The Emulated Ensemble

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

Ph.D. Dissertation
Silvin Willemsen

Dissertation submitted July 2, 2021

Thesis submitted: July 2, 2021

PhD Supervisor: Prof. Stefania Serafin
Aalborg University

PhD Committee: Assoc. Prof. Olga Timcenko
Aalborg University

Prof. Julius O. Smith
Stanford University

Prof. Augusto Sarti
Politecnico di Milano

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.

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

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.

Could be fun to include this :) : "I'm a musician. Will I be out of a job if you keep making physical models?" 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.

Acknowledgements

I would like to thank my mom..

Silvin Willemsen
Aalborg University, July 2, 2021

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 VIII, 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, pp. 100–107.

[S3] M. G. Onofrei, S. Willemsen, and S. Serafin, "Implementing physical models in real-time using partitioned convolution: An adjustable spring reverb," in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021, pp. 108–114.

[S4] M. G. Onofrei, S. Willemsen, and S. Serafin, "Real-time implementation of a friction drum inspired instrument 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.

Contents

FULL DOC SWEEP: check capitalisation of headings throughout document

FULL DOC SWEEP: check hyphen in titles

Todo list

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

At the time of writing, an uncountable number of digital musical instruments exist. From digital keyboards that create sounds from a various real (and non-real) instruments to digital instrument plug-ins that music producers use in their Digital Audio Workstations (DAWs) to create their music. Until recently, many of these digital instruments were based on samples, or recordings, of their real-life counterparts. As computing power increased over the last few decades, using *physical models* rather than samples gained more and more popularity in these applications.

Physical modelling, in the context of sound and music, is a way to generate sound based on physical processes. This includes string vibrations in a guitar, air propagation in a trumpet, or even the reflections in a concert hall. This work focuses on simulating[1] traditional musical instruments using physical modelling. The interest in physically modelling traditional musical instruments is twofold: sound generation, and understanding of the underlying physical processes, the former of which is the main focus of this PhD project. One of the reasons why one would use physical models rather than samples of the real instrument, is that a model is much more flexible to player control. Consider the violin as an example. The performer controls bow force, velocity and position along the string, as well as the finger determining the pitch of the string. A physical model can generate the sound on the spot based on these performance parameters. If samples were to be used, every single combination of these parameters would need to be recorded in order to capture the entire instrument. A more in-depth reasoning behind using physical models for sound generation will be given in Section 1.4.

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

This chapter continues by giving a brief overview of the history of physical modelling for sound synthesis.

1.1 A Brief History

Digital sound synthesis gained an increased interest in the last few decades. In the 1960s, efficient sinusoidal-based sound synthesis techniques such as additive synthesis [stefania? :)], subtractive synthesis [stefania? :)], 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. This section is a little off, both history-wise and the fact that it's not really 'physical modelling'

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 musical sound is the "Bicycle Built for Two" by Kelly, Lochbaum, and Matthews in 1961[2]. It uses what later got known as the Kelly-Lochbaum vocal-tract model to generate a voice and was published the year thereafter [7].

The very first musical instrument simulations were based on discretisation of differential equations using *finite-difference time-domain* (FDTD) methods. These were carried out around 1970 by Hiller and Ruiz [8, 9, 10] and applied to the wave equation to simulate string sounds. The sound generation, however, was far from real-time and it took several minutes to generate only 1 second of sound. In 1983, Cadoz et al. introduced CORDIS, a real-time sound generating system based on *mass-spring networks* [11]. The first physical model of the bowed string was due to McIntyre et al. in their 1983 publication [12]. In the same year Karplus and Strong devised an extremely efficient way to generate a string sound in [13] later known as the Karplus-Strong algorithm. Based on these ideas, Smith coined the term *digital waveguides* around the late 1980s and early 1990s in [14, 15] and continued to develop the method [2]. Around the same time, Adrien in [16] and later Morrison and Adrien in [17] introduced *modal synthesis*, a way to generate an object's sound by decomposing it into its modes of vibration.

Although more techniques have been developed in the last 20-30 years, most of the developments in the field of physical modelling for musical instruments are based on those presented in this section. Before moving on to further details about these methods in Section 1.3, a modular approach to subdivide a musical instrument will be presented.

[2] `http://ccrma.stanford.edu/~jos/wav/daisy-klm.wav`

1.2 Exciter-Resonator Approach

Nearly any musical instrument can be subdivided into a resonator component and an exciter component, both of which can be simulated individually. This modular approach to musical instruments was first introduced by Borin, De Poli and Sarti in [18] and and later developed by De Poli and Rocchesso in [19] 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.

A resonator is a passive system, in this work mostly assumed to be linear, and does not emit sound unless triggered by an external source. Exciters can be seen as these external sources, and generally have a nonlinear element.[3] 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.

The next section will talk about various techniques that can be used to implement the resonator. Details on modelling excitations are left for Part III.

1.3 Physical Modelling Techniques

The time-evolution of dynamic systems, including that of musical instruments, can be well described by partial differential equations (PDEs) [20, 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 [20]. 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, also mentioned in Section 1.1 that are described in this literature can be found below:

[3] A quick explanation of (non)linear systems: The behaviour of linear systems does not change with the level of the input. Instead, it only scales (linearly) with the input level: an input to a linear system with twice the amplitude yields an output of twice the amplitude. The behaviour of nonlinear systems, however, does change depending on the level of the input. Although linear systems are rarely found in the real world, under low amplitude excitations they can still be considered linear and their nonlinear effects can be ignored.

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 [17], 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 [21, 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 and used as a sound-generating physical model [22].

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 by Hiller and Ruiz in [8, 9, 10] and later by Chaigne in [23, 24]. Bilbao extensively describes this method in [3, 21]. 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 technique that discretises wave propagation and scattering. The technique was first presented by Smith in [15], 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 FDTD methods (such as in [25, 26]) 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 [27, 11, 28] it is currently being further developed by Leonard and Villeneuve in a real-time, interactive environment [29, 30].

Discussion

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 to be directly numerically simulated without making any assumptions regarding travelling wave solutions or modes. DWGs, for example, assume a travelling wave solution, which makes complex 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 [3]. For a wrong choice of parameters, the implemented system could become unstable and "explode"[4]. 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 are much more computationally heavy than other methods, such as DWGs or modal synthesis techniques. The bright side, if one believes in Moore's law [31], 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, will not be an issue.

1.4 Applications of Physical Modelling

So why would we go through all this hassle of modelling musical instruments? Could we not use a recording of the original instrument 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? This section aims to answer those questions, by providing some applications of physical modelling for musical instrument simulations. Very informal section, but I kinda like it :)

1.4.1 Samples vs. Physical Modelling

First of all, physical modelling can be used to generate sound in audio plug-ins or digital instrument controllers (such as keyboards). Although many techniques to digitally simulate musical instruments exist, the bulk of the currently

[4] I learned the hard way that one should always implement a limiter when working with real-time physical models.

available digital musical instruments are still based on samples. This is due to the computational power needed to generate sounds as opposed to simple playback of recordings. Furthermore, digital musical instruments based on samples of an actual instrument, 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.

That said, it can be argued that 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 with one playing style or technique, using one specific microphone to record the sample, etc. Even if one accepts this, 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 and take an incredible amount of time to record.

On the other hand, using physical models to simulate the musical instrument allows the sound to be generated on the spot based on all the aforementioned interaction parameters. One is not stuck to a single recording of the instrument and, given the right tools or controller, one can alter the sound just as one can with its real-life counterpart.

A drawback of physical models is that, in order to generate a realistic sound, a highly accurate physical description of the original instrument is needed. Apart from (potentially) taking a lot of time to develop this model and tuning its parameters, the eventual implementation will be (much) more computationally expensive than if samples were to be used. Generally, the more accurate the model is, and thus the more true the sound is to the original, the higher the computational cost becomes.

The main trade-off between samples and physical models is thus storage versus speed, or hard-disk versus CPU. Whether one method should be used over the other depends on the situation. If efficiency is required and the lack of flexibility in the sound is not an issue, samples might be the better choice. If, on the other hand, one wants to create a full digital version of a traditional instrument that responds to player-interaction in the same way as the original instrument would, a physical model should be chosen instead.

1.4.2 Resurrect Old or Rare Instruments

Many instruments exist that are too old, too rare, or too valuable to be played. Some live behind museum glass only to be looked at by visitors, never to be played again. In these cases, it might even be hard to record samples of the musical instrument. If, however, the physics of the instrument, including

the geometry and material properties, are available, a physical model of the instrument could be created and its sound brought back too life.

Even popular musical instruments require maintenance and might need to be replaced after years of usage. A simulation of these instruments will not age, unless that is of course desired and included in the model.

1.4.3 Go beyond what is physically possible

As a digital simulation is not restricted by the laws of physics of the real world, this opens up a substantial amount 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. A physical model of a violin could potentially change size and 'morph' into a cello while the simulation is running and a player is interacting with it. New ways of interaction and expression could be devised that control the physics of the instrument, expanding the range of possibilities for the musician.

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 result in unique sounds that can only be created using physical models.

1.5 Project Objectives and Main Contributions

This section presents several research questions and provides the main objectives and contributions of the project.

How can computationally expensive physical models be made playable in real-time?

The biggest challenge in real-time audio applications, as opposed to those only involving graphics for example, is that the sample rate required is extremely high. As Nyquist's sampling theory states, a sampling rate of at least 40 kHz is necessary to produce frequencies up to the human hearing limit of 20 kHz [32]. The perceptual limit of a temporal sample rate for visuals (mostly referred to as frames per second (FPS)) is at around 60 Hz **[human visual limit]**, which is orders of magnitude smaller than the auditory sample rate.

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' [33]. Several virtual string instruments and different electric pianos have been made real-time by Pfeifle and Bader in [34, 35, 36]. 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 [37] that implement real-time physical models in hardware instruments. In the NESS project, Stefan Bilbao and his team focused on implementing systems using FDTD methods in real-time using parallelisation techniques and the GPU [38, 39].

The main objective of this work is to implement physical models simulated using FDTD methods in real time without the need of special hardware, i.e., on a regular personal computer. The objective is not to renew the underlying models themselves, but novel combinations could be made to simulate relatively unknown instruments as test cases for this objective. The instruments modelled over the course of this project are the esraj (Bowed Sitar), hammered dulcimer, hurdy gurdy, tromba Marina and the trombone, all implemented in real time using FDTD methods. A detailed description of all of these can be found in Part V and publications [A], [D] and [H].

How can (the sound of) traditional instruments be extended upon?

As mentioned in Section 1.4.3, using physical modelling to simulate real-life instruments relieves the physical limitations that the real world imposes on them. As FDTD methods are quite rigid, dynamically changing parameters 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 1.3. Therefore, one of the main objectives of this project was to devise a method to allow parameters in musical instrument simulations based on FDTD methods to be dynamically varied.

Throughout the PhD project, indeed a novel method was devised to smoothly change parameters over time, introducing this to FDTD methods. This method was published in [G] and will be described in detail in Part **??**.

How can the now-virtual instruments be controlled in a natural way?

A great challenge in musical instrument simulations is their control. In many physical instruments, one interacts immediately with the sound-creating object, such as a string on a guitar or a membrane on a drum. This allows the musician to be much more expressive than if they only used the keyboard and mouse. Expressivity, however, is not the only thing that makes an instrument interesting and enjoyable to play. The interaction with a musical instrument simulations could feel very 'dry' or unnatural as there is no haptic feedback; something which is the case in (nearly) all physical musical instruments.

The last objective of this PhD project is thus to find ways to control the instrument simulations in a natural way. Over the course of this PhD project, experiments have been done with the Sensel Morph, which is a controller containing ca. 20,000 pressure sensors that allow for highly expressive control of the instruments [40]. This controller has been used in papers [A], [B], [C]

and [D].

Although the Sensel Morph allows for more expressive control than a keyboard and mouse, it does not resemble any of the interaction paradigms of the original instruments. It was thus attempted to include a controller that would be more suited for controlling the musical instrument simulation and allow for a more intuitive control. For one of the projects, described in paper [E], a Virtual Reality implementation of the tromba marina is controlled by the PHANTOM Omni [41] which is a six-degrees-of-freedom haptic device. The controller contains a hand-held pen-like object that is attached to a robotic arm. The controller can be linked to a virtual environment and proviDe haptic feedback through the arm based on this environment.

The controllers and their implementations will be discussed in-depth in **??** or V.

1.6 Thesis Outline

The dissertation is divided into several parts which in their turn are divided in chapters. WILL SAVE THIS SECTION FOR THE END

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

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

The above parts are used as an introduction for the main contributions of the PhD project. The parts are written for beginners in the field of physical modelling for sound synthesis using FDTD methods and – with the exception of Chapter **??** do not contain any novelty.

The next part contains the main contributions of the PhD project:

Part V: Contributions presents the contributions of the PhD project to the scientific field

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

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 theory presented in this chapter and the notation have been taken from [3].

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, such as displacement from the equilibrium of a string, or the pressure in a tube, 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 [42] 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. For this reason, Newton's notation will be used for ODEs in isolation. 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 14). Also, state-of-the-art literature in the field of FDTD methods for sound synthesis use this notation [21].

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 models describing physical systems 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. Nonlinear effects such as pitch glides due to tension modulation caused by high-amplitude string vibration are not present in the simplified model and are not presented in this work.

check whether this needs to be per point along a system

Work has been done on strings with dual (transverse) polarisation by Desvages [43] and Desvages and Bilbao [44] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are couples can be found in [3, 45]. In [29], 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 sim-

plifications. 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].

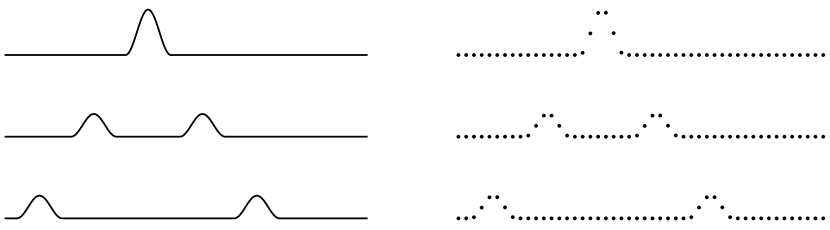


Fig. 2.1: A continuous PDE is discretised...

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. See Figure 2.2. 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).

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.

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

To summarise, for a 1D system

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

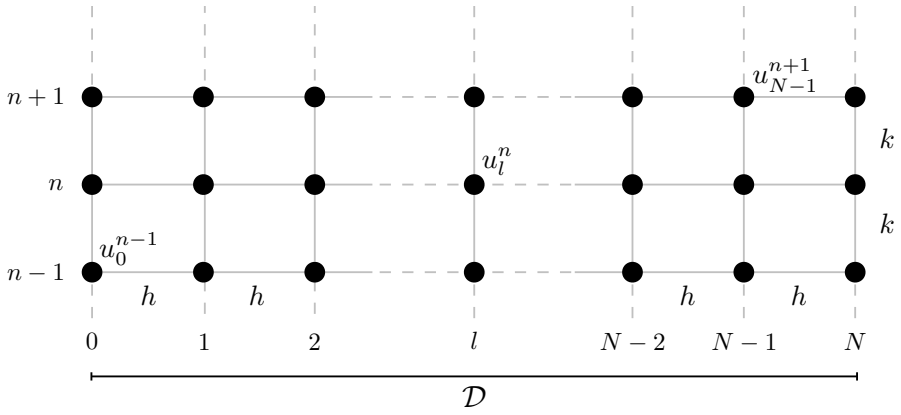


Fig. 2.2: The spatio-temporal grid that appears when a 1D system $u(x,t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n. The spatial domain $\mathcal{D}$ is divided into N intervals of length h and spatial range of interest $l = \{0, \ldots, N\}$. Time is subdivided into time steps of duration k and together with the discretised domain, forms a grid over space and time. Some grid points are labelled with the appropriate grid function.

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.

Similar operators exist for a first-order derivative in space, where the forward, backward and centred difference are

figure here visualising operators (with reference to grid figure)

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

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.[2] 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

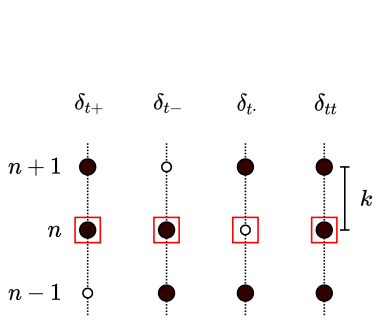
$$\partial_x^2 \approx \delta_{x+}\delta_{x-} = \delta_{xx} \triangleq \frac{1}{h^2}\left(e_{x+} - 2 + e_{x-}\right), \tag{2.8}$$

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.

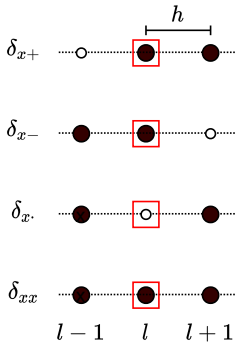
Also useful are averaging operators, all of which approximate the identity

in energy analysis, interleaved grids, etc.

[2] Alternatively, one could first apply one operator to a grid function, expand it, and apply the other operator to all individual grid functions in the result of the first expansion thereafter.



(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.

operation. The temporal forward, backward and centred averaging operators are

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

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

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

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

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

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

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

and

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

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 applies 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

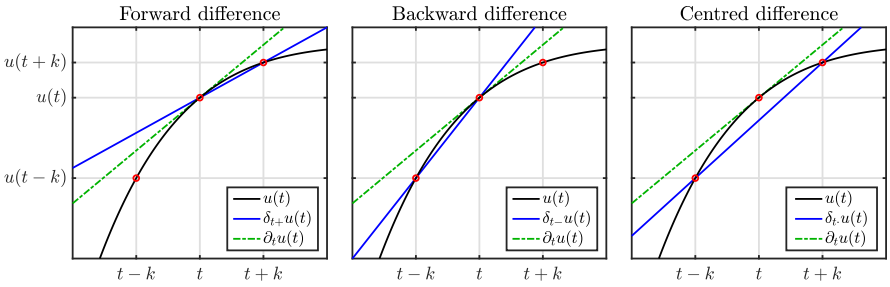


Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. One can observe that the centred difference operator much more closely approximates the derivative, or the slope, of u at t than the forward and backward difference operators.

can be observed that the derivative approximation – the slope – of the centred operator matches much more closely the true derivative of u at t.

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:

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

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

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

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

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'.

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 follows the discretisation process to a 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

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*.

move section up (stefan's comment)

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,

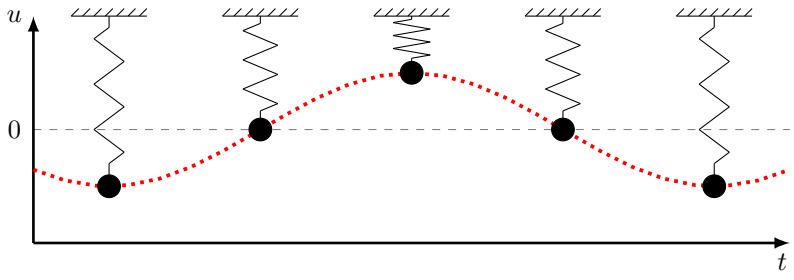


Fig. 2.5: Mass spring system over time. The system follows a harmonic (sinusoidal) motion.

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 expanded and 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.

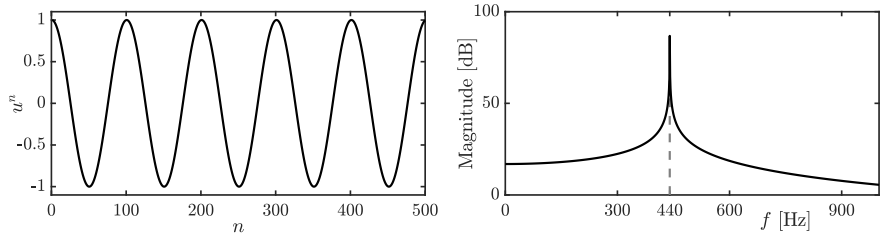


Fig. 2.6: The time-domain and frequency-domain output of a mass-spring system with $f_0 = 440$ Hz.

FULL DOC SWEEP: check hyphen in titles

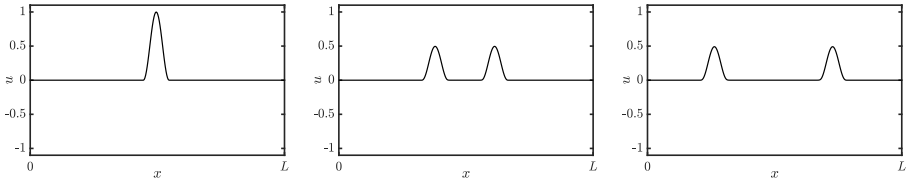
2.4.1 Continuous time

unit?

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 $\mathcal{D} = [0, L]$. 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



(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.

Intuition

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, 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.

different wording in caption

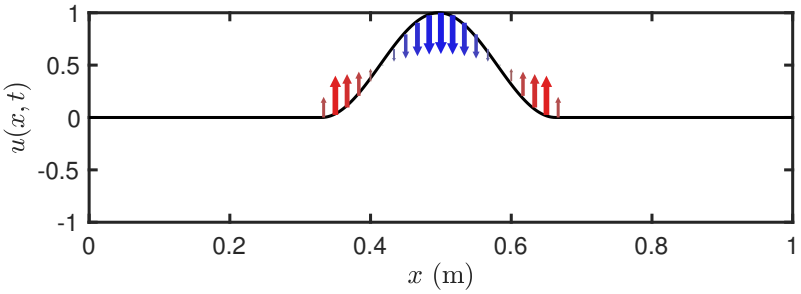


Fig. 2.8: The forces acting on the 1D wave equation described by $u(x,t)$ due to curvature. The arrows indicate the direction and magnitude of the force, and simultaneously the acceleration as these are connected through Eq. (2.38).

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 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 invert after reaching the boundary whereas in the latter incoming waves are reflected un-inverted. See Figure 2.9.

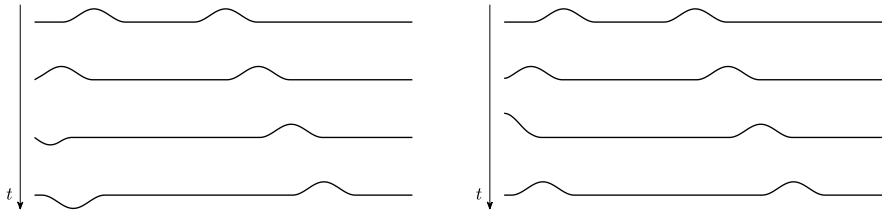
add why this is relevant?

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. [46, 45]) and can be useful to reduce the amount of parameters used to describe a system.



(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.

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 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 IV), 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 [47]

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

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 Eq. (2.44). One could visualise this stencil to be placed on the left-most point sof the grid shown in Figure 2.2. The update equation then iterates this stencil over the entire domain and calculates all values of u_l^{n+1} based on known values of u_l^n and u_l^{n-1}.

