The Emulated Ensemble

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

Willemsen, Silvin

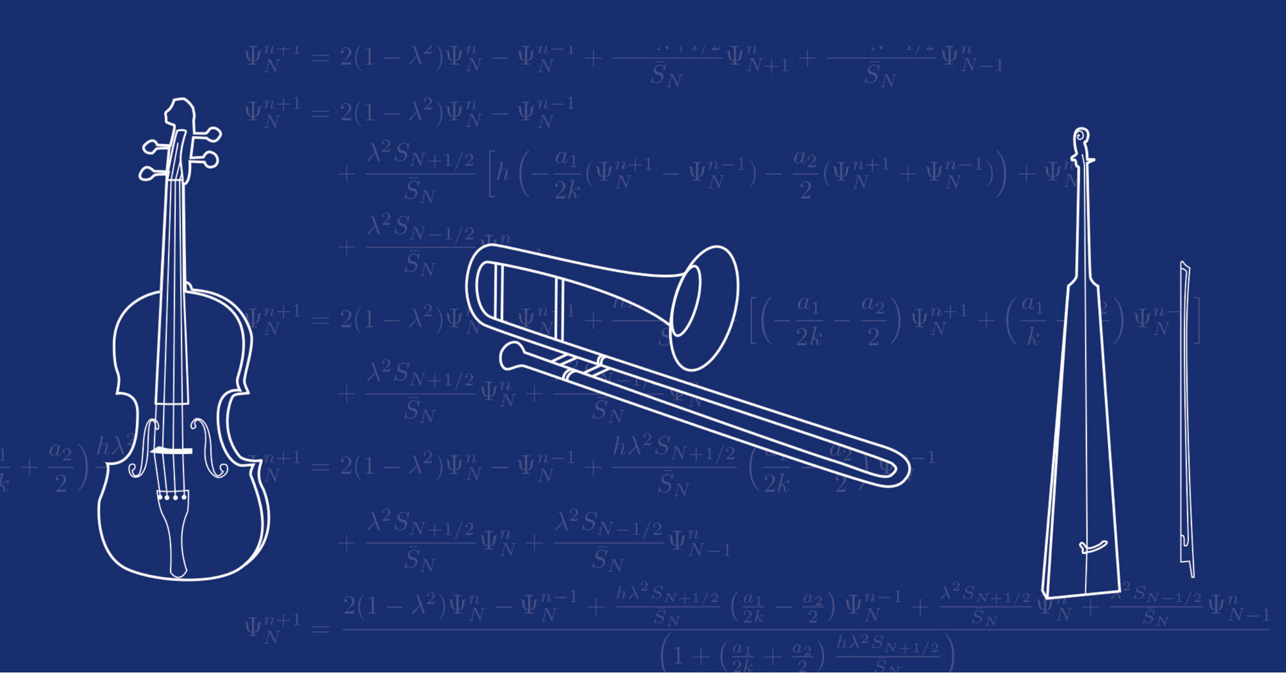
DOI (link to publication from Publisher):
10.54337/aau451025772

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Willemsen, S. (2021). *The Emulated Ensemble: Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods*. Aalborg Universitetsforlag. https://doi.org/10.54337/aau451025772



THE EMULATED ENSEMBLE

REAL-TIME SIMULATION OF MUSICAL INSTRUMENTS USING FINITE-DIFFERENCE TIME-DOMAIN METHODS

BY
SILVINWILLEMSEN

DISSERTATION SUBMITTED 2021

AALBORG UNIVERSITY
DENMARK

The Emulated Ensemble

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

Ph.D. Dissertation
Silvin Willemsen

Dissertation submitted July 31, 2021

Dissertation submitted:	July 31, 2021
PhD supervisor:	Prof. Stefania Serafin Aalborg University
Co-Supervisor (external):	Prof. Stefan Bilbao University of Edinburgh
PhD committee:	Associate Professor Olga Timcenko (chair) Aalborg University Professor Julius O. Smith III Stanford University Professor Augusto Sarti Polytechnic University of Milan
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Architecture, Design and Media Technology

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-976-3

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2021

Author CV

Silvin Willemsen received his BSc. in Industrial Design from Eindhoven University of Technology in 2015 and his MSc. in Sound and Music Computing from Aalborg University in 2017. In 2018, he was appointed as a PhD Stipend at the Department of Architecture, Design and Media Technology at Aalborg University Copenhagen and was affiliated with the Multisensory Experience Lab. During the PhD project he supervised students following the Sound and Music Computing master's programme, and was involved with teaching the courses *Physical Modelling for Sound Synthesis* and *Sound Processing*.

Author CV

Abstract

Over the past few decades, numerous strategies to virtualise traditional instruments have been developed. Although one could create digital musical instruments using pre-recorded samples of their real-life counterparts, the playability will not be captured. Instead, a simulation of the underlying physics of the instrument could be created, and is much more flexible to player interaction. This *physical model* will allow a musician to be much more expressive when playing the digital instrument than if static samples were to be used. Using ad hoc hardware to control the simulation could potentially make the simulated instrument feel identical to the original.

Applications of physical modelling for musical instruments include simulating instruments that are unplayable as they are too rare or vulnerable. A model of the underlying physics of the instrument could potentially resurrect the instrument making it available to the public again. Furthermore, as a simulation is not restricted by the laws of physics, one could extend the possibilities of the original instrument. Properties such as the material or geometry of an instrument could be dynamically changed which broadens the range of expression of the musician. One could even imagine physically impossible musical instruments which still exhibit a natural sound due to the underlying models.

In this project, finite-difference time-domain (FDTD) methods have been chosen, as they have an advantage in terms of generality and flexibility regarding the systems they can model. A drawback of these methods is that they are quite computationally expensive, and although many highly accurate models based on these methods have existed for years, the computing power to run them in real time has only recently become available. The main challenge is thus to run the simulations in real time to allow for proper player interaction.

This thesis presents the development and real-time implementation of various physical models of traditional musical instruments based on FDTD methods. These instruments include the trombone and more obscure instruments such as the hurdy gurdy and the tromba marina. Furthermore, a novel method is presented that paves the way for dynamic FDTD-based musical instrument simulations allowing for physically impossible instrument manipulations. Finally, this work doubles as an aid for beginners in the field of musical instrument simulations based on FDTD methods, and aims to provide a low-entry-level explanation of the literature and theory that the physical models are based on.

Resumé

I løbet af de sidste årtier, er der blevet udviklet adskillige strategier til at lave virtuelle udgaver af traditionelle musikinstrumenter. Selvom digitale musikinstrumenter baseret på traditionelle musikinstrumenter, kan skabes ved hjælp af lydoptagelser af deres virkelige modstykke, er det ofte på bekostning instrumenternes spilbarhed. En anden strategi ville være at implementere en digital simulering af instrumentets underliggende fysik, hvilket ville give en mere fleksibel og naturlig interaktion. Denne digitale simulering, en fysisk model af instrumentet, ville gøre det muligt for en musiker at være mere udtryksfuld når han eller hun spiller på det digitale musikinstrument end med statiske lydoptagelser. Derudover, ved at bruge ad hoc hardware til at styre simuleringen, kunne man potentielt få det digitale musikinstrument til at føles identisk med originalen.

Fysisk modellering af musikinstrumenter kan også anvendes til at simulere musikinstrumenter, der er sjældne eller for sårbare til at må spilles på. Her ville en model af instrumentets underliggende fysik potentielt kunne genoplive instrumentet ved gøre det tilgængeligt og spilbart igen. Ydermere, kunne man forbedre det originale instrument, eftersom en digital simulering ikke er begrænset af fysikkens love. Egenskaber som instruments materiale eller geometri kunne dynamisk ændres og udvide musikerens udtryksmuligheder. Man kunne endda forestille sig fysisk umulige musikinstrumenter, der stadig har en naturlig klang på grund af de underliggende modelleringsprincipper.

Til dette projekt er finite-difference time-domain (FDTD) metoderne blevet valgt som modelleringsteknik, siden disse metoder er generelle og fleksible og derfor har en fordel i forhold til de forskellige typer af systemer som de kan modellere. En ulempe ved FDTD metoderne er at de er beregningstunge, og selvom der har eksisteret nøjagtige modeller baseret på disse metoder i årevis, er computer regnekraften til at køre dem i realtid først blevet tilgængelig for nyligt. Den største udfordring er derfor at køre simuleringerne i realtid og at opnå naturlig interaktion imellem udøveren og simuleringen.

Denne afhandling beskriver udviklingen og implementeringen af forskellige fysiske modeller af traditionelle musikinstrumenter baseret på FDTD metoder i realtid. Disse instrumenter inkluderer trombone og mindre kendte instrumenter som drejelire og tromba marina. Desuden præsenteres en ny metode, der muliggør dynamiske parametre i FDTD-baserede musikinstrumentsimuleringer og tillader instrumentmanipulationer som er umulige i den virkelige verden. Derudover, kan denne afhandling bruges som et hjælpemiddel til begyndere inden for simuleringer af musikinstrumenter, og sigter mod at give en begyndervenlig forklaring af den litteratur og teori, som de fysiske modeller er baseret på.

Preface

This PhD thesis is the product of a 3-year long endeavour on physical modelling musical instruments using finite-difference time-domain (FDTD) methods. The main contributions include several real-time implementations of traditional musical instruments ranging from string instruments to brass instruments. Furthermore, a novel method has been created to change the material properties of the simulations in real time, which allows for physically impossible manipulations of the instrument. This thesis is structured as a collection of papers preceded by an introduction describing the methods on which the publications are based in extended detail. A comprehensive overview of the structure of this thesis appears at the end of Chapter 1.

A personal note

The field of physical modelling for sound synthesis is cross-disciplinary and combines mathematics, physics and computer science. As my background did not include any of these disciplines, the terminology and different notations used by the literature were slightly overwhelming at first.

After overcoming the initial steep learning curve, I discovered that existing work lacks a lot of intuition needed for readers without a background in any of these disciplines. Rather, much of the literature assumes that the reader has a degree in at least one of the aforementioned topics. In his seminal work *Numerical Sound Synthesis*, which is the most complete work to date on physically modelling musical instruments using FDTD methods, Stefan Bilbao says that, in order to read his book, *"a strong background in digital signal processing, physics, and computer programming is essential."* This inspired me to use this opportunity to write a large part as an aid for beginners in the field, while simultaneously relating it to the contributions made during the PhD project. Additionally, I hope that this enhanced level of detail supports the reproducibility of my contributions without overshadowing them.

Acknowledgements

First and foremost, I would wholeheartedly like to thank my supervisor and friend Stefania Serafin, without whom none of this would have been possible. I quickly found out that this PhD project was a "golden ticket" to go in whichever direction I found interesting as long as it was scientifically relevant. She provided me with this opportunity and allowed me to be free in my decisions throughout this project, and I can't thank her enough for that. Also, I would like to thank her for the laughs, drinks, nerdy jokes, and overall good times at the Multisensory Experience (ME) Lab. Working at AAU with Stefania and the entire ME-Lab team has truly been a 'dream job', for which I am extremely grateful.

Next, I would like to thank Stefan Bilbao for his invaluable supervision throughout this PhD project. He has been an incredible mentor throughout the process and a great personal inspiration to me. His critical look raised this thesis and several publications to a higher level. Needless to say, the results of this project would not have been possible without him.

Furthermore, I would like to thank Michele Ducceschi for our collaborations on several papers. His patience and willingness to help have been exceptional, and I hope that our collaboration continues in the future.

In January 2019, I visited the University of Edinburgh and want to thank – in addition to Stefan and Michele – Craig Webb, Brian Hamilton, Charlotte Desvages and Giulia Fratoni for welcoming me into their research group for an amazing two weeks.

Writing this thesis, I received an incredible amount of help from my mother Madeleine Koudstaal, who tirelessly went through every single line of this document, taking out nearly invisible punctuation and spelling errors (including this sentence). Others who helped in this regard are Marius Onofrei, Razvan Paisa, Niels Willemsen, Nikolaj Andersson and Pelle Juul Christensen, and I can not thank them enough for their help in polishing this work.

Next, I would like to thank my colleagues, visitors and friends from the ME-Lab not mentioned before: Ali Adjorlu, Lars Andersen, Nicklas Andersen, Lui Thomsen, Jonas Wang, Simone Spagnol, Emil Høeg, Elisha Anne Teo, Camilla Jaller, Luis Vieira, Cumhur Erkut, Sofia Dahl, Dan Overholt, Niels Nilsson, Jon Bruun-Pedersen, Eva Triantafyllou, Sebastian Boring, Smilen Dimitrov, Rolf Nordahl, James Leonard, Jérôme Villeneuve, Federico Fontana, Romain Michon, Nolan Lem and Nathaly Betancourt. Thanks for the interesting and helpful discussions, the drinks, and overall amazing times at the Lab and AAU.

Furthermore, I would like to thank the co-authors of the publications made over the course of this project: Karolina Prawda, Vesa Välimäki, Anca-Simona Horvath, Mauro Nascimben, Titas Lasickas, Rares Stefan Alecu, Emanuele Parravicini, Stefano Lucato, Jacob Møller Hjerrild and Mads Græsbøll Christensen, and I look forward to future collaborations.

I would also like to take this opportunity to thank my PhD committee: Olga Timcenko, Julius O. Smith III and Augusto Sarti for their time and effort in assessing my PhD project.

This project was financially supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC under project number 86892.

Finally, I would like to thank my family for having always supported the decisions I made and providing the freedom I needed to pursue my dreams. Specifically, I would like to thank Anna Katarina Weber, for her unconditional support over the last few years, including sitting through my try-out lectures and conference presentations, and enduring headaches from my unfinished (and finished) implementations.

Silvin Willemsen
Copenhagen, July 31, 2021

Thesis Details

Thesis title:	The Emulated Ensemble: Real-Time Simulation of Musical Instruments using Finite-Difference Time-Domain Methods
PhD Student:	Silvin Willemsen
PhD Supervisor:	Prof. Stefania Serafin, Aalborg University
Co-supervisor (external):	Prof. Stefan Bilbao, University of Edinburgh

This thesis has been submitted for assessment in partial fulfilment of the PhD degree. The thesis is based on the published scientific papers that are listed below; papers listed under 'Main publications' on the next page have been included as a part of this thesis. Parts of the papers are used directly or indirectly in the main body of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty.

List of Publications

Listed below are the publications made during the PhD project, (co-)authored by the PhD student. These are grouped by: the main publications included in this thesis, papers with a supervisory role, and 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.

List of Abbreviations

Abbreviation	Definition
1D	one-dimensional or one dimension
2D	two-dimensional or two dimensions
3D	three-dimensional or three dimensions
DoF	Degrees of freedom
DSP	Digital signal processing
Eq.	Equation
Eqs.	Equations
FD	Finite-difference
FDTD	Finite-difference time-domain
GUI	Graphical user interface
LSI	Linear shift-invariant
LTI	Linear time-invariant
ODE	Ordinary differential equation
PDE	Partial differential equation
VR	Virtual reality

Contents

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

The earliest musical instruments date back to the start of human civilisation [1]. It has only been over the last sixty years, that technological advances have allowed for the creation of digital versions of traditional musical instruments. At the time of writing, an uncountable number of digital musical instruments exists, including digital keyboards that produce sounds from various real (and non-real) instruments, as well as digital instrument plug-ins used by music producers. Many of these digital instruments are based on samples, or recordings, of their real-life counterparts, while others use computationally efficient methods to generate sounds, some inspired by physical musical instruments. The earliest sound synthesis techniques date back to the late 1950s where Max Mathews proposed a technique called wavetable synthesis [2]. Not long after, in the 1960s, efficient sinusoidal-based and filter-based sound synthesis techniques such as additive synthesis, subtractive synthesis, and FM (frequency modulation) synthesis were invented [3, 4]. The latter became widely popular through the Yamaha DX7 synthesiser created in 1983, that synthesised sounds based solely on this technique [5]. Through a simple change of variables, FM synthesis can generate sounds ranging from brass instruments to drums.

Most of these techniques are referred to as *spectral modelling* methods, where the manipulation of sinusoids or filtering noise produces (harmonic) sounds, which could be perceived by the listener as originating from a physical instrument. This top-down approach which starts at the perception of the listener, has advantages in terms of computational efficiency, but is quite limited by the systems it can model [6].

As computing power increased over the last few decades, using *physical models* rather than samples or spectral modelling methods gained an increased popularity. Physical modelling, in the context of sound and music, is a way to generate sound based on physical processes, including string vibrations in a guitar, air propagation in a trumpet, or even the reflections in a concert hall. When compared to spectral modelling, this is a bottom-up approach that attempts to model the sound from the source.

This work focuses on simulating[1] traditional musical instruments using physical modelling. The interest in physically modelling traditional musical instruments is twofold: 1) sound generation, and 2) understanding of the underlying physical processes. The main focus of this PhD project is the former.

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, where the performer controls the 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 in real time 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.

This chapter continues by giving a brief overview of the history of physical modelling for sound synthesis after which an overview of various modelling techniques will be given. Next, several applications of physical modelling will be presented, which aim to explain *why* musical instrument simulations exist in the first place. Finally, an overview of the objectives and contributions of this PhD project will be given, as well as an overview of this thesis.

1.1 A brief history

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 were 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 one 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 proposed by McIntyre et al. in their 1983 publication

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

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

[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* in the late 1980s and early 1990s in [14, 15] and continued to develop the method [16]. Around the same time, Adrien in [17] and later together with Morrison and Adrien in [18] introduced *modal synthesis*, a way to synthesise the sound of an object 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.

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 [19] and later developed by De Poli and Rocchesso in [20] and is used to structure this thesis. 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 project 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, and the method of excitation greatly influences the sound of the resonator. In the real world, the interaction between the exciter and the resonator is bidirectional. 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 be modelled in this project.

The next section discusses various techniques that can be used to implement the resonator. Details on excitation modelling are left for Part III.

[3] The difference between linear and nonlinear systems is in their response to input level or amplitude. The behaviour of linear systems does not change with the level of the input. Instead, it only scales (linearly) with the input level, i.e., 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 most systems can still be considered linear and their nonlinear effects can be ignored.

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) [1, 21]. Examples of dynamic systems are a guitar string, a drum-membrane, or air propagation in a concert hall; three very different concepts, but can all be 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 [1]. 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, Välimäki et al. in [22] and Smith in [6, 16].

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

Modal Synthesis decomposes a system into a series of uncoupled 'modes of vibration' and can be seen as a physically-based additive synthesis technique. First used in a musical context by Morrison and Adrien in [18], 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 [23, 6]. 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 [24].

Finite-Difference Time Domain (FDTD) methods 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 [25, 26]. An extensive overview of FDTD methods in the context of sound synthesis is given by Bilbao in [21]. Although computationally expensive, especially when working with higher-dimensional systems, this technique could potentially 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 [14, 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, such as the waveguide mesh [27], but is superior in efficiency when used in the one-dimensional case [22]. Some authors have combined DWGs with FDTD methods (such as in [28, 29]) to accurately model nonlinear behaviour while maintaining a 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 [30, 11, 31] it is currently being further developed by Leonard and Villeneuve in a real-time, interactive environment [32, 33].

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 number 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. Moreover, FDTD methods allow for various PDEs, e.g. a violin body and four strings, to be connected in a fairly straightforward manner. DWGs, for example, assume a travelling wave solution, which makes complex nonlinear effects extremely hard to model using this technique. To use modal synthesis for modelling a PDE, it requires the system to have a 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 [34].

The main drawback of FDTD methods is the fact that they require great attention to numerical stability of the solution [21]. 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 thesis.

A final drawback of using FDTD methods is that – especially for higher-dimensional systems – they are much more computationally expensive than other methods, such as DWGs or modal synthesis techniques. The bright side

[4] The author learned the hard way that one should always implement a limiter when working with real-time physical models to avoid dangerously loud sounds due to unstable implementations.

– if one believes in Moore's law [35] – is that it can be assumed that computing power will continue to increase and that within several years, running high-quality real-time simulations of musical instruments based on FDTD methods will not be an issue.

1.4 Applications of physical modelling

It might not be clear why one would go through the hassle of modelling a musical instrument. Why could one 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.

1.4.1 Samples vs. physical modelling

Despite the existence of many techniques to simulate musical instruments mentioned in the previous section, the bulk of the currently available digital musical instruments are still based on samples. This is mainly due to the computational power needed to generate sounds, as opposed to simple playback of recordings. Furthermore, digital musical instruments based on samples have an optimally realistic sound. As the output of the digitised instrument is exactly that of the original instrument, the digital version should thus sound indistinguishable from the original.

That said, it can be argued that these are the only advantages of using samples over physical models in this context. 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, and so on. Even if one accepts this, capturing 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.[5]

Using physical models to simulate the musical instrument instead, allows the sounds to be generated 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.

[5] This was actually done in 2019 for four century-old bowed-string instruments in the Italian city of Cremona. The recordings lasted five weeks, six days a week, eight hours a day, and required the city center to be as quiet as possible to record the instruments [36, 37].

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

Applications of physical modelling are not limited to old or rare instruments. Popular musical instruments also 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 [6]). 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?

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' [38]. Several virtual string instruments and different electric pianos have been made real-time by Pfeifle and Bader in [39, 40, 41]. The authors 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 [42] 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 [34, 43].

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 [44]. Most graphics applications are made with a temporal sample rate (mostly referred to as frames per second (FPS)) of around 60 Hz [45], which is orders of magnitude smaller than the auditory sample rate. For comparison, for a commonly used auditory sample rate of 44100 Hz, running a simulation for audio requires 735x as many iterations as if this simulation was done for graphics only.

The main objective of this project is to implement physical models using FDTD methods in real time without the need of special hardware, i.e., on a regular personal computer or laptop. The objective is not to renew the underlying models themselves, but to create novel combinations of existing models 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 and hurdy gurdy presented in paper [A], the tromba marina presented in paper [D] and the trombone presented in paper [H], all implemented in real time using FDTD methods. An extended summary of these papers can be found in Part V and the complete papers are included in Part VII. Details on the real-time implementation can be found in Chapter 13.

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, see e.g. [38, 46], 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.

Indeed, during this PhD project, 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 elaborated on in Chapter 12.

How can the now-virtual instruments be controlled in an expressive way?

A substantial challenge in the field of 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 present in (nearly) all physical musical instruments.

The last objective of this PhD project is thus to find ways to control the instrument simulations in an expressive way. Over the course of this PhD project, the Sensel Morph, or Sensel for short, has been used extensively [47]. The Sensel is a controller containing ca. 20,000 pressure sensors that allow for highly expressive control of the instruments. This controller has been used in papers [A], [B], [C] and [D].

Although the Sensel 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, a virtual-reality (VR) implementation of the tromba marina was controlled by the PHANTOM Omni, which is a six-degrees-of-freedom (6-DoF) haptic device [48]. The controller contains a hand-held pen-like object that is attached to a robotic arm, and can be linked to a virtual environment to provide force and vibrotactile feedback through the arm, based on this environment. This project is presented in paper [E].

1.6 Thesis outline

The thesis uses a ‘collection of papers’ format, and is divided into several parts. See Figure 1.1 for a visual overview of the thesis structure. Parts I, II, III and IV are used as an introduction for the main contributions of the PhD project in Part V, and – with the exception of Chapter 8 – do not contain any of the contributions made during this project. Much effort has been put in explaining the existing methods and models from the literature used in this work, in a way that is slightly more in-depth and pedagogical than might be common for a PhD thesis, while simultaneously relating this existing work to the contributions of this project. It is the hope of the author, that going this extra mile could make this thesis (and these parts in particular) a contribution in itself: To put this research field into reach for beginners in the area of physical modelling for sound synthesis using FDTD methods, without the need of much experience in the fields of physics, mathematics or computer science. To this end, although a ‘collection of papers’ format has been used, the introductory part has been written in an extended and detailed form.

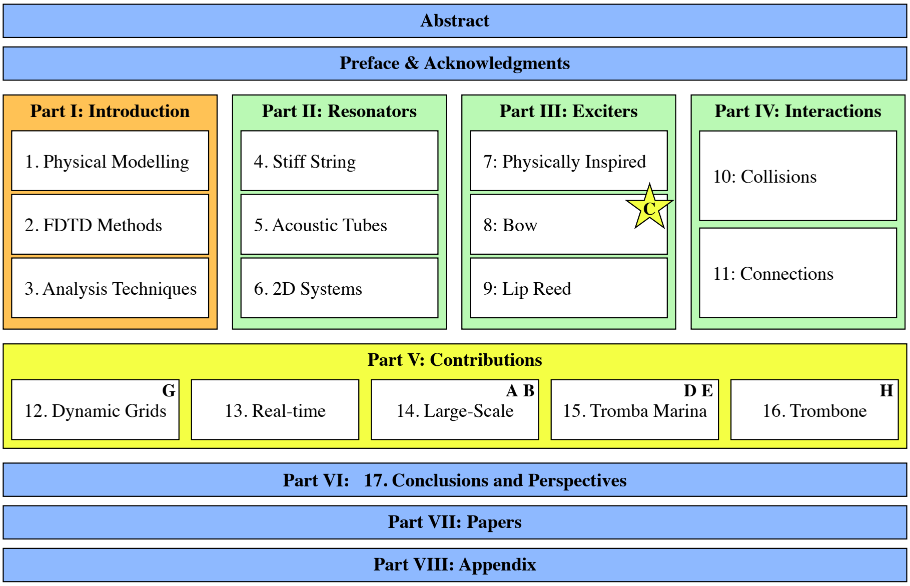


Fig. 1.1: The outline of this thesis. The contributions made throughout the PhD project are marked in yellow. Most are collected in Part V, though the novel work done on the bow will already appear in Chapter 8. The parts marked in green, describe the physical models on which the contributions are based. The basics of the methods used for these models are introduced in Part I marked in orange. The chapters that can be seen as an extended summary of the papers in Part VII are indicated by the letter of the respective paper.

Part I: Introduction

This part introduces the field of physical modelling for musical instruments in Chapter 1, by giving a brief history of the field and providing background for the project. Furthermore, the objectives and main contributions of the PhD project are detailed. Chapter 2 provides a thorough introduction to FDTD methods, using simple sound-generating systems as examples, after which Chapter 3 introduces several analysis techniques in a tutorial-like fashion.

Part II: Resonators

The resonator component of a musical instrument, as introduced in Section 1.2, can – for most instruments – be further decomposed into more basic resonators. In order to model the violin, for example, one can decompose the entire resonator into four strings and its body. This part presents the various resonators used for the contributions: Chapter 4 presents a model for the stiff string, Chapter 5 introduces acoustic tubes that can be used to model brass instruments, and Chapter 6 introduces two-dimensional systems such as membranes and plates which can be used to simulate simplified instrument bodies.

Part III: Exciters

As stated in Section 1.2, the excitation greatly determines the behaviour of the resonator. This part presents various ways in which the resonators introduced in Part II can be excited. Chapter 7 introduces physically inspired excitations as an easy way to excite the resonators, Chapter 8 introduces the bow and presents the contribution made in paper [C], and finally, Chapter 9 presents the lip reed used to excite brass instruments.

Part IV: Interactions

To properly model a full instrument, the interactions between the various resonators in isolation must be taken into account. This part describes two different ways that the resonators can interact with each other: collisions between various resonators are presented in Chapter 10 and connections between them in Chapter 11.

Part V: Contributions

This part contains extended summaries of the main contributions of the PhD project. Chapter 12 summarises paper [G] and extends it by providing some implementation details and design considerations. Chapter 13 explains the considerations necessary for real-time implementation of physical models. An extended summary of papers [A] and [B] is provided in Chapter 14. Papers [D] and [E] are summarised in Chapter 15 and are extended with details on the implementation, and finally, paper [H] is summarised in Chapter 16, and is extended with design considerations and details on the implementation.

Part VI: Conclusions and Perspectives

Chapter 17 concludes by providing a summary of the thesis. Furthermore, perspectives for the future and possible continuations of this work are given as well.

Finally, **Part VII: Papers** contains the main publications made over the course of this PhD project and an appendix appears in **Part VIII: Appendix** including (among other topics) additional information on matrices, code examples, and derivations.

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. Furthermore, some code examples using the `MATLAB` programming language [49] will be used. Unless denoted otherwise, the theory presented in this chapter and the notation have been taken from [21].

2.1 Differential equations

Differential equations can be 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 (see e.g. [50]) 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 the 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

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

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 dots above the function to denote time-derivatives. For this reason, Newton's notation will be used for ODEs in isolation. The drawback of this notation is that it can 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

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

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). Also, it allows for more compactness when creating bigger operators with multiple (connected) systems (see e.g. Chapter 15). Moreover, state-of-the-art literature in the field of FDTD methods for sound synthesis use this notation as well [23].

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.[1]

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) are 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 have not been included in this project.

Work has been done on strings with dual (transverse) polarisation by Desvages [51] and Desvages and Bilbao [52] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are coupled, can be found in [21, 53]. In [32], Villeneuve and Leonard present a mass-spring network where the state of every individual mass has 3 translational DoF. Due to these additional DoF, these networks do capture the aforementioned effects, but greatly increase the computational complexity of the models.

[1] In this work, '1D' and '2D' will also be used to abbreviate 'one dimension' and 'two dimensions'.

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

Notation

When describing the state of a system, the spatial dimensions it is distributed over appear 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 number 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 is defined for $t \geq 0$, meaning that the system is defined for non-negative time. This will be the case for all systems presented in this work.

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 to be among the most general and flexible techniques in terms of the systems they can model, and 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 Chapter 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. See Figure 2.1. In the following, for generality and ease of explanation, a 1D system will be used, and (again) the theory and notation follows [21].

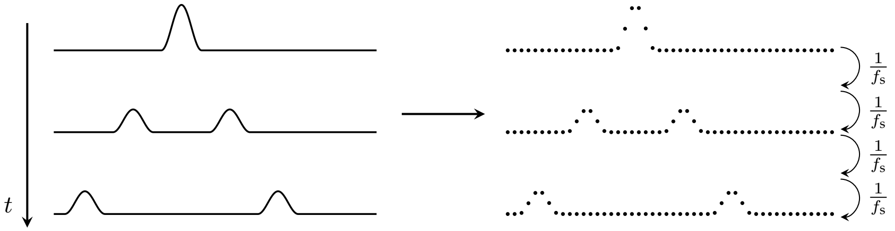


Fig. 2.1: A continuous PDE is discretised to a FD scheme. In a PDE, time passes continuously, whereas for a FD scheme, time passes in finite increments with a duration of the reciprocal of the sample rate f_s.

2.2.1 Grid functions

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 a state variable $u = u(x, t)$ 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, or *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 is the sample rate that 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 $n \in \mathbb{N}^0$.[2] 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

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

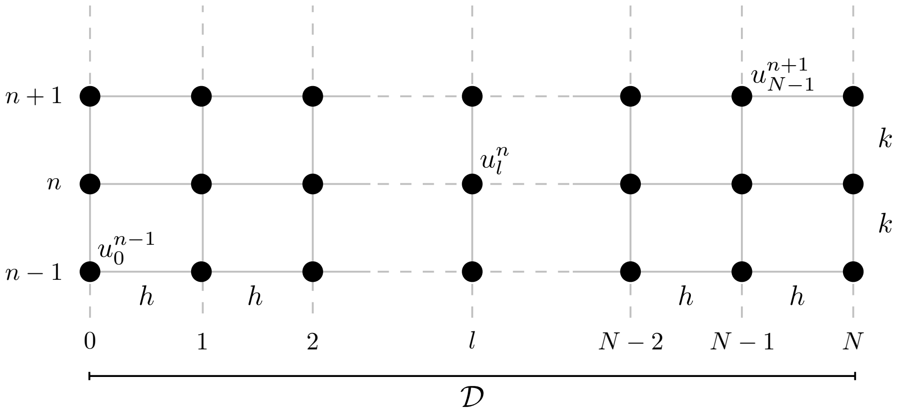


Fig. 2.2: The spatio-temporal grid that appears when a 1D system $u(x,t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n. The spatial domain $\mathcal{D}$ is divided into N intervals of length h and spatial range of interest $l = \{0, \dots, N\}$. Time is subdivided into time steps 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.

end points. The spatial range of interest then becomes $l \in \{0, \dots, N\}$ and the total number of grid points is $N+1$, which is one more than the number of intervals.

To summarise, for a 1D system:

$$u(x,t) \approxeq u_l^n, \quad \text{with} \quad x = lh \quad \text{and} \quad t = nk,$$

$$l \in \{0, \dots, N\} \quad \text{and} \quad n \in \mathbb{N}^0.$$

2.2.2 Finite-difference operators

Now that the state variable has a discrete counterpart, it leaves the derivatives to be discretised, or approximated. First, *shift operators* are introduced, which can be applied to a grid function and 'shift' its indexing, either temporally or spatially. These are denoted by e_s, where subscript s denotes the type and direction of the shift. 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

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

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 in three different ways. The forward, backward and centred difference operators are

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

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

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

and all approximate the first-order time derivative of u. Note that the centred difference has a division by $2k$ as the time difference between $n+1$ and $n-1$ is, indeed, twice the time step. Figure 2.3a shows the *stencils* of the operators introduced above. A stencil shows the grid points needed to perform the operation of a FD operator.

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

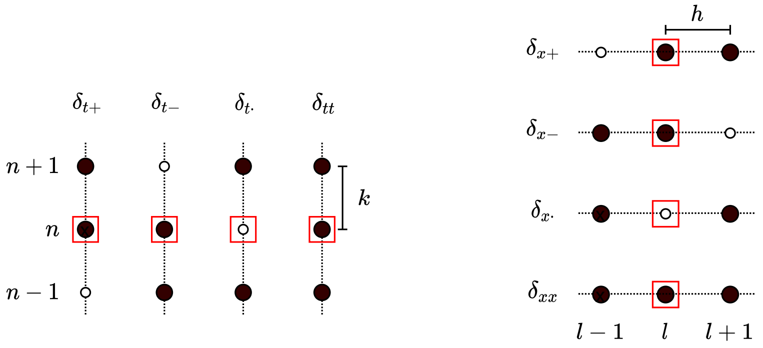
$$\partial_x \approxeq \begin{cases} \delta_{x+} \triangleq \frac{1}{h}\left(e_{x+} - 1\right), & (2.5a) \\ \delta_{x-} \triangleq \frac{1}{h}\left(1 - e_{x-}\right), & (2.5b) \\ \delta_{x\cdot} \triangleq \frac{1}{2h}\left(e_{x+} - e_{x-}\right), & (2.5c) \end{cases}$$

and when applied to u_l^n are

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

Higher-order differences can be approximated through a composition of first-order difference operators where their definitions are multiplied.[3] The second-

[3] 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.

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 be done similarly 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 2.3b shows the stencils of the spatial operators introduced above.

Also useful are averaging operators, all of which approximate the identity 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 Eq. (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. Also notice that the centred averaging operator does not have an extra division by 2 as in Eq.

(2.3c). Applied to u_l^n, the definitions in Eq. (2.9) become

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

Similarly, spatial averaging operators are

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

and when applied to u_l^n

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

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

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

The stencils of the averaging operators follow the same pattern as the difference operators.

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 [21], 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 $k = 1/f_\text{s}$).

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

$$\begin{aligned}
u(t-k) &= u(t) + (-k)\dot{u} + \frac{(-k)^2}{2}\ddot{u} + \frac{(-k)^3}{6}\dddot{u} + \mathcal{O}(k^4), && (2.20)\\
\frac{u(t-k) - u(t)}{k} &= -\dot{u} + \frac{k}{2}\ddot{u} - \frac{k^2}{6}\dddot{u} + \mathcal{O}(k^3), &&\\
\delta_{t-}u(t) &= \dot{u} + \mathcal{O}(k). && (2.21)
\end{aligned}$$

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

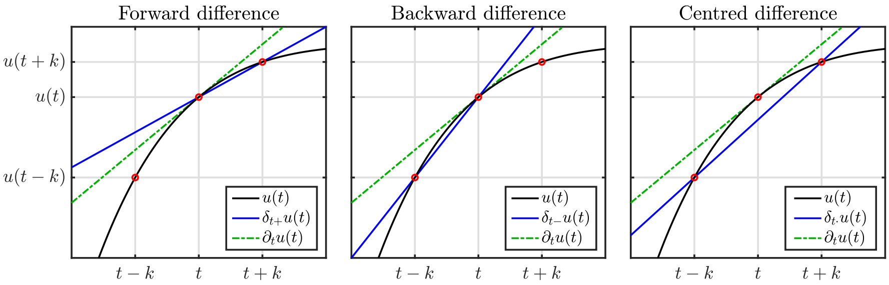


Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. The centred difference operator much more closely approximates the derivative, or the slope, of u at t when compared to the forward and backward difference operators.

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

$$\begin{aligned}
u(t+k) - u(t-k) &= 2k\dot{u} - \frac{2k^3}{6}\dddot{u} + 2\mathcal{O}(k^5),\\
\frac{u(t+k) - u(t-k)}{2k} &= \dot{u} + \mathcal{O}(k^2),\\
\delta_{t\cdot}u(t) &= \dot{u} + \mathcal{O}(k^2),
\end{aligned} \tag{2.23}$$

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

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

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

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:[4]

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

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

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

$$\delta_{t\pm} = \pm\frac{2}{k}(\mu_{t\pm}-1), \tag{2.27d}$$

$$\mu_{t\cdot} = k\delta_{t\cdot}+e_{t-}. \tag{2.27e}$$

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, the current section follows the discretisation process to a FD scheme using the operators described in Section 2.2.2. Finally, the scheme

[4] The ± operators in a single equation are either all '$+$' or all '$-$' and are used for a more compact notation.

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) also referred to as the spring stiffness. An alternative way to write Eq. (2.28) is

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

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.

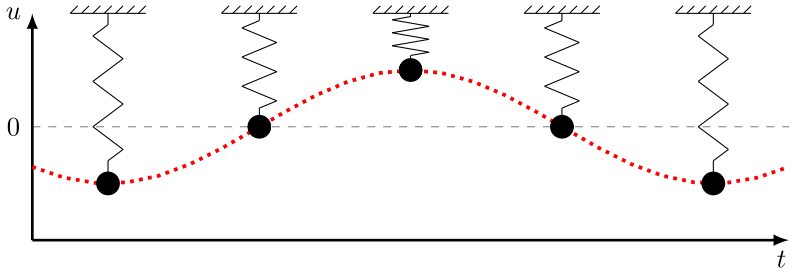


Fig. 2.5: Mass-spring system over time. The system follows a harmonic (sinusoidal) motion.

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*.

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 $\ddot{u}$ $\mathrm{m/s}^2$. 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 yield the equation for the mass-spring system in (2.28).

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

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

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 for the discrete initial velocity. 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:

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

In short, setting $u^0 = u^1 = 1$ 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

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

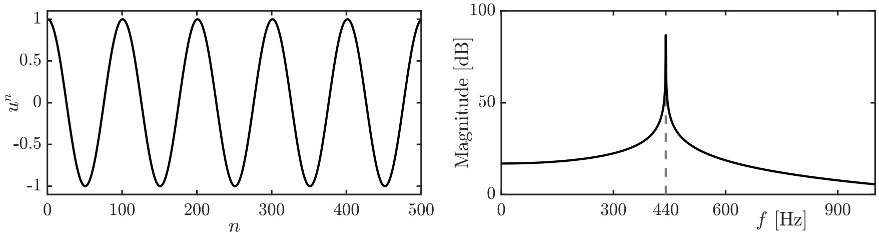


Fig. 2.6: The time domain and frequency domain output of a mass-spring system with $f_0 = 440$ Hz.

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

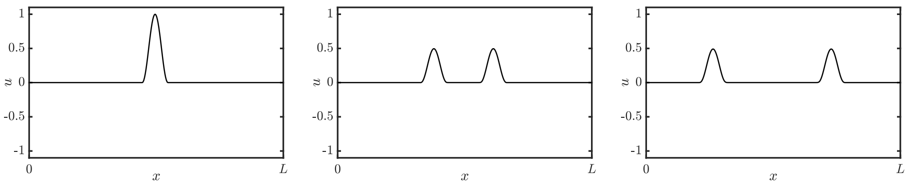
Arguably the most important PDE in the field of physical modelling for sound synthesis is the 1D wave equation. It can be used to describe transverse vibration in an ideal string, longitudinal vibration in an ideal bar or the pressure in an acoustic tube (see Chapter 5) [21]. Although the behaviour of this equation alone does not appear in the real world as such – as no physical system is ideal – it is extremely useful as a test case and a basis for more complicated models.

2.4.1 Continuous time

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). If the PDE is used to model an ideal string, the wave speed can be defined as $c = \sqrt{T/\rho A}$, with tension T (in N), material density ρ (in kg/m^3) and cross-sectional area A (in m^2). If instead, it is used to model pressure in an acoustic tube c is the speed of sound in air. 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) with $c \approx 127$ m/s.

Intuition

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

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

How the 1D wave equation relates to Newton's second law, becomes apparent by slightly rewriting Eq. (2.38). Recalling the definition of c for an ideal string, one can rewrite the 1D wave equation to

$$\rho A \partial_t^2 u = T \partial_x^2 u,$$

where ρA describes the *mass per unit length* of the string. As the forces present in the system act on infinitesimally small portions of the string Newton's second law appears by a multiplication of dx

$$\underbrace{\rho A\partial_t^2 u dx}_{ma} = \underbrace{T\partial_x^2 u dx}_{F},$$

where $\rho A dx$ is the mass of a (tiny) portion of the string of length dx (in m), $\partial_t^2 u$ is the acceleration of that portion and $T\partial_x^2 u dx$ describes the force acting on that portion, yielding Newton's second law.

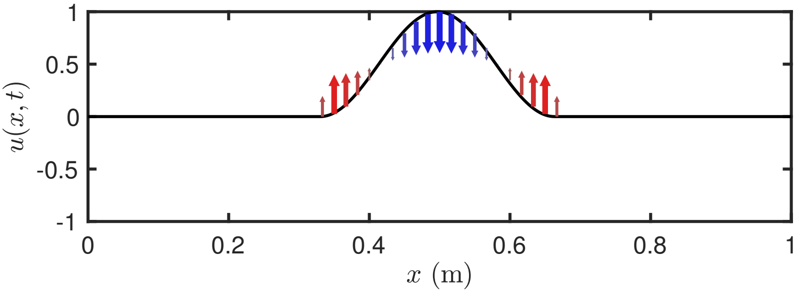


Fig. 2.8: The forces acting on the 1D wave equation described by $u(x,t)$ due to curvature. The arrows indicate the direction and magnitude of the force along the system, and consequently the acceleration through Eq. (2.38).

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

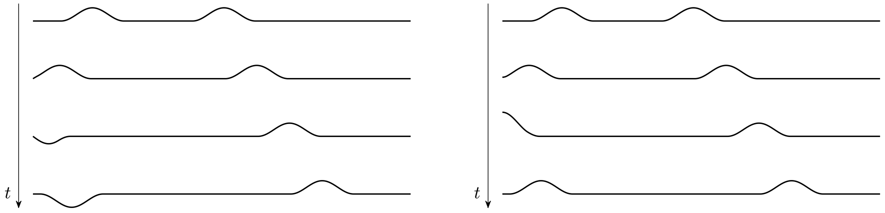
$$u(0,t) = u(L,t) = 0 \quad \text{(Dirichlet, fixed)}, \tag{2.39a}$$

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

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.

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

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



(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

As much of this thesis follows Bilbao's *Numerical Sound Synthesis* [21], 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 [21] and much of the literature published around that time (e.g. [54, 53]) and can be useful to reduce the number of parameters used to describe a system.

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

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

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 parametrise 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 in this thesis.

2.4.2 Discrete time

Coming back to the PDE presented in Eq. (2.38), one continues by finding a discrete-time approximation for it. As explained in Section 2.2.1, a continuous state variable $u = u(x,t)$ can be discretised using $x = lh$ with grid spacing h (in m) and $t = nk$ with time step k (in s). The grid function u_l^n approximating u can then be indexed by spatial index $l \in \{0, \ldots, N\}$ with number of intervals between the grid points N and temporal index $n \in \mathbb{N}^0$. Continuing with the approximations of the derivatives in the 1D wave equation, the most straightforward discretisation of Eq. (2.38) is the following FD scheme

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

Other schemes exist (see e.g. [21]), but are excluded as they have not been used in this project. 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 [55]

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

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 points of 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}.

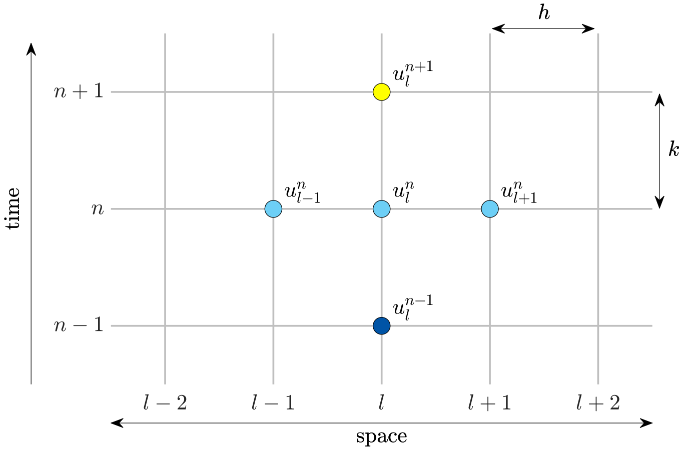


Fig. 2.10: The stencil, or region of operation, for the FD scheme in (2.42). The time steps of the various grid points are colour-coded by yellow ($n+1$), light blue (n) and dark blue ($n-1$).

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 conditions in Eq. (2.39). Discretising these (using the most accurate centred spatial difference operator for the Neumann condition) yields

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

$$\delta_{x\cdot} u_0^n = \delta_{x\cdot} u_N^n = 0 \quad \text{(Neumann, free)}. \tag{2.48b}$$

If Dirichlet boundary conditions are used, the states of the boundary points will always be zero and can therefore 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:

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

At the boundaries, the update equation in Eq. (2.44) will then have 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

This section provides information on the excitation and output of the system. See Appendix C.2 for a MATLAB implementation of the 1D wave equation.

Excitation

A simple way to excite the system is to initialise the state using a raised cosine, or Hann window. More information on this will be given in Chapter 7, 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_0 and width w (in 'number of grid spacings') from which the start index l_s and end index l_e can be calculated, according to

$$l_\text{s} = l_0 - \lfloor w/2 \rfloor \quad \text{and} \quad l_\text{e} = l_0 + \lfloor w/2 \rfloor, \tag{2.51}$$

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

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

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` (in grid points).[5]

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 the given sample rate f_s. An example using the parameters in Table 2.1 and Dirichlet boundary conditions is shown in Figure 2.11.

Name	Symbol (unit)	Value
User-defined parameters		
Length	L (m)	1
Wave speed	c (m/s)	1470
Sample rate	f_s (Hz)	44100
Derived parameters		
Fundamental frequency	f_0 (Hz)	735
No. of intervals	N (-)	30
Time step	k (s)	$\approx 2.27 \cdot 10^{-5}$
Grid spacing	h (m)	≈ 0.033
Courant number	λ (-)	1
Excitation and output		
Center location	l_0 (-)	$0.2N$
Width	w (-)	4
Output location	l_out	3

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

As can be seen from Figure 2.11, the output of the 1D wave equation contains many peaks in the frequency spectrum on top of the fundamental frequency. These are called *harmonic partials* or *harmonics* for short and arise

[5] This slight discrepancy requires to use `hann(w+1)` in order to get the same result as Eq. (2.52).

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.[6]

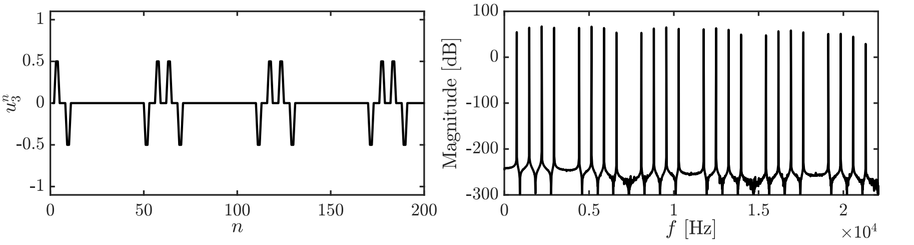


Fig. 2.11: The time-domain and frequency domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_\text{s} = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_0 = 0.2N$) and $w = 4$ and the output is retrieved at $l_\text{out} = 3$.

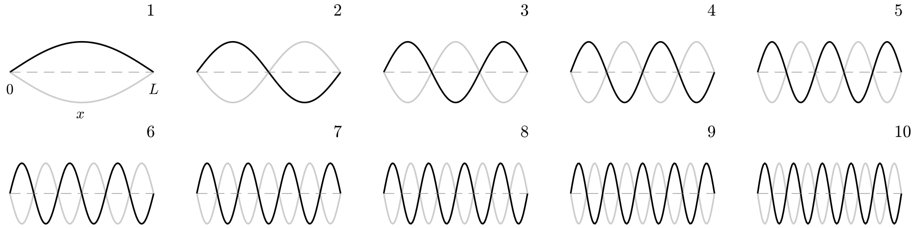


Fig. 2.12: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions defined for $x \in [0, L]$ (only shown for mode 1). The modes are normalised to have the same amplitude and vibrate at their respective modal frequencies with the extremes indicated by the black and the grey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

The number of modes present in the continuous PDE of the 1D wave equation is theoretically infinite. The number 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 $m \in \{1, \dots, N-1\}$ for Dirichlet

[6] Modes of the vibrating string were first discovered by Sauveur in 1701 who said that *"especially at night"* he observed *"other small sounds"* on top of the fundamental frequency and coined the terms 'node' and 'harmonic' [56].

and $m \in \{0, \dots, N\}$ when using 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 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 $1/5^{\text{th}}$ of 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 2nd 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.

2.4.4 Stability and simulation quality

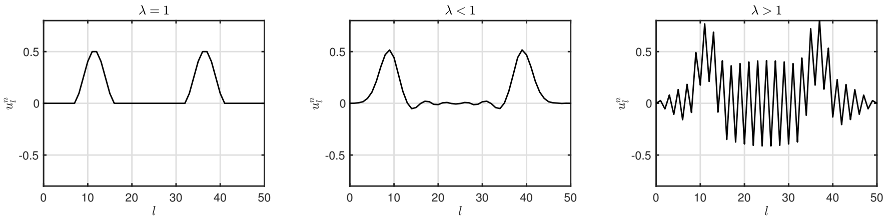
As shown in Eq. (2.46), the Courant number needs to abide the CFL condition in order for the scheme to be stable. A system is regarded *unstable* if it exhibits (exponential) unbounded growth. If Neumann boundary conditions (free) are used, it is possible that the system drifts off over time. This does not mean that the system is unstable, it is actually entirely physically possible.[7]

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 *numerical dispersion*. Dispersion is a phenomenon where some frequencies travel faster through a medium than others, which is desired in some models (see e.g. 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.

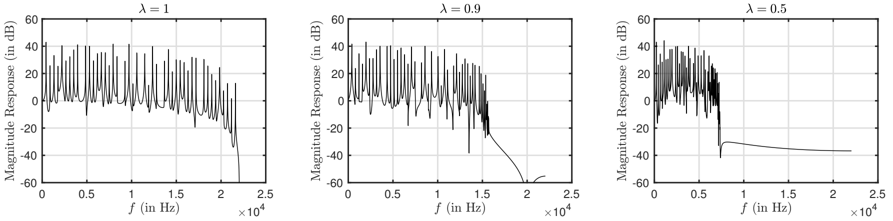
Finally, if $\lambda > 1$ the system becomes unstable. An example is shown in Figure 2.13c. Unstable behaviour usually comes in the form of high frequencies

[7] 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.

(around the Nyquist frequency of $f_s/2$) growing without bounds.

In what situation would the stability condition then 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, the system should produce a fundamental frequency of $f_0 = 750$ Hz which requires a wave speed of $c = 1500$ m/s according to Eq. (2.40). If a sample rate of $f_\text{s} = 44100$ Hz is used, and recalling that $k = 1/f_\text{s}$, these values can be filled into Eq. (2.47) satisfied with equality, and yields $h \approx 0.034$. Dividing the length by the grid spacing, yields $L/h = 29.4$, 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), yields $N = 29$. If h is not recalculated according to Eq. (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 the condition in Eq. (2.46), and the decrease in bandwidth and quality will be perceptually irrelevant – or at the very least, less perceptually relevant than the change in f_0 if h is not recalculated.

Possible solution

One of the main contributions of the PhD project is published in paper [G] and summarised in Chapter 12, 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.

Intuition

It might not be immediately clear why a too low value for h might cause instability. Some intuition is provided in [21, Fig. 6.9], but here, an alternative way to see this is given.

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 2.1 as an example, $N = 30$ and if $\lambda = 1$, it takes exactly 30 samples, or iterations of Eq. (2.44), 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 FD scheme in Eq. (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 which 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 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.

Chapter 3

Analysis Techniques

This chapter provides some useful techniques to analyse FD schemes. Techniques to analyse PDEs also exist, but this work is more practically oriented, and will focus on 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 in a FDTD context and other mathematical tools, this chapter continues to introduce

- *Frequency domain analysis*, which can be used to determine stability conditions of (linear and time/shift-invariant) FD schemes,
- *Energy analysis*, which can both be 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.

3.1 Matrices in a FDTD context

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

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \tag{3.1}$$

[1] Appendix B provides some basic knowledge on matrices and linear algebra for those unfamiliar with this.

Expanding this operation results in

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} a_{00}u_0 + a_{01}u_1 + \ldots + a_{0N}u_N \\ a_{10}u_0 + a_{11}u_1 + \ldots + a_{1N}u_N \\ \vdots \\ a_{N0}u_0 + a_{N1}u_1 + \ldots + a_{NN}u_N \end{bmatrix}}_{\mathbf{w}} \tag{3.2}$$

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

3.1.1 FD 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, as well as the boundary conditions. Not assuming a specific size for now, the FD operators in (2.5) can be written in matrix form according to

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

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

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

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

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

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

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

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

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

and has the following special property

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

3.1.2 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

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 1 & -2 \end{bmatrix}. \tag{3.3}$$

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 to 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

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} -2 & 2 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 2 & -2 \end{bmatrix}, \tag{3.4}$$

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

$$\frac{1}{k^2}\left(\mathbf{u}^{n+1} - 2\mathbf{u} + \mathbf{u}^{n-1}\right) = c^2\mathbf{D}_{xx}\mathbf{u}^n, \tag{3.5}$$

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

$$\mathbf{u}^{n+1} = (2\mathbf{I} + c^2k^2\mathbf{D}_{xx})\mathbf{u}^n - \mathbf{u}^{n-1}. \tag{3.6}$$

The identity matrix is necessary here for correct matrix addition.

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 [21].

3.2.1 Inner product

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

$$\langle f, g\rangle_{\mathcal{D}} = \int_{\mathcal{D}} fg dx \quad \text{and} \quad \|f\|_{\mathcal{D}} = \sqrt{\langle f, f\rangle_{\mathcal{D}}}. \tag{3.7}$$

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 state variables with a derivative applied to it (such as $\partial_t u(x,t)$).

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

$$\langle f_l^n, g_l^n\rangle_d = \sum_{l=0}^{N} h f_l^n g_l^n, \tag{3.8}$$

where the multiplication by h is the discrete counterpart of dx in the continuous definition in (3.7). Also useful are the primed inner product

$$\langle f_l^n, g_l^n\rangle_d' = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{h}{2} f_0^n g_0^n + \frac{h}{2} f_N^n g_N^n, \tag{3.9}$$

and the more general weighted inner product

$$\langle f_l^n, g_l^n\rangle_d^{\epsilon_{\text{l}},\epsilon_{\text{r}}} = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{\epsilon_{\text{l}}}{2} h f_0^n g_0^n + \frac{\epsilon_{\text{r}}}{2} h f_N^n g_N^n, \tag{3.10}$$

where free parameters $0 < \epsilon_{\text{l}}, \epsilon_{\text{r}} \leq 2$ scale the boundary points of the regular inner product. Naturally, if $\epsilon_{\text{l}} = \epsilon_{\text{r}} = 1$, Eq. (3.10) reduces to Eq. (3.9), and if $\epsilon_{\text{l}} = \epsilon_{\text{r}} = 2$, (3.10) reduces to (3.8).

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 only be apparent when actually performing an energy analysis (see e.g. 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. (3.7), integration by parts is defined as

$$\langle f, \partial_x g\rangle_{\mathcal{D}} = -\langle \partial_x f, g\rangle_{\mathcal{D}} + fg|_0^L \tag{3.11}$$

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

In discrete time, using the same two (1D) functions as before: f_l^n and g_l and are defined for $l \in d$ with discrete domain $d = \{0, \dots, N\}$. Then, using the discrete inner product in Eq. (3.8), two variants of summation by parts are defined as

$$\langle f_l^n, \delta_{x-} g_l^n\rangle_d = -\langle \delta_{x+} f_l^n, g_l^n\rangle_d + f_{N+1}^n g_N^n - f_0^n g_{-1}^n, \tag{3.12a}$$
$$\langle f_l^n, \delta_{x+} g_l^n\rangle_d = -\langle \delta_{x-} f_l^n, g_l^n\rangle_d + f_N^n g_{N+1}^n - f_{-1}^n g_0^n. \tag{3.12b}$$

A derivation of Eq. (3.12a) is given in Appendix F.1. As in the case of integration by parts in Eq. (3.11), the process of summation by parts causes the derivative to be applied to the other function and causes the sign of the resulting inner product to change. 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 $\underline{d} = \{0, \dots, N-1\}$, $\overline{d} = \{1, \dots, N\}$ and $\underline{\overline{d}} = \{1, \dots, N-1\}$. The following identities can be shown to hold

$$\langle f_l^n, \delta_{x-} g_l^n\rangle_d = -\langle \delta_{x+} f_l^n, g_l^n\rangle_{\underline{d}} + f_N^n g_N^n - f_0^n g_{-1}^n, \tag{3.13a}$$
$$\langle f_l^n, \delta_{x+} g_l^n\rangle_d = -\langle \delta_{x-} f_l^n, g_l^n\rangle_{\overline{d}} + f_N^n g_{N+1}^n - f_0^n g_0^n, \tag{3.13b}$$

and, using the primed inner product in Eq. (3.9),

$$\langle f_l^n, \delta_{x-} g_l^n\rangle_d' = -\langle \delta_{x+} f_l^n, g_l^n\rangle_{\underline{d}} + f_N^n \mu_{x-} g_N^n - f_0^n \mu_{x-} g_0^n, \tag{3.14a}$$
$$\langle f_l^n, \delta_{x+} g_l^n\rangle_d' = -\langle \delta_{x-} f_l^n, g_l^n\rangle_{\overline{d}} + f_N^n \mu_{x+} g_N^n - f_0^n \mu_{x+} g_0^n, \tag{3.14b}$$

or the more general weighted inner product in Eq. (3.10)

$$\begin{aligned}\langle f_l^n, \delta_{x-} g_l^n\rangle_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}} = &-\langle \delta_{x+} f_l^n, g_l^n\rangle_{\underline{d}} + f_N^n g_{N-1}^n - f_0^n g_0^n \\ &+ \frac{\epsilon_\mathrm{r}}{2} f_N^n (g_N^n - g_{N-1}^n) + \frac{\epsilon_\mathrm{l}}{2} f_0^n (g_0^n - g_{-1}^n),\end{aligned} \tag{3.15a}$$

$$\begin{aligned}\langle f_l^n, \delta_{x+} g_l^n\rangle_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}} = &-\langle \delta_{x-} f_l^n, g_l^n\rangle_{\overline{d}} + f_N^n g_N^n - f_0^n g_1^n \\ &+ \frac{\epsilon_\mathrm{r}}{2} f_N^n (g_{N+1}^n - g_N^n) + \frac{\epsilon_\mathrm{l}}{2} f_0^n (g_1^n - g_0^n).\end{aligned} \tag{3.15b}$$

The above identities will prove useful in energy analysis techniques later on. A derivation of Eq. (3.13a) is given in Appendix F.1.

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

$$\langle f, \delta_{xx} g\rangle_d = \langle \delta_{xx} f, g\rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \tag{3.16a}$$

$$\langle f, \delta_{xx} g\rangle_d = \langle \delta_{xx} f, g\rangle_{\underline{\overline{d}}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0, \tag{3.16b}$$

$$\langle f, \delta_{xx} g\rangle_d' = \langle \delta_{xx} f, g\rangle_d' + f_N \delta_{x\cdot} g_N - g_N \delta_{x\cdot} f_N - f_0 \delta_{x\cdot} g_0 + g_0 \delta_{x\cdot} f_0. \tag{3.16c}$$

3.2.3 Product identities

Some useful identities used in this work are

$$(\delta_{t\cdot} u_l^n)(\delta_{tt} u_l^n) = \delta_{t+}\left(\frac{1}{2}(\delta_{t-} u_l^n)^2\right), \tag{3.17a}$$

$$(\delta_{t\cdot} u_l^n) u_l^n = \delta_{t+}\left(\frac{1}{2} u_l^n e_{t-} u_l^n\right), \tag{3.17b}$$

$$(\delta_{t+} u_l^n)(\mu_{t+} u_l^n) = \delta_{t+}\left(\frac{1}{2}(u_l^n)^2\right), \tag{3.17c}$$

$$(\delta_{t\cdot} u_l^n)(\mu_{t\cdot} u_l^n) = \delta_{t+}\left(\frac{1}{2}\mu_{t-}(u_l^n)^2\right), \tag{3.17d}$$

$$(\delta_{t\cdot} u_l^n)(\mu_{tt} u_l^n) = \delta_{t+}\left(\frac{1}{8}(u_l^n + e_{t-} u_l^n)^2\right), \tag{3.17e}$$

$$u_l^n e_{t-} u_l^n = (\mu_{t-} u_l^n)^2 - \frac{k^2}{4}(\delta_{t-} u_l^n)^2. \tag{3.17f}$$

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

$$\delta_{t+}(u_l^n w_l^n) = (\delta_{t+} u_l^n)(\mu_{t+} w_l^n) + (\mu_{t+} u_l^n)(\delta_{t+} w_l^n). \tag{3.18}$$

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. (3.18) 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 [21], 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 [57, 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 [58] co-authored by John von Neumann. Later, this technique got a more general treatment in [59] and is heavily used in [21]. The discrete-time z-transform and discrete spatial Fourier transform performed on a 1D grid function are defined as [21]

$$\hat{u} = \sum_{n=-\infty}^{\infty} u_l^n z^{-n} \quad \text{and} \quad \tilde{u} = \sum_{l=-\infty}^{\infty} u_l^n e^{-jl\beta h} \tag{3.19}$$

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 6.2.4.

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) [21]

$$u_l^n \overset{\mathcal{A}}{\Longrightarrow} z^n e^{jl\beta h} \tag{3.20}$$

where "$\overset{\mathcal{A}}{\Longrightarrow}$" indicates to replace the grid function with the ansatz.

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 grid functions with their temporal and spatial indices shifted in different ways.

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 (3.20) with frequently appearing temporal and spatial shifts.

Using these definitions, the effect of various operators on a grid function can be written in their frequency domain representation. For systems distributed in space, the following trigonometric identities are extremely useful when performing the analyses [60, p. 71]:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \quad \Rightarrow \quad \sin^2(x) = \frac{e^{j2x} + e^{-j2x}}{-4} + \frac{1}{2}, \tag{3.21a}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \quad \Rightarrow \quad \cos^2(x) = \frac{e^{j2x} + e^{-j2x}}{4} + \frac{1}{2}. \tag{3.21b}$$

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 (3.21a) 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

$$\begin{aligned} \delta_{t+}u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(z-1\right), & \delta_{t-}u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k}\left(1-z^{-1}\right), \\ \delta_{t\cdot}u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{2k}\left(z-z^{-1}\right), & \delta_{tt}u_l^n &\overset{\mathcal{A}}{\Longrightarrow} \frac{1}{k^2}\left(z-2+z^{-1}\right), \end{aligned} \tag{3.22}$$

and for spatial operators, identity (3.21a) can be used to obtain

$$\delta_{xx}u_l^n \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2}\sin^2(\beta h/2), \tag{3.23a}$$

$$\delta_{xxxx}u_l^n \overset{\mathcal{A}}{\Longrightarrow} \frac{16}{h^4}\sin^4(\beta h/2). \tag{3.23b}$$

Proving stability

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

$$|z| \le 1. \tag{3.24}$$

In a 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 (3.24) for all wave numbers β. It can be shown that for a polynomial of the form

$$z^2 + a^{(1)}z + a^{(2)} \tag{3.25}$$

its roots satisfy condition (3.24) when it abides the following condition [21]

$$|a^{(1)}| - 1 \le a^{(2)} \le 1. \tag{3.26}$$

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

$$|a^{(1)}| \le 2, \tag{3.27}$$

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 Eq. (3.20) with $l = 0$. Using Table 3.1 and Eqs. (3.22) 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:

$$z - \left(2 - \frac{Kk^2}{M}\right) + z^{-1} = 0. \tag{3.28}$$

To begin to prove stability, this equation needs to be written in the form found in (3.25). Multiplying all the terms by z, and noticing that $a^{(2)} = 1$, one could continue with condition (3.27). However, the scheme used here is a special case where the roots of the characteristic equation can not be identical [21]. 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 (3.27) 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

$$k < 2\sqrt{\frac{M}{K}}. \tag{3.29}$$

Recalling that $\omega_0 = \sqrt{K/M}$ (see Eq. (2.30)), Eq (3.29) can be more compactly written as

$$k < \frac{2}{\omega_0}. \tag{3.30}$$

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. (3.22) and (3.23a):

$$\frac{1}{k^2}\left(z-2+z^{-1}\right) = -\frac{4c^2}{h^2}\sin^2\left(\beta h/2\right). \tag{3.31}$$

Also recalling that $\lambda = ck/h$ (see Eq. (2.45)), the characteristic equation of the 1D wave equation is

$$z + \left(4\lambda^2\sin^2(\beta h/2) - 2\right) + z^{-1} = 0. \tag{3.32}$$

The scheme is then stable if the roots satisfy condition (3.24). As the characteristic equation is of the form in Eq. (3.25) (after multiplication with z) with $a^{(2)} = 1$, stability is shown by abiding condition (3.27) for all β. When applied to the characteristic equation (3.32), 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 will 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, yields 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

$$h \geq ck, \tag{3.33}$$

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

3.3.3 Discussion

Although frequency domain analysis is very useful, it can only be applied to linear and time-invariant (LTI) systems and linear and shift-invariant (LSI) systems. These, respectively, describe systems whose properties do not change over time (LTI) and in space (LSI).[2] Furthermore, the analysis assumes systems with infinite domains, so boundary conditions are not included. Energy analysis techniques, on the other hand, allow these types of systems, even nonlinear systems, to be analysed. Moreover, these techniques can work with finite domains, such that boundary conditions can be handled as well. Energy analysis techniques will be presented below.

3.4 Energy analysis

Of all analysis techniques described in this chapter, energy analysis is, without a doubt, the most important when working with FD schemes. 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 locating the errors. 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 and LSI systems with infinite domains (for distributed systems), energy analysis can be applied to nonlinear systems and include boundary conditions [21].

Gustafsson et al. in (the first edition of) [61] worked with energy to find stability conditions for FD schemes. The authors referred to this 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 elaborate overview in [21] where this has been extensively used to show stability of many FD schemes.

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 practically oriented and only the discrete time case will be considered.

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.

[2] Acoustic tubes with a spatially-varying cross-section presented in Chapter 5 are examples of non-LSI systems.

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. (3.7)), which yields an 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 $(\delta_{t\cdot}u_l^n)$ over a discrete domain d. 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.[3]

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.[4]

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

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q} - \mathfrak{p}. \tag{3.34}$$

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:

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.35}$$

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 are extremely useful, 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:

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \tag{3.36}$$

As a rule of thumb, the definition for kinetic energy contains 'velocity squared'

[3] [Bilbao, verbally]

[4] So using the discretised version of e.g. Eq. (2.28) rather than Eq. (2.29).

(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 kg $\cdot$ m$^2\cdot$ 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: kg $\cdot$ m$^2\cdot$ s^{-3}. As mentioned in Step 1, it is therefore useful to have all physical parameters written out so that the units will be correct in this step. Some information about operators and grid functions and how they 'add' units are given in Table 3.2. Finally, the appearance of a grid function u_l^n 'adds' the unit of whatever it describes. For example, if u_l^n describes a displacement in m, it will 'add' a unit of m to the equation.

Name	Operator	Unit
Inner product (1D)	$\langle\cdot,\cdot\rangle_d$	m
Norm (1D)	$\lVert\cdot,\cdot\rVert_d^2$	m
First order ops. in time	$\delta_{t+},\delta_{t-},\delta_{t\cdot}$	s^{-1}
Second order op. in time	δ_{tt}	s^{-2}
First order ops. in space	$\delta_{x+},\delta_{x-},\delta_{x\cdot}$	m^{-1}
Second order op. in space	δ_{xx}	m^{-2}
Shift operators	$e_{t-},e_{x+},\dots$	-
Averaging operators	$\mu_{t+},\mu_{tt},\mu_{x\cdot},\dots$	-

Table 3.2: Units of operators.

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 main loop and plotted after the simulation. For a system without losses or energy inputs, the energy should be unchanged according to Eq. (3.35) and can be plotted according to

$$\mathfrak{h}_{\text{e}}^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0, \tag{3.37}$$

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, minute 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 e.g. Figures 3.1 and 3.2.

For a system with losses or energy inputs, a discrete integration, or summed form can be used (as done in e.g. [62]):

$$\mathfrak{h}_\text{e}^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0 + k\sum_{m=0}^{n-1}(\mathfrak{q}^m + \mathfrak{p}^m)}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0. \tag{3.38}$$

3.4.2 Mass-spring system

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

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

an energy analysis can be performed using the four 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 $(\delta_{t\cdot}u^n)$:

$$\delta_{t+}\mathfrak{h} = M(\delta_{t\cdot}u^n)(\delta_{tt}u^n) + K(\delta_{t\cdot}u^n)(u^n) = 0. \tag{3.39}$$

Note that an inner product is not necessary here, 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 Eq. (3.39), one can use identities (3.17a) and (3.17b) to get the following:

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n\right) = 0, \tag{3.40}$$

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

$$\mathfrak{h} = \frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n = 0. \tag{3.41}$$

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-}u^n)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}u^n e_{t-}u^n. \tag{3.42}$$

Step 3: Check units

As mentioned above, the energy $\mathfrak{h}$ needs to be in Joules, or $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$. Taking the terms in Eq. (3.42) one-by-one and writing them in their units results in

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

which indeed have the correct units.

Step 4: Implementation

Equation (3.47) can then be implemented in the same for-loop recursion where the update is calculated.

```
%% Calculate the energy using Eq. (3.42)

% 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 3.1 shows the normalised energy (according to Eq. (3.37)) of the mass-spring system and shows that the deviation is indeed within machine precision.

3.4.3 1D wave equation

Energy analysis could be directly performed on the FD scheme in Eq. (2.42). However, as mentioned above, it is useful to write out all physical parameters such that the units of the scheme add up to energy in Joules. 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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n, \tag{3.43}$$

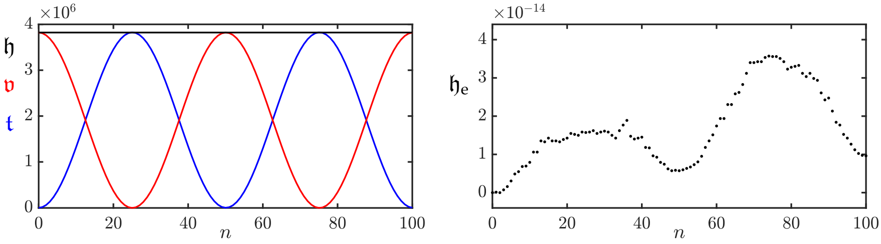


Fig. 3.1: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within machine precision.

where $l \in d$ with discrete domain $d \in \{0, \dots, N\}$, and $N+1$ is number of grid points. 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. (3.43) 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:

$$\delta_{t+}\mathfrak{h} = \rho A \langle \delta_{t\cdot}u_l^n, \delta_{tt}u_l^n \rangle_d - T\langle \delta_{t\cdot}u_l^n, \delta_{xx}u_l^n\rangle_d = 0. \tag{3.44}$$

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. (3.44), the terms have to be rewritten in a way that it fits the product identities in Section 3.2.3. Summation by parts as described in Section 3.2.2 can be used. Using identity (3.13a) 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

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

and reduced domain $\underline{d} = \{0, \dots, N-1\}$. 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. (3.8), Eq. (3.44) can be expanded to

$$\delta_{t+}\mathfrak{h} = \rho A \sum_{l=0}^{N} h(\delta_{t\cdot} u_l^n)(\delta_{tt} u_l^n) + T \sum_{l=0}^{N-1} h(\delta_{t\cdot}\delta_{x+} u_l^n)(\delta_{x+} u_l^n) \tag{3.45}$$

Then, using identities (3.17a) and (3.17b), δ_{t+} can be isolated

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}\right), \tag{3.46}$$

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

$$\begin{gathered}\mathfrak{h} = \mathfrak{t} + \mathfrak{v},\\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad \text{and} \quad \mathfrak{v} = \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}.\end{gathered} \tag{3.47}$$

Step 3: Check units

Writing out the definitions for kinetic and potential energy in Eq. (3.47) respectively, yields

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

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. (3.47) can be implemented with the following code in the for-loop recursion:

```
%% Calculate the energy using Eq. (3.47)

% 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 $\mathbf{u} = [u_1^n, \dots, u_{N-1}^n]^T$ (as Dirichlet boundary conditions are used) and need to be concatenated with 0 in the calculation of the potential energy as the boundaries needs to be included in the calculation, despite them being 0.[5] Figure 3.2 shows the plot of the normalised energy according to Eq. (3.37) and shows that the deviation of $\mathfrak{h}^n$ is within machine precision.

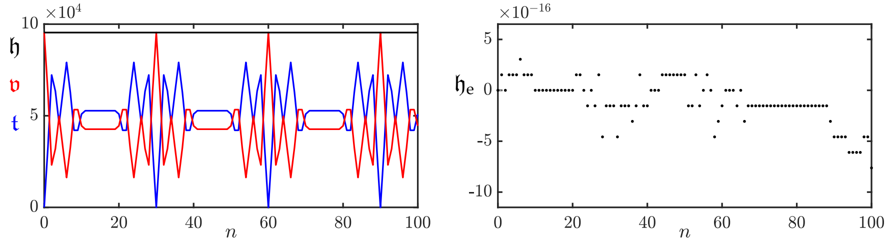


Fig. 3.2: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

Neumann boundary conditions

If Neumann boundary conditions – as per Eq. (2.48b) – are used instead, the primed inner product in Eq. (3.9) needs to be used in Step 1. Using the identity in (3.14a), summation by parts of the second term results in

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

[5] As can be seen from the definition of $\mathfrak{v}$ in Eq. (3.47), the domain used for the inner product is $\underline{d} = \{0, \dots, N-1\}$ and $\mathfrak{v}$ contains a forward difference in its definition requiring u_N^n as well.

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

$$\begin{gathered} \mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \\ \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\left(\|\delta_{t-}u_l^n\|'_d\right)^2, \quad \text{and} \quad \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}. \end{gathered} \tag{3.48}$$

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

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

% 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 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. These techniques might even be considered more powerful than the frequency domain approach, as it can also be used to analyse spatially varying and 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

$$x^2 + y^2 + 2axy \tag{3.49}$$

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

Equation (3.49) can be used to prove stability for the mass spring system using the energy balance in Eq. (3.42). 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. (3.42) yields

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

Dividing all terms by $M/2k^2$ this equation is of the form in Eq. (3.49):

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

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

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

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

$$k < 2\sqrt{\frac{M}{K}} \tag{3.50}$$

which is identical to the definition in Eq. (3.29).

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. (3.47) 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 (3.17f) 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 [21]

$$\|\delta_{x+}u_l^n\|_{\underline{d}} \le \frac{2}{h}\|u_l^n\|_d' \le \frac{2}{h}\|u_l^n\|_d, \tag{3.51}$$

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

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

Substituting this condition into the energy balance in Eq. (3.47) yields

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

Recalling that $c = \sqrt{T/\rho A}$ and $\lambda = ck/h$ (see Section 2.4), all terms can be divided by ρA which yields

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \ge \frac{1}{2}\left(1-\lambda^2\right)\|\delta_{t-}u_l^n\|_d^2 + \frac{c^2}{2}\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2, \tag{3.52}$$

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 number 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. Although this method is only fully accurate for LTI systems, it can still provide valuable information about systems with slow (sub-audio rate) parameter changes.[6] This section will show how to numerically obtain the modal frequencies of a FD scheme using the 1D wave equation as a test case.

Recall the matrix form of the 1D wave equation from Eq. (3.5)

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

Following [21], one can insert a test solution of the form $\mathbf{u}^n = z^n\boldsymbol{\phi}$ into the above equation, which yields the following characteristic equation:

$$(z - 2 + z^{-1})\boldsymbol{\phi} = c^2k^2\mathbf{D}_{xx}\boldsymbol{\phi}. \tag{3.53}$$

This is an eigenvalue problem (see Section B.4) where the p^{th} solution $\boldsymbol{\phi}_p$ may be interpreted as the modal shape of mode p. The corresponding modal frequencies (or eigenfrequencies) are the solutions to the following equations:

$$\begin{aligned} z_p - 2 + z_p^{-1} &= c^2k^2\text{eig}_p(\mathbf{D}_{xx}), \\ z_p + \Big(-2 - c^2k^2\text{eig}_p(\mathbf{D}_{xx})\Big) + z_p^{-1} &= 0. \end{aligned} \tag{3.54}$$

Furthermore, one 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

[6] The modal analysis techniques presented here have indeed been used extensively in Chapter 12, precisely for this reason.

$\sigma_p \leq 0$ of the p^{th} mode.[7] As there is no damping present in the system, the test solution reduces to $z_p = e^{j\omega_p k}$ which can be substituted into Eq (3.5) to get

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

Finally, using the trigonometric identity in Eq. (3.21a) yields

$$
\begin{aligned}
\sin^2(\omega_p k/2) + \frac{c^2k^2}{4} \text{eig}_p(\mathbf{D}_{xx}) &= 0,\\
\sin(\omega_p k/2) &= \frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})},\\
\omega_p &= \frac{2}{k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right), \qquad (3.55)
\end{aligned}
$$

and can be rewritten to

$$
f_p = \frac{1}{\pi k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_{xx})}\right) \qquad (3.56)
$$

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

See Figure 3.3 for a plot of the modal frequencies of an implementation of the 1D wave equation with the parameters given in Table 2.1. The figure shows one great advantage of performing modal analysis on a FD scheme, as opposed to only obtaining the spectrum of its output. Although the values from the analysis do correspond to the partials shown in the frequency domain output of the 1D wave equation in Figure 2.11, the latter does not show all modes present in the system. This is due to the input and output locations of the system as discussed in Section 2.4.3. The modal analysis does obtain the frequency data regardless of the aforementioned input and output locations.

3.5.1 One-step form

For more complicated systems, specifically those containing damping terms, it is useful to rewrite the update in a *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 (3.21) can not be used directly. Although the eigenvalue 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.

[7] 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.

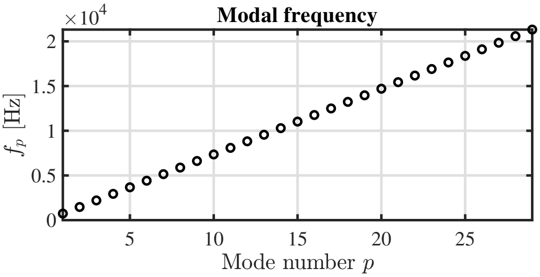


Fig. 3.3: Modal frequencies of the 1D wave equation with the parameters given in Table 2.1.

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

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{3.57}$$

can be rewritten to

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^n\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{A}^{-1}\mathbf{B} & \mathbf{A}^{-1}\mathbf{C}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}}\underbrace{\begin{bmatrix}\mathbf{u}^n\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^n} \tag{3.58}$$

which relates the unknown state of the system to the known state through matrix $\mathbf{Q}$. 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)

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{3.59}$$

which can be solved for the pth eigenvalue as

$$z_p = \text{eig}_p(\mathbf{Q}). \tag{3.60}$$

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

$$\begin{aligned} e^{s_p k} &= \text{eig}_p(\mathbf{Q}),\\ s_p &= \frac{1}{k}\ln\left(\text{eig}_p(\mathbf{Q})\right). \end{aligned} \tag{3.61}$$

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

$$\omega_p = \Im(s_p) \quad \text{and} \quad \sigma_p = \Re(s_p), \tag{3.62}$$

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. (3.61) 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 Conclusion

This chapter presented three different analysis techniques in discrete time, that are of extreme utility when working with FD schemes. Frequency domain analysis, or von Neumann analysis in the distributed case, can be used to obtain stability conditions for LTI and LSI systems. Energy analysis techniques can also be used to prove stability and passivity, but for a larger range of systems including LTI, LSI, and nonlinear systems. Furthermore, energy analysis can be used in a practical manner to debug implementations of FD schemes and ensure that no programming errors have been made. Finally, modal analysis can be used to analyse the behaviour of a scheme in terms of its modal frequencies and modal shapes. This can be used to determine whether the auditory output matches the predictions of the analysis. Although analogous techniques in continuous time also exist, a more practical angle has been chosen for this work and only the discrete time methods have been presented. For more information about the techniques in continuous time, see [21].

All three analysis techniques will be extensively used in the rest of this document to analyse the FD schemes used in this project.

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, these can not be found in the real world as effects such as losses or frequency dispersion are not included.

This part presents the different resonators used over the course of the project that better include these non-ideal physical processes and is structured as follows: Chapter 4 introduces the stiff string, an extension of the 1D wave equation, Chapter 5 introduces acoustic tubes, used to model brass instruments, and finally, Chapter 6 introduces 2D systems which, in this project, have been used to simulate (simplified) instrument bodies. The analysis techniques introduced in the previous part will be applied to all models and described in detail.

Chapter 4

The Stiff String

In earlier chapters, the case of the ideal string was presented, and was modelled using the 1D wave equation. This system generates an output with harmonic partials that are integer multiples of the fundamental frequency (if the CFL condition is satisfied with equality). In the real world, however, strings exhibit a phenomenon called *dispersion* due to stiffness in the material, hence the name *stiff string*. 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. Finally, an example of an implicit scheme will be given and comparison to the earlier FD scheme will be made. Unless denoted otherwise, this chapter follows [21].

4.1 Continuous time

Consider a lossless stiff string of length L and with 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

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

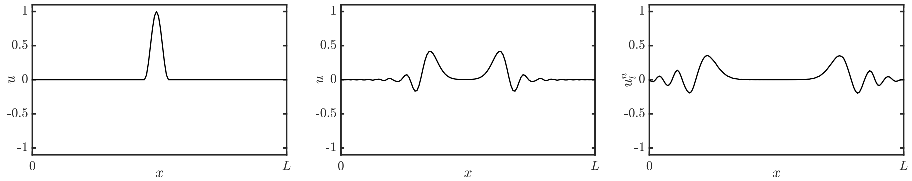
and is parametrised by material density ρ (in kg/m^3), cross-sectional area $A = \pi r^2$ (in m^2), radius r (in m), tension T (in N), Young's modulus E (in Pa) and area moment of inertia $I = \pi r^4/4$ (in m^4). In the limit as $r \to 0$, Eq (4.1) reduces to the 1D wave equation in Eq. (2.38) where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

The difference between the ideal string and the stiff string is the term containing a 4th-order spatial derivative. This term adds *stiffness* to the system and causes 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 further apart as frequency increases (see Eq. (4.6) below). 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: Frequency 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 e.g. 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 [63] and has been used extensively in the literature since (see e.g. [64, 65]). A damped stiff string can be modelled as

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \tag{4.3}$$

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. Appendix D attempts to provide some intuition on workings of these damping terms.

A more compact way to write Eq. (4.3), is to divide all terms by ρA to get

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa^2\partial_x^4 u - 2\sigma_0\partial_t u + 2\sigma_1\partial_t\partial_x^2 u. \tag{4.4}$$

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

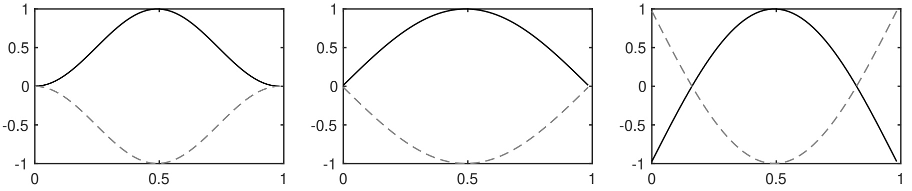
$$u = \partial_x u = 0 \quad \text{(clamped)} \tag{4.5a}$$
$$u = \partial_x^2 u = 0 \quad \text{(simply supported)} \tag{4.5b}$$
$$\partial_x^2 u = \partial_x^3 u = 0 \quad \text{(free)} \tag{4.5c}$$

at $x = 0, L$. See Figure 4.2 for plots of the first modal shape for each respective boundary condition.[1] If simply supported boundary conditions are chosen, and for low values of κ, the frequencies exhibited by the system 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 [66]

$$f_p = f_0 p\sqrt{1 + Bp^2}, \tag{4.6}$$

with inharmonicity coefficient

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



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

Fig. 4.2: Plots of the first (normalised) modal shape for the three boundary conditions in Eqs. (4.5). The extremes are indicated with solid black and dashed grey lines respectively.

[1] Note that there is a zero-frequency mode if free boundary conditions are used for both ends of the string, which is technically the first mode.

4.2 Discrete time

For the sake of compactness, Eq. (4.4) will be used in the following. Naturally, the same process can be followed for Eq. (4.3), the only difference being a multiplication by ρA of all terms.

Following Section 2.2.1 and using the FD operators presented in Section 2.2.2, Eq. (4.4) 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.

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

$$\begin{aligned}\delta_{t-}\delta_{xx} &= \frac{1}{k}\left(1 - e_{t-}\right)\frac{1}{h^2}\left(e_{x+} - 2 + e_{x-}\right),\\ &= \frac{1}{kh^2}\left(e_{x+} - 2 + e_{x-} - e_{t-}(e_{x+} - 2 + e_{x-})\right).\end{aligned} \tag{4.9}$$

To have two different shift operators multiplied together still simply means to apply each of them to the grid function individually. 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 be calculated from known values at times n and $n-1$. If this is not the case and values of e.g. u_{l+1}^{n+1} and u_{l-1}^{n+1} are required to calculate u_l^{n+1}, the scheme is called *implicit*. An example of an implicit scheme, that uses the centred operator for the temporal derivative in the frequency-dependent damping term instead, can be found in Section 4.6.

Using the definitions above, the operators in scheme (4.7) can be expanded, and after a multiplication of all terms by k^2 and collecting the terms, this yields

$$\begin{aligned}(1+\sigma_0 k)u_l^{n+1} = {} & \left(2 - 2\lambda^2 - 6\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u_l^n\\ & + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)(u_{l+1}^n + u_{l-1}^n)\\ & - \mu^2(u_{l+2}^n + u_{l-2}^n) + \left(-1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}\right)u_l^{n-1}\\ & - \frac{2\sigma_1 k}{h^2}(u_{l+1}^{n-1} + u_{l-1}^{n-1}),\end{aligned} \tag{4.10}$$

with

$$\lambda = \frac{ck}{h} \quad \text{and} \quad \mu = \frac{\kappa k}{h^2}. \tag{4.11}$$

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

$$h \geq \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}, \tag{4.12}$$

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. (4.12) should be satisfied with equality, after which the following calculations are performed:

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

which can then be used to calculate λ and μ in Eq. (4.11).

Stencil

As done in Section 2.4.2, a stencil for the FD scheme in Eq. (4.7) can be created, and 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 4th-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 4th-order spatial derivative, two virtual grid points need to be accounted for at the boundaries of the system. Discretising the boundary

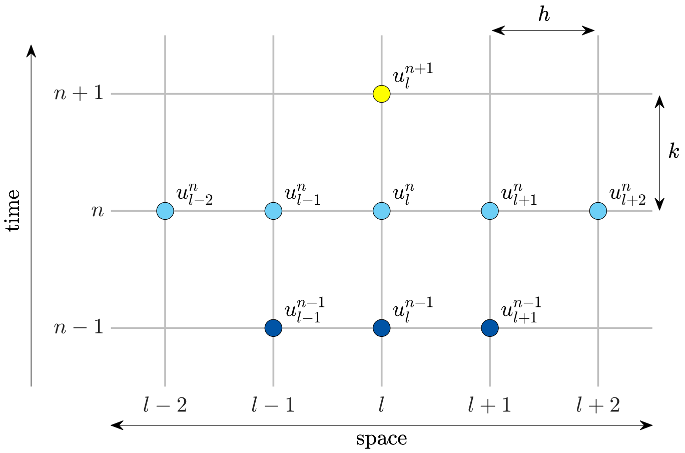


Fig. 4.3: The stencil for the damped stiff string scheme in Eq. (4.7) (adapted from [A]).

conditions in (4.5) yields

$$u_l^n = \delta_{x\pm} u_l^n = 0 \quad \text{(clamped)} \tag{4.13a}$$

$$u_l^n = \delta_{xx} u_l^n = 0 \quad \text{(simply supported)} \tag{4.13b}$$

$$\delta_{xx} u_l^n = \delta_{x\cdot}\delta_{xx} u_l^n = 0 \quad \text{(free)} \tag{4.13c}$$

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 ($l = N$). Notice that to discretise ∂_x^3 in the free boundary condition in Eq. (4.5c), the more accurate $\delta_{x\cdot}\delta_{xx}$ 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 obtain definitions for the virtual grid points.

Clamped

Expanding the operators for the clamped condition yields

$$u_0^n = u_1^n = 0 \quad \text{and} \quad u_{N-1}^n = u_N^n = 0. \tag{4.14}$$

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 $l \in$

$\{1,\ldots,N-1\}$. Evaluating the update equation in Eq. (4.10) at $l=1$ and $l=N-1$ shows that definitions for the virtual grid points u_{-1}^n and u_{N+1}^n are required. A definition for u_{-1}^n can be found by expanding Eq. (4.13b) at $l=0$:

$$\begin{aligned} &\frac{1}{h^2}\left(u_1^n - 2u_0^n + u_{-1}^n\right) = 0,\\ \overset{u_0^n=0}{\Longleftrightarrow}\quad & u_1^n + u_{-1}^n = 0,\\ & u_{-1}^n = -u_1^n, \end{aligned} \tag{4.15}$$

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

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

Substituting the first definition into the expanded scheme in Eq. (4.10) at $l=1$, yields

$$\begin{aligned}(1+\sigma_0 k)u_1^{n+1} = &\left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u_1^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)u_2^n\\ &- \mu^2 u_3^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2}\right)u_1^{n-1} - \frac{2\sigma_1 k}{h^2}u_2^{n-1}.\end{aligned} \tag{4.16}$$

Doing the same for $l=N-1$ yields

$$\begin{aligned}(1+\sigma_0 k)u_{N-1}^{n+1} = &\left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u_{N-1}^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)u_{N-2}^n\\ &- \mu^2 u_{N-3}^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2}\right)u_{N-1}^{n-1} - \frac{2\sigma_1 k}{h^2}u_{N-2}^{n-1}.\end{aligned} \tag{4.17}$$

Free

Although rarely used for the stiff string (rather for the ideal bar), free boundary conditions are given here for completeness. The free boundary condition requires all points to be calculated and the range of calculation remains $l\in\{0,\ldots,N\}$. At each respective boundary, two virtual grid points are needed: u_{-1}^n and u_{-2}^n at the left and u_{N+1}^n and u_{N+2}^n at the right boundary respectively. The third-order spatial FD operator in Eq. (4.13c) is defined as:

$$\begin{aligned}\delta_{x\cdot}\delta_{xx} &= \frac{1}{2h^3}\left(e_{x+} - e_{x-}\right)\left(e_{x+} - 2 + e_{x-}\right),\\ &= \frac{1}{2h^3}\left(e_{x+}^2 - 2e_{x+} + 2e_{x-} - e_{x-}^2\right),\end{aligned} \tag{4.18}$$

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

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

As u_0^n is not necessarily 0 at all times, solving the first part of the boundary condition (i.e., $\delta_{xx}u_0^n = 0$) 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.

Discussion

In practice, the simply supported boundary condition is mostly chosen as this most realistically reflects string terminations in the real world. 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 the boundaries of (damped) ideal bar (Eq. (4.3) 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 (see Section 3.1.2). The FD scheme of the stiff string in (4.7) can be written as

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{4.19}$$

where

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

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 Eq. (3.3) and the matrix form of the δ_{xxxx} operator is

$$\mathbf{D}_{xxxx} = \frac{1}{h^4}\begin{bmatrix} 6 & -4 & 1 & & \mathbf{0} \\ -4 & 6 & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & 6 & -4 \\ \mathbf{0} & & 1 & -4 & 6 \end{bmatrix}. \tag{4.20}$$

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 Eq. (3.3) and $\mathbf{D}_{xxxx}$ can be obtained by multiplying two $\mathbf{D}_{xx}$ matrices according to

$$\mathbf{D}_{xxxx} = \mathbf{D}_{xx}\mathbf{D}_{xx} = \frac{1}{h^4}\begin{bmatrix} 5 & -4 & 1 & & & \mathbf{0} & \\ -4 & 6 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & -4 & 1 & & \\ & \ddots & -4 & 6 & -4 & \ddots & \\ & & 1 & -4 & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & 6 & -4 \\ & \mathbf{0} & & & 1 & -4 & 5 \end{bmatrix}. \tag{4.21}$$

Finally for free boundary conditions given in Eq. (4.13c), 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 Eq. (3.4) instead, and

$$\mathbf{D}_{xxxx} = \frac{1}{h^4}\begin{bmatrix} 2 & -4 & 2 & & & \mathbf{0} & \\ -2 & 5 & -4 & 1 & & & \\ 1 & -4 & 6 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 5 & -2 \\ & \mathbf{0} & & & 2 & -4 & 2 \end{bmatrix}. \tag{4.22}$$

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 (through c and L in Eq. (2.40)), the stiff string has many more aspects that can be changed. See Table 4.1 for parameters most commonly used in this project.

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)	$100 \leq T \leq 10^4$
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.

A formula exists to calculate the loss coefficients σ_0 and σ_1 from T_{60} values at different frequencies (see [21, 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 $l = \lfloor N/20 \rfloor$) of an implementation of the stiff string excited using a raised-cosine (see Chapter 7). The parameters used can be found in Table 4.1 where $T = 3951$ N, and $r = 9.35 \cdot 10^{-4}$ m 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 frequency domain (the right panel in Figure 4.4), this shows in the fact that the partials are not 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.

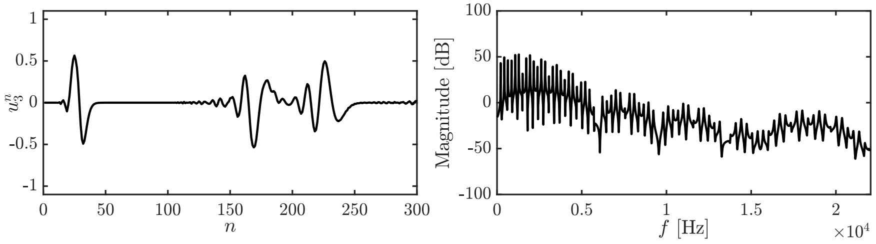


Fig. 4.4: The time-domain and frequency domain output of the stiff string. The parameters are set as in Table 4.1 where $r = 9.35 \cdot 10^{-4}$ m to highlight dispersive effects and $T = 3951$ N.

4.3 von Neumann analysis and stability condition

In order to obtain the stability condition for the damped stiff string, one can perform a von Neumann analysis, as presented in Section 3.3, on the FD scheme in Eq. (4.7). A detailed derivation can be found in Appendix F.2, and a compact version will be presented here.

Using the definitions found in Eq. (3.22) for the temporal operators, and Eqs. (3.23a) and (3.23b) 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 is as follows

$$\begin{aligned}(1+\sigma_0 k)z + \left(16\mu^2\sin^4(\beta h/2) + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right)\sin^2(\beta h/2) - 2\right)&\\ + \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z^{-1} = 0.&\qquad (4.23)\end{aligned}$$

Rewriting this to the form in Eq. (3.25), and using condition (3.26), its roots can be shown to be bounded by unity for all β under the following condition (see Appendix F.2)

$$4\mu^2 + \lambda^2 + \frac{4\sigma_1 k}{h^2} \leq 1.$$

Recalling the definitions for λ and μ from Eq. (4.11), one yields a quadratic equation in h^2 which can be shown to be bounded by

$$h \geq \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}. \tag{4.24}$$

This is the stability condition for the damped stiff string also shown in Eq. (4.12).

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 Eq. (4.3) yields

$$\rho A \delta_{tt} u_l^n = T\delta_{xx} u_l^n - EI\delta_{xxxx} u_l^n - 2\sigma_0 \rho A \delta_{t\cdot} u_l^n + 2\sigma_1 \rho A \delta_{t-}\delta_{xx} u_l^n, \tag{4.25}$$

defined for $l \in d$ with discrete domain $d = \{0, \dots, N\}$. 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. (3.8)) of the scheme with $(\delta_{t\cdot} u_l^n)$ over discrete domain d:

$$\begin{aligned} \delta_{t+}\mathfrak{h} = \rho A \langle \delta_{t\cdot} u_l^n, \delta_{tt} u_l^n \rangle_d - T\langle \delta_{t\cdot} u_l^n, \delta_{xx} u_l^n \rangle_d + EI\langle \delta_{t\cdot} u_l^n, \delta_{xxxx} u_l^n \rangle_d \\ + 2\sigma_0 \rho A \langle \delta_{t\cdot} u_l^n, \delta_{t\cdot} u_l^n \rangle_d - 2\sigma_1 \rho A \langle \delta_{t\cdot} u_l^n, \delta_{t-}\delta_{xx} u_l^n \rangle_d = 0. \end{aligned} \tag{4.26}$$

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

As there is damping present in the system, and the system is distributed, the energy balance will be of the form

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q}, \tag{4.27}$$

with boundary term $\mathfrak{b}$ and damping term $\mathfrak{q}$. The latter is defined as

$$\mathfrak{q} = 2\sigma_0 \rho A \|\delta_{t\cdot} u_l^n\|_d^2 - 2\sigma_1 \rho A \langle \delta_{t\cdot} u_l^n, \delta_{t-}\delta_{xx} u_l^n \rangle_d, \tag{4.28}$$

where the virtual grid points needed to calculate the second term are defined by the boundary conditions given in Eq. (4.13). Furthermore, $\mathfrak{b}$ appears after rewriting Eq. (4.26) using summation by parts (see Section 3.2.2), specifically, using Eq. (3.13a) for the second term and Eq. (3.16b) for the third, yields

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

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 (4.13a) and (4.13b) it can easily be shown that $\mathfrak{b} = 0$. If free conditions as in Eq. (4.13c) are used, the boundary conditions will vanish when the primed inner product in Eq. (3.9) is used in Step 1 and identity (3.16c) is used when performing summation by parts. Below, only the simply supported case will be considered.

Isolating δ_{t+} to obtain the total energy $\mathfrak{h}$ in the definition for $\delta_{t+}\mathfrak{h}$ above, requires identities (3.17a) and (3.17b) 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:

$$\begin{gathered}\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad \text{and}\\ \mathfrak{v} = \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}} + \frac{EI}{2}\langle \delta_{xx}u_l^n, e_{t-}\delta_{xx}u_l^n\rangle_{\underline{\overline{d}}}\ ,\end{gathered} \tag{4.29}$$

and can be shown to be non-negative if condition (4.12) is satisfied.

Step 3: Check units

Comparing the acquired definitions in Eq. (4.29) to those for the 1D wave equation in Eq. (3.47), one can observe that the definitions are nearly identical, the only difference being the second term in the definition for $\mathfrak{v}$ in Eq. (4.29). Writing this term out in units, and recalling that Pa (the unit for E) in SI units is kg·m^{-1}·s^{-2}, yields

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

and indeed has the correct units.

As described in Section 3.4, the damping terms in $\mathfrak{q}$ need to have units of Joules per second, or kg · m^2· s^{-3}. Writing the terms in Eq. (4.28) out in their units yields

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

which also have the correct units.

Step 4: Implementation

An implementation of the energy calculation for the simply supported boundary condition is given in Algorithm 4.1. The calculation of the damping is omitted, but the full algorithm (including other boundary conditions) can be found online [67]. Figure 4.5 shows that the damping present in the system causes $\mathfrak{h}$ to decrease in the left panel. The right panel shows that the deviation of the total energy calculated using Eq. (3.38) is within machine precision.

```
%%%% Before the main loop: %%%%

% Initialise Dx+ operator to calculate potential energy due to tension
% As the domain is reduced by one, the matrix needs to be of size N x N
Dxp = sparse(1:N, 1:N, -ones(1, N), N, N) + ...
sparse(1:N-1, 2:N, ones(1, N-1), N, N);

%%%% In the main loop: %%%%

% energy in the system
kinEnergy(n) = rho * A * h / 2 * sum((1/k * (u - uPrev)).^2);
potEnergy(n) = T / 2 * h * sum((Dxp * [0; u]) .* (Dxp * [0; uPrev]))
    ... + E * I * h / 2 * sum((Dxx * u) .* (Dxx * uPrev));
```

Algorithm 4.1: Calculating $\mathfrak{h}$ for the simply supported boundary condition.

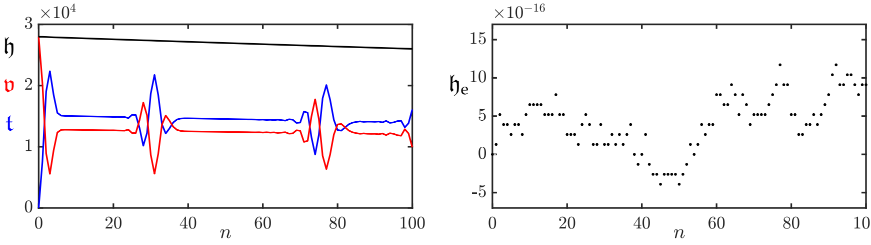


Fig. 4.5: The kinetic (blue), potential (red), and total (black) energy of an implementation of the stiff string are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

4.5 Modal analysis

To be able to perform a modal analysis on the FD scheme in Eq. (4.7), it must be written in a one-step form – introduced in Section 3.5.1 – due to the damping present in the system. Using the matrix form of the damped stiff string in Eq. (4.19), the one-step form can be written as

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^{n}\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{B}/A & \mathbf{C}/A\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}}\underbrace{\begin{bmatrix}\mathbf{u}^{n}\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^{n}}, \tag{4.30}$$

where the definitions for $\mathbf{B}$, $\mathbf{C}$ and A can be found in Section 4.2.2. In this analysis, the definitions for $\mathbf{D}_{xx}$ and $\mathbf{D}_{xxxx}$ for simply supported boundary conditions will be used.

Assuming test solutions of the form $\mathbf{w}^n = z^n\boldsymbol{\phi}$, and recalling that $z = e^{sk}$ and complex frequency $s = j\omega + \sigma$ (see Section 3.5.1), yields the following eigenvalue problem (see Section B.4):

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{4.31}$$

which can be solved for the p^{th} complex modal frequency

$$s_p = \frac{1}{k}\ln\left(\text{eig}_p(\mathbf{Q})\right). \tag{4.32}$$

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 $T = 1.88 \cdot 10^6$ N, and $r = 1.58 \cdot 10^{-2}$ m, which are unnaturally high values to highlight inharmonic behaviour. The left panel shows that the system is indeed inharmonic, i.e.,

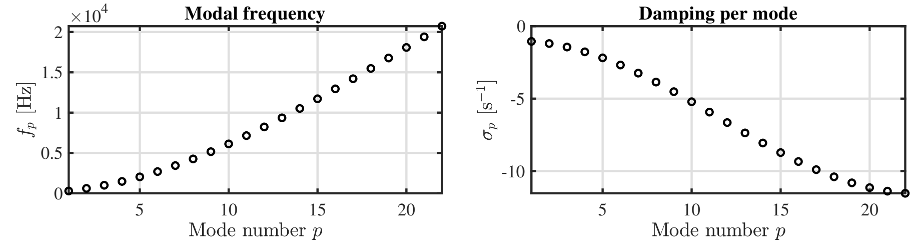


Fig. 4.6: The modal frequencies and damping per mode for the stiff string using the values in Table 4.1 and $T = 1.88 \cdot 10^6$ N and $r = 1.58 \cdot 10^{-2}$ m to highlight effects of stiffness.

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 Eq. (4.7), it can be shown that $\sigma_p = \sigma_0$ for every mode p (in this case $\sigma_0 = -1$).

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. (4.4) where the (more accurate) centred operator is used for the frequency-dependent damping term:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t\cdot}\delta_{xx}u_l^n. \tag{4.33}$$

Using the centred operator in the mixed-spatio-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 B.3. The drawback is that this requires one matrix inversion per iteration which can be extremely costly.[2] 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.

Considering simply supported boundary conditions such that the region of operation is $l \in \{1, \dots, N-1\}$, the system will have $N-1$ unknowns (u_l^{n+1} for $l \in \{1, \dots, N-1\}$) that can be calculated using $N-1$ (update) equations.

[2] In the context of FDTD methods, the matrices to be inverted are *diagonally dominant*, which means that if the off-diagonals are small, specialised methods such as the iterative Jacobi method (see e.g. [68]) could, with a few iterations, yield an answer for the inverse.

Writing this in matrix form using column vector $\mathbf{u}^n = [u_1^n, u_2^n, \dots, u_{N-1}^n]$ yields

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{4.34}$$

where

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

Equation (4.34) can be considered a system of linear equations (see Section B.3) and the state at the next time step $\mathbf{u}^{n+1}$ can then be retrieved using a matrix inversion (see B.2)

$$\mathbf{u}^{n+1} = \mathbf{A}^{-1}\left(\mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}\right). \tag{4.35}$$

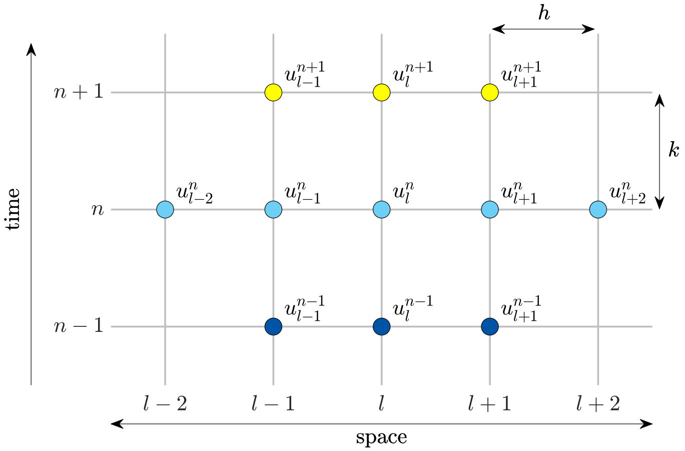


Fig. 4.7: The stencil for the damped stiff string scheme in (4.33).

4.6.1 von Neumann analysis

This section follows the same process as in Section 4.3. A full derivation is given in Appendix F.3 and a compact version is given here.

The definitions in Section 3.3 can be used to obtain a frequency domain representation of the FD scheme in Eq. (4.33):

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

and collecting the terms, yields the following characteristic equation:

$$\left(1+\sigma_0 k+\frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z+\left(16\mu^2\sin^4(\beta h/2)+4\lambda^2\sin^2(\beta h/2)-2\right)$$
$$+\left(1-\sigma_0 k-\frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z^{-1}=0. \tag{4.36}$$

Rewriting this to the form found in Eq. (3.25) and using condition (3.26), one can show that its roots are bounded by unit for all β under the following condition (see Appendix F.3):

$$4\mu^2+\lambda^2\leq 1,$$

and places the following condition on the grid spacing:

$$h\geq\sqrt{\frac{c^2k^2+\sqrt{c^4k^4+16\kappa^2k^2}}{2}}. \tag{4.37}$$

Comparing this to the stability condition for the explicit scheme in Eq. (4.12), 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 and the condition is more relaxed. 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. (4.34) matches the form in Eq. (3.57), one can perform a modal analysis by writing the scheme in a one-step form as explained in Section 3.5.1. The results of the analysis are shown in Figure 4.8 using the same values for T and r as in Section 4.5. 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. (4.6) 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. (4.12), 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 is introduced.

As the stability condition for the implicit scheme in Eq. (4.37) 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 a more accurate simulation can be obtained with fewer numerically dispersive effects, because the frequency-dependent damping term no longer affects the stability condition for the implicit scheme.

