
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

Thesis submitted: July 31, 2021

PhD Supervisor: Prof. Stefania Serafin
Aalborg University

Co-Supervisor (external): Prof. Stefan Bilbao
University of Edinburgh

PhD Committee: Assoc. Prof. Olga Timcenko
Aalborg University

Prof. Julius O. Smith III
Stanford University

Prof. Augusto Sarti
Politecnico di Milano

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

Department: Department of Architecture, Design and Media Technology

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

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

© Copyright: Silvin Willemsen

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 and interaction of the instruments 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 work 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

Abstract

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 og interaktion. 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 sårbar 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 modelleringssteknik, 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

Resumé

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 my 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, Luis Vieira, Simone Spagnol, Emil Høeg, Elisha Anne Teo, Camilla Jaller, 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 Chris-

tensen, 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 Willemse
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. Willemesen, 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. Willemesen, 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. Willemesen, 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. Willemesen, S. Serafin, S. Bilbao, and M. Duccheschi, “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. Willemesen, 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. Willemesen, 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

List of Abbreviations

Contents

Abstract	v
Resumé	vii
Preface	ix
Contents	xix
I Introduction	1
1 Physical Modelling of Musical Instruments	3
1.1 A brief history	4
1.2 Exciter-resonator approach	5
1.3 Physical modelling techniques	6
1.4 Applications of physical modelling	8
1.5 Project objectives and main contributions	10
1.6 Thesis outline	12
2 Introduction to Finite-Difference Time-Domain Methods	15
2.1 Differential equations	15
2.2 Discretisation using FDTD methods	18
2.3 The mass-spring system	26
2.4 The 1D wave equation	30
3 Analysis Techniques	43
3.1 Matrices in a FDTD context	43
3.2 Mathematical tools and product identities	46
3.3 Frequency domain analysis	51
3.4 Energy analysis	56
3.5 Modal analysis	67
3.6 Conclusion	70

II Resonators	71
4 The Stiff String	75
4.1 Continuous time	75
4.2 Discrete time	78
4.3 von Neumann analysis and stability condition	85
4.4 Energy analysis	86
4.5 Modal analysis	89
4.6 Implicit scheme	90
5 Acoustic Tubes	95
5.1 Webster's equation	95
5.2 First-order system	107
6 2D Systems	115
6.1 PDEs and FD schemes in 2D	115
6.2 The 2D wave equation	117
6.3 The thin plate	128
6.4 The stiff membrane	136
III Exciters	141
7 Physically-Inspired Excitations	145
7.1 Initial conditions	145
7.2 Time-varying excitations	150
8 The Bow	155
8.1 Brief history of bowed-string simulation	156
8.2 Interpolation and spreading operators	157
8.3 The Newton-Raphson method	160
8.4 Static friction models	162
8.5 Dynamic friction models	168
8.6 Discussion and conclusion	176
9 The Lip Reed	177
9.1 Mass-spring systems revisited: Damping	178
9.2 Continuous time	179
9.3 Discrete time	181
9.4 Energy analysis	184

IV Interactions	189
10 Collisions	193
10.1 The mass – rigid barrier collision	195
10.2 Mass-spring – string collision	200
10.3 Two-sided collision: A connection	205
11 Connections	207
11.1 Interpolation and spreading in 2D	207
11.2 Connected ideal strings	209
11.3 Rigid connection	211
11.4 Spring connection	216
11.5 String-plate connection	220
V Contributions	227
12 The Dynamic Grid	231
12.1 Background and motivation	231
12.2 Extended summary	232
12.3 Interpolation experiments	243
12.4 Examples of use cases	248
13 Real-Time Implementation and Control	249
13.1 Real-time FD schemes	250
13.2 Code structure	254
13.3 Hardware devices	257
14 Large Scale Modular Physical Models	259
14.1 Physical models	259
14.2 Implementation	260
14.3 Summary	261
15 The Tromba Marina	263
15.1 Summary	263
15.2 Physical model	264
15.3 Implementation details	269
16 The Trombone	273
16.1 Summary	273
16.2 Dynamic grid considerations	274
16.3 Lip-reed with collision	275

Contents

VI Conclusions and Perspectives	279
17 Conclusions and Perspectives	281
17.1 Summary	281
17.2 Perspectives and future work	282
References	297
VII Papers	299
A Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph	301
B Physical Models and Real-Time Control with the Sensel Morph	303
C Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes	305
D Real-time Implementation of a Physical Model of the Tromba Marina	307
E Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling	309
F DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study	311
G Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations	313
H A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes	315
VIII Appendix	317
A Paper Errata	319
B Matrices	321
B.1 Operations	321
B.2 Matrix inverse	323
B.3 Systems of linear equations	324
B.4 Eigenvalue problems	325

Contents

C Code Snippets	327
C.1 Mass-spring system (Section 2.3)	327
C.2 1D wave equation (Section 2.4)	328
C.3 2D wave equation (Section 6.2)	329
D Intuition for the Damping Terms in the Stiff String PDE	331
E Considerations in real-time FD schemes	333
F Derivations	337
F.1 von Neumann analysis damped stiff string (Section 4.3)	337
F.2 von Neumann analysis implicit damped stiff string (Section 4.6.1)	339
F.3 Webster's update equation (Eq. (5.9))	340
F.4 Boundary terms Webster's equation (Eq. (5.31))	341
F.5 Levine and Schwinger radiation model update (Eq. (5.59)) . . .	342
F.6 Derivatives for Newton-Raphson for the elasto-plastic friction model (Eq. (8.41))	344

Contents

Todo list

■ Find double words: using the following commented 'regular expression':	3
■ set tocdepth back after doing capitalisation and dashes	3
■ FULL DOC SWEEP: this work clear about referencing to the PhD project	5
■ Maybe add table of pros and cons OR when to use which	7
■ think about references to this project/work in this chapter that should only be focused on background	17
■ FULL DOC SWEEP: check capitalisation of headings throughout document (DOING NOW)	19
■ many figures for shift and FD operators	20
■ FULL DOC SWEEP: check centred instead of centered	21
■ figure here visualising operators (with reference to grid figure)	21
■ in energy analysis, interleaved grids, etc.	22
■ move section up (stefan's comment)	27
■ FULL DOC SWEEP: check hyphen in titles (DOING)	30
■ unit?	30
■ different wording in caption	31
■ add why this is relevant?	32
■ check whether still correct	33
■ FULL DOC SWEEP: check straightforward or straight-forward	33
■ <i>w</i> does not have to be floored	36
■ look at this	36
■ figure?	41
■ is this how you explain it?	44
■ check with Stefan	47
■ check	52
■ check if the sum should indeed go until $n - 1$ and why	59
■ ask stefania about these figure captions being identical	63
■ some citation here	67
■ check with Stefan	68
■ look at wording	107

Contents

■ stencil?	109
■ temperature dependent parameters somewhere?	113
■ table	113
■ FULL DOC SWEEP: check what equations have numbers when performing energy analysis (and stability for that matter)	125
■ check hyphens after 'frequency'	129
■ FULL DOC SWEEP: check figure alignment	132
■ FULL DOC SWEEP: check for SWcomments	135
■ look at this	136
■ take an extra look at these intros	143
■ w does not have to be floored...	147
■ look at this	149
■ pulse train figure placement	152
■ FULL DOC SWEEP: check figure centering	163
■ check time varying x_B^n	165
■ look at implementation	168
■ figure layout	186
■ check with stefan if this is a correct notation for this	216
■ shortly describe	281
■ check whether all references are used	285
■ check whether to sort references or not	297

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

Find double words: using the following commented 'regular expression':

set tocdepth back after doing capitalisation and dashes

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 simulating¹ 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 1961². It uses what later got known as the Kelly-Lochbaum vocal-tract model to generate a voice and was published the year thereafter [7].

The very first musical instrument simulations were based on discretisation of differential equations using *finite-difference time-domain* (FDTD) methods. These were carried out around 1970 by Hiller and Ruiz [8, 9, 10] and applied to the wave equation to simulate string sounds. The sound generation, however, was far from real-time and it took several minutes to generate only 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

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

²<http://ccrma.stanford.edu/~jos/wav/daisy-klm.wav>

1.2. Exciter-resonator approach

[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 work. Examples of 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 bi-directional. 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 work.

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

FULL DOC
SWEEP: this
work clear
about refer-
encing to the
PhD project

³The 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 all based on the same types of equations of motion. Many of these equations and other knowledge currently available on the physics of musical instruments have been collected by Fletcher and Rossing in [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 methods (FDTD) aim to solve PDEs by approximating them with difference equations, discretising a continuous system into grid-points in space and time. In a musical context, this technique was first used for the case of string vibration by Hiller and Ruiz in [8, 9, 10] and later by Chaigne in [25, 26]. Bilbao extensively describes this method in [21, 23]. 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 pre-

1.3. Physical modelling techniques

sented 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].

Maybe add table of pros and cons OR when to use which

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 work.

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 – 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-

⁴I learned the hard way that one should always implement a limiter when working with real-time physical models to avoid dangerously loud sounds.

quality simulations of musical instruments based on FDTD methods in real time, will not be an issue.

1.4 Applications of physical modelling

It might not be clear why one would go through all the hassle of modelling musical instruments. 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, etc. 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.

A drawback of physical models is that, in order to generate a realistic sound, a highly accurate physical description of the original instrument is

⁵This 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].

1.4. Applications of physical modelling

needed. Apart from (potentially) taking a lot of time to develop this model and tuning its parameters, the eventual implementation will be (much) more computationally expensive than if samples were to be used. Generally, the more accurate the model is, and thus the more true the sound is to the original, the higher the computational cost becomes.

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

1.4.2 Resurrect old or rare instruments

Many instruments exist that are too old, too rare, or too valuable to be played. Some live behind museum glass only to be looked at by visitors, never to be played again. In these cases, it might even be hard to record samples of the musical instrument. If, however, the physics (geometry and material properties) of the instrument are available, a physical model of the instrument could be created, bringing its sound back to 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 work 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 (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 is structured using several parts, which in turn are divided into chapters. 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 novelty. Much effort has been put in explaining the existing methods and models from the literature used in this project, 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 work (and these parts in particular) to be 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.

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 project objectives and contributions to the field are detailed. Chapter 2 provides a thorough introduction to finite-difference time-domain 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. 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

1.6. Thesis outline

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 extended with implementation details. Finally, paper [H] is summarised in Chapter 16, extended with details on the implementation.

Part VI: Conclusions and Perspectives

Chapter 17 concludes the thesis, and puts the contributions into context of the physical modelling field. Future perspectives 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 **Part VIII: Appendix** contains additional information on matrices, code examples and derivations.

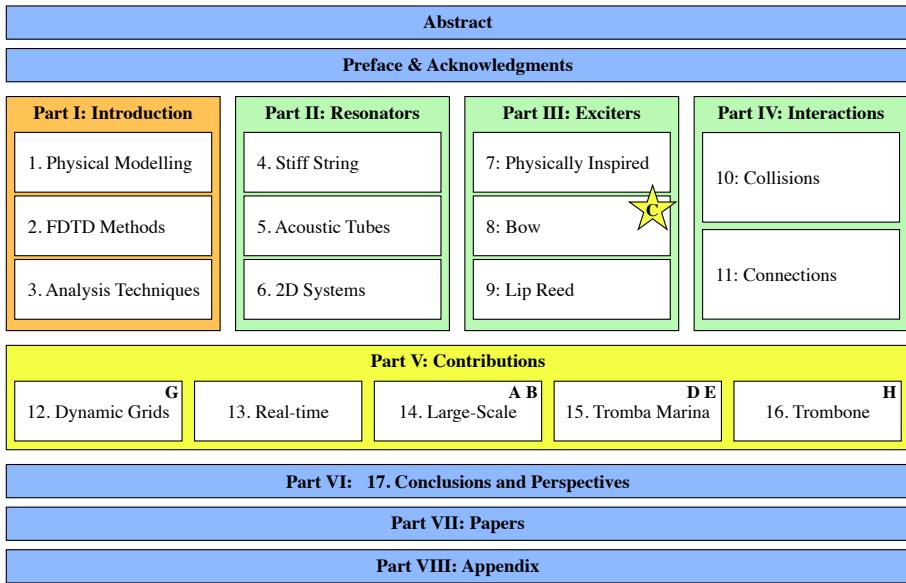


Fig. 1.1: The outline of this thesis. The contributions made throughout the PhD project are marked in yellow. Most are collected in Part V, though the novel work done on the bow will already appear in Chapter 8. The parts marked in green, describe the physical models on which the contributions are based. 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.

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 will be used [49]. Unless denoted otherwise, the theory presented in this chapter and the notation have been taken from [21].

2.1 Differential equations

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

A characteristic feature of these equations is that, rather than an absolute value or *state* of a system, such as displacement from the equilibrium of a string, or the pressure in a tube, the time derivative of its state – its velocity

– or the second time derivative – its acceleration – is described. From this, the absolute state of the system can then be computed. This state is usually described by the variable u which is always a function of time, i.e., $u = u(t)$. If the system is distributed in space, u also becomes a function of space, i.e., $u = u(x, t)$, or with two spatial dimensions, $u = u(x, y, t)$, etc. Though this work only describes systems of up to two spatial dimensions, one can easily extend to three dimensions [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

$$\frac{d^2u}{dt^2} \quad (\text{Leibniz's notation}),$$

$$\ddot{u} \quad (\text{Newton's notation}),$$

$$D_t^2u \quad (\text{Euler's notation}).$$

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 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

$$\frac{\partial^2u}{\partial t^2} \quad (\text{Leibniz's notation}),$$

$$u_{tt} \quad (\text{subscript notation}),$$

$$\partial_t^2u \quad (\text{Euler's notation}),$$

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 three translational DoF. Due to these additional DoF, these networks do capture the aforementioned effects, but greatly increase the computational complexity of the models.

think about
references to
this project/-
work in this
chapter that
should only
be focused on
background

¹In 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 $t \geq 0$, meaning that the system is defined for non-negative time.

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

2.2 Discretisation using FDTD methods

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

approximation. Conditions for stability can be mathematically derived and will be introduced in Section 3.3.