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) 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

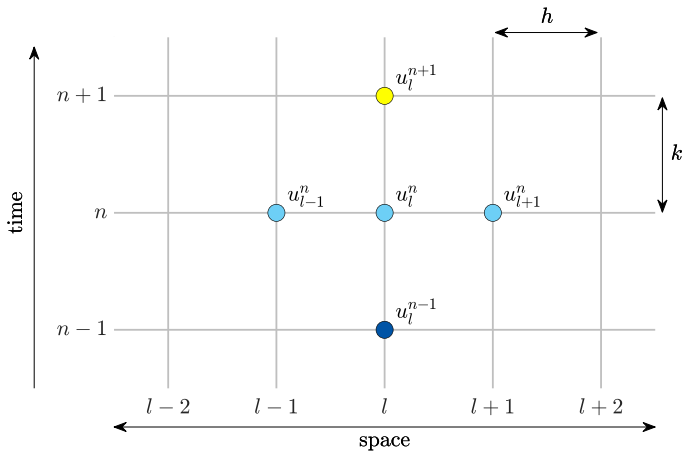


Fig. 2.10: The stencil, or region of operation, for the FD scheme in (2.42).

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 III, 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.

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 implemented using modal synthesis (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.

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 $\lambda = 1$.

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 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.

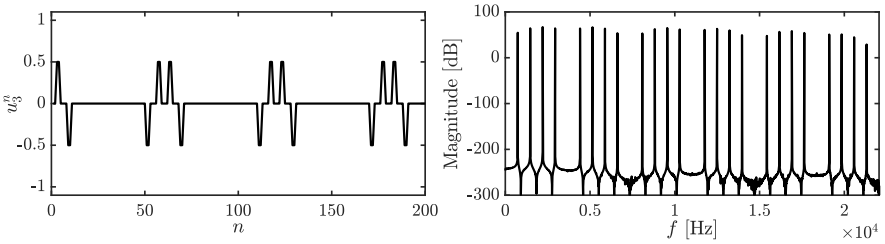


Fig. 2.11: The time-domain and frequency-domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_s = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_c = 0.2N$) and $w = 4$ and the output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

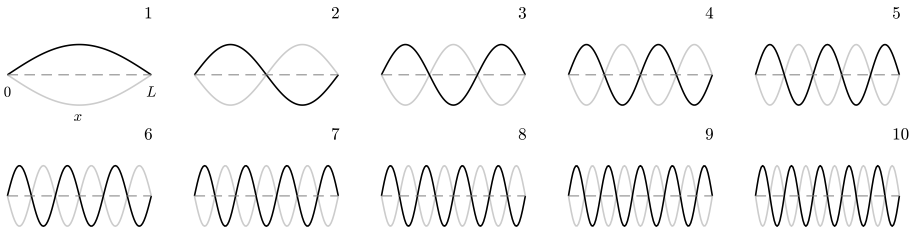


Fig. 2.12: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions defined for $x \in [0, L]$ (only shown for mode 1). The modes are normalised to have the same amplitude and vibrate at their respective modal frequencies with the extremes indicated by the black and the grey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

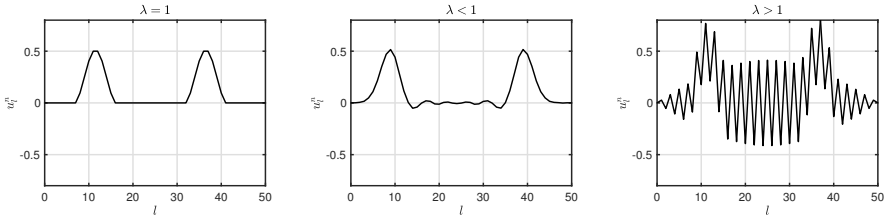
2.4.4 Stability and Simulation Quality

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![3]

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.

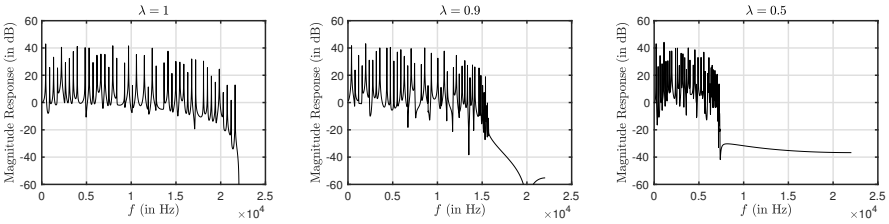
If $\lambda < 1$, the quality of the simulation decreases in an effect called *numer-*

[3]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. 2.13: 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.



(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.

ical 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 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 $c = 1500\,\mathrm{m/s}$ 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.

Intuition

It might not be immediately clear why a too low value for h might cause instability. Some intuition is provided in [3, Fig. 6.9], but here I would like to provide an alternative, hopefully more tangible way to see this.

In a FD implementation of the 1D wave equation, grid points can only affect their neighbours as seen in update equation (2.44). Using the values in Table 2.1as an example, $N = 30$ and if $\lambda = 1$, it takes exactly 30 samples, or iterations, for a wave to travel from one boundary to the other.

If h were to be chosen to be twice as big so that there are only half as many intervals between the grid points as per Eq. (2.53) ($N = 15$), the grid points could be set to 'affect' their neighbours to a lesser degree. This way, the wave still takes the same amount of time to travel between the boundaries and

the fundamental frequency stays approximately the same. This is essentially what happens when $\lambda < 1$ (in this case $\lambda = 0.5$) and can be observed from the update equation in Eq. (2.44); the effect that the neighbouring grid points have on each other will indeed be less. The output of the system will have approximately the same fundamental frequency as if $\lambda = 1$, but its partials will be detuned due to numerical dispersion as explained in this section.

If, on the other hand, h were to be chosen to be twice as small so that there are twice as many intervals between grid points ($N = 60$), it is impossible for the waves to travel from one boundary to the other in 30 samples. If they could interact with their second neighbour, this would be possible, but the working of the FD scheme in (2.42) does not allow for this. Indeed, as $\lambda = 2$ in this case, the effect that the grid points have on each other will be disproportionate. In a way, grid points have too much energy that they can not lose to their neighbours, because their effect should have reached their second neighbour over the course of one sample. The way to solve this would be to halve the time step k (or double the sample rate f_s), which would allow grid points to interact with their second neighbours over the course of once the the old time step (as this is now divided into two time steps). This also shows in the fact that $\lambda = 1$ again (as halving k cancels out halving h) and grid points transfer their energy to their neighbours proportionately again.

figure?

Possible solution

One of the main contributions of the PhD project is presented in Chapter 11, where a 'fractional' number of intervals is introduced. This removes the necessity of the flooring operation in Eq. (2.53) and circumvents the recalculation of h to always satisfy the stability condition with equality while retaining the correct fundamental frequency.

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, it 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 $\mathbf{A}^T$. This 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. The multiplication of two matrices $\mathbf{A}$ and $\mathbf{B}$ to a resulting matrix $\mathbf{C}$ is defined as

$$c_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj}, \tag{3.2}$$

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

(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).

where K is both the number columns of matrix $\mathbf{A}$ and the number of rows in matrix $\mathbf{B}$. It thus follows that, 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.

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

$$\overbrace{(L \times M)}^{\mathbf{A}} \cdot \overbrace{(M \times N)}^{\mathbf{B}} = \overbrace{(L \times N)}^{\mathbf{C}}. \tag{3.3}$$

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 $\mathbf{w}$:

$$\mathbf{Au} = \mathbf{w}. \tag{3.4}$$

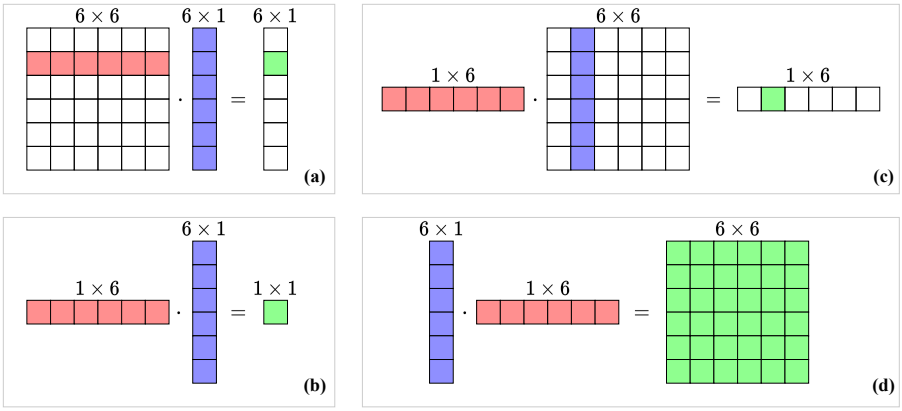


Fig. 3.2: Visualisation of valid matrix multiplications (see Eq. (3.2)). The "inner" dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of "outer" dimensions (rows of the left matrix and columns of the right).

Expanding this operation (with reference to Eq. (3.2)) results in

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} a_{00}u_0 + a_{01}u_1 + \dots + a_{0N}u_N \\ a_{10}u_0 + a_{11}u_1 + \dots + a_{1N}u_N \\ \vdots \\ a_{N0}u_0 + a_{N1}u_1 + \dots + a_{NN}u_N \end{bmatrix}}_{\mathbf{w}} \tag{3.5}$$

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 on the boundary conditions. Not assuming a specific size for now, the FD operators in (2.6) can be written in matrix form according to

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

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

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:

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

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

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:

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

and has the following special property

$$\mathbf{IA} = \mathbf{AI} = \mathbf{A}.$$

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

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

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

$$\mathbf{D}_{xx} = \frac{1}{h^2} \begin{bmatrix} -2 & 2 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 2 & -2 \end{bmatrix}, \tag{3.7}$$

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

$$\frac{1}{k^2}\left(\mathbf{u}^{n+1} - 2\mathbf{u} + \mathbf{u}^{n-1}\right) = c^2 \mathbf{D}_{xx} \mathbf{u}^n, \tag{3.8}$$

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

$$\mathbf{u}^{n+1} = (2\mathbf{I} + c^2k^2\mathbf{D}_{xx})\mathbf{u}^n - \mathbf{u}^{n-1}. \tag{3.9}$$

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

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \tag{3.10}$$

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:

$$\mathbf{Au} = \mathbf{w}. \tag{3.11}$$

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 $\mathbf{w}$

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \tag{3.12}$$

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

ask stefan

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.

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

$$\mathbf{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}. \tag{3.13}$$

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

$$\lambda_p = \text{eig}_p(\mathbf{A}), \tag{3.14}$$

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. Unless denoted otherwise, the notation and theory will follow [3].

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

$$\langle f, g \rangle_{\mathcal{D}} = \int_{\mathcal{D}} fg dx \quad \text{and} \quad \|f\|_{\mathcal{D}} = \sqrt{\langle f, f \rangle_{\mathcal{D}}}. \tag{3.15}$$

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 $d = \{0, \dots, N\}$, is

$$\langle f_l^n, g_l^n \rangle_d = \sum_{l=0}^{N} h f_l^n g_l^n, \tag{3.16}$$

where the multiplication by h is the discrete counterpart of dx in the continuous definition in (3.15). Also useful are the primed inner product

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

and the more general weighted inner product

$$\langle f_l^n, g_l^n \rangle_d^{\epsilon_\mathrm{l}, \epsilon_\mathrm{r}} = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{\epsilon_\mathrm{l}}{2} h f_0^n g_0^n + \frac{\epsilon_\mathrm{r}}{2} h f_N^n g_N^n, \tag{3.18}$$

which scale the boundary points of the regular inner product. Naturally, if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 1$, Eq. (3.18) reduces to Eq. (3.17), and if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 2$, (3.18) reduces to (3.16).

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. (3.15), integration by parts is defined as

$$\langle f, \partial_x g \rangle_{\mathcal{D}} = -\langle \partial_x f, g \rangle_{\mathcal{D}} + fg|_0^L \tag{3.19}$$

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. (3.16), two variants of summation by parts are defined as

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_d + f_{N+1}^n g_N^n - f_0^n g_{-1}^n, \tag{3.20a}$$
$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_d + f_N^n g_{N+1}^n - f_{-1}^n g_0^n. \tag{3.20b}$$

A derivation of Eq. (3.20a) is given below. As in the case of integration by parts in Eq. (3.19), 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+1}^n and f_{-1}^n. 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 $\underline{d} = \{0, \ldots, N-1\}$, $\overline{d} = \{1, \ldots, N\}$ and $\underline{\overline{d}} = \{1, \ldots, N-1\}$. The following identities can be shown to hold

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_{\underline{d}} + f_N^n g_N^n - f_0^n g_{-1}^n, \tag{3.21a}$$
$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\overline{d}} + f_N^n g_{N+1}^n - f_0^n g_0^n, \tag{3.21b}$$

and, using the primed inner product in Eq. (3.17),

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d' = -\langle \delta_{x+} f_l^n, g_l^n \rangle_{\underline{d}} + f_N^n \mu_{x-} g_N^n - f_0^n \mu_{x-} g_0^n, \tag{3.22a}$$
$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d' = -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\overline{d}} + f_N^n \mu_{x+} g_N^n - f_0^n \mu_{x+} g_0^n, \tag{3.22b}$$

all of which will prove useful in energy analysis techniques later on. A derivation of (3.21a) is given below.

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

$$\langle f, \delta_{xx} g\rangle_d = \langle \delta_{xx} f, g\rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \quad (3.23a)$$

$$\langle f, \delta_{xx} g\rangle_d = \langle \delta_{xx} f, g\rangle_{\underline{d}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0, \quad (3.23b)$$

$$\langle f, \delta_{xx} g\rangle'_d = \langle \delta_{xx} f, g\rangle'_d + f_N \delta_{x\cdot} g_N - g_N \delta_{x\cdot} f_N - f_0 \delta_{x\cdot} g_0 + g_0 \delta_{x\cdot} f_0 \quad (3.23c)$$

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. (3.20a) and (3.21a) as they have the same inner product as a starting point, but yield different results. In the following, $d = \{0, \ldots, N\}$ and $N = 2$ are used.

Starting with Eq. (3.20a), suppressing the n superscript for brevity, and using the definition for the discrete inner product in Eq. (3.16), 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. (3.20a).

Similarly, identity (3.21a) 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

$$(\delta_{t\cdot} u_l^n)(\delta_{tt} u_l^n) = \delta_{t+}\left(\frac{1}{2}(\delta_{t-} u_l^n)^2\right), \tag{3.24a}$$

$$(\delta_{t\cdot} u_l^n) u_l^n = \delta_{t+}\left(\frac{1}{2} u_l^n e_{t-} u_l^n\right), \tag{3.24b}$$

$$(\delta_{t+} u_l^n)(\mu_{t+} u_l^n) = \delta_{t+}\left(\frac{1}{2}(u_l^n)^2\right), \tag{3.24c}$$

$$(\delta_{t\cdot} u_l^n)(\mu_{t\cdot} u_l^n) = \delta_{t\cdot}\left(\frac{1}{2}(u_l^n)^2\right), \tag{3.24d}$$

$$u_l^n e_{t-} u_l^n = (\mu_{t-} u_l^n)^2 - \frac{k^2}{4}(\delta_{t-} u_l^n)^2 \tag{3.24e}$$

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

$$\delta_{t+}(u_l^n w_l^n) = (\delta_{t+} u_l^n)(\mu_{t+} w_l^n) + (\mu_{t+} u_l^n)(\delta_{t+} w_l^n). \tag{3.25}$$

The same rule applies when the backward operator $\delta_{t-}(u_l^n w_l^n)$ or centred operator $\delta_{t\cdot}(u_l^n w_l^n)$ is used. In that case, the forward operators δ_{t+} and μ_{t+} in Eq. (3.25) need to be substituted for the backward or centred versions of the operators respectively.

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 [48, 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 [49] co-authored by John von Neumann. Later, this technique got a more general treatment in [50] 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]

$$\hat{u} = \sum_{n=-\infty}^{\infty} u_l^n z^{-n} \quad \text{and} \quad \tilde{u} = \sum_{l=-\infty}^{\infty} u_l^n e^{-jl\beta h} \tag{3.26}$$

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) [50]

check reference

$$u_l^n \overset{\mathcal{A}}{\Longrightarrow} z^n e^{jl\beta h} \tag{3.27}$$

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

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 [51, p. 71]:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \quad \Rightarrow \quad \sin^2(x) = \frac{e^{j2x} + e^{-j2x}}{-4} + \frac{1}{2}, \tag{3.28a}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \quad \Rightarrow \quad \cos^2(x) = \frac{e^{j2x} + e^{-j2x}}{4} + \frac{1}{2}. \tag{3.28b}$$

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 (3.27) with frequently appearing temporal and spatial shifts.

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 (3.28a) 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

$$\begin{aligned} \delta_{t+} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(z - 1\right), & \delta_{t-} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(1 - z^{-1}\right), \\ \delta_{t\cdot} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{2k}\left(z - z^{-1}\right), & \delta_{tt} u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k^2}\left(z - 2 + z^{-1}\right) \end{aligned} \tag{3.29}$$

and for spatial operators identity (3.28a) can be used to obtain

$$\delta_{xx} u_l^n \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2}\sin^2(\beta h/2), \tag{3.30a}$$

$$\delta_{xxxx} u_l^n \overset{\mathcal{A}}{\Longrightarrow} \frac{16}{h^4}\sin^4(\beta h/2). \tag{3.30b}$$

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, nonlinear 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

$$|z| \leq 1. \tag{3.31}$$

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 (3.31) for all wave numbers β. It can be shown that for a polynomial of the form

$$z^2 + a^{(1)}z + a^{(2)} \tag{3.32}$$

its roots satisfy condition (3.31) when it abides the following condition [3]

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

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

$$|a^{(1)}| \leq 2, \tag{3.34}$$

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 (3.27) with $l = 0$. Using Table 3.1 and Eqs. (3.29) 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:

$$z - \left(2 - \frac{Kk^2}{M}\right) + z^{-1} = 0. \tag{3.35}$$

To begin to prove stability, this equation needs to be written in the form found in (3.32). Multiplying all the terms by z, and noticing that $a^{(2)} = 1$, we could continue with condition (3.34). 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 (3.34) 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 (as they are already defined as being non-negative) and the first condition is always satisfied. The second condition is then easily solved for k by

$$k < 2\sqrt{\frac{M}{K}}. \tag{3.36}$$

Recalling that $\omega_0 = \sqrt{K/M}$, Eq (3.36) can be more compactly written as

$$k < \frac{2}{\omega_0}. \tag{3.37}$$

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. (3.29) and (3.30a):

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

Also recalling that

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

the characteristic equation of the 1D wave equation is

$$z + \left(4\lambda^2 \sin^2(\beta h/2) - 2\right) + z^{-1} = 0. \tag{3.39}$$

The scheme is then stable if the roots satisfy condition (3.31). As the characteristic equation is of the form in (3.32) (after multiplication with z) with $a^{(2)} = 1$,

stability is shown by abiding condition (3.34) for all β and when applied to the characteristic equation (3.39), 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

$$h \geq ck, \tag{3.40}$$

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) [52] 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. (3.15)), 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

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q} - \mathfrak{p}. \tag{3.41}$$

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,

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.42}$$

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:

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \tag{3.43}$$

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 (3.15). 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. (3.42) and can be plotted according

$$\mathfrak{h}_{\text{e}}^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0, \tag{3.44}$$

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. [53]):

$$\mathfrak{h}_{\mathrm{e}}^{n} = \frac{\mathfrak{h}^{n} - \mathfrak{h}^{0} + k\sum_{m=0}^{n-1}(\mathfrak{q}^{m} + \mathfrak{p}^{m})}{\mathfrak{h}^{0}}, \quad \text{if } \mathfrak{h}^{0} \neq 0. \tag{3.45}$$

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 $(\delta_{t\cdot}u^{n})$:

$$\delta_{t+}\mathfrak{h} = M(\delta_{t\cdot}u^{n})(\delta_{tt}u^{n}) + K(\delta_{t\cdot}u^{n})(u^{n}) = 0. \tag{3.46}$$

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 (3.46), one can use identities (3.24a) and (3.24b) to get the following:

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

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

$$\mathfrak{h} = \frac{M}{2}(\delta_{t-}u^{n})^{2} + \frac{K}{2}u^{n}e_{t-}u^{n} = 0. \tag{3.48}$$

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-}u^{n})^{2}, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}u^{n}e_{t-}u^{n}. \tag{3.49}$$

Step 3: Check units

As mentioned above, the energy $\mathfrak{h}$ needs to be in Joules, or $\mathrm{kg} \cdot \mathrm{m}^2 \cdot \mathrm{s}^{-2}$. Taking the terms in Eq. (3.49) one-by-one and writing them in their units results in

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

and indeed have the correct units.

