anything that comes to mind, please speak out loud". Furthermore, the user was instructed to bow above the damping finger (visualised as a white sphere) at all times, as this is also the interaction with the real instrument.

The interaction part was divided into two phases. Firstly, the participants were asked to freely explore the instrument on their own. Then, when they felt they are ready to move on, an audio recording made by the authors using the application was played, showcasing the system's capabilities, aiming to inspire the second phase of free exploration. It was stressed that the participants did not have to recreate what they heard, but to merely use it as inspiration. The experiment concluded with participants completing a questionnaire covering usability and playability of the system. Finally, a semi-structured interview was held which lasted 5 minutes on average.

Throughout the interaction phase, the participants' actions were observed and noted by the authors, and their comments written down. Most participants were encouraged again to think aloud during their exploration. The full experiment lasted ~30 minutes for all participants.

3.3 Measurements

Because the goal of the study was to investigate the overall experience of bowing in VR, as well as evaluate the playability of the instrument, self-reporting measurements were used in combination with the observations, interview and *think aloud* notations. Specifically, after exposed to the instrument, the participants were asked to fill out a questionnaire containing 20 items related to the experience of interacting with the VRMI. The items can be broadly segmented into four categories: overall experience, haptic feedback, auditory feedback and visual feedback. Table 1 presents the questions.

Questionnaire items:
Overall experience:
(1) It was easy to understand how to play the instrument.
(2) I felt the instrument was hard to play.
(3) I felt the instrument was expressive.
(4) The instrument's capabilities did not match my expectations.
(5) I felt I could easily achieve my goals.
(6) I made many errors playing the instrument.
(7) I am satisfied with the instrument.
(8) I felt the instrument was boring.
(9) Interacting with the instrument was frustrating.
Haptic feedback:
(10) I felt the haptic feedback was realistic.
(11) I felt the haptic feedback was too strong.
(12) I felt the haptic feedback was natural.
Auditory feedback:
(13) I felt I was in control of the sound.
(14) I felt the audio was matching my actions.
(15) I felt the sound was matching the haptic feedback.
(16) I felt the sound was matching the visuals.
(17) I felt the sound was static.
Visual feedback:
(18) I felt the visual feedback was helping me play.
(19) I felt the visuals were confusing.
(20) I felt the visuals were matching my actions.

Table 1: The questionnaire items and corresponding anchors of the 5 point (1 – 5) rating scales (Strongly disagree – Strongly agree).

4 Results

This section presents the results obtained from the self-reported measure regarding the participants' experience as well as the qualitative findings from interview, observations and think aloud.

4.1 Quantitative Data

The data obtained for the questionnaire items was treated as ordinal and analysed in terms of central tendency (medians and mode), interquartile ranges, minimum and maximum ratings. Figure 7 visualises the collected data. The mode was considered only when different from the median, specifically question 8, 13 and 14. It is worth noting that most of the items show a skewed normal distribution.

Question 1-9 paint a picture of how the instrument was perceived by the users. Questions 1, 2, 4, 5, 6 and 9 cover the perceived difficulty of using the

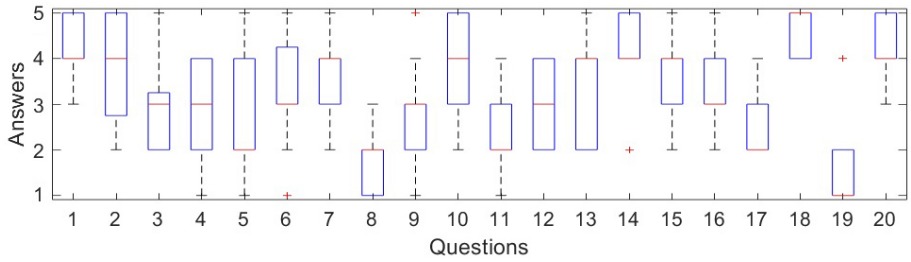


Fig. 7: Boxplots visualizing the results related to the 20 questionnaire items (shown in Table 1) in terms of medians (red lines), interquartile ranges (blue rectangles), minimum and maximum ratings (dashed lines), and outliers (red crosses). The y-axis maps "Strongly disagree – Strongly agree" to a 1 – 5 interval.

system as a musical instrument. The answers to these questions show that even though participants generally found the instrument easy to understand, they had difficulty playing it and reaching their goals. Questions 3, 7 and 8 cover their general opinion about the instrument. Participants generally felt satisfied and not bored with the instrument. Questions 10-12 cover exclusively the impressions about haptic feedback. It can be seen that most participants found the haptic feedback to be realistic and generally natural and the force to be not too strong. The questions 13-17 approach the auditory aspect of the instrument, focusing on its perceived characteristics. As can be seen from questions 13, 14 and 17, the participants felt a high level of command over the sound, and were satisfied with mapping between the haptic and auditory feedback. The same thing can be said about the visual mapping, as indicated by question 16. Items 18-20 investigate the perceived visual quality. It can be seen that the visuals helped the participants play and were implemented well, i.e., not confusing and matching their actions.

4.2 Qualitative Data

In order to present an accurate representation of the findings, this section will be split into two categories: actions – covering the observed activities during the interaction phase, and oral feedback – presenting the findings from the think aloud protocol and interviews.

4.2.1 Observed Actions

Since there were no tasks given to the participants, all actions were noted and analysed. That said, most users performed similar actions in their interaction phase. All participants experimented with bowing at different heights, but only a few of them tried to explore bowing heights for all discrete pitches. Most of them were satisfied with trying different heights on whatever pitch

they found themselves at that time. In a similar fashion, all participants experimented with playing different pitches, both using the discrete buttons as well as the joystick. It is worth mentioning that many users tried to investigate the limits of the pitches they could play. Higher pitches usually resulted in little or no sound which was commented on by most. This behaviour is true to a real tromba marina, where higher harmonics are harder to excite than lower ones. The majority tried to perform some form of glissando, as well as bowing with different velocities, usually commenting on the findings. Due to the non-intrusive nature of observation, it was impossible to notice the pressure applied with the bow, but some participants explicitly mentioned that they tried to experiment with different forces. This was especially true in the second phase of interaction, when they experimented with a higher dynamic range of sounds. Another common occurrence was the attempt to play some sort of melody or riff. Simple melodies like *Mary had a little lamb*, or *Twinkle twinkle little star* were attempted, with various degrees of success. One participant tried to play a Mozart segment. The last commonality was found in the attempt to perform a sustained tone, with a constant bowing speed and a back-and-forth motion.

When it comes to seldom or individual actions, a great variance in experimentation was observed. Participants tried to hit the string with the bow, rotate the bow upwards to the point of it being parallel to the string, move the bow in an up-down (y-axis) motion, bow on the damping finger indicator and underneath it or play some form of vibrato or staccato. No one tried to tilt the stand supporting the Omni.

4.2.2 Oral Feedback

Generally the overall impression of the instrument was positive, described with words like: *cool*, *fun*, *interesting*, *weird*, as well as *hard* or *difficult to play*. One participant's answer encapsulates this very well by saying: "I got to express my ideas, but not perfect them".

Just as described in the previous section, there was a general consensus on several reported characteristics. All participants that attempted to play the highest harmonic said it is hard to play, and that it felt frustrating. At the other end, several participants expressed their preference towards lower pitches, where some said that they prefer the sound produced when bowing under the damping finger (essentially playing the lower-pitched open string). Besides that, many reported that is was hard to maintain a sustained tone, regardless of the pitch. Another sound-related report was the inability to re-create the buzzing sound heard in the recording; one of the participants familiar with the tromba marina's mechanism even mentioned specifically that he "couldn't get the bridge to rattle". When it comes to the pitch selection interface, the reports are very polarised between the joystick and the buttons. On one hand

some describe the buttons as being more, *fun, musical, melodic, useful* or *easier*, while describing the joystick as *useless, unrealistic, too hard* or *meaningless*. On the other hand some participants clearly preferred the joystick describing it as *natural, intuitive, interesting, expressive* or *humane*, but everyone mentioned that the sensitivity of the joystick is too high, making it hard to land on the desired pitches. Due to the incremental nature of the damping fingers' position, it was impossible to skip over notes, a fact that was mentioned in different forms by several participants. Some noted that the control of the damping finger ('Y' for up the string and 'X' for down) should have been inverted. Furthermore, some would have liked a more physical interaction for the damping finger, such as moving the controller up and down rather than using buttons.

When it comes to the haptic feedback, the majority was satisfied with it, mentioning that "it feels nice", "it feels good", "is great", "impressive - it felt natural", or "it feels real", while one participant found it to be "wild and a bit too powerful". A special case related to the haptic feedback was the bounce obtained by hitting the virtual string with the bow. Most subjects found it pleasing and were intrigued by its realistic feel, but the violin player repeatedly mentioned that it is "unrealistic and way to powerful". One participant explicitly mentioned that the haptic feedback matches the auditory one, and his expectations.

Several participants noticed that the bow could rotate along its axis and asked whether it made a sonic difference or not, to which they were answered negatively. Besides that, there were very few comments regarding the visual aspect of the system, but most of these were positive. One participant mentioned that sometimes there's a gap between the string and the bow, and that it would be nice to observe one's hands. No one mentioned anything related to the visual indication or the damping finger seen in Figure 3.

The overall interaction was described offering a high degree of freedom on the bowing hand, but the pitch selecting hand was either not mentioned, or described as *disconnected* several times. Some agreed that the instrument is hard to play, mentioning that it is *frustrating*. However, most people estimated that they can perform better after practising more.

5 Discussion

In this section, both the evaluation procedure itself and the results from Section 4 will be discussed.

5.1 Procedure

It is acknowledged that the cognitive load of speech and playing an instrument are overlapping [29], therefore the *think aloud protocol* might have not generated

in the most abundant data possible. Most participants alternated between playing and speaking. This resulted in occasionally long breaks in either activities, and required participants to be encouraged to think aloud. Retrospectively, a structured activity schedule allocating time for playing and feedback could have been more productive. Similarly, using self report through Likert scales require large sample sizes to achieve a high level of accuracy[29]. Therefore the interpretation of results rooted into the qualitative data, and then validated using the quantitative data.

Furthermore, as the data we obtained was purely through non-intrusive methods, it would have been useful to log the raw data provided by the Omni (such as the bowing pressure). This could then have been analysed to obtain a better understanding of the user's feedback.

Lastly, the audio did not fully match the sounds that were possible to create with the application. As the recording was quite distorted, the volume of the audio plugin was turned down during the test, but the recording was not remade. This will be elaborated on below.

5.2 User Feedback

The generally positive oral feedback about the overall experience is backed up by the quantitative data which showed that participants were satisfied and not bored with the instrument. They attempted to perform fundamental tasks as producing a sustained tone or playing simple melodies with various degrees of success, and when exposed to the example recording, some tried to recreate the sounds heard from the audio clip. Several participants mentioned that it was difficult to achieve this particular goal, a problem that finds its explanation in the difference in volume between the recording and the experiment scenario as mentioned above. This would be a point of improvement for future testing, as it could have impacted the answers for question 5 – the lowest scoring question regarding the overall experience. Another reason for this question's answers could be linked to the inability to play the higher notes, or the limited pitch range, as presented in Section 4, but these characteristics are inherited from the physical characteristics of the real instrument, so could be expected.

Interestingly, many participants believed that it was easy to understand how to play the instrument, but that they could become better after some more practice. This indicates that the setup has a low "entry-level", with an envisioned high virtuosity ceiling. This is believed something desirable when creating computer based instruments [30]. Even though the implementation was inspired by a real instrument with a possibly different learning curve, as mentioned, it was not our goal to recreate it.

The haptic feedback was considered positive and generally having an appropriate level of resistance to movements. The answers to questions 10 (realistic haptic feedback) and 12 (natural haptic feedback) correlate positively

and question 15 (sound matching the haptic feedback) was also answered positively, giving a strong indication that the participants considered the haptic feedback real and according to their expectations. It can be understood that the realism of the haptic feedback is estimated considering the multisensory experience and this result reassures that bowing in VR with our setup is possible.

Furthermore, many participants noticed that they could 'bounce' the bow onto the string. Even though this behaviour was a byproduct of the implementation, participants generally liked this interaction and found it to be realistic and exciting.

Many users noted that the full range of the bow could not be used. As mentioned in Section 2.5, the virtual tromba marina was aligned with the physical position of the Omni and as the bow is held at one end, about half of the range could not be used for bowing. This was a commonly reported issue, and it could have impacted the answers for questions 4, 5, and 9. The reason for aligning the physical setup with the virtual tromba marina was the tilting interaction, so that if people wanted to interact with the entire instrument, they would be able to grab the physical setup. As none of the participants used this, we could discard the aforementioned alignment to be able to account for the entire range of the bow.

The polarisation of the participants' opinion on the pitch control – joystick versus buttons – was backed up by the answers individuals gave on question 9 (interaction was frustrating). It could be argued that users preferring the joystick over the buttons had a harder time interacting with the instrument than the people preferring the buttons. As mentioned, all participants who mentioned the joystick interaction said it was too fast, explaining the above.

6 Conclusions

This paper presents a virtual reality implementation of the tromba marina and its evaluation. Our goal was to evaluate the general experience of bowing in VR and to evaluate the playability of our implementation. The results show that the implementation was successful with participants finding the haptic feedback realistic and the general experience enjoyable and interesting on one hand, and difficult and frequently frustrating on the other hand. Nevertheless, all sensory modalities we focused on (auditory, haptic and visual) seemed to reinforce each other, inspiring participants to attempt to play melodies with the instrument. This was considered to be an important achievement. Improvements on our application include the pitch control, which should either be more physical, i.e., moving the pitch hand physically up and down the virtual string, or simply slower continuous control. Besides that, a better physical setup, allowing the users to utilise the entire bow is desired. The

findings of this paper prove that it is possible to create a satisfactory bowed VRMI using off-the-shelf hardware and accurate physical modelling.

Acknowledgments

We would like to thank Peter Williams for his valuable insights on playing the tromba marina, feedback on our application and providing access to his instrument replica.

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

7 References

[1] The Editors of Encyclopædia Britannica, *Encyclopædia Britannica*. Encyclopædia Britannica, Inc., 2020.

[2] D. K. Pai, "Multisensory interaction: Real and virtual," *Robotics Research. The Eleventh International Symposium*, pp. 489–498, 2005.

[3] F. Danieau, A. Lécuyer, P. Guillotel, J. Fleureau, N. Mollet, and M. Christie, "Enhancing audiovisual experience with haptic feedback: a survey on hav," *IEEE transactions on haptics*, vol. 6, no. 2, pp. 193–205, 2012.

[4] M. S. O'Modhrain, "Playing by feel: incorporating haptic feedback into computer-based musical instruments," Ph.D. dissertation, Stanford University, 2001.

[5] C. Nichols, "The vbow: development of a virtual violin bow haptic human-computer interface," pp. 1–4, 2002.

[6] J.-L. Florens and C. Cadoz, "Modèles et simulation en temps réel de corde frottée," *Le Journal de Physique Colloques*, vol. 51, no. C2, pp. C2–873–C2–876, 1990.

[7] A. Luciani, J.-L. Florens, and N. Castagné, "From action to sound: a challenging perspective for haptics," pp. 592–595, 2005.

[8] J. Villeneuve and J. Leonard, "Mass-interaction physical models for sound and multi-sensory creation: Starting anew," *Proc. Int. Conf. Sound and Music Computing*, 2019.

[9] S. Sinclair, G. P. Scavone, and M. M. Wanderley, "Audio-haptic interaction with the digital waveguide bowed string," *ICMC*, 2009.

[10] A. Askenfelt and E. V. Jansson, "On vibration sensation and finger touch in stringed instrument playing," *Music Perception: An Interdisciplinary Journal*, vol. 9, no. 3, pp. 311–349, 1992.

[11] 3D Systems, Inc, "3D Systems Touch Haptic Device," accessed February 12, 2020, available at https://www.3dsystems.com/haptics-devices/touch.

[12] F. Avanzini and P. Crosato, "Haptic-auditory rendering and perception of contact stiffness," *Lecture Notes in Computer Science*, vol. 4129, pp. 24–35, 2006.

[13] F. Avanzini and D. Rocchesso, "Physical modeling of impacts: theory and experiments on contact time and spectral centroid," *Proc. Int. Conf. Sound and Music Computing*, pp. 287–293, 2004.

[14] A. Passalenti, R. Paisa, N. Nilsson, N. Andersson, F. Fontana, R. Nordahl, and S. Serafin, "No strings attached: Force and vibrotactile feedback in a virtual guitar simulation," *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1116–1117, 2019.

[15] ——, "No strings attached: Force and vibrotactile feedback in a guitar simulation," *Proc. Int. Conf. Sound and Music Computing*, 2019.

[16] F. Fontana, R. Paisa, R. Ranon, and S. Serafin, "Multisensory plucked instrument modeling in Unity3D: From keytar to accurate string prototyping," *Applied Sciences*, vol. 10, 2020.

[17] S. Serafin, C. Erkut, J. Kojs, N. Nilsson, , and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, 2016. [Online]. Available: http://www.mitpressjournals.org/doi/pdfplus/10.1162/COMJ_a_00372

[18] Facebook Technologies, LLC, "Oculus Rift: VR Headset for VR-ready PCs | Oculus," accessed March 6, 2020, available at https://www.oculus.com/rift/.

[19] Razvan P, "Resurrecting the Tromba Marina," accessed March 6, 2020, available at https://www.youtube.com/watch?v=SlHqvaaPCyU.

[20] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, "Real-time implementation of a physical model of the tromba marina," *submitted to the 17th Sound and Music Computing (SMC) Conference*, 2020.

[21] Unity Technologies, "Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations," accessed February 10, 2020, available at https://unity.com/.

[22] A. Baldwin, T. Hammer, E. Peciulis, P. Williams, D. Overholt, and S. Serafin, "Tromba moderna: A digitally augmented medieval instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, vol. 16, pp. 14–19, 2016.

[23] 3D Systems Inc., "Openhaptics unity plugin user guide (toolkit version 3.5.0)," 2018, available at http://s3.amazonaws.com/dl.3dsystems.com/binaries/Sensable/OH/3.5/OpenHaptics_Toolkit_ProgrammersGuide.pdf.

[24] J. Barbosa, J. Malloch, M. Wanderley, and S. Huot, "What does "evaluation" mean for the NIME community?" *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2015.

[25] D. Young and S. Serafin, "Playability evaluation of a virtual bowed string instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pp. 104–108, 2003.

[26] M. Someren, Y. Barnard, and J. Sandberg, *The Think Aloud Method - A Practical Guide to Modelling Cognitive Processes*. London: Academic Press, 1994.

[27] K. Finstad, "The usability metric for user experience," *Interacting with Computers*, vol. 22, pp. 323–327, 2010.

[28] S. K. Card, A. Newell, and T. P. Moran, *The Psychology of Human-Computer Interaction*. New Jersey: Lawrence Erlbaum Associates Publishers, 1983.

[29] D. Stowell, A. Robertson, N. Bryan-Kinns, and M. Plumbley, "Evaluation of live human–computer music-making: Quantitative and qualitative approaches," *International Journal of Human-Computer Studies*, vol. 67, pp. 960–975, 2009.

[30] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, 2002.

Paper F

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study

Silvin Willemsen, Anca-Simona Horvath and Mauro Nascimben

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 292–299, 2020.

Abstract

This paper presents DigiDrum – a novel virtual reality musical instrument (VRMI) which consists of a physical drum augmented by virtual reality (VR) to produce enhanced auditory and haptic feedback. The physical drum membrane is driven by a simulated membrane of which the parameters can be changed on the fly. The design and implementation of the instrument setup are detailed together with the preliminary results of a user study which investigates users' haptic perception of the material stiffness of the drum membrane. The study tests whether the tension in the membrane simulation and the sound damping (how fast the sound dies out) changes users' perception of drum membrane stiffness. Preliminary results show that higher values for both tension and damping give the illusion of higher material stiffness in the drum membrane, where the damping appears to be the more important factor. The goal and contribution of this work is twofold: on the one hand it introduces a musical instrument which allows for enhanced musical expression possibilities through VR. On the other hand, it presents an early investigation on how haptics influence users' interaction in VRMIs by presenting a preliminary study.