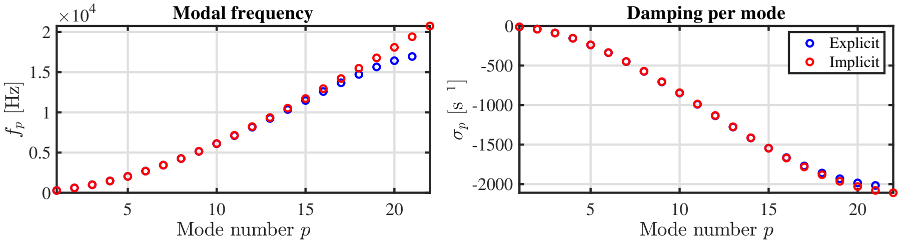


Fig. 4.8: A comparison between the modal frequencies and damping per mode of the explicit (blue) and implicit (red) scheme.

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, several 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 much more complex if the implicit scheme is used.

Chapter 5

Acoustic Tubes

The dynamics of woodwind and brass instruments is based on wave propagation in acoustic tubes. Although the physical processes that generate the sound are fundamentally different from those in strings, the underlying models have many similarities. The main difference between acoustic tubes and (ideal) strings, is that tubes have a varying cross-sectional area, causing wave dispersion and greatly influencing behaviour of the system.

In this project, the behaviour of acoustic tubes is approximated using 1D systems. Although higher-dimensional models might better capture some physical effects (see e.g. [69]), 1D systems already show good agreement between model and measurement [70]. Moreover, looking towards real-time implementation of these models, the choice to simplify to 1D has been made due to the low relative computational cost.

This chapter first presents Webster's equation, which extends the 1D wave equation presented in Section 2.4 by introducing a spatially varying cross-section. Although not used for the contributions in Part V, Webster's equation forms a good basis for the second part of this chapter, which decomposes Webster's equation into a system of two coupled first-order PDEs. This has been used to model the trombone in paper [H].

5.1 Webster's equation

For an (axially symmetric) acoustic tube of length L (in m), where the wavelengths of the frequencies at interest are much larger than the radius of the tube, one can simplify the system to be one-dimensional [23]. For low-amplitude vibrations, the air propagation in this tube can be described using *Webster*'s equation [71]

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

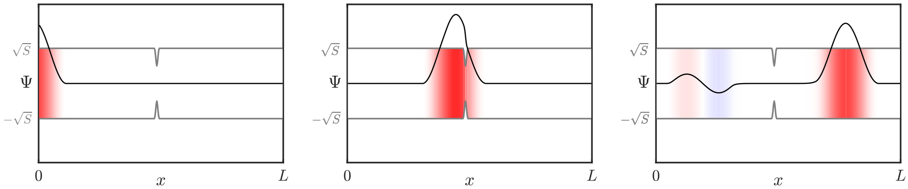
with *acoustic potential* $\Psi = \Psi(x,t)$ (in m^2/s), the cross-sectional area along the tube (or bore profile) $S = S(x)$ (in m^2) and the speed of sound in air c (in m/s). The state variable Ψ is defined for $t \geq 0$ and $x \in \mathcal{D}$ where domain $\mathcal{D} = [0, L]$. If $S(x)$ is constant, Eq. (5.1) reduces to the 1D wave equation in Eq. (2.38). This shows that for a cylindrical acoustic tube, the fundamental frequency is not affected by the cross-sectional area, but solely relies on length L and wave speed c according to Eq. (2.40).[1]

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 [23]

$$p = \rho_0 \partial_t \Psi, \quad \text{and} \quad v = -\partial_x \Psi, \tag{5.2}$$

with air density ρ_0 (in kg/m^3).

The interesting thing about the presence of a variable cross-section, is that it causes dispersive or scattering behaviour, especially at locations of high (spatial) variation of S. See Figure 5.1.



(a) $t = 1$ ms. **(b)** $t = 5$ ms. **(c)** $t = 8$ ms.

Fig. 5.1: Wave propagation and dispersion in an acoustic tube of varying cross-section (shown in grey) modelled by Webster's equation in Eq. (5.1). Positive acoustic potential Ψ is shown in red and negative in blue, highlighted by a black line for clarity.

Boundary conditions

The choices for boundary conditions in an acoustic tube are open and closed, defined as [23][2]

$$\partial_t \Psi(0,t) = 0, \qquad \partial_t \Psi(L,t) = 0, \qquad \text{(Dirichlet, open)}, \tag{5.3a}$$

$$\partial_x \Psi(0,t) = 0, \qquad \partial_x \Psi(L,t) = 0, \qquad \text{(Neumann, closed)}. \tag{5.3b}$$

This might be slightly counter-intuitive when compared to the 1D wave equation, as "closed" might imply the "fixed" or Dirichlet boundary condition. The

[1] Equation (5.1) also reduces to the 1D wave equation if the acoustic tube is conical, i.e., if $\partial_x S(x)$ is constant.

[2] The Dirichlet condition is identical to the one shown in Eq. (2.39a), but is derived from a boundary condition for the pressure $p(0,t) = p(L,t) = 0$. The time-derivative has thus been kept here.

opposite can be intuitively shown by imagining a wave front with a positive acoustic potential moving through a tube and hitting a closed end. What reflects is also a wave front with a positive acoustic potential, i.e., the sign of the potential does not flip. This also happens using the free or Neumann condition for the 1D wave equation (see Figure 2.9). Here, the following conditions are chosen:

$$\partial_x \Psi(0,t) = 0 \quad \text{and} \quad \partial_t \Psi(L,t) = 0, \tag{5.4}$$

i.e. closed at the left end and open at the right end.

5.1.1 Discrete time

The state variable is discretised to the grid function Ψ_l^n and is defined for $n \in \mathbb{N}^0$ and $l = \{0, \dots, N\}$, where N is the number of intervals between the grid points. As the cross-section is distributed in space, $S(x)$ needs to be discretised to a grid function as well, albeit only in space (as it is not time-varying). Following [23], it is useful to introduce *interleaved grid points* at $l - 1/2$ and $l + 1/2$ for S and are defined as

$$S_{l-1/2} = \mu_{x-} S(x = lh) \quad \text{and} \quad S_{l+1/2} = \mu_{x+} S(x = lh). \tag{5.5}$$

These approximate a 'true' (possibly measured) bore profile $S(x)$ sampled at $x = lh$ with grid spacing h (see Figure 5.2). Using these definitions, one can discretise Eq. (5.1) to the following FD scheme [21]:[3]

$$\bar{S}_l \delta_{tt} \Psi_l^n = c^2 \delta_{x-} \left(S_{l+1/2} (\delta_{x+} \Psi_l^n) \right), \tag{5.6}$$

where

$$\bar{S}_l = \mu_{x+} S_{l-1/2} = \mu_{x-} S_{l+1/2} = \mu_{xx} S(x = lh), \tag{5.7}$$

the choice of which will become apparent in Section 5.1.6. The right-hand side of the scheme contains an operator applied to two grid functions (S and Ψ) multiplied onto each other. In order to expand this, the product rule must be used. Recalling Eq. (3.18) and applying this to backwards spatial operators instead yields

$$\delta_{x-}(u_l^n w_l^n) = (\delta_{x-} u_l^n)(\mu_{x-} w_l^n) + (\mu_{x-} u_l^n)(\delta_{x-} w_l^n). \tag{5.8}$$

Using the product rule, the right-hand side of Eq. (5.6) can be expanded to

$$\bar{S} \delta_{tt} \Psi_l^n = c^2 \left[(\delta_{x-} S_{l+1/2})(\mu_{x-}(\delta_{x+} \Psi_l^n)) + (\mu_{x-} S_{l+1/2})(\delta_{x-}(\delta_{x+} \Psi_l^n)) \right],$$

[3] Notice that in [21], Webster's equation is $\bar{S}_l \delta_{tt} \Psi_l^n = c^2 \delta_{x+}(S_{l-1/2}(\delta_{x-} \Psi_l^n))$ but is identical to Eq. (5.6). This discretisation has been chosen for a more straightforward energy analysis in Section 5.1.5.

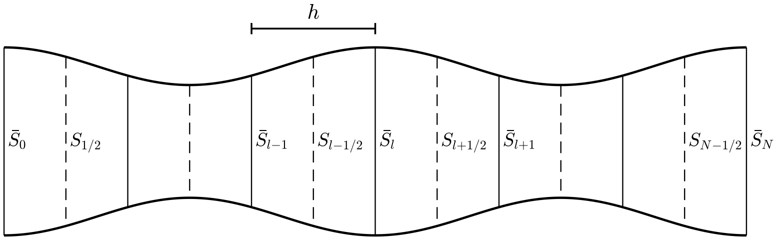


Fig. 5.2: Approximations to $S(x)$ used in Eq. (5.6). Dashed lines indicate the interleaved grid on which S is sampled (Eq. (5.5)) and solid lines indicate $\bar{S}$ which are averages of these (Eq. (5.7)).

and solving for Ψ_l^{n+1} yields the following update equation (see Appendix F.4):

$$\Psi_l^{n+1} = 2(1-\lambda^2)\Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}_l}\Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}_l}\Psi_{l-1}^n, \tag{5.9}$$

where

$$\lambda = \frac{ck}{h}, \tag{5.10}$$

and, similar to the 1D wave equation in Section 2.4.2, needs to abide

$$\lambda \leq 1 \tag{5.11}$$

in order for the scheme to be stable. See Section 5.1.6 for a derivation. The number of grid points N can then be calculated in the same way as for the 1D wave equation in Eq. (2.53), and the stencil is similar to Figure 2.10.

Notice that at the boundaries, Eq. (5.9) requires values of S outside of the defined domain through its definition in Eq. (5.7) (i.e., $S_{N+1/2}$ and $S_{-1/2}$). To solve this, one can set $\bar{S}_0 = S(0)$ and $\bar{S}_N = S(L)$ from which $S_{-1/2}$ and $S_{N+1/2}$ can be calculated according to

$$\bar{S}_0 = \frac{1}{2}(S_{1/2} + S_{-1/2}) \;\Rightarrow\; S_{-1/2} = 2\bar{S}_0 - S_{1/2}, \tag{5.12a}$$

$$\bar{S}_N = \frac{1}{2}(S_{N+1/2} + S_{N-1/2}) \;\Rightarrow\; S_{N+1/2} = 2\bar{S}_N - S_{N-1/2}. \tag{5.12b}$$

Although these values will not be needed when discretising the boundary conditions in Eq. (5.3), they will be useful at a later point.

Boundary conditions

One can discretise the continuous boundary conditions in Eq. (5.4) (closed at $x = 0$, open at $x = L$) using centred difference operators for higher accuracy according to

$$\delta_{x\cdot}\Psi_0^n = 0 \quad \Rightarrow \quad \Psi_{-1}^n = \Psi_1^n, \qquad \text{(Neumann, closed)}, \tag{5.13a}$$

$$\delta_{t\cdot}\Psi_N^n = 0 \quad \Rightarrow \quad \Psi_N^n = 0, \qquad \text{(Dirichlet, open)}. \tag{5.13b}$$

At the left boundary, Eq. (5.9) can be expanded to

$$\begin{aligned} &\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\Psi_{-1}^n, \\ \xLeftrightarrow{\text{Eq. (5.13a)}} \quad &\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2(S_{1/2}+S_{-1/2})}{\bar{S}_0}\Psi_1^n, \end{aligned}$$

and as $\bar{S}_0 = \frac{1}{2}(S_{1/2}+S_{-1/2})$ through Eq. (5.7), this can be solved to

$$\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + 2\lambda^2\Psi_1^n. \tag{5.14}$$

One can implement the right boundary condition by simply reducing the range of operation to $l = \{0, \dots, N-1\}$, as $\Psi_N^n = 0$ according to Eq. (5.13b). A more realistic boundary condition for the open end is presented in the following.

5.1.2 Radiation

One of the ways that an acoustic tube loses energy is through radiation. The right boundary condition presented in Eq. (5.4) can be changed to be radiating according to [21]

$$\partial_x\Psi(L,t) = -a_1\partial_t\Psi(L,t) - a_2\Psi(L,t), \tag{5.15}$$

where, for a tube terminating on an infinite plane [72]

$$a_1 = \frac{1}{2(0.8216)^2 c}\,, \quad \text{and} \quad a_2 = \frac{L}{0.8216\sqrt{S_0 S(1)/\pi}}\,, \tag{5.16}$$

which determine the amount of loss and inertia at the radiating boundary respectively.

The radiating boundary in Eq. (5.15) can then be discretised to [21]

$$\delta_{x\cdot}\Psi_N^n = -a_1\delta_{t\cdot}\Psi_N^n - a_2\mu_{t\cdot}\Psi_N^n, \tag{5.17}$$

which can be expanded and solved for Ψ_{N+1}^n according to

$$\Psi_{N+1}^n = h\left(-\frac{a_1}{k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - a_2(\Psi_N^{n+1} + \Psi_N^{n-1})\right) + \Psi_{N-1}^n. \tag{5.18}$$

Substitution into Eq. (5.9) at $l = N$ yields the following update equation:

$$\Psi_N^{n+1} = \frac{2(1-\lambda^2)\Psi_N^n - \Psi_N^{n-1} + \alpha_-\Psi_N^{n-1} + 2\lambda^2\Psi_{N-1}^n}{(1+\alpha_+)}, \tag{5.19}$$

where

$$\alpha_\pm = h\left(\frac{a_1}{k} \pm a_2\right)\frac{\lambda^2 S_{N+1/2}}{\bar{S}_N}. \tag{5.20}$$

One can observe that $S_{N+1/2}$ is needed, which is outside the defined domain. As mentioned before, setting $\bar{S}_N = S(L)$, one can calculate $S_{N+1/2}$ using Eq. (5.12b) to solve the issue.

5.1.3 Excitation

Although excitations will be discussed more in-depth in Chapter 7, a simple way to excite Webster's equation will be presented here.

Following [23], one can create an input signal $v_{\text{in}} = v_{\text{in}}(t)$ that interacts with the particle velocity of the tube. As this relates to the acoustic potential as in Eq. (5.2), one can change the boundary condition of the left boundary to

$$\partial_x\Psi(0,t) = -v_{\text{in}}. \tag{5.21}$$

Discretising this using the centred spatial operator, yields

$$\delta_{x\cdot}\Psi_0^n = -v_{\text{in}}^n \quad \Rightarrow \quad \Psi_{-1}^n = 2hv_{\text{in}}^n + \Psi_1^n, \tag{5.22}$$

and can be substituted into the update equation in Eq. (5.9) at $l = 0$ to get

$$\begin{aligned}\Psi_0^{n+1} &= 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1}\frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\left(2hv_{\text{in}}^n + \Psi_1^n\right),\\ \Psi_0^{n+1} &= 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + 2\lambda^2\Psi_1^n + \frac{2h\lambda^2 S_{-1/2}}{\bar{S}_0}v_{\text{in}}^n.\end{aligned} \tag{5.23}$$

The input signal is arbitrary, but looking towards lip excitation, and following [21], one can set the input to a pulse train as shown in Figure 5.3. More details on the pulse train can be found in Section 7.2.2.

5.1.4 Matrix form and output

One can write scheme (5.6) in matrix form by saving the state in a vector $\boldsymbol{\Psi}^n = [\Psi_0^n, \ldots, \Psi_N^n]^T$ and creating a $\mathbf{D}_{xx}$ matrix that includes the effect of the

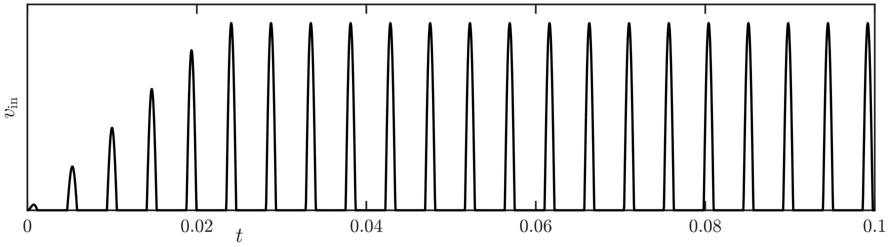


Fig. 5.3: A pulse train with a frequency of 213 Hz a duty cycle of 25% and an attack of 22 ms, used to generate the output in Figure 5.4.

cross-sectional area S. Assuming Neumann boundary conditions yields

$$\mathbf{D}_{xx} = \frac{1}{h^2}\begin{bmatrix} -2 & 2 & & & & \mathbf{0} & \\ \frac{S_{1/2}}{\bar{S}_1} & -2 & \frac{S_{3/2}}{\bar{S}_1} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \frac{S_{l-1/2}}{\bar{S}_l} & -2 & \frac{S_{l+1/2}}{\bar{S}_l} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{S_{N-3/2}}{\bar{S}_{N-1}} & -2 & \frac{S_{N-1/2}}{\bar{S}_{N-1}} \\ & \mathbf{0} & & & & 2 & -2 \end{bmatrix}. \tag{5.24}$$

Notice that there are no appearances of S at the boundaries as these vanish due to the boundary conditions as in Eq. (5.14). Using $\mathbf{I}_N$ as the $N \times N$ identity matrix, one can write scheme (5.6) in matrix form as

$$\mathbf{A}\mathbf{\Psi}^{n+1} = \mathbf{B}\mathbf{\Psi}^n + \mathbf{C}\mathbf{\Psi}^{n-1} + \mathbf{v}^n, \tag{5.25}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} \\ \mathbf{0} & 1+\alpha_+ \end{bmatrix}, \quad \mathbf{B} = 2\mathbf{I} + c^2k^2\mathbf{D}_{xx}, \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} -\mathbf{I}_N & \mathbf{0} \\ \mathbf{0} & -1+\alpha_- \end{bmatrix},$$

and the $(N+1) \times 1$ input vector $\mathbf{v}^n$ consists of zeros except for the first index:

$$\mathbf{v}_i^n = \begin{cases} \frac{2h\lambda^2 S_{-1/2}}{\bar{S}_0} v_{\text{in}}^n, & \text{if } i = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.26}$$

Notice how the radiation is included by changing the last entry of matrices **A** and **C**. The output of an implementation of Webster's equation is shown in Figure 5.4. The parameters used for the scheme, the input signal and the geometry used to obtain the output can be found in Table 5.1, Figure 5.3, and Figure 5.5 respectively. Notice that the frequencies of the partials are not integer multiples of the fundamental (as for the 1D wave equation in Figure 2.11) due to the varying geometry of the acoustic tube and the radiating boundary.

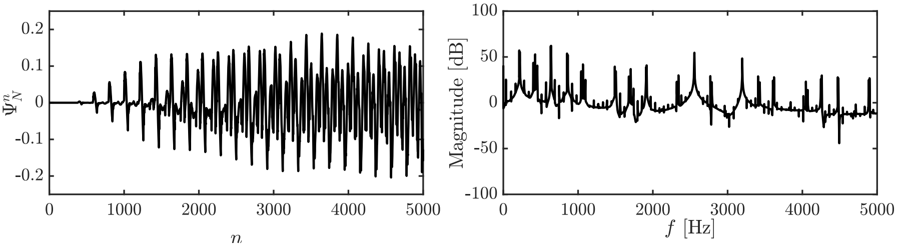


Fig. 5.4: The output of Webster's equation at Ψ_N^n using the input in Figure 5.3, the parameters in Table 5.1, and the geometry in Figure 5.5.

Name	Symbol (unit)	Value
Length	L (m)	≈ 3
Wave speed	c (m/s)	343
Cross-sectional area	$S(x)$	See paper [H]

Table 5.1: Parameters for the implementation of Webster's equation. The length is slightly below 3 m to yield $\lambda = 1$ in Eq. (5.11).

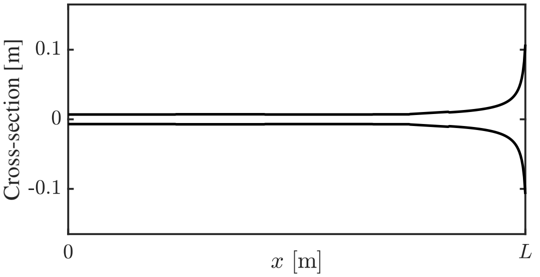


Fig. 5.5: The geometry used for the implementation. See paper [H] for more details.

5.1.5 Energy analysis

The energy analysis of Webster's equation with a radiating boundary might seem straightforward. However, due to the varying cross-sectional area, the energy balance deserves a more detailed treatment, especially at the boundaries. For this analysis, (centred) Neumann boundary conditions are used for both boundaries (for generality) and the input is ignored. This section follows the steps presented in Section 3.4.

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

Usually, to ensure vanishing boundary terms when using centred Neumann boundary conditions, the primed inner product in Eq. (3.9) is chosen. However, as the system has a spatially varying cross-section, the more general weighted inner product in Eq. (3.10) has to be chosen instead.

Taking an inner product weighted by free parameters $0 < \epsilon_{\mathrm{l}}, \epsilon_{\mathrm{r}} \leq 2$ at the left and right boundary respectively, of scheme (5.6) with $(\delta_{t\cdot}\Psi_l^n)$ over discrete domain d, yields

$$\delta_{t+}\mathfrak{h} = \langle \delta_{t\cdot}\Psi_l^n, \bar{S}_l \delta_{tt}\Psi_l^n \rangle_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}} - c^2 \langle \delta_{t\cdot}\Psi_l^n, \delta_{x-}(S_{l+1/2}(\delta_{x+}\Psi_l^n)) \rangle_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}} = 0. \tag{5.27}$$

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

As the right boundary is set to be radiating according to Eq. (5.17), the energy balance will eventually be of the following form:

$$\delta_{t+}(\mathfrak{h} + \mathfrak{h}_{\mathrm{b}}) = \mathfrak{b} - \mathfrak{q}_{\mathrm{b}}, \tag{5.28}$$

where $\mathfrak{h}_{\mathrm{b}}$ is the energy stored by the radiating boundary through the inertia term, $\mathfrak{q}_{\mathrm{b}}$ describes the energy losses through radiation and $\mathfrak{b}$ is the general boundary term.

Starting at Eq. (5.27), the last term can – using identity (3.15a) – be rewritten to

$$c^2 \langle S_{l+1/2}\delta_{t\cdot}\delta_{x+}\Psi_l^n, (\delta_{x+}\Psi_l^n) \rangle_{\underline{d}} + \mathfrak{b}_{\mathrm{r}} - \mathfrak{b}_{\mathrm{l}},$$

where

$$\mathfrak{b}_{\mathrm{r}} = c^2(\delta_{t\cdot}\Psi_N^n)\left(\frac{\epsilon_{\mathrm{r}}}{2} S_{N+1/2}(\delta_{x+}\Psi_N^n) + \left(1 - \frac{\epsilon_{\mathrm{r}}}{2}\right) S_{N-1/2}(\delta_{x-}\Psi_N^n)\right), \tag{5.29a}$$

$$\mathfrak{b}_{\mathrm{l}} = c^2(\delta_{t\cdot}\Psi_0^n)\left(\frac{\epsilon_{\mathrm{l}}}{2} S_{-1/2}(\delta_{x-}\Psi_0^n) + \left(1 - \frac{\epsilon_{\mathrm{l}}}{2}\right) S_{1/2}(\delta_{x+}\Psi_0^n))\right), \tag{5.29b}$$

are the right and left boundary term respectively (notice that $\mathfrak{b}_{\mathrm{l}}$ is subtracted). One can immediately observe that the boundary terms vanish if Dirichlet boundary conditions would be used.

Then, using identities (3.17a) and (3.17b) yields

$$\delta_{t+}\mathfrak{h} = \mathfrak{b}_{\mathrm{r}} - \mathfrak{b}_{\mathrm{l}},$$

where

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{1}{2}\left(\|\sqrt{\bar{S}_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}}\right)^2 \quad \text{and}$$
$$\mathfrak{v} = \frac{c^2}{2}\langle S_{l+1/2}\delta_{x+}\Psi_l^n, e_{t-}\delta_{x+}\Psi_l^n \rangle_{\underline{d}}. \tag{5.30}$$

Notice that $\bar{S}_l$ is included in the norm by using its square-root.

The next step is to find definitions for ϵ_l and ϵ_r such that the boundary terms vanish if radiation were to be ignored. In other words, the boundary terms need to be rewritten such that

$$\begin{aligned}\delta_{x\cdot}\Psi_0^n = 0 \;\Rightarrow\; \mathfrak{b}_\mathrm{l} = 0\\ \delta_{x\cdot}\Psi_N^n = 0 \;\Rightarrow\; \mathfrak{b}_\mathrm{r} = 0\end{aligned}$$

for the left and right boundary respectively. It can be shown that, for the special cases of $\epsilon_\mathrm{r} = S_{N-1/2}/\mu_{xx}S_N$ and $\epsilon_\mathrm{l} = S_{1/2}/\mu_{xx}S_0$, the boundary terms vanish:

$$\mathfrak{b}_\mathrm{r} = c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}(2-\epsilon_\mathrm{r})(\delta_{x\cdot}\Psi_N^n), \tag{5.31a}$$

$$\mathfrak{b}_\mathrm{l} = c^2(\delta_{t\cdot}\Psi_0^n)S_{1/2}(2-\epsilon_\mathrm{l})(\delta_{x\cdot}\Psi_0^n). \tag{5.31b}$$

See Appendix F.5 for a derivation of this.

To add the energy stored and dissipated by the radiating boundary, its definition in Eq. (5.17) can be substituted into the right boundary term $\mathfrak{b}_\mathrm{r}$ in Eq. (5.31a) as

$$\begin{aligned}\mathfrak{b}_\mathrm{r} &= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}(2-\epsilon_\mathrm{r})(-a_1\delta_{t\cdot}\Psi_N^n - a_2\mu_{t\cdot}\Psi_N^n),\\ &= c^2S_{N-1/2}(2-\epsilon_\mathrm{r})\left(-a_1(\delta_{t\cdot}\Psi_N^n)^2 - a_2(\delta_{t\cdot}\Psi_N^n)(\mu_{t\cdot}\Psi_N^n)\right).\end{aligned}$$

This can then be decomposed in $\mathfrak{h}_\mathrm{b}$ and $\mathfrak{q}_\mathrm{b}$ used in Eq. (5.27).

Using identity (3.17d) yields the definitions for $\mathfrak{h}_\mathrm{b}$ and $\mathfrak{q}_\mathrm{b}$ in Eq. (5.28)

$$\mathfrak{h}_\mathrm{b} = \frac{c^2S_{N-1/2}(2-\epsilon_\mathrm{r})a_2}{2}\mu_{t-}(\Psi_N^n)^2, \quad \text{and} \quad \mathfrak{q}_\mathrm{b} = c^2S_{N-1/2}(2-\epsilon_\mathrm{r})a_1(\delta_{t\cdot}\Psi_N^n)^2. \tag{5.32}$$

Finally, $\mathfrak{b} = \mathfrak{b}_\mathrm{l}$ and can be shown to vanish for both Dirichlet and (centred) Neumann conditions.

Step 3: Check units

To obtain the correct units, the quantities for pressure and particle velocity in Eq. (5.2) need to be substituted into the scheme. As this substitution will be made in Section 5.2, the unit check will be omitted here.

Step 4: Implementation

Figure 5.6 shows the energetic output of Webster's equation with a radiating boundary at $x = L$. To highlight the effect of the radiation, the parameters are set to $L = 1$ m and $S(x) = 0.01$ m^2 for all $x \in \mathcal{D}$. The system is excited with a raised cosine close to the left boundary, and when the excitation reaches the radiating boundary, the total energy in the system decreases due to the losses.

The energy stored by the boundary $\mathfrak{h}_\mathrm{b}$ is also shown and indeed increases when the wave reaches the boundary.

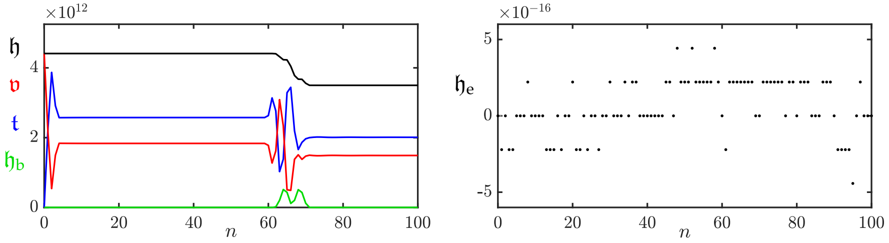


Fig. 5.6: The kinetic (blue), potential (red), and total (black) energy as well as the energy stored by the radiation condition (green) of an implementation of Webster's equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

5.1.6 Stability through energy analysis

Frequency domain analysis as presented in Section 3.3, or more specifically, von Neumann analysis, can not be performed on Webster's equation due to the varying cross-section of the system [21]. Instead, stability conditions can be obtained through energy analysis explained in Section 3.4.4. This section follows the process presented in [21, Sec. 9.1.5, pp. 255–256].

Consider the following scheme

$$[S]_l\delta_{tt}\Psi_l^n = c^2\delta_{x-}(S_{l+1/2}(\delta_{x+}\Psi_l^n)), \tag{5.33}$$

where $[S]_l$ is a yet undetermined second-order approximation to the true geometry of the acoustic tube and will be shown to be $\bar{S}_l$ below. As done for the 1D wave equation in Section 3.4.4, the potential energy $\mathfrak{v}$ in Eq. (5.30) can be rewritten using identity (3.17f) as

$$\mathfrak{v} = \frac{c^2}{2}\left(\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|^2_{\underline{d}} - \frac{k^2}{4}\|\sqrt{S_{l+1/2}}\delta_{t-}\delta_{x+}\Psi_l^n\|^2_{\underline{d}}\right).$$

For spatially varying systems, one can use the following extension of the bound given in Eq. (3.51) [21]

$$\|\sqrt{\phi_l}\delta_{x+}u_l^n\|_{\underline{d}} \le \frac{2}{h}\|\sqrt{\mu_{x-}\phi_l}u_l^n\|_d, \tag{5.34}$$

where spatially varying function $\phi_l > 0$ is defined over the same domain as u.

The following condition can then be put on $\mathfrak{v}$

$$\begin{aligned}
\mathfrak{v} &\geq \frac{c^2}{2}\left(\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|_{\underline{d}}^2 - \frac{k^2}{4}\left(\frac{2}{h}\|\sqrt{\mu_{x-}S_{l+1/2}}\delta_{t-}\Psi_l^n\|_d\right)^2\right),\\
\mathfrak{v} &\geq \frac{c^2}{2}\left(\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|_{\underline{d}}^2 - \frac{k^2}{h^2}\|\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n\|_d^2\right)\\
&\geq \frac{c^2}{2}\left(\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|_{\underline{d}}^2 - \frac{k^2}{h^2}\left(\|\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}}\right)^2\right),
\end{aligned}$$

where the last step is possible because $0 < \epsilon_\mathrm{l}, \epsilon_\mathrm{r} \leq 2$. Substituting this into the energy balance in Eq. (5.30) yields

$$\begin{aligned}
\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \geq \frac{1}{2}&\left(\|\sqrt{[S]_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}}\right)^2\\
&+ \frac{c^2}{2}\left(\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|_{\underline{d}}^2 - \frac{k^2}{h^2}\left(\|\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}}\right)^2\right),
\end{aligned}$$

and as $\|\sqrt{S_{l+1/2}}\mu_{t-}\delta_{x+}\Psi_l^n\|_{\underline{d}}^2$ is non-negative, the following is also true:

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \geq \frac{1}{2}\left(\|\sqrt{[S]_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}}\right)^2 - \frac{\lambda^2}{2}\left(\|\sqrt{\mu_{xx}S_l}\delta_{t-}\Psi_l^n\|_d^{\epsilon_\mathrm{l},\epsilon_\mathrm{r}}\right)^2.$$

This can be written as

$$\mathfrak{h} \geq \frac{1}{2}\sum_d\left(\sqrt{[S]_l} - \lambda^2\sqrt{\mu_{xx}S_l}\right)(\delta_{t-}\Psi_l^n)^2, \tag{5.35}$$

which is non-negative if

$$\begin{aligned}
\min\!\left(\sqrt{[S]_l} - \lambda^2\sqrt{\mu_{xx}S_l}\right) &\geq 0,\\
\lambda &\leq \min\left(\sqrt{\frac{[S]_l}{\mu_{xx}S_l}}\right).
\end{aligned}$$

For the special choice of $[S]_l = \mu_{xx}S_l$, this condition reduces to

$$\lambda \leq 1, \tag{5.36}$$

also given in (5.11). This choice of $[S]_l$ is equal to $\bar{S}_l$ through Eq. (5.7), hence, its choice in Eq. (5.6).

5.2 First-order system

Until now, only PDEs that are second-order in time have been presented, i.e., that are dependent on the acceleration of the state variable. This section presents a system of two coupled first-order PDEs which are instead dependent on the velocity. State-of-the-art research on brass instruments in the context of FDTD methods also uses this coupled system (see e.g. [73, 62]), and has been used in this project to model the trombone in paper [H].

5.2.1 Continuous time

Using the same variables as before for cross-sectional area $S = S(x)$, wave speed c, and air density ρ_0, a system of coupled PDEs that describes air propagation in an acoustic tube can be defined as follows:

$$\frac{S}{\rho_0 c^2}\partial_t p = -\partial_x(Sv), \tag{5.37a}$$

$$\rho_0 \partial_t v = -\partial_x p. \tag{5.37b}$$

Here, the pressure $p = p(x,t)$ (Pa) and particle velocity $v = v(x,t)$ (m/s) are defined for $x \in \mathcal{D}$ with domain $\mathcal{D} = [0, L]$ and tube length L (in m). These state variables are related to the acoustic potential Ψ, as shown in Eq. (5.2), as

$$p = \rho_0 \partial_t \Psi, \quad v = -\partial_x \Psi. \tag{5.38}$$

Indeed it can be shown by substituting these definitions into Eq. (5.37a), Webster's equation can be obtained:

$$\frac{S}{\rho_0 c^2}\partial_t(\rho_0 \partial_t \Psi) = -\partial_x(S(-\partial_x \Psi)) \quad \Longrightarrow \quad S\partial_t^2 \Psi = c^2 \partial_x(S\partial_x \Psi).$$

Boundary conditions

For the first-order PDE system in Eq. (5.37), the boundary conditions are defined as follows:

$$p(0,t) = 0, \qquad p(L,t) = 0, \qquad \text{(Dirichlet, open)}, \tag{5.39a}$$

$$S(0)v(0) = 0, \qquad S(L)v(L) = 0, \qquad \text{(Neumann, closed)}, \tag{5.39b}$$

which, through Eq. (5.38), relate to the boundary conditions of Webster's equation in Eq. (5.3).

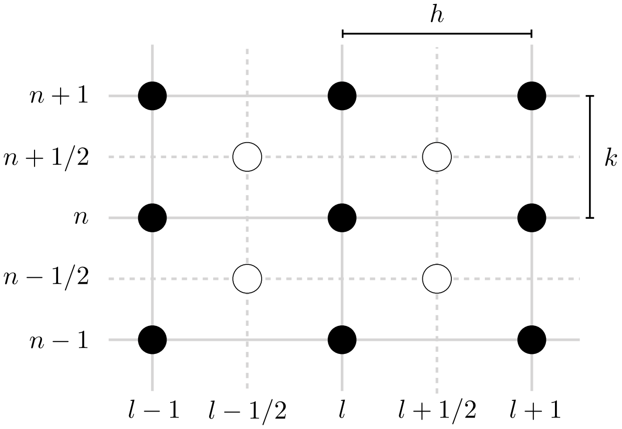


Fig. 5.7: The interleaved grid used for the system of FD schemes in Eq. (5.40). Grid points on the regular grid (in black) are used for pressure p, while points on the interleaved grid (in white) are used for particle velocity v.

5.2.2 Discrete time

It is useful to place either p or v on an interleaved grid (see Figure 5.7). Following [62], v is placed on this interleaved grid both in space and time. Accordingly, system (5.37) is discretised as

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p_l^n = -\delta_{x-}(S_{l+1/2}v_{l+1/2}^{n+1/2}), \tag{5.40a}$$

$$\rho_0\delta_{t-}v_{l+1/2}^{n+1/2} = -\delta_{x+}p_l^n, \tag{5.40b}$$

after which the update equations become

$$p_l^{n+1} = p_l^n - \frac{\rho_0 c\lambda}{\bar{S}_l}(S_{l+1/2}v_{l+1/2}^{n+1/2} - S_{l-1/2}v_{l-1/2}^{n+1/2}), \tag{5.41a}$$

$$v_{l+1/2}^{n+1/2} = v_{l+1/2}^{n-1/2} - \frac{\lambda}{\rho_0 c}(p_{l+1}^n - p_l^n), \tag{5.41b}$$

where (again) $\lambda = ck/h \leq 1$ for stability. The pressure is defined for $l = \{0,\ldots,N\}$ and the velocity for $l = \{0,\ldots,N-1\}$ where N is the number of intervals between the grid points on the pressure grid. Notice that the range of calculation for the particle velocity one index smaller than that of the pressure.

An advantage of using an interleaved grid like this, is that the forward and backward FD operators are second-order accurate, and can be shown through a Taylor series expansion as done in 2.2.2 (also see [62]).

Boundary conditions

The boundary conditions in Eq. (5.39a) can be discretised as follows

$$p_0^n = 0, \qquad p_N^n = 0, \quad \text{(Dirichlet, open)}, \tag{5.42a}$$
$$\mu_{x-}(S_{1/2}v_{1/2}^n) = 0, \quad \mu_{x+}(S_{N-1/2}v_{N-1/2}^n) = 0, \quad \text{(Neumann, closed)}. \tag{5.42b}$$

5.2.3 Matrix form

System (5.40) can be written in matrix form, by saving the states of p_l^n and $v_{l+1/2}^{n+1}$ in vectors as

$$\mathbf{p}^n = [p_0^n, \dots, p_N^n]^T, \quad \text{and} \quad \mathbf{v}^{n+1/2} = [v_{1/2}^{n+1/2}, \dots, v_{N-1/2}^{n+1/2}]^T, \tag{5.43}$$

which are of sizes $(N+1)\times 1$ and $N\times 1$ respectively. One may then write the scheme in matrix form as

$$\mathbf{v}^{n+1/2} = \mathbf{v}^{n-1/2} + \mathbf{B}_p\mathbf{p}^n \tag{5.44a}$$
$$\mathbf{p}^{n+1} = \mathbf{p}^n + \mathbf{B}_v\mathbf{v}^{n+1/2} \tag{5.44b}$$

where

$$\mathbf{B}_v = \frac{\rho_0 c}{\lambda}\begin{bmatrix} -\frac{2S_{1/2}}{\bar{S}_0} & & & \mathbf{0} \\ \frac{S_{1/2}}{\bar{S}_1} & -\frac{S_{3/2}}{\bar{S}_1} & & \\ & \ddots & \ddots & \\ & & \frac{S_{N-3/2}}{\bar{S}_{N-1}} & -\frac{S_{N-1/2}}{\bar{S}_{N-1}} \\ \mathbf{0} & & & \frac{2S_{N-1/2}}{\bar{S}_N} \end{bmatrix} \tag{5.45}$$

is of size $(N+1)\times N$, and

$$\mathbf{B}_p = \frac{\lambda}{\rho c}\begin{bmatrix} 1 & -1 & & & \mathbf{0} & \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & -1 & \\ & \mathbf{0} & & & 1 & -1 \end{bmatrix} \tag{5.46}$$

is of size $N\times(N+1)$.

Alternatively, one can write the scheme in a one-step form by concatenating the states of the pressure and particle velocity into one vector. The matrix form will then be

$$\begin{bmatrix} \mathbf{p}^{n+1} \\ \mathbf{v}^{n+1/2} \end{bmatrix} = \mathbf{B}\begin{bmatrix} \mathbf{p}^{n} \\ \mathbf{v}^{n-1/2} \end{bmatrix}, \tag{5.47}$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_{N+1} + \mathbf{B}_v\mathbf{B}_p & \mathbf{B}_v \\ \mathbf{B}_p & \mathbf{I}_N \end{bmatrix} \tag{5.48}$$

is of size $(2N+1) \times (2N+1)$, and may be directly used as matrix $\mathbf{Q}$ for the modal analysis using a one-step form described in Section 3.5.1.

5.2.4 Energy analysis

This section presents an energy analysis of the first-order system presented above using the techniques presented in Section 3.4. The bulk of the analysis follows [62, Sec. 3.4.1, pp. 80–81].

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

To obtain the correct energy balance, an inner product of Eq. (5.40a) with $\mu_{t+}p_l^n$ needs to be taken over discrete domain $d = \{0, \dots, N\}$. Using the primed inner product in Eq. (3.9) and, after taking all terms to the left-hand side, yields[4]

$$\delta_{t+}\mathfrak{h} = \frac{1}{\rho_0 c^2}\langle \mu_{t+}p_l^n, \bar{S}\delta_{t+}p_l^n\rangle_d' + \langle \mu_{t+}p_l^n, \delta_{x-}(S_{l+1/2}v_{l+1/2}^{n+1/2})\rangle_d' = 0. \tag{5.49}$$

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

For the rest of the analysis, the following superscripts and subscripts will be assumed unless denoted otherwise: n and l for p, l for $\bar{S}$, $l+1/2$ for S, and $l+1/2$ and $n+1/2$ for v. After performing summation by parts of the last term using identity (3.14a), Eq. (5.49) becomes

$$\delta_{t+}\mathfrak{h} = \frac{1}{\rho_0 c^2}\langle \mu_{t+}p, \bar{S}\delta_{t+}p\rangle_d' - \langle \mu_{t+}\delta_{x+}p, Sv\rangle_{\underline{d}} = -\mathfrak{b}, \tag{5.50}$$

where the boundary term is

$$\begin{aligned} \mathfrak{b} &= \mathfrak{b}_\mathrm{r} - \mathfrak{b}_\mathrm{l}, \quad \text{with} \\ \mathfrak{b}_\mathrm{r} &= (\mu_{t+}p_N)\mu_{x+}(S_{N-1/2}v_{N-1/2}) \quad \text{and} && (5.51) \\ \mathfrak{b}_\mathrm{l} &= (\mu_{t+}p_0)\mu_{x-}(S_{1/2}v_{1/2}), && (5.52) \end{aligned}$$

[4] The primed rather than the weighted inner product can be used here as the eventual boundary terms can be shown to vanish when using the boundary conditions in Eq. (5.42a).

and can be shown to vanish under the boundary conditions shown in Eq. (5.42a). Then, Eq. (5.40b) can be substituted into Eq. (5.50) to get

$$\delta_{t+}\mathfrak{h} = \frac{1}{\rho_0 c^2}\langle \mu_{t+}p, \bar{S}\delta_{t+}p\rangle'_d - \langle \mu_{t+}(-\rho_0\delta_{t-}v), Sv\rangle_{\underline{d}} = 0 \tag{5.53}$$

$$= \frac{1}{\rho_0 c^2}\langle \mu_{t+}p, \bar{S}\delta_{t+}p\rangle'_d + \rho_0\langle \delta_{t\cdot}v, Sv\rangle_{\underline{d}} = 0. \tag{5.54}$$

Finally, one can use identities (3.17c) and (3.17b) for the first and second term respectively to get

$$\begin{gathered}\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{where} \\ \mathfrak{t} = \frac{\rho_0}{2}\langle Sv, e_{t-}v\rangle_{\underline{d}} \quad \text{and} \quad \mathfrak{v} = \frac{1}{2\rho_0 c^2}\left(\|\sqrt{\bar{S}}p\|'_d\right)^2.\end{gathered} \tag{5.55}$$

Step 3: Check units

Writing the terms in Eq. (5.55) in their units yields

$$\begin{aligned}\mathfrak{t} = \frac{\rho_0}{2}\langle Sv, e_{t-}v\rangle_{\underline{d}} &\xrightarrow{\text{in units}} \mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}\cdot(\mathrm{m}^2\cdot\mathrm{m}\cdot\mathrm{s}^{-1}\cdot\mathrm{m}\cdot\mathrm{s}^{-1}),\\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},\\ \mathfrak{v} = \frac{1}{2\rho_0 c^2}\left(\|\sqrt{\bar{S}}p\|'_d\right)^2 &\xrightarrow{\text{in units}} (\mathrm{kg}\cdot\mathrm{m}^{-3}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2})^{-1}\cdot(\mathrm{m}\cdot\mathrm{kg}\cdot\mathrm{m}^{-1}\cdot\mathrm{s}^{-2})^2,\\ &= \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},\end{aligned}$$

and indeed have the correct units.

Step 4: Implementation

Figure 5.8 shows the energetic output of an implementation of the first order system in Eq. (5.40), and shows that the energy is within machine precision.

5.2.5 Levine and Schwinger radiation model

As done in Section 5.1, radiation can be added to the right boundary acoustic tube. The radiation model used in this project was proposed by Levine and Schwinger in [74], discretised by Silva et al. [75]. Although other radiation models exist, state-of-the-art work by Bilbao and Harrison on brass instruments based on the first-order system in (5.37) also use this model [76, 62]. See Figure 5.9 for a circuit representation of the Levine and Schwinger radiation model. This section will only go into practical details necessary for implementation of the algorithm. Further background is provided in [62].

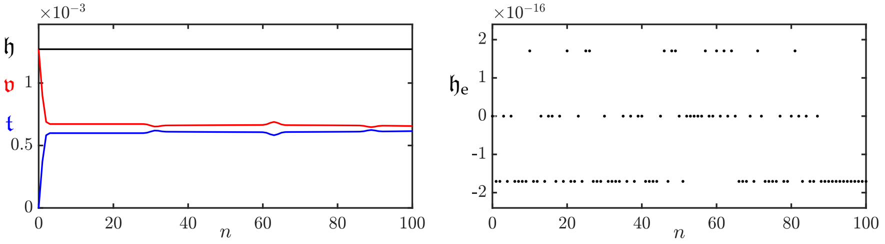


Fig. 5.8: The kinetic (blue), potential (red), and total (black) energy of an implementation of the first-order system in Eq. (5.40) are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

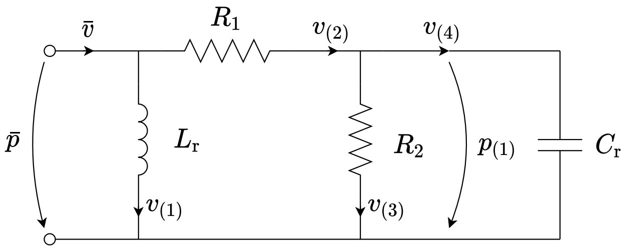


Fig. 5.9: The circuit representation of the Levine and Schwinger radiation model.

The system can be described as

$$\bar{v} = \mu_{t+}v_{(1)} + \frac{1}{R_2}\mu_{t+}p_{(1)} + C_\mathrm{r}\delta_{t+}p_{(1)}, \tag{5.56a}$$

$$\bar{p} = L_\mathrm{r}\delta_{t+}v_{(1)}, \tag{5.56b}$$

$$\bar{p} = \left(1 + \frac{R_1}{R_2}\right)\mu_{t+}p_{(1)} + R_1C_\mathrm{r}\delta_{t+}p_{(1)}, \tag{5.56c}$$

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

$$\bar{p} = \mu_{t+}p_N^n, \quad \bar{S}_N\bar{v} = \mu_{x-}\left(S_{N+1/2}v_{N+1/2}^{n+1/2}\right). \tag{5.57}$$

Using the above, an update for p_N^{n+1} based on known values of the system can be obtained (see Appendix (F.6) for a derivation):

$$p_N^{n+1} = \frac{1-\rho_0c\lambda\zeta_3}{1+\rho_0c\lambda\zeta_3}p_N^n - \frac{2\rho_0c\lambda}{1+\rho_0c\lambda\zeta_3}\left(v_{(1)}^n + \zeta_4p_{(1)}^n - \frac{S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right), \tag{5.58}$$

where

$$\zeta_1 = \frac{2R_2k}{2R_1R_2C_\mathrm{r} + k(R_1+R_2)}, \quad \zeta_2 = \frac{2R_1R_2C_\mathrm{r} - k(R_1+R_2)}{2R_1R_2C_\mathrm{r} + k(R_1+R_2)}$$
$$\zeta_3 = \frac{k}{2L_\mathrm{r}} + \frac{\zeta_1}{2R_2} + \frac{C_\mathrm{r}\zeta_1}{k} \quad \text{and} \quad \zeta_4 = \frac{\zeta_2+1}{2R_2} + \frac{C_\mathrm{r}\zeta_2 - C_\mathrm{r}}{k}.$$

Once p_N^{n+1} is known, the internal states of the system can be updated as follows

$$v_{(1)}^{n+1} = v_{(1)}^n + \frac{k}{L_\mathrm{r}}\mu_{t+}p_N^n, \tag{5.59a}$$

$$p_{(1)}^{n+1} = \zeta_1\mu_{t+}p_N^n + \zeta_2 p_{(1)}^n. \tag{5.59b}$$

The various parameters are taken from [62] and are

$$R_1 = \rho_0 c, \quad R_2 = 0.505\rho_0 c,$$
$$L_\mathrm{r} = 0.613\rho_0\sqrt{\bar{S}_N/\pi}, \quad \text{and} \quad C_\mathrm{r} = 1.111\frac{\sqrt{\bar{S}_N}}{\rho_0 c^2\sqrt{\pi}} \; .$$

5.2.6 Energy analysis

This section shows the result of the energy analysis of the above radiation model. The full derivation will not be given here, but can be found in [62]. The result is included in this thesis for completeness.

One can perform an energy analysis of the radiation model by recalling the condition at the right boundary from Eq. (5.51)

$$\mathfrak{b}_\mathrm{r} = (\mu_{t+}p_N)\underbrace{\mu_{x+}(S_{N-1/2}v_{N-1/2})}_{\mu_{x-}S_{N+1/2}v_{N+1/2}}, \tag{5.60}$$

which, using Eq. (5.57), can be rewritten to

$$\mathfrak{b}_\mathrm{r} = \bar{p}\bar{S}_N\bar{v}. \tag{5.61}$$

Following the derivation in [62, Sec. 4.1.4, pp. 111–112], this can eventually be rewritten to

$$\delta_{t+}\mathfrak{h}_\mathrm{rad} + \mathfrak{q}_\mathrm{rad} = 0, \tag{5.62}$$

where

$$\mathfrak{h}_\mathrm{rad} = \frac{\bar{S}_N}{2}\left(L_\mathrm{r}v_{(1)}^2 + C_\mathrm{r}p_{(1)}^2\right), \quad \text{and}$$
$$\mathfrak{q}_\mathrm{rad} = \bar{S}_N\left(R_1(\mu_{t+}v_{(2)}^n)^2 + R_2(\mu_{t+}v_{(3)})^2\right),$$

with

$$v_{(3)}^n = \frac{p_{(1)}^n}{R_2}, \quad \text{and} \quad \mu_{t+}v_{(2)}^n = \frac{\mu_{t+}p_N^n - \mu_{t+}p_{(1)}^n}{R_1}. \tag{5.63}$$

Notice that $\mathfrak{h}_{\text{rad}}$ and $\mathfrak{q}_{\text{rad}}$ are non-negative and thus do not affect the stability of the system.

Chapter 6

2D Systems

The previous chapters considered systems distributed over one spatial dimension. As not all musical instruments or instrument-components can be simplified to this, higher dimensional systems need to be taken into consideration, such as 2D systems. 2D PDEs can be used to model drum membranes, plate reverbs or simplified instrument bodies.

Apart from being slightly more complex than 1D models, the main issue with 2D systems is that their implementations are orders of magnitude heavier to compute than 1D schemes. This chapter will therefore also provide details on how to best implement these schemes in `MATLAB`. Implementation in C++ will be detailed in Chapter 13.

This chapter starts by providing some additional information about 2D grid functions and operators. Then, the 2D wave equation is presented, which is used to extend the analysis techniques presented in Chapter 3 to 2D. Afterwards, two 2D models that have been used in this project, the thin plate and the stiff membrane, will be described in a similar fashion. Unless denoted otherwise, the theory in this chapter follows [21].

6.1 PDEs and FD schemes in 2D

The systems modelled in this work are simplified to be rectangular and are defined on a Cartesian coordinate system. Consider a rectangular 2D system with side lengths L_x and L_y (both in m) and its state described by $u = u(x, y, t)$. The system is defined for $t \geq 0$ and $(x, y) \in \mathcal{D}$ where domain $\mathcal{D} \in [0, L_x] \times [0, L_y]$ is two-dimensional.

Similar to the 1D case explained in Section 2.2.1, the state variable can be discretised to a 2D grid function according to $u(x, y, t) \approxeq u_{l,m}^n$ with space $x = lh$ and $y = mh$ and time $t = nk$ with $k = 1/f_\text{s}$. The temporal index $n \in \mathbb{N}^0$ and the spatial indices $l \in \{0, \dots, N_x\}$ and $m \in \{0, \dots, N_y\}$ index the grid function in space in the x and y directions respectively. Here, N_x is the number of intervals between grid points in the x direction and N_y in the y direction. For simplicity, the grid spacing h is set to be the same in both the x and y directions in this work.

Additional operators

In continuous time, an additional operator referred to as the *Laplacian*, can be defined as

$$\Delta = \partial_x^2 + \partial_y^2, \tag{6.1}$$

which describes a second-order spatial derivative in 2D. A 4th-order spatial derivative in 2D, used to model stiffness like in the stiff string in Chapter 4, is called the *biharmonic* operator, and is defined as the Laplacian applied to itself:

$$\Delta\Delta = \partial_x^4 + 2\partial_x^2\partial_y^2 + \partial_y^4. \tag{6.2}$$

In discrete time, the same temporal and spatial shift operators as defined in Section 2.2.2 can be applied to grid function $u_{l,m}^n$ the latter of which only affects the spatial index l. Additional operators affecting spatial index m for the y direction are

$$e_{y+}u_{l,m}^n = u_{l,m+1}^n, \quad \text{and} \quad e_{y-}u_{l,m}^n = u_{l,m-1}^n. \tag{6.3}$$

Using these shift operators, a discrete approximation of the Laplacian in Eq. (6.1) can be made[1]

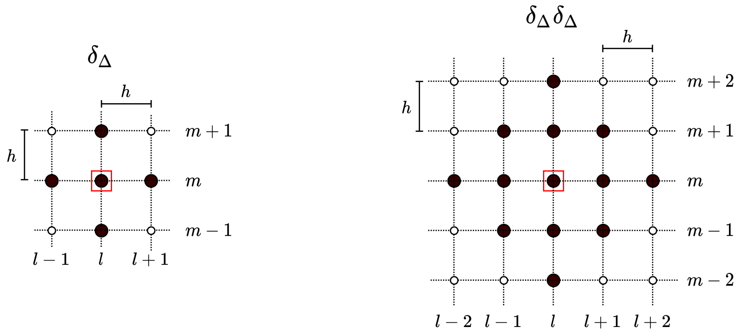
$$\Delta \approxeq \delta_\Delta \triangleq \frac{1}{h^2}\left(e_{x+} + e_{x-} + e_{y+} + e_{y-} - 4\right). \tag{6.4}$$

Also see Figure 6.1a. Similarly, an approximation of the biharmonic operator in Eq. (6.2) can be made as

$$\begin{aligned}\Delta\Delta \approxeq \delta_\Delta\delta_\Delta \triangleq \frac{1}{h^4}\bigg[&\left(e_{x+}^2 + e_{x-}^2 + e_{y+}^2 + e_{y-}^2\right) \\ &+ 2\left(e_{x+}e_{y+} + e_{x+}e_{y-} + e_{x-}e_{y+} + e_{x-}e_{y-}\right) \\ &- 8\left(e_{x+} + e_{x-} + e_{y+} + e_{y-}\right) + 20\bigg].\end{aligned} \tag{6.5}$$

Also see Figure 6.1b.

[1] Notice that the δ_Δ operator is identical to $\delta_{\Delta\boxplus}$ in [21]. The former will be used as it is more compact, and it is the only discretisation to the Laplacian used in this work.



(a) The δ_Δ operator in Eq. (6.4).

(b) The $\delta_\Delta\delta_\Delta$ operator in Eq. (6.5).

Fig. 6.1: The stencils of the 2D spatial FD operators in Eqs. (6.4) and (6.5) respectively. The red square denotes what grid point the operator is applied to. The stencils follow the same layout as Figure 2.3, but the vertical axis denotes a second spatial dimension rather than time.

6.2 The 2D wave equation

The 2D wave equation is the simplest 2D model in the context of musical acoustics and, using the operators presented above, it is a fairly straightforward extension to the 1D wave equation. Similar to how the 1D wave equation is used to model an ideal string, the 2D wave equation can be used to model an ideal membrane.

The first appearance of an implementation of the 2D wave equation in a musical context was proposed by van Duyne and Smith who used digital waveguides, or more specifically a waveguide mesh, to discretise it [27]. The implementation is identical to the FD scheme that will be presented here (if the stability condition of the latter is satisfied with equality).

This section will present the 2D wave equation in continuous time and its discretisation afterwards. The resulting FD scheme is then used as a test-case to extend the various analysis techniques presented in Chapter 3 to 2D.

6.2.1 Continuous time

Consider a system modelling the 2D wave equation with side lengths L_x and L_y (both in m) and its state described by $u = u(x, y, t)$. The system 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:

$$\partial_t^2 u = c^2 \Delta u, \tag{6.6}$$

with wave speed c (in m/s) and the Laplacian operator as defined in Eq. (6.1). If the 2D wave equation is used to model an ideal membrane, the wave speed is defined as $c = \sqrt{T/\rho H}$ (in m/s), with tension per unit length T (in N/m), material density ρ (in kg/m^3) and thickness H (in m).

Boundary conditions

Similar to the 1D wave equation, two alternatives for boundary conditions are

$$\left.\begin{aligned} u(0,y,t) = u(L_x,y,t) = 0 \quad \forall y,\\ u(x,0,t) = u(x,L_y,t) = 0 \quad \forall x,\end{aligned}\right\} \quad \text{(Dirichlet, fixed)}, \tag{6.7a}$$

$$\left.\begin{aligned} \partial_x u(0,y,t) = \partial_x u(L_x,y,t) = 0 \quad \forall y,\\ \partial_y u(x,0,t) = \partial_y u(x,L_y,t) = 0 \quad \forall x,\end{aligned}\right\} \quad \text{(Neumann, free)}, \tag{6.7b}$$

where $\forall$ means 'for all values of'.

6.2.2 Discrete time

Using the definition for the approximation of the Laplacian in Eq. (6.4), the 2D wave equation PDE in Eq. (6.6) can be discretised to

$$\delta_{tt} u_{l,m}^n = c^2 \delta_\Delta u_{l,m}^n, \tag{6.8}$$

with $l \in \{0,\dots,N_x\}$ and $m \in \{0,\dots,N_y\}$, where N_x and N_y are the number of intervals between grid points in the x and y direction respectively. The operators can then be expanded (see Eq. (6.4)) and solving for $u_{l,m}^n$ yields the following update equation

$$u_{l,m}^{n+1} = 2u_{l,m}^n - u_{l,m}^{n-1} + \lambda^2\left(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n - 4u_{l,m}^n\right), \tag{6.9}$$

where the Courant number

$$\lambda = \frac{ck}{h}, \tag{6.10}$$

and needs to abide

$$\lambda \le \frac{1}{\sqrt{2}} \tag{6.11}$$

for the scheme to be stable. See Section 6.2.4 for a derivation of this. Writing this condition in terms of the grid spacing, places the following condition on h:

$$h \ge \sqrt{2}ck\,. \tag{6.12}$$

Discrete boundary conditions

The boundary conditions in Eqs. (6.7) can be discretised to

$$\left.\begin{aligned} u_{0,m}^n = u_{N_x,m}^n &= 0 \quad \forall m,\\ u_{l,0}^n = u_{l,N_y}^n &= 0 \quad \forall l, \end{aligned}\right\} \text{(Dirichlet, fixed)}, \tag{6.13a}$$

$$\left.\begin{aligned} \delta_{x\cdot}u_{0,m}^n = \delta_{x\cdot}u_{N_x,m}^n &= 0 \quad \forall m,\\ \delta_{y\cdot}u_{l,0}^n = \delta_{y\cdot}u_{l,N_y}^n &= 0 \quad \forall l, \end{aligned}\right\} \text{(Neumann, free)}. \tag{6.13b}$$

If the Dirichlet boundary conditions are used (for all sides), the domain of calculation can simply be reduced to $l \in \{1, \ldots N_x - 1\}$ and $m \in \{1, \ldots N_y - 1\}$.

Stencil

Figure 6.2 shows the stencil of the 2D wave equation FD scheme in Eq. (6.8). The grid points use the same colour-coding as previous stencils (see e.g. Figure 2.10).

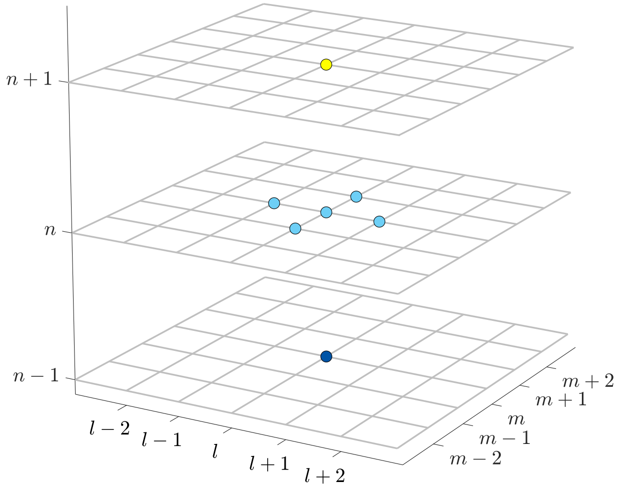


Fig. 6.2: The stencil for the 2D wave equation FD scheme in Eq. (6.8).

6.2.3 Matrix form and output

Similar to how the number of intervals between grid points is calculated for 1D systems in Eq. (2.53), it can be calculated in 2D using the following operations:

$$h := \sqrt{2}ck,\ \ N_x := \left\lfloor \frac{L_x}{h} \right\rfloor,\ \ N_y := \left\lfloor \frac{L_y}{h} \right\rfloor,\ \ h := \min\left(\frac{L_x}{N_x}, \frac{L_y}{N_y}\right),\ \ \lambda := \frac{ck}{h}, \tag{6.14}$$

where the 'min' operator selects the smallest value of L_x/N_x and L_y/N_y to stay as close to the stability condition as possible.

To implement the update equation in Eq. (6.9), one could save the states of the system in matrices (as opposed to vectors in the 1D case such as done in Section 4.2.2) and directly work with these. Using Dirichlet boundary conditions the $(N_x - 1) \times (N_y - 1)$ state matrix at time index n would be

$$\mathbf{U}^n = \begin{bmatrix} u_{1,1}^n & \cdots & u_{1,N_x-1}^n \\ \vdots & \ddots & \vdots \\ u_{N_y-1,1}^n & \cdots & u_{N_y-1,N_x-1}^n \end{bmatrix}, \tag{6.15}$$

and could be used to make a 'for-loop implementation' of the update equation. This could indeed be the strategy if one would implement the scheme in e.g. C++ (see Chapter 13). For a more compact implementation in `MATLAB`, however, one could 'stack' or 'flatten' the state matrices to vectors and update the scheme using matrix-vector multiplication (as done for the stiff string in Section 4.2.2 for example). Again using Dirichlet boundary conditions, the stacked state vector will be structured as

$$\boldsymbol{u}^n = [(\mathbf{u}_1^n)^T, \ldots, (\mathbf{u}_{N_x-1}^n)^T]^T, \quad \text{with} \quad \mathbf{u}_l^n = [u_{l,1}^n, \ldots, u_{l,N_y-1}^n]^T, \tag{6.16}$$

and has a size of $(N_x - 1) \cdot (N_y - 1) \times 1$. See Figure 6.3 for a visualisation of the matrix-stacking process.

To obtain a matrix form of the δ_Δ operator that can be applied to this stacked state vector, the *Kronecker product* and *Kronecker sum* must be introduced [77]. The Kronecker product between two arbitrarily-sized matrices (using their dimensions as a subscript) is

$$\mathbf{A}_{M\times N} \otimes \mathbf{B}_{K\times L} = \begin{bmatrix} a_{11}\mathbf{B} & \ldots & a_{1N}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{M1}\mathbf{B} & \ldots & a_{MN}\mathbf{B} \end{bmatrix}_{MK\times NL}. \tag{6.17}$$

The Kronecker sum between two square matrices is [50]

$$\mathbf{A}_{M\times M} \oplus \mathbf{B}_{N\times N} = \mathbf{I}_N \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}_M, \tag{6.18}$$

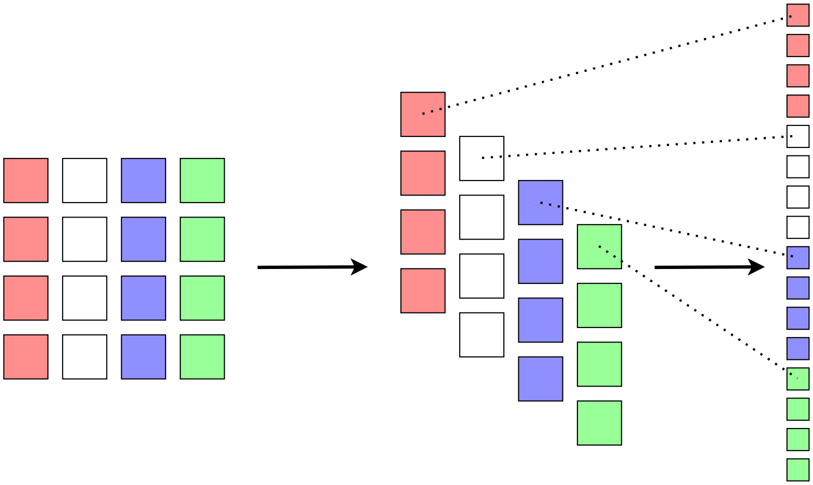


Fig. 6.3: Stacking, or 'flattening' a 4×4 matrix to a 16-element vector.

where $\mathbf{I}_P$ is the identity matrix of size $P \times P$.

For Dirichlet boundary conditions, the $\mathbf{D}_{xx}$ matrix of size $(N_x-1)\times(N_x-1)$ and the $\mathbf{D}_{yy}$ matrix of size $(N_y-1)\times(N_y-1)$ can be defined (similar to Eq. (3.3)) as

$$\mathbf{D}_{xx} = \frac{1}{h^2}\underbrace{\begin{bmatrix} -2 & 1 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 1 & -2 \end{bmatrix}}_{(N_x-1)\times(N_x-1)} \quad \text{and} \quad \mathbf{D}_{yy} = \frac{1}{h^2}\underbrace{\begin{bmatrix} -2 & 1 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 1 & -2 \end{bmatrix}}_{(N_y-1)\times(N_y-1)}. \tag{6.19}$$

Following [50], the matrix form of the δ_Δ operator can then be defined as the Kronecker sum of $\mathbf{D}_{yy}$ and $\mathbf{D}_{xx}$, yielding

$$\mathbf{D}_\Delta = \mathbf{D}_{yy} \oplus \mathbf{D}_{xx} = \begin{bmatrix} \ddots & & & & \mathbf{0} \\ & \mathbf{D}_{yy} & & & \\ & & \mathbf{D}_{yy} & & \\ & & & \mathbf{D}_{yy} & \\ \mathbf{0} & & & & \ddots \end{bmatrix} + \frac{1}{h^2}\begin{bmatrix} \ddots & \ddots & & & \mathbf{0} \\ \ddots & -2\mathbf{I} & \mathbf{I} & & \\ & \mathbf{I} & -2\mathbf{I} & \mathbf{I} & \\ & & \mathbf{I} & -2\mathbf{I} & \ddots \\ \mathbf{0} & & & \ddots & \ddots \end{bmatrix}, \tag{6.20}$$

where the identity matrix $\mathbf{I} = \mathbf{I}_{N_x-1}$. The $\mathbf{D}_\Delta$ matrix is square and of size $(N_x-1)\cdot(N_y-1) \times (N_x-1)\cdot(N_y-1)$.

Using the above, the FD scheme in Eq. (6.8) can then be compactly written in matrix form as

$$\boldsymbol{u}^{n+1} = \left(2\mathbf{I} + c^2k^2\mathbf{D}_\Delta\right)\boldsymbol{u}^n - \boldsymbol{u}^{n-1}, \tag{6.21}$$

where the identity matrix is of the same size as $\mathbf{D}_\Delta$. See Appendix C.3 for a MATLAB implementation of the 2D wave equation.[2]

If one would like to visualise the system state as a 2D grid, one can revert the stacked vector back to a matrix by using the `reshape` function in MATLAB:

```
uMatrix = reshape(u, Ny-1, Nx-1);
```

A 2D raised-cosine excitation can be implemented in the same way by reshaping an excitation matrix to a vector (see Section 7.1.4).

Output

Figure 6.4 shows the wave propagation of an implementation of the 2D wave equation with Dirichlet boundary conditions. Parameter values are $L_x = 1.5$ m, $L_y = 1$ m and $c = 360$ m/s. Waves reflect at the boundaries at an increasing rate. This is also shown in Figure 6.5, where the output – taken at $(x, y) = (0.15, 0.85)$ – in time domain shows an increase in oscillations over time, due to these reflections. The right panel shows that the output contains many partials that are close together, i.e., the output is highly inharmonic. As opposed to the output of the 1D wave equation shown in Figure 2.11, where the partials are integer multiples of the fundamental frequency, the 2D wave equation exhibits aperiodic behaviour due to the aforementioned reflections, causing this inharmonicity.

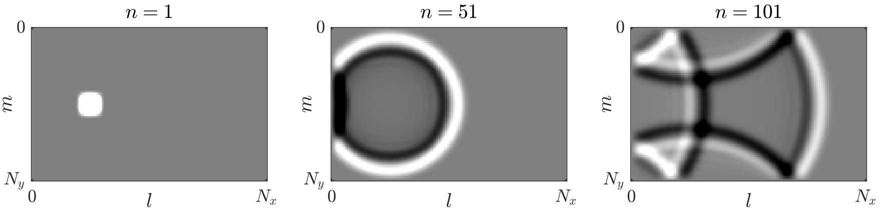


Fig. 6.4: Wave propagation of an implementation of the 2D wave equation with $L_x = 1.5$ m, $L_y = 1$ m and $c = 360$ m/s. The system is excited with a 2D raised cosine at $(0.25L_x, 0.5L_y)$.

[2] As the matrices are extremely sparse (many 0-entries), it is useful to utilise MATLABs optimisation for sparse matrices using the `sparse()` function. One can use `speye()` for sparse identity matrices.

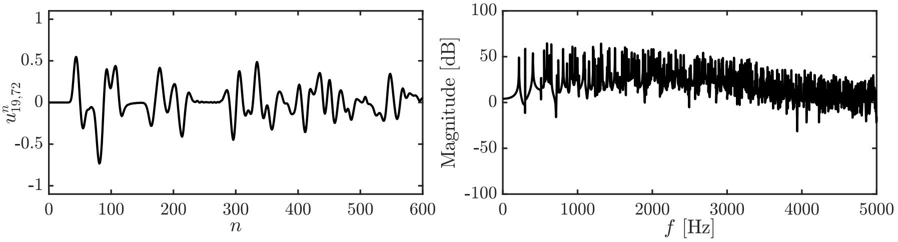


Fig. 6.5: The output of the 2D wave equation at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.4. The partials are extremely close together (notice that only frequencies up to 5000 Hz are shown) which is related to the aperiodic nature of the system behaviour.

6.2.4 Frequency domain analysis in 2D

Section 3.3 showed how to perform frequency domain analysis to obtain stability conditions for a FD scheme. This section shows extensions to this in 2D and follows [21, Ch. 10].

In 2D, the ansatz in Eq. (3.20) can be extended to

$$u^n_{l,m} \overset{\mathcal{A}}{\Longrightarrow} z^n e^{jh(l\beta_x + m\beta_y)} \tag{6.22}$$

where β_x and β_y are components of a 2D wavenumber $\boldsymbol{\beta}$ in the x and y directions respectively. Frequency domain representations of temporal operators shown in Eq. (3.22) do not change in the 2D case. Using

$$p_x = \sin^2(\beta_x h/2) \quad \text{and} \quad p_y = \sin^2(\beta_y h/2) \tag{6.23}$$

for brevity, the following frequency domain representation of spatial operators can be obtained through the ansatz in Eq. (6.22)

$$\delta_{xx} u^n_{l,m} \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2} p_x u^n_{l,m} \quad \text{and} \quad \delta_{yy} u^n_{l,m} \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2} p_y u^n_{l,m}, \tag{6.24}$$

from which it follows that

$$\delta_\Delta u^n_{l,m} \overset{\mathcal{A}}{\Longrightarrow} -\frac{4}{h^2}(p_x + p_y) u^n_{l,m}, \tag{6.25}$$

$$\delta_\Delta \delta_\Delta u^n_{l,m} \overset{\mathcal{A}}{\Longrightarrow} \frac{16}{h^4}(p_x + p_y)^2 u^n_{l,m}. \tag{6.26}$$

Using these definitions, a frequency domain interpretation of the 2D wave equation FD scheme in Eq. (6.8) can be obtained

$$\frac{1}{k^2}\left(z - 2 + z^{-1}\right) = -\frac{4c^2}{h^2}(p_x + p_y).$$

Recalling λ in Eq. (6.10), this can be rewritten to the following characteristic equation

$$z + \big(4\lambda^2(p_x + p_y) - 2\big) + z^{-1} = 0. \tag{6.27}$$

As (after multiplication by z) the characteristic equation is of the form in Eq. (3.25) and $a^{(2)} = 1$, its roots are bounded by condition (3.27)

$$\big|4\lambda^2(p_x + p_y) - 2\big| \leq 2.$$

Further derivation yields

$$\begin{aligned} -2 \leq 4\lambda^2(p_x + p_y) - 2 \leq 2,\\ 0 \leq 4\lambda^2(p_x + p_y) \leq 4, \end{aligned}$$

and as the middle term is non-negative the first condition is always satisfied, yields

$$\lambda^2(p_x + p_y) \leq 1.$$

Finally, as p_x and p_y are bounded by 1 for all wavenumbers β_x and β_y respectively, the following condition must hold

$$\begin{aligned} 2\lambda^2 &\leq 1,\\ \lambda &\leq \frac{1}{\sqrt{2}} \end{aligned} \tag{6.28}$$

which is the stability condition given in Eq. (6.11).

6.2.5 Energy analysis in 2D

Energy analysis for the 1D case is introduced in Section 3.4. Extensions for the analysis in 2D will be given here.

Analogous to the 1D inner product presented in Section 3.2.1, one can define a 2D inner product. For two functions $f = f(x, y, t)$ and $g(x, y, t)$ defined for a 2D domain $\mathcal{D}$ their inner product over this domain is defined as

$$\langle f, g\rangle_{\mathcal{D}} = \iint_{\mathcal{D}} fg dx dy. \tag{6.29}$$

Like in the 1D case, these functions do not have to be a function of time, but they are here, for coherence.

For two (grid) functions $f_{l,m}^n$ and $g_{l,m}^n$ defined over a discrete domain $d \in \{0, \dots, N_x\} \times \{0, \dots, N_y\}$ their discrete inner product is defined as

$$\langle f_{l,m}^n, g_{l,m}^n\rangle_d = \sum_{l=0}^{N_x}\sum_{m=0}^{N_y} h^2 f_{l,m}^n g_{l,m}^n. \tag{6.30}$$

Notice that the multiplication with the grid spacing is squared due to the inner product over a 2D domain (and is the discrete counterpart of $dxdy$). Useful for energy analysis are the following reduced 2D domains

$$
\begin{aligned}
\underline{d_x} &= \{0, \dots, N_x - 1\} \times \{0, \dots, N_y\}, && \text{(6.31a)}\\
\overline{\underline{d_x}} &= \{1, \dots, N_x - 1\} \times \{0, \dots, N_y\}, && \text{(6.31b)}\\
\underline{d_y} &= \{0, \dots, N_x\} \times \{0, \dots, N_y - 1\}, && \text{(6.31c)}\\
\overline{\underline{d_y}} &= \{0, \dots, N_x\} \times \{1, \dots, N_y - 1\} && \text{(6.31d)}\\
\overline{\underline{d}} &= \{1, \dots, N_x - 1\} \times \{1, \dots, N_y - 1\} && \text{(6.31e)}
\end{aligned}
$$

Below, the steps to perform energy analysis presented in Section 3.4 will be followed:

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

Using the definition of wave speed for the ideal membrane, i.e., $c = \sqrt{T/\rho H}$, the FD scheme in Eq. (6.8) can be multiplied by ρH and a 2D inner product (see Eq. (6.30)) with $(\delta_{t\cdot}u_{l,m}^n)$ over discrete domain d can be taken to yield a definition for $\delta_{t+}\mathfrak{h}$:

$$\delta_{t+}\mathfrak{h} = \rho H \langle \delta_{t\cdot}u_{l,m}^n, \delta_{tt}u_{l,m}^n\rangle_d - T\langle \delta_{t\cdot}u_{l,m}^n, \delta_{\Delta}u_{l,m}^n\rangle_d = 0,$$

which can be rewritten to

$$\delta_{t+}\mathfrak{h} = \rho H \langle \delta_{t\cdot}u_{l,m}^n, \delta_{tt}u_{l,m}^n\rangle_d - T\left(\langle \delta_{t\cdot}u_{l,m}^n, \delta_{xx}u_{l,m}^n\rangle_d + \langle \delta_{t\cdot}u_{l,m}^n, \delta_{yy}u_{l,m}^n\rangle_d\right) = 0.$$

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

Summation by parts as described in Section 3.2.2 can also be applied to δ_{yy} and the following energy balance follows

$$\delta_{t+}\mathfrak{h} = \mathfrak{b},$$

where

$$
\begin{gathered}
\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \quad \text{with} \quad \mathfrak{t} = \frac{\rho H}{2}\|\delta_{t-}u_{l,m}^n\|_d^2 \quad \text{and}\\
\mathfrak{v} = \frac{T}{2}\left(\langle \delta_{x+}u_{l,m}^n, e_{t-}\delta_{x+}u_{l,m}^n\rangle_{\underline{d_x}} + \langle \delta_{y+}u_{l,m}^n, e_{t-}\delta_{y+}u_{l,m}^n\rangle_{\underline{d_y}}\right).
\end{gathered}
\tag{6.32}
$$

Here, the reduced domains $\underline{d_x}$ and $\underline{d_y}$ are as defined in Eqs. (6.31). The boundary term is

$$\begin{aligned}\mathfrak{b} = \frac{T}{2}\Bigg[&\langle \delta_{t\cdot}u^n_{N_x,m}, \delta_{x+}u^n_{N_x,m}\rangle_{(N_x,y)} - \langle \delta_{t\cdot}u^n_{0,m}, \delta_{x-}u^n_{0,m}\rangle_{(0,y)} \\ &+ \langle \delta_{t\cdot}u^n_{l,N_y}, \delta_{y+}u^n_{l,N_y}\rangle_{(x,N_y)} - \langle \delta_{t\cdot}u^n_{l,0}, \delta_{y-}u^n_{l,0}\rangle_{(x,0)}\Bigg],\end{aligned}$$

where $(l,y) = \{l\} \times \{0,\ldots,N_y\}$ and $(x,m) = \{0,\ldots,N_x\} \times \{m\}$ are slices of domain d. The boundary term can be shown to vanish under Dirichlet boundary conditions in Eq. (6.13a). Neumann conditions will not be considered here.

Step 3: Check units

As the addition of the two inner products in the definition for $\mathfrak{v}$ in Eq. (6.32) does not affect the units, only one term is used to check the units. Recalling that, as opposed to the 1D case, the symbol T is tension per unit length and thus in N/m, one can write the terms in Eq. (6.32) in their units:

$$\begin{aligned}\mathfrak{t} = \frac{\rho H}{2}\|\delta_{t-}u^n_{l,m}\|^2_d &\xrightarrow{\text{in units}} \text{kg}\cdot\text{m}^{-3}\cdot\text{m}\cdot\text{m}^2\cdot(\text{s}^{-1}\cdot\text{m})^2 \\ &\qquad\quad = \text{kg}\cdot\text{m}^2\cdot\text{s}^{-2} \\ \mathfrak{v} = \frac{T}{2}\langle\delta_{x+}u^n_{l,m}, e_{t-}\delta_{x+}u^n_{l,m}\rangle_{\underline{d_x}} &\xrightarrow{\text{in units}} \text{N}\cdot\text{m}^{-1}\cdot\text{m}^2\cdot(\text{m}^{-1}\cdot\text{m})\cdot(\text{m}^{-1}\cdot\text{m}) \\ &\qquad\quad = \text{kg}\cdot\text{m}^2\cdot\text{s}^{-2}\end{aligned}$$

which have the correct units.

Step 4: Implementation

Figure 6.6 shows the energetic output of an implementation of the 2D wave equation and shows that the energy deviation is within machine precision.

6.2.6 Modal analysis in 2D

Given that the state vector is stacked as described in Section 6.2.3 and the update equation is written in matrix form as in Eq. (6.21), performing a modal analysis on a 2D system does not differ from a 1D system and follows the same process presented in Section 3.5.

Inserting a test solution of $\boldsymbol{u}^n = z^n\boldsymbol{\phi}$ into the matrix form of the 2D wave

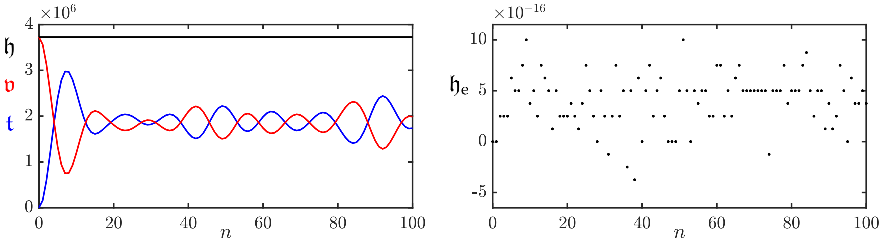


Fig. 6.6: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 2D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

equation in Eq. (6.21) yields the following characteristic equation

$$\left(z - 2 + z^{-1}\right)\boldsymbol{\phi} = c^2k^2\mathbf{D}_\triangle\boldsymbol{\phi}. \tag{6.33}$$

The p^{th} modal frequency can then be obtained by finding the roots of

$$z_p + \left(-2 - c^2k^2\text{eig}_p(\mathbf{D}_\triangle)\right) + z_p^{-1} = 0, \tag{6.34}$$

which, using test solution $z_p = e^{j\omega_p k}$ for (angular) frequency ω_p, can be shown to be

$$f_p = \frac{1}{\pi k}\sin^{-1}\left(\frac{ck}{2}\sqrt{-\text{eig}_p(\mathbf{D}_\triangle)}\right). \tag{6.35}$$

Notice the similarity to the equation for the modal frequencies of the 1D wave equation in Eq. (3.56). Again, the number of modes is equal to the number of moving grid points in the system.

See Figure 6.7 for the result of a modal analysis of the 2D wave equation. One can observe that the modes do not follow a linear pattern as opposed to those of the 1D wave equation shown in Figure 3.3. This confirms the inharmonic behaviour of the 2D wave equation discussed 6.2.3. Notice that the modal density is higher around $f_s/4$ and that the modal pattern is symmetric around this frequency. This was also observed by van Duyne and Smith in the case of the waveguide mesh [27].

Modal shapes

Using the line of code in Appendix B.4 and the `reshape` function, the modal shapes of the system can also be obtained. Figure 6.8 shows the six lowest-frequency modes of the 2D wave equation with $L_x = 1.5$ m and $L_y = 1$ m. The mode number (x, y) corresponds to the modal number in the x and y direction.

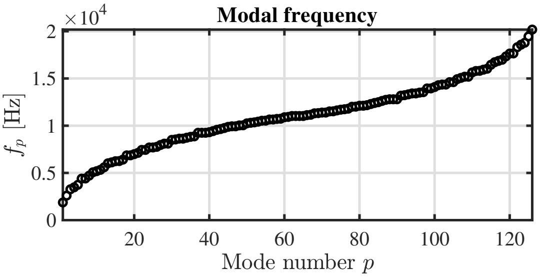


Fig. 6.7: Modal frequencies of the FD scheme implementing the 2D wave equation in Eq. (6.8) with $L_x = 1.5$ m, $L_y = 1$ m and $c \approx 3118$ m/s, such that $\lambda = 1/\sqrt{2}$ according to Eq. (6.11).

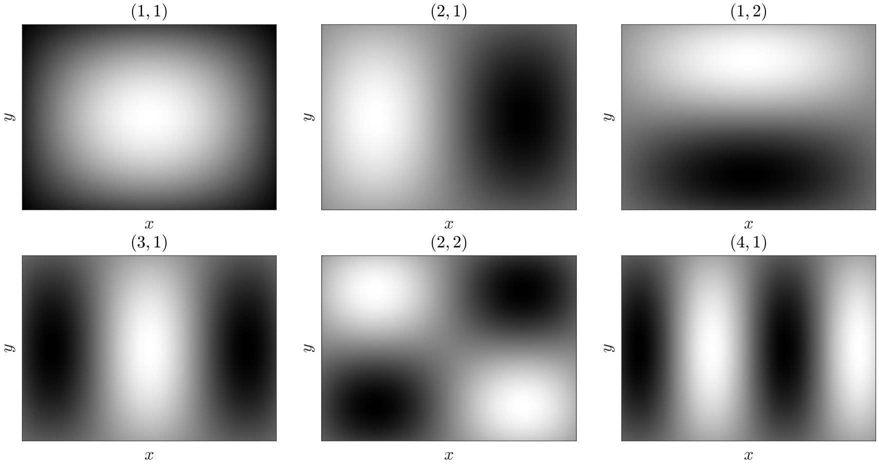


Fig. 6.8: The first six (lowest-frequency) modal shapes of 2D wave equation with $L_x = 1.5$ m and $L_y = 1$ m.

6.3 The thin plate

The thin plate, also known as the Kirchhoff model [78], differs from the 2D wave equation in that its restoring force is solely due to stiffness rather than tension. Like for the stiffness term in the stiff string (see Chapter 4), this causes frequency dispersion, and exhibits interesting timbres.

The plate model is quite versatile and can be used to model a plate reverb [79] as well as simplified instrument bodies, as done in [A], [B], [D] and [E]. This section presents the thin plate PDE and FD scheme, after which it will be subjected to the various analysis techniques extended to 2D in the previous section.

6.3.1 Continuous time

Consider a rectangular thin plate with side lengths L_x and L_y (both in m) and its transverse displacement described by $u = u(x, y, t)$ (in m). The system is defined for $(x, y) \in \mathcal{D}$ where 2D domain $\mathcal{D} = [0, L_x] \times [0, L_y]$. Using the biharmonic operator introduced in Eq. (6.2), the PDE for the thin plate can be defined as [78]

$$\rho H \partial_t^2 u = -D\Delta\Delta u, \tag{6.36}$$

where $D = EH^3/12(1-\nu^2)$ is a stiffness coefficient (in kg $\cdot$ m^2$\cdot$s^{-2}) parametrised by Young's Modulus E (in Pa), thickness H (in m) and the dimensionless Poisson's ratio ν. Although Eq. (6.36) does not hold for thick plates and only accounts for low-amplitude vibration (as it is linear), these properties can be assumed in musical instrument simulations, making this model sufficient in this work.

Adding losses to Eq. (6.36) yields

$$\rho H \partial_t^2 u = -D\Delta\Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u \tag{6.37}$$

where, as in the case of the stiff string in Eq. (4.3), σ_0 and σ_1 are the frequency-independent (in s^{-1}) and frequency-dependent damping coefficient (in m^2/s) respectively.

Boundary conditions

Similar to the stiff string, clamped and simply supported boundary conditions can be defined as

$$\left.\begin{array}{lll} u = \partial_x u = 0, & \text{if } y = \{0, L_y\}, & \forall x \\ u = \partial_y u = 0, & \text{if } x = \{0, L_x\}, & \forall y \end{array}\right\} \quad \text{(Clamped)}, \tag{6.38a}$$

$$\left.\begin{array}{lll} u = \partial_x^2 u = 0, & \text{if } y = \{0, L_y\}, & \forall x \\ u = \partial_y^2 u = 0, & \text{if } x = \{0, L_x\}, & \forall y \end{array}\right\} \quad \text{(Simply supported)}. \tag{6.38b}$$

Naturally, a free condition can be added too, but is much less trivial. As it will not be used in this work, it will not be given here, and the interested reader is instead referred to [21, Ch. 12].

6.3.2 Discrete time

Equation (6.37) can be discretised to the following FD scheme:

$$\rho H \delta_{tt} u_{l,m}^n = -D\delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-} \delta_\Delta u_{l,m}^n \tag{6.39}$$

where $l \in \{0, \ldots, N_x\}$ and $m \in \{0, \ldots, N_y\}$. Like for the stiff string FD scheme in Eq. (4.7), the backwards difference operator is used for the frequency-dependent damping term to yield an explicit scheme. A more compact way to write this scheme is after a division by ρH which yields

$$\delta_{tt}u_{l,m}^n = -\kappa^2\delta_\Delta\delta_\Delta u_{l,m}^n - 2\sigma_0\delta_{t\cdot}u_{l,m}^n + 2\sigma_1\delta_{t-}\delta_\Delta u_{l,m}^n, \tag{6.40}$$

with

$$\kappa = \sqrt{\frac{D}{\rho H}}\,. \tag{6.41}$$

Using the expansion of the discrete biharmonic operator in Eq. (6.5), Eq. (6.40) can be expanded and solved for $u_{l,m}^{n+1}$ according to

$$\begin{aligned} u_{l,m}^{n+1} = {} & (2 - 20\mu^2 - 4S)u_{l,m}^n \\ & + (8\mu^2 + S)(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n) \\ & - 2\mu^2(u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n) \\ & - \mu^2(u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n), \\ & + (\sigma_0 k - 1 + 4S)u_{l,m}^{n-1} \\ & - S(u_{l+1,m}^{n-1} + u_{l-1,m}^{n-1} + u_{l,m+1}^{n-1} + u_{l,m-1}^{n-1}) \end{aligned} \tag{6.42}$$

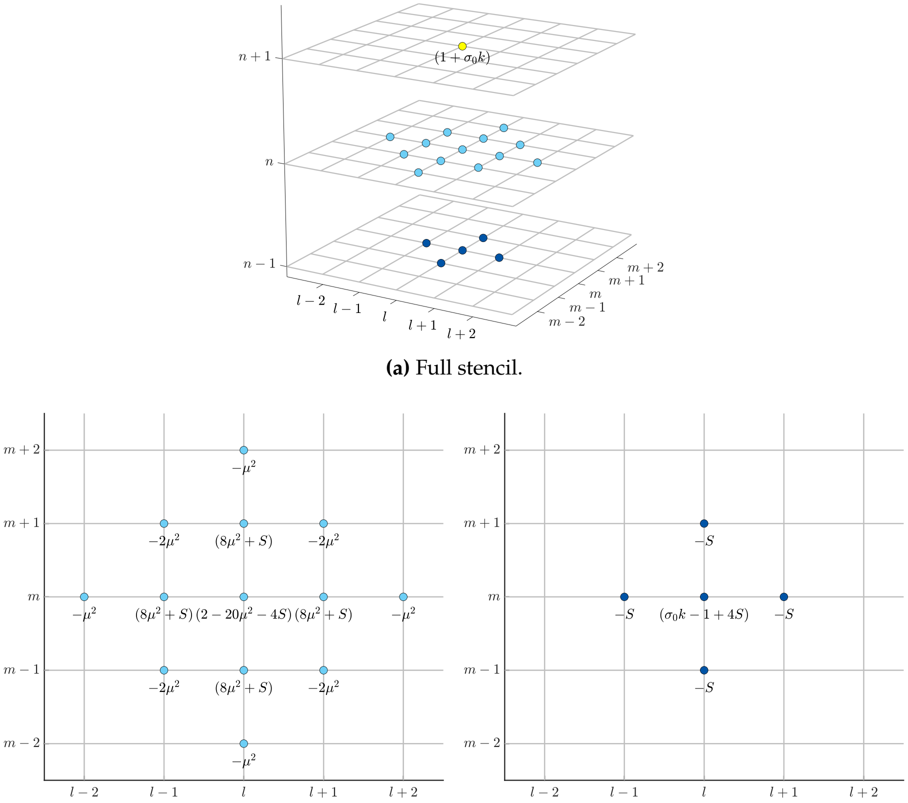
where

$$\mu = \frac{\kappa k}{h^2} \tag{6.43}$$

and $S = 2\sigma_1 k/h^2$ for compactness. See Figure 6.9 for the stencil of this scheme. The stability condition of the scheme can be shown to be

$$h \geq 2\sqrt{k\left(\sigma_1 + \sqrt{\kappa^2 + \sigma_1^2}\right)}, \tag{6.44}$$

and will be derived in Section 6.3.4.



(a) Full stencil.

(b) Stencil of $u_{l,m}^n$.

(c) Stencil of $u_{l,m}^{n-1}$.

Fig. 6.9: The stencil of the plate with coefficients corresponding to those in update equation (6.42). (a) A full overview of the stencil. (b) The current time-step n highlighted. (c) The previous time-step $n-1$ highlighted.

Discrete boundary conditions

The boundary conditions shown in Eq. (6.38) can be discretised to

$$\left.\begin{aligned}
u_{l,m}^n &= \delta_{x+}u_{l,m}^n = 0 \quad \text{if } m = 0 \quad &&\forall l\\
u_{l,m}^n &= \delta_{x-}u_{l,m}^n = 0 \quad \text{if } m = N_y \quad &&\forall l\\
u_{l,m}^n &= \delta_{y+}u_{l,m}^n = 0 \quad \text{if } l = 0 \quad &&\forall m\\
u_{l,m}^n &= \delta_{y-}u_{l,m}^n = 0 \quad \text{if } l = N_x \quad &&\forall m
\end{aligned}\right\} \quad \text{(Clamped)}, \tag{6.45a}$$

$$\left.\begin{aligned}
u_{l,m}^n &= \delta_{xx}u_{l,m}^n = 0 \quad \text{if } m = \{0, N_y\} \quad &&\forall l\\
u_{l,m}^n &= \delta_{yy}u_{l,m}^n = 0 \quad \text{if } l = \{0, N_x\} \quad &&\forall m
\end{aligned}\right\} \quad \text{(Simply supported)}. \tag{6.45b}$$

The clamped condition can be implemented by simply reducing the discrete range of operation to $l = \{2, \ldots, N_x - 2\}$ and $m = \{2, \ldots, N_y - 2\}$. For the simply supported case, the range of operation reduces to $l = \{1, \ldots, N_x - 1\}$ and $m = \{1, \ldots, N_y - 1\}$, and similar to the simply supported stiff string described in Section 4.2.1, the virtual grid points needed for this condition become

$$\begin{aligned} u^n_{-1,m} &= -u^n_{1,m} \quad \text{and} \quad u^n_{N_x+1,m} = -u^n_{N_x-1,m} \quad \forall m, \\ u^n_{l,-1} &= -u^n_{l,-1} \quad \text{and} \quad u^n_{l,N_y+1} = -u^n_{l,N_y-1} \qquad \forall l. \end{aligned}$$

6.3.3 Matrix form and output

Similar to the implementation of the 2D wave equation in Section 6.2.3, one can use a stacked state vector. If simply supported boundary conditions are used, one can easily obtain a matrix form of the $\delta_\Delta\delta_\Delta$ operator by multiplying two $\mathbf{D}_\Delta$ matrices presented in Eq. (6.20) to get $\mathbf{D}_{\Delta\Delta} = \mathbf{D}_\Delta\mathbf{D}_\Delta$.

Using a stacked form of the state as described in Eq. (6.16) the scheme in Eq. (6.40) in matrix form is

$$A\boldsymbol{u}^{n+1} = \mathbf{B}\boldsymbol{u}^n + \mathbf{C}\boldsymbol{u}^{n-1}, \tag{6.46}$$

where

$$\begin{aligned} A = (1 + \sigma_0 k), \quad \mathbf{B} = 2\mathbf{I} - \kappa^2 k^2 \mathbf{D}_{\Delta\Delta} + 2\sigma_1 k \mathbf{D}_\Delta, \\ \text{and} \quad \mathbf{C} = -(1 - \sigma_0 k)\mathbf{I} - 2\sigma_1 k \mathbf{D}_\Delta, \end{aligned}$$

and the identity matrix $\mathbf{I}$ is of the same size as $\mathbf{D}_{\Delta\Delta}$ and $\mathbf{D}_\Delta$.

Name	Symbol (unit)	Value
Side length x	L_x (m)	1.5
Side length y	L_y (m)	1
Material density	ρ (kg/m^3)	7850
Thickness	H (m)	$5 \cdot 10^{-3}$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Poisson's ratio	ν (-)	0.3
Freq.-independent damping	σ_0 (s^{-1})	1
Freq.-dependent damping	σ_1 (m^2/s)	0.005

Table 6.1: Parameters for the thin plate and possible values to use as a starting point for the simulation.

As a starting point for implementation, possible parameters are given in Table 6.1. Figure 6.10 shows the wave propagation of a thin plate using these parameters, and excited using the same excitation as used for the 2D wave

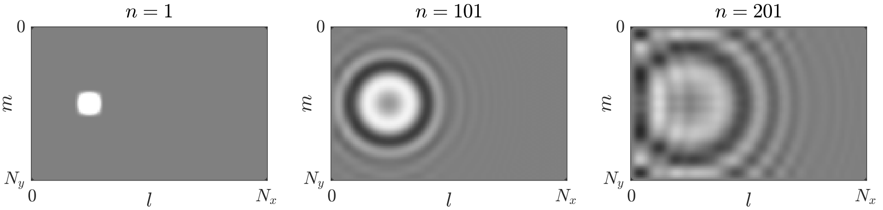


Fig. 6.10: Wave propagation of a thin plate with simply supported boundary conditions and parameters as shown in Table 6.1. The system is excited with a 2D raised cosine at $(x, y) = (0.25L_x, 0.5L_y)$ and dispersive effects are apparent.

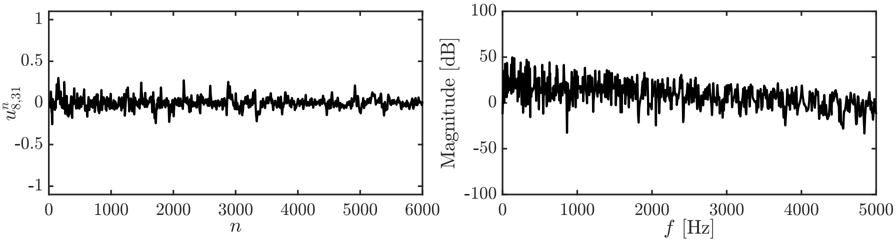


Fig. 6.11: The output of the thin plate at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.10.

equation in Section 6.2.3 (a 2D raised cosine at $(x, y) = (0.25L_x, 0.5L_y)$). When compared to Figure 6.4, dispersive effects – where higher-frequency components travel faster than lower-frequency ones – are apparent due to stiffness.

Figure 6.11 shows the time domain and frequency domain output of the thin plate at $(x, y) = (0.15L_x, 0.85L_y)$. Compared to the output of the 2D wave equation in Figure 6.5, there are several interesting differences. The amplitude is much lower, waves are closer together and the first wave arrives much earlier, all due to dispersive effects.

Modal analysis

Although the results will not be given here, a modal analysis can be performed on the thin plate by using the matrix form in Eq. (6.46), and writing this in a one-step form described in Section 3.5.1.

6.3.4 Frequency domain analysis

This section follows the process presented in Section 3.3 with the extensions to 2D shown in 6.2.4.

Using Eqs. (6.25) and (6.26) one can obtain a frequency domain representation of the FD scheme in Eq. (6.40) and obtain the following characteristic equation

$$\begin{aligned}(1+\sigma_0 k)z + &\left(16\mu^2(p_x+p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x+p_y) - 2\right)\\ &+ \left(1-\sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x+p_y)\right)z^{-1} = 0.\end{aligned} \tag{6.47}$$

This can, similar to the damped stiff string in Section 4.3, be solved to

$$4\mu^2(p_x+p_y)^2 + \frac{4\sigma_1 k}{h^2}(p_x+p_y) \leq 1.$$

Recalling the definitions p_x and p_y in Eq. (6.23), and given the fact that these are bounded by 1, the following can be written:

$$4\mu^2(1+1)^2 + \frac{4\sigma_1 k}{h^2}(1+1) \leq 1$$

and solved for h:

$$h \geq 2\sqrt{k\left(\sigma_1 + \sqrt{\sigma_1^2+\kappa^2}\right)}, \tag{6.48}$$

which is the stability condition given in Eq. (6.44).

6.3.5 Energy analysis

Using the steps described in Section 3.4 with the extensions to 2D presented in Section 6.2.5, one can obtain the total energy of the FD scheme in Eq. (6.40).

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

To obtain the rate of change of the total energy, one can take an inner product of the scheme in Eq. (6.40) with $(\delta_{t\cdot}u_{l,m}^n)$ over discrete (2D) domain $d = \{0,\ldots,N_x\}\times\{0,\ldots,N_y\}$ to get

$$\begin{aligned}\delta_{t+}\mathfrak{h} = \rho H\langle \delta_{t\cdot}u_{l,m}^n, \delta_{tt}u_{l,m}^n\rangle_d + D\langle \delta_{t\cdot}u_{l,m}^n, \delta_\Delta\delta_\Delta u_{l,m}^n\rangle_d\\ + 2\sigma_0\rho H\langle \delta_{t\cdot}u_{l,m}^n, \delta_{t\cdot}u_{l,m}^n\rangle_d - 2\sigma_1\rho H\langle \delta_{t\cdot}u_{l,m}^n, \delta_{t-}\delta_\Delta u_{l,m}^n\rangle_d = 0.\end{aligned} \tag{6.49}$$

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

Due to the damping present in the system and because the system is distributed in space, the energy balance will be of the following form

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q},$$

with boundary term $\mathfrak{b}$ and damping term

$$\mathfrak{q} = 2\sigma_0 \rho H \|\delta_{t\cdot} u_{l,m}^n\|_d^2 - 2\sigma_1 \rho H \langle \delta_{t\cdot} u_{l,m}^n, \delta_{t-}\delta_\Delta u_{l,m}^n \rangle_d. \tag{6.50}$$

Expanding the stiffness term in Eq. (6.49) to

$$\begin{aligned} & D\langle \delta_{t\cdot} u_{l,m}^n, (\delta_{xx} + \delta_{yy})\delta_\Delta u_{l,m}^n \rangle_d \\ \iff \quad & D\left(\langle \delta_{t\cdot} u_{l,m}^n, \delta_{xx}\delta_\Delta u_{l,m}^n \rangle_d + \langle \delta_{t\cdot} u_{l,m}^n, \delta_{yy}\delta_\Delta u_{l,m}^n \rangle_d \right), \end{aligned}$$

one can perform summation by parts twice, using Eq. (3.16b) for both terms to get

$$D\left(\langle \delta_{t\cdot}\delta_{xx} u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{\overline{d_x}}} + \langle \delta_{t\cdot}\delta_{yy} u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{\overline{d_y}}} \right) + \mathfrak{b}.$$

The definitions for the reduced domains can be found in Eqs. (6.31). Finally, as the boundaries are always 0 due to the boundary conditions in Eq. (6.45), $\underline{\overline{d_x}}$ and $\underline{\overline{d_y}}$ can be further reduced to $\underline{d}$ and the terms can be combined as

$$D\langle \delta_{t\cdot}\delta_\Delta u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{d}} + \mathfrak{b},$$

and using identities (3.17a) and (3.17b) a definition for the total energy can be found:

$$\begin{aligned} \mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho H}{2} \left\| \delta_{t-} u_{l,m}^n \right\|_d^2 \quad \text{and} \\ \mathfrak{v} = \frac{D}{2} \langle \delta_\Delta u_{l,m}^n, e_{t-}\delta_\Delta u_{l,m}^n \rangle_{\underline{d}}\,. \end{aligned} \tag{6.51}$$

The definition of the boundary term $\mathfrak{b}$ will not be given here, but can be shown to vanish under the boundary conditions given in Eq. (6.45) [21].

Step 3: Check units

As $\mathfrak{t}$ is identical to its definition in Eq. (6.32), only the units for $\mathfrak{v}$ will be checked here. Recalling that $D = EH^3/12(1-\nu^2)$, which in units is kg$\cdot$ m$^2\cdot$s^{-2}, yields

$$\begin{aligned} \mathfrak{v} = \frac{D}{2} \langle \delta_\Delta u_{l,m}^n, e_{t-}\delta_\Delta u_{l,m}^n \rangle_{\underline{d}} \xrightarrow{\text{in units}} & \, \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{m}^2 \cdot (\text{m}^{-2} \cdot \text{m}) \cdot (\text{m}^{-2} \cdot \text{m}) \\ = & \, \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \end{aligned}$$

and shows that $\mathfrak{v}$ indeed has the correct units.

Step 4: Implementation

Figure 6.12 shows the energetic output of an implementation of the thin plate. Notice that the damping present in the system causes $\mathfrak{h}$ to decrease.

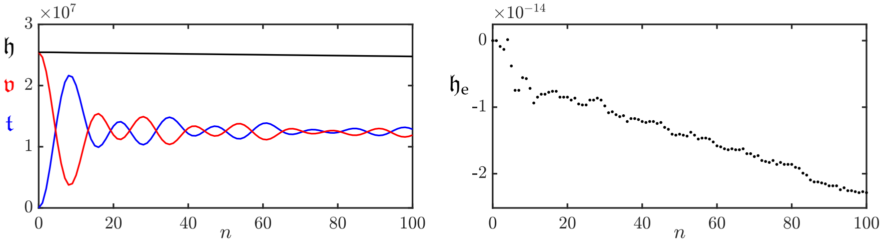


Fig. 6.12: The kinetic (blue), potential (red), and total (black) energy of an implementation of the thin plate are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

6.4 The stiff membrane

The term *stiff membrane*, as defined in [1], is essentially a 2D version of the stiff string. It can be used to model membranes with dispersive effects or provide tension control for thin plates. In this work, the stiff membrane has only been used in paper [F] to model a drum membrane.

Similar to previous sections, this section will provide the continuous-time and discrete-time equations of the model. Only a frequency domain analysis will be given, as energy analysis (and modal analysis) are too similar to those previously presented.

6.4.1 Continuous time

The PDE for a stiff membrane can be obtained as a combination of the 2D wave equation in Eq. (6.6) and the thin plate in Eq. (6.36). Adding losses as in Eq. (6.37) yields the following PDE

$$\rho H \partial_t^2 u = T\Delta u - D\Delta\Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u, \tag{6.52}$$

where the parameters are identical to those in Eqs. (6.6) (with $c = \sqrt{T/\rho H}$) and (6.37).

6.4.2 Discrete time

Using familiar operators, Eq. (6.52) can be discretised to

$$\rho H \delta_{tt} u_{l,m}^n = T\delta_\Delta u_{l,m}^n - D\delta_\Delta\delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-}\delta_\Delta u_{l,m}^n, \tag{6.53}$$

or, using a more compact form after division by ρH, to

$$\delta_{tt} u_{l,m}^n = c^2 \delta_\Delta u_{l,m}^n - \kappa^2 \delta_\Delta\delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \delta_{t-}\delta_\Delta u_{l,m}^n, \tag{6.54}$$

where $c = \sqrt{T/\rho H}$ and $\kappa = \sqrt{D/\rho H}$.

The update equation can then be obtained using the expansions of the Laplacian and biharmonic operators in Eqs. (6.4) and (6.5) respectively, to get

$$\begin{aligned} u_{l,m}^{n+1} = {} & (2 - 4\lambda^2 - 20\mu^2 - 4S)u_{l,m}^n \\ & + (\lambda^2 + 8\mu^2 + S)(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n) \\ & - 2\mu^2(u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n) \\ & - \mu^2(u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n), \\ & + (\sigma_0 k - 1 + 4S)u_{l,m}^{n-1} \\ & - S(u_{l+1,m}^{n-1} + u_{l-1,m}^{n-1} + u_{l,m+1}^{n-1} + u_{l,m-1}^{n-1}) \end{aligned} \tag{6.55}$$

where

$$\lambda = \frac{ck}{h} \quad \text{and} \quad \mu = \frac{\kappa k}{h^2} \tag{6.56}$$

and again, $S = 2\sigma_1 k/h^2$ for compactness. The stability condition for this scheme will be given in Section 6.4.4.

6.4.3 Implementation

Writing Eq. (6.54) in matrix form, yields

$$A\boldsymbol{u}^{n+1} = \mathbf{B}\boldsymbol{u}^n + \mathbf{C}\boldsymbol{u}^{n-1}, \tag{6.57}$$

with

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

Notice that the only difference with Eq. (6.46) is the addition of the wave speed term in the definition of the $\mathbf{B}$ matrix.