Step 4: Implementation

Equation (3.54) can then be implemented in same for-loop recursion where the update is calculated.

```
%% Calculate the energy using Eq. (3.49)

% 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. (3.44)) of the mass-spring system and shows that the deviation is indeed within machine precision.

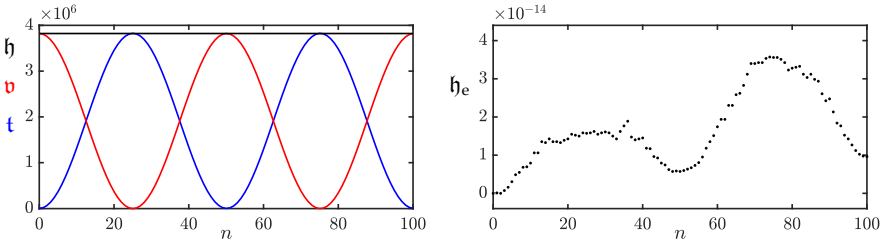


Fig. 3.3: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.44)). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within 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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n, \tag{3.50}$$

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. (3.50) 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:

$$\delta_{t+}\mathfrak{h} = \rho A \langle \delta_{t\cdot} u_l^n, \delta_{tt} u_l^n \rangle_d - T \langle \delta_{t\cdot} u_l^n, \delta_{xx} u_l^n \rangle_d = 0. \tag{3.51}$$

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. (3.51), 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 (3.21a) 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. (3.16), Eq. (3.51) can be expanded to

$$\delta_{t+}\mathfrak{h} = \rho A \sum_{l=0}^{N} h(\delta_{t\cdot} u_l^n)(\delta_{tt} u_l^n) + T \sum_{l=0}^{N} h(\delta_{t\cdot}\delta_{x+} u_l^n)(\delta_{x+} u_l^n) \tag{3.52}$$

Then, using identities (3.24a) and (3.24b), δ_{t+} can be isolated

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}\right), \tag{3.53}$$

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

$$\begin{gathered}\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad \text{and} \quad \mathfrak{v} = \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}.\end{gathered} \tag{3.54}$$

Step 3: Check units

Writing out the definitions for kinetic and potential energy in Eq. (3.54) 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. (3.54) can be implemented with the following code in the for-loop recursion:

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

% 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![1] Figure 3.4 shows the plot of the normalised energy according to Eq. (3.44) and shows that the deviation of $\mathfrak{h}^n$ is within machine precision.

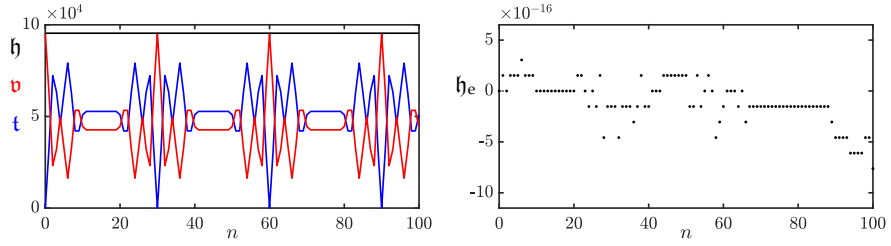


Fig. 3.4: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.44)) and shows that the deviation of the energy is within machine precision.

Neumann boundary conditions

If Neumann boundary conditions – as per Eq. (2.48b) – are used instead, the primed inner product in Eq. (3.17) needs to be used in Step 1. Using the identity in (3.22a), summation by parts of the second term results in

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

where the boundary term

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

As the Neumann boundary condition states that

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

the boundary term vanishes and the energy balance results in

$$\begin{gathered} \mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\left(\|\delta_{t-} u_l^n\|'_d\right)^2, \quad \text{and} \quad \frac{T}{2}\langle \delta_{x+} u_l^n, e_{t-}\delta_{x+} u_l^n \rangle_{\underline{d}}. \end{gathered} \tag{3.55}$$

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

[1] As can be seen from the definition of $\mathfrak{v}$ in Eq. (3.54), the domain used for the inner product is $\underline{d} = \{0, \ldots, N-1\}$ and $\mathfrak{v}$ contains a forward difference in its definition requiring u_N^n as well.

```
%% Calculate the energy using Eq. (3.55)

% 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!

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

$$x^2 + y^2 + 2axy \tag{3.56}$$

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

Equation (3.56) can be used to prove stability for the mass spring system using the energy balance in Eq. (3.49). 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. (3.49)

yields

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

Dividing all terms by $M/2k^2$ this equation is of the form in Eq. (3.56):

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

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

$$k < 2\sqrt{\frac{M}{K}} \tag{3.57}$$

which is identical to the definition in Eq. (3.36).

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. (3.54) 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 (3.24e) 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]

$$\|\delta_{x+}u_l^n\|_{\underline{d}} \leq \frac{2}{h}\|u_l^n\|'_d \leq \frac{2}{h}\|u_l^n\|_d, \tag{3.58}$$

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

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

Substituting this condition into the energy balance in Eq. (3.54) yields

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \geq \frac{1}{2}\left(1-\lambda^2\right)\|\delta_{t-}u_l^n\|^2_d + \frac{c^2}{2}\|\mu_{t-}\delta_{x+}u_l^n\|^2_{\underline{d}}, \tag{3.59}$$

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. Modal analysis can be used to obtain objective data on what modes a FD scheme should contain. This can then be used to determine whether this matches one's expectations or whether the output of the system matches what the analysis predicted. This section will show how to numerically obtain the modal frequencies of an FD implementation using the 1D wave equation as a test case.

some citation here

We start by using the matrix form of the 1D wave equation from Eq. (3.8)

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

more explanation, perhaps refer to von neumann analysis in 3.3

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

$$(z - 2 + z^{-1})\boldsymbol{\phi} = c^2k^2\mathbf{D}_{xx}\boldsymbol{\phi}. \tag{3.60}$$

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:

$$\begin{aligned} z_p - 2 + z_p^{-1} &= c^2k^2\text{eig}_p(\mathbf{D}_{xx}), \\ z_p + \Big(- 2 - c^2k^2\text{eig}_p(\mathbf{D}_{xx})\Big) + z_p^{-1} &= 0. \end{aligned} \tag{3.61}$$

check with Stefan

Furthermore, we can substitute a test solution $z_p = e^{s_pk}$ 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.[2] As there is no damping present in the system, the test solution reduces to $z_p = e^{j\omega_pk}$ which can be substituted into Eq (3.5) to get

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

Finally, using the trigonometric identity in Eq. (3.28a) we get

$$\begin{aligned} \sin^2(\omega_pk/2) + \frac{c^2k^2}{4}\text{eig}_p(\mathbf{D}_{xx}) &= 0, \\ \sin(\omega_pk/2) &= \frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}, \\ \omega_p &= \frac{2}{k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right). \end{aligned} \tag{3.62}$$

and can be rewritten to

$$f_p = \frac{1}{\pi k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right) \tag{3.63}$$

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

[2] Notice that regardless of the possible damping coefficient per mode, the eventual amplitude of each will mostly be determined by the locations of the excitation and output as discussed in Section 2.4.3.

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 (3.28) 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

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^{n} + \mathbf{C}\mathbf{u}^{n-1}, \tag{3.64}$$

can be rewritten to

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^{n}\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{A}^{-1}\mathbf{B} & \mathbf{A}^{-1}\mathbf{C}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix}\mathbf{u}^{n}\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^{n}} \tag{3.65}$$

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)

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{3.66}$$

which can be solved for the pth eigenvalue as

$$z_p = \text{eig}_p(\mathbf{Q}). \tag{3.67}$$

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

$$\begin{aligned} e^{s_p k} &= \text{eig}_p(\mathbf{Q}),\\ s_p &= \frac{1}{k}\ln\left(\text{eig}_p(\mathbf{Q})\right). \end{aligned} \tag{3.68}$$

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

$$\omega_p = \Im(s_p) \quad \text{and} \quad \sigma_p = \Re(s_p), \tag{3.69}$$

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. (3.68) 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 real world 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 $\kappa = \sqrt{EI/\rho A}$.

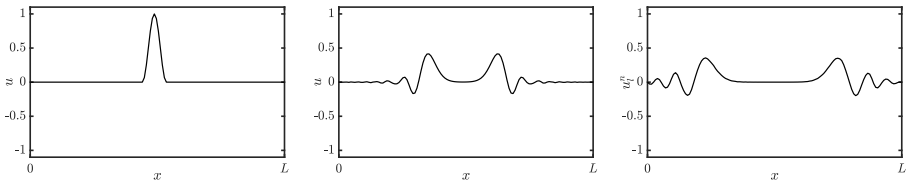
The difference between the ideal string (1D wave equation) and the stiff string is the presence of the 4$^{\text{th}}$-order spatial derivative in the stiffness term which causes frequency dispersion. As opposed to unwanted numerical dispersion due to numerical error (see Section 2.4.4) this type of 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 [54] 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

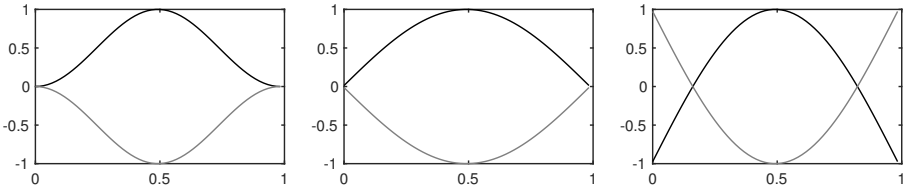
Section 2.4 presents two types of boundary conditions for the 1D wave equation in Eq. (2.39). In the case of the stiff string, these can be extended to

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

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

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

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*. A scheme is explicit if the values of u_l^{n+1} can explicitly be calculated from known values. If this is not the case and values of fx. u_{l+1}^{n+1} and u_{l-1}^{n+1} are required to calculate u_l^{n+1}, the scheme is called an *implicit*. An example of an implicit scheme using the centred operator for the temporal derivative in the frequency-dependent damping term 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

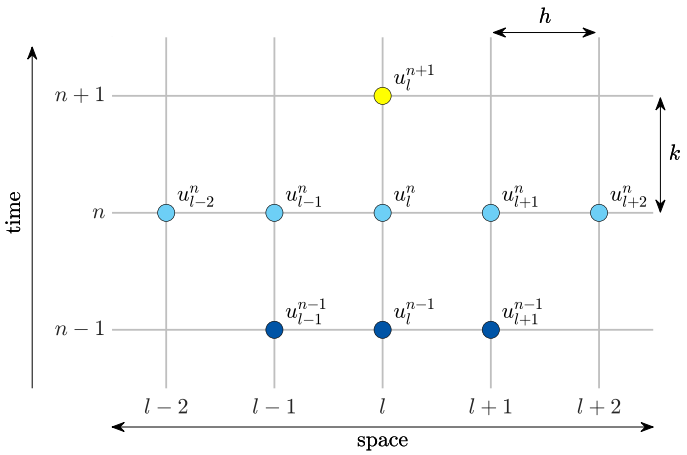


Fig. 4.3: The stencil for the damped stiff string scheme in Eq. (4.7).

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^n_{-1} and u^n_{N+1} are needed. A definition for u^n_{-1} can be found by expanding Eq. (4.16b) at $l = 0$:

$$\begin{aligned} & \frac{1}{h^2}\left(u^n_1 - 2u^n_0 + u^n_{-1}\right) = 0, \\ \overset{u^n_0=0}{\Longleftrightarrow} \quad & u^n_1 + u^n_{-1} = 0, \\ & u^n_{-1} = -u^n_1, \end{aligned} \tag{4.18}$$

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

$$u^n_{N+1} = -u^n_{N-1}.$$

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

$$\begin{aligned} (1+\sigma_0 k)u^{n+1}_1 = & \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u^n_1 + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)u^n_2 \\ & - \mu^2 u^n_3 + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2}\right)u^{n-1}_1 - \frac{2\sigma_1 k}{h^2}u^{n-1}_2. \end{aligned} \tag{4.19}$$

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

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

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^n_{-1} and u^n_{-2} at the left and u^n_{N+1} and u^n_{N+2} at the right boundary respectively. The combined operator in Eq. (4.16c) is defined as:

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

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

$$\begin{aligned} & \frac{1}{2h^3}\left(u^n_2 - 2u^n_1 + 2u^n_{-1} - u^n_{-2}\right) = 0, \\ & u^n_{-2} = u^n_2 - 2u^n_1 + 2u^n_{-1}. \end{aligned}$$

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 (3.6) 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 (3.6) 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 (3.7) 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.

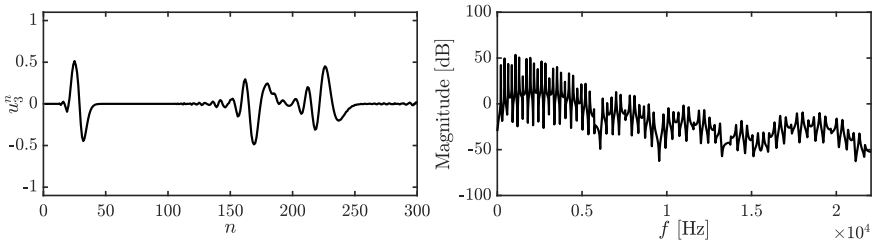


Fig. 4.4: The time-domain and frequency-domain output of the stiff string. The parameters are set as in Table 4.1 with $E = 7 \cdot 10^{11}$ to highlight dispersive effects and $T = 1129$. From the left panel, one can observe that over time, dispersive effects show due to the stiffness in the string. 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. (3.29) for the temporal operators and Eqs. (3.30a) and (3.30b) 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. (3.32), 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 (3.33) 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

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

which is a quadratic equation in h^2. Using the quadratic formula, the grid spacing h can then be shown to be bounded by

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

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. (3.16)) 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. (3.21a) for the second term and Eq. (3.23b) 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. (3.17) is used in Step 1 and identity (3.23c) 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 (3.24a) and (3.24b) 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. (3.54), 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

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

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

$$\begin{aligned}2\sigma_0\rho A\|\delta_{t\cdot}u_l^n\|_d^2 &\xrightarrow{\text{in units}} \mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2\cdot\mathrm{m}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})^2\\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},\\ -2\sigma_1\rho A\langle\delta_{t\cdot}u_l^n, \delta_{t-}\delta_{xx}u_l^n\rangle_d &\xrightarrow{\text{in units}} \mathrm{m}^2\cdot\mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2\\ &\quad\cdot\mathrm{m}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})(\mathrm{s}^{-1}\cdot\mathrm{m}^{-2}\cdot\mathrm{m}),\\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},\end{aligned}$$

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. (3.45) is within machine precision.

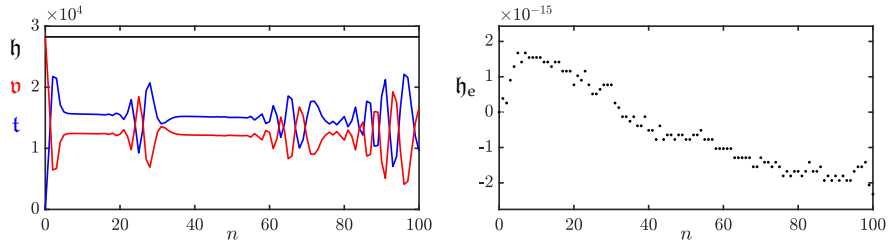


Fig. 4.5: The kinetic (blue), potential (red), and total (black) energy of an implementation of the stiff string are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.45)) and shows that the deviation of the energy is within machine precision.

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

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

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:

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

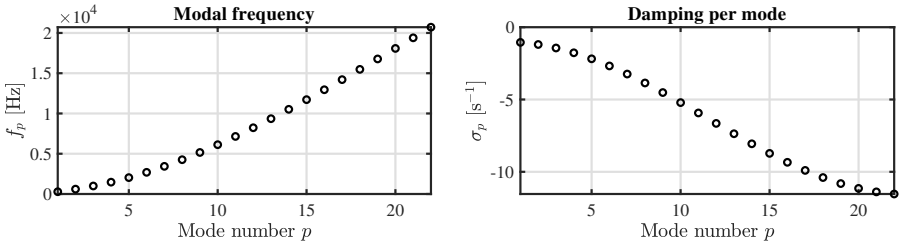


Fig. 4.6: The modal frequencies and damping per mode for the stiff string using the values in 4.1 and $T = 1885$ and $E = 2 \cdot 10^{14}$ to highlight effects of stiffness. Notice form the left panel that the frequency increases exponentially with the mode number. The right panel shows that higher modes 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 12.4.1). 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 $l \in \{1, \ldots, N-1\}$, the sytem will have $N-1$ unknowns (u_l^{n+1} for $l \in \{1, \ldots, N-1\}$) that can be calculated using $N-1$ (updat) equations. Writing this in matrix form using column vector $\mathbf{u}^n = [u_1^n, u_2^n, \ldots, u_{N-1}^n]$ 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.

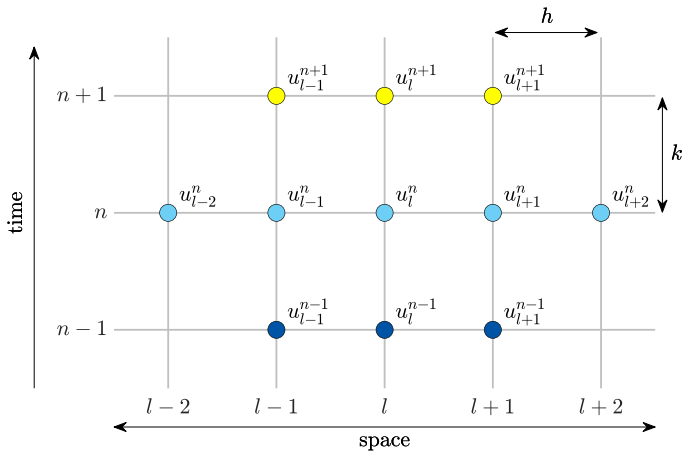


Fig. 4.7: The stencil for the damped stiff string scheme in (4.36).

(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. (3.32) 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 (3.33), 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. (3.64), 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, which is much higher than one would normally use.

One can observe from Figure 4.8 that especially higher-frequency modes in the explicit scheme are affected by σ_1. In the continuous case, the modal frequencies should only be affected by values for c and κ as per Eq. (4.3) and the damping should not influence the frequencies of the partials, as one could expect. However, as σ_1 increases, h increases due to Eq. (4.15), causing λ and μ to decrease. This introduces numerical dispersion as explained in Section 2.4.4, and the higher the value of σ_1, the more numerical dispersion it introduces in the scheme.

As the stability condition for the implicit scheme in Eq. (4.40) does not contain σ_1, this value will not affect λ and μ and will thus not affect the modal frequencies. As can be observed from the figure, it even allows for one more grid point to be included in the simulation. It can be concluded that because the frequency-dependent damping term no longer affects the stability condition for the implicit scheme, a more accurate simulation can be obtained with fewer numerically dispersive effects.

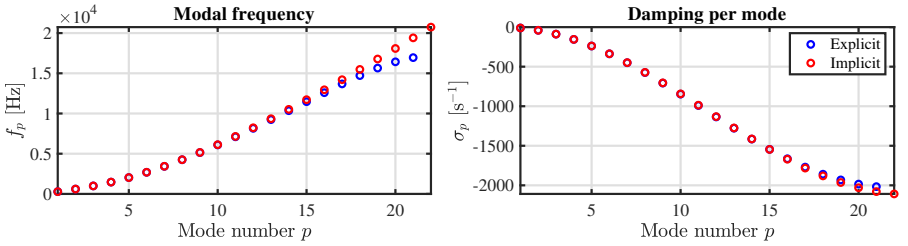


Fig. 4.8: A comparison between the modal frequencies and damping per mode of the explicit (blue) and implicit (red) scheme. Here, $T = 1885$, $E = 2 \cdot 10^{14}$ and $\sigma_1 = 1$ to highlight differences between the two schemes. One can observe that the modes of the implicit scheme follow the expected 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. 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 [55]

$$S\partial_t^2\Psi = c^2\partial_x(S\partial_x\Psi), \tag{5.1}$$

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 [21]) 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 [21]. Also see Figure 5.1

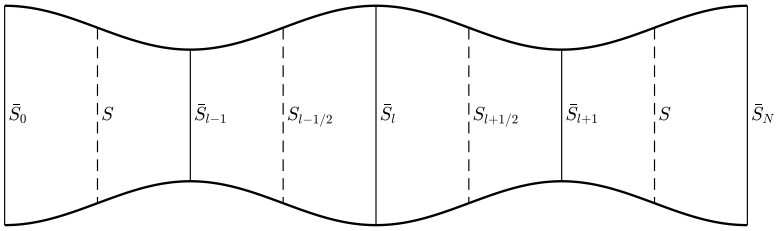


Fig. 5.1: Approximations to $S(x)$..left off here!

5.1.2 Boundary Conditions

The choices for boundary conditions in an acoustic tube are open and closed, defined as [21]

$$\begin{aligned} \partial_t \Psi &= 0 \text{ (open, Dirichlet)} \\ \partial_x \Psi &= 0 \text{ (closed, Neumann),} \end{aligned} \tag{5.7}$$

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 $u(x, y, t) \approx u^n_{l,m}$ 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 $u^n_{l,m}$. Additional ones are

$$e_{y+}u^n_{l,m} = u^n_{l,m+1}, \quad \text{and} \quad e_{y-}u^n_{l,m} = u^n_{l,m-1}. \tag{6.2}$$

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:

$$u^n_{l,m} = z^n e^{jh(l\beta_x + m\beta_y)} \tag{6.3}$$

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 [56].

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

$$\partial_t^2 u = c^2 \Delta u \tag{6.5}$$

check whether this is right..

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.

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]

Part III

Exciters

Exciters

Now that a plethora of resonators have been introduced in part II, different mechanisms to excite them will be introduced here. First, different examples of

and have a great effect on the eventual timbre of the sound.

- Simple pluck ((half) raised-cos)
- Bow Models
- Lip reed

Chapter 7

Unmodelled excitations

7.1 Impulse

The simplest way of exciting a system is to add an impulse to

7.1.1 The raised cosine

7.1.2 Pluck

- Cut-off raised cosine
- Triangle (for string)

7.2 Signals

7.2.1 Pulse train

For brass

7.2.2 Noise

Noise input

Chapter 8

The Bow

The bow is a very interesting excitation mechanism from a simulation perspective. The bow excites the resonator at hand with a force due to friction, which introduces a nonlinear element that is dependent on the relative velocity between the bow and the string. The string 'sticks' to the bow after which it 'slips' when the restoring force of the string is too great and overcomes the friction force. This 'stick-slip' behaviour, first coined by Bowden and Leben in 1939 [57] causes the string to move in a characteristic triangular motion where the corner of the triangle moves back and forth along the string (see Figure 8.1). Herman Helmholtz was the first to discover this behaviour, which later got named *Helmholtz motion* in his honour [58].[1]

The Helmholtz motion gives bowed string instruments, such as the violin and cello, their characteristic sound. helmholtz output figure

Brief history of bowed-string simulation

The first nonlinear systems in the context of musical instrument simulations, including the bowed string, were presented by McIntyre, et al. in 1983 [12]. The first real-time implementation of the bowed string was due to Smith in 1986 and used digital waveguides for the string and a look-up table for the friction model [59]. Simultaneously, Florens et al. published a real-time implementation of the bowed string, but instead, the string was modelled using mass-spring systems and the friction model used a static friction model [60] (see Section 8.3). One of the most complex friction models applied in a musical context to-date is the elasto-plastic friction model due to Dupont [61], which Serafin et al. [62, 63] applied to a digital waveguide implementation of the string.

The first ← check with stefania appearance of FDTD methods in bowed string simulations was by Maestre et al. in [26] where these methods were

[1] Also see `https://www.youtube.com/watch?v=6JeyiM0YNo4`

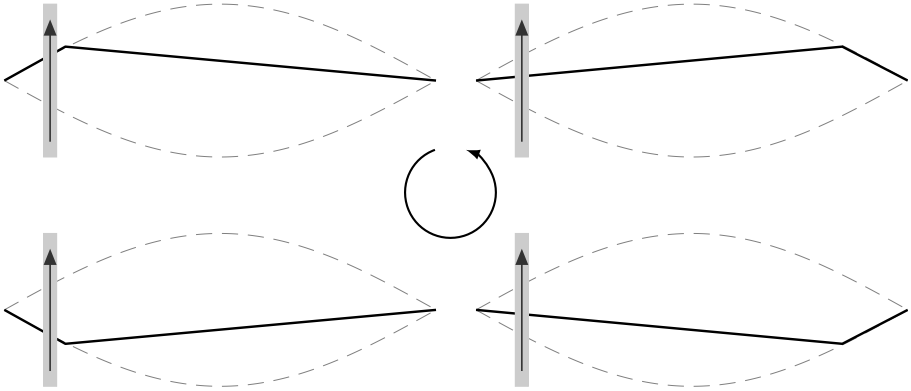


Fig. 8.1: Helmholtz motion. If the bow moves up on the left side of the string, the 'Helmholtz corner' travels anti-clockwise.

used to implement a thermal friction model proposed by Woodhouse in [64]. The string, however, was implemented using digital waveguides. Desvages implemented a bowed string model using a static friction model and a two-polarisation FDTD model for the string in [44, 43], but did not implement this in real-time. This chapter presents the first contribution of this dissertation published in papers [A] and [C]. Firstly, paper [A] shows the first real-time implementation of a bowed stiff string fully modelled using FDTD methods. Secondly, [C] presents the first (real-time) implementation of the elasto-plastic friction model applied to a FD scheme in a musical context.