1 Introduction

Virtual Reality (VR) is described as an immersive environment provided by technology and experienced through sensory stimuli [1]. Different types of technologies are available for creating VR experiences, and head-mounted displays (HMDs) are among the most popular. VR has been used as a platform for the creation of perceptual illusions, and much research has gone into producing realistic or otherwise compelling visual and auditory experiences. By comparison, the sense of touch has been neglected in spite of its obvious potential to increase a sense of presence in a simulated world [1].

Virtual musical instruments (VMIs) are defined as software simulations or extensions of existing musical instruments with a focus on sonic emulation. Virtual reality musical instruments (VRMIs), are those which also include a simulated visual component [2].

The design and evaluation of DigiDrum – a novel VRMI where a physical darbuka (a djembe-like drum) is enhanced by VR is presented. The user wears a HMD which puts them in a recording studio where a virtual drum is aligned with the physical drum, so that both drums can be played at the same time. Interaction with the (physical + virtual) drum triggers a virtually simulated sound of a drum membrane. This sound is sent to the user through sound-isolating headphones for auditory feedback, and a vibration motor (haptuator) attached to the inside of the physical drum's membrane creating a vibrotactile response in the physical drum – similar to the haptic response a real membrane would produce. As the drum's sound is being simulated, its properties can be

changed on the fly, something which is impossible to do in the physical world.

An initial user study was conducted on DigiDrum with a twofold goal. On the one hand, in order to study how users interact with the installation and use this feedback to improve the drum, and on the other hand, for trying to understand whether there is a correlation between material stiffness perception and the way users interact with the drum. More specifically, the study investigated which parameters influence the perception of the material stiffness of the drum membrane. Different combinations of values for: (1) tension in the virtual membrane and (2) damping, or how quickly the sound dies out were used. The initial hypothesis was that higher values for both tension and damping would influence the perception of stiffness positively. In other words, higher tension and higher damping (sound dying out faster) will result in users perceiving the drum membrane as being more stiff. It was suspected that tension would be the most important parameter in the perception of stiffness. In the test, the auditory and haptic cues were linked, or matching.

The research question which guides this work is:

Can a user's perception of
material stiffness in an enhanced drum membrane change
by using auditory and haptic cues?

The ultimate goal of the paper is to (1) present an installation which helps to enhance musical expression possibilities through a novel VRMI, and (2) investigate users' interaction with a VRMI focused not only on the visual and auditory experience, but also on haptics.

The paper is structured as follows: Section 2 presents a selection of related work. Section 3 is an introduction to haptic perception. Section 4 describes the design criteria used in for DigiDrum. In Section 5 we describe the system overview and Section 6 details the implementation of the visual virtual environment. In Section 7, the physical model sound algorithm is described. In Section 8 a user study looking at the interaction with the setup is presented and preliminary results are shown and discussed in Section 9. Conclusive remarks and future development are made in Section 10.

2 Related Work

Several investigations on the connection between haptic and auditory cues and perception of material stiffness have been done. Some look specifically at playing percussion musical instruments as the music community has long had a strong interest in haptic technology [3], where others investigate haptics, visuals and sound in relation to human computer interaction.

In [4], Sile O'Mohrain describes a series of studies where experienced musicians played VMIs with both haptic and auditory feedback with the aim of

finding out whether adding haptic feedback to these instruments would improve their playability. The results indicate that the presence of haptic feedback can improve a player's ability to learn the behavior of a VMI.

In [5], Dahl gives a detailed analysis of four experienced drummers performing the same musical sequence using drumsticks on drums with three striking surfaces (soft, medium and hard). The study finds that the main parameter influencing the preparatory movement and the striking velocity was the dynamic level and, to a lesser extent, the striking surface.

The work of Avanzini and Crosato's [6], that of Passalenti *et al.* [7], and that of Liu et. al. use the haptic device PHANTOM® Omni™ (now Touch) [8]. Avanzini and Crosato test the influence of haptic and auditory cues on perception of material stiffness separately, in an experiment where subjects had to tap on virtual surfaces, and were presented with audio-haptic feedback. In each condition the haptic stiffness had the same value while the acoustic stiffness was varied. The study indicates that subjects consistently ranked surfaces according to the auditory stimuli. Passalenti *et al.*'s experiment focuses on haptics and guitar strings. In [9], a multimodal interface that synchronizes visual, haptic and auditory stimuli to give users a feeling of presence of virtual objects is presented and thoroughly detailed. The study notices that although the stiffness parameters of different materials were set to be the same, the sound effects biased user's judgment of the hardness of surfaces.

The preliminary experiment conducted in relation to DigiDrum takes inspiration from these works, but looks at the specificity of a VR-enhanced drum.

In [10], the design of a physically intuitive haptic drumstick is presented. The paper suggests that physically intuitive new musical instruments may help performers transfer motor skills from familiar, traditional musical instruments.

3 Haptics

The sense of touch is the first to develop in humans – a sense we cannot shut down. Vision is the last sense to develop, a sense we are able to "turn off" [11] by closing our eyes. Despite this, tactile awareness generally receives less attention than other sensory modalities when it comes to technological development [12]. We live in a world over-saturated by visuals, and VR is a technology where this has been the case notably. In this section, we describe in further detail haptic perception and how it works from a neurophysiological point of view as well as the basis for subjective decision making on tactile sensation.

3.1 Haptic Perception

The peripheral nervous system gathers environmental stimuli in form of visual, audible, tactile, olfactory (smell) and gustatory (taste) inputs and transfers them to the central nervous system for further elaboration and integration. Tactile information is collected in the skin, muscles, and joints and sent to an area in the brain called the primary somato-sensory cortex[13]. This cortical area is the first stage for the tactile awareness occurring across the surface of the body. Several other structures of the central nervous system take part in the generation of tactile feedback, as generally, a single brain area is never responsible for information awareness [14]. Light touch and tactile attention are processed in the secondary somato-sensory cortex – an area directly connected with the primary somato-sensory cortex [15]. Literature reports that people undergoing tactile training improve their perception but also strengthen the connections and cortical representations of the stimulated body area [16]. There is a direct relationship between size of cortical region and haptic performance.

A specific area of the central parietal lobe, placed in the back of the primary somato-sensory cortex, integrates the information from the visual and haptic regions to help locate objects in space.

The sense of hearing is connected to the sense of touch and touching objects in different ways produce abundant sounds which convey information about the object and the interaction, such as material, shape, roughness, stiffness, the gesture, rate and strength of our actions. In VR systems, users may immediately notice the unnaturalness if the interface has no sound or provides mismatched sound [9].

As Cao *et al.* explain in [17], skilled interactions with sounding objects, such as drumming, rely on resolving the uncertainty in the acoustical and tactual feedback signals generated by vibrating objects.

3.2 Notes on Experiments Involving Haptics

Conducting experiments on haptics can prove difficult because there are no proper technological devices for delivering controlled and reliable tactile stimuli [12]. When users interact with a physical object, uncertainty may arise from mis-estimation of the objects' geometry-independent mechanical properties, such as surface stiffness. How multisensory information feeds back into the fine-tuning of sound-generating actions remains unexplored [17].

In virtual environments (as used in VR) and using hand tracking devices such as Leap Motion [18] (see Section 5), subjects are able to move their hands freely, which could confound somato-sensory processing with activations related to motor planning and movement [19]. These uncontrolled motor activities result in uncontrolled somatic stimulation. There is an anatomical explanation of this close somato-motor functional relationship: areas involved

in the perception of touch on the hands in the primary somato-sensory cortex are located mostly in front of the areas responsible for hand movements [20]. Another problem with haptics is the subjective quantification of the stimuli. Contents of tactile consciousness vary between individuals and a common lexicon to evaluate haptic sensation through surveys still seems far to be conceived [21].

3.3 Interaction between Visual Information and Tactile Feedback

In a famous experiment, Pavani *et al.* [22], asked a group of participants to detect the position of vibro-tactile stimuli on their arm. The participant's own arm was placed under a table (out of sight) while a fake rubber hand was laid in front of them. The rubber hand was laid out in a position that was anatomically compatible with participant's real hand. When seeing the mannequin hand being touched, all participants reported that their own hand was being touched, even though that was not the case. In short, the perception of tactile stimulation was simulated through visuals. A similar experiment was conducted by [23] asking subjects to watch a video of a hand being touched on the first finger while their own hand was stimulated synchronously. Brain activity during synchronous stimulation showed an improved tactile acuity. Taking into account previous literature findings, we can conclude that in virtual environments hand manipulations and interactions are important factors that enhance realism and user experience.

4 Design Criteria for DigiDrum

As explained by [10], a new musical instrument is physically intuitive if the physics of haptic interaction are similar to those supported by a traditional musical instrument. Physically intuitive new musical instruments may help performers transfer motor skills from familiar, traditional musical instruments. This is why we choose to augment an existing drum, instead of suggesting a completely new musical instrument – seeing a physical drum will invite users to play the new instrument in an intuitive way and as a regular drum. The mechanics of a musical instrument's interface – what the instrument feels like – determines much of its playability [24].

In creating DigiDrum, the design criteria for VRMIs suggested by Serafin *et al.* were used as guidelines [2]. The setup integrates visuals, audio and haptics and extends an existing musical instrument using VR seeking to create a "magical interaction". Creating a sense of presence is attempted by mapping the virtual drum's location to that of the physical one, and by representing the user's hands in the simulated world. DigiDrum was designed to create

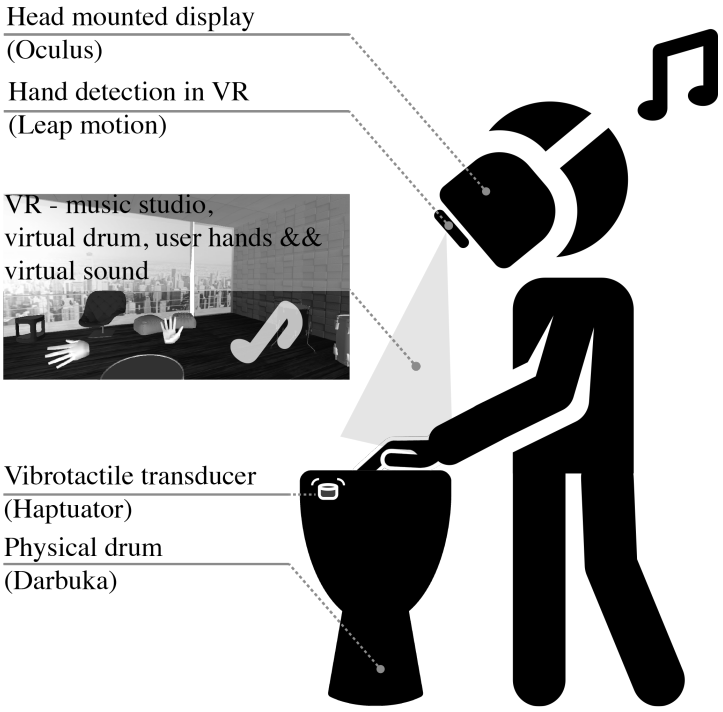


Fig. 1: The physical setup of the system. The Leap Motion is mounted to the front of the HMD.

three types of illusions: (1) a place illusion – users should feel like they are in a music production studio, (2) a plausibility illusion – users should feel like the experience is really happening, and (3) virtual body ownership – users should see their own body in the virtual world and feel ownership of their virtual body.

5 System Overview

Figure 1 shows the overall design of DigiDrum and its setup and Figure 2 shows a user interacting with the setup. For hand-tracking, the Leap Motion [18], which is an infrared-sensor-based camera that allows for accurate hand tracking is used. It is mounted to the front of an Oculus Rift HMD so that the user's hands are in the field of view when they look at the virtual drum. The drum is fixed in-place and played like a djembe. In the application, the virtual drum was placed slightly higher than the physical drum to make sure the physical model was triggered when the physical drum was hit.

A detailed overview of the system is given in Figure 3. The hand movement

Fig. 2: A user interacting with the setup.

data is retrieved by a PC which runs the cross-platform game engine Unity [25]. The Unity 'scene' contains the virtual environment (see Section 6) that the user will see through the HMD and the physical model used for the sound and haptics (see Section 7). The HMD also sends data back to the PC regarding location and head rotation. Once the tracked hand touches (or collides with) the virtual drum, the physical model is triggered and its output sound is sent to a haptuator which is attached to the inside of the drum membrane. This effectively causes the physical membrane to be actuated by a virtual membrane. To accommodate for the plausability illusion mentioned in Section 4, the chosen haptuator has a very high fidelity, i.e. can play realistic audio signals as opposed to non-realistic 'buzzes'. Other forms of haptic feedback have been considered, but – according to the authors – the use of this actuator attached to the drum membrane had the highest potential of resembling realistic drum membrane vibration in the end.

Finally, the same sound that is sent to the haptuator is also sent to sound-isolating headphones. Sound-isolation is important as the sound coming from the physical drum should not interfere with the audio coming from the simulated drum.

6 Unity Implementation

The virtual environment was created using Unity. All the hardware drivers and software components were linked together using this platform. Here, a

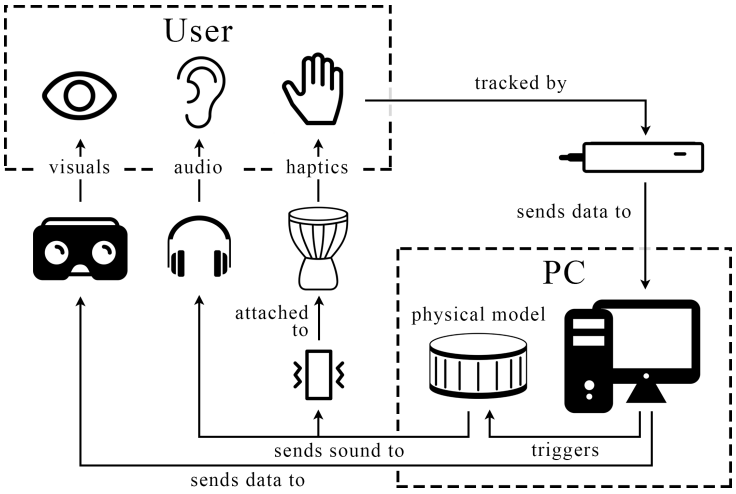


Fig. 3: Detailed system layout. The user interacts with the system using their hands and gets haptic feedback from the haptuator attached to the drum membrane, auditory feedback from closed headphones and visual feedback from the Oculus Rift headset. A detailed explanation can be found in Section 5.

virtual drum playable with hand motion using Leap Motion was created. The user enters the VR environment (rendered as a recording studio) and Leap Motion reconstructs (in VR) the subject's own hands. In the virtual recording studio a drum was placed at the center and programmed to detect collision with the reconstructed hands. When a collision was detected, a C# script, in which a physical model of a drum membrane was programmed, was activated to reproduce the beating sound of the drum through an actuator placed inside the drum skin.

7 Physical Model

The behaviour of musical instruments can be well described by partial differential equations (PDEs) [26]. In this section, the continuous-time PDE for a drum-membrane is given and explained. This is followed by an explanation of the discretisation method used. Finally, the parameter values used for the implementation are given.

7.1 Continuous Time

A rectangular (stiff) membrane with dimensions L_x (m) and L_y (m) can be described by the following equation [27]:

$$\rho H \frac{\partial^2 u}{\partial t^2} = T\Delta u - D\Delta\Delta u - 2\sigma_0 \frac{\partial u}{\partial t} + 2\sigma_1 \Delta \frac{\partial u}{\partial t}. \quad (1)$$

Here, state variable, $u = u(x, y, t)$ is a function of horizontal coordinate $x \in [0, L_x]$, vertical coordinate $y \in [0, L_y]$ and time $t \geq 0$ and is parameterised in terms of material density ρ (kg/m^3), membrane thickness H (m), tension T (N) and frequency independent and dependent damping coefficients σ_0 (s^{-1}) and σ_1 (m^2/s). Furthermore, $D = EH^3/12(1 - \nu^2)$ with Young's modulus E (Pa) and Poisson's ratio ν. Lastly, Δ represents the 2D Laplacian [27]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \quad (2)$$

Furthermore, clamped boundary conditions – i.e., the state u at all plate edges and their gradients are 0 – have been chosen for simplicity:

$$u = \nabla u = 0 \quad \text{with} \quad \nabla = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}. \quad (3)$$

7.2 Discretisation

For implementing the physical model, finite-difference time-domain (FDTD) methods were used [27]. These methods were chosen over others, such as the 2D waveguide mesh [28], as they allow parameters and real-time changes of these to be better controlled. FDTD methods discretise $u(x, y, t)$ shown in Equation (1) to $u^n_{(l,m)}$ using $t = nk$ with sample n and time step k (s), $x = lh$ where $l \in [0, ..., N_x - 1]$ and $y = mh$ where $m \in [0, ..., N_y - 1]$ where N_x and N_y are the number of horizontal and vertical grid points respectively. Furthermore, grid spacing h (m) can be calculated using

$$h \geq h_{\text{min}} = 2\sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 4\kappa^2 k^2}}{2}}, \quad (4)$$

where $c = \sqrt{T/\rho H}$ and $\kappa = \sqrt{D/\rho H}$. The closer h is to h_{min}, the higher the accuracy of the implementation.

7.3 Parameters

Most parameters used in the simulation were chosen empirically and can be found in Table 1. With these parameters a small (30×30 cm) membrane with a low density and stiffness is simulated. For the purpose of getting the model to work in real time, the minimum grid spacing h_{min} in Equation (4) is multiplied by 4 (h_{min} in (4) is calculated based on the highest value of T and $\sigma_1 = 0.005$). The values for T and σ_0 correspond to the cases used in the experiment. The

Parameter	Symbol (unit)	Value
Membrane width	L_x (m)	0.3
Membrane length	L_y (m)	0.3
Material density	ρ (kg/m^3)	10
Thickness	H (m)	0.001
Tension	T (N)	$\{15, 40, 80\}$
Young's modulus	E (Pa)	$2 \cdot 10^3$
Poisson's ratio	ν (-)	0.3
Freq. indep. damping	σ_0 (s^{-1})	$\{0.5, 2, 5\}$
Freq. dep. damping	σ_1 (m^2/s)	$[0, 0.005]$
Time step	k (s)	$1/44100$
Grid spacing	h (m)	$4h_{\text{min}}$

Table 1: Table showing parameter values.

frequency dependent damping σ_1 follows an exponentially decaying curve,

$$\sigma_1(t) = 0.005e^{-0.01t}, \tag{5}$$

where $t = 0$ at the time of excitation. This allows for very low damping, i.e., very long sound, while taking away some of the high frequency content present immediately after excitation. This ultimately results in a more natural drum sound, even when σ_0 is set low.

8 User study

This work hopes to add to the corpus of design guidelines for VRMIs, more specifically those VRMIs which involve a touch based stroking movement. In [2], Serafin et al. describe three layers of evaluation for VRMIs, namely: (1) investigating modalities of interaction, (2) evaluating VR specific aspects, with engagement being the most interesting from a VRMI perspective, and (3) looking at quality and goals of interaction.

An initial user study was conducted with a towfold goal: on the one hand - to study how users interact with DigiDrum and create guidelines for improving the setup, and on the other hand to investigate the relationship between tension and frequency independent damping coefficient (T and σ_0 respectively in Section 7) and user's perception of material stiffness.