6.4.4 Frequency domain analysis

Following familiar techniques from Sections 3.3 and 6.2.4, the characteristic equation of the FD scheme in Eq. (6.54) can be obtained:

$$\begin{aligned} (1 + \sigma_0 k)z + & \left(4\lambda^2(p_x + p_y) + 16\mu^2(p_x + p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x + p_y) - 2\right) \\ & + \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x + p_y)\right) z^{-1} = 0. \end{aligned} \tag{6.58}$$

Similar to the stiff string in Section 4.3 and the thin plate in Section 6.3.4, this can be solved to

$$\lambda^2(p_x+p_y)+4\mu^2(p_x+p_y)^2+\frac{4\sigma_1 k}{h^2}(p_x+p_y)\leq 1,$$

and recalling that p_x and p_y are bounded by 1, yields

$$2\lambda^2+16\mu^2+\frac{8\sigma_1 k}{h^2}\leq 1.$$

Finally, recalling the definitions for λ and μ from Eq. (6.56), this can be rewritten in terms of h to get the following stability condition for the stiff membrane:[3]

$$h\geq\sqrt{c^2k^2+4\sigma_1 k+\sqrt{(c^2k^2+4\sigma_1 k)^2+16\kappa^2k^2}}. \tag{6.59}$$

[3] This stability condition was wrong in paper [F]. It has been corrected here and included in Appendix A.

Part III

Exciters

Exciters

As mentioned in Chapter 1, nearly all musical instruments can be subdivided into an exciter and a resonator component. Several resonators have been introduced in Part II, and different mechanisms to excite these are introduced in this part. First, various examples of physically inspired excitations will be presented in Chapter 7, some of which made a brief appearance in Parts I and II. Chapter 8 introduces the bow as an excitation mechanism and presents the contribution made in paper [C]: the elasto-plastic friction model applied to a FDTD-based stiff strings. Finally, Chapter 9 presents the lip reed as a way to excite brass instruments.

Chapter 7

Physically-Inspired Excitations

This short chapter introduces several simple ways to excite the different resonators presented in Part II. First, various ways to excite resonators using initial conditions are provided, after which examples of time-varying excitations are given, such that the systems can also be excited later during the simulation.

7.1 Initial conditions

The easiest way to excite a system is to set its initial conditions to non-zero values. This has been done several times before using a raised cosine (see e.g. Section 2.4.3). To give the system an initial displacement, not an initial velocity, one initialises both u_l^0 and u_l^1 with the same values. In the following, the 1D wave equation will be used as an example (Eq. (2.38)):

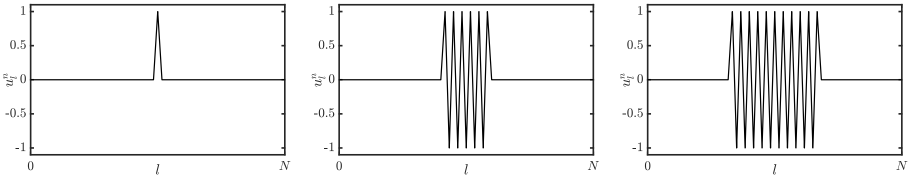
$$\partial_t^2 u = c^2 \partial_x^2 u \tag{7.1}$$

where $u = u(x,t)$ is the state of the system defined for $t \geq 0$ and $x \in D$ with domain $\mathcal{D} = [0, L]$ and length L (in m). Furthermore, $c = 735$ m/s. Following 2.2.1, the state variable can be discretised to u_l^n where $n \in \mathbb{N}^0$ and $l \in d$ with discrete domain $d = \{0, \dots, N\}$ and number of grid points $N+1$. The FD scheme is (Eq. (2.42))

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n. \tag{7.2}$$

7.1.1 Impulse

The simplest way to excite a system is to set the value of one grid point to non-zero, which is referred to as an *impulse* excitation. Figure 7.1 shows an implementation of the 1D wave equation where $u_l^0 = u_l^1 = 1$ at $l = \lfloor 0.5N \rfloor$. One can observe that the variations between two neighbouring grid points are extremely high. If the CFL condition in Eq. (2.46) is satisfied with equality, the system will exhibit high amounts of energy around the Nyquist frequency, which is generally unwanted.



(a) $n = 1$. **(b)** $n = 6$. **(c)** $n = 11$.

Fig. 7.1: The 1D wave equation initialised with an impulse at $l = \lfloor 0.5N \rfloor$.

7.1.2 Raised cosine

To avoid the high-frequency behaviour caused by the impulse, a spatially smooth excitation must be created. The *raised cosine* is most often used for this property and is extensively used throughout the literature [21]. Initialising the displacement of a distributed system with a raised cosine can be interpreted as a pluck. A different pluck excitation is presented in Section 7.1.3.

A raised cosine is parametrised by its amplitude e_{amp}, its center location x_0 and its width x_{w}. Applied to a distributed 1D system defined over domain $x \in \mathcal{D}$, the (continuous-time) excitation function containing a raised cosine is defined as

$$e_{\text{rc}}(x) = \begin{cases} \frac{e_{\text{amp}}}{2}\left(1 - \cos\left(\frac{2\pi(x - x_{\text{s}})}{x_{\text{w}}}\right)\right), & \text{if } x_{\text{s}} \leq x \leq x_{\text{e}}, \\ 0, & \text{otherwise,} \end{cases} \tag{7.3}$$

where

$$x_{\text{s}} = x_0 - \frac{x_{\text{w}}}{2}, \quad \text{and} \quad x_{\text{e}} = x_0 + \frac{x_{\text{w}}}{2} \tag{7.4}$$

are the start and end locations of the excitation. Furthermore, $x_{\text{s}}, x_{\text{e}} \in \mathcal{D}$, which puts a constraint on the width and location of the excitation.

In discrete time[1], the center location is defined as $l_0 = \lfloor x_0/h \rfloor$, where $\lfloor \cdot \rfloor$ denotes the flooring operation, h is the grid spacing. The discrete start and

[1] One could sample the continuous function in Eq. (7.3) at locations $x = lh$. In this chapter, looking towards straightforward implementation, a more practical approach has been chosen, and discrete definitions are given separately.

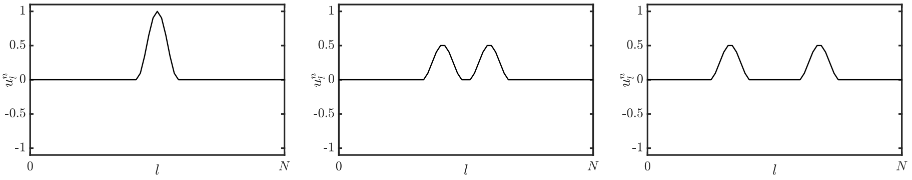
end locations of the raised cosine are

$$l_{\text{s}} = l_0 - \lfloor w/2 \rfloor \quad \text{and} \quad l_{\text{e}} = l_0 + \lfloor w/2 \rfloor, \tag{7.5}$$

with $l_{\text{s}}, l_{\text{e}} \in d$ and, finally, $w = \lfloor x_{\text{w}}/h \rfloor$.[2] Equation (7.3) can then be discretised as

$$E_{l,\text{rc}} = \begin{cases} \frac{e_{\text{amp}}}{2}\left(1 - \cos\left(\frac{2\pi(l-l_{\text{s}})}{w}\right)\right), & \text{if } l_{\text{s}} \leq l \leq l_{\text{e}}, \\ 0, & \text{otherwise.} \end{cases} \tag{7.6}$$

Figure 7.2 shows the 1D wave equation initialised with a raised cosine with $e_{\text{amp}} = 1$, $l_0 = \lfloor 0.5N \rfloor$ and $w = \lfloor 0.1N \rfloor$. Notice that the behaviour is much more smooth than the impulse shown in Figure 7.1.



(a) $n = 1$.

(b) $n = 6$.

(c) $n = 11$.

Fig. 7.2: The 1D wave equation initialised with a raised cosine at the center of the system.

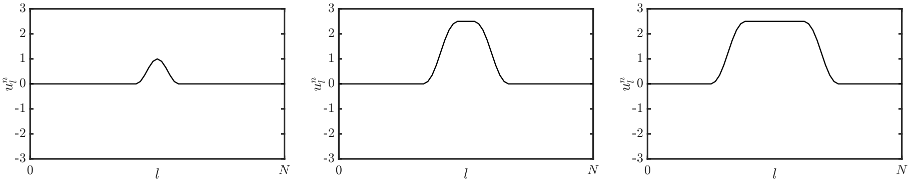
Strike

If one initialises the system with an initial velocity, i.e., only setting a displacement at $n = 1$, and leaving $u_l^0 = 0$ for $l \in d$, one can use the raised cosine to model a strike. Figure 7.3 shows a strike using the same values as for the pluck in Figure 7.2 at $n = 1$, but leaving $u_l^0 = 0$. One can observe that, for the pluck, the displacement of the system stays high rather than going back to 0 as in the case of the pluck. Furthermore, the amplitude of the displacement is higher than for the pluck (notice the scaling of the y-axis).

7.1.3 Triangular pluck

Another way to initialise the string, which is closer to reality, is to use a triangular shape to model a pluck [1, 21].Using e_{amp} as the maximum displacement – at the corner of the triangle – and $x_0 \in \mathcal{D}$ as the plucking position, the

[2] Notice that the width is given in 'number of grid spacings' and will thus affect $w + 1$ grid points. This is also why the range in Eq. (7.6) includes both end points.



(a) $n = 1$.   **(b)** $n = 6$.   **(c)** $n = 11$.

Fig. 7.3: The 1D wave equation initialised with a strike at the center of the system. Notice the scaling of the y-axis compared to Figure 7.2.

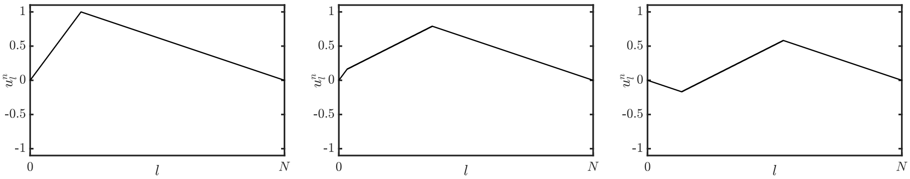
triangular excitation can be defined as

$$e_{\text{tri}}(x) = \begin{cases} \frac{e_{\text{amp}}}{x_0}x, & \text{if } 0 \leq x \leq x_0, \\ \frac{e_{\text{amp}}}{x_0 - L}(x - L), & \text{if } x_0 < x \leq L. \end{cases} \tag{7.7}$$

In discrete time, Eq. (7.7) becomes

$$E_{l,\text{tri}} = \begin{cases} \frac{e_{\text{amp}}}{l_0}l, & \text{if } 0 \leq l \leq l_0, \\ \frac{e_{\text{amp}}}{l_0 - N}(l - N), & \text{if } l_0 < l \leq N. \end{cases} \tag{7.8}$$

Due to the spatial discontinuity at the corner, some high-frequency oscillations (similar to the impulse) might emerge. Figure 7.4 shows an implementation of the 1D wave equation initialised with a triangular pluck excitation. The sample rate is 441 kHz (10x the usual) to prevent these high-frequency oscillations from appearing in the plot (though they still exist to some degree).



(a) $n = 1$.   **(b)** $n = 101$.   **(c)** $n = 201$.

Fig. 7.4: The 1D wave equation initialised with a triangular pluck with $l_0 = \lfloor 0.2N \rfloor$. Note that sample rate has been set to $f_{\text{s}} = 441000$ Hz to show a more ideal triangular motion.

7.1.4 2D raised cosine

Introducing an extra coordinate y, one can extend the raised cosine presented in Section 7.1.2 to 2D according to

$$e_{\text{rc}}(x,y) = \begin{cases} \frac{e_{\text{amp}}}{2}\left(1-\cos\left(\frac{2\pi(x-x_{\text{s}})}{r_{\text{w}}}\right)\right)\left(1-\cos\left(\frac{2\pi(y-y_{\text{s}})}{r_{\text{w}}}\right)\right), & \text{if } x_{\text{s}} \leq x \leq x_{\text{e}}, \\ & \text{and } y_{\text{s}} \leq y \leq y_{\text{e}}, \\ 0, & \text{otherwise,} \end{cases} \tag{7.9}$$

where r_{w} is the excitation radius. Similar to Eq. (7.4), the start and end locations of the raised cosine in the x and y direction can be calculated as

$$x_{\text{s}} = x_0 - \frac{r_{\text{w}}}{2}, \quad x_{\text{e}} = x_0 + \frac{r_{\text{w}}}{2}, \quad y_{\text{s}} = y_0 - \frac{r_{\text{w}}}{2}, \quad \text{and} \quad y_{\text{e}} = y_0 + \frac{r_{\text{w}}}{2},$$

where (x_0, y_0) is the center coordinate and $x_{\text{s}}, x_{\text{e}}, y_{\text{s}}, y_{\text{e}} \in \mathcal{D}$ for a 2D domain $\mathcal{D}$.

In discrete time, Eq. (7.9) becomes

$$E_{(l,m),\text{rc}} = \begin{cases} \frac{e_{\text{amp}}}{2}\left(1-\cos\left(\frac{2\pi(l-l_{\text{s}})}{r}\right)\right)\left(1-\cos\left(\frac{2\pi(m-m_{\text{s}})}{r}\right)\right), & \text{if } l_{\text{s}} \leq l \leq l_{\text{e}}, \text{ and} \\ & m_{\text{s}} \leq m \leq m_{\text{e}}, \\ 0, & \text{otherwise,} \end{cases} \tag{7.10}$$

where $r = \lfloor r_{\text{w}}/h \rfloor$ is the discrete excitation radius (in 'number of grid spacings'). Furthermore, similar to Eq. (7.5),

$$l_{\text{s}} = l_0 - \lfloor r/2 \rfloor, \; l_{\text{e}} = l_0 + \lfloor r/2 \rfloor, \; m_{\text{s}} = m_0 - \lfloor r/2 \rfloor, \; \text{and } m_{\text{e}} = m_0 + \lfloor r/2 \rfloor \tag{7.11}$$

for discrete center coordinate (l_0, m_0) and $l_{\text{s}}, l_{\text{e}}, m_{\text{s}}, m_{\text{e}} \in d$ for a discrete 2D domain d. As in the 1D case, this excitation can be used to model a simple 'pluck' and strike for a 2D system. See Figure 7.5 for a visualisation of a 2D raised cosine. A simple way to implement the 2D raised cosine in `MATLAB` using the `hann` function is shown in Algorithm 7.1.

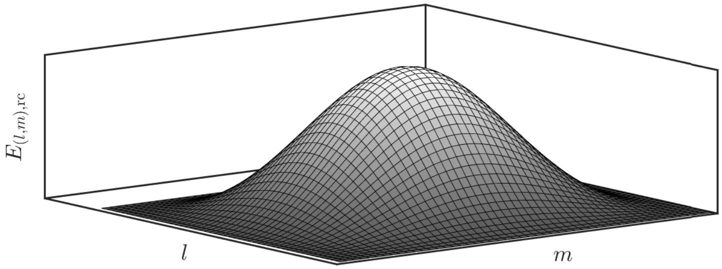


Fig. 7.5: A 2D raised cosine created using Eq. (7.10).

```
% Assuming Dirichlet boundary conditions and having initialised the
    following
% - center locations for the x and y-directions: l0 and m0
% - radius of the excitation r (in grid points)

ls = l0 - floor(r/2); % start location x-direction
le = l0 + floor(r/2); % end location x-direction
ms = m0 - floor(r/2); % start location y-direction
me = m0 + floor(r/2); % end location x-direction

% Create excitation matrix
e = zeros(Ny-1, Nx-1);

% Add one to hann function as the width is given in 'grid spacings'
% and affects r+1 grid points
e(ms:me, ls:le) = hann(r+1) * hann(r+1)';

% Applying excitation to stacked matrix form as in Section 6.2.3
u = reshape(e, (Nx-1) * (Ny-1), 1);
```

Algorithm 7.1: A MATLAB implementation of a 2D raised cosine.

7.2 Time-varying excitations

Although various types of excitation can already be modelled using the initial conditions presented in the previous section, they are temporally rigid. In other words, the time of excitation is fixed to be at the start of the simulation. In order to excite the system while the simulation is running, one can create excitations that – on top of being spatially distributed – have a temporal profile as well.

For the following, consider the ideal string of length L (in m), where its transverse displacement is described by $u = u(x,t)$ (in m). The system is defined for $t \geq 0$ and $x \in \mathcal{D}$ with domain $\mathcal{D} = [0, L]$. The PDE of the ideal string with a time varying external force $f(t)$ (in N) is defined as (Eq. (2.38))

$$\rho A \partial_t^2 u = T \partial_x^2 u + e(x) f(t) \tag{7.12}$$

where $e(x)$ is a spatial distribution function such as those presented in Section 7.1 (in m^{-1}).

In discrete time, Eq. (7.12) becomes

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n + E_l f^n \tag{7.13}$$

with discrete spatial distribution E_l and discrete time-varying external force f^n.

7.2.1 Raised cosine

To yield a smooth excitation over time, one can, similar to the spatially distributed raised cosine in Eq. 7.1.2, define a temporally distributed raised cosine. Using the time of excitation $t_0 \geq 0$ and excitation duration $t_\mathrm{d} > 0$ (both in s), the temporal raised cosine can be used as a force function, as

$$f(t) = \begin{cases} \frac{f_\mathrm{amp}}{2}\left(1 - \cos\left(\frac{q\pi(t-t_0)}{t_\mathrm{d}}\right)\right), & t_\mathrm{s} \leq t \leq t_0 + t_\mathrm{d} \\ 0, & \text{otherwise.} \end{cases} \tag{7.14}$$

As done in [64, 65], q alters the excitation to be a pluck when $q = 1$ and a strike when $q = 2$. Finally, f_amp is the maximum force (in N).

As done in papers [A] and [B], the force function can be used in conjunction with the distribution functions shown in Section 7.1. When used to scale a spatially distributed raised cosine as in Eq. (7.3), one can model a pluck or a strike, by setting $q = 1$ or $q = 2$ respectively. These alternatives are visualised in Figure 7.6. Note, that if one would like f_amp to be the maximum force, one must set $e_\mathrm{amp} = 1$ in Eq. (7.3).

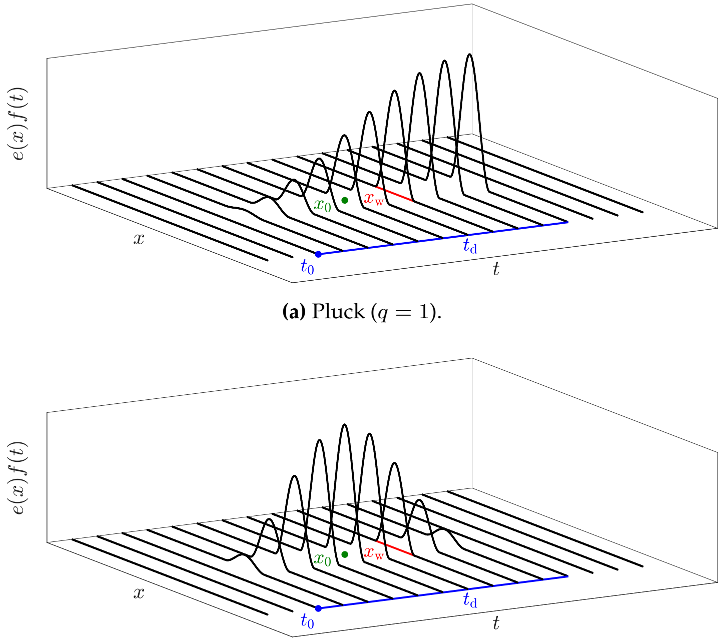
In discrete time, the force function in Eq. (7.14) becomes

$$f^n = \begin{cases} \frac{f_\mathrm{amp}}{2}\left(1 - \cos\left(\frac{q\pi(n-n_0)}{n_\mathrm{d}}\right)\right), & n_0 \leq n \leq n_0 + n_\mathrm{d}, \\ 0, & \text{otherwise.} \end{cases} \tag{7.15}$$

where $n_\mathrm{d} = \lfloor t_\mathrm{d}/k \rfloor$ is the duration of the excitation in samples.

7.2.2 Pulse train

As already briefly introduced in Section 5.1.3, one can create a pulse train to excite an acoustic tube. This is inspired by [21] where the signal represents the opening and closing of the glottis. As the characteristics of the lip reed are



(a) Pluck ($q = 1$).

(b) Strike ($q = 2$).

Fig. 7.6: The time-varying raised cosine showing a (a) pluck and a (b) strike. The location of excitation x_0 is shown in green, the width x_{w} in red and the excitation start t_0 and duration t_{d} in blue.

similar to the vocal folds [80], the pulse train has been used as a test signal here (also see Section 9.3.2). A more complete model of the lip reed can be found in Chapter 9.

The pulse train can be created using a clipped sinusoid, which can be used as the input velocity to an acoustic tube. Algorithm 7.2 shows an example of how to generate a pulse train. The frequency as well as the duty cycle (how much of the signal is non-zero) can be set. Figure 7.7 shows the output of the algorithm.

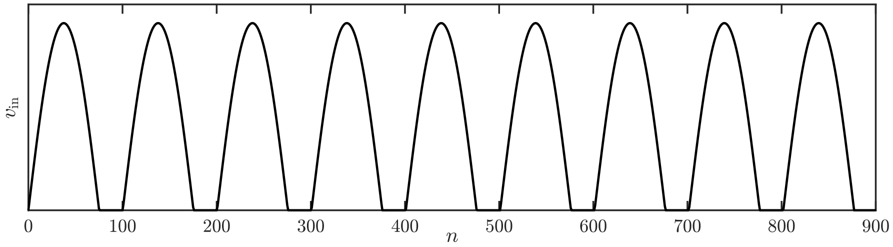


Fig. 7.7: The pulse train generated using Algorithm 7.2 ($f = 440$ Hz, duty cycle $= 75\%$).

```
%% Pulse train generator

fs = 44100;                 % Sample rate [Hz]
lengthSound = fs;           % Length of the sound [samples]
f = 440;                    % Pulse train frequency
dutyC = 0.75;               % Duty cycle [0-1]
amp = 1;                    % Amplitude

%% Create input signal
for n = 0:lengthSound
    if mod(n, fs / f) <= fs / f * dutyC
        vIn(n+1) = amp * sin(f * pi / dutyC * mod(n, fs / f) / fs);
    else
        vIn(n+1) = 0;
    end
end
```

Algorithm 7.2: `MATLAB` code to generate a pulse train.

Chapter 8

The Bow

The bow is an extremely interesting excitation mechanism from a simulation perspective. A bow excites a string with a force due to friction, which introduces a nonlinear element into the system. The string 'sticks' to the bow and 'slips' again when the restoring force of the string is too great and consequently overcomes the friction force. This 'stick-slip' behaviour, first coined by Bowden and Leben in 1939 [81] 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 in the 19th century, which later got named *Helmholtz motion* in his honour [82].[1]

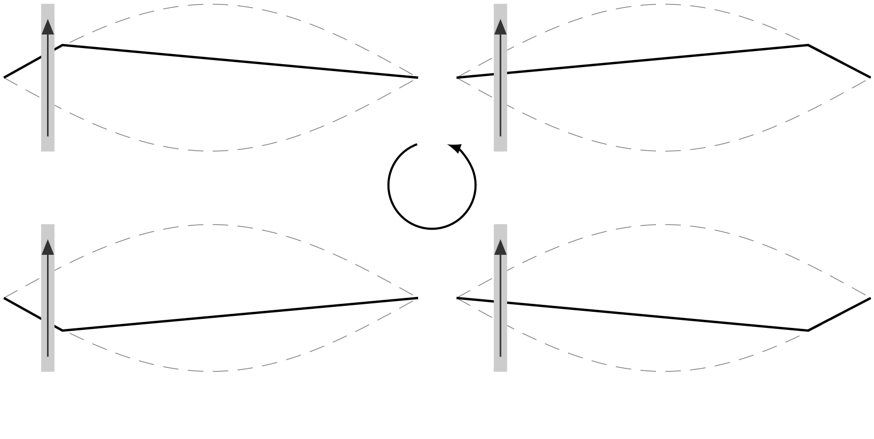


Fig. 8.1: Helmholtz motion. If the bow moves up on the left side of the string, the 'Helmholtz corner' travels anti-clockwise.

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

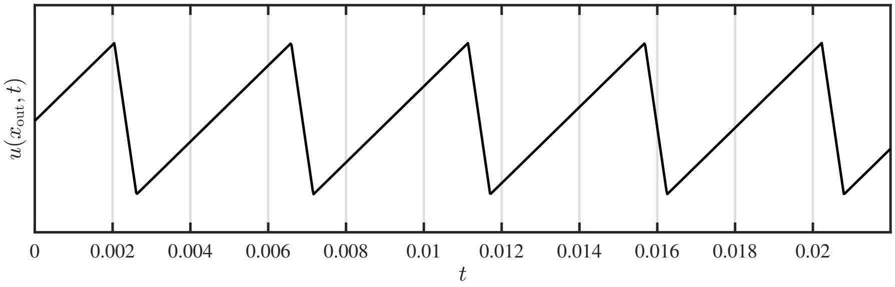


Fig. 8.2: An example of the (ideal) Helmholtz motion of a location x_{out} along a bowed string.

The Helmholtz motion gives bowed string instruments, such as the violin and cello, their characteristic sound. An ideal case is shown in Figure 8.2, where, for a string described by $u(x, t)$, the location of a point x_{out} along the string follows a sawtooth-like motion due to the 'stick-slip' behaviour.

The rest of this chapter is structured as follows: first, a brief history of bowed-string simulations is presented. Then, an introduction to interpolation and spreading operators, as well as an introduction to the Newton-Raphson method will be given, both of which are necessary to work with bow-string interaction. Finally, a static and a dynamic friction model are presented. The latter, the elasto-plastic friction model, was applied to a FD scheme of the stiff string during this project and is one of the contributions, published in paper [C].

8.1 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 proposed by Smith in 1986 and used digital waveguides for the string and a look-up table for the friction model [83]. Simultaneously, Florens et al. presented 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 [84] (see Section 8.4). One of the most complex friction models applied in a musical context to-date is the elasto-plastic friction model proposed by Dupont [85], which Serafin et al. [86, 87] applied to a digital waveguide implementation of the string.

The first appearance of FDTD methods in bowed string simulations was in a publication by Pitteroff and Woodhouse in [88]. Later, Maestre et al. in [29] used FDTD methods to implement a thermal friction model proposed by Woodhouse in [89]. In both cases, the string 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 [52, 51], but did not implement this in real-time. The first real-time implementation of a bowed stiff string fully modelled using FDTD methods was presented in paper [A]. Finally, paper [C] presented the first (real-time) implementation of the elasto-plastic friction model applied to a FD scheme in a musical context.

8.2 Interpolation and spreading operators

This section summarises and extends [21, Sec. 5.2.4 pp. 101–104].

To listen to, or interact with a FD scheme between grid points, it is necessary to use some form of interpolation. To this end, an *interpolation operator* $I(x_\text{i})$ can be introduced and can applied to a grid function [21]. This operator is a function of x_i, the (continuous) location of interest and can be defined in various levels of accuracy. In this section, a 1D system $u(x,t)$ is assumed where $x \in \mathcal{D}$ for spatial domain $\mathcal{D}$. The theory presented in this section will be extended to 2D in Section 11.1.

An interpolation operator can be applied to a grid function u_l^n, which performs the following operation:

$$I_{l,o}(x_\text{i})u_l^n = \sum_{l\in d} I_{l,o}(x_\text{i}) \cdot u_l^n. \tag{8.1}$$

Here, o is the order of the operator, and $l \in d$ with discrete domain d, which needs to be the same for both $I_{l,o}(x_\text{i})$ and u_l^n.

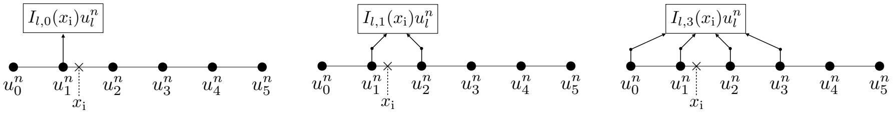
The simplest interpolation operator is of '0th'-order and is defined as

$$I_{l,0}(x_\text{i}) = \begin{cases} 1, & \text{if } l = l_\text{i}, \\ 0, & \text{otherwise}, \end{cases} \tag{8.2}$$

where the grid location of interest is defined as $l_\text{i} = \lfloor x_\text{i}/h \rfloor$ (see Figure 8.3a). Instead of actually performing an interpolation operation, I_0 simply floors its input to the grid location below. A slightly more accurate way to perform 0th-order interpolation is to round x_i/h to the nearest integer, rather than to use the flooring operation.

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

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


$I_{l,0}(x_\mathrm{i})u_l^n$
u_0^n u_1^n u_2^n u_3^n u_4^n u_5^n
x_i
$I_{l,1}(x_\mathrm{i})u_l^n$
u_0^n u_1^n u_2^n u_3^n u_4^n u_5^n
x_i
$I_{l,3}(x_\mathrm{i})u_l^n$
u_0^n u_1^n u_2^n u_3^n u_4^n u_5^n
x_i

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

Fig. 8.3: Interpolation with varying orders of accuracy.

See Figure 8.3b.

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

$$I_{l,3}(x_\mathrm{i}) = \begin{cases} -\alpha_\mathrm{i}(\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}-2)/6, & l = l_\mathrm{i}-1, \\ (\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i}, \\ -\alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i}+1, \\ \alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-1)/6, & l = l_\mathrm{i}+2, \\ 0, & \text{otherwise.} \end{cases} \tag{8.4}$$

See Figure 8.3c. Notice that the sum of all values of $I_{l,o}(x_\mathrm{i})$ add up to 1, regardless of the order of the interpolator or the value of α_i.

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

Apart from interpolation operators, one may define *spreading operators* which can be interpreted as an inverse interpolation operation. A spreading operator $J(x_\mathrm{i})$ is used to interact with a distributed FD scheme in the form of an excitation or other interactions, such as connections or collisions between multiple schemes (also 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

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

a linear spreading operator

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

and a Lagrange cubic spreading operator

$$J_{l,3}(x_\mathrm{i}) = \frac{1}{h}\begin{cases} -\alpha_\mathrm{i}(\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}-2)/6, & l = l_\mathrm{i}-1, \\ (\alpha_\mathrm{i}-1)(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i}, \\ -\alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-2)/2, & l = l_\mathrm{i}+1, \\ \alpha_\mathrm{i}(\alpha_\mathrm{i}+1)(\alpha_\mathrm{i}-1)/6, & l = l_\mathrm{i}+2, \\ 0, & \text{otherwise}. \end{cases} \tag{8.7}$$

Notice that the spreading operator uses a scaling by $1/h$. An intuition this will be given in Section 11.1.1. As is the case for interpolation operators, higher-order spreading operators reduce to Eq. (8.5) if $\alpha_\mathrm{i} = 0$.

Spreading operators approximate the spatial Dirac delta function $\delta(x - x_\mathrm{i})$ (in m^{-1}), which is a test function defined as

$$\delta(x) = \begin{cases} \infty, & x = 0, \\ 0, & \text{otherwise}, \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x)dx = 1, \tag{8.8}$$

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 (8.8) will not be used directly. Instead, it can be approximated using the spreading operators presented in this section.

Identities

The following identity is extremely useful when solving systems including interpolation and spreading operators of the same order o for any (grid) function f and discrete domain d:

$$\langle f_l, J_{l,o}(x_\mathrm{i})\rangle_d = I_{l,o}(x_\mathrm{i})f_l, \tag{8.9}$$

where $l \in d$. From this, it follows that taking the norm of a spreading operator over a given domain, is identical to applying to its 'dual' interpolation operator (of the same order o and same input x_i):

$$\langle J_{l,o}(x_\mathrm{i}), J_{l,o}(x_\mathrm{i})\rangle_d = \|J_{l,o}(x_\mathrm{i})\|_d^2 = I_{l,o}(x_\mathrm{i})J_{l,o}(x_\mathrm{i}). \tag{8.10}$$

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

Other distributions

This section presented interpolation and spreading operators that interact with the state of a FD scheme at a single location x_i and either interpolate or distribute over a range of points. Although multiple grid points might be used for these operations, the interpolation or spreading is not *distributed*. Physical exciters such as mallets or bows have a non-zero width and thus interact with a larger part of the system. One could make an arbitrary distribution function E with elements e_l (in 1D) where $l \in d$ for discrete domain d of the system at hand. The *distribution* and spreading operators become

$$I_l = \frac{e_l}{\sum_d e_l} \quad \text{and} \quad J_l = \frac{1}{h} I_l. \tag{8.11}$$

Although any values for E would work, the sum of E needs to be normalised to 1 to retain correct scaling, as shown in Eq. (8.11).

8.3 The Newton-Raphson method

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

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

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \tag{8.12}$$

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:

$$|x_{i+1} - x_i| < \epsilon, \tag{8.13}$$

where ϵ is small, but its value depends on the situation at hand. To prevent Newton Raphson from iterating endlessly (if the iteration can not converge), 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 (non-uniqueness) or many local variations.