Before introducing various friction models, a brief introduction on interpolation and spreading operators will be given, which will be necessary to work with this type of excitation.

8.1 Interpolation and Spreading operators

As FD schemes are an approximation to continuous space (and time) using a finite amount of points, working with the system between grid points is not impossible, but requires an extra step. If one would like to listen to a location between specified grid points, one can use interpolation. For this end, an *interpolation operator* $I(x_\mathrm{i})$ can be introduced and can be applied to a grid function. This operator is a function of x_i, the (continuous) location of interest and can be defined in various levels of accuracy. Heere, a 1D system $u(x,t)$ is assumed where $x \in \mathcal{D}$ for spatial domain $\mathcal{D}$.

The simplest interpolation operator is of '0^{th}'-order and is defined as

$$I_0(x_\text{i}) = \begin{cases} 1, & \text{if } l = l_\text{i}, \\ 0, & \text{otherwise}, \end{cases} \tag{8.1}$$

where the grid location of interest is defined as $l_\text{i} = \lfloor x_\text{i}/h \rfloor$. See Figure 8.2a. Instead of actually performing an interpolation operation, I_0 simply floors its input to the grid location below. As an example, for a 1D system of length $L = 1$ described by $u(x,t)$, the location of interest is at $x_\text{i} = 0.25$. If the grid spacing $h = 0.2$ the grid point of interest will be $l_\text{i} = \lfloor 0.25/0.2 \rfloor = \lfloor 1.25 \rfloor = 1$. Writing this in operator form yields

$$u(0.25, t) \approxeq I_0(0.25) u_l^n = u_1^n.$$

A slightly more accurate way to perform 0^{th}-order interpolation is to *round* x_i/h to the nearest neighbour, rather than using the flooring operation.

First-order or linear interpolation also uses the fractional part of the flooring operation according to $\alpha_\text{i} = x_\text{i}/h - l_\text{i}$ and is defined as

$$I_1(x_\text{i}) = \begin{cases} (1 - \alpha_\text{i}), & \text{if } l = l_\text{i}, \\ \alpha_\text{i}, & \text{if } l = l_\text{i} + 1, \\ 0 & \text{otherwise}. \end{cases} \tag{8.2}$$

See Figure 8.2b. When applied to a grid function, all multiplications get added to obtain the interpolated value. Using the same example as before, $\alpha = 1.25 - \lfloor 1.25 \rfloor = 0.25$ and

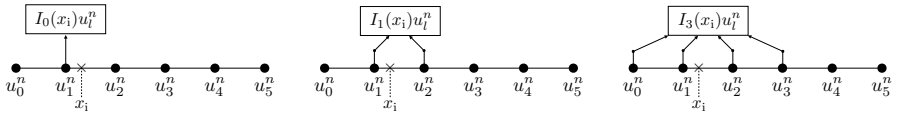
$$u(0.25, t) \approxeq I_1(0.25) u_l^n = 0.75 u_1^n + 0.25 u_2^n,$$

The highest order interpolator used in this project is the Lagrange cubic interpolator:

$$I_3(x_\text{i}) = \begin{cases} -\alpha_\text{i}(\alpha_\text{i} - 1)(\alpha_\text{i} - 2)/6, & l = l_\text{i} - 1, \\ (\alpha_\text{i} - 1)(\alpha_\text{i} + 1)(\alpha_\text{i} - 2)/2, & l = l_\text{i}, \\ -\alpha_\text{i}(\alpha_\text{i} + 1)(\alpha_\text{i} - 2)/2, & l = l_\text{i} + 1, \\ \alpha_\text{i}(\alpha_\text{i} + 1)(\alpha_\text{i} - 1)/6, & l = l_\text{i} + 2, \\ 0, & \text{otherwise}. \end{cases} \tag{8.3}$$

See Figure 8.2c. One could potentially create higher-order interpolation operators, but as one is restricted to a finite domain, the flexibility of the implementation will become less. Notice that if $\alpha_\text{i} = 0$, the higher-order interpolators reduce to the 0^{th}-order one in Eq. (8.1).

Apart form interpolation operators, one may define *spreading operators* which can be interpreted as an inverse interpolation operation. A spreading operator $J(x_\text{i})$ is used to interact with a distributed FD scheme in the form



(a) 0^{th}-order interpolation. **(b)** Linear interpolation. **(c)** Cubic interpolation.

Fig. 8.2: Interpolation with varying orders of accuracy.

of an excitation or other interactions such as connections or collisions between multiple schemes (see Part IV).

The spreading operators can be defined in the same way as the interpolation operators described above, yielding a 0^{th}-order spreading operator

$$J_0(x_\mathrm{i}) = \frac{1}{h}\begin{cases}1, & \text{if } l = l_\mathrm{i},\\ 0, & \text{otherwise,}\end{cases} \tag{8.4}$$

a linear spreading operator

$$J_1(x_\mathrm{i}) = \frac{1}{h}\begin{cases}(1-\alpha_\mathrm{i}), & \text{if } l = l_\mathrm{i},\\ \alpha_\mathrm{i}, & \text{if } l = l_\mathrm{i}+1,\\ 0 & \text{otherwise,}\end{cases} \tag{8.5}$$

and a Lagrange cubic spreading operator

$$J_3(x_\mathrm{i}) = \frac{1}{h}\begin{cases}-\alpha_\mathrm{i}(\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}-2)/6, & l = l_\mathrm{i}-1,\\ (\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i},\\ -\alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i}+1,\\ \alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-1)/6, & l = l_\mathrm{i}+2,\\ 0, & \text{otherwise.}\end{cases} \tag{8.6}$$

Notice the scaling by $1/h$ which will be more elaborated on in Chapter 9. As is the case with the interpolation operators, higher-order spreading operators reduce to Eq. (8.4) if $\alpha_\mathrm{i} = 0$.

The spreading operators, $J(x_\mathrm{i})$ approximate the spatial Dirac delta function $\delta(x - x_\mathrm{i})$, a test function defined as

$$\delta(x) = \begin{cases}\infty, & x = 0,\\ 0, & \text{otherwise,}\end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty}\delta(x)dx = 1, \tag{8.7}$$

used in continuous time to locate an external force to a location x_i along a system distributed over space x. Note that the definition in (8.7) will not be used in practice. Instead, it can be approximated using the spreading operators presented in this section.

The following identity is extremely useful when solving systems including interpolation and spreading operators of the same order o

$$\langle f, J_o(x_\mathrm{i})\rangle_\mathcal{D} = I_o(x_\mathrm{i})f, \tag{8.8}$$

for any (grid) function f. From this, it follows that taking the norm of a spreading operator $J_o(x_\mathrm{i})$ over a given domain is identical to applying to its 'dual' interpolation operator (of the same order o and same input x_i):

$$\langle J_o(x_\mathrm{i}), J_o(x_\mathrm{i})\rangle_\mathcal{D} = \|J_o(x_\mathrm{i})\|^2_\mathcal{D} = I_o(x_\mathrm{i})J_o(x_\mathrm{i}). \tag{8.9}$$

See Section (3.2.1) for more details on the inner product and the norm.

8.2 The Newton-Raphson Method

Before moving on to more complex excitation mechanisms, it is useful to go over the process of how to solve some of these using en iterative root-finding method called the *Newton-Raphson* method (or Newton's method for short).

If the root of function can not be solved explicitly, due to the presence of a nonlinear dependence on a variable for example, the Newton-Raphson method can be used. For a continuous and differentiable function $f(x) = 0$ its root can be approached using the following iteration

different wording here

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}\,, \tag{8.10}$$

with iteration number i and the tick is used to denote a derivation with respect to x. This iteration will then be carried out until the difference between the values of two consecutive iterations is smaller than a given threshold:

$$|x_{i+1} - x_i| < \epsilon, \tag{8.11}$$

where ϵ is small, but its exact value depends on the situation at hand. To prevent Newton's method from iterating endlessly (which can happen in some cases), one can put a cap on the number of iterations allowed.

Preferably, the starting point of the iteration, x_0, should be close to the value of where the root is expected to be. This is especially the case for a higher-ordered function with multiple roots or many local variations.

Algorithm 8.1 shows an example of an implementation of the Newton-Raphson method using $f(x) = e^x - 1 \;\Rightarrow\; f'(x) = e^x$.

```
% An example of the Newton Raphson method using f(x) = exp(x) - 1

x = 1;          % starting point
eps = 1e-4;     % threshold

```

```
% if the threshold has not been crossed before this amount of
    iterations, do not iterate more
maxIterations = 100;

% loop until a maximum number of iterations
for i = 1:maxIterations

    % calculate next iteration (Eq. (8.10)
    xNext = x - (exp(x) - 1) / (exp(x));

    % threshold check (Eq. (8.11)
    if abs(xNext - x) < eps
        break; % break out of the for loop
    end

    % update the value of x
    x = xNext;
end
disp("The root of f(x) is at x = " + xNext)
```

Algorithm 8.1: Example of an implementation of the Newton-Raphson method using $f(x) = e^x - 1$.

8.2.1 Multivariate Newton-Raphson

For M functions f_m dependent on M independent variables x_m with $m = \{1, \ldots, M\}$, the Newton-Raphson method can be extended to the following citation?:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \underbrace{\begin{bmatrix} \frac{\partial f_1(\mathbf{x}_i)}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x}_i)}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x}_i)}{\partial x_1} & \cdots & \frac{\partial f_M(\mathbf{x}_i)}{\partial x_M} \end{bmatrix}^{-1}}_{\mathbf{J}(\mathbf{x})} \begin{bmatrix} f_1(\mathbf{x}_i) \\ \vdots \\ f_M(\mathbf{x}_i) \end{bmatrix}, \tag{8.12}$$

where the independent variables are collected in a column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix}, \tag{8.13}$$

and the iteration number is again denoted by i. The matrix in Eq. (8.12) is referred to as the *Jacobian matrix* $\mathbf{J}$ and contains the derivatives of all functions with respect to each individual independent variable.

As an example, consider the following system of equations[2]:

$$f_1(\mathbf{x}) = 3x_1 - \cos(x_2 x_3) - 3/2 = 0, \tag{8.14a}$$
$$f_2(\mathbf{x}) = 4x_1^2 - 625x_2^2 + 2x_3 - 1 = 0, \tag{8.14b}$$
$$f_3(\mathbf{x}) = 20x_3 + e^{-x_1 x_2} + 9 = 0. \tag{8.14c}$$

The Jacobian matrix will be

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 8x_1 & -1250x_2 & 2 \\ -x_2 e^{-x_1 x_2} & -x_1 e^{-x_1 x_2} & 20 \end{bmatrix},$$

and its roots can be found by iteratively calculating Eq. (8.12) .

8.3 Static Friction Models

A friction model is a nonlinear function that is (at least) dependent on the relative velocity v_{rel} between the bow and the string. This function scales how much the bow force affects the bowed object. In static friction models, the friction force is defined as a function of this relative velocity only. The first mathematical description of friction was proposed by Coulomb in 1773 [65]to which static friction, or *stiction*, was added by Morin in 1833 [66] and viscous friction, or velocity-dependent friction, by Reynolds in 1886 [67]. In 1902, Stribeck found a smooth transition between the static and the coulomb part of the friction curve now referred to as the Stribeck effect [68]. The latter is still the standard for static friction models today.

check references here

In this project, only the following static friction model has been used [3]

$$\Phi(v_{\text{rel}}) = \sqrt{2a} v_{\text{rel}} e^{-a v_{\text{rel}}^2 + 1/2}. \tag{8.15}$$

See Figure 8.3. Consider a string, its transverse displacement described by $u(x,t)$ defined for $x \in \mathcal{D}$ (see Chapter 4). The relative velocity between the string at bow position $x_{\text{B}} = x_{\text{B}}(t) \in \mathcal{D}$ and the bow is described as

$$v_{\text{rel}} = \partial_t u(x_{\text{B}}, t) - v_{\text{B}}(t) \tag{8.16}$$

(in m/s) with bow velocity $v_{\text{B}} = v_{\text{B}}(t)$ (in m/s).

FULL DOC SWEEP: check figure centering

Many of the friction models contain a discontinuity where the relative velocity $v_{\text{rel}} = 0$ due to a multiplication with $\text{sgn}(v_{\text{rel}})$ in their definition. Equation (8.15) approximates discontinuous bow models using a continuous and differentiable function. This makes it easier to work with when trying to implement the system as will be explained next.

[2]Taken from `http://fourier.eng.hmc.edu/e176/lectures/NM/node21.html`

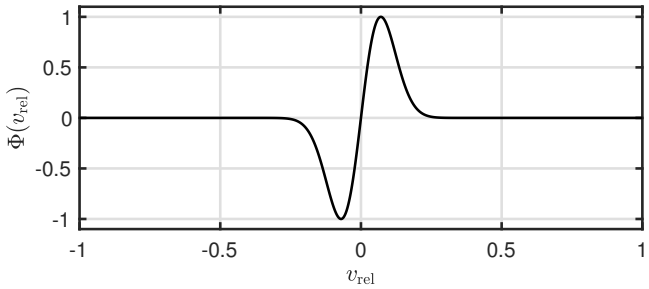


Fig. 8.3: The friction model in (8.15) with $a = 100$.

8.3.1 The bowed stiff string

Recalling the PDE of the stiff string in Eq. (4.4)

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \tag{8.17}$$

one can add the bow force to the equation according to

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u - \delta(x - x_{\text{B}}) f_{\text{B}} \Phi(v_{\text{rel}}) \tag{8.18}$$

where spatial Dirac delta function $\delta(x - x_{\text{B}})$ (in m^{-1}) (see Section 8.1) positions the bow along the string and $f_{\text{B}} = f_{\text{B}}(t)$ is the bow force (in N).

Intuition

From Eq. (8.18) it can be seen that the bow force gets scaled by the friction model $\Phi(v_{\text{rel}})$ shown in Figure 8.3. The figure shows that if v_{rel} is too large (either positively or negatively) the bow term in (8.18) becomes 0. If, on the other hand, v_{rel} is closer to 0, the bow will have an effect on the string. This can be interpreted in terms of static and dynamic friction[3]. A stationary object requires more force to be moved than a moving object, i.e., the static friction coefficient is always higher than the dynamic friction coefficient. This is essentially what the friction model tries to simulate.

Discrete time

Dividing all terms in Eq. (8.19) by ρA and discretising the system yields

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n + 2\sigma_1 \delta_{t-} \delta_{xx} u_l^n - J(x_{\text{B}}^n) F_{\text{B}}^n \Phi(v_{\text{rel}}^n), \tag{8.19}$$

with $F_{\text{B}}^n = f_{\text{B}}^n / \rho A$ and spreading operator $J(x_{\text{B}})$ (in m^{-1}) as described in Section 8.1, the order of which remains undetermined for now. Notice that, as

[3] 'Static' and 'dynamic' friction are unrelated to 'static' and 'dynamic' friction models.

the bow position, bow velocity and bow force are time-dependent, these have received a superscript n. These parameters are called control parameters and will be supplied by the performer in the eventual implementation.

The relative velocity in Eq. (8.16) is discretised using a centred difference operator according to

$$v_{\text{rel}}^n = I(x_{\text{B}}^n)\delta_{t\cdot}u_l^n - v_{\text{B}}^n. \tag{8.20}$$

The main issue with Eq. (8.20) is that, due to the centred difference operator, the FD scheme is now nonlinearly dependent on u_l^{n+1}. To solve Eq. (8.19), an iterative solver is required, such as the Newton-Raphson method described in Section 8.2. This process could be circumvented by using a backward difference operator in Eq. (8.20), but this will affect accuracy of the bow model.

Solution

To find a solution for u_l^{n+1} at the bow location, an inner product of the scheme in Eq. (8.19) with spreading operator $J(x_{\text{B}}^n)$ must be taken which isolates the scheme at the bowing location. Performing this operation and using identity (8.8) yields

$$\begin{aligned} I(x_{\text{B}}^n)\delta_{tt}u_l^n = c^2 I(x_{\text{B}}^n)\delta_{xx}u_l^n - \kappa^2 I(x_{\text{B}}^n)\delta_{xxxx}u_l^n - 2\sigma_0 I(x_{\text{B}}^n)\delta_{t\cdot}u_l^n \\ + 2\sigma_1 I(x_{\text{B}}^n)\delta_{t-}\delta_{xx}u_l^n - \|J(x_{\text{B}}^n)\|_{\mathcal{D}}^2 F_{\text{B}}^n \Phi(v_{\text{rel}}^n). \end{aligned} \tag{8.21}$$

One can rewrite Eq. (8.20) to

$$I(x_{\text{B}}^n)\delta_{t\cdot}u_l^n = v_{\text{rel}}^n + v_{\text{B}}^n, \tag{8.22}$$

and using identity (2.27a), Eq. (8.21) can be rewritten and assigned to a function $g(v_{\text{rel}}^n)$

$$g(v_{\text{rel}}^n) = \left(\frac{2}{k} + 2\sigma_0\right) v_{\text{rel}}^n + \|J(x_{\text{B}}^n)\|_{\mathcal{D}}^2 F_{\text{B}}^n \Phi(v_{\text{rel}}^n) + b^n = 0, \tag{8.23}$$

where the terms not dependent on v_{rel} are collected in

$$\begin{aligned} b^n = &-\frac{2}{k} I(x_{\text{B}}^n)\delta_{t-}u_l^n - c^2 I(x_{\text{B}}^n)\delta_{xx}u_l^n + \kappa^2 I(x_{\text{B}}^n)\delta_{xxxx}u_l^n \\ &+ \left(\frac{2}{k} + 2\sigma_0\right) v_{\text{B}}^n - 2\sigma_1 I(x_{\text{B}}^n)\delta_{t-}\delta_{xx}u_l^n. \end{aligned}$$

One can then perform the Newton-Raphson method detailed in Section 8.2 to iteratively solve for v_{rel}

$$(v_{\text{rel}}^n)_{i+1} = (v_{\text{rel}}^n)_i - \frac{g((v_{\text{rel}}^n)_i)}{g'((v_{\text{rel}}^n)_i)}\,, \tag{8.24}$$

where

$$g'(v_{\text{rel}}^n) = \frac{2}{k} + 2\sigma_0 + \|J(x_{\text{B}}^n)\|_{\mathcal{D}}^2 F_{\text{B}}^n \Phi'(v_{\text{rel}}^n), \tag{8.25}$$

and

$$\Phi'(v_{\text{rel}}^n) = \sqrt{2a}\left(1 - 2a(v_{\text{rel}}^n)^2\right) e^{-a(v_{\text{rel}}^n)^2 + 1/2}.$$

Implementation

still need to write this bit The Newton-Raphson iteration needs to be performed for every sample

$\epsilon = 10^{-4}$
Limit of iterations is set to 100

$$I(x_{\mathrm{B}}^n)\delta_{xx}u_l^n = (1-\alpha)\delta_{xx}u_{l_{\mathrm{B}}}^n + \alpha\delta_{xx}u_{l_{\mathrm{B}}+1}^n \tag{8.26}$$

8.4 Dynamic Friction Models

As opposed to less complex static friction models, dynamic friction models relate the relative velocity to the friction force using a differential equation. Dynamic friction models exhibit a phenomenon called *hysteresis*, which is the dependence of a system on its history.

The first dynamic friction model was due to Dahl [69] and captured hysteresis effects. The Stribeck effect was, however, not taken into account. The LuGre model (named after the collaboration between Lund and Grenoble) was then proposed by Canudas de Wit et al. in [70, 71] and extended the Dahl model to take the Stribeck effect into account. The model assumes a large ensemble of bristles between the two sliding surfaces, each of which contributes a tiny amount to the total friction force. The drawback of this model is that it exhibits drift for extremely small external forces. In [61], Dupont et al. extended the LuGre model by allowing for a purely elastic regime that solves the drift issue. This model is referred to as the *elasto-plastic* friction model and is used in this project.

8.4.1 The Elasto-Plastic friction model

In a musical context, the elasto-plastic friction has been thoroughly investigated by Serafin et al. in [62, 63, 72]. Like the LuGre model, the elasto-plastic friction model assumes that the friction between the bow and the string is caused by a large quantity of bristles, each of which contributes to the total amount of friction. See Figure 8.4.

Unless denoted otherwise, this section follows the original model by Dupont et al. in [61], but with the appropriate corrections added as presented in paper [C]. As opposed to the static friction model described in the previous section, the friction force f (in N) is now dependent on the average bristle displacement $z = z(t)$ (in m) on top of the relative velocity $v = v(t)$ (in m/s). The force is defined as

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{8.27}$$

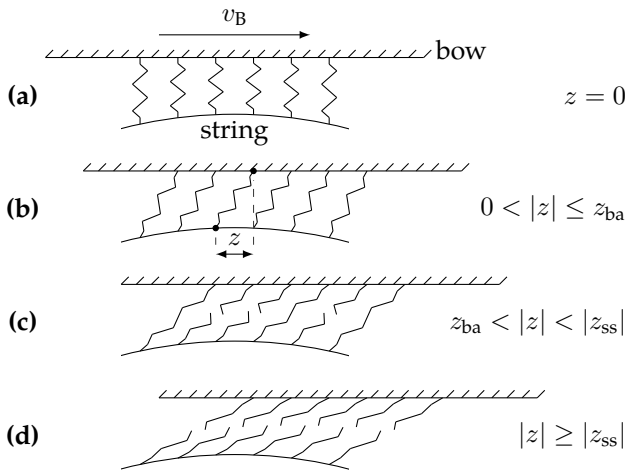


Fig. 8.4: Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B. (a) The initial state is where the average bristle displacement $z = 0$. (b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (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. (Adapted from paper [C].)exactly the same caption as paper

with bristle stiffness s_0 (in N/m), bristle damping s_1 (in kg/s), viscous friction s_2 (in kg/s) and, as presented in [63], a dimensionless noise coefficient s_3 multiplied onto a pseudorandom function $w = w(t)$ (in N) generating values between -1 and 1. Moreover, the relative velocity between the string at bowing location $x_B = x_B(t)$ and the bow is (similar to Eq. (8.16))

$$v = \partial_t u(x_B, t) - v_B, \tag{8.28}$$

with bow velocity $v_B = v_B(t)$ (in m/s). Lastly, $\dot{z}$ is the rate of change of the bristle displacement (in m/s) and is related to v according to

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_{ss}(v)}\right]. \tag{8.29}$$

Here, z_{ss} is the steady-state function

$$z_{ss}(v) = \frac{\operatorname{sgn}(v)}{s_0}\left[f_C + (f_S - f_C)e^{-(v/v_S)^2}\right], \tag{8.30}$$

where the v_S is the Stribeck velocity (in m/s). Furthermore, $f_C = f_N\mu_C$ and $f_S = f_N\mu_S$ are the Coulomb force and stiction force respectively (both in N). In these definitions μ_C and μ_S are the dimensionless dynamic and static friction

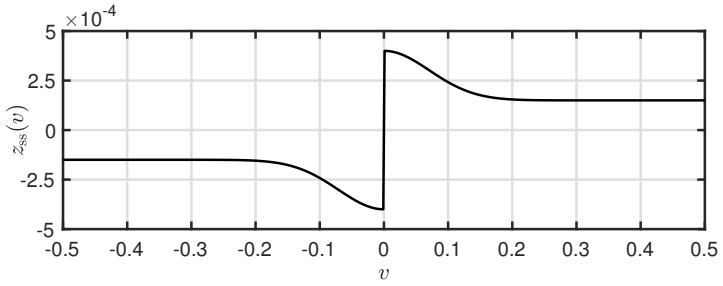


Fig. 8.5: A plot of the steady-state function $z_{ss}(v)$ with $s_0 = 10^4$, $\mu_C = 0.3$, $\mu_S = 0.8$, $v_S = 0.1$ and $f_N = 5$.

coefficients respectively and $f_N = f_N(t)$ is the normal force (in N). A plot of the steady state function can be found in Figure 8.5.