As shown in Section 7, there are 3 different cases for both tension T and frequency independent damping σ_0. All combinations were tested, resulting in 9 different cases. Sound examples of each individual case can be found in [29]. Participants' experiences were evaluated through both qualitative and quantitative methods, namely by: (1) asking them during the test how they rate the stiffness of the material in each of the 9 cases, (2) a questionnaire

including questions about their relationship and experience with playing a musical instrument, virtual body ownership and whether they thought their interaction patterns changed between the different cases and (3) observation while the participants interacted with the setup to retrieve data on engagement and stroke patterns which possibly correlate to the haptics and sound.

8.1 Process for the User Study

Before the experiment, participants were told that they would be “drumming in VR", that their perception of the stiffness of the material they were interacting with was tested and that their performances did not need to be musical in any way. Furthermore, participants were told they would hear 9 different cases in between which the “parameters of the experience" would be changed and that for each of these cases they would have to rate the stiffness of the material they were interacting with on a scale of 1 to 7, 1 being “extremely soft or loose", 7 being “extremely stiff or hard". The order in which the cases were presented was randomised to reduce bias. Between cases, the participants did not take off the headset or headphones, and the authors noted their answers. After the test, the participants filled out a questionnaire with the following questions (the last two taken from [6]):

- I felt like the hands in the simulation were my own. (1-7 rating)
- In order to express your judgements to the questions during the simulation, you relied mainly on... (multiple answers possible: visuals | audio | haptics)
- In your opinion what was varying between each condition? (multiple answers possible: visuals | audio | haptics)

From participant-observation during the experiment and the the final two questions of the questionnaire, “Did your behaviour change between different cases, and if so what did you do differently?" and “Anything you would like to add?", information on the user interaction and the quality of the setup was collected.

The experiment was done on 16 participants, 9 of which were experienced musicians ($>$ 5 years of instrument practice). Three participants were drummers.

9 Results and Discussion

This section will give the results of the user study and discuss these. Due to the small sample size and some issues regarding interaction described at the end of this section, the presented results should be considered preliminary.

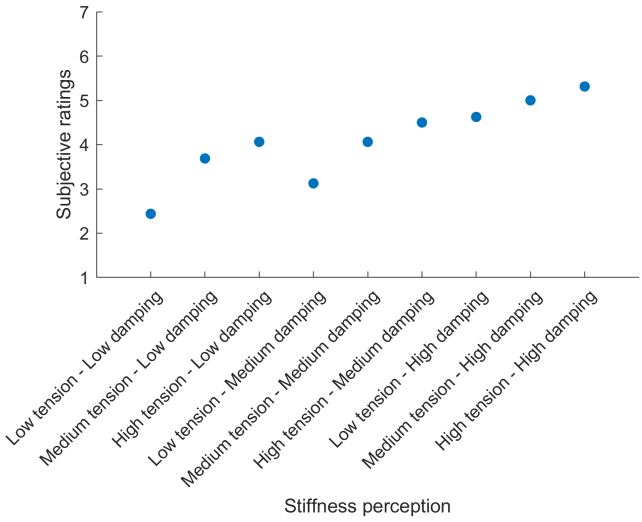


Fig. 4: Relation between stiffness perception and subjective ratings.

9.1 Statistical Analysis

The results of the stiffness ratings can be found in Figure 4. Intriguingly, there was a significant correlation between the cases sorted by damping first and then by tension (both sorted from low to high) and the subjective ratings (ρ = 0.9372, p < 0.01) using Spearman correlation. The Spearman methodology was used because the low number of values did not allow modelling a normal distribution [30]. A quasi-linear relationship between subjective stiffness perception and the values for tension and damping used by the simulation can be observed.

Figure 5 shows the average participant scores for each level of damping and tension both grouped in levels (low, medium and high). As previously hypothesised, the ratings of material stiffness increases with tension and damping.

A statistical analysis was run on each single level based on non-parametric Mann-Whitney U-test with the results reported in Table 2. This test helps to identify significant differences between groups in presence of small samples made by ordinal variables. Abbreviations are T for “tension" and D for “damping" while letters L, M and H mean the levels “low", “medium" and “high". It is important to take into account the multi-comparison problem and in this case the threshold level for significance should be equal to 0.0056 following the Bonferroni correction.

As we can observe from Table 2, there isn’t a significant difference between “high tension – low damping" and “low tension – medium damping" (p =

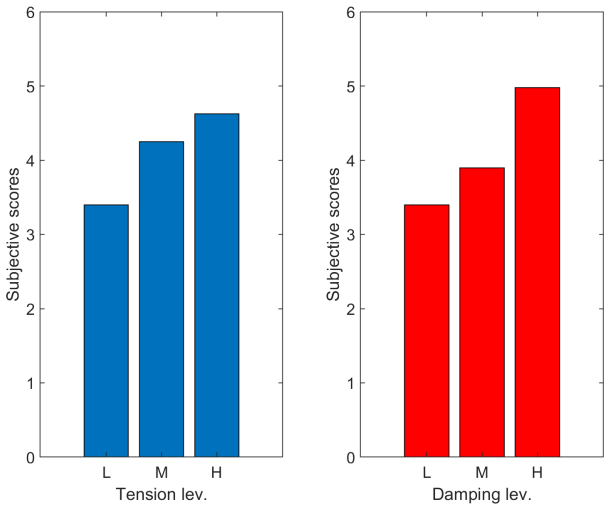


Fig. 5: Subjective ratings grouped by different tension and damping levels.

Mann-Whitney U-test [p-values]									
	LT LD	MT LD	HT LD	LT MD	MT MD	HT MD	LT HD	MT HD	HT HD
LT LD		.0642	.0163	.1268	.0049	.0041	.0034	.0004	.0002
MT LD	.0642		.6463	.3465	.6582	.1802	.1584	.034	.0091
HT LD	.0163	.6463		.2123	.8933	.5413	.4455	.1737	.0915
LT MD	.1268	.3465	.2123		.0791	.0254	.0283	.0012	.0009
MT MD	.0049	.6582	.8933	.0791		.3191	.3114	.0449	.0174
HT MD	.0041	.1802	.5413	.0254	.3191		.7732	.5214	.1383
LT HD	.0034	.1584	.4455	.0283	.3114	.7732		.7725	.3424
MT HD	.0004	.034	.1737	.0012	.0449	.5214	.7725		.2991
HT HD	.0002	.0091	.0915	.0009	.0174	.1383	.3424	.2991	
Note: Bonferroni adjusted significance threshold for multi-comparison p<0.0056									

Table 2: Mann-Whitney U-test (p-values).

0.2123) suggesting that the linear relation shown in Figure 5 holds despite the discontinuity between points as seen on the scatterplot. However, we should consider it similar to a monotonically increasing function rather than a pure linear trend.

Lastly, Table 3 shows group comparisons between the three different values of damping and tension. A significant difference in participant's ratings between medium-high damping and low-high damping levels can be noticed while tension shows significance only between low to high tension. It can be deducted from the results that damping is a more important factor than tension in material stiffness perception. This result was unexpected, as it was hypothesised that tension would be the most dominant factor in stiffness perception. Additionally, it appears difficult for participants to evaluate low to

Different levels of tension/damping		p-values
Low Damping	Medium Damping	0.2560
Medium Damping	High Damping	0.0078
Low Damping	High Damping	6.5071e-04
Low Tension	Medium Tension	0.0950
Medium Tension	High Tension	0.4268
Low Tension	High Tension	0.0297

Table 3: Comparison between different levels of tension and damping.

medium levels of both damping and tension. In a future test, the values could be chosen differently, or more alternatives for the parameter values could be investigated to better see the perceptual differences between these values.

9.2 Statistical Analysis: Reliability

Individual ratings were initially analysed with Cronbach's alpha [31] to test the internal consistency of the responses. This measure is generally known as a metric to validate a questionnaire with higher values of alpha as those more desirable. The non-standardised Cronbach's alpha value was 0.6348 while the standardised value reached 0.6589. According to [32], a value between 0.6 to 0.7 is questionable (questionnaire scale is not fully reliable) with 0.7 as the threshold for an acceptable test. Despite the outcomes being slightly below threshold (probably caused by subjective difficulties in evaluating stiffness), it appears that in future a good reliability can be reached by increasing the sample size. Moreover, if we don't consider all factors loadings as evenly distributed, we could assume that the Cronbach's alpha underestimates the true reliability.

9.3 Questionnaire

The questionnaire results in Table 4 show that the participants generally found that the hands in the simulation were their own. This proves that the Leap Motion is a good way to track the hands and that it was well implemented. The visuals had no influence on participants' judgement, probably because they were unchanged. The audio seemed to be the most predominant feature the participants focused on when expressing their judgements (93.8%). Haptics for expressing judgements was only chosen by 5 participants (31.3%). In the future, removing the audio, only leaving the haptics might be a better way to force the participants to use their sense of touch and test the influence of this modality on perception.

Question	Result
I felt like the hands in the simulation were my own. (1-7 rating)	$\mu = 5.44$, $\sigma = 1.26$
In order to express your judgements to the questions during the simulation, you relied mainly on... (visuals | audio | haptics)	visuals: 0, audio: 15, haptics: 5
In your opinion what was varying between each condition? (visuals | audio | haptics)	visuals: 0, audio: 14, haptics: 10

Table 4: Questionnaire results. The last two questions were taken from [6].

9.4 Qualitative Observations

From participant observation during the experiment, comments they gave during and after the test, and the two last (open) questions of the questionnaire (see Section 8) additional findings were compiled.

Due to the fact that the virtual drum was placed slightly higher than the physical drum (see Section 5), many participants interacted with the air above the drum rather than finishing their stroke to actually hit the drum. This was an issue, as the haptic sensation would not be felt in that case. This might also explain the result of the second question in Table 4. Either before or during the test, the participants were instructed to finish their stroke to actually physically interact with the drum.

The interaction was programmed in such a way, that when a tracked hand collides with the virtual drum, this hand would not be able to trigger the physical model until it was completely out of the “collision zone". Due to the misalignment mentioned above, many interactions were not captured. Again, either before or during the experiment, the participants were instructed to make longer movements to ensure that their hands were completely outside of this “collision zone" before interacting with the drum again.

Another technical issue was that sometimes participants would look forward rather than down to the hands. This caused the hands not to be tracked anymore as the Leap Motion was mounted on the HMD. A solution for this would be to mount it at a lower angle rather than straight forward (as is the current case).

The experiment could be improved by addressing the above interaction issues to yield stronger data. The issues could potentially be solved by adding a more precise and reliable sensor to the setup, such as a contact microphone placed on the drum membrane. Even though a feedback loop could occur due to the haptuator being present on the same membrane, there is a potential to filter out its vibrations and only use the transients due to the interaction with

the membrane for control.

Some participants commented that they would have liked to have reference points for “the stiffest" and “the softest" cases before testing as they said they would have judged the first few cases differently if they had known these references in advance. This could, however, bias the participants’ answers.

The movements of participants were observed during the test and sporadically noted. There was a small tendency towards slower and longer movements in the case of lower tension and faster and shorter movements in the opposite case, but as these observations were not done systematically, to be able to say anything about this, this should be properly tested, possibly using raw data from the hand tracking.

10 Conclusion

In this paper, we presented and evaluated a novel VRMI where a physical drum was enhanced by VR. The physical drum was augmented by a vibration motor and the sound was simulated using a physical model of a drum membrane. In an experiment run during the study, preliminary results show that higher values for both tension and damping increase the perception of material stiffness of the drum membrane, as hypothesised. However, the damping appeared to be a more important factor in this perception than the tension, which was contrary to expectations.

In future work, improving the experiment by, for example, adding a contact microphone to the membrane for more accurate control and re-conducting the experiment with a larger sample size will be necessary to validate or improve the results presented in this paper.

Other future work includes decoupling the audio and the haptics, to test the perceptual influence of each individual modality separately. More alternatives of the parameter values could be presented in a future test to more deeply investigate the connection between parameter values and stiffness perception.

Additionally, the tracking of the user’s hands should be improved by mounting the Leap Motion more downwards on the HMD. Furthermore, the virtual and physical drum should be better aligned in space as to make the interaction less confusing and more intuitive. Lastly, in order to test whether the interaction patterns change depending on the changes in parameters, the raw data from the hand tracking should be analysed.

Acknowledgments

The authors wish to thank Stefania Serafin, Cumhur Erkut, Niels Nilsson, Rolf Nordahl, and Michele Geronazzo – our teachers of the “Virtual, Augmented, Mixed realities" PhD course at Aalborg University Copenhagen – for their

help, and all the participants who were kind enough to participate in our experiment.

11 References

[1] S. Serafin, N. Nilsson, C. Erkut, and R. Nordahl, "Virtual reality and the senses - whitepaper," *Danish Sound Innovation Network*, 2017.

[2] S. Serafin, C. Erkut, J. Kojs, N. Nilsson, , and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, 2016.

[3] E. Berdahl, H.-C. Steiner, and C. Oldham, "Practical hardware and algorithms for creating haptic musical instruments," in *NIME*, 2008.

[4] M. S. O'Modhrain, *Playing by Feel: Incorporating Haptic Feedback into Computer-Based musical Instruments*. Ph.D. Dissertation, Stanford University, 2001.

[5] S. Dahl, "Playing the accent - comparing striking velocity and timing in an ostinato rhythm performed by four drummers," *Acta Acustica united with Acustica*, vol. 90, pp. 762 – 776, 2004.

[6] F. Avanzini and P. Crosato, "Haptic-auditory rendering and perception of contact stiffness," *Lecture Notes in Computer Science*, vol. 4129, pp. 24–35, 2006.

[7] A. Passalenti, R. Paisa, N. Nilsson, N. Andersson, F. Fontana, R. Nordahl, and S. Serafin, "No strings attached: Force and vibrotactile feedback in a virtual guitar simulation," *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1116–1117, 2019.

[8] 3D Systems, Inc, "3D Systems Touch Haptic Device," accessed November 4, 2019, available at https://www.3dsystems.com/haptics-devices/touch.

[9] J. Liu and H. Ando, "Hearing how you touch: Real-time synthesis of contact sounds for multisensory interaction," *Conference on Human System Interactions*, pp. 275–280, 2008.

[10] E. Berdahl, B. Verplank, J. O. Smith, and G. Niemeyer, "A physically intuitive haptic drumstick," in *ICMC*, 2007.

[11] K. Barnett, "A theoretical construct of the concepts of touch as they relate to nursing," *Nursing Research*, vol. 21, pp. 102–110, 1972.

[12] A. Gallace, M. K. Ngo, J. Sulaitis, and C. Spence, "Multisensory presence in virtual reality: possibilities & limitations," *Multiple sensorial media advances and applications: New developments in MulSeMedia*, pp. 1–38, 2012.

[13] M. Blatow, E. Nennig, A. Durst, K. Sartor, and C. Stippich, "fMRI reflects functional connectivity of human somatosensory cortex," *Neuroimage*, vol. 37, pp. 927–936, 2007.

[14] T. Manzoni, F. Conti, and M. Fabri, "Callosal projections from area SII to SI in monkeys: Anatomical organization and comparison with association projections," *Journal of Comparative Neurology*, vol. 252, pp. 245–263, 1986.

[15] S. B. Eickhoff, A. Schleicher, K. Zilles, and K. Amunts, "The human parietal operculum. I. Cytoarchitectonic mapping of subdivisions," *Cerebral cortex*, vol. 16, no. 2, pp. 254–267, 2005.

[16] D. N. Saito, T. Okada, M. Honda, Y. Yonekura, and N. Sadato, "Practice makes perfect: The neural substrates of tactile discrimination by Mah-Jong experts include the primary visual cortex," *BMC Neuroscience*, vol. 7, no. 79, 2007.

[17] Y. Cao, B. L. Giordano, F. Avanzini, and S. McAdams, "The dominance of haptics over audition in controlling wrist velocity during striking movements," *Experimental Brain Research*, vol. 234, pp. 1145—1158, 2016.

[18] UltraLeap Ltd, "Leap Motion," accessed November 4, 2019, available at https://www.leapmotion.com/.

[19] A. Bodegard, S. Geyer, C. Grefkes, K. Zilles, and P. Roland, "Hierarchical processing of tactile shape in the human brain," *Neuron*, vol. 31, pp. 317–328, 2001.

[20] W. Penfield and T. L. Rasmussen, *The cerebral cortex of man: A clinical study of localization of function*. London: Macmillan, 1950.

[21] A. Gallace and C. Spence, *Touch and the body: The role of the somatosensory cortex in tactile awareness*. Psyche: An Interdisciplinary Journal of Research on Consciousness, 2010.

[22] F. Pavani, C. Spence, and J. Driver, "Visual capture of touch: Out-of-the-body experiences with rubber gloves," *Psychological Science*, vol. 11, pp. 353–359, 2000.

[23] M. Schaefer, H. Flor, H. J. Heinze, and M. Rotte, "Dynamic modulation of the primary somatosensory cortex during seeing and feeling a touched hand," *Neuroimage*, vol. 29, pp. 587–592, 2006.

[24] S. O'Modhrain and R. B. Gillespie, *Once More, with Feeling: Revisiting the Role of Touch in Performer-Instrument Interaction*, 2018.

[25] Unity Technologies, "Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations," accessed November 4, 2019, available at https://unity.com/.

[26] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1998.

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

[28] S. Van Duyne and J. O. Smith, "The 2-D digital waveguide mesh," *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 177–180, 1993.

[29] S. Willemsen, "Sound Files (MembraneOutputs.zip)," accessed April 26, 2020, available at https://github.com/SilvinWillemsen/Membrane/blob/Paper/Sound_Files/MembraneOutputs.zip.

[30] R. Kirk, *Statistics: an introduction*. Nelson Education, 2007.

[31] L. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297—-334, 1951.

[32] P. Kline, *The handbook of psychological testing (2nd ed.)*. London: Routledge, 2000.

Paper G

Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21), 2021.

Abstract

For physical modelling sound synthesis, many techniques are available; time-stepping methods (e.g., finite-difference time-domain (FDTD) methods) have an advantage of flexibility and generality in terms of the type of systems they can model. These methods do, however, lack the capability of easily handling smooth parameter changes while retaining optimal simulation quality and stability, something other techniques are better suited for. In this paper, we propose an efficient method to smoothly add and remove grid points from a FDTD simulation under sub-audio rate parameter variations. This allows for dynamic parameter changes in physical models of musical instruments. An instrument such as the trombone can now be modelled using FDTD methods, as well as physically impossible instruments where parameters such as e.g. material density or its geometry can be made time-varying. Results show that the method does not produce (visible) artifacts and stability analysis is ongoing.

1 Introduction

The operation of most musical instruments can be subdivided into excitation and resonator components [1]. Examples of excitation-resonator combinations are the bow and violin and the lips and trumpet. In most instruments, the parameters describing the excitation are continuously varied by the performer to play the instrument. As an example, the bow velocity, bow position and bow force for stringed instruments, and lip pressure and frequency for brass instruments. Naturally, the resonator is also altered by fingering the strings of the violin or pressing valves on the trumpet to change the instrument pitch. But, even under such variable playing conditions, physical properties of the resonators do not change: the string length and tension stay the same and the total tube length remains unchanged; it is only the portions that resonate that are shortened or lengthened.

There are several examples where the parameters of the resonator are also modified. A prime example of this is the trombone, where the tube length is dynamically changed in order to generate different pitches. The slide whistle is another example in this category. Guitar strings are another category where the tension can be smoothly modulated during performance using the fretting finger or a whammy bar to create smooth pitch glides. The same kind of tension modulation is used for the membranes of timpani or “hourglass drums" to change the pitch. It is these direct parameter modifications of the resonators that we are interested in simulating. In addition to simulating existing instruments, one could potentially simulate instruments that can be manipulated in physically impossible ways. Examples of this could be to dynamically change material properties such as density or stiffness, or even the geometry and size of the instrument where this is physically impossible.