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

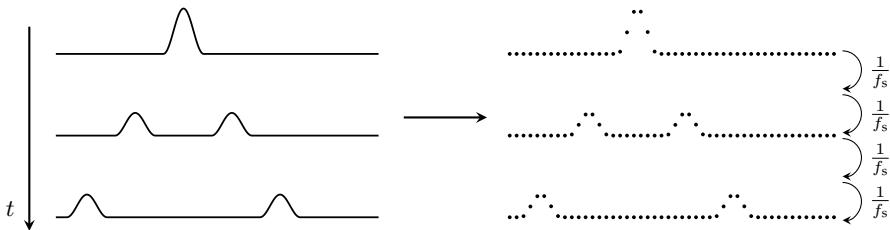


Fig. 2.1: A continuous PDE is discretised to a FD scheme. In a PDE, time passes continuously, whereas for a FD scheme, time passes in finite increments with a duration of the reciprocal of the sample rate f_s .

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

FULL DOC
SWEEP: check capitalisation of headings throughout document (DOING NOW)

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 end points. The spatial range of interest then becomes $l \in \{0, \dots, N\}$ and the

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

total number of grid points is $N + 1$, which is one more than the number of intervals.

To summarise, for a 1D system

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

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

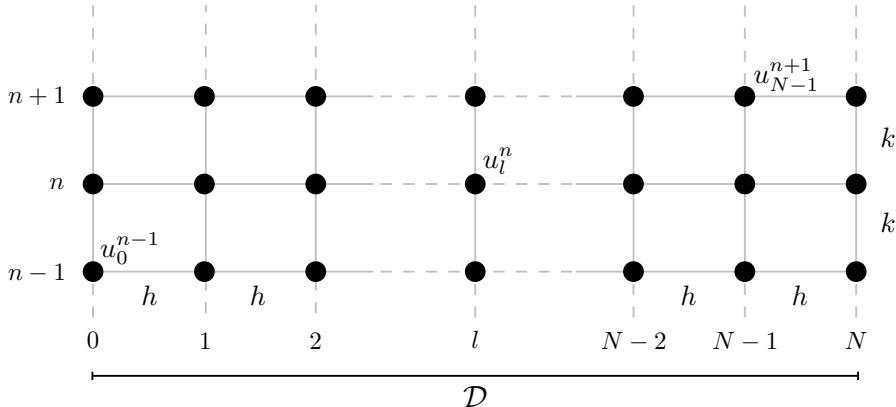


Fig. 2.2: The spatio-temporal grid that appears when a 1D system $u(x, t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n . The spatial domain \mathcal{D} is divided into N intervals of length h and spatial range of interest $l = \{0, \dots, N\}$. Time is subdivided into time steps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

2.2.2 Finite-difference operators

Now that the state variable has a discrete counterpart, it leaves the derivatives to be discretised, or approximated. We start by introducing *shift operators* that 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

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

Similarly, forward and backward shifts in space are

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

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,

many figures
for shift and
FD operators

2.2. Discretisation using FDTD methods

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

$$\partial_t \approx \begin{cases} \delta_{t+} \triangleq \frac{1}{k} (e_{t+} - 1), \\ \delta_{t-} \triangleq \frac{1}{k} (1 - e_{t-}), \\ \delta_{t.} \triangleq \frac{1}{2k} (e_{t+} - e_{t-}), \end{cases} \quad (2.3a)$$

$$\partial_t \approx \begin{cases} \delta_{t+} \triangleq \frac{1}{k} (e_{t+} - 1), \\ \delta_{t-} \triangleq \frac{1}{k} (1 - e_{t-}), \\ \delta_{t.} \triangleq \frac{1}{2k} (e_{t+} - e_{t-}), \end{cases} \quad (2.3b)$$

$$\partial_t \approx \begin{cases} \delta_{t+} \triangleq \frac{1}{k} (e_{t+} - 1), \\ \delta_{t-} \triangleq \frac{1}{k} (1 - e_{t-}), \\ \delta_{t.} \triangleq \frac{1}{2k} (e_{t+} - e_{t-}), \end{cases} \quad (2.3c)$$

FULL DOC
SWEEP: check
centred instead
of centered

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

$$\partial_t u \approx \begin{cases} \delta_{t+} u_l^n = \frac{1}{k} (u_l^{n+1} - u_l^n), \\ \delta_{t-} u_l^n = \frac{1}{k} (u_l^n - u_l^{n-1}), \\ \delta_{t.} u_l^n = \frac{1}{2k} (u_l^{n+1} - u_l^{n-1}), \end{cases} \quad (2.4a)$$

$$\partial_t u \approx \begin{cases} \delta_{t+} u_l^n = \frac{1}{k} (u_l^{n+1} - u_l^n), \\ \delta_{t-} u_l^n = \frac{1}{k} (u_l^n - u_l^{n-1}), \\ \delta_{t.} u_l^n = \frac{1}{2k} (u_l^{n+1} - u_l^{n-1}), \end{cases} \quad (2.4b)$$

$$\partial_t u \approx \begin{cases} \delta_{t+} u_l^n = \frac{1}{k} (u_l^{n+1} - u_l^n), \\ \delta_{t-} u_l^n = \frac{1}{k} (u_l^n - u_l^{n-1}), \\ \delta_{t.} u_l^n = \frac{1}{2k} (u_l^{n+1} - u_l^{n-1}), \end{cases} \quad (2.4c)$$

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

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

$$\partial_x \approx \begin{cases} \delta_{x+} \triangleq \frac{1}{h} (e_{x+} - 1), \\ \delta_{x-} \triangleq \frac{1}{h} (1 - e_{x-}), \\ \delta_{x.} \triangleq \frac{1}{2h} (e_{x+} - e_{x-}), \end{cases} \quad (2.5a)$$

$$\partial_x \approx \begin{cases} \delta_{x+} \triangleq \frac{1}{h} (e_{x+} - 1), \\ \delta_{x-} \triangleq \frac{1}{h} (1 - e_{x-}), \\ \delta_{x.} \triangleq \frac{1}{2h} (e_{x+} - e_{x-}), \end{cases} \quad (2.5b)$$

$$\partial_x \approx \begin{cases} \delta_{x+} \triangleq \frac{1}{h} (e_{x+} - 1), \\ \delta_{x-} \triangleq \frac{1}{h} (1 - e_{x-}), \\ \delta_{x.} \triangleq \frac{1}{2h} (e_{x+} - e_{x-}), \end{cases} \quad (2.5c)$$

figure here vi-
sualising op-
erators (with
reference to
grid figure)

and when applied to u_l^n are

$$\partial_x u \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h} (u_{l+1}^n - u_l^n), \\ \delta_{x-} u_l^n = \frac{1}{h} (u_l^n - u_{l-1}^n), \\ \delta_{x.} u_l^n = \frac{1}{2h} (u_{l+1}^n - u_{l-1}^n). \end{cases} \quad (2.6a)$$

$$\partial_x u \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h} (u_{l+1}^n - u_l^n), \\ \delta_{x-} u_l^n = \frac{1}{h} (u_l^n - u_{l-1}^n), \\ \delta_{x.} u_l^n = \frac{1}{2h} (u_{l+1}^n - u_{l-1}^n). \end{cases} \quad (2.6b)$$

$$\partial_x u \approx \begin{cases} \delta_{x+} u_l^n = \frac{1}{h} (u_{l+1}^n - u_l^n), \\ \delta_{x-} u_l^n = \frac{1}{h} (u_l^n - u_{l-1}^n), \\ \delta_{x.} u_l^n = \frac{1}{2h} (u_{l+1}^n - u_{l-1}^n). \end{cases} \quad (2.6c)$$

Higher-order differences can be approximated through a composition of first-order difference operators where their definitions are multiplied.³ The second-order difference in time may be approximated using

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

where “2” is the identity operator applied twice. This can be done similarly

³Alternatively, 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.

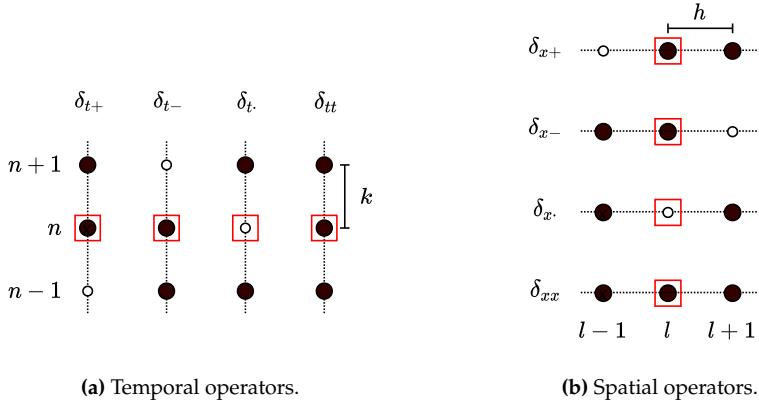


Fig. 2.3: The stencils of various FD operators applied to the grid point highlighted with a red square. Black grid points are used in the calculation, and white grid points are not. The averaging operators follow the same pattern.

for the second-order difference in space

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

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

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

$$1 \approx \begin{cases} \mu_{t+} \triangleq \frac{1}{2} (e_{t+} + 1), \\ \mu_{t-} \triangleq \frac{1}{2} (1 + e_{t-}), \\ \mu_{t.} \triangleq \frac{1}{2} (e_{t+} + e_{t-}). \end{cases} \quad (2.9a)$$

$$1 \approx \begin{cases} \mu_{t+} \triangleq \frac{1}{2} (e_{t+} + 1), \\ \mu_{t-} \triangleq \frac{1}{2} (1 + e_{t-}), \\ \mu_{t.} \triangleq \frac{1}{2} (e_{t+} + e_{t-}). \end{cases} \quad (2.9b)$$

$$1 \approx \begin{cases} \mu_{t+} \triangleq \frac{1}{2} (e_{t+} + 1), \\ \mu_{t-} \triangleq \frac{1}{2} (1 + e_{t-}), \\ \mu_{t.} \triangleq \frac{1}{2} (e_{t+} + e_{t-}). \end{cases} \quad (2.9c)$$

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

$$u_l^n \approx \begin{cases} \mu_{t+} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^n), \\ \mu_{t-} u_l^n = \frac{1}{2} (u_l^n + u_l^{n-1}), \\ \mu_{t.} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^{n-1}). \end{cases} \quad (2.10a)$$

$$u_l^n \approx \begin{cases} \mu_{t+} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^n), \\ \mu_{t-} u_l^n = \frac{1}{2} (u_l^n + u_l^{n-1}), \\ \mu_{t.} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^{n-1}). \end{cases} \quad (2.10b)$$

$$u_l^n \approx \begin{cases} \mu_{t+} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^n), \\ \mu_{t-} u_l^n = \frac{1}{2} (u_l^n + u_l^{n-1}), \\ \mu_{t.} u_l^n = \frac{1}{2} (u_l^{n+1} + u_l^{n-1}). \end{cases} \quad (2.10c)$$

in energy analysis,
interleaved grids,
etc.

2.2. Discretisation using FDTD methods

Similarly, spatial averaging operators are

$$1 \approx \begin{cases} \mu_{x+} \triangleq \frac{1}{2} (e_{x+} + 1), \\ \mu_{x-} \triangleq \frac{1}{2} (1 + e_{x-}), \end{cases} \quad (2.11a)$$

$$\mu_x \triangleq \frac{1}{2} (e_{x+} + e_{x-}), \quad (2.11b)$$

$$1 \approx \begin{cases} \mu_{x+} u_l^n = \frac{1}{2} (u_{l+1}^n + u_l^n), \\ \mu_{x-} u_l^n = \frac{1}{2} (u_l^n + u_{l-1}^n), \\ \mu_x u_l^n = \frac{1}{2} (u_{l+1}^n + u_{l-1}^n). \end{cases} \quad (2.12a)$$

$$u_l^n \approx \begin{cases} \mu_{x+} u_l^n = \frac{1}{2} (u_{l+1}^n + u_l^n), \\ \mu_{x-} u_l^n = \frac{1}{2} (u_l^n + u_{l-1}^n), \\ \mu_x u_l^n = \frac{1}{2} (u_{l+1}^n + u_{l-1}^n). \end{cases} \quad (2.12b)$$

$$u_l^n \approx \begin{cases} \mu_{x+} u_l^n = \frac{1}{2} (u_{l+1}^n + u_l^n), \\ \mu_{x-} u_l^n = \frac{1}{2} (u_l^n + u_{l-1}^n), \\ \mu_x u_l^n = \frac{1}{2} (u_{l+1}^n + u_{l-1}^n). \end{cases} \quad (2.12c)$$

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

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

and

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

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

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

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

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

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

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

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

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

and using Eq. (2.16) can be written to

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

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

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

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

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

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

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

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

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

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

indicating that to find the order of accuracy for this operator, both Eqs. (2.17)

2.2. Discretisation using FDTD methods

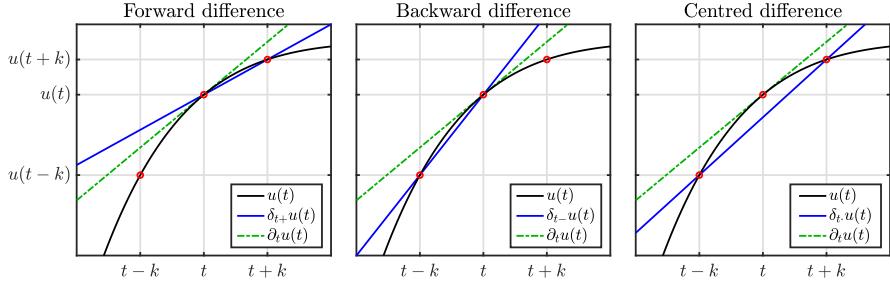


Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. The centred difference operator much more closely approximates the derivative, or the slope, of u at t when compared to the forward and backward difference operators.

and (2.20) are needed. Subtracting these and substituting their definitions yields

$$\begin{aligned} u(t+k) - u(t-k) &= 2k\dot{u} - \frac{2k^3}{6}\ddot{u} + 2\mathcal{O}(k^5), \\ \frac{u(t+k) - u(t-k)}{2k} &= \dot{u} + \mathcal{O}(k^2), \\ \delta_t u(t) &= \dot{u} + \mathcal{O}(k^2), \end{aligned} \quad (2.23)$$

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

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

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

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

The accuracy of averaging operators can be found in the same way and

follow a similar pattern.

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

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

$$\delta_{tt} = \frac{2}{k} (\delta_{t.} - \delta_{t-}), \quad (2.27a)$$

$$\delta_{t.} = \delta_{t+}\mu_{t-} = \delta_{t-}\mu_{t+}, \quad (2.27b)$$

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

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

$$\mu_{t.} = k\delta_{t.} + e_{t-}. \quad (2.27e)$$

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 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

$$M\ddot{u} = -Ku, \quad (2.28)$$

⁴The \pm operators in a single equation are either all ‘+’ or all ‘-’ and are used for a more compact notation.