Algorithm 8.1 shows an example of an implementation of Newton-Raphson using $f(x) = e^x - 1 \;\Rightarrow\; f'(x) = e^x$ and Figure 8.4 visualises the iterative algorithm.

```
% 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 number of
% iterations, do not iterate more
maxIterations = 100;

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

    % calculate next iteration (Eq. (8.12)
    xNext = x - (exp(x) - 1) / (exp(x));

    % threshold check (Eq. (8.13)
    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 $f(x) = e^x - 1$.

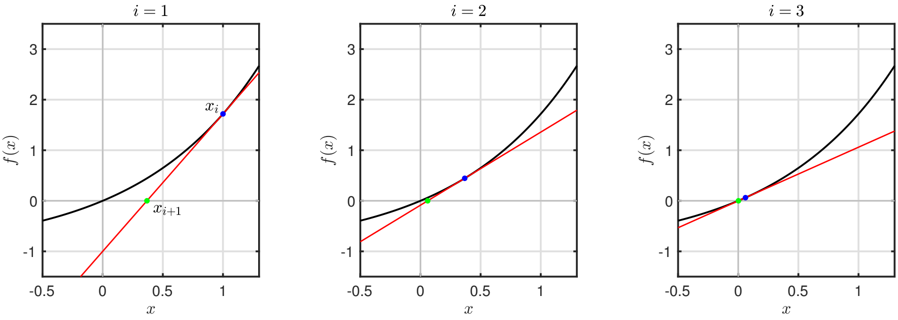


Fig. 8.4: The Newton-Raphson method. The x-value of the root of the tangent line at $f(x_i)$ is used to evaluate the next iteration.

8.3.1 Multivariate Newton-Raphson

For M functions f_m dependent on the same number of independent variables x_m with $m = \{1, \dots, M\}$, Newton-Raphson can be extended to

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \underbrace{\begin{bmatrix} \frac{\partial f_1(\mathbf{x}_i)}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x}_i)}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x}_i)}{\partial x_1} & \dots & \frac{\partial f_M(\mathbf{x}_i)}{\partial x_M} \end{bmatrix}^{-1}}_{\mathbf{J}(\mathbf{x}_i)} \begin{bmatrix} f_1(\mathbf{x}_i) \\ \vdots \\ f_M(\mathbf{x}_i) \end{bmatrix}, \tag{8.14}$$

where the column vector $\mathbf{x} = [x_1, \dots, x_M]^T$ is the collection of independent variables, and the iteration number is (again) denoted by i. The matrix in Eq. (8.14) 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]:

$$f_1(\mathbf{x}) = 3x_1 - \cos(x_2 x_3) - 3/2 = 0, \tag{8.15a}$$
$$f_2(\mathbf{x}) = 4x_1^2 - 625x_2^2 + 2x_3 - 1 = 0, \tag{8.15b}$$
$$f_3(\mathbf{x}) = 20x_3 + e^{-x_1 x_2} + 9 = 0, \tag{8.15c}$$

where $\mathbf{x} = [x_1, x_2, x_3]^T$. 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. (8.14).

8.4 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 [90] to which static friction, or *stiction*, was added by Morin in 1833 [91] and viscous friction, or velocity-dependent friction, by Reynolds in 1886 [92]. 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 [93]. The latter is still the standard for static friction models today.

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

Many static friction models contain a discontinuity where the relative velocity between the $v_{\text{rel}} = 0$, due to a multiplication with sgn(v_{rel}) in their definition. In this project, only the following static friction model has been used [21]

$$\Phi(v_{\text{rel}}) = \sqrt{2a}v_{\text{rel}}e^{-av_{\text{rel}}^2+1/2}, \tag{8.16}$$

as it is continuous and differentiable, but still approximating discontinuous bow models. These characteristics make this model easier to work with in implementation. A definition for v_{rel} will be given below.

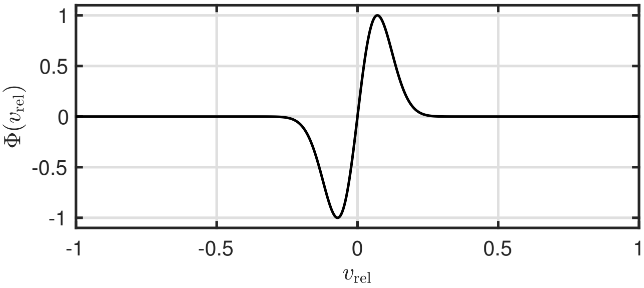


Fig. 8.5: The friction model in Eq. (8.16) with $a = 100$.

8.4.1 The bowed stiff string

Consider a stiff 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 can then be described as

$$v_{\text{rel}} = \partial_t u(x_{\text{B}}, t) - v_{\text{B}}(t) \tag{8.17}$$

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

Recalling the PDE of the stiff string in Eq. (4.3)

$$\rho A \partial_t^2 u = T\partial_x^2 u - EI\partial_x^4 u - 2\sigma_0\rho A\partial_t u + 2\sigma_1\rho A\partial_t\partial_x^2 u, \tag{8.18}$$

one can add the bow force to the equation according to

$$\rho A \partial_t^2 u = T\partial_x^2 u - EI\partial_x^4 u - 2\sigma_0\rho A\partial_t u + 2\sigma_1\rho A\partial_t\partial_x^2 u - \delta(x - x_{\text{B}})f_{\text{B}}\Phi(v_{\text{rel}}) \tag{8.19}$$

where spatial Dirac delta function $\delta(x - x_{\text{B}})$ (in m^{-1}) (see Section 8.2) positions the bow along the string and $f_{\text{B}} = f_{\text{B}}(t) \geq 0$ is the bow force (in N).[3]

[3] If the spatial Dirac delta function were omitted, the bow force would be applied to the entire string domain rather than only the bow location x_{B}.

Intuition

From Eq. (8.19) it can be seen that the bow force gets scaled by the friction model $\Phi(v_{\text{rel}})$ shown in Figure 8.5. The figure shows that if v_{rel} is too large (either positively or negatively), the bow term in (8.19) 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[4]. 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.

It might seem counter-intuitive that the bow term is subtracted from the scheme. This is due to the definition of the relative velocity in Eq. (8.17). For a negative bow velocity v_{B}, v_{rel} becomes positive. As $\text{sgn}\left(\Phi(v_{\text{rel}})\right) = \text{sgn}(v_{\text{rel}})$ through Eq. (8.16) and $f_{\text{B}} \geq 0$, the term $\delta(x - x_{\text{B}})f_{\text{B}}\Phi(v_{\text{rel}})$ will be positive for a positive v_{rel}. As a negative v_{B} needs to have a downwards, or negative, effect on the string, the sign of the bow term also needs to be negative to achieve this.

Discrete time

Dividing all terms in Eq. (8.20) by ρA and discretising the system yields

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_l(x_{\text{B}}^n)F_{\text{B}}^n\Phi(v_{\text{rel}}^n), \tag{8.20}$$

with $F_{\text{B}}^n = f_{\text{B}}^n/\rho A$ and spreading operator $J_l(x_{\text{B}}^n) = J_{l,o}(x_{\text{B}}^n)$ (in m^{-1}) as described in Section 8.2, where the order o remains undetermined for now. As the bow position, bow velocity and bow force are time-dependent, these have a superscript n in discrete time. These parameters are called *control parameters* and will be supplied by the performer in an eventual implementation.

The relative velocity in Eq. (8.17) is discretised using a centred difference operator according to

$$v_{\text{rel}}^n = I_l(x_{\text{B}}^n)\delta_{t\cdot}u_l^n - v_{\text{B}}^n, \tag{8.21}$$

with interpolation operator $I_l(x_{\text{B}}^n) = I_{l,o}(x_{\text{B}}^n)$ and is of the same order as $J_l(x_{\text{B}}^n)$. Equation (8.21) makes the scheme implicit due to the centred difference operator as the FD scheme in Eq. (8.20) is now nonlinearly dependent on u_l^{n+1}. To solve Eq. (8.20) for u_l^{n+1}, an iterative root-finding algorithm is required, such as Newton-Raphson described in Section 8.3. This process could be circumvented by using a backward difference operator in Eq. (8.21), but this will affect the accuracy and behaviour of the bow model.

[4] The terms 'static' and 'dynamic' used in this sentence, do not relate to a 'static' or 'dynamic' friction model. Rather it refers to the friction for an object at rest (static), versus that of an object in motion (dynamic).

Solution

To find a solution for u_l^{n+1} at the bow location, an inner product of the scheme in Eq. (8.20) with spreading operator $J_l(x_{\text{B}}^n)$ must be taken over the discrete domain of the string d which isolates the scheme at the bowing location. Performing this operation and using identity (8.9) yields

$$\begin{aligned} I_l(x_{\text{B}}^n)\delta_{tt}u_l^n = {} & c^2 I_l(x_{\text{B}}^n)\delta_{xx}u_l^n - \kappa^2 I_l(x_{\text{B}}^n)\delta_{xxxx}u_l^n - 2\sigma_0 I_l(x_{\text{B}}^n)\delta_{t\cdot}u_l^n \\ & + 2\sigma_1 I_l(x_{\text{B}}^n)\delta_{t-}\delta_{xx}u_l^n - \|J_l(x_{\text{B}}^n)\|_d^2 F_{\text{B}}^n \Phi(v_{\text{rel}}^n). \end{aligned} \tag{8.22}$$

One can rewrite Eq. (8.21) to

$$I_l(x_{\text{B}}^n)\delta_{t\cdot}u_l^n = v_{\text{rel}}^n + v_{\text{B}}^n, \tag{8.23}$$

and using identity (2.27a), Eq. (8.22) can be rewritten and assigned to a function $g(v_{\text{rel}}^n)$ according to

$$g(v_{\text{rel}}^n) = \left(\frac{2}{k} + 2\sigma_0\right) v_{\text{rel}}^n + \|J_l(x_{\text{B}}^n)\|_d^2 F_{\text{B}}^n \Phi(v_{\text{rel}}^n) + b^n = 0, \tag{8.24}$$

where the terms not dependent on v_{rel} are collected in[5]

$$\begin{aligned} b^n = {} & -\frac{2}{k} I_l(x_{\text{B}}^n)\delta_{t-}u_l^n - c^2 I_l(x_{\text{B}}^n)\delta_{xx}u_l^n + \kappa^2 I_l(x_{\text{B}}^n)\delta_{xxxx}u_l^n \\ & + \left(\frac{2}{k} + 2\sigma_0\right) v_{\text{B}}^n - 2\sigma_1 I_l(x_{\text{B}}^n)\delta_{t-}\delta_{xx}u_l^n. \end{aligned} \tag{8.25}$$

One can then perform the Newton-Raphson method detailed in Section 8.3 to iteratively solve for v_{rel}

$$(v_{\text{rel}}^n)_{i+1} = (v_{\text{rel}}^n)_i - \frac{g\left((v_{\text{rel}}^n)_i\right)}{g'\left((v_{\text{rel}}^n)_i\right)}\ , \tag{8.26}$$

where

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

and

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

[5] An interpolation operator applied to a spatial derivative can be expanded in a similar fashion to when it is applied to a grid function. A first-order interpolator applied to $\delta_{xx}u_l^n$ thus yields $I_{l,1}(x_{\text{i}})\delta_{xx}u_l^n = (1-\alpha_{\text{i}})\delta_{xx}u_{l_{\text{i}}}^n + \alpha\delta_{xx}u_{l_{\text{i}}+1}^n$.

8.4.2 Implementation and output

To implement the bowed stiff string, one must perform the Newton-Raphson iteration every sample. For the implementation shown in this section, the threshold in Eq. (8.13) has been set to $\epsilon = 10^{-7}$ and the maximum number of iterations to 100. In the following, the parameters used for the string are listed in Table 4.1 with $T = 1000$ N and those for the bow are

$$x_{\mathrm{B}}^n = 1/8, \quad f_{\mathrm{B}}^n = 1\ \mathrm{N}, \quad v_{\mathrm{B}}^n = 0.2\ \mathrm{m/s}, \quad \text{and} \quad a = 100.$$

Furthermore, 0th-order interpolation and spreading operator are used.

Figure 8.6 shows the behaviour of a bowed stiff string at the beginning of the simulation and shows the characteristic stick-slip behaviour. Figure 8.7 shows the state of the same simulation 3 seconds later. One can observe the Helmholtz corner move in an anti-clockwise direction as presented in Figure 8.1. Finally, the time domain output of the string at the bowing location shown in Figure 8.8 generally follows the Helmholtz motion shown in Figure 8.2.

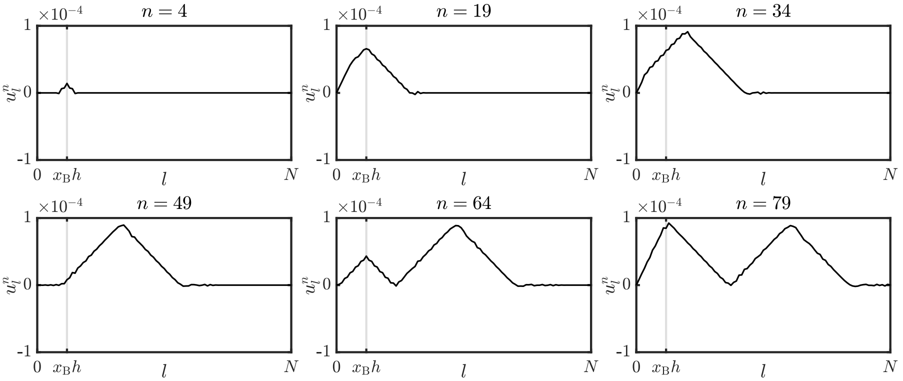


Fig. 8.6: Behaviour of a bowed stiff string at the start of the simulation.

8.4.3 Energy analysis

Following the energy analysis of the stiff string presented in Section 4.4, taking an inner product of Eq. (8.20) (after multiplication by ρA) with $(\delta_{t\cdot} u_l^n)$ over

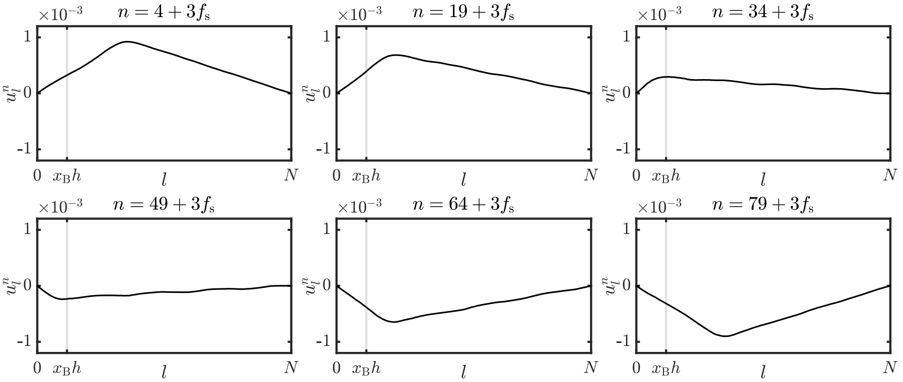


Fig. 8.7: Behaviour of a bowed stiff string simulation after 3 seconds. The string exhibits a Helmholtz motion as presented in Figure 8.1.

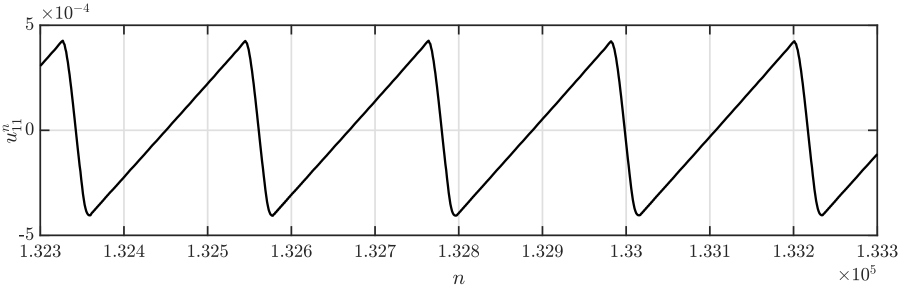


Fig. 8.8: Time domain output at the bowing location of a bowed stiff string using the static friction model in Eq. (8.16).

discrete domain d one arrives at the following

$$
\begin{aligned}
\delta_{t+}\mathfrak{h} + \mathfrak{q} &= -\langle (\delta_{t\cdot}u_l^n), J_l(x_\text{B}^n) f_\text{B}^n \Phi(v_\text{rel}^n)\rangle_d \\
\xLeftrightarrow{\text{Eq. (8.9)}} \quad &= -I_l(x_\text{B}^n)(\delta_{t\cdot}u_l^n) f_\text{B}^n \Phi(v_\text{rel}^n) \\
\xLeftrightarrow{\text{Eq. (8.23)}} \quad &= \underbrace{-f_\text{B}^n \Phi(v_\text{rel}^n) v_\text{rel}^n}_{\text{loss}} \underbrace{-f_\text{B}^n \Phi(v_\text{rel}^n) v_\text{B}^n}_{\text{power}},
\end{aligned}
$$

where $\mathfrak{h}$ and $\mathfrak{q}$ are as defined in Eqs. (4.29) and (4.28) respectively. As $\text{sgn}\left(\Phi(v_\text{rel}^n)\right) = \text{sgn}\left(v_\text{rel}^n\right)$ through Eq. (8.16), one can observe that the first term on the right-hand side always has a negative effect on the rate of change of the total energy. This term can therefore be interpreted as the loss of power through the bow (as indicated). The last term is of indeterminate sign and can thus be interpreted as the power supplied by the bow.

The final energy balance can thus be written as

$$\delta_{t+}\mathfrak{h} = -\mathfrak{q} - \mathfrak{q}_\mathrm{B} - \mathfrak{p} \tag{8.27}$$

where

$$\mathfrak{q}_\mathrm{B} = f_\mathrm{B}^n \Phi(v_\mathrm{rel}^n) v_\mathrm{rel}^n \quad \text{and} \quad \mathfrak{p} = f_\mathrm{B}^n \Phi(v_\mathrm{rel}^n) v_\mathrm{B}^n.$$

Figure 8.9 shows the energy of the bowed stiff string corresponding to the wave propagation in Figure 8.6. The bow only injects energy into the system when it sticks to the string.

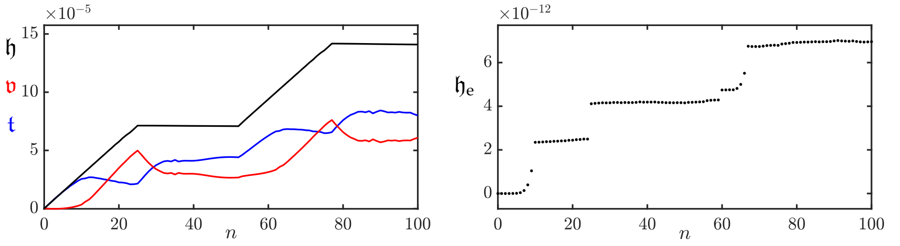


Fig. 8.9: The kinetic (blue), potential (red), and total (black) energy of the bowed stiff string. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

8.5 Dynamic friction models

As opposed to 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. This hysteresis loop is in the force versus velocity plane and has been confirmed by measurements using a bowing machine in [89].

The first dynamic friction model was proposed by Dahl [94] and captured hysteresis effects. The Stribeck effect (mentioned in Section 8.4) 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 [95, 96], and extended the Dahl model by taking 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 [85], 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.

The first appearance of a contribution in this thesis is the elasto-plastic friction model applied to a stiff string implemented using FDTD methods. This has been presented in paper [C] and will be summarised in this section. Furthermore, this section extends the paper by providing a stability analysis using the techniques presented in Section 3.4.4.

8.5.1 The elasto-plastic friction model

In a musical context, the elasto-plastic friction has been investigated in-depth by Serafin et al. in [86, 87, 97]. 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, all contributing a fraction of the total amount of friction. See Figure 8.10.

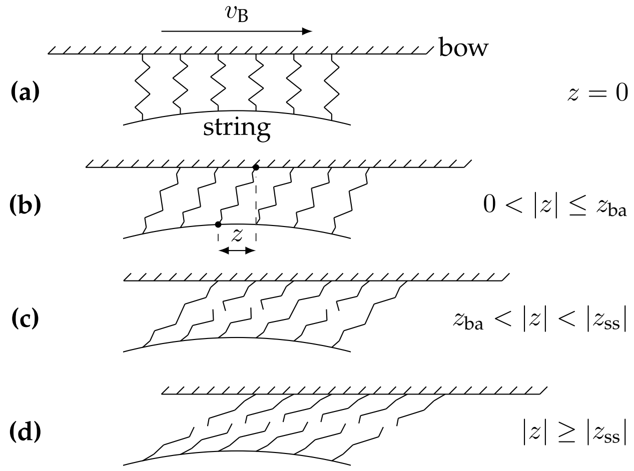

v_B
bow
(a)
$z = 0$
string
(b)
$0 < |z| \leq z_\mathrm{ba}$
z
(c)
$z_\mathrm{ba} < |z| < |z_\mathrm{ss}|$
(d)
$|z| \geq |z_\mathrm{ss}|$

Fig. 8.10: A visualisation of the microscopic displacements of the bristles between the bow and the string assumed by the elasto-plastic friction model. The bow moves right with a velocity of v_B. (a) The average bristle displacement $z = 0$. (b) The bow moves right relative to the string and the purely elastic, or 'presliding' regime, is entered (stick). (c) After $|z|$ gets larger than the break-away displacement z_ba, more and more bristles start to 'break'. This is defined as the elasto-plastic regime. (d) After $|z| \geq |z_\mathrm{ss}|$ all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered. (Adapted from paper [C].)

Unless denoted otherwise, this section follows the original model in [85], 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) as well as the relative velocity $v = v(t)$ (in m/s). The friction force is defined as

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{8.28}$$

with bristle stiffness $s_0 \geq 0$ (in N/m), bristle damping $s_1 \geq 0$ (in kg/s), viscous friction $s_2 \geq 0$ (in kg/s) and, as presented in [87], a dimensionless noise coefficient s_3 multiplied onto a pseudorandom function $w = w(t)$ (in N) generating values between -1 and 1. Moreover, for a string defined over domain $\mathcal{D}$, the relative velocity between the string at bowing location $x_\text{B} = x_\text{B}(t) \in \mathcal{D}$ and the bow is defined as (similar to Eq. (8.17))

$$v = \partial_t u(x_\text{B}, t) - v_\text{B}, \tag{8.29}$$

with bow velocity $v_\text{B} = v_\text{B}(t)$ (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

$$\dot{z} = r(v, z) = v\left[1 - \alpha(v, z)\frac{z}{z_\text{ss}(v)}\right]. \tag{8.30}$$

Here, z_ss is the steady-state function

$$z_\text{ss}(v) = \frac{\text{sgn}(v)}{s_0}\left[f_\text{C} + (f_\text{S} - f_\text{C})e^{-(v/v_\text{S})^2}\right], \tag{8.31}$$

where the v_S is the Stribeck velocity (in m/s). Furthermore, using the normal force $f_\text{N} = f_\text{N}(t)$ (in N), the Coulomb force and stiction force can be calculated according to $f_\text{C} = f_\text{N}\mu_\text{C}$ and $f_\text{S} = f_\text{N}\mu_\text{S}$ respectively (both in N). In these definitions, μ_C and μ_S are the dimensionless dynamic and static friction coefficients respectively. A plot of the steady state function can be found in Figure 8.11.

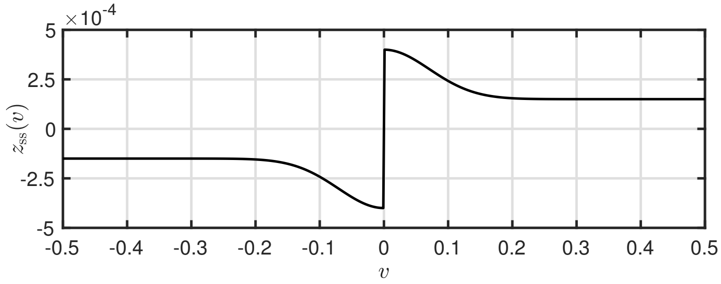


Fig. 8.11: The steady-state function $z_\text{ss}(v)$ plotted against relative velocity v with $s_0 = 10^4$, $\mu_\text{C} = 0.3$, $\mu_\text{S} = 0.8$, $v_\text{S} = 0.1$ and $f_\text{N} = 5$.

Finally, $\alpha(v, z)$ in Eq. (8.30) is an adhesion map between the bow and the string and is defined as

$$\alpha(v, z) = \begin{cases} \left.\begin{array}{ll} 0 & |z| \leq z_\text{ba} \\ \alpha_\text{m}(v, z) & z_\text{ba} < |z| < |z_\text{ss}(v)| \\ 1 & |z| \geq |z_\text{ss}(v)| \end{array}\right\} & \text{if sgn}(v) = \text{sgn}(z) \\ 0 & \text{if sgn}(v) \neq \text{sgn}(z), \end{cases} \tag{8.32}$$

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

$$\alpha_\mathrm{m} = \frac{1}{2}\left[1 + \mathrm{sgn}(z)\sin\left(\pi\frac{z - \mathrm{sgn}(z)\frac{1}{2}(|z_\mathrm{ss}(v)| + z_\mathrm{ba})}{|z_\mathrm{ss}(v)| - z_\mathrm{ba}}\right)\right], \tag{8.33}$$

where the break-away displacement $z_\mathrm{ba} = z_\mathrm{ba}(t) = 0.7 f_\mathrm{C}/s_0$ determines the value of z before bristles start to break. The adhesion map is visualised in Figure 8.12 and relates to Figure 8.10 as described in its caption.

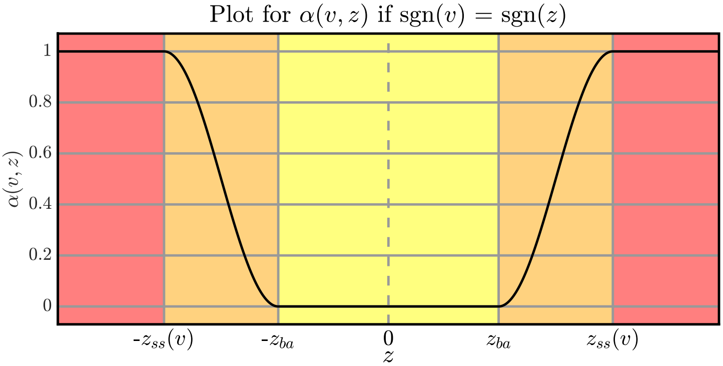


Fig. 8.12: A plot of the adhesion map $\alpha(v,z)$ in Eq. (8.32) plotted against z when $\mathrm{sgn}(v) = \mathrm{sgn}(z)$. The different coloured regions correspond to Figure 8.10 according to: yellow - a) & b), orange - c) and red - d). (Adapted from paper [C].)

Discrete time

Equation (8.28) can be discretised to

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n, \tag{8.34}$$

where Eq. (8.29) can be discretised to

$$v^n = I_l(x_\mathrm{B}^n)\delta_{t\cdot}u_l^n - v_\mathrm{B}^n, \tag{8.35}$$

with interpolation operator $I_l(x_\mathrm{B}^n) = I_{l,o}(x_\mathrm{B}^n)$ (of unspecified order o) and

$$r^n = r(v^n, z^n) = v^n\left[1 - \alpha(v^n, z^n)\frac{z^n}{z_\mathrm{ss}(v^n)}\right] \tag{8.36}$$

is the discrete counterpart of Eq. (8.30). The discrete adhesion map is identical to the continuous definition given in Eqs. (8.32) and (8.33), but with superscripts n added for appearances of v and z.

8.5.2 Applied to a FDTD stiff string

In the same way as done with the static friction model in Section 8.4, one can add the friction force to the stiff string PDE in Eq. (4.3) and discretise the system as follows:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_l(x_{\text{B}}^n)\frac{f(v^n,z^n)}{\rho A}, \tag{8.37}$$

where spreading operator $J_l(x_{\text{B}}^n) = J_{l,o}(x_{\text{B}}^n)$ and is of the same order as $I_l(x_{\text{B}}^n)$ in Eq. (8.35). Following the same procedure as for the static friction model in Section 8.4, one takes an inner product with $J_l(x_{\text{B}}^n)$ over discrete string domain d and using identities (8.9) and (2.27a), one can rewrite this similar to the static friction model in Eq. (8.24) as

$$g_1(v^n,z^n) = \left(\frac{2}{k}+2\sigma_0\right)v^n + \|J_l(x_{\text{B}}^n)\|_d^2\frac{f(v^n,z^n)}{\rho A} + b^n = 0, \tag{8.38}$$

where

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

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 8.3.1 must be performed. To be able to do this, an extra function, which is dependent on v^n and z^n, must be included.

As r describes $\dot{z}$ in Eq. (8.30), one can take another approach to approximate $\dot{z}$ using the trapezoid rule [21]

$$a^n = (\mu_{t-})^{-1}\delta_{t-}z^n \quad\Longrightarrow\quad a^n = \frac{2}{k}(z^n - z^{n-1}) + a^{n-1}. \tag{8.39}$$

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:

$$g_2(v^n,z^n) = r^n - a^n = 0. \tag{8.40}$$

Finally, one can use the multivariate Newton-Raphson method described in Section 8.3.1, which results in

$$\begin{bmatrix} v^n \\ z^n \end{bmatrix}_{i+1} = \begin{bmatrix} v^n \\ z^n \end{bmatrix}_i - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{8.41}$$

The derivatives in the Jacobian matrix are given in Appendix F.7.

8.5.3 Output

In the following, the same parameters as presented for the static friction model in Section 8.4.2 have been used for the string and the bow (where $f_N = f_\text{B}$). Additional parameters used for the elasto-plastic friction model are given in paper [C].

Figure 8.13 shows that the implementation of the elasto-plastic friction model exhibits a hysteresis loop in the force versus velocity plane, as desired from a dynamic friction model. The values around $v = 0$ are due to sticking behaviour and the loop on the left is due to slipping behaviour. Figure 8.14 shows the output of the implementation and follows the characteristic Helmholtz motion shown in Figure 8.2. When compared to the output of the static friction model in Figure 8.8, the output of the elasto-plastic implementation seems to be more 'smooth' overall, which could be explained by the inclusion of the bristles and their elasticity.

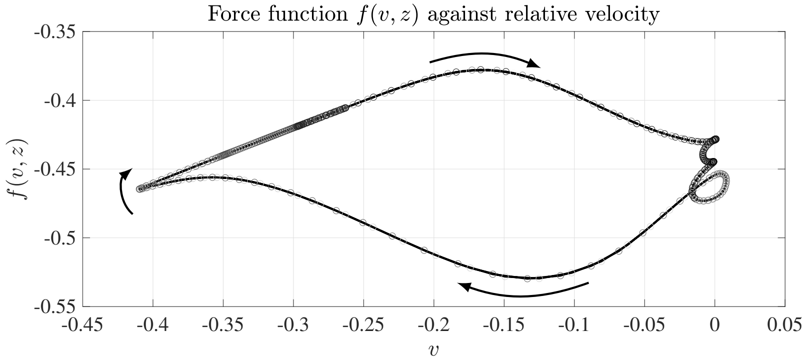


Fig. 8.13: Hysteresis loop showing 500 values up to $n = 3f_\text{s}$. (Adapted from paper [C].)

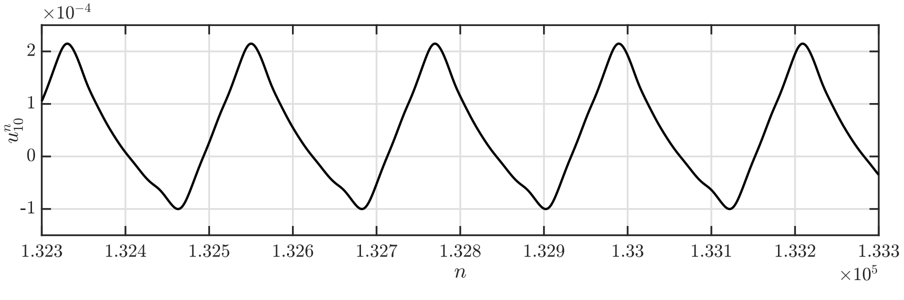


Fig. 8.14: Time domain output at the bow location of a stiff string bowed using an elasto-plastic friction model.

8.5.4 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 condition for the elasto-plastic friction model. A similar process for finding stability for the LuGre model has been done by Olsson in continuous time [98, p. 55]. The derivation below is inspired by his.

First, all terms of Eq. (8.37) 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 discrete string domain d is taken to get

$$\delta_{t+}\mathfrak{h}_\text{s} + \mathfrak{q}_\text{s} = -\mathfrak{p}_\text{B} \tag{8.42}$$

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

$$\mathfrak{p}_\text{B} = \langle(\delta_{t\cdot} u_l^n), J(x_\text{B}^n)f^n\rangle_d$$

which, using identity (8.9), can be written as

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

Finally, using Eq. (8.35) yields

$$\mathfrak{p}_\text{B} = f^n v^n + f^n v_\text{B}^n. \tag{8.43}$$

The term $f^n v^n$ is the important one as $f^n v_\text{B}^n$ 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 brevity.

Substituting Eq. (8.34) into fv and ignoring the noise term $s_3 w^n$ for now, yields

$$\mathfrak{p}_\text{B} = fv = \sigma_0 zv + \sigma_1 rv + \sigma_2 v^2. \tag{8.44}$$

The definition for r^n in Eq. (8.36) may be rewritten as

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

and (following Olsson) may be substituted in Eq. (8.44) as

$$\mathfrak{p}_{\text{B}} = s_0 z\left(r+\frac{v\alpha z}{z_{\text{ss}}(v)}\right)+s_1 r\left(r+\frac{v\alpha z}{z_{\text{ss}}(v)}\right)+s_2v^2, \tag{8.45}$$

or

$$\mathfrak{p}_{\text{B}} = s_0 zr+s_1\left(r+\frac{v\alpha z}{2z_{\text{ss}}(v)}\right)^2+\frac{v\alpha z^2}{z_{\text{ss}}(v)}\left(s_0-\frac{s_1v\alpha}{4z_{\text{ss}}(v)}\right)+s_2v^2. \tag{8.46}$$

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

$$\delta_{t+}\mathfrak{h}_{\text{brist}}+\mathfrak{q}_{\text{brist}}\geq 0,$$

which needs to be non-negative for passivity.

Starting with the damping term, which is defined as

$$\mathfrak{q}_{\text{brist}} = s_1\left(r+\frac{v\alpha z}{2z_{\text{ss}}(v)}\right)^2+\frac{v\alpha z^2}{z_{\text{ss}}(v)}\left(s_0-\frac{s_1v\alpha}{4z_{\text{ss}}(v)}\right)+s_2v^2, \tag{8.47}$$

one can show that, 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. (8.31)), $\mathfrak{q}_{\text{brist}}$ is non-negative under the following condition:

$$s_1\leq\frac{4s_0z_{\text{ss}}(v)}{v}\,. \tag{8.48}$$

This is the same condition as Olsson presents for the LuGre model [98], and means that as long as one knows the limit of the velocity of the system, the coefficient s_1 can be set accordingly.

Using identity (3.17b), the energy stored in the bristles can be shown to be

$$\mathfrak{h}_{\text{brist}}=\frac{s_0}{2}z^n e_{t-}z^n, \tag{8.49}$$

which is not necessarily non-negative. In [98], Olsson performs the analysis in continuous time, where the energy in the bristles is defined as

$$\mathfrak{H}_{\text{brist}} = \frac{s_0}{2} z^2, \tag{8.50}$$

which is clearly non-negative. Further work needs to be done to prove stability for the discretisation of the elasto-plastic friction model in Eq. (8.34).[6]

8.6 Discussion and conclusion

This chapter presented the bow as a mechanism to excite stiff strings. Two friction models have been presented: a static friction model where the friction force is only a function of the relative velocity between the string and the bow, and a dynamic elasto-plastic friction model, which relates this relative velocity to the friction force using a differential equation. The latter has been presented in paper [C] where it was first applied to stiff strings based on FDTD methods. Stability analysis for the elasto-plastic friction model is ongoing, although a stability condition for parameters of the model has been presented.

Although a successful implementation of the elasto-plastic friction model has been made, it has not been used in the project beyond paper [C]. It was found that small changes in parameters, both control and model parameters, already yield large behavioural changes. An attempt was made at using the elasto-plastic friction model with a fully modelled instrument (the tromba marina presented in papers [D] and [E]), but this did not yield the desired results, and the predictable static friction model was chosen instead. Future work includes tuning the many parameters that the elasto-plastic model relies on, and to apply this to fully modelled instruments.

An in-depth comparison between the static and the elasto-plastic friction model in terms of perceptual differences, is also left for future work. A preliminary comparison has been carried out by Onofrei in [99], where the author noted that *"the elasto-plastic model seems to behave more smoothly"* than the static friction model. This is also what is observed when comparing the output of the friction models in Figures 8.8 and 8.14. As said, these observations are preliminary, and further work needs to be done to compare the various friction models.

Finally, it must be noted that the applications of the bow are not limited to strings, and can very well be extended to other resonators. In [S4], for example, the authors apply the elasto-plastic friction model to a 2D drum membrane to simulate a friction drum inspired instrument.

[6] This could possibly include a different discretisation of $s_0 z$ in Eq. (8.28) or adding a mass to the bristles.

Chapter 9

The Lip Reed

The dynamics of wind instruments can be modelled by acoustic tubes as presented in Chapter 5. Excitation of these instruments happens either by blowing a jet of air across an opening, such as in a flute, or by the buzzing of a *reed*. In [1], the authors state that all wind-instrument reeds fall into one of three categories: the single reed, (clarinet, saxophone), the double reed (oboe, bassoon) and the lip reed (trumpet, trombone). The latter will be the focus of this chapter.

Sections 5.1.3 and 7.2.2 presented a physically inspired pulse train that attempts to model the opening and closing of the lips, using a clipped sinusoidal signal. A more physical approach, which is bidirectional, is to model the lips as a mass-spring-damper system that interacts with the left boundary of the tube. The literature describes lip reed models with varying degrees of freedom (DoF) (see [1, 62] for an overview). Recent work includes vortex-induced vibration into the lip reed model, that allows for the buzzing of the lips without the need of an acoustic tube [S1].

As the contribution of this project to a brass instrument model was mainly focused on the resonator, the simple 'outward striking door model' was chosen, which is a simple single one-DoF mass-spring-damper system. The model is as presented in [100], but excluding the collision. An alternative collision model was added in paper [H] and will be elaborated on in Chapter 16.

This chapter starts by introducing the mass-spring-damper system, after which the lip reed model will be given in continuous and discrete time. The lip reed will be coupled to the first-order system of equations presented in 5.2. Unless denoted otherwise, this chapter follows [62].

9.1 Mass-spring systems revisited: Damping

Before moving on to the lip reed system, the mass-spring system given in Section 2.3 will be extended to contain damping.

Recall the mass spring system presented in Eq. (2.28), where $u = u(t)$ is the displacement of the mass from its equilibrium position (in m). Damping can be easily be added to yield a mass-spring-damper system as follows:

$$M\ddot{u} = -Ku - R\dot{u}, \tag{9.1}$$

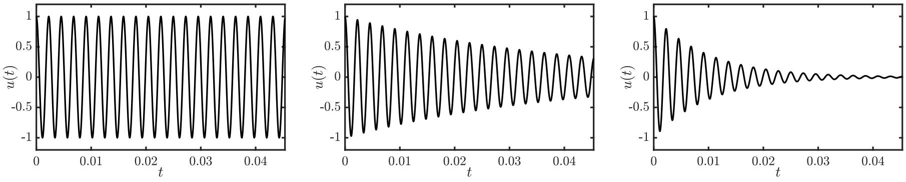
with mass M (in kg), spring constant K (in N/m) and damping coefficient R (in kg/s). Figure 9.1 shows the behaviour of the system for different values of R.

Equation (9.1) can then be discretised to the following FD scheme:

$$M\delta_{tt}u^n = -Ku^n - R\delta_{t\cdot}u^n. \tag{9.2}$$

Expanding and solving for u^{n+1} yields the following update equation:

$$\left(1 + \frac{Rk}{2M}\right)u^{n+1} = 2u^n - u^{n-1} - \frac{Kk^2}{M}u^n + \frac{Rk}{2M}u^{n-1}. \tag{9.3}$$



(a) $R = 0$.

(b) $R = 50$.

(c) $R = 200$.

Fig. 9.1: The mass-spring-damper system in Eq. (9.1) with $f_0 = 440$ Hz for different values of R.

9.1.1 Energy analysis

Following Section 3.4 (without explicitly following the steps for brevity), one can obtain the energy of Eq. (9.2) through a multiplication of the scheme by $(\delta_{t\cdot}u^n)$ to get

$$M(\delta_{tt}u^n)(\delta_{t\cdot}u^n) = -K(\delta_{t\cdot}u^n)u^n - R(\delta_{t\cdot}u^n)^2. \tag{9.4}$$

As there is damping present in the system, the energy balance will be of the form

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

Using identities (3.17a) and (3.17b), $\mathfrak{h}$ and $\mathfrak{q}$ can be obtained from Eq. (9.4)

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-}u^n)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}u^n e_{t-}u^n, \tag{9.5}$$

and

$$\mathfrak{q} = R(\delta_{t\cdot}u^n)^2. \tag{9.6}$$

Figure 9.2 shows the energy output of the mass spring damper system with $R = 50$ and $f_0 = 2\pi\sqrt{K/M} = 440$ Hz. One can observe that the damping term causes the system to lose energy when the mass is in motion (high kinetic energy).

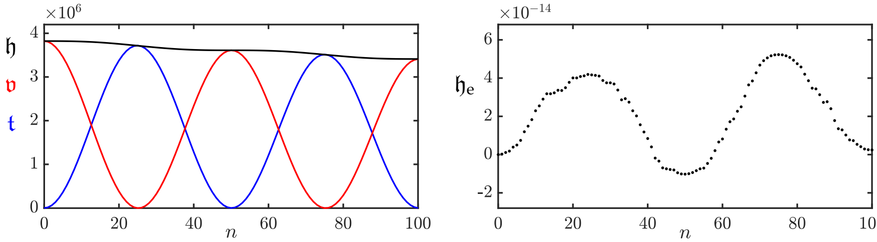


Fig. 9.2: The potential (red), kinetic (blue), and total (black) energy of the mass-spring-damper system. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

9.2 Continuous time

As mentioned at the beginning of this chapter, the lip reed will be modelled as a mass-spring-damper system as in Eq. (9.1). The system will be coupled to an acoustic tube described by the first-order system of PDEs described in Section 5.2, Eq. (5.37).

Using dots to denote derivatives with respect to time t, the PDE of the lip reed connected to an acoustic tube is defined as

$$M\ddot{y} = -Ky - R\dot{y} + S_\mathrm{r}\Delta p, \tag{9.7}$$

with displacement of the lip reed from equilibrium $y = y(t)$ (in m), mass of the lip reed $M > 0$ (in kg), lip stiffness $K \geq 0$ (in N/m), damping coefficient $R \geq 0$ (in kg/s), and effective surface area of the lip $S_\mathrm{r} \geq 0$ (in m^2). Furthermore,

$$\Delta p = \Delta p(t) = P_\mathrm{m} - p(0,t) \tag{9.8}$$

is the difference between the pressure in the mouth $P_\mathrm{m} = P_\mathrm{m}(t)$ and the pressure at the left boundary of the acoustic tube $p(0,t)$ (all in Pa). The

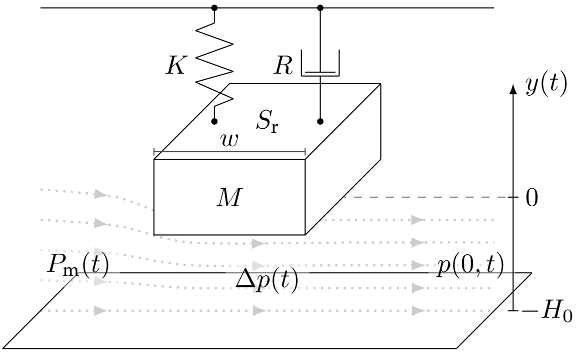


Fig. 9.3: Lip-reed system with parameters as appear in Section 9.2. (Adapted from paper [H].)

acoustic tube can be described by the first-order system presented in Section 5.2. See Figure 9.3 for a schematic representation of the lip reed.

The pressure difference in Eq. (9.8) causes a volume flow velocity (in m^3/s) and follows the Bernoulli equation

$$U_{\mathrm{B}} = U_{\mathrm{B}}(t) = w[y + H_0]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}}, \tag{9.9}$$

with effective lip-reed width w (in m), density of air ρ_0 (in kg/m^3), static equilibrium separation H_0 (in m). Moreover, $[\cdot]_+$ describes the 'positive part of' (see Chapter 10). The negative equilibrium separation $-H_0$ can be seen as the location of the lower lip, and when $y + H_0 \leq 0$, the lips are closed and U_{B} is 0. Another volume flow (in m^3/s) is generated by the lip reed itself according to

$$U_{\mathrm{r}} = U_{\mathrm{r}}(t) = S_{\mathrm{r}}\dot{y}, \tag{9.10}$$

and assuming that the volume flow velocity is conserved, the total air volume entering the acoustic tube at the left boundary is defined as

$$S(0)v(0,t) = U_{\mathrm{B}}(t) + U_{\mathrm{r}}(t). \tag{9.11}$$

Compact PDE

To reduce the number of variables in later derivations in this chapter, one can divide all terms in Eq. (9.7) by M to obtain

$$\ddot{y} = -\omega_0^2 y - \sigma_{\mathrm{r}}\dot{y} + \frac{S_{\mathrm{r}}}{M}\Delta p, \tag{9.12}$$

with angular frequency of the lip reed $\omega_0 = \sqrt{K/M}$ (in rad/s) and loss parameter $\sigma_\mathrm{r} = R/M$ (in s^{-1}).

9.3 Discrete time

This section follows the discretisation and derivation given in [62, Sec. 5.1.3, pp. 140–141], with a slight change in notation.

The variables y, Δp, and thereby U_B and U_r are placed on the interleaved temporal grid[1], and the equations presented above can be discretised to the following system:

$$\begin{aligned}
\delta_{tt} y^{n+1/2} &= -\omega_0^2 \mu_{t\cdot} y^{n+1/2} - \sigma_\mathrm{r} \delta_{t\cdot} y^{n+1/2} + \frac{S_\mathrm{r}}{M} \Delta p^{n+1/2}, && \text{(9.13a)}\\
\Delta p^{n+1/2} &= P_\mathrm{m} - \mu_{t+} p_0^n, && \text{(9.13b)}\\
U_\mathrm{B}^{n+1/2} &= w[y^{n+1/2} + H_0]_+ \mathrm{sgn}(\Delta p^{n+1/2}) \sqrt{\frac{2|\Delta p^{n+1/2}|}{\rho_0}}, && \text{(9.13c)}\\
U_\mathrm{r}^{n+1/2} &= S_\mathrm{r} \delta_{t\cdot} y^{n+1/2}, && \text{(9.13d)}\\
\mu_{x-}(S_{1/2} v_{1/2}^{n+1/2}) &= U_\mathrm{B}^{n+1/2} + U_\mathrm{r}^{n+1/2}. && \text{(9.13e)}
\end{aligned}$$

Here, p_0^n and $S_{1/2} v_{1/2}^{n+1/2}$ are discrete values at the left boundary of an acoustic tube described by system (5.40). Expanding the operators in Eq. (9.13a) and solving for $y^{n+3/2}$, yields

$$\alpha_\mathrm{r} y^{n+3/2} = 4 y^{n+1/2} + \beta_\mathrm{r} y^{n-1/2} + \xi_\mathrm{r} \Delta p^{n+1/2}, \tag{9.14}$$

where[2]

$$\alpha_\mathrm{r} = 2 + \omega_0^2 k^2 + \sigma_\mathrm{r} k\ , \quad \beta_\mathrm{r} = \sigma_\mathrm{r} k - 2 - \omega_0^2 k^2\ , \quad \text{and} \quad \xi_\mathrm{r} = \frac{2 S_\mathrm{r} k^2}{M}. \tag{9.15}$$

Although Eq. (9.14) seems to be implicitly dependent on the pressure difference $\Delta p^{n+1/2}$, it is possible to explicitly solve it. A derivation is shown below.

9.3.1 Obtaining Δp

In the following, the superscript $n + 1/2$ will be suppressed for y, Δp, U_B, U_r, $S_{1/2}$ and $v_{1/2}$ for brevity.

[1] The variables are placed on the non-interleaved spatial grid, as the lip reed interacts with the boundary of the tube ($x = 0$).

[2] Notice that all terms are multiplied by 2 to reduce fractions.

Rewriting Eq. (9.13a)

Using identities (2.27a) and (2.27e), Eq. (9.13a) can be rewritten to

$$\frac{2}{k}(\delta_{t\cdot} - \delta_{t-})y = -\omega_0^2(k\delta_{t\cdot} + e_{t-})y - \sigma_\mathrm{r}\delta_{t\cdot}y + \frac{S_\mathrm{r}}{M}\Delta p,$$

and, after grouping the terms,

$$a_1\delta_{t\cdot}y - a_2\Delta p - a_3^n = 0, \tag{9.16}$$

where

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

Note that the non-negativity property can be applied to a_1 and a_2 as these are calculated solely from non-negative parameters. Equation (9.13d) can then be substituted into Eq. (9.16)

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

and consequently Eq. (9.13e), to get

$$\frac{a_1}{S_\mathrm{r}}\left(\mu_{x-}(S_{1/2}v_{1/2}) - U_\mathrm{B}\right) - a_2\Delta p - a_3^n = 0. \tag{9.17}$$

Obtaining $\mu_{x-}(S_{1/2}v_{1/2})$

To obtain a definition for $\mu_{x-}(S_{1/2}v_{1/2})$, one can use the FD scheme for the pressure of the first-order system in (5.40a) and evaluate this at $l = 0$ to get

$$\frac{\bar{S}_0}{\rho_0 c^2}\delta_{t+}p_0^n = -\delta_{x-}(S_{1/2}v_{1/2}). \tag{9.18}$$

Using identity (2.27d) for δ_{x-} and δ_{t+}, this can be rewritten to

$$\frac{2\bar{S}_0}{\rho_0 c^2 k}(\mu_{t+}p_0^n - p_0^n) = \frac{2}{h}\left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2}\right). \tag{9.19}$$

and substituting Eq. (9.13b) yields

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

$$\mu_{x-}(S_{1/2}v_{1/2}) = b_1^n - b_2\Delta p \tag{9.20}$$

where

$$b_1^n = S_{1/2}v_{1/2} + \frac{\bar{S}_0 h}{\rho_0 c^2 k}(P_\mathrm{m} - p_0^n), \quad \text{and} \quad b_2 = \frac{\bar{S}_0 h}{\rho_0 c^2 k} \geq 0\,. \tag{9.21}$$

Final steps

Equations (9.20) and (9.13c) can be substituted into Eq. (9.17) to get

$$\begin{aligned} \frac{a_1}{S_\mathrm{r}}\left(b_1^n - b_2\Delta p - w[y+H_0]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}}\right) - a_2\Delta p - a_3^n &= 0, \\ -w[y+H_0]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}} - b_2\Delta p - \frac{a_2 S_\mathrm{r}}{a_1}\Delta p + b_1^n - \frac{a_3^n S_\mathrm{r}}{a_1} &= 0, \\ -c_1^n\mathrm{sgn}(\Delta p)\sqrt{|\Delta p|} - c_2\Delta p + c_3^n &= 0, \end{aligned} \tag{9.22}$$

where

$$c_1^n = w[y+H_0]_+\sqrt{\frac{2}{\rho_0}} \geq 0, \quad c_2 = b_2 + \frac{a_2 S_\mathrm{r}}{a_1} \geq 0, \quad \text{and} \quad c_3^n = b_1^n - \frac{a_3^n S_\mathrm{r}}{a_1}\,. \tag{9.23}$$

Equation (9.22) can be divided by $-\mathrm{sgn}(\Delta p)$ to get a quadratic equation in $\sqrt{|\Delta p|}$:

$$c_2|\Delta p| + c_1^n\sqrt{|\Delta p|} - \frac{c_3^n}{\mathrm{sgn}(\Delta p)} = 0. \tag{9.24}$$

As $c_1^n, c_2 \geq 0$, the following must be true for any real solutions to exist

$$\mathrm{sgn}(c_3^n) = \mathrm{sgn}(\Delta p) \quad \Longrightarrow \quad \frac{c_3^n}{\mathrm{sgn}(\Delta p)} = |c_3^n|, \tag{9.25}$$

and one can solve for $\sqrt{|\Delta p|}$:

$$\sqrt{|\Delta p|} = \frac{-c_1^n \pm \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2}\,. \tag{9.26}$$

Finally, because $\sqrt{(c_1^n)^2 + 4c_2|c_3^n|} \geq c_1^n$, one can only guarantee a positive solution if the square root term is added. Using Eq. (9.25), the definition for the pressure difference can be found:

$$\Delta p = \mathrm{sgn}(c_3^n)\left(\frac{-c_1^n + \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2}\right)^2, \tag{9.27}$$

which can be used in the update of the lip reed in Eq. (9.13a).

9.3.2 Coupling to the tube

The coupling of the lip reed to the acoustic tube is easily done by rewriting Eq. (5.41a) evaluated at $l = 0$ to

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c\lambda}{\bar{S}_0}\left(-2\mu_{x-}(S_{1/2}v_{1/2}) + 2S_{1/2}v_{1/2}\right). \tag{9.28}$$

Equation (9.13e) can then be substituted to get

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c\lambda}{\bar{S}_0}\left(-2(U_\mathrm{B} + U_\mathrm{r}) + 2S_{1/2}v_{1/2}\right). \tag{9.29}$$

Figure 9.4 shows an implementation of the lip reed connected to an acoustic tube. The lip reed is shown on the left and the left boundary of the tube is on the right side of the lip reed. The frequency of the lips is set to $f_0 = 600$ Hz ($\omega_0 = 1200\pi$ rad/s), the input pressure $P_\mathrm{m} = 2000$ Pa and the other parameters are as listed in paper [H]. The lip is initialised using $y^{1/2} = y^{3/2} = -H_0$ such that the lips are closed at the start of the simulation. Furthermore, the tube is set to be cylindrical with a circular cross-section of $S(x) = 5 \cdot 10^{-5}$ m^2. The figure shows that the lips oscillate, and that when the lips are closed, i.e. when $y \leq H_0$, no energy enters the acoustic tube.[3]

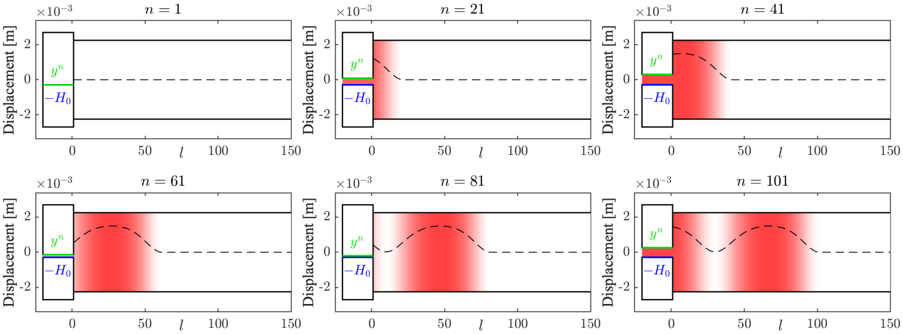


Fig. 9.4: A lip reed (shown at the left side of the plots) exciting a cylindrical acoustic tube. The y-axis refers to the displacement of the lips and the pressure in the tube p_l^n is shown in red and highlighted with a dashed line (not related to the y-axis).

9.4 Energy analysis

This section performs an energy analysis on the lip reed system, coupled to an acoustic tube using the steps described Section 3.4. The analysis follows [62, Sec 5.1.3, p. 139].

[3] This is similar behaviour to what the pulse train in Section 7.2.2 attempts to model.

As all physical parameters need to be written out to obtain the correct units, Eq. (9.7) is discretised to get

$$M\delta_{tt}y^{n+1/2} = -K\mu_{t\cdot}y^{n+1/2} - R\delta_{t\cdot}y^{n+1/2} + S_{\mathrm{r}}\Delta p^{n+1/2}, \tag{9.30}$$

and will be used in this analysis. Again, the superscript $n+1/2$ will be suppressed for y, Δp, U_{B}, U_{r}, $S_{1/2}$ and $v_{1/2}$ for brevity.

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

Multiplying Eq. (9.30) by $(\delta_{t\cdot}y)$, and moving all terms to the left-hand side, yields the rate of change of the energy in the lip reed $\mathfrak{h}_{\mathrm{r}}$:

$$\delta_{t+}\mathfrak{h}_{\mathrm{r}} = M(\delta_{t\cdot}y)(\delta_{tt}y) + K(\delta_{t\cdot}y)(\mu_{t\cdot}y) + R(\delta_{t\cdot}y)^2 - S_{\mathrm{r}}(\delta_{t\cdot}y)\Delta p = 0.$$

One can substitute Eqs (9.13d), and (9.13e) thereafter, to get

$$\begin{aligned}\delta_{t+}\mathfrak{h}_{\mathrm{r}} = M(\delta_{t\cdot}y)(\delta_{tt}y) + K(\delta_{t\cdot}y)(\mu_{t\cdot}y) + R(\delta_{t\cdot}y)^2 \\ - \left(\mu_{x-}(S_{1/2}v_{1/2}) - U_{\mathrm{B}}\right)\Delta p = 0.\end{aligned}$$

Finally, substituting Eq. (9.13b), yields

$$\begin{aligned}\delta_{t+}\mathfrak{h}_{\mathrm{r}} = M(\delta_{t\cdot}y)(\delta_{tt}y) + K(\delta_{t\cdot}y)(\mu_{t\cdot}y) + R(\delta_{t\cdot}y)^2 \\ + U_{\mathrm{B}}\Delta p - \mu_{x-}(S_{1/2}v_{1/2})(P_{\mathrm{m}} - \mu_{t+}p_0^n) = 0.\end{aligned}$$

One can then include the tube by recalling that $\delta_{t+}\mathfrak{h}_{\mathrm{t}} = -\mathfrak{b}_{\mathrm{r}} + \mathfrak{b}_{\mathrm{l}}$, and that the left boundary term is defined as (Eq. (5.52))

$$\mathfrak{b}_{\mathrm{l}} = (\mu_{t+}p_0)\mu_{x-}(S_{1/2}v_{1/2}),$$

and substituting this (ignoring the right boundary term, i.e., $\mathfrak{b}_{\mathrm{r}} = 0$) to get

$$\begin{aligned}\delta_{t+}(\mathfrak{h}_{\mathrm{r}} + \mathfrak{h}_{\mathrm{t}}) = M(\delta_{t\cdot}y)(\delta_{tt}y) + K(\delta_{t\cdot}y)(\mu_{t\cdot}y) + R(\delta_{t\cdot}y)^2 \\ + U_{\mathrm{B}}\Delta p - \mu_{x-}(S_{1/2}v_{1/2})P_{\mathrm{m}} = 0.\end{aligned}$$

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

Using identities (3.17a) and (3.17d), the energy balance can be shown to be

$$\delta_{t+}\left(\mathfrak{h}_{\mathrm{t}} + \mathfrak{h}_{\mathrm{r}}\right) = -\mathfrak{q}_{\mathrm{r}} - \mathfrak{p}_{\mathrm{r}}, \tag{9.31}$$

where the energy of the tube $\mathfrak{h}_{\mathrm{t}}$ is as defined in Eq. (5.55) and the energy of the mass is

$$\mathfrak{h}_{\mathrm{r}} = \mathfrak{t}_{\mathrm{r}} + \mathfrak{v}_{\mathrm{r}}, \quad \text{with} \quad \mathfrak{t}_{\mathrm{r}} = \frac{M}{2}(\delta_{t-}y)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}\mu_{t-}(y^2). \tag{9.32}$$

Furthermore, the damping term is defined as

$$\mathfrak{q}_\mathrm{r} = R(\delta_{t\cdot}y)^2 + U_\mathrm{B}\Delta p, \tag{9.33}$$

and the input power as

$$\mathfrak{p}_\mathrm{r} = -(U_\mathrm{B} + U_\mathrm{r})P_\mathrm{m}. \tag{9.34}$$

It is interesting to note that due to the choice of discretisation of the lip reed, $\mathfrak{t}_\mathrm{r}$, $\mathfrak{v}_\mathrm{r}$ and $\mathfrak{q}_\mathrm{r}$ are non-negative, making the lip reed strictly dissipative and thus inherently stable.

Step 3: Check units

The kinetic and potential energy of the lip reed can be written in their units as

$$\begin{aligned}
\mathfrak{t}_\mathrm{r} = \frac{M}{2}(\delta_{t-}y)^2 &\xrightarrow{\text{in units}} \quad \mathrm{kg}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})^2 = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},\\
\mathfrak{v}_\mathrm{r} = \frac{K}{2}\mu_{t-}(y^2) &\xrightarrow{\text{in units}} \quad \mathrm{N}\cdot\mathrm{m}^{-1}\cdot\mathrm{m}^2 = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2},
\end{aligned}$$

and have the correct units. Recalling that the damping and input power terms need to have units of kg$\cdot$ m$^2\cdot$s^{-3}, writing the individual components of these terms in their respective units, yields

$$\begin{aligned}
R(\delta_{t\cdot}y)^2 &\xrightarrow{\text{in units}} \quad \mathrm{kg}\cdot\mathrm{s}^{-1}\cdot(\mathrm{s}^{-1}\cdot\mathrm{m})^2 = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},\\
U_\mathrm{B}\Delta p &\xrightarrow{\text{in units}} \quad \mathrm{m}^3\cdot\mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-1}\cdot\mathrm{s}^{-2} = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},\\
-(U_\mathrm{B}+U_\mathrm{r})P_\mathrm{m} &\xrightarrow{\text{in units}} \quad \mathrm{m}^3\cdot\mathrm{s}^{-1}\cdot\mathrm{kg}\cdot\mathrm{m}^{-1}\cdot\mathrm{s}^{-2} = \mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-3},
\end{aligned}$$

and shows that the units are indeed correct.

Step 4: Implementation

Figure 9.5 shows the energetic output of the lip reed coupled to an acoustic tube, corresponding to the behaviour shown in Figure 9.4. The total energy of the system increases due to the input pressure and is mainly transferred to the tube. That the lip reed is oscillating can be observed from the oscillations in its kinetic and potential energy. The normalised energy (using Eq. (3.38)) does not include the first time index, as one full iteration of the coupling is necessary to yield a correct energy calculation. Instead, one starts at $n = 1$ and uses $\mathfrak{h}^1$ instead of $\mathfrak{h}^0$ in Eq. (3.38).

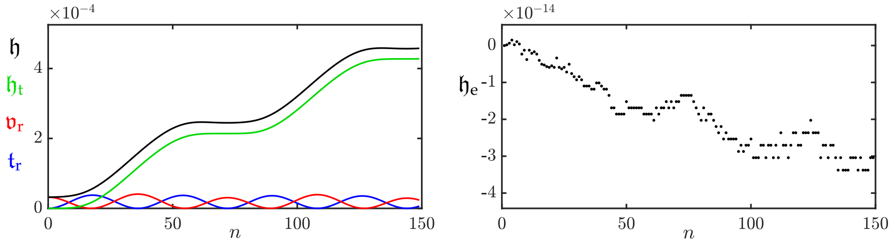


Fig. 9.5: The energy of the acoustic tube (green), the potential energy (red) the kinetic energy of the lip reed (blue), and the total energy (black) of the system corresponding to Figure 9.4. The right panel shows the normalised energy (according to Eq. (3.38), starting at $n = 1$) and shows that the deviation of the energy is within machine precision.

Part IV

Interactions

Interactions

Most musical instruments are composed of several individual resonators which interact with one another. Part II introduced various resonators in isolation, and this part describes different ways to model the interaction between these.

Chapter 10 describes collision interactions between different systems and has been used for the interactions between the different components of the tromba marina in papers [D] and [E]. Furthermore, the theory described in this chapter has been used for the collision between the lips that excite the trombone in paper [H]. Then, Chapter 11 describes various to connections between models and has been used extensively for papers [A] and [B]. Furthermore, the dynamic grid presented in paper [G], uses the principles described in this chapter.

Chapter 10

Collisions

Many musical instruments rely on collisions in some shape or form. Examples are the collision between a hammer and a piano string, a guitar pick and the string, and even the collision between the lips of a trumpet player.

This work uses collision models that rely on penalising methods. The colliding objects – although possibly perfectly rigid – are supposed to interpenetrate, and collision is interpreted as a *penalty*. The eventual force acting on the colliding objects is then dependent on the level of penetration. For deformable objects, such as the hammer felt tip of a piano, the penalty is dependent on the level of deformation. These collision models were first used in a musical context by e.g. [101, 102].

The discretisations proposed in [101, 102] rely on implicit nonlinear schemes which require an iterative method, such as the Newton-Raphson method presented in Section 8.3, to obtain their solution. The exact number of iterations required per time step, especially in interactive applications, is usually unknown. This could be detrimental to real-time applications, as the number of iterations, and consequently the extra number of computations, could be very large in a particular situation. Furthermore, and perhaps more importantly, existence and uniqueness of the solution might not be available.

In [103] (co-authored by the PhD student [O3]), Ducceschi et al. propose a method based on quadratisation of the collision potential energy, that circumvents the need of an iterative method to solve nonlinear collisions. Energy quadratisation for explicit schemes first appeared in the context of Port-Hamiltonian systems and was proposed by Lopes et al. and Falaize et al. in [104, 105]. The introduction of an additional state variable, which is what Ducceschi's work is based on, was introduced in [106, 107]. Papers [D] and [E] follow an earlier iteration of the non-iterative collision algorithm from [108, 65] which exhibited spurious oscillations that Ducceschi et al. resolve in [103]. Paper [H] uses the corrected collision model for the collision between the lips exciting the trombone. The corrected model will be used in this work and presented in this chapter.

This chapter first provides a definition for the collision potential as well as its quadratisation, used as the basis for the explicit method. Then, the method will be applied to a simple mass-barrier collision and finally, to a mass-spring – string collision which can be used to model a finger-fretted string.

Collision potential

Collisions can be modelled using a nonlinear *collision potential*, which can be defined as [109]

$$\phi(\eta) = \frac{K_\text{c}}{\alpha_\text{c}+1}[\eta]_+^{\alpha_\text{c}+1}, \tag{10.1}$$

with collision stiffness $K_\text{c} \geq 0$ (in N/m$^{\alpha_\text{c}}$) and dimensionless nonlinear collision coefficient $\alpha_\text{c} \geq 1$. Here $\eta = \eta(t)$ describes the relative displacement between the two colliding bodies (in m). The $[\cdot]_+$ operator, defined as

$$[\cdot]_+ = \frac{\cdot + |\cdot|}{2}, \tag{10.2}$$

describes the 'positive part of' and when applied to η in Eq. (10.1) causes the potential ϕ to only be non-zero when the two colliding bodies are in contact.

The derivative of Eq. (10.1) with respect to η is defined as

$$\phi'(\eta) = K_\text{c}[\eta]_+^{\alpha_\text{c}} \tag{10.3}$$

and can then be used in the PDE at hand.

The issue with this form of the collision potential, is that an iterative method, such as Newton-Raphson presented in Section 8.3, needs to be used in order to solve the system [103].

Quadratic form

Based on earlier work by Lopez and Falaize et al. [104, 105], the authors propose in [103] to rewrite the potential in Eq. (10.3) in a quadratic form. Using the chain rule and $\psi = \psi(\eta)$, Eq. (10.3) can be rewritten as

$$\phi'(\eta) = \psi\psi' \quad \text{where} \quad \psi = \sqrt{2\phi} \quad \text{and} \quad \psi' = \frac{\dot{\psi}}{\dot{\eta}}, \tag{10.4}$$

where a dot denotes a single derivative with respect to time.

This form of the potential can be discretised to a FD scheme that can be solved explicitly. This process will be shown below, using an example of the simple mass – rigid barrier collision.

10.1 The mass – rigid barrier collision

As a test case, a mass colliding with a rigid barrier is presented here, which is arguably the simplest case of a collision. Consider a mass at location $u = u(t)$ (in m) colliding with a barrier at location b (in m).

If the barrier is placed above the mass, the force it exerts on the mass will be negative and its system would be described as

$$M\ddot{u} = -\psi\psi', \tag{10.5}$$

with mass M (in kg) and $\psi = \psi(\eta)$ and ψ' are as defined in Eq. (10.4) with $\eta = \eta(t) = u(t) - b$.

Looking towards the discretisation the mass-barrier collision, one could use the definitions in Eq. (10.4) to rewrite Eq. (10.5) to the following system of equations

$$M\ddot{u} = -\psi g, \tag{10.6a}$$

$$\dot{\psi} = g\dot{\eta}, \tag{10.6b}$$

$$\eta(t) = u(t) - b, \tag{10.6c}$$

where $g = \psi'$.

Relative location of objects

When working with multiple interacting objects, it is important to consider whether an object is located 'above' or 'below' the other, i.e., which (generally) has a more positive or negative displacement than the other. A mass with a displacement of 0.01 m will thus be 'above' a barrier with a displacement of -0.05 m. Along these lines, a positive force acting on an element will accelerate it upwards and a negative force will accelerate it downwards.

The relative location of the two colliding objects will affect two things in system (10.6):

Firstly, the location of the object determines the direction of the collision force, i.e., the sign of the right-hand side in system (10.6). In this case, the barrier is placed above the mass, and will exert a downwards (negative) force on the mass during collision. If the barrier was placed below the mass, the opposite would have applied.

Secondly, the definition of η in (10.6c) is affected by the relative location of the objects. The collision potential in Eq. (10.1) is only non-zero when η is positive. If the barrier is placed above the mass, $u(t) - b$ will be positive on collision. It is thus important to remember that η should be defined as the element above subtracted from the element below.

10.1.1 Discrete time

Before discretising system (10.6) in full, the discrete approximation to the collision potential will be elaborated on. Following [103], ψ is placed on an interleaved temporal grid (see Section 5.2.2) using

$$\psi^{n-1/2} = \mu_{t-}\psi^n, \tag{10.7}$$

where the interleaved temporal grid is used here as it results in energy conservation in discrete time (see Section 10.1.3). Approximations to ψ and g in Eq. (10.6) can then be made as

$$\psi \approxeq \mu_{t+}\psi^{n-1/2} \tag{10.8}$$

and

$$g \approxeq g^n = \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}, \tag{10.9}$$

respectively. Notice that applying a first-order difference operator to a grid function on an interleaved grid is second-order accurate.[1] The result of the approximation in Eq. (10.9) allows ψ to be treated as an independent time series:

$$\delta_{t+}\psi^{n-1/2} = g^n\delta_{t\cdot}\eta^n. \tag{10.10}$$

With the above approximations in place, system (10.6) can be discretised and yields the following system of equations:

$$M\delta_{tt}u^n = -\left(\mu_{t+}\psi^{n-1/2}\right)g^n, \tag{10.11a}$$

$$\delta_{t+}\psi^{n-1/2} = g^n\delta_{t\cdot}\eta^n, \tag{10.11b}$$

$$\eta^n = u^n - b. \tag{10.11c}$$

An explicit definition for g^n

To be able to calculate $\psi^{n+1/2}$ and u^{n+1} in system (10.11) explicitly, a definition for g^n only based on known values must be found. As $g^n \approxeq \psi'$ as per Eq. (10.9), the derivative can be computed analytically according to

$$g^n = \psi'\bigg|_{\eta=\eta^n} \overset{\text{Eq. (10.4)}}{=} \frac{\phi'}{\sqrt{2\phi}}\bigg|_{\eta=\eta^n}. \tag{10.12}$$

[1] $\delta_{t+}\psi^{n-1/2} \overset{\text{Eq. (10.9)}}{=} \delta_{t+}\mu_{t-}\psi^n \overset{\text{Eq. (2.27b)}}{=} \delta_{t\cdot}\psi^n$ which is second-order accurate (see Section 2.2.2).

Recalling (10.3) and (10.1), this can conveniently be rewritten to

$$g^n = \frac{K_\text{c}[\eta^n]_+^{\alpha_\text{c}}}{\sqrt{\frac{2K_\text{c}}{\alpha_\text{c}+1}[\eta^n]_+^{\alpha_\text{c}+1}}} = K_\text{c}\sqrt{\frac{\alpha_\text{c}+1}{2K_\text{c}}}[\eta^n]_+^{\alpha_\text{c}}[\eta^n]_+^{\frac{-(\alpha_\text{c}+1)}{2}} = \sqrt{\frac{K_\text{c}(\alpha_\text{c}+1)}{2}}[\eta^n]_+^{\frac{\alpha_\text{c}-1}{2}} \, . \tag{10.13}$$

This implementation is the one presented in [108], but exhibited spurious oscillations and 'sticking' behaviour. This is due to the possibility of negative forces for positive penetrations due to the discontinuity in the definition for g^n at $\eta^n = 0$.

In [103], the definition for g^n is extended, starting out by using an implicit equation for g^n by directly discretising Eq. (10.9)

$$g^n_\text{imp} = 2\frac{\psi^{n+1/2} - \psi^{n-1/2}}{\eta^{n+1} - \eta^{n-1}} \, . \tag{10.14}$$

If there is, however, no collision at $n + 1/2$, $\psi^{n+1/2} = 0$ and Eq. (10.14) reduces to

$$g^n_\text{imp} = -2\frac{\psi^{n-1/2}}{\eta^{n+1} - \eta^{n-1}} \, .$$

Furthermore, due to the fact that there is no collision, η^{n+1} can be calculated according to $\eta^{n+1} = \eta^\star = u^\star - b$, where $u^\star$ is the value of u^{n+1} calculated using the scheme in Eq. (10.11a) without the collision force. Expanding Eq. (10.11a) without the collision force yields

$$\frac{M}{k^2}\left(u^\star - 2u^n + u^{n-1}\right) = 0 \quad \Longrightarrow \quad u^\star = 2u^n - u^{n-1}.$$

Thus, if there is no collision, g^n_imp can now be explicitly calculated from known values and be used in the definition for g^n according to [103]

$$g^n = \begin{cases} \kappa\sqrt{\dfrac{K_\text{c}(\alpha_\text{c}+1)}{2}} \cdot (\eta^n)^{\frac{\alpha_\text{c}-1}{2}}, & \text{if } \eta^n \geq 0, \qquad \text{(10.15a)} \\ -2\dfrac{\psi^{n-1/2}}{\eta^\star - \eta^{n-1}}, & \text{if } \eta^n < 0 \text{ and } \eta^\star \neq \eta^{n-1}, \qquad \text{(10.15b)} \\ 0, & \text{if } \eta^n < 0 \text{ and } \eta^\star = \eta^{n-1}. \qquad \text{(10.15c)} \end{cases}$$

Here, $\kappa = 1$ if $\psi^{n-1/2} \geq 0$, otherwise $\kappa = -1$ and aims to resolve the 'sticking' behaviour by forcing an outwardly-directed force at all times. As was done in paper [H], condition (10.15c) has been added to the definition of g^n from [103] to prevent a division by 0 in Eq. (10.15b).

This definition for g^n does not exhibit the spurious oscillations that the old definition did, and can still be explicitly calculated from known values of the system.

10.1.2 Solving the system

To implement the system in Eq. (10.11), its definitions need to be slightly rewritten. Using identity (2.27c), $\mu_{t+}\psi^{n-1/2}$ can be rewritten to

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}\delta_{t+}\psi^{n-1/2} + \psi^{n-1/2}.$$

Then, substituting (10.11b) into this, yields

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}g^n\delta_{t\cdot}\eta^n + \psi^{n-1/2},$$

and inserting this into (10.11a), yields

$$M\delta_{tt}u^n = -\left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n + \psi^{n-1/2}\right)g^n\ . \tag{10.16}$$

As the position of barrier b is static, the following is true:

$$\frac{d\eta}{dt} = \frac{d}{dt}\Big(u - b\Big) \quad\Longrightarrow\quad \delta_{t\cdot}\eta^n = \delta_{t\cdot}u^n, \tag{10.17}$$

i.e., the time derivative of η equals the time derivative of u.[2] Eq. (10.16) can now be solved explicitly as u^{n+1} is the only unknown in the system

$$\left(\frac{M}{k^2} + \frac{(g^n)^2}{4}\right)u^{n+1} = \frac{M}{k^2}(2u^n - u^{n-1}) + \frac{(g^n)^2}{4}u^{n-1} - \psi^{n-1/2}g^n\ , \tag{10.18}$$

and can be solved by a simple division.

Finally, u^{n+1} can be used to calculate η^{n+1} by evaluating (10.11c) at $n+1$:

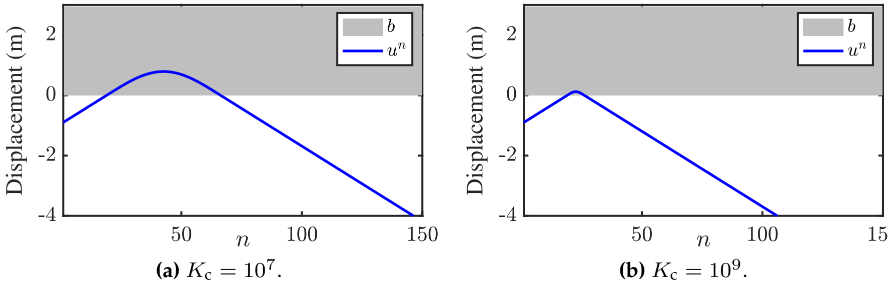
$$\eta^{n+1} = u^{n+1} - b, \tag{10.19}$$

which is used to calculate $\psi^{n+1/2}$ by expanding and rewriting (10.11b) to

$$\psi^{n+1/2} = \psi^{n-1/2} + \frac{\eta^{n+1} - \eta^{n-1}}{2}\ . \tag{10.20}$$

Figure 10.1 shows the mass – rigid barrier collision over time for two values of K_c. The mass is initialised with an initial (upwards) velocity using $u^0 = -1$ m and $u^1 = -0.95$ m. The figure shows that the penetration of the mass with the barrier causes a downwards force on the mass. As expected, this force is higher for a larger value of K_c and causes the mass to accelerate downwards more quickly.

[2] Note that if the barrier was placed underneath the mass, making (10.11c) $\eta^n = b - u^n$, this would result in $\delta_{t\cdot}\eta^n = -\delta_{t\cdot}u^n$.



(a) $K_\mathrm{c} = 10^7$.

(b) $K_\mathrm{c} = 10^9$.

Fig. 10.1: The mass – rigid barrier collision over time with $\alpha_\mathrm{c} = 1.3$ for different values of K_c.

10.1.3 Energy analysis

To prove that the collision term does not add any additional energy into the system (retaining passivity) and that it does not add additional constraints on the stability of the system, the energy analysis techniques presented in Section 3.4 can be used. Notice that for brevity, the steps presented in Section 3.4 will not explicitly be followed.

Multiplying Eq. (10.11a) by $(\delta_{t\cdot}u^n)$ yields

$$\delta_{t+}\mathfrak{h}_\mathrm{m} = -\left(\mu_{t+}\psi^{n-1/2}\right) g^n(\delta_{t\cdot}u^n)$$

where the energy of the mass is defined as (see Eq. (3.42))

$$\mathfrak{h}_\mathrm{m} = \frac{M}{2}(\delta_{t-}u^n)^2. \tag{10.21}$$

Expanding g^n yields

$$\begin{aligned}\delta_{t+}\mathfrak{h}_\mathrm{m} &= -\left(\mu_{t+}\psi^{n-1/2}\right)\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}(\delta_{t\cdot}u^n)\\ \overset{\text{Eq. (10.17)}}{\Longleftrightarrow} \quad &= -\left(\mu_{t+}\psi^{n-1/2}\right)\delta_{t+}\psi^{n-1/2},\end{aligned}$$

which, using identity (3.17c), can be rewritten to

$$\delta_{t+}(\mathfrak{h}_\mathrm{m} + \mathfrak{h}_\mathrm{c}) = 0, \tag{10.22}$$

with collision energy

$$\mathfrak{h}_\mathrm{c} = \frac{(\psi^{n-1/2})^2}{2}. \tag{10.23}$$

Recall that in order for a scheme to be passive, its energy must be non-negative, and the fact that ψ is squared proves passivity for system (10.11).

Figure 10.2 shows the energetic output of the mass – rigid barrier collision corresponding to Figure 10.1a. The left panel shows that the kinetic energy of the mass is transferred into the energy of the collision, after which it is converted into kinetic energy of the mass again.

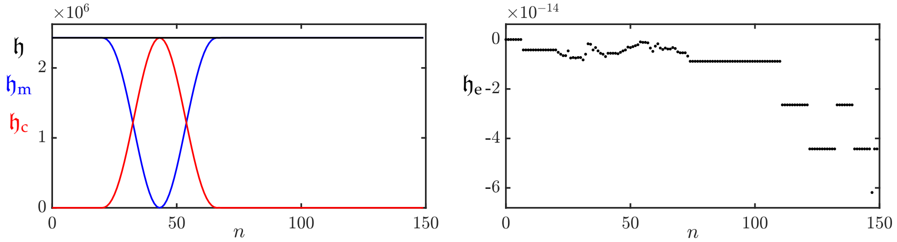


Fig. 10.2: The energy of the mass (blue), the collision (green) and the total energy (black) of the mass – rigid barrier collision. The energy corresponds to the behaviour in Figure 10.1a. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

10.2 Mass-spring – string collision

The mass-spring – string collision is slightly trickier than the mass – rigid barrier collision, as there are two moving components rather than one. This system is chosen as an example as it has the interesting use-case of fretting a string to change the pitch, modelling the fretting finger as a mass.

Consider a lossless stiff string of length L, its transverse displacement described by $u = u(x,t)$ (in m) and defined for $t \geq 0$ and $x \in \mathcal{D}$ with domain $\mathcal{D} = [0, L]$. The mass with displacement $w = w(t)$ (in m) and $t \geq 0$ will model the fretting finger. The PDE for the stiff string and its parameter definitions can be found in Eq. (4.1) and for the mass-spring system in Eq. (2.28). Placing the string above the mass, the following system emerges:

$$\rho A \partial_t^2 u = T\partial_x^2 u - EI\partial_x^4 u + \delta(x - x_\mathrm{m})\psi g \tag{10.24a}$$

$$M\ddot{w} = -Kw - \psi g \tag{10.24b}$$

$$\dot{\psi} = g\dot{\eta}, \tag{10.24c}$$

$$\eta(t) = w(t) - u(x_\mathrm{m}, t), \tag{10.24d}$$

where spatial Dirac delta function $\delta(x - x_\mathrm{m})$ localises the mass (finger) along the string at location $x_\mathrm{m} \in \mathcal{D}$ (see Eq. (8.8)). Furthermore, $\psi = \psi(\eta)$ and $g = \psi'$ are as defined in Eq. (10.4).

Discretising system (10.24), with the collision discretised according to the process explained in Section 10.1, yields

$$\rho A\delta_{tt}u_l^n = T\delta_{xx}u_l^n - EI\delta_{xxxx}u_l^n + J_l(x_\text{m})\big(\mu_{t+}\psi^{n-1/2}\big)g^n, \tag{10.25a}$$

$$M\delta_{tt}w^n = -Kw^n - \big(\mu_{t+}\psi^{n-1/2}\big)g^n, \tag{10.25b}$$

$$\delta_{t+}\psi^{n-1/2} = g^n\delta_{t\cdot}\eta^n, \tag{10.25c}$$

$$\eta^n = w^n - I_l(x_\text{m})u_l^n, \tag{10.25d}$$

where $l \in d$ with discrete domain $d = \{0, \ldots, N\}$ and number of grid points along the string $N+1$. Furthermore, spreading and interpolation operators $I_l(x_\text{m}) = I_{l,o}(x_\text{m})$ and $J_l(x_\text{m}) = J_{l,o}(x_\text{m})$ are as defined in Section 8.2. The order o is left unspecified. Following the same process as in Section 10.1.2, Eqs. (10.25a) and (10.25b) can be rewritten to

$$\rho A\delta_{tt}u_l^n = T\delta_{xx}u_l^n - EI\delta_{xxxx}u_l^n + J_l(x_\text{m})\left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n + \psi^{n-1/2}\right)g^n, \tag{10.26a}$$

$$M\delta_{tt}w^n = -Kw^n - \left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n + \psi^{n-1/2}\right)g^n, \tag{10.26b}$$

which can be used as a starting point for solving the system.

10.2.1 Solving the system

As the colliding objects are both moving, Eq. (10.17) is not valid anymore and another strategy needs to be used. To start, one must isolate the string at the collision location x_m by taking an inner product of Eq. (10.26a) with $J_l(x_\text{m})$ over discrete domain d. Using identity (8.9) and dividing all terms by ρA yields

$$\begin{aligned}\delta_{tt}I_l(x_\text{m})u_l^n = c^2 I_l(x_\text{m})\delta_{xx}u_l^n &- \kappa^2 I_l(x_\text{m})\delta_{xxxx}u_l^n\\ &+ \frac{\|J_l(x_\text{m})\|_d^2}{\rho A}\left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n + \psi^{n-1/2}\right)g^n.\end{aligned}$$

with $c = \sqrt{T/\rho A}$ and $\kappa = \sqrt{EI/\rho A}$. Expanding the temporal FD operators yields

$$I_l(x_\text{m})u_l^{n+1} = u^\star + \underbrace{\frac{\|J_l(x_\text{m})\|_d^2 k^2}{\rho A}}_{\mathfrak{J}_l}\left(\frac{(g^n)^2}{4}\left(\eta^{n+1} - \eta^{n-1}\right) + \psi^{n-1/2}g^n\right), \tag{10.27}$$

where

$$u^\star = I_l(x_\text{m})(2u_l^n - u_l^{n-1}) + c^2k^2 I_l(x_\text{m})\delta_{xx}u_l^n - \kappa^2k^2 I_l(x_\text{m})\delta_{xxxx}u_l^n$$

is the result of the update equation of the string at x_m without the collision term. Then, Eq. (10.25d) evaluated at $n+1$, which is $\eta^{n+1} = w^{n+1} - I(x_\mathrm{m})u_l^{n+1}$, can be substituted into Eq. (10.27), which results in

$$\begin{aligned}\left(1 + \mathfrak{J}_l \frac{(g^n)^2}{4}\right) I_l(x_\mathrm{m})u_l^{n+1} - \mathfrak{J}_l \frac{(g^n)^2}{4} w^{n+1} &= u^\star \\ &+ \mathfrak{J}_l \left(-\frac{(g^n)^2}{4}\eta^{n-1} + \psi^{n-1/2}g^n\right).\end{aligned} \tag{10.28}$$

Performing this same process on the FD scheme of the mass in Eq. (10.26b) yields

$$\begin{aligned}-\frac{(g^n)^2k^2}{4M} I_l(x_\mathrm{m})u_l^{n+1} + \left(1 + \frac{(g^n)^2k^2}{4M}\right) w^{n+1} &= w^\star \\ &- \frac{k^2}{M}\left(-\frac{(g^n)^2}{4}\eta^{n-1} + \psi^{n-1/2}g^n\right),\end{aligned} \tag{10.29}$$

where

$$w^\star = 2w^n - w^{n-1} - \frac{Kk^2}{M}w^n$$

is (again) the result of the update equation of the mass without the collision term. Equations (10.28) and (10.29) can be treated as a system of linear equations (see Section B.3) with unknowns $I_l(x_\mathrm{m})u^{n+1}$ and w^{n+1}. Writing the aforementioned equations in matrix form yields

$$\begin{bmatrix} I_l(x_\mathrm{m})u_l^{n+1} \\ w^{n+1} \end{bmatrix} = \mathbf{A}^{-1}\mathbf{v} \tag{10.30}$$

where

$$\mathbf{A} = \begin{bmatrix} \left(1 + \mathfrak{J}_l \frac{(g^n)^2}{4}\right) & -\mathfrak{J}_l \frac{(g^n)^2}{4} \\ -\frac{(g^n)^2k^2}{4M} & \left(1 + \frac{(g^n)^2k^2}{4M}\right) \end{bmatrix} \quad \text{and}$$

$$\mathbf{v} = \begin{bmatrix} u^\star + \mathfrak{J}_l\left(-\frac{(g^n)^2}{4}\eta^{n-1} + \psi^{n-1/2}g^n\right) \\ w^\star - \frac{k^2}{M}\left(-\frac{(g^n)^2}{4}\eta^{n-1} + \psi^{n-1/2}g^n\right) \end{bmatrix}.$$

From this, η^{n+1} can be calculated, and can consequently be applied to the string and mass in system (10.26).

Figure 10.3 shows an implementation of the mass spring collision. The mass is initialised with an upwards initial velocity, where $w^0 = -0.2, w^1 = -0.1$, and collides with the string almost instantly after the start of the simulation. As the first panel shows, the collision model allows for interpenetration of the objects. Immediately after, the collision force accelerates the string upwards and the mass downwards.

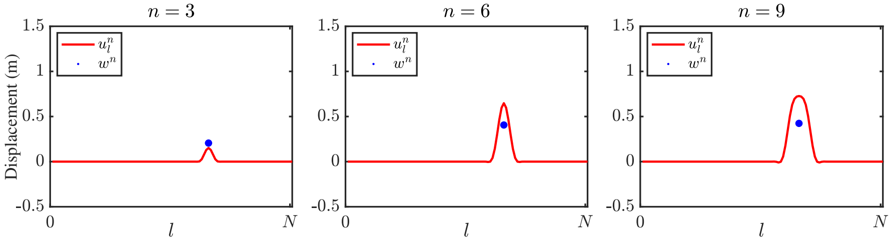


Fig. 10.3: The collision of the mass (blue) and the string (red). The collision model allows for interpenetration of the objects as shown in the left panel.

10.2.2 Energy analysis

This section follows Section 3.4 without explicitly following the steps for brevity.

One can obtain the energy of the stiff string FD scheme in Eq. (10.25a) by taking the inner product of scheme by $(\delta_{t\cdot}u_l^n)$ over discrete domain d to obtain

$$\delta_{t+}\mathfrak{h}_\mathrm{s} = \left\langle (\delta_{t\cdot}u_l^n), J_l(x_\mathrm{m})\left(\mu_{t+}\psi^{n-1/2}\right)g^n \right\rangle_d \tag{10.31}$$

where the energy of the string is (see Eq. (4.29))

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

Energy analysis for the mass in Eq. (10.25b) can be done by multiplying the scheme by $(\delta_{t\cdot}w^n)$ to get

$$\delta_{t+}\mathfrak{h}_\mathrm{m} = -(\delta_{t\cdot}w^n)\left(\mu_{t+}\psi^{n-1/2}\right)g^n, \tag{10.32}$$

where (see Eq. (3.42))

$$\mathfrak{h}_\mathrm{m} = \mathfrak{t}_\mathrm{m} + \mathfrak{v}_\mathrm{m}, \quad \text{with} \quad \mathfrak{t}_\mathrm{m} = \frac{M}{2}(\delta_{t-}w^n)^2, \quad \text{and} \quad \mathfrak{v}_\mathrm{m} = \frac{K}{2}w^n e_{t-}w^n.$$

The total energy in the system is the addition of Eqs. (10.31) and (10.32), which, using identity (8.9) for the former, can be written as:

$$\begin{aligned}\delta_{t+}(\mathfrak{h}_\mathrm{s} + \mathfrak{h}_\mathrm{m}) &= \left(I_l(x_\mathrm{m})(\delta_{t\cdot}u_l^n) - (\delta_{t\cdot}w^n)\right)\left(\mu_{t+}\psi^{n-1/2}\right)g^n,\\ &= \underbrace{\delta_{t\cdot}\left(I_l(x_\mathrm{m})u_l^n - w^n\right)}_{-\delta_{t\cdot}\eta^n}\left(\mu_{t+}\psi^{n-1/2}\right)g^n.\end{aligned}$$

Then, expanding g^n according to Eq. (10.9) yields

$$\begin{aligned}\delta_{t+}(\mathfrak{h}_{\mathrm{s}}+\mathfrak{h}_{\mathrm{m}}) &= -\delta_{t\cdot}\eta^n\left(\mu_{t+}\psi^{n-1/2}\right)\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}\\ &= -\left(\mu_{t+}\psi^{n-1/2}\right)\delta_{t+}\psi^{n-1/2}\end{aligned}$$

which, using identity (3.17c), can be rewritten as

$$\delta_{t+}(\mathfrak{h}_{\mathrm{s}}+\mathfrak{h}_{\mathrm{m}}+\mathfrak{h}_{\mathrm{c}}) = 0, \tag{10.33}$$

with collision energy

$$\mathfrak{h}_{\mathrm{c}} = \frac{(\psi^{n-1/2})^2}{2}.$$

Again, the fact that ψ is squared here, means that $\mathfrak{h}_{\mathrm{c}}$ is non-negative, and proves passivity of the system.

Figure 10.4 shows the energy of the mass-spring collision corresponding to the behaviour shown in Figure 10.3. One can observe that energy of the mass is transferred to the string almost immediately after the start of the simulation. Furthermore, the figure shows that the mass and the string collide again at $n \approx 110$. The interpenetration of the two colliding objects can be observed from the small peaks in the value for $\mathfrak{h}_{\mathrm{c}}$ at these times.

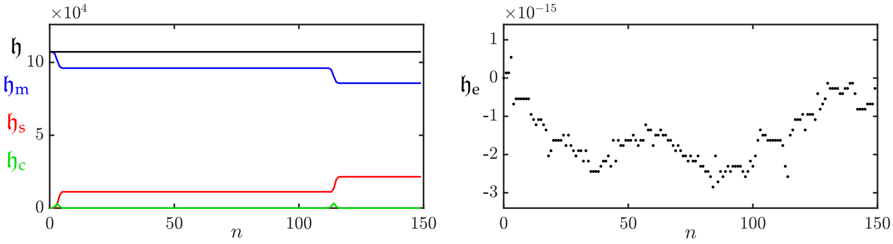


Fig. 10.4: The energy of the mass (blue), the string (red), the collision (green) and the total energy (black) of the mass-string collision. The energy corresponds to the system in Figure 10.3. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

10.3 Two-sided collision: A connection

Using the methods presented in this chapter, one could devise a two-sided collision and alter the collision potential in Eq. (10.1) to [65][3]

$$\phi(\eta) = \frac{K}{\alpha_\text{c}+1}|\eta|^{\alpha_\text{c}+1}, \tag{10.34}$$

and taking its derivative with respect to η yields

$$\phi'(\eta) = \text{sgn}(\eta)K|\eta|^{\alpha_\text{c}}. \tag{10.35}$$

One can observe that, as opposed to the (one-sided) potential presented in Eq. (10.1), the collision force will be non-zero, for both a positive and negative η. This two-sided collision can be used as a connection – as an alternative to connections presented in Chapter 11 – and has been used in papers [D] and [E] in combination with Eq. (10.1) to model the mechanics of the tromba marina. See Chapter 15 for more details.

[3] The equation proposed in [65] contains a 'dead zone' that has been set to 0 here.

Chapter 11

Connections

Many musical instruments consist of multiple subsystems like the ones presented in Part II. For example, one could simulate a guitar by modelling six separate instances of the stiff string presented in Chapter 4, and the sound board (as a simplified instrument body), using a thin plate presented in Section 6.3. The interaction between the strings and the body can then be modelled using *connections*.

Examples of connected resonators based on FDTD methods are shown in e.g. [21] and [54]. The latter presents a modular approach to connect any number of resonators in arbitrary ways using an extremely compact matrix form of the entire system.

The first example presented in this chapter is the case of two ideal strings, connected using a rigid and spring-like connection. Afterwards, the connection between a stiff string and a thin plate using a nonlinear spring will be presented, and has been used extensively in papers [A] and [B]. Before moving on to these examples, interpolation and spreading operators in 2D will be introduced.

11.1 Interpolation and spreading in 2D

This section summarises and extends [21, Sec. 10.2.1, pp. 293–294].

One can extend the interpolation and spreading operators presented in Section 8.2 to 2D by adding an additional argument to the operators. Using $l_\text{i} = \lfloor x_\text{i}/h \rfloor$ and $m_\text{i} = \lfloor y_\text{i}/h \rfloor$, a 0th-order interpolation operator $I_0(x_\text{i}, y_\text{i}) = I_{(l,m),0}(x_\text{i}, y_\text{i})$ is defined as

$$I_0(x_\text{i}, y_\text{i}) = \begin{cases} 1, & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}, \\ 0, & \text{otherwise.} \end{cases} \tag{11.1}$$

Notice that the same value for the grid spacing h is used for both the x and y direction.

Using the fractional part of the flooring operations $\alpha_x = x_\text{i}/h - l_\text{i}$ and $\alpha_y = y_\text{i}/h - m_\text{i}$, a 2D linear interpolator $I_1(x_\text{i}, y_\text{i}) = I_{(l,m),1}(x_\text{i}, y_\text{i})$ can then be composed as

$$I_1(x_\text{i}, y_\text{i}) = \begin{cases} (1-\alpha_x)(1-\alpha_y) & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}, \\ (1-\alpha_x)\alpha_y & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}+1, \\ \alpha_x(1-\alpha_y) & \text{if } l = l_\text{i}+1 \text{ and } m = m_\text{i}, \\ \alpha_x\alpha_y & \text{if } l = l_\text{i}+1 \text{ and } m = m_\text{i}+1, \\ 0, & \text{otherwise.} \end{cases} \tag{11.2}$$

Spreading operators are defined in the same way as in Section 8.2. A 0^th-order spreading operator $J_0(x_\text{i}, y_\text{i}) = J_{(l,m),0}(x_\text{i}, y_\text{i})$ can be defined as

$$J_0(x_\text{i}, y_\text{i}) = \frac{1}{h^2}\begin{cases} 1, & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}, \\ 0, & \text{otherwise,} \end{cases} \tag{11.3}$$

as well as a linear spreading operator $J_1(x_\text{i}, y_\text{i}) = J_{(l,m),1}(x_\text{i}, y_\text{i})$ as

$$J_1(x_\text{i}, y_\text{i}) = \frac{1}{h^2}\begin{cases} (1-\alpha_x)(1-\alpha_y) & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}, \\ (1-\alpha_x)\alpha_y & \text{if } l = l_\text{i} \text{ and } m = m_\text{i}+1, \\ \alpha_x(1-\alpha_y) & \text{if } l = l_\text{i}+1 \text{ and } m = m_\text{i}, \\ \alpha_x\alpha_y & \text{if } l = l_\text{i}+1 \text{ and } m = m_\text{i}+1, \\ 0, & \text{otherwise.} \end{cases} \tag{11.4}$$

Notice that the scaling is by $1/h^2$ (due to the 2D system) rather than $1/h$ in the 1D case. Some intuition on this will be given below.

As in the 1D case, the spreading operator approximates a spatial Dirac delta function, which – in 2D – is defined as

$$\delta(x,y) = \begin{cases} \infty, & x = y = 0, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\delta(x,y)dxdy = 1. \tag{11.5}$$

where $\delta(x,y)$ has units of m^{-2}. Again, as described in Section 8.2, this definition will be approximated by spreading operators, rather than be used directly.

11.1.1 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) through the discretisation process. An alternative way to see grid points after discretisation is shown in Figure 11.1. Rather than grid 'points' with a spacing h between them, a continuous system is divided into grid 'sections' of length h. This interpretation allows for the 'weight' of a grid point to be calculated from its material properties and geometry. Notice that boundaries have a length 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 system can be calculated as ρHh^2. As these grid points interact with each other, the forces resulting from this interaction will be scaled by their respective weight per grid point as will be shown in Section 11.5. This interpretation hopefully provides a better intuition for the interactions between components shown in this chapter.

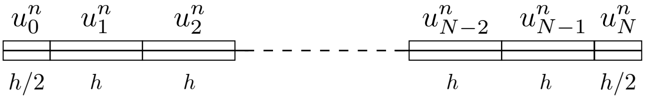


Fig. 11.1: Alternative interpretation of the discretisation of $u(x,t)$ to a grid function u_l^n. The continuous system is divided into $N-1$ sections of length h plus 2 sections of length $h/2$ at the boundaries. Through this interpretation, the 'weight' of a grid point can be calculated from its physical parameters.

11.2 Connected ideal strings

When working with multiple interacting systems, one finds that notation becomes extremely important. Subscripts will be extensively used in the following for extra clarity, and although this results in something of a notational jungle, it is better to be explicit and avoid confusion in the end.

As a test case for the following sections, consider two ideal strings[1] of length L_u and L_w (both in m) their transverse displacement denoted as $u = u(x,t)$ and $w = w(\chi, t)$ (both in m) respectively (see Section 2.4). The systems are defined for $x \in \mathcal{D}_u$ with domain $\mathcal{D}_u = [0, L_u]$ and $\chi \in \mathcal{D}_w$ with domain $\mathcal{D}_w = [0, L_w]$ respectively. Notice that χ is used as the spatial coordinate for w to denote that the two systems use different coordinate systems. Connecting these systems at $x_\text{c} \in \mathcal{D}_u$ and $\chi_\text{c} \in \mathcal{D}_w$ yields the following system of PDEs:

$$\rho_u A_u \partial_t^2 u = T_u \partial_x^2 u - \delta(x - x_\text{c}) f, \tag{11.6a}$$

$$\rho_w A_w \partial_t^2 w = T_w \partial_\chi^2 w + \delta(\chi - \chi_\text{c}) f, \tag{11.6b}$$

[1] Recall that the ideal string is the 1D wave equation with $c = \sqrt{T/\rho A}$.

where subscripts u and w denote whether a variable belongs to system u or w respectively. Notice that the ∂_χ in Eq. (11.6b) denotes a partial derivative with respect to χ and is an identical operation to ∂_x, but on a different coordinate system. Furthermore, $f = f(t)$ is the connection force (in N) which should be equal and opposite for the connected systems according to Newton's third law (hence the inverse signs). The definition for f depends on the connection type, and two alternatives will be given shortly. Finally, the spatial Dirac delta function δ is defined as in Eq. (8.8), and localises the connection force along the systems.

Relative location of objects

As explained in Chapter 10 it is important to keep in mind the relative location of two interacting objects, as this will affect the signs of the force terms added to the PDEs. As opposed to the case of collisions, the connection will have a negative effect on the object 'above' and a positive effect on the one 'below' due to the 'pulling' behaviour of a connection. From the signs of the force terms in system (11.6), it can thus be concluded that u has been placed above w. If f is positively dependent on η (the relative displacement between the two objects at their respective connection locations), this will be defined as the object below subtracted from the object above. For system (11.6) this will be

$$\eta(t) = u(x_\text{c}, t) - w(\chi_\text{c}, t), \tag{11.7}$$

and will be used for a spring connection in Section 11.4.

11.2.1 Discrete time

One can then discretise the state variables u and w to grid functions u_l^n and w_m^n, using $x = lh_u$ and $\chi = mh_w$, where $l \in \{0, \dots, N_u\}$ and $m \in \{0, \dots, N_w\}$.[2] Also see Section 2.2.1. Furthermore, h_u and h_w are the values of the grid spacing (both in m) and $N_u + 1$ and $N_w + 1$ are the number of grid points for u_l^n and w_m^n respectively. Dividing Eqs. (11.6a) and (11.6b) by $\rho_u A_u$ and $\rho_w A_w$ respectively, yields

$$\delta_{tt} u_l^n = c_u^2 \delta_{xx} u_l^n - J_{l,u}(x_\text{c}) \frac{f^n}{\rho_u A_u}, \tag{11.8a}$$

$$\delta_{tt} w_m^n = c_w^2 \delta_{\chi\chi} w_m^n + J_{m,w}(\chi_\text{c}) \frac{f^n}{\rho_w A_w}, \tag{11.8b}$$

where $c_u = \sqrt{T_u/\rho_u A_u}$ and $c_w = \sqrt{T_w/\rho_w A_w}$. The spreading operators $J_{l,u}(x_\text{c}) = J_{l,o_u,u}(x_\text{c})$ and $J_{m,w}(\chi_\text{c}) = J_{m,o_w,w}(\chi_\text{c})$ are as defined in Section 8.2, and their orders o_u and o_w are left unspecified.

[2] Here, m is used for the spatial index of w_m^n to avoid double subscripts l_u and l_w.

The next step would be to solve for connection force f^n. First, one needs to isolate the schemes in system (11.8) at their respective connection locations x_c and χ_c. This is done by taking an inner product of each scheme in system (11.8) with their respective spreading operator $J_{l,u}(x_\text{c})$ and $J_{m,w}(\chi_\text{c})$ over discrete domains $d_u = \{0, \dots, N_u\}$ and $d_w = \{0, \dots, N_w\}$ respectively. Using identity (8.9) one can write

$$I_{l,u}(x_\text{c})\delta_{tt}u_l^n = c_u^2 I_{l,u}(x_\text{c})\delta_{xx}u_l^n - \|J_{l,u}(x_\text{c})\|_{d_u}^2 \frac{f^n}{\rho_u A_u}, \tag{11.9a}$$

$$I_{m,w}(\chi_\text{c})\delta_{tt}w_m^n = c_w^2 I_{m,w}(\chi_\text{c})\delta_{\chi\chi}w_m^n + \|J_{m,w}(\chi_\text{c})\|_{d_w}^2 \frac{f^n}{\rho_w A_w}. \tag{11.9b}$$

Here, interpolation operators $I_{l,u}(x_\text{c}) = I_{l,o_u,u}(x_\text{c})$ and $I_{m,w}(\chi_\text{c}) = I_{m,o_w,w}(\chi_\text{c})$ are as defined in Section 8.2. Notice that the order of these operators need to match their 'dual' spreading operator, but the orders o_u and o_w may differ.

The definition of the force depends on the connection type. Below, two alternatives will be presented: the rigid connection and the spring connection.

11.3 Rigid connection

The simplest connection-type is the *rigid connection*. This connection type states that the displacement of two connected points should always be equal, and thus the distance between them should be 0 at all times. For the rigid connection, the following is true:

$$u(x_\text{c}, t) = w(\chi_\text{c}, t), \tag{11.10}$$

which in discrete time becomes

$$I_{l,u}(x_\text{c})u_l^n = I_{m,w}(\chi_\text{c})w_m^n. \tag{11.11}$$

For a rigid connection, the following must also hold:

$$I_{l,u}(x_\text{c})\delta_{tt}u_l^n = I_{m,w}(\chi_\text{c})\delta_{tt}w_m^n. \tag{11.12}$$

In other words, if the displacement of two objects is equal, their acceleration must also be. This definition can then immediately be used to solve for f^n and the right-hand sides in system (11.9) can be substituted in Eq. (11.12) to get

$$c_u^2 I_{l,u}(x_\text{c})\delta_{xx}u_l^n - \|J_{l,u}(x_\text{c})\|_{d_u}^2 \frac{f^n}{\rho_u A_u} = c_w^2 I_{m,w}(\chi_\text{c})\delta_{\chi\chi}w_m^n + \|J_{m,w}(\chi_\text{c})\|_{d_w}^2 \frac{f^n}{\rho_w A_w},$$

which can be explicitly solved for f^n according to

$$f^n = \frac{c_u^2 I_{l,u}(x_\mathrm{c})\delta_{xx}u_l^n - c_w^2 I_{m,w}(\chi_\mathrm{c})\delta_{\chi\chi}w_m^n}{\frac{\|J_{l,u}(x_\mathrm{c})\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_\mathrm{c})\|_{d_w}^2}{\rho_w A_w}} \; . \tag{11.13}$$

This value can then be used in the update equation obtained after expanding system (11.8) as

$$u_l^{n+1} = \left(2 - 2\lambda_u^2\right) u_l^n + \lambda_u^2 \left(u_{l+1}^n + u_{l-1}^n\right) - u_l^{n-1} - J_{l,u}(x_\mathrm{c})\frac{k^2 f^n}{\rho_u A_u}, \tag{11.14a}$$

$$w_m^{n+1} = \left(2 - 2\lambda_w^2\right) w_m^n + \lambda_w^2 \left(w_{m+1}^n + w_{m-1}^n\right) - w_m^{n-1} + J_{m,w}(\chi_\mathrm{c})\frac{k^2 f^n}{\rho_w A_w}, \tag{11.14b}$$

where $\lambda_u = c_u k/h_u \leq 1$ and $\lambda_w = c_w k/h_w \leq 1$ are the Courant numbers for each individual scheme (see Section 2.4).

Figure 11.2 shows an implementation of system (11.8) with $x_\mathrm{c} = 0.25$ m and $\chi_\mathrm{c} = 0.75$ m. The ideal strings have the same mass per unit length, i.e., $\rho_u A_u = \rho_w A_w$, and the same length $L_u = L_w = 1$ m, but operate at different wave speeds $c_u = 300$ m/s and $c_w = 400$ m/s. The offset between the systems is made for clarity, and the locations connected by the grey line should have the same displacement as posed by the rigid connection.

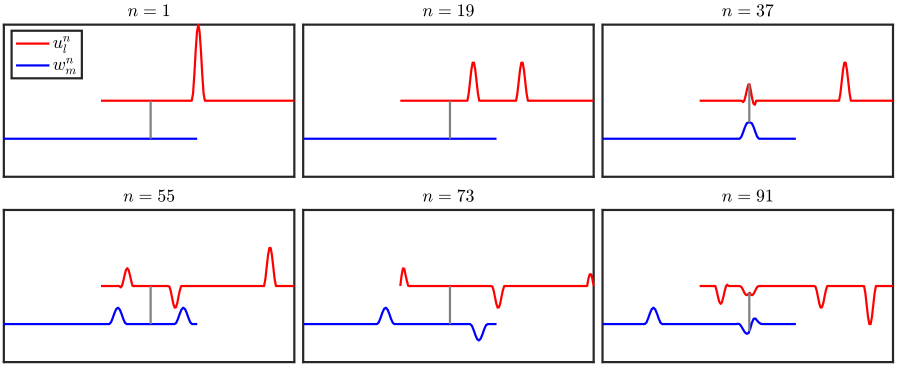


Fig. 11.2: The behaviour of two connected 1D wave equations. The systems are offset for clarity, but the relative displacement at the connection location is 0.

11.3.1 Notation simplification

In the above equations, the orders of the spreading and interpolation operators have been left unspecified to retain generality. If one would like to connect two systems at specified grid points (so not in-between), the notation can be greatly simplified.

Recalling that $I_{l,u}(x_{\text{c}}) = I_{l,o_u,u}(x_{\text{c}})$ and $I_{m,w}(\chi_{\text{c}}) = I_{m,o_w,w}(\chi_{\text{c}})$, one can set the interpolation orders to 0, i.e., $o_u = o_w = 0$, to yield the following short-hand notations

$$I_{l,0,u}(x_{\text{c}})u_l^n = u_{l_{\text{c}}}^n, \quad \text{and} \quad I_{m,0,w}(\chi_{\text{c}})w_m^n = w_{m_{\text{c}}}^n, \tag{11.15}$$

where $l_{\text{c}} = \lfloor x_{\text{c}}/h_u \rfloor$ and $m_{\text{c}} = \lfloor \chi_{\text{c}}/h_w \rfloor$,

$$\|J_{l,0,u}(x_{\text{c}})\|_{d_u}^2 = \frac{1}{h_u} \quad \text{and} \quad \|J_{m,0,w}(\chi_{\text{c}})\|_{d_w}^2 = \frac{1}{h_w}. \tag{11.16}$$

This simplifies Eqs. (11.9) to

$$\delta_{tt}u_{l_{\text{c}}}^n = c_u^2\delta_{xx}u_{l_{\text{c}}}^n - \frac{f^n}{\rho_u A_u h_u}, \tag{11.17a}$$

$$\delta_{tt}w_{m_{\text{c}}}^n = c_w^2\delta_{\chi\chi}w_{m_{\text{c}}}^n + \frac{f^n}{\rho_w A_w h_w}, \tag{11.17b}$$

which, after rewriting Eq. (11.12) to

$$\delta_{tt}u_{l_{\text{c}}}^n = \delta_{tt}w_{m_{\text{c}}}^n, \tag{11.18}$$

one can solve for f^n, yielding the following simplified form of Eq. (11.13)

$$f^n = \frac{c_u^2\delta_{xx}u_{l_{\text{c}}}^n - c_w^2\delta_{\chi\chi}w_{m_{\text{c}}}^n}{\frac{1}{\rho_u A_u h_u} + \frac{1}{\rho_w A_w h_w}}. \tag{11.19}$$

Through this simplification, one can now clearly see that the connection forces acting on each respective ideal string in Eq. (11.17) are scaled by the mass of one grid 'section' as explained in Section 11.1.1.

11.3.2 Energy Analysis

This section follows the energy analysis techniques shown in Section 3.4, though not explicitly following the steps for brevity. As the analysis has previously been performed on the 1D wave equation, this part will not be detailed here.

Starting with the FD scheme in Eq. (11.8a), one can take an inner product of the scheme (after a multiplication with $\rho_u A_u$) with $(\delta_{t\cdot}u_l^n)$ over discrete domain d_u, to get

$$\delta_{t+}\mathfrak{h}_u = \langle \delta_{t\cdot}u_l^n, -J_{l,u}(x_{\text{c}})f^n\rangle_{d_u}, \tag{11.20}$$

where $\mathfrak{h}_u$ is the total energy in system u and is as defined in Eq. (3.47). The same can be done for Eq. (11.8b) (after a multiplication with $\rho_w A_w$) by taking an inner product with $(\delta_{t\cdot} w_m^n)$ over discrete domain d_w to get

$$\delta_{t+}\mathfrak{h}_w = \langle \delta_{t\cdot} w_m^n, J_{m,w}(\chi_{\mathrm{c}}) f^n \rangle_{d_w}, \tag{11.21}$$

where $\mathfrak{h}_w$ is the total energy in system w. As the total energy in the system is an addition of $\mathfrak{h}_u$ and $\mathfrak{h}_w$, and using identity (8.9) for the right hand sides of Eqs. (11.20) and (11.21), one can write

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -I_{l,u}(x_{\mathrm{c}})\delta_{t\cdot} u_l^n f^n + I_{m,w}(\chi_{\mathrm{c}})\delta_{t\cdot} w_m^n f^n. \tag{11.22}$$

Finally, due to the rigid connection in Eq. (11.11), $I_{l,u}(x_{\mathrm{c}})\delta_{t\cdot} u_l^n = I_{m,w}(\chi_{\mathrm{c}})\delta_{t\cdot} w_m^n$ (if the displacements are equal, their velocities must also be) and the right hand side vanishes:

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = 0.$$

This shows that the rigid connection does not affect the total energy in the system and thus does not affect the stability of the scheme.

Figure 11.3 shows the energy of an implementation of the 1D wave system in (11.8) corresponding to the behaviour shown in Figure 11.2. One can observe that energy is transferred from

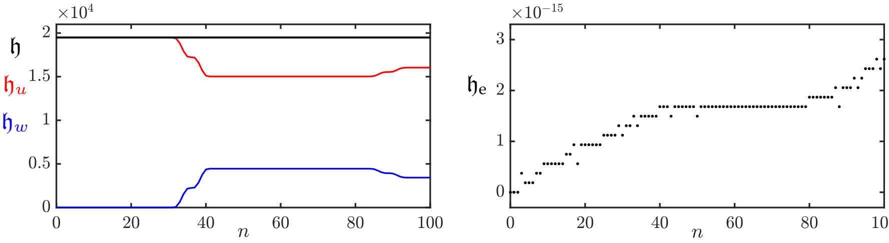


Fig. 11.3: The energy of u (red), the energy of w (blue), and the total (black) energy of the system of connected 1D wave equations in (11.8). The energy corresponds to Figure 11.2. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

11.3.3 Matrix form

One can write the system in Eq. (11.8) with a rigid connection in matrix form, albeit slightly more involved due to the interconnection of the schemes.

To start, the definition of the force in Eq. (11.13) must be substituted into the system, which, after expansion of the left hand side of the system, becomes

$$
\begin{aligned}
u_l^{n+1} &= 2u_l^n - u_l^{n-1} + c_u^2 k^2 \delta_{xx} u_l^n \\
&\quad - J_{l,u}(x_\text{c}) \frac{k^2}{\rho_u A_u} \left(\frac{c_u^2 I_{l,u}(x_\text{c}) \delta_{xx} u_l^n - c_w^2 I_{m,w}(\chi_\text{c}) \delta_{\chi\chi} w_m^n}{\frac{\|J_{l,u}(x_\text{c})\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_\text{c})\|_{d_w}^2}{\rho_w A_w}} \right), \qquad (11.23\text{a}) \\
w_m^{n+1} &= 2w_m^n - w_m^{n-1} + c_w^2 k^2 \delta_{xx} w_m^n \\
&\quad + J_{m,w}(\chi_\text{c}) \frac{k^2}{\rho_w A_w} \left(\frac{c_u^2 I_{l,u}(x_\text{c}) \delta_{xx} u_l^n - c_w^2 I_{m,w}(\chi_\text{c}) \delta_{\chi\chi} w_m^n}{\frac{\|J_{l,u}(x_\text{c})\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_\text{c})\|_{d_w}^2}{\rho_w A_w}} \right). \qquad (11.23\text{b})
\end{aligned}
$$

Using Dirichlet boundary conditions for both ideal strings, their values can be stored in the following vectors:

$$
\mathbf{u}^n = [u_1^n, \dots, u_{N_u-1}^n]^T, \quad \text{and} \quad \mathbf{w}^n = [w_1^n, \dots, w_{N_w-1}^n]^T.
$$

These vectors can then be concatenated to one larger state vector, and after the terms in Eqs. (11.23) are grouped by the grid functions at various time indices, one obtains the following compact matrix form of system (11.8):

$$
\begin{bmatrix} \mathbf{u}^{n+1} \\ \mathbf{w}^{n+1} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \mathbf{u}^{n} \\ \mathbf{w}^{n} \end{bmatrix} - \begin{bmatrix} \mathbf{u}^{n-1} \\ \mathbf{w}^{n-1} \end{bmatrix}, \qquad (11.24)
$$

where

$$
\mathbf{B} = \begin{bmatrix} \mathbf{B}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w \end{bmatrix} + \begin{bmatrix} -\mathbf{j}_u \\ \mathbf{j}_w \end{bmatrix} \begin{bmatrix} \mathbf{f}_u & -\mathbf{f}_w \end{bmatrix}.
$$

The matrix in the definition of $\mathbf{B}$ contains the operations of the 1D wave equation (also see Eq. (3.5)),

$$
\mathbf{B}_u = 2\mathbf{I}_{N_u-1} + c_u^2 k^2 (\mathbf{D}_{xx})_u, \quad \text{and} \quad \mathbf{B}_w = 2\mathbf{I}_{N_w-1} + c_w^2 k^2 (\mathbf{D}_{xx})_w, \qquad (11.25)
$$

where matrices $(\mathbf{D}_{xx})_u$ and $(\mathbf{D}_{xx})_w$ are as defined in Eq. (3.3) and are of the appropriate sizes. The vector multiplication in the definition of $\mathbf{B}$ results in a matrix, and adds the effect of the connection force to the system. Here, $\mathbf{j}_u$ and $\mathbf{j}_w$ are column vectors of size $(N_u - 1) \times 1$ and $(N_w - 1) \times 1$ containing the values of the spreading operators $J_{l,u}(x_\text{c})$ and $J_{m,w}(\chi_\text{c})$ respectively. Finally,

$$
\mathbf{f}_u = \frac{k^2}{\rho_u A_u} \left(\frac{c_u^2 \mathbf{i}_u (\mathbf{D}_{xx})_u}{\frac{\mathbf{i}_u \mathbf{j}_u}{\rho_u A_u} + \frac{\mathbf{i}_w \mathbf{j}_w}{\rho_w A_w}} \right), \quad \text{and} \quad \mathbf{f}_w = \frac{k^2}{\rho_w A_w} \left(\frac{c_w^2 \mathbf{i}_w (\mathbf{D}_{xx})_w}{\frac{\mathbf{i}_u \mathbf{j}_u}{\rho_u A_u} + \frac{\mathbf{i}_w \mathbf{j}_w}{\rho_w A_w}} \right),
$$

where $\mathbf{i}_u$ and $\mathbf{i}_w$ are row vectors of size $1 \times (N_u - 1)$ and $1 \times (N_w - 1)$ containing the values of the interpolation operators $I_{l,u}(x_\text{c})$ and $I_{m,w}(\chi_\text{c})$ respectively. Here, $\mathbf{i}_u \mathbf{j}_u$ and $\mathbf{i}_w \mathbf{j}_w$ are matrix-vector forms of $\|J_{l,u}(x_\text{c})\|_{d_u}^2$ and $\|J_{m,w}(\chi_\text{c})\|_{d_w}^2$

respectively (see Eq. (8.10)), and reduce to a scalar.

Equation (11.24) can then easily be rewritten in a one-step form as described in Section 3.5.1 and used for modal analysis.[3]

11.4 Spring connection

An alternative connection type is the *spring connection*. As in the rigid case, forces are still equal and opposite, but spring connections allow the relative displacement between the two connected elements to be non-zero. This relative displacement is used to determine the connection force. Interestingly, a nonlinear component can be added to this connection without making the system implicit. The most complex springs used in this project have a linear and a nonlinear (cubic) component, as well as a damping term. For ease of explanation, this section will only use a linear spring. A damped nonlinear spring will appear in Section 11.5.

The force between two components connected by a linear spring can be defined as

$$f = f(t) = K\eta, \tag{11.26}$$

where $K \geq 0$ is the spring constant (in N/m) and

$$\eta = \eta(t) = u(x_\text{c}, t) - w(\chi_\text{c}, t) \tag{11.27}$$

is the relative displacement between the two systems at their respective connection locations (in m).[4]

In discrete time, Eq. (11.26) becomes

$$f^n = K\mu_{t\cdot}\eta^n, \tag{11.28}$$

where

$$\eta^n = I_{l,u}(x_\text{c})u_l^n - I_{m,w}(\chi_\text{c})w_m^n. \tag{11.29}$$

Here, the centred averaging operator is used for stability (see Section 11.4.2), but when substituted into system (11.9) seems to make the system implicit. However, one can find an explicit solution, even for an arbitrary amount of connections [54]. These systems are therefore referred to as being *semi-implicit*, and the process of how to solve the system explicitly will be shown below.

[3] As the system does not exhibit damping, it could be rewritten and analysed directly, not using a one-step form.

[4] Note that if u was placed 'below' w (see Section 11.2), the signs of the force terms in system (11.8) would have been flipped and $u(x_\text{c}, t)$ would have been subtracted from $w(\chi_\text{c}, t)$ in Eq. (11.27) instead.

11.4.1 Explicit solution

Compared to the rigid connection in Section 11.3, solving for f^n requires an extra step. After isolating the schemes at their respective connection locations – resulting in Eqs. (11.9) – one needs to expand the scheme and solve for the states at $n+1$:

$$I_{l,u}(x_{\mathrm{c}})u_l^{n+1} = u^\star - \|J_{l,u}(x_{\mathrm{c}})\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u}, \tag{11.30a}$$

$$I_{m,w}(\chi_{\mathrm{c}})w_m^{n+1} = w^\star + \|J_{m,w}(\chi_{\mathrm{c}})\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w}, \tag{11.30b}$$

where

$$u^\star = I_{l,u}(x_{\mathrm{c}})(2u_l^n - u_l^{n-1}) + c_u^2 k^2 I_{l,u}(x_{\mathrm{c}})\delta_{xx}u_l^n,$$

and

$$w^\star = I_{m,w}(\chi_{\mathrm{c}})(2w_m^n - w_m^{n-1}) + c_w^2 k^2 I_{m,w}(\chi_{\mathrm{c}})\delta_{\chi\chi}w_m^n,$$

are the update equations of the system at their respective connection locations, without the term containing the connection force (as done in Chapter 10). Evaluating Eq. (11.29) at $n+1$ yields

$$\eta^{n+1} = I_{l,u}(x_{\mathrm{c}})u_l^{n+1} - I_{m,w}(\chi_{\mathrm{c}})w_m^{n+1},$$

into which Eqs. (11.30) can be substituted, as

$$\eta^{n+1} = u^\star - \|J_{l,u}(x_{\mathrm{c}})\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u} - \left(w^\star + \|J_{m,w}(\chi_{\mathrm{c}})\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w}\right). \tag{11.31}$$

A second definition for η^{n+1} can be obtained after expanding Eq. (11.28):

$$\eta^{n+1} = \frac{2f^n}{K} - \eta^{n-1} \tag{11.32}$$

and can be substituted into Eq. (11.31) to get

$$\frac{2f^n}{K} - \eta^{n-1} = u^\star - \|J_{l,u}(x_{\mathrm{c}})\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u} - \left(w^\star + \|J_{m,w}(\chi_{\mathrm{c}})\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w}\right). \tag{11.33}$$

Finally, one can group the terms for f^n

$$\left(\frac{2}{K} + \frac{\|J_{l,u}(x_{\mathrm{c}})\|_{d_u}^2 k^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_{\mathrm{c}})\|_{d_w}^2 k^2}{\rho_w A_w}\right) f^n = u^\star - w^\star + \eta^{n-1}$$

and solve for the force, solely based on known values of the system

$$f^n = \frac{u^\star - w^\star + \eta^{n-1}}{\frac{2}{K} + \frac{\|J_{l,u}(x_\text{c})\|^2_{d_u} k^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_\text{c})\|^2_{d_w} k^2}{\rho_w A_w}}\,. \tag{11.34}$$

Figure 11.4 shows the behaviour of system Eq. (11.8), connected with a spring with spring constant $K = 5\cdot 10^4$ N/m. The same parameters, excitation and connection locations are used as for the rigid connection in Section 11.3. Compared to the behaviour of the system with a rigid connection in Figure 11.2, one can observe that the distance between the two connected points gets larger as the wave passes the connection point, which corresponds to the extension of the spring.

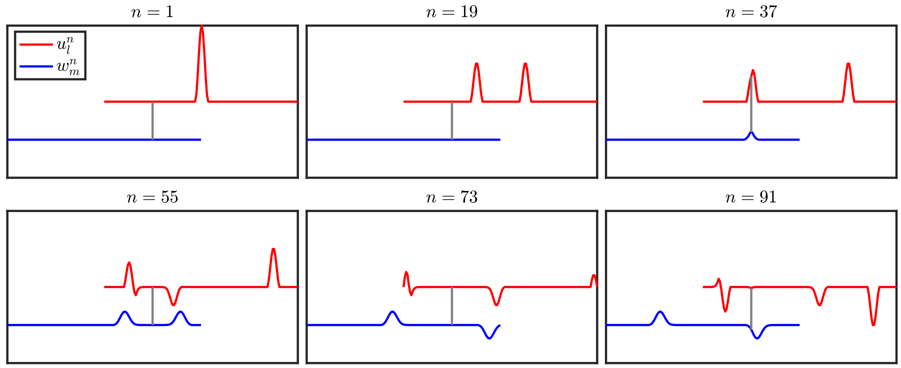


Fig. 11.4: The behaviour of two ideal strings connected using a spring. The systems are offset for clarity, but the relative displacement at the connection location at the start of the simulation is 0.

11.4.2 Energy analysis

This section follows the energy analysis techniques presented in Section 3.4 (without explicitly following the steps for brevity) and shows that the discretisation of the spring force chosen in Eq. (11.28) is inherently stable.

Following the same process as in Section 11.3.2, one can analyse system (11.8), and arrive at Eq. (11.22):

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -I_{l,u}(x_\text{c})\delta_{t\cdot}u_l^n f^n + I_{m,w}(\chi_\text{c})\delta_{t\cdot}w_m^n f^n,$$

which can be rewritten to

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -\delta_{t\cdot}\left(I_{l,u}(x_\text{c})u_l^n - I_{m,w}(\chi_\text{c})w_m^n\right) f^n.$$

One can then substitute the definitions for f^n and η^n from Eqs. (11.28) and

(11.29) to get

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -K(\delta_{t\cdot}\eta^n)(\mu_{t\cdot}\eta^n), \tag{11.35}$$

which, using identity (3.17d), can be rewritten to

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w + \mathfrak{h}_\mathrm{c}) = 0, \tag{11.36}$$

where

$$\mathfrak{h}_\mathrm{c} = \frac{K}{2}\left(\mu_{t-}(\eta^n)^2\right) \tag{11.37}$$

is the energy stored by the connection. As this definition is non-negative it does not affect the stability of the system. The spring constant K could potentially be infinitely large, which would effectively reduce the spring connection to a rigid connection presented in Section 11.3.

Figure 11.5 shows the energy of an implementation of system (11.8) connected with a spring with spring constant $K = 5 \cdot 10^4$. other parameters are the same as for the rigid connection in Section 11.3. One can observe that when compared to Figure 11.3, less energy is transferred from u to w, and some energy is stored in the spring shown in green.

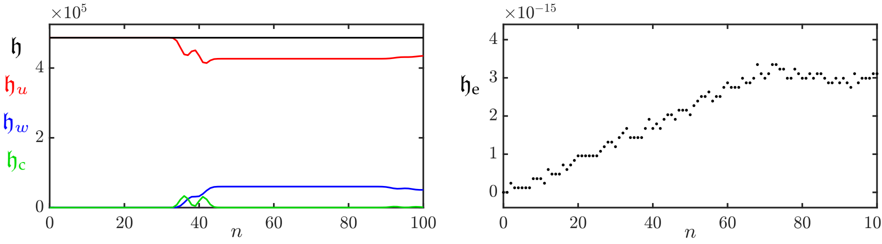


Fig. 11.5: The energy of u (red), the energy of w (blue), the energy of the spring connection (green) and the total energy (black) of the system of connected 1D wave equations in (11.8). The energy corresponds to the system in Figure 11.4. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

Unstable discretisation

To show why an averaging operator is used in Eq. (11.28), consider a more straightforward discretisation of Eq. (11.26) without the averaging operator, such that

$$f^n = K\eta^n.$$

Performing an energy analysis of the system would yield

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -K(\delta_{t\cdot}\eta^n)(\eta^n),$$

(instead of Eq. (11.35)) and using identity (3.17b), this can be rewritten to

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w + \mathfrak{h}_\mathrm{c}) = 0,$$

where

$$\mathfrak{h}_\mathrm{c} = \frac{K}{2}\left(\eta^n e_{t-}\eta^n\right).$$

As this is not necessarily non-negative, the connection places a larger restriction on the stability of the system at the connection location. In other words, $\lambda \leq 1$ for the ideal strings does not ensure stability at the connection location. Also see [21, pp. 190–192].

11.5 String-plate connection

As an example of a more complicated connected system used in papers [A] and [B], consider a stiff string connected to a plate using a nonlinear damped spring. This could be interpreted as a simplified form of how the string would be connected to the body in a stringed instrument. In the following, subscripts 's' and 'p' are used to denote a string or plate parameter respectively.

11.5.1 Continuous time

Consider a damped stiff string of length L (in m), its transverse displacement described by $u = u(\chi, t)$ (in m) defined for $t \geq 0$ and $\chi \in \mathcal{D}_\mathrm{s}$ where domain $\mathcal{D}_\mathrm{s} = [0, L]$. Its PDE is described by

$$\rho_\mathrm{s} A \partial_t^2 u = T\partial_\chi^2 u - E_\mathrm{s} I \partial_\chi^4 u - 2\sigma_{0,\mathrm{s}}\rho_\mathrm{s} A\partial_t u + 2\sigma_{1,\mathrm{s}}\rho_\mathrm{s} A \partial_t \partial_\chi^2 u, \tag{11.38}$$

where parameters are as in Eq. (4.3).

The transverse displacement of a damped rectangular thin plate of side lengths L_x and L_y (both in m) can be described as $w = w(x, y, t)$ (in m), which is defined for $t \geq 0$ and $(x, y) \in \mathcal{D}_p$ where domain $\mathcal{D}_p = [0, L_x] \times [0, L_y]$. Its PDE is defined as

$$\rho_\mathrm{p} H \partial_t^2 w = -D\Delta\Delta w - 2\sigma_{0,\mathrm{p}}\rho_\mathrm{p} H \partial_t w + 2\sigma_{1,\mathrm{p}}\rho_\mathrm{p} H \partial_t \partial_x^2 w, \tag{11.39}$$

where parameters are as in Eq. (6.37).

One can connect the above PDEs, by adding a localised connection force. After a division by $\rho_\mathrm{s} A$ and $\rho_\mathrm{p} H$ respectively the connected string-plate system becomes

$$\partial_t^2 u = c^2 \partial_\chi^2 u - \kappa_\text{s}^2 \partial_\chi^4 u - 2\sigma_{0,\text{s}} \partial_t u + 2\sigma_{1,\text{s}} \partial_t \partial_\chi^2 u - \delta(\chi - \chi_\text{c}) \frac{f}{\rho_\text{s} A}, \tag{11.40a}$$

$$\partial_t^2 w = -\kappa_\text{p}^2 \Delta\Delta w - 2\sigma_{0,\text{p}} \partial_t w + 2\sigma_{1,\text{p}} \partial_t \partial_x^2 w + \delta(x - x_\text{c}, y - y_\text{c}) \frac{f}{\rho_\text{p} H}, \tag{11.40b}$$

where $\delta(\chi - \chi_\text{c})$ (in m^{-1}) and $\delta(x - x_\text{c}, y - y_\text{c})$ (in m^{-2}) are the 1D and 2D spatial Dirac delta functions defined in Eqs. (8.8) and (11.5) respectively and locate the connection force at $\chi_\text{c} \in \mathcal{D}_s$ (in m) along the string and $(x_\text{c}, y_\text{c}) \in \mathcal{D}_p$ (in (m, m)) on the plate.

The force between the two components (in N) is set to be a nonlinear (cubic) damped spring defined as (used in e.g. [64] and in scaled form in [54])

$$f = f(t) = K_1 \eta + K_3 \eta^3 + R\dot{\eta}, \tag{11.41}$$

with linear and nonlinear spring coefficients K_1 (in N/m) and K_3 (in N/m^3) and damping coefficient R (in kg/s). Furthermore, the distance (in m) between the string and the plate at their respective connection locations is defined as

$$\eta = \eta(t) = u(\chi_\text{c}, t) - w(x_\text{c}, y_\text{c}, t). \tag{11.42}$$

11.5.2 Discrete time

To discretise $u(\chi, t)$, one can use grid function u_q^n where $n \in \mathbb{N}^0$ and $q \in \{0, \dots, N\}$ with number of grid points $N + 1$ (see Section 2.2.1). Next, $w(x, y, t)$ can be discretised using grid function $w_{l,m}^n$ with $l \in \{0, \dots, N_x\}$ and $m \in \{0, \dots, N_y\}$ where $N_x + 1$ and $N_y + 1$ are the number of grid points in the x and y direction respectively (see Section 6.1).

Using these grid functions, system (11.40) can then be discretised as

$$\begin{aligned} \delta_{tt} u_q^n = {} & c^2 \delta_{\chi\chi} u_q^n - \kappa_\text{s}^2 \delta_{\chi\chi\chi\chi} u_q^n - 2\sigma_{0,\text{s}} \delta_{t\cdot} u_q^n + 2\sigma_{1,\text{s}} \delta_{t-} \delta_{\chi\chi} u_q^n \\ & - J_q(\chi_\text{c}) \frac{f^n}{\rho_\text{s} A}, \end{aligned} \tag{11.43}$$

$$\begin{aligned} \delta_{tt} w_{l,m}^n = {} & -\kappa_\text{p}^2 \delta_\Delta \delta_\Delta w_{l,m}^n - 2\sigma_{0,\text{p}} \delta_{t\cdot} w_{l,m}^n + 2\sigma_{1,\text{p}} \delta_{t-} \delta_{xx} w_{l,m}^n \\ & + J_{l,m}(x_\text{c}, y_\text{c}) \frac{f^n}{\rho_\text{p} H}, \end{aligned} \tag{11.44}$$

where $J_q(\chi_\text{c}) = J_{q,o_\text{s}}(\chi_\text{c})$ is a 1D spreading operator of order o_s as defined in Section 8.2 and $J_{l,m}(x_\text{c}, y_\text{c}) = J_{(l,m),o_\text{p}}(x_\text{c}, y_\text{c})$ is a 2D spreading operator of order o_p as defined in Section 11.1.

The definition of the force in Eq. (11.41) can be discretised as[5]

$$f^n = K_1\mu_{tt}\eta^n + K_3(\eta^n)^2\mu_{t\cdot}\eta^n + R\delta_{t\cdot}\eta^n, \tag{11.45}$$

where

$$\eta^n = I_q(\chi_\text{c})u_q^n - I_{l,m}(x_\text{c}, y_\text{c})w_m^n. \tag{11.46}$$

Here, $I_q(\chi_\text{c}) = I_{q,o_\text{s}}(\chi_\text{c})$ and $I_{l,m}(x_\text{c}, y_\text{c}) = I_{(l,m),o_\text{p}}(x_\text{c}, y_\text{c})$ are interpolation operators of the same order as $J_q(\chi_\text{c})$ and $J_{l,m}(x_\text{c})$ respectively.

11.5.3 Solving for f

Following the same process as in Section 11.4.1, system (11.43) needs to be isolated at the connection locations. This is done by taking an inner product of the schemes in (11.43) with their respective spreading operators over discrete domains $d_u = \{0, \ldots, N\}$ and $d_w = \{0, \ldots, N_x\} \times \{0, \ldots, N_y\}$ respectively. Taking these inner products, expanding the δ_{tt} and $\delta_{t\cdot}$ operators (as these contain u_q^{n+1}) and solving for the states at $n+1$ yields

$$I_q(\chi_\text{c})u_q^{n+1} = u^\star - \|J_q(\chi_\text{c})\|_{d_u}^2 \frac{k^2 f^n}{\rho_\text{s}A(1+\sigma_{0,\text{s}}k)}\ , \tag{11.47a}$$

$$I_{l,m}(x_\text{c})w_{l,m}^n = w^\star + \|J_{l,m}(x_\text{c}, y_\text{c})\|_{d_w}^2 \frac{k^2 f^n}{\rho_\text{p}H(1+\sigma_{0,\text{p}}k)}\ , \tag{11.47b}$$

where

$$\begin{aligned} u^\star = \Big(&I_q(\chi_\text{c})(2u_q^n - u_q^{n-1}) + c^2k^2 I_q(\chi_\text{c})\delta_{\chi\chi}u_q^n - \kappa_\text{s}^2 k^2 I_q(\chi_\text{c})\delta_{\chi\chi\chi\chi}u_q^n \\ &+ \sigma_{0,\text{s}}kI_q(\chi_\text{c})u_q^{n-1} + 2\sigma_{1,\text{s}}k^2 I_q(\chi_\text{c})\delta_{t-}\delta_{\chi\chi}u_q^n\Big)/(1+\sigma_{0,\text{s}}k), \end{aligned} \tag{11.48a}$$

$$\begin{aligned} w^\star = \Big(&-\kappa_\text{p}^2 I_{l,m}(x_\text{c})\delta_\Delta\delta_\Delta w_{l,m}^n + \sigma_{0,\text{p}}kI_{l,m}(x_\text{c})w_{l,m}^{n-1} \\ &+ 2\sigma_{1,\text{p}}I_{l,m}(x_\text{c})\delta_{t-}\delta_{xx}w_{l,m}^n\Big)/(1+\sigma_{0,\text{p}}k), \end{aligned} \tag{11.48b}$$

are the update equations of the schemes at their respective connection locations without the force term.

Evaluating Eq. (11.46) at $n+1$ and substituting Eqs. (11.47) yields

$$\begin{aligned} \eta^{n+1} = u^\star &- \|J_q(\chi_\text{c})\|_{d_u}^2 \frac{k^2 f^n}{\rho_\text{s}A(1+\sigma_{0,\text{s}}k)} \\ &- \left(w^\star + \|J_{l,m}(x_\text{c}, y_\text{c})\|_{d_w}^2 \frac{k^2 f^n}{\rho_\text{p}H(1+\sigma_{0,\text{p}}k)}\right). \end{aligned} \tag{11.49}$$

[5] The second-order averaging operator has been chosen here to show an alternative discretisation of the linear term, but it could be replaced by a centred first-order averaging operator as used in Eq. (11.28).

Then expanding Eq. (11.45) to

$$f^n = \underbrace{\left(\frac{K_1}{4} + \frac{K_3(\eta^n)^2}{2} + \frac{R}{2k}\right)}_{r_+^n} \eta^{n+1} + \frac{K_1}{2}\eta^n + \underbrace{\left(\frac{K_1}{4} + \frac{K_3(\eta^n)^2}{2} - \frac{R}{2k}\right)}_{r_-^n} \eta^{n-1},$$

and solving for η^{n+1} yields

$$\eta^{n+1} = \frac{f^n}{r_+^n} - \frac{K_1}{2r_+^n}\eta^n - \frac{r_-^n}{r_+^n}\eta^{n-1}. \tag{11.50}$$

Substituting this into Eq. (11.49), one can find a definition for the connecting force

$$f^n = \frac{u^\star - w^\star + \frac{K_1}{2r_+^n}\eta^n + \frac{r_-^n}{r_+^n}\eta^{n-1}}{\frac{1}{r_+^n} + \frac{\|J_q(\chi_\text{c})\|_{d_u}^2 k^2}{\rho_\text{s} A(1+\sigma_{0,\text{s}}k)} + \frac{\|J_{l,m}(x_\text{c},y_\text{c})\|_{d_w}^2 k^2}{\rho_\text{p} H(1+\sigma_{0,\text{p}}k)}}\,. \tag{11.51}$$

11.5.4 Implementation

This section shows an example of an implementation of the string-plate system. The parameters for the stiff string can be found in Table 4.1 (with $L = 1.5$ m and $T = 555$ N) and for the thin plate in Table 6.1 (with $H = 5 \cdot 10^{-4}$). Both systems use simply supported boundary conditions. Additional parameters used for the connection are

$$K_1 = 10^4 \text{ N/m}, \quad K_3 = 10^7 \text{ N/m}^3, \quad \text{and} \quad R = 10 \text{ kg/s}.$$

The `MATLAB` code of the implementation can be found online [110].[6] Figure 11.6 shows a visualisation of the string plate system excited with a raised cosine.

In the following, the **i** and **j** vectors are as used in Section 11.3.3 and are of the appropriate sizes. The matrices for the string can be found in Eq. (4.19) and those for the plate in Eq. (6.46). When implementing connections (or any other interactions for that matter), one mostly performs the following steps in the main loop:

1. Calculate the entire scheme without force terms:

$$\mathbf{u}^\star = \frac{\mathbf{B}_\text{s}\mathbf{u}^n + \mathbf{C}_\text{s}\mathbf{u}^{n-1}}{A_\text{s}}, \quad \text{and} \quad \mathbf{w}^\star = \frac{\mathbf{B}_\text{p}\mathbf{w}^n + \mathbf{C}_\text{p}\mathbf{w}^{n-1}}{A_\text{p}}\,. \tag{11.52}$$

2. Obtain $u^\star$ and $w^\star$:

$$u^\star = \mathbf{i}_u\mathbf{u}^\star \quad \text{and} \quad w^\star = \mathbf{i}_w\mathbf{w}^\star. \tag{11.53}$$

[6] Note that in the online code, L, L_x and L_y have been halved to reduce computations and avoid crashes on some machines.

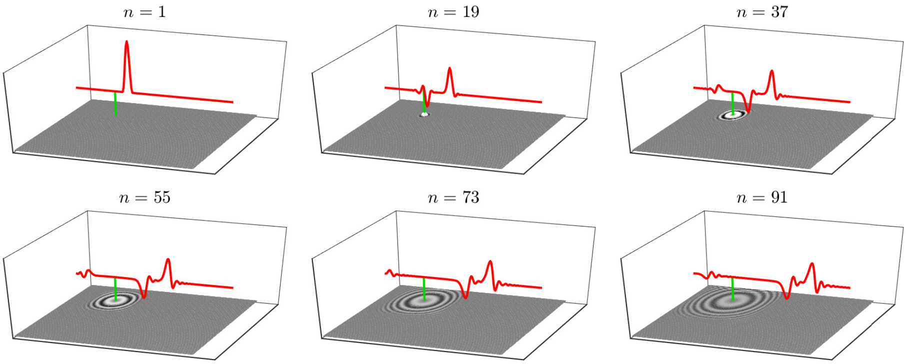


Fig. 11.6: The behaviour of the string-plate system connected with a nonlinear damped spring. The string is shown in red, the plate in gray and the connection in green.

3. Calculate the connection force f^n (Eq. (11.51)).
4. Add force terms to the schemes in Eq. (11.52):

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \mathbf{j}_u \frac{f^n k^2}{\rho_\text{s} A(1+\sigma_{0,\text{s}})}, \quad \text{and} \quad \mathbf{w}^{n+1} = \mathbf{w}^\star + \mathbf{j}_w \frac{f^n k^2}{\rho_\text{p} H(1+\sigma_{0,\text{p}})}. \tag{11.54}$$

Calculating $\mathbf{u}^\star$ and $\mathbf{w}^\star$ beforehand reduces computations, and allows $u^\star$ and $w^\star$ to be more easily obtained.

11.5.5 Energy analysis

Recalling the total energy and damping terms for the string and plate in Sections 4.4 and 6.3.5 respectively, one can – similar to Eq. (11.22) – arrive at the following:

$$\delta_{t+}(\mathfrak{h}_\text{s} + \mathfrak{h}_\text{p}) + \mathfrak{q}_\text{s} + \mathfrak{q}_\text{p} = -I_q(\chi_\text{c})(\delta_{t\cdot} u_q^n) f^n + I_{l,m}(x_\text{c})(\delta_{t\cdot} w_{l,m}^n) f^n, \tag{11.55}$$

which can be rewritten to

$$\delta_{t+}(\mathfrak{h}_\text{s} + \mathfrak{h}_\text{p}) + \mathfrak{q}_\text{s} + \mathfrak{q}_\text{p} = -\delta_{t\cdot}\left(I_q(\chi_\text{c}) u_q^n - I_{l,m}(x_\text{c}) w_{l,m}^n\right) f^n.$$

Substituting the definitions for f^n and η^n from Eqs. (11.45) and (11.46) respectively, yields

$$\delta_{t+}(\mathfrak{h}_\text{s} + \mathfrak{h}_\text{p}) + \mathfrak{q}_\text{s} + \mathfrak{q}_\text{p} = -(\delta_{t\cdot}\eta^n)(K_1 \mu_{tt}\eta^n + K_3(\eta^n)^2 \mu_{t\cdot}\eta^n + R\delta_{t\cdot}\eta^n).$$

Due to the nonlinear dependency on η^n one must isolate δ_{t+} from the cubic term manually, according to

$$
\begin{aligned}
& K_3(\eta^n)^2(\delta_{t\cdot}\eta^n)(\mu_{t\cdot}\eta^n) \\
&= \frac{K_3(\eta^2)}{2k}(\eta^{n+1}-\eta^{n-1})\frac{1}{2}(\eta^{n+1}+\eta^{n-1}) \\
&= \frac{K_3(\eta^n)^2}{4k}\left((\eta^{n+1})^2-(\eta^{n-1})^2\right) \\
&= \frac{K_3}{4k}\left((\eta^{n+1}\eta^n)^2-(\eta^n\eta^{n-1})^2\right) \\
&= \delta_{t+}\left(\frac{K_3}{4}(\eta^n\eta^{n-1})^2\right).
\end{aligned}
$$

Finally, using identity (3.17e) for the linear term, the following balance follows

$$\delta_{t+}(\mathfrak{h}_\mathrm{s}+\mathfrak{h}_\mathrm{p}+\mathfrak{h}_\mathrm{c}) = -\mathfrak{q}_\mathrm{s}-\mathfrak{q}_\mathrm{p}-\mathfrak{q}_\mathrm{c}, \tag{11.56}$$

where the energy stored by the spring connection is

$$\mathfrak{h}_\mathrm{c} = \frac{K_1}{8}(\eta^n+\eta^{n-1})^2+\frac{K_3}{4}(\eta^n\eta^{n-1})^2,$$

and the damping term of the connection is

$$\mathfrak{q}_\mathrm{c} = R(\delta_{t\cdot}\eta^n)^2.$$

Figure 11.7 shows the energetic output of the string-plate system corresponding to the behaviour in Figure 11.6. One can observe that energy is transferred from the string to the plate due to the connection. Furthermore, due to the high value for spring-damping R, the total energy decreases substantially as the excitation reaches the connection location along the string.

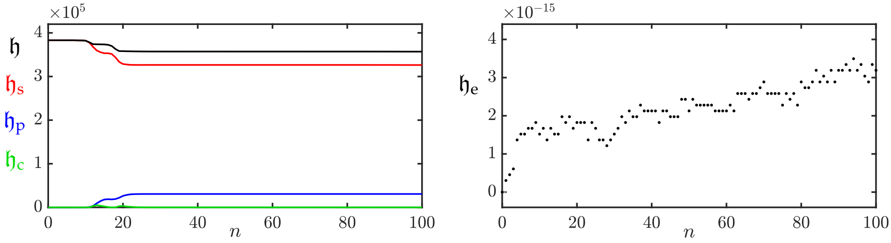


Fig. 11.7: The energy of the string (red), the plate (blue), the spring connection (green) and the total energy (black) of the connected string-plate system (11.43). The energy corresponds to the system in Figure 11.6. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

Part V

Contributions

Contributions

This part presents the contributions made throughout this PhD project, and can be seen as an extended summary of the published work in Part VII. As much as possible, the summaries relate the content of the papers to the theory presented in the rest of the thesis.

This part starts by introducing the dynamic grid in Chapter 12, a method to dynamically vary grid configurations in FDTD simulations published in paper [G], and extends the paper by providing implementation details and design considerations. Chapter 13 provides details on real-time implementation of physical models using FDTD methods, which have been used for many of the contributions. Finally, several real-time implementations of full instrument models that have been developed during the PhD will be presented: Chapter 14 provides an extended summary of papers [A] and [B], which present three instrument-inspired case-studies using a large-scale modular environment. Chapter 15 summarises papers [D] and [E], presenting the tromba marina, and provides extra information on the implementation of the algorithm. Finally, Chapter 16 provides an extended summary of paper [H] and extends the paper by providing design considerations and implementation details.

Chapter 12

The Dynamic Grid

This chapter provides an extended summary to the paper "Dynamic grids for Finite-Difference Schemes in Musical Instrument Simulations" [G]. The paper presents a novel method to smoothly add and remove grid points from a FD scheme, which allows for dynamic parameter variations without compromising stability or quality of the simulation (see Section 2.4.4).

After a brief introduction, this chapter summarises paper [G] and extends it by providing details on the implementation of the displacement correction and the modal analysis shown in the paper. This chapter continues by providing additional results of various experiments to substantiate choices made in the paper. Finally, this chapter lists several examples of potential future use cases for the method presented here.

12.1 Background and motivation

Simulating musical instruments using physical modelling – as mentioned in Chapter 1 – 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 (strings, acoustic tubes), thickness (membranes, plates) and size of the system in general. Using FDTD methods to discretise PDEs, constrains the simulation to a grid of a finite amount of points. As explained in Section 2.4.4, the definition of this grid ties the parameters set by the user to the stability and quality of the simulation, making FDTD methods extremely inflexible to parameter changes.

Apart from being potentially sonically interesting, dynamic parameter changes also happen in the real world. A prime example is the trombone published in paper [H] using the method presented here. A collection of examples for potential future use cases of the method are listed in Section 12.4.

12.2 Extended summary

This section summarises the *dynamic grid method* presented in paper [G] and extends the paper by providing details on the implementation.

12.2.1 Problem statement

Consider the 1D wave equation as presented in Section 2.4, describing the motion of a system of length L (in m), its state is denoted by $q = q(x,t)$, and is defined for $t \geq 0$ and $x \in \mathcal{D}$ with domain $\mathcal{D} = [0, L]$. This state variable can be discretised to a grid function q_l^n with $n \in \mathbb{N}^0$ and $l \in \{0, \ldots, N\}$, where N is the number of intervals between the grid points. The PDE in Eq. (2.38) (using state variable q) can then be discretised to the following FD scheme:

$$\delta_{tt} q_l^n = c^2 \delta_{xx} q_l^n. \tag{12.1}$$

If one would like to dynamically vary the wave speed c, several issues arise. Performing the usual calculations for the number of intervals N, Courant number $\lambda \leq 1$ and grid spacing h (Eq. (2.53)),

$$h := ck, \quad N := \left\lfloor \frac{L}{h} \right\rfloor, \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}, \tag{12.2}$$

shows that a change in c can cause abrupt changes in N due to the flooring operation. One could avoid this issue by fixing N at the start of the simulation and decrease c, i.e., tune it away from the stability condition. The issue here, is that the simulation quality and bandwidth decreases very rapidly as explained in Section 2.4.4.

Paper [G] proposes a *fractional* number of intervals $\mathcal{N}$, where $N = \lfloor \mathcal{N} \rfloor$, such that grid points could potentially be added and removed from the grid without causing artefacts. As the number of intervals is fractional, the flooring operation in Eq. (12.2) can be removed, and h does not have to be recalculated. Equation (12.2) can be changed to

$$h := ck, \quad \mathcal{N} := \frac{L}{h}, \quad \lambda := \frac{ck}{h}, \tag{12.3}$$

which results in that the stability condition is always satisfied with equality. Issues regarding simulation quality and bandwidth could thus be resolved. In the following, the variables c, h, $\mathcal{N}$ and N will receive a superscript n as they are time-varying.

12.2.2 Splitting the scheme

Rather than working with the scheme in Eq. (12.1) directly, paper [G] proposes to split it into two separate subsystems, according to

$$\delta_{tt}u_{l_u}^n = (c^n)^2\delta_{xx}u_{l_u}^n, \tag{12.4a}$$

$$\delta_{tt}w_{l_w}^n = (c^n)^2\delta_{xx}w_{l_w}^n, \tag{12.4b}$$

where $l_u \in \{0, \dots, M^n\}$ and $l_w \in \{0, \dots, M_w^n\}$ and integers

$$M^n = \lceil 0.5\mathcal{N}^n \rceil \quad \text{and} \quad M_w^n = \lfloor 0.5\mathcal{N}^n \rfloor \tag{12.5}$$

are the number of intervals between grid points of each respective subsystem. This yields a total of $M^n + M_w^n + 2$ grid points, which is one more than the original scheme in Eq. (12.1). Paper [G] shows that this system can be shown to exhibit identical behaviour to the original system, using the boundary conditions described shortly.

Locations of grid points

The schemes in Eq. (12.4) are placed on the same domain x, where the left boundary of $u_{l_u}^n$ and the right boundary of $w_{l_w}^n$ – referred to as the *outer boundaries* – are placed at the following locations:

$$x_{u_0}^n = 0, \quad x_{w_{M_w^n}}^n = L. \tag{12.6}$$

Here, $x_{q_l}^n$ is the location of grid point q_l (in m from the left boundary) at time index n.

The right boundary of u and the left boundary of w – referred to as the *inner boundaries* – are placed at the following locations:

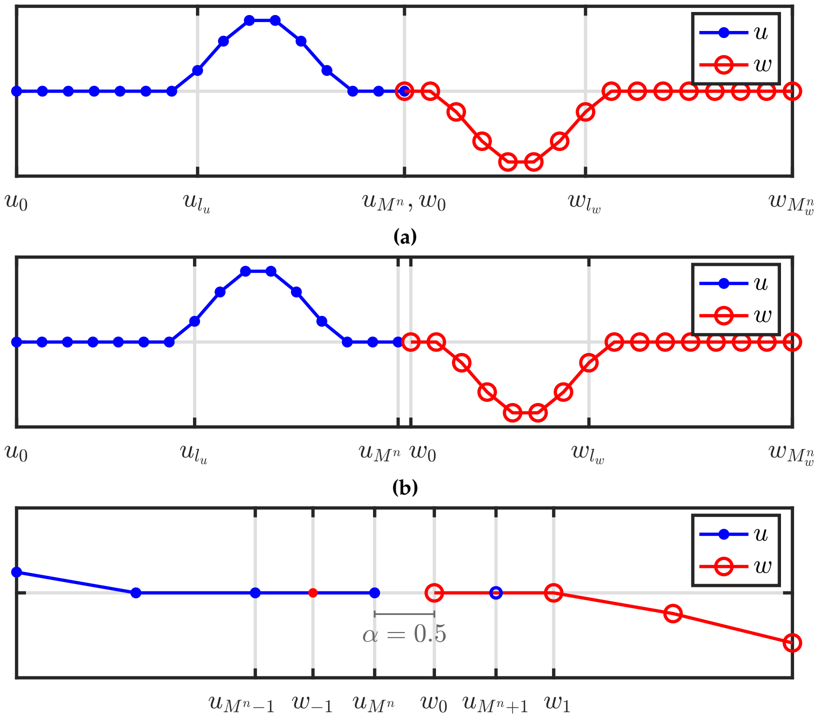
$$x_{u_{M^n}}^n = M^n h^n, \quad x_{w_0}^n = L - M_w^n h^n. \tag{12.7}$$

If $\mathcal{N}^n$ is an integer, the inner boundaries perfectly overlap (see Figure 12.1a). If the wave speed c^n changes, which consequently changes h^n according to Eq. (12.3), the outer boundaries will remain at their respective locations given by Eq. (12.6) and all other grid points will move to or from the outer boundary of their respective system according to[1]

$$x_{u_{l_u}}^n = l_u h^n, \quad x_{w_{l_w}}^n = L - (M_w^n - l_w)h^n. \tag{12.8}$$

See Figure 12.1b. As an example, if c^n decreases, h^n decreases, causing all grid points of system u to move towards the left boundary and all grid points of

[1] Notice that Eq. (12.8) applies to all grid points, and includes the definitions for the outer and inner boundaries in Eqs. (12.6) and (12.7).



(a)

(b)

(c)

Fig. 12.1: Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but 'x^n' is omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($\mathcal{N}^n = 30$, $x^n_{u_{M^n}} = x^n_{w_0}$). (b) When c^n, and consequently h^n, are decreased and the positions of the grid points change ($\mathcal{N}^n = 30.5$, $x^n_{u_{M^n}} \neq x^n_{w_0}$). (c) Figure 12.1b zoomed-in around the inner boundaries. The virtual grid points $u^n_{M^n+1}$ and w^n_{-1} are shown together with the distance between them, expressed using α in Eq. (12.9). (Taken from paper [G].)

system w to move towards the right boundary (except for grid points at the outer boundaries).

The distance between the inner boundaries is expressed as the fractional part of $\mathcal{N}^n$

$$\alpha = \alpha^n = \mathcal{N}^n - N^n \tag{12.9}$$

and essentially states how many times the grid spacing h^n can fit between the inner boundaries, which is always less than once (see Figure 12.1c). If the value of N^n changes, a grid point will be added to or removed from the grid as will be shown in Section 12.2.3.

Boundary conditions

For the outer boundaries, Dirichlet boundary conditions are used (see Eq. (2.48a)). The conditions for the inner boundaries are slightly more involved, and play a large part in the working of the method. In order for the system to exhibit the same behaviour as the original system in Eq. (12.1), the inner boundaries must have an identical displacement if they are perfectly overlapping, i.e.,

$$u_{M^n}^n = w_0^n, \quad \text{if } \alpha = 0, \tag{12.10}$$

and acts like a rigid connection between the inner boundaries (see Section 11.3). It is shown in paper [G], that in the perfectly overlapping case, identical behaviour to the original system can be obtained if: 1) the virtual grid points $u_{M^n+1}^n$ and w_{-1}^n are defined by the (centred) Neumann boundary condition (Eq. (2.48b)), and 2) the rigid connection in Eq. (12.10) is imposed on $u_{M^n}^n$ and w_0^n.

If parameters are varied, and the inner boundaries are no longer overlapping, the rigid connection will not be imposed anymore, and other definitions for the virtual grid points must be found. Using quadratic Lagrange interpolation to calculate the virtual grid points, shows excellent behaviour when parameters are varied, and continues to satisfy the requirement in Eq. (12.10) for perfectly overlapping boundaries (see Section 12.3 for experiments with other interpolators). At the inner boundaries, a definition of the virtual grid points is given as

$$u_{M^n+1}^n = \frac{\alpha-1}{\alpha+1}u_{M^n}^n + w_0^n - \frac{\alpha-1}{\alpha+1}w_1^n, \tag{12.11a}$$

$$w_{-1}^n = -\frac{\alpha-1}{\alpha+1}u_{M^n-1}^n + u_{M^n}^n + \frac{\alpha-1}{\alpha+1}w_0^n. \tag{12.11b}$$

How these coefficients are obtained will be shown below.

Lagrange interpolation

The coefficients in Eq. (12.11) were obtained using the Lagrange interpolation formula, where the coefficient at the i^{th} interpolation index is calculated as

$$I_i(x) = \prod_{j=0, j\neq i}^{o} \frac{x - x_j}{x_i - x_j}\ , \tag{12.12}$$

with interpolation order o, which, in this case, is 2. As u_{M^n} is the leftmost grid point used for the interpolation, its location is set to 0 and the values used are normalised by h^n for simplicity. To calculate the coefficients in Eq. (12.11a), the following locations were used in Eq. (12.12) (refer to Figure 12.1c):

$$\begin{aligned} x_0 &= x_{u_{M^n}} = 0, \\ x_1 &= x_{w_0} = \alpha, \\ x_2 &= x_{w_1} = \alpha + 1. \end{aligned} \tag{12.13}$$

The virtual grid point is the point calculated by the interpolation and will be at

$$x = x_{u_{M^n+1}} = 1. \tag{12.14}$$

Writing out the interpolation coefficient for index $i = 0$, yields

$$\begin{aligned} I_0(x) &= \left(\frac{1-\alpha}{0-\alpha}\right)\left(\frac{1-(\alpha+1)}{0-(\alpha+1)}\right), \\ &= \frac{\alpha-1}{\alpha+1}. \end{aligned}$$

This process can be repeated to obtain the other coefficients.

To obtain the coefficients for Eq. (12.11b), one can alter the locations in Eq. (12.13), or simply reverse the interpolator and apply it to the appropriate grid points.

12.2.3 Adding and removing grid points

Before continuing, it is useful to write the state of the system in vectors:

$$\mathbf{u}^n = [u_1^n, \dots, u_{M^n}^n]^T, \text{ and } \mathbf{w}^n = [w_0^n, \dots, w_{M_w^n-1}^n]^T, \tag{12.15}$$

and have M^n and M_w^n entries respectively. Notice that u_0^n and $w_{M_w^n}^n$ are not included, as Dirichlet boundary conditions are used.

If c^n and – through Eq. (12.2) – h^n are decreased, it might happen that the inner boundaries surpass the virtual grid points and $N^n > N^{n-1}$. In this case, a grid point must be added to the system, which will be done according to the following:

$$\text{if } N^n > N^{n-1} \begin{cases} \mathbf{u}^n = [(\mathbf{u}^n)^T, I_3\mathbf{v}^n]^T & \text{if } N^n \text{ is odd,} \\ \mathbf{w}^n = [I_3^{\leftarrow}\mathbf{v}^n, (\mathbf{w}^n)^T]^T & \text{if } N^n \text{ is even.} \end{cases} \tag{12.16}$$

Here,

$$\mathbf{v}^n = [u_{M^n-1}^n, u_{M^n}^n, w_0^n, w_1^n]^T$$

and cubic Lagrangian interpolator (see Eq. (12.12))

$$I_3 = \begin{bmatrix} -\frac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} & \frac{2\alpha}{\alpha+2} & \frac{2}{\alpha+2} & -\frac{2\alpha}{(\alpha+3)(\alpha+2)} \end{bmatrix}, \tag{12.17}$$

where $I_3^{\leftarrow}$ is a flipped version of (12.17).

If the opposite happens: c^n and h^n increase, and $N^n < N^{n-1}$, a grid point can be removed from the system according to the following:

$$\text{if } N^n < N^{n-1} \begin{cases} \mathbf{u}^n = [u_0^n, u_1^n ..., u_{M^n-1}^n]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w_1^n, w_2^n ..., w_{M_w^n}^n]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{12.18}$$

As mentioned in [G], the split of the original system does not have to be at the center (as presented here), but can be anywhere along the system, the limit being one point from the boundary. Thus, if M^n and M_w^n are calculated using

$$M^n = N^n - 1, \quad \text{and} \quad M_w^n = 1, \tag{12.19}$$

the method is still valid. Notice that if this definition is used, grid points will only be added and removed from $\mathbf{u}^n$, and Eqs. (12.16) and (12.18) reduce to

$$\mathbf{u}^n = [(\mathbf{u}^n)^T, I_3\mathbf{v}^n]^T, \quad \text{if } N^n > N^{n-1} \tag{12.20}$$

and

$$\mathbf{u}^n = [u_0^n, u_1^n ..., u_{M^n-1}^n]^T, \quad \text{if } N^n < N^{n-1}, \tag{12.21}$$

respectively.

12.2.4 Displacement correction

An issue that arises when increasing c^n, is that $u_{M^n}^n \not\approx w_0^n$ at the time of removal. As $\alpha \approx 0$ at this moment, this violates the rigid connection in Eq. (12.10). Paper [G] proposes to 'correct' the state of the grid points at the inner boundaries, which is referred to as *displacement correction*, and will be detailed here.[2]

Using 0th-order spreading interpolators $J_{l_u}(x_{u_{M^n}}^n) = J_{l_u,0}(x_{u_{M^n}}^n)$ and $J_{l_w}(x_{w_0}^n) = J_{l_w,0}(x_{w_0}^n)$ as defined in Section 8.2, system (12.4) can be extended to contain an artificial spring connection at the inner boundaries as[3]

$$\delta_{tt}u_{l_u}^n = (c^n)^2\delta_{xx}u_{l_u}^n + J_{l_u}(x_{u_{M^n}}^n)F_\text{c}^n, \tag{12.22a}$$

$$\delta_{tt}w_{l_w}^n = (c^n)^2\delta_{xx}w_{l_w}^n - J_{l_w}(x_{w_0}^n)F_\text{c}^n, \tag{12.22b}$$

[2] The term *state correction* (as used in paper [H]) might be more appropriate here, as the state of a system described by a FD scheme might not refer to a 'displacement'.

[3] Paper [G] uses subscripts u and w rather than l_u and l_w for the spreading operators, but this has been changed here, for coherency with the rest of this thesis.

where the correction effect is determined by a linear damped spring (see Chapter 11)

$$F_{\text{c}}^n = \beta\left(\mu_{t\cdot}\eta^n + \sigma_0\delta_{t\cdot}\eta^n\right). \tag{12.23}$$

Here, σ_0 is a damping coefficient and the difference between the states of the system at the inner boundaries is defined as

$$\eta^n \triangleq w_0^n - u_{M^n}^n. \tag{12.24}$$

Notice that, as the spreading operators are of 0th-order, one can simplify the notation as done in Section 11.3.1 and use subscripts rather than interpolation operators.

Furthermore, $\beta = \beta^n = \beta(\alpha^n)$ scales the correction effect depending on the value of α. This function has to be defined such that when $\alpha = 0$, $\beta \to \infty$ and the correction effect acts like a rigid connection. If, on the other hand $\alpha \to 1$, the correction effect should vanish, according to $\beta \to 0$. The following function that satisfies these conditions was proposed in paper [G]:

$$\beta = \frac{1-\alpha}{\alpha+\varepsilon}\,, \tag{12.25}$$

where $0 \leq \varepsilon \ll 1$ prevents a division by 0. Paper [G] states that it can be shown that when implementing the correction effect, a division by 0 can be prevented, and $\varepsilon = 0$ will still yield a defined solution. As an extension to paper [G], details on this implementation will be given here.

Implementation

Following a similar process to Section 11.4.1, one can expand the temporal FD operators of system (12.22) at the connection location to get

$$u_{M^n}^{n+1} = 2u_{M^n}^n - u_{M^n}^{n-1} + (c^n)^2k^2\delta_{xx}u_{M^n}^n + \frac{k^2}{h^n}F_{\text{c}}^n, \tag{12.26a}$$

$$w_0^{n+1} = 2w_0^n - w_0^{n-1} + (c^n)^2k^2\delta_{xx}w_{l_w}^n - \frac{k^2}{h^n}F_{\text{c}}^n, \tag{12.26b}$$

and substituting this into Eq. (12.24) evaluated at $n+1$, yields

$$\eta^{n+1} = w^\star - \frac{k^2}{h^n}F_{\text{c}}^n - \left(u^\star + \frac{k^2}{h^n}F_{\text{c}}^n\right), \tag{12.27}$$

where

$$u^\star = 2u_{M^n}^n - u_{M^n}^{n-1} + (c^n)^2k^2\delta_{xx}u_{M^n}^n,$$

and

$$w^\star = 2w_0^n - w_0^{n-1} + (c^n)^2k^2\delta_{xx}w_0^n,$$

are the update equations of the schemes without the connection force.

Another definition of η^{n+1} can be obtained by expanding Eq. (12.23) and solving for η^{n+1} according to

$$\begin{aligned}
F_\text{c}^n &= \beta\left(\frac{1}{2}\left(\eta^{n+1}+\eta^{n-1}\right)+\frac{\sigma_0}{2k}\left(\eta^{n+1}-\eta^{n-1}\right)\right),\\
F_\text{c}^n &= \left(\frac{\beta(1+\sigma_0/k)}{2}\right)\eta^{n+1}+\left(\frac{\beta(1-\sigma_0/k)}{2}\right)\eta^{n-1},\\
\xLeftrightarrow{\text{Eq. (12.25)}}\quad \eta^{n+1} &= \left(\frac{2(\alpha+\varepsilon)}{(1+\sigma_0/k)(1-\alpha)}\right)F_\text{c}^n-\underbrace{\frac{1-\sigma_0/k}{1+\sigma_0/k}}_{r}\eta^{n-1}. \qquad (12.28)
\end{aligned}$$

This definition can be substituted into Eq. (12.27) and solved for F_c^n according to

$$\begin{aligned}
\left(\frac{2(\alpha+\varepsilon)}{(1+\sigma_0/k)(1-\alpha)}\right)F_\text{c}^n-r\eta^{n-1} &= w^\star-u^\star-\frac{2k^2}{h^n}F,\\
\left(\frac{2h^n(\alpha+\varepsilon)+2k^2(1+\sigma_0/k)(1-\alpha)}{h^n(1+\sigma_0/k)(1-\alpha)}\right)F_\text{c}^n &= w^\star-u^\star+r\eta^{n-1},
\end{aligned}$$

and finally

$$F_\text{c}^n=\left(w^\star-u^\star+r\eta^{n-1}\right)\left(\frac{h^n(1+\sigma_0/k)(1-\alpha)}{2h^n(\alpha+\varepsilon)+2k^2(1+\sigma_0/k)(1-\alpha)}\right).$$

It is clear now that, even if $\varepsilon=0$, no division by 0 will occur no matter the value of α. In other words, if $\alpha=\varepsilon=0$ in Eq. (12.25), this will still yield a defined solution. The final equation for F_c^n can thus be written as

$$F_\text{c}^n=\left(w^\star-u^\star+r\eta^{n-1}\right)\left(\frac{h^n(1+\sigma_0/k)(1-\alpha)}{2h^n\alpha+2k^2(1+\sigma_0/k)(1-\alpha)}\right). \qquad (12.29)$$

This can then be substituted into system (12.22) and used to calculate $u_{M^n}^{n+1}$ and w_0^{n+1} respectively.

12.2.5 Matrix form

The vectors in Eq. (12.15) can be concatenated to one larger state vector as[4]

$$\boldsymbol{\mathcal{U}}^n=\begin{bmatrix}\mathbf{u}^n\\ \mathbf{w}^n\end{bmatrix}, \qquad (12.30)$$

[4] Note that, although $\boldsymbol{\mathcal{U}}^n$ is not a matrix, it is capitalised for coherency with paper [G].

and has $\mathcal{M}^n = M^n + M_w^n$ elements. Using $\mathcal{M}^n \times \mathcal{M}^n$ identity matrix $\mathbf{I}_{\mathcal{M}^n}$, system (12.22) can then be written in matrix form as

$$\begin{aligned}\mathbf{I}_{\mathcal{M}^n}\boldsymbol{\mathcal{U}}^{n+1} =& \mathbf{B}^n\boldsymbol{\mathcal{U}}^n - \mathbf{I}_{\mathcal{M}^n}\boldsymbol{\mathcal{U}}^{n-1} \\ &+ (\mathbf{j}\boldsymbol{\eta})\frac{\beta k^2}{2}\Big((1+\sigma_0/k)\boldsymbol{\mathcal{U}}^{n+1} + (1-\sigma_0/k)\boldsymbol{\mathcal{U}}^{n-1}\Big)\end{aligned} \tag{12.31}$$

where $\mathcal{M}^n \times 1$ column vector

$$\mathbf{j} = \mathbf{j}^n = [\mathbf{0}_{M^n-1}, 1/h^n, -1/h^n, \mathbf{0}_{M_w^n-1}]^T$$

contains the effect of the spreading operators with $P \times 1$ row vector $\mathbf{0}_P$. Furthermore, $1 \times \mathcal{M}^n$ row vector

$$\boldsymbol{\eta} = \boldsymbol{\eta}^n = [\mathbf{0}_{M^n-1}, -1, 1, \mathbf{0}_{M_w^n-1}]$$

vectorises the effect that η^n in Eq. (12.24) has on $\boldsymbol{\mathcal{U}}$. Note that as $\mathbf{j}$ is a column vector and $\boldsymbol{\eta}$ a row vector, the multiplication of the two yields an $\mathcal{M}^n \times \mathcal{M}^n$ diagonal matrix. Finally, $\mathbf{B}^n$ contains the usual $\mathbf{D}_{xx}$ matrices for both schemes in its top-left and bottom-right quadrants, as well as the definitions for the virtual grid points given in Eqs. (12.11):

$$\begin{aligned}\mathbf{B}^n =& 2\mathbf{I}_{\mathcal{M}^n} \\ &+ \lambda^2 \left[\begin{array}{cccc|cccc} \ddots & \ddots & \ddots & & & & \mathbf{0} & \\ & 1 & -2 & 1 & & & & \\ & & 1 & \left(\frac{\alpha-1}{\alpha+1}-2\right) & 1 & -\frac{\alpha-1}{\alpha+1} & & \\ \hline & & -\frac{\alpha-1}{\alpha+1} & 1 & \left(\frac{\alpha-1}{\alpha+1}-2\right) & 1 & & \\ & & & & 1 & -2 & 1 & \\ & \mathbf{0} & & & & \ddots & \ddots & \ddots \end{array}\right]\end{aligned}$$

which, as the method allows $\lambda = 1$ at all times, can be simplified to

$$\mathbf{B}^n = \left[\begin{array}{cccc|cccc} \ddots & \ddots & \ddots & & & & \mathbf{0} & \\ & 1 & 0 & 1 & & & & \\ & & 1 & \frac{\alpha-1}{\alpha+1} & 1 & -\frac{\alpha-1}{\alpha+1} & & \\ \hline & & -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1} & 1 & & \\ & & & & 1 & 0 & 1 & \\ \mathbf{0} & & & & & \ddots & \ddots & \ddots \end{array}\right]. \tag{12.32}$$

Equation (12.31) can then be rewritten to

$$\mathbf{A}^n\boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}^n\boldsymbol{\mathcal{U}}^n + \mathbf{C}^n\boldsymbol{\mathcal{U}}^{n-1}, \tag{12.33}$$

with

$$\mathbf{A}^n = \mathbf{I}_{\mathcal{M}^n} - \frac{\beta k^2(1+\sigma_0/k)}{2}\mathbf{j}\boldsymbol{\eta}, \quad \text{and} \quad \mathbf{C}^n = -\left(\mathbf{I}_{\mathcal{M}^n} - \frac{\beta k^2(1-\sigma_0/k)}{2}\mathbf{j}\boldsymbol{\eta}\right).$$

Notice that if points are added and removed at the boundary, and Eq. (12.19) is used, the $\mathbf{B}^n$ matrix can be written as

$$\mathbf{B}^n = \left[\begin{array}{cccc|c} \ddots & \ddots & \ddots & & \\ & 1 & 0 & 1 & \\ & & 1 & \frac{\alpha-1}{\alpha+1} & 1 \\ \hline & & -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1} \end{array}\right], \tag{12.34}$$

due to the Dirichlet boundary condition.

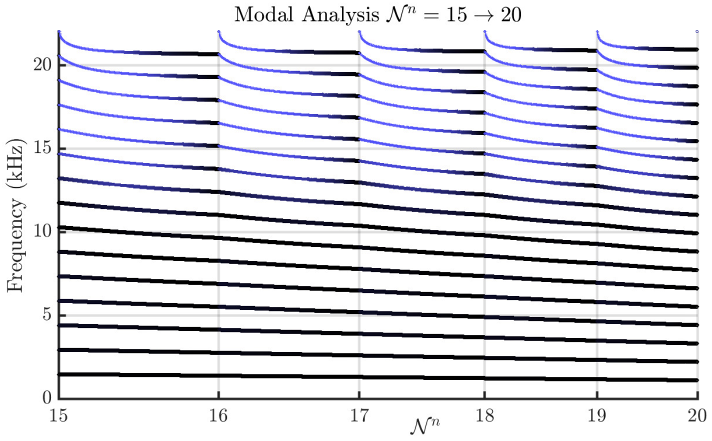
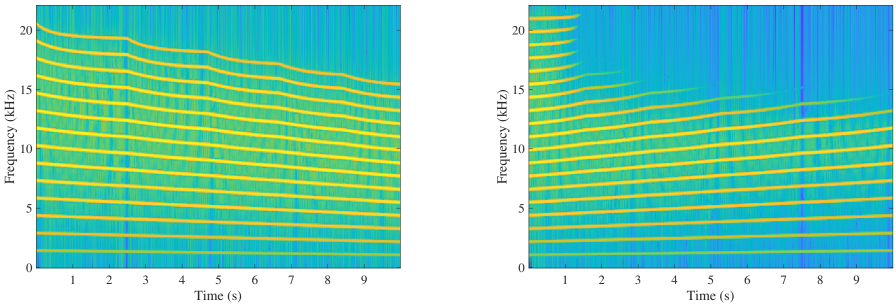


Fig. 12.2: Results of a modal analysis performed on the one-step form of the dynamic grid system in (12.35). Thinner and bluer lines indicate a higher amount of damping.

12.2.6 Modal analysis and results

Although the modal analysis techniques presented in Section 3.5 are only accurate for LTI systems, they can still be applied in the case of slow (sub-audio rate) parameter variations to obtain useful information about the scheme behaviour. In the process of creating the proposed dynamic grid method,



(a) $\mathcal{N}^n = 15 \to 20$, without displacement correction.

(b) $\mathcal{N}^n = 20 \to 15$, with displacement correction.

Fig. 12.3: Output spectra of an implementation of the dynamic grid.

modal analysis was indeed a key component in determining whether the method yielded satisfactory results.

One can perform a modal analysis on the scheme by writing Eq. (12.33) in a one-step form (see Section 3.5.1),

$$\begin{bmatrix} \boldsymbol{\mathcal{U}}^{n+1} \\ \boldsymbol{\mathcal{U}}^{n} \end{bmatrix} = \underbrace{\begin{bmatrix} (\mathbf{A}^n)^{-1}\mathbf{B}^n & (\mathbf{A}^n)^{-1}\mathbf{C}^n \\ \mathbf{I}_{\mathcal{M}^n} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}^n} \begin{bmatrix} \boldsymbol{\mathcal{U}}^{n} \\ \boldsymbol{\mathcal{U}}^{n-1} \end{bmatrix}, \tag{12.35}$$

where $\mathbf{0}$ is a $\mathcal{M}^n \times \mathcal{M}^n$ matrix of zeros. [5]

Figure 12.2 shows the result of a modal analysis performed on the one-step form shown in Eq. (12.35) according to Section 3.5.1. The split of the original system is done as close to the right boundary as possible, such that the number of intervals M^n and M_w^n are calculated using Eq. (12.19). For a simulation lasting n_{end} samples, the wave speed is varied linearly between $c^0 = 2940$, corresponding to $\mathcal{N}^0 = 15$, and $c^{n_{\text{end}}} = 2205$, corresponding to $\mathcal{N}^{n_{\text{end}}} = 20$.

Figure 12.3 shows the output spectra of an implementation of the dynamic grid with the same parameter variation as used for the modal analysis. Figure 12.3a presents the spectral output of an implementation, where $\mathcal{N} = 20 \to 15$ without displacement correction, and shows that the modes follow the pattern predicted by the analysis. Figure 12.3b presents the spectral output where $\mathcal{N} = 15 \to 20$ with displacement correction activated. Although high-frequency modes get damped by the displacement correction, no artefacts are visible when changing grid configurations. Furthermore, the modes follow the pattern predicted by the modal analysis in reverse (as expected).

[5] Notice that, in the definitions for $\mathbf{A}^n$ and $\mathbf{C}^n$, β can go to infinity, when $\alpha = \varepsilon = 0$ through Eq. (12.25). In this case, ε is set to a tiny non-zero value to avoid undefined results.

12.2.7 Discussion and conclusion

Overall, the method satisfies the requirements detailed in the paper. The fundamental frequency of the system follows Eq. (2.40) with only very small deviations, as desired. Although larger deviations occur for higher frequency modes, these are not substantial. Looking at Figure 12.3b, the displacement correction seems to successfully prevent artefacts, but proper listening tests will need to be carried out to confirm this.

The results show that drawbacks of the method, such as frequency deviations and damping due to the displacement correction, mainly happen in higher frequency regions. As these are less relevant than lower frequencies due to the reasons presented in [G], the method can be concluded to work satisfactory. A more detailed discussion of the results can be found in the paper.

One important aspect that is still missing from the presented method, is under what conditions it is stable. As stated in [21], stability for time-varying schemes are difficult to show using the current energy analysis framework (presented in Section 3.4). The first step would be to perform a *frozen coefficient analysis* [59]. In the case of this method, this means to fix c^n at different values and prove stability in those cases. Finding these conditions has been left for future work, but would be a logical next step in the development of the dynamic grid method.

Other future work includes to extend the method to other systems, such as stiff strings (Chapter 4) or even 2D systems such as membranes and plates (Chapter 6). An interesting use case for this method is that of nonlinear systems, such as the Kirchhoff-Carrier string model [111] where parameters are varied based on the state of the system.

12.3 Interpolation experiments

This section presents the results of experiments using different orders of interpolation to calculate the virtual grid points at the inner boundaries of the system, and aims to provide insight as to why quadratic interpolation has been chosen in the end.

Four different Lagrangian interpolators are presented, ranging from linear to quartic (fourth order), and are made using the Lagrange interpolation formula in Eq. (12.12). Different orders of interpolation are included in the method by changing the definitions of the virtual grid points given in Eq. (12.11), and thereby the definition of the $\mathbf{B}^n$ matrix in Eq. (12.32). As the effect of the displacement correction on the modal frequencies is negligible, this is excluded for clarity. Equation (12.33) can then be rewritten to

$$\boldsymbol{\mathcal{U}}^{n+1} = \mathbf{B}^n \boldsymbol{\mathcal{U}}^n - \boldsymbol{\mathcal{U}}^{n-1} \tag{12.36}$$

and, using a test solution $\boldsymbol{\mathcal{U}}^n = z^n \boldsymbol{\phi}$ (following Section 3.5), the eigenfrequencies can be calculated according to (using trigonometric identity (3.21b))

$$f_p^n = \frac{1}{2\pi k}\cos^{-1}\left(\frac{1}{2}\text{eig}_p(\mathbf{B}^n)\right). \tag{12.37}$$

In the following, results will be shown for a varying wave speed corresponding $\mathcal{N}^n = 15 \to 20$, as was done in Section 12.2.6, but using Eq. (12.37). For each interpolator, the case where the original system is split in the middle using Eq. (12.16), and where the split is at the right boundary using Eq. (12.20), are considered. The black and red colours used in the figures do not carry extra meaning, and are only used for clarity of the plots. The results will be discussed at the end of this section.

Linear interpolation

One can implement linear interpolation by changing the definitions in Eq. (12.11) to

$$u_{M^n+1}^n = \alpha w_0^n + (1-\alpha)w_1^n, \tag{12.38a}$$

$$w_{-1}^n = (1-\alpha)u_{M^n-1}^n + \alpha u_{M^n}^n, \tag{12.38b}$$

such that the value of the virtual grid points of one system are fully defined by values of the other.

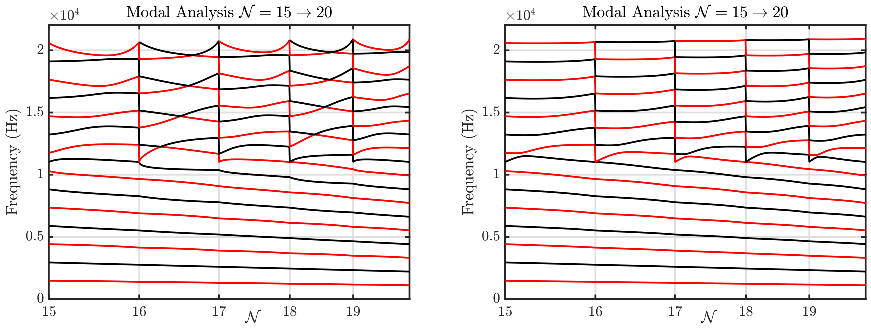
The $\mathbf{B}$ matrix in Eq. (12.32) can be changed to

$$\mathbf{B}^n = \left[\begin{array}{ccc|ccc} \ddots & \ddots & & & & 0 \\ 1 & 0 & 1 & & & \\ & 1 & 0 & \alpha & (1-\alpha) & \\ \hline & (1-\alpha) & \alpha & 0 & 1 & \\ & & & 1 & 0 & 1 \\ 0 & & & & \ddots & \ddots \end{array}\right] \tag{12.39}$$

and results are shown in Figure 12.4.

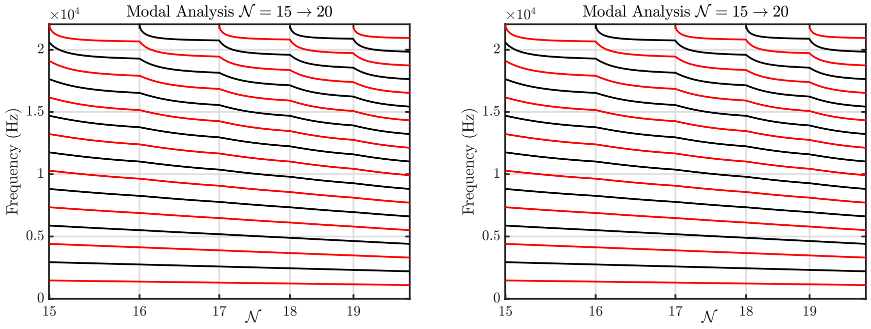
Quadratic interpolation

As quadratic interpolation is what the dynamic grid method is based on, no new definitions for Eqs. (12.11) and (12.32) have to be given. Results of the modal analysis using Eq. (12.37) are shown in Figure 12.5.



(a) Split is in middle. **(b)** Split is close to the boundary.

Fig. 12.4: Results of modal analysis for linear interpolation.



(a) Split is in middle. **(b)** Split is close to the boundary.

Fig. 12.5: Results of modal analysis for quadratic interpolation.

Cubic interpolation

A cubic interpolator, using two points at either side of the virtual grid point, can be created using the Lagrangian interpolation formula in Eq. (12.12). The locations to calculate u_{M^n+1} will be set to

$$\begin{aligned} x_0 &= x_{u_{M^n}} = 0, \\ x_1 &= x_{w_0} = \alpha, \\ x_2 &= x_{w_1} = \alpha + 1, \\ x_3 &= x_{w_2} = \alpha + 2, \end{aligned} \tag{12.40}$$

and the virtual grid point is at

$$x = x_{u_{M^n+1}} = 1. \tag{12.41}$$

The definitions for the virtual grid points can then be shown to be

$$u^n_{M^n+1} = \frac{\alpha-1}{\alpha+2}u^n_{M^n} + \frac{\alpha+1}{2}w^n_0 + (1-\alpha)w^n_1 + \frac{\alpha(\alpha-1)}{2(\alpha+2)}w^n_2, \tag{12.42a}$$

$$w^n_{-1} = \frac{\alpha(\alpha-1)}{2(\alpha+2)}u^n_{M^n-2} + (1-\alpha)u^n_{M^n-1} + \frac{\alpha+1}{2}u^n_{M^n} + \frac{\alpha-1}{\alpha+2}w^n_0, \tag{12.42b}$$

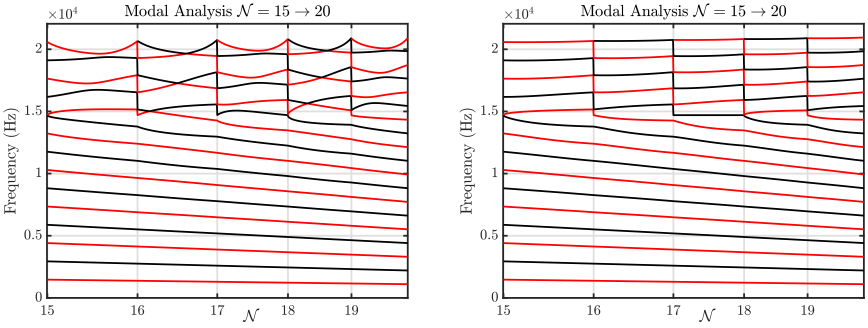
and the $\mathbf{B}^n$ matrix can be set accordingly.

If Eq. (12.19) is used, and the split is placed close to the right boundary, the Dirichlet condition at this boundary needs to be extended to be simply supported, such that $w^n_2 = -w^n_0$. As the value at the boundary remains 0, i.e., $w_1 = 0$, this alters Eqs. (12.42) to be

$$u^n_{M^n+1} = \frac{\alpha-1}{\alpha+2}u^n_{M^n} + \left(\frac{\alpha+1}{2} - \frac{\alpha(\alpha-1)}{2(\alpha+2)}\right)w^n_0$$

$$w^n_{-1} = \frac{\alpha(\alpha-1)}{2(\alpha+2)}u^n_{M^n-2} + (1-\alpha)u^n_{M^n-1} + \frac{\alpha+1}{2}u^n_{M^n}.$$

The results are shown in Figure 12.6.



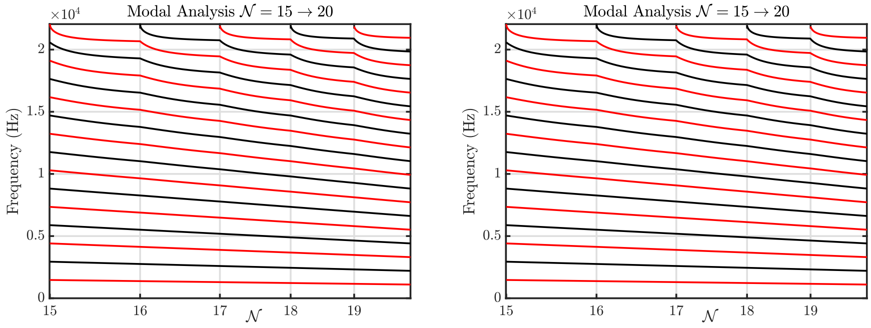
(a) Split is in middle.

(b) Split is close to the boundary.

Fig. 12.6: Results of modal analysis for cubic interpolation.

Quartic interpolation

Finally, a quartic interpolator can be created from Eq. (12.12). At the boundary, it can be implemented similar to the cubic interpolator. Results of a quartic interpolator can be found in Figure 12.7.



(a) Split is in middle.

(b) Split is close to the boundary.

Fig. 12.7: Results of modal analysis for quartic (fourth-order) interpolation.

12.3.1 Discussion

The results show a large behavioural difference between odd-ordered (linear, cubic) interpolators and even-ordered (quadratic, quartic) interpolators.

Figure 12.4 shows that for linear interpolation, modes with frequencies below $f_s/4$ Hz ($1/2$ the Nyquist frequency) move downwards as $\mathcal{N}^n$ increases, as expected, but modes above this frequency generally move upwards. If the split is in the middle, 'modal crossings' appear for even values of $\lfloor \mathcal{N}^n \rfloor$, but they do not occur when the split is close the boundary. Similar behaviour occurs for cubic interpolation in Figure 12.6, but the split happens around $3/8 f_s$ Hz ($3/4$ the Nyquist frequency). Again, adding points close to the boundary resolves the modal crossings.

The modal patterns created by even-ordered interpolators, exhibit behaviour closer to what is desired (see Figures 12.5 and 12.7). All modal frequencies move down and no modal crossings occur. Interestingly, the even-ordered interpolators show no differences in their modal behaviour when the split is in the middle of the original system versus close to the boundary (the differences are within machine precision).

Frequency deviations happen to a lesser degree for the quartic interpolator than for the quadratic interpolator, and continue to decrease for higher-ordered (even) interpolators. The differences have been found negligible and as flexibility of the method decreases as the interpolation order increases (as more grid points are required for the virtual grid point calculation), the quadratic interpolator has been chosen in the eventual method.

12.4 Examples of use cases

This section provides several examples of instruments using dynamic parameter variations and can be used as inspiration for future work. As mentioned in paper [G], the interest lies in dynamic changes in the defining parameters of the resonator, as opposed to a pitch change using a fretting finger, for example.

1D systems

A defining property that can physically be changed in strings is the tension. There are various ways to smoothly change the pitch by changing the tension in the string. A straightforward way to do this would be to move the fretting finger perpendicular to the string while pressing down to create a pitch bend. Other, less common ways are to modulate the tension by pressing down on the small length of string between the tuning peg and the nut[6], or turn the tuning pegs directly while playing.[7,8]

The hammered dulcimer is another example where the strings are placed 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.[9]

Apart from strings, acoustic tubes also lend themselves to applications of the dynamic grid. The main example is the trombone as published in [H], where the method presented in this chapter is used. The slide whistle falls in the same category.

2D systems

One could potentially extend the dynamic grid method presented here to 2D, and model physical tension changes in membranes. The membrane tension in timpani, for example, can be changed using a footpedal. The Bodhrán is a membranophone where the player hits the membrane on one side and can change the pitch by pressing on the other side.[10] The pitch of a talking drum (hourglass drum) can be changed be squeezing the laces attached to the membrane.[11] An example of a pitch-modulated thin plate is the musical saw, where the curvature of the saw is changed to create different pitches.[12]

[6] John Mayer - Gravity (Live in L.A.): `https://youtu.be/dBFW8OvciIU?t=284`

[7] Jon Gomm - Passionflower: `https://youtu.be/nY7GnAq6Znw?t=49`

[8] gr4yhound - servo bender: `https://youtu.be/fSQ9Dg65EFo`

[9] Amazing Hammered Dulcimer Musician - Joshua Messick: `https://youtu.be/veuGTnzgNRU?t=215`

[10] John Joe Kelly Bodhran Solo - Christ church Dublin 2012: `https://youtu.be/b9HyB5yNS1A?t=146`

[11] Ayan Bisi Adeleke - Master talking drummer - drum talks: `https://youtu.be/B4oQJZ2TEVI?t=9`

[12] Musical Saw performance by Sakita Hajime: `https://youtu.be/-6nv0iDrAis`

Chapter 13

Real-Time Implementation and Control

> *"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."*
> *- Donald E. Knuth*

A large contribution of this PhD project has been to implement novel combinations of existing FD schemes in real time. As opposed to many FDTD-based musical instruments found in the literature, those presented in this work allow for real-time control such that the virtual instrument can be played. 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 [112] place the upper limit of latency when interacting with computers for musical purposes at 10 ms. Moreover, they place the limit on the *jitter*, the variation of the latency, at 1 ms. It is thus important to keep the CPU usage at a fixed level as much as possible, and different ways of controlling and interacting with the application should not influence the number of computations much.

Until now, this thesis has made several references to implementations of FD schemes in the `MATLAB` programming language. Although `MATLAB` is a great tool for prototyping, it is a *high-level* programming language (i.e., has a high level of abstraction). Generally, higher-level programming languages are easier to program, but more control – and speed – is gained by using a lower-level programming language such as C++.

A great tool for real-time audio programming, which has extensively been used in this project, is the JUCE Framework.[1] This framework, written in C++, is specifically developed for programming audio applications and plug-ins. The framework provides the back end of an audio application that handles the audio and graphics threads, and ensures that they can run simultaneously with minimal interference. All real-time physical models that were made over the course of this PhD project have been implemented using the JUCE framework.

This chapter can be considered as a general contribution that can be applied to all papers in Part VII (except paper [G]). Firstly, details on the structure of a class implementing a FD scheme will be provided, using the damped stiff string as an example. Secondly, an overall code structure is given that can be applied to many of the applications created during this PhD project. Finally, this chapter will present the Sensel Morph and the PHANTOM Omni, two hardware devices which have been used to control the physical models presented in papers [A], [B], [C], [D] and [E]. Some additional considerations on programming real-time FD schemes can be found in Appendix E.

13.1 Real-time FD schemes

Until now, this thesis has presented many resonators in matrix form (see e.g. Eqs. (4.19) and (6.46)) for a compact implementation in `MATLAB`. Although libraries for handling matrices in C++ exist (see e.g. *Eigen* [113]), the highest algorithm speed is obtained by calculating the update equations directly.

In any of the real-time applications created during this project, the update equation implementing a FD scheme is always the most computationally expensive part (given a low refresh-rate for the graphics). This algorithm needs to run at a rate of 44100 Hz, whereas the rest of the implementation can run at a much lower rate. This section provides details and considerations on the real-time implementation of FD schemes in C++ during this project. The damped stiff string presented in Chapter 4 will be used an example. The full implementation can be found online and parts will be presented here to aid the explanation.[2] The FD scheme is implemented in a separate class called `SimpleString`, and will be used in the following.

13.1.1 System states and pointer switches

In any FD scheme implementation, at the end of every iteration, the system states must be updated, i.e., the following operations must be performed:

$$\mathbf{u}^{n-1} := \mathbf{u}^n \quad \text{and} \quad \mathbf{u}^n := \mathbf{u}^{n+1}.$$

[1] `https://juce.com/`

[2] `https://github.com/SilvinWillemsen/SimpleStringApp/` (using JUCE v6.0.8)

In MATLAB, one would simply perform these operations according to

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

In C++, however, one has the ability to perform a *pointer switch* to update the system states. For a 1D FD scheme with $N+1$ grid points, the number of copy-operations it takes to update the system states manually would be $2(N+1)$, as shown in Figure 13.1a. A pointer switch, as shown in Figure 13.1b, only needs 4 copy-operations per iteration and can be carried out in C++ as follows:

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

Here, u is a vector containing 3 pointers, each of which points a state vector at a certain time step:

$$\texttt{u[0]} \rightarrow \mathbf{u}^{n+1}, \quad \texttt{u[1]} \rightarrow \mathbf{u}^{n}, \quad \text{and} \quad \texttt{u[2]} \rightarrow \mathbf{u}^{n-1}.$$

A temporary pointer is assigned to the memory location that u[2] points at, to be able to assign that location in memory to u[0] in the end. The values of that vector will be overwritten by the update equation in the next iteration (see Section 13.1.3 and Figure 13.1).

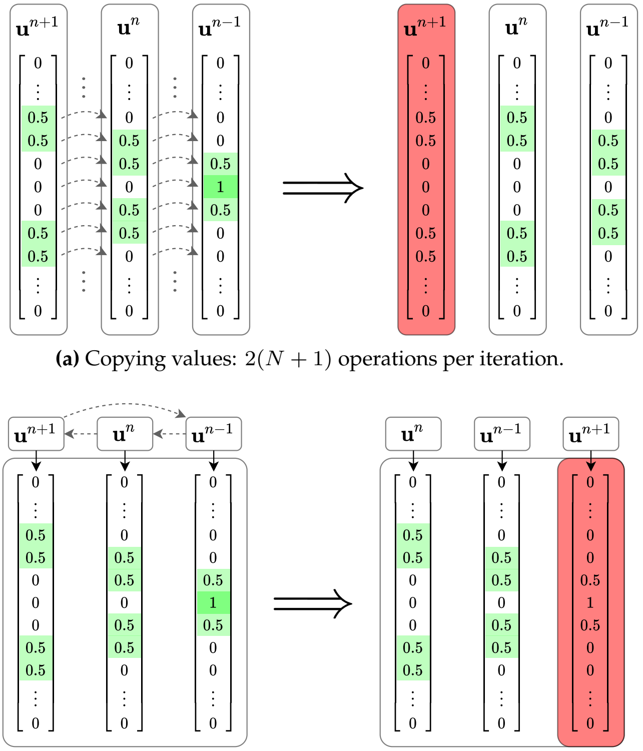
The state vectors themselves are stored in a matrix (which is a 'vector of vectors' in C++). This matrix will have 3 columns related to the 3 time steps required to calculate the FD scheme, and $N+1$ rows, which corresponds to the number of grid points.[3] The matrix is initialised as follows