Finally, $\alpha(v, z)$ in Eq. (8.29) is an adhesion map between the bow and the string and is defined as

$$\alpha(v,z) = \begin{cases} \left.\begin{array}{ll} 0 & |z| \le z_{ba} \\ \alpha_m(v,z) & z_{ba} < |z| < |z_{ss}(v)| \\ 1 & |z| \ge |z_{ss}(v)| \end{array}\right\} & \text{if } \operatorname{sgn}(v) = \operatorname{sgn}(z) \\ 0 & \text{if } \operatorname{sgn}(v) \neq \operatorname{sgn}(z), \end{cases} \tag{8.31}$$

where the transition between the elastic and plastic behaviour is defined as

$$\alpha_m = \frac{1}{2}\left[1 + \operatorname{sgn}(z)\sin\left(\pi \frac{z - \operatorname{sgn}(z)\frac{1}{2}(|z_{ss}(v)| + z_{ba})}{|z_{ss}(v)| - z_{ba}}\right)\right], \tag{8.32}$$

so is $z_{ba} = z_{ba}(t)$

with break-away displacement $z_{ba} = 0.7 f_C / s_0$ determines the value of z before bristles start to break. The adhesion map is visualised in Figure 8.6 and relates to Figure 8.4 as described in its caption.

Discrete time

Equation (8.27) can be discretised to

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n \tag{8.33}$$

where discrete relative velocity in Eq. (8.28)

$$v^n = I(x_B^n)\delta_t . u_l^n - v_B^n, \tag{8.34}$$

and

$$r^n = r(v^n, z^n) = v^n\left[1 - \alpha(v^n, z^n)\frac{z^n}{z_{ss}(v^n)}\right] \tag{8.35}$$

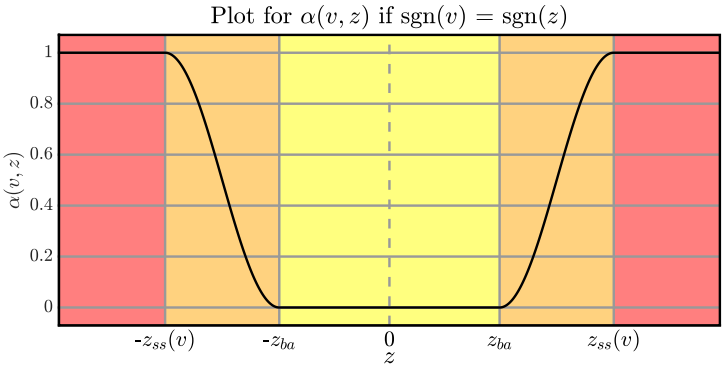


Fig. 8.6: A plot of the adhesion map $\alpha(v,z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Figure 8.4 according to: yellow - a) & b), orange - c) and red - d). (Adapted from paper [C])

is the discrete counterpart of (8.29). The discrete adhesion map is identical to the continuous definition given in Eqs. (8.31) and (8.32), but with superscripts n added for appearances of v and z.

8.4.2 Applied to a FDTD stiff string

The first appearance contribution of the PhD project in this dissertation

In the same way as done with the static friction model in Section 8.3, one can add the friction force to the stiff string PDE in Eq. (4.4) and discretise the system as follows:

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n + 2\sigma_1 \delta_{t-} \delta_{xx} u_l^n - J(x_{\mathrm{B}}^n) \frac{f(v^n, z^n)}{\rho A}. \quad (8.36)$$

Following the same procedure as for the static friction model in Section 8.3, one takes an inner product with $J(x_{\mathrm{B}}^n)$ and using identities (8.8) and (2.27a), one can rewrite this similarly to the static friction model in Eq. (8.23) as

$$g_1(v^n, z^n) = \left(\frac{2}{k} + 2\sigma_0 \right) v^n + \| J(x_{\mathrm{B}}^n) \|_{\mathcal{D}}^2 \frac{f(v^n, z^n)}{\rho A} + b^n = 0, \quad (8.37)$$

where

$$b^n = -\frac{2}{k} I(x_{\mathrm{B}}^n) \delta_{t-} u_l^n - c^2 I(x_{\mathrm{B}}^n) \delta_{xx} u_l^n + \kappa^2 I(x_{\mathrm{B}}^n) \delta_{xxxx} u_l^n + \left(\frac{2}{k} + 2\sigma_0 \right) v_{\mathrm{B}}^n - 2\sigma_1 I(x_{\mathrm{B}}^n) \delta_{t-} \delta_{xx} u_l^n.$$

As g_1 contains two unknown variables v^n and z^n that need to be solved for, the multivariate Newton-Raphson method presented in 8.2.1 must be performed. To be able to do this, an extra function must be included.

As r describes $\dot{z}$ in Eq. (8.29), one can take another approach to approximate $\dot{z}$ using the trapezoid rule [3]

$$a^n = (\mu_{t-})^{-1}\delta_{t-}z^n \quad \Longrightarrow \quad a^n = \frac{2}{k}(z^n - z^{n-1}) + a^{n-1}. \tag{8.38}$$

As both a^n and r^n approximate $\dot{z}$ these can be used to create the second function necessary to solve the full system

$$g_2(v^n, z^n) = r^n - a^n = 0. \tag{8.39}$$

Performing the multivariate Newton-Raphson method described in yields

$$\begin{bmatrix} v^n \\ z^n \end{bmatrix}_{i+1} = \begin{bmatrix} v^n \\ z^n \end{bmatrix}_{i} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{8.40}$$

The derivatives can be shown to be... should I give those here..? Could have an appendix with derivations haha

8.4.3 Implementation and output

still need to write

8.4.4 Stability through Energy Analysis

As the elasto-plastic bow model is a differential equation in itself, its approximation will need to abide a stability condition as well. As the system at hand is nonlinear, frequency domain analysis as described in Section 3.3 can not be performed. Energy analysis, on the other hand, can be used here to determine the necessary stability condition for this model. This section follows the concepts introduced in Section 3.4.4 to obtain a stability conditions for the elasto-plastic friction model. A similar process for finding stability for the LuGre model has been done by Olsson in [73, p. 55] and the derivation below is inspired by his.

First, all terms of Eq. (8.36) are multiplied by ρA to get the appropriate units for the analysis. Then, the inner product with $\delta_{t\cdot}u_l^n$ over the string domain $\mathcal{D}$ is taken to get

$$\delta_{t+}\mathfrak{h}_\mathrm{s} + \mathfrak{q}_\mathrm{s} = -\mathfrak{p}_\mathrm{B} \tag{8.41}$$

where the definitions for the discrete Hamiltonian $\mathfrak{h}_\mathrm{s}$ and the damping term $\mathfrak{q}_\mathrm{s}$ for the string can be found in Section 4.4. The input power introduced by the bow is defined as (writing $f(v^n, z^n) = f^n$)

$$\mathfrak{p}_\mathrm{B} = \langle (\delta_{t\cdot}u_l^n), J(x_\mathrm{B}^n)f^n \rangle_\mathcal{D}$$

which, using identity (8.8) can be written as

$$\mathfrak{p}_\mathrm{B} = I(x_\mathrm{B}^n)\delta_{t\cdot}u_l^n f^n.$$

Finally, using Eq. (8.34) yields

$$\mathfrak{p}_{\mathrm{B}} = f^n v^n + f^n v_{\mathrm{B}}^n. \tag{8.42}$$

The term $f^n v^n$ is the important one as $f^n v_{\mathrm{B}}^n$ is a driving term, and is zero when the external bow velocity is zero. This means that this does not affect the internal stability of the system. In the following, the superscript n is suppressed for clarity.

Substituting Eq. (8.33) into fv and ignoring the noise term $s_3 w^n$ for now, yields

$$\mathfrak{p}_{\mathrm{B}} = fv = \sigma_0 z v + \sigma_1 r v + \sigma_2 v^2. \tag{8.43}$$

The definition for r^n in Eq. (8.35) may be rewritten as

$$\begin{aligned} r &= v\left[1 - \alpha\frac{z}{z_{\mathrm{ss}}(v)}\right], \\ r &= v - \frac{v\alpha z}{z_{\mathrm{ss}}(v)}, \\ v &= r + \frac{v\alpha z}{z_{\mathrm{ss}}(v)}, \end{aligned}$$

and (following Olsson) may be substituted in Eq. (8.43) as

$$\mathfrak{p}_{\mathrm{B}} = s_0 z\left(r + \frac{v\alpha z}{z_{\mathrm{ss}}(v)}\right) + s_1 r\left(r + \frac{v\alpha z}{z_{\mathrm{ss}}(v)}\right) + s_2 v^2, \tag{8.44}$$

or

$$\mathfrak{p}_{\mathrm{B}} = s_0 z r + s_1\left(r + \frac{v\alpha z}{2z_{\mathrm{ss}}(v)}\right)^2 + \frac{v\alpha z^2}{z_{\mathrm{ss}}(v)}\left(s_0 - \frac{s_1 v\alpha}{4z_{\mathrm{ss}}(v)}\right) + s_2 v^2. \tag{8.45}$$

The power introduced by the bow can then be subdivided into the total energy in the bristles and their damping. As r approximates $\dot{z}$ the first term, one can rewrite this to

NEED TO LOOK AT THE BELOW

$$\delta_{t+}\mathfrak{h}_{\mathrm{brist}} = s_0 z^n(\delta_t . z^n) + \mathfrak{q}_{\mathrm{brist}} \geq 0$$

and needs to be non-negative for passivity. Using identity (3.24b)

$$\begin{aligned} \mathfrak{h}_{\mathrm{B}} &= \frac{s_0}{2} z^n e_{t-} z^n, \\ &= \frac{s_0}{2}\left((\mu_{t-} z^n)^2 - \frac{k^2}{4}(\delta_{t-} z^n)^2\right) \end{aligned}$$

and

$$\mathfrak{q}_{\mathrm{brist}} = s_1\left(r + \frac{v\alpha z}{2z_{\mathrm{ss}}(v)}\right)^2 + \frac{v\alpha z^2}{z_{\mathrm{ss}}(v)}\left(s_0 - \frac{s_1 v\alpha}{4z_{\mathrm{ss}}(v)}\right) + s_2 v^2. \tag{8.46}$$

Finally, for the system to be passive, $\mathfrak{q}$ must be non-negative. As all coefficients are non-negative and $\text{sgn}(v) = \text{sgn}(z_{\text{ss}}(v))$ (through a multiplication by $\text{sgn}(v)$ in the definition of in Eq. (8.30)) the following must hold

$$s_0 - \frac{s_1 v \alpha}{4 z_{\text{ss}}(v)} \geq 0,$$

where both terms are non-negative. Finally, as α is bounded by 1 the following condition must hold for stability

$$s_1 \leq \frac{4 s_0 z_{\text{ss}}(v)}{v} \,. \tag{8.47}$$

This is the same stability condition as Olsson presents in [73]. So as long as one knows the limit of the velocity of the system, the coefficient s_1 can be set accordingly.

8.5 Lip-reed

Lip-reed model

Coupling to Tube

To excite the system we can use a pulse train. A more physical approach, that is bidirectional, is to model a lip as a mass-spring system that interacts with the left boundary of the tube (see Figure 8.7). Following [53] we get

$$M_{\text{r}} \frac{d^2 y}{dt^2} = -M_{\text{r}} \omega_0^2 y - M_{\text{r}} \sigma_{\text{r}} \frac{dy}{dt} + S_{\text{r}} \Delta p, \tag{8.48}$$

with displacement of the lip reed from equilibrium $y = y(t)$, mass of the lip reed M_{r} (kg) natural angular frequency of the lip reed $\omega_0 = \sqrt{K/M_{\text{r}}}$ (rad/s), spring stiffness of the lip K (N/m), loss parameter σ_{r} (s^{-1}), effective surface area of the lip S_{r} (m^2) and

$$\Delta p = P_{\text{m}} - p(0, t) \tag{8.49}$$

is the difference between the pressure in the mouth P_{m} (kPa) and the pressure in the mouth piece $p(0, t)$ (kPa). This pressure difference causes a volume flow velocity following the Bernoulli equation

$$U_{\text{B}} = w[y + H_0]_+ \text{sgn}(\Delta p) \sqrt{\frac{2|\Delta p|}{\rho_0}}, \tag{8.50}$$

with effective lip-reed width w (m), static equilibrium separation H_0 (m) and $[x]_+ = 0.5(x + |x|)$ describes the "positive part of". Notice that when $y + H_0 \leq 0$,

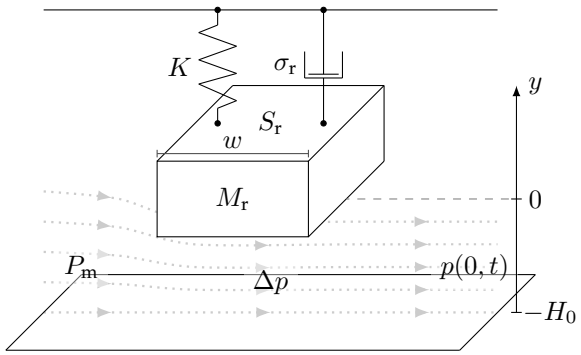


Fig. 8.7: Lipsystem with the equilibrium at 0 and the distance from the lower lip H_0.

the lips are closed and the volume velocity U_B is 0. Another volume flow is generated by the lip reed itself according to

$$U_r = S_r \frac{dy}{dt}. \tag{8.51}$$

Assuming that the volume flow velocity is conserved we define the total air volume entering the system as

$$S(0)v(0,t) = U_B(t) + U_r(t). \tag{8.52}$$

8.5.1 Discrete Time

Placing y, Δp, and thereby U_B and U_r on the interleaved temporal grid (but on the non-interleaved spatial grid), we discretise the equations above to get the following system

$$\begin{cases} M_r \delta_{tt} y^{n+1/2} & = -M_r \omega_0^2 \mu_{t\cdot} y^{n+1/2} - M_r \sigma_r \delta_{t\cdot} y^{n+1/2} + S_r \Delta p^{n+1/2} & \text{(8.53a)} \\ \Delta p^{n+1/2} & = P_m - \mu_{t+} p_0^n & \text{(8.53b)} \\ U_B^{n+1/2} & = w[y^{n+1/2} + H_0]_+ \operatorname{sgn}(\Delta p^{n+1/2}) \sqrt{\frac{2|\Delta p^{n+1/2}|}{\rho_0}} & \text{(8.53c)} \\ U_r^{n+1/2} & = S_r \delta_{t\cdot} y^{n+1/2} & \text{(8.53d)} \\ \mu_{x-}(S_{1/2} v_{1/2}^{n+1/2}) & = U_B^{n+1/2} + U_r^{n+1/2} & \text{(8.53e)} \end{cases}$$

In the following we will suppress the superscript $n+1/2$ for the aforementioned variables. Expanding and solving (8.53a) for $y^{n+3/2}$ yields

$$\left(1 + \frac{\omega_0^2 k^2}{2} + \frac{\sigma_r k}{2}\right) y^{n+3/2} = 2y^{n+1/2} - \left(1 + \frac{\omega_0^2 k^2}{2} - \frac{\sigma_r k}{2}\right) y^{n-1/2} + \frac{S_r k^2}{M_r} \Delta p$$

$$\alpha_r y^{n+3/2} = 4y^{n+1/2} + \beta_r y^{n-1/2} + \xi_r \Delta p \tag{8.54}$$

where

$$\alpha_{\mathrm{r}} = 2+\omega_0^2 k^2+\sigma_{\mathrm{r}} k\,, \qquad \beta_{\mathrm{r}} = \sigma_{\mathrm{r}} k-2-\omega_0^2 k^2\,, \qquad \text{and} \qquad \xi_{\mathrm{r}} = \frac{2S_{\mathrm{r}}k^2}{M_{\mathrm{r}}}. \tag{8.55}$$

Obtaining Δp

With all other parameters user-defined, the only unknown in our system is now Δp. Using the following identities

$$\delta_{tt} = \frac{2}{k}(\delta_{t\cdot} - \delta_{t-}), \quad \text{and} \quad \mu_{t\cdot} = k\delta_{t\cdot} + e_{t-} \tag{8.56}$$

where e_{t-} is a backwards time-shift of one sample (so $e_{t-}y^{n+1/2} = y^{n-1/2}$) we can rewrite (8.53a) to

$$\begin{aligned}\frac{2}{k}(\delta_{t\cdot} - \delta_{t-})y = -\omega_0^2(k\delta_{t\cdot} + e_{t-})y - \sigma_{\mathrm{r}}\delta_{t\cdot}y + \frac{S_{\mathrm{r}}}{M_{\mathrm{r}}}\Delta p \\ a_1\delta_{t\cdot}y - a_2\Delta p - a_3^n = 0,\end{aligned} \tag{8.57}$$

where

$$a_1 = \frac{2}{k} + \omega_0^2 k + \sigma_{\mathrm{r}} \geq 0, \quad a_2 = \frac{S_{\mathrm{r}}}{M_{\mathrm{r}}} \geq 0\,, \quad \text{and} \quad a_3^n = \left(\frac{2}{k}\delta_{t\cdot} - \omega_0^2 e_{t-}\right) y\,. \tag{8.58}$$

Note that because a_1 and a_2 are calculated solely from non-negative parameters we can apply the condition that these are greater than or equal to 0. The same will be done for other coefficients below. We can substitute Eq. (8.53d) into Eq. (8.89)

$$\frac{a_1}{S_{\mathrm{r}}}U_{\mathrm{r}} - a_2\Delta p - a_3^n = 0 \tag{8.59}$$

and consequently (8.53e) to get

$$\frac{a_1}{S_{\mathrm{r}}}\left(\mu_{x-}(S_{1/2}v_{1/2}) - U_{\mathrm{B}}\right) - a_2\Delta p - a_3^n = 0 \tag{8.60}$$

To get a definition for $\mu_{x-}(S_{1/2}v_{1/2})$, we include the expression for Webster's equation at $l = 0$

$$\frac{\bar{S}_0}{\rho_0 c^2}\delta_{t+}p_0^n = -\delta_{x-}(S_{1/2}v_{1/2}), \tag{8.61}$$

which, using the following identity (derived from Eq. (2.7d) from [**?**])

$$\delta_{x\pm} = \pm\frac{2}{h}(\mu_{x\pm} - 1), \tag{8.62}$$

can be rewritten to

$$\frac{\bar{S}_0}{\rho_0 c^2}\delta_{t+}p_0^n = \frac{2}{h}\left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2}\right). \tag{8.63}$$

Then using the same identity (8.62) but for δ_{t+} we get

$$\frac{2\bar{S}_0}{\rho_0 c^2 k}(\mu_{t+}p_0^n - p_0^n) = \frac{2}{h}\left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2}\right). \tag{8.64}$$

Using Eq. (8.53b) we can rewrite this to

$$\begin{aligned}\frac{\bar{S}_0 h}{\rho_0 c^2 k}(P_\mathrm{m} - \Delta p - p_0^n) &= \frac{2}{h}\left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2}\right).\\ \mu_{x-}(S_{1/2}v_{1/2}) &= b_1^n - b_2\Delta p\end{aligned} \tag{8.65}$$

where

$$b_1^n = S_{1/2}v_{1/2} + \frac{\bar{S}_0 h}{\rho_0 c^2 k}(P_\mathrm{m} - p_0^n), \quad \text{and} \quad b_2 = \frac{\bar{S}_0 h}{\rho_0 c^2 k} \geq 0\,. \tag{8.66}$$

We can then substitute Eqs. (8.65) and (8.53c) into Eq. (8.60) to get

$$\begin{aligned}\frac{a_1}{S_\mathrm{r}}\left(b_1^n - b_2\Delta p - w[y+H_0]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}}\right) - a_2\Delta p - a_3^n = 0,\\ -w[y+H_0]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}} - b_2\Delta p - \frac{a_2 S_\mathrm{r}}{a_1}\Delta p + b_1^n - \frac{a_3^n S_\mathrm{r}}{a_1} = 0,\\ -c_1^n\mathrm{sgn}(\Delta p)\sqrt{|\Delta p|} - c_2\Delta p + c_3^n = 0\end{aligned} \tag{8.67}$$

where

$$c_1^n = w[y+H_0]_+\sqrt{\frac{2}{\rho_0}} \geq 0, \quad c_2 = b_2 + \frac{a_2 S_\mathrm{r}}{a_1} \geq 0, \quad \text{and} \quad c_3^n = b_1^n - \frac{a_3^n S_\mathrm{r}}{a_1}\,. \tag{8.68}$$

We can then divide Eq. (8.67) by $-\mathrm{sgn}(\Delta p)$ to get a quadratic equation in $\sqrt{|\Delta p|}$

$$c_2|\Delta p| + c_1^n\sqrt{|\Delta p|} - \frac{c_3^n}{\mathrm{sgn}(\Delta p)} = 0. \tag{8.69}$$

Now, as we know that $c_1^n, c_2 \geq 0$, for any real solutions to exist the following must be true

$$\mathrm{sgn}(c_3^n) = \mathrm{sgn}(\Delta p) \quad \Longrightarrow \quad \frac{c_3^n}{\mathrm{sgn}(\Delta p)} = |c_3^n|. \tag{8.70}$$

Now we can solve for $\sqrt{|\Delta p|}$:

$$\sqrt{|\Delta p|} = \frac{-c_1^n \pm \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2}\,. \tag{8.71}$$

As $\sqrt{(c_1^n)^2 + 4c_2|c_3^n|} \geq c_1^n$, we can only guarantee that the solution is positive if we take the positive solution the square root. Furthermore, using Eq. (8.70) we can solve for the pressure difference

$$\Delta p = \mathrm{sgn}(c_3^n) \left(\frac{-c_1^n + \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2} \right)^2 . \tag{8.72}$$

This we can then apply to the update of the lip reed in Eq. (8.53a).