2.3. The mass-spring system

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. Equation (2.28) can be written as

$$\ddot{u} = -\omega_0^2 u, \quad (2.29)$$

with angular frequency (in rad/s)

$$\omega_0 = \sqrt{K/M}. \quad (2.30)$$

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

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

Intuition

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

move section up (stefan's comment)

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

$$F = M\ddot{u}. \quad (2.31)$$

This equation in isolation can be used to, for example, calculate the force

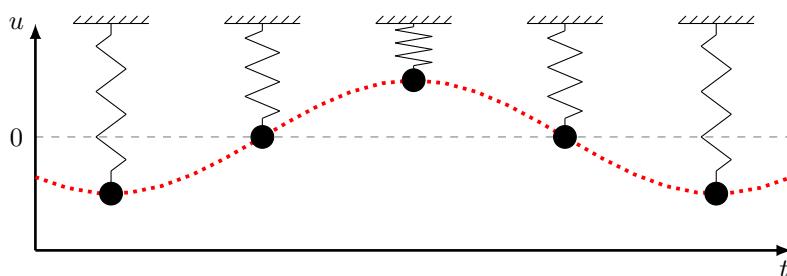


Fig. 2.5: Mass-spring system over time. The system follows a harmonic (sinusoidal) motion.

necessary to accelerate a mass of M kg to \ddot{u} m/s². Next, the force generated by the spring follows Hooke's law:

$$F = -Ku, \quad (2.32)$$

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 at 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

$$u(t) \approx u^n, \quad (2.33)$$

with time $t = nk$, time step $k = 1/f_s$, 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:

$$M\delta_{tt}u^n = -Ku^n, \quad (2.34)$$

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

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

2.3. The mass-spring system

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

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

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

2.3.3 Implementation and output

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

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

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

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

$$\begin{aligned} \frac{1}{k} (u^1 - u^0) &= 0, \\ \xleftarrow{u^0=1} u^1 - 1 &= 0, \\ u^1 &= 1. \end{aligned}$$

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

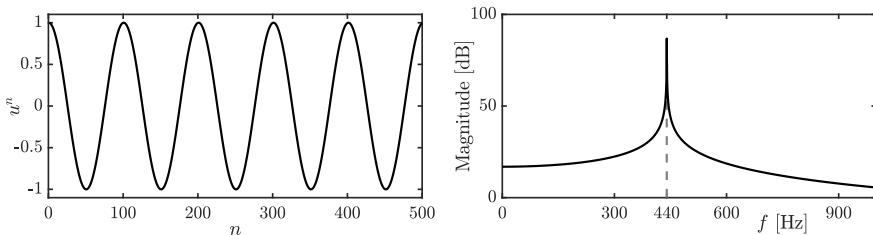


Fig. 2.6: The time domain and frequency domain output of a mass-spring system with $f_0 = 440$ Hz.

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

$$k < 2\sqrt{\frac{M}{K}}, \quad (2.37)$$

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

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

2.4 The 1D wave equation

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

2.4.1 Continuous time

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

$$\partial_t^2 u = c^2 \partial_x^2 u, \quad (2.38)$$

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/m³) and cross-sectional area A (in m²). 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.

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

FULL DOC
SWEEP: check
hyphen in ti-
tles (DOING)

unit?

2.4. The 1D wave equation

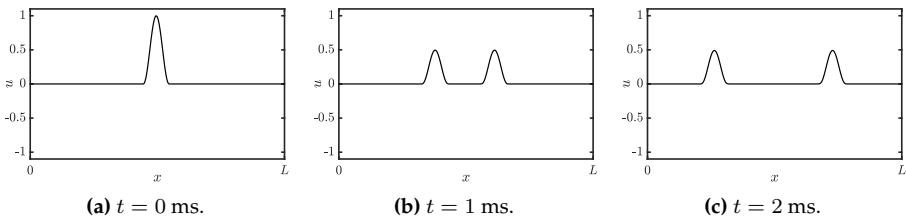


Fig. 2.7: Wave propagation in the 1D wave equation in Eq. (2.38) with $c \approx 127 \text{ m/s.}$

location (scaled by a constant c^2). In the case that u describes the transverse displacement of an ideal string, this second-order derivative denotes the *curvature* of this string. As c^2 is always positive, the sign (or direction) of the acceleration is fully determined by the sign of the curvature. In other words, a ‘positive’ curvature at location x along the ideal string yields a ‘positive’ or upwards acceleration at that same location.

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

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

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 ρAdx 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.

different wording in caption

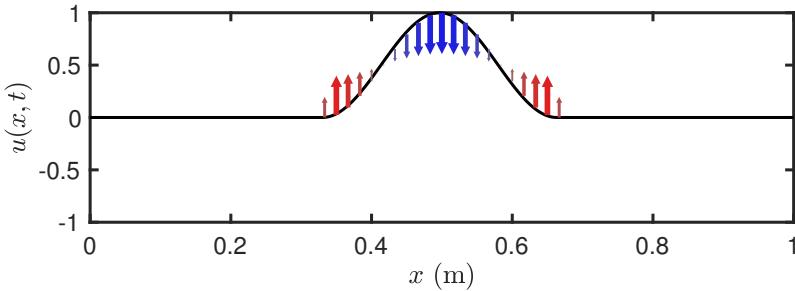


Fig. 2.8: The forces acting on the 1D wave equation described by $u(x, t)$ due to curvature. The arrows indicate the direction and magnitude of the force along the system, and consequently the acceleration through Eq. (2.38).

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

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

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

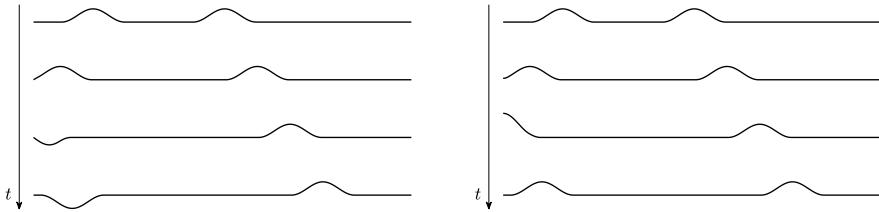
The Dirichlet boundary condition says that at the end points of the system, the state is 0 at all times. The Neumann condition on the other hand, says that rather the slope of these points needs to be 0, but that the end points are free to move transversely. In the former case, incoming waves invert 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

$$f_0 = \frac{c}{2L}. \quad (2.40)$$

Scaling

As this work follows much of 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.



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

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

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

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

$$\partial_t^2 u = \gamma^2 \partial_{x'x'} u, \quad (2.41)$$

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 here.

check whether
still correct

2.4.2 Discrete time

Coming back to the PDE presented in Eq. (2.38), we continue 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 $l \in \{0, \dots, N\}$ 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

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n. \quad (2.42)$$

FULL DOC
SWEEP: check
straightforward
or straight-
forward

Other schemes exist (see e.g. [21]), but are excluded as they have not been used in this work. Expanding the operators using the definitions given in Section 2.2.2 yields

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

and solving for u_l^{n+1} yields

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

Here,

$$\lambda = \frac{ck}{h} \quad (2.45)$$

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-Lowy or *CFL condition* for short [55]

$$\lambda \leq 1, \quad (2.46)$$

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 :

$$h \geq ck, \quad (2.47)$$

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} .

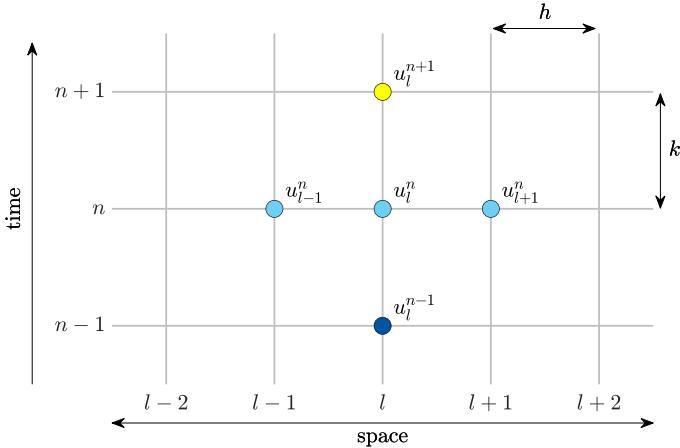


Fig. 2.10: The stencil, or region of operation, for the FD scheme in (2.42). The time steps of the various grid points are colour-coded by yellow ($n + 1$), light blue (n) and dark blue ($n - 1$).

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

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

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

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 $l \in \{1, \dots, N - 1\}$ 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 $l \in \{0, \dots, N\}$ 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:

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

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

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

and

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

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.

w does not have to be floored look at this The discrete raised cosine can be parametrised by its center location l_0 and width w (in' grid spacings') from which the start index l_s and end index l_e can be calculated, according to

$$l_s = l_0 - \lfloor w/2 \rfloor \quad \text{and} \quad l_e = l_0 + \lfloor w/2 \rfloor, \quad (2.51)$$

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

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

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).

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 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.⁵

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,

⁵Modes 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].

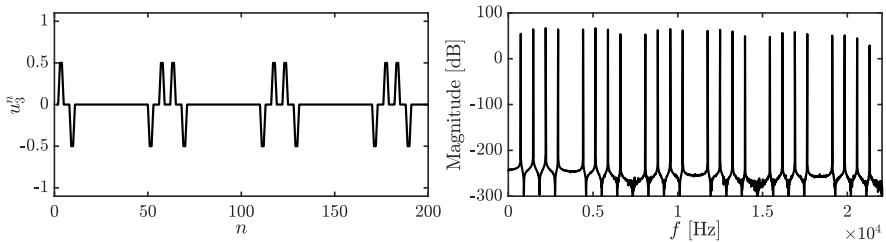


Fig. 2.11: The time-domain and frequency domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_s = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_0 = 0.2N$) and $w = 4$ and the output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

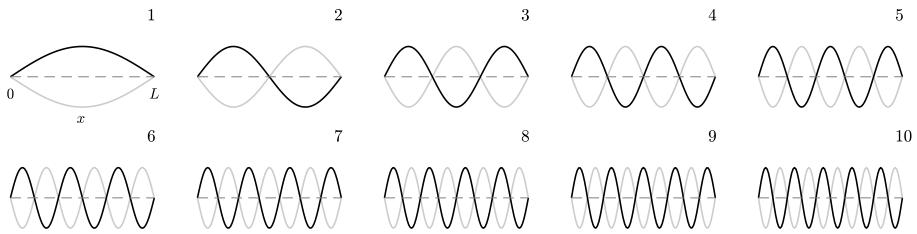


Fig. 2.12: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions defined for $x \in [0, L]$ (only shown for mode 1). The modes are normalised to have the same amplitude and vibrate at their respective modal frequencies with the extremes indicated by the black and the grey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

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: $f_m = m f_0$ for mode number $m \in \{1, \dots, N - 1\}$ for Dirichlet 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 2^{nd} modal frequency will be attenuated as there is a node present in the corresponding modal shape.

The output would then only contain odd-numbered modes.

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.⁶

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 (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:

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

⁶Imagine 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.

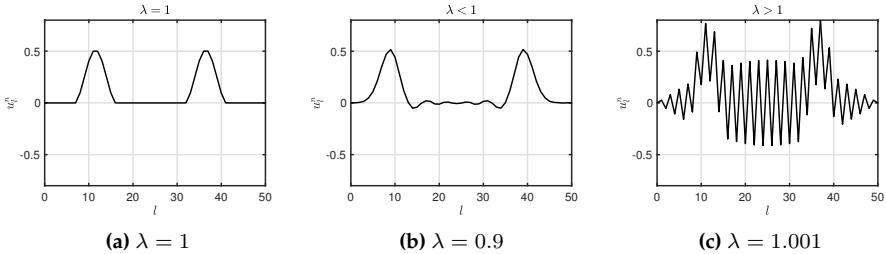


Fig. 2.13: Grid function u_i^n visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (2.46) is not satisfied and the system is unstable.

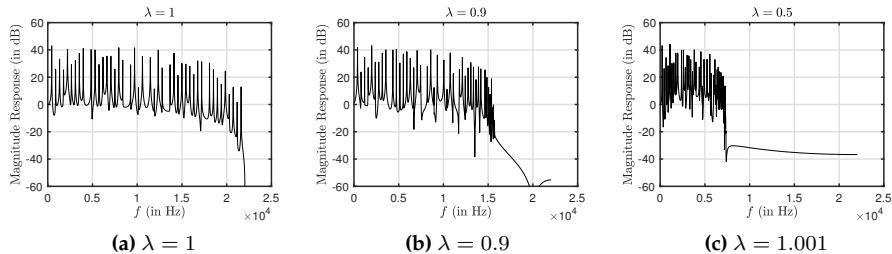


Fig. 2.14: Frequency spectra of the simulation output. The Courant number is set to (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$. One can observe that for lower values of λ the bandwidth of the output decreases drastically.

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

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

desired: $f_0 \approx 760.34$ Hz. If h is recalculated based on N , then L and f_0 will be unchanged, and the system will have the correct fundamental frequency. The Courant number $\lambda \approx 0.986$ is still very close to satisfying the condition in Eq. (2.46), and the decrease in quality will be perceptually irrelevant – or at the very least, less perceptually relevant than the change in f_0 if h is not recalculated.

Intuition

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

In a FD implementation of the 1D wave equation, grid points can only affect their neighbours as seen in update equation (2.44). Using the values in Table 2.1 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 (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.

figure?

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.

Chapter 3

Analysis Techniques

This chapter provides some useful techniques to analyse FD schemes. Techniques to analyse PDEs also exist, but the focus here is of a practical nature and will especially revolve around the discrete schemes. This chapter can be seen as a ‘tutorial’ on how to use these techniques. Starting off with some necessary theory on matrices 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-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} :

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \quad (3.1)$$

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

Expanding this operation results in

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{bmatrix}}_A \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_u = \underbrace{\begin{bmatrix} a_{00}u_0 + a_{01}u_1 + \dots + a_{0N}u_N \\ a_{10}u_0 + a_{11}u_1 + \dots + a_{1N}u_N \\ \vdots \\ a_{N0}u_0 + a_{N1}u_1 + \dots + a_{NN}u_N \end{bmatrix}}_w \quad (3.2)$$

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

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

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

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:

is this how you
explain it?