Finite-difference time-domain (FDTD) methods are flexible and generalisable techniques which have seen increased use in physical modelling sound synthesis applications [2]. The normal approach, for a given system such as a musical instrument, is to describe its motion by a set of partial differential equations (PDEs). The instrument is then represented over a spatial grid, and a time-stepping method is developed, yielding a fully discrete approximation to the target PDE system.

In many cases, the system itself is static, so that the defining parameters do not change over time. In others, such as the trombone and others mentioned above, this is not the case, and various technical challenges arise when trying to design a simulation using FDTD methods; all relate to the choice of the spatial grid. For example, the grid density is usually closely tied to the parameters themselves through a stability condition. Also, adding and removing points from the grid is nontrivial and can cause audible artifacts and new stability concerns. The default approach of defining a grid globally, according to a very conservative stability condition, as done in [3], is possible, but introduces numerical dispersion and bandlimiting effects. Full-grid interpolation [2, Ch. 5] could be used to change between grid configurations, but extremely high sample rates are necessary to avoid audible artifacts and low-passing effects, rendering any implementation offline.

In this paper, a new method is proposed, allowing the efficient and smooth insertion and deletion of grid points from 1D finite-difference grids to allow for dynamic parameter changes. We are interested in varying parameters 'slowly' (i.e., at sub-audio rate corresponding to human gestural control). In a companion paper we present a physical model of the trombone using the method proposed in this paper [4]. Notice that other techniques do allow for dynamic parameter changes but come with their own drawbacks [2]. Examples of dynamic parameters using modal synthesis [5] are shown in [6, 7] and digital waveguides [8] are shown in [9].

This paper is structured as follows: Section 2 presents the 1D wave equation, to be used as an illustrative example for the proposed method. Section 3 gives an introduction to numerical methods, stability and simulation quality. The proposed method for dynamic grids is then presented in Section 4 and applied to the 1D wave equation. Section 5 shows the results of an analysis performed on the method, which are discussed in Section 6. Finally, concluding remarks and future perspectives are given in 7.

2 Continuous Systems

The wave equation is a useful starting point for investigations of time-varying behaviour in musical instruments. In 1D, the wave equation may be written as

$$\frac{\partial^2 q}{\partial t^2} = c^2 \frac{\partial^2 q}{\partial x^2} \,, \tag{1}$$

and is defined over spatial domain $x \in [0, L]$, for length L (in m) and time $t \geq 0$ (in s). c (in m/s) is the wave speed. The dependent variable $q = q(x, t)$ in Eq. (1) may be interpreted as the transverse displacement of an ideal string, or the acoustic pressure in the case of a cylindrical tube. Two possible choices of boundary conditions are

$$q(0, t) = q(L, t) = 0 \quad \text{(Dirichlet)}, \tag{2a}$$

$$\frac{\partial}{\partial x} q(0, t) = \frac{\partial}{\partial x} q(L, t) = 0 \quad \text{(Neumann)}, \tag{2b}$$

and describe 'fixed' or 'free' boundary respectively in the case of an ideal string, and 'open' or 'closed' conditions respectively in the case of a cylindrical acoustic tube.

2.1 Dynamic parameters

In the case of the 1D wave equation, only the wave speed c and length L can be altered (in the case of an acoustic tube, only L is variable, and for a string, c could exhibit variations through changes in tension). If the same boundary condition is used at both ends of the domain, and under static conditions, the fundamental frequency f_0 of vibration can be calculated according to

$$f_0 = \frac{c}{2L} \,. \tag{3}$$

In the dynamic case, and under slow (sub-audio rate) variations of c or L, Eq. (3) still holds approximately. From Eq. (3), one can easily conclude that in terms of fundamental frequency, halving the length in Eq. (1) is identical to doubling the wave speed and vice versa. Looking at Eq. (1) in isolation, f_0 is the only behaviour that can be changed. One can thus leave L fixed and allow time variation in c, so that $c = c(t)$, which will prove easier to work with in the following sections. This fact can more easily be seen if Eq. (1) is scaled or non-dimensionalised as in [2], where scaled domain $x' = x/L \;\Rightarrow\; x' \in [0, 1]$ and $\gamma = c/L$ such that $f_0 = \gamma/2$. For clarity, however, we will employ a fully dimensional representation here.

3 Numerical methods

This section will provide a brief introduction to physical modelling using FDTD methods, including details on stability and quality of the simulations based on these methods. In this section, c is assumed constant.

3.1 Discretisation

In FDTD methods, the first step is the definition of a grid. The spatial variable can be discretised using $x_l = lh$ with integer $l \in \{0, \ldots, N\}$. The grid spacing h (in m) is the distance between adjacent grid points, and the total number of points covering the domain, including endpoints, is $N + 1$. Here, integer N describes the total number of intervals between the grid points, and thus the total domain length is $L = Nh$. The temporal variable can be discretised using $t_n = nk$ with positive integer n, time step $k = 1/f_\mathrm{s}$ (in s) for sample rate f_s (in Hz). The state variable q can then be approximated using $q_l^n \approxeq q(x = lh, t = nk)$.

The following operators can then be applied to q_l^n to get the following approximations to the derivatives in Eq. (1)

$$\delta_{tt} q_l^n = \frac{1}{k^2} \left(q_l^{n+1} - 2q_l^n + q_l^{n-1} \right) \approx \frac{\partial^2 q}{\partial t^2} , \tag{4a}$$

$$\delta_{xx} q_l^n = \frac{1}{h^2} \left(q_{l+1}^n - 2q_l^n + q_{l-1}^n \right) \approx \frac{\partial^2 q}{\partial x^2} . \tag{4b}$$

Substituting these definitions into Eq. (1) yields the following finite-difference (FD) scheme

$$\delta_{tt} q_l^n = c^2 \delta_{xx} q_l^n . \tag{5}$$

Expanding the operators as in (4) and solving for q_l^{n+1} yields the following update equation

$$q_l^{n+1} = 2q_l^n - q_l^{n-1} + \lambda^2 \left(q_{l+1}^n - 2q_l^n + q_{l-1}^n \right) , \tag{6}$$

which is suitable for direct software implementation. Here,

$$\lambda = \frac{ck}{h} \tag{7}$$

is referred to as the Courant number, constrained by numerical stability conditions, and also has an impact on the quality and behaviour of the simulation. This will be described in detail in Sections 3.2 and 3.3.

In the FD scheme described in Eq. (5), the boundary locations are at $l = 0$ and $l = N$. Substituting these locations into Eq. (6) seemingly introduces the need of grid points outside of the defined domain, namely q_{-1}^n and q_{N+1}^n.

These can be referred to as *virtual grid points* and can be accounted for using the boundary conditions in Eq. (2). Discretising these yields

$$q_0^n = q_N^n = 0, \quad \text{(Dirichlet)} \tag{8a}$$

$$\delta_x . q_0^n = \delta_x . q_N^n = 0, \quad \text{(Neumann)} \tag{8b}$$

where

$$\delta_x . q_l^n = \frac{1}{2h}\left(q_{l+1}^n - q_{l-1}^n\right) \quad \approx \quad \frac{\partial q}{\partial x} \tag{9}$$

is a second-order accurate approximation of the first-order spatial derivative. The Dirichlet condition in (8a) says that the displacements of q at the boundary locations are always 0. In practice, this means that these grid points do not need to be updated and the spatial range of calculation for Eq. (6) then becomes $l \in \{1, \ldots, N-1\}$. If the Neumann condition is used, the boundary points do need to be updated as these are not necessarily 0; rather, their 'slope' is 0. Eq. (8b) can then be expanded to yield defnitions for these virtual grid points

$$q_{-1}^n = q_1^n \quad \text{and} \quad q_{N+1}^n = q_{N-1}^n \,. \tag{10}$$

Now that the full system is described, audio output at sample rate f_s can be drawn from the state q_l^n in Eq. (6) at $0 < l < N$ (when using Dirichlet boundary conditions).

3.2 Stability

Explicit FDTD methods for hyperbolic systems such as the 1D wave equation must necessarily satisfy a stability condition. In the case of the update in Eq. (6) it can be shown – using von Neumann analysis [10] – that the system is stable if

$$\lambda \leq 1, \tag{11}$$

which is referred to as the Courant-Friedrichs-Lewy (CFL) condition. The more closely λ approaches this condition with equality, the higher the quality of the simulation (see Section 3.3) and if $\lambda = 1$, Eq. (6) yields an exact solution to Eq. (1). If $\lambda > 1$ the system will become unstable. Recalling (7), Eq. (11) can be rewritten in terms of grid spacing h to get

$$h \geq ck. \tag{12}$$

This shows that the CFL condition in (11) puts a lower bound on the grid spacing, determined by the sample rate and wave speed. Usually, the following steps are taken to calculate λ:

$$h := ck, \;\; N := \left\lfloor \frac{L}{h} \right\rfloor, \;\; h := \frac{L}{N}, \;\; \lambda := \frac{ck}{h}, \tag{13}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. In other words, condition (12) is first satisfied with equality and used to calculate number of intervals N. Thereafter, h is recalculated based on integer N and used to calculate λ. The calculation of λ in Eq. (13) can be compactly rewritten as

$$\lambda = \frac{ck}{L} \cdot \left\lfloor \frac{L}{ck} \right\rfloor . \tag{14}$$

The flooring operation causes the CFL condition in (11) to not always be satisfied with equality and results in a reduced simulation quality described in the following section.

3.3 Simulation Quality

Choosing $\lambda < 1$ in Eq. (6) will decrease the simulation quality in two ways. Firstly, it will decrease the maximum frequency that the simulation is able to produce, i.e., it will decrease the bandwidth of the output sound of the system.

By analysing the scheme in Eq. (6), it can be shown that the maximum frequency produced by the system can be calculated using $f_{\text{max}} = f_{\text{s}} \sin^{-1}(\lambda)/\pi$ [2, Chap. 6]. Note that only a small deviation of λ from condition (11) leads to a large reduction in output bandwidth. Secondly, choosing $\lambda < 1$ causes numerical dispersion. Harmonic partials become unnaturally closely spaced at higher frequencies (i.e. spurious inharmonicity increases) as λ decreases, which is generally undesirable.

4 The Dynamic Grid

The time variation of the wave speed c leads to various complications in the simulation framework presented above. First of all, a change in c causes a change in λ according to Eq. (14), affecting the simulation quality and bandwidth. Secondly, and more importantly, a change in c could result in a change in N through Eq. (13). As N directly relates to the number of grid points, this raises questions as to *where* and especially *how* one would add and remove points to the grid according to the now-dynamic wave speed.

We propose a method that allows for a non-integer number of intervals to smoothly change between grid configurations, i.e, the number of grid points used. This removes the necessity of the flooring operation in Eqs. (13) and (14), and consequently satisfies the CFL condition in (11) with equality at all times. Introducing fractional number of intervals $\mathcal{N}$, where $N = \lfloor \mathcal{N} \rfloor$, Eq. (3) can be rewritten in terms of $\mathcal{N}$ by substituting the calculation of N from (13)

into Eq. (3) (using $h = ck$) yielding

$$f_0 = \frac{1}{2\mathcal{N}k} \quad \text{with} \quad \mathcal{N} = L/h. \tag{15}$$

This shows that if $\lambda = 1$, $\mathcal{N}$ solely determines the fundamental frequency of the simulation.

Ideally, a method that dynamically changes the grid size of a FD scheme should

r1. generate an output with a fundamental frequency f_0 which is proportional to the wave speed c ($f_0 \propto c$),

r2. allow for a fractional number of intervals $\mathcal{N}$ to smoothly (without audible artifacts) transition between different grid configurations,

r3. generate an output containing $N - 1$ modes which are integer multiples of the fundamental ($f_p = f_0 p$ with integer p),

r4. work in real time to have a playable simulation.

These requirements will be used in Section 6 to evaluate the proposed method.

4.1 Proposed Method

In the following, the location of a grid point (in m from the left boundary) q_l at time index n is denoted by $x^n_{q_l}$. Furthermore, some variables are now time dependent as indicated by superscript n. These are c^n, h^n, $\mathcal{N}^n$, N^n and f_0^n.

4.1.1 System Setup

Consider two grid functions, $u^n_{l_u}$ and $w^n_{l_w}$ defined over discrete domains $l_u \in \{0, \ldots, M^n\}$ and $l_w \in \{0, \ldots, M^n_w\}$ respectively with integers $M^n = \lceil 0.5N^n \rceil$ with $\lceil \cdot \rceil$ denoting the ceiling operation and $M^n_w = \lfloor 0.5N^n \rfloor$, i.e., half the number of points allowed by the stability condition, plus one for overlap. The two grid functions are assumed to lie adjacent to each other on the same domain x. For now, the grid locations $l_u = M^n$ and $l_w = 0$ are assumed to overlap so that $x^n_{u_{M^n}} = x^n_{w_0} = M^n h^n$, and are referred to as the inner boundaries. The grid locations $l_u = 0$ and $l_w = M^n_w$ are placed at $x^n_{u_0} = 0$ and $x^n_{w_{M^n_w}} = L$ and will be referred to as the outer boundaries. See Figure 1a. The following boundary conditions are then imposed:

$$u^n_0 = w^n_{M^n_w} = 0, \quad \text{(Dirichlet)} \tag{16a}$$

$$\delta_{x\cdot} u^n_{M^n} = \delta_{x\cdot} w^n_0 = 0. \quad \text{(Neumann)} \tag{16b}$$

In other words, grid points at the outer boundaries are fixed, according to the usual Dirichlet condition, and those at the inner boundaries are free. It is important to note that the Neumann condition is just used as a starting point for the method here, but will be modified in Section 4.1.2. The systems can then be connected at the inner boundaries using a rigid connection

$$u^n_{M^n} = w^n_0, \quad \text{if } x^n_{u_{M^n}} = x^n_{w_0}. \tag{17}$$

Notice that this condition only needs to be satisfied when the inner boundaries perfectly overlap, which is not always the case when c^n is varied (see Section 4.1.2).

To sum up, a grid function with N intervals as per Eq. (13) is divided into two separate subsystems connected at their respective inner boundaries.

With the above boundary conditions imposed, the following state vectors can be defined:

$$\mathbf{u}^n = [u^n_1, \dots, u^n_{M^n}]^T, \text{ and } \mathbf{w}^n = [w^n_0, \dots, w^n_{M^n_w - 1}]^T, \tag{18}$$

with T denoting the transpose operation, and have M^n and M^n_w points respectively. Note that the grid points at the outer boundaries are excluded as they are 0 at all times due to the Dirichlet boundary condition in (8a). A vector concatenating (18) is then defined as

$$\boldsymbol{\mathcal{U}}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix}. \tag{19}$$

Even though the new system has an extra (overlapping) grid point, the behaviour of the new system should be identical to that of the original system in Eq. (5) with (static) $\mathcal{N}^n = N^n$. That this holds will be shown below.

Using $u^n_{l_u}$ and $w^n_{l_w}$ in the context of the 1D wave equation, a system of FD schemes can be defined as

$$\begin{cases} \delta_{tt} u^n_{l_u} = (c^n)^2 \delta_{xx} u^n_{l_u} + J_u(x^n_{u_{M^n}}) F^n, \\ \delta_{tt} w^n_{l_w} = (c^n)^2 \delta_{xx} w^n_{l_w} - J_w(x^n_{w_0}) F^n, \end{cases} \tag{20}$$

with spreading operators

$$\begin{aligned} J_u(x^n_i) &= \begin{cases} \frac{1}{h^n}, & l_u = \lfloor x^n_i / h^n \rfloor \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \\ J_w(x^n_i) &= \begin{cases} \frac{1}{h^n}, & l_w = \lfloor x^n_i / h^n \rfloor - M^n \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{21}$$

applying the effect of the connection F^n (in $\mathrm{m}^2/\mathrm{s}^2$) to grid points $u^n_{M^n}$ and

w_0^n respectively. Expanding the spatial operators in system (20) at the inner boundaries, recalling the Neumann condition in (16b) and the definition for the virtual grid points needed for this condition in Eq. (10) yields

$$\begin{cases} \delta_{tt} u_{M^n}^n = \frac{\lambda^2}{k^2}(2u_{M^n-1}^n - 2u_{M^n}^n) + \frac{1}{h^n} F^n, \\ \delta_{tt} w_0^n = \frac{\lambda^2}{k^2}(2w_1^n - 2w_0^n) - \frac{1}{h^n} F^n. \end{cases} \tag{22}$$

It is important to note that the time index n in M^n will not be affected by the δ_{tt} operator and all obtained terms after expansion (Eq. (4a)) will use the same value for M^n. Because of the rigid connection in (17), it is also true that $\delta_{tt} u_{M^n}^n = \delta_{tt} w_0^n$ (if $x_{u_{M^n}}^n = x_{w_0}^n$), and F^n can be calculated by setting the right side of the equations in (22) equal to each other:

$$\begin{aligned} \frac{\lambda^2}{k^2}(2u_{M^n-1}^n - 2u_{M^n}^n) + \frac{1}{h^n} F^n &= \frac{\lambda^2}{k^2}(2w_1^n - 2w_0^n) - \frac{1}{h^n} F^n, \\ F^n &= h^n \frac{\lambda^2}{k^2}(w_1^n - u_{M^n-1}^n). \end{aligned}$$

Substituting this into system (22) after expansion of the second-time derivative yields the update of the inner boundaries

$$\begin{cases} u_{M^n}^{n+1} = 2u_{M^n}^n - u_{M^n}^{n-1} + \lambda^2(u_{M^n-1}^n - 2u_{M^n}^n + w_1^n), & \text{(23a)} \\ w_0^{n+1} = 2w_0^n - w_0^{n-1} + \lambda^2(u_{M^n-1}^n - 2w_0^n + w_1^n), & \text{(23b)} \end{cases}$$

which, (again, recalling Eq. (17)) are indeed equivalent expressions for the connected point which is necessary to satisfy the rigid connection. System (20) can be shown to exhibit behaviour identical to that of the original scheme in Eq. (5) using (static) $\mathcal{N}^n = N^n$. In (23), w_1^n in Eq. (23a) acts as virtual grid point $u_{M^n+1}^n$, and $u_{M^n-1}^n$ in (23b) as virtual grid point w_{-1}^n. This important fact is what the proposed method relies on and will be extensively used in the following.

4.1.2 Changing the Grid

The previous section describes the case in which $\mathcal{N}^n$ is an integer. We now continue by varying c^n such that this is not the case.