Coupling to the tube

To couple the reed to the tube, we take Eq. (**??**) at $l = 0$ and rewrite it to

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0} \left(-2\mu_{x-}(S_{1/2} v_{1/2}) + 2S_{1/2} v_{1/2} \right), \tag{8.73}$$

and substitute Eq. (8.53e) to get

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0} \left(-2(U_\mathrm{B} + U_\mathrm{r}) + 2S_{1/2} v_{1/2} \right). \tag{8.74}$$

8.5.2 Energy analysis

We start by multiplying Eq. (8.53a) by $\delta_{t\cdot} y$ (superscript $n + 1/2$ is again suppressed):

$$\overset{\text{Eq. (8.53b)}}{\Longleftrightarrow} \quad M_\mathrm{r} \delta_{t\cdot} y \delta_{tt} y + M_\mathrm{r} \omega_0^2 \delta_{t\cdot} y \mu_{t\cdot} y + M_\mathrm{r} \sigma_\mathrm{r} (\delta_{t\cdot} y)^2 - S_\mathrm{r} \delta_{t\cdot} y \Delta p =$$

$$\overset{\text{Eqs. (8.53d) \& (8.53e)}}{\Longleftrightarrow} \quad M_\mathrm{r} \delta_{t\cdot} y \delta_{tt} y + M_\mathrm{r} \omega_0^2 \delta_{t\cdot} y \mu_{t\cdot} y + M_\mathrm{r} \sigma_\mathrm{r} (\delta_{t\cdot} y)^2 - \left(\mu_{x-}(S_{1/2} v_{1/2}) - U_\mathrm{B} \right) \Delta p =$$

$$\overset{\text{Eq. (8.53b)}}{\Longleftrightarrow} \quad M_\mathrm{r} \delta_{t\cdot} y \delta_{tt} y + M_\mathrm{r} \omega_0^2 \delta_{t\cdot} y \mu_{t\cdot} y + M_\mathrm{r} \sigma_\mathrm{r} (\delta_{t\cdot} y)^2 + U_\mathrm{B} \Delta p - \mu_{x-}(S_{1/2} v_{1/2})(P_\mathrm{m} - \mu_{t+} p_0) =$$

Recalling that the energy of the tube $\delta_{t+} \mathfrak{h}_\mathrm{t} = \mathfrak{b}_\mathrm{r} + \mathfrak{b}_\mathrm{l}$ and $\mathfrak{b}_\mathrm{l} = -(\mu_{t+} p_0)\mu_{x-}(S_{1/2} v_{1/2})$ we get, assuming that $\mathfrak{b}_\mathrm{r} = 0$

$$\overset{\text{Eq. (??)}}{\Longleftrightarrow} \quad M_\mathrm{r} \delta_{t\cdot} y \delta_{tt} y + M_\mathrm{r} \omega_0^2 \delta_{t\cdot} y \mu_{t\cdot} y + M_\mathrm{r} \sigma_\mathrm{r} (\delta_{t\cdot} y)^2 + U_\mathrm{B} \Delta p - \mu_{x-}(S_{1/2} v_{1/2}) P_\mathrm{m} + \delta_{t+} \mathfrak{h}_\mathrm{t} =$$

Then we arrive at the following energy balance

$$\delta_{t+} \left(\mathfrak{h}_\mathrm{t} + \mathfrak{h}_\mathrm{r} \right) + \mathfrak{Q}_\mathrm{r} + \mathfrak{p}_\mathrm{r} = 0 \tag{8.75}$$

where

$$\mathfrak{h}_\mathrm{r} = \frac{M_\mathrm{r}}{2} \left((\delta_{t-} y)^2 + \omega_0^2 \mu_{t-}(y^2) \right) \geq 0 \tag{8.76}$$

$$\mathfrak{Q}_\mathrm{r} = M_\mathrm{r} \sigma_\mathrm{r} (\delta_{t\cdot} y)^2 + U_\mathrm{B} \Delta p \geq 0 \tag{8.77}$$

$$\mathfrak{p}_\mathrm{r} = -(U_\mathrm{B} + U_\mathrm{r}) P_\mathrm{m} \tag{8.78}$$

8.5.3 Adding lip collision

We can use Michele's tricks to add a non-linear collision to the lip. As the collision happens at the interleaved temporal grid, we move the potential half a time step forward. We extend Eq. (8.53a) to be

$$M_{\mathrm{r}}\delta_{tt}y = -M_{\mathrm{r}}\omega_0^2\mu_{t\cdot}y - M_{\mathrm{r}}\sigma_{\mathrm{r}}\delta_{t\cdot}y + S_{\mathrm{r}}\Delta p + \psi^{n+1/2}(\psi^{n+1/2})', \tag{8.79}$$

which, when using $\mu_{t+}\psi^n = \psi^{n+1/2}$, we can then rewrite to

$$M_{\mathrm{r}}\delta_{tt}y = -M_{\mathrm{r}}\omega_0^2\mu_{t\cdot}y - M_{\mathrm{r}}\sigma_{\mathrm{r}}\delta_{t\cdot}y + S_{\mathrm{r}}\Delta p + (\mu_{t+}\psi^n)g^{n+1/2} \tag{8.80}$$

with

$$g^{n+1/2} = \frac{\delta_{t+}\psi^n}{\delta_{t\cdot}\eta^{n+1/2}}\ , \tag{8.81}$$

distance between the lips

$$\eta^{n+1/2} = b - y^{n+1/2} \tag{8.82}$$

and the location of the lower lip $b = -H_0$. Rewriting Eq. (8.81) to

$$\delta_{t+}\psi^n = g^{n+1/2}\delta_{t\cdot}\eta^{n+1/2} \tag{8.83}$$

and using identity

$$\mu_{t+}\psi^n = \frac{k}{2}\delta_{t+}\psi^n + \psi^n \tag{8.84}$$

we arrive at

$$M_{\mathrm{r}}\delta_{tt}y = -M_{\mathrm{r}}\omega_0^2\mu_{t\cdot}y - M_{\mathrm{r}}\sigma_{\mathrm{r}}\delta_{t\cdot}y + S_{\mathrm{r}}\Delta p + \left(\frac{k}{2}g^{n+1/2}\delta_{t\cdot}\eta^{n+1/2} + \psi^n\right)g^{n+1/2}, \tag{8.85}$$

where $g^{n+1/2}$ can be analytically obtained through

$$g^{n+1/2} = (\psi^{n+1/2})'\Big|_{\eta=\eta^{n+1/2}} = \sqrt{\frac{K_{\mathrm{c}}(\alpha_{\mathrm{c}}+1)}{2}}[\eta^{n+1/2}]_+^{\frac{\alpha_{\mathrm{c}}-1}{2}}, \tag{8.86}$$

with collision stiffness K_{c} (in N/m if $\alpha_{\mathrm{c}} = 1$) and non-linear collision coefficient α_{c}. Finally, because the barrier is static and below y through (8.82), this implies that

$$\delta_{t\cdot}\eta = -\delta_{t\cdot}y \tag{8.87}$$

and we can solve for $y^{n+3/2}$ (suppressing the $n+1/2$ superscript for) g

$$\begin{aligned}\frac{1}{k^2}(y^{n+3/2} - 2y^{n+1/2} + y^{n-1/2}) = &-\frac{\omega_0^2}{2}(y^{n+3/2} + y^{n-1/2}) - \frac{\sigma_{\mathrm{r}}}{2k}(y^{n+3/2} - y^{n-1/2}) \\ &+ \frac{S_{\mathrm{r}}}{M_{\mathrm{r}}}\Delta p - \frac{g^2}{4M_{\mathrm{r}}}(y^{n+3/2} - y^{n-1/2}) + \frac{g}{M_{\mathrm{r}}}\psi^n \\ \left(2 + \omega_0^2k^2 + \sigma_{\mathrm{r}}k + \frac{g^2k^2}{2M_{\mathrm{r}}}\right)y^{n+3/2} = &\ 4y^{n+1/2} + \left(\sigma_{\mathrm{r}}k - 2 - \omega_0^2k^2 + \frac{g^2k^2}{2M_{\mathrm{r}}}\right)y^{n-1/2} \\ &+ \frac{2S_{\mathrm{r}}k^2}{M_{\mathrm{r}}}\Delta p + \frac{2gk^2}{M_{\mathrm{r}}}\psi^n.\end{aligned}$$

Finally we get

$$\alpha_{\mathrm{r}} y^{n+3/2} = 4y^{n+1/2} + \beta_{\mathrm{r}} y^{n-1/2} + \xi_{\mathrm{r}} \Delta p + 4\psi^n \gamma_{\mathrm{r}} \tag{8.88}$$

with

$$\alpha_{\mathrm{r}} = 2 + \omega_0^2 k^2 + \sigma_{\mathrm{r}} k + g\gamma_{\mathrm{r}}, \quad \beta_{\mathrm{r}} = \sigma_{\mathrm{r}} k - 2 - \omega_0^2 k^2 + g\gamma_{\mathrm{r}},$$

$$\xi_{\mathrm{r}} = \frac{2S_{\mathrm{r}} k^2}{M_{\mathrm{r}}}, \quad \text{and} \quad \gamma_{\mathrm{r}} = \frac{gk^2}{2M_{\mathrm{r}}}\,.$$

For calculating Δp we follow the same steps as in Section 8.5.1 to get

$$\frac{2}{k}(\delta_{t\cdot} - \delta_{t-})y = -\omega_0^2(k\delta_{t\cdot} + e_{t-})y - \sigma_{\mathrm{r}}\delta_{t\cdot}.y + \frac{S_{\mathrm{r}}}{M_{\mathrm{r}}}\Delta p + \frac{1}{M_{\mathrm{r}}}\left(-\frac{k}{2}g\delta_{t\cdot}.y + \psi^n\right) g$$
$$a_1 \delta_{t\cdot}.y - a_2 \Delta p - a_3^n = 0, \tag{8.89}$$

with

$$a_1^n = \frac{2}{k} + \omega_0^2 k + \sigma_{\mathrm{r}} + \frac{g^2 k}{2M_{\mathrm{r}}} \geq 0, \quad a_2 = \frac{S_{\mathrm{r}}}{M_{\mathrm{r}}} \geq 0\,, \quad \text{and} \quad a_3^n = \left(\frac{2}{k}\delta_{t\cdot} - \omega_0^2 e_{t-}\right) y + \frac{g}{M_{\mathrm{r}}}\psi^n\,. \tag{8.90}$$

Note that a_1^n is now time-dependent through g but remains non-negative. The rest of the variables and process in Section 8.5.1 are unchanged.

Knowing $y^{n+3/2}$ we can calculate ψ^{n+1} through Eqs. (8.82) and (8.83) with

$$\psi^{n+1} = \psi^n - \frac{g}{2}\left(y^{n+3/2} - y^{n-1/2}\right) \tag{8.91}$$

Energy

The added energy to the system can be calculated by multiplying the added term with $\delta_{t\cdot}y$

$$\begin{aligned}
&\delta_{t+}(\mathfrak{h}_{\mathrm{t}} + \mathfrak{h}_{\mathrm{r}}) + \mathfrak{Q}_{\mathrm{r}} + \mathfrak{p}_{\mathrm{r}} - (\mu_{t+}\psi^n)\frac{\delta_{t+}\psi^n}{\delta_{t\cdot}\eta^{n+1/2}}(\delta_{t\cdot}y^{n+1/2}) = 0 \\
&\xleftrightarrow{\text{Eq. (8.87)}} \quad \ldots + (\mu_{t+}\psi^n)(\delta_{t+}\psi^n) = 0 \\
&\qquad\qquad \ldots + \frac{1}{2k}(\psi^{n+1} + \psi^n)(\psi^{n+1} - \psi^n) = 0 \\
&\qquad\qquad \ldots + \frac{1}{2k}((\psi^{n+1})^2 - (\psi^n)^2) = 0 \\
&\qquad\qquad \ldots + \frac{1}{2}\delta_{t+}\left((\psi^n)^2\right) = 0 \\
&\delta_{t+}(\mathfrak{h}_{\mathrm{t}} + \mathfrak{h}_{\mathrm{r}} + \mathfrak{h}_{\mathrm{c}}) + \mathfrak{Q}_{\mathrm{r}} + \mathfrak{p}_{\mathrm{r}} = 0
\end{aligned} \tag{8.92}$$

with

$$\mathfrak{h}_{\mathrm{c}} = \frac{(\psi^n)^2}{2}$$

8.6 Radiation

Following [53] we can add a radiation to the schemes using a circuit representation of the Levine and Schwinger radiation model (See Figure \ref{fig:radiation}). The system can be described as

$$\bar{v} = \mu_{t+} v_{(1)} + \frac{1}{R_2} \mu_{t+} p_{(1)} + C_\mathrm{r} \delta_{t+} p_{(1)}, \tag{8.93a}$$

$$\bar{p} = L_\mathrm{r} \delta_{t+} v_{(1)}, \tag{8.93b}$$

$$\bar{p} = \left(1 + \frac{R_1}{R_2}\right) \mu_{t+} p_{(1)} + R_1 C_\mathrm{r} \delta_{t+} p_{(1)}, \tag{8.93c}$$

where $\bar{p}^{n+1/2}$ and $\bar{v}^{n+1/2}$ lie on the interleaved temporal grid and are related to the tube by

$$\bar{p} = \mu_{t+} p_N^n, \quad \bar{S}_N \bar{v} = \mu_{x-} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} \right). \tag{8.94}$$

We can couple this to the tube by taking Eq. (**??**) at $l = N$

$$p_N^{n+1} = p_N^n - \frac{\rho_0 c \lambda}{\bar{S}_N} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} - S_{N-1/2} v_{N-1/2}^{n+1/2} \right), \tag{8.95}$$

and, similarly to (8.73), rewriting this to

$$\begin{aligned} p_N^{n+1} &= p_N^n - \frac{\rho_0 c \lambda}{\bar{S}_N} \left(2\mu_{x-} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} \right) - 2 S_{N-1/2} v_{N-1/2}^{n+1/2} \right), \\ p_N^{n+1} &= p_N^n - \frac{2\rho_0 c \lambda}{\bar{S}_N} \left(\bar{S}_N \bar{v} - S_{N-1/2} v_{N-1/2}^{n+1/2} \right). \end{aligned} \tag{8.96}$$

We can then find a definition for $\bar{v}$ by expanding system (8.93) and make Eq. (8.93a) solely dependent on known values of $v_{(1)}$, $p_{(1)}$ and p_N^n and the unknown p_N^{n+1} (as we can solve for the latter using (8.96)).

$$\bar{v} = \frac{1}{2} \left(v_{(1)}^{n+1} + v_{(1)}^n \right) + \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k} \right) p_{(1)}^{n+1} + \left(\frac{1}{2R_2} - \frac{C_\mathrm{r}}{k} \right) p_{(1)}^n \tag{8.97}$$

where, after expanding Eq. (8.93b)

$$v_{(1)}^{n+1} = \frac{k}{L_\mathrm{r}} \bar{p} + v_{(1)}^n, \tag{8.98}$$

and Eq. (8.93c)

$$\begin{aligned}
\bar{p} &= \left(1 + \frac{R_1}{R_2}\right)\mu_{t+}p_{(1)} + R_1 C_\mathrm{r}\delta_{t+}p_{(1)} \\
\bar{p} &= \frac{1}{2}\left(1 + \frac{R_1}{R_2}\right)\left(p_{(1)}^{n+1} + p_{(1)}^{n}\right) + \frac{R_1 C_\mathrm{r}}{k}\left(p_{(1)}^{n+1} - p_{(1)}^{n}\right) \\
\left(\frac{1}{2} + \frac{R_1}{2R_2} + \frac{R_1 C_\mathrm{r}}{k}\right) p_{(1)}^{n+1} &= \bar{p} + \left(\frac{R_1 C_\mathrm{r}}{k} - \frac{1}{2} - \frac{R_1}{2R_2}\right) p_{(1)}^{n} \\
p_{(1)}^{n+1} &= \underbrace{\left(\frac{2R_2 k}{2R_1 R_2 C_\mathrm{r} + k(R_1+R_2)}\right)}_{\zeta_1}\bar{p} + \underbrace{\left(\frac{2R_1 R_2 C_\mathrm{r} - k(R_1+R_2)}{2R_1 R_2 C_\mathrm{r} + k(R_1+R_2)}\right)}_{\zeta_2} p_{(1)}^{n}.
\end{aligned} \tag{8.99}$$

Filling these into Eq. (8.97) and using the definition of $\bar{p}$ from Eq. (8.94) yields

$$\begin{aligned}
\bar{v} &= \frac{1}{2}\left(\frac{k}{L_\mathrm{r}}(\mu_{t+}p_N^n) + 2v_{(1)}^n\right) + \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k}\right)\zeta_1\mu_{t+}p_N^n + \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k}\right)\zeta_2 p_{(1)}^n + \left(\frac{1}{2R_2} - \frac{C_\mathrm{r}}{k}\right)p_{(} \\
\bar{v} &= \underbrace{\left(\frac{k}{2L_\mathrm{r}} + \frac{\zeta_1}{2R_2} + \frac{C_\mathrm{r}\zeta_1}{k}\right)}_{\zeta_3}\mu_{t+}p_N^n + v_{(1)}^n + \underbrace{\left(\frac{\zeta_2+1}{2R_2} + \frac{C_\mathrm{r}\zeta_2 - C_\mathrm{r}}{k}\right)}_{\zeta_4} p_{(1)}^n.
\end{aligned} \tag{8.100}$$

Finally, filling in this definition for $\bar{v}$ into Eq. (8.96)

$$\begin{aligned}
p_N^{n+1} &= p_N^n - \frac{2\rho_0 c\lambda}{\bar{S}_N}\left(\bar{S}_N\left[\zeta_3\left(\frac{p_N^{n+1}+p_N^n}{2}\right) + v_{(1)}^n + \zeta_4 p_{(1)}^n\right] - S_{N-1/2}v_{N-1/2}^{n+1/2}\right) \\
p_N^{n+1} &= p_N^n - \rho_0 c\lambda\left(\zeta_3(p_N^{n+1}+p_N^n) + 2(v_{(1)}^n + \zeta_4 p_{(1)}^n) - \frac{2S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right) \\
(1+\rho_0 c\lambda\zeta_3)p_N^{n+1} &= (1-\rho_0 c\lambda\zeta_3)p_N^n - 2\rho_0 c\lambda\left(v_{(1)}^n + \zeta_4 p_{(1)}^n - \frac{S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right)
\end{aligned} \tag{8.101}$$

8.6.1 Energy

Recalling the condition at the right boundary from (**??**)

$$\mathfrak{b}_\mathrm{r} = (\mu_{t+}p_N)\underbrace{\mu_{x+}(S_{N-1/2}v_{N-1/2})}_{\mu_{x-}S_{N+1/2}v_{N+1/2}}, \tag{8.102}$$

using Eq. (8.94) we can rewrite this to

$$\mathfrak{b}_\mathrm{r} = \bar{p}\bar{S}_N\bar{v}. \tag{8.103}$$

then we can

Part IV

Interactions

Interactions

still unsure as to whether I want to have equations be denoted by roman numerals.. Probably want to move the equations to the chapter

ε

Newtons third law (action reaction)

Subscripts are needed

Somewhere in this Chapter have a section about fretting and how to generate different pitches using only one string

The models described in Part II already sound quite convincing on their own. However, these are just individual components that can be combined to approximate a fully functional (virtual) instrument. The following chapters will describe different ways of interaction between individual systems. Chapter 9 describes ways to connect different systems and Chapter 10 describes collision interactions between models.

Chapter 9

Connections

Something about connections

One can use connections to simulate a point-like damping finger on a string to create different pitches.

Pointlike $\delta(x - x_c)$ or distributed E_c

Alternative interpretation of grid points

Section 2.2.1 gives an introduction to how a continuous 1D system is subdivided into grid points in space (see Figure 2.2) in a process called discretisation. An alternative way to see grid points after discretisation is shown in Figure 9.1. Rather than grid 'points' with a spacing h between them, a continuous system is divided into grid 'sections' of width h. This interpretation allows the 'weight' of a grid point to be calculated from its material properties and geometry. The boundaries have a width of $h/2$ such that the total length $L = Nh$ m.

As an example, the weight of one grid point (or now rather grid section) of a string can be calculated as ρAh. The weight of one grid point of a 2D plane can be calculated as ρHh^2. As these grid points interact with each other, the forces resulting from this interaction are scaled by their respective weight per grid point....

This provides a better intuition for the interactions between components shown in this part.

9.1 Rigid connection

The simplest connection is Forces should be equal and opposite.

If component a is located 'above' component b, and their relative displacement is defined as $\eta = a - b$, then a positive η is going to have a negative effect

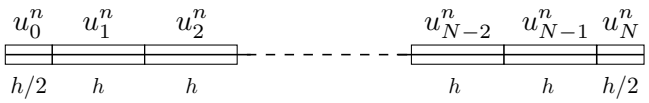


Fig. 9.1: Alternative interpretation of the discretisation of $u(x,t)$. The continuous system is divided into $N-1$ sections of width h plus 2 sections of width $h/2$ at the boundaries. Through this interpretation, the 'weight' of a grid point can be calculated from its physical parameters.

on a and a positive effect on b and vice-versa. This is important for the signs when adding the force terms to the schemes.