```
//// In the constructor of SimpleString ////

// initialise vectors
uStates = std::vector<std::vector<double>> (3,
                                        std::vector<double>(N+1, 0));
```

Next, the aforementioned pointers are initialised such that they contain the memory addresses of the start of the three state vectors in the matrix.

[3] These are not actual rows and columns as in a matrix, but these terms are used here for ease of explanation.



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

(b) Pointer switch: 4 operations per iteration.

Fig. 13.1: 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.

```
//// In the constructor of SimpleString ////

// Initialise pointer vector
u.resize (3, nullptr);

// Make set memory addresses to first index of the state vectors.
for (int i = 0; i < 3; ++i)
    u[i] = &uStates[i][0];
```

One will then be able to work with the pointers directly in the eventual update equation (see Section 13.1.3).

13.1.2 Pre-calculation of coefficients

To prevent extra computations in the FD scheme, it is useful to calculate as many of the coefficients as possible beforehand (provided that these do not vary over time). Recalling the update equation for a stiff string in Eq. (4.10), one can write this as

$$\begin{aligned}A_{\text{div}}u_l^{n+1} &= B_1 u_l^n + B_2(u_{l+1}^n + u_{l-1}^n) + B_3(u_{l+2}^n + u_{l-2}^n)\\ &\quad + C_1 u_l^{n-1} + C_2(u_{l+1}^{n-1} + u_{l-1}^{n-1}),\end{aligned} \tag{13.1}$$

where

$$A_{\text{div}} = 1 + \sigma_0 k, \quad B_0 = 2 - 2\lambda^2 - 6\mu^2 - \frac{4\sigma_1 k}{h^2}, \quad B_1 = \lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2},$$
$$B_2 = -\mu^2, \quad C_0 = -1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}, \quad \text{and} \quad C_1 = -\frac{2\sigma_1 k}{h^2}.$$

All of these coefficients can be calculated in the constructor of the stiff string class. One can also already divide the B and C coefficients by A_{div} as done in the code below.

```
//// In the constructor of SimpleString ////

// Coefficients used for damping
S0 = sigma0 * k;
S1 = (2.0 * sigma1 * k) / (h * h);

// Scheme coefficients
B0 = 2.0 - 2.0 * lambdaSq - 6.0 * muSq - 2.0 * S1; // u_l^n
B1 = lambdaSq + 4.0 * muSq + S1;                   // u_{l+-1}^n
B2 = -muSq;                                        // u_{l+-2}^n
C0 = -1.0 + S1 + 2.0 * S2;                         // u_l^{n-1}
C1 = -S1;                                          // u_{l+-1}^{n-1}

Adiv = 1.0 / (1.0 + S0);                           // u_l^{n+1}

// Divide by u_l^{n+1} term
B0 *= Adiv;
B1 *= Adiv;
B2 *= Adiv;
C0 *= Adiv;
C1 *= Adiv;
```

13.1.3 Update equation

With all the above set up, the update equation can be implemented as follows:

```
void SimpleString::calculateScheme()
{
    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] + C2 * (u[2][l + 1] + u[2][l - 1]);
}
```

This function and the pointer switch in Section 13.1.1 (in that order) will then have to be called once per sample.

13.1.4 Acceleration strategies

Apart from implementing the physical models in C++ rather than MATLAB, additional acceleration strategies can be used for even faster algorithms. FD schemes, especially explicit schemes, are highly parallelisable, i.e., many of the operations done are identical and can run simultaneously. A great overview is given in [43] and provides advantages and disadvantages of using the GPU, multicore processing and vector instructions (SIMD, AVX). An in-depth evaluation of FD schemes (both 1D and 2D), implemented using multicore processing and AVX instructions, has been carried out in [64].

For this project, none of these acceleration strategies have been used, but could be investigated in the future. Occasionally, the quality of 2D systems has been lowered to allow for a real-time implementation (see papers [A] and [D]). However, as strings were used as the main resonators of many of the implementations, these decreases in quality were deemed unimportant for the eventual output sound of the simulation.

13.2 Code structure

This section presents the general code structure used for the real-time applications created in this project. As an example, consider a simple instrument consisting of one string and one plate, and a connection between them as presented in Section 11.5. The string is excited using the Sensel Morph (see Section 13.3.1). This is a simplified case of the contribution made in papers [A] and [B].

The structure of the code is visualised in Figure 13.2. The white boxes denote various classes or components of the application, which will be described in detail shortly. The black arrows indicate instructions, and hollow arrows indicate data flows. All arrows are accompanied by a box denoting the type

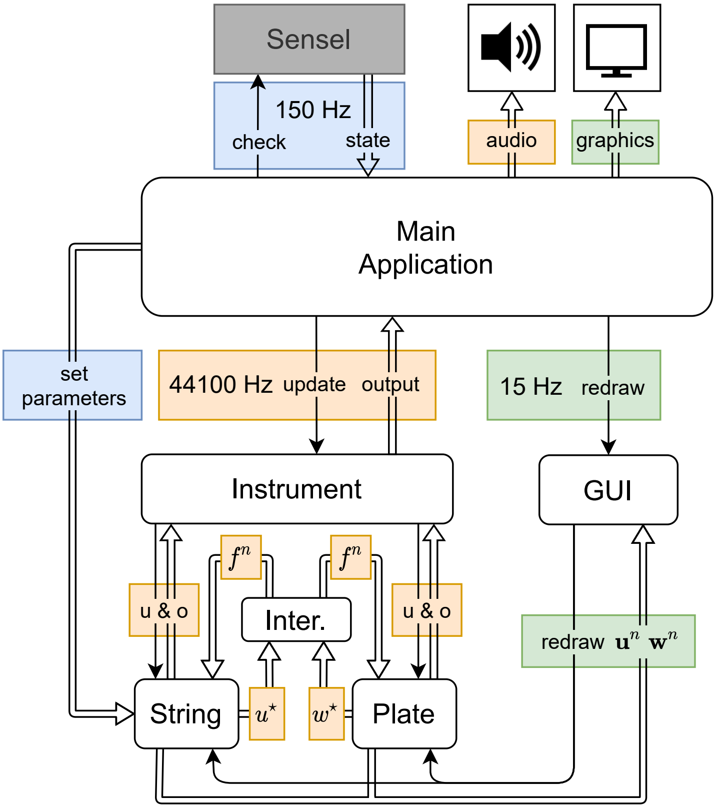


Fig. 13.2: The structure of a real-time implementation of a physical model. Boxes with 'u & o' refer to 'update and output' and 'Inter.' is short for 'Interaction'. A detailed description of the figure is given in Section 13.2.

of instruction / dataflow and the colour of the box denotes at what rate this happens.

Threads

The application contains three different threads, all handled by the JUCE back end. The highest priority thread is the audio thread, which is denoted by orange blocks. It runs at 44100 Hz and handles the calculations of the FD schemes. Denoted by blue, is the control thread, which runs at 150 Hz. The input from the Sensel will be applied to the application at this rate. This rate corresponds to a maximum control latency of ~7 ms, which is below the upper limit for latency in musical applications proposed in [112]. Finally, the thread updating the graphical user interface (GUI) is set to run at 15 Hz, which is a value that was heuristically found to be a good balance between good visuals and a fast application. Papers [A], [C], and [H], contain a comparison between the speed of the application with and without the graphics. In all cases, results showed that the graphics take up a large part of the computational power available.

String and Plate

The String and Plate classes implement the FD schemes of the stiff string and thin plate resonators respectively.[4] See Section 13.1 for an example of an implementation of the stiff string; a similar code structure can be used for the Plate class. As this section follows Section 11.5, the states of the string and the plate will be denoted by u and w, respectively.

For the resonators to work in isolation, both classes require a function that calculates the scheme, and a function that updates their system states (see Section 13.1). The interactions between the classes is handled by the Instrument class (see below). Therefore, both classes require additional functions that return $u^\star$ and $w^\star$, as well as a function that adds the interaction force f^n to the schemes. A final function returns the output of the resonators.

Lastly, the classes contain a `paint` function which is used to draw the state of the system on the screen. This function is called by the JUCE back end at the rate that the graphics thread is set to.

Instrument

The Instrument class contains instances of the individual String and Plate resonators. Rather than performing the calculations of the FD schemes, the Instrument class handles all interactions between the individual resonators. In Figure 13.2, this is denoted by the 'Inter.' block and it follows a similar process to Section 11.5.4 (albeit not in matrix-vector form):

1. The Instrument class instructs the resonators to calculate their respective schemes without the connection force (Eq. (11.52)).

2. The intermediate states at the connection location, $u^\star$ and $w^\star$, are retrieved (Eq. (11.53)).

3. The connection force f^n is calculated using Eq. (11.51).

4. The force is added to the scheme (Eq. (11.54)).

[4] Note that the class can not actually be called 'String' as this is already an existing variable type.

Main Application and Control

The Main Application (also called `MainComponent` in JUCE), is the top-level class of the application that handles control input and audio and graphics output.[5] The Main Application contains an instance of the Instrument class and instructs it to calculate the schemes and their interactions once per sample. Furthermore, it retrieves the output from the Instrument and sends this to the audio device at the same rate.

Finally, the application is controlled by the Sensel Morph. The Main Application checks the state of the Sensel at a rate of 150 Hz and maps this to control parameters used by the scheme.

13.3 Hardware devices

Throughout this project, two hardware devices to expressively control the simulated instruments have been investigated. These are the Sensel Morph and the PHANTOM Omni, both of which will be briefly described here. The mapping of these controllers to the various instrument simulations can be found in the respective papers described below.

13.3.1 Sensel Morph

The *Sensel Morph,* or Sensel for short, is a high-accuracy pressure sensitive touch controller, containing ~20,000 pressure-sensitive sensors that allow for high-fidelity control (see Figure 13.3) [47]. Above the touch-sensitive area, the controller contains an array of 24 LEDs that can be programmed and used to provide information to the user.

Papers [A] and [B] were the first scientific papers to use the Sensel to control a musical instrument simulation. Afterwards, the controller was used for other applications in the Sound and Music Computing field [114, 115, 116].

For this project, the Sensel has mainly been used to control the bow to excite stiff strings in papers [A], [B], [C] and [D]. Additionally, it is used to excite strings using simple pluck and hammer excitations as described in Section 7.2. Details on the mapping of the Sensel to the various implementations can be found in the respective papers.

13.3.2 PHANTOM Omni

The PHANTOM Omni, or Omni for short, by SenseAble Technologies (now 3D Systems), is a six-degrees-of-freedom (6-DoF) device that provides force and vibrotactile feedback (see Figure 13.4) [117]. Moreover, it allows for highly

[5] This section assumes that the JUCE Audio Application template has been chosen – not the Audio Plug-in template.

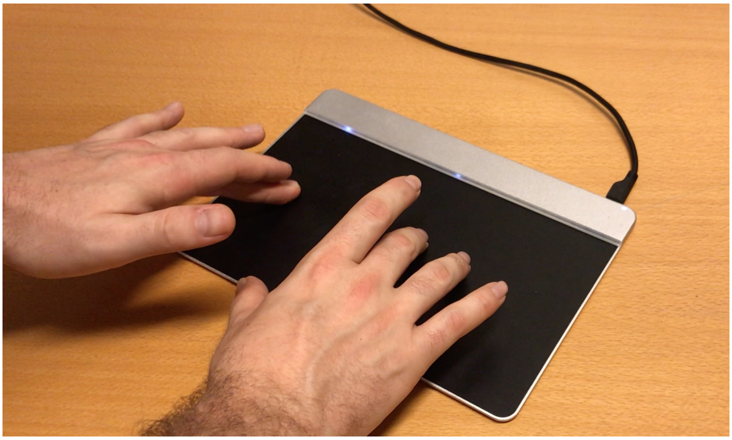


Fig. 13.3: The Sensel Morph.

accurate tracking in a 3D application, and has been used in paper [E] to control the bow of the tromba marina.

Other work using the Omni in a musical context, specifically for plucking a virtual guitar string, was done by Passalenti et al. in [118, 119] and Fontana et al. in [120].

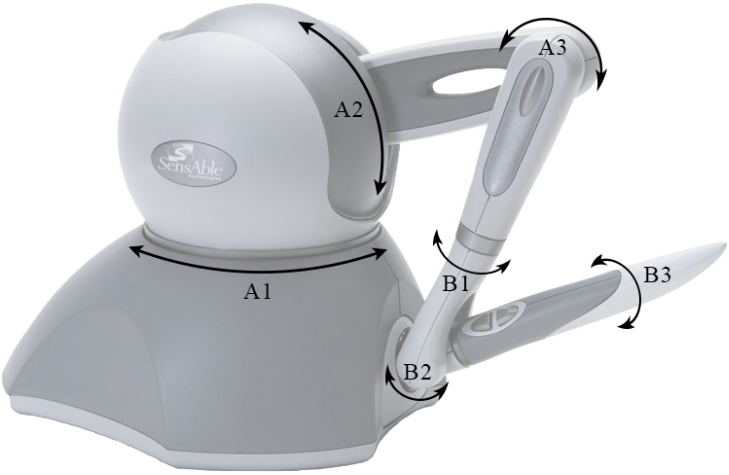


Fig. 13.4: The PHANTOM Omni haptic device. The device has six axes of rotation (6-DoF), three of which provide force feedback (A1-3), and three that only track position (B1-3). (Adapted from paper [E].)

Chapter 14

Large Scale Modular Physical Models

This chapter provides an extended summary for the work presented in the papers "Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph" [A] and "Physical Models and Real-Time Control with the Sensel Morph" [B]. Paper [A] presents the work done on various physical models connected by nonlinear springs using three instruments as case studies: the esraj (bowed sitar), the hammered dulcimer and the hurdy gurdy. The implementations and a video showcasing the hurdy gurdy can be found online.[1,2] Paper [A] follows [21] and [54] and uses 'scaling' (see Section 2.4.1). To relate the paper to the theory presented in this thesis, this chapter presents the models in a non-scaled, dimensional form. The eventual implementation of the models is equivalent. Then, a summary of the remaining parts of the paper is provided, including descriptions of the instruments.

14.1 Physical models

All instruments use multiple instances of the stiff string presented in Chapter 4 and one instance of the thin plate presented in Section 6.3. The latter was used as a simplified instrument body for the resulting simulations (see Section 14.2.1). Theory on connections can be found in Chapter 11, and information on the string-plate connection specifically, is presented in Section 11.5.

Consider a set of strings, where the transverse displacement of string s is described by $u_s = u_s(\chi_s, t)$ (in m) defined for $t \geq 0$ and $\chi_s \in \mathcal{D}_s$ for domain $\mathcal{D}_s = [0, L_s]$ and length L_s (in m). Notice that every string is defined for a

[1] `https://github.com/SMC-AAU-CPH/ConnectedElements/releases/tag/v5.0`

[2] `https://youtu.be/BkxLji2ap1w`

separate coordinate system χ_s. In the following, spatial derivatives ∂_{χ_s} are the same as those described in Section 2.2.2, but with respect to coordinate χ_s (similar to Chapter 11). The PDE of string s with an external connection force is defined as

$$\partial_t^2 u_s = c_s^2 \partial_{\chi_s}^2 u_s - \kappa_s^2 \partial_{\chi_s}^4 u_s - 2\sigma_{0,s}\partial_t u_s + 2\sigma_{1,s}\partial_{\chi_s}^2 u_s - \delta(\chi_s - \chi_{s,\text{c}})\frac{f_s}{\rho_s A_s}, \quad (14.1)$$

where spatial Dirac delta function $\delta(\chi_s - \chi_{s,\text{c}})$ (in m^{-1}) localises the connection force between the string s and the plate to connection location $\chi_{s,\text{c}}$. Other parameters are as defined in Eq. (4.3) but have a subscript s to denote that they can be different for each string.

As all connections in the implementation are between an individual string and the plate, the PDE of the thin plate in Eq. (6.37) can be extended to

$$\partial_t^2 w = -\kappa_\text{p}^2 \Delta\Delta w - 2\sigma_{0,\text{p}}\partial_t w + 2\sigma_{1,\text{p}}\partial_t\partial_x^2 w + \sum_s \delta(x - x_{\text{c},s}, y - y_{\text{c},s})\frac{f_s}{\rho_\text{p} H}, \quad (14.2)$$

where 2D spatial Dirac delta function $\delta(x_s - x_{s,\text{c}}, y_s - y_{s,\text{c}})$ (in m^{-2}) localises the connection force of between the plate and string s to coordinate (x_s, y_s) on the plate. Other parameters are as defined in Eq. (6.37).

Finally, the connection force between the plate and string s is defined as a nonlinear damped spring (see Eq. (11.41))

$$f_s = K_1\eta_s + K_3\eta_s^3 + R\dot{\eta}_s, \quad (14.3)$$

where

$$\eta_s = u_s(\chi_{\text{c},s}, t) - w(x_{\text{c},s}, y_{\text{c},s}, t) \quad (14.4)$$

is the relative displacement between string s and the plate at their respective connection locations. Notice that the plate is placed below the strings such that the sign of the force term is negative for the strings and positive for the plate.

14.1.1 Implementation

Here, some considerations for implementing the above models are provided. Details on discretisation of the models and how to solve for the connection forces f_s, are presented in Section 11.5 and are not given here.

The spatial Dirac delta functions are discretised using 0th-order spreading operators for simplicity (see Section 8.2 (1D) and Section 11.1 (2D)). Furthermore, the connection locations on the plate are implemented to be non-overlapping. Overlaps would require to solve a system of linear equations to obtain the connection forces (see e.g. [54]). Looking towards real-time implementation, an explicit solution for each connection is desired.

14.2 Summary

This section provides a summary of the instrument simulations presented in paper [A]. All instruments were implemented in real-time in C++ using the JUCE framework (see Chapter 13). Finally, a summary of the results and the conclusion will be given.

14.2.1 Instruments

Using the setup presented in Section 14.1, various configurations inspired by real instruments were made. The choices of simulated instruments were aimed at those containing many (sympathetic) strings.[3] Another condition was that no FDTD-based physical models of these instruments existed in the literature at the time of writing the papers.

Three implementations inspired by real-life instruments were created and their setups are presented here. The implementations were controlled by a pair of Sensel Morph controllers (see Section 13.3.1). The mapping between the controllers and the instruments is explained in papers [A] and [B].

Esraj: bowed sitar

The first instrument simulation was inspired by the *esraj*: the bowed sitar. This instrument uses many strings, some of which are bowed and others are sympathetic strings that resonate when the instrument is played. As one can also interact with the latter, several strings in the implementation could be plucked as well.

In total, 20 strings were implemented, all connected to a thin plate: 2 strings could be bowed, 5 strings could be plucked, and 13 strings are sympathetic. The bow was implemented using the static friction model presented in Section 8.4 and the pluck was modelled as a time-varying raised cosine found in Section 7.2.1.

Hammered dulcimer

The hammered dulcimer, or santur, can be seen as an 'open piano' where the player hammers several strings at once. In the implementation, 20 pairs of strings were implemented, and one in each pair was connected to the plate. This caused a slight detuning between the strings, resulting in a characteristic 'chorus' effect exhibited by the instrument. To excite the strings, the time-varying strike presented in Section 7.2.1 was used.

[3] Sympathetic strings – apart from being friendly – are strings that add resonances to the instrument without being excited directly.

Hurdy gurdy

The hurdy gurdy is a bowed string instrument, that also uses sympathetic strings. Rather than a bow, the instrument uses a rosined wheel attached to a crank that bows the strings as it is turned. As for the esraj, the static friction model presented in Section 8.4 was used to implement the wheel.

The instrument simulation consisted of 5 bowed strings and 13 sympathetic strings, all connected to a plate.

14.2.2 Results and conclusion

All instrument simulations were able to run in real time on a MacBook Pro with a 2.2 GHz Intel i7 processor. Interaction with the implementations shows that when exciting one string, the connections with the plate cause other (sympathetic) strings to vibrate as well. Specifically, strings tuned to one of the harmonic partials of the excited string were found to resonate to a high degree. This phenomenon is consistent to real-world processes.

Finally, informal evaluations of the instruments were carried out on experts in the sound and music computing field, and showed that the mapping between the Sensels and the instruments, specifically the bowing interaction, was considered to be intuitive.

Chapter 15

The Tromba Marina

This chapter provides an extended summary for the paper "Real-time Implementation of a Physical Model of the Tromba Marina" [D] and the paper "Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling" [E]. After a general summary of these papers, the physical model will be summarised, with reference to the theory explained in the previous parts of this thesis. Finally, this chapter extends the contents of paper [D] by providing more details on the implementation and provides an energy analysis.

15.1 Summary

The tromba marina is a bowed monochord instrument from medieval Europe (see Figure 1 in paper [D]). 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 (see Figure 2 in paper [D]). 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.

Real-time implementation and control

Paper [D] presents a physical model of the tromba marina and its real-time implementation in C++ using the JUCE framework (see Chapter 13). The application and a video showcasing it can be found online.[1],[2] The paper uses

[1] https://github.com/SilvinWillemsen/TrombaMarina/releases/tag/2.0

[2] https://youtu.be/x72Xh-nUoVc

the Sensel morph for control of the algorithm. See Section 13.3.1 for more information on this controller. To approach the interaction paradigm of the real-life instrument, a virtual reality (VR) application was made from where the algorithm could be controlled. To this end, the PHANTOM Omni haptic device was used to control the implementation and presented in paper [E]. See Section 13.3.2 for more details on this device. A demo of the VR application can be found online.[3] The VR implementation has been evaluated and the results will be summarised below.

Evaluation and results

The VR implementation of the tromba marina was evaluated on 14 participants. The goals of the evaluation was to determine the general experience of bowing in a VR environment, and to evaluate the playability of a VR monochord instrument. The data was collected using a combination of qualitative and quantitative methods as detailed in paper [E]. The questions asked in the quantitative part of the evaluation were divided into haptics, visuals, audio, and overall experience of the application.

The results show that the various modalities (visuals, audio and haptics) seemed to reinforce each other and that the haptic feedback provided by the PHANTOM Omni was deemed "realistic" and "natural". Many considered the instrument "hard to play", but this is what could be expected when one starts. The overall playability of the instrument was thus concluded to be satisfactory, though improvements could be made.

15.2 Physical model

This section presents the physical model of the tromba marina with reference to the theory presented in Parts II, III and IV. As much as possible, this chapter follows the notation of paper [D], as long as it is coherent with what has already been presented in this thesis. Any discrepancies between this chapter and the paper will be clearly highlighted.

15.2.1 Continuous time

The full musical instrument has been divided into three parts: the string, the bridge and the body, each of which will briefly be presented here. Each resonator in isolation is of the form

$$\mathcal{L}q = 0 \tag{15.1}$$

[3] https://www.youtube.com/watch?v=SlHqvaaPCyU

where $\mathcal{L}$ is a linear (partial) differential operator and $q(\boldsymbol{x}, t)$ describes the state of the component in isolation. q is defined for time $t \geq 0$ and spatial coordinate $\boldsymbol{x} \in \mathcal{D}$ where the dimensions of domain $\mathcal{D}$ depend on the dimensions of the system at hand. Using the form in Eq. (15.1) allows for a much more compact notation later on. In the following, subscripts 's', 'm' and 'p' will be used to denote the 'string', 'bridge' (mass), and 'body' (plate) respectively.

String

The string of the tromba marina is modelled as a damped stiff string of length L (in m) (see Chapter 4). With reference to the form in (15.1), its transverse displacement is described by $q = u(\chi, t)$ (in m) and is defined for $\chi \in \mathcal{D}_\text{s}$ where domain $\mathcal{D}_\text{s} = [0, L]$.[4] Using partial differential operator $\mathcal{L} = \mathcal{L}_\text{s}$, the PDE describing its motion is defined as

$$\mathcal{L}_\text{s} u = 0, \tag{15.2}$$

where (see Eq. (4.3))

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

The parameters are as defined for Eq. (4.3) with the possible addition of the 's' subscript. Compared to the schemes presented in previous chapters, using a partial differential operator to combine all operators like this, is solely different from a notational point of view, and does not change the behaviour of the system. If Eq. (15.2) is expanded and all terms except for $\rho_\text{s} A \partial_t^2 u$ are moved to the right-hand side, one arrives at the damped stiff string PDE in Eq. (4.3) again.

As the string is bowed, one can extend the PDE in (15.2) to

$$\mathcal{L}_\text{s} u = -\delta(\chi - \chi_\text{B}) F_\text{B} \Phi(v_\text{rel}), \tag{15.3}$$

with bow position $\chi_\text{B} = \chi_\text{B}(t) \in \mathcal{D}_\text{s}$ (in m), bow force $F_\text{B} = F_\text{B}(t)$ and relative velocity between the bow and the string $v_\text{rel} = v_\text{rel}(t)$. The static friction model in Eq. (8.16) has been chosen for simplicity. For more details on this bow model, see Section 8.4.

Bridge

The bridge is modelled using a mass-spring-damper system (see Section 9.1). With reference to Eq. (15.1), the transverse displacement of the mass is described by $q = w(t)$ (in m) and using differential operator $\mathcal{L} = \mathcal{L}_\text{m}$, its PDE

[4] Notice that χ rather than x is used here. As x will be used as a spatial coordinate for the body later on, a different symbol is used for the string to differentiate between two coordinate-systems.

is

$$\mathcal{L}_{\text{m}} w = 0, \tag{15.4}$$

where (see Eq. (9.1))

$$\mathcal{L}_{\text{m}} = M\frac{d^2}{dt^2} + M\omega_0^2 + M\sigma_{\text{m}}\frac{d}{dt}.$$

Here, $\omega_0 = \sqrt{K/M}$ (in s^{-1}) and $\sigma_{\text{m}} = R/M$ (in s^{-1}) and other parameters are as in Eq. (9.1).[5]

Notice that when using a differential operator for an ODE, the dots to denote a temporal derivative (as e.g. Eq. (9.1)) need to be written in an operator form. Here, Leibniz's notation is chosen (see Section 2.1).

Body

The body of the instrument is simplified to a damped rectangular thin plate of side lengths L_x and L_y (in m) (see Section 6.3). Again, with reference to Eq. (15.1), its transverse displacement is described by $q = z(x, y, t)$ (in m) and is defined for $(x, y) \in \mathcal{D}_{\text{p}}$ where domain $\mathcal{D}_{\text{p}} = [0, L_x] \times [0, L_y]$. Using partial differential operator $\mathcal{L} = \mathcal{L}_{\text{p}}$, the PDE describing the motion of the body is

$$\mathcal{L}_{\text{p}} z = 0, \tag{15.5}$$

where (see Eq. (6.37))

$$\mathcal{L}_{\text{p}} = \rho_{\text{p}} H \partial_t^2 + D\Delta\Delta + 2\rho_{\text{p}} H \sigma_{0,\text{p}} \partial_t - 2\rho_{\text{p}} H \sigma_{1,\text{p}} \partial_t \Delta,$$

with $D = E_{\text{p}} H^3/12(1-\nu^2)$. Again, parameters are as defined in Eq. (6.37).

Interactions

The three components interact using the non-iterative collision methods presented in Chapter 10. Recalling the importance of the relative location of the various objects (see Section 10.1), the string is placed above the bridge, which is placed above the body. This arrangement will, in turn, determine the definitions of η and the directions of the forces in the eventual system. In the following, subscripts 'sm' and 'mp' are used to denote a 'string-mass' and 'mass-plate' interaction respectively.

The bridge-body interaction is modelled using the collision-potential in Eq. (10.1):

$$\phi(\eta) = \frac{K_{\text{mp}}}{\alpha_{\text{mp}} + 1} [\eta_{\text{mp}}]_+^{\alpha_{\text{mp}}+1}, \tag{15.6}$$

[5] In paper [D], the symbol R is used for the damping coefficient, but for coherence in this work, σ_{m} is used instead.

where $\eta_{\text{mp}} = \eta_{\text{mp}}(t) = w(t) - z(x_{\text{mp}}, y_{\text{mp}}, t)$ and the location on the plate where the bridge collides is $(x_{\text{mp}}, y_{\text{mp}}) \in \mathcal{D}_{\text{p}}$.

The string interacts with the bridge using the two-sided collision potential presented in Eq. (10.34):

$$\phi(\eta_{\text{sm}}) = \frac{K_{\text{sm}}}{\alpha_{\text{sm}} + 1} |\eta_{\text{sm}}|^{\alpha_{\text{sm}}+1}, \tag{15.7}$$

which acts as a connection. Here, $\eta_{\text{sm}} = \eta_{\text{sm}}(t) = u(\chi_{\text{sm}}, t) - w(t)$ (in m) and the location of where the bridge is connected along the string is $\chi_{\text{sm}} \in \mathcal{D}_{\text{s}}$.

Recalling the process of writing the collision potential in quadratic form in Eq. (10.4)

$$\phi'(\eta) = \psi\psi' \quad \text{where} \quad \psi = \sqrt{2\phi} \quad \text{and} \quad \psi' = \frac{\dot{\psi}}{\dot{\eta}}, \tag{15.8}$$

the complete system can be written next.

Complete system

Looking towards discretisation, the separate variables $g_{\text{sm}} = \psi'_{\text{sm}}$ and $g_{\text{mp}} = \psi'_{\text{mp}}$ are used and the full system can be written as

$$\mathcal{L}_{\text{s}} u = -\delta(\chi - \chi_{\text{B}}) F_{\text{B}} + \delta(\chi - \chi_{\text{sm}}) \psi_{\text{sm}} g_{\text{sm}}, \tag{15.9a}$$

$$\mathcal{L}_{\text{m}} w = -\psi_{\text{sm}} g_{\text{sm}} + \psi_{\text{mp}} g_{\text{mp}}, \tag{15.9b}$$

$$\mathcal{L}_{\text{p}} z = -\delta(x - x_{\text{mp}}, y - y_{\text{mp}}) \psi_{\text{mp}} g_{\text{mp}}, \tag{15.9c}$$

$$\dot{\psi}_{\text{sm}} = g_{\text{sm}} \dot{\eta}_{\text{sm}}, \tag{15.9d}$$

$$\dot{\psi}_{\text{mp}} = g_{\text{mp}} \dot{\eta}_{\text{mp}}, \tag{15.9e}$$

$$\eta_{\text{sm}}(t) = w(t) - u(\chi_{\text{sm}}, t), \tag{15.9f}$$

$$\eta_{\text{mp}}(t) = z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t). \tag{15.9g}$$

See Figure 15.1 for a visual overview of system (15.9). Notice that when compared to the system presented in paper [D], Eqs. (15.9d) and (15.9e) have been added for coherence with the theory presented in Chapter 10, as well as the introduction of g_{sm} and g_{mp} already in the continuous system.

15.2.2 Discrete time

Using the process explained in Section 10.1.2 for discretising the collision terms and introducing[6]

$$\xi^n = \frac{k}{2} g^n \delta_{t\cdot} \eta^n + \psi^{n-1/2}, \tag{15.10}$$

[6] The definition for ξ^n was wrong in paper [D], where $\psi^{n-1/2}$ was subtracted rather than added. It has been corrected here and included in Appendix A.

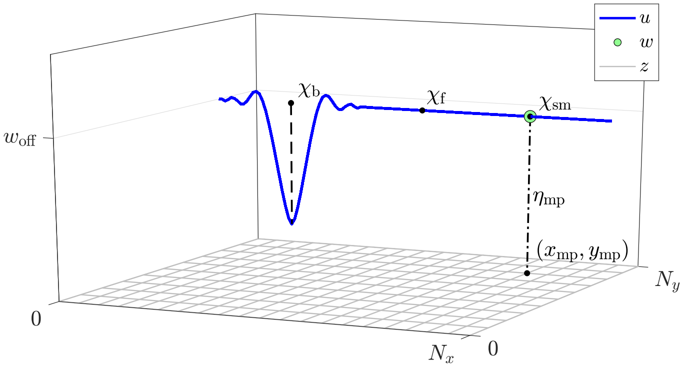


Fig. 15.1: A visualisation of system (15.9) with various important coordinates highlighted. The figure includes the offset and the damping finger described in paper [D]. Note that η_sm (Eq. (15.9f)) is not shown as it is close to 0 at all times. (Figure taken from [D].)

for brevity, system (15.9) can be discretised as follows:[7]

$$
\begin{aligned}
\ell_\mathrm{s} u_l^n &= -J_{l,3}(\chi_\mathrm{B})F_\mathrm{B} + J_{l,0}(\chi_\mathrm{sm})\xi_\mathrm{sm}^n g_\mathrm{sm}^n, && (15.11a)\\
\ell_\mathrm{m} w^n &= -\xi_\mathrm{sm}^n g_\mathrm{sm}^n + \xi_\mathrm{mp}^n g_\mathrm{mp}^n, && (15.11b)\\
\ell_\mathrm{p} z_{l,m}^n &= -J_{(l,m),0}(x_\mathrm{mp}, y_\mathrm{mp})\xi_\mathrm{mp}^n g_\mathrm{mp}^n, && (15.11c)\\
\delta_{t+}\psi_\mathrm{sm}^{n-1/2} &= g_\mathrm{sm}^n \delta_{t\cdot}\eta_\mathrm{sm}^n, && (15.11d)\\
\delta_{t+}\psi_\mathrm{mp}^{n-1/2} &= g_\mathrm{mp}^n \delta_{t\cdot}\eta_\mathrm{mp}^n, && (15.11e)\\
\eta_\mathrm{sm}^n &= w^n - u_\mathrm{br}^n, && (15.11f)\\
\eta_\mathrm{mp}^n &= z_\mathrm{br}^n - w^n, && (15.11g)
\end{aligned}
$$

where the discrete linear (partial) differential operators ℓ are the discrete counterparts of $\mathcal{L}$ in system (15.9) and are discretised as shown in their respective chapters as

$$
\begin{aligned}
\ell_\mathrm{s} &= \rho_\mathrm{s} A\delta_{tt} - T\delta_{\chi\chi} + EI\delta_{\chi\chi\chi\chi} + 2\rho_\mathrm{s} A\sigma_{0,\mathrm{s}}\delta_{t\cdot} - 2\rho_\mathrm{s} A\sigma_{1,\mathrm{s}}\delta_{t-}\delta_{\chi\chi}, && (15.12a)\\
\ell_\mathrm{b} &= M\delta_{tt} + M\omega_0^2 + M\sigma_\mathrm{m}\delta_{t\cdot}, && (15.12b)\\
\ell_\mathrm{p} &= \rho_\mathrm{p} H\delta_{tt} + D\delta_\Delta\delta_\Delta + 2\rho_\mathrm{p} H\sigma_{0,\mathrm{p}}\delta_{t\cdot} - 2\rho_\mathrm{p} H\sigma_{1,\mathrm{p}}\delta_{t-}\delta_\Delta. && (15.12c)
\end{aligned}
$$

Furthermore, the definitions for g_sm^n and g_mp^n can be found in Eq. (10.15)[8] and $J_{l,0}(\chi_\mathrm{sm})$ and $J_{l,3}(\chi_\mathrm{B})$ are the 0th-order and cubic spreading operators respectively, as defined in Section 8.2 and $J_{(l,m),0}(x_\mathrm{mp}, y_\mathrm{mp})$ is a 0th-order 2D

[7] Notice that subscript l is used as a spatial index for both the string and the plate for coherency with the paper, but are used as an index for different coordinate systems.

[8] Paper [D] still uses the old definition of g^n in Eq. (10.13).

spreading operator as defined in Section 11.1.

As 0th-order spreading operators are used for the collision terms in the string and body FD schemes, the definitions of Eqs. (15.11f) and (15.11g) do not use interpolation operators to obtain the string and plate values. Instead, for brevity, $u_{\text{br}}^n = u_{l_{\text{sm}}}^n$ with $l_{\text{sm}} = \lfloor \chi_{\text{sm}}/h_{\text{s}} \rfloor$ and $z_{\text{br}}^n = z_{(l_{\text{mp}},m_{\text{mp}})}^n$ with $(l_{\text{mp}}, m_{\text{mp}}) = (\lfloor x_{\text{mp}}/h_{\text{p}} \rfloor, \lfloor y_{\text{mp}}/h_{\text{p}} \rfloor)$ are introduced for the string and the plate respectively. Here h_{s} is the grid spacing for the string and h_{p} that for the plate.

There are a few components presented in paper [D] that the system does not include here. These are the offset w_{off} between the mass and the plate as well as the damping finger that interacts with the string to play different pitches. As the focus of this chapter is on the process of solving the system at the bridge for which these two components are not important, these will be ignored to avoid additional complexity. Further details on these components are provided in the paper.

15.3 Implementation details

This section extends paper [D] by providing more details on the solution of the string-bridge-body interaction.

Following Section 10.2 one could expand Eqs. (15.11a), (15.11b) and (15.11c) and treat these as a system of linear equations (see Section B.3). This would require an inversion of a 3 × 3 matrix each iteration. To simplify things slightly, and looking towards real-time implementation, one could instead solve for $\delta_{t\cdot}\eta_{\text{sm}}^n$ and $\delta_{t\cdot}\eta_{\text{mp}}^n$ directly, which would require a 2 × 2 matrix inversion each iteration and requires much fewer operations (at best, less than half).

As it is assumed that one will not bow the bridge, i.e. $\chi_{\text{B}} \neq \chi_{\text{sm}}$, the bowing term will be disregarded when calculating the interaction between the components.

Recalling (15.11f) and (15.11g), one can write the following:

$$\delta_{t\cdot}\eta_{\text{sm}}^n = \delta_{t\cdot}(w^n - u_{\text{br}}^n) \quad \text{and} \quad \delta_{t\cdot}\eta_{\text{mp}}^n = \delta_{t\cdot}(z_{\text{br}}^n - w^n),$$

and expanding the right-hand sides yields

$$\begin{aligned} \delta_{t\cdot}\eta_{\text{sm}}^n &= \frac{w^{n+1} - w^{n-1} - u_{\text{br}}^{n+1} + u_{\text{br}}^{n-1}}{2k} \quad \text{and} \\ \delta_{t\cdot}\eta_{\text{mp}}^n &= \frac{z_{\text{br}}^{n+1} - z_{\text{br}}^{n-1} - w^{n+1} + w^{n-1}}{2k}. \end{aligned} \tag{15.13}$$

To solve the system, inner product of Eqs. (15.11a) and (15.11c) are taken with $J_{l,0}(\chi_{\text{sm}})$ and $J_{(l,m),0}(x_{\text{mp}}, y_{\text{mp}})$ respectively. As 0th-order spreading operators are used, their norms over discrete string domain d_{s} and discrete plate domain d_{p}, respectively, reduce as follows (also see Eq. (11.16)): $\|J_{l,0}(\chi_{\text{sm}})\|_{d_{\text{s}}}^2 = 1/h_{\text{s}}$

and $\|J_{(l,m),0}(x_{\text{mp}},y_{\text{mp}})\|^2_{d_{\text{p}}} = 1/h_{\text{p}}^2$. This yields the following updates for Eqs. (15.11a), (15.11b) and (15.11c) at the locations of interaction

$$u_{\text{br}}^{n+1} = u_{\text{br}}^{\star} + \frac{k^2}{h_{\text{s}}\rho_{\text{s}}A(1+\sigma_{0,\text{s}}k)}\left(\frac{(g_{\text{sm}}^n)^2k}{2}\delta_{t\cdot}\eta_{\text{sm}}^n + \psi_{\text{sm}}^{n-1/2}g_{\text{sm}}^n\right), \tag{15.14a}$$

$$\begin{aligned} w^{n+1} = w^{\star} &- \frac{k^2}{M\left(1+\frac{\sigma_{\text{m}}k}{2}\right)}\left(\frac{(g_{\text{sm}}^n)^2k}{2}\delta_{t\cdot}\eta_{\text{sm}}^n + \psi_{\text{sm}}^{n-1/2}g_{\text{sm}}^n\right) \\ &+ \frac{k^2}{M\left(1+\frac{\sigma_{\text{m}}k}{2}\right)}\left(\frac{(g_{\text{mp}}^n)^2k}{2}\delta_{t\cdot}\eta_{\text{mp}}^n + \psi_{\text{mp}}^{n-1/2}g_{\text{mp}}^n\right), \end{aligned} \tag{15.14b}$$

$$z_{\text{br}}^{n+1} = z_{\text{br}}^{\star} - \frac{k^2}{h_{\text{p}}^2\rho_{\text{p}}H(1+\sigma_{0,\text{p}}k)}\left(\frac{(g_{\text{mp}}^n)^2k}{2}\delta_{t\cdot}\eta_{\text{mp}}^n + \psi_{\text{mp}}^{n-1/2}g_{\text{mp}}^n\right), \tag{15.14c}$$

where the update equations of the components in isolation (without the collision terms) at their respective interaction locations are

$$\begin{aligned} u_{\text{br}}^{\star} &= \frac{2u_{\text{br}}^n - u_{\text{br}}^{n-1} + c^2k^2\delta_{\chi\chi}u_{\text{br}}^n - \kappa_{\text{s}}^2k^2\delta_{\chi\chi\chi\chi}u_{\text{br}}^n + \sigma_{0,\text{s}}ku_{\text{br}}^{n-1} + 2\sigma_{1,\text{s}}k^2\delta_{t-}\delta_{\chi\chi}u_{\text{br}}^n}{1+\sigma_{0,\text{s}}k}, \\ w^{\star} &= \frac{2w^n - w^{n-1} - k^2\omega_0^2w^n + \frac{\sigma_{\text{m}}k}{2}w^{n-1}}{1+\frac{\sigma_{\text{m}}k}{2}}, \\ z_{\text{br}}^{\star} &= \frac{2z_{\text{br}}^n - z_{\text{br}}^{n-1} - \kappa_{\text{p}}^2k^2\delta_{\Delta}\delta_{\Delta}z_{\text{br}}^n + \sigma_{0,\text{p}}kz_{\text{br}}^{n-1} + 2\sigma_{1,\text{p}}k^2\delta_{t-}\delta_{\Delta}z_{\text{br}}^n}{1+\sigma_{0,\text{p}}k}, \end{aligned}$$

with $c=\sqrt{T/\rho_{\text{s}}A}$, $\kappa_{\text{s}}=\sqrt{E_{\text{s}}I/\rho_{\text{s}}A}$ and $\kappa_{\text{p}}=\sqrt{D/\rho_{\text{p}}H}$.

The definitions in Eqs. (15.14) can then be inserted into Eqs. (15.13) and solved for $\delta_{t\cdot}\eta_{\text{sm}}^n$ and $\delta_{t\cdot}\eta_{\text{mp}}^n$. This can be treated as a system of linear equations and be solved according to

$$\begin{bmatrix}\delta_{t\cdot}\eta_{\text{sm}}^n \\ \delta_{t\cdot}\eta_{\text{mp}}^n\end{bmatrix} = \mathbf{A}^{-1}\mathbf{v}, \tag{15.15}$$

where

$$\mathbf{A} = \begin{bmatrix} 1+\frac{(g_{\text{sm}}^n)^2k^2}{2M(2+\sigma_{\text{m}}k)} + \frac{(g_{\text{sm}}^n)^2k^2}{4\rho_{\text{s}}Ah_{\text{s}}(1+\sigma_{0,\text{s}}k)} & -\frac{(g_{\text{mp}}^n)^2k^2}{2M(2+\sigma_{\text{m}}k)} \\ -\frac{(g_{\text{sm}}^n)^2k^2}{2M(2+\sigma_{\text{m}}k)} & 1+\frac{(g_{\text{mp}}^n)^2k^2}{2M(2+\sigma_{\text{m}}k)} + \frac{(g_{\text{mp}}^n)^2k^2}{4h_{\text{p}}^2\rho_{\text{p}}H(1+\sigma_{0,\text{p}}k)} \end{bmatrix},$$

$$\mathbf{v} = \begin{bmatrix} \frac{w^{\star}-w^{n-1}-u_{\text{br}}^{\star}+u_{\text{br}}^{n-1}}{2k} - \frac{k(\psi_{\text{sm}}^{n-1/2}g_{\text{sm}}^n-\psi_{\text{mp}}^{n-1/2}g_{\text{mp}}^n)}{M(2+\sigma_{\text{m}}k)} - \frac{\psi_{\text{sm}}^{n-1/2}g_{\text{sm}}^nk}{2\rho_{\text{s}}Ah_{\text{s}}(1+\sigma_{0,\text{s}}k)} \\ \frac{z_{\text{br}}^{\star}-z_{\text{br}}^{n-1}-w^{\star}+w^{n-1}}{2k} + \frac{k(\psi_{\text{sm}}^{n-1/2}g_{\text{sm}}^n-\psi_{\text{mp}}^{n-1/2}g_{\text{mp}}^n)}{M(2+\sigma_{\text{m}}k)} - \frac{\psi_{\text{mp}}^{n-1/2}g_{\text{mp}}^nk}{2h_{\text{p}}^2\rho_{\text{p}}H(1+\sigma_{0,\text{p}}k)} \end{bmatrix}.$$

The solutions obtained for $\delta_{t\cdot}\eta_{\text{sm}}^n$ and $\delta_{t\cdot}\eta_{\text{mp}}^n$, can then be used directly in ξ_{sm}^n and ξ_{mp}^n in Eqs. (15.11a), (15.11b) and (15.11c) to calculate u_l^{n+1}, w^{n+1} and $z_{(l,m)}^{n+1}$

respectively. They can also be used directly to calculate $\psi_{\text{sm}}^{n+1/2}$ and $\psi_{\text{mp}}^{n+1/2}$ in Eqs. (15.11d) and (15.11e) respectively.

Figure 15.2 shows three consecutive plots of the system excited with a raised cosine. The bow is not used here, to highlight the collision behaviour. One can observe that the string pulls the mass downwards through their connection, causing the latter to collide with the plate.

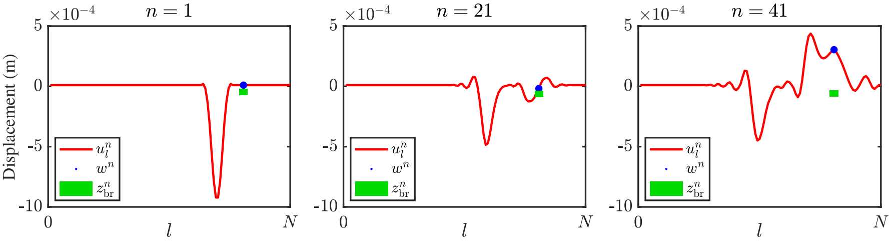


Fig. 15.2: The string of the tromba marina (red) excited using a raised cosine. The string-mass interaction forces the mass (blue) downwards, which collides with the plate (green).

15.3.1 Energy analysis

Using the energy analysis techniques presented in Section 3.4, the total energy of the system can be shown to be of the following form:

$$\delta_{t+}\left(\mathfrak{h}_{\text{s}}+\mathfrak{h}_{\text{m}}+\mathfrak{h}_{\text{p}}+\mathfrak{h}_{\text{sm}}+\mathfrak{h}_{\text{mp}}\right)=-\mathfrak{q}_{\text{s}}-\mathfrak{q}_{\text{m}}-\mathfrak{q}_{\text{p}}-\mathfrak{q}_{\text{B}}-\mathfrak{p}_{\text{B}}. \tag{15.16}$$

Here, the total energy of the string is (Eq. (4.29)),

$$\mathfrak{h}_{\text{s}}=\frac{\rho_{\text{s}}A}{2}\|\delta_{t-}u_l^n\|_{d_{\text{s}}}^2+\frac{T}{2}\langle\delta_{x+}u_l^n,e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d_{\text{s}}}}+\frac{E_{\text{s}}I}{2}\langle\delta_{xx}u_l^n,e_{t-}\delta_{xx}u_l^n\rangle_{\overline{\underline{d_{\text{s}}}}},$$

the total energy of the mass is (Eq. (9.5))

$$\mathfrak{h}_{\text{m}}=\frac{M}{2}(\delta_{t-}w^n)^2+\frac{K}{2}w^n e_{t-}w^n,$$

and the total energy of the plate is (Eq. (6.51))

$$\mathfrak{h}_{\text{p}}=\frac{\rho_{\text{p}}H}{2}\left\|\delta_{t-}z_{l,m}^n\right\|_{d_{\text{p}}}^2+\frac{D}{2}\langle\delta_{\Delta}z_{l,m}^n,e_{t-}\delta_{\Delta}z_{l,m}^n\rangle_{\overline{\underline{d_{\text{p}}}}}.$$

The energy of the string-mass connection and the mass-plate collision are (Eq. (10.23))

$$\mathfrak{h}_{\text{sm}}=\frac{(\psi_{\text{sm}}^{n-1/2})^2}{2},\quad\text{and}\quad\mathfrak{h}_{\text{mp}}=\frac{(\psi_{\text{mp}}^{n-1/2})^2}{2},$$

respectively.

The damping terms are defined for the string as (Eq. (4.28))

$$\mathfrak{q}_\mathrm{s} = 2\sigma_{0,\mathrm{s}}\rho_\mathrm{s}A\|\delta_{t\cdot}u_l^n\|_{d_\mathrm{s}}^2 - 2\sigma_{1,\mathrm{s}}\rho_\mathrm{s}A\langle\delta_{t\cdot}u_l^n, \delta_{t-}\delta_{\chi\chi}u_l^n\rangle_{d_\mathrm{s}},$$

for the mass as (Eq. (9.6))

$$\mathfrak{q}_\mathrm{m} = R\left(\delta_{t\cdot}w^n\right)^2$$

and for the plate as (Eq. (6.50))

$$\mathfrak{q}_\mathrm{p} = 2\sigma_0\rho_\mathrm{p}H\|\delta_{t\cdot}z_{l,m}^n\|_{d_\mathrm{p}}^2 - 2\sigma_1\rho_\mathrm{p}H\langle\delta_{t\cdot}z_{l,m}^n, \delta_{t-}\delta_\Delta z_{l,m}^n\rangle_{d_\mathrm{p}}.$$

Finally, the power dissipated and supplied by the bow are defined as (Eq. (8.27))

$$\mathfrak{q}_\mathrm{B} = f_\mathrm{B}^n\Phi(v_\mathrm{rel}^n)v_\mathrm{rel}^n \quad \text{and} \quad \mathfrak{p}_\mathrm{B} = f_\mathrm{B}^n\Phi(v_\mathrm{rel}^n)v_\mathrm{B}^n,$$

respectively.

Figure 15.3 shows the plots of the energy for an implementation of the tromba marina presented in this chapter without losses. The system is excited with a raised cosine for clarity of the figures and plots correspond to the system states shown in Figure 15.2.

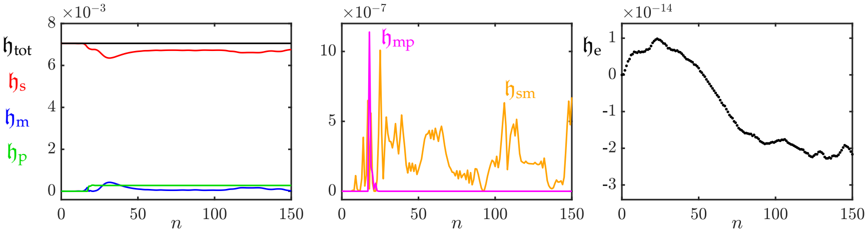


Fig. 15.3: The energy of the tromba marina excited with a raised cosine (see Figure 15.2). The left panel shows the total energy (black) present in the system as well as the energy of the string (red), mass (blue) and plate (green) respectively. The second panel shows the energy of the mass-plate (magenta) and string-mass (orange) interactions and the right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

Chapter 16

The Trombone

This chapter provides an extended summary for the paper "A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes" [H].

The trombone is an extremely interesting instrument from a simulation perspective, due to the time-varying length of the acoustic tube. In FDTD-based simulations, the trombone poses a challenge due to issues regarding adding and removing points to the grid, or simulation quality, as also pointed out by Harrison in [62]. As the dynamic grid method presented in paper [G] (also see Chapter 12) attempts to resolve these issues, the trombone is an excellent use case for this method to be applied to.

The air propagation in the trombone has been modelled using a coupled system of two first-order PDEs, presented in Section 5.2. Although Webster's equation presented in Section 5.1) could have been used, the state-of-the-art models for brass instruments using FDTD methods use first-order PDEs [73, 62]. Some extensions, such as the Levine and Schwinger radiation model presented in 5.2.5 and used in [62] and viscothermal losses in [73], could then easily be added, although the latter has been left for future work.

This chapter first presents a short summary of paper [H] that relates its contents to the rest of this thesis. Then, as the process of applying the dynamic grid method to the trombone is not straightforward, several considerations made during this process will be given. Finally, the collision method from Chapter 10 is applied to the lip reed presented in Chapter 9, and details on the implementation will be given.

16.1 Summary

The paper presents a real-time implementation of the trombone based on FDTD methods, and using the dynamic grid method presented in Chapter 12. The acoustic tube has been modelled using a system of coupled first-order equations as described in Section 5.2. The lip reed has been modelled using the 1-DoF mass spring system presented in Chapter 9, and has been extended using the collision method described in Chapter 10, details of which will be given in Section 16.3.

The physical model of the trombone was implemented in real time in C++ using the JUCE framework (see Chapter 13). The application can be found online, as well as a demo showcasing it.[1],[2]

The implementation allowed for the total range of the tube length (2.593-3.653 m) to be traversed in 0.06 s (corresponding to 20 samples between grid configurations). An informal evaluation by the authors showed that no noticeable artefacts were observed, but naturally, proper listening tests need to confirm this. It must be noted that to ensure stability, especially at fast changes between grid configurations, the Courant number has been set slightly lower than the stability condition ($\lambda = 0.999$). This decrease in λ does cause a small, but negligible decrease in simulation quality and bandwidth (see Section 2.4.4).

Future work includes to add viscothermal losses [73] and nonlinear effects [121], as well as investigating intuitive mapping to control the simulation.

16.2 Dynamic grid considerations

The main contribution of the paper is the use of the dynamic grid method to implement the time-varying acoustic tube. As paper [H] provides extensive details on the implementation of the dynamic grid method applied to the case of the trombone, these will not be given here. Instead, this section presents several considerations that needed to be taken into account in order to use the dynamic grid method in the case of the acoustic tube.

First-order system

The dynamic grid method presented in paper [G] uses the 1D wave equation as a test case. Although this is not used to model the trombone, the underlying behaviour is the same for a cylindrical tube. Section 5.2.1 shows that the first-order system in Eq. (5.37) can be reduced to Webster's equation, which – for a cylindrical tube – reduces to the 1D wave equation. It was therefore concluded, that the dynamic grid method could be applied to the acoustic tube, under the condition that its cross-section does not vary at the location where the method is applied. In other words, as long as the system locally reduces to the 1D wave equation (at the location of the split) the method can be applied in a similar fashion.

[1] https://github.com/SilvinWillemsen/cppBrass/releases/

[2] https://youtu.be/Ht5gVNrshYo

As the state of the pressure variable in Eq. (5.37a) is discretised to the non-interleaved grid, it was chosen to apply the dynamic grid method to the pressure grid, rather than that of the particle velocity. That said, if the aforementioned constraint regarding a constant cross-section is kept, applying the method to the velocity grid should work just as well.

Location of split

It was chosen to split the system in the middle of the slide crook of the trombone, i.e., at the far end of the trombone slide. First of all, it could be reasonable to assume that the air in the tube would "go away from" or "go towards" that point, while the slide extends or contracts. Secondly, the cross-sectional area is constant at this location, and satisfies the aforementioned condition.

Time-varying length

In paper [G], the time-varying parameter is the wave speed c, whereas the trombone has a time-varying length L. As stated in paper [G], c and L are linked by the fundamental frequency in Eq. (2.40) and a change in one yields identical behaviour as an inverse change in the other. The length could thus be made time-varying in a straightforward manner, and the resulting equations could consequently be applied to the acoustic tube.

16.3 Lip-reed with collision

To excite the trombone, the lip reed model presented in Chapter 9 was used, and extended using the collision method presented in Chapter 10. This section provides details on this combination and the implementation and follows the notation of the thesis for consistency.

16.3.1 Continuous time

In continuous time, a collision can be added to Eq. (9.7) in the same way as for the mass-barrier collision presented in Section 10.1 as

$$M\ddot{y} = -Ky - R\dot{y} + S_{\mathrm{r}}\Delta p + \psi g \tag{16.1}$$

where $g = \psi'$ and ψ and ψ' are as defined in Eq. (10.4).[3] In the implementation, the frequency of the lip reed is made to be time varying and causes $K = K(t)$ to be time-dependent. Other parameters are the same as in Eq. (9.7).

[3] Notice that, as y is the displacement of the upper lip, the 'barrier' modelling the lower lip is placed below, resulting in a positive collision force on y.

16.3.2 Discrete time

The discretisation follows Section 10.1 for the collision and Section 9.3 for the lip reed.

As the lip reed is discretised on the interleaved (temporal) grid, the collision term needs to be as well. Dividing all terms by M, and using $\omega_0 = \omega_0^{n+1/2} = \sqrt{K^{n+1/2}/M}$ and $\sigma_\mathrm{r} = R/M$, yields[4]

$$\delta_{tt} y^{n+1/2} = -\omega_0^2 \mu_{t\cdot} y^{n+1/2} - \sigma_\mathrm{r} \delta_{t\cdot} y^{n+1/2} + \frac{S_\mathrm{r}}{M} \Delta p^{n+1/2} + \frac{\psi^{n+1/2} g^{n+1/2}}{M}. \quad (16.2)$$

Here,

$$g^{n+1/2} = \frac{\delta_{t+} \psi^n}{\delta_{t\cdot} \eta^{n+1/2}} \, , \quad (16.3)$$

and the distance between the lips

$$\eta^{n+1/2} = -H_0 - y^{n+1/2} \quad (16.4)$$

with static equilibrium separation H_0. Here $-H_0$ can be interpreted as the location of the lower lip. Using $\mu_{t+} \psi^n = \psi^{n+1/2}$ (which is Eq. (10.7) shifted to the interleaved grid), Eq. (16.2) can be rewritten to

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

In the following, the superscript $n + 1/2$ is suppressed for y, Δp, g and η for brevity. Rewriting Eq. (16.3) to

$$\delta_{t+} \psi^n = g \delta_{t\cdot} \eta \quad (16.5)$$

and using identity (2.27c), one arrives at

$$\delta_{tt} y = -\omega_0^2 \mu_{t\cdot} y - \sigma_\mathrm{r} \delta_{t\cdot} y + \frac{S_\mathrm{r}}{M} \Delta p + \left(\frac{k}{2} g \delta_{t\cdot} \eta + \psi^n \right) \frac{g}{M}. \quad (16.6)$$

As the barrier is static and placed below y, this implies that

$$\delta_{t\cdot} \eta = -\delta_{t\cdot} y, \quad (16.7)$$

and a solution for $y^{n+3/2}$ can be obtained:

$$\alpha_\mathrm{r} y^{n+3/2} = 4 y^{n+1/2} + \beta_\mathrm{r} y^{n-1/2} + \xi_\mathrm{r} \Delta p + 4 \psi^n \gamma_\mathrm{r}, \quad (16.8)$$

[4] Note that the paper uses ω_r and M_r instead of ω_0 and M.

with

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

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

To be able to calculate $y^{n+3/2}$, definitions for $g^{n+1/2}$ and $\Delta p^{n+1/2}$ need to be found.

Calculating $g^{n+1/2}$

Following Chapter 10, g can be calculated using (Eq. (10.15) shifted to the interleaved grid)

$$g^{n+1/2} = \begin{cases} \kappa\sqrt{\dfrac{K_{\text{c}}(\alpha_{\text{c}}+1)}{2}} \cdot (\eta^{n+1/2})^{\frac{\alpha_{\text{c}}-1}{2}}, & \text{if } \eta^{n+1/2} \geq 0, \\ -2\dfrac{\psi^n}{\eta^\star - \eta^{n-1/2}}, & \text{if } \eta^{n+1/2} < 0 \ \text{ and } \ \eta^\star \neq \eta^{n-1/2}, \\ 0, & \text{if } \eta^{n+1/2} < 0 \ \text{ and } \ \eta^\star = \eta^{n-1/2}, \end{cases}$$

where parameters are as in Eq. (10.15). Furthermore, $\eta^\star = -H_0 - y^\star$ where

$$y^\star = \frac{4}{\alpha_{\text{r}}^\star} y^{n+1/2} + \frac{\beta_{\text{r}}^\star}{\alpha_{\text{r}}^\star} y^{n-1/2} + \frac{\xi_{\text{r}}}{\alpha_{\text{r}}^\star}\Delta p^\star, \tag{16.10}$$

is the update equation of the system without the effect of the collision and

$$\alpha_{\text{r}}^\star = 2 + \omega_0^2 k^2 + \sigma_{\text{r}} k, \quad \text{and} \quad \beta_{\text{r}}^\star = \sigma_{\text{r}} k - 2 - \omega_0^2 k^2,$$

are the coefficients in Eq. (16.8) without the collision terms (as found in Eq. (9.15)).[5] Finally, $\Delta p^\star$ is the pressure difference calculated using Eq. (9.27), i.e., without the effect of the collision. Once $g^{n+1/2}$ is calculated, $\Delta p^{n+1/2}$ can be obtained.

Calculating $\Delta p^{n+1/2}$

Following Section 9.3.1, to calculate $\Delta p^{n+1/2}$, one starts by rewriting the scheme in Eq. (16.6) to

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

[5] Notice that ξ_{r} is unchanged.

with

$$a_1^{n+1/2} = \frac{2}{k} + \omega_0^2 k + \sigma_\mathrm{r} + \frac{g^2 k}{2M} \geq 0, \quad a_2 = \frac{S_\mathrm{r}}{M} \geq 0\,,$$
$$\text{and} \quad a_3^{n+1/2} = \left(\frac{2}{k}\delta_{t-} - \omega_0^2 e_{t-}\right) y + \frac{g}{M}\psi^n\,.$$

Note that $a_1^{n+1/2}$ is now time-dependent through g and ω_0 but remains non-negative. The rest of the variables and process in Section 9.3.1 are unchanged. Notice that the calculation for $\Delta p^{n+1/2}$ has to be performed twice: once to obtain the pressure difference without the collision effect $\Delta p^\star$, and once to obtain the final pressure difference $\Delta p^{n+1/2}$.

Last steps

After the definitions for g and Δp are found using the steps above, and $y^{n+3/2}$ is calculated using Eq. (16.8), ψ^{n+1} can be calculated by expanding Eq. (16.5) and substituting Eq. (16.7) according to

$$\psi^{n+1} = \psi^n - \frac{g}{2}\left(y^{n+3/2} - y^{n-1/2}\right). \tag{16.11}$$

16.3.3 Energy analysis

The energy analysis for the lip reed shown in Section 9.4 can be extended to contain the collision. Equation (9.31) can be extended to

$$\delta_{t+}(\mathfrak{h}_\mathrm{t} + \mathfrak{h}_\mathrm{r}) = -\mathfrak{q}_\mathrm{r} - \mathfrak{p}_\mathrm{r} + (\mu_{t+}\psi^n)\frac{\delta_{t+}\psi^n}{\delta_{t\cdot}\eta^{n+1/2}}(\delta_{t\cdot}y^{n+1/2})$$

which, using Eq. (16.7), yields

$$\delta_{t+}(\mathfrak{h}_\mathrm{t} + \mathfrak{h}_\mathrm{r}) = -\mathfrak{q}_\mathrm{r} - \mathfrak{p}_\mathrm{r} - (\mu_{t+}\psi^n)(\delta_{t+}\psi^n),$$

with

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

Part VI

Conclusions and Perspectives

Chapter 17

Conclusions and Perspectives

This chapter concludes this work by providing a summary of the thesis. Furthermore, perspectives for the future and possible continuations of this project will be given.

17.1 Summary

This thesis presents the result of a PhD project on physical modelling of musical instruments using FDTD methods. Part I provided an introduction to the field of physical modelling, after which the basics of FDTD methods and analysis techniques were described in a tutorial-like fashion. Part II provided detailed information about the resonators used for the contributions in this PhD project, and Part III did the same for the exciters. Part IV presented collisions and connections as various ways in which the resonators could interact. Part V summarised most papers included in Part VII and related the contributions to the theory presented in the parts before. Part V is summarised below.

Contributions

Chapter 12 summarised paper [G], which presents a novel method to dynamically vary grid configurations in FDTD-based musical instrument simulations. The chapter extended the paper by providing information on experiments done, which substantiate choices made in the paper. Chapter 13 presented considerations on the real-time implementation of FD schemes as well as their control. Chapter 14 summarised papers [A] and [B], which present the real-time implementation and control of a large-scale modular environment using the esraj, the hammered dulcimer and the hurdy gurdy as instrument test-cases. Chapter 15 summarised papers [D] and [E], which present the real-time implementation of the tromba marina using the Sensel Morph and the

PHANTOM Omni as controllers respectively. The chapter extended on the papers by providing more details on the implementation. Finally, Chapter 16 summarised paper [H], which presents a real-time implementation of the trombone using the dynamic grid method, and provided additional design considerations and implementation details.

17.2 Perspectives and future work

Both an advantage and a disadvantage of the field of physical modelling musical instruments, is that one is never done. There are always more instruments to model or existing models to improve. This section contains several possibilities for continuations of this work and some perspectives on how to move forward.

17.2.1 Parameter design

One could argue that the sound produced by musical instrument simulations depends in equal parts on the model describing the system and the parameters used. Parameter design and tuning is therefore an extremely important aspect in creating physical models that sound good.

Some models might contain many parameters that are nonlinearly interconnected, such as the elasto-plastic friction model presented in paper [C], causing the tuning of the parameters to become extremely time-consuming. A possible solution for this, could be to tune the parameters based on a recording of the physical system one tries to model. One could automate this process using machine learning methods and a 'gray-box' approach where the model is known, but the parameters are fitted to the input. This was done for virtual analog models in e.g. [122, 123].

For parameters controlling the exciter, such as force, velocity, and position of a bow, a real-time implementation can be extremely helpful to judge the sound qualities of the simulation. Several times during this project, the simulation did not sound good, due to the use of static control parameters in `MATLAB`. The real-time implementation, where the control parameters were varied using human control, sounded much better and more natural. Parameters of the resonator could also be exposed and tuned in real-time, but this causes stability concerns in FDTD-based instrument simulations.[1]

[1] One could always set the states of the system to 0 when a parameter is changed, and reinitialise the system based on the new parameter. Alternatively, one could use the dynamic grid method from paper [G], but this might be a slightly overkill to implement only for parameter tuning.

17.2.2 Realism

The focus of this project was to create real-time simulations of musical instruments. Although a natural or realistic sound was, of course, desired, this was not the main focus, and more work could be done to achieve this.

If the goal is to create an extremely realistic musical instrument simulation, one would have to spend much time tuning the model parameters (see Section 17.2.1) and use more accurate and complex models. It is safe to say, that at the time of writing, accurately simulating the complete physics of a musical instrument in 3D, including nonlinear effects, is not something that personal computers will be able to do any time soon.

That said, simulations using simplified models, which are able to run in real time (such as those presented in this work), can already sound quite realistic. From a perceptual point of view, using more complex models might thus be unnecessary. Furthermore, various components, such as the instrument body or the room it is played in, could be included as an impulse response, either obtained from a recording or generated using a highly-accurate (non-real-time) physical model. Comparisons to real instruments and perceptual evaluations could then be used to verify the realism of the sound produced by the physical model.

It must be said that realistic-sounding implementations do not have to be the goal of physical modelling. Another angle, rather than trying to recreate traditional musical instruments, is to make completely new instruments. As mentioned several times throughout this thesis (see Section 1.4.3 and Chapter 12), one could even create physically impossible instruments, where realism is the opposite of what one aims for.

17.2.3 Control

Much work could still be done on investigating natural and intuitive control of the physical models. This PhD project explored two ways of controlling the musical instrument simulations.

Firstly, the Sensel Morph (see Section 13.3.1) was used for bowing, striking and plucking strings. Even though, bowing strings using this controller does not resemble the act of bowing a string in the physical world, the interaction could be deemed intuitive, as shown in paper [A].

Secondly, the PHANTOM Omni (see Section 13.3.2) was used to control the bow in a way that more closely resembled physical bowing. The evaluation in paper [E] shows that the interaction with the instrument, especially the haptic feedback, was "realistic" and "natural". Other aspects of the instrument, such as pitch control were deemed less natural, as one did not physically interact with a string.

Future work includes exploring other hardware to control musical instrument simulations, or even building new ones. Custom controllers could be made to control physical models of traditional musical instruments. These could either be inspired by their real-life counterpart, similar to what a digital keyboard is to a piano, or be made completely different, as the sound of the instrument is no longer coupled to its shape or form.

17.2.4 Evaluation

There are various ways to evaluate real-time implementations of musical instrument simulations. Throughout the PhD project, the main focus was on technical evaluations and revolved around the analysis techniques presented in Chapter 3 and the computational speed of the algorithm. Together with informal evaluations (by the authors of the respective paper) about the sound quality and the interaction mapping, these formed the success criteria for most of the published work.

Although user-evaluation of the real-time instrument simulations was not a large part of this project, it is an important aspect in all of the aforementioned future work. Evaluations can, for example, be used as a part of an iterative design process, and aimed at parameter tuning (either for realism or not) or intuitive instrument control.

The most elaborate user evaluation performed during this project is presented in paper [E] and tested the playability of the tromba marina using the PHANTOM Omni device. As this instrument was also controlled by the Sensel Morph in paper [D], an evaluation that compares the two controllers could be made in the future. Apart from evaluating the differences in natural or intuitive control, one could investigate cross-modal effects between control (haptics) and audio, where one could investigate whether the way that the simulation is controlled has an influence on sound perception.

One interesting observation, shown in paper [E], was that players found the virtual tromba marina hard to play. This could indeed be expected, as no participant ever played the instrument before, neither the virtual one, nor the physical one. For other unknown instruments, including completely novel ones, this shows that their evaluation is challenging. Like a real instrument, it might take months, or even years of practice to produce well-sounding auditory results, or to determine whether the chosen mapping is satisfactory.

17.2.5 Dynamic grid

Finally, where the author personally sees most potential for continuations of this work, is the further development of the dynamic grid method presented in paper [G]. During this project, only the cases of the 1D wave equation and the acoustic tube have been explored. Section 12.4 provides many real-world examples that the method could be applied to. These consist 1D systems, such as stiff strings, but also 2D systems, such as the musical saw and the talking drum. Furthermore, finding conditions under which the method is stable will allow the method to be safely integrated into other applications.

References

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

[2] M. Puckette, "Max at seventeen," *Computer Music Journal*, vol. 26, pp. 31–43, 2002.

[3] C. Roads, *The Computer Music Tutorial*. MIT Press, Cambridge, Massachusetts, 1996.

[4] 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.

[5] 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.

[6] J. O. Smith, "Virtual acoustic musical instruments: Review and update," *Center for computer research in music and acoustics (CCRMA)*, 2010.

[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] ——, "Physical audio signal processing (online book)," 2010. [Online]. Available: http://ccrma.stanford.edu/~jos/pasp/

[17] 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.

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

[19] 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.

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

[21] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley & Sons, 2009.

[22] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Institute of Physics Publishing*, 2006.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

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

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

[36] M. Paradiso, "To save the sound of a stradivarius, a whole city must keep quiet," 2019. [Online]. Available: https://www.nytimes.com/2019/01/17/arts/music/stradivarius-sound-bank-recording-cremona.html

[37] C. Livesay, "An italian town fell silent so the sounds of a stradivarius could be preserved," 2019. [Online]. Available: https://www.npr.org/2019/02/17/694056444/

[38] 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.

[39] 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.

[40] ——, "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.

[41] 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.

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

[43] 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.

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

[45] S. Yantis and R. A. Abrams, *Sensation and Perception*, 2nd ed. Worth Publishers, 2016.

[46] S. Willemsen, S. Serafin, and J. R. Jensen, "Virtual analog simulation and extensions of plate reverberation," in *Proc. of the 14th Sound and Music Computing Conference*, 2017, pp. 314–319.

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

[48] 3D Systems, Inc, "3D Systems Touch Haptic Device," 2021. [Online]. Available: https://www.3dsystems.com/haptics-devices/touch

[49] *MATLAB version 9.9.0.1592791 (R2020b) Update 5*, The Mathworks, Inc., Natick, Massachusetts, 2020.

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

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

[52] 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.

[53] 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.

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

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

[56] J. Sauveur, "Systeme général des intervalles des sons, et son application à tous les systèmes et à tous les instrumens de musique," *Histoire de L'Académie Royale des Sciences. Année 1701, Mémoires de Mathematique & de Physique*, pp. 297–364, 1701.

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

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

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

[60] 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.

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

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

[63] 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.

[64] 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.

[65] S. Bilbao, M. Ducceschi, and C. Webb, “Large-scale real-time modular physical modeling sound synthesis,” in *Proceedings of the 22th International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[66] H. Fletcher, “Normal vibration frequencies of a stiff string,” *Journal of the Acoustical Society of America*, vol. 36, no. 1, pp. 203–209, 1964.

[67] S. Willemsen, “stiffString.m,” 2021. [Online]. Available: https://gist.github.com/SilvinWillemsen/58f7353103ab6930da6139ed08f68c8f

[68] Y. Saad, *Iterative methods for sparse linear systems.* Philadelphia: SIAM, 2003.

[69] J. A. Kemp, “Theoretical and experimental study of wave propagation in brass musical instruments,” Ph.D. dissertation, The University of Edinburgh, 2002.

[70] P. Eveno, J. P. Dalmont, R. Caussé, and J. Gilbert, “Wave propagation and radiation in a horn: Comparisons between models and measurements,” *Acta Acustica united with Acustica*, vol. 98, pp. 158–165, 2012.

[71] 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.

[72] M. Atig, J.-P. Dalmont, and J. Gilbert, “Termination impedance of open-ended cylindrical tubes at high sound pressure level,” *Comptes Rendus Mécanique*, vol. 332, pp. 299–304, 2004.

[73] S. Bilbao and R. Harrison, “Passive time-domain numerical models of viscothermal wave propagation in acoustic tubes of variable cross section,” *The Journal of the Acoustical Society of America*, vol. 140, pp. 728–740, 2016.

[74] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe," *Physical Review*, vol. 73, no. 2, pp. 383–406, 1948.

[75] F. Silva, P. Guillemain, J. Kergomard, B. Mallaroni, and A. Norris, "Approximation formulae for the acoustic radiation impedance of a cylindrical pipe," *Journal of Sound and Vibration*, vol. 322, pp. 255–263, 2009.

[76] S. Bilbao and J. Chick, "Finite difference time domain simulation for the brass instrument bore," *Journal of the Acoustical Society of America*, vol. 134, no. 6, pp. 3860–3871, 2013.

[77] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[78] P. Morse and U. Ingard., *Theoretical Acoustics*. Princeton University Press, 1968.

[79] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, "Virtual analog effects," in *DAFX: Digital Audio Effects*, 2nd ed., U. Zölzer, Ed. John Wiley & Sons Ltd., 2011, pp. 473–522.

[80] O. Richards, "Investigation of the lip reed using computational modelling and experimental studies with an artificial mouth," Ph.D. dissertation, The University of Edinburgh, 2003.

[81] 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.

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

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

[84] 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.

[85] 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.

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

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

[88] R. Pitteroff and J. Woodhouse, "Mechanics of the contact area between a violin bow and a string. Part II: Simulating the bowed string," *Acta Acustica united with Acustica*, vol. 84, no. 4, pp. 744–757, 1998.

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

[90] 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.

[91] 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.

[92] 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.

[93] 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.

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

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

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

[97] 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.

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

[99] M. G. Onofrei, "Real-time implementation of bowed instruments using static and dynamic friction models," Master's thesis, Aalborg University, 2021.

[100] S. Bilbao, "Direct simulation of reed wind instruments," *Computer Music Journal*, vol. 33, no. 4, 2009.

[101] V. Chatziioannou and M. van Walstijn, "An energy conserving finite difference scheme for simulation of collisions," in *Proceedings of the Sound and Music Computing Conference 2013*, 2013, pp. 584–591.

[102] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2014.

[103] 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.

[104] N. Lopes, T. Hèlie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," in *Proceedings of the 5th IFAC 2015*, 2015, pp. 223–228.

[105] A. Falaize and T. Hélie, "Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach," *Applied Sciences*, vol. 6, p. 273, 2016.

[106] X. Yang, "Linear and unconditionally energy stable schemes for the binary fluid-surfactant phase field model," *Computer Methods in Applied Mechanics and Engineering*, vol. 318, pp. 1005–1029, 2017.

[107] C. Jiang, W. Cai, and Y. Wang, "A linearly implicit and local energy-preserving scheme for the sine-gordon equation based on the invariant energy quadratization approach," *Journal of Scientific Computing*, vol. 80, 2019.

[108] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: The case of collisions," *Proceedings of the 22th International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[109] H. Hertz, "Über die berührung fester elastischer körper," *Journal für die Reine und Angewandte Mathematik*, vol. 92, pp. 156–171, 1881.

[110] S. Willemsen, "connectedStringPlate.m," 2021. [Online]. Available: https://gist.github.com/SilvinWillemsen/91d10f6279af7fd0da7fb65fbd7647da

[111] G. F. Carrier, "On the non-linear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, pp. 157–165, 1945.

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

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

[114] R. Paisa and D. Overholt, "Enhancing the expressivity of the sensel morph via audio-rate sensing," in *Proceedings of the New Interfaces for Musical Expression conference*, 2019.

[115] L. Pardue, M. Ortiz, M. van Walstijn, P. Stapleton, and M. Rodger, "Vodhrán: collaborative design for evolving a physical model and interface into a proto-instrument," in *New Interfaces for Musical Expression*, 2020.

[116] M. van Walstijn, A. Bhanuprakash, and P. Stapleton, "Finite-difference simulation of linear plate vibration with dynamic distributed contact," in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021.

[117] 3D Systems Inc., "Openhaptics unity plugin user guide (toolkit version 3.5.0)," 2018. [Online]. Available: http://s3.amazonaws.com/dl.3dsystems.com/binaries/Sensable/OH/3.5/OpenHaptics_Toolkit_ProgrammersGuide.pdf

[118] A. Passalenti, R. Paisa, N. Nilsson, N. Andersson, F. Fontana, R. Nordahl, and S. Serafin, "No strings attached: Force and vibrotactile feedback in a virtual guitar simulation," *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1116–1117, 2019.

[119] ——, "No strings attached: Force and vibrotactile feedback in a guitar simulation," *Proceedings of the 16th Sound and Music Computing (SMC) Conference*, 2019.

[120] F. Fontana, R. Paisa, R. Ranon, and S. Serafin, "Multisensory plucked instrument modeling in Unity3D: From keytar to accurate string prototyping," *Applied Sciences*, vol. 10, 2020.

[121] R. Msallam, S. Dequidt, S. Tassart, and R. Causse, "Physical model of the trombone including non-linear propagation effects," in *ISMA: International Symposium of Music Acoustics*, 1997.

[122] F. Eichas, S. Möller, and U. Zölzer, "Block-oriented gray box modeling of guitar amplifiers," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, 2017.

[123] J. D. Parker, F. Esqueda, and A. Bergner, “Modelling of nonlinear state-space systems using a deep neural network,” in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[124] D. E. Knuth, “Computer programming as an art,” *Communications of the ACM*, vol. 17, no. 12, pp. 667–673, 1974.

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

Part VII

Papers

Paper A

Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Silvin Willemsen, Nikolaj Andersson, Stefania Serafin
and Stefan Bilbao

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp. 275–280, 2019.

Abstract

In this paper, implementation, instrument design and control issues surrounding a modular physical modelling synthesis environment are described. The environment is constructed as a network of stiff strings and a resonant plate, accompanied by user-defined connections and excitation models. The bow, in particular, is a novel feature in this setting. The system as a whole is simulated using finite difference (FD) methods. The mathematical formulation of these models is presented, alongside several new instrument designs, together with a real-time implementation in JUCE using FD methods. Control is through the Sensel Morph.

1 Introduction

Physical models for sound synthesis have been researched for several decades to mathematically simulate the sonic behaviour of musical instruments and everyday sounds. Various techniques and methodologies have developed, ranging from mass-spring models [1, 2, 3] to modal synthesis [4] and waveguide based models [5]. The latter two techniques may be viewed as numerical simulation techniques applied to the systems of partial differential equations (PDEs). These equations define the dynamics of a musical instrument, either real or imagined.

Mainstream time-domain simulation techniques, such as finite difference (FD) methods, were first applied to the case of string vibration by Ruiz [6] and Hiller and Ruiz [7, 8], and then later by other authors [9] including, most notably Chaigne [10] and Chaigne and Askenfelt [11]. The general use of finite-difference schemes (FDSs) in sound synthesis is described in [12]. Modularized physical modelling sound synthesis, whereby the user may construct a virtual instrument using basic canonical components dates back to the work of Cadoz and collaborators [1, 2, 3]. It has been also used as a design principle in the context of FD methods [13, 14, 15], where the canonical elements are strings and plates, with a non-linear connection mechanism. Though computational cost of such methods is high, standard computing power is now approaching a level suitable for real-time performance for simpler systems.

We are interested in bridging the gap between large-scale modular physical modelling synthesis and sonic interaction design [16], to be able to play with such simulations in real-time. Specifically, we are interested in using the expressivity of the Sensel Morph [17] to control our simulations, using both percussive and bowing excitations. Our ultimate goal is to create models that are both mathematically accurate and efficient. This goal is nowadays possible thanks to improvements in hardware and software technologies for sound synthesis, yet it has rarely been achieved. The ultimate goal is to provide a modular efficient synthesizer based on accurate simulations, where real-time expressivity can also be achieved. This synthesizer has already been informally evaluated by composers and sound designers, who appreciated the current sonic palette.

This paper is structured as follows: Section 2 describes the physical models used in the implementation and Section 3 shows a general description of the FD methods used to digitally implement these models. Furthermore, Section 4 elaborates on the real-time implementation, Section 5 shows several different configurations of the physical models inspired by real musical instruments, Section 6 will present the results on CPU usage and evaluation and discuss this and finally, in Section 7, some concluding remarks appear.

2 Models

In this section, the PDEs for the damped stiff string and plate will be presented. The notation used will be the one found in [12] where the subscript for state variable u denotes a single derivative with respect to time t or space x respectively. Furthermore, to simplify the presented physical models, non-dimensionalization (or scaling) will be used [12].

2.1 Stiff string

A basic model of the linear transverse motion of a string of circular cross section may be described in terms of several parameters: the total length L (in m), the material density ρ (in kg·m^{-3}), string radius r (in m), Young's modulus E (in Pa), tension T (in N), and two loss parameters σ_0 and σ_1. The PDE for a damped stiff string may be written as [12]

$$u_{tt} = \gamma^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \tag{1}$$

In this representation, spatial scaling has been employed using a length L, so the solution $u = u(x,t)$ is defined for $t \geq 0$ and for dimensionless coordinate $x \in [0,1]$. Furthermore, parameters $\gamma = \sqrt{T/\rho\pi r^2 L^2}$ and $\kappa = \sqrt{Er^2/4\rho L^4}$ and have units s^{-1}.

In this work, the string is assumed clamped at both ends, so that

$$u = u_x = 0 \quad \text{where} \quad x = \{0,1\}. \tag{2}$$

A model of a bowed string [12] may be incorporated into (1) as

$$u_{tt} = \ldots - \delta(x - x_\text{B}) F_\text{B} \phi(v_\text{rel}), \quad \text{with} \tag{3a}$$

$$v_\text{rel} = u_t|_{(x=x_\text{B})} - v_\text{B}, \tag{3b}$$

where $F_\text{B} = f_\text{B}/M_\text{s}$ is the excitation function (in m/s^2) with externally-supplied bowing force $f_\text{B} = f_\text{B}(t)$ (in N) and total string mass $M_\text{s} = \rho\pi r^2 L$ (in kg).

The relative velocity v_{rel} is defined as the difference between the velocity of the string at bowing point x_{B} and the externally-supplied bowing velocity $v_{\text{B}} = v_{\text{B}}(t)$ (in m/s) and ϕ is a dimensionless friction characteristic, chosen here as [12]

$$\phi(v_{\text{rel}}) = \sqrt{2a}v_{\text{rel}}e^{-av_{\text{rel}}^2+1/2}. \tag{4}$$

Furthermore, $\delta(x - x_{\text{B}})$ is a spatial Dirac delta function selecting the bowing location $x = x_{\text{B}}$. The single bowing point can be extended to a bowing area [12]. More detailed models of string dynamics, again in a bowed string context, have been proposed by Desvages [18].

Another, and more simple way to excite the string is by extending Equation (1) to

$$u_{tt} = \ldots + E_{\text{e}}F_{\text{e}} \tag{5}$$

using an externally-supplied distribution function $E_{\text{e}} = E_{\text{e}}(x)$ and excitation function $F_{\text{e}} = F_{\text{e}}(t)$. In this case, the excitation region is allowed to be of finite width.

2.2 Plate

Under linear conditions, a rectangular plate of dimensions L_x and L_y may be parameterized in terms of density ρ (in kg· m^{-3}), thickness H (in m), Young's modulus E (in Pa) and a dimensionless Poisson's ratio ν, as well as two loss parameters σ_0 and σ_1.

In terms of dimensionless spatial coordinates x and y scaled by $\sqrt{L_xL_y}$, the equation of motion of a damped plate is a variant of the Kirchhoff model [19]

$$u_{tt} = -\kappa^2\Delta\Delta u - 2\sigma_0 u_t + 2\sigma_1\Delta u_t. \tag{6}$$

Here, $u(x, y, t)$ is the transverse displacement of the plate as a function of dimensionless coordinates $x \in [0, \sqrt{a}]$, $y \in [0, 1/\sqrt{a}]$, where $a = L_x/L_y$ is the plate aspect ratio, as well as time t. Furthermore, Δ represents the 2D Laplacian [12]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{7}$$

The stiffness parameter κ, with dimensions of s^{-1}, is defined by $\kappa = \sqrt{D/\rho H L_x^2 L_y^2}$ where $D = EH^3/12\left(1 - \nu^2\right)$. As in the case of the stiff string, we chose to use clamped boundary conditions:

$$u = \mathbf{n} \cdot \nabla u = 0 \tag{8}$$

over any plate edge with outward normal direction $\mathbf{n}$ and where ∇u is the gradient of u.

2.3 Connections

Adding connections between different physical models, further referred to as elements, adds another term to Equation (3a), (5) or (6). Assuming that element α is a stiff string and β is a plate, the following terms are added to the aforementioned equations:

$$u_{tt} = ... + E_{\mathrm{c},\alpha} F_\alpha, \tag{9a}$$

$$u_{tt} = ... + E_{\mathrm{c},\beta} F_\beta, \tag{9b}$$

with force-functions $F_\alpha = F_\alpha(t)$ and $F_\beta = F_\beta(t)$ (in m/s^2) and distribution functions $E_{\mathrm{c},\alpha}$ and $E_{\mathrm{c},\beta}$ which have chosen to be highly localised in our application and reduce to $\delta(x - x_{\mathrm{c},\alpha})$ and $\delta(x - x_{\mathrm{c},\beta}, y - y_{\mathrm{c},\beta})$ respectively, but can be extended to be connection areas [13]. We use the implementation as presented in [13] where the connection between two elements is a non-linear spring. The forces it imposes on the elements it connects are defined as

$$F_\alpha = -\omega_0^2 \eta - \omega_1^4 \eta^3 - 2\sigma_\times \dot{\eta}, \tag{10a}$$

$$F_\beta = -\mathcal{M} F_\alpha, \tag{10b}$$

where ω_0 and ω_1 are the linear (in s^{-1}) and non-linear (in (m·s)$^{-1/2}$) frequencies of oscillation respectively, $\sigma_\times$ is a damping factor (in s^{-1}), $\mathcal{M}$ is the mass ratio between the two elements and η is the relative displacement between the connected elements at the point of connection (in m). Lastly, the dot above η denotes a derivative with respect to time.

3 Finite-Difference Schemes

To be able to digitally implement the continuous equations shown in the previous section, they need to be approximated. In this section, a high-level review of a finite difference approximation to a connected system of strings and plates is presented. For more technical details, see [13].

In the case of the stiff string, state variable $u(x,t)$ can be discretised at times $t = nk$, where $n \in \mathbb{N}$ and $k = 1/f_\mathrm{s}$ is the time step (at sample-rate f_s) and locations $x = lh$, with $l \in [0, \ldots, N]$ where the total number of points is $N + 1$ and grid spacing h. We can now write the discretised state variable as u_l^n, representing an approximation to $u(x,t)$.

In the case of the plate, $u(x,y,t)$ is discretised to $u_{(l,m)}^n$ using $x = lh$ where $l \in [0, \ldots, N_x]$ with $N_x + 1$ being the total horizontal number of points and $y = mh$ where $m \in [0, \ldots, N_y]$ with $N_y + 1$ being the total vertical number of points.

In a general sense, when discretising PDEs as presented in Equations (1) and (6), we will need to solve for $\mathbf{u}^{n+1}$, i.e., $\mathbf{u}$ at the next time step, where $\mathbf{u}$ is a vector of size $N-1$ containing values of u_l $\forall l$ for a string and $(N_x-1)(N_y-1)$ containing values of $u_{(l,m)}$ $\forall (l,m)$ for a plate. Note that the vector sizes are smaller than the total number of grid points as we do not include the values at the boundaries (which are always 0). For a PDE expressed as a function of u_{tt}, its FDS will be of the form

$$\mathbf{u}^{n+1} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}^n, \tag{11}$$

where

$$K = \frac{k^2}{1+\sigma_0 k}, \tag{12}$$

and $\boldsymbol{\mathcal{F}}^n$ is a combination of the discretised PDE (excluding terms containing u^{n+1}) together with connection and excitation terms.

3.1 Stiff String

In the case of the stiff string, $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (1) $\mathbf{f}_\alpha^n$, connection term (9a) and bowing (3a)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha} F_\alpha^n - \mathbf{J}(x_\mathrm{B}^n) F_\mathrm{B}^n \phi(v_\mathrm{rel}), \tag{13a}$$

or excitation (5) term

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\alpha^n + \mathbf{E}_{\mathrm{c},\alpha} F_\alpha^n + \mathbf{E}_\mathrm{e} F_\mathrm{e}^n, \tag{13b}$$

where $\mathbf{E}_{\mathrm{c},\alpha}$ contains the discretised distribution function for the connection ($1/h$ at connection index $l_{\mathrm{c},\alpha}$, rest 0's [12]), $\mathbf{E}_\mathrm{e}$ contains the discretised distribution function for the excitation (which will be presented in Equation (25) in the next section) and $\mathbf{J}(x_\mathrm{B}^n)$ is a spreading operator containing the discretised bowing distribution ($1/h$ at time-varying bowing position x_B). If x_B is between grid points, cubic interpolation is used to spread the bow-force over neighbouring grid points [12]. All vectors are columns of size $N-1$.

It can be useful to talk about the *region of operation* of a FDS in terms of a 'stencil'. A stencil describes the number of grid points needed to calculate a single point at the next time step. The stiff string FDS has a stencil of 5 grid points. In other words, two grid points at either side of l – and l itself – are necessary to calculate u_l^{n+1}. See Figure 1 for a visualisation of this.

In order for the scheme to be stable, the grid spacing needs to abide the following condition [12]

$$h \geq \sqrt{\frac{\gamma^2 k^2 + 4\sigma_1 k + \sqrt{(\gamma^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}. \tag{14}$$

The closer h is to this limit, the higher the quality of the implementation. The number of points N can then be calculated using

$$N = \text{floor}\left(\frac{1}{h}\right). \tag{15}$$

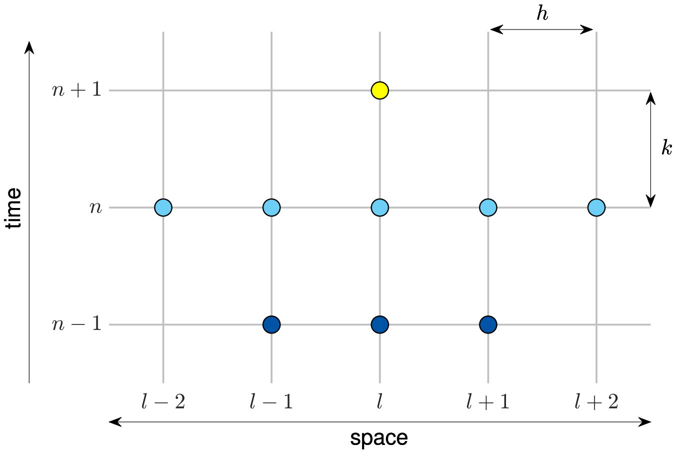


Fig. 1: Stencil for a stiff string FDS with grid spacing h and time step k. The point l at the next time step (yellow) is calculated using 5 points at the current time step (blue) and 3 at the previous time step (dark blue).

3.2 Plate

In the case of the plate, $\mathbf{u}$ is a column vector of concatenated vertical 'strips' of the plate state as in [13] of size $(N_x-1)(N_y-1)$ and $\boldsymbol{\mathcal{F}}^n$ in Equation (11) is a combination of the discretised PDE (6) $\mathbf{f}_\beta^n$ and connection term (9b)

$$\boldsymbol{\mathcal{F}}^n = \mathbf{f}_\beta^n + \mathbf{E}_{\mathrm{c},\beta} F_\beta^n. \tag{16}$$

Here, $\mathbf{E}_{\mathrm{c},\beta}$ contains the discretised distribution function for the connection ($1/h^2$ at connection index $(l_{\mathrm{c},\beta}, m_{\mathrm{c},\beta})$, rest 0's [13]) and is a column vector of size $(N_x-1)(N_y-1)$. For the plate, the stencil will consist of 13 grid points as can be seen in Figure 2.

The grid spacing needs to abide the following condition [13]

$$h \geq 2\sqrt{k\left(\sigma_1^2 + \sqrt{\kappa^2+\sigma_1^2}\right)}, \tag{17}$$

(again, the closer h is to this limit the better) from which N_x and N_y can be derived using

$$N_x = \text{floor}\left(\frac{\sqrt{a}}{h}\right) \quad \text{and} \quad N_y = \text{floor}\left(\frac{1}{h\sqrt{a}}\right). \tag{18}$$

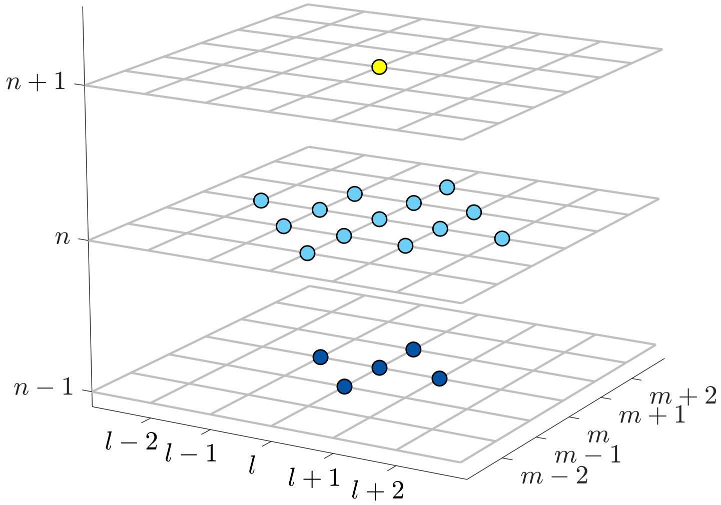


Fig. 2: Stencil for a plate FDS. The point (l, m) at the next time step (yellow) is calculated using 13 points at the current time step (blue) and 5 at the previous time step (dark blue).

3.3 Connections

In the following, we discretise the equations in (10) as shown in [13]. However, as these equations are not expressed as a function of u_{tt}, their FDS counterpart will be different. Moreover, instead of solving for $\mathbf{u}^{n+1}$, we need to solve for η^{n+1}, i.e., the relative displacement at the next time step, which will be in the form of

$$\eta^{n+1} = p^n F_\alpha^n + r^n \eta^{n-1}, \tag{19}$$

where $p^n = p(\eta^n)$ and $r^n = r(\eta^n)$ are functions of the relative displacement η if $\omega_1 \neq 0$ and constants if $\omega_1 = 0$. Again, assuming that element α is a stiff string and β is a plate, η can be calculated using

$$\eta^n = h_\alpha u^n_{\alpha, l_{\text{c},\alpha}} - h_\beta^2 u^n_{\beta,(l_{\text{c},\beta}, m_{\text{c},\beta})}. \tag{20}$$

In other words, this is the difference between the state of element α at $l_{\text{c},\alpha}$ and the state of element β at $(l_{\text{c},\beta}, m_{\text{c},\beta})$ scaled by their respective (for plates, squared) grid spacings h_α and h_β. The next step is to obtain F_α^n, which can be used to easily calculate F_β^n. We first obtain values for $\mathbf{u}^{n+1}$ by solving (11) using (13a), (13b) or (16) (without the connection term!) for a string or plate respectively. As, at this point, no connection forces have been added yet, this state will be referred to as an intermediate state $\mathbf{u}^{\text{I}}$. This intermediate state can be used to obtain η^{n+1} using (20)

$$\eta^{n+1} = h_\alpha(u^{\mathrm{I}}_{\alpha,l_{\mathrm{c},\alpha}} + K_\alpha F^n_\alpha) - \Big[h^2_\beta(u^{\mathrm{I}}_{\beta,(l_{\mathrm{c},\beta},m_{\mathrm{c},\beta})} + K_\beta F^n_\beta)\Big], \tag{21}$$

where K_α and K_β are as described in (12) using the damping coefficient σ_0 of their respective element. This can then be set equal to (19). Using Equation (10b), solving for F_α yields

$$F^n_\alpha = \frac{r^n\eta^{n-1} - (h_\alpha u^{\mathrm{I}}_{\alpha,l_{\mathrm{c},\alpha}} - h^2_\beta u^{\mathrm{I}}_{\beta,(l_{\mathrm{c},\beta},m_{\mathrm{c},\beta})})}{h_\alpha K_\alpha + \mathcal{M}h^2_\beta K_\beta - p^n}. \tag{22}$$

4 Implementation

In this section, we elaborate more on the chosen values for the parameters described in the previous two sections and present the system architecture of the real-time application. The values for most parameters have been arbitrarily chosen and can – as long as they satisfy the conditions in Equations (14) and (17) – be changed. We used C++ along with the JUCE framework [20] for implementing the physical models and connections in real-time. The main hardware used was a MacBook Pro with a 2.2 GHz Intel Core i7 processor.

4.1 Stiff String

As many string properties stay constant, we chose to set the following parameters directly, rather than calculating them from their physical properties: $\kappa = 2$, $\sigma_0 = 1$, $\sigma_1 = 0.005$. An interesting parameter to make dynamic is the fundamental frequency f_0 (in s^{-1}) of the string. According to [12], the fundamental frequency can be approximately calculated using

$$f_0 \approx \frac{\gamma}{2}. \tag{23}$$

However, as the grid spacing h is dependent on the wave speed γ according to the condition found in (14), we must put a lower limit on the number of points N if we plan to dynamically increase γ.

Another way to change frequency is to add damping to the model at specific points acting as a (simplified) fretting finger. The advantage of this is that the condition (14) will never be violated. On top of this, a tapping sound will be introduced when fretting the string making it more realistic than changing the wave speed. If the string is fretted at single location $x_{\mathrm{f}} \in [0,1]$ and $l_{\mathrm{f}} = \mathrm{floor}(x_{\mathrm{f}}/h)$ we use

$$u^n_l = \begin{cases} 0, & l = l_{\mathrm{f}} - 1 \lor l = l_{\mathrm{f}} \\ (1-\alpha^\epsilon_{\mathrm{f}})u^n_l, & l = l_{\mathrm{f}} + 1 \\ u^n_l, & \text{otherwise} \end{cases} \tag{24}$$

where $\alpha_\text{f} = x_\text{f}/h - l_\text{f}$ describes the fractional location of x_f between two grid points. Note that the grid point at the finger location and the grid point before are set to 0 to (recalling the stencil) prevent the states at either side of the finger to influence each other. The disadvantage of using this technique over regular linear interpolation, is that the effect of damping between grid points does not linearly scale to pitch. We thus added $\epsilon = 7$ as a heuristic value to more properly map finger position to pitch.

In some cases, N is fixed to a certain value (as opposed to calculating it from Equations (14) and (15)) for multiple strings of different pitches. Even though some bandwidth will be lost (in the higher frequency range), this will allow the strings to be perfectly tuned to each other.

4.1.1 *Bowed String*

Parameters for the bowed strings abide the following conditions: $|v_\text{B}| \leq 1$ m/s and $0 \leq F_\text{B} \leq 100$ N. It was chosen to discretise Equation (3b) implicitly making it necessary to use an iterative root-finding method such as Newton-Raphson [21].

4.1.2 *Excited string*

If simply excited, we set the distribution function to a raised cosine with width w_e (in grid points)

$$E_\text{e}(l) = \begin{cases} \frac{1-\cos(\frac{2\pi(l-(l_\text{e}-w_\text{e}/2))}{w_\text{e}})}{2}, & l_\text{e} - \frac{w_\text{e}}{2} < l < l_\text{e} + \frac{w_\text{e}}{2} \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

scaled by the excitation function over time with excitation duration d_e (in samples)

$$F_\text{e}(n) = \begin{cases} \frac{1-\cos(\frac{\pi(n-n_\text{e})}{d_\text{e}})}{2}, & n_\text{e} \leq n < n_\text{e} + d_\text{e} \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

A visualisation of this can be found in Figure 3.

4.2 Plate

For the plate, the damping coefficients have been decided to be $\sigma_0 = 0.1$ and $\sigma_1 = 0.005$ and the aspect ratio is set to $a = 2$. The plate stiffness κ has been left as a user parameter to be changed dynamically and will be between the following bounds: $0.1 \leq \kappa \leq 50$ s^{-1}. In Equation (17), the grid spacing is calculated using the maximum value of κ to prevent stability issues. Using a sample rate of 44,100 Hz results in a plate with dimensions $N_x = 20$ and $N_y = 10$ (in grid points).

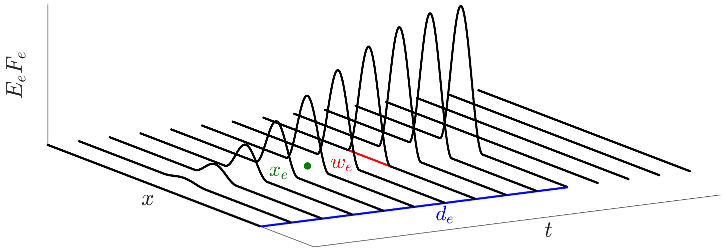


Fig. 3: A visualisation of the excitation used in our implementation presented in Equation (5). The location of excitation x_e is shown in green, excitation width w_e in red and excitation duration d_e in blue (also see Equations (25) and (26)).

4.3 Connections

Increasing $\omega_1 \gtrsim 100,000$ $(\text{m}{\cdot}\text{s})^{-1/2}$ while keeping $0 < \omega_0 \lesssim 100$ s^{-1} will cause audible non-linear behaviour, such as pitch-glides and rattling sounds. These effects will be more dominant when the plate stiffness is higher. In our implementation we set $\omega_0 = 100$ s^{-1} and $\omega_1 = 100,000$ $(\text{m}{\cdot}\text{s})^{-1/2}$. The spring-damping $\sigma_\times = 1$ s^{-1} is kept to a minimum ($0 \leq \sigma_\times \leq 10$ s^{-1}).

4.4 System Architecture

The system architecture can be seen in Figure 4. The top box denotes the Sensel Morph (described in more detail in the next section) controlling the application, and the white boxes show the different classes or components of the application. The black arrows indicate instructions that one class can give to another and the hollow arrows show data flows between classes. All arrows are accompanied by a coloured box indicating which thread the instruction / dataflow is associated with and at what rate this thread runs.

The lowest priority thread, the graphics-thread, is shown by green boxes and runs at 15 Hz. This draws the states of the strings, connections and the plate on the screen.

Checking and retrieving the Sensel state happens at a rate of 150 Hz and is denoted by blue boxes. The parameters that the user controls by means of the Sensel, such as bowing position, force and velocity, will be updated in the models at this rate as well.

The highest priority thread is the audio-thread and runs at commonly-used sample rate 44,100 Hz. The main application gives an 'update' (u) instruction to the instrument, which in turn updates the FDSs in its strings and plate. After the FDS update is done, the intermediate state at the connection points $u^\text{I}_{x_\text{c}}$ (where $x_\text{c} = l_{\text{c},\alpha}$ for the string or $x_\text{c} = (l_{\text{c},\beta}, m_{\text{c},\beta})$ for the plate) are sent to

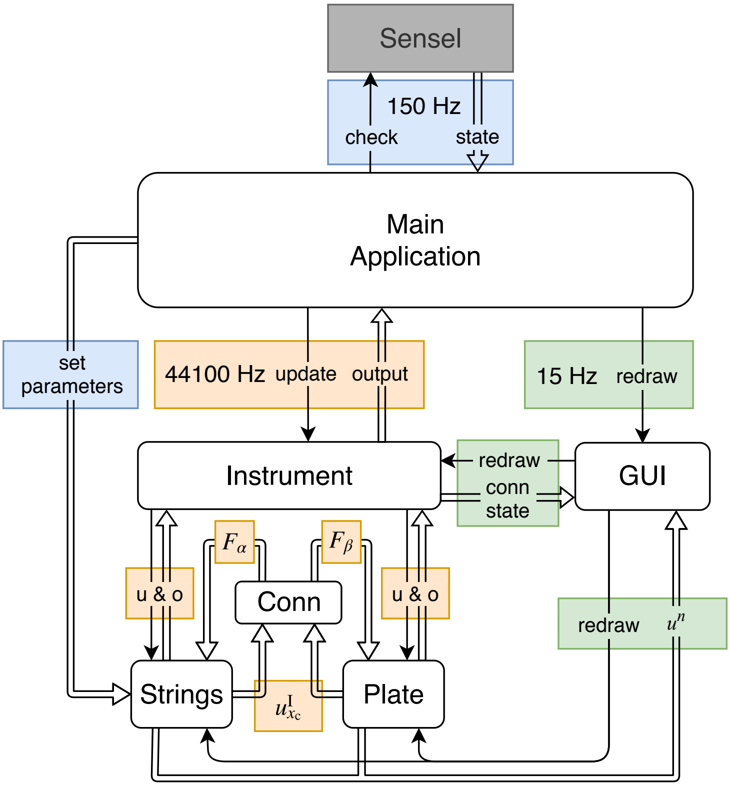


Fig. 4: System architecture flowchart. See Section 4.4 for a thorough explanation.

the connection (Conn) class which calculates the force-functions F_α and F_β. These values are then sent back to the string and plate classes and added to their respective states after which their outputs (o) (at arbitrary points) are sent back to the main application. See Algorithm 1 for this 'order of calculation'.

5 Instruments and User Interaction

In this section, the Sensel Morph (or simply Sensel) and user interface will be described in more detail. Furthermore, several configurations of strings, plates and connections that are inspired by real-life instruments will be presented. A demonstration of one of the instruments can be found in [22].

5.1 Sensel Morph

The Sensel Morph is a highly accurate touch controller that senses position and force of objects [17] (see Figure 5). We use the Sensel as an expressive interface