The locations of the outer boundaries $x_{u_0}^n$ and $x_{w_{M_w}}^n$ are fixed:

$$x_{u_0}^n = x_{u_0}^0 = 0 \quad \text{and} \quad x_{w_{M_w^n}}^n = x_{w_{M_w^n}}^0 = L \quad \forall n.$$

If the wave speed c^n is then decreased, and consequently the grid spacing h^n according to Eq. (12) (with equality), all other points move towards their respective outer boundary (see Figure 1b). Calculating h^n this way allows this method to always satisfy the CFL condition in Eq. (11) with equality, solving

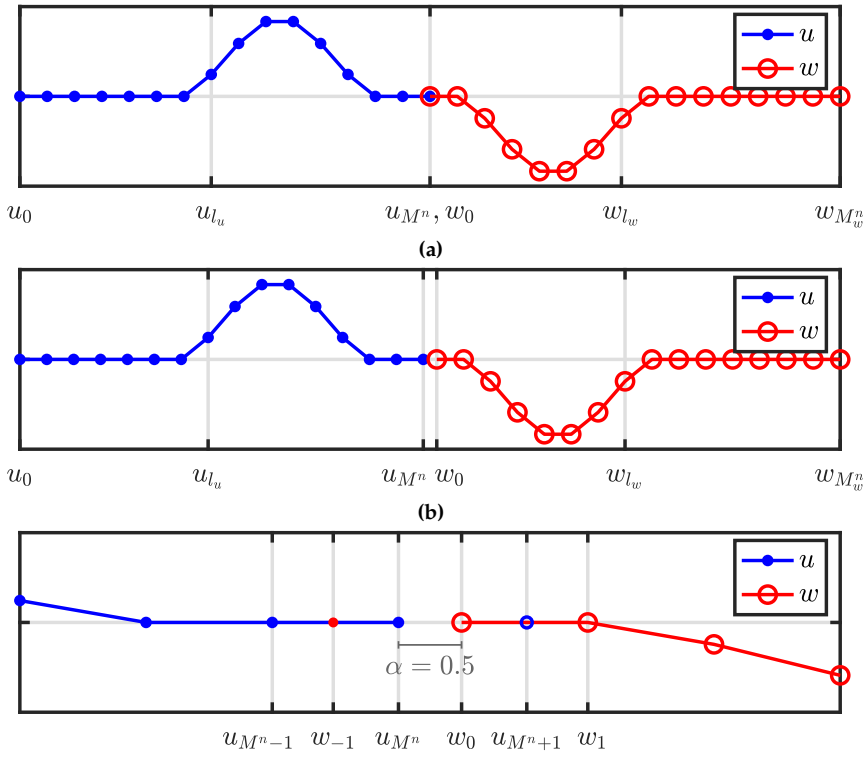


Fig. 1: *Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but 'x^n' is omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($\mathcal{N}^n = 30$, $x^n_{u_{M^n}} = x^n_{w_0}$). (b) When c^n – and consequently h^n – are decreased and the positions of the grid points change ($\mathcal{N}^n = 30.5$, $x^n_{u_{M^n}} \neq x^n_{w_0}$). (c) Figure 1b zoomed-in around the inner boundaries. The virtual grid points $u^n_{M^n+1}$ and w^n_{-1} are shown together with the distance between them expressed using α in Eq.* (24).

issues regarding simulation quality and numerical dispersion described in Section 3.3.

As mentioned in Section 4.1.1, the state of the virtual grid points at the inner boundaries are defined as $u^n_{M^n+1} = w^n_1$ and $w^n_{-1} = u^n_{M^n-1}$ when the inner boundaries perfectly overlap (i.e., $x^n_{u_{M^n}} = x^n_{w_0}$). If this is not the case ($x^n_{u_{M^n}} \neq x^n_{w_0}$) a Lagrangian interpolator $I(x^n_i)$ at location x^n_i (in m from the left boundary) can be used to calculate the value of these virtual grid points (also see Figure 1c for reference). The interpolator I is a row-vector with the same length as $\boldsymbol{\mathcal{U}}^n$ (from Eq. (19)) and its values depend on the interpolation order. In the following, the fractional part of $\mathcal{N}^n$ is defined as

$$\alpha = \alpha^n = \mathcal{N}^n - N^n, \tag{24}$$

and for clarity, I and $\boldsymbol{\mathcal{U}}^n$ are indexed by m. Now, consider the following quadratic interpolator

$$I_2(x_i^n) = \begin{cases} -(\alpha-1)/(\alpha+1), & m = m_i^n - 1 \\ 1, & m = m_i^n \\ (\alpha-1)/(\alpha+1), & m = m_i^n + 1 \\ 0, & \text{otherwise} \end{cases} \tag{25a}$$

and its flipped version

$$I_2^{\leftarrow}(x_i^n) = \begin{cases} (\alpha-1)/(\alpha+1), & m = (m_i^{\leftarrow})^n - 1 \\ 1, & m = (m_i^{\leftarrow})^n \\ -(\alpha-1)/(\alpha+1), & m = (m_i^{\leftarrow})^n + 1 \\ 0, & \text{otherwise} \end{cases} \tag{25b}$$

with $m_i^n = \lfloor x_i^n/h^n \rfloor$ and $(m_i^{\leftarrow})^n = \lfloor x_i^n/h^n + (1-\alpha) \rfloor$, where the shift in the latter is necessary to transform the location x_i^n to the correct indices of $\boldsymbol{\mathcal{U}}^n$. When applied to Eq. (19) this yields the definitions for the virtual grid points

$$u_{M^n+1}^n = I_2^{\leftarrow}(x_{u_{M^n+1}}^n)\boldsymbol{\mathcal{U}}^n = \frac{\alpha-1}{\alpha+1}u_{M^n}^n + w_0^n - \frac{\alpha-1}{\alpha+1}w_1^n, \tag{26a}$$

$$w_{-1}^n = I_2(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n = -\frac{\alpha-1}{\alpha+1}u_{M^n-1}^n + u_{M^n}^n + \frac{\alpha-1}{\alpha+1}w_0^n. \tag{26b}$$

These definitions for the virtual grid points at the inner boundaries will replace the Neumann condition in Eq. (16b). One can show that when $\mathcal{N}^n$ is an integer, and thus $\alpha = 0$, Eqs. (26a) and (26b) can be substituted as w_1^n and $u_{M^n-1}^n$ into Eqs. (23a) and (23b) respectively (as these acted as virtual grid points $u_{M^n+1}^n$ and w_{-1}^n). Then recalling Eq. (17) it can be seen that the system reduces to (23) and exhibits the same exact behaviour as the usual case in Eq. (5).

Now that the virtual grid points at the inner boundaries are not determined by the Neumann boundary condition in (16b), but rather by the definitions in Eqs. (26), system (20) can simply be re-written to

$$\begin{cases} \delta_{tt}u_{l_u}^n = (c^n)^2\delta_{xx}u_{l_u}^n, \\ \delta_{tt}w_{l_w}^n = (c^n)^2\delta_{xx}w_{l_w}^n, \end{cases} \tag{27}$$

where the Dirichlet condition in (16a) is (still) used for the outer boundaries and the Neumann condition at the inner boundaries in (16b) is replaced by the definitions in (26):

$$u_{M^n+1}^n = I_2^{\leftarrow}(x_{u_{M^n+1}}^n)\boldsymbol{\mathcal{U}}^n \quad \text{and} \quad w_{-1}^n = I_2(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n. \tag{28}$$

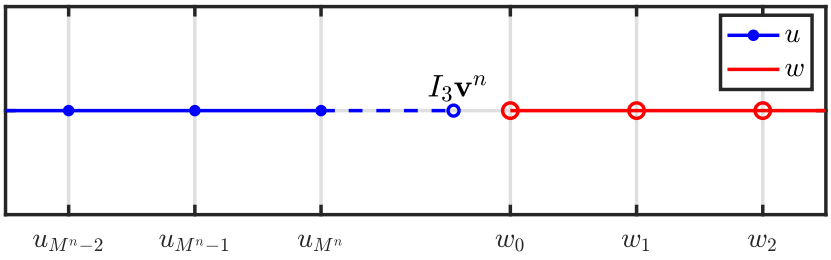


Fig. 2: *The moment when a point is added to* $\mathbf{u}$ *at location* $x^n_{u_{M^n+1}}$ *in Eq.* (29)*. This figure shows an extreme case where this location is far from* $x^n_{w_0}$*, i.e.,* $\alpha \not\approx 0$ *in Eq.* (30)*.*

4.1.3 *Adding and Removing Grid Points*

When c^n, and consequently h^n, are decreased and the inner boundary points surpass the virtual points (i.e. $x^n_{u_{M^n}} \leq x^n_{w_{-1}}$ and $x^n_{w_0} \geq x^n_{u_{M^n+1}}$), this means that $N^n > N^{n-1}$. A point is then added to the right boundary of u and the left boundary of w (for both time indices n and $n-1$) in an alternating fashion:

$$\begin{cases} \mathbf{u}^n = [(\mathbf{u}^n)^T, I_3\mathbf{v}^n]^T & \text{if } N^n \text{ is odd,} \\ \mathbf{w}^n = [I_3^{\leftarrow}\mathbf{v}^n, (\mathbf{w}^n)^T]^T & \text{if } N^n \text{ is even.} \end{cases} \tag{29}$$

Here,

$$\mathbf{v}^n = [u^n_{M^n-1}, u^n_{M^n}, w^n_0, w^n_1]^T,$$

and cubic Lagrangian interpolator

$$I_3 = \left[-\tfrac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} \quad \tfrac{2\alpha}{\alpha+2} \quad \tfrac{2}{\alpha+2} \quad -\tfrac{2\alpha}{(\alpha+3)(\alpha+2)} \right], \tag{30}$$

with $I_3^{\leftarrow}$ being a flipped, not shifted (as $I_2^{\leftarrow}$ in Eq. (25b)) version of (30). See Figure 2. Notice that $\mathcal{N}^n$ is only going to be slightly bigger than an integer at the moment that a point is added and Eq. (24) will return $\alpha \gtrsim 0$. This means that that $I_3 \approx [0, 0, 1, 0]$ and the displacement of the newly added point is nearly fully based on the grid point at the inner boundary of the other system.

Removing grid points happens when c^n, and consequently h^n, are increased and $x^n_{u_{M^n}} > x^n_{w_0}$ (or $N^n < N^{n-1}$). Grid points are simply removed from $\mathbf{u}$ and $\mathbf{w}$ (again for both n and $n-1$) in an alternating fashion according to

$$\begin{cases} \mathbf{u}^n = [u^n_0, u^n_1 ..., u^n_{M^n-1}]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w^n_1, w^n_2 ..., w^n_{M^n_w}]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{31}$$

In Eqs. (29) and (31), the even and odd conditions can be inverted. To keep the difference between u and w a maximum of one grid point, the ceiling and

flooring operations when calculating M^n and M_w^n will need to be inverted as well.

Until now, only adding and removing points in the center of the original system has been considered. This location could be moved anywhere along the grid, the limit being one point from the boundary. In other words, both $u_{l_u}^n$ and $w_{l_w}^n$ need to have at least one point (excluding the grid points at the outer boundaries). Furthermore, one does not have to add and remove points from $\mathbf{u}$ and $\mathbf{w}$ in an alternating fashion as in (29), but can just add to and remove from, for example, $\mathbf{u}$ leaving $\mathbf{w}$ the same size throughout the simulation. In the extreme case where $M^n = N^n - 1$ and $M_w^n = 1$ (leaving $w_{l_w}^n$ with only one moving grid point, w_0^n) the method still works.

4.1.4 Displacement correction

A problem that arises when increasing c^n, is that it is possible that the displacements $u_{M^n}^n \not\approx w_0^n$ at the time when a grid point needs to be removed. As the grid locations $x_{u_{M^n}}^n \approx x_{w_0}^n$ at the time of removal, this violates the rigid connection in (17) and causes audible artifacts. A method is proposed that decreases the relative displacement of the inner boundaries the closer their grid-locations are together, i.e., the closer α in (24) is to 0. We thus extend system (27) with an artificial spring force as

$$\begin{cases} \delta_{tt} u_{l_u}^n = (c^n)^2 \delta_{xx} u_{l_u}^n + J_u(x_{u_{M^n}}^n) F_{\mathrm{c}}^n, \\ \delta_{tt} w_{l_w}^n = (c^n)^2 \delta_{xx} w_{l_w}^n - J_w(x_{w_0}^n) F_{\mathrm{c}}^n. \end{cases} \tag{32}$$

Using centred temporal averaging and difference operators

$$\mu_{t\cdot} q_l^n = \frac{1}{2} \left(q_l^{n+1} + q_l^{n-1} \right), \tag{33a}$$

$$\delta_{t\cdot} q_l^n = \frac{1}{2k} \left(q_l^{n+1} - q_l^{n-1} \right), \tag{33b}$$

the correction effect is defined as

$$F_{\mathrm{c}}^n = \beta \left(\mu_{t\cdot} \eta^n + \sigma_0 \delta_{t\cdot} \eta^n \right), \tag{34}$$

with the difference in displacement between the inner boundaries

$$\eta^n \triangleq w_0^n - u_{M^n}^n, \tag{35}$$

and damping coefficient σ_0. Furthermore, β scales the effect of the displacement correction and is defined as

$$\beta = \beta(\alpha) = \frac{1 - \alpha}{\alpha + \varepsilon}, \tag{36}$$

where $\varepsilon \ll 1$ prevents a division by 0. Despite the operators in (33) introducing states at $n+1$, it is possible to calculate the force explicitly (such as in [2] or [11]). Furthermore, it can be shown that even when $\varepsilon = 0$ this calculation is always defined. In that case, as $\alpha \to 0$, $\beta \to \infty$ which acts as a rigid connection such as Eq. (17). Essentially, the displacement correction attempts to have $\eta^n \to 0$ in Eq. (35) as $\alpha \to 0$ to satisfy the rigid connection in Eq. (17). Although the correction presented here is not based on some physical process, it can be justified by the fact that large differences in displacement between two spatially adjacent points is not physical.

Notice that when c^n is decreased, the rigid connection will not be violated as $u^n_{M^n} \approx w^n_0$ when a point is added. This is due to the fact that $I_3 \approx [0, 0, 1, 0]$ and either $u^n_{M^n}$ or w^n_0 is the newly added point which almost solely based on the other.

4.2 Summary

Here, Section 4.1 is summarised and describes the final version of the proposed method.

The proposed method subdivides a grid function q^n_l with N intervals into two grid functions $u^n_{l_u}$ and $w^n_{l_w}$ with M^n and M^n_w intervals respectively for a total of $N^n + 2$ grid points. Knowing that $\lambda = 1\ \forall n$, Eq. (6), written for both grid functions, becomes

$$u^{n+1}_{l_u} = u^n_{l_u+1} + u^n_{l_u-1} - u^{n-1}_{l_u}, \tag{37a}$$

$$w^{n+1}_{l_w} = w^n_{l_w+1} + w^n_{l_w-1} - w^{n-1}_{l_w}. \tag{37b}$$

Due to the Dirichlet boundary condition in (16a) imposed at the outer boundaries of the system, u^n_0 and $w^n_{M_w}$ are 0 at all times and do not have to be included in the calculation. The ranges of calculation for Eq. (37a) and (37b) then become $l_u \in \{1, \dots, M^n\}$ and $l_w \in \{0, \dots, M^n_w - 1\}$ respectively.

The grid points at the inner boundaries are calculated by expanding (27) (ignoring the displacement correction for now)

$$u^{n+1}_{M^n} = u^n_{M^n+1} + u^n_{M^n-1} - u^{n-1}_{M^n}, \tag{38a}$$

$$w^{n+1}_0 = w^n_{-1} + w^n_1 - w^{n-1}_0, \tag{38b}$$

where virtual grid points $u^n_{M^n+1}$ and w^n_{-1} can be calculated using Eq. (26).

Then, when $N^n > N^{n-1}$, a point is added to $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ (or $\mathbf{w}^n$ and $\mathbf{w}^{n-1}$) using Eq. (29), and when $N^n < N^{n-1}$, a point is removed from the same vectors using Eq. (31). In order to prevent audible artifacts when increasing c^n (and thus decreasing $\mathcal{N}^n$) due to a violation of the rigid connection in (17), a method is proposed in Eq. (32) to ensure that the grid points at the inner boundaries have a similar displacement when one of them is removed.

4.3 Implementation

A MATLAB implementation of the proposed method and audio examples can be found online[1] and Algorithm 1 shows the order of calculation of this implementation. Especially important to take into account, is to only retrieve a change in c^n at time index n once before all other calculations. This is to ensure that $u_{l_u}^n$ and $w_{l_w}^n$ are calculated with the same α and β for all l_u and l_w.

while *application is running* **do**
  Retrieve new c^n
  Calc. h^n (Eq. (12) with equality)
  Calc. $\mathcal{N}^n$ and N^n (Eqs. (15) and (13))
  Calc. α (Eq. (24))
  if $N^n \neq N^{n-1}$ **then**
    Add or remove point (Eq. (29) or (31))
    Update M^n and M_w^n
  end
  Calc. virtual grid points (Eqs. (26))
  Calc. $u_{l_u}^{n+1}$ and $w_{l_w}^{n+1}$ (Eqs. (37) and (38))
  Calc. and apply displacement corr. (Eq. (34))
  Retrieve output
  Update states ($\boldsymbol{\mathcal{U}}^{n-1} = \boldsymbol{\mathcal{U}}^n, \boldsymbol{\mathcal{U}}^n = \boldsymbol{\mathcal{U}}^{n+1}$)
  Update N^{n-1} ($N^{n-1} = N^n$)
  Increment n
end

Algorithm 1: *Pseudocode showing the order of calculations.*

5 Analysis and Results

5.1 Modes

Writing system (32) in matrix form, one can perform a modal analysis while changing c^n to obtain the frequencies and damping coefficients for each mode over time. As a test case, the wave speed of a system running at $f_\mathrm{s} = 44100$ Hz is linearly varied from $c^0 = 2940$ ($\mathcal{N}^0 = 15$) to $c^{n_\mathrm{end}} = 2205$ ($\mathcal{N}^{n_\mathrm{end}} = 20$) where $n_\mathrm{end} = t_\mathrm{dur} f_\mathrm{s}$ is the simulation length in samples and $t_\mathrm{dur} = 10$ s. Grid points are added and removed as close to the right boundary as possible, i.e., $M^n = N^n - 1$ and $M_w^n = 1$ (similar behaviour can be observed if $M^n = 1$ and $M_w^n = N^n - 1$). The result of the analysis is shown in Figure 3a where

[1] https://github.com/SilvinWillemsen/DAFx21DynamicGridFiles/

higher damping (induced by the displacement correction) is indicated using thinner and bluer lines. Figure 3b shows the resulting spectrogram, with the displacement correction deactivated, of the system excited with $u_1^0 = 1$ and the output retrieved at u_1^n, and Figure 3c shows a system with the same excitation but the change in c^n inverted ($\mathcal{N}^n = 20 \rightarrow 15$) and displacement correction activated.

In the following, the lowest mode generated by the analysis is referred to as f_1^n and should ideally be equal to f_0^n calculated using Eq. (3). The first thing one can observe from Figure 3a is that the frequencies of the modes decrease as c^n decreases (as desired). The lower the mode, the more linear this decrease happens. Between $\mathcal{N}^n = 15$ and $\mathcal{N}^n = 16$, f_1^n maximally deviates by -0.15 cents. In this same interval f_{15}^n maximally deviates by -67 cents. This deviation gets less as $\mathcal{N}^n$ increases. Experiments with higher even-ordered Lagrange interpolators show that these frequency deviations become smaller, but not by a substantial amount. The quadratic interpolator has thus been chosen for being simpler and more flexible while not being substantially worse than higher order interpolators.

Another observation from Figure 3a is that there are always N^n modes present, corresponding to the number of moving points of the system. As can be seen in Figure 3b the highest mode is not excited. If the system is excited when $\mathcal{N}^n$ is not an integer, the highest mode will also be excited. Comparing the implementation of the system using this method with integer $\mathcal{N}^n$ (without changing c^n) to a normal implementation of the 1D wave equation (shown in Section 3) with (static) $N^n = \mathcal{N}^n$, identical outputs are observed, even though the latter has $N^n - 1$ moving points.

5.2 Displacement Correction

In the experiments, $\sigma_0 = 1$ in Eq. (34). The displacement correction has a low-pass-comb-filtering effect on the system, where the position and amount of damped regions directly relates to the position of where grid points are added and removed. The best behaviour, i.e., least affecting lower frequencies, is when grid points are added and removed as close to the boundary as possible, i.e., $M^n = N^n - 1$, and only has one damped region as shown in Figures 3a and 3c.

5.3 Limit on Rate of Change of c

The current implementation of the proposed method can only add or remove a maximum one point per sample using Eqs. (29) and (31). The rate of change of f_0^n according to (15) is thus limited by $|\mathcal{N}^n - \mathcal{N}^{n-1}| \leq 1$. Though this is the maximum limitation on speed, a much lower limitation needs to be placed to keep the system well-behaved. The usual stability and energy analyses

performed on FD schemes are not valid anymore in the time-varying case. Frozen coefficient analysis as in [10] could be applied here and hold for slowly varying coefficients, but is left for future work.