9.2 Spring-like connections

9.2.1 Connection with rigid barrier (scaled)

Consider the (scaled) 1D wave equation with an additional force term F^n

$$\delta_{tt} u_l^n = \gamma^2 \delta_{xx} u_l^n + J(x_\mathrm{c}) F^n \tag{9.1}$$

where

$$F^n = -\omega_0^2 \mu_{t\cdot} \eta^n - \omega_1^4 (\eta^n)^2 \mu_{t\cdot} \eta^n - 2\sigma_\times \delta_{t\cdot} \eta^n \tag{9.2}$$

and

$$\eta^n = I(x_\mathrm{c}) u_l^n. \tag{9.3}$$

To obtain F^n, an inner product of scheme (9.1) needs to be taken with $J(x_\mathrm{c})$ over domain $\mathcal{D}$ which, using identity (8.8) yields

$$\delta_{tt} I(x_\mathrm{c}) u_l^n = \gamma^2 I(x_\mathrm{c}) \delta_{xx} u_l^n + \underbrace{I(x_\mathrm{c}) J(x_\mathrm{c})}_{\|J(x_\mathrm{c})\|_{\mathcal{D}}^2} F^n. \tag{9.4}$$

As u is connected to a rigid barrier according to (9.3), a shortcut can be taken and Eqs. (9.2) and (9.3) can be directly substituted into Eq. (9.4) to get

$$\delta_{tt} \eta^n = \gamma^2 I(x_\mathrm{c}) \delta_{xx} u_l^n + \|J(x_\mathrm{c})\|_{\mathcal{D}}^2 \left(-\omega_0^2 \mu_{t\cdot} \eta^n - \omega_1^4 (\eta^n)^2 \mu_{t\cdot} \eta^n - 2\sigma_\times \delta_{t\cdot} \eta^n \right). \tag{9.5}$$

and solved for η^{n+1}:

$$\begin{aligned} &\Big(1 + \|J(x_\mathrm{c})\|_{\mathcal{D}}^2 k^2 [\omega_0^2/2 + \omega_1^4 (\eta^n)^2/2 + \sigma_\times/k]\Big) \eta^{n+1} \\ &\quad = 2\eta^n - \Big(1 + \|J(x_\mathrm{c})\|_{\mathcal{D}}^2 k^2 [\omega_0^2/2 + \omega_1^4 (\eta^n)^2/2 - \sigma_\times/k]\Big) \eta^{n-1} \\ &\qquad + \gamma^2 k^2 I(x_\mathrm{c}) \delta_{xx} u_l^n \end{aligned} \tag{9.6}$$

This can then be used to calculate F^n in (9.2) and can in turn be used to calculate u_l^{n+1} in (9.1).

9.2.2 String-plate connection

In this example, let's consider a string connected to a plate using a nonlinear damped spring. This could be interpreted as a simplified form of how guitar string would be connected to the body.

Continuous

The systems in isolation are as in (4.4) and (6.6), but with an added force term:

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa_\mathrm{s}^2 \partial_x^4 u - 2\sigma_{0,\mathrm{s}} \partial_t u + 2\sigma_{1,\mathrm{s}} \partial_t \partial_x^2 u - \delta(x - x_\mathrm{c}) \frac{f}{\rho_\mathrm{s} A} \tag{9.7a}$$

$$\partial_t^2 w = -\kappa_\mathrm{p}^2 \Delta\Delta w - 2\sigma_{0,\mathrm{p}} \partial_t w + 2\sigma_{1,\mathrm{p}} \partial_t \partial_x^2 w + \delta(x - x_\mathrm{c}, y - y_\mathrm{c}) \frac{f}{\rho_\mathrm{p} H} \tag{9.7b}$$

where

$$f = f(t) = K_1 \eta + K_3 \eta^3 + R\dot{\eta} \tag{9.8}$$

and

$$\eta = \eta(t) = u(x_\mathrm{c}, t) - w(x_\mathrm{c}, y_\mathrm{c}, t) \tag{9.9}$$

Discrete

System (9.7) can then be discretised as

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa_\mathrm{s}^2 \delta_{xxxx} u_l^n - 2\sigma_{0,\mathrm{s}} \delta_{t\cdot} u_l^n + 2\sigma_{1,\mathrm{s}} \delta_{t-} \delta_{xx} u_l^n - J_\mathrm{s}(x_\mathrm{c}) \frac{f^n}{\rho_\mathrm{s} A} \,, \tag{9.10}$$

$$\delta_{tt} w_l^n = -\kappa_\mathrm{p}^2 \delta_\Delta \delta_\Delta w_l^n - 2\sigma_{0,\mathrm{p}} \delta_{t\cdot} w_l^n + 2\sigma_{1,\mathrm{p}} \delta_{t-} \delta_{xx} w_l^n + J_\mathrm{p}(x_\mathrm{c}, y_\mathrm{c}) \frac{f^n}{\rho_\mathrm{p} H} \,, \tag{9.11}$$

where

$$f^n = K_1 \mu_{tt} \eta^n + K_3 (\eta^n)^2 \mu_{t\cdot} \eta^n + R\delta_{t\cdot} \eta^n, \tag{9.12}$$

and

$$\eta^n = I(x_\mathrm{c}) u_l^n - I(x_\mathrm{c}, y_\mathrm{c}) w_l^n. \tag{9.13}$$

Expansion

System (9.10) can be expanded at the connection location x_c by taking an inner product of the schemes with their respective spreading operators.

9.2.3 Solving for f

9.2.4 Non-dimensional

The scaled system can be written as:

$$\partial_t^2 u = \gamma^2 \partial_x^2 u - \kappa_{\mathrm{s}}^2 \partial_x^4 u - 2\sigma_{0,\mathrm{s}} \partial_t u + 2\sigma_{1,\mathrm{s}} \partial_t \partial_x^2 u - \delta(x - x_{\mathrm{c}}) F \tag{9.14}$$

$$\partial_t w = -\kappa_{\mathrm{p}}^2 \Delta\Delta w - 2\sigma_{0,\mathrm{p}} \partial_t w + 2\sigma_{1,\mathrm{p}} \partial_t \partial_x^2 w + \delta(x - x_{\mathrm{c}}, y - y_{\mathrm{c}}) F \tag{9.15}$$

where

$$F = F(t) = \omega_1^2 \eta + \omega_3^4 \eta^3 + \sigma_{\mathrm{c}} \dot{\eta} \tag{9.16}$$

and

$$\eta = \eta(t) = u(x_{\mathrm{c}}, t) - w(x_{\mathrm{c}}, y_{\mathrm{c}}, t) \tag{9.17}$$

$$I(x_{\mathrm{c}}) \delta_{tt} u_l^n = c^2 \left(I(x_{\mathrm{c}}) \delta_{xx} u_l^n \right) + I(x_{\mathrm{c}}) J(x_{\mathrm{c}}) F \tag{9.18}$$

Chapter 10

Collisions

Something about collisions

10.0.1 Mass-spring Systems Revisited: Adding Damping

Damping can be added to Eq. (2.28)

$$M\ddot{u} = -Ku - R\dot{u} \tag{10.1}$$

with damping coefficient R (in kg/s). To this system we can add an external force:

$$M\ddot{u} = -Ku - R\dot{u} + F \tag{10.2}$$

where

10.1 Classic models

Note that when using Eq. (7.37) in [3]

10.2 Michele's tricks

Part V

Contributions

This part presents the contributions made throughout this PhD project.

It can be seen as an extended summary of the published work in VIII.

This part starts by introducing the dynamic grid in Chapter 11, a method to dynamically vary grid configurations in FDTD simulations. After this, several full instrument models that have been developed during the PhD will be presented. Chapter 13 shows a three instrument-inspired case-studies using a large-scale modular environment, Chapter 14 describes the implementation of the tromba marina and Chapter 15 that of the trombone.

Chapter 11

Dynamic Grid

Often in math, you should view the definition not as a starting point, but as a target. Contrary to the structure of textbooks, mathematicians do not start by making definitions and then listing a lot of theorems, and proving them, and showing some examples. The process of discovering math typically goes the other way around. They start by chewing on specific problems, and then generalising those problems, then coming up with constructs that might be helpful in those general cases,and only then you write down a new definition (or extend an old one). - Grant Sanderson (AKA 3Blue1Brown) https://youtu.be/O85OWBJ2ayo?t=359

11.1 Background and Motivation

Simulating musical instruments using physical modelling – as mentioned in Part I– allows for manipulations of the instrument that are impossible in the physical world. Examples of this are changes in material density or stiffness, cross-sectional area (1D), thickness (2D) and size. Apart from being potentially sonically interesting, there are examples in the physical world where certain aspects of the instrument are manipulated in real-time.

check if is still true

Tension in a string is changed when tuning it

Some artists even use this in their performances [74, 75] Or gr4yhound: https://www.youtube.com/watch?v=fSQ9Dg65EFo

The hammered dulcimer is another example where the strings are tensioned over a bridge where one can play the string at one side of the bridge, while pushing down on the same string on the other side [76].

1D:

- Trombone
- Slide whistle

- Guitar strings
  - Fretting finger pitch bend
  - Above the nut [75]
  - Tuning pegs directly [74]
- Hammered dulcimer [76]
- Erhu?

2D:

- Timpani
- Bodhrán: https://youtu.be/b9HyB5yNS1A?t=146
- Talking drum (hourglass drum): https://youtu.be/B4oQJZ2TEVI?t=9
- Flex-a-tone (could also be 1D tbh..): https://www.youtube.com/watch?v=HEW1aG8XJQ

A more relevant example is that of the trombone, where the size of the instrument is changed in order to play different pitches. Modelling this using FDTD methods would require

In his thesis, Harrison points out that in order to model the trombone, grid points need to be introduced

Something about time-dependent variable coefficient Stokes flow: https://arxiv.org/abs/10

Time-varying propagation speed in waveguides: https://quod.lib.umich.edu/cgi/p/pod/idx/fractional-delay-application-time-varying-propagation-speed.pdf?c=icmc;idno=bbp2372.

Special boundary conditions (look at!): Modeling of Complex Geometries and Boundary Conditions in Finite Difference/Finite Volume Time Domain Room Acoustics Simulation (`https://www.researchgate.net/publication/260701231_Modeling_of_Complex_Geometries_and_Boundary_Conditions_in_Finite_DifferenceFinite_Volume_Time_Domain_Room_Acoustics_Simulation`)

11.2 Method

Iterations have been:

- Interpolated boundary conditions
- Linear interpolation

These sections are taken from the JASA appendix

In this appendix, some iterations done over the course of this project will be shown in more detail. In the following, the 1D wave equation with a wave speed of $c = 1470$ m/s, a length of $L = 1$ m, Dirichlet boundary conditions

and a sample rate of $f_s = 44100$ Hz is considered, and – through Eq. (**??**) – satisfies the CFL condition with equality. These values result in $N = 30$, or a grid of 31 points including the boundaries. Then, the wave speed is decreased to $c \approx 1422.6$ m/s, i.e., the wave speed that results in $N = 31$ and satisfies the stability condition with equality again.

11.2.1 Full-Grid Interpolation

One way to go from one grid to another is performing a full-grid interpolation [3, Chap. 5]. If the number of points changes according to Eq. (**??**), i.e., if $N^n \neq N^{n-1}$ the full state of the system ($u_l^n, u_l^{n-1}\ \forall l$) can be interpolated to the new state. See Figure 11.1.

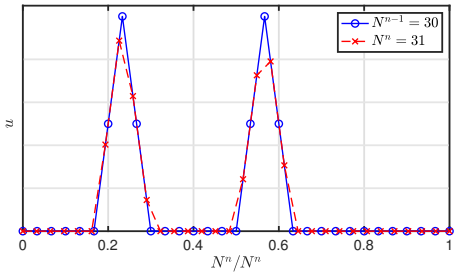


Fig. 11.1: Upsampling u (with an arbitrary state) using (linear) full-grid interpolation with $N^{n-1} = 30$ and $N^n = 31$. The horizontal axis is normalised with respect to N^n.

An issue that arises using this method is that the Courant number λ will slightly deviate from the CFL condition as c changes. Using Eq. (**??**) with L/ck approaching 31 (from below), the minimum value of $\lambda \approx 30/31 \approx 0.9677$. This, employing Eq. (**??**), has a maximum frequency output of $f_{\text{max}} \approx 18,475$ Hz. The Courant number will deviate more for higher values of c and thus lower values for N – for instance, if N approaches 11 (from below), $\lambda \approx 10/11 \approx 0.9091$ and $f_{\text{max}} \approx 16,018$ Hz.

Another problem with full-grid interpolation, is that it has a low-passing effect on the system state, and thus on the output sound. Furthermore, this state-interpolation causes artefacts or 'clicks' in the output sound as the method causes sudden variations in the states.

All the aforementioned issues could be solved by using a (much) higher sample rate and thus more grid points, but this would render this method impossible to work in real time.

11.2.2 Adding and removing Points at the Boundary

To solve the issues exhibited by a full-grid interpolation, points can be added and removed at a single location and leave most points unaffected by the

parameter changes. A good candidate for a location to do this is at a fixed (Dirichlet) boundary. The state u at this location is always 0 so points can be added smoothly.

As c decreases, h can be calculated according to Eq. (**??**) and decreases as well.

This has a physical analogy with tuning a guitar string. Material enters and exits the neck (playable part of the string) at the nut, which in discrete time means grid points appearing and disappearing at one boundary.

To yield smooth changes between grid configurations, an interpolated boundary has been developed, the possibility of which has been briefly mentioned in [3, p. 145]. The Dirichlet condition in Eq. (2.39a) can be extended to be the simply supported boundary condition:

$$u(x,t) = \frac{\partial^2}{\partial x^2} u(x,t) = 0 \quad \text{where} \quad x = 0, L, \tag{11.1}$$

or, when discretised,

$$u_l^n = \delta_{xx} u_l^n = 0, \quad \text{where} \quad l = 0, N. \tag{11.2}$$

This means that on top of that the state of the boundary should be 0, the curvature around it should also be 0. One can again solve for the virtual grid points at the boundary locations, yielding

$$u_{-1}^n = -u_1^n \quad \text{and} \quad u_{N+1}^n = -u_{N-1}^n. \tag{11.3}$$

This is visualised in Figure 11.2.

If the flooring operation in Eq. (**??**) is removed this introduces a fractional number of grid points.

The by-product of using a fractional N this is that the CFL condition in (2.46) can now always be satisfied with equality no matter what the wave speed is.

An issue with this method is that removing points is much harder than adding.

their interactions change through a change in the grid spacing and wave speed. This interaction, though, is defined by λ which

11.2.3 Cubic interpolation

11.2.4 Sinc interpolation

11.2.5 Displacement correction

The displacement correction can be interpreted as a spring force pulling u_M^n and w_0^n to the average displacement.

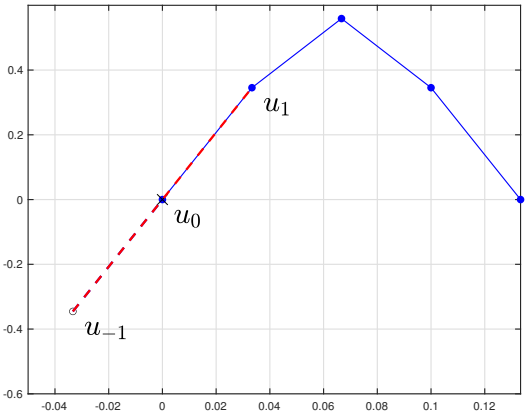


Fig. 11.2: The simply supported boundary condition: both the state and the curvature at the boundary – at $l = 0$ – should be 0.

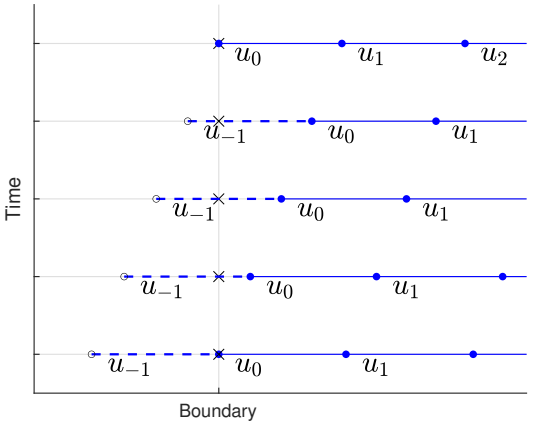


Fig. 11.3: The grid changing over time

$$
\begin{aligned}
u_M^{n+1} &= 2u_M^n + \lambda^2(u_{M-1}^n - 2u_M^n + u_{M+1}^n) - K\left(u_M^n - \frac{u_M^n + w_0^n}{2}\right) \\
w_0^{n+1} &= 2u_M^n + \lambda^2(w_{-1}^n - 2w_0^n + w_1^n) - K\left(w_0^n - \frac{u_M^n + w_0^n}{2}\right)
\end{aligned} \tag{11.4}
$$

$$
\begin{aligned}
u_M^{n+1} &= 2u_M^n + \lambda^2(u_{M-1}^n - 2u_M^n + u_{M+1}^n) + \frac{K}{2}(w_0^n - u_M^n) \\
w_0^{n+1} &= 2u_M^n + \lambda^2(w_{-1}^n - 2w_0^n + w_1^n) - \frac{K}{2}(w_0^n - u_M^n)
\end{aligned} \tag{11.5}
$$

with $K = K(\alpha)$

$$K = (1 - \alpha)^{\epsilon}. \tag{11.6}$$

11.3 Analysis and Experiments

11.3.1 Interpolation technique

11.3.2 Interpolation range

11.3.3 Location

... where to add and remove points
Using the whole range, we can still add/remove points at the sides.

11.4 Discussion and Conclusion

Chapter 12

Real-Time Implementation

JUCE Give overall structure of code

Implementation of the physical models using FDTD methods

As mentioned in Chapter 1, FDTD methods are used for high-quality and accurate simulations, rather than for real-time applications. This is due to their lack of simplifications.

Usually, `MATLAB` is used for simulating

Here, an interactive application is considered real-time when

Control of the application generates or manipulates audio with no noticeable latency.

For human-computer interaction, the task at hand greatly determines how much latency is acceptable. Wessel and Wright [77] place the upper limit of latency when interacting with computers for musical purposes at 10 ms. Moreover, they place the a limit on the *jitter*, or of variation of the latency, at 1 ms. It is thus important to keep the CPU usage at a fixed level, and different ways of controlling and interacting with the application should not influence the amount of computations much...

Also the application needs to be controlled continuously

Naming conventions: CamelCase...

Finite representation: $1.80 \cdot 10^{308}$

Helps to (informally) evaluate the models by interacting with it in a natural way (rather than static parameters)

Refer to [78] somewhere here

Figure with programming languages sorted by speed

12.1 MATLAB vs. C++

It is usually a good idea to prototype a physical modelling application in MATLAB for several reasons:

- Easier to debug
  - Plotting functionality
  - No need for memory handling
- Instability due to programming errors

12.1.1 Speed

Here is where the power of C++

12.1.2 Syntax

Indexing in Matlab is 1-based, meaning that the index of a vector starts at 1. If `u` is a vector with 10 elements, the first element is retrieved as `u(1)` and the last as `u(10)`. C++, on the other hand, is 0-based and retrieving the first and last element of a size-10 vector happens through `u[0]` and `u[9]` respectively.

12.2 Do's and don'ts in Real-Time FD schemes

"The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming." [Knuth 1974]

(Knuth, Donald E. (Dec. 1974). "Computer Programming As an Art". In: Commun. ACM 17.12, pp. 667–673. ISSN: 0001-0782. DOI: 10.1145/361604.361612. URL: http://doi.acm.org/10.1145/361604.361612.) Some of the things I learned (the hard way)...

- Create a limiter
- Structure your application into classes
- Use pointer switches
- Comment your code (hehe)
- Working with JUCE: select your audio output before building the application...

Create a limiter

Programming errors happen. To save your speakers, headphones or – most importantly – your ears, create a limiter.

```
double limit (double val)
{
    if (val < -1)
    {
        val = -1;
        return val;
    }
    else if (val > 1)
    {
        val = 1;
        return val;
    }
    return val;
}
```

```
for  i = 0:lengthSound
    uNext = ...
end
```

Use pointer switches

One of the most important things in working with FD schemes for real-time audio applications is to

The maximum number of copy-operations this takes is $2(N+1)$ if the boundaries also need updated in the case of free or Neumann boundary conditions. For a string with simply supported or clamped boundary conditions this can be reduced to $2N$ or $2(N-1)$ respectively, but does not maka big difference in computational time.

A pointer switch, however, only needs 4 copy-operations per iteration as shown in Algorithm 12.1

```
for n = 1:lengthSound
    ...
    uPrev = u;
    u = uNext;
end
```

In C++ this is done using

```
double updateStates()
{
    double* uTmp = u[2];
    u[2] = u[1];
    u[1] = u[0];
```

```
    u[0] = uTmp;
}
```

Algorithm 12.1: Implementation of a pointer switch also shown in Figure 12.1b. A temporary pointer is assigned to where the $\mathbf{u}^{n-1}$ pointer is currently pointing at to be able to assign that location in memory to the $\mathbf{u}^{n+1}$ pointer in the end.

Grouping terms and Precalculating coefficients

Generally in implementations of FD schemes the most computationally expensive part of the algorithm is the calculation of the scheme itself. This is due to the rate at which it needs to be updated which usually is 44100 Hz. Graphics can be updated at rates orders of magnitude lower than the audio (say 10-20 Hz) and still be considered smooth enough.

Recall the update equation for the 1D wave update equation in Eq. (2.44)

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

Grouping the terms like this allows for the coefficients multiplied onto the grid function at different temporal and spatial indices to be precomputed. This can significantly decrease the amount of operations per sample.

The undamped stiff string FD scheme

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n, \tag{12.1}$$

can be expanded to an update equation as

$$\begin{aligned} u_l^{n+1} = 2u_l^n - u_l^{n-1} &+ \lambda^2 \left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right) \\ &- \mu^2 \left(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n + -4u_{l-1}^n + u_{l-2}^n\right) \end{aligned} \tag{12.2}$$

where $\lambda = ck/h$ and $\mu = \kappa k/h^2$.

For implementation purposes there is a better way to write this scheme that reduces the amount of computations. This is done by collecting the terms based on the grid function and pre-calculating the coefficients multiplied onto these. As the schemes are spatially symmetric, "neighbouring points" relative to u_l^n can also be grouped to get

$$u_l^{n+1} = \underbrace{(2 - 2\lambda^2 - 6\mu^2)}_{\texttt{B1}} u_l^n + \underbrace{(\lambda^2 + 4\mu^2)}_{\texttt{B2}} \left(u_{l+1}^n + u_{l-1}^n\right) \underbrace{-\mu^2}_{\texttt{B3}} \left(u_{l+2}^n + u_{l-2}^n\right) \underbrace{-}_{\texttt{C1}} u_l^{n-1}. \tag{12.3}$$

These coefficients can then be pre-calculated and do not have to be

```
//// Constructor ////
B1 = 2.0 - 2.0 * lambdaSq - 6.0 * muSq; // u_l^n
B2 = lambdaSq + 4.0 * muSq;             // u_{l+-1}^n
B3 = -muSq;                             // u_{l+-2}^n
C1 = -1.0;                              // u_l^{n-1}

//// Function where scheme is calculated ////
for (int l = 2; l < N-1; ++l) // clamped boundaries
{
    u[0][l] = B1 * u[1][l]
            + B2 * (u[1][l+1] + u[1][l-1])
            + B3 * (u[1][l+2] + u[1][l-2])
            + C1 * u[2][l];
```

```
}
```

Algorithm 12.2: Precalculation

Denormalised numbers

The damping present in FD schemes causes the state of the system to exponentially decay. What this means for the values of the state vectors in implementation, is that they keep getting closer to 0 but never reach it.