3.1. Matrices in a FDTD context

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

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

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:

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

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 $l \in \{1, \dots, N-1\}$ can then be stored in a column vector according to $\mathbf{u}^n = [u_1^n, \dots, u_{N-1}^n]^T$. Furthermore, $(N-1) \times (N-1)$ matrix

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\langle f_l^n, g_l^n \rangle'_d = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{h}{2} f_0^n g_0^n + \frac{h}{2} f_N^n g_N^n, \quad (3.9)$$

and the more general weighted inner product

$$\langle f_l^n, g_l^n \rangle_d^{\epsilon_l, \epsilon_r} = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{\epsilon_l}{2} h f_0^n g_0^n + \frac{\epsilon_r}{2} h f_N^n g_N^n, \quad (3.10)$$

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

check with Stefan

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

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

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

$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_d + f_{N+1}^n g_N^n - f_0^n g_{-1}^n, \quad (3.12a)$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_d + f_N^n g_{N+1}^n - f_{-1}^n g_0^n. \quad (3.12b)$$

A derivation of Eq. (3.12a) is given below. 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 $\underline{d} = \{0, \dots, N-1\}$, $\bar{d} = \{1, \dots, N\}$ and $\bar{\underline{d}} = \{1, \dots, N-1\}$. The following identities can be shown to hold

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_{\underline{d}} + f_N^n g_N^n - f_0^n g_{-1}^n, \quad (3.13a)$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\bar{d}} + f_N^n g_{N+1}^n - f_0^n g_0^n, \quad (3.13b)$$

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

$$\langle f_l^n, \delta_{x-} g_l^n \rangle'_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_{\underline{d}} + f_N^n \mu_{x-} g_N^n - f_0^n \mu_{x-} g_0^n, \quad (3.14a)$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle'_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\bar{d}} + f_N^n \mu_{x+} g_{N+1}^n - f_0^n \mu_{x+} g_0^n, \quad (3.14b)$$

3.2. Mathematical tools and product identities

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

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

$$\begin{aligned} \langle f_l^n, \delta_{x+} g_l^n \rangle_d^{\epsilon_l, \epsilon_r} &= -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\bar{d}} + f_N^n g_N^n - f_0^n g_1^n \\ &\quad + \frac{\epsilon_r}{2} f_N^n (g_{N+1}^n - g_N^n) + \frac{\epsilon_l}{2} f_0^n (g_1^n - g_0^n). \end{aligned} \quad (3.15b)$$

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

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

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

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

$$\langle f, \delta_{xx} g \rangle'_d = \langle \delta_{xx} f, g \rangle'_d + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0. \quad (3.16c)$$

Derivations

To see why the above identities 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, $d = \{0, \dots, N\}$ 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

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

As $N = 2$, the result is identical to Eq. (3.12a).

Similarly, identity (3.13a) can be proven to hold:

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

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 this section.

3.2.3 Product identities

Some useful identities used in this work are

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

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

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

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

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

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

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

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

The same rule applies when the backward operator $\delta_{t-}(u_l^n w_l^n)$ or centred operator $\delta_t.(u_l^n w_l^n)$ is used. In that case, the forward operators δ_{t+} and μ_{t+} in

3.3. Frequency domain analysis

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]

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

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]

$$u_l^n \xrightarrow{\mathcal{A}} z^n e^{jl\beta h} \quad (3.20)$$

where “ $\xrightarrow{\mathcal{A}}$ ” 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]:

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

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

Take for example

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

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

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

3.3. Frequency domain analysis

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

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

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

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

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

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)

$$|z| \leq 1. \quad (3.24)$$

In the FDTD context, the frequency domain representation of a FD scheme results in a *characteristic equation* – which is usually a second-order polynomial – in z and needs to satisfy condition (3.24) for all wave numbers β . It can be shown that for a polynomial of the form

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

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

$$|a^{(1)}| - 1 \leq a^{(2)} \leq 1. \quad (3.26)$$

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

$$|a^{(1)}| \leq 2, \quad (3.27)$$

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 (3.20) with $l = 0$. Using Table 3.1 and Eqs. (3.22) as a reference and substituting the definitions yields

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

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

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

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

$$k < 2\sqrt{\frac{M}{K}}. \quad (3.29)$$

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

$$k < \frac{2}{\omega_0}. \quad (3.30)$$

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.

3.3. Frequency domain 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):

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

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

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

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

$$h \geq ck, \quad (3.33)$$

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) sys-

tems. 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 FDTD methods. First of all, from a practical point of view, it is essential for debugging implementations of FD schemes. Especially when trying to model more complex systems, programming errors are unavoidable, and energy analysis can be extremely helpful in pinpointing where the error lies. Secondly, energy analysis techniques can be used to obtain stability conditions in a much more general sense than the frequency domain analysis techniques presented in Section 3.3. Where frequency domain analysis is restricted to LTI 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 extensive overview in [21] where this has been extensively used to show stability of the FD schemes used.

One of the main goals when performing energy analysis is to find an expression for the total energy present in the system. This is referred to as the *Hamiltonian* and denoted by \mathcal{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.

²Acoustic 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 (δ_t, 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 (so the discretised version of Eq. (2.28) rather than Eq. (2.29)).

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

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

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

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:

$$\delta_{t+}\mathfrak{h} = 0 \implies \mathfrak{h}^n = \mathfrak{h}^0. \quad (3.35)$$

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 will come in handy, as well as summation by parts described in Section 3.2.2 for distributed systems.

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \quad (3.36)$$

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

³[Bilbao, verbally]

(as in the classical-mechanics definition $E_{\text{kin}} = \frac{1}{2} M \dot{u}$) and the potential energy includes the restoring forces of the system.

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

To know that the previous steps have been carried out correctly, it is good to check whether the units of the resulting expression for \mathfrak{h} is indeed in Joules, or $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$. The other quantities such as energy losses q and inputs p , should be in Joules per second or in SI units: $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$. 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. It is important to note that the time shift operator (e_{t-}) does not influence the units. Finally, the appearance of a grid function u_l^n ‘adds’ whatever it describes. Usually, as u_l^n describes a displacement in m, it will ‘add’ this to the equation. If it describes anything else, it will ‘add’ that.

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

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 for 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

$$\mathfrak{h}_e^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0, \quad (3.37)$$

where \mathfrak{h}_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

3.4. Energy analysis

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]):

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

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

3.4.2 Mass-spring system

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

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

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 $(\delta_t.u^n)$:

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

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

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

As there are no losses or externally supplied energy present in the system, all terms are part of the Hamiltonian \mathfrak{h} . To isolate δ_{t+} from (3.39), one can use identities (3.17a) and (3.17b) to get the following:

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

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

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

This can be rewritten in terms of the kinetic energy t and potential energy v according to

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

Step 3: Check units

As mentioned above, the energy 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

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

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

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

3.4. Energy analysis

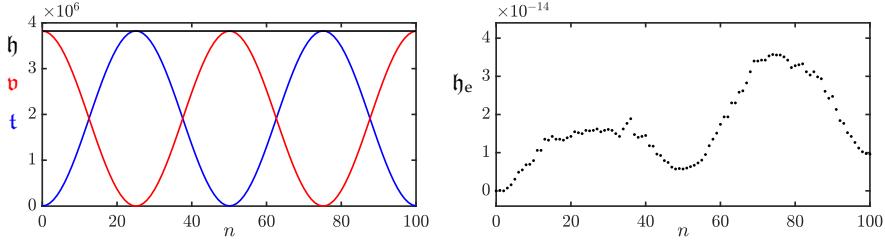


Fig. 3.1: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy according to Eq. (3.37). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within machine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

Taking an inner product using Eq. (3.43) with $(\delta_t.u_l^n)$ and moving all terms to the left-hand side yields the definition for the rate of change of the Hamiltonian:

$$\delta_{t+}h = \rho A \langle \delta_t.u_l^n, \delta_{tt}u_l^n \rangle_d - T \langle \delta_t.u_l^n, \delta_{xx}u_l^n \rangle_d = 0. \quad (3.44)$$

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 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 $f_l^n \triangleq \delta_t.u_l^n$ and $g_l^n \triangleq \delta_{x+}u_l^n$, the second term can be rewritten to

$$-T \langle \delta_t.u_l^n, \delta_{xx}u_l^n \rangle_d = T \langle \delta_{x+}(\delta_t.u_l^n), \delta_{x+}u_l^n \rangle_d - b,$$

where the boundary term

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

and reduced domain $d = \{0, \dots, N - 1\}$. As Dirichlet boundary conditions

are used, the boundary term vanishes as

$$u_0^n = u_N^n = 0 \implies \delta_t \cdot u_0^n = \delta_t \cdot u_N^n = 0.$$

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

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

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

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

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

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \\ \text{with } \mathfrak{t} &= \frac{\rho A}{2} \|\delta_{t-} u_l^n\|_d^2, \quad \text{and} \quad \mathfrak{v} = \frac{T}{2} \langle \delta_{x+} u_l^n, e_{t-} \delta_{x+} u_l^n \rangle_d. \end{aligned} \quad (3.47)$$

Step 3: Check units

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

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

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:

3.4. Energy analysis

```
%> 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, \mathbf{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 0.⁴ 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.

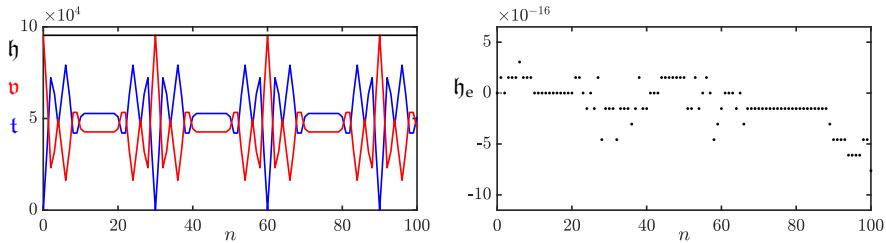


Fig. 3.2: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

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

$$-T\langle \delta_t u_l^n, \delta_{xx} u_l^n \rangle'_d = T\langle \delta_{x+}(\delta_t u_l^n), \delta_{x+} u_l^n \rangle_d - \mathfrak{b},$$

⁴As can be seen from the definition of \mathfrak{v} in Eq. (3.47), the domain used for the inner product is $d = \{0, \dots, N-1\}$ and \mathfrak{v} contains a forward difference in its definition requiring u_N^n as well.

ask stefania
about these
figure captions
being identical

where the boundary term

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

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

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \\ \text{with } \mathfrak{t} &= \frac{\rho A}{2} \left(\|\delta_t \cdot u_l^n\|'_d \right)^2, \quad \text{and} \quad \frac{T}{2} \langle \delta_{x+} u_l^n, e_t - \delta_{x+} u_l^n \rangle_d. \end{aligned} \tag{3.48}$$

Using \mathbf{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.

3.4. Energy analysis

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

$$x^2 + y^2 + 2axy \quad (3.49)$$

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{h} 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

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

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

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

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

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

This can then be written as

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

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

$$k < 2\sqrt{\frac{M}{K}} \quad (3.50)$$

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]

$$\|\delta_{x+} u_l^n\|_{\underline{d}} \leq \frac{2}{h} \|u_l^n\|_d' \leq \frac{2}{h} \|u_l^n\|_d, \quad (3.51)$$

to put a condition on \mathfrak{v}

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

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

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

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

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

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.⁵ This section will show how to numerically obtain the modal frequencies of a FD scheme using the 1D wave equation as a test case.

some citation here

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

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

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

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

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

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

Furthermore, one can substitute a test solution $z_p = e^{s_p k}$ with complex frequency $s_p = j\omega_p + \sigma_p$ which contains the (angular) frequency ω_p and damping

⁵The modal analysis techniques presented here have indeed been used extensively in Chapter 12, precisely for this reason.

check with Stefan

$\sigma_p \leq 0$ of the p^{th} mode.⁶ 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

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

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

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

and can be rewritten to

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

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 *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.

⁶Notice that regardless of the possible damping coefficient per mode, the eventual amplitude of each will mostly be determined by the locations of the excitation and output as discussed in Section 2.4.3.

3.5. Modal analysis

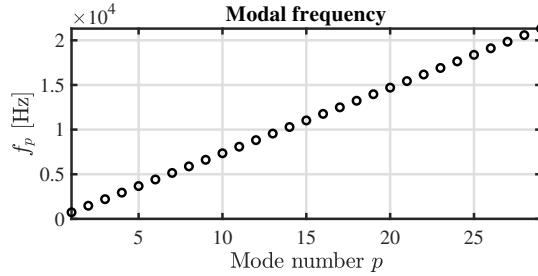


Fig. 3.3: Modal frequencies of the 1D wave equation with the parameters given in Table 2.1.

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

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

can be rewritten to

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

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

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

$$z\phi = \mathbf{Q}\phi, \quad (3.59)$$

which can be solved for the p th eigenvalue as

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

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

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

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

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

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

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u, \quad (4.1)$$

and is parametrised by material density ρ (in kg/m³), cross-sectional area $A = \pi r^2$ (in m²), 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 m⁴). In the limit as $r \rightarrow 0$, 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

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u \quad (4.2)$$

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.

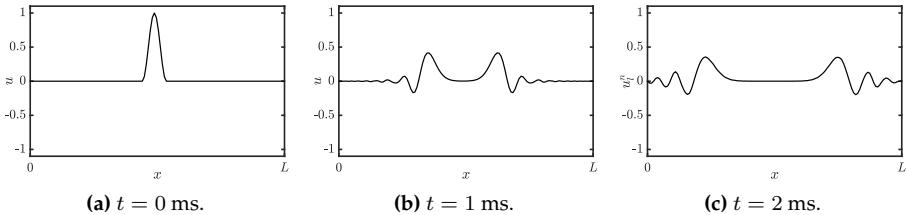


Fig. 4.1: Frequency dispersion in a stiff string due to stiffness.

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

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \quad (4.3)$$

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

4.1. Continuous time

terms.

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

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u - 2\sigma_0 \partial_t u + 2\sigma_1 \partial_t \partial_x^2 u. \quad (4.4)$$

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

$$u = \partial_x u = 0 \quad (\text{clamped}) \quad (4.5a)$$

$$u = \partial_x^2 u = 0 \quad (\text{simply supported}) \quad (4.5b)$$

$$\partial_x^2 u = \partial_x^3 u = 0 \quad (\text{free}) \quad (4.5c)$$

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]

$$f_p = f_0 p \sqrt{1 + B p^2}, \quad (4.6)$$

with inharmonicity coefficient

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

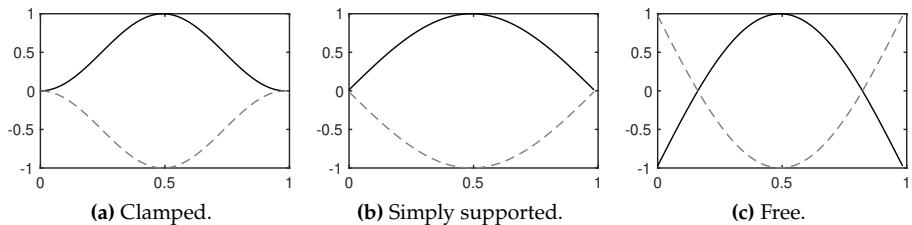


Fig. 4.2: Plots of the first (normalised) modal shape for the three boundary conditions in Eqs. (4.5). The extremes are indicated with solid black and dashed grey lines respectively.

¹Note 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

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_t u_l^n + 2\sigma_1 \delta_{t-} \delta_{xx} u_l^n, \quad (4.7)$$

and is defined for domain $l \in \{0, \dots, N\}$ 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:

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

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

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

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,

4.2. Discrete time

and after a multiplication of all terms by k^2 and collecting the terms, this yields

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

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

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

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

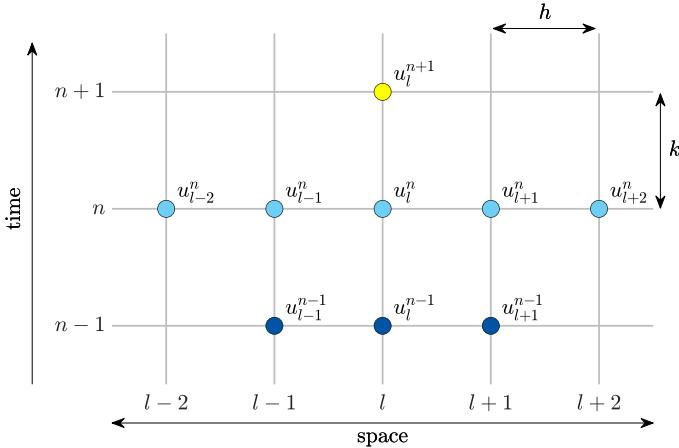


Fig. 4.3: The stencil for the damped stiff string scheme in Eq. (4.7) (adapted from [A]).

conditions in (4.5) yields

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

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

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

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

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

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

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 $l \in$

4.2. Discrete time

$\{1, \dots, N - 1\}$. 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 $l = 0$:

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

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

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

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

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

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 $l \in \{0, \dots, N\}$. 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:

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

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

$$\frac{1}{2h^3} (u_2^n - 2u_1^n + 2u_{-1}^n - u_{-2}^n) = 0,$$

$$u_{-2}^n = u_2^n - 2u_1^n + 2u_{-1}^n.$$

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:

$$\frac{1}{h^2} (u_1^n - 2u_0^n + u_{-1}^n) = 0,$$

$$u_{-1}^n = 2u_0^n - u_1^n.$$

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

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

where

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

$$\text{and} \quad \mathbf{C} = -(1 - \sigma_0 k)\mathbf{I} - 2\sigma_1 k \mathbf{D}_{xx}.$$

Notice that A is a scalar rather than a matrix.

The size of the state vectors and the matrix-form operators depend on the

4.2. Discrete time

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

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

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

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

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

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

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.

4.3. von Neumann analysis and stability condition

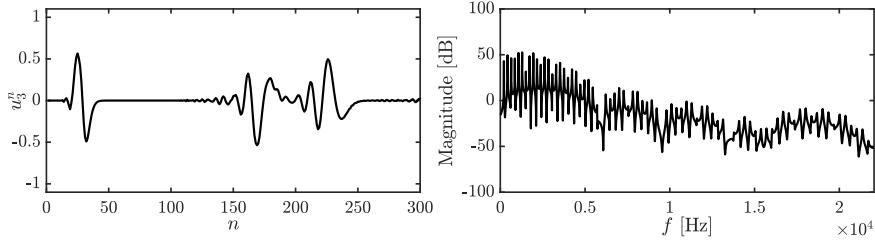


Fig. 4.4: The time-domain and frequency domain output of the stiff string. The parameters are set as in Table 4.1 where $r = 9.35 \cdot 10^{-4}$ m to highlight dispersive effects and $T = 3951$ N.

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.1, 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:

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

and after collecting the terms, the characteristic equation is as follows

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

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.1)

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

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

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

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

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

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 $(\delta_t u_l^n)$ over discrete domain d :

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

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

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q}, \quad (4.27)$$

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

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

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,

4.4. Energy analysis

using Eq. (3.13a) for the second term and Eq. (3.16b) for the third, yields

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

where the boundary term becomes

$$\begin{aligned}\mathfrak{b} &= T \left((\delta_t \cdot u_N^n)(\delta_{x+} u_N^n) - (\delta_t \cdot u_0^n)(\delta_{x+} u_{-1}^n) \right) \\ &\quad + EI \left((\delta_t \cdot u_N^n)(\delta_{x+} \delta_{xx} u_N^n) - (\delta_{xx} u_N^n)(\delta_{x-} \delta_t \cdot u_N^n) \right) \\ &\quad - EI \left((\delta_t \cdot u_0^n)(\delta_{x-} \delta_{xx} u_0^n) - (\delta_{xx} u_0^n)(\delta_{x+} \delta_t \cdot u_0^n) \right).\end{aligned}$$

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

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

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

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

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 $\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}$, 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 \mathbf{q} need to have units of Joules per second, or $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$. Writing the terms in Eq. (4.28) out in their units yields

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

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

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.