6 Discussion

To decide whether the proposed method is satisfactory, the results presented in the previous section are compared to the method requirements listed in Section 4.

It can be argued that the frequency deviations of f_1^n from f_0^n are sufficiently small to say that r1 is satisfied. As for r2, a fractional number of intervals $\mathcal{N}^n$ has been introduced and smooth transitions are indeed observed from Figure 3b, in the case when c^n is decreased and $\mathcal{N}^n$ is increased. When c^n is increased instead, the displacement correction prevents (visible) artifacts when grid points are removed as seen in Figure 3c. Despite this, the filtering effect that the displacement correction has on the system (mentioned in Section 5.2) is not ideal as it creates damped regions in the spectrum of the output sound. The least intrusive filtering happens when points are added and removed as close to the boundary as possible, i.e., when $M^n = 1$ or $M_w^n = 1$ where the damping only occurs in the higher end of the spectrum. Although artifacts do not show in Figure 3c, to confirm the absence of audible artifacts, formal listening tests have to be carried out. Furthermore, higher speeds of parameter variation might cause artifacts anyway. The value of σ_0 could therefore also be made dynamic and depending on the rate of change of c^n to have a higher effect when c^n is increased faster and vice versa. Either way, as this is still not ideal, another method for reducing artifacts that less affects the frequency content of the system should be devised, if possible. Furthermore, higher modes will be lost after decreasing $\mathcal{N}$ and will not return after increasing $\mathcal{N}$ again. They can, however, be activated again by re-exciting the system.

The modal analysis in Figure 3a shows that the method generates N^n rather than $N^n - 1$ modes as set by r3. However, the output does contain the correct number of modes as shown in Figure 3b due to the highest mode not being excited. This is a result of the rigid connection imposed on the inner boundaries, forcing them to have the same displacement and act as one point. The latter part of r3, however, is not satisfied. The modes deviate from integer multiples of f_0^n, moreso for higher modes. Other interpolation techniques could be investigated to improve the behaviour and decrease this deviation.

Finally, the method only adds a few extra calculations for the inner boundaries so r4 is also easily satisfied.

Although the results bring forward some drawbacks of the proposed method, such as modal frequency deviations, and filtering effects, most of

these affect the higher frequencies of the output. First of all, human frequency sensitivity becomes very limited above 3000 Hz [12] making high-frequency deviations much less important perceptually. Secondly, the physical systems one usually tries to model contain high-frequency losses, causing higher modes to usually not have very high amplitudes to begin with. Finally, $\mathcal{N}^n$ is usually much bigger in the systems one models, where frequency deviations happen to a much smaller degree.

As of now, some aspects of the proposed method still lack physical justification (such as the displacement correction), but are shown to yield the desired behaviour and fulfil the requirements to a satisfactory degree. Despite this, further work needs to be done to physically justify the choices made in this paper.

7 Conclusions and Perspectives

This paper presents a method to change grid configurations of finite-difference schemes to allow for dynamic parameter changes. The method allows the stability condition that these schemes rely on can be satisfied with equality at all times, minimising numerical dispersion and bandlimiting issues. Grid points are shown to be added and removed smoothly and do not cause artifacts when switching between grid configurations. Listening tests will need to be performed to carried out to confirm the lack of audible artifacts.

The proposed method might not provide an exact solution to the problem of time-varying systems, and not all choices are physically justified, but it does circumvent the need for upsampling and higher orders of computations necessary to approximate this solution with, for example, full-grid interpolation.

Although this method has only been applied to the 1D wave equation it could be applied to many other 1D systems. Other parameters, such as material density or stiffness could also be made dynamic, going beyond what is physically possible. An application of the method that could be investigated is that of non-linear systems, such as the Kirchhoff-Carrier string model [13] where the tension is modulated based on the state of the system.

Other future work includes creating an adaptive version of the displacement correction that changes its effect depending on the speed at which the grid is changed. Finally, stability and energy analyses will have to be performed to show the limits on changes in parameters and grid configurations.

8 Acknowledgments

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

[1] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitator-resonator interaction in physical models syntheses," in *Proceedings of the International Computer Music Conference*, 1989.

[2] S. Bilbao, *Numerical Sound Synthesis*. United Kingdom: John Wiley & Sons, 2009.

[3] S. Willemsen, N. Andersson, S. Serafin, and S. Bilbao, "Real-time control of large-scale modular physical models using the sensel morph," in *Proc. of the 16th Sound and Music Computing Conference*, 2019, pp. 275–280.

[4] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "A physical model of the trombone using dynamic grids for finite-difference schemes," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx)*, 2021.

[5] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[6] S. Mehes, M. van Walstijn, and P. Stapleton, "Towards a virtual-acoustic string instrument," in *Proceedings of the 13th Sound and Music Computing Conference (SMC)*, 2016.

[7] S. Willemsen, S. Serafin, and J. R. Jensen, "Virtual analog simulation and extensions of plate reverberation," in *Proc. of the 14th Sound and Music Computing Conference*, 2017, pp. 314–319.

[8] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

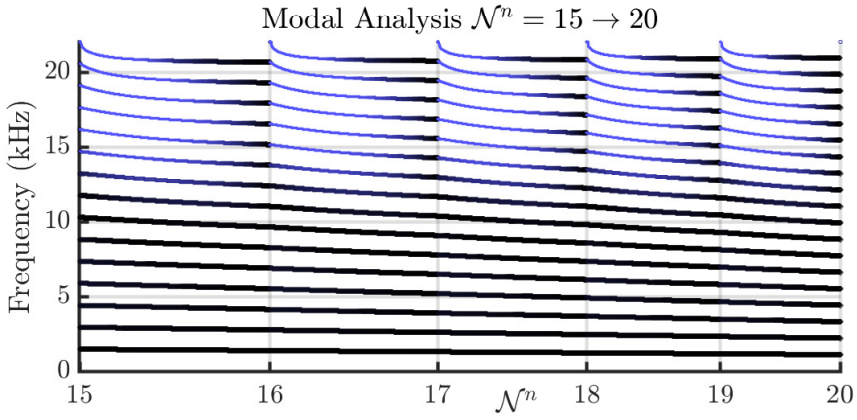
[9] R. Michon and J. Smith, "A hybrid guitar physical model controller: The BladeAxe," in *Proceedings ICMC | SMC | 2014*, 2014.

[10] J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, California: Wadsworth & Brooks/Cole Advanced Books & Software, 1989.

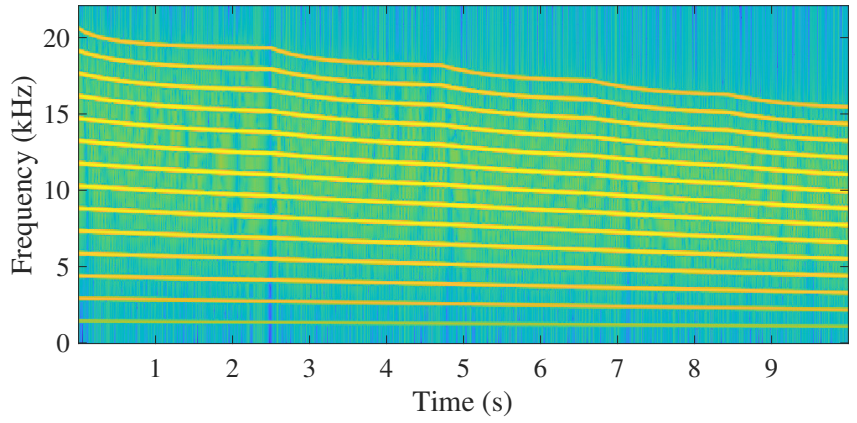
[11] S. Bilbao, "A modular percussion synthesis environment," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx)*, 2009.

[12] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. Berlin-Heidelberg, Germany: Springer-Verlag, 1990.

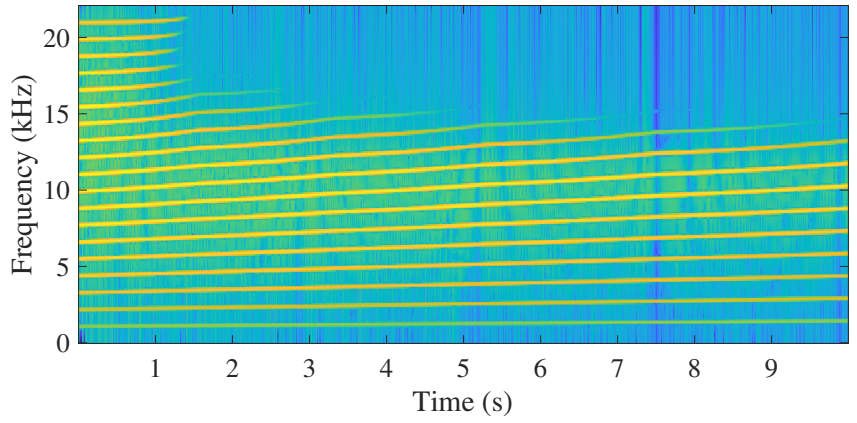
[13] G. F. Carrier, "On the non-linear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, pp. 157–165, 1945.



(a)



(b)



(c)

Fig. 3: *Experiments showing (linearly) varying wave speed between $c^0 = 2940$ ($\mathcal{N}^0 = 15$) and $c^{n_{\text{end}}} = 2205$ ($\mathcal{N}^{n_{\text{end}}} = 20$) with $M^n = N^n - 1$ and $M_w^n = 1$ running at $f_s = 44100$ Hz for 10 s ($n_{\text{end}} = 10 f_s$). (a) Modal analysis of system* (32)*. Thinner and bluer lines indicate a higher amount of damping. (b) Output of the system while decreasing c^n ($\mathcal{N}^n = 15 \rightarrow 20$) without displacement correction, excited using $u_1^0 = 1$ and retrieved at u_1^n. The sound output follows the same pattern as predicted by the analysis shown in Figure 3a. (c) Output of the system while increasing c^n ($\mathcal{N}^n = 20 \rightarrow 15$) with displacement correction activated (essentially flipping the analysis in Figure 3a along the x-axis and applying this to a 10 s simulation).*

Paper H

A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21), 2021.

Abstract

In this paper, a complete simulation of a trombone using finite-difference time-domain (FDTD) methods is proposed. In particular, we propose the use of a novel method to dynamically vary the number of grid points associated to the FDTD method, to simulate the fact that the physical dimension of the trombone's resonator dynamically varies over time. We describe the different elements of the model and present the results of a real-time simulation.

1 Introduction

The trombone is a musical instrument that presents distinct challenges from the perspective of physical modelling synthesis. In particular, the excitation mechanism between the lips and the player has been extensively studied, and simulated mostly using a simple mass-spring damper system [1]. Because the majority of the bore is cylindrical, nonlinear effects can appear at high blowing pressures [2], leading to changes in timbre, or brassiness; such effects have been investigated and simulated [1, 3, 4]. However, the defining characteristic of the trombone is that the physical dimensions of the resonator vary during playing. Synthesis techniques such as digital waveguides allow an approach to dynamic resonator changes in a simple and computationally efficient way, simply by varying the length of the corresponding delay line. This feature has been used in real-time sound synthesis [5], for simplified bore profiles suitable for modelling in terms of travelling waves.

However, when attempting more fine-grained modelling of the trombone resonator using finite-difference time-domain (FDTD) methods, the issue of the change in the tube length is not trivial. Previous implementations of brass instruments using these methods focus on the trumpet [6] and various brass instruments (including the trombone bore) under static conditions [7]. To our knowledge, the simulation of a trombone varying the shape of the resonator in real time using FDTD methods has not been approached. We can tackle this problem by having a grid that dynamically changes while the simulation is running as presented in a companion paper [8]. Briefly described, we modify the grid configurations of the FDTD method by adding and removing grid points based on parameters describing the system.

In this paper, we propose a full simulation of a trombone, describing all its elements in detail with a specific focus on the dynamic grid simulation. Section 2 presents the models for the tube and lip reed interaction in continuous time. Section 3 briefly introduces FDTD methods and the discretisation of the aforementioned continuous equations. Section 4 presents the dynamic grid used to simulate the trombone slide and details on the implementation are provided in Section 5. Section 6 presents simulation results, and some

concluding remarks appear in Section 7.

2 Continuous System

Wave propagation in an acoustic tube can be approximated using a 1-dimensional (1D) model, for wavelengths that are long relative to the largest lateral dimension of the tube. Consider a tube of time-varying length $L = L(t)$ (in m) defined over spatial domain $x \in [0, L]$ and time $t \geq 0$. Using operators ∂_t and ∂_x denoting partial derivatives with respect to time t and spatial coordinate x, respectively, a system of first-order partial differential equations (PDEs) describing the wave propagation in an acoustic tube can then be written as:

$$\frac{S}{\rho_0 c^2}\partial_t p = -\partial_x(Sv), \tag{1a}$$

$$\rho_0 \partial_t v = -\partial_x p, \tag{1b}$$

with acoustic pressure $p = p(x,t)$ (in N/m^2), particle velocity $v = v(x,t)$ (in m/s) and (circular) cross-sectional area $S(x)$ (in m^2). Furthermore, ρ_0 is the density of air (in kg/m^3) and c is the speed of sound in air (in m/s). System (1) can be condensed into a second-order equation in p alone, often referred to as Webster's equation [9]. For simplicity, effects of viscothermal losses have been neglected in (1). For a full time domain model of such effects in an acoustic tube, see, e.g. [10].

System (1) requires two boundary conditions, one at either end of the domain. The left boundary condition, at $x = 0$, will be set according to an excitation model to be described in Section 2.1. The right boundary, at $x = L$, is set according to a radiation condition. The radiation model used here, is the one for the unflanged cylindrical pipe proposed by Levine and Schwinger in [11] and discretised by Silva *et al.* in [12]. As this model is not important for the contribution of this work it will not be detailed here in full. The interested reader is instead referred to [7, 13] for a comprehensive explanation.

2.1 Coupling to a Lip Reed

To excite the system, a lip reed can be modelled as a mass-spring-damper system including two nonlinearities due to flow, and the collision of the lip against the mouthpiece. In the following, y can be seen as the moving upper lip where the lower lip is left static and rigid. A diagram of the full lip-reed model is shown in Figure 1. Using dots to indicate time-derivatives, the lip reed is modelled as

$$M_\mathrm{r}\ddot{y} = -M_\mathrm{r}\omega_\mathrm{r}^2 y - M_\mathrm{r}\sigma_\mathrm{r}\dot{y} + \psi(\dot{\psi}/\dot{\eta}) + S_\mathrm{r}\Delta p, \tag{2}$$

with displacement from the equilibrium $y = y(t)$, lip mass M_r (in kg), externally supplied (angular) frequency of oscillation $\omega_\mathrm{r} = \omega_\mathrm{r}(t) = \sqrt{K_\mathrm{r}/M_\mathrm{r}}$ (in rad/s) and stiffness $K_\mathrm{r} = K_\mathrm{r}(t)$ (in N/m).

We extend the existing models of lip reeds [1] by introducing a nonlinear collision between the lips based on potential quadratisation proposed by [14]. The collision potential is defined as

$$\psi(\eta) = \left(\frac{2K_\mathrm{c}}{\alpha_\mathrm{c}+1} [\eta]_+^{\alpha_\mathrm{c}+1} \right)^{1/2}, \tag{3}$$

with collision stiffness $K_\mathrm{c} > 0$ and dimensionless nonlinear collision coefficient $\alpha_\mathrm{c} \geq 1$, The inverted distance between the lips $\eta = \eta(t) \triangleq -y - H_0$ (in m), for static equilibrium separation H_0 (in m). $[\eta]_+ = 0.5(\eta + |\eta|)$ indicates the "positive part of η". Notice, that if $\eta \geq 0$, the lips are closed and the collision potential will be non-zero. This quadratic form of a collision potential allows for a non-iterative implementation [14]. This will be explained further in Section 3.

Finally, S_r (in m^2) is the effective surface area and

$$\Delta p = P_\mathrm{m} - p(0,t) \tag{4}$$

is the difference between the externally supplied pressure in the mouth $P_\mathrm{m} = P_\mathrm{m}(t)$ and the pressure in the mouthpiece $p(0,t)$ (all in Pa). This pressure difference causes a volume flow velocity following the Bernoulli equation

$$U_\mathrm{B} = w_\mathrm{r}[-\eta]_+ \mathrm{sgn}(\Delta p) \sqrt{\frac{2|\Delta p|}{\rho_0}}, \tag{5}$$

(in m^3/s) with effective lip-reed width w_r (m). Another volume flow is generated by the lip reed itself according to

$$U_\mathrm{r} = S_\mathrm{r} \frac{dy}{dt} \tag{6}$$

(in m^3/s). Assuming that the volume flow velocity is conserved, the total air volume entering the system is defined as

$$S(0)v(0,t) = U_\mathrm{B}(t) + U_\mathrm{r}(t). \tag{7}$$

This condition serves as a boundary condition at $x = 0$ for system (1).

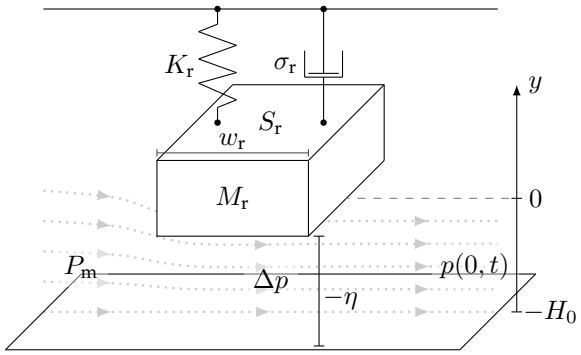


Fig. 1: *Diagram of the lip-reed system with the equilibrium at 0 and the distance from the lower lip H_0. The various symbols relate to those used in Eq.* (2)*.*

3 Discretisation

The continuous system described in the previous section is discretised using FDTD methods, through an approximation over a grid in space and time. Before presenting this discretisation, we briefly summarize the operation of FDTD methods.

3.1 Numerical Methods

Consider a 1D system of (static) length L described by state variable $u = u(x,t)$ with spatial domain $x \in [0, L]$ and time $t \geq 0$. The spatial domain can be disctretised according to $x = lh$ with spatial index $l \in \{0, \ldots, N\}$, number of intervals between the grid points N, grid spacing h (in m) and time as $t = nk$ with temporal index $n \in \mathbb{Z}^{0+}$ and time step k (in s). The grid function u_l^n represents an approximation to $u(x,t)$ at $x = lh$ and $t = nk$.

Shift operators can then be applied to grid function u_l^n. Temporal and spatial shift operators are

$$\begin{aligned} e_{t+}u_l^n &= u_l^{n+1}, \quad e_{t-}u_l^n = u_l^{n-1}, \\ e_{x+}u_l^n &= u_{l+1}^n, \quad e_{x-}u_l^n = u_{l-1}^n, \end{aligned} \tag{8}$$

from which more complex operators can be derived. First-order derivatives can be approximated using forward, backward and centred difference operators in time

$$\delta_{t+} = \frac{e_{t+} - 1}{k}, \; \delta_{t-} = \frac{1 - e_{t-}}{k}, \; \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k}, \tag{9}$$

(all approximating ∂_t) and space

$$\delta_{x+} = \frac{e_{x+} - 1}{h},\ \delta_{x-} = \frac{1 - e_{x-}}{h},\ \delta_{x\cdot} = \frac{e_{x+} - e_{x-}}{2h}, \tag{10}$$

(all approximating ∂_x) where 1 is the identity operator.