After a long period of time, which depends on the damping coefficients, state values can get in the range of $\sim 10^{-307}$! Numbers in this range are referred to as *denormalised numbers* and operations with these are "extremely slow" [79].

Although it rarely happens that numbers end up in this range, especially when the application is continuously interacted with, it is good to account for the possibility. For example, due to the very high damping of the body in the Tromba Marina application in Chapter 14, denormalised numbers appear after ~ 10 s of not interacting with the instrument, and the CPU usage shoots up. There are specific processor flags that can be activated to truncate denormalised numbers to 0. To retain generality (cross-platform, various processors), one could implement a simple check per buffer to see whether values are smaller than fx. 10^{-250} and truncate all values of that system to 0.

12.3 Graphics

```
void Simple1DWave::paint (juce::Graphics& g)
{
    // clear the background
    g.fillAll (getLookAndFeel().findColour
    (juce::ResizableWindow::backgroundColourId));

    Path stringState = visualiseState (g, 100);

    // choose your favourite colour
    g.setColour (Colours::cyan);

    // visualScaling depends on your excitation
    g.strokePath (visualiseState (g, 100), PathStrokeType(2.0f));

}

Path Simple1DWave::visualiseState (Graphics& g, double visualScaling)
{
    // String-boundaries are in the vertical middle of the component
    double stringBoundaries = getHeight() / 2.0;
```

```
    // initialise path
    Path stringPath;

    // start path
    stringPath.startNewSubPath (0, -u[1][0] * visualScaling +
    stringBoundaries);

    double spacing = getWidth() / static_cast<double>(N);
    double x = spacing;

    for (int l = 1; l <= N; l++)
    {
        // Needs to be -u, because a positive u would visually go down
        float newY = -u[1][l] * visualScaling + stringBoundaries;
        if (isnan(x) || isinf(abs(x)) || isnan(u[1][l]) ||
    isinf(abs(u[1][l])))
            std::cout << "Wait" << std::endl;

        // if we get NAN values, make sure that we don't get an
        // exception
        if (isnan(newY))
            newY = 0;

        stringPath.lineTo (x, newY);
        x += spacing;
    }

    return stringPath;
}
```

```
void Simple1DWave::excite()
{
    //// Raised cosine excitation ////

    // width (in grid points) of the excitation
    double width = 10;
    double excitationLoc = 0.2;
    // make sure we're not going out of bounds at the left boundary
    int start = std::max (floor((N+1) * excitationLoc) - floor(width *
    0.5), 1.0);

    for (int l = 0; l < width; ++l)
    {
        // make sure we're not going out of bounds
        // at the right boundary (this does 'cut off'
        // the raised cosine)

        if (l+start > N - 1)
            break;

        u[1][l+start] += 0.5 * (1 - cos(2.0 * double_Pi * l /
    (width-1.0)));
        u[2][l+start] += 0.5 * (1 - cos(2.0 * double_Pi * l /
    (width-1.0)));
    }
}
```

12.4 Matrices

Library called *Eigen* [80]

12.4.1 Matrix Inversions in Real-Time

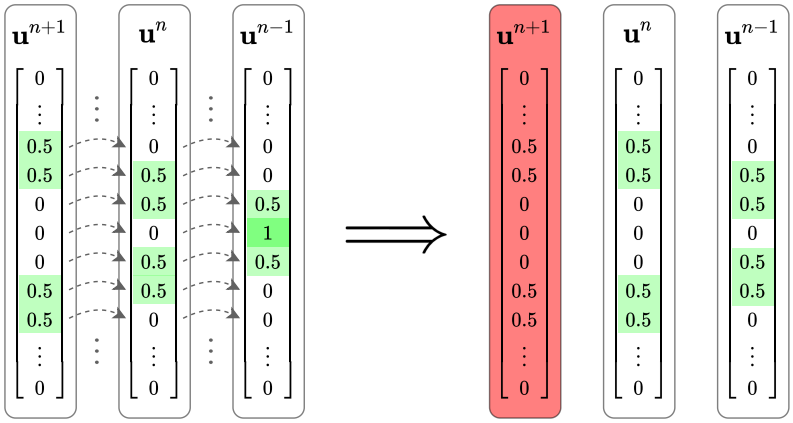
For small (2×2 and 3×3) matrices it is doable to do the inversion 'by hand'. This requires finding the *determinant* of a matrix

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w},$$

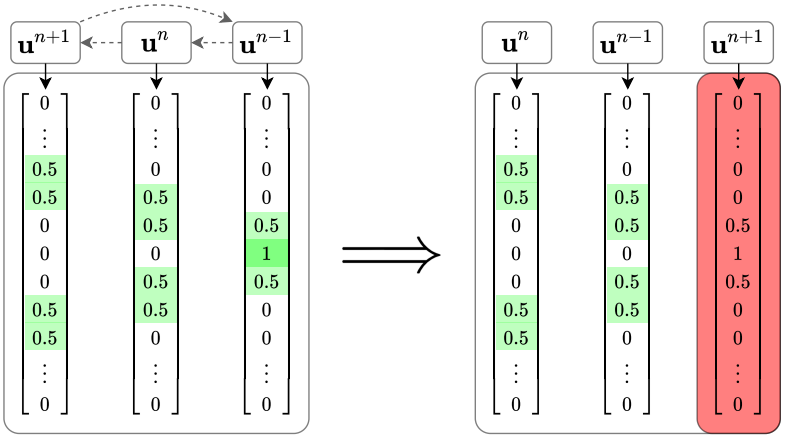
where

$$\mathbf{A}^{-1} = \frac{1}{a_{00} \cdot a_{11} - a_{01} \cdot a_{10}} \begin{bmatrix} a_{11} & -a_{01} \\ -a_{10} & a_{00} \end{bmatrix} \tag{12.4}$$

```
det = a00 * a11 - a01 * a10;
u0 = (w0 * a11 - w1 * a01) / det;
u1 = (-w0 * a10 + w1 * a00) / det;
```



(a) Copying values: $2(N+1)$ operations per iteration.



(b) Pointer switch: 4 operations per iteration.

Fig. 12.1: Updating the state vectors by (a) copying all values individually, or (b) performing a pointer switch. Non-zero values are highlighted in green for clarity. The values of the red vector will be overwritten by the update of the scheme in the next iteration so these values will no longer be used.

Chapter 13

Large Scale Modular Physical models

In the paper "Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph" [A] we presented a modular physical modelling environment using three instruments as case studies.

13.1 Bowed Sitar

- Stiff String
- Thin Plate
- Pluck
- Bow
- Non-linear spring connections

13.2 Dulcimer

- Stiff String
- Thin Plate
- Hammer (simple)
- Non-linear spring connections

13.3 Hurdy Gurdy

- Stiff String
- Thin Plate
- Pluck
- Bow
- Non-linear spring connections

Chapter 14

Tromba Marina

14.1 Introduction

The tromba marina is a bowed monochord instrument from medieval Europe. It has a long quasi-trapezoidal body and is unique due to its oddly-shaped bridge that the string rests on. The bridge is often called a 'shoe' due to its shape and is free to rattle against the body in sympathy with the movement of the vibrating string. This rattling causes a sound with brass or trumpet-like qualities, hence the name *tromba* which stems from the Italian word trumpet. The rarity of the instrument as well as its interesting physics makes it an ideal case for a physical modelling implementation.

14.2 Physical Model

Using linear (partial) differential operator $\mathcal{L}$

$$\mathcal{L}q = 0 \tag{14.1}$$

where $q(\boldsymbol{x}, t)$

14.2.1 Continuous

$$\mathcal{L}_{\mathrm{s}} = \rho_{\mathrm{s}} A \partial_t^2 - T \partial_\chi^2 + E_{\mathrm{s}} I \partial_\chi^4 + 2\rho_{\mathrm{s}} A \sigma_{0\mathrm{s}} \partial_t - 2\rho_{\mathrm{s}} A \sigma_{1\mathrm{s}} \partial_t \partial_\chi^2 \,. \tag{14.2}$$

Complete system

Test

14.2.2 Discrete

14.3 Real-Time Implementation

14.3.1 Control using Sensel Morph

14.3.2 VR Application

Chapter 15

Trombone

Published in [H]

15.1 Introduction

Interesting read: https://newt.phys.unsw.edu.au/jw/brassacoustics.html

15.2 Physical Model

Most has been described in Chapter 5

15.2.1 Continuous

Just to save the conversation with Stefan about Webster's equation:

Using operators ∂_t and ∂_x denoting partial derivatives with respect to time t and spatial coordinate x, respectively, a system of first-order PDEs describing the wave propagation in an acoustic tube can then be written as [A]

$$\frac{S}{\rho_0 c^2}\partial_t p = -\partial_x(Sv) \tag{15.1a}$$

$$\rho_0 \partial_t v = -\partial_x p \tag{15.1b}$$

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 (15.1) can be condensed into a second-order equation in p alone, often referred to as Webster's equation [**?**]. Interesting! In NSS it is the acoustic potential right? Can you go from that to a second-order PDE in p? There is a time-derivative hidden there somewhere right? (Just wondering :))Yes, the form

in p alone is the one you usually see. You get it by differentiating the first equation, giving you a $\dot{v}$ on the RHS, and then you can substitute the second equation in...I used the velocity potential one because it has direct energy balance properties. Right. So Webster's eq. in p and Ψ are identical (will exhibit identical behaviour), except for the unit of the state variable..?yes that's right...using the velocity potential allows you to do all the energy analysis easily, in terms of physical impedances. But the scheme you get to in the end is the same, just one derivative down. Alright cool! Thanks for the explanation :) For simplicity, effects of viscothermal losses have been neglected in (15.1). For a full time domain model of such effects in an acoustic tube, see, e.g. [**?**].

15.2.2 Discrete

15.3 Real-Time Implementation

Unity??

15.4 Discussion

more for your info, don't think I want to include this: To combat the drift, experiments have been done involving different ways of connecting the left and right tube. One involved alternating between applying the connection to the pressures and the velocity. Here, rather than adding points to the left and right system in alternating fashion, points were added to pressures p and q and velocities v and w in an alternating fashion. Another experiment involved a "staggered" version of the connection where (fx.) for one system (either left or right), a virtual grid point of the velocity was created from known values according to (**??**), rather than both from pressures. This, however, showed unstable behaviour. No conclusory statements can be made about these experiments at this point. ← which is exactly why I don't want to include this section

As the geometry varies it matters a lot where points are added and removed as this might influence the way that the method is implemented. speculative section coming up The middle of the slide crook was chosen, both because it would be reasonable for the air on the tube to "go away from" or "go towards" that point as the slide is extended or contracted, and because the geometry does not vary there. Experiments with adding / removing grid points where the geometry varies have been left for future work. even more speculative.. → It could be argued that it makes more sense to add points at the ends of the inner slides as "tube material" is also added there. This would mean that the system should be split in three parts: "inner slide", "outer slide" and "rest", and would complicate things even more.

Part VI

Conclusions and Perspectives

Chapter 16

Conclusions and Perspectives

Both the good and bad thing about physical modelling musical instruments is that you are never done..

title is the exact same as [3]

There is always more work to be done

16.1 Applications

The implementations presented in this work, as well as real-time physical models of musical instruments in general, have several applications in the real world. First of all, the implementations can be used as audio plug-ins that music producers could use if they do not have

The now-digital musical instrument

What a keyboard is to a piano, many other controllers inspired by their real-life counterpart could be made to trigger physical models of their respective instrument. Put this work into perspective of the literature (higher level)

16.2 Realism

We are not there yet..

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.

Parameter design

Many parameters that can always be improved

Possible perspectives could be machine learning based on audio files (literature...)

16.3 Dynamic grid

In this work, the dynamic grid in Chapter 11 the test case of the 1D wave equation has been explored.

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] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.

[9] 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.

[10] ——, "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.

[11] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the CORDIS system," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.

[12] 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.

[13] K. Karplus and A. Strong, "Digital synthesis of plucked-string and drum timbres," *Computer Music Journal*, vol. 7, pp. 43–55, 1983.

[14] J. O. Smith, "Music applications of digital waveguides," Technical Report, CCRMA Stanford University, 1987.

[15] ——, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[16] J.-M. Adrien, "The missing link: Modal synthesis," in *Representations of Musical Signals*, G. De Poli, A. Picalli, and C. Roads, Eds. MIT Press, 1991, pp. 269–298.

[17] J. D. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

[18] 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.

[19] G. De Poli and D. Rocchesso, "Physically based sound modelling," *Organised Sound*, vol. 3, no. 1, pp. 61–76, 1998.

[20] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1998.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] C. Erkut and M. Karjalainen, "Finite difference method vs. digital waveguide method in string instrument modeling and synthesis," *International Symposium on Musical Acoustics*, 2002.

[26] E. Maestre, C. Spa, and J. O. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC|SMC|2014*, pp. 1305–1311, 2014.

[27] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," 1979, thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France.

[28] 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.

[29] 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.

[30] 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.

[31] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114—117, 1965.

[32] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[33] S. Mehes, M. van Walstijn, and P. Stapleton, "Towards a virtual-acoustic string instrument," *Proceedings of the 13th Sound and Music Computing Conference (SMC)*, 2016.

[34] F. Pfeifle and R. Bader, "Real-time finite difference physical models of musical instruments on a field programmable gate array (FPGA)," *Proceedings of the 15th International Conference on Digital Audio Effects (DAFx)*, 2012.

[35] ——, "Real-time finite-difference method physical modeling of musical instruments using field-programmable gate array hardware," *Journal of the Audio Engineering Society (JASA)*, vol. 63, no. 12, pp. 1001–1016, 2015.

[36] F. Pfeifle, "Real-time physical model of a wurlitzer and a rhodes electric piano," *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx)*, 2017.

[37] J. Bybee, "COSM REVISITED," Accessed June 28, 2021. [Online]. Available: https://www.boss.info/us/community/boss_users_group/1319/

[38] 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.

[39] 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.

[40] Sensel Inc., "Sensel | Interaction Evolved," 2021. [Online]. Available: https://sensel.com/

[41] 3D Systems, Inc, "3D Systems Touch Haptic Device," 2021. [Online]. Available: https://www.3dsystems.com/haptics-devices/touch

[42] B. Hamilton, "Finite difference and finite volume methods for wave-based modelling of room acoustics," Ph.D. dissertation, The University of Edinburgh, 2016.

[43] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2018.

[44] 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.

[45] 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.

[46] S. Bilbao, "A modular percussion synthesis environment," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.

[47] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen de mathematischen physik," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928.

[48] T. H. Park, *Introduction To Digital Signal Processing: Computer Musically Speaking*. World Scientific Publishing Co. Pte. Ltd, 2010.

[49] J. G. Charney, R. Fjörtoft, and J. V. Neumann, "Numerical integration of the barotropic vorticity equation," *Tellus*, vol. 2, no. 4, 1950.

[50] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, California: Wadsworth and Brooks/Cole Advanced Books and Software, 1989.

[51] 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.

[52] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time-Dependent Problems and Difference Methods*, 2nd ed. John Wiley & Sons, 2013.

[53] R. L. Harrison-Harsley, "Physical modelling of brass instruments using finite-difference time-domain methods," Ph.D. dissertation, University of Edinburgh, 2018.

[54] 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.

[55] 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.

[56] S. A. van Duyne and J. O. Smith, "Physical modeling with the 2-D digital waveguide mesh," in *ICMC Proceedings*, 1993.

[57] F. P. Bowden and L. Leben, "The nature of sliding and the analysis of friction," in *Proceedings of the Royal Society of London*, vol. 169, 1939.

[58] H. L. F. von Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, 1954, english translation of 1863 (German) edition by A. J. Ellis.

[59] J. O. Smith, "Efficient simulation of the reed bore and bow string mechanics," *ICMC 86*, pp. 275–280, 1986.

[60] 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.

[61] P. Dupont, V. Hayward, B. Armstrong, and F. Altpeter, "Single state elastoplastic friction models," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, pp. 787–792, 2002.

[62] S. Serafin, F. Avanzini, and D. Rocchesso, "Bowed string simulation using an elasto-plastic friction model," *SMAC 03*, pp. 95–98, 2003.

[63] S. Serafin, "The sound of friction: Real-time models, playability and musical applications," Ph.D. dissertation, CCRMA, 2004.

[64] J. Woodhouse, "Bowed string simulation using a thermal friction model," *Acta Acustica united with Acustica*, vol. 89, pp. 355–368, 2003.

[65] C. A. Coulomb, "Sur une application des règles de maximis et minimis à quelques problèmes de statique, relatifs à l'architecture [On maximums and minimums of rules to some static problems relating to architecture]," *Mémoires de Mathematique de l'Académie Royale de Science*, vol. 7, pp. 343–382, 1773.

[66] A. Morin, "New friction experiments carried out at metz in 1831-1833," in *Proceedings of the French Royal Academy of Sciences*, 1833, pp. 1–128.

[67] O. Reynolds, "On the theory of lubrication and its application to Mr. Beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil," in *Proceedings of the Royal Society of London*, vol. 40, 1886, pp. 191–203.

[68] R. Stribeck, "Die wesentlichen eigenschaften der gleit- und rollenlager [The essential properties of sliding and roller bearings]," *Zeitschrift des Vereins Deutscher Ingenieure*, vol. 46, no. (38,39), pp. 1342–48,1432–37, 1902.

[69] P. Dahl, "A solid friction model," Technical report, The Aerospace Corporation, 1968.

[70] C. Canudas de Wit, H. Olsson, K. J. rAström, and P. Lischinsky, "Dynamic friction models and control design," *Proceedings of the 1993 American Control Conference*, pp. 1920–1926, 1993.

[71] ——, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

[72] 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.

[73] H. Olsson, "Control systems with friction," Ph.D. dissertation, Lund University, 1996.

[74] J. Gomm, "Passionflower," 2011. [Online]. Available: https://www.youtube.com/watch?v=nY7GnAq6Znw

[75] J. Mayer, "Gravity (Live in L.A.)," 2008. [Online]. Available: https://youtu.be/dBFW8OvciIU?t=284

[76] T. L. Glenn, "Amazing hammered dulcimer musician - joshua messick," 2014. [Online]. Available: https://youtu.be/veuGTnzgNRU?t=215

[77] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, vol. 26, no. 3, 2002.

[78] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx)*, 2015.

[79] exception, "double - and how to use it," 2008. [Online]. Available: http://www.cplusplus.com/articles/Nb07M4Gy/

[80] G. Guennebaud, B. Jacob *et al.*, "Eigen," 2021. [Online]. Available: http://eigen.tuxfamily.org

check whether to sort references or not

Part VII

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 15 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
```

Appendix D

Intuition for the Damping Terms in the Stiff String PDE

This appendix will delve deeper into the damping terms in the PDE of the stiff string presented in Chapter 4, especially the frequency-dependent damping term $2\sigma_1 \partial_t \partial_x^2 u$. Recalling the compact version of the PDE of the stiff string in Eq. (4.5):

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u - 2\sigma_0 \partial_t u + 2\sigma_1 \partial_t \partial_x^2 u \tag{4.1}$$

Consider first the frequency-independent damping term $-2\sigma_0 \partial_t u$. The more positive the velocity $\partial_t u$ is, i.e., the string is moving upwards the more this term applies a negative, or downwards force (/effect) on the string. Vice versa, a more negative velocity will make this term apply a more positive force on the string.

As for the frequency-dependent damping term, apart from the obvious σ_1, the effect of the term increases with an increase of $\partial_t \partial_x^2 u$ which literally describes the 'rate of change of the curvature' of the string.

Let's first talk about positive and negative curvature, i.e., when $\partial_x^2 u > 0$ or $\partial_x^2 u < 0$. Counterintuitively, in the positive case, the curve points downwards. Think about the function $f(x) = x^2$. It has a positive curvature (at any point), but has a minimum. We can prove this by taking $x = 0$ and setting grid spacing $h = 1$.

$$\begin{aligned} \delta_{xx} f(x) &= \frac{1}{h^2} \left(f(-1) - 2f(0) + f(1) \right), \\ &= \frac{1}{1^2} \left((-1)^2 - 2 \cdot 0^2 + 1^2 \right), \\ &= (1 - 0 + 1) = 2. \end{aligned} \tag{4.2}$$

In other words, the second derivative of the function $f(x) = x^2$ around $x = 0$ is positive.

As our term does not only include a second-order spatial derivative but also a first-order time derivative, we are now talking about a positive or negative *rate of change* of the curvature, i.e., when $\partial_t \partial_x^2 u > 0$ or $\partial_t \partial_x^2 u < 0$. A positive rate of change of curvature means that the string either has a positive curvature and is getting more positive, i.e., the string gets more curved over time, or that the string has a negative curvature and is getting less negative, i.e., the string gets less curved or 'loosens up' over time. In the same way, a negative rate of change of curvature means that the string either has a negative curvature and is getting more negative, or that the string has a positive curvature and is getting less positive.

Let's see some examples. Take the same function described before, but now f changes over time, fx. $f(x,t) = tx^2$. When t increases over time, the curvature gets bigger. Repeating what we did above with $x = 0$ and grid spacing $h = 1$, but now with $t = 2$ and step size $k = 1$, but now with a backwards time derivative we get:

$$\begin{aligned}
\delta_{t-}\delta_{xx} f(x,t) &= \frac{1}{kh^2}\Big(\; f(-1,2) - 2f(0,2) + f(1,2) \\
&\qquad\qquad - \Big(f(-1,1) - 2f(0,1) + f(1,1)\Big)\Big), \\
&= \frac{1}{1 \cdot 1^2}\Big(2 \cdot (-1)^2 - 2 \cdot 2 \cdot (0)^2 + 2 \cdot 1^2 \\
&\qquad\qquad - \Big(1 \cdot (-1)^2 - 2 \cdot 1 \cdot (0) + 1 \cdot (1^2)\Big)\Big), \\
&= 2 + 2 - (1 + 1) = 2.
\end{aligned}$$

So the rate of change of the curvature is positive, i.e., the already positively curved function x^2 gets more curved over time.

If the curvature around a point along a string gets more positive (or less negative) over time, the force applied to that point will be positive, effectively 'trying' to reduce the curvature. Vice versa, if the curvature around a point along a string gets more negative (or less positive) over time, the force applied will be negative, again 'trying' to reduce the curvature.

From an auditory point of view, higher curvature generally means higher frequency. As the frequency-dependent damping term reduces curvature along the string it effectively damps higher frequencies.

Part VIII

Papers

Paper Errata

Here, some errors in the published papers will be listed:

Elasto-plastic [C]

- The authors in reference [15] are a bit shuffled.

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:

$$h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}} \tag{0.3}$$

- Unit for membrane tension is $\mathrm{N/m}$.

Dynamic grids [G]

- Reference in intro for 'recently gained popularity' should go to [38]
  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.

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.

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.

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.

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.

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.

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.

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.

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...