4.5. Modal analysis

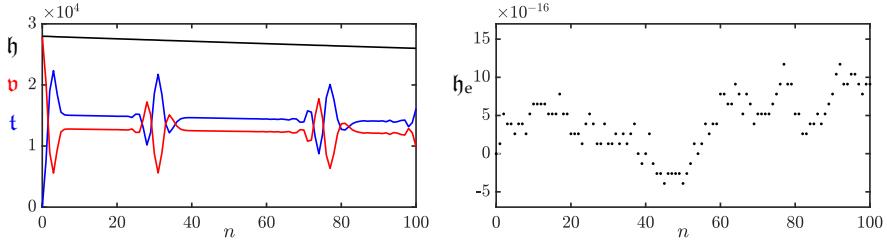


Fig. 4.5: The kinetic (blue), potential (red), and total (black) energy of an implementation of the stiff string are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

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 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

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

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 $\mathbf{w}^n = z^n \phi$, 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):

$$z\phi = \mathbf{Q}\phi, \quad (4.31)$$

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

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

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.,

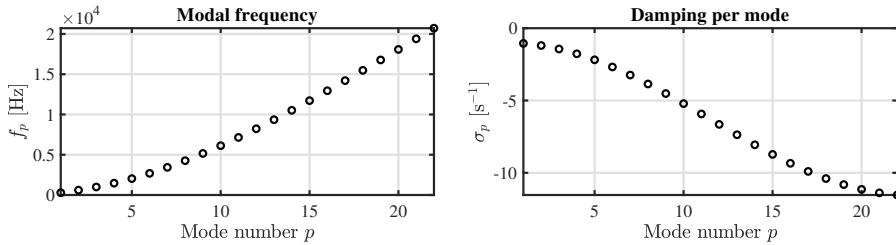


Fig. 4.6: The modal frequencies and damping per mode for the stiff string using the values in Table 4.1 and $T = 1.88 \cdot 10^6$ N and $r = 1.58 \cdot 10^{-2}$ m to highlight effects of stiffness.

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:

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t,x} u_l^n + 2\sigma_1 \delta_{x,t} u_l^n. \quad (4.33)$$

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.

²In 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.

4.6. Implicit scheme

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

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

where

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

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)

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

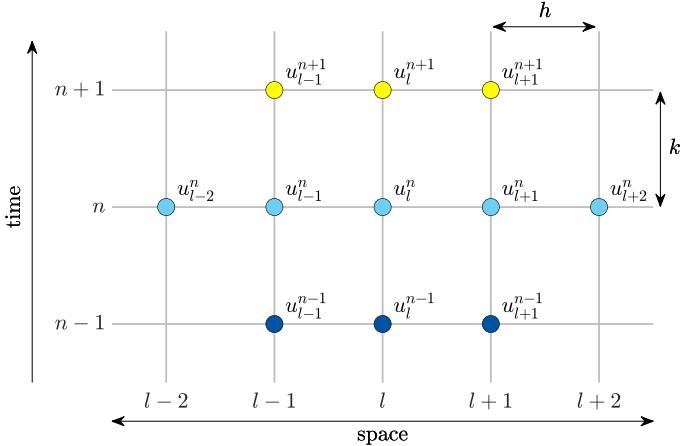


Fig. 4.7: The stencil for the damped stiff string scheme in (4.33).

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.2 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):

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

and collecting the terms, yields the following characteristic equation:

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

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.2):

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

and places the following condition on the grid spacing:

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

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 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

4.6. Implicit scheme

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.

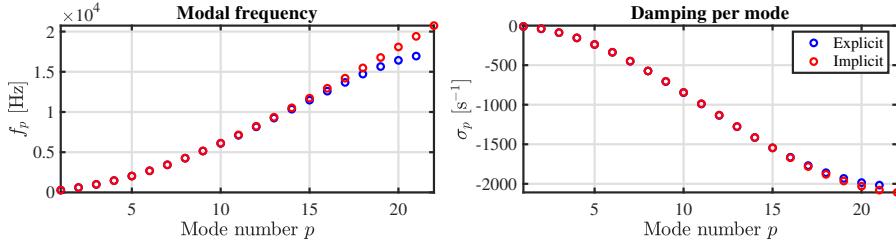


Fig. 4.8: A comparison between the modal frequencies and damping per mode of the explicit (blue) and implicit (red) scheme.

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 4. The Stiff String

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 work, 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]

$$S\partial_t^2\Psi = c^2\partial_x(S\partial_x\Psi), \quad (5.1)$$

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]

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

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.

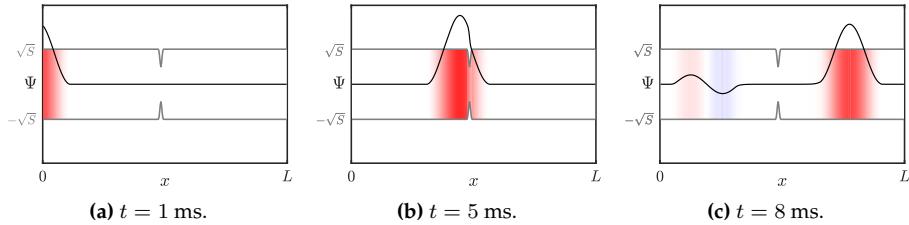


Fig. 5.1: Wave propagation and dispersion in an acoustic tube of varying cross-section (shown in grey) modelled by Webster's equation in Eq. (5.1). Positive acoustic potential Ψ is shown in red and negative in blue, highlighted by a black line for clarity.

Boundary conditions

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

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

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

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

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

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

5.1. Webster's equation

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:

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

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

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

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

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

where

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

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

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

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

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

³Notice 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.

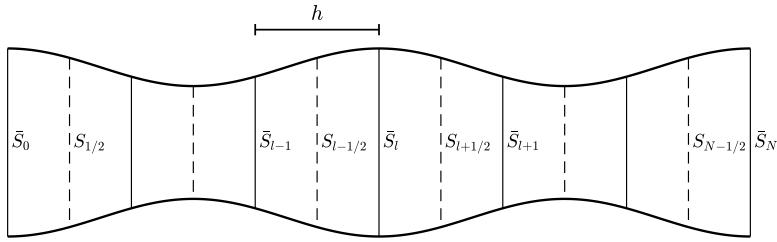


Fig. 5.2: Approximations to $S(x)$ used in Eq. (5.6). Dashed lines indicate the interleaved grid on which S is sampled (Eq. (5.5)) and solid lines indicate \bar{S} which are averages of these (Eq. (5.7)).

and solving for Ψ_l^{n+1} yields the following update equation (see Appendix F.3):

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

where

$$\lambda = \frac{ck}{h}, \quad (5.10)$$

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

$$\lambda \leq 1 \quad (5.11)$$

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

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

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

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

5.1. Webster's equation

according to

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

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

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

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

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

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

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]

$$\partial_x \Psi(L, t) = -a_1 \partial_t \Psi(L, t) - a_2 \Psi(L, t), \quad (5.15)$$

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

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

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]

$$\delta_x \cdot \Psi_N^n = -a_1 \delta_t \cdot \Psi_N^n - a_2 \mu_t \cdot \Psi_N^n, \quad (5.17)$$

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

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

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

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

where

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

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

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

Discretising this using the centred spatial operator, yields

$$\delta_x \cdot \Psi_0^n = -v_{\text{in}}^n \Rightarrow \Psi_{-1}^n = 2hv_{\text{in}}^n + \Psi_1^n, \quad (5.22)$$

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

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

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 $\Psi^n = [\Psi_0^n, \dots, \Psi_N^n]^T$ and creating a \mathbf{D}_{xx} matrix that includes the effect of the

5.1. Webster's equation

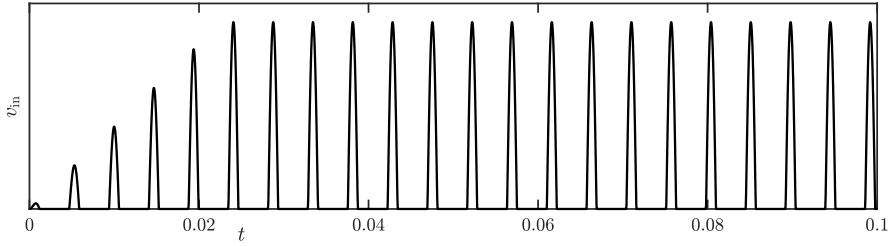


Fig. 5.3: A pulse train with a frequency of 213 Hz a duty cycle of 25% and an attack of 22 ms, used to generate the output in Figure 5.4.

cross-sectional area S . Assuming Neumann boundary conditions yields

$$\mathbf{D}_{xx} = \frac{1}{h^2} \begin{bmatrix} -2 & 2 & & & \mathbf{0} \\ \frac{S_{1/2}}{S_1} & -2 & \frac{S_{3/2}}{S_1} & & \\ \ddots & \ddots & \ddots & & \\ & \frac{S_{l-1/2}}{S_l} & -2 & \frac{S_{l+1/2}}{S_l} & \\ & & \ddots & \ddots & \\ & & \frac{S_{N-3/2}}{S_{N-1}} & -2 & \frac{S_{N-1/2}}{S_{N-1}} \\ \mathbf{0} & & & 2 & -2 \end{bmatrix}. \quad (5.24)$$

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

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

where

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

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

$$\mathbf{v}_i^n = \begin{cases} \frac{2h\lambda^2 S_{-1/2}}{S_0} v_{in}^n, & \text{if } i = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (5.26)$$

Notice how the radiation is included by changing the last entry of matrices \mathbf{A} and \mathbf{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.

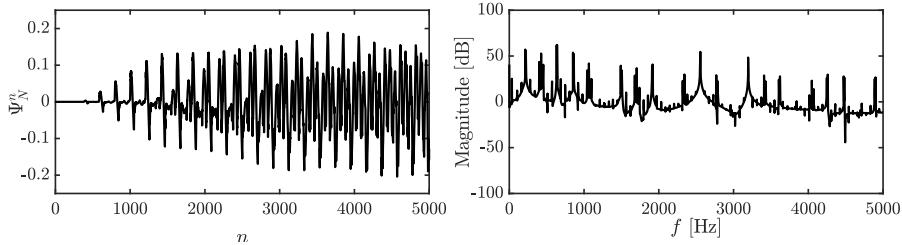


Fig. 5.4: The output of Webster's equation at Ψ_N^n using the input in Figure 5.3, the parameters in Table 5.1, and the geometry in Figure 5.5.

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).

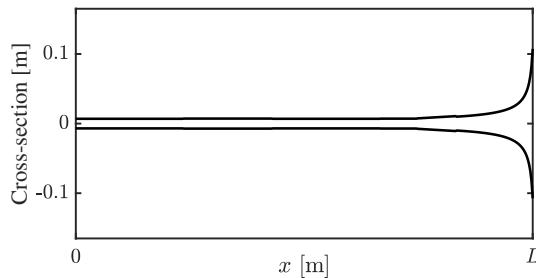


Fig. 5.5: The geometry used for the implementation. See paper [H] for more details.

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.

5.1. Webster's equation

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 $0 < \epsilon_l, \epsilon_r \leq 2$ at the left and right boundary respectively, of scheme (5.6) with $(\delta_t \cdot \Psi_l^n)$ over discrete domain d , yields

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

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:

$$\delta_{t+}(\mathfrak{h} + \mathfrak{h}_b) = \mathfrak{b} - \mathfrak{q}_b, \quad (5.28)$$

where \mathfrak{h}_b is the energy stored by the radiating boundary through the inertia term, \mathfrak{q}_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

$$c^2 \langle S_{l+1/2} \delta_t \cdot \delta_{x+} \Psi_l^n, (\delta_{x+} \Psi_l^n) \rangle_d + \mathfrak{b}_r - \mathfrak{b}_l,$$

where

$$\mathfrak{b}_r = c^2 (\delta_t \cdot \Psi_N^n) \left(\frac{\epsilon_r}{2} S_{N+1/2} (\delta_{x+} \Psi_N^n) + \left(1 - \frac{\epsilon_r}{2} \right) S_{N-1/2} (\delta_{x-} \Psi_N^n) \right), \quad (5.29a)$$

$$\mathfrak{b}_l = c^2 (\delta_t \cdot \Psi_0^n) \left(\frac{\epsilon_l}{2} S_{-1/2} (\delta_{x-} \Psi_0^n) + \left(1 - \frac{\epsilon_l}{2} \right) S_{1/2} (\delta_{x+} \Psi_0^n) \right), \quad (5.29b)$$

are the right and left boundary term respectively (notice that \mathfrak{b}_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

$$\delta_{t+}\mathfrak{h} = \mathfrak{b}_r - \mathfrak{b}_l,$$

where

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{1}{2} \left(\|\sqrt{\bar{S}_l} \delta_{t-} \Psi_l^n\|_d^{\epsilon_l, \epsilon_r} \right)^2 \quad \text{and} \\ \mathfrak{v} &= \frac{c^2}{2} \langle S_{l+1/2} \delta_{x+} \Psi_l^n, e_t - \delta_{x+} \Psi_l^n \rangle_d. \end{aligned} \quad (5.30)$$

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

$$\begin{aligned}\delta_x \cdot \Psi_0^n = 0 &\Rightarrow b_l = 0 \\ \delta_x \cdot \Psi_N^n = 0 &\Rightarrow b_r = 0\end{aligned}$$

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

$$b_r = c^2(\delta_t \cdot \Psi_N^n)S_{N-1/2}(2 - \epsilon_r)(\delta_x \cdot \Psi_N^n), \quad (5.31a)$$

$$b_l = c^2(\delta_t \cdot \Psi_0^n)S_{1/2}(2 - \epsilon_l)(\delta_x \cdot \Psi_0^n). \quad (5.31b)$$

See Appendix F.4 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 b_r in Eq. (5.31a) as

$$\begin{aligned}b_r &= c^2(\delta_t \cdot \Psi_N^n)S_{N-1/2}(2 - \epsilon_r)(-a_1\delta_t \cdot \Psi_N^n - a_2\mu_t \cdot \Psi_N^n), \\ &= c^2S_{N-1/2}(2 - \epsilon_r)(-a_1(\delta_t \cdot \Psi_N^n)^2 - a_2(\delta_t \cdot \Psi_N^n)(\mu_t \cdot \Psi_N^n)).\end{aligned}$$

This can then be decomposed in h_b and q_b used in Eq. (5.27).

Using identity (3.17d) yields the definitions for h_b and q_b in Eq. (5.28)

$$h_b = \frac{c^2S_{N-1/2}(2 - \epsilon_r)a_2}{2}\mu_t \cdot (\Psi_N^n)^2, \quad \text{and} \quad q_b = c^2S_{N-1/2}(2 - \epsilon_r)a_1(\delta_t \cdot \Psi_N^n)^2. \quad (5.32)$$

Finally, $b = b_l$ 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 $S(x) = 0.01$ m² 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.

5.1. Webster's equation

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

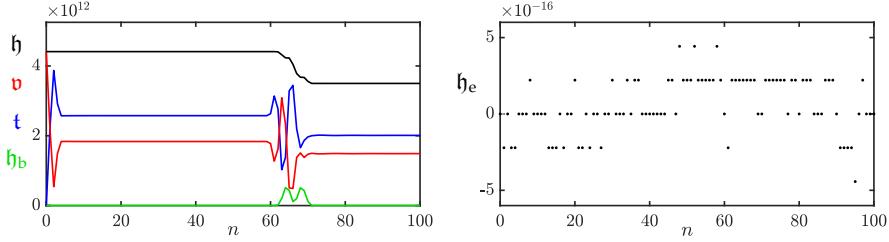


Fig. 5.6: The kinetic (blue), potential (red), and total (black) energy as well as the energy stored by the radiation condition (green) of an implementation of Webster's equation are plotted in the left panel. Notice that the energy decreases between $n = 60$ and $n = 70$ as the excitation reached the boundary where damping is included. 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

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

where $[S]_l$ is a still 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

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

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

$$\|\sqrt{\phi_l} \delta_{x+} u_l^n\|_{\underline{d}} \leq \frac{2}{h} \|\sqrt{\mu_{x-} \phi_l} u_l^n\|_{\underline{d}}, \quad (5.34)$$

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}