```
while application runs do
    for all elements do
        calculate intermediate state $\mathbf{u}^\text{I}$ using previous state values (as in Equation (11))
            $\mathbf{u}_\text{s}^\text{I} = 2\mathbf{u}^n - \mathbf{u}^{n-1} + K\boldsymbol{\mathcal{F}}$
    end
    if element is excited/bowed then
        calculate excitation term $\mathbf{E}$ and add to intermediate state of the element
            $\mathbf{u}_\text{s+e}^\text{I} = \mathbf{u}_\text{s}^\text{I} + \mathbf{E}$
    end
    for all connections do
        calculate connection forces and add connection term $\mathbf{C}$ to elements to obtain the state at the next time step
            $\mathbf{u}_\text{s+e+c}^{n+1} = \mathbf{u}_\text{s+e}^\text{I} + \mathbf{C}$
    end
    update state vectors
        $\mathbf{u}^{n-1} = \mathbf{u}^n$
        $\mathbf{u}^n = \mathbf{u}_\text{s+e+c}^{n+1}$
    increment time step
        $n$++
end
```

Algorithm 1: Pseudocode showing the correct order of calculation. The subscripts for state vector $\mathbf{u}$ shows what it consists of ('s' for previous state, 'e' for excitation and 'c' for connection).

for interacting with the instrument configurations. Right above the touch-sensitive area, the Sensel contains an array of 24 LEDs that can be controlled from the application.

5.2 User interface

Strings are shown as coloured paths (see Figure 6 for a descriptive visualisation). The state $\mathbf{u}^n$ of the string is visualised using the vertical displacement of the paths. Bowed strings are shown in cyan on the top left. The bow is shown as a yellow rectangle and moves on interaction. The fretting position is shown as a yellow circle. Plucked strings are shown in purple in the top right, underneath which the sympathetic strings are shown in light green. The plate is shown in the bottom using a grid of rectangles (clamped grid points are not shown). Its state is visualised using a grey-scale. Furthermore, connections are shown using orange circles/squares for the points of connection and dotted

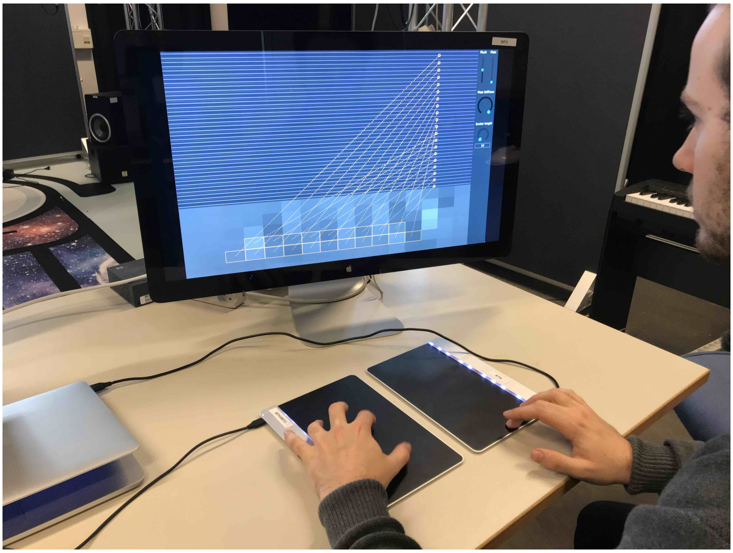


Fig. 5: Player using the Sensel Morphs to interact with one of the instruments.

lines between these points. Lastly, all parameters that are controlled by the mouse such as output-level and plate-stiffness are located in a column on the right side of the application.

5.3 Instruments

We subdivide string-elements into three types: bowed, plucked and sympathetic strings. All strings will be connected to one plate acting as an instrument body of which the user can control the plate-stiffness. Furthermore, the user can change the output-level of each element type. Apart from these parameters, which are controlled by the mouse, the instruments are fully controlled by two Sensels. The instruments we have chosen as our inspiration are the sitar, the hammered dulcimer and the hurdy gurdy.

5.3.1 Bowed Sitar

The sitar is originally an Indian string instrument that has both fretted strings and sympathetic strings. Instead of plucking the fretted strings, we extended the model to bow them. Our implementation consists of 2 bowed strings (tuned to A3 and E4), 13 sympathetic strings (tuned according to [23]) and 5 plucked strings (tuned A3-E4 following an A-major scale) as it is also possible to strum the sympathetic strings. See Figure 6 for a visual of the implementation. One Sensel is vertically subdivided into two sections; one for each bowed string. The first finger registered by the Sensel is mapped to a bow and the second is mapped to a fretting finger controlling pitch. The horizontal position of both fingers is visualised using the Sensel's LED array. The frets are not implemented as such (the pitch is continuous), but they are visualised for reference. The horizontal position of the first finger is mapped to the bowing position on the string, the vertical velocity to the bow velocity v_{B} and the finger force is linked to the excitation function F_{B} (both in Equation (3a)). The other Sensel is subdivided into 5 sections mapped to the plucked strings. These sections are visualised by the LED array for reference.

The mass ratio for the bowed/plucked string to plate connections has been set to $\mathcal{M} = 2$ and ratio for the sympathetic string to plate connections has been set to $\mathcal{M} = 0.5$ to increase the effect that the playable strings have on the sympathetic strings.

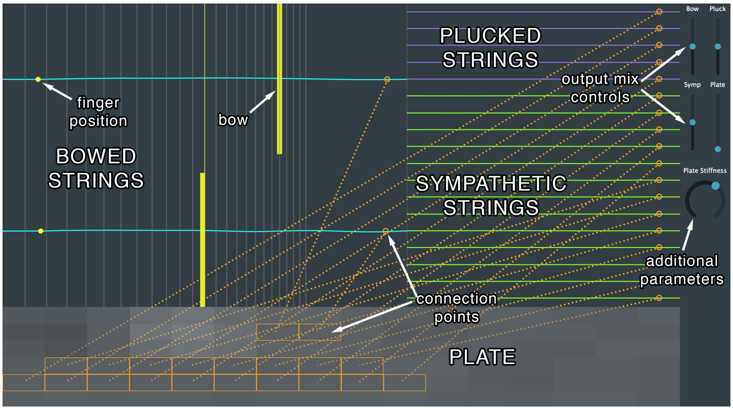


Fig. 6: The bowed sitar application. The descriptions of the different elements and other objects are shown in the image, but will (naturally) not be visible in the application.

5.3.2 *Hammered Dulcimer*

The hammered dulcimer is an instrument that can be seen as an 'open piano' where the musician has the hammers in their hand. Just like the piano, the strings are grouped in pairs or triplets that are played simultaneously. In our implementation, we have 20 pairs of plucked strings. Even though most hammered dulcimers have more strings, we decided that this configuration has the highest number of strings while maintaining playability. One of each pair is connected to the plate which slightly detunes it, creating a desired 'chorusing' effect. See Figure 7 for a visual of the implementation. In order for the excitation to more resemble a strike of a hammer than a pluck, the contents of the cosine in (26) will be multiplied by 2 for the excitation to have a less abrupt ending, something desired for a hammered interaction. Moreover, the excitation-length can be changed to simulate short and long hammer-times.

The Sensels are placed vertically next to each other (see Figure 5). The pair with the lowest frequency will then be located in the bottom right and the highest in the top left, as in the real instrument. As with the plucked strings of the bowed sitar, the LED array is used to visualise the way that the Sensel is subdivided, which is especially useful here as one Sensel controls 10 string-pairs.

The mass ratio is set relatively high ($\mathcal{M} = 100$) to amplify the non-linear interaction between the strings and the detuning of the strings connected to the plate.

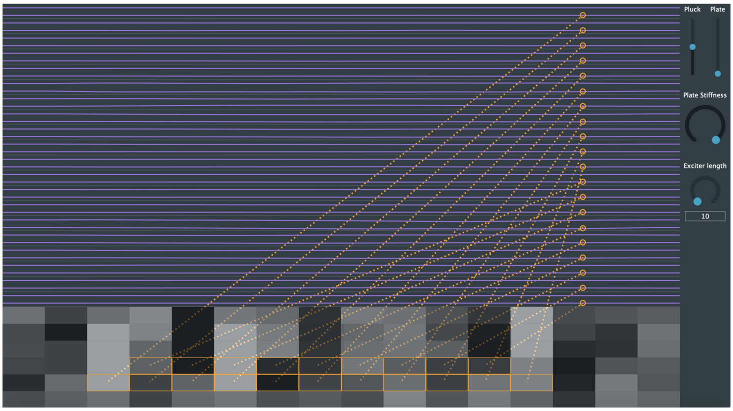


Fig. 7: The hammered dulcimer application.

5.3.3 *Hurdy Gurdy*

The hurdy gurdy is an instrument that consists of bowed and sympathetic strings. The bowing happens through a rosined wheel attached to a crank and bows these strings as the crank is turned. It is possible to change the pitch of a few bowed strings - the melody strings - using buttons that press tangent pins on the strings at different positions. The other strings, referred to as drone strings, are mostly tuned lower than the melody strings and provide the bass frequencies of the instrument. The musician can place the bowed strings on rests that keep the wheel from interacting with it.

Our implementation consists of 5 bowed strings subdivided into 2 drone strings tuned to A2, E3 and 3 melody strings tuned to A3, E4 and A4 and 13 sympathetic strings tuned the same way as the sympathetic strings in bowed sitar. Furthermore, the mass ratios have been set the same as in the bowed sitar application. See Figure 8 for a visual of the implementation.

The Sensel is vertically subdivided into 5 rows that control whether the strings are placed on the wheel. The bowing velocity is mapped to the average pressure of the fingers. The fundamental frequency (in the model $\gamma/2$) of the melody-strings is changed by a Sensel with a piano-overlay acting as a midi controller. A demonstration of this instrument can be found in [22]. It is interesting to note here that the sympathetic strings that are in tune with the harmonics of the bowed strings resonate most, which is expected to happen in the real world as well.

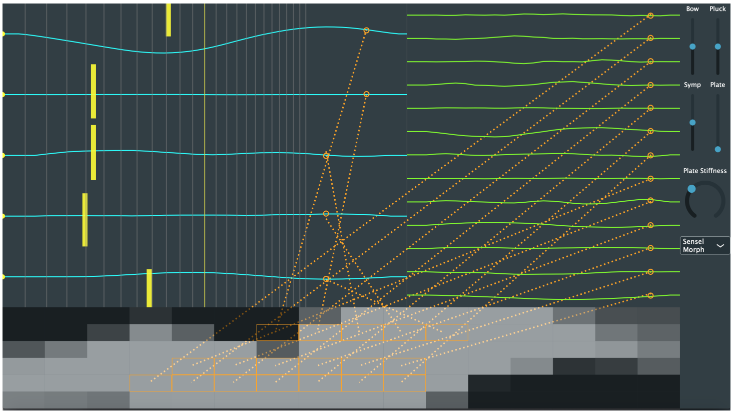


Fig. 8: The hurdy gurdy application.

6 Results and Discussion

Table 1 shows the CPU usage (on the same MacBook Pro 2.2 GHz i7 as described before) for the three instruments presented in the previous section. As the Sensel thread contributes a negligible amount to the CPU usage, this is not shown in the table.

Application	No Sound	No Graphics	Total
Bowed Sitar	32	63	85
Dulcimer	30	66	85
Hurdy Gurdy	28	58	78

Table 1: CPU usage (in %) for the instruments found in Section 5. Values show usage of one (virtual) thread and have been taken as an average (with a margin of ~5%) over a short period of time. The two middle columns show usage when the sound or graphics thread has been turned off.

As can be seen from the table, all instruments use about the same amount of CPU and none of them have audible dropouts (CPU $< 100\%$). It can be observed that the graphics use about 20% of the CPU, indicating that there is still much room to increase the complexity of the instrument-configurations before dropouts will occur. On the other hand, should the instruments be used in parallel with other audio applications or plug-ins, the CPU usage has to be greatly reduced. The first step towards this would be to vectorise the FDSs using AVX instructions.

While our instruments have been not formally evaluated yet, we have performed some qualitative evaluations with sound and music computing experts. The goals of the evaluations were to explore the playability of the instrument, sonic quality and intuitiveness of control. These evaluations showed that especially the bowing interaction feels intuitive and creates a natural sound. The overall sound of the instruments was generally judged to be interesting, but not “sounding like a real-life instrument". This makes sense, as we did not seek to perfectly model each instrument, but rather used them as an inspiration for the configurations of the physical models. The next step for sound quality would be to replace the thin plate with a more realistic element, such as a wooden instrument body.

7 Conclusion

In this paper, a real-time modular physical modelling synthesis environment structured as a network of connected strings and plates has been presented. Several instruments have been created in the context of this environment which can be played by a pair of Sensel Morphs allowing for highly expressive control of these instruments. Informal evaluations with professional musicians have confirmed that the interaction is found natural and the output sound interesting. Further steps to improve this project are to optimise the algorithm and to replace the plate with a more realistic instrument body.

8 Acknowledgments

This work is supported by NordForsk’s Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

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

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

[3] C. Cadoz, A. Luciani, and J. L. Florens, "Cordis-anima: a modeling and simulation system for sound and image synthesis: the general formalism," *Computer music journal*, vol. 17, no. 1, pp. 19–29, 1993.

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

[5] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

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

[7] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[8] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[9] R. Bacon and J. Bowsher, "A discrete model of a struck string," *Acustica*, vol. 41, pp. 21–27, 1978.

[10] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[11] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[12] S. Bilbao, *Numerical Sound Synthesis, Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley and Sons, Ltd, 2009.

[13] ——, "A modular percussion synthesis environment," *Proc. of the 12th Int. Conf. on Digital Audio Effects (DAFx-09)*, 2009.

[14] S. Bilbao, A. Torin, P. Graham, J. Perry, and G. Delap, "Modular physical modeling synthesis on gpu," in *Proceedings of the 40th International Computer Music Conference and the 11th Sound and Music Computing Conference*, 2013.

[15] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," *Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx-15)*, 2015.

[16] K. Franinović and S. Serafin, *Sonic interaction design*. Mit Press, 2013.

[17] Sensel Inc. (2018) Sensel morph. [Online]. Available: https://sensel.com/

[18] C. Desvages and S. Bilbao, "Two-polarisation physical model of bowed strings with nonlinear contact and friction forces, and application to gesture-based sound synthesis," *Applied Sciences*, 2016.

[19] K. Graff, *Wave Motion in Elastic Solids*. New York, New York: Dover, 1975.

[20] JUCE ROLI. (2019) JUCE. [Online]. Available: https://juce.com/

[21] J. Wallis, *A treatise of algebra, both historical and practical*. London, 1685.

[22] S. Willemsen. (2019) Hurdy gurdy demo. [Online]. Available: https://www.youtube.com/watch?v=BkxLji2ap1w

[23] Joe Rizzo. (2015) How to tune a sitar. [Online]. Available: http://www.joerizzo.com/sitar/

Paper B

Physical Models and Real-Time Control with the Sensel Morph

Silvin Willemsen, Stefan Bilbao, Nikolaj Andersson
and Stefania Serafin

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp. 95–96, 2019.

Abstract

In this demonstration we present novel physical models controlled by the Sensel Morph interface.

1 Introduction

The Sensel Morph is a highly accurate touch controller that senses position and force of objects [1]. Figure 1 shows one player interacting with two Sensel Morph devices to interact with the developed physical models. We use the Sensel as an expressive interface for interacting with different physical models described in a companion paper accepted to SMC 2019. Right above the touch-sensitive area, the Sensel contains an array of 24 LEDs that can be controlled from the application.

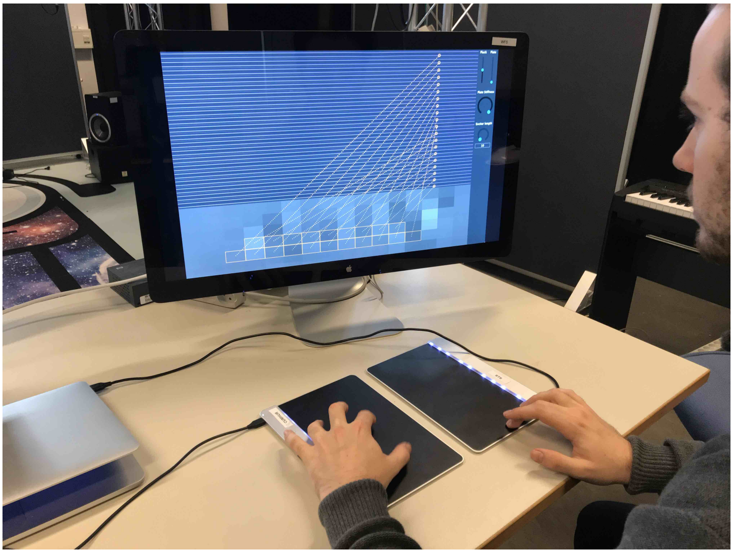


Fig. 1: Player using the Sensel Morphs to interact with one of the instruments.

Strings are shown as coloured paths (see Figure 2 for a descriptive visualisation). The state of the string is visualised using the vertical displacement of the paths. Bowed strings are shown in cyan on the top left. The bow is shown as a yellow rectangle and moves while interacting. The fretting position is shown as a yellow circle. Plucked strings are shown in purple in the top right, underneath which the sympathetic strings are shown in light green. The plate is shown in the bottom using a grid of rectangles (clamped grid points are not shown). Its state is visualised using a grey-scale. Furthermore, connections are shown using orange circles/squares for the points of connection and dotted lines between these points. Lastly, all parameters that are controlled by the mouse such as output-level and plate-stiffness are located in a column on the right side of the application.

2 Implemented Instruments

We subdivide string-elements into three types: bowed, plucked and sympathetic strings. All strings will be connected to one plate acting as an instrument body of which the user can control the plate-stiffness. Furthermore, the user can change the output-level of each element type. Apart from these parameters, which are controlled by the mouse, the instruments are fully controlled by two Sensels. The instruments we have chosen as our inspiration are the sitar, the hammered dulcimer and the hurdy gurdy.

2.1 Bowed Sitar

The sitar is originally an Indian string instrument that has both fretted strings and sympathetic strings. Instead of plucking the fretted strings, we extended the model to bow them. Our implementation consists of 2 bowed strings (tuned to A3 and E4), 13 sympathetic strings (tuned according to [2]) and 5 plucked strings (tuned A3-E4 following an A-major scale) as it is also possible to strum the sympathetic strings. Figure 2 shows the visual interface of the implementation. One Sensel is vertically subdivided into two sections; one for each bowed string. The first finger registered by the Sensel is mapped to a bow and the second is mapped to a fretting finger controlling pitch. The horizontal position of both fingers is visualised using the Sensel's LED array. The frets are not implemented as such (the pitch is continuous), but they are visualised for reference.

2.2 Hammered Dulcimer

The hammered dulcimer is an instrument that can be seen as an 'open piano' where the musician has the hammers in their hand. Just like the piano, the strings are grouped in pairs or triplets that are played simultaneously. The interface for the hammered dulcimer can be seen in Figure 3. In our implementation, we have 20 pairs of plucked strings. Even though most hammered dulcimers have more strings, we decided that this configuration has the highest number of strings while maintaining playability. One of each pair is connected to a plate which slightly detunes it, creating a desired 'chorusing' effect. Two Sensel boards are placed vertically next to each other (see Figure 1). The pair with the lowest frequency is located in the bottom right and the highest in the

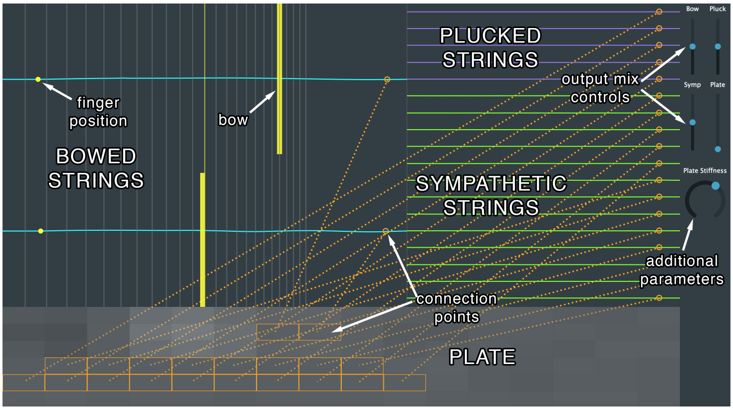


Fig. 2: The bowed sitar application. The descriptions of the different elements and other objects are shown in the image, but will (naturally) not be visible in the application.

top left, as in the real instrument. As with the plucked strings of the bowed sitar, the LED array is used to visualise the way that the Sensel is subdivided, which is especially useful here as one Sensel controls 10 string-pairs.

The mass ratio is set relatively high ($\mathcal{M} = 100$) to amplify the non-linear interaction between the strings and the detuning of the strings connected to the plate.

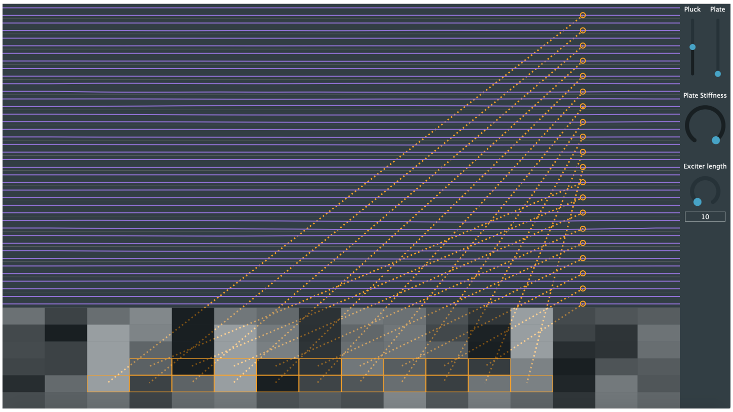


Fig. 3: The hammered dulcimer application.

2.3 Hurdy Gurdy

The hurdy gurdy is an instrument that consists of bowed and sympathetic strings. The bowing happens through a rosined wheel attached to a crank and bows these strings as the crank is turned. It is possible to change the pitch of a few bowed strings - the melody strings - using buttons that press tangent pins on the strings at different positions. The other strings, referred to as drone strings, are mostly tuned lower than the melody strings and provide the bass frequencies of the instrument. The musician can place the bowed strings on rests that keep the wheel from interacting with it. The visual interface can be seen in Figure 4. Our implementation consists of 5 bowed strings subdivided into 2 drone strings tuned to A2, E3 and 3 melody strings tuned to A3, E4 and A4 and 13 sympathetic strings tuned the same way as the sympathetic strings in bowed sitar. Furthermore, the mass ratios have been set the same as in the bowed sitar application.

The Sensel is vertically subdivided into 5 rows that control whether the strings are placed on the wheel.

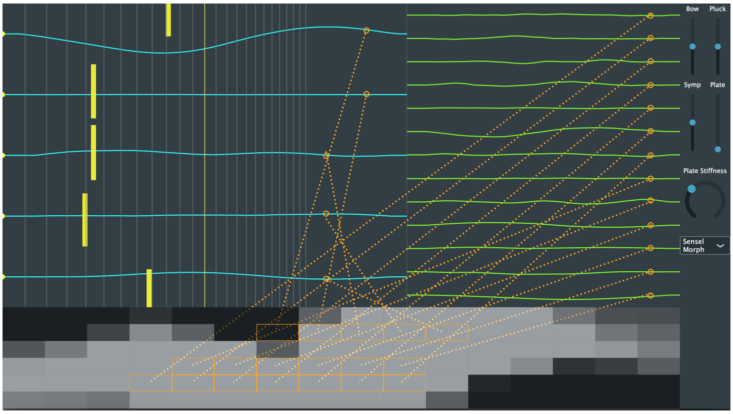


Fig. 4: The hurdy gurdy application.

3 References

[1] Sensel Inc. (2018) Sensel morph. [Online]. Available: https://sensel.com/

[2] Joe Rizzo. (2015) How to tune a sitar. [Online]. Available: http://www.joerizzo.com/sitar/

Paper C

Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao and Stefania Serafin

The paper has been published in the
Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19), pp. 40–46, 2019.

Abstract

The simulation of a bowed string is challenging due to the strongly non-linear relationship between the bow and the string. This relationship can be described through a model of friction. Several friction models in the literature have been proposed, from simple velocity dependent to more accurate ones. Similarly, a highly accurate technique to simulate a stiff string is the use of finite-difference time-domain (FDTD) methods. As these models are generally computationally heavy, implementation in real-time is challenging. This paper presents a real-time implementation of the combination of a complex friction model, namely the elasto-plastic friction model, and a stiff string simulated using FDTD methods. We show that it is possible to keep the CPU usage of a single bowed string below 6 percent. For real-time control of the bowed string, the Sensel Morph is used.

1 Introduction

In physical modelling sound synthesis applications, the simulation of a bowed string is a challenging endeavour. This is mainly due to the strongly non-linear relationship between the bow and the string, through a model of friction. Such friction models can be categorised as static or dynamic; models of the latter type have only recently seen a major effort. As opposed to static friction models, where friction depends only on the relative velocity of the two bodies in contact, dynamic models describe the friction force through a differential equation.

A recently popular dynamic model is the elasto-plastic model, first proposed in [1]. The model assumes that the friction between the two objects in contact is caused by a large ensemble of bristles, each of which contributes to the total friction force. The average bristle deflection is used as an extra independent variable for calculating the friction force. As shown in [2], the elasto-plastic model can be applied to a bowed string simulation and it exhibits a hysteresis loop in the force versus velocity plane due to this multivariable dependency. This is consistent with measurements performed using a bowing machine in [3]. The elasto-plastic model has been thoroughly investigated in a musical context by Serafin et al. in [2, 4, 5].

Regarding bowed string simulations, the first musical non-linear systems, including bowed strings, were presented by McIntyre, et al. in [6]. Smith published the first real-time implementation of the bowed string using a digital waveguide (DW) for the string and a look-up table for the friction model in [7]. Simultaneously, Florens, et al. published a real-time implementation using mass-spring systems for the string and a static friction model for the bow in [8].

The dynamics of musical instruments are generally described by systems of partial differential equations (PDEs). Specialised synthesis methods such as DWs [9] and modal synthesis [10] are derived from particular solutions. Mainstream time-stepping methods such as finite-difference time-domain (FDTD) methods were first proposed in [11, 12, 13], and developed subsequently [14, 15]. In [16] the authors adapted the thermal model proposed by Woodhouse in [3] for real-time applications using a DW for the string implementation and a combination of the DW and FDTD methods for the bowing interaction. In [17, 18], Desvages used FDTD methods for the implementation of the string in two polarizations and a static double exponential friction model introduced in [19]. This was, however, not implemented in real-time. To the best of the authors' knowledge, the only known real-time implementation of any bow model applied to complete FDTD strings was presented in [20] where the soft exponential friction function presented in [14] was used. The current work can be considered an extension of this work.

We are interested in bridging the gap between highly accurate physical models and efficient implementations so that these models can be played in real-time. In this work, we present an implementation of the elasto-plastic friction model in conjunction with a finite-difference implementation of the damped stiff string. Furthermore, we show that it is possible to play the string in real-time using the Sensel Morph controller [21].

This paper is structured as follows. In Section 2, the elasto-plastic bow model in conjunction with a PDE model for a stiff string is described. Discretisation is covered in Section 3, and implementation details appear in Section 4. In Section 5, simulated results are presented and discussed. Some concluding remarks appear in Section 6.

2 Elasto-Plastic Bow Model

Consider a linear model of transverse string vibration in a single polarization, where $u(x,t)$ represents string displacement as a function of time $t \geq 0$, in s, and coordinate $x \in [0,\, L]$ (in m) for some string length L (in m). Using the subscripts t and x to denote differentiation with respect to time and space respectively, a partial differential equation describing the dynamics of the damped stiff string is [14]

$$u_{tt} = c^2 u_{xx} - \kappa^2 u_{xxxx} - 2\sigma_0 u_t + 2\sigma_1 u_{txx}. \tag{1}$$

Here, $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) with tension T (in N), material density ρ (in kg·m^{-3}) and cross-sectional area A (in m^2). Furthermore, $\kappa = \sqrt{EI/\rho A}$ is the stiffness coefficient (in m^2/s) with Young's Modulus E (in Pa) and area moment of inertia I (in m^4). For a string of circular cross section we have radius r (in m), cross-sectional area $A = \pi r^2$ and area moment of inertia

$I = \pi r^4/4$. Lastly, $\sigma_0 \geq 0$ (in s^{-1}) and $\sigma_1 \geq 0$ (in m^2/s) are coefficients allowing for frequency-independent and frequency-dependent damping respectively.

In our implementation we assume simply supported boundary conditions, which are defined as

$$u = u_{xx} = 0 \quad \text{where} \quad x = 0, L\,. \tag{2}$$

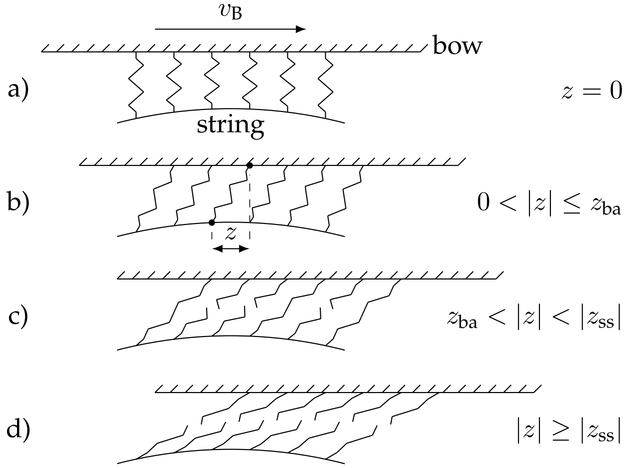

v_B
bow
a)
$z = 0$
string
b)
$0 < |z| \leq z_\mathrm{ba}$
z
c)
$z_\mathrm{ba} < |z| < |z_\mathrm{ss}|$
d)
$|z| \geq |z_\mathrm{ss}|$

Fig. 1: *Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B. a) The initial state is where the average bristle displacement $z = 0$. b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (stick). c) After the break-away displacement z_{ba}, more and more bristles start to 'break'. This is defined as the elasto-plastic regime. d) After all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered.*

As mentioned in the introduction, the elasto-plastic bow model assumes that the friction between the bow and the string is due to a large ensemble of bristles, each of which contributes to the total friction force. See Figure 1 for a graphical representation of this. The bristles are assumed to be damped stiff springs and can 'break' after a given break-away displacement threshold. An extra term can be added to (1) to include the bowing interaction

$$u_{tt} = \ldots - \delta(x - x_\mathrm{B}) f(v, z)/\rho A. \tag{3}$$

Here, the spatial Dirac delta function $\delta(x - x_\mathrm{B})$ (in m^{-1}) allows for the pointwise application of the force f (in N) at externally supplied bowing position $x_\mathrm{B}(t)$ (in m).

In the following we will use the definitions found in [1]. The force f is defined in terms of the relative velocity v (in m/s) and average bristle displacement z (in m) (see Figure 1) as

$$f(v,z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \tag{4}$$

where

$$v = u_t(x_\mathrm{B}) - v_\mathrm{B}, \tag{5}$$

where $v_\mathrm{B}(t)$ is an externally supplied bow velocity, s_0 is the bristle stiffness (in N/m), s_1 is the damping coefficient (in kg/s), s_2 is the viscous friction (in kg/s) and s_3 is a dimensionless noise coefficient multiplied onto pseudorandom function $w(t)$ (in N) as done in [4] and adds noise to the friction force. Here, $\dot{z}$ indicates a time derivative of z, and is related to v through

$$\dot{z} = r(v,z) = v\left[1 - \alpha(v,z)\frac{z}{z_\mathrm{ss}(v)}\right], \tag{6}$$

where z_ss is the steady-state function

$$z_\mathrm{ss}(v) = \frac{\mathrm{sgn}(v)}{s_0}\left[f_\mathrm{C} + (f_\mathrm{S} - f_\mathrm{C})e^{-(v/v_\mathrm{S})^2}\right], \tag{7}$$

with Stribeck velocity v_S (in m/s), Coulomb force $f_\mathrm{C} = f_\mathrm{N}\mu_\mathrm{C}$ and stiction force $f_\mathrm{S} = f_\mathrm{N}\mu_\mathrm{S}$ (both in N). Here μ_C and μ_S are the dynamic and static friction coefficient respectively and $f_\mathrm{N}(t)$ is the normal force (in N) which is, like $v_\mathrm{B}(t)$, externally supplied. See Figure 2 for a plot of (7).

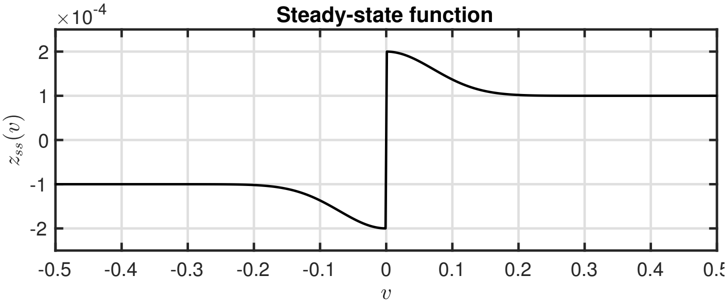


Fig. 2: *A plot of the steady-state function $z_{ss}(v)$ with a force of 5 N.*

Furthermore, the adhesion map between the bow and the string is defined as

$$\alpha(v,z) = \begin{cases} \left.\begin{matrix} 0 & |z| \le z_\mathrm{ba} \\ \alpha_\mathrm{m}(v,z) & z_\mathrm{ba} < |z| < |z_\mathrm{ss}(v)| \\ 1 & |z| \ge |z_\mathrm{ss}(v)| \end{matrix}\right\} & \text{if } \mathrm{sgn}(v) = \mathrm{sgn}(z) \\ 0 & \text{if } \mathrm{sgn}(v) \neq \mathrm{sgn}(z), \end{cases} \tag{8}$$

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

$$\alpha_\mathrm{m} = \frac{1}{2}\left[1 + \mathrm{sgn}(z)\sin\left(\pi\frac{z - \mathrm{sgn}(z)\frac{1}{2}(|z_\mathrm{ss}(v)| + z_\mathrm{ba})}{|z_\mathrm{ss}(v)| - z_\mathrm{ba}}\right)\right], \tag{9}$$

with break-away displacement z_ba, i.e., where the bristles start to break (see Figure 1 c)). A plot of the adhesion map can be found in Figure 3.[1]

One of the difficulties in working with this model is that, due to the many approximations, the notion of an energy balance, relating the rate of stored energy in the system to power input and loss is not readily available. Such energy methods are used frequently in the context of physical modeling synthesis and virtual analog modeling as a means of arriving at numerical stability conditions for strongly nonlinear systems, as is the present case. See, e.g., [14]. This means that we do not have a means of ensuring numerical stability in the algorithm development that follows. This does not mean, however, that an energy balance is not available.

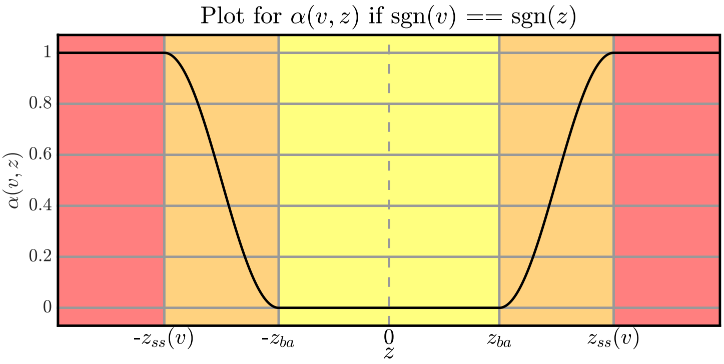


Fig. 3: *A plot of the adhesion map $\alpha(v,z)$ plotted against z when the signs of v and z are the same. The different regions of the map are shown with the coloured areas and correspond to Figure 1 according to: yellow - a) & b), orange - c) and red - d).*

3 Discretisation

Finite-difference schemes for the stiff string in isolation are covered by various authors [13, 14].

Equation (1) can be discretised at times $t = nk$, with sample $n \in \mathbb{N}$ and time-step $k = 1/f_\mathrm{s}$ (in s) with sample-rate f_s (in Hz) and locations $x = lh$,

[1] It is interesting to note is that in the literature on this topic such as [1, 2, 4, 5], a few inaccuracies can be found in the definition of $\alpha(v,z)$: 1) all uses of z_ss in (8) and (9) lack the absolute value operator, 2) the multiplications with $\mathrm{sgn}(z)$ in (9) are excluded, 3) $\alpha(v,z)$ is undefined for $|z| = z_\mathrm{ba}$ and $|z| = |z_\mathrm{ss}(v)|$ (correct in the original paper by Dupont et al. [1]). It can be shown that only with the definitions presented here, is it possible to obtain the curve shown in Figure 3.

where grid spacing h (in m) needs to abide the following condition [14]

$$h \geq h_{\text{min}} = \sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}} \tag{10}$$

and grid points $l \in [0, ..., N]$, where $N = \text{floor}(L/h)$ and $N+1$ is the total number of grid points. It is important to note that the closer h is to h_{min}, the more accurate the scheme will be. Approximations for the derivatives found in (1) are described in the following way [14]:

$$u_t \approx \delta_{t\cdot} u_l^n = \frac{1}{2k}\big(u_l^{n+1} - u_l^{n-1}\big), \tag{11a}$$

$$u_{tt} \approx \delta_{tt} u_l^n = \frac{1}{k^2}\big(u_l^{n+1} - 2u_l^n + u_l^{n-1}\big), \tag{11b}$$

$$u_{xx} \approx \delta_{xx} u_l^n = \frac{1}{h^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n\big), \tag{11c}$$

$$u_{txx} \approx \delta_{t-}\delta_{xx} u_l^n = \frac{1}{hk^2}\big(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}\big), \tag{11d}$$

$$u_{xxxx} \approx \delta_{xxxx} u_l^n = \frac{1}{h^4}\big(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n\big), \tag{11e}$$

with grid function u_l^n denoting a discretised version of $u(x,t)$ at the nth time step and the lth point on the string. Note that in (11d), the backwards time difference operator is used to keep (12) explicit and thus computationally cheaper to update. Using the approximations shown in (11), (3) can be discretised to

$$\begin{aligned} \delta_{tt} u_l^n &= c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t\cdot} u_l^n \\ &\quad + 2\sigma_1 \delta_{t-}\delta_{xx} u_l^n - J(x_{\text{B}}^n) f(v^n, z^n)/\rho A, \end{aligned} \tag{12}$$

where the relative velocity described in (5) can be discretised as

$$v^n = I(x_{\text{B}}^n)\delta_{t\cdot} u_l^n - v_{\text{B}}^n. \tag{13}$$

Here, $I(x_{\text{B}}^n)$ and $J(x_{\text{B}}^n)$ are weighting functions where the former interpolates the string displacement and velocity and the latter distributes the bowing term around time-varying bowing position x_{B}^n (see Figure 4 and [14] for more details on this). Furthermore,

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n \tag{14}$$

is the discrete counterpart of (4) where

$$r^n = r(v^n, z^n) = v^n \left[1 - \alpha(v^n, z^n)\frac{z^n}{z_{\text{ss}}(v^n)}\right] \tag{15}$$

is the discrete counterpart of (6).

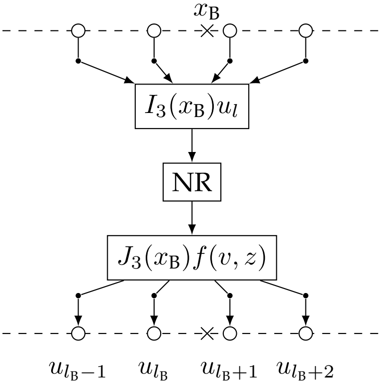


Fig. 4: *Cubic interpolation at bowing point x_B. The interpolator I retrieves the values of four grid points which are then used in the Newton-Raphson (NR) solver. This outputs the force function $f(v,z)$ that the spreading function J in turn distributes over the same four grid points. This process happens every single sample.*

At the bowing point we need to iteratively solve for two unknown variables: the relative velocity between the bow and the string v^n and the mean bristle displacement z^n of the bow at sample n. We can solve (12) at x_B^n using (13) and identity [14]

$$\delta_{tt}u_l^n = \frac{2}{k}\big(\delta_{t\cdot}u_l^n - \delta_{t-}u_l^n\big) \qquad (16)$$

resulting in

$$I(x_\text{B}^n)J(x_\text{B}^n)f(v^n,z^n)/\rho A + \Big(\frac{2}{k}+2\sigma_0\Big)v^n + b^n = 0, \qquad (17)$$

where

$$\begin{aligned} b^n = &\frac{2}{k}v_\text{B}^n - \frac{2}{k}I(x_\text{B}^n)\delta_{t-}u_l^n - c^2 I(x_\text{B}^n)\delta_{xx}u_l^n + \kappa^2 I(x_\text{B}^n)\delta_{xxxx}u_l^n \\ &+ 2\sigma_0 v_\text{B}^n - 2\sigma_1 I(x_\text{B}^n)\delta_{t-}\delta_{xx}u_l^n \end{aligned} \qquad (18)$$

and can be pre-computed as its terms are not dependent on v^n or z^n. Recalling (4), this can be rewritten to

$$I(x_\text{B}^n)J(x_\text{B}^n)\left(\frac{s_0z^n + s_1r^n + s_2v^n + s_3w^n}{\rho A}\right) + \Big(\frac{2}{k}+2\sigma_0\Big)v^n + b^n = 0. \qquad (19)$$

To obtain the values of v^n and z^n, multivariate Newton-Raphson (NR) is used. If (19) is defined to be $g_1 = g_1(v^n, z^n)$ and

$$g_2(v^n, z^n) = r^n - a^n = 0, \tag{20}$$

with

$$a^n = (\mu_{t-})^{-1}\delta_{t-}z^n \tag{21}$$

(where the operators applied to z^n denote the trapezoid rule [14]) we obtain the following iteration

$$\begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} = \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{22}$$

where i is the iteration number capped by 50 iterations, and the convergence threshold is set to 10^{-7}.

4 Implementation

In this section, we will elaborate on the implementation; the parameters used and the system architecture. The real-time implementation of the discrete-time model shown in the previous section has been done using C++ together with the JUCE framework [22]. The application is shown in Figure 5. The parameters we used can be found in Table 1, most of which are based on implementations by Serafin in [4]. These parameters will be static, i.e., are not user-controlled (except for z_{ba} and s_3 which rely on f_{N}). A demonstrative video can be found in [23]. We use the passivity condition proposed by

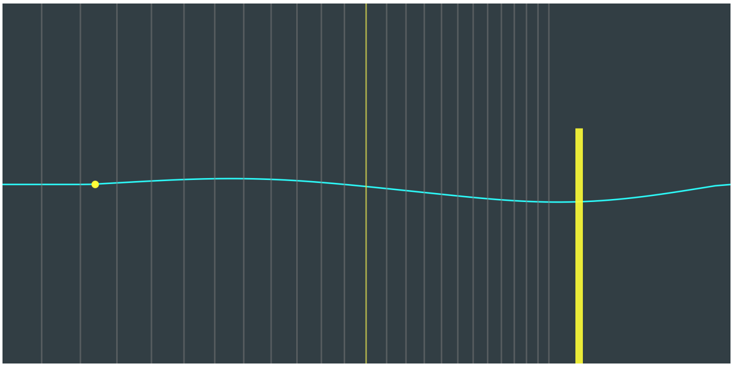


Fig. 5: *The elasto-plastic bowed string application. The bow is shown as a yellow rectangle, moves on interaction and its opacity depends on the finger force. The state* $\mathbf{u}^n$ *is visualised using the cyan curve and stopping-finger position is shown as a yellow circle. The grey lines show the 'frets' corresponding to semi-tones as a visual reference for the stopping position and do not influence the model.*

[24] for our choices of different parameter-values. As this condition applies to

Table 1: *Parameter values. Values for the fundamental frequency f_0 can be found in Section 5.*

Parameter	Symb. (unit)	Value (notes)
Material Density	ρ (kg·m^{-3})	7850
Radius	r (m)	$5 \cdot 10^{-4}$
String length	L (m)	1
Wave speed	c (m/s)	$2f_0/L$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq. indep. damping	σ_0 (s^{-1})	1
Freq. dep. damping	σ_1 (m^2/s)	$5 \cdot 10^{-3}$
Coulomb friction	μ_C (-)	0.3 ($< \mu_\text{S}$)
Static friction	μ_S (-)	0.8 ($> \mu_\text{C}$)
Normal force	f_N (N)	10
Bow velocity	v_B (m/s)	0.1
Bow position	x_B (m)	0.25
Stribeck velocity	v_S (m/s)	0.1
Bristle stiffness	s_0 (N/m)	10^4
Bristle damping	s_1 (kg/s)	$0.001\sqrt{s_0}$
Viscous friction	s_2 (kg/s)	0.4
Noise coefficient	s_3 (-)	$0.02f_\text{N}$
Pseudorandom func.	w (N)	$-1 < w < 1$
Break-away disp.	z_ba (m)	$0.7f_\text{C}/s_0$ ($< f_\text{C}/s_0$)
Sample rate	f_s (Hz)	44,100
Time step	k (s)	$1/f_\text{s}$

the LuGre model first proposed in [25, 26] from which the elasto-plastic model evolved, further investigation is required to conclude whether these conditions are identical for the elasto-plastic model.

4.1 Sensel Morph

As mentioned in Section 1, the Sensel Morph (or Sensel for short) is used as an interface to control the bowed string (see Figure 6). The Sensel is a highly sensitive touch controller containing ca. 20,000 pressure sensitive sensors that allow for expressive control of the implementation [21].

4.2 Interaction

The first finger the Sensel registers is linked to the following parameters: the normal force of the bow f_N (finger pressure), the bowing velocity v_B (vertical finger velocity) and bowing position x_B (horizontal finger position). The parameters are limited by the following conditions: $0 \leq f_\text{N} \leq 10$, $-0.3 \leq v_\text{B} \leq 0.3$ and $0 < x_\text{B} < L$. The second finger acts as a stopping finger on the string. As done in [20], for a string stopped at location $x_\text{f} \in [0, L]$ and $l_\text{f} = \text{floor}(x_\text{f}/h)$ we

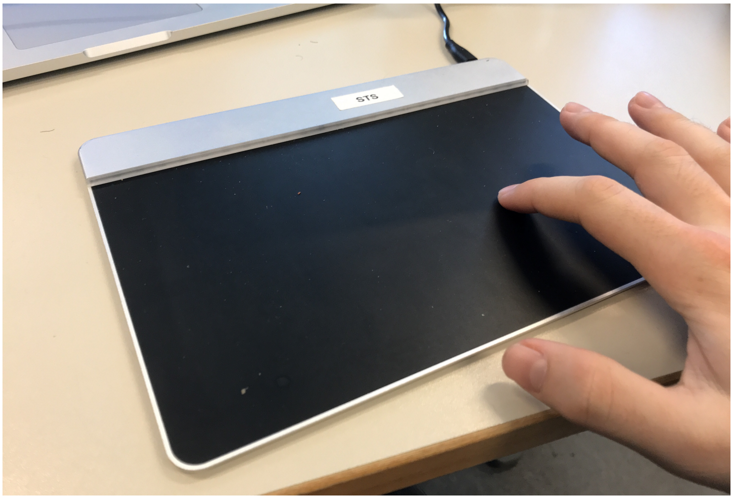


Fig. 6: *The Sensel Morph: an expressive touch sensitive controller used for controlling the real-time elasto-plastic bowed string implementation.*

use

$$u_l^n = \begin{cases} 0, & l = l_\text{f} - 1 \vee l = l_\text{f} \\ (1-\alpha_\text{f}^\epsilon)u_l^n, & l = l_\text{f}+1 \\ u_l^n, & \text{otherwise} \end{cases} \tag{23}$$

where $\alpha_\text{f} = x_\text{f}/h - l_\text{f}$ and $\epsilon = 7$ is a heuristic value that has been found to most linearly alter pitch between grid points.

4.3 System Architecture

Implementation of the scheme shown in (12) starts by expanding the operators shown in (11) and solving for the state at the next sample $\mathbf{u}^{n+1}$ where $\mathbf{u}$ is a vector containing the values for all grid points $l \in [0, ..., N]$.

An overview of the system architecture can be found in Figure 7. The three main components of the application are the Sensel controlling the application, the violin string class that performs the simulation and the main application class that moderates between these and the auditory and visual outputs. The black arrows indicate instructions that one of these components can give to another and the hollow arrows indicate data flows. Moreover, the arrows are accompanied by coloured boxes, depicting what thread the instruction or data flow is associated with and at what rate this runs.

The graphics thread has the lowest priority, is denoted by the green boxes and runs at 15 Hz. The redraw instruction merely retrieves the current string state $\mathbf{u}^n$ and bow and finger position and visualises this as shown in Figure 5.

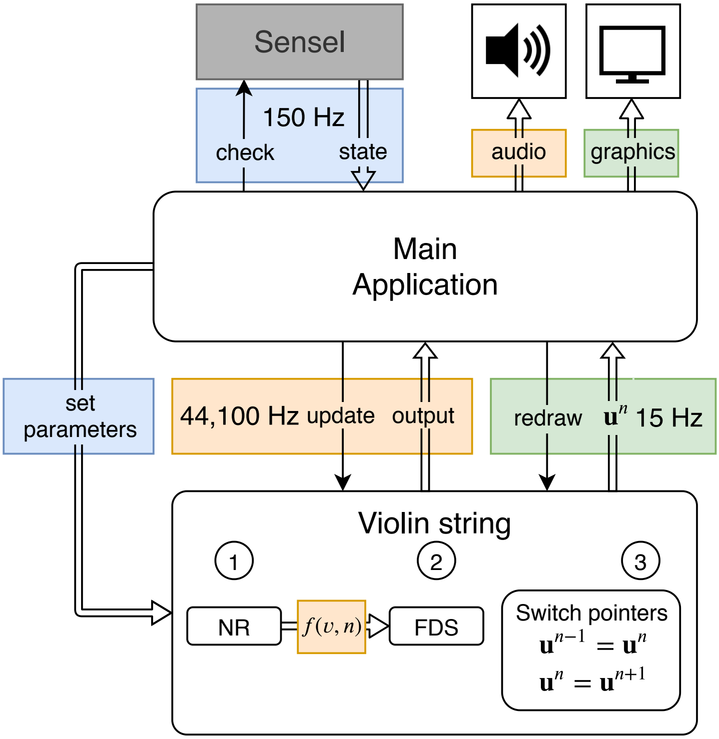


Fig. 7: *The system architecture. See Section 4.3 for a thorough explanation.*

The thread checking and receiving data from the Sensel runs at 150 Hz and is denoted by the blue boxes. The parameters that the user interacts with (bowing force, velocity and position) are also updated at this rate.

The highest priority thread is the audio thread denoted by the orange boxes and runs at 44,100 Hz. The violin string class gets updated at this rate and performs operations in the order shown in Algorithm 1.

5 Results and Discussion

Figure 8 shows the output waveforms for a string with $f_0 = 440$ Hz at different points along the string. The bowing parameters are $f_\text{N} = 5$ N and $v_\text{B} = 0.1$ m/s. The figure shows the traditional Helmholtz motion, which is the characteristic motion of a bowed string.

To test whether the implementation exhibits a hysteresis loop, the force vs. relative velocity plane was visualised. In Figure 9, this plot can be found for which the same parameters have been used. The figure shows values for 500 samples around $t = 0.5 f_\text{s}$. As can be seen from the figure, the hysteresis loop

```
for $t$ = $1{:}lengthSound$ do
    calculate computable part $b^n$ (Eq. (18))
    $\epsilon = 1$
    $i = 0$
    while $\epsilon < tol \wedge i < 50 \wedge f_C > 0$ do
        calculate..
        1. $z_{\text{ss}}(v^n_{(i)})$               (Eq. (7) in discrete-time)
        2. $\alpha(v^n_{(i)}, z^n_{(i)})$           (Eq. (8) in discrete-time)
        3. $r(v^n_{(i)}, z^n_{(i)})$                (Eq. (15))
        4. $g_1, g_2$                               (Eqs. (19) and (20))
        5.–9. Compute derivatives of 1.–4. in the same order.
        10. Perform vector NR to obtain $v^n_{(i+1)}$ and $z^n_{(i+1)}$
        11. Calculate $\epsilon$: $\epsilon = \left\| \begin{bmatrix} v^n_{(i+1)} \\ z^n_{(i+1)} \end{bmatrix} - \begin{bmatrix} v^n_{(i)} \\ z^n_{(i)} \end{bmatrix} \right\|$
        12. Increment $i$: $i = i + 1$
    end
    Repeat 1.–3. using the values for $v^n$ and $z^n$ from the NR iteration.
    Calculate $f(v^n, z^n)$                   (Eq. (14))
    Calculate $\mathbf{u}^{n+1}$              (Eq. (12) expanded)
    $\mathbf{u}^{n-1} = \mathbf{u}^n$
    $\mathbf{u}^n = \mathbf{u}^{n+1}$
end
```

Algorithm 1: Pseudocode showing the order of calculations.

is achieved and is similar to the one observed in [19]. The group of values around $v = 0$ are due to the sticking behaviour, and the others (the loop on the left) to the slipping behaviour.

For testing the speed of the algorithm, a MacBook Pro with a 2.2 GHz Intel Core i7 processor was used. The algorithm was tested using different frequencies according to the violin tuning of empty strings: $f_0 = 196.0$ (G3), 293.66 (D4), 440.0 (A4) and 659.26 (E5) Hz corresponding to $N = 95$, 71, 49, and 33 grid points respectively. The results can be seen in Table 2. When the total number of strings is smaller than 4, always the lowest frequency strings are used.

From Table 2 it can be observed that for one string, the CPU usage is $< 6\%$ with the graphics thread disabled. This is a great result, given the fact that both the bow and the string model are computationally complex. Empirical

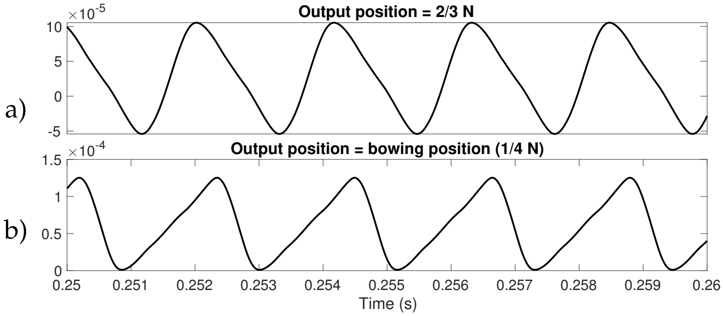


Fig. 8: *Output waveforms of the simulation at different positions along the string where N denotes the number of points of the string ($f_0 = 440$ Hz, $f_N = 5$ N and $v_B = 0.1$ m/s).*

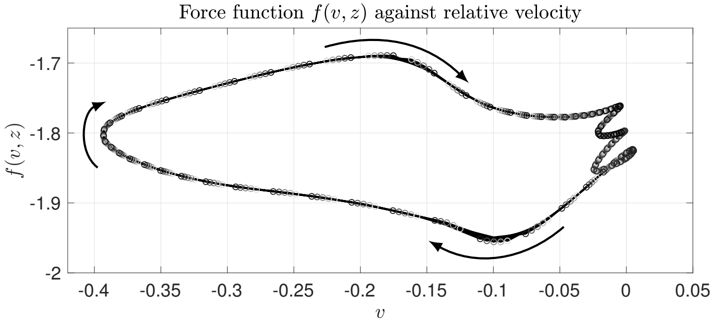


Fig. 9: *Hysteresis loop showing 500 values. The values around $v = 0$ are due to sticking behaviour and the loop on the left is due to slipping behaviour.*

investigation shows that the NR algorithm converges after ca. 3-4 iterations and the capping of 50 iterations never has to be used. A single string (but also more) could thus safely be used as an audio plugin in parallel to others without the user having to worry about auditory dropouts.

6 Conclusions

In this paper, we presented a real-time implementation of an elasto-plastic friction model with applications to a bow exciting a string, discretised using a finite-difference approach.

With a single string we are able to keep the CPU usage down to $< 6\%$ making for an efficient implementation that could be used in parallel with other virtual instruments or plugins.

Future work includes parameter design and including an instrument body for more realistic sounding results, as well as listening tests to verify the perceivable differences between simpler friction models versus the elasto-plastic

Table 2: *CPU usage for different amounts of strings. The values are averages over a 10 s period both for the enabled and disabled graphics thread. All strings are bowed simultaneously (polyphonically).*

Amount of strings	Graphics (%)	No graphics (%)
1	44.8	5.95
2	47.7	9.54
3	52.8	12.1
4	60.9	17.9

model.

7 Acknowledgments

Many thanks to the anonymous reviewers for giving their valuable input. This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

8 References

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

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

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

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

[5] F. Avanzini, S. Serafin, and D. Rocchesso, "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1073–1080, 2005.

[6] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments," *Journal of the Acoustical Society of America*, vol. 74, no. 5, pp. 1325–1345, 1983.

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

[8] J.-L. Florens, A. Razafindrakoto, A. Luciani, and C. Cadoz, "Optimized real time simulation of objects for musical synthesis and animated image synthesis," *ICMC 86*, pp. 65–70, 1986.

[9] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

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

[11] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[12] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[13] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. a physical model for a struck string using finite difference methods," *JASA*, vol. 95, no. 2, pp. 1112–1118, 1994.

[14] S. Bilbao, *Numerical Sound Synthesis*. J. Wiley & Sons, 2009.

[15] S. Bilbao, R. Hamilton, B and. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial," in *Springer Handbook of Systematic Musicology*. Springer, 2018, ch. 19, pp. 349–384.

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

[17] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2017.

[18] C. Desvages and S. Bilbao, "Two-polarisation finite difference model of bowed strings with nonlinear contact and friction forces," *Proceedings of the International Conference on Digital Audio Effects*, 2015.

[19] J. H. Smith and J. Woodhouse, "The tribology of rosin," *Journal of the Mechanics and Physics of Solids*, vol. 48, pp. 1633–1681, 2000.

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

[21] Sensel Inc., "Sensel Morph," accessed April 01, 2019, available at https://sensel.com/.

[22] JUCE ROLI, "JUCE," accessed April 04, 2019, available at https://juce.com/.

[23] S. Willemsen, "Elasto-Plastic Bow Model acting on a Finite-Difference Stiff String," accessed April 06, 2019, available at https://www.youtube.com/watch?v=-5bDebCW1Qg.

[24] K. J. Åström and C. Canudas de Wit, "Revisiting the lugre friction model," *IEEE Control Systems Magazine*, vol. 28, no. 6, pp. 101–114, 2008.

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

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

Paper D

Real-time Implementation of a Physical Model of the Tromba Marina

Silvin Willemsen, Stefania Serafin, Stefan Bilbao and Michele Ducceschi

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 161–168, 2020.

Abstract

The tromba marina is a medieval bowed monochord instrument. The string of the instrument rests on a rattling bridge that, due to the collision with the body, creates a trumpet-like sound. This paper presents a real-time implementation of a physical model of the tromba marina. The goal of the simulation is to make the instrument accessible to a larger audience. The physical model is implemented using finite-difference time-domain (FDTD) methods and non-iterative collision methods. A real-time implementation of the instrument is also presented. The simulation exhibits brass-like qualities and sounds similar to a real tromba marina, but requires further testing to validate the realism.

1 Introduction

The tromba marina (see Figure 1) is a medieval bowed monochord instrument with a long quasi-trapezoidal body and a uniquely fashioned bridge (often called a shoe, because of its shape – see Figure 2). The name of the instrument derives from the fact that *tromba* means *trumpet* in Italian. A peculiarity of the instrument is that a foot of the bridge is free to rattle against the soundboard in sympathy with the vibrating string. This unusual bridge creates a trumpet-like sound. The frequency produced by the instrument is varied by placing the side of the knuckle of the non-dominant hand, lightly, at specific nodal points on the string, in order to select various harmonics of the open string. The dominant hand controls the bow, which is drawn across the string above the non-dominant hand [1].

In this paper, we present a real-time implementation of a physical model of the tromba marina. One of the ultimate goals is the emulation of an instrument that, due to its rarity, is not accessible to a large audience.

Physical modelling for sound synthesis has a long history. Various techniques have been developed to simulate real-world instruments, including mass-spring systems [2], digital waveguides [3] and modal synthesis [4]. Finite-difference time-domain (FDTD) methods were first used for sound synthesis by Hiller and Ruiz in [5, 6, 7], later by Chaigne et al. in [8, 9] and elaborated upon by Bilbao and colleagues in [10, 11]. Compared with other techniques, FDTD methods are more computationally expensive, but easily generalisable and flexible—no assumptions of linearity of travelling wave solutions are employed. Our goal is to implement these techniques in real time and thereby make the simulations playable for the users. For this purpose, we use the expressive Sensel Morph controller [12]. Other work in real-time control of FDTD methods using this controller includes [13].

The emulation of nonlinear collision interactions in musical instruments normally requires the use of iterative solvers (such as the Newton-Raphson

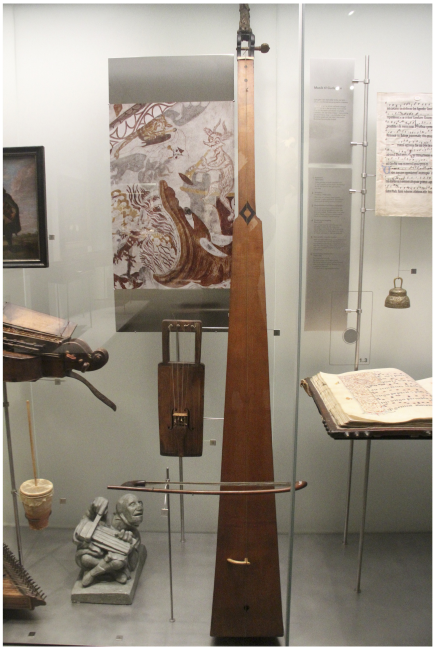


Fig. 1: The tromba marina from the Danish Music Museum in Copenhagen.

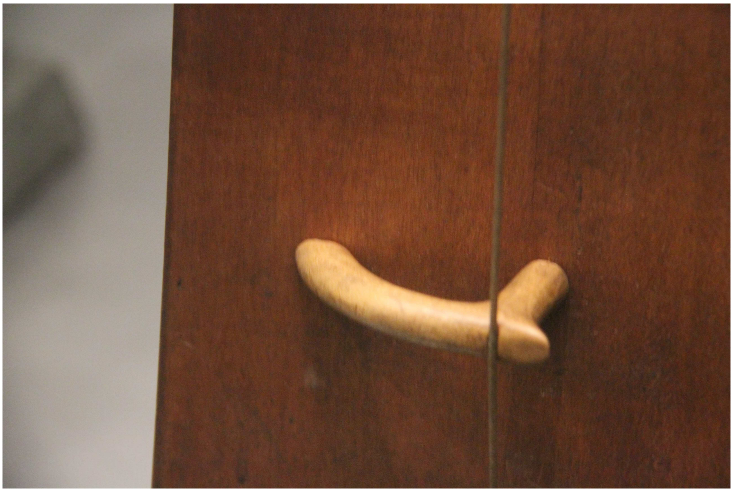


Fig. 2: The bridge of the tromba marina from the Danish Music Museum in Copenhagen. The right side is pressed against the body by the string while the left side is free and can rattle against the body.

algorithm) [14]. For the nonlinear collisions present in the instrument, a method recently proposed in the field of audio by Lopes and Falaize in [15, 16, 17] and later by Ducceschi and Bilbao in [18] allows such iterative methods to be sidestepped. It is thus suited to creating a real-time implementation of the tromba marina.

This paper is structured as follows: Section 2 presents the models used and Section 3 shows the discretisation of these. Section 4 provides information about implementation, parameter choices, the graphical user interface and control and mapping. Section 5 shows the results and discusses these. Concluding remarks and future work are presented in Section 6.

2 Models

The tromba marina can be subdivided into three main components: the string, the bridge and the body. In this section, the partial differential equations (PDEs) of the different components in isolation, under zero-input conditions, will be of the form

$$\mathcal{L}q = 0. \tag{1}$$

Here, $q = q(\boldsymbol{x}, t)$ represents the state of the component at time t and spatial coordinate $\boldsymbol{x} \in \mathcal{D}$, where the dimensions of domain $\mathcal{D}$ depend on the component at hand. Furthermore, $\mathcal{L}$ is a partial differential operator. (Subscripts 's', 'm' and 'p' used subsequently indicate that (1) applies to the string, bridge (mass) or body (plate), respectively.)

2.1 Bowed Stiff String

Consider a damped stiff string of length L (m), with domain $\mathcal{D} = \mathcal{D}_\text{s} = [0, L]$ and state variable $q = u(\chi, t)$. With reference to (1), we define the operator $\mathcal{L} = \mathcal{L}_\text{s}$ as [10]

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

Here, ∂_t and ∂_χ indicate partial differentiation with respect to t and χ. The various parameters appear as: material density ρ_s (kg·m^{-3}), cross-sectional area $A = \pi r^2$ (m^2), radius r (m), tension $T = (2f_{0,\text{s}}L)^2\rho_\text{s}A$ (N),[1] fundamental frequency $f_{0,\text{s}}$ (s^{-1}), Young's modulus E_s (Pa), area moment of inertia $I = \pi r^4/4$ (m^4), and loss coefficients $\sigma_{0,\text{s}}$ (s^{-1}) and $\sigma_{1,\text{s}}$ (m^2/s). We set the boundary

[1] Even though this definition for T from the fundamental frequency $f_{0,\text{s}}$ is only valid for a simply supported string without stiffness, the effect of the stiffness eventually chosen for $f_{0,\text{s}}$ is negligible.

conditions to be simply supported so that

$$u = \partial_\chi^2 u = 0 \quad \text{for} \quad \chi = 0, L. \tag{3}$$

As the string is excited using a bow, Equation (1) may be augmented as [10]

$$\mathcal{L}_\text{s} u = -\delta(\chi - \chi_\text{b}) F_\text{b} \Phi(v_\text{rel}), \tag{4}$$

with externally supplied downward bow force $F_\text{b} = F_\text{b}(t)$ (N), spatial Dirac delta function $\delta(\chi - \chi_\text{b})$ (m) selecting the bow position $\chi_\text{b} = \chi_\text{b}(t) \in \mathcal{D}_\text{s}$ (m) and dimensionless friction characteristic

$$\Phi(v_\text{rel}) = \sqrt{2a} v_\text{rel} e^{-a v_\text{rel}^2 + 1/2}, \tag{5}$$

with free parameter a. The relative velocity between the string at bow location χ_b and the externally supplied bow velocity $v_\text{b} = v_\text{b}(t)$ (m/s) is defined as

$$v_\text{rel} = \partial_t u(\chi_\text{b}, t) - v_\text{b}. \tag{6}$$

2.2 Bridge

The bridge is modelled as a simple mass-spring-damper system. As this system is point-like, or zero-dimensional, the state variable $q = w(t)$ and the definition of domain $\mathcal{D}$ is unnecessary. The operator $\mathcal{L} = \mathcal{L}_\text{m}$ is defined as

$$\mathcal{L}_\text{m} = M \frac{d^2}{dt^2} + M\omega_0^2 + MR\frac{d}{dt}, \tag{7}$$

with mass M (kg), linear angular frequency of oscillation $\omega_0 = 2\pi f_{0,\text{m}}$, (s^{-1}), fundamental frequency $f_{0,\text{m}}$ (s^{-1}) and damping coefficient R (s^{-1}).

2.3 Body

The body is simplified to a two-dimensional plate with side-lengths L_x and L_y, domain $\mathcal{D} = \mathcal{D}_\text{p} = [0, L_x] \times [0, L_y]$ and state variable $q = z(x, y, t)$. Using the 2D Laplacian

$$\Delta \triangleq \partial_x^2 + \partial_y^2, \tag{8}$$

the operator $\mathcal{L} = \mathcal{L}_\text{p}$ can be defined as [10]

$$\mathcal{L}_\text{p} = \rho_\text{p} H \partial_t^2 + D\Delta\Delta + 2\rho_\text{p} H \sigma_{0,\text{p}} \partial_t - 2\rho_\text{p} H \sigma_{1,\text{p}} \partial_t \Delta, \tag{9}$$

with material density ρ_p (kg·m^{-3}), plate thickness H (m), stiffness coefficient $D = E_\text{p} H^3 / 12(1 - \nu^2)$, Young's modulus E_p (Pa), dimensionless Poisson's ratio ν, and loss coefficients $\sigma_{0,\text{p}}$ (s^{-1}) and $\sigma_{1,\text{p}}$ (m^2/s). The boundary conditions of the plate are set to be clamped so that

$$z = \mathbf{n} \cdot \nabla z = 0. \tag{10}$$

where ∇z is the gradient of z, and where $\mathbf{n}$ indicates a normal to the plate area at the boundary.

2.4 Collisions

It can be argued that the greatest contributor to the characteristic sound of the tromba marina is the rattling bridge colliding with the body. A diagram of the bridge with important parts highlighted can be found in Figure 3. A collision can be modelled by including a term to the PDEs mentioned above describing the potential energy of the system (further referred to as *the potential*) [18]. For the bridge-body (mass-plate) interaction this potential is defined as follows

$$\phi_{\text{mp}}(\eta_{\text{mp}}) = \frac{K_{\text{mp}}}{\alpha_{\text{mp}} + 1}[\eta_{\text{mp}}]_+^{\alpha_{\text{mp}}+1}, \tag{11}$$

$$K_{\text{mp}} > 0, \quad \alpha_{\text{mp}} \geq 1, \quad \eta_{\text{mp}} \triangleq z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t)$$

where K_{mp} is the collision stiffness (N/m if $\alpha_{\text{mp}} = 1$), α_{mp} is the dimensionless nonlinear collision coefficient, and $\eta_{\text{mp}} = \eta_{\text{mp}}(t)$ is the distance between the rattling part of the bridge and the body at the point of collision (m). Furthermore, $[\eta_{\text{mp}}]_+ = 0.5(\eta_{\text{mp}} + |\eta_{\text{mp}}|)$ is the positive part of η_{mp}. Note that penalty methods are employed here, where a positive η_{mp}, i.e., interpenetration of the colliding objects, is intended [19]. The term which can then be included in the PDEs is $\phi'_{\text{mp}} = d\phi_{\text{mp}}/d\eta_{\text{mp}}$. As described in [15, 16, 17, 18], using this form of the potential requires using iterative methods for solving its discrete

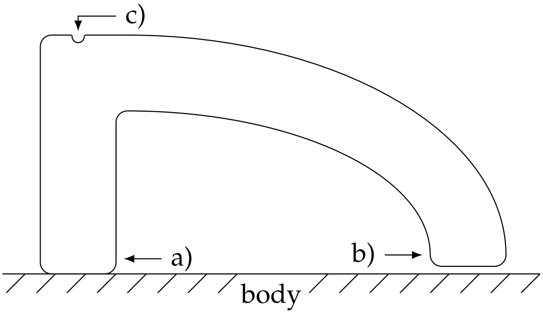


Fig. 3: Diagram of the bridge while rattling (view from top of the tromba marina). Indicated are: a) the pivoting point always in contact with the body, b) the rattling point colliding with the body (currently not colliding), and c) the string cavity straight above the middle of the pivoting point.

counterpart. In [18], the authors propose to rewrite the potential to

$$\psi = \sqrt{2\phi}, \tag{12}$$

and the term included in the PDEs to

$$\phi' = \psi\psi' = \psi\frac{d\psi}{d\eta} \xrightarrow{\text{chain rule}} \psi\frac{\dot{\psi}}{\dot{\eta}}\ , \tag{13}$$

where the dot above ψ and ϕ denotes a single time derivative. Equation (13), as can be seen in Section 3, leads to guaranteed stable and explicitly computable simulation algorithms without the need for iterative solvers.

As the string rests on the bridge, the interaction between these components needs to be modelled as well. Even though the bridge-body interaction is perpendicular to the string-bridge interaction, we can model them as being parallel, assuming that a “horizontal" movement of the string causes a “vertical" movement of the rattling part of the bridge. We can use an alternative version of the potential in Equation (11) described in [20] to make the collision two-sided acting as a connection:

$$\phi_{\text{sm}}(\eta_{\text{sm}}) = \frac{K_{\text{sm}}}{\alpha_{\text{sm}}+1}|\eta_{\text{sm}}|^{\alpha_{\text{sm}}+1}, \tag{14}$$

$$K_{\text{sm}} > 0, \quad \alpha_{\text{sm}} \geq 1, \quad \eta_{\text{sm}} \triangleq w(t) - u(\chi_{\text{sm}}, t)$$

where $\eta_{\text{sm}} = \eta_{\text{sm}}(t)$ is the distance between the string at the location of the bridge and the bridge itself.

2.5 Complete System

A complete system for the tromba marina may be written, in continuous-time as:

$$\begin{cases} \mathcal{L}_{\text{s}}u &= -\delta(\chi - \chi_{\text{b}})F_{\text{b}}\Phi(v_{\text{rel}}) \qquad \text{(15a)} \\ &\quad + \delta(\chi - \chi_{\text{sm}})\psi_{\text{sm}}\psi'_{\text{sm}} \\ \mathcal{L}_{\text{m}}w &= -\psi_{\text{sm}}\psi'_{\text{sm}} + \psi_{\text{mp}}\psi'_{\text{mp}}, \qquad \text{(15b)} \\ \mathcal{L}_{\text{p}}z &= -\delta(x - x_{\text{mp}}, y - y_{\text{mp}})\psi_{\text{mp}}\psi'_{\text{mp}}, \qquad \text{(15c)} \\ \eta_{\text{sm}} &= w(t) - u(\chi_{\text{sm}}, t), \qquad \text{(15d)} \\ \eta_{\text{mp}} &= z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t), \qquad \text{(15e)} \end{cases}$$

where $\chi_{\text{sm}} \in \mathcal{D}_{\text{s}}$ is the location of the bridge along the string and $(x_{\text{mp}}, y_{\text{mp}}) \in \mathcal{D}_{\text{p}}$ is the location on the body with which the bridge collides.

3 Discretisation

System (15) is discretised using FDTD methods. These methods subdivide the continuous system in grid points in space and samples in time. Before going into the discretisation of the models, and collision and connection terms in the system described in (15), some finite difference operators are introduced.

3.1 Operators

The identity and temporal shift operators are defined as

$$1\eta^n = \eta^n, \quad e_{t+}\eta^n = \eta^{n+1}, \quad e_{t-}\eta^n = \eta^{n-1}. \tag{16}$$

Using these, the operators for the forward, backward and centered time differences can be defined as

$$\delta_{t+} = \frac{e_{t+} - 1}{k},\ \delta_{t-} = \frac{1 - e_{t-}}{k},\ \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k}, \tag{17}$$

and are all approximations to a first-order time derivative. Furthermore, forwards and backwards averaging operators are defined as

$$\mu_{t+} = \frac{e_{t+} + 1}{2}, \quad \mu_{t-} = \frac{1 + e_{t-}}{2}. \tag{18}$$

and can be used to describe interleaved grid points $n + 1/2$ and $n - 1/2$ respectively.

3.2 Discrete Models

To approximate the state of a system in isolation we use

$$q(\boldsymbol{x}, t) \approx q_{\boldsymbol{l}}^n, \tag{19}$$

where grid function $q_{\boldsymbol{l}}^n$ is a discrete approximation to $q(\boldsymbol{x}, t)$ at $t = nk$ with time step k (s), time index $n \geq 0$ and grid location $\boldsymbol{l}$ that depends on domain $\mathcal{D}$ of the system at hand. In the case of the string, we use $\chi = lh_\text{s}$ with grid spacing h_s (m), $\boldsymbol{l} = l \in [0, \dots, N]$ and total number of grid points $N = L/h_\text{s}$ to yield $u(\chi, t) \approx u_l^n$.

In the case of the body, we use $x = lh_\text{p}$ and $y = mh_\text{p}$ to get $z(x, y, t) \approx z_{(l,m)}^n$ where $\boldsymbol{l} = (l, m)$ with $l \in [0, \dots, N_x]$ and $m \in [0, \dots, N_y]$. Here, $N_x = L_x/h_\text{p}$ and $N_y = L_y/h_\text{p}$ are the horizontal and vertical number of grid points respectively with grid spacing h_p (m).

The discretisation of and expansion of operator $\mathcal{L} \approx \ell$ in the case of stiff strings, mass-spring systems and plates using FDTD methods are well covered in the literature [10] and will not be described in detail in this paper. To obtain the highest accuracy possible while keeping the system explicit (except for the bow), centered differences – which are second-order accurate – have been chosen where possible.

For stability, grid spacings h_s and h_p should satisfy the conditions below. In the case of the damped stiff string,

$$h_\text{s} \geq \sqrt{\frac{c^2k^2 + 4\sigma_{1,\text{s}}k + \sqrt{(c^2k^2 + 4\sigma_{1,\text{s}}k)^2 + 16\kappa_\text{s}^2k^2}}{2}}, \tag{20}$$

with wave speed $c = \sqrt{T/\rho_\text{s}A}$ and stiffness coefficient $\kappa_\text{s} = \sqrt{E_\text{s}I/\rho_\text{s}A}$ and in the case of the plate,

$$h_\text{p} \geq 2\sqrt{k\left(\sigma_{1,\text{s}} + \sqrt{\kappa_\text{p}^2 + \sigma_{1,\text{s}}^2}\right)}\ , \tag{21}$$

with stiffness coefficient $\kappa_\text{p} = \sqrt{D/\rho_\text{p}H}$. The closer the grid spacings are to these conditions, the higher the accuracy of the approximation.

In order to discretise the Dirac delta functions found in system (15) we introduce a spreading operator $J(\boldsymbol{x}_\text{c})$ that applies a force to coordinate $\boldsymbol{x}_\text{c}$, which, in the simplest case, is defined as [10]

$$J(\boldsymbol{x}_\text{c}) = \begin{cases} \frac{1}{h^d}, & \boldsymbol{l} = \boldsymbol{l}_\text{c} = \text{round}(\boldsymbol{x}_\text{c}/h) \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

Here, d is the number of dimensions of domain $\mathcal{D}$ that $\boldsymbol{x}$ is defined for, i.e., $d = 0$ for the bridge, $d = 1$ for the string, and $d = 2$ for the plate. For finer control, a cubic spreading operator J_3 can be introduced [10]. This is used for the bowing term in Equation (4), which is discretised as follows

$$\ell_\text{s}u_l^n = -J_3(\chi_\text{b})F_\text{b}^n\phi(v_\text{rel}^n) \tag{23}$$

where, using the centered difference operator from Equation (17),

$$v_\text{rel}^n = \delta_{t\cdot}u_{l_\text{b}}^n - v_\text{b}^n, \tag{24}$$

with coordinate $l_\text{b} = \chi_\text{b}/h_\text{s}$. Equation (24) needs to be calculated using iterative methods.

3.3 Collisions using Non-Iterative Methods

For the discrete-time definitions of the potential in (13) we can use

$$\psi \approx \mu_{t+}\psi^{n-1/2} \quad \text{and} \quad \psi' \approx \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}, \tag{25}$$

where ψ at interleaved grid point $n-1/2$ is defined as

$$\psi^{n-1/2} = \mu_{t-}\psi^n. \tag{26}$$

Note that applying a forward or backward difference operator to an interleaved grid – such as $\delta_{t+}\psi^{n-1/2}$ in Equation (25) – is second-order accurate.

For a system that has a single (upward) collision we get

$$\ell q_{\boldsymbol{l}}^n = J(\boldsymbol{x}_{\mathrm{c}})\left(\mu_{t+}\psi^{n-1/2}\right)\frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}. \tag{27}$$

Here, we use the identity

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}\delta_{t+}\psi^{n-1/2} - \psi^{n-1/2} \tag{28}$$

and define

$$g^n = \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t\cdot}\eta^n}, \tag{29}$$

which can be rewritten to

$$\delta_{t+}\psi^{n-1/2} = g^n\delta_{t\cdot}\eta^n. \tag{30}$$

Then, inserting (30) into (28) and this together with (29) into (27) we get

$$\ell q_{\boldsymbol{l}}^n = J(\boldsymbol{x}_{\mathrm{c}})\left(\frac{k}{2}g^n\delta_{t\cdot}\eta^n - \psi^{n-1/2}\right)g^n \tag{31}$$

where g^n may be explicitly calculated using the analytic expressions for ψ and ϕ [18]:

$$g^n = \psi'\Big|_{\eta=\eta^n} = \frac{\phi'}{\sqrt{2\phi}}\Big|_{\eta=\eta^n}. \tag{32}$$

Numerical stability of this scheme is shown in [18]. When writing out (32) we can obtain definitions for g_{sm}^n using (14)

$$g_{\text{sm}}^n = \text{sgn}(\eta_{\text{sm}}^n)\sqrt{\frac{K_{\text{sm}}(\alpha_{\text{sm}}+1)}{2}}|\eta_{\text{sm}}^n|^{\frac{\alpha_{\text{sm}}-1}{2}}, \tag{33}$$

and g_{mp}^n using (11)

$$g_{\mathrm{mp}}^n = \sqrt{\frac{K_{\mathrm{mp}}(\alpha_{\mathrm{mp}}+1)}{2}}[\eta_{\mathrm{mp}}^n]_+^{\frac{\alpha_{\mathrm{mp}}-1}{2}}\;. \tag{34}$$

3.4 Complete Discrete System

Introducing for brevity,

$$\xi^n = \frac{k}{2}g^n\delta_{t\cdot}\eta^n - \psi^{n-1/2}, \tag{35}$$

the discrete counterpart of the complete system described in (15) will be

$$\begin{cases} \ell_{\mathrm{s}} u_l^n &= -J_3(\chi_{\mathrm{b}}^n)F_{\mathrm{b}}\Phi(v_{\mathrm{rel}}^n) + J(\chi_{\mathrm{sm}})\xi_{\mathrm{sm}}^n g_{\mathrm{sm}}^n, & \text{(36a)}\\ \ell_{\mathrm{m}} w^n &= -\xi_{\mathrm{sm}}^n g_{\mathrm{sm}}^n + \xi_{\mathrm{mp}}^n g_{\mathrm{mp}}^n, & \text{(36b)}\\ \ell_{\mathrm{p}} z_{(l,m)}^n &= -J(x_{\mathrm{mp}}, y_{\mathrm{mp}})\xi_{\mathrm{mp}}^n g_{\mathrm{mp}}^n, & \text{(36c)}\\ \eta_{\mathrm{sm}}^n &= w^n - u_{l_{\mathrm{sm}}}^n, & \text{(36d)}\\ \eta_{\mathrm{mp}}^n &= z_{(l_{\mathrm{mp}},m_{\mathrm{mp}})}^n - w^n, & \text{(36e)} \end{cases}$$

where discrete counterparts of connection and collision locations in Equations (36d) and (36e) are described as $l_{\mathrm{sm}} = \chi_{\mathrm{sm}}/h_{\mathrm{s}}$ and $(l_{\mathrm{mp}}, m_{\mathrm{mp}}) = (x_{\mathrm{mp}}/h_{\mathrm{p}}, y_{\mathrm{mp}}/h_{\mathrm{p}})$. This leaves us with two different types of update equations, one where q_l^{n+1} is calculated and one where $\psi^{n+1/2}$ is calculated.

One might think that due to the centered differences $\delta_{t\cdot}\eta^n$ still present in Equation (35), our system remains implicit, but as we can insert the definitions for Equations (36d) and (36e) evaluated at the next time index $n+1$, which are already present in $\ell_{\mathrm{s}} u_l^n$, $\ell_{\mathrm{m}} w^n$ and $\ell_{\mathrm{p}} z_{(l,m)}^n$, the Equations in (36) reduce to a system of linear equations that can be solved by a single division.

4 Implementation

The real-time implementation of the system has been done in C++ using the JUCE framework [21] and will be controlled using the Sensel Morph (or simply Sensel) – an expressive touch controller. A demo of the application can be found in [22]. This section will first elaborate some important considerations regarding the setup of the system. Then, the algorithm together with the parameter design will be presented. Finally, the graphical user interface (GUI) will be detailed together with the Sensel and its mapping to the application.

4.1 Introducing an Offset

Firstly, for more realistic and expressive sounds, we model the bridge – and with that, the string – to rest slightly above the body. Expanding $\ell_{\mathrm{m}} w^n$ in (36b) and including the offset yields

$$\ell_{\mathrm{m}} w^n \Rightarrow M\delta_{tt} w^n + M\omega_0^2 (w^n - w_{\mathrm{off}}) + MR\delta_{t\cdot} w^n \tag{37}$$

where $w_{\mathrm{off}} \geq 0$ is a predefined offset between the body and the bridge. Furthermore, the second-order time derivative can be defined from the definitions in (17) as

$$\delta_{tt} = \delta_{t+}\delta_{t-} \ . \tag{38}$$

The boundary condition of the string defined in Equation (3) will also change depending on the bridge offset:

$$u = w_{\mathrm{off}} \quad \text{and} \quad \partial_\chi^2 u = 0. \tag{39}$$

4.2 Pitch Control

Secondly, as briefly mentioned in Section 1, the way that different pitches are played on the tromba marina, is to slightly rest a knuckle or finger on nodal points along the string to induce harmonics. Thus, a damping finger is implemented. Using the cubic interpolation operator I_3 [10], Equation (36a) can be extended to

$$\ell_{\mathrm{s}} u_l^n = \ldots - J_3(\chi_{\mathrm{f}}) I_3(\chi_{\mathrm{f}}) \sigma_{\mathrm{f}} (u_l^n - w_{\mathrm{off}}), \quad \text{where} \quad 0 \leq \sigma_{\mathrm{f}} \leq 1, \tag{40}$$

which essentially subtracts its own state at location $\chi_{\mathrm{f}} \in [0, 0.5\chi_{\mathrm{sm}}]$ according to the damping coefficient σ_{f} (kg· s^{-2}) applied. As done in [13], the fractional part used in the spreading operator ($\alpha_{\mathrm{i}} = \chi_{\mathrm{f}}/h - \mathrm{floor}(\chi_{\mathrm{f}}/h)$) is raised to the 7th power as it has been found to scale finger position to pitch more properly in the context of FDTD. As the string is bowed above the damping finger (at the other side of the rattling bridge) it is essential that the energy from the bow reaches the rattling bridge, which is still the case for lower values of σ_{f}. A more realistic approach that could be investigated is to model the finger as a mass colliding with the string, rather than imposing the damping directly to the state of the string as presented here.

A schematic plot of the full system, including the offset described in Equations (37) and (39) and the damping finger from Equation (40) can be found in Figure 4.

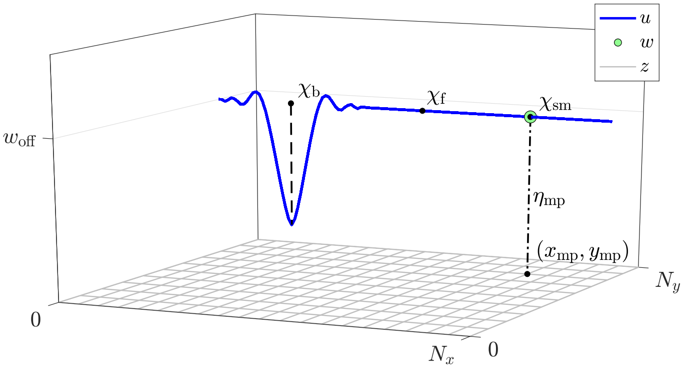


Fig. 4: The virtual system in (36) including the offset in Equation (37) and the damping finger in Equation (40), with different important coordinates highlighted. Note that η_sm (Equation (36d)) is not shown as it is close to 0 at all times.

4.3 Other Considerations

Realistic initialisation of both η_sm and η_mp is essential. In this case (at $n = 0$) $\eta^0_\mathrm{sm} = 0$ and $\eta^0_\mathrm{mp} \leq 0$ so that no collision is present at initialisation.

After h_p is calculated in Equation (21), we check whether it is smaller than a set value $h_\mathrm{p,min} = 0.01$. This reduces the quality of the model, but increases the speed, ultimately allowing for real-time implementation.

4.4 Order of Calculation

The order of calculation is shown in the pseudocode in Algorithm 1. In theory, in order to iteratively calculate the bow force, the collision and connection forces should be included in this. However, as the string is practically never bowed at the bridge position χ_sm, these can be calculated independently.

4.5 Parameter Design

The list of parameters used in the implementation can be found in Table 1. As the authors had a real (recreated) tromba marina (presented in [23]) at their disposal, some parameters have been measured in accordance to the real instrument. The others have been tuned by ear by one of the authors.

Regarding the output of the system, through informal testing it was decided to retrieve the output from the state of the plate right at the point of collision $z_\mathrm{out} = (l_\mathrm{mp}, m_\mathrm{mp})$ combined with the sound of the string at $u_\mathrm{out} = L - \chi_\mathrm{sm}$ at a lower volume. It can be argued that the loudest sound comes from the collision between the bridge and the body making it logical to select this point as the main sound source.