Furthermore, forward, backward and centred averaging operators can be defined in time

$$\mu_{t+} = \frac{e_{t+} + 1}{2},\ \mu_{t-} = \frac{1 + e_{t-}}{2},\ \mu_{t\cdot} = \frac{e_{t+} + e_{t-}}{2}, \tag{11}$$

and space

$$\mu_{x+} = \frac{e_{x+} + 1}{2},\ \mu_{x-} = \frac{1 + e_{x-}}{2},\ \mu_{x\cdot} = \frac{e_{x+} + e_{x-}}{2}. \tag{12}$$

Finally, an approximation δ_{tt} to a second time derivative may be defined as

$$\delta_{tt} = \delta_{t+}\delta_{t-} = \frac{1}{k^2}\left(e_{t+} - 2 + e_{t-}\right). \tag{13}$$

3.2 Discrete Tube

As a first step, the domain $x \in [0, L]$ can be subdivided into N equal segments of length h (the grid spacing). Interleaved grid functions approximating p and v may then be defined. Grid function p_l^n with $l \in \{0, \ldots, N\}$ approximates $p(x,t)$ at coordinates $x = lh$, $t = nk$ and $v_{l+1/2}^{n+1/2}$ with $l \in \{0 \ldots, N-1\}$ approximates $v(x,t)$ at coordinates $x = (l+1/2)h$, $t = (n+1/2)k$. In addition, a discrete cross-sectional area $S_l \approx S(x = lh)$ with $l \in \{0, \ldots, N\}$ is assumed known. System (1) can then be discretised as

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p_l^n = -\delta_{x-}(S_{l+1/2}v_{l+1/2}^{n+1/2}), \tag{14a}$$

$$\rho_0\delta_{t-}v_{l+1/2}^{n+1/2} = -\delta_{x+}p_l^n, \tag{14b}$$

where $S_{l+1/2} = \mu_{x+}S_l$ and $\bar{S}_l = \mu_{x-}S_{l+1/2}$ are approximations to the continuous cross-sectional area $S(x)$. The values for $\bar{S}_l$ at the boundaries, i.e., $\bar{S}_0$ and $\bar{S}_N$, are set equal to $S(0)$ and $S(L)$.

Expanding the operators, we obtain the following recursion

$$p_l^{n+1} = p_l^n - \frac{\rho_0 c\lambda}{\bar{S}_l}(S_{l+1/2}v_{l+1/2}^{n+1/2} - S_{l-1/2}v_{l-1/2}^{n+1/2}), \tag{15a}$$

$$v_{l+1/2}^{n+1/2} = v_{l+1/2}^{n-1/2} - \frac{\lambda}{\rho_0 c}(p_{l+1}^n - p_l^n), \tag{15b}$$

where $\lambda = ck/h$ is referred to as the Courant number and

$$\lambda \leq 1 \quad \Longrightarrow \quad h \geq ck \tag{16}$$

in order for the scheme to be stable [15]. In implementation, the following steps are taken to calculate λ:

$$h := ck, \;\; N := \left\lfloor \frac{L}{h} \right\rfloor, \;\; h := \frac{L}{N}, \;\; \lambda := \frac{ck}{h}, \tag{17}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation and is necessary because N is an integer. This causes (16) to not be satisfied with equality for all choices of L.

Equations (15a) and (15b) hold for $l \in \{0, \ldots, N\}$ and $l \in \{0, \ldots, N-1\}$ respectively, and thus, in analogy with the continuous case, two numerical boundary conditions are required in order to update p_0^{n+1} and p_N^{n+1}. These are provided by numerical equivalents of the excitation condition (see Section 3.3 below) and the radiation condition (in [13]).

3.3 Lip reed

As the lip reed interacts with the particle velocity of the tube via Eq. (7), it is discretised to the interleaved temporal grid, but to the regular spatial grid as it interacts with the boundary at $x = 0$. Equations (2) - (7) are then discretised as follows:

$$\begin{aligned} M_\mathrm{r}\delta_{tt} y^{n+1/2} = &- M_\mathrm{r}(\omega_\mathrm{r}^{n+1/2})^2 \mu_{t\cdot} y^{n+1/2} \\ &- M_\mathrm{r}\sigma_\mathrm{r}\delta_{t\cdot} y^{n+1/2} + (\mu_{t+}\psi^n)\, g^{n+1/2} + S_\mathrm{r}\Delta p^{n+1/2}, \end{aligned} \tag{18a}$$

$$\Delta p^{n+1/2} = P_\mathrm{m}^{n+1/2} - \mu_{t+} p_0^n, \tag{18b}$$

$$U_\mathrm{B}^{n+1/2} = w_\mathrm{r}[-\eta^{n+1/2}]_+ \mathrm{sgn}(\Delta p^{n+1/2}) \cdot \sqrt{2|\Delta p^{n+1/2}|/\rho_0}, \tag{18c}$$

$$U_\mathrm{r}^{n+1/2} = S_\mathrm{r}\delta_{t\cdot} y^{n+1/2}, \tag{18d}$$

$$\mu_{x-}(S_{1/2} v_{1/2}^{n+1/2}) = U_\mathrm{B}^{n+1/2} + U_\mathrm{r}^{n+1/2}. \tag{18e}$$

Here, following [14],

$$g^{n+1/2} = \begin{cases} \kappa\sqrt{\dfrac{K_\mathrm{c}(\alpha_\mathrm{c}+1)}{2}} \cdot (\eta^{n+1/2})^{\frac{\alpha_\mathrm{c}-1}{2}} & \text{if } \eta^{n+1/2} \geq 0 \qquad (19a) \\ -2\dfrac{\psi^n}{\eta^\star - \eta^{n-1/2}} & \text{if } \eta^{n+1/2} < 0 \qquad (19b) \\ 0, & \text{if } \eta^{n+1/2} < 0 \text{ and } \eta^\star = \eta^{n-1/2} \qquad (19c) \end{cases}$$

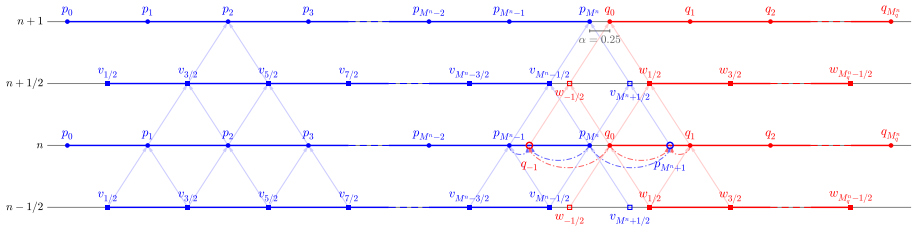


Fig. 2: *Schematic showing data flow of how different grid points at time index $n+1$ are calculated with $\alpha = 0.25$ in Eq. (25). To prevent cluttering, arrows going straight up (indicating that the state of a grid point at time step n is needed to calculate the state of that grid point at $n+1$) are suppressed. As an example of the usual case (refer to Eq. (15)), the points required to calculate p_2^{n+1} are shown. Furthermore, the points needed to calculate $p_{M^n}^{n+1}$ and q_0^{n+1} are shown. The most important difference with the usual case is that the virtual grid points $p_{M^n+1}^n$ and q_{-1}^n are the result of the interpolation of known pressure values at n using Eq. (27).*

where $\kappa = 1$ if $\psi^n \geq 0$, otherwise $\kappa = -1$. It should be noted that condition (19c) has been added to the definition of g from [14] to prevent a division by 0 in (19b). Finally, $\eta^\star = -y^\star - H_0$ where $y^\star$ is the value of $y^{n+3/2}$ calculated using system (18) (after expansion) without the collision potential. This means that system (18) needs to be calculated twice every iteration, once without the collision term and once with. The process of calculating the pressure difference $\Delta p^{n+1/2}$ in (18) will not be given here, but the interested reader is referred to [13, Ch. 5] for a derivation.

4 Dynamic grid

The defining feature of the trombone is its slide that alters the length of the tube, changing the resonant frequencies. In a companion paper [8], we present a method to dynamically change grid configurations of FD schemes by inserting and deleting grid points based on an instantaneous value of the time-varying wave speed $c(t)$. Although here, the tube length $L(t)$ is varied, the method still applies. Note that this method only works for slow (sub-audio rate) parameter changes.

We can split a tube with time-varying length L^n into two smaller sections with lengths L_p^n and L_q^n (in m) such that $L^n = L_p^n + L_q^n$. Splitting the schemes in (14) in this way yields two sets of first-order systems. The pressure and particle velocity of the first (left) system $p_{l_p}^n$ and $v_{l_p+1/2}^{n+1/2}$ are both defined over discrete domain $l_p \in \{0, \ldots, M^n\}$, and those of the second (right) system $q_{l_q}^n$ and $w_{l_q-1/2}^{n+1/2}$ are defined over discrete domain $l_q \in \{0, \ldots, M_q^n\}$, with

$$M^n = \lceil L_p^n / h \rceil, \quad \text{and} \quad M_q^n = \lfloor L_q^n / h \rfloor \tag{20}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation. Note, that the domains for v and w have an extra grid point when compared to the regular case in (14) and that w is indexed with $l_q - 1/2$ rather than $l_q + 1/2$. The resulting system of FD schemes then becomes

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+} p^n_{l_p} = -\delta_{x-}(S_{l+1/2} v^{n+1/2}_{l_p+1/2}), \tag{21a}$$

$$\rho_0 \delta_{t-} v^{n+1/2}_{l_p+1/2} = -\delta_{x+} p^n_{l_p}, \tag{21b}$$

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+} q^n_{l_q} = -\delta_{x+}(S_{l-1/2} w^{n+1/2}_{l_q-1/2}), \tag{21c}$$

$$\rho_0 \delta_{t-} w^{n+1/2}_{l_q-1/2} = -\delta_{x-} q^n_{l_q}. \tag{21d}$$

Here, due to the different indexing for w, the spatial derivatives for the right system are flipped (δ_{x+} became δ_{x-} and vice versa). Also note, that l is still used for the spatial indices of $\bar{S}$ and S which now approximate $S(x)$ according to

$$S_l \approx \begin{cases} S(x = lh) & \text{for } x \in [0, L^n_p], \\ S(x = L^n - (M^n_q - l)h) & \text{for } x \in [L^n_p, L^n]. \end{cases} \tag{22}$$

The conditions for the outer boundaries of this system, i.e., at $l_p = 0$ and $l_q = M^n_q$, are the same as for the full system. The inner boundaries, $l_p = M^n$ and $l_q = 0$ are connected according to the method described in [8] to be explained shortly. To be able to calculate $p^{n+1}_{M^n}$ and q^{n+1}_0, the domains of v and w have been extended at the inner boundaries to include $v^{n+1/2}_{M^n+1/2}$ and $w^{n+1/2}_{-1/2}$. These, however, require points outside of the domains of $p^n_{l_p}$ and $q^n_{l_q}$, i.e., $p^n_{M^n+1}$ and q^n_{-1}. In [8] we propose to calculate these *virtual grid points* based on known values of the system. Despite the fact that [8] presents the method using a second-order system, it can still be applied here. The process of how $p^{n+1}_{M^n}$ and q^{n+1}_0 are calculated is visualised in in Figure 2. Notice that all time steps use the same value of M^n and M^n_q. In other words, the expansion of the temporal operators in (9) do not affect the temporal indices n in M^n and M^n_q.

4.1 Changing the Tube Length

In the following, the location of a grid point u_l along the grid (in m from the left boundary) at time index n is denoted as $x^n_{u_l}$.

The two pairs of first order systems in (21) are placed on the same domain x with

$$x^n_{p_{l_p}} = l_p h, \quad \text{and} \quad x^n_{q_{l_q}} = L^n - (M^n_q - l_q)h, \tag{23}$$

describing the locations of the left system and right system respectively. Here, it can be observed that as the tube length L^n changes, the locations of the grid points of the right system will change. More specifically, as the trombone-slide

is extended and L^n increases, all grid points of the right system move to the right, and to the left for a contracting slide. If L^n is changed in a smooth fashion, the continuous domain $x \in [0, L^n]$ will not necessarily be subdivided into an integer amount of intervals N^n (of size $h = ck$). This is where a *fractional* number of intervals is introduced and is defined as

$$\mathcal{N}^n = L^n/h, \tag{24}$$

which is essentially the calculation of N in Eq. (17) without the flooring operation, and $N^n = \lfloor \mathcal{N}^n \rfloor$. The fractional part of $\mathcal{N}^n$ can then be calculated using

$$\alpha = \alpha^n = \mathcal{N}^n - N^n, \tag{25}$$

which describes the distance between the inner boundaries along the grid in terms of how many times h would fit in-between (which is always less than once). If $\mathcal{N}^n = N^n$ and $\alpha = 0$, the inner boundary locations perfectly overlap, and $x^n_{p_{M^n}} = x^n_{q_0}$. This also means that the domain x can be exactly divided into N^n equal intervals of size $h = ck$. As the virtual grid points $p^n_{M^n+1}$ and q^n_{-1} perfectly overlap with q^n_1 and $p^n_{M^n-1}$ respectively, these values can be used directly to calculate the grid points at the inner boundaries. This situation effectively acts as a rigid connection between the grid points at the inner boundaries defined as

$$p^n_{M^n} = q^n_0, \quad \text{if } \alpha = 0. \tag{26}$$

If $\alpha \neq 0$, some other definition for $p^n_{M^n+1}$ and q^n_{-1} needs to be found. We use quadratic Lagrangian interpolation according to

$$p^n_{M^n+1} = \frac{\alpha - 1}{\alpha + 1} p^n_{M^n} + q^n_0 - \frac{\alpha - 1}{\alpha + 1} q^n_1, \tag{27a}$$

$$q^n_{-1} = -\frac{\alpha - 1}{\alpha + 1} p^n_{M^n-1} + p^n_{M^n} + \frac{\alpha - 1}{\alpha + 1} q^n_0, \tag{27b}$$

which can then be used to calculate $v^{n+1/2}_{M^n+1/2}$ and $w^{n+1/2}_{-1/2}$ and consequently $p^{n+1}_{M^n}$ and q^{n+1}_0 (see Figure 2). This process is repeated every sample. It can be shown through the rigid connection in (26), that if $\alpha = 0$, the definitions in (27) reduce to $p^n_{M^n+1} = q^n_1$ and $q^n_{-1} = p^n_{M^n-1}$ as stated before.

4.2 Adding and removing grid points

As the tube length L^n changes, L^n_p and L^n_q also change according to

$$L^n_p = L^{n-1}_p + 0.5 L^n_{\text{diff}}, \quad L^n_q = L^{n-1}_q + 0.5 L^n_{\text{diff}}, \tag{28}$$

where

$$L^n_{\text{diff}} = L^n - L^{n-1}, \tag{29}$$

which causes the number of intervals between grid points M^n and M^n_q to change as well, according to Eq. (20).

The following state vectors are introduced for the pressure, defined for $n+1$ and n

$$\mathbf{p}^n = [p^n_0, p^n_1, ..., p^n_{M^n}]^T,\ \mathbf{q}^n = [q^n_0, q^n_1, ..., q^n_{M^n_q}]^T, \tag{30}$$

and for the velocity, defined for $n + 1/2$ and $n - 1/2$

$$\begin{aligned} \mathbf{v}^{n-1/2} &= [v^{n-1/2}_{1/2}, v^{n-1/2}_{3/2}, ..., v^{n-1/2}_{M^n+1/2}]^T, \\ \mathbf{w}^{n-1/2} &= [w^{n-1/2}_{-1/2}, w^{n-1/2}_{1/2}, ..., w^{n-1/2}_{M^n_q-1/2}]^T, \end{aligned} \tag{31}$$

and contain the different states over the discrete domains defined at the beginning of this section. Here, T denotes the transpose operation.

If $N^n > N^{n-1}$, points are added to the left and right system in an alternating fashion:

$$\begin{cases} \mathbf{p}^n = [(\mathbf{p}^n)^T, I_3\mathbf{r}^n]^T \\ \mathbf{v}^{n-1/2} = [(\mathbf{v}^{n-1/2})^T, I_3\mathbf{z}^{n-1/2}_v]^T \end{cases} \quad \text{if } N^n \text{ is odd,}$$
$$\begin{cases} \mathbf{q}^n = [I^{\leftarrow}_3\mathbf{r}^n, (\mathbf{q}^n)^T]^T \\ \mathbf{w}^{n-1/2} = [I^{\leftarrow}_3\mathbf{z}^{n-1/2}_w, (\mathbf{w}^{n-1/2})^T]^T \end{cases} \quad \text{if } N^n \text{ is even,} \tag{32}$$

where

$$\begin{aligned} \mathbf{r}^n &= [p^n_{M^n-1}, p^n_{M^n}, q^n_0, q^n_1]^T, \\ \mathbf{z}^{n-1/2}_v &= [v^{n-1/2}_{M^n-1/2}, v^{n-1/2}_{M^n+1/2}, w^{n-1/2}_{1/2}, w^{n-1/2}_{3/2}]^T - \boldsymbol{\eta}, \\ \mathbf{z}^{n-1/2}_w &= [v^{n-1/2}_{M^n-3/2}, v^{n-1/2}_{M^n-1/2}, w^{n-1/2}_{-1/2}, w^{n-1/2}_{1/2}]^T + \boldsymbol{\eta}^{\leftarrow}, \end{aligned} \tag{33}$$

and cubic Lagrangian interpolator

$$I_3 = \left[-\tfrac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} \quad \tfrac{2\alpha}{\alpha+2} \quad \tfrac{2}{\alpha+2} \quad -\tfrac{2\alpha}{(\alpha+3)(\alpha+2)}\right]. \tag{34}$$

Here,

$$\boldsymbol{\eta} = \boldsymbol{\eta}^{n-1/2} = \left(w^{n-1/2}_{-1/2} - v^{n-1/2}_{M^n+1/2}\right) \cdot [0, 0, 1, 1]^T \tag{35}$$

adds an offset to half of the elements in the $\mathbf{z}$ vectors depending on the difference between $v^{n-1/2}_{M^n+1/2}$ and $w^{n-1/2}_{-1/2}$. Why this is necessary will be further explained in Section 4.3. Finally, $I^{\leftarrow}_3$ and $\boldsymbol{\eta}^{\leftarrow}$ are flipped versions of (34) and (35) respectively.

If $N^n < N^{n-1}$, points are simply removed from the vectors according to

$$\begin{cases} \mathbf{p}^n = [p_0^n, \ldots, p_{M^n-1}^n]^T \\ \mathbf{v}^{n-1/2} = [v_{1/2}^{n-1/2}, \ldots, v_{M^n-1/2}^{n-1/2}]^T \end{cases} \quad \text{if } N^n \text{ is even,}$$
$$\begin{cases} \mathbf{q}^n = [q_1^n, \ldots, q_{M_q^n}^n]^T \\ \mathbf{w}^{n-1/2} = [w_{1/2}^{n-1/2}, \ldots, w_{M_q^n-1/2}^{n-1/2}]^T \end{cases} \quad \text{if } N^n \text{ is odd.} \tag{36}$$

Notice that the even and odd conditions in Eqs. (32) and (36) can be swapped. To stay as close to the desired location of adding and removing grid points as possible, this requires the ceiling and flooring operations in (20) to be swapped as well.

4.3 Drift of w

The inner boundaries of the pressure states p and q are connected by (27), but no such connection exists for the velocity states v and w. As the radiating boundary is implemented on the pressure grid, this leaves w without any boundary condition; it is only "held in place" by the pressure values of q, or more specifically, by derivatives (both spatial and temporal). As FD schemes are an approximation, it does not give a perfect solution and w tends to 'drift' during the simulation, especially when L^n is changed.

Luckily, as the pressure values are also calculated from derivatives of the velocity, the absolute state of w does not matter. The difference in values at the connection point is also irrelevant as there is no spatial derivative taken between v and w (refer to Figure 2). Finally, the pressure values are used for the output audio of the simulation, so the drift does not affect the audio.