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

where the last step is possible because $0 < \epsilon_l, \epsilon_r \leq 2$. Substituting this into the energy balance in Eq. (5.30) yields

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

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

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

This can be written as

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

which is non-negative if

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

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

$$\lambda \leq 1, \quad (5.36)$$

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].

look at word-ing

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 PDEs that describes air propagation in an acoustic tube can be defined as follows:

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

$$\rho_0 \partial_t v = -\partial_x p. \quad (5.37b)$$

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

$$p = \rho_0 \partial_t \Psi, \quad v = -\partial_x \Psi. \quad (5.38)$$

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

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

Boundary conditions

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

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

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

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

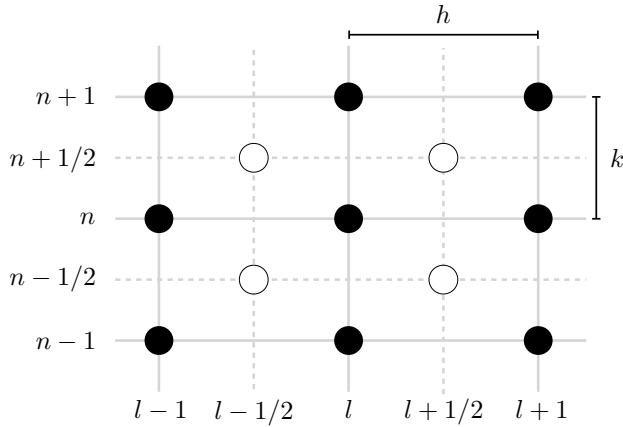


Fig. 5.7: The interleaved grid used for the system of FD schemes in Eq. (5.40). Grid points on the regular grid (in black) are used for pressure p , while points on the interleaved grid (in white) are used for particle velocity v .

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

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

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

after which the update equations become

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

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

where (again) $\lambda = ck/h \leq 1$ for stability. The pressure is defined for $l = \{0, \dots, N\}$ and the velocity for $l = \{0, \dots, N-1\}$ 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 is one fewer grid point 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

$$p_0^n = 0, \quad p_N^n = 0, \quad (\text{Dirichlet, open}), \quad (5.42\text{a})$$

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

stencil?

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

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

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

$$\mathbf{v}^{n+1/2} = \mathbf{v}^{n-1/2} + \mathbf{B}_p \mathbf{p}^n \quad (5.44\text{a})$$

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \mathbf{B}_v \mathbf{v}^{n+1/2} \quad (5.44\text{b})$$

where

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

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

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

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

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

will then be

$$\underbrace{\begin{bmatrix} \mathbf{p}^{n+1} \\ \mathbf{v}^{n+1/2} \end{bmatrix}}_{\mathbf{u}^{n+1}} = \mathbf{B} \underbrace{\begin{bmatrix} \mathbf{p}^n \\ \mathbf{v}^{n-1/2} \end{bmatrix}}_{\mathbf{u}^n}, \quad (5.47)$$

where

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

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.

Finally, the concatenated vector is defined as

$$\mathbf{u}^n = [p_0^n, \dots, p_N^n, v_{1/2}^{n-1/2}, \dots, v_{N-1/2}^{n-1/2}]^T \quad (5.49)$$

and will be of size $(2N + 1) \times 1$.

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, yields⁴

$$\delta_{t+}\mathfrak{h} = \frac{1}{\rho_0 c^2} \langle \mu_{t+} p_l^n, \bar{S} \delta_{t+} p_l^n \rangle'_d + \langle \mu_{t+} p_l^n, \delta_{x-} (S_{l+1/2} v_{l+1/2}^{n+1/2}) \rangle'_d = 0. \quad (5.50)$$

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. (5.50) becomes

$$\delta_{t+}\mathfrak{h} = \frac{1}{\rho_0 c^2} \langle \mu_{t+} p, \bar{S} \delta_{t+} p \rangle'_d - \langle \mu_{t+} \delta_{x+} p, S v \rangle_d = -\mathfrak{b}, \quad (5.51)$$

⁴The 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).

5.2. First-order system

where the boundary term is

$$\mathfrak{b} = \mathfrak{b}_r - \mathfrak{b}_l, \quad \text{with}$$

$$\mathfrak{b}_r = (\mu_{t+} p_N) \mu_{x+} (S_{N-1/2} v_{N-1/2}) \quad \text{and} \quad (5.52)$$

$$\mathfrak{b}_l = (\mu_{t+} p_0) \mu_{x-} (S_{1/2} v_{1/2}), \quad (5.53)$$

and can be shown to vanish under the boundary conditions shown in Eq. (5.42a). Then, Eq. (5.40b) can be substituted into Eq. (5.51) to get

$$\delta_{t+} \mathfrak{h} = \frac{1}{\rho_0 c^2} \langle \mu_{t+} p, \bar{S} \delta_{t+} p \rangle'_d - \langle \mu_{t+} (-\rho_0 \delta_{t-} v), S v \rangle_d = 0 \quad (5.54)$$

$$= \frac{1}{\rho_0 c^2} \langle \mu_{t+} p, \bar{S} \delta_{t+} p \rangle'_d + \rho_0 \langle \delta_{t-} v, S v \rangle_d = 0. \quad (5.55)$$

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

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \quad \text{where} \\ \mathfrak{t} &= \frac{\rho_0}{2} \langle S v, e_{t-} v \rangle_d \quad \text{and} \quad \mathfrak{v} = \frac{1}{2\rho_0 c^2} \left(\|\sqrt{\bar{S}} p\|'_d \right)^2. \end{aligned} \quad (5.56)$$

Step 3: Check units

Writing the terms in Eq. (5.56) in their units yields

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

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 work was proposed by Levine and Schwinger in [74], discretised by Silva et al. [75]. Although other radiation

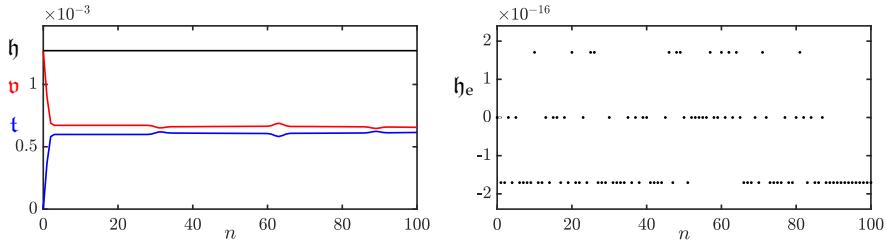


Fig. 5.8: The kinetic (blue), potential (red), and total (black) energy of an implementation of the first-order system in Eq. (5.40) are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

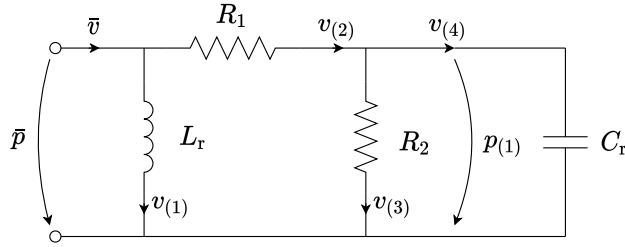


Fig. 5.9: The circuit representation of the Levine and Schwinger radiation model.

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].

The system can be described as

$$\bar{v} = \mu_{t+} v_{(1)} + \frac{1}{R_2} \mu_{t+} p_{(1)} + C_r \delta_{t+} p_{(1)}, \quad (5.57a)$$

$$\bar{p} = L_r \delta_{t+} v_{(1)}, \quad (5.57b)$$

$$\bar{p} = \left(1 + \frac{R_1}{R_2}\right) \mu_{t+} p_{(1)} + R_1 C_r \delta_{t+} p_{(1)}, \quad (5.57c)$$

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

$$\bar{p} = \mu_{t+} p_N^n, \quad \bar{S}_N \bar{v} = \mu_{x-} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} \right). \quad (5.58)$$

Using the above, an update for p_N^{n+1} based on known values of the system can

be obtained (see Appendix (F.5) for a derivation):

$$p_N^{n+1} = \frac{1 - \rho_0 c \lambda \zeta_3}{1 + \rho_0 c \lambda \zeta_3} p_N^n - \frac{2 \rho_0 c \lambda}{1 + \rho_0 c \lambda \zeta_3} \left(v_{(1)}^n + \zeta_4 p_{(1)}^n - \frac{S_{N-1/2} v_{N-1/2}^{n+1/2}}{\bar{S}_N} \right), \quad (5.59)$$

where

$$\begin{aligned} \zeta_1 &= \frac{2R_2 k}{2R_1 R_2 C_r + k(R_1 + R_2)}, & \zeta_2 &= \frac{2R_1 R_2 C_r - k(R_1 + R_2)}{2R_1 R_2 C_r + k(R_1 + R_2)} \\ \zeta_3 &= \frac{k}{2L_r} + \frac{\zeta_1}{2R_2} + \frac{C_r \zeta_1}{k} \quad \text{and} \quad \zeta_4 = \frac{\zeta_2 + 1}{2R_2} + \frac{C_r \zeta_2 - C_r}{k}. \end{aligned}$$

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

$$v_{(1)}^{n+1} = v_{(1)}^n + \frac{k}{L_r} \mu_t + p_N^n, \quad (5.60a)$$

$$p_{(1)}^{n+1} = \zeta_1 \mu_t + p_N^n + \zeta_2 p_{(1)}^n. \quad (5.60b)$$

The various parameters are taken from [62] and are

$$\begin{aligned} R_1 &= \rho_0 c, & R_2 &= 0.505 \rho_0 c, \\ L_r &= 0.613 \rho_0 \sqrt{\bar{S}_N / \pi}, & \text{and} \quad C_r &= 1.111 \frac{\sqrt{\bar{S}_N}}{\rho_0 c^2 \sqrt{\pi}}. \end{aligned}$$

temperature
dependent
parameters somewhere?
[table](#)

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, and such that it can be used for implementation and debugging purposes.

One can perform an energy analysis of the radiation model by recalling the condition at the right boundary from Eq. (5.52)

$$\mathbf{b}_r = (\mu_t + p_N) \underbrace{(\mu_{x+}(S_{N-1/2} v_{N-1/2}) - \mu_{x-}(S_{N+1/2} v_{N+1/2}))}_{\mathbf{b}_r}, \quad (5.61)$$

which, using Eq. (5.58), can be rewritten to

$$\mathbf{b}_r = \bar{p} \bar{S}_N \bar{v}. \quad (5.62)$$

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

rewritten to

$$\delta_{t+} \mathfrak{h}_{\text{rad}} + \mathfrak{q}_{\text{rad}} = 0, \quad (5.63)$$

where

$$\begin{aligned}\mathfrak{h}_{\text{rad}} &= \frac{\bar{S}_N}{2} \left(L_r v_{(1)}^2 + C_r p_{(1)}^2 \right), \quad \text{and} \\ \mathfrak{q}_{\text{rad}} &= \bar{S}_N \left(R_1 (\mu_{t+} v_{(2)}^n)^2 + R_2 (\mu_{t+} v_{(3)})^2 \right),\end{aligned}$$

with

$$v_{(3)}^n = \frac{p_{(1)}^n}{R_2}, \quad \text{and} \quad \mu_{t+} v_{(2)}^n = \frac{\mu_{t+} p_N^n - \mu_{t+} p_{(1)}^n}{R_1}. \quad (5.64)$$

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.¹ Afterwards, two 2D models that are used in this work, 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 $u(x, y, t) \cong u_{l,m}^n$ with space

¹The abbreviation 2D will also be used for ‘two dimensions’.

$x = lh$ and $y = mh$ and time $t = nk$ with $k = 1/f_s$. 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

$$\Delta = \partial_x^2 + \partial_y^2, \quad (6.1)$$

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:

$$\Delta\Delta = \partial_x^4 + 2\partial_x^2\partial_y^2 + \partial_y^4. \quad (6.2)$$

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

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

Using these shift operators, a discrete approximation of the Laplacian in Eq. (6.1) can be made²

$$\Delta \approx \delta_\Delta \triangleq \frac{1}{h^2} (e_{x+} + e_{x-} + e_{y+} + e_{y-} - 4). \quad (6.4)$$

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

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

Also see Figure 6.1b.

²Notice that the δ_Δ operator is identical to $\delta_{\Delta\Box}$ in [21], but will not be used here as Eq. (6.4) is the only discretisation to the Δ operator used in this work.

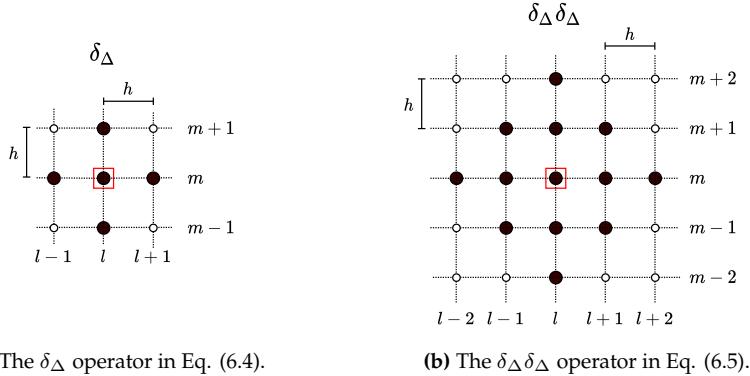


Fig. 6.1: The stencils of the 2D spatial FD operators in Eqs. (6.4) and (6.5) respectively. The red square denotes what grid point the operator is applied to. The stencils follow the same layout as Figure 2.3, but the vertical axis denotes a second spatial dimension rather than time.

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:

$$\partial_t^2 u = c^2 \Delta u, \quad (6.6)$$

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/m³) and thickness H (in m).

Boundary conditions

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

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

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

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

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

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

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

where the Courant number

$$\lambda = \frac{ck}{h}, \quad (6.10)$$

and needs to abide

$$\lambda \leq \frac{1}{\sqrt{2}} \quad (6.11)$$

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 limit on h :

$$h \geq \sqrt{2}ck. \quad (6.12)$$

Discrete boundary conditions

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

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

$$\left. \begin{array}{l} \delta_x \cdot u_{0,m}^n = \delta_x \cdot u_{N_x,m}^n = 0 \quad \forall m, \\ \delta_y \cdot u_{l,0}^n = \delta_y \cdot u_{l,N_y}^n = 0 \quad \forall l, \end{array} \right\} \text{(Neumann, free).} \quad (6.13b)$$

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

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).

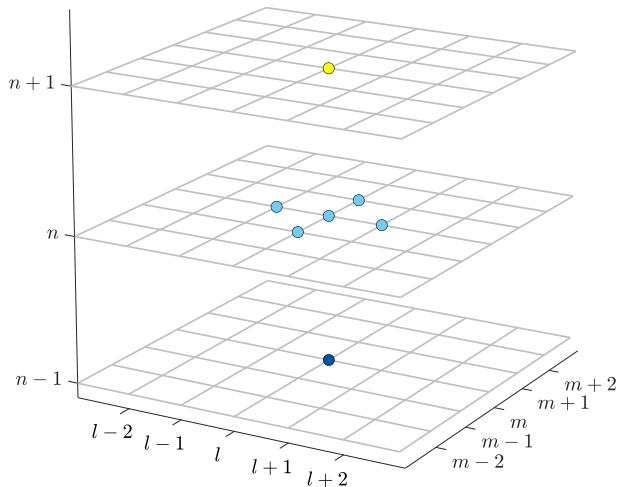


Fig. 6.2: The stencil for the 2D wave equation FD scheme in Eq. (6.8).

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:

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

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

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

and could be used to make a ‘for-loop implementation’ of the update equation. This would 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