```
while application is running do
    1. calculate schemes                          ($\ell q$ in Eqs. (36a-c))
    2. apply bow to string                        (Eq. (36a))
    3. apply damping finger                       (Eq. (40))
    4. calculate $g_{\text{sm}}^n$ and $g_{\text{mp}}^n$               (Eqs. (33) and (34))
    5. calculate collision and connec-            (Eqs. (36a-c))
       tion forces and add to schemes
    6. Update states                              $\boldsymbol{q}^{n-1} = \boldsymbol{q}^n$
                                                  $\boldsymbol{q}^n = \boldsymbol{q}^{n+1}$
                                                  $\boldsymbol{\psi}^{n-1/2} = \boldsymbol{\psi}^{n+1/2}$
end
```

Algorithm 1: Pseudocode showing the order of calculation after initialisation. Bold symbols denote the collection of states of the entire system ($\boldsymbol{q}$) and potentials ($\boldsymbol{\psi}$).

4.6 Graphical User Interface

A screenshot of the GUI is shown in Figure 5. The GUI is divided in four sections, three showing the states of the string, bridge and body respectively and one control section.

Firstly, the string section shows the state of the string u as a cyan-coloured path and the bow as a yellow rectangle with bow position χ_{b} and its opacity depending on the bow force F_{b}. Furthermore, the bridge state w^n is shown as a green circle at location (of the bridge along the string) χ_{sm}. Finally, the position of the damping finger χ_{f} is displayed as a yellow circle, the size of which depends on damping coefficient σ_{f}. The position of the finger triggers lines showing the locations of the closest nodes along the string according to the following equation

$$\chi_{\text{node}}^i = \frac{i \cdot \chi_{\text{sm}}}{n} \quad \text{for} \quad i = [1, \dots, n-1], \tag{41}$$

where $n = \text{round}(\chi_{\text{sm}}/\chi_{\text{f}})$ is an integer closest to the ratio between the string length until the bridge location and the damping finger position. These lines are drawn to help the user place the damping finger at nodes along the string.

Secondly, the bridge section shows the displacement of the bridge w as a green circle, the state of the body at the collision location $z_{(l_{\text{sm}}, m_{\text{sm}})}$, both moving vertically according to their respective displacements and finally, a

Name	Symbol (unit)	Value
String		
Length	L (m)	1.90*
Material density	ρ_s (kg·m^{-3})	7850
Radius	r (m)	0.0005
Fundamental freq.	f_0 (s^{-1})	32*
Young's modulus	E_s (Pa)	$2 \cdot 10^{11}$
Freq. indep. loss	$\sigma_{0,\text{s}}$ (s^{-1})	0.1
Freq. dep. loss	$\sigma_{1,\text{s}}$ (m^2/s)	0.05
Bow		
Bow force	F_b (N)	$0 \leq F_\text{b} \leq 0.1$
Bow velocity	v_b (m/s)	$-0.5 \leq v_\text{b} \leq 0.5$
Free parameter	a (-)	100
Bridge		
Mass	M (kg)	0.001
Fundamental freq.	$f_{0,\text{m}}$ (s^{-1})	500
Damping	R (s^{-1})	0.05
Body		
Length	L_x (m)	1.35*
Width	L_y (m)	0.18*
Material density	ρ_p (kg·m^{-3})	50
Thickness	H (m)	0.01
Young's modulus	E_p (Pa)	$2 \cdot 10^5$
Poisson's ratio	ν (-)	0.3
Freq. indep. loss	$\sigma_{0,\text{p}}$ (s^{-1})	2
Freq. dep. loss	$\sigma_{1,\text{p}}$ (m^2/s)	0.05
Min. grid spacing	$h_{\text{p,min}}$ (m)	0.01
String-bridge connection		
Stiffness coefficient	K_sm (N/m)	$5 \cdot 10^6$
Nonlin. col. coeff.	α_sm (-)	1
Bridge location	χ_sm (m)	1.65*
Bridge-body collision		
Stiffness coefficient	K_mp (N/m)	$5 \cdot 10^8$
Nonlin. col. coeff.	α_mp (-)	1
Bridge location	$(x_\text{mp}, y_\text{mp})$ (m,m)	$(1.08, 0.135)$*
Other		
Offset	w_off (m)	$5 \cdot 10^{-6}$
Damp. finger coeff.	σ_f (kg·s^{-2})	$0 \leq \sigma_\text{f} \leq 1$
Output loc. string	u_out (m)	$L - \chi_\text{sm}$
Output loc. body	z_out (m, m)	$(x_\text{mp}, y_\text{mp})$

Table 1: List of parameter values used for the simulation. *These values have been taken from a real (recreated) tromba marina [23].

static grey horizontal line denoting the offset w_off, i.e., the resting position of the bridge.

Thirdly, the body section shows the state of the body z as a grid of rectangles

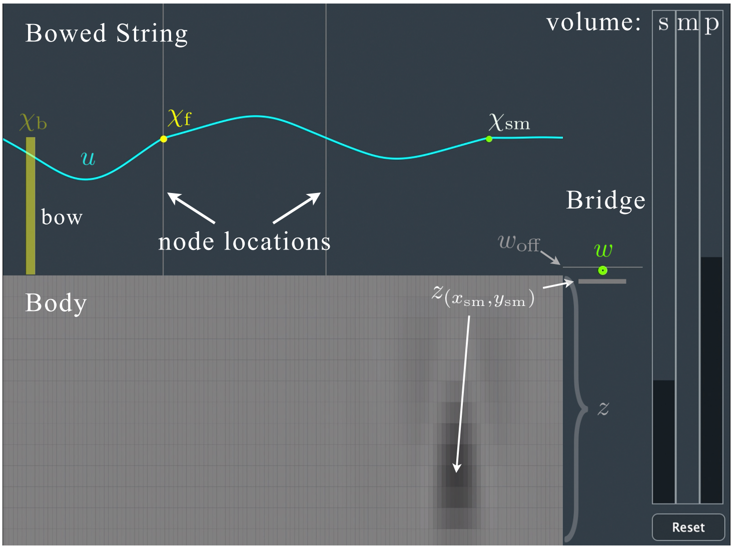


Fig. 5: The GUI showing the excited system with components highlighted. A more detailed description can be found in Section 4.6.

changing (grey-scale) colour according to their displacement.

Finally, the control section contains three sliders that control the volume-levels of the string (s), bridge (m) and body (p) respectively (for experimentation of volume ratios between the components) and a reset button to re-initialise the system.

4.7 Sensel Morph and Mapping

The Sensel is an expressive touch controller using ~20,000 pressure-sensitive sensors laid out in an hexagonal grid [12]. It retrieves x and y-positions and pressure at a rate of 150 Hz from which velocities and accelerations can be obtained.

The first finger registered by the Sensel is mapped to the bow: x-position is mapped to bow position χ_{b}, y-velocity to bow velocity v_{b} (y-position is shown in the GUI but does not influence the model directly) and pressure to bow force F_{b}. The second finger is mapped to the damping finger: x-position is mapped to finger location x_{f} and pressure to damping coefficient σ_{f}.

5 Results and Discussion

Informal listening by the authors has confirmed that the sound has brass-like qualities and comparison with the recreated tromba marina showed that the sound exhibited similar qualities. Naturally, formal listening tests need to be conducted to verify this.

Disabling the graphics of the application, its CPU usage is 68.9% on a MacBook Pro with a 2.2 GHz Intel i7 processor, easily allowing it to work in real-time. As the heaviest part of the algorithm is the calculation of the body, the minimum grid spacing $h_{\text{p,min}}$ could be set to a higher value to decrease the CPU usage. However, as mentioned, this will decrease the quality of the output sound.

Through using the application, the authors found some odd behaviour, where the bridge 'gets stuck' behind the plate, i.e., values for ψ_{mp} would be negative for a short period of time (one to several samples). The explicit technique used in this work allows for this to happen (and can be proven to still be stable in this case [18]), but it is 'unphysical' to have a negative potential as this implies a 'pulling' collision. As can be seen from Table 1, the nonlinear collision coefficients α_{sm} and α_{mp} are set to 1. When increasing these values, this behaviour would arise much more often, and even occur for a prolonged period of time (several seconds to indefinitely). This is also the reason why the reset button presented in Section 4.6 has been implemented. As mentioned in [18], oversampling increases the accuracy of the explicit collision method, and could be a solution to this issue. However, in order for the application to run in real time, this solution can not be afforded without decreasing the quality of the implementation, e.g. increasing $h_{\text{p,min}}$. Further investigation will be necessary to solve this issue without oversampling.

Lastly, it has been found that when $|z^n_{(l,m)}| \lessapprox 10^{-306}$ (but non-zero) for any coordinate (l, m) (which happens when the body has not been collided with for a prolonged period of time), the CPU usage increases considerably. This could be explained by the fact that calculations with extremely small values are handled differently by the application. This is solved by implementing a limit to how small a value for $z^n_{(l,m)}$ can be. If the value of $z^n_{(l_{\text{sm}},m_{\text{sm}})}$ is lower than this limit, the total plate state is set to 0.

6 Conclusion and Future Work

In this paper, a real-time implementation of a simulation of the tromba marina has been presented. The output sound has been found natural and brass-like by the authors and exhibited similar qualities when compared to a real (recreated) tromba marina.

Future work includes a comparison between the non-iterative methods used in this paper and iterative methods (such as and Newton-Raphson) both regarding algorithm speed and sound quality.

Lastly, for a more physical implementation of the damping finger, it would be good to model it as another mass colliding with the string rather than directly imposing damping onto the string state.

Acknowledgments

The authors would like to thank Peter Williams for his valuable feedback on our application.

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892. Ducceschi's work was supported by an Early Career Fellowship from the Leverhulme Trust.

7 References

[1] D. Munrow, *Instruments of the Middle Ages and Renaissance.* Oxford University Press, USA, 1976.

[2] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," Ph.D. dissertation, Grenoble INP, 1979.

[3] J. O. Smith, "Physical modeling using digital waveguides," *Computer music journal*, vol. 16, no. 4, pp. 74–91, 1992.

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

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

[6] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society*, vol. 19, no. 6, pp. 462–470, 1971.

[7] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society*, vol. 19, no. 7, pp. 542–550, 1971.

[8] A. Chaigne, "On the use of finite differences for musical synthesis. Application to plucked stringed instruments," *Journal d'Acoustique*, vol. 5, no. 2, pp. 181–211, 1992.

[9] A. Chaigne and A. Askenfelt, "Numerical simulations of struck strings. I. A physical model for a struck string using finite difference methods," *Journal of Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

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

[11] S. Bilbao, B. Hamilton, R. Harrison, and A. Torin, "Finite-difference schemes in musical acoustics: A tutorial." *Springer handbook of systematic musicology*, 2018.

[12] Sensel Inc. (2020) Sensel morph. [Online]. Available: https://sensel.com/

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

[14] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2015.

[15] N. Lopes, T. Hélie, and A. Falaize, "Explicit second-order accurate method for the passive guaranteed simulation of port-hamiltonian systems," *Proc. 5th IFAC*, 2015.

[16] A. Falaize and T. Hélie, "Passive guaranteed simulation of analog audio circuits: A port-hamiltonian approach," *Applied Sciences*, vol. 6, pp. 273–273, 2016.

[17] A. Falaize, "Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation hamiltonienne à ports," Ph.D. dissertation, Université Pierre et Marie Curie, Paris, 2016.

[18] M. Ducceschi and S. Bilbao, "Non-iterative solvers for nonlinear problems: The case of collisions," *Proc. of the 22th Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[19] S. Bilbao, A. Torin, and V. Chatziioannou, "Numerical modeling of collisions in musical instruments," *Acta Acustica united with Acustica*, vol. 101, no. 1, pp. 155–173, 2014.

[20] S. Bilbao and M. Ducceschi, "Large-scale real-time modular physical modeling sound synthesis," *Proc. of the 22th Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[21] JUCE ROLI. (2020) JUCE. [Online]. Available: https://juce.com/

[22] S. Willemsen. (2020) Virtual tromba marina - sensel morph. [Online]. Available: https://www.youtube.com/watch?v=x72Xh-nUoVc

[23] A. Baldwin, T. Hammer, E. Peciulis, P. Williams, D. Overholt, and S. Serafin, "Tromba moderna: A digitally augmented medieval instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, vol. 16, pp. 14–19, 2016.

Paper E

Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling

Silvin Willemsen, Razvan Paisa and Stefania Serafin

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 300–307, 2020.

Abstract

This paper proposes a multisensory simulation of a tromba marina – a bowed string instrument in virtual reality. The auditory feedback is generated by an accurate physical model, the haptic feedback is provided by the PHANTOM Omni, and the visual feedback is rendered through an Oculus Rift CV1 head-mounted display (HMD). Moreover, a user study exploring the experience of interacting with a virtual bowed string instrument is presented, as well as evaluating the playability of the system. The study comprises of both qualitative (observations, think aloud and interviews) and quantitative (survey) data collection methods. The results indicate that the implementation was successful, offering participants realistic feedback, as well as a satisfactory multisensory experience, allowing them to use the system as a musical instrument.

1 Introduction

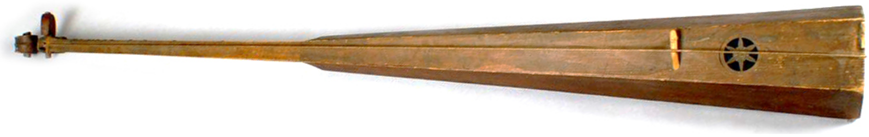


Fig. 1: A tromba marina owned by *Nationalmuseet* in Copenhagen, Denmark.

The tromba marina is a bowed monochord from medieval Europe [1] (see Figure 1). The string rests on a loose bridge that rattles against the body. This rattling mechanism creates a sound with brass- or trumpet-like qualities. Unlike other bowed string instruments, different frequencies are created by slightly damping the string with a finger of the non-bowing hand as opposed to pressing the string fully against the neck. This interaction at different locations along the string triggers the different harmonics of the open string. Furthermore, the tromba marina is bowed closer to the nut, and the finger determining the frequency is closer to the bridge (below the bow). As the tromba marina is a rare instrument which can be merely found in museums, very few have the opportunity to play it and discover its interesting timbral possibilities. We wish to recreate the feeling of playing this instrument by using physics based multisensory simulations [2].

In the context of musical applications, physics based multisensory simulations have shown some interest in the sound and music computing community. As stated in [3], the combination of haptics and audio visual content has its own specific challenges worth investigating. Sile O'Modhrain is one of the pioneers that noticed the tight connection between auditory and haptic feedback and investigated how haptic feedback can improve the playability of virtual instruments [4]. At the same time, Charles Nichols developed the vBow, a haptic human computer interface for bowing [5]. For several years, researchers from ACROE in Grenoble have developed multisensory instruments based on the mass-spring-system paradigm, with custom-made bowing interfaces [6, 7]. Such multisensory simulations have recently been made open source [8]. Haptic feedback has also been combined with digital waveguide models for simulating bowed string interactions [9].

Simulating the feeling of string-instrument vibrations is particularly important since it has been shown how vibrations' level can be strongly perceived [10]. We use democratized VR technologies controlled by a commercial device called the PHANTOM Omni (or simply Omni) by SenseAble Technologies (now 3D Systems) [11]. The Omni is a six-degrees-of-freedom system providing the tracking and haptic feedback (up to 3.3 N) in our application. Using the same device, Avanzini and Crosato tested the influence of haptic and auditory cues on perception of material stiffness [12]. Auditory stimuli were obtained using a physically-based audio model of impact, in which the colliding objects are described as modal resonators that interact through a non-linear impact force [13]. Auditory stiffness was varied while haptic stiffness was kept constant. Results show a significant interaction between auditory stiffness and haptic stiffness, the first affecting the perception of the second. Passalenti et. al's also used the Omni to simulate the act of plucking a virtual guitar string [14, 15, 16].

The goal of this project is to explore the experience of interacting with virtual bowed instrument by using physics based simulations and haptic feedback, together with a visual virtual reality (VR) experience. This effectively makes the implementation a virtual reality musical instrument (VRMI) [17]. The tromba marina is used solely as inspiration because it affords itself to being a solid starting point by having only one string. Besides that, the rarity of the instrument ensures that the participants do not have prior experience playing a tromba marina, nullifying possible comparisons between a real instrument and the virtual one. At no point the system was evaluated as an alternative to the real tromba marina. The system (and its evaluation) is targeted towards musicians in order to avoid discouragement frequently encountered when non-musicians interact with musical instruments. It is assumed that musicians acknowledge that mastering any instrument require extended study, therefore it is expected that they will not evaluate this system exclusively based on its difficulty to play.

We start by describing the implementation of the system, both from the hardware and software perspective in Section 2, followed by presenting a study that evaluates the setup in Section 3. Section 4 shows the results of the evaluation and Section 5 discusses these. Finally, concluding remarks appear in 6.

2 Implementation

The virtual tromba marina consists of three main components: auditory, visual and haptic feedback, all of which will be elaborated on in this section. For visuals, the Oculus Rift CV1 setup was used [18]. The setup consists of a head-mounted display (HMD) and a pair of of wireless controllers that provide tracking information and user input through several buttons and a joystick. A diagram showing the full setup of the system can be found in Figure 2. The controls, their mapping to the system and the final setup of the system will also be presented. A video showing the implementation can be found in [19].

2.1 Auditory Feedback

The audio is generated by a physical model of the tromba marina presented in a companion paper [20]. Some parameters of the model are exposed and can be controlled by the user. These are the velocity, force and position of the bow and the position of the finger inducing the harmonics. The algorithm will not be discussed in detail here, but the mapping to the various parameters of the model will be described in Section 2.4.

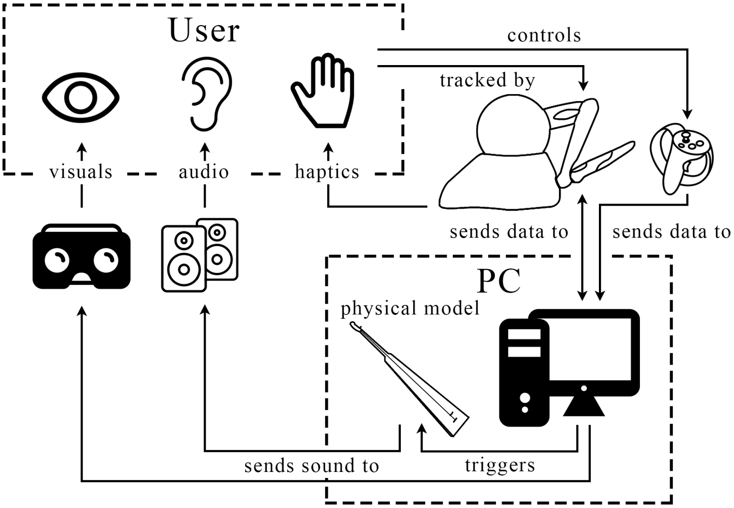


Fig. 2: Diagram showing the system layout of the application. The user interacts with the system using the Omni – which in turn provides haptic feedback – and the Oculus Touch controller. These trigger the physical model of the tromba marina. Auditory feedback then comes from speakers and visual feedback from the Oculus Rift headset. A detailed explanation can be found in Section 2.5.

2.2 Visual Feedback

The application was built using the cross-platform game engine Unity3D (or simply Unity) [21] which can be used to build VR applications. Even though the visual feedback is not the focus of the implementation and eventual evaluation, it was used to guide the users' movements and give them a sense of where the virtual instrument was located. Figure 3 shows a screenshot of the view from the HMD, depicting the virtual instrument, the bow and the damping finger indicator. A 3D model of the tromba marina was made inspired by a real-life instrument (presented in [22]) available to the authors. The overall environment resembled a medieval room, providing context to tromba marina's historical nature.

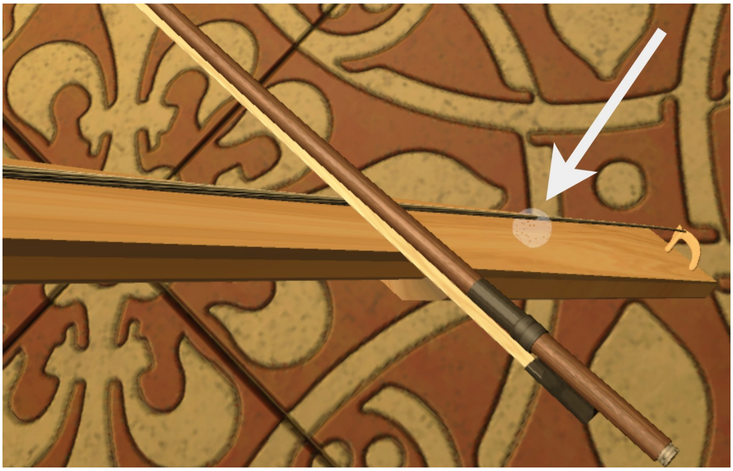


Fig. 3: The view from the head-mounted display (HMD). The damping finger is highlighted and shown as a transparent white sphere.

2.3 Haptic Feedback

The PHANTOM Omni (or simply Omni) is a six-degrees-of-freedom tracking and haptic system developed by SensAble Technologies (see Figure 4). The device has a pen-shaped arm that a user interacts with.

The raw data provided by the Omni are 1) the absolute position of pivot point B2 (three degrees of freedom), 2) the rotation (three degrees of freedom), and 3) the pressure (touching depth). The latter is calculated from the absolute euclidean distance between the virtual collision point of the object (in our case the bow) and the virtual position of the pen.

The axes are labelled as follows in relation to the virtual tromba marina (also see global coordinate system in Figure 5): x-axis (width): horizontally

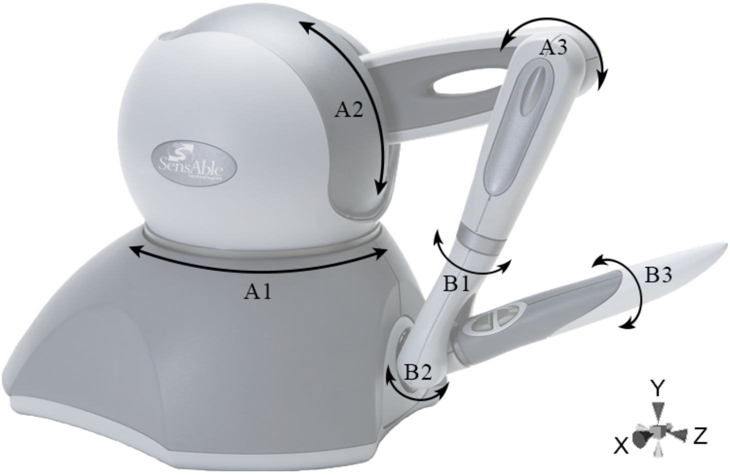


Fig. 4: The PHANTOM Omni has six axes of rotation, three of which provide force feedback (A1-3), and three only tracking position (B1-3). Together, these axes provide six degrees of freedom: x, y and z positions of B2 (according to the shown coordinate system) and rotations of the pen.

across the soundboard (the common interaction direction), y-axis (height): floor to ceiling, and z-axis (depth): perpendicular to the soundboard. The orientation of the Omni with respect to the aforementioned axis can be seen from the coordinate system in Figure 4.

The fact that pivot points B1-3 do not provide force feedback gives rise to an issue in our application. The virtual bow's frog (where it is held by the player) has been placed at the pivot point B2, whereas the interaction between the virtual bow and string happens at an offset as seen in Figure 5. To solve this issue, we created a separate game object with which the bow (pivot point B2 to be exact) will interact with in the virtual world Figure 5. This '(hidden) collision block' lives in a local coordinate system and its x and y-position exactly follow that of the Omni-pen. The y-rotation will change the rotation of the local coordinate system and uses the virtual string as the center point. If, for any reason, the bow ends up behind the string, the collision block will be offset to the left along the (local) x-axis so that no collision occurs when trying to return the bow to the normal playing area.

Through a list of pseudophysical parameters, the collision forces computed by Unity's physics engine are mapped to the haptic feedback produced by the Omni. Through empirical testing, the following pseudophysical parameters have been found: *Stiffness*: 0.003, *Damping*: 0.0071, *Static Friction*: 0, *Dynamic Friction*: 0.109, and *Pop-through*: 0. For more information, please refer to [23].

Throughout implementation, it was considered to actuate the Omni's pen with the output of the physical model used for the auditory feedback, in order to replicate the stick-slip interaction encountered in a real bowing scenario.

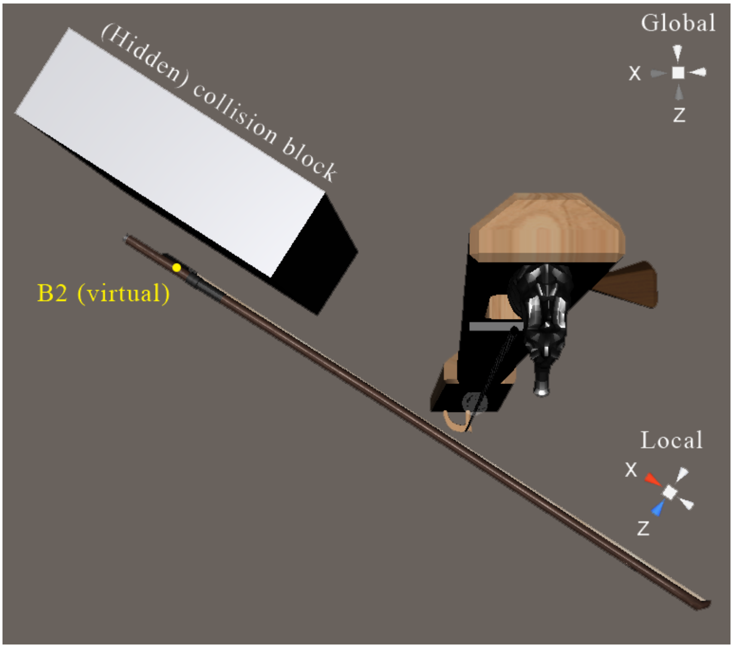


Fig. 5: Top-down view of the global and local coordinate system (x-z–plane). The rotation of the local coordinate system around the (global) y-axis is determined by the y-rotation of the bow. The (normally hidden) collision block lives in the local coordinate system. Its (local) x and y-position follows the (local) x and y-position of B2.

This was deemed unnecessary, as the Omni's internal gearing systems provide a similar, though uncorrelated, haptic feedback, which satisfied the authors.

2.4 Controls and Mapping

As most people are right-handed, it was chosen to also have the bow in the right hand in the application. The (now-local) x-velocity of the Omni is mapped to the bow velocity, pressure to bow force and y-position (including rotation around the local z-axis) to bow position. The left hand is used to control the pitch by changing the position of the damping finger along the string. This position is defined as

$$x_{\mathrm{f}} = L \cdot n^{-1}, \tag{1}$$

where L is the length of the string and $n \in [2, 8]$. If n is an integer, it is the number of the harmonic we want to induce. The lowest harmonic has been set at half the string length $L/2$, meaning that the string is never completely open. The highest harmonic (8 in this case) has been chosen to be the one that can still be (comfortably) reached. The location of the damping finger x_{f} is controlled using the 'X' and 'Y' buttons and the joystick on the left Oculus

Touch controller. The buttons are used for “discrete harmonic" control of the damping finger, i.e. integer values of n in Equation (1), where ‘Y’ increases n and ‘X’ decreases it. The joystick allows for fine pitch control, i.e., decimal values of n, and moves the damping finger up and down the string. The latter could potentially create pitch glides in the output sound of the application, but make it harder to ‘hit’ a perfect harmonic according to Equation (1). If a button is pressed while the current finger position is between two discrete points, the position will move to the next or previous discrete position, depending on the button pressed.

2.5 Physical Setup

The physical setup is shown in Figure 6. The Omni is mounted on a stand at ~125 cm to match the approximate bowing height of the real instrument. As can be seen in Figure 6, the right Oculus Touch controller is mounted right underneath the Omni. This is used to align the physical setup with the virtual tromba marina, both in the x-z–plane but also the height of the bow in the application. After the scene is initialised the controller is used for a tilting interaction so that the instrument can rest on the user’s body, as is done with the real instrument. The aforementioned alignment came with a drawback – as the center of the x-axis range of the Omni was aligned with the tromba marina and B2 was aligned with one end of the bow, only half of the range of the Omni could be used for bowing.

The setup shown in Figure 2 is implemented as follows: the user controls the application using the Omni (for tracking) and the left Oculus Touch controller which sends data to the computer running the application. The Omni produces haptic feedback based on Unity’s physics engine calculating the interaction force between the ‘(hidden) collision block’ and the virtual bow as shown in Figure 5. This data simultaneously triggers the physical model which sends its output to a pair of speakers. The user wears a HMD that gives visual information about the location of the tromba marina (and medieval scene). The user’s position in the VR environment is controlled by the HMD, but this dataflow is not visualised in the diagram. Lastly, the right Oculus Touch controller is attached to the stand the Omni is attached to, and sends position and tilting data to the application.

3 Evaluation

The goal of the study was to (1) evaluate the general experience of bowing in a VR environment using haptic feedback and accurate physical modelling and (2) to evaluate the playability of a VR monochord instrument. This was done by exploring the quality of the software, the acoustic model, the interface and

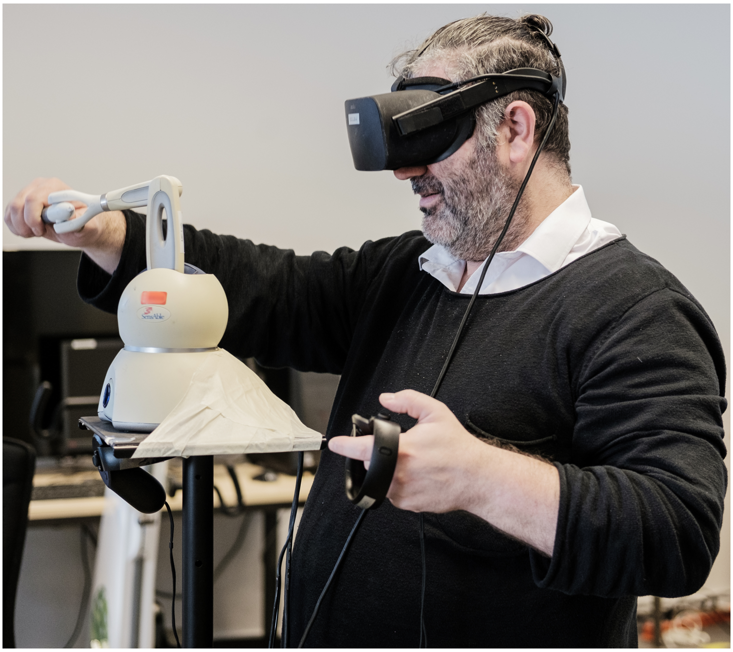


Fig. 6: User interacting with the physical setup. The Omni is mounted on a ~125cm stand together with the right Oculus Touch controller used for location and tilting information.

the mapping, as proposed by [24] and implemented previously in a similar study [25]. To meet this aim, an investigative study was performed through which feedback on the virtual instrument was collected. In order to ensure a high level of validity and reliability, a triangulation of methods has been used: think aloud protocol [26] throughout the interaction, observation and post-study self report through a modified Usability Metric for User Experience survey [27]. The study concluded with an semi-structured interview based on the observed actions, noted comments and questions loosely revolving around *goals, operators, methods and selection* method [28].

3.1 Participants

A total of 14 people (12 male, 2 female), 23-48 years old (M=29.5, SD=7.65) participated in the study. All participants were students or staff at Aalborg University Copenhagen. The selection of participants was based on the single criterion that one had to have experience playing a musical instrument. Over 70% of the participants have been playing an instrument for more than 5 years, guitar being the most common occurrence (25%). There was only one participant experienced in playing bowed instruments (violin). The same participant mentioned playing the tromba marina briefly before, but the majority of the other participants had never heard (of) it. All but one participant have had tried VR experiences before joining the study.

3.2 Procedure and Task

The experiment started with the participant reading an introduction about the experiment and completing a questionnaire covering several demographic questions (age, gender, musical experience, familiarity with the tromba marina and VR experience). They were then introduced to the setup and task, and controls were explained. The participants were informed that the study is exploring the experience of bowing in a VR environment. It was emphasised that the most important part of the experiment was for the participant to talk aloud with the phrase: "anything positive, negative, basically anything that comes to mind, please speak out loud". Furthermore, the user was instructed to bow above the damping finger (visualised as a white sphere) at all times, as this is also the interaction with the real instrument.

The interaction part was divided into two phases. Firstly, the participants were asked to freely explore the instrument on their own. Then, when they felt they are ready to move on, an audio recording made by the authors using the application was played, showcasing the system's capabilities, aiming to inspire the second phase of free exploration. It was stressed that the participants did not have to recreate what they heard, but to merely use it as inspiration. The experiment concluded with participants completing a questionnaire covering usability and playability of the system. Finally, a semi-structured interview was held which lasted 5 minutes on average.

Throughout the interaction phase, the participants' actions were observed and noted by the authors, and their comments written down. Most participants were encouraged again to think aloud during their exploration. The full experiment lasted ~30 minutes for all participants.

3.3 Measurements

Because the goal of the study was to investigate the overall experience of bowing in VR, as well as evaluate the playability of the instrument, self-reporting measurements were used in combination with the observations, interview and *think aloud* notations. Specifically, after exposed to the instrument, the participants were asked to fill out a questionnaire containing 20 items related to the experience of interacting with the VRMI. The items can be broadly segmented into four categories: overall experience, haptic feedback, auditory feedback and visual feedback. Table 1 presents the questions.

Questionnaire items:

Overall experience:
(1) It was easy to understand how to play the instrument.
(2) I felt the instrument was hard to play.
(3) I felt the instrument was expressive.
(4) The instrument's capabilities did not match my expectations.
(5) I felt I could easily achieve my goals.
(6) I made many errors playing the instrument.
(7) I am satisfied with the instrument.
(8) I felt the instrument was boring.
(9) Interacting with the instrument was frustrating.

Haptic feedback:
(10) I felt the haptic feedback was realistic.
(11) I felt the haptic feedback was too strong.
(12) I felt the haptic feedback was natural.

Auditory feedback:
(13) I felt I was in control of the sound.
(14) I felt the audio was matching my actions.
(15) I felt the sound was matching the haptic feedback.
(16) I felt the sound was matching the visuals.
(17) I felt the sound was static.

Visual feedback:
(18) I felt the visual feedback was helping me play.
(19) I felt the visuals were confusing.
(20) I felt the visuals were matching my actions.

Table 1: The questionnaire items and corresponding anchors of the 5 point (1 – 5) rating scales (Strongly disagree – Strongly agree).

4 Results

This section presents the results obtained from the self-reported measure regarding the participants' experience as well as the qualitative findings from interview, observations and think aloud.

4.1 Quantitative Data

The data obtained for the questionnaire items was treated as ordinal and analysed in terms of central tendency (medians and mode), interquartile ranges, minimum and maximum ratings. Figure 7 visualises the collected data. The mode was considered only when different from the median, specifically question 8, 13 and 14. It is worth noting that most of the items show a skewed normal distribution.

Question 1-9 paint a picture of how the instrument was perceived by the users. Questions 1, 2, 4, 5, 6 and 9 cover the perceived difficulty of using the

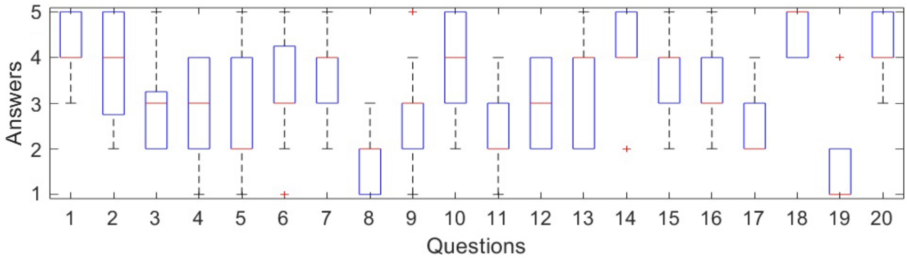


Fig. 7: Boxplots visualizing the results related to the 20 questionnaire items (shown in Table 1) in terms of medians (red lines), interquartile ranges (blue rectangles), minimum and maximum ratings (dashed lines), and outliers (red crosses). The y-axis maps "Strongly disagree – Strongly agree" to a 1 – 5 interval.

system as a musical instrument. The answers to these questions show that even though participants generally found the instrument easy to understand, they had difficulty playing it and reaching their goals. Questions 3, 7 and 8 cover their general opinion about the instrument. Participants generally felt satisfied and not bored with the instrument. Questions 10-12 cover exclusively the impressions about haptic feedback. It can be seen that most participants found the haptic feedback to be realistic and generally natural and the force to be not too strong. The questions 13-17 approach the auditory aspect of the instrument, focusing on its perceived characteristics. As can be seen from questions 13, 14 and 17, the participants felt a high level of command over the sound, and were satisfied with mapping between the haptic and auditory feedback. The same thing can be said about the visual mapping, as indicated by question 16. Items 18-20 investigate the perceived visual quality. It can be seen that the visuals helped the participants play and were implemented well, i.e., not confusing and matching their actions.

4.2 Qualitative Data

In order to present an accurate representation of the findings, this section will be split into two categories: actions – covering the observed activities during the interaction phase, and oral feedback – presenting the findings from the think aloud protocol and interviews.

4.2.1 Observed Actions

Since there were no tasks given to the participants, all actions were noted and analysed. That said, most users performed similar actions in their interaction phase. All participants experimented with bowing at different heights, but only a few of them tried to explore bowing heights for all discrete pitches. Most of them were satisfied with trying different heights on whatever pitch they found themselves at that time. In a similar fashion, all participants experimented with playing different pitches, both using the discrete buttons as well as the joystick. It is worth mentioning that many users tried to investigate the limits of the pitches they could play. Higher pitches usually resulted in little or no sound which was commented on by most. This behaviour is true to a real tromba marina, where higher harmonics are harder to excite than lower ones. The majority tried to perform some form of glissando, as well as bowing with different velocities, usually commenting on the findings. Due to the non-intrusive nature of observation, it was impossible to notice the pressure applied with the bow, but some participants explicitly mentioned that they tried to experiment with different forces. This was especially true in the second phase of interaction, when they experimented with a higher dynamic range of sounds. Another common occurrence was the attempt to play some sort of melody or riff. Simple melodies like *Mary had a little lamb*, or *Twinkle twinkle little star* were attempted, with various degrees of success. One participant tried to play a Mozart segment. The last commonality was found in the attempt to perform a sustained tone, with a constant bowing speed and a back-and-forth motion.

When it comes to seldom or individual actions, a great variance in experimentation was observed. Participants tried to hit the string with the bow, rotate the bow upwards to the point of it being parallel to the string, move the bow in an up-down (y-axis) motion, bow on the damping finger indicator and underneath it or play some form of vibrato or staccato. No one tried to tilt the stand supporting the Omni.

4.2.2 Oral Feedback

Generally the overall impression of the instrument was positive, described with words like: *cool, fun, interesting, weird*, as well as *hard* or *difficult to play*. One participant's answer encapsulates this very well by saying: "I got to express my ideas, but not perfect them".

Just as described in the previous section, there was a general consensus on several reported characteristics. All participants that attempted to play the highest harmonic said it is hard to play, and that it felt frustrating. At the other end, several participants expressed their preference towards lower pitches, where some said that they prefer the sound produced when bowing under the damping finger (essentially playing the lower-pitched open string). Besides that, many reported that is was hard to maintain a sustained tone, regardless of the pitch. Another sound-related report was the inability to re-create the buzzing sound heard in the recording; one of the participants familiar with the tromba marina's mechanism even mentioned specifically that he "couldn't get the bridge to rattle". When it comes to the pitch selection interface, the reports are very polarised between the joystick and the buttons. On one hand some describe the buttons as being more, *fun, musical, melodic, useful* or *easier*, while describing the joystick as *useless, unrealistic, too hard* or *meaningless*. On the other hand some participants clearly preferred the joystick describing it as *natural, intuitive, interesting, expressive* or *humane*, but everyone mentioned that the sensitivity of the joystick is too high, making it hard to land on the desired pitches. Due to the incremental nature of the damping fingers' position, it was impossible to skip over notes, a fact that was mentioned in different forms by several participants. Some noted that the control of the damping finger ('Y' for up the string and 'X' for down) should have been inverted. Furthermore, some would have liked a more physical interaction for the damping finger, such as moving the controller up and down rather than using buttons.

When it comes to the haptic feedback, the majority was satisfied with it, mentioning that "it feels nice", "it feels good", "is great", "impressive - it felt natural", or "it feels real", while one participant found it to be "wild and a bit too powerful". A special case related to the haptic feedback was the bounce obtained by hitting the virtual string with the bow. Most subjects found it pleasing and were intrigued by its realistic feel, but the violin player repeatedly mentioned that it is "unrealistic and way to powerful". One participant explicitly mentioned that the haptic feedback matches the auditory one, and his expectations.

Several participants noticed that the bow could rotate along its axis and asked whether it made a sonic difference or not, to which they were answered negatively. Besides that, there were very few comments regarding the visual aspect of the system, but most of these were positive. One participant mentioned that sometimes there's a gap between the string and the bow, and that it would be nice to observe one's hands. No one mentioned anything related to the visual indication or the damping finger seen in Figure 3.

The overall interaction was described offering a high degree of freedom on the bowing hand, but the pitch selecting hand was either not mentioned, or described as *disconnected* several times. Some agreed that the instrument is hard to play, mentioning that it is *frustrating*. However, most people estimated that they can perform better after practising more.

5 Discussion

In this section, both the evaluation procedure itself and the results from Section 4 will be discussed.

5.1 Procedure

It is acknowledged that the cognitive load of speech and playing an instrument are overlapping [29], therefore the *think aloud protocol* might have not generated in the most abundant data possible. Most participants alternated between playing and speaking. This resulted in occasionally long breaks in either activities, and required participants to be encouraged to think aloud. Retrospectively, a structured activity schedule allocating time for playing and feedback could have been more productive. Similarly, using self report through Likert scales require large sample sizes to achieve a high level of accuracy[29]. Therefore the interpretation of results rooted into the qualitative data, and then validated using the quantitative data.

Furthermore, as the data we obtained was purely through non-intrusive methods, it would have been useful to log the raw data provided by the Omni (such as the bowing pressure). This could then have been analysed to obtain a better understanding of the user's feedback.

Lastly, the audio did not fully match the sounds that were possible to create with the application. As the recording was quite distorted, the volume of the audio plugin was turned down during the test, but the recording was not remade. This will be elaborated on below.

5.2 User Feedback

The generally positive oral feedback about the overall experience is backed up by the quantitative data which showed that participants were satisfied and not bored with the instrument. They attempted to perform fundamental tasks as producing a sustained tone or playing simple melodies with various degrees of success, and when exposed to the example recording, some tried to recreate the sounds heard from the audio clip. Several participants mentioned that it was difficult to achieve this particular goal, a problem that finds its explanation in the difference in volume between the recording and the experiment scenario as mentioned above. This would be a point of improvement for future testing, as it could have impacted the answers for question 5 – the lowest scoring question regarding the overall experience. Another reason for this question's answers could be linked to the inability to play the higher notes, or the limited pitch range, as presented in Section 4, but these characteristics are inherited from the physical characteristics of the real instrument, so could be expected.

Interestingly, many participants believed that it was easy to understand how to play the instrument, but that they could become better after some more practice. This indicates that the setup has a low "entry-level", with an envisioned high virtuosity ceiling. This is believed something desirable when creating computer based instruments [30]. Even though the implementation was inspired by a real instrument with a possibly different learning curve, as mentioned, it was not our goal to recreate it.

The haptic feedback was considered positive and generally having an appropriate level of resistance to movements. The answers to questions 10 (realistic haptic feedback) and 12 (natural haptic feedback) correlate positively and question 15 (sound matching the haptic feedback) was also answered positively, giving a strong indication that the participants considered the haptic feedback real and according to their expectations. It can be understood that the realism of the haptic feedback is estimated considering the multisensory experience and this result reassures that bowing in VR with our setup is possible.

Furthermore, many participants noticed that they could 'bounce' the bow onto the string. Even though this behaviour was a byproduct of the implementation, participants generally liked this interaction and found it to be realistic and exciting.

Many users noted that the full range of the bow could not be used. As mentioned in Section 2.5, the virtual tromba marina was aligned with the physical position of the Omni and as the bow is held at one end, about half of the range could not be used for bowing. This was a commonly reported issue, and it could have impacted the answers for questions 4, 5, and 9. The reason for aligning the physical setup with the virtual tromba marina was the tilting interaction, so that if people wanted to interact with the entire instrument, they would be able to grab the physical setup. As none of the participants used this, we could discard the aforementioned alignment to be able to account for the entire range of the bow.

The polarisation of the participants' opinion on the pitch control – joystick versus buttons – was backed up by the answers individuals gave on question 9 (interaction was frustrating). It could be argued that users preferring the joystick over the buttons had a harder time interacting with the instrument than the people preferring the buttons. As mentioned, all participants who mentioned the joystick interaction said it was too fast, explaining the above.

6 Conclusions

This paper presents a virtual reality implementation of the tromba marina and its evaluation. Our goal was to evaluate the general experience of bowing in VR and to evaluate the playability of our implementation. The results show that the implementation was successful with participants finding the haptic feedback realistic and the general experience enjoyable and interesting on one hand, and difficult and frequently frustrating on the other hand. Nevertheless, all sensory modalities we focused on (auditory, haptic and visual) seemed to reinforce each other, inspiring participants to attempt to play melodies with the instrument. This was considered to be an important achievement. Improvements on our application include the pitch control, which should either be more physical, i.e., moving the pitch hand physically up and down the virtual string, or simply slower continuous control. Besides that, a better physical setup, allowing the users to utilise the entire bow is desired. The findings of this paper prove that it is possible to create a satisfactory bowed VRMI using off-the-shelf hardware and accurate physical modelling.

Acknowledgments

We would like to thank Peter Williams for his valuable insights on playing the tromba marina, feedback on our application and providing access to his instrument replica.

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

7 References

[1] The Editors of Encyclopædia Britannica, *Encyclopædia Britannica*. Encyclopædia Britannica, Inc., 2020.

[2] D. K. Pai, "Multisensory interaction: Real and virtual," *Robotics Research. The Eleventh International Symposium*, pp. 489–498, 2005.

[3] F. Danieau, A. Lécuyer, P. Guillotel, J. Fleureau, N. Mollet, and M. Christie, "Enhancing audiovisual experience with haptic feedback: a survey on hav," *IEEE transactions on haptics*, vol. 6, no. 2, pp. 193–205, 2012.

[4] M. S. O'Modhrain, "Playing by feel: incorporating haptic feedback into computer-based musical instruments," Ph.D. dissertation, Stanford University, 2001.

[5] C. Nichols, "The vbow: development of a virtual violin bow haptic human-computer interface," pp. 1–4, 2002.

[6] J.-L. Florens and C. Cadoz, "Modèles et simulation en temps réel de corde frottée," *Le Journal de Physique Colloques*, vol. 51, no. C2, pp. C2–873–C2–876, 1990.

[7] A. Luciani, J.-L. Florens, and N. Castagné, "From action to sound: a challenging perspective for haptics," pp. 592–595, 2005.

[8] J. Villeneuve and J. Leonard, "Mass-interaction physical models for sound and multi-sensory creation: Starting anew," *Proc. Int. Conf. Sound and Music Computing*, 2019.

[9] S. Sinclair, G. P. Scavone, and M. M. Wanderley, "Audio-haptic interaction with the digital waveguide bowed string," *ICMC*, 2009.

[10] A. Askenfelt and E. V. Jansson, “On vibration sensation and finger touch in stringed instrument playing,” *Music Perception: An Interdisciplinary Journal*, vol. 9, no. 3, pp. 311–349, 1992.

[11] 3D Systems, Inc, “3D Systems Touch Haptic Device,” accessed February 12, 2020, available at https://www.3dsystems.com/haptics-devices/touch.

[12] F. Avanzini and P. Crosato, “Haptic-auditory rendering and perception of contact stiffness,” *Lecture Notes in Computer Science*, vol. 4129, pp. 24–35, 2006.

[13] F. Avanzini and D. Rocchesso, “Physical modeling of impacts: theory and experiments on contact time and spectral centroid,” *Proc. Int. Conf. Sound and Music Computing*, pp. 287–293, 2004.

[14] A. Passalenti, R. Paisa, N. Nilsson, N. Andersson, F. Fontana, R. Nordahl, and S. Serafin, “No strings attached: Force and vibrotactile feedback in a virtual guitar simulation,” *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1116–1117, 2019.

[15] ——, “No strings attached: Force and vibrotactile feedback in a guitar simulation,” *Proc. Int. Conf. Sound and Music Computing*, 2019.

[16] F. Fontana, R. Paisa, R. Ranon, and S. Serafin, “Multisensory plucked instrument modeling in Unity3D: From keytar to accurate string prototyping,” *Applied Sciences*, vol. 10, 2020.

[17] S. Serafin, C. Erkut, J. Kojs, N. Nilsson, , and R. Nordahl, “Virtual reality musical instruments: State of the art, design principles, and future directions,” *Computer Music Journal*, 2016. [Online]. Available: http://www.mitpressjournals.org/doi/pdfplus/10.1162/COMJ_a_00372

[18] Facebook Technologies, LLC, “Oculus Rift: VR Headset for VR-ready PCs | Oculus,” accessed March 6, 2020, available at https://www.oculus.com/rift/.

[19] Razvan P, “Resurrecting the Tromba Marina,” accessed March 6, 2020, available at https://www.youtube.com/watch?v=SlHqvaaPCyU.

[20] S. Willemsen, S. Serafin, S. Bilbao, and M. Ducceschi, “Real-time implementation of a physical model of the tromba marina,” *submitted to the 17th Sound and Music Computing (SMC) Conference*, 2020.

[21] Unity Technologies, “Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations,” accessed February 10, 2020, available at https://unity.com/.

[22] A. Baldwin, T. Hammer, E. Peciulis, P. Williams, D. Overholt, and S. Serafin, "Tromba moderna: A digitally augmented medieval instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, vol. 16, pp. 14–19, 2016.

[23] 3D Systems Inc., "Openhaptics unity plugin user guide (toolkit version 3.5.0)," 2018, available at http://s3.amazonaws.com/dl.3dsystems.com/binaries/Sensable/OH/3.5/OpenHaptics_Toolkit_ProgrammersGuide.pdf.

[24] J. Barbosa, J. Malloch, M. Wanderley, and S. Huot, "What does "evaluation" mean for the NIME community?" *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2015.

[25] D. Young and S. Serafin, "Playability evaluation of a virtual bowed string instrument," *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pp. 104–108, 2003.

[26] M. Someren, Y. Barnard, and J. Sandberg, *The Think Aloud Method - A Practical Guide to Modelling Cognitive Processes*. London: Academic Press, 1994.

[27] K. Finstad, "The usability metric for user experience," *Interacting with Computers*, vol. 22, pp. 323–327, 2010.

[28] S. K. Card, A. Newell, and T. P. Moran, *The Psychology of Human-Computer Interaction*. New Jersey: Lawrence Erlbaum Associates Publishers, 1983.

[29] D. Stowell, A. Robertson, N. Bryan-Kinns, and M. Plumbley, "Evaluation of live human–computer music-making: Quantitative and qualitative approaches," *International Journal of Human-Computer Studies*, vol. 67, pp. 960–975, 2009.

[30] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, 2002.

Paper F

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study

Silvin Willemsen, Anca-Simona Horvath and Mauro Nascimben

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp. 292–299, 2020.

Abstract

This paper presents DigiDrum – a novel virtual reality musical instrument (VRMI) which consists of a physical drum augmented by virtual reality (VR) to produce enhanced auditory and haptic feedback. The physical drum membrane is driven by a simulated membrane of which the parameters can be changed on the fly. The design and implementation of the instrument setup are detailed together with the preliminary results of a user study which investigates users' haptic perception of the material stiffness of the drum membrane. The study tests whether the tension in the membrane simulation and the sound damping (how fast the sound dies out) changes users' perception of drum membrane stiffness. Preliminary results show that higher values for both tension and damping give the illusion of higher material stiffness in the drum membrane, where the damping appears to be the more important factor. The goal and contribution of this work is twofold: on the one hand it introduces a musical instrument which allows for enhanced musical expression possibilities through VR. On the other hand, it presents an early investigation on how haptics influence users' interaction in VRMIs by presenting a preliminary study.

1 Introduction

Virtual Reality (VR) is described as an immersive environment provided by technology and experienced through sensory stimuli [1]. Different types of technologies are available for creating VR experiences, and head-mounted displays (HMDs) are among the most popular. VR has been used as a platform for the creation of perceptual illusions, and much research has gone into producing realistic or otherwise compelling visual and auditory experiences. By comparison, the sense of touch has been neglected in spite of its obvious potential to increase a sense of presence in a simulated world [1].

Virtual musical instruments (VMIs) are defined as software simulations or extensions of existing musical instruments with a focus on sonic emulation. Virtual reality musical instruments (VRMIs), are those which also include a simulated visual component [2].

The design and evaluation of DigiDrum – a novel VRMI where a physical darbuka (a djembe-like drum) is enhanced by VR is presented. The user wears a HMD which puts them in a recording studio where a virtual drum is aligned with the physical drum, so that both drums can be played at the same time. Interaction with the (physical + virtual) drum triggers a virtually simulated sound of a drum membrane. This sound is sent to the user through sound-isolating headphones for auditory feedback, and a vibration motor (haptuator) attached to the inside of the physical drum's membrane creating a vibrotactile response in the physical drum – similar to the haptic response a real membrane would produce. As the drum's sound is being simulated, its properties can be changed on the fly, something which is impossible to do in the physical world.

An initial user study was conducted on DigiDrum with a twofold goal. On the one hand, in order to study how users interact with the installation and use this feedback to improve the drum, and on the other hand, for trying to understand whether there is a correlation between material stiffness perception and the way users interact with the drum. More specifically, the study investigated which parameters influence the perception of the material stiffness of the drum membrane. Different combinations of values for: (1) tension in the virtual membrane and (2) damping, or how quickly the sound dies out were used. The initial hypothesis was that higher values for both tension and damping would influence the perception of stiffness positively. In other words, higher tension and higher damping (sound dying out faster) will result in users perceiving the drum membrane as being more stiff. It was suspected that tension would be the most important parameter in the perception of stiffness. In the test, the auditory and haptic cues were linked, or matching.

The research question which guides this work is:

Can a user's perception of
material stiffness in an enhanced drum membrane change
by using auditory and haptic cues?

The ultimate goal of the paper is to (1) present an installation which helps to enhance musical expression possibilities through a novel VRMI, and (2) investigate users' interaction with a VRMI focused not only on the visual and auditory experience, but also on haptics.

The paper is structured as follows: Section 2 presents a selection of related work. Section 3 is an introduction to haptic perception. Section 4 describes the design criteria used in for DigiDrum. In Section 5 we describe the system overview and Section 6 details the implementation of the visual virtual environment. In Section 7, the physical model sound algorithm is described. In Section 8 a user study looking at the interaction with the setup is presented and preliminary results are shown and discussed in Section 9. Conclusive remarks and future development are made in Section 10.

2 Related Work

Several investigations on the connection between haptic and auditory cues and perception of material stiffness have been done. Some look specifically at playing percussion musical instruments as the music community has long had a strong interest in haptic technology [3], where others investigate haptics, visuals and sound in relation to human computer interaction.

In [4], Sile O'Mohrain describes a series of studies where experienced musicians played VMIs with both haptic and auditory feedback with the aim of finding out whether adding haptic feedback to these instruments would improve their playability. The results indicate that the presence of haptic feedback can improve a player's ability to learn the behavior of a VMI.

In [5], Dahl gives a detailed analysis of four experienced drummers performing the same musical sequence using drumsticks on drums with three striking surfaces (soft, medium and hard). The study finds that the main parameter influencing the preparatory movement and the striking velocity was the dynamic level and, to a lesser extent, the striking surface.

The work of Avanzini and Crosato's [6], that of Passalenti *et al.* [7], and that of Liu et. al. use the haptic device PHANTOM® Omni$^{\text{TM}}$ (now Touch) [8]. Avanzini and Crosato test the influence of haptic and auditory cues on perception of material stiffness separately, in an experiment where subjects had to tap on virtual surfaces, and were presented with audio-haptic feedback. In each condition the haptic stiffness had the same value while the acoustic stiffness was varied. The study indicates that subjects consistently ranked surfaces according to the auditory stimuli. Passalenti *et al.*'s experiment focuses on haptics and guitar strings. In [9], a multimodal interface that synchronizes visual, haptic and auditory stimuli to give users a feeling of presence of virtual objects is presented and thoroughly detailed. The study notices that although the stiffness parameters of different materials were set to be the same, the sound effects biased user's judgment of the hardness of surfaces.

The preliminary experiment conducted in relation to DigiDrum takes inspiration from these works, but looks at the specificity of a VR-enhanced drum.

In [10], the design of a physically intuitive haptic drumstick is presented. The paper suggests that physically intuitive new musical instruments may help performers transfer motor skills from familiar, traditional musical instruments.

3 Haptics

The sense of touch is the first to develop in humans – a sense we cannot shut down. Vision is the last sense to develop, a sense we are able to "turn off" [11] by closing our eyes. Despite this, tactile awareness generally receives less attention than other sensory modalities when it comes to technological development [12]. We live in a world over-saturated by visuals, and VR is a technology where this has been the case notably. In this section, we describe in further detail haptic perception and how it works from a neurophysiological point of view as well as the basis for subjective decision making on tactile sensation.

3.1 Haptic Perception

The peripheral nervous system gathers environmental stimuli in form of visual, audible, tactile, olfactory (smell) and gustatory (taste) inputs and transfers them to the central nervous system for further elaboration and integration. Tactile information is collected in the skin, muscles, and joints and sent to an area in the brain called the primary somato-sensory cortex[13]. This cortical area is the first stage for the tactile awareness occurring across the surface of the body. Several other structures of the central nervous system take part in the generation of tactile feedback, as generally, a single brain area is never responsible for information awareness [14]. Light touch and tactile attention are processed in the secondary somato-sensory cortex – an area directly connected with the primary somato-sensory cortex [15]. Literature reports that people undergoing tactile training improve their perception but also strengthen the connections and cortical representations of the stimulated body area [16]. There is a direct relationship between size of cortical region and haptic performance.

A specific area of the central parietal lobe, placed in the back of the primary somato-sensory cortex, integrates the information from the visual and haptic regions to help locate objects in space.

The sense of hearing is connected to the sense of touch and touching objects in different ways produce abundant sounds which convey information about the object and the interaction, such as material, shape, roughness, stiffness, the gesture, rate and strength of our actions. In VR systems, users may immediately notice the unnaturalness if the interface has no sound or provides mismatched sound [9].

As Cao *et al.* explain in [17], skilled interactions with sounding objects, such as drumming, rely on resolving the uncertainty in the acoustical and tactual feedback signals generated by vibrating objects.

3.2 Notes on Experiments Involving Haptics

Conducting experiments on haptics can prove difficult because there are no proper technological devices for delivering controlled and reliable tactile stimuli [12]. When users interact with a physical object, uncertainty may arise from mis-estimation of the objects' geometry-independent mechanical properties, such as surface stiffness. How multisensory information feeds back into the fine-tuning of sound-generating actions remains unexplored [17].

In virtual environments (as used in VR) and using hand tracking devices such as Leap Motion [18] (see Section 5), subjects are able to move their hands freely, which could confound somato-sensory processing with activations related to motor planning and movement [19]. These uncontrolled motor activities result in uncontrolled somatic stimulation. There is an anatomical explanation of this close somato-motor functional relationship: areas involved in the perception of touch on the hands in the primary somato-sensory cortex are located mostly in front of the areas responsible for hand movements [20]. Another problem with haptics is the subjective quantification of the stimuli. Contents of tactile consciousness vary between individuals and a common lexicon to evaluate haptic sensation through surveys still seems far to be conceived [21].

3.3 Interaction between Visual Information and Tactile Feedback

In a famous experiment, Pavani *et al.* [22], asked a group of participants to detect the position of vibro-tactile stimuli on their arm. The participant's own arm was placed under a table (out of sight) while a fake rubber hand was laid in front of them. The rubber hand was laid out in a position that was anatomically compatible with participant's real hand. When seeing the mannequin hand being touched, all participants reported that their own hand was being touched, even though that was not the case. In short, the perception of tactile stimulation was simulated through visuals. A similar experiment was conducted by [23] asking subjects to watch a video of a hand being touched on the first finger while their own hand was stimulated synchronously. Brain activity during synchronous stimulation showed an improved tactile acuity. Taking into account previous literature findings, we can conclude that in virtual environments hand manipulations and interactions are important factors that enhance realism and user experience.

4 Design Criteria for DigiDrum

As explained by [10], a new musical instrument is physically intuitive if the physics of haptic interaction are similar to those supported by a traditional musical instrument. Physically intuitive new musical instruments may help performers transfer motor skills from familiar, traditional musical instruments. This is why we choose to augment an existing drum, instead of suggesting a completely new musical instrument – seeing a physical drum will invite users to play the new instrument in an intuitive way and as a regular drum. The mechanics of a musical instrument's interface – what the instrument feels like – determines much of its playability [24].

In creating DigiDrum, the design criteria for VRMIs suggested by Serafin *et al.* were used as guidelines [2]. The setup integrates visuals, audio and haptics and extends an existing musical instrument using VR seeking to create a "magical interaction". Creating a sense of presence is attempted by mapping the virtual drum's location to that of the physical one, and by representing the user's hands in the simulated world. DigiDrum was designed to create

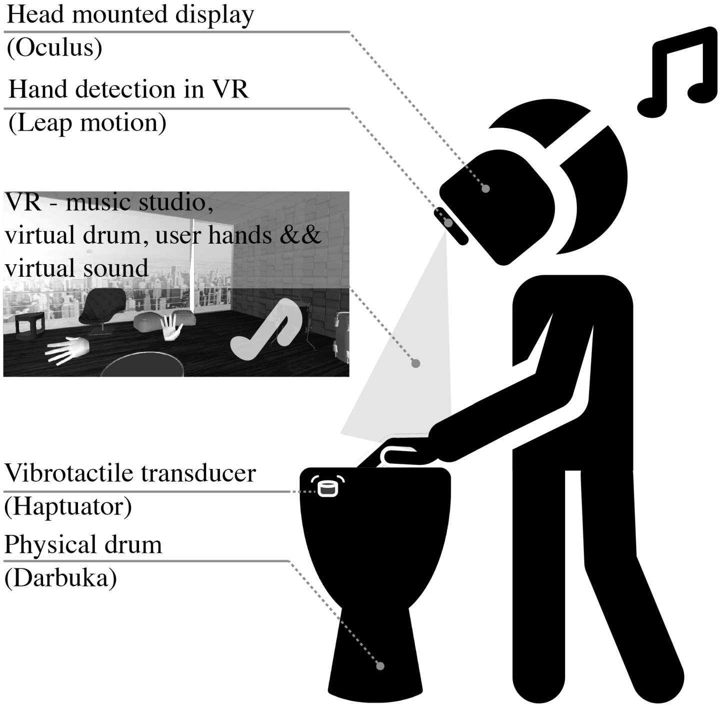


Fig. 1: The physical setup of the system. The Leap Motion is mounted to the front of the HMD.

three types of illusions: (1) a place illusion – users should feel like they are in a music production studio, (2) a plausibility illusion – users should feel like the experience is really happening, and (3) virtual body ownership – users should see their own body in the virtual world and feel ownership of their virtual body.

5 System Overview

Figure 1 shows the overall design of DigiDrum and its setup and Figure 2 shows a user interacting with the setup. For hand-tracking, the Leap Motion [18], which is an infrared-sensor-based camera that allows for accurate hand tracking is used. It is mounted to the front of an Oculus Rift HMD so that the user's hands are in the field of view when they look at the virtual drum. The drum is fixed in-place and played like a djembe. In the application, the virtual drum was placed slightly higher than the physical drum to make sure the physical model was triggered when the physical drum was hit.

A detailed overview of the system is given in Figure 3. The hand movement

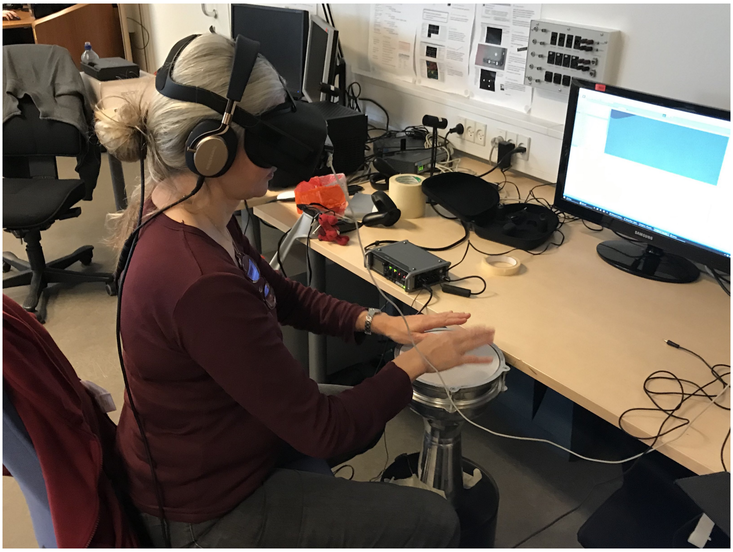


Fig. 2: A user interacting with the setup.

data is retrieved by a PC which runs the cross-platform game engine Unity [25]. The Unity 'scene' contains the virtual environment (see Section 6) that the user will see through the HMD and the physical model used for the sound and haptics (see Section 7). The HMD also sends data back to the PC regarding location and head rotation. Once the tracked hand touches (or collides with) the virtual drum, the physical model is triggered and its output sound is sent to a haptuator which is attached to the inside of the drum membrane. This effectively causes the physical membrane to be actuated by a virtual membrane. To accommodate for the plausability illusion mentioned in Section 4, the chosen haptuator has a very high fidelity, i.e. can play realistic audio signals as opposed to non-realistic 'buzzes'. Other forms of haptic feedback have been considered, but – according to the authors – the use of this actuator attached to the drum membrane had the highest potential of resembling realistic drum membrane vibration in the end.

Finally, the same sound that is sent to the haptuator is also sent to sound-isolating headphones. Sound-isolation is important as the sound coming from the physical drum should not interfere with the audio coming from the simulated drum.

6 Unity Implementation

The virtual environment was created using Unity. All the hardware drivers and software components were linked together using this platform. Here, a

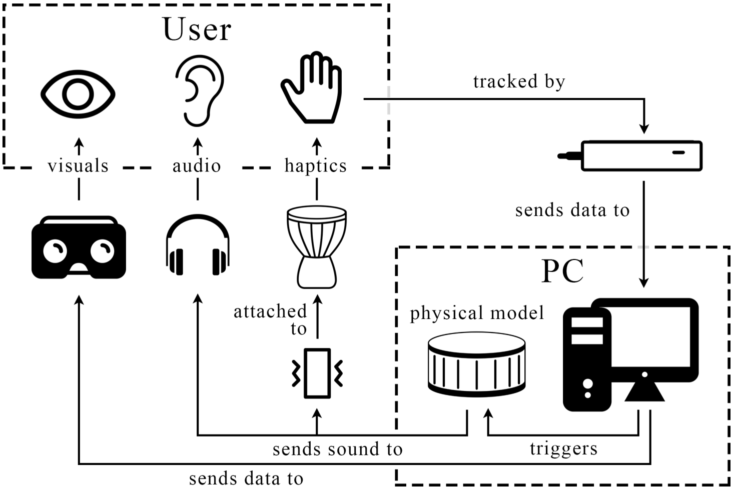


Fig. 3: Detailed system layout. The user interacts with the system using their hands and gets haptic feedback from the haptuator attached to the drum membrane, auditory feedback from closed headphones and visual feedback from the Oculus Rift headset. A detailed explanation can be found in Section 5.

virtual drum playable with hand motion using Leap Motion was created. The user enters the VR environment (rendered as a recording studio) and Leap Motion reconstructs (in VR) the subject's own hands. In the virtual recording studio a drum was placed at the center and programmed to detect collision with the reconstructed hands. When a collision was detected, a C# script, in which a physical model of a drum membrane was programmed, was activated to reproduce the beating sound of the drum through an actuator placed inside the drum skin.

7 Physical Model

The behaviour of musical instruments can be well described by partial differential equations (PDEs) [26]. In this section, the continuous-time PDE for a drum-membrane is given and explained. This is followed by an explanation of the discretisation method used. Finally, the parameter values used for the implementation are given.

7.1 Continuous Time

A rectangular (stiff) membrane with dimensions L_x (m) and L_y (m) can be described by the following equation [27]:

$$\rho H \frac{\partial^2 u}{\partial t^2} = T\Delta u - D\Delta\Delta u - 2\sigma_0 \frac{\partial u}{\partial t} + 2\sigma_1 \Delta \frac{\partial u}{\partial t}. \tag{1}$$

Here, state variable, $u = u(x, y, t)$ is a function of horizontal coordinate $x \in [0, L_x]$, vertical coordinate $y \in [0, L_y]$ and time $t \geq 0$ and is parameterised in terms of material density ρ (kg/m^3), membrane thickness H (m), tension T (N) and frequency independent and dependent damping coefficients σ_0 (s^{-1}) and σ_1 (m^2/s). Furthermore, $D = EH^3/12(1-\nu^2)$ with Young's modulus E (Pa) and Poisson's ratio ν. Lastly, Δ represents the 2D Laplacian [27]:

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{2}$$

Furthermore, clamped boundary conditions – i.e., the state u at all plate edges and their gradients are 0 – have been chosen for simplicity:

$$u = \nabla u = 0 \quad \text{with} \quad \nabla = \frac{\partial}{\partial x} + \frac{\partial}{\partial y}. \tag{3}$$

7.2 Discretisation

For implementing the physical model, finite-difference time-domain (FDTD) methods were used [27]. These methods were chosen over others, such as the 2D waveguide mesh [28], as they allow parameters and real-time changes of these to be better controlled. FDTD methods discretise $u(x, y, t)$ shown in Equation (1) to $u^n_{(l,m)}$ using $t = nk$ with sample n and time step k (s), $x = lh$ where $l \in [0, ..., N_x - 1]$ and $y = mh$ where $m \in [0, ..., N_y - 1]$ where N_x and N_y are the number of horizontal and vertical grid points respectively. Furthermore, grid spacing h (m) can be calculated using

$$h \geq h_{\text{min}} = 2\sqrt{\frac{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 4\kappa^2 k^2}}{2}}, \tag{4}$$

where $c = \sqrt{T/\rho H}$ and $\kappa = \sqrt{D/\rho H}$. The closer h is to h_{min}, the higher the accuracy of the implementation.

7.3 Parameters

Most parameters used in the simulation were chosen empirically and can be found in Table 1. With these parameters a small (30×30 cm) membrane with a low density and stiffness is simulated. For the purpose of getting the model to work in real time, the minimum grid spacing h_{min} in Equation (4) is multiplied by 4 (h_{min} in (4) is calculated based on the highest value of T and $\sigma_1 = 0.005$). The values for T and σ_0 correspond to the cases used in the experiment. The

Parameter	Symbol (unit)	Value
Membrane width	L_x (m)	0.3
Membrane length	L_y (m)	0.3
Material density	ρ (kg/m^3)	10
Thickness	H (m)	0.001
Tension	T (N)	$\{15, 40, 80\}$
Young's modulus	E (Pa)	$2 \cdot 10^3$
Poisson's ratio	ν (-)	0.3
Freq. indep. damping	σ_0 (s^{-1})	$\{0.5, 2, 5\}$
Freq. dep. damping	σ_1 (m^2/s)	$[0, 0.005]$
Time step	k (s)	$1/44100$
Grid spacing	h (m)	$4h_{\text{min}}$

Table 1: Table showing parameter values.

frequency dependent damping σ_1 follows an exponentially decaying curve,

$$\sigma_1(t) = 0.005e^{-0.01t}, \tag{5}$$

where $t = 0$ at the time of excitation. This allows for very low damping, i.e., very long sound, while taking away some of the high frequency content present immediately after excitation. This ultimately results in a more natural drum sound, even when σ_0 is set low.

8 User study

This work hopes to add to the corpus of design guidelines for VRMIs, more specifically those VRMIs which involve a touch based stroking movement. In [2], Serafin et al. describe three layers of evaluation for VRMIs, namely: (1) investigating modalities of interaction, (2) evaluating VR specific aspects, with engagement being the most interesting from a VRMI perspective, and (3) looking at quality and goals of interaction.

An initial user study was conducted with a towfold goal: on the one hand - to study how users interact with DigiDrum and create guidelines for improving the setup, and on the other hand to investigate the relationship between tension and frequency independent damping coefficient (T and σ_0 respectively in Section 7) and user's perception of material stiffness.

As shown in Section 7, there are 3 different cases for both tension T and frequency independent damping σ_0. All combinations were tested, resulting in 9 different cases. Sound examples of each individual case can be found in [29]. Participants' experiences were evaluated through both qualitative and quantitative methods, namely by: (1) asking them during the test how they rate the stiffness of the material in each of the 9 cases, (2) a questionnaire including questions about their relationship and experience with playing a musical instrument, virtual body ownership and whether they thought their interaction patterns changed between the different cases and (3) observation while the participants interacted with the setup to retrieve data on engagement and stroke patterns which possibly correlate to the haptics and sound.

8.1 Process for the User Study

Before the experiment, participants were told that they would be “drumming in VR", that their perception of the stiffness of the material they were interacting with was tested and that their performances did not need to be musical in any way. Furthermore, participants were told they would hear 9 different cases in between which the “parameters of the experience" would be changed and that for each of these cases they would have to rate the stiffness of the material they were interacting with on a scale of 1 to 7, 1 being “extremely soft or loose", 7 being “extremely stiff or hard". The order in which the cases were presented was randomised to reduce bias. Between cases, the participants did not take off the headset or headphones, and the authors noted their answers. After the test, the participants filled out a questionnaire with the following questions (the last two taken from [6]):

- I felt like the hands in the simulation were my own. (1-7 rating)
- In order to express your judgements to the questions during the simulation, you relied mainly on... (multiple answers possible: visuals | audio | haptics)
- In your opinion what was varying between each condition? (multiple answers possible: visuals | audio | haptics)

From participant-observation during the experiment and the the final two questions of the questionnaire, “Did your behaviour change between different cases, and if so what did you do differently?" and “Anything you would like to add?", information on the user interaction and the quality of the setup was collected.

The experiment was done on 16 participants, 9 of which were experienced musicians (> 5 years of instrument practice). Three participants were drummers.

9 Results and Discussion

This section will give the results of the user study and discuss these. Due to the small sample size and some issues regarding interaction described at the end of this section, the presented results should be considered preliminary.

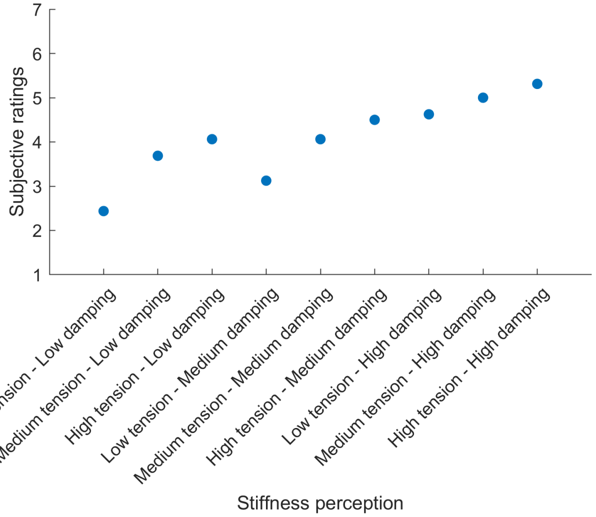


Fig. 4: Relation between stiffness perception and subjective ratings.

9.1 Statistical Analysis

The results of the stiffness ratings can be found in Figure 4. Intriguingly, there was a significant correlation between the cases sorted by damping first and then by tension (both sorted from low to high) and the subjective ratings (ρ = 0.9372, p < 0.01) using Spearman correlation. The Spearman methodology was used because the low number of values did not allow modelling a normal distribution [30]. A quasi-linear relationship between subjective stiffness perception and the values for tension and damping used by the simulation can be observed.

Figure 5 shows the average participant scores for each level of damping and tension both grouped in levels (low, medium and high). As previously hypothesised, the ratings of material stiffness increases with tension and damping.

A statistical analysis was run on each single level based on non-parametric Mann-Whitney U-test with the results reported in Table 2. This test helps to identify significant differences between groups in presence of small samples made by ordinal variables. Abbreviations are T for “tension" and D for “damping" while letters L, M and H mean the levels “low", “medium" and “high". It is important to take into account the multi-comparison problem and in this case the threshold level for significance should be equal to 0.0056 following the Bonferroni correction.

As we can observe from Table 2, there isn’t a significant difference between “high tension – low damping" and “low tension – medium damping" (p =

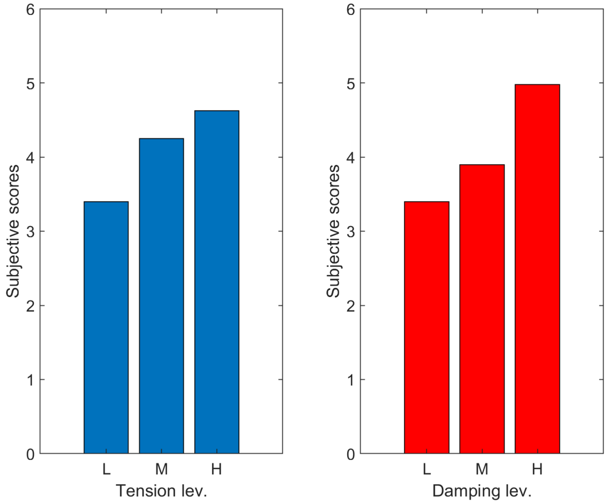


Fig. 5: Subjective ratings grouped by different tension and damping levels.

Mann-Whitney U-test [p-values]									
	LT	MT	HT	LT	MT	HT	LT	MT	HT
	LD	LD	LD	MD	MD	MD	HD	HD	HD
LT LD		.0642	.0163	.1268	.0049	.0041	.0034	.0004	.0002
MT LD	.0642		.6463	.3465	.6582	.1802	.1584	.034	.0091
HT LD	.0163	.6463		.2123	.8933	.5413	.4455	.1737	.0915
LT MD	.1268	.3465	.2123		.0791	.0254	.0283	.0012	.0009
MT MD	.0049	.6582	.8933	.0791		.3191	.3114	.0449	.0174
HT MD	.0041	.1802	.5413	.0254	.3191		.7732	.5214	.1383
LT HD	.0034	.1584	.4455	.0283	.3114	.7732		.7725	.3424
MT HD	.0004	.034	.1737	.0012	.0449	.5214	.7725		.2991
HT HD	.0002	.0091	.0915	.0009	.0174	.1383	.3424	.2991	
Note: Bonferroni adjusted significance threshold for multi-comparison p<0.0056									

Table 2: Mann-Whitney U-test (p-values).

0.2123) suggesting that the linear relation shown in Figure 5 holds despite the discontinuity between points as seen on the scatterplot. However, we should consider it similar to a monotonically increasing function rather than a pure linear trend.

Lastly, Table 3 shows group comparisons between the three different values of damping and tension. A significant difference in participant's ratings between medium-high damping and low-high damping levels can be noticed while tension shows significance only between low to high tension. It can be deducted from the results that damping is a more important factor than tension in material stiffness perception. This result was unexpected, as it was hypothesised that tension would be the most dominant factor in stiffness perception. Additionally, it appears difficult for participants to evaluate low to

Different levels of tension/damping		p-values
Low Damping	Medium Damping	0.2560
Medium Damping	High Damping	0.0078
Low Damping	High Damping	6.5071e-04
Low Tension	Medium Tension	0.0950
Medium Tension	High Tension	0.4268
Low Tension	High Tension	0.0297

Table 3: Comparison between different levels of tension and damping.

medium levels of both damping and tension. In a future test, the values could be chosen differently, or more alternatives for the parameter values could be investigated to better see the perceptual differences between these values.

9.2 Statistical Analysis: Reliability

Individual ratings were initially analysed with Cronbach's alpha [31] to test the internal consistency of the responses. This measure is generally known as a metric to validate a questionnaire with higher values of alpha as those more desirable. The non-standardised Cronbach's alpha value was 0.6348 while the standardised value reached 0.6589. According to [32], a value between 0.6 to 0.7 is questionable (questionnaire scale is not fully reliable) with 0.7 as the threshold for an acceptable test. Despite the outcomes being slightly below threshold (probably caused by subjective difficulties in evaluating stiffness), it appears that in future a good reliability can be reached by increasing the sample size. Moreover, if we don't consider all factors loadings as evenly distributed, we could assume that the Cronbach's alpha underestimates the true reliability.

9.3 Questionnaire

The questionnaire results in Table 4 show that the participants generally found that the hands in the simulation were their own. This proves that the Leap Motion is a good way to track the hands and that it was well implemented. The visuals had no influence on participants' judgement, probably because they were unchanged. The audio seemed to be the most predominant feature the participants focused on when expressing their judgements (93.8%). Haptics for expressing judgements was only chosen by 5 participants (31.3%). In the future, removing the audio, only leaving the haptics might be a better way to force the participants to use their sense of touch and test the influence of this modality on perception.

Question	Result
I felt like the hands in the simulation were my own. (1-7 rating)	$\mu = 5.44$, $\sigma = 1.26$
In order to express your judgements to the questions during the simulation, you relied mainly on... (visuals | audio | haptics)	visuals: 0, audio: 15, haptics: 5
In your opinion what was varying between each condition? (visuals | audio | haptics)	visuals: 0, audio: 14, haptics: 10

Table 4: Questionnaire results. The last two questions were taken from [6].

9.4 Qualitative Observations

From participant observation during the experiment, comments they gave during and after the test, and the two last (open) questions of the questionnaire (see Section 8) additional findings were compiled.

Due to the fact that the virtual drum was placed slightly higher than the physical drum (see Section 5), many participants interacted with the air above the drum rather than finishing their stroke to actually hit the drum. This was an issue, as the haptic sensation would not be felt in that case. This might also explain the result of the second question in Table 4. Either before or during the test, the participants were instructed to finish their stroke to actually physically interact with the drum.

The interaction was programmed in such a way, that when a tracked hand collides with the virtual drum, this hand would not be able to trigger the physical model until it was completely out of the “collision zone". Due to the misalignment mentioned above, many interactions were not captured. Again, either before or during the experiment, the participants were instructed to make longer movements to ensure that their hands were completely outside of this “collision zone" before interacting with the drum again.

Another technical issue was that sometimes participants would look forward rather than down to the hands. This caused the hands not to be tracked anymore as the Leap Motion was mounted on the HMD. A solution for this would be to mount it at a lower angle rather than straight forward (as is the current case).

The experiment could be improved by addressing the above interaction issues to yield stronger data. The issues could potentially be solved by adding a more precise and reliable sensor to the setup, such as a contact microphone placed on the drum membrane. Even though a feedback loop could occur due to the haptuator being present on the same membrane, there is a potential to filter out its vibrations and only use the transients due to the interaction with the membrane for control.

Some participants commented that they would have liked to have reference points for “the stiffest" and “the softest" cases before testing as they said they would have judged the first few cases differently if they had known these references in advance. This could, however, bias the participants’ answers.

The movements of participants were observed during the test and sporadically noted. There was a small tendency towards slower and longer movements in the case of lower tension and faster and shorter movements in the opposite case, but as these observations were not done systematically, to be able to say anything about this, this should be properly tested, possibly using raw data from the hand tracking.

10 Conclusion

In this paper, we presented and evaluated a novel VRMI where a physical drum was enhanced by VR. The physical drum was augmented by a vibration motor and the sound was simulated using a physical model of a drum membrane. In an experiment run during the study, preliminary results show that higher values for both tension and damping increase the perception of material stiffness of the drum membrane, as hypothesised. However, the damping appeared to be a more important factor in this perception than the tension, which was contrary to expectations.

In future work, improving the experiment by, for example, adding a contact microphone to the membrane for more accurate control and re-conducting the experiment with a larger sample size will be necessary to validate or improve the results presented in this paper.

Other future work includes decoupling the audio and the haptics, to test the perceptual influence of each individual modality separately. More alternatives of the parameter values could be presented in a future test to more deeply investigate the connection between parameter values and stiffness perception.

Additionally, the tracking of the user’s hands should be improved by mounting the Leap Motion more downwards on the HMD. Furthermore, the virtual and physical drum should be better aligned in space as to make the interaction less confusing and more intuitive. Lastly, in order to test whether the interaction patterns change depending on the changes in parameters, the raw data from the hand tracking should be analysed.

Acknowledgments

The authors wish to thank Stefania Serafin, Cumhur Erkut, Niels Nilsson, Rolf Nordahl, and Michele Geronazzo – our teachers of the “Virtual, Augmented, Mixed realities" PhD course at Aalborg University Copenhagen – for their

help, and all the participants who were kind enough to participate in our experiment.

11 References

[1] S. Serafin, N. Nilsson, C. Erkut, and R. Nordahl, "Virtual reality and the senses - whitepaper," *Danish Sound Innovation Network*, 2017.

[2] S. Serafin, C. Erkut, J. Kojs, N. Nilsson, , and R. Nordahl, "Virtual reality musical instruments: State of the art, design principles, and future directions," *Computer Music Journal*, 2016.

[3] E. Berdahl, H.-C. Steiner, and C. Oldham, "Practical hardware and algorithms for creating haptic musical instruments," in *NIME*, 2008.

[4] M. S. O'Modhrain, *Playing by Feel: Incorporating Haptic Feedback into Computer-Based musical Instruments*. Ph.D. Dissertation, Stanford University, 2001.

[5] S. Dahl, "Playing the accent - comparing striking velocity and timing in an ostinato rhythm performed by four drummers," *Acta Acustica united with Acustica*, vol. 90, pp. 762 – 776, 2004.

[6] F. Avanzini and P. Crosato, "Haptic-auditory rendering and perception of contact stiffness," *Lecture Notes in Computer Science*, vol. 4129, pp. 24–35, 2006.

[7] A. Passalenti, R. Paisa, N. Nilsson, N. Andersson, F. Fontana, R. Nordahl, and S. Serafin, "No strings attached: Force and vibrotactile feedback in a virtual guitar simulation," *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1116–1117, 2019.

[8] 3D Systems, Inc, "3D Systems Touch Haptic Device," accessed November 4, 2019, available at https://www.3dsystems.com/haptics-devices/touch.

[9] J. Liu and H. Ando, "Hearing how you touch: Real-time synthesis of contact sounds for multisensory interaction," *Conference on Human System Interactions*, pp. 275–280, 2008.

[10] E. Berdahl, B. Verplank, J. O. Smith, and G. Niemeyer, "A physically intuitive haptic drumstick," in *ICMC*, 2007.

[11] K. Barnett, "A theoretical construct of the concepts of touch as they relate to nursing," *Nursing Research*, vol. 21, pp. 102–110, 1972.

[12] A. Gallace, M. K. Ngo, J. Sulaitis, and C. Spence, "Multisensory presence in virtual reality: possibilities & limitations," *Multiple sensorial media advances and applications: New developments in MulSeMedia*, pp. 1–38, 2012.

[13] M. Blatow, E. Nennig, A. Durst, K. Sartor, and C. Stippich, "fMRI reflects functional connectivity of human somatosensory cortex," *Neuroimage*, vol. 37, pp. 927–936, 2007.

[14] T. Manzoni, F. Conti, and M. Fabri, "Callosal projections from area SII to SI in monkeys: Anatomical organization and comparison with association projections," *Journal of Comparative Neurology*, vol. 252, pp. 245–263, 1986.

[15] S. B. Eickhoff, A. Schleicher, K. Zilles, and K. Amunts, "The human parietal operculum. I. Cytoarchitectonic mapping of subdivisions," *Cerebral cortex*, vol. 16, no. 2, pp. 254–267, 2005.

[16] D. N. Saito, T. Okada, M. Honda, Y. Yonekura, and N. Sadato, "Practice makes perfect: The neural substrates of tactile discrimination by Mah-Jong experts include the primary visual cortex," *BMC Neuroscience*, vol. 7, no. 79, 2007.

[17] Y. Cao, B. L. Giordano, F. Avanzini, and S. McAdams, "The dominance of haptics over audition in controlling wrist velocity during striking movements," *Experimental Brain Research*, vol. 234, pp. 1145—1158, 2016.

[18] UltraLeap Ltd, "Leap Motion," accessed November 4, 2019, available at https://www.leapmotion.com/.

[19] A. Bodegard, S. Geyer, C. Grefkes, K. Zilles, and P. Roland, "Hierarchical processing of tactile shape in the human brain," *Neuron*, vol. 31, pp. 317–328, 2001.

[20] W. Penfield and T. L. Rasmussen, *The cerebral cortex of man: A clinical study of localization of function*. London: Macmillan, 1950.

[21] A. Gallace and C. Spence, *Touch and the body: The role of the somatosensory cortex in tactile awareness*. Psyche: An Interdisciplinary Journal of Research on Consciousness, 2010.

[22] F. Pavani, C. Spence, and J. Driver, "Visual capture of touch: Out-of-the-body experiences with rubber gloves," *Psychological Science*, vol. 11, pp. 353–359, 2000.

[23] M. Schaefer, H. Flor, H. J. Heinze, and M. Rotte, "Dynamic modulation of the primary somatosensory cortex during seeing and feeling a touched hand," *Neuroimage*, vol. 29, pp. 587–592, 2006.

[24] S. O'Modhrain and R. B. Gillespie, *Once More, with Feeling: Revisiting the Role of Touch in Performer-Instrument Interaction*, 2018.

[25] Unity Technologies, "Unity Real-Time Development Platform | 3D, 2D VR & AR Visualizations," accessed November 4, 2019, available at https://unity.com/.

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

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

[28] S. Van Duyne and J. O. Smith, "The 2-D digital waveguide mesh," *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 177–180, 1993.

[29] S. Willemsen, "Sound Files (MembraneOutputs.zip)," accessed April 26, 2020, available at https://github.com/SilvinWillemsen/Membrane/blob/Paper/Sound_Files/MembraneOutputs.zip.

[30] R. Kirk, *Statistics: an introduction*. Nelson Education, 2007.

[31] L. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297—-334, 1951.

[32] P. Kline, *The handbook of psychological testing (2nd ed.)*. London: Routledge, 2000.

Paper G

Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21), 2021.

Abstract

For physical modelling sound synthesis, many techniques are available; time-stepping methods (e.g., finite-difference time-domain (FDTD) methods) have an advantage of flexibility and generality in terms of the type of systems they can model. These methods do, however, lack the capability of easily handling smooth parameter changes while retaining optimal simulation quality and stability, something other techniques are better suited for. In this paper, we propose an efficient method to smoothly add and remove grid points from a FDTD simulation under sub-audio rate parameter variations. This allows for dynamic parameter changes in physical models of musical instruments. An instrument such as the trombone can now be modelled using FDTD methods, as well as physically impossible instruments where parameters such as e.g. material density or its geometry can be made time-varying. Results show that the method does not produce (visible) artifacts and stability analysis is ongoing.

1 Introduction

The operation of most musical instruments can be subdivided into excitation and resonator components [1]. Examples of excitation-resonator combinations are the bow and violin and the lips and trumpet. In most instruments, the parameters describing the excitation are continuously varied by the performer to play the instrument. As an example, the bow velocity, bow position and bow force for stringed instruments, and lip pressure and frequency for brass instruments. Naturally, the resonator is also altered by fingering the strings of the violin or pressing valves on the trumpet to change the instrument pitch. But, even under such variable playing conditions, physical properties of the resonators do not change: the string length and tension stay the same and the total tube length remains unchanged; it is only the portions that resonate that are shortened or lengthened.

There are several examples where the parameters of the resonator are also modified. A prime example of this is the trombone, where the tube length is dynamically changed in order to generate different pitches. The slide whistle is another example in this category. Guitar strings are another category where the tension can be smoothly modulated during performance using the fretting finger or a whammy bar to create smooth pitch glides. The same kind of tension modulation is used for the membranes of timpani or "hourglass drums" to change the pitch. It is these direct parameter modifications of the resonators that we are interested in simulating. In addition to simulating existing instruments, one could potentially simulate instruments that can be manipulated in physically impossible ways. Examples of this could be to dynamically change material properties such as density or stiffness, or even the geometry and size of the instrument where this is physically impossible.

Finite-difference time-domain (FDTD) methods are flexible and generalisable techniques which have seen increased use in physical modelling sound synthesis applications [2]. The normal approach, for a given system such as a musical instrument, is to describe its motion by a set of partial differential equations (PDEs). The instrument is then represented over a spatial grid, and a time-stepping method is developed, yielding a fully discrete approximation to the target PDE system.

In many cases, the system itself is static, so that the defining parameters do not change over time. In others, such as the trombone and others mentioned above, this is not the case, and various technical challenges arise when trying to design a simulation using FDTD methods; all relate to the choice of the spatial grid. For example, the grid density is usually closely tied to the parameters themselves through a stability condition. Also, adding and removing points from the grid is nontrivial and can cause audible artifacts and new stability concerns. The default approach of defining a grid globally, according to a very conservative stability condition, as done in [3], is possible, but introduces numerical dispersion and bandlimiting effects. Full-grid interpolation [2, Ch. 5] could be used to change between grid configurations, but extremely high sample rates are necessary to avoid audible artifacts and low-passing effects, rendering any implementation offline.

In this paper, a new method is proposed, allowing the efficient and smooth insertion and deletion of grid points from 1D finite-difference grids to allow for dynamic parameter changes. We are interested in varying parameters 'slowly' (i.e., at sub-audio rate corresponding to human gestural control). In a companion paper we present a physical model of the trombone using the method proposed in this paper [4]. Notice that other techniques do allow for dynamic parameter changes but come with their own drawbacks [2]. Examples of dynamic parameters using modal synthesis [5] are shown in [6, 7] and digital waveguides [8] are shown in [9].

This paper is structured as follows: Section 2 presents the 1D wave equation, to be used as an illustrative example for the proposed method. Section 3 gives an introduction to numerical methods, stability and simulation quality. The proposed method for dynamic grids is then presented in Section 4 and applied to the 1D wave equation. Section 5 shows the results of an analysis performed on the method, which are discussed in Section 6. Finally, concluding remarks and future perspectives are given in 7.

2 Continuous Systems

The wave equation is a useful starting point for investigations of time-varying behaviour in musical instruments. In 1D, the wave equation may be written as

$$\frac{\partial^2 q}{\partial t^2} = c^2 \frac{\partial^2 q}{\partial x^2} \,, \tag{1}$$

and is defined over spatial domain $x \in [0, L]$, for length L (in m) and time $t \geq 0$ (in s). c (in m/s) is the wave speed. The dependent variable $q = q(x,t)$ in Eq. (1) may be interpreted as the transverse displacement of an ideal string, or the acoustic pressure in the case of a cylindrical tube. Two possible choices of boundary conditions are

$$q(0,t) = q(L,t) = 0 \quad \text{(Dirichlet)}, \tag{2a}$$

$$\frac{\partial}{\partial x} q(0,t) = \frac{\partial}{\partial x} q(L,t) = 0 \quad \text{(Neumann)}, \tag{2b}$$

and describe 'fixed' or 'free' boundary respectively in the case of an ideal string, and 'open' or 'closed' conditions respectively in the case of a cylindrical acoustic tube.

2.1 Dynamic parameters

In the case of the 1D wave equation, only the wave speed c and length L can be altered (in the case of an acoustic tube, only L is variable, and for a string, c could exhibit variations through changes in tension). If the same boundary condition is used at both ends of the domain, and under static conditions, the fundamental frequency f_0 of vibration can be calculated according to

$$f_0 = \frac{c}{2L} \,. \tag{3}$$

In the dynamic case, and under slow (sub-audio rate) variations of c or L, Eq. (3) still holds approximately. From Eq. (3), one can easily conclude that in terms of fundamental frequency, halving the length in Eq. (1) is identical to doubling the wave speed and vice versa. Looking at Eq. (1) in isolation, f_0 is the only behaviour that can be changed. One can thus leave L fixed and allow time variation in c, so that $c = c(t)$, which will prove easier to work with in the following sections. This fact can more easily be seen if Eq. (1) is scaled or non-dimensionalised as in [2], where scaled domain $x' = x/L \;\Rightarrow\; x' \in [0,1]$ and $\gamma = c/L$ such that $f_0 = \gamma/2$. For clarity, however, we will employ a fully dimensional representation here.

3 Numerical methods

This section will provide a brief introduction to physical modelling using FDTD methods, including details on stability and quality of the simulations based on these methods. In this section, c is assumed constant.

3.1 Discretisation

In FDTD methods, the first step is the definition of a grid. The spatial variable can be discretised using $x_l = lh$ with integer $l \in \{0, \dots, N\}$. The grid spacing h (in m) is the distance between adjacent grid points, and the total number of points covering the domain, including endpoints, is $N + 1$. Here, integer N describes the total number of intervals between the grid points, and thus the total domain length is $L = Nh$. The temporal variable can be discretised using $t_n = nk$ with positive integer n, time step $k = 1/f_\text{s}$ (in s) for sample rate f_s (in Hz). The state variable q can then be approximated using $q_l^n \approxeq q(x = lh, t = nk)$.

The following operators can then be applied to q_l^n to get the following approximations to the derivatives in Eq. (1)

$$\delta_{tt} q_l^n = \frac{1}{k^2}\left(q_l^{n+1} - 2q_l^n + q_l^{n-1}\right) \quad \approx \quad \frac{\partial^2 q}{\partial t^2}\,, \tag{4a}$$

$$\delta_{xx} q_l^n = \frac{1}{h^2}\left(q_{l+1}^{n} - 2q_l^n + q_{l-1}^{n}\right) \quad \approx \quad \frac{\partial^2 q}{\partial x^2}\,. \tag{4b}$$

Substituting these definitions into Eq. (1) yields the following finite-difference (FD) scheme

$$\delta_{tt} q_l^n = c^2 \delta_{xx} q_l^n. \tag{5}$$

Expanding the operators as in (4) and solving for q_l^{n+1} yields the following update equation

$$q_l^{n+1} = 2q_l^n - q_l^{n-1} + \lambda^2\left(q_{l+1}^n - 2q_l^n + q_{l-1}^n\right), \tag{6}$$

which is suitable for direct software implementation. Here,

$$\lambda = \frac{ck}{h} \tag{7}$$

is referred to as the Courant number, constrained by numerical stability conditions, and also has an impact on the quality and behaviour of the simulation. This will be described in detail in Sections 3.2 and 3.3.

In the FD scheme described in Eq. (5), the boundary locations are at $l = 0$ and $l = N$. Substituting these locations into Eq. (6) seemingly introduces the need of grid points outside of the defined domain, namely q_{-1}^n and q_{N+1}^n.

These can be referred to as *virtual grid points* and can be accounted for using the boundary conditions in Eq. (2). Discretising these yields

$$q_0^n = q_N^n = 0, \quad \text{(Dirichlet)} \tag{8a}$$

$$\delta_{x\cdot} q_0^n = \delta_{x\cdot} q_N^n = 0, \quad \text{(Neumann)} \tag{8b}$$

where

$$\delta_{x\cdot} q_l^n = \frac{1}{2h}\left(q_{l+1}^n - q_{l-1}^n\right) \quad \approx \quad \frac{\partial q}{\partial x} \tag{9}$$

is a second-order accurate approximation of the first-order spatial derivative. The Dirichlet condition in (8a) says that the displacements of q at the boundary locations are always 0. In practice, this means that these grid points do not need to be updated and the spatial range of calculation for Eq. (6) then becomes $l \in \{1, \ldots, N-1\}$. If the Neumann condition is used, the boundary points do need to be updated as these are not necessarily 0; rather, their 'slope' is 0. Eq. (8b) can then be expanded to yield defnitions for these virtual grid points

$$q_{-1}^n = q_1^n \quad \text{and} \quad q_{N+1}^n = q_{N-1}^n\,. \tag{10}$$

Now that the full system is described, audio output at sample rate f_s can be drawn from the state q_l^n in Eq. (6) at $0 < l < N$ (when using Dirichlet boundary conditions).

3.2 Stability

Explicit FDTD methods for hyperbolic systems such as the 1D wave equation must necessarily satisfy a stability condition. In the case of the update in Eq. (6) it can be shown – using von Neumann analysis [10] – that the system is stable if

$$\lambda \leq 1, \tag{11}$$

which is referred to as the Courant-Friedrichs-Lewy (CFL) condition. The more closely λ approaches this condition with equality, the higher the quality of the simulation (see Section 3.3) and if $\lambda = 1$, Eq. (6) yields an exact solution to Eq. (1). If $\lambda > 1$ the system will become unstable. Recalling (7), Eq. (11) can be rewritten in terms of grid spacing h to get

$$h \geq ck. \tag{12}$$

This shows that the CFL condition in (11) puts a lower bound on the grid spacing, determined by the sample rate and wave speed. Usually, the following steps are taken to calculate λ:

$$h := ck, \;\; N := \left\lfloor \frac{L}{h} \right\rfloor, \;\; h := \frac{L}{N}, \;\; \lambda := \frac{ck}{h}, \tag{13}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. In other words, condition (12) is first satisfied with equality and used to calculate number of intervals N. Thereafter, h is recalculated based on integer N and used to calculate λ. The calculation of λ in Eq. (13) can be compactly rewritten as

$$\lambda = \frac{ck}{L} \cdot \left\lfloor \frac{L}{ck} \right\rfloor . \tag{14}$$

The flooring operation causes the CFL condition in (11) to not always be satisfied with equality and results in a reduced simulation quality described in the following section.

3.3 Simulation Quality

Choosing $\lambda < 1$ in Eq. (6) will decrease the simulation quality in two ways. Firstly, it will decrease the maximum frequency that the simulation is able to produce, i.e., it will decrease the bandwidth of the output sound of the system.

By analysing the scheme in Eq. (6), it can be shown that the maximum frequency produced by the system can be calculated using $f_{\text{max}} = f_{\text{s}} \sin^{-1}(\lambda)/\pi$ [2, Chap. 6]. Note that only a small deviation of λ from condition (11) leads to a large reduction in output bandwidth. Secondly, choosing $\lambda < 1$ causes numerical dispersion. Harmonic partials become unnaturally closely spaced at higher frequencies (i.e. spurious inharmonicity increases) as λ decreases, which is generally undesirable.

4 The Dynamic Grid

The time variation of the wave speed c leads to various complications in the simulation framework presented above. First of all, a change in c causes a change in λ according to Eq. (14), affecting the simulation quality and bandwidth. Secondly, and more importantly, a change in c could result in a change in N through Eq. (13). As N directly relates to the number of grid points, this raises questions as to *where* and especially *how* one would add and remove points to the grid according to the now-dynamic wave speed.

We propose a method that allows for a non-integer number of intervals to smoothly change between grid configurations, i.e, the number of grid points used. This removes the necessity of the flooring operation in Eqs. (13) and (14), and consequently satisfies the CFL condition in (11) with equality at all times. Introducing fractional number of intervals $\mathcal{N}$, where $N = \lfloor \mathcal{N} \rfloor$, Eq. (3) can be rewritten in terms of $\mathcal{N}$ by substituting the calculation of N from (13) into Eq. (3) (using $h = ck$) yielding

$$f_0 = \frac{1}{2\mathcal{N}k} \quad \text{with} \quad \mathcal{N} = L/h. \tag{15}$$

This shows that if $\lambda = 1$, $\mathcal{N}$ solely determines the fundamental frequency of the simulation.

Ideally, a method that dynamically changes the grid size of a FD scheme should

r1. generate an output with a fundamental frequency f_0 which is proportional to the wave speed c ($f_0 \propto c$),

r2. allow for a fractional number of intervals $\mathcal{N}$ to smoothly (without audible artifacts) transition between different grid configurations,

r3. generate an output containing $N - 1$ modes which are integer multiples of the fundamental ($f_p = f_0 p$ with integer p),

r4. work in real time to have a playable simulation.

These requirements will be used in Section 6 to evaluate the proposed method.

4.1 Proposed Method

In the following, the location of a grid point (in m from the left boundary) q_l at time index n is denoted by $x^n_{q_l}$. Furthermore, some variables are now time dependent as indicated by superscript n. These are c^n, h^n, $\mathcal{N}^n$, N^n and f_0^n.

4.1.1 System Setup

Consider two grid functions, $u^n_{l_u}$ and $w^n_{l_w}$ defined over discrete domains $l_u \in \{0, \ldots, M^n\}$ and $l_w \in \{0, \ldots, M^n_w\}$ respectively with integers $M^n = \lceil 0.5N^n \rceil$ with $\lceil \cdot \rceil$ denoting the ceiling operation and $M^n_w = \lfloor 0.5N^n \rfloor$, i.e., half the number of points allowed by the stability condition, plus one for overlap. The two grid functions are assumed to lie adjacent to each other on the same domain x. For now, the grid locations $l_u = M^n$ and $l_w = 0$ are assumed to overlap so that $x^n_{u_{M^n}} = x^n_{w_0} = M^n h^n$, and are referred to as the inner boundaries. The grid locations $l_u = 0$ and $l_w = M^n_w$ are placed at $x^n_{u_0} = 0$ and $x^n_{w_{M^n_w}} = L$ and will be referred to as the outer boundaries. See Figure 1a. The following boundary conditions are then imposed:

$$u^n_0 = w^n_{M^n_w} = 0, \quad \text{(Dirichlet)} \tag{16a}$$

$$\delta_{x\cdot} u^n_{M^n} = \delta_{x\cdot} w^n_0 = 0. \quad \text{(Neumann)} \tag{16b}$$

In other words, grid points at the outer boundaries are fixed, according to the usual Dirichlet condition, and those at the inner boundaries are free. It is important to note that the Neumann condition is just used as a starting point for the method here, but will be modified in Section 4.1.2. The systems can then be connected at the inner boundaries using a rigid connection

$$u_{M^n}^n = w_0^n, \quad \text{if } x_{u_{M^n}}^n = x_{w_0}^n. \tag{17}$$

Notice that this condition only needs to be satisfied when the inner boundaries perfectly overlap, which is not always the case when c^n is varied (see Section 4.1.2).

To sum up, a grid function with N intervals as per Eq. (13) is divided into two separate subsystems connected at their respective inner boundaries.

With the above boundary conditions imposed, the following state vectors can be defined:

$$\mathbf{u}^n = [u_1^n, \dots, u_{M^n}^n]^T, \text{ and } \mathbf{w}^n = [w_0^n, \dots, w_{M_w^n-1}^n]^T, \tag{18}$$

with T denoting the transpose operation, and have M^n and M_w^n points respectively. Note that the grid points at the outer boundaries are excluded as they are 0 at all times due to the Dirichlet boundary condition in (8a). A vector concatenating (18) is then defined as

$$\boldsymbol{\mathcal{U}}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix}. \tag{19}$$

Even though the new system has an extra (overlapping) grid point, the behaviour of the new system should be identical to that of the original system in Eq. (5) with (static) $\mathcal{N}^n = N^n$. That this holds will be shown below.

Using $u_{l_u}^n$ and $w_{l_w}^n$ in the context of the 1D wave equation, a system of FD schemes can be defined as

$$\begin{cases} \delta_{tt} u_{l_u}^n = (c^n)^2 \delta_{xx} u_{l_u}^n + J_u(x_{u_{M^n}}^n) F^n, \\ \delta_{tt} w_{l_w}^n = (c^n)^2 \delta_{xx} w_{l_w}^n - J_w(x_{w_0}^n) F^n, \end{cases} \tag{20}$$

with spreading operators

$$\begin{aligned} J_u(x_i^n) &= \begin{cases} \frac{1}{h^n}, & l_u = \lfloor x_i^n / h^n \rfloor \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \\ J_w(x_i^n) &= \begin{cases} \frac{1}{h^n}, & l_w = \lfloor x_i^n / h^n \rfloor - M^n \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{21}$$

applying the effect of the connection F^n (in m^2/s^2) to grid points $u_{M^n}^n$ and

w_0^n respectively. Expanding the spatial operators in system (20) at the inner boundaries, recalling the Neumann condition in (16b) and the definition for the virtual grid points needed for this condition in Eq. (10) yields

$$\begin{cases} \delta_{tt}u_{M^n}^n = \frac{\lambda^2}{k^2}(2u_{M^n-1}^n - 2u_{M^n}^n) + \frac{1}{h^n}F^n, \\ \delta_{tt}w_0^n = \frac{\lambda^2}{k^2}(2w_1^n - 2w_0^n) - \frac{1}{h^n}F^n. \end{cases} \tag{22}$$

It is important to note that the time index n in M^n will not be affected by the δ_{tt} operator and all obtained terms after expansion (Eq. (4a)) will use the same value for M^n. Because of the rigid connection in (17), it is also true that $\delta_{tt}u_{M^n}^n = \delta_{tt}w_0^n$ (if $x_{u_{M^n}}^n = x_{w_0}^n$), and F^n can be calculated by setting the right side of the equations in (22) equal to each other:

$$\frac{\lambda^2}{k^2}(2u_{M^n-1}^n - 2u_{M^n}^n) + \frac{1}{h^n}F^n = \frac{\lambda^2}{k^2}(2w_1^n - 2w_0^n) - \frac{1}{h^n}F^n,$$

$$F^n = h^n\frac{\lambda^2}{k^2}(w_1^n - u_{M^n-1}^n).$$

Substituting this into system (22) after expansion of the second-time derivative yields the update of the inner boundaries

$$\begin{cases} u_{M^n}^{n+1} = 2u_{M^n}^n - u_{M^n}^{n-1} + \lambda^2(u_{M^n-1}^n - 2u_{M^n}^n + w_1^n), & \text{(23a)} \\ w_0^{n+1} = 2w_0^n - w_0^{n-1} + \lambda^2(u_{M^n-1}^n - 2w_0^n + w_1^n), & \text{(23b)} \end{cases}$$

which, (again, recalling Eq. (17)) are indeed equivalent expressions for the connected point which is necessary to satisfy the rigid connection. System (20) can be shown to exhibit behaviour identical to that of the original scheme in Eq. (5) using (static) $\mathcal{N}^n = N^n$. In (23), w_1^n in Eq. (23a) acts as virtual grid point $u_{M^n+1}^n$, and $u_{M^n-1}^n$ in (23b) as virtual grid point w_{-1}^n. This important fact is what the proposed method relies on and will be extensively used in the following.

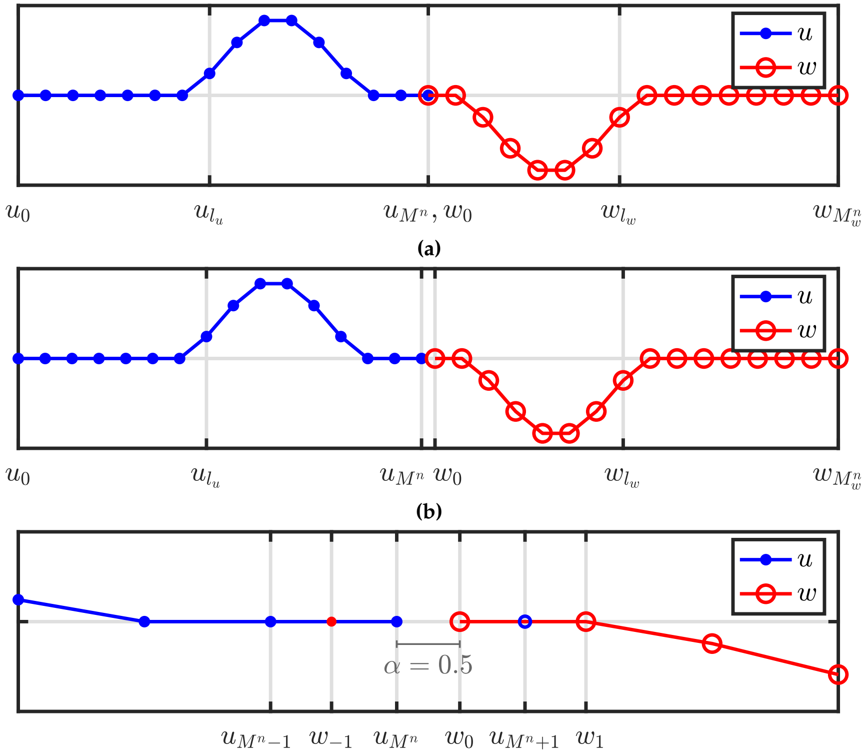
4.1.2 Changing the Grid

The previous section describes the case in which $\mathcal{N}^n$ is an integer. We now continue by varying c^n such that this is not the case.

The locations of the outer boundaries $x_{u_0}^n$ and $x_{w_{M_w}}^n$ are fixed:

$$x_{u_0}^n = x_{u_0}^0 = 0 \quad \text{and} \quad x_{w_{M_w^n}}^n = x_{w_{M_w^n}}^0 = L \quad \forall n.$$

If the wave speed c^n is then decreased, and consequently the grid spacing h^n according to Eq. (12) (with equality), all other points move towards their respective outer boundary (see Figure 1b). Calculating h^n this way allows this method to always satisfy the CFL condition in Eq. (11) with equality, solving



(a)

(b)

(c)

Fig. 1: *Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but 'x^n' is omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($\mathcal{N}^n = 30$, $x^n_{u_{M^n}} = x^n_{w_0}$). (b) When c^n – and consequently h^n – are decreased and the positions of the grid points change ($\mathcal{N}^n = 30.5$, $x^n_{u_{M^n}} \neq x^n_{w_0}$). (c) Figure 1b zoomed-in around the inner boundaries. The virtual grid points $u^n_{M^n+1}$ and w^n_{-1} are shown together with the distance between them expressed using α in Eq.* (24).

issues regarding simulation quality and numerical dispersion described in Section 3.3.

As mentioned in Section 4.1.1, the state of the virtual grid points at the inner boundaries are defined as $u^n_{M^n+1} = w^n_1$ and $w^n_{-1} = u^n_{M^n-1}$ when the inner boundaries perfectly overlap (i.e., $x^n_{u_{M^n}} = x^n_{w_0}$). If this is not the case ($x^n_{u_{M^n}} \neq x^n_{w_0}$) a Lagrangian interpolator $I(x^n_i)$ at location x^n_i (in m from the left boundary) can be used to calculate the value of these virtual grid points (also see Figure 1c for reference). The interpolator I is a row-vector with the same length as $\boldsymbol{\mathcal{U}}^n$ (from Eq. (19)) and its values depend on the interpolation order. In the following, the fractional part of $\mathcal{N}^n$ is defined as

$$\alpha = \alpha^n = \mathcal{N}^n - N^n, \tag{24}$$

and for clarity, I and $\boldsymbol{\mathcal{U}}^n$ are indexed by m. Now, consider the following quadratic interpolator

$$I_2(x_i^n) = \begin{cases} -(\alpha-1)/(\alpha+1), & m = m_i^n - 1 \\ 1, & m = m_i^n \\ (\alpha-1)/(\alpha+1), & m = m_i^n + 1 \\ 0, & \text{otherwise} \end{cases} \tag{25a}$$

and its flipped version

$$I_2^{\leftarrow}(x_i^n) = \begin{cases} (\alpha-1)/(\alpha+1), & m = (m_i^{\leftarrow})^n - 1 \\ 1, & m = (m_i^{\leftarrow})^n \\ -(\alpha-1)/(\alpha+1), & m = (m_i^{\leftarrow})^n + 1 \\ 0, & \text{otherwise} \end{cases} \tag{25b}$$

with $m_i^n = \lfloor x_i^n/h^n \rfloor$ and $(m_i^{\leftarrow})^n = \lfloor x_i^n/h^n + (1-\alpha) \rfloor$, where the shift in the latter is necessary to transform the location x_i^n to the correct indices of $\boldsymbol{\mathcal{U}}^n$. When applied to Eq. (19) this yields the definitions for the virtual grid points

$$u_{M^n+1}^n = I_2^{\leftarrow}(x_{u_{M^n+1}}^n)\boldsymbol{\mathcal{U}}^n = \frac{\alpha-1}{\alpha+1}u_{M^n}^n + w_0^n - \frac{\alpha-1}{\alpha+1}w_1^n, \tag{26a}$$

$$w_{-1}^n = I_2(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n = -\frac{\alpha-1}{\alpha+1}u_{M^n-1}^n + u_{M^n}^n + \frac{\alpha-1}{\alpha+1}w_0^n. \tag{26b}$$

These definitions for the virtual grid points at the inner boundaries will replace the Neumann condition in Eq. (16b). One can show that when $\mathcal{N}^n$ is an integer, and thus $\alpha = 0$, Eqs. (26a) and (26b) can be substituted as w_1^n and $u_{M^n-1}^n$ into Eqs. (23a) and (23b) respectively (as these acted as virtual grid points $u_{M^n+1}^n$ and w_{-1}^n). Then recalling Eq. (17) it can be seen that the system reduces to (23) and exhibits the same exact behaviour as the usual case in Eq. (5).

Now that the virtual grid points at the inner boundaries are not determined by the Neumann boundary condition in (16b), but rather by the definitions in Eqs. (26), system (20) can simply be re-written to

$$\begin{cases} \delta_{tt}u_{l_u}^n = (c^n)^2\delta_{xx}u_{l_u}^n, \\ \delta_{tt}w_{l_w}^n = (c^n)^2\delta_{xx}w_{l_w}^n, \end{cases} \tag{27}$$

where the Dirichlet condition in (16a) is (still) used for the outer boundaries and the Neumann condition at the inner boundaries in (16b) is replaced by the definitions in (26):

$$u_{M^n+1}^n = I_2^{\leftarrow}(x_{u_{M^n+1}}^n)\boldsymbol{\mathcal{U}}^n \quad \text{and} \quad w_{-1}^n = I_2(x_{w_{-1}}^n)\boldsymbol{\mathcal{U}}^n. \tag{28}$$

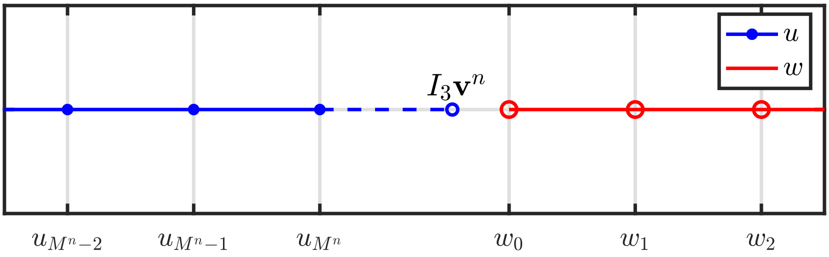


Fig. 2: *The moment when a point is added to* $\mathbf{u}$ *at location* $x^n_{u_{M^n+1}}$ *in Eq.* (29). *This figure shows an extreme case where this location is far from* $x^n_{w_0}$*, i.e.,* $\alpha \not\approx 0$ *in Eq.* (30).

4.1.3 *Adding and Removing Grid Points*

When c^n, and consequently h^n, are decreased and the inner boundary points surpass the virtual points (i.e. $x^n_{u_{M^n}} \leq x^n_{w_{-1}}$ and $x^n_{w_0} \geq x^n_{u_{M^n+1}}$), this means that $N^n > N^{n-1}$. A point is then added to the right boundary of u and the left boundary of w (for both time indices n and $n-1$) in an alternating fashion:

$$\begin{cases} \mathbf{u}^n = [(\mathbf{u}^n)^T, I_3\mathbf{v}^n]^T & \text{if } N^n \text{ is odd,} \\ \mathbf{w}^n = [I_3^{\leftarrow}\mathbf{v}^n, (\mathbf{w}^n)^T]^T & \text{if } N^n \text{ is even.} \end{cases} \tag{29}$$

Here,

$$\mathbf{v}^n = [u^n_{M^n-1}, u^n_{M^n}, w^n_0, w^n_1]^T,$$

and cubic Lagrangian interpolator

$$I_3 = \begin{bmatrix} -\frac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} & \frac{2\alpha}{\alpha+2} & \frac{2}{\alpha+2} & -\frac{2\alpha}{(\alpha+3)(\alpha+2)} \end{bmatrix}, \tag{30}$$

with $I_3^{\leftarrow}$ being a flipped, not shifted (as $I_2^{\leftarrow}$ in Eq. (25b)) version of (30). See Figure 2. Notice that $\mathcal{N}^n$ is only going to be slightly bigger than an integer at the moment that a point is added and Eq. (24) will return $\alpha \gtrsim 0$. This means that that $I_3 \approx [0, 0, 1, 0]$ and the displacement of the newly added point is nearly fully based on the grid point at the inner boundary of the other system.

Removing grid points happens when c^n, and consequently h^n, are increased and $x^n_{u_{M^n}} > x^n_{w_0}$ (or $N^n < N^{n-1}$). Grid points are simply removed from $\mathbf{u}$ and $\mathbf{w}$ (again for both n and $n-1$) in an alternating fashion according to

$$\begin{cases} \mathbf{u}^n = [u^n_0, u^n_1 ..., u^n_{M^n-1}]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w^n_1, w^n_2 ..., w^n_{M^n_w}]^T & \text{if } N^n \text{ is odd.} \end{cases} \tag{31}$$

In Eqs. (29) and (31), the even and odd conditions can be inverted. To keep the difference between u and w a maximum of one grid point, the ceiling and flooring operations when calculating M^n and M_w^n will need to be inverted as well.

Until now, only adding and removing points in the center of the original system has been considered. This location could be moved anywhere along the grid, the limit being one point from the boundary. In other words, both $u_{l_u}^n$ and $w_{l_w}^n$ need to have at least one point (excluding the grid points at the outer boundaries). Furthermore, one does not have to add and remove points from $\mathbf{u}$ and $\mathbf{w}$ in an alternating fashion as in (29), but can just add to and remove from, for example, $\mathbf{u}$ leaving $\mathbf{w}$ the same size throughout the simulation. In the extreme case where $M^n = N^n - 1$ and $M_w^n = 1$ (leaving $w_{l_w}^n$ with only one moving grid point, w_0^n) the method still works.

4.1.4 Displacement correction

A problem that arises when increasing c^n, is that it is possible that the displacements $u_{M^n}^n \not\approx w_0^n$ at the time when a grid point needs to be removed. As the grid locations $x_{u_{M^n}}^n \approx x_{w_0}^n$ at the time of removal, this violates the rigid connection in (17) and causes audible artifacts. A method is proposed that decreases the relative displacement of the inner boundaries the closer their grid-locations are together, i.e., the closer α in (24) is to 0. We thus extend system (27) with an artificial spring force as

$$\begin{cases} \delta_{tt} u_{l_u}^n = (c^n)^2 \delta_{xx} u_{l_u}^n + J_u(x_{u_{M^n}}^n) F_\text{c}^n, \\ \delta_{tt} w_{l_w}^n = (c^n)^2 \delta_{xx} w_{l_w}^n - J_w(x_{w_0}^n) F_\text{c}^n. \end{cases} \tag{32}$$

Using centred temporal averaging and difference operators

$$\mu_{t\cdot} q_l^n = \frac{1}{2}\left(q_l^{n+1} + q_l^{n-1}\right), \tag{33a}$$

$$\delta_{t\cdot} q_l^n = \frac{1}{2k}\left(q_l^{n+1} - q_l^{n-1}\right), \tag{33b}$$

the correction effect is defined as

$$F_\text{c}^n = \beta\left(\mu_{t\cdot}\eta^n + \sigma_0 \delta_{t\cdot}\eta^n\right), \tag{34}$$

with the difference in displacement between the inner boundaries

$$\eta^n \triangleq w_0^n - u_{M^n}^n, \tag{35}$$

and damping coefficient σ_0. Furthermore, β scales the effect of the displacement correction and is defined as

$$\beta = \beta(\alpha) = \frac{1-\alpha}{\alpha + \varepsilon}, \tag{36}$$

where $\varepsilon \ll 1$ prevents a division by 0. Despite the operators in (33) introducing states at $n+1$, it is possible to calculate the force explicitly (such as in [2] or [11]). Furthermore, it can be shown that even when $\varepsilon = 0$ this calculation is always defined. In that case, as $\alpha \to 0$, $\beta \to \infty$ which acts as a rigid connection such as Eq. (17). Essentially, the displacement correction attempts to have $\eta^n \to 0$ in Eq. (35) as $\alpha \to 0$ to satisfy the rigid connection in Eq. (17). Although the correction presented here is not based on some physical process, it can be justified by the fact that large differences in displacement between two spatially adjacent points is not physical.

Notice that when c^n is decreased, the rigid connection will not be violated as $u^n_{M^n} \approx w^n_0$ when a point is added. This is due to the fact that $I_3 \approx [0,0,1,0]$ and either $u^n_{M^n}$ or w^n_0 is the newly added point which almost solely based on the other.

4.2 Summary

Here, Section 4.1 is summarised and describes the final version of the proposed method.

The proposed method subdivides a grid function q^n_l with N intervals into two grid functions $u^n_{l_u}$ and $w^n_{l_w}$ with M^n and M^n_w intervals respectively for a total of $N^n + 2$ grid points. Knowing that $\lambda = 1\ \forall n$, Eq. (6), written for both grid functions, becomes

$$u^{n+1}_{l_u} = u^n_{l_u+1} + u^n_{l_u-1} - u^{n-1}_{l_u}, \tag{37a}$$

$$w^{n+1}_{l_w} = w^n_{l_w+1} + w^n_{l_w-1} - w^{n-1}_{l_w}. \tag{37b}$$

Due to the Dirichlet boundary condition in (16a) imposed at the outer boundaries of the system, u^n_0 and $w^n_{M_w}$ are 0 at all times and do not have to be included in the calculation. The ranges of calculation for Eq. (37a) and (37b) then become $l_u \in \{1, \ldots, M^n\}$ and $l_w \in \{0, \ldots, M^n_w - 1\}$ respectively.

The grid points at the inner boundaries are calculated by expanding (27) (ignoring the displacement correction for now)

$$u^{n+1}_{M^n} = u^n_{M^n+1} + u^n_{M^n-1} - u^{n-1}_{M^n}, \tag{38a}$$

$$w^{n+1}_0 = w^n_{-1} + w^n_1 - w^{n-1}_0, \tag{38b}$$

where virtual grid points $u^n_{M^n+1}$ and w^n_{-1} can be calculated using Eq. (26).

Then, when $N^n > N^{n-1}$, a point is added to $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$ (or $\mathbf{w}^n$ and $\mathbf{w}^{n-1}$) using Eq. (29), and when $N^n < N^{n-1}$, a point is removed from the same vectors using Eq. (31). In order to prevent audible artifacts when increasing c^n (and thus decreasing $\mathcal{N}^n$) due to a violation of the rigid connection in (17), a method is proposed in Eq. (32) to ensure that the grid points at the inner boundaries have a similar displacement when one of them is removed.

4.3 Implementation

A MATLAB implementation of the proposed method and audio examples can be found online[1] and Algorithm 1 shows the order of calculation of this implementation. Especially important to take into account, is to only retrieve a change in c^n at time index n once before all other calculations. This is to ensure that $u_{l_u}^n$ and $w_{l_w}^n$ are calculated with the same α and β for all l_u and l_w.