The absolute states of the velocity vectors do, however, need to be accounted for when adding points to the $\mathbf{v}$ and $\mathbf{w}$ using (32). The current drift can be approximated by observing the difference between $w_{-1/2}^{n-1/2}$ and $v_{M^n+1/2}^{n-1/2}$, as these have approximately the same x location ($x_{w_{-1/2}}^n \approx x_{v_{M^n+1/2}}^n$) when a grid point is added. This is then used in a drift-correction vector $\boldsymbol{\eta}^{n-1/2}$ presented in (35). When a point is added to v, the values of w in $\mathbf{z}_v$ are offset by the aforementioned difference and when a point is added to w the same happens (inverted) for the values of v in $\mathbf{z}_w$. This way, the drift is allowed, but does not affect the state of the newly added grid points. Notice that the drift does not affect the operations of point removal in (36).

4.4 State Correction

As L^n, and consequently the number of grid points, is decreased, it might occur that the grid points at the inner boundaries $p_{M^n}^n$ and q_0^n have a very

different value when $\alpha \gtrsim 0$, i.e., right before a point is removed. This violates the rigid connection in Eq. (26).

We propose in [8] to add an artificial spring-like connection between the grid points at the inner boundaries that "corrects" the state of these points. Applying this to system (21) extends Eqs. (21a) and (21c) according to

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+} p_{l_p}^n = -\delta_{x-}(S_{l+1/2} v_{l_p+1/2}^{n+1/2}) + J_p(x_{p_{M^n}}^n) F_{\text{sc}}^n, \tag{37a}$$

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+} q_{l_q}^n = -\delta_{x+}(S_{l-1/2} w_{l_q-1/2}^{n+1/2}) - J_q(x_{q_0}^n) F_{\text{sc}}^n, \tag{37b}$$

where the spreading operators are defined as

$$\begin{aligned} J_p(x_i^n) &= \begin{cases} \frac{1}{h}, & l_p = \lfloor x_i^n/h \rfloor \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \\ J_q(x_i^n) &= \begin{cases} \frac{1}{h}, & l_q = \lfloor x_i^n/h \rfloor - M^n \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{38}$$

Furthermore, the correction effect is defined as

$$F_{\text{sc}}^n = \beta\left(\mu_t . \eta_{\text{sc}}^n + \sigma_{\text{sc}} \delta_t . \eta_{\text{sc}}^n\right), \tag{39}$$

with spring damping σ_{sc}, pressure difference

$$\eta_{\text{sc}}^n \triangleq q_0^n - p_{M^n}^n, \tag{40}$$

and scaling coefficient

$$\beta = \beta(\alpha) = \frac{1-\alpha}{\alpha+\varepsilon}. \tag{41}$$

Here, $\varepsilon \ll 1$ to prevent division by 0. Just like in [8], the implementation of the correction effect allows for an infinite β when $\alpha = \varepsilon = 0$ acting like a rigid connection between Eqs. (37a) and (37b).

5 Implementation

The implementation has been done in C++ using the JUCE framework [1], and is available online[2] as well as a demo showcasing it.[3] The audio output of the system can be retrieved by selecting a grid point on the pressure grid and listening to this at the given sample rate f_{s}. Here, the radiating boundary $q_{M_q^n}^n$ is chosen, as this is where the sound enters the listening space in the real

[1] https://juce.com/
[2] https://github.com/SilvinWillemsen/cppBrass/releases/
[3] https://youtu.be/Ht5gVNrshYo

Table 1: *Geometry of a measured trombone taken from [16]. Numbers correspond to Figure 3.*

Part of tube	Length (cm)	Radius (cm)
Inner slide (1)	70.8	0.69
Outer slide (extended) (2)	53	0.72
Slide crook (3)	17.7	0.74
Outer slide (extended) (4)	53	0.72
Inner slide (5)	71.1	0.69
Gooseneck (6)	24.1	0.71
Tuning slide (7)	25.4	0.75, 1.07
Bell flare (8)	50.2	1, 10.8

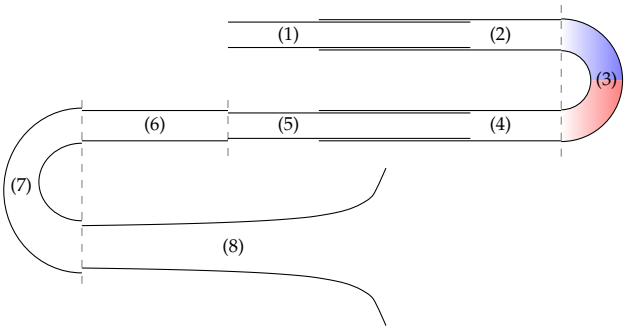


Fig. 3: *Diagram showing the trombone geometry (not to scale). Numbers correspond to the parts of the tube found in Table 1 and dashed lines highlight where the different parts are separated. The tube is split in the middle of the slide crook with the colours corresponding to those in Figure 2.*

world. To mimic low-pass filtering happening due to a distributed radiating area, a 4$^{\text{th}}$-order low-passing Butterworth filter with a cutoff frequency of $f_{\text{c}} = \sqrt{c^2\pi/S(L)} \approx 3245$ Hz is used. This equation is retrieved by choosing the listening point to be at the bell surface and integrating over the bell area.

5.1 Parameters

For the most part, the parameters used in the simulation have been obtained from [13, 16, 17]. The lengths and radii of different parts of the tube can be found in Table 1 and a diagram showing this geometry is shown in Figure 3. The system is split in the middle of the slide crook such that the ranges for the lengths of the two tubes are $L_p^n \in [0.797, 1.327]$ and $L_q^n \in [1.796, 2.326]$.

Other parameters used in the simulation can be found in Table 2. Not included here is λ, which has been set slightly lower than the stability condition in (16), i.e., $\lambda = 0.999$. Although the implementation works when $\lambda = 1$, this is done to tolerate (much) higher speeds of change in L^n before instability occurs (see Section 5.2). Not satisfying condition (16) causes bandlimiting and

Table 2: *List of parameter values used for the simulation. Taken from *[16], *[13] or **[17] with temperature* $T = 26.85°C$.

Name	Symbol (unit)	Value
Tube		
Length	L (m)	$2.593 \leq L \leq 3.653$*
Air density	ρ_0 (kg/m^3)	1.1769**
Wave speed	c (m/s)	347.23**
Geometry	S (m^2)	See Table 1.
Lip reed		
Mass	M_r (kg)	$5.37 \cdot 10^{-5}$*
Frequency	ω_r (rad/s)	$20 \leq \omega_\mathrm{r}/2\pi \leq 1000$
Mouth pressure	P_m (Pa)	$0 \leq P_\mathrm{m} \leq 6000$
Damping	σ_r (s^{-1})	5*
Eff. surface area	S_r (m^2)	$1.46 \cdot 10^{-5}$*
Width	w_r (m)	0.01*
Equilibrium sep.	H_0 (m)	$2.9 \cdot 10^{-4}$*
Coll. stiffness	K_c (N/m)	10^4
Nonlin. coll. coeff.	α_c (-)	3
Other		
State corr. damping	σ_sc	1
Sample rate	f_s (Hz)	44100

dispersive effects [15], but such a small deviation from the condition has no perceptual influence on the output sound and outweighs the problems caused by instability.

As the tube acts mainly as an amplifier for specific resonant frequencies it is important to match the frequency of the lip reed to a resonating mode of the tube. This frequency depends on L^n in the following way

$$\omega_\mathrm{r}^{n+1/2} = \mathcal{F}\frac{2\pi c}{\rho_0 L^{n+1/2}}\ , \tag{42}$$

where $L^{n+1/2} = L^n$ and scalar multiplier $\mathcal{F} = 2.4$ was heuristically found to best match the 4$^\mathrm{th}$ resonating mode of the tube and generates a recognisable brass sound.

5.2 Limit on speed of change

To reduce audible artifacts and instability issues from adding and removing points, and to stay in the sub-audio rate regime, a limit can be placed on (29) as

$$L^n_\mathrm{diff} \leq \mathcal{N}_\mathrm{maxdiff} h, \tag{43}$$

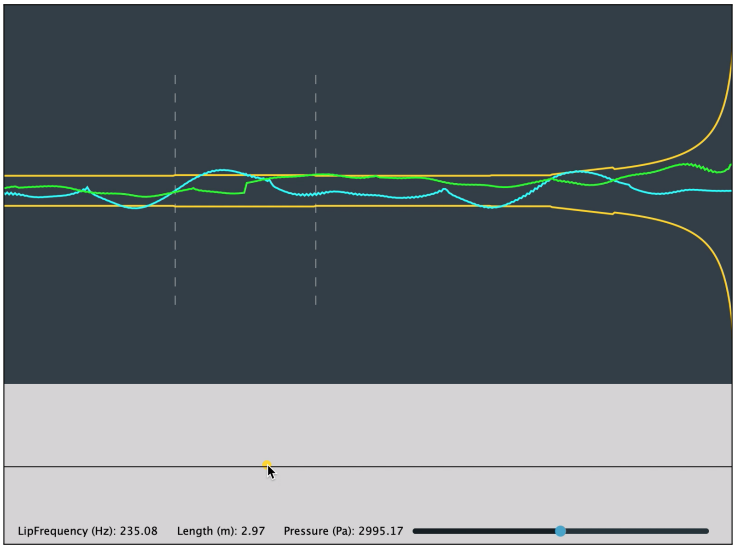


Fig. 4: *Screenshot of the graphical user interface (GUI). The geometry (in orange) as well as the states of the pressure (in blue) and velocity scaled by S (in green) are shown. For clarity, the start and end of the outer slide are denoted by dashed lines. The drift of w as explained in Section 4.3 is visible from the "kink" in the green line exactly in the middle of the outer slide.*

where $\mathcal{N}_{\text{maxdiff}}$ is the maximum change in $\mathcal{N}$ per sample and has been set to $\mathcal{N}_{\text{maxdiff}} = 1/20$. This means that a grid point can be added or removed every 20 samples and allows the entire range of L to be traversed in ca. 0.06 s at a sample rate of $f_{\text{s}} = 44100$ Hz.

5.3 State correction

The introduction of system states at $n+1$ through the centred operators in Eq. (39) seem to make the scheme implicit. It is, however, possible to calculate F_{sc} explicitly [15, 18]. The same operators also introduce the need for values at $n-1$, i.e., $p_{M^n}^{n-1}$ and q_0^{n-1}. Therefore, the vectors $\mathbf{p}^{n-1}$ and $\mathbf{q}^{n-1}$ will need to be stored, and the operations to add and remove grid points as described in 4.2 need to be applied to these as well. One could argue that only two points at the inner boundaries are needed for the calculation and to create $\mathbf{r}$ in (33) at $n-1$. For generality, we continue with the entire vectors defined over the same domains as $\mathbf{p}^n$ and $\mathbf{q}^n$ respectively.

5.4 Graphical User Interface and Control Mapping

A screenshot of the graphical user interface (GUI) is shown in Figure 4. The geometry of the tube is plotted along with paths showing the pressure states

while *application is running* **do**

Retrieve new parameters	(L^n, ω_r^n and P_m^n)
Update L_p^n and L_q^n	(Eqs. (29), (43) & (28))
Calc. $\mathcal{N}^n$ and N^n	(Eqs. (24) and (17))
Calc. α^n	(Eq. (25))
if $N^n \neq N^{n-1}$ **then**	
Add or remove point	(Eq. (32) or (36))
Update M^n and M_q^n	(Eq. (20))
end	
Calc. $p_{M^n+1}^n$ and q_{-1}^n	(Eqs. (27))
Calc. $\mathbf{v}^{n+1/2}$ and $\mathbf{w}^{n+1/2}$	(Eqs. (21b) and (21d))
Calc. $y^{n+3/2}$ w/o collision	(Eqs. (18))
Calc $g^{n+1/2}$	(Eq. (19))
Calc. $y^{n+3/2}$ with collision	(Eqs. (18))
Calc. $U_\mathrm{B}^{n+1/2}$ and $U_\mathrm{r}^{n+1/2}$	(Eqs. (18c) and (18d))
Calc. $\mathbf{p}^{n+1}$ and $\mathbf{q}^{n+1}$	(Eqs. (37))
Retrieve output	
Update system states	($\mathbf{p}^{n-1} = \mathbf{p}^n$, $\mathbf{p}^n = \mathbf{p}^{n+1}$) (same for $\mathbf{v}^{n-1/2} = \ldots$, $y^{n-1/2}$, $y^{n+1/2}$, and ψ^n)
Update N^{n-1}	($N^{n-1} = N^n$)
Increment n	

end

Algorithm 1: *Pseudocode showing the order of calculations of the algorithm implementing the trombone.*

in blue and the velocity (scaled by the geometry S) in green. The audio thread of the application runs at 44100 Hz whereas the graphics are updated at a rate of 15 Hz.

The real-time application is controlled by interacting with the bottom panel using the mouse. The x-axis is mapped to tube-length L^n and also modifies the lip-reed frequency ω_r according to Eq. (42). The y-axis changes the multiplier $\mathcal{F}$ in Eq. (42) and the black line in the vertical middle of the control panel is mapped to $\mathcal{F} = 2.4$. The pressure is modulated by a slider at the bottom of the control panel. As of now, no focus has been put on intuitive parameter mapping; it has only been implemented for simple parameter exploration.

5.5 Order of Calculation

Algorithm 1 shows the order in which the different parts of the system presented in this paper are calculated.

6 Results and Discussion

The real-time implementation has been tested on a MacBook Pro with a 2.2 GHz Intel i7 processor and was informally evaluated by the authors. The speed of the algorithm was tested with and without the graphics-thread and using three different styles of interaction: static excitation at the shortest and longest length, and rapidly (and continuously) changing L and ω_r between their minimum and maximum values given in Table 2. The pressure was kept at $P_{\mathrm{m}} = 3000$ Pa at all times. The results are shown in Table 3. Differences in CPU usage between a short and long tube length are because more grid points need to be calculated in the long case. The recalculation of the geometry maximally once every 20 samples in the rapidly moving case explains the increase in CPU usage there. These results show that the implementation can easily be used as an audio plugin, with or without graphics.

Table 3: *Average CPU usage (in %) for different graphics settings and various interactions with the application.*

Tube length	Graphics (%)	No graphics (%)
Short ($L^n = 2.593$ m)	12.1	4.3
Long ($L^n = 3.653$ m)	14.4	5.2
Rapidly changing	17.7	10.1

Informal listening tests by the authors confirm that the audio output of the simulation exhibits brass-like qualities. However, the implementation requires some further refinements to be considered as a complete trombone. Possible extensions to improve the realism of the simulation sound could be to add viscothermal losses [19] or nonlinear effects [3]. Furthermore, for lower values of the lip frequency ω_{r}, the sound exhibits extra oscillatory behaviour making the output "non-smooth". This might be due a higher average displacement of y for lower ω_{r} and the nonlinear collision present in the lip model will have a greater effect on its displacement. Variable collision stiffness might solve this issue but is left for future work.

Informal listening by the authors shows that the method used to implement the dynamic grid does not introduce perceivable audible artifacts, even when L^n is changed very rapidly. Naturally, this needs to be confirmed by formal listening tests. Despite the limit placed on the speed of change of L^n in (43) the control of the application does not exhibit a noticeable delay and changes in L^n feel immediate.

The main difference between the method in [8] and the version used here, is that the method is applied to a system of first-order equations rather than the second-order 1D wave equation. Because the connection between the inner boundaries is only applied to the grid functions describing pressure, a drift

occurs in w as it is left without boundary conditions. Although this drift does not have an effect on the output sound, as discussed in Section 4.3, too high or low values might cause rounding errors in the simulation. As it is expected that this only happens at extremely high or low values after a long simulation length, the drift is not considered an issue at this point.

7 Conclusion

In this paper, we have presented a full implementation of the trombone including a lip reed, radiation and a tube, discretised using FDTD methods on a dynamic grid. Informal evaluation by the authors shows that the implementation exhibits no audible artifacts when grid points are added and removed, even under relatively fast variation in tube length. Naturally, this needs to be confirmed by formal listening tests. Moreover, the simulation easily runs in real-time allowing it to be used as an audio plugin.

Future work will include extending the tube model to include more realistic viscothermal and nonlinear effects and variable collision stiffness in the lip model. Furthermore, the investigation of more intuitive control parameter mappings is a necessary step towards a real-time instrument.

8 Acknowledgements

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

[1] M. Campbell, "Brass instruments as we know them today," *Acta Acustica united with Acustica*, vol. 90, no. 4, pp. 600–610, 2004.

[2] A. Hirschberg, J. Gilbert, R. Msallam, and A. Wijnands, "Shock waves in trombones," *J. Acoust. Soc. Am.*, vol. 99, no. 3, pp. 1754–1758, 1996.

[3] R. Msallam, S. Dequidt, S. Tassart, and R. Causse, "Physical model of the trombone including non-linear propagation effects," in *ISMA: International Symposium of Music Acoustics*, 1997.

[4] R. Msallam, S. Dequidt, R. Causse, and S. Tassart, "Physical model of the trombone including nonlinear effects. Application to the sound synthesis of loud tones," *Acta Acustica united with Acustica*, vol. 86, no. 4, pp. 725–736, 2000.

[5] P. R. Cook, *Real sound synthesis for interactive applications*. CRC Press, 2002.

[6] R. L. Harrison, S. Bilbao, J. Perry, and T. Wishart, "An environment for physical modeling of articulated brass instruments," *Computer Music Journal*, vol. 39, no. 4, pp. 80–95, 2015.

[7] S. Bilbao and J. Chick, "Finite difference time domain simulation for the brass instrument bore," *J. Acoust. Soc. Am.*, vol. 134, no. 6, pp. 3860–3871, 2013.

[8] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "Dynamic grids for finite-difference schemes in musical instrument simulations," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx)*, 2021.

[9] A. Webster, "Acoustical impedance, and the theory of horns and of the phonograph," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 5, no. 7, pp. 275–282, 1919.

[10] S. Bilbao and R. Harrison, "Passive time-domain numerical models of viscothermal wave propagation in acoustic tubes of variable cross section," *J. Acoust. Soc. Am.*, vol. 140, pp. 728–740, 2016.

[11] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe," *Physical Review*, vol. 73, no. 2, pp. 383–406, 1948.

[12] F. Silva, P. Guillemain, J. Kergomard, B. Mallaroni, and A. Norris, "Approximation formulae for the acoustic radiation impedance of a cylindrical pipe," *Journal of Sound and Vibration*, vol. 322, pp. 255–263, 2009.

[13] R. L. Harrison-Harsley, "Physical modelling of brass instruments using finite-difference time-domain methods," Ph.D. dissertation, University of Edinburgh, 2018.

[14] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, "Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation," *Journal of the Acoustical Society of America (JASA)*, vol. 149, pp. 3502–3516, 2021.

[15] S. Bilbao, *Numerical Sound Synthesis*. United Kingdom: John Wiley & Sons, 2009.

[16] T. Smyth and F. S. Scott, "Trombone synthesis by model and measurement," *EURASIP Journal on Advances in Signal Processing*, 2011.

[17] A. H. Benade, "On the propagation of sound waves in a cylindrical conduit," *Journal of the Acoustical Society of America*, vol. 44, no. 2, pp. 616–623, 1968.

[18] S. Bilbao, "A modular percussion synthesis environment," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx)*, 2009.

[19] S. Bilbao and R. L. Harrison, "Passive time-domain numerical models of viscothermal wave propagation in acoustic tubes of variable cross section," *The Journal of the Acoustical Society of America*, vol. 140, pp. 728–740, 2016.

format the blurb

Digital versions of musical instruments have been created for several decades, and for good reason! They are more compact, more easy to maintain, and less difficult to play than their real-life counterparts. One way to digitise an instrument is to record it and play back the samples, but this does not capture the entire range of expression of the real instrument. Simulating an instrument based on its physics, including its geometry and material properties, is much more flexible to player control. Although it requires more computational power to generate the sound in real time, the simulation could possibly go beyond what is physically possible. A violin growing into a cello, bowing your trumpet, your imagination is the limit...