```
while application is running do
    Retrieve new $c^n$
    Calc. $h^n$ (Eq. (12) with equality)
    Calc. $\mathcal{N}^n$ and $N^n$ (Eqs. (15) and (13))
    Calc. $\alpha$ (Eq. (24))
    if $N^n \neq N^{n-1}$ then
        Add or remove point (Eq. (29) or (31))
        Update $M^n$ and $M_w^n$
    end
    Calc. virtual grid points (Eqs. (26))
    Calc. $u_{l_u}^{n+1}$ and $w_{l_w}^{n+1}$ (Eqs. (37) and (38))
    Calc. and apply displacement corr. (Eq. (34))
    Retrieve output
    Update states ($\boldsymbol{\mathcal{U}}^{n-1} = \boldsymbol{\mathcal{U}}^n, \boldsymbol{\mathcal{U}}^n = \boldsymbol{\mathcal{U}}^{n+1}$)
    Update $N^{n-1}$ ($N^{n-1} = N^n$)
    Increment $n$
end
```

Algorithm 1: *Pseudocode showing the order of calculations.*

5 Analysis and Results

5.1 Modes

Writing system (32) in matrix form, one can perform a modal analysis while changing c^n to obtain the frequencies and damping coefficients for each mode over time. As a test case, the wave speed of a system running at $f_\text{s} = 44100$ Hz is linearly varied from $c^0 = 2940$ ($\mathcal{N}^0 = 15$) to $c^{n_\text{end}} = 2205$ ($\mathcal{N}^{n_\text{end}} = 20$) where $n_\text{end} = t_\text{dur} f_\text{s}$ is the simulation length in samples and $t_\text{dur} = 10$ s. Grid points are added and removed as close to the right boundary as possible, i.e., $M^n = N^n - 1$ and $M_w^n = 1$ (similar behaviour can be observed if $M^n = 1$ and $M_w^n = N^n - 1$). The result of the analysis is shown in Figure 3a where

[1] https://github.com/SilvinWillemsen/DAFx21DynamicGridFiles/

higher damping (induced by the displacement correction) is indicated using thinner and bluer lines. Figure 3b shows the resulting spectrogram, with the displacement correction deactivated, of the system excited with $u_1^0 = 1$ and the output retrieved at u_1^n, and Figure 3c shows a system with the same excitation but the change in c^n inverted ($\mathcal{N}^n = 20 \to 15$) and displacement correction activated. In the following, the lowest mode generated by the analysis is referred to as f_1^n and should ideally be equal to f_0^n calculated using Eq. (3). The first thing one can observe from Figure 3a is that the frequencies of the modes decrease as c^n decreases (as desired). The lower the mode, the more linear this decrease happens. Between $\mathcal{N}^n = 15$ and $\mathcal{N}^n = 16$, f_1^n maximally deviates by -0.15 cents. In this same interval f_{15}^n maximally deviates by -67 cents. This deviation gets less as $\mathcal{N}^n$ increases. Experiments with higher even-ordered Lagrange interpolators show that these frequency deviations become smaller, but not by a substantial amount. The quadratic interpolator has thus been chosen for being simpler and more flexible while not being substantially worse than higher order interpolators.

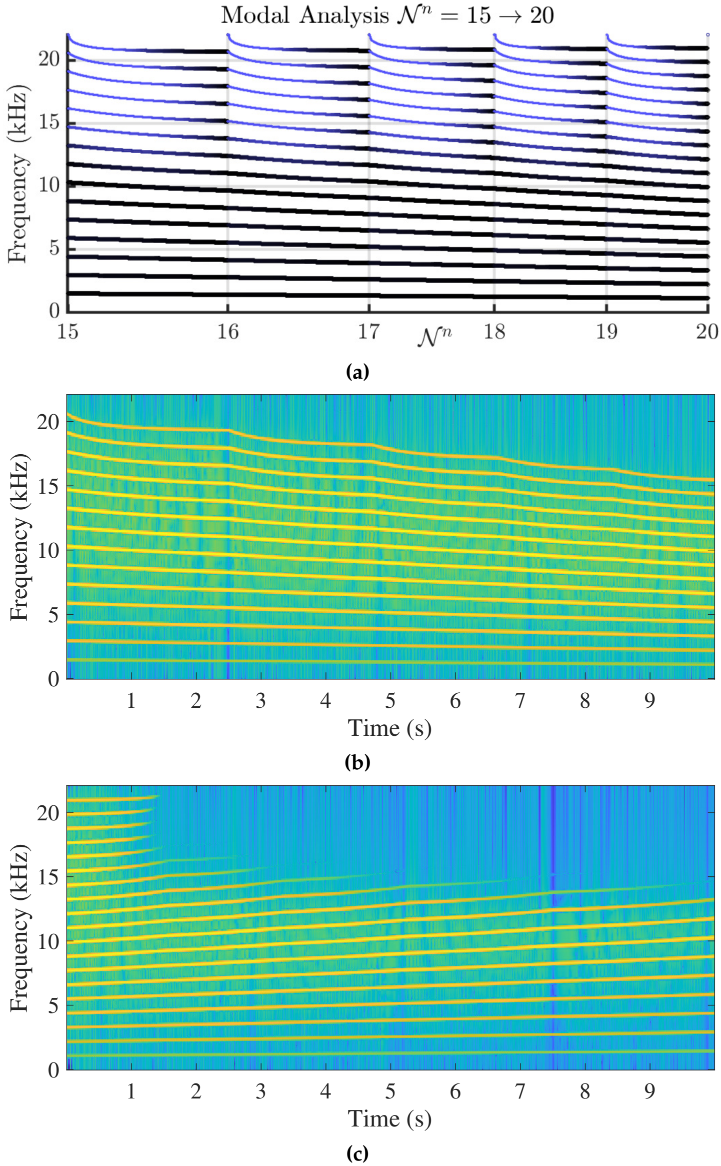
Another observation from Figure 3a is that there are always N^n modes present, corresponding to the number of moving points of the system. As can be seen in Figure 3b the highest mode is not excited. If the system is excited when $\mathcal{N}^n$ is not an integer, the highest mode will also be excited. Comparing the implementation of the system using this method with integer $\mathcal{N}^n$ (without changing c^n) to a normal implementation of the 1D wave equation (shown in Section 3) with (static) $N^n = \mathcal{N}^n$, identical outputs are observed, even though the latter has $N^n - 1$ moving points.

5.2 Displacement Correction

In the experiments, $\sigma_0 = 1$ in Eq. (34). The displacement correction has a low-pass-comb-filtering effect on the system, where the position and amount of damped regions directly relates to the position of where grid points are added and removed. The best behaviour, i.e., least affecting lower frequencies, is when grid points are added and removed as close to the boundary as possible, i.e., $M^n = N^n - 1$, and only has one damped region as shown in Figures 3a and 3c.

5.3 Limit on Rate of Change of c

The current implementation of the proposed method can only add or remove a maximum one point per sample using Eqs. (29) and (31). The rate of change of f_0^n according to (15) is thus limited by $|\mathcal{N}^n - \mathcal{N}^{n-1}| \leq 1$. Though this is the maximum limitation on speed, a much lower limitation needs to be placed to keep the system well-behaved. The usual stability and energy analyses performed on FD schemes are not valid anymore in the time-varying case.



(a)

(b)

(c)

Fig. 3: *Experiments showing (linearly) varying wave speed between* $c^0 = 2940$ *(*$\mathcal{N}^0 = 15$*) and* $c^{n_\text{end}} = 2205$ *(*$\mathcal{N}^{n_\text{end}} = 20$*) with* $M^n = N^n - 1$ *and* $M_w^n = 1$ *running at* $f_\text{s} = 44100$ *Hz for 10 s (*$n_\text{end} = 10f_\text{s}$*). (a) Modal analysis of system* (32)*. Thinner and bluer lines indicate a higher amount of damping. (b) Output of the system while decreasing* c^n *(*$\mathcal{N}^n = 15 \to 20$*) without displacement correction, excited using* $u_1^0 = 1$ *and retrieved at* u_1^n*. The sound output follows the same pattern as predicted by the analysis shown in Figure 3a. (c) Output of the system while increasing* c^n *(*$\mathcal{N}^n = 20 \to 15$*) with displacement correction activated (essentially flipping the analysis in Figure 3a along the x-axis and applying this to a 10 s simulation).*

Frozen coefficient analysis as in [10] could be applied here and hold for slowly varying coefficients, but is left for future work.

6 Discussion

To decide whether the proposed method is satisfactory, the results presented in the previous section are compared to the method requirements listed in Section 4.

It can be argued that the frequency deviations of f_1^n from f_0^n are sufficiently small to say that r1 is satisfied. As for r2, a fractional number of intervals $\mathcal{N}^n$ has been introduced and smooth transitions are indeed observed from Figure 3b, in the case when c^n is decreased and $\mathcal{N}^n$ is increased. When c^n is increased instead, the displacement correction prevents (visible) artifacts when grid points are removed as seen in Figure 3c. Despite this, the filtering effect that the displacement correction has on the system (mentioned in Section 5.2) is not ideal as it creates damped regions in the spectrum of the output sound. The least intrusive filtering happens when points are added and removed as close to the boundary as possible, i.e., when $M^n = 1$ or $M_w^n = 1$ where the damping only occurs in the higher end of the spectrum. Although artifacts do not show in Figure 3c, to confirm the absence of audible artifacts, formal listening tests have to be carried out. Furthermore, higher speeds of parameter variation might cause artifacts anyway. The value of σ_0 could therefore also be made dynamic and depending on the rate of change of c^n to have a higher effect when c^n is increased faster and vice versa. Either way, as this is still not ideal, another method for reducing artifacts that less affects the frequency content of the system should be devised, if possible. Furthermore, higher modes will be lost after decreasing $\mathcal{N}$ and will not return after increasing $\mathcal{N}$ again. They can, however, be activated again by re-exciting the system.

The modal analysis in Figure 3a shows that the method generates N^n rather than $N^n - 1$ modes as set by r3. However, the output does contain the correct number of modes as shown in Figure 3b due to the highest mode not being excited. This is a result of the rigid connection imposed on the inner boundaries, forcing them to have the same displacement and act as one point. The latter part of r3, however, is not satisfied. The modes deviate from integer multiples of f_0^n, moreso for higher modes. Other interpolation techniques could be investigated to improve the behaviour and decrease this deviation.

Finally, the method only adds a few extra calculations for the inner boundaries so r4 is also easily satisfied.

Although the results bring forward some drawbacks of the proposed method, such as modal frequency deviations, and filtering effects, most of these affect the higher frequencies of the output. First of all, human frequency sensitivity becomes very limited above 3000 Hz [12] making high-frequency deviations much less important perceptually. Secondly, the physical systems one usually tries to model contain high-frequency losses, causing higher modes to usually not have very high amplitudes to begin with. Finally, $\mathcal{N}^n$ is usually much bigger in the systems one models, where frequency deviations happen to a much smaller degree.

As of now, some aspects of the proposed method still lack physical justification (such as the displacement correction), but are shown to yield the desired behaviour and fulfil the requirements to a satisfactory degree. Despite this, further work needs to be done to physically justify the choices made in this paper.

7 Conclusions and Perspectives

This paper presents a method to change grid configurations of finite-difference schemes to allow for dynamic parameter changes. The method allows the stability condition that these schemes rely on can be satisfied with equality at all times, minimising numerical dispersion and bandlimiting issues. Grid points are shown to be added and removed smoothly and do not cause artifacts when switching between grid configurations. Listening tests will need to be performed to carried out to confirm the lack of audible artifacts.

The proposed method might not provide an exact solution to the problem of time-varying systems, and not all choices are physically justified, but it does circumvent the need for upsampling and higher orders of computations necessary to approximate this solution with, for example, full-grid interpolation.

Although this method has only been applied to the 1D wave equation it could be applied to many other 1D systems. Other parameters, such as material density or stiffness could also be made dynamic, going beyond what is physically possible. An application of the method that could be investigated is that of non-linear systems, such as the Kirchhoff-Carrier string model [13] where the tension is modulated based on the state of the system.

Other future work includes creating an adaptive version of the displacement correction that changes its effect depending on the speed at which the grid is changed. Finally, stability and energy analyses will have to be performed to show the limits on changes in parameters and grid configurations.

8 Acknowledgments

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

[1] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitator-resonator interaction in physical models syntheses," in *Proceedings of the International Computer Music Conference*, 1989.

[2] S. Bilbao, *Numerical Sound Synthesis*. United Kingdom: John Wiley & Sons, 2009.

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

[4] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "A physical model of the trombone using dynamic grids for finite-difference schemes," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx)*, 2021.

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

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

[7] S. Willemsen, S. Serafin, and J. R. Jensen, "Virtual analog simulation and extensions of plate reverberation," in *Proc. of the 14th Sound and Music Computing Conference*, 2017, pp. 314–319.

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

[9] R. Michon and J. Smith, "A hybrid guitar physical model controller: The BladeAxe," in *Proceedings ICMC | SMC | 2014*, 2014.

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

[11] S. Bilbao, "A modular percussion synthesis environment," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx)*, 2009.

[12] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*. Berlin-Heidelberg, Germany: Springer-Verlag, 1990.

[13] G. F. Carrier, "On the non-linear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, pp. 157–165, 1945.

Paper H

A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21), 2021.

Abstract

In this paper, a complete simulation of a trombone using finite-difference time-domain (FDTD) methods is proposed. In particular, we propose the use of a novel method to dynamically vary the number of grid points associated to the FDTD method, to simulate the fact that the physical dimension of the trombone's resonator dynamically varies over time. We describe the different elements of the model and present the results of a real-time simulation.

1 Introduction

The trombone is a musical instrument that presents distinct challenges from the perspective of physical modelling synthesis. In particular, the excitation mechanism between the lips and the player has been extensively studied, and simulated mostly using a simple mass-spring damper system [1]. Because the majority of the bore is cylindrical, nonlinear effects can appear at high blowing pressures [2], leading to changes in timbre, or brassiness; such effects have been investigated and simulated [1, 3, 4]. However, the defining characteristic of the trombone is that the physical dimensions of the resonator vary during playing. Synthesis techniques such as digital waveguides allow an approach to dynamic resonator changes in a simple and computationally efficient way, simply by varying the length of the corresponding delay line. This feature has been used in real-time sound synthesis [5], for simplified bore profiles suitable for modelling in terms of travelling waves.

However, when attempting more fine-grained modelling of the trombone resonator using finite-difference time-domain (FDTD) methods, the issue of the change in the tube length is not trivial. Previous implementations of brass instruments using these methods focus on the trumpet [6] and various brass instruments (including the trombone bore) under static conditions [7]. To our knowledge, the simulation of a trombone varying the shape of the resonator in real time using FDTD methods has not been approached. We can tackle this problem by having a grid that dynamically changes while the simulation is running as presented in a companion paper [8]. Briefly described, we modify the grid configurations of the FDTD method by adding and removing grid points based on parameters describing the system.

In this paper, we propose a full simulation of a trombone, describing all its elements in detail with a specific focus on the dynamic grid simulation. Section 2 presents the models for the tube and lip reed interaction in continuous time. Section 3 briefly introduces FDTD methods and the discretisation of the aforementioned continuous equations. Section 4 presents the dynamic grid used to simulate the trombone slide and details on the implementation are provided in Section 5. Section 6 presents simulation results, and some concluding remarks appear in Section 7.

2 Continuous System

Wave propagation in an acoustic tube can be approximated using a 1-dimensional (1D) model, for wavelengths that are long relative to the largest lateral dimension of the tube. Consider a tube of time-varying length $L = L(t)$ (in m) defined over spatial domain $x \in [0, L]$ and time $t \geq 0$. Using operators ∂_t and ∂_x denoting partial derivatives with respect to time t and spatial coordinate x, respectively, a system of first-order partial differential equations (PDEs) describing the wave propagation in an acoustic tube can then be written as:

$$\frac{S}{\rho_0 c^2}\partial_t p = -\partial_x(Sv), \tag{1a}$$

$$\rho_0 \partial_t v = -\partial_x p, \tag{1b}$$

with acoustic pressure $p = p(x,t)$ (in N/m^2), particle velocity $v = v(x,t)$ (in m/s) and (circular) cross-sectional area $S(x)$ (in m^2). Furthermore, ρ_0 is the density of air (in kg/m^3) and c is the speed of sound in air (in m/s). System (1) can be condensed into a second-order equation in p alone, often referred to as Webster's equation [9]. For simplicity, effects of viscothermal losses have been neglected in (1). For a full time domain model of such effects in an acoustic tube, see, e.g. [10].

System (1) requires two boundary conditions, one at either end of the domain. The left boundary condition, at $x = 0$, will be set according to an excitation model to be described in Section 2.1. The right boundary, at $x = L$, is set according to a radiation condition. The radiation model used here, is the one for the unflanged cylindrical pipe proposed by Levine and Schwinger in [11] and discretised by Silva *et al.* in [12]. As this model is not important for the contribution of this work it will not be detailed here in full. The interested reader is instead referred to [7, 13] for a comprehensive explanation.

2.1 Coupling to a Lip Reed

To excite the system, a lip reed can be modelled as a mass-spring-damper system including two nonlinearities due to flow, and the collision of the lip against the mouthpiece. In the following, y can be seen as the moving upper lip where the lower lip is left static and rigid. A diagram of the full lip-reed model is shown in Figure 1. Using dots to indicate time-derivatives, the lip reed is modelled as

$$M_\mathrm{r}\ddot{y} = -M_\mathrm{r}\omega_\mathrm{r}^2 y - M_\mathrm{r}\sigma_\mathrm{r}\dot{y} + \psi(\dot{\psi}/\dot{\eta}) + S_\mathrm{r}\Delta p, \tag{2}$$

with displacement from the equilibrium $y = y(t)$, lip mass M_r (in kg), externally supplied (angular) frequency of oscillation $\omega_\mathrm{r} = \omega_\mathrm{r}(t) = \sqrt{K_\mathrm{r}/M_\mathrm{r}}$ (in rad/s) and stiffness $K_\mathrm{r} = K_\mathrm{r}(t)$ (in N/m).

We extend the existing models of lip reeds [1] by introducing a nonlinear collision between the lips based on potential quadratisation proposed by [14]. The collision potential is defined as

$$\psi(\eta) = \left(\frac{2K_\mathrm{c}}{\alpha_\mathrm{c}+1}[\eta]_+^{\alpha_\mathrm{c}+1}\right)^{1/2}, \tag{3}$$

with collision stiffness $K_\mathrm{c} > 0$ and dimensionless nonlinear collision coefficient $\alpha_\mathrm{c} \geq 1$, The inverted distance between the lips $\eta = \eta(t) \triangleq -y - H_0$ (in m), for static equilibrium separation H_0 (in m). $[\eta]_+ = 0.5(\eta + |\eta|)$ indicates the "positive part of η". Notice, that if $\eta \geq 0$, the lips are closed and the collision potential will be non-zero. This quadratic form of a collision potential allows for a non-iterative implementation [14]. This will be explained further in Section 3.

Finally, S_r (in m^2) is the effective surface area and

$$\Delta p = P_\mathrm{m} - p(0,t) \tag{4}$$

is the difference between the externally supplied pressure in the mouth $P_\mathrm{m} = P_\mathrm{m}(t)$ and the pressure in the mouthpiece $p(0,t)$ (all in Pa). This pressure difference causes a volume flow velocity following the Bernoulli equation

$$U_\mathrm{B} = w_\mathrm{r}[-\eta]_+\mathrm{sgn}(\Delta p)\sqrt{\frac{2|\Delta p|}{\rho_0}}, \tag{5}$$

(in m^3/s) with effective lip-reed width w_r (m). Another volume flow is generated by the lip reed itself according to

$$U_\mathrm{r} = S_\mathrm{r}\frac{dy}{dt} \tag{6}$$

(in m^3/s). Assuming that the volume flow velocity is conserved, the total air volume entering the system is defined as

$$S(0)v(0,t) = U_\mathrm{B}(t) + U_\mathrm{r}(t). \tag{7}$$

This condition serves as a boundary condition at $x = 0$ for system (1).

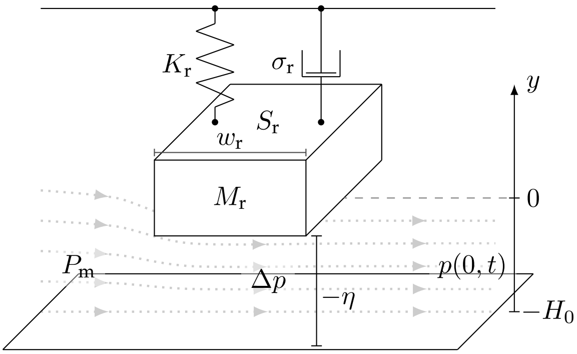


Fig. 1: *Diagram of the lip-reed system with the equilibrium at 0 and the distance from the lower lip* H_0*. The various symbols relate to those used in Eq.* (2)*.*

3 Discretisation

The continuous system described in the previous section is discretised using FDTD methods, through an approximation over a grid in space and time. Before presenting this discretisation, we briefly summarize the operation of FDTD methods.

3.1 Numerical Methods

Consider a 1D system of (static) length L described by state variable $u = u(x,t)$ with spatial domain $x \in [0, L]$ and time $t \geq 0$. The spatial domain can be disctretised according to $x = lh$ with spatial index $l \in \{0, \ldots, N\}$, number of intervals between the grid points N, grid spacing h (in m) and time as $t = nk$ with temporal index $n \in \mathbb{Z}^{0+}$ and time step k (in s). The grid function u_l^n represents an approximation to $u(x,t)$ at $x = lh$ and $t = nk$.

Shift operators can then be applied to grid function u_l^n. Temporal and spatial shift operators are

$$\begin{aligned} e_{t+}u_l^n &= u_l^{n+1}, \quad e_{t-}u_l^n = u_l^{n-1}, \\ e_{x+}u_l^n &= u_{l+1}^n\,, \quad e_{x-}u_l^n = u_{l-1}^n, \end{aligned} \tag{8}$$

from which more complex operators can be derived. First-order derivatives can be approximated using forward, backward and centred difference operators in time

$$\delta_{t+} = \frac{e_{t+} - 1}{k},\ \delta_{t-} = \frac{1 - e_{t-}}{k},\ \delta_{t\cdot} = \frac{e_{t+} - e_{t-}}{2k}, \tag{9}$$

(all approximating ∂_t) and space

$$\delta_{x+} = \frac{e_{x+} - 1}{h}, \; \delta_{x-} = \frac{1 - e_{x-}}{h}, \; \delta_{x\cdot} = \frac{e_{x+} - e_{x-}}{2h}, \tag{10}$$

(all approximating ∂_x) where 1 is the identity operator.

Furthermore, forward, backward and centred averaging operators can be defined in time

$$\mu_{t+} = \frac{e_{t+} + 1}{2}, \; \mu_{t-} = \frac{1 + e_{t-}}{2}, \; \mu_{t\cdot} = \frac{e_{t+} + e_{t-}}{2}, \tag{11}$$

and space

$$\mu_{x+} = \frac{e_{x+} + 1}{2}, \; \mu_{x-} = \frac{1 + e_{x-}}{2}, \; \mu_{x\cdot} = \frac{e_{x+} + e_{x-}}{2}. \tag{12}$$

Finally, an approximation δ_{tt} to a second time derivative may be defined as

$$\delta_{tt} = \delta_{t+}\delta_{t-} = \frac{1}{k^2}\left(e_{t+} - 2 + e_{t-}\right). \tag{13}$$

3.2 Discrete Tube

As a first step, the domain $x \in [0, L]$ can be subdivided into N equal segments of length h (the grid spacing). Interleaved grid functions approximating p and v may then be defined. Grid function p_l^n with $l \in \{0, \ldots, N\}$ approximates $p(x,t)$ at coordinates $x = lh$, $t = nk$ and $v_{l+1/2}^{n+1/2}$ with $l \in \{0 \ldots, N-1\}$ approximates $v(x,t)$ at coordinates $x = (l+1/2)h$, $t = (n+1/2)k$. In addition, a discrete cross-sectional area $S_l \approx S(x = lh)$ with $l \in \{0, \ldots, N\}$ is assumed known. System (1) can then be discretised as

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p_l^n = -\delta_{x-}(S_{l+1/2}v_{l+1/2}^{n+1/2}), \tag{14a}$$

$$\rho_0\delta_{t-}v_{l+1/2}^{n+1/2} = -\delta_{x+}p_l^n, \tag{14b}$$

where $S_{l+1/2} = \mu_{x+}S_l$ and $\bar{S}_l = \mu_{x-}S_{l+1/2}$ are approximations to the continuous cross-sectional area $S(x)$. The values for $\bar{S}_l$ at the boundaries, i.e., $\bar{S}_0$ and $\bar{S}_N$, are set equal to $S(0)$ and $S(L)$.

Expanding the operators, we obtain the following recursion

$$p_l^{n+1} = p_l^n - \frac{\rho_0 c\lambda}{\bar{S}_l}(S_{l+1/2}v_{l+1/2}^{n+1/2} - S_{l-1/2}v_{l-1/2}^{n+1/2}), \tag{15a}$$

$$v_{l+1/2}^{n+1/2} = v_{l+1/2}^{n-1/2} - \frac{\lambda}{\rho_0 c}(p_{l+1}^n - p_l^n), \tag{15b}$$

where $\lambda = ck/h$ is referred to as the Courant number and

$$\lambda \leq 1 \quad \implies \quad h \geq ck \tag{16}$$

in order for the scheme to be stable [15]. In implementation, the following steps are taken to calculate λ:

$$h := ck, \;\; N := \left\lfloor \frac{L}{h} \right\rfloor, \;\; h := \frac{L}{N}, \;\; \lambda := \frac{ck}{h}, \tag{17}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation and is necessary because N is an integer. This causes (16) to not be satisfied with equality for all choices of L.

Equations (15a) and (15b) hold for $l \in \{0, \ldots, N\}$ and $l \in \{0, \ldots, N-1\}$ respectively, and thus, in analogy with the continuous case, two numerical boundary conditions are required in order to update p_0^{n+1} and p_N^{n+1}. These are provided by numerical equivalents of the excitation condition (see Section 3.3 below) and the radiation condition (in [13]).

3.3 Lip reed

As the lip reed interacts with the particle velocity of the tube via Eq. (7), it is discretised to the interleaved temporal grid, but to the regular spatial grid as it interacts with the boundary at $x = 0$. Equations (2) - (7) are then discretised as follows:

$$\begin{aligned} M_\text{r}\delta_{tt}y^{n+1/2} =& -M_\text{r}(\omega_\text{r}^{n+1/2})^2\mu_{t\cdot}y^{n+1/2} \\ & - M_\text{r}\sigma_\text{r}\delta_{t\cdot}y^{n+1/2} + (\mu_{t+}\psi^n)\, g^{n+1/2} + S_\text{r}\Delta p^{n+1/2}, \end{aligned} \tag{18a}$$

$$\Delta p^{n+1/2} = P_\text{m}^{n+1/2} - \mu_{t+}p_0^n, \tag{18b}$$

$$U_\text{B}^{n+1/2} = w_\text{r}[-\eta^{n+1/2}]_+ \text{sgn}(\Delta p^{n+1/2}) \cdot \sqrt{2|\Delta p^{n+1/2}|/\rho_0}, \tag{18c}$$

$$U_\text{r}^{n+1/2} = S_\text{r}\delta_{t\cdot}y^{n+1/2}, \tag{18d}$$

$$\mu_{x-}(S_{1/2}v_{1/2}^{n+1/2}) = U_\text{B}^{n+1/2} + U_\text{r}^{n+1/2}. \tag{18e}$$

Here, following [14],

$$g^{n+1/2} = \begin{cases} \kappa\sqrt{\dfrac{K_\text{c}(\alpha_\text{c}+1)}{2}} \cdot (\eta^{n+1/2})^{\frac{\alpha_\text{c}-1}{2}} & \text{if } \eta^{n+1/2} \geq 0 \quad (19a) \\ -2\dfrac{\psi^n}{\eta^\star - \eta^{n-1/2}} & \text{if } \eta^{n+1/2} < 0 \quad (19b) \\ 0, & \begin{array}{r}\text{if } \eta^{n+1/2} < 0 \\ \text{and } \eta^\star = \eta^{n-1/2}\end{array} \quad (19c) \end{cases}$$

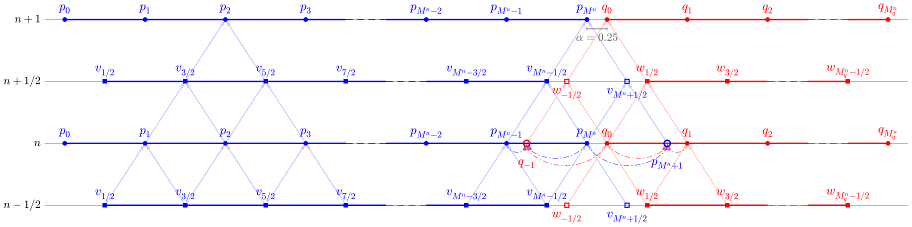


Fig. 2: *Schematic showing data flow of how different grid points at time index $n+1$ are calculated with $\alpha = 0.25$ in Eq.* (25). *To prevent cluttering, arrows going straight up (indicating that the state of a grid point at time step n is needed to calculate the state of that grid point at $n+1$) are suppressed. As an example of the usual case (refer to Eq.* (15)*), the points required to calculate p_2^{n+1} are shown. Furthermore, the points needed to calculate $p_{M^n}^{n+1}$ and q_0^{n+1} are shown. The most important difference with the usual case is that the virtual grid points $p_{M^n+1}^n$ and q_{-1}^n are the result of the interpolation of known pressure values at n using Eq.* (27).

where $\kappa = 1$ if $\psi^n \geq 0$, otherwise $\kappa = -1$. It should be noted that condition (19c) has been added to the definition of g from [14] to prevent a division by 0 in (19b). Finally, $\eta^\star = -y^\star - H_0$ where $y^\star$ is the value of $y^{n+3/2}$ calculated using system (18) (after expansion) without the collision potential. This means that system (18) needs to be calculated twice every iteration, once without the collision term and once with. The process of calculating the pressure difference $\Delta p^{n+1/2}$ in (18) will not be given here, but the interested reader is referred to [13, Ch. 5] for a derivation.

4 Dynamic grid

The defining feature of the trombone is its slide that alters the length of the tube, changing the resonant frequencies. In a companion paper [8], we present a method to dynamically change grid configurations of FD schemes by inserting and deleting grid points based on an instantaneous value of the time-varying wave speed $c(t)$. Although here, the tube length $L(t)$ is varied, the method still applies. Note that this method only works for slow (sub-audio rate) parameter changes.

We can split a tube with time-varying length L^n into two smaller sections with lengths L_p^n and L_q^n (in m) such that $L^n = L_p^n + L_q^n$. Splitting the schemes in (14) in this way yields two sets of first-order systems. The pressure and particle velocity of the first (left) system $p_{l_p}^n$ and $v_{l_p+1/2}^{n+1/2}$ are both defined over discrete domain $l_p \in \{0, \dots, M^n\}$, and those of the second (right) system $q_{l_q}^n$ and $w_{l_q-1/2}^{n+1/2}$ are defined over discrete domain $l_q \in \{0, \dots, M_q^n\}$, with

$$M^n = \lceil L_p^n / h \rceil, \quad \text{and} \quad M_q^n = \lfloor L_q^n / h \rfloor \tag{20}$$

where $\lceil\cdot\rceil$ denotes the ceiling operation. Note, that the domains for v and w have an extra grid point when compared to the regular case in (14) and that w is indexed with $l_q - 1/2$ rather than $l_q + 1/2$. The resulting system of FD schemes then becomes

$$\begin{aligned}\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p^n_{l_p} &= -\delta_{x-}(S_{l+1/2}v^{n+1/2}_{l_p+1/2}), && \text{(21a)}\\ \rho_0\delta_{t-}v^{n+1/2}_{l_p+1/2} &= -\delta_{x+}p^n_{l_p}, && \text{(21b)}\\ \frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}q^n_{l_q} &= -\delta_{x+}(S_{l-1/2}w^{n+1/2}_{l_q-1/2}), && \text{(21c)}\\ \rho_0\delta_{t-}w^{n+1/2}_{l_q-1/2} &= -\delta_{x-}q^n_{l_q}. && \text{(21d)}\end{aligned}$$

Here, due to the different indexing for w, the spatial derivatives for the right system are flipped (δ_{x+} became δ_{x-} and vice versa). Also note, that l is still used for the spatial indices of $\bar{S}$ and S which now approximate $S(x)$ according to

$$S_l \approx \begin{cases} S(x = lh) & \text{for } x \in [0, L^n_p],\\ S(x = L^n - (M^n_q - l)h) & \text{for } x \in [L^n_p, L^n]. \end{cases} \tag{22}$$

The conditions for the outer boundaries of this system, i.e., at $l_p = 0$ and $l_q = M^n_q$, are the same as for the full system. The inner boundaries, $l_p = M^n$ and $l_q = 0$ are connected according to the method described in [8] to be explained shortly. To be able to calculate $p^{n+1}_{M^n}$ and q^{n+1}_0, the domains of v and w have been extended at the inner boundaries to include $v^{n+1/2}_{M^n+1/2}$ and $w^{n+1/2}_{-1/2}$. These, however, require points outside of the domains of $p^n_{l_p}$ and $q^n_{l_q}$, i.e., $p^n_{M^n+1}$ and q^n_{-1}. In [8] we propose to calculate these *virtual grid points* based on known values of the system. Despite the fact that [8] presents the method using a second-order system, it can still be applied here. The process of how $p^{n+1}_{M^n}$ and q^{n+1}_0 are calculated is visualised in in Figure 2. Notice that all time steps use the same value of M^n and M^n_q. In other words, the expansion of the temporal operators in (9) do not affect the temporal indices n in M^n and M^n_q.

4.1 Changing the Tube Length

In the following, the location of a grid point u_l along the grid (in m from the left boundary) at time index n is denoted as $x^n_{u_l}$.

The two pairs of first order systems in (21) are placed on the same domain x with

$$x^n_{p_{l_p}} = l_p h, \quad \text{and} \quad x^n_{q_{l_q}} = L^n - (M^n_q - l_q)h, \tag{23}$$

describing the locations of the left system and right system respectively. Here, it can be observed that as the tube length L^n changes, the locations of the grid points of the right system will change. More specifically, as the trombone-slide is extended and L^n increases, all grid points of the right system move to the right, and to the left for a contracting slide. If L^n is changed in a smooth fashion, the continuous domain $x \in [0, L^n]$ will not necessarily be subdivided into an integer amount of intervals N^n (of size $h = ck$). This is where a *fractional* number of intervals is introduced and is defined as

$$\mathcal{N}^n = L^n/h, \tag{24}$$

which is essentially the calculation of N in Eq. (17) without the flooring operation, and $N^n = \lfloor \mathcal{N}^n \rfloor$. The fractional part of $\mathcal{N}^n$ can then be calculated using

$$\alpha = \alpha^n = \mathcal{N}^n - N^n, \tag{25}$$

which describes the distance between the inner boundaries along the grid in terms of how many times h would fit in-between (which is always less than once). If $\mathcal{N}^n = N^n$ and $\alpha = 0$, the inner boundary locations perfectly overlap, and $x^n_{p_{M^n}} = x^n_{q_0}$. This also means that the domain x can be exactly divided into N^n equal intervals of size $h = ck$. As the virtual grid points $p^n_{M^n+1}$ and q^n_{-1} perfectly overlap with q^n_1 and $p^n_{M^n-1}$ respectively, these values can be used directly to calculate the grid points at the inner boundaries. This situation effectively acts as a rigid connection between the grid points at the inner boundaries defined as

$$p^n_{M^n} = q^n_0, \quad \text{if } \alpha = 0. \tag{26}$$

If $\alpha \neq 0$, some other definition for $p^n_{M^n+1}$ and q^n_{-1} needs to be found. We use quadratic Lagrangian interpolation according to

$$p^n_{M^n+1} = \frac{\alpha-1}{\alpha+1}p^n_{M^n} + q^n_0 - \frac{\alpha-1}{\alpha+1}q^n_1, \tag{27a}$$

$$q^n_{-1} = -\frac{\alpha-1}{\alpha+1}p^n_{M^n-1} + p^n_{M^n} + \frac{\alpha-1}{\alpha+1}q^n_0, \tag{27b}$$

which can then be used to calculate $v^{n+1/2}_{M^n+1/2}$ and $w^{n+1/2}_{-1/2}$ and consequently $p^{n+1}_{M^n}$ and q^{n+1}_0 (see Figure 2). This process is repeated every sample. It can be shown through the rigid connection in (26), that if $\alpha = 0$, the definitions in (27) reduce to $p^n_{M^n+1} = q^n_1$ and $q^n_{-1} = p^n_{M^n-1}$ as stated before.

4.2 Adding and removing grid points

As the tube length L^n changes, L^n_p and L^n_q also change according to

$$L^n_p = L^{n-1}_p + 0.5L^n_{\text{diff}}, \quad L^n_q = L^{n-1}_q + 0.5L^n_{\text{diff}}, \tag{28}$$

where

$$L^n_{\text{diff}} = L^n - L^{n-1}, \tag{29}$$

which causes the number of intervals between grid points M^n and M^n_q to change as well, according to Eq. (20).

The following state vectors are introduced for the pressure, defined for $n+1$ and n

$$\mathbf{p}^n = [p^n_0, p^n_1, ..., p^n_{M^n}]^T, \ \mathbf{q}^n = [q^n_0, q^n_1, ..., q^n_{M^n_q}]^T, \tag{30}$$

and for the velocity, defined for $n + 1/2$ and $n - 1/2$

$$\begin{aligned} \mathbf{v}^{n-1/2} &= [v^{n-1/2}_{1/2}, v^{n-1/2}_{3/2}, ..., v^{n-1/2}_{M^n+1/2}]^T, \\ \mathbf{w}^{n-1/2} &= [w^{n-1/2}_{-1/2}, w^{n-1/2}_{1/2}, ..., w^{n-1/2}_{M^n_q-1/2}]^T, \end{aligned} \tag{31}$$

and contain the different states over the discrete domains defined at the beginning of this section. Here, T denotes the transpose operation.

If $N^n > N^{n-1}$, points are added to the left and right system in an alternating fashion:

$$\begin{aligned} &\begin{cases} \mathbf{p}^n = [(\mathbf{p}^n)^T, I_3\mathbf{r}^n]^T \\ \mathbf{v}^{n-1/2} = [(\mathbf{v}^{n-1/2})^T, I_3\mathbf{z}^{n-1/2}_v]^T \end{cases} \quad \text{if } N^n \text{ is odd,} \\ &\begin{cases} \mathbf{q}^n = [I^{\leftarrow}_3\mathbf{r}^n, (\mathbf{q}^n)^T]^T \\ \mathbf{w}^{n-1/2} = [I^{\leftarrow}_3\mathbf{z}^{n-1/2}_w, (\mathbf{w}^{n-1/2})^T]^T \end{cases} \quad \text{if } N^n \text{ is even,} \end{aligned} \tag{32}$$

where

$$\begin{aligned} \mathbf{r}^n &= [p^n_{M^n-1}, p^n_{M^n}, q^n_0, q^n_1]^T, \\ \mathbf{z}^{n-1/2}_v &= [v^{n-1/2}_{M^n-1/2}, v^{n-1/2}_{M^n+1/2}, w^{n-1/2}_{1/2}, w^{n-1/2}_{3/2}]^T - \boldsymbol{\eta}, \\ \mathbf{z}^{n-1/2}_w &= [v^{n-1/2}_{M^n-3/2}, v^{n-1/2}_{M^n-1/2}, w^{n-1/2}_{-1/2}, w^{n-1/2}_{1/2}]^T + \boldsymbol{\eta}^{\leftarrow}, \end{aligned} \tag{33}$$

and cubic Lagrangian interpolator

$$I_3 = \begin{bmatrix} -\frac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} & \frac{2\alpha}{\alpha+2} & \frac{2}{\alpha+2} & -\frac{2\alpha}{(\alpha+3)(\alpha+2)} \end{bmatrix}. \tag{34}$$

Here,

$$\boldsymbol{\eta} = \boldsymbol{\eta}^{n-1/2} = \left(w^{n-1/2}_{-1/2} - v^{n-1/2}_{M^n+1/2}\right) \cdot [0, 0, 1, 1]^T \tag{35}$$

adds an offset to half of the elements in the $\mathbf{z}$ vectors depending on the difference between $v^{n-1/2}_{M^n+1/2}$ and $w^{n-1/2}_{-1/2}$. Why this is necessary will be further explained in Section 4.3. Finally, $I^{\leftarrow}_3$ and $\boldsymbol{\eta}^{\leftarrow}$ are flipped versions of (34) and (35) respectively.

If $N^n < N^{n-1}$, points are simply removed from the vectors according to

$$\begin{cases} \mathbf{p}^n = [p_0^n, \dots, p_{M^n-1}^n]^T \\ \mathbf{v}^{n-1/2} = [v_{1/2}^{n-1/2}, \dots, v_{M^n-1/2}^{n-1/2}]^T \end{cases} \quad \text{if } N^n \text{ is even,}$$
$$\begin{cases} \mathbf{q}^n = [q_1^n, \dots, q_{M_q^n}^n]^T \\ \mathbf{w}^{n-1/2} = [w_{1/2}^{n-1/2}, \dots, w_{M_q^n-1/2}^{n-1/2}]^T \end{cases} \quad \text{if } N^n \text{ is odd.} \tag{36}$$

Notice that the even and odd conditions in Eqs. (32) and (36) can be swapped. To stay as close to the desired location of adding and removing grid points as possible, this requires the ceiling and flooring operations in (20) to be swapped as well.

4.3 Drift of w

The inner boundaries of the pressure states p and q are connected by (27), but no such connection exists for the velocity states v and w. As the radiating boundary is implemented on the pressure grid, this leaves w without any boundary condition; it is only "held in place" by the pressure values of q, or more specifically, by derivatives (both spatial and temporal). As FD schemes are an approximation, it does not give a perfect solution and w tends to 'drift' during the simulation, especially when L^n is changed.

Luckily, as the pressure values are also calculated from derivatives of the velocity, the absolute state of w does not matter. The difference in values at the connection point is also irrelevant as there is no spatial derivative taken between v and w (refer to Figure 2). Finally, the pressure values are used for the output audio of the simulation, so the drift does not affect the audio.

The absolute states of the velocity vectors do, however, need to be accounted for when adding points to the $\mathbf{v}$ and $\mathbf{w}$ using (32). The current drift can be approximated by observing the difference between $w_{-1/2}^{n-1/2}$ and $v_{M^n+1/2}^{n-1/2}$, as these have approximately the same x location ($x_{w_{-1/2}}^n \approx x_{v_{M^n+1/2}}^n$) when a grid point is added. This is then used in a drift-correction vector $\boldsymbol{\eta}^{n-1/2}$ presented in (35). When a point is added to v, the values of w in $\mathbf{z}_v$ are offset by the aforementioned difference and when a point is added to w the same happens (inverted) for the values of v in $\mathbf{z}_w$. This way, the drift is allowed, but does not affect the state of the newly added grid points. Notice that the drift does not affect the operations of point removal in (36).

4.4 State Correction

As L^n, and consequently the number of grid points, is decreased, it might occur that the grid points at the inner boundaries $p_{M^n}^n$ and q_0^n have a very different value when $\alpha \gtrsim 0$, i.e., right before a point is removed. This violates the rigid connection in Eq. (26).

We propose in [8] to add an artificial spring-like connection between the grid points at the inner boundaries that "corrects" the state of these points. Applying this to system (21) extends Eqs. (21a) and (21c) according to

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}p_{l_p}^n = -\delta_{x-}(S_{l+1/2}v_{l_p+1/2}^{n+1/2}) + J_p(x_{p_{M^n}}^n)F_{\text{sc}}^n, \tag{37a}$$

$$\frac{\bar{S}_l}{\rho_0 c^2}\delta_{t+}q_{l_q}^n = -\delta_{x+}(S_{l-1/2}w_{l_q-1/2}^{n+1/2}) - J_q(x_{q_0}^n)F_{\text{sc}}^n, \tag{37b}$$

where the spreading operators are defined as

$$\begin{aligned} J_p(x_i^n) &= \begin{cases} \frac{1}{h}, & l_p = \lfloor x_i^n/h \rfloor \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \\ J_q(x_i^n) &= \begin{cases} \frac{1}{h}, & l_q = \lfloor x_i^n/h \rfloor - M^n \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{38}$$

Furthermore, the correction effect is defined as

$$F_{\text{sc}}^n = \beta\left(\mu_{t\cdot}\eta_{\text{sc}}^n + \sigma_{\text{sc}}\delta_{t\cdot}\eta_{\text{sc}}^n\right), \tag{39}$$

with spring damping σ_{sc}, pressure difference

$$\eta_{\text{sc}}^n \triangleq q_0^n - p_{M^n}^n, \tag{40}$$

and scaling coefficient

$$\beta = \beta(\alpha) = \frac{1-\alpha}{\alpha+\varepsilon}. \tag{41}$$

Here, $\varepsilon \ll 1$ to prevent division by 0. Just like in [8], the implementation of the correction effect allows for an infinite β when $\alpha = \varepsilon = 0$ acting like a rigid connection between Eqs. (37a) and (37b).

5 Implementation

The implementation has been done in C++ using the JUCE framework [1], and is available online[2] as well as a demo showcasing it.[3] The audio output of the system can be retrieved by selecting a grid point on the pressure grid and listening to this at the given sample rate f_s. Here, the radiating boundary $q_{M_q^n}^n$ is chosen, as this is where the sound enters the listening space in the real

[1] https://juce.com/

[2] https://github.com/SilvinWillemsen/cppBrass/releases/

[3] https://youtu.be/Ht5gVNrshYo

Table 1: *Geometry of a measured trombone taken from [16]. Numbers correspond to Figure 3.*

Part of tube	Length (cm)	Radius (cm)
Inner slide (1)	70.8	0.69
Outer slide (extended) (2)	53	0.72
Slide crook (3)	17.7	0.74
Outer slide (extended) (4)	53	0.72
Inner slide (5)	71.1	0.69
Gooseneck (6)	24.1	0.71
Tuning slide (7)	25.4	0.75, 1.07
Bell flare (8)	50.2	1, 10.8

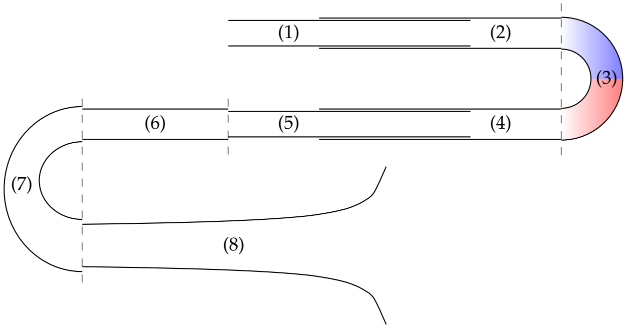


Fig. 3: *Diagram showing the trombone geometry (not to scale). Numbers correspond to the parts of the tube found in Table 1 and dashed lines highlight where the different parts are separated. The tube is split in the middle of the slide crook with the colours corresponding to those in Figure 2.*

world. To mimic low-pass filtering happening due to a distributed radiating area, a 4th-order low-passing Butterworth filter with a cutoff frequency of $f_\text{c} = \sqrt{c^2\pi/S(L)} \approx 3245$ Hz is used. This equation is retrieved by choosing the listening point to be at the bell surface and integrating over the bell area.

5.1 Parameters

For the most part, the parameters used in the simulation have been obtained from [13, 16, 17]. The lengths and radii of different parts of the tube can be found in Table 1 and a diagram showing this geometry is shown in Figure 3. The system is split in the middle of the slide crook such that the ranges for the lengths of the two tubes are $L_p^n \in [0.797, 1.327]$ and $L_q^n \in [1.796, 2.326]$.

Other parameters used in the simulation can be found in Table 2. Not included here is λ, which has been set slightly lower than the stability condition in (16), i.e., $\lambda = 0.999$. Although the implementation works when $\lambda = 1$, this is done to tolerate (much) higher speeds of change in L^n before instability occurs (see Section 5.2). Not satisfying condition (16) causes bandlimiting and

Table 2: *List of parameter values used for the simulation. Taken from ⋆ [16], *[13] or **[17] with temperature* $T = 26.85°C$.

Name	Symbol (unit)	Value
Tube		
Length	L (m)	$2.593 \leq L \leq 3.653^\star$
Air density	ρ_0 (kg/m^3)	1.1769**
Wave speed	c (m/s)	347.23**
Geometry	S (m^2)	See Table 1.
Lip reed		
Mass	M_r (kg)	$5.37 \cdot 10^{-5}$*
Frequency	ω_r (rad/s)	$20 \leq \omega_\mathrm{r}/2\pi \leq 1000$
Mouth pressure	P_m (Pa)	$0 \leq P_\mathrm{m} \leq 6000$
Damping	σ_r (s^{-1})	5*
Eff. surface area	S_r (m^2)	$1.46 \cdot 10^{-5}$*
Width	w_r (m)	0.01*
Equilibrium sep.	H_0 (m)	$2.9 \cdot 10^{-4}$*
Coll. stiffness	K_c (N/m)	10^4
Nonlin. coll. coeff.	α_c (-)	3
Other		
State corr. damping	σ_sc	1
Sample rate	f_s (Hz)	44100

dispersive effects [15], but such a small deviation from the condition has no perceptual influence on the output sound and outweighs the problems caused by instability.

As the tube acts mainly as an amplifier for specific resonant frequencies it is important to match the frequency of the lip reed to a resonating mode of the tube. This frequency depends on L^n in the following way

$$\omega_\mathrm{r}^{n+1/2} = \mathcal{F}\frac{2\pi c}{\rho_0 L^{n+1/2}}\,, \tag{42}$$

where $L^{n+1/2} = L^n$ and scalar multiplier $\mathcal{F} = 2.4$ was heuristically found to best match the 4th resonating mode of the tube and generates a recognisable brass sound.

5.2 Limit on speed of change

To reduce audible artifacts and instability issues from adding and removing points, and to stay in the sub-audio rate regime, a limit can be placed on (29) as

$$L^n_\text{diff} \leq \mathcal{N}_\text{maxdiff}h, \tag{43}$$

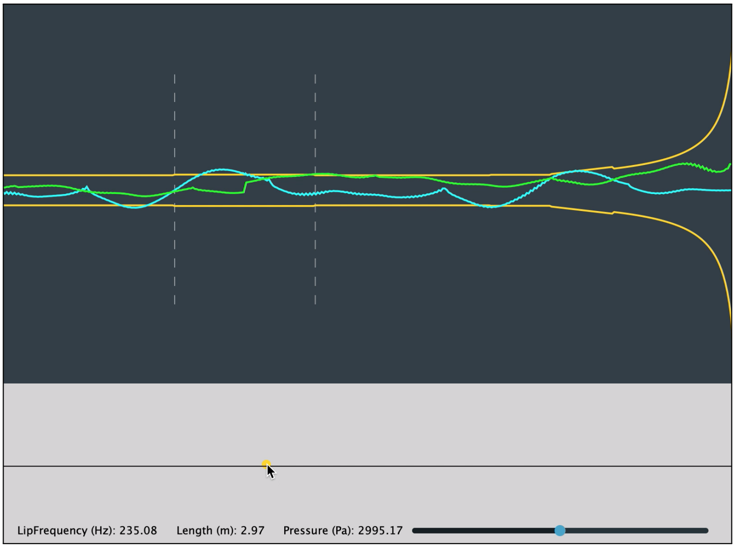

LipFrequency (Hz): 235.08 Length (m): 2.97 Pressure (Pa): 2995.17

Fig. 4: *Screenshot of the graphical user interface (GUI). The geometry (in orange) as well as the states of the pressure (in blue) and velocity scaled by S (in green) are shown. For clarity, the start and end of the outer slide are denoted by dashed lines. The drift of w as explained in Section 4.3 is visible from the "kink" in the green line exactly in the middle of the outer slide.*

where $\mathcal{N}_{\text{maxdiff}}$ is the maximum change in $\mathcal{N}$ per sample and has been set to $\mathcal{N}_{\text{maxdiff}} = 1/20$. This means that a grid point can be added or removed every 20 samples and allows the entire range of L to be traversed in ca. 0.06 s at a sample rate of $f_s = 44100$ Hz.

5.3 State correction

The introduction of system states at $n+1$ through the centred operators in Eq. (39) seem to make the scheme implicit. It is, however, possible to calculate F_{sc} explicitly [15, 18]. The same operators also introduce the need for values at $n-1$, i.e., $p_{M^n}^{n-1}$ and q_0^{n-1}. Therefore, the vectors $\mathbf{p}^{n-1}$ and $\mathbf{q}^{n-1}$ will need to be stored, and the operations to add and remove grid points as described in 4.2 need to be applied to these as well. One could argue that only two points at the inner boundaries are needed for the calculation and to create $\mathbf{r}$ in (33) at $n-1$. For generality, we continue with the entire vectors defined over the same domains as $\mathbf{p}^n$ and $\mathbf{q}^n$ respectively.

5.4 Graphical User Interface and Control Mapping

A screenshot of the graphical user interface (GUI) is shown in Figure 4. The geometry of the tube is plotted along with paths showing the pressure states

```
while application is running do
    Retrieve new parameters                 ($L^n$, $\omega_\mathrm{r}^n$ and $P_\mathrm{m}^n$)
    Update $L_p^n$ and $L_q^n$                      (Eqs. (29), (43) & (28))
    Calc. $\mathcal{N}^n$ and $N^n$                     (Eqs. (24) and (17))
    Calc. $\alpha^n$                              (Eq. (25))
    if $N^n \neq N^{n-1}$ then
        Add or remove point                 (Eq. (32) or (36))
        Update $M^n$ and $M_q^n$                  (Eq. (20))
    end
    Calc. $p_{M^n+1}^n$ and $q_{-1}^n$                 (Eqs. (27))
    Calc. $\mathbf{v}^{n+1/2}$ and $\mathbf{w}^{n+1/2}$             (Eqs. (21b) and (21d))
    Calc. $y^{n+3/2}$ w/o collision              (Eqs. (18))
    Calc $g^{n+1/2}$                         (Eq. (19))
    Calc. $y^{n+3/2}$ with collision             (Eqs. (18))
    Calc. $U_\mathrm{B}^{n+1/2}$ and $U_\mathrm{r}^{n+1/2}$             (Eqs. (18c) and (18d))
    Calc. $\mathbf{p}^{n+1}$ and $\mathbf{q}^{n+1}$                 (Eqs. (37))
    Retrieve output
    Update system states              ($\mathbf{p}^{n-1} = \mathbf{p}^n$, $\mathbf{p}^n = \mathbf{p}^{n+1}$) (same for
                                      $\mathbf{v}^{n-1/2} = \ldots$,
                                      $y^{n-1/2}$, $y^{n+1/2}$, and $\psi^n$)
    Update $N^{n-1}$                        ($N^{n-1} = N^n$)
    Increment $n$
end
```

Algorithm 1: *Pseudocode showing the order of calculations of the algorithm implementing the trombone.*

in blue and the velocity (scaled by the geometry S) in green. The audio thread of the application runs at 44100 Hz whereas the graphics are updated at a rate of 15 Hz.

The real-time application is controlled by interacting with the bottom panel using the mouse. The x-axis is mapped to tube-length L^n and also modifies the lip-reed frequency ω_r according to Eq. (42). The y-axis changes the multiplier $\mathcal{F}$ in Eq. (42) and the black line in the vertical middle of the control panel is mapped to $\mathcal{F} = 2.4$. The pressure is modulated by a slider at the bottom of the control panel. As of now, no focus has been put on intuitive parameter mapping; it has only been implemented for simple parameter exploration.

5.5 Order of Calculation

Algorithm 1 shows the order in which the different parts of the system presented in this paper are calculated.

6 Results and Discussion

The real-time implementation has been tested on a MacBook Pro with a 2.2 GHz Intel i7 processor and was informally evaluated by the authors. The speed of the algorithm was tested with and without the graphics-thread and using three different styles of interaction: static excitation at the shortest and longest length, and rapidly (and continuously) changing L and ω_r between their minimum and maximum values given in Table 2. The pressure was kept at $P_{\mathrm{m}} = 3000$ Pa at all times. The results are shown in Table 3. Differences in CPU usage between a short and long tube length are because more grid points need to be calculated in the long case. The recalculation of the geometry maximally once every 20 samples in the rapidly moving case explains the increase in CPU usage there. These results show that the implementation can easily be used as an audio plugin, with or without graphics.

Table 3: *Average CPU usage (in %) for different graphics settings and various interactions with the application.*

Tube length	Graphics (%)	No graphics (%)
Short ($L^n = 2.593$ m)	12.1	4.3
Long ($L^n = 3.653$ m)	14.4	5.2
Rapidly changing	17.7	10.1

Informal listening tests by the authors confirm that the audio output of the simulation exhibits brass-like qualities. However, the implementation requires some further refinements to be considered as a complete trombone. Possible extensions to improve the realism of the simulation sound could be to add viscothermal losses [19] or nonlinear effects [3]. Furthermore, for lower values of the lip frequency ω_{r}, the sound exhibits extra oscillatory behaviour making the output "non-smooth". This might be due a higher average displacement of y for lower ω_{r} and the nonlinear collision present in the lip model will have a greater effect on its displacement. Variable collision stiffness might solve this issue but is left for future work.

Informal listening by the authors shows that the method used to implement the dynamic grid does not introduce perceivable audible artifacts, even when L^n is changed very rapidly. Naturally, this needs to be confirmed by formal listening tests. Despite the limit placed on the speed of change of L^n in (43) the control of the application does not exhibit a noticeable delay and changes in L^n feel immediate.

The main difference between the method in [8] and the version used here, is that the method is applied to a system of first-order equations rather than the second-order 1D wave equation. Because the connection between the inner boundaries is only applied to the grid functions describing pressure, a drift occurs in w as it is left without boundary conditions. Although this drift does not have an effect on the output sound, as discussed in Section 4.3, too high or low values might cause rounding errors in the simulation. As it is expected that this only happens at extremely high or low values after a long simulation length, the drift is not considered an issue at this point.

7 Conclusion

In this paper, we have presented a full implementation of the trombone including a lip reed, radiation and a tube, discretised using FDTD methods on a dynamic grid. Informal evaluation by the authors shows that the implementation exhibits no audible artifacts when grid points are added and removed, even under relatively fast variation in tube length. Naturally, this needs to be confirmed by formal listening tests. Moreover, the simulation easily runs in real-time allowing it to be used as an audio plugin.

Future work will include extending the tube model to include more realistic viscothermal and nonlinear effects and variable collision stiffness in the lip model. Furthermore, the investigation of more intuitive control parameter mappings is a necessary step towards a real-time instrument.

8 Acknowledgements

This work is supported by NordForsk's Nordic University Hub Nordic Sound and Music Computing Network NordicSMC, project number 86892.

9 References

[1] M. Campbell, "Brass instruments as we know them today," *Acta Acustica united with Acustica*, vol. 90, no. 4, pp. 600–610, 2004.

[2] A. Hirschberg, J. Gilbert, R. Msallam, and A. Wijnands, "Shock waves in trombones," *J. Acoust. Soc. Am.*, vol. 99, no. 3, pp. 1754–1758, 1996.

[3] R. Msallam, S. Dequidt, S. Tassart, and R. Causse, "Physical model of the trombone including non-linear propagation effects," in *ISMA: International Symposium of Music Acoustics*, 1997.

[4] R. Msallam, S. Dequidt, R. Causse, and S. Tassart, "Physical model of the trombone including nonlinear effects. Application to the sound synthesis of loud tones," *Acta Acustica united with Acustica*, vol. 86, no. 4, pp. 725–736, 2000.

[5] P. R. Cook, *Real sound synthesis for interactive applications*. CRC Press, 2002.

[6] R. L. Harrison, S. Bilbao, J. Perry, and T. Wishart, "An environment for physical modeling of articulated brass instruments," *Computer Music Journal*, vol. 39, no. 4, pp. 80–95, 2015.

[7] S. Bilbao and J. Chick, "Finite difference time domain simulation for the brass instrument bore," *J. Acoust. Soc. Am.*, vol. 134, no. 6, pp. 3860–3871, 2013.

[8] S. Willemsen, S. Bilbao, M. Ducceschi, and S. Serafin, "Dynamic grids for finite-difference schemes in musical instrument simulations," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects (DAFx)*, 2021.

[9] A. Webster, "Acoustical impedance, and the theory of horns and of the phonograph," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 5, no. 7, pp. 275–282, 1919.

[10] S. Bilbao and R. Harrison, "Passive time-domain numerical models of viscothermal wave propagation in acoustic tubes of variable cross section," *J. Acoust. Soc. Am.*, vol. 140, pp. 728–740, 2016.

[11] H. Levine and J. Schwinger, "On the radiation of sound from an unflanged circular pipe," *Physical Review*, vol. 73, no. 2, pp. 383–406, 1948.

[12] F. Silva, P. Guillemain, J. Kergomard, B. Mallaroni, and A. Norris, "Approximation formulae for the acoustic radiation impedance of a cylindrical pipe," *Journal of Sound and Vibration*, vol. 322, pp. 255–263, 2009.

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

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

[15] S. Bilbao, *Numerical Sound Synthesis*. United Kingdom: John Wiley & Sons, 2009.

[16] T. Smyth and F. S. Scott, "Trombone synthesis by model and measurement," *EURASIP Journal on Advances in Signal Processing*, 2011.

[17] A. H. Benade, "On the propagation of sound waves in a cylindrical conduit," *Journal of the Acoustical Society of America*, vol. 44, no. 2, pp. 616–623, 1968.

[18] S. Bilbao, "A modular percussion synthesis environment," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx)*, 2009.

[19] S. Bilbao and R. L. Harrison, "Passive time-domain numerical models of viscothermal wave propagation in acoustic tubes of variable cross section," *The Journal of the Acoustical Society of America*, vol. 140, pp. 728–740, 2016.

Part VIII

Appendix

Appendix A

Paper Errata

This appendix presents a few errors that appeared in the papers [D] and [F].

Real-time Implementation of a Physical Model of the Tromba Marina [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 be $\sigma_{1,\mathrm{p}}$.
- The unit of the spatial Dirac delta function δ should be m^{-1}.

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study [F]

- σ_0 and σ_1 in Eq. (1) should be multiplied by ρH in order for the stability condition to hold.
- The stability condition is wrong. It should be:

$$h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}. \tag{A.1}$$

- The unit for membrane tension is N/m.

Appendix B

Matrices

This appendix aims to provide some fundamental knowledge on matrices and linear algebra used throughout this thesis.

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 B.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.

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.

B.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 B.2 for more information.

Matrix transpose

A matrix or vector can be *transposed,* which is indicated by 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

$$a_{ij} = a_{ji}. \tag{B.1}$$

Also see Figure B.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.

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

(a) A 3×5 matrix $\mathbf{A}$.

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

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

Fig. B.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. (B.1).

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

$$c_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj}, \tag{B.2}$$

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

$$\overbrace{(L \times M)}^{\mathbf{A}} \cdot \overbrace{(M \times N)}^{\mathbf{B}} = \overbrace{(L \times N)}^{\mathbf{C}}. \tag{B.3}$$

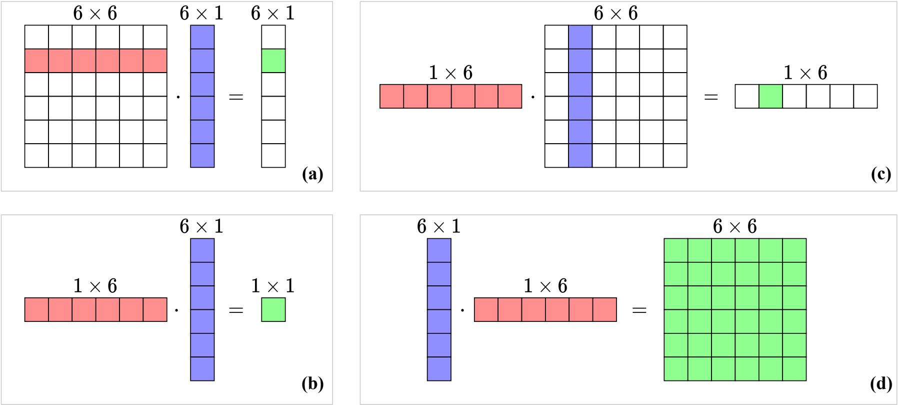


Fig. B.2: Visualisation of valid matrix multiplications (see Eq. (B.2)). The "inner" dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of "outer" dimensions (rows of the left matrix and columns of the right).

B.2 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 invertible if there exists a matrix $\mathbf{B}$ such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \tag{B.4}$$

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:

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

B.3 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:

$$\mathbf{Au} = \mathbf{w}. \tag{B.5}$$

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:

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

This can be solved for $\mathbf{u}$ by taking the inverse of $\mathbf{A}$ (see Section B.2) and multiplying this onto $\mathbf{w}$

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \tag{B.6}$$

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

Solving a system of linear equations can be implemented in `MATLAB` by using the code given in Section B.2 and multiplying this onto a vector `w`

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

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

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

B.4 Eigenvalue problems

A square matrix **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 provides 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

$$\mathbf{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}. \tag{B.7}$$

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

$$\lambda_p = \text{eig}_p(\mathbf{A}), \tag{B.8}$$

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 corresponding eigenvalues are given in a $p \times 1$ column vector `lambda`.

Appendix C

Code Examples

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
```

C.3 2D wave equation (Section 6.2)

```
%% Initialise variables
fs = 44100;         % Sample rate [Hz]
k = 1 / fs;         % Time step [s]
lengthSound = fs;   % Length of the simulation (1 second) [samples]

rho = 7850;                     % Material density [kg/m^3]
H =  0.0005;                    % Thickness [m]
T = 1000000;                    % Tension per unit length [N/m]
c = sqrt(T / (rho * H));        % Wave speed [m/s]

Lx = 1;                         % Length in x direction [m]
Ly = 2;                         % Length in y direction [m]

h = sqrt(2) * c * k;            % Grid spacing [m]
Nx = floor(Lx/h);               % Number of intervals in x direction
Ny = floor(Ly/h);               % Number of intervals in y direction
h = min(Lx/Nx, Ly/Ny);          % Recalculation of grid spacing

lambdaSq = c^2 * k^2 / h^2;     % Courant number squared
h = max(Lx/Nx, Ly/Ny);          % Recalculation of grid spacing

%% Create scheme matrices with Dirichlet boundary conditions
Nxu = Nx - 1;
Nyu = Ny - 1;
Dxx = toeplitz([-2, 1, zeros(1, Nxu-2)]);
Dyy = toeplitz([-2, 1, zeros(1, Nyu-2)]);

% Kronecker sum
D = kron(speye(Nxu), Dyy) + kron(Dxx, speye(Nyu));
D = D / h^2;
```

```
% Total number of grid points
Nu = Nxu * Nyu;

%% Initialise state vectors (one more grid point than the number of
    intervals)
uNext = zeros(Nu, 1);
u = zeros(Nu, 1);

%% Initial conditions (2D raised cosine)
halfWidth = floor(min(Nx, Ny) / 5);
width = 2 * halfWidth + 1;
xLoc = floor(0.3 * Nx);
yLoc = floor(0.6 * Ny);
xRange = xLoc-halfWidth : xLoc+halfWidth;
yRange = yLoc-halfWidth : yLoc+halfWidth;

rcMat = zeros(Nyu, Nxu);
rcMat(yRange, xRange) = hann(width) * hann(width)';

% initialise current state
u = reshape(rcMat, Nu, 1);

% Set initial velocity to zero
uPrev = u;

% Output location
xOut = 0.45;
yOut = 0.25;
outLoc = round((xOut + yOut * Nyu) * Nxu);
out = zeros(lengthSound, 1);

%% Simulation loop
for n = 1:lengthSound

    %% Update equation Eq. (6.21)
    uNext = (2 * eye(Nu) + c^2 * k^2 * D) * u - uPrev;

    % 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. (4.4):

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa^2\partial_x^4 u - 2\sigma_0\partial_t u + 2\sigma_1\partial_t\partial_x^2 u, \tag{D.1}$$

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.

First, consider the meaning behind positive and negative curvature, i.e., when $\partial_x^2 u > 0$ or $\partial_x^2 u < 0$. Counter-intuitively, 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. This can be proven by taking $x = 0$ and setting grid spacing $h = 1$.

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

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

is positive.

As the term does not only include a second-order spatial derivative but also a first-order time derivative, this is referred to as 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 the 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, e.g. $f(x,t) = tx^2$. When t increases over time, the curvature gets bigger. Repeating the above with $x = 0$ and grid spacing $h = 1$, but now with $t = 2$ and step size $k = 1$, and with a backwards time derivative yields:

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

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, a higher curvature in space (generally) corresponds to a higher frequency in time. As the frequency-dependent damping term aims to reduces the curvature along the string, it effectively damps higher frequencies.

Appendix E

Considerations in real-time FD schemes

As an addition to what has been presented in Chapter 13, this appendix provides some considerations when working with FD schemes in real-time applications. Consider this some '*tips and tricks*'.

Prototyping in MATLAB

Before creating any real-time FD scheme, it is usually a good idea to prototype a physical modelling application in `MATLAB`. Various reasons include, but are not limited to:

1. The code is easier to write and debug.
2. Data is more easily visualised, allowing for better insight to, e.g., the state of your system (one could use `drawnow` for real-time plotting).
3. There is no need for memory handling.
4. Programming errors causing unstable schemes do not happen in real time.

Creating a limiter

Stability issues in FD schemes have been mentioned several times in this work. To protect speakers, headphones, or – most importantly – ears, it is thus important to implement a limiter as one of the first things in any real-time application.

Below is an example of a limiter that limits an input value between -1 and 1.

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

Vector indexing

One of the most common sources of error when translating `MATLAB` code to C++ (apart from the differences in syntax), is the fact that `MATLAB` is 1-based, and C++ is 0-based, which refers to the indexing of a vector. If `u` is a vector with 10 elements, the first element is retrieved as `u(1)` and the last as `u(10)`. In C++, on the other hand, retrieving the first and last element of a size-10 vector happens through `u[0]` and `u[9]` respectively.

Real-time control

For the real-time control of any application using FD schemes, whether it uses the mouse or an external controller, the important thing is to not affect the scheme while it is performing the update equation. If the system is excited while the scheme is calculated, this might cause instability, or at the very least, unpredictable behaviour. The best thing to do is to work with flags that get triggered by outside control and get checked once per sample (or even only once per buffer), before the calculation of the scheme.

Premature optimisation

As Donald E. Knuth states "premature optimization is the root of all evil [...] in programming" [124] (see full quote at the start of Chapter 13). Often during this project, much time was spent trying to find a variable mistake or a sign error that could have easily been prevented if the code was not prematurely optimised.

Most often it will take less time to write the code in a non-optimised, but understandable way, followed by several iterations of optimisation. One might even find that the difference in speed is extremely small and might not even need the optimisation at all.

Debug and Release modes

This might seem trivial for C++ developers, but whether one builds the application in 'Debug' or 'Release' matters a lot for the eventual speed of the algorithm. Depending on the mode, the compiler uses different optimisation flags and could increase the speed of the application tremendously. The Debug mode is still useful, as – apart from being able to debug the code – building the application takes (much) less time, depending on the size of the application.

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 value of the damping coefficients, state values can get in the range of ~10^{-307}! Numbers in this range are referred to as *denormalised numbers* and operations performed with these are "extremely slow" [125].

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 strong damping in the body in the tromba marina presented in Chapter 15, denormalised numbers appear after only ~10 s of not interacting with the instrument, and the CPU usage increases considerably. 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 closer to zero than e.g. 10^{-250}, and truncate all values of that system to 0, as was done in paper [D].

Appendix F

Derivations

This appendix contains derivations of several equations used in this thesis.

F.1 Summation by parts

To see why some of the summation by parts identities in Section 3.2.2 hold true, it is useful to briefly go through a derivation. As an example, Eqs. (3.12a) and (3.13a) are derived 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. (3.12a), suppressing the n superscript for brevity, and using the definition for the discrete inner product in Eq. (3.8), yields

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

As $N = 2$, the result is identical to Eq. (3.12a).

Similarly, identity (3.13a) 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 $\underline{d} = \{0, \dots, N-1\}$. Similar processes can be used to prove the other identities presented in Section 3.2.2.

F.2 von Neumann analysis damped stiff string

This section performs the von Neumann analysis presented in Section 4.3 in greater detail and derives the stability condition for the damped stiff string.

Starting with the characteristic equation in Eq. (4.23):

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

one can rewrite this to the form in Eq. (3.25), 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 (3.26), and substituting the coefficients into this condition yields

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

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

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

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:

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

As the left-hand side takes its maximum value for $\mathcal{S}=1$, one can substitute this and continue with the substituted definitions for λ and μ from Eq. (4.11) to yield

$$
\begin{gathered}
\frac{4\kappa^2k^2}{h^4}+\frac{c^2k^2+4\sigma_1 k}{h^2}\le 1,\\
4\kappa^2k^2+(c^2k^2+4\sigma_1 k)h^2\le h^4,\\
h^4-(c^2k^2+4\sigma_1 k)h^2-4\kappa^2k^2\ge 0,
\end{gathered}
$$

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

$$
h\ge\sqrt{\frac{c^2k^2+4\sigma_1 k+\sqrt{(c^2k^2+4\sigma_1 k)^2+16\kappa^2k^2}}{2}}, \tag{F.1}
$$

which is the stability condition for the damped stiff string also shown in Eq. (4.12).

F.3 von Neumann analysis implicit damped stiff string

This section performs the von Neumann analysis presented in Section 4.6.1 in greater detail and derives the stability condition for the damped stiff string, using a centred difference operator for the frequency-dependent damping term.

Recalling the characteristic equation in Eq. (4.36):

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

one can rewrite this to the form in Eq. (3.25) and, using $\mathcal{S}=\sin^2(\beta h/2)$ for brevity, 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 (3.26), 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 yields

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

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

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

and yields the following stability condition:

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

F.4 Webster's update equation

This section derives the update equation for Webster's equation in Eq. (5.9):

$$\begin{aligned} \frac{\bar{S}}{k^2}(\Psi_l^{n+1} - 2\Psi_l^n + \Psi_l^{n-1}) &= c^2\bigg((\delta_{x-}S_{l+1/2})(\mu_{x-}\delta_{x+}\Psi_l^n) \\ &\qquad + (\mu_{x-}S_{l+1/2})(\delta_{x-}\delta_{x+}\Psi_l^n)\bigg), \\ \Psi_l^{n+1} - 2\Psi_l^n + \Psi_l^{n-1} &= \frac{c^2k^2}{\bar{S}}\bigg(\frac{1}{h}(S_{l+1/2} - S_{l-1/2})\frac{1}{2h}\overbrace{(\Psi_{l+1}^n - \Psi_{l-1}^n)}^{\mu_{x+}\delta_{x-}\Psi_l^n=\delta_{x\cdot}\Psi_l^n} \\ &\qquad + \frac{1}{2}(S_{l+1/2} + S_{l-1/2})\frac{1}{h^2}(\Psi_{l+1}^n - 2\Psi_l^n + \Psi_{l-1}^n)\bigg), \\ \Psi_l^{n+1} = 2\Psi_l^n - \Psi_l^{n-1} + &\frac{\overbrace{\lambda^2}^{\lambda=\frac{ck}{h}}}{2\bar{S}}\Big(S_{l+1/2}\Psi_{l+1}^n - S_{l+1/2}\Psi_{l-1}^n \\ &- S_{l-1/2}\Psi_{l+1}^n + S_{l-1/2}\Psi_{l-1}^n + S_{l+1/2}\Psi_{l+1}^n + S_{l+1/2}\Psi_{l-1}^n \\ &+ S_{l-1/2}\Psi_{l+1}^n + S_{l-1/2}\Psi_{l-1}^n - 2(S_{l+1/2} + S_{l-1/2})\Psi_l^n\Big), \\ \Psi_l^{n+1} = 2\Psi_l^n - \Psi_l^{n-1} + &\frac{\lambda^2}{2\bar{S}}\Big(2S_{l+1/2}\Psi_{l+1}^n + 2S_{l-1/2}\Psi_{l-1}^n - 4\bar{S}\Psi_l^n\Big), \\ \Psi_l^{n+1} = 2\Psi_l^n - \Psi_l^{n-1} + &\frac{\lambda^2 S_{l+1/2}}{\bar{S}}\Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}}\Psi_{l-1}^n - 2\lambda^2\Psi_l^n, \\ \Psi_l^{n+1} = 2(1-\lambda^2)\Psi_l^n - &\Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}}\Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}}\Psi_{l-1}^n. \end{aligned}$$

F.5 Boundary terms Webster's equation

This section derives the process of obtaining the values for ϵ_{l} and ϵ_{r} such that the boundary terms in Webster's equation are strictly dissipative. The result is presented in Eq. (5.31) and will be derived here.

Starting with the second term in the energy balance in Eq. (5.27):

$$-\,c^2\langle \delta_{t\cdot}\Psi_l^n, \delta_{x-}\big(S_{l+1/2}(\delta_{x+}\Psi_l^n)\big)\rangle_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}}, \tag{F.2}$$

and using identity (3.15a):

$$\begin{aligned}\langle f_l^n, \delta_{x-}g_l^n\rangle_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}} = -\langle \delta_{x+}f_l^n, g_l^n\rangle_{\underline{d}} + f_N^n g_{N-1}^n - f_0^n g_0^n \\ + \frac{\epsilon_{\mathrm{r}}}{2} f_N^n(g_N^n - g_{N-1}^n) + \frac{\epsilon_{\mathrm{l}}}{2} f_0^n(g_0^n - g_{-1}^n),\end{aligned}$$

this can be rewritten to (with $f = \delta_{t\cdot}\Psi$ and $g = S_{l+1/2}(\delta_{x+}\Psi)$)

$$-c^2\langle \delta_{t\cdot}\Psi_l^n, \delta_{x-}\big(S_{l+1/2}(\delta_{x+}\Psi_l^n)\big)\rangle_d^{\epsilon_{\mathrm{l}},\epsilon_{\mathrm{r}}} = c^2\langle \delta_{t\cdot}\delta_{x+}\Psi_l^n, \big(S_{l+1/2}(\delta_{x+}\Psi_l^n)\big)\rangle_{\underline{d}} - \mathfrak{b},$$

where

$$\mathfrak{b} = \mathfrak{b}_{\mathrm{r}} - \mathfrak{b}_{\mathrm{l}}, \tag{F.3}$$

with

$$\begin{aligned}\mathfrak{b}_{\mathrm{r}} = c^2(\delta_{t\cdot}\Psi_N^n)\Big(S_{N-1/2}\overbrace{(\delta_{x+}\Psi_{N-1}^n)}^{(\delta_{x-}\Psi_N^n)}\Big) \\ + \frac{\epsilon_{\mathrm{r}}}{2}(\delta_{t\cdot}\Psi_N^n)\Big(S_{N+1/2}(\delta_{x+}\Psi_N^n) - S_{N-1/2}\underbrace{(\delta_{x+}\Psi_{N-1}^n)}_{(\delta_{x-}\Psi_N^n)}\Big),\end{aligned}$$

and

$$\begin{aligned}\mathfrak{b}_{\mathrm{l}} = c^2(\delta_{t\cdot}\Psi_0^n)\Big(S_{1/2}(\delta_{x+}\Psi_0^n)\Big) \\ - \frac{\epsilon_{\mathrm{l}}}{2}(\delta_{t\cdot}\Psi_0^n)\Big(S_{1/2}(\delta_{x+}\Psi_0^n) - S_{-1/2}\underbrace{(\delta_{x+}\Psi_{-1}^n)}_{(\delta_{x-}\Psi_0^n)}\Big).\end{aligned}$$

This can be rewritten to

$$\mathfrak{b}_{\mathrm{r}} = c^2(\delta_{t\cdot}\Psi_N^n)\left(\frac{\epsilon_{\mathrm{r}}}{2}S_{N+1/2}(\delta_{x+}\Psi_N^n) + \left(1 - \frac{\epsilon_{\mathrm{r}}}{2}\right)S_{N-1/2}(\delta_{x-}\Psi_N^n)\right), \tag{F.4}$$

$$\mathfrak{b}_{\mathrm{l}} = c^2(\delta_{t\cdot}\Psi_0^n)\left(\frac{\epsilon_{\mathrm{l}}}{2}S_{-1/2}(\delta_{x-}\Psi_0^n) + \left(1 - \frac{\epsilon_{\mathrm{l}}}{2}\right)S_{1/2}(\delta_{x+}\Psi_0^n))\right). \tag{F.5}$$

Then, for the centred radiating boundary condition in Eq. (5.17) to be strictly dissipative, i.e., $\delta_{x\cdot}\Psi_l^n = 0 \;\Rightarrow\; \mathfrak{b}_{\mathrm{r}} = 0$, the special choice for $\epsilon_{\mathrm{r}} =$

$S_{N-1/2}/\mu_{xx}S_N$ needs to be made:

$$\begin{aligned}
\mathfrak{b}_\mathrm{r} &= c^2(\delta_{t\cdot}\Psi_N^n)\left(\frac{S_{N-1/2}}{2\mu_{xx}S_N}S_{N+1/2}(\delta_{x+}\Psi_N^n) + \left(1-\frac{S_{N-1/2}}{2\mu_{xx}S_N}\right)S_{N-1/2}(\delta_{x-}\Psi_N^n)\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(\frac{S_{N+1/2}}{2\mu_{xx}S_N}(\delta_{x+}\Psi_N^n) + \left(1-\frac{S_{N-1/2}}{2\mu_{xx}S_N}\right)(\delta_{x-}\Psi_N^n)\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{S_{N-1/2}}{2\mu_{xx}S_N}\right)\left(\frac{\frac{S_{N+1/2}(\delta_{x+}\Psi_N^n)}{2\mu_{xx}S_N}}{\left(1-\frac{S_{N-1/2}}{2\mu_{xx}S_N}\right)} + \delta_{x-}\Psi_N^n\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)\left(\frac{\frac{S_{N+1/2}(\delta_{x+}\Psi_N^n)}{2\mu_{xx}S_N}}{\left(\frac{2\mu_{xx}S_N-S_{N-1/2}}{2\mu_{xx}S_N}\right)} + \delta_{x-}\Psi_N^n\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)\left(\frac{S_{N+1/2}(\delta_{x+}\Psi_N^n)}{2\mu_{xx}S_N-S_{N-1/2}} + \delta_{x-}\Psi_N^n\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)\left(\frac{S_{N+1/2}(\delta_{x+}\Psi_N^n)}{S_{N+1/2}+S_{N-1/2}-S_{N-1/2}} + \delta_{x-}\Psi_N^n\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)(\delta_{x+}\Psi_N^n + \delta_{x-}\Psi_N^n),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)\left(\frac{1}{h}\left(\Psi_{N+1}^n - \Psi_N^n + \Psi_N^n - \Psi_{N-1}^n\right)\right),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(1-\frac{\epsilon_\mathrm{r}}{2}\right)2(\delta_{x\cdot}\Psi_N^n),\\
&= c^2(\delta_{t\cdot}\Psi_N^n)S_{N-1/2}\left(2-\epsilon_\mathrm{r}\right)(\delta_{x\cdot}\Psi_N^n).
\end{aligned} \tag{F.6}$$

The same can be done for $\mathfrak{b}_\mathrm{l}$ with $\epsilon_\mathrm{l} = S_{1/2}/\mu_{xx}S_0$ to get

$$\mathfrak{b}_\mathrm{l} = c^2(\delta_{t\cdot}\Psi_0^n)S_{1/2}\left(2-\epsilon_\mathrm{l}\right)(\delta_{x\cdot}\Psi_0^n). \tag{F.7}$$

F.6 Levine and Schwinger radiation model update equation

This section provides a derivation for the update equation in Eq. (5.58) and follows [62, Sec. 4.1.3, pp. 109–111]. Recalling system (5.56)

$$\bar{v} = \mu_{t+}v_{(1)} + \frac{1}{R_2}\mu_{t+}p_{(1)} + C_\mathrm{r}\delta_{t+}p_{(1)}, \tag{F.8a}$$

$$\bar{p} = L_\mathrm{r}\delta_{t+}v_{(1)}, \tag{F.8b}$$

$$\bar{p} = \left(1+\frac{R_1}{R_2}\right)\mu_{t+}p_{(1)} + R_1C_\mathrm{r}\delta_{t+}p_{(1)}, \tag{F.8c}$$

where $\bar{p}^{n+1/2}$ and $\bar{v}^{n+1/2}$ are related to the tube by (see Eq. (5.57))

$$\bar{p} = \mu_{t+}p_N^n, \tag{F.9}$$

$$\bar{S}_N\bar{v} = \mu_{x-}\left(S_{N+1/2}v_{N+1/2}^{n+1/2}\right), \tag{F.10}$$

one can start the derivation.

The radiation can be applied to the right boundary of the tube by evaluating the update equation of the pressure in Eq. (5.41a) at $l = N$

$$p_N^{n+1} = p_N^n - \frac{\rho_0 c\lambda}{\bar{S}_N}\left(S_{N+1/2}v_{N+1/2}^{n+1/2} - S_{N-1/2}v_{N-1/2}^{n+1/2}\right), \tag{F.11}$$

rewriting this to

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

and substituting Eq. (F.10) to get

$$p_N^{n+1} = p_N^n - \frac{2\rho_0 c\lambda}{\bar{S}_N}\left(\bar{S}_N\bar{v} - S_{N-1/2}v_{N-1/2}^{n+1/2}\right). \tag{F.12}$$

A definition for $\bar{v}$ can then be found by first expanding Eq. (F.8a) to

$$\bar{v} = \frac{1}{2}\left(v_{(1)}^{n+1} + v_{(1)}^n\right) + \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k}\right)p_{(1)}^{n+1} + \left(\frac{1}{2R_2} - \frac{C_\mathrm{r}}{k}\right)p_{(1)}^n, \tag{F.13}$$

after which it should be made solely dependent on the known values $v_{(1)}^n$, $p_{(1)}^n$ and p_N^n and the unknown p_N^{n+1} (as this can be obtained using Eq. (F.12)).

Equation (F.8b) can be expanded to

$$v_{(1)}^{n+1} = \frac{k}{L_\mathrm{r}}\bar{p} + v_{(1)}^n, \tag{F.14}$$

and Eq. (F.8c) to

$$\bar{p} = \left(1 + \frac{R_1}{R_2}\right)\mu_{t+}p_{(1)} + R_1C_\mathrm{r}\delta_{t+}p_{(1)},$$

$$\bar{p} = \frac{1}{2}\left(1 + \frac{R_1}{R_2}\right)\left(p_{(1)}^{n+1} + p_{(1)}^n\right) + \frac{R_1C_\mathrm{r}}{k}\left(p_{(1)}^{n+1} - p_{(1)}^n\right),$$

$$\left(\frac{1}{2} + \frac{R_1}{2R_2} + \frac{R_1C_\mathrm{r}}{k}\right)p_{(1)}^{n+1} = \bar{p} + \left(\frac{R_1C_\mathrm{r}}{k} - \frac{1}{2} - \frac{R_1}{2R_2}\right)p_{(1)}^n,$$

and finally solved for $p_{(1)}^{n+1}$ as

$$p_{(1)}^{n+1} = \underbrace{\left(\frac{2R_2k}{2R_1R_2C_\mathrm{r} + k(R_1+R_2)}\right)}_{\zeta_1}\bar{p} + \underbrace{\left(\frac{2R_1R_2C_\mathrm{r} - k(R_1+R_2)}{2R_1R_2C_\mathrm{r} + k(R_1+R_2)}\right)}_{\zeta_2} p_{(1)}^{n}. \quad \text{(F.15)}$$

Equations (F.14) and (F.15) can then be substituted into Eq. (F.13) and, using the definition of $\bar{p}$ from Eq. (F.9), yields

$$\bar{v} = \frac{1}{2}\left(\frac{k}{L_\mathrm{r}}(\mu_{t+}p_N^n) + 2v_{(1)}^n\right) + \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k}\right)\zeta_1\mu_{t+}p_N^n$$

$$+ \left(\frac{1}{2R_2} + \frac{C_\mathrm{r}}{k}\right)\zeta_2 p_{(1)}^n + \left(\frac{1}{2R_2} - \frac{C_\mathrm{r}}{k}\right)p_{(1)}^n,$$

$$\bar{v} = \underbrace{\left(\frac{k}{2L_\mathrm{r}} + \frac{\zeta_1}{2R_2} + \frac{C_\mathrm{r}\zeta_1}{k}\right)}_{\zeta_3}\mu_{t+}p_N^n + v_{(1)}^n + \underbrace{\left(\frac{\zeta_2+1}{2R_2} + \frac{C_\mathrm{r}\zeta_2 - C_\mathrm{r}}{k}\right)}_{\zeta_4} p_{(1)}^n. \quad \text{(F.16)}$$

Finally, substituting this definition for $\bar{v}$ into Eq. (F.12), yields

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

$$p_N^{n+1} = p_N^n - \rho_0 c\lambda\left(\zeta_3(p_N^{n+1}+p_N^n) + 2(v_{(1)}^n + \zeta_4 p_{(1)}^n) - \frac{2S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right),$$

and yields a definition for p_N^{n+1} based on known values of the system

$$p_N^{n+1} = \frac{1-\rho_0 c\lambda\zeta_3}{1+\rho_0 c\lambda\zeta_3}p_N^n - \frac{2\rho_0 c\lambda}{1+\rho_0 c\lambda\zeta_3}\left(v_{(1)}^n + \zeta_4 p_{(1)}^n - \frac{S_{N-1/2}v_{N-1/2}^{n+1/2}}{\bar{S}_N}\right), \quad \text{(F.17)}$$

which is Eq. (5.58). After p_N^{n+1} is calculated, $v_{(1)}^{n+1}$ and $p_{(1)}^{n+1}$ can be updated according to Eqs. (F.14) and (F.15), respectively.

F.7 Derivatives for Newton-Raphson for the elasto-plastic friction model

This section provides the derivatives for the Newton-Raphson iteration for the elasto-plastic friction model in Eq. (8.41).

Recalling the functions needed to compute the Newton-Raphson iteration for the elasto-plastic bow model in Section 8.5.1, being Eq. (8.38):

$$g_1(v^n, z^n) = \left(\frac{2}{k} + 2\sigma_0\right) v^n + \|J_l(x_\text{B}^n)\|_d^2 \frac{f(v^n, z^n)}{\rho A} + b^n = 0,$$

and Eq. (8.40)

$$g_2(v^n, z^n) = r^n - a^n = 0,$$

the derivatives needed to solve the Newton-Raphson iteration in Eq. (8.41)

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

can be shown to be

$$\begin{aligned}
\frac{\partial g_1}{\partial v} &= \frac{2}{k} + 2\sigma_0 + \frac{s_1\|J_l(x_\text{B}^n)\|_d^2}{\rho A}\frac{\partial r}{\partial v} + \frac{s_2\|J_l(x_\text{B}^n)\|_d^2}{\rho A},\\
\frac{\partial g_1}{\partial z} &= \frac{s_0\|J_l(x_\text{B}^n)\|_d^2}{\rho A} + \frac{s_1\|J_l(x_\text{B}^n)\|_d^2}{\rho A}\frac{\partial r}{\partial z},\\
\frac{\partial g_2}{\partial v} &= \frac{\partial r}{\partial v}\\
\frac{\partial g_2}{\partial z} &= \frac{\partial r}{\partial z} - \frac{2}{k}.
\end{aligned}$$

Recalling from Eq. (8.36) that

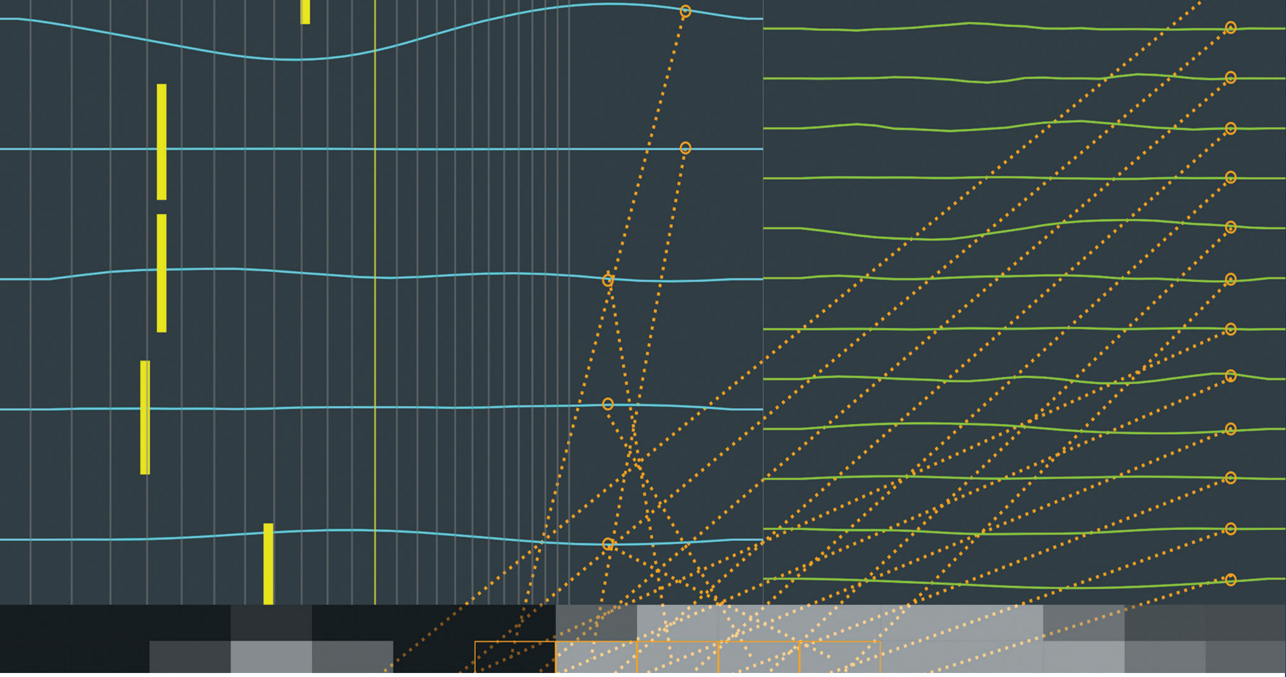
$$r^n = r(v^n, z^n) = v^n\left[1 - \alpha(v^n, z^n)\frac{z^n}{z_\text{ss}(v^n)}\right],$$

its derivatives can be computed as

$$\begin{aligned}
\frac{\partial r}{\partial v} &= 1 - z^n\left(\frac{(\alpha^n + \frac{\partial \alpha^n}{\partial v}v^n)z_\text{ss}^n - \frac{\partial z_\text{ss}^n}{\partial v}\alpha^n v^n}{(z_\text{ss}^n)^2}\right),\\
\frac{\partial r}{\partial z} &= -\frac{v^n}{z_\text{ss}^n}\left(\frac{\partial \alpha^n}{\partial z}z^n + \alpha^n\right),
\end{aligned}$$

with

$$\begin{aligned}
\frac{\partial \alpha^n}{\partial v} &= \text{sgn}(z_\text{ss}^n)\frac{\partial z_\text{ss}^n}{\partial v}\frac{z_\text{ba} - |z^n|}{(|z_\text{ss}^n| - z_\text{ba})^2}\frac{\pi}{2}\cos\big(\text{sgn}(z^n)\Phi\big),\\
\frac{\partial \alpha^n}{\partial z} &= \frac{\text{sgn}(z^n)\pi\cos\big(\text{sgn}(z^n)\Phi\big)}{2(|z_\text{ss}^n| - z_\text{ba})},\\
\frac{\partial z_\text{ss}^n}{\partial v} &= -\frac{2|v^n|}{v_\text{S}^2 s_0}(f_\text{S} - f_\text{C})e^{-(v^n/v_\text{S})^2}.
\end{aligned}$$



SUMMARY

Digital versions of musical instruments have been created for several decades, and for good reasons! 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...

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-976-3

AALBORG UNIVERSITY PRESS