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

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

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

The Kronecker sum between two square matrices is [50]

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

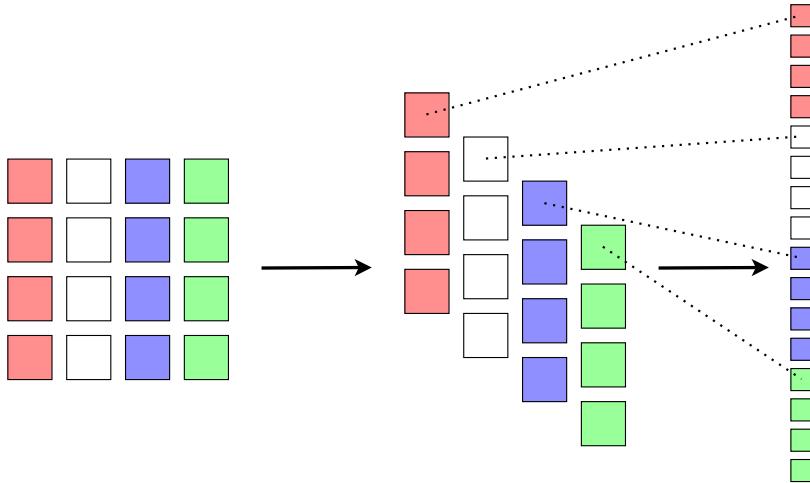


Fig. 6.3: Stacking, or ‘flattening’ a 4×4 matrix to a 16-element vector.

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

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

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

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

where the identity matrix $\mathbf{I} = \mathbf{I}_{N_x-1}$. The \mathbf{D}_Δ 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

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

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

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.

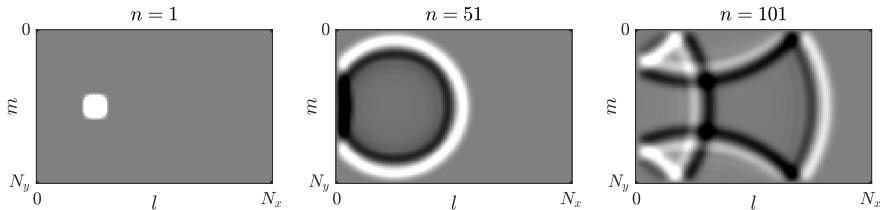


Fig. 6.4: Wave propagation of an implementation of the 2D wave equation with $L_x = 1.5$ m, $L_y = 1$ m and $c = 360$ m/s. The system is excited with a 2D raised cosine at $(0.25L_x, 0.5L_y)$.

³As the matrices are extremely sparse (many 0-entries), it is useful to utilise MATLAB's optimisation for sparse matrices using the `sparse()` function. One can use `speye()` for sparse identity matrices.

6.2. The 2D wave equation

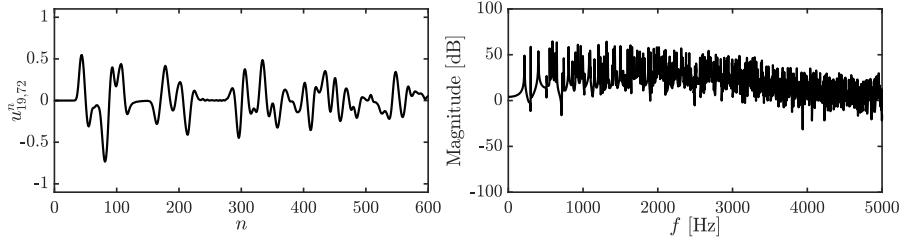


Fig. 6.5: The output of the 2D wave equation at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.4. The partials are extremely close together (notice that only frequencies up to 5000 Hz are shown) which is related to the aperiodic nature of the system behaviour.

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

$$u_{l,m}^n \xrightarrow{\mathcal{A}} z^n e^{jh(l\beta_x + m\beta_y)} \quad (6.22)$$

where β_x and β_y are components of a 2D wavenumber β 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

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

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

$$\delta_{xx} u_{l,m}^n \xrightarrow{\mathcal{A}} -\frac{4}{h^2} p_x u_{l,m}^n \quad \text{and} \quad \delta_{yy} u_{l,m}^n \xrightarrow{\mathcal{A}} -\frac{4}{h^2} p_y u_{l,m}^n, \quad (6.24)$$

from which it follows that

$$\delta_\Delta u_{l,m}^n \xrightarrow{\mathcal{A}} -\frac{4}{h^2} (p_x + p_y) u_{l,m}^n, \quad (6.25)$$

$$\delta_\Delta \delta_\Delta u_{l,m}^n \xrightarrow{\mathcal{A}} \frac{16}{h^4} (p_x + p_y)^2 u_{l,m}^n. \quad (6.26)$$

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

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

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

$$z + (4\lambda^2(p_x + p_y) - 2) + z^{-1} = 0. \quad (6.27)$$

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)

$$|4\lambda^2(p_x + p_y) - 2| \leq 2.$$

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

$$\begin{aligned} 2\lambda^2 &\leq 1, \\ \lambda &\leq \frac{1}{\sqrt{2}} \end{aligned} \quad (6.28)$$

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

$$\langle f, g \rangle_{\mathcal{D}} = \iint_{\mathcal{D}} f g dx dy. \quad (6.29)$$

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

$$\langle f_{l,m}^n, g_{l,m}^n \rangle_d = \sum_{l=0}^{N_x} \sum_{m=0}^{N_y} h^2 f_{l,m}^n g_{l,m}^n. \quad (6.30)$$

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

$$\underline{d}_x = \{0, \dots, N_x - 1\} \times \{0, \dots, N_y\}, \quad (6.31a)$$

$$\overline{d}_x = \{1, \dots, N_x - 1\} \times \{0, \dots, N_y\}, \quad (6.31b)$$

$$\underline{d}_y = \{0, \dots, N_x\} \times \{0, \dots, N_y - 1\}, \quad (6.31c)$$

$$\overline{d}_y = \{0, \dots, N_x\} \times \{1, \dots, N_y - 1\} \quad (6.31d)$$

$$\bar{d} = \{1, \dots, N_x - 1\} \times \{1, \dots, N_y - 1\} \quad (6.31e)$$

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 $(\delta_t u_{l,m}^n)$ over discrete domain d can be taken to yield a definition for $\delta_{t+}\mathfrak{h}$:

$$\delta_{t+}\mathfrak{h} = \rho H \langle \delta_t u_{l,m}^n, \delta_{tt} u_{l,m}^n \rangle_d - T \langle \delta_t u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_d = 0,$$

which can be rewritten to

$$\delta_{t+}\mathfrak{h} = \rho H \langle \delta_t u_{l,m}^n, \delta_{tt} u_{l,m}^n \rangle_d - T (\langle \delta_t u_{l,m}^n, \delta_{xx} u_{l,m}^n \rangle_d + \langle \delta_t u_{l,m}^n, \delta_{yy} u_{l,m}^n \rangle_d) = 0.$$

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

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v} \quad \text{with} \quad \mathfrak{t} = \frac{\rho H}{2} \|\delta_{t-} u_{l,m}^n\|_d^2 \quad \text{and} \\ \mathfrak{v} &= \frac{T}{2} \left(\langle \delta_{x+} u_{l,m}^n, e_{t-} \delta_{x+} u_{l,m}^n \rangle_{\underline{d}_x} + \langle \delta_{y+} u_{l,m}^n, e_{t-} \delta_{y+} u_{l,m}^n \rangle_{\underline{d}_y} \right). \end{aligned} \quad (6.32)$$

FULL DOC
SWEEP: check what equations have numbers when performing energy analysis (and stability for that matter)

Here, the reduced domains \underline{d}_x and \underline{d}_y are as defined in Eqs. (6.31). The boundary term is

$$\mathfrak{b} = \frac{T}{2} \left[\langle \delta_t u_{N_x, m}^n, \delta_x + u_{N_x, m}^n \rangle_{(N_x, y)} - \langle \delta_t u_{0, m}^n, \delta_x - u_{0, m}^n \rangle_{(0, y)} \right. \\ \left. + \langle \delta_t u_{l, N_y}^n, \delta_y + u_{l, N_y}^n \rangle_{(x, N_y)} - \langle \delta_t u_{l, 0}^n, \delta_y - u_{l, 0}^n \rangle_{(x, 0)} \right],$$

where $(l, y) = \{l\} \times \{0, \dots, N_y\}$ and $(x, m) = \{0, \dots, N_x\} \times \{m\}$ 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:

$$\mathfrak{t} = \frac{\rho H}{2} \|\delta_t - u_{l, m}^n\|_d^2 \xrightarrow{\text{in units}} \text{kg} \cdot \text{m}^{-3} \cdot \text{m} \cdot \text{m}^2 \cdot (\text{s}^{-1} \cdot \text{m})^2 \\ = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$$

$$\mathfrak{v} = \frac{T}{2} \langle \delta_x + u_{l, m}^n, e_t - \delta_x + u_{l, m}^n \rangle_{\underline{d}_x} \xrightarrow{\text{in units}} \text{N} \cdot \text{m}^{-1} \cdot \text{m}^2 \cdot (\text{m}^{-1} \cdot \text{m}) \cdot (\text{m}^{-1} \cdot \text{m}) \\ = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$$

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 $u^n = z^n \phi$ into the matrix form of the 2D wave

6.2. The 2D wave equation

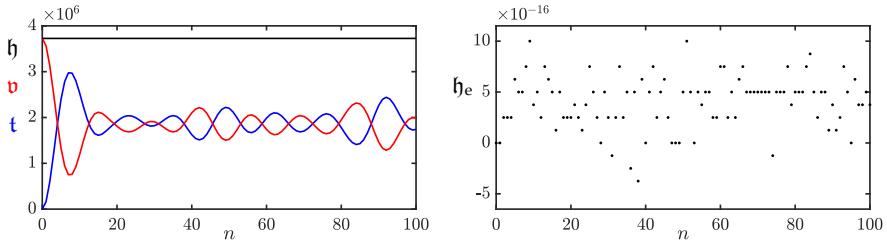


Fig. 6.6: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 2D wave equation are plotted in the left panel. The right panel shows the normalised energy according to Eq. (3.37) and shows that the deviation of the energy is within machine precision.

equation in Eq. (6.21) yields the following characteristic equation

$$(z - 2 + z^{-1}) \phi = c^2 k^2 \mathbf{D}_\Delta \phi. \quad (6.33)$$

The p^{th} modal frequency can then be obtained by finding the roots of

$$z_p + \left(-2 - c^2 k^2 \text{eig}_p(\mathbf{D}_\Delta) \right) + z_p^{-1} = 0, \quad (6.34)$$

which, using test solution $z_p = e^{j\omega_p k}$ for (angular) frequency ω_p , can be shown to be

$$f_p = \frac{1}{\pi k} \sin^{-1} \left(\frac{ck}{2} \sqrt{-\text{eig}_p(\mathbf{D}_\Delta)} \right). \quad (6.35)$$

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.

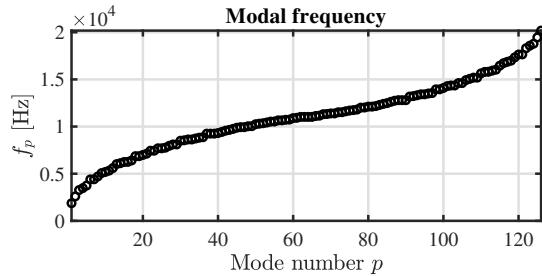


Fig. 6.7: Modal frequencies of the FD scheme implementing the 2D wave equation in Eq. (6.8) with $L_x = 1.5$ m, $L_y = 1$ m and $c \approx 3118$ m/s, such that $\lambda = 1/\sqrt{2}$ according to Eq. (6.11).

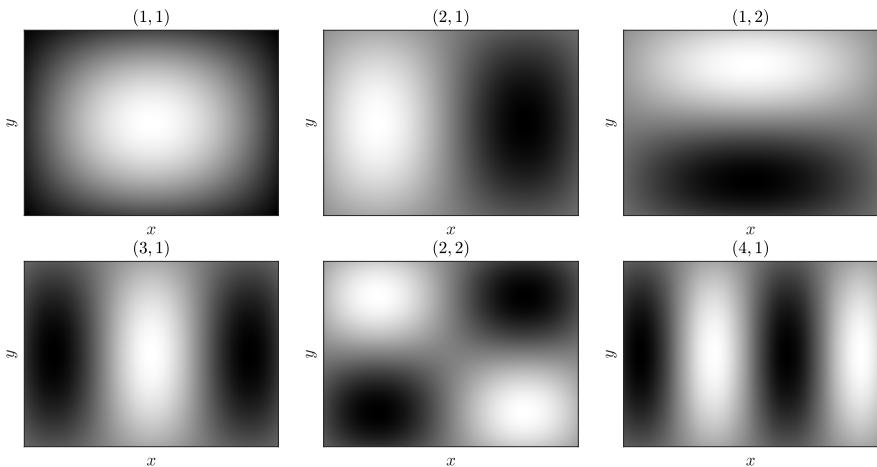


Fig. 6.8: The first six (lowest-frequency) modal shapes of 2D wave equation with $L_x = 1.5$ m and $L_y = 1$ m.

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 reverberation [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]

$$\rho H \partial_t^2 u = -D \Delta \Delta u, \quad (6.36)$$

where $D = EH^3/12(1-\nu^2)$ is a stiffness coefficient (in $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$) 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

$$\rho H \partial_t^2 u = -D \Delta \Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u \quad (6.37)$$

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.

check hyphens after 'frequency'

Boundary conditions

Similar to the stiff string, clamped and simply supported boundary conditions can be defined as

$$\left. \begin{array}{l} u = \partial_x u = 0, \quad \text{if } y = \{0, L_y\}, \quad \forall x \\ u = \partial_y u = 0, \quad \text{if } x = \{0, L_x\}, \quad \forall y \end{array} \right\} \quad (\text{Clamped}), \quad (6.38a)$$

$$\left. \begin{array}{l} u = \partial_x^2 u = 0, \quad \text{if } y = \{0, L_y\}, \quad \forall x \\ u = \partial_y^2 u = 0, \quad \text{if } x = \{0, L_x\}, \quad \forall y \end{array} \right\} \quad (\text{Simply supported}). \quad (6.38b)$$

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:

$$\rho H \delta_{tt} u_{l,m}^n = -D \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_t u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-} \delta_\Delta u_{l,m}^n \quad (6.39)$$

where $l \in \{0, \dots, N_x\}$ and $m \in \{0, \dots, N_y\}$. 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

$$\delta_{tt} u_{l,m}^n = -\kappa^2 \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_t u_{l,m}^n + 2\sigma_1 \delta_{t-} \delta_\Delta u_{l,m}^n \quad (6.40)$$

with

$$\kappa = \sqrt{\frac{D}{\rho H}}. \quad (6.41)$$

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

$$\begin{aligned} u_{l,m}^{n+1} &= (2 - 20\mu^2 - 4S)u_{l,m}^n \\ &\quad + (8\mu^2 + S)(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n) \\ &\quad - 2\mu^2(u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n) \\ &\quad - \mu^2(u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n), \\ &\quad + (\sigma_0 k - 1 + 4S)u_{l,m}^{n-1} \\ &\quad - S(u_{l+1,m}^{n-1} + u_{l-1,m}^{n-1} + u_{l,m+1}^{n-1} + u_{l,m-1}^{n-1}) \end{aligned} \quad (6.42)$$

where

$$\mu = \frac{\kappa k}{h^2} \quad (6.43)$$

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

$$h \geq 2 \sqrt{k \left(\sigma_1 + \sqrt{\kappa^2 + \sigma_1^2} \right)}, \quad (6.44)$$

and will be derived in Section 6.3.4.

6.3. The thin plate

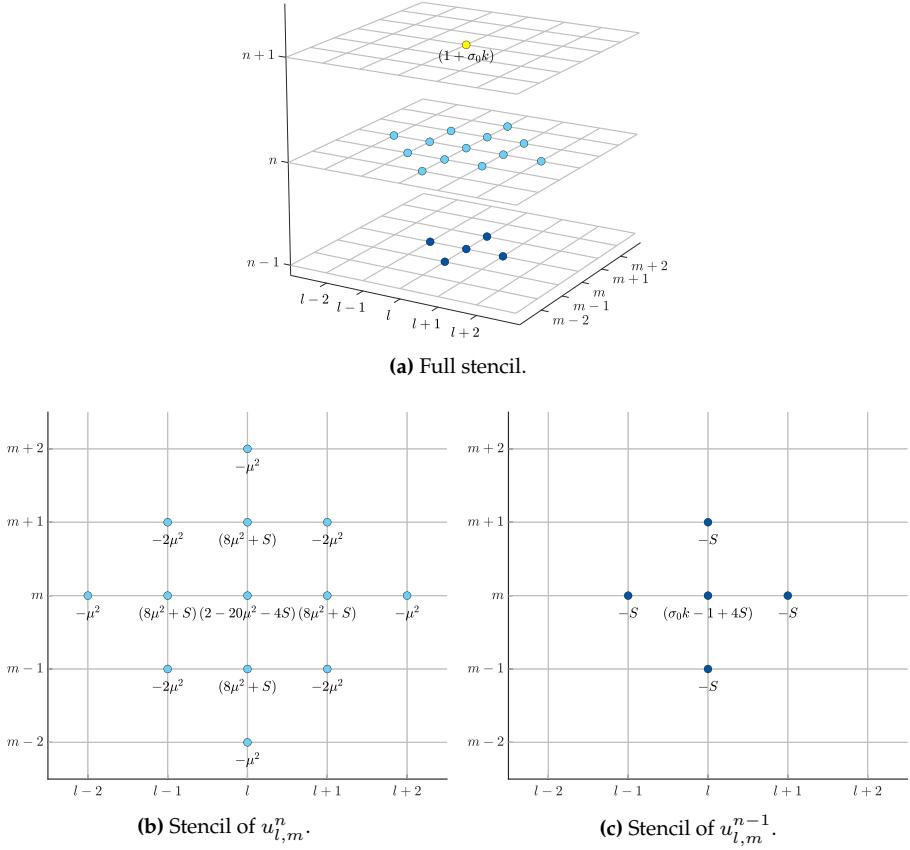


Fig. 6.9: The stencil of the plate with coefficients corresponding to those in update equation (6.42). (a) A full overview of the stencil. (b) The current time-step n highlighted. (c) The previous time-step $n - 1$ highlighted.

Discrete boundary conditions

The boundary conditions shown in Eq. (6.38) can be discretised to

$$\left. \begin{array}{ll} u_{l,m}^n = \delta_{x+} u_{l,m}^n = 0 & \text{if } m = 0 \\ u_{l,m}^n = \delta_{x-} u_{l,m}^n = 0 & \text{if } m = N_y \\ u_{l,m}^n = \delta_{y+} u_{l,m}^n = 0 & \text{if } l = 0 \\ u_{l,m}^n = \delta_{y-} u_{l,m}^n = 0 & \text{if } l = N_x \end{array} \right\} \quad (\text{Clamped}), \quad (6.45a)$$

$$\left. \begin{array}{ll} u_{l,m}^n = \delta_{xx} u_{l,m}^n = 0 & \text{if } m = \{0, N_y\} \\ u_{l,m}^n = \delta_{yy} u_{l,m}^n = 0 & \text{if } l = \{0, N_x\} \end{array} \right\} \quad (\text{Simply supported}). \quad (6.45b)$$

The clamped condition can be implemented by simply reducing the discrete range of operation to $l = \{2, \dots, N_x - 2\}$ and $m = \{2, \dots, N_y - 2\}$. For the simply supported case, the range of operation reduces to $l = \{1, \dots, N_x - 1\}$ and $m = \{1, \dots, N_y - 1\}$, and similar to the simply supported stiff string described in Section 4.2.1, the virtual grid points needed for this condition become

$$\begin{aligned} u_{-1,m}^n &= -u_{1,m}^n \quad \text{and} \quad u_{N_x+1,m}^n = -u_{N_x-1,m}^n \quad \forall m, \\ u_{l,-1}^n &= -u_{l,-1}^n \quad \text{and} \quad u_{l,N_y+1}^n = -u_{l,N_y-1}^n \quad \forall l. \end{aligned}$$

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}_Δ 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

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \quad (6.46)$$

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}_Δ .

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 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.

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.

FULL DOC
SWEEP: check
figure align-
ment

6.3. The thin plate

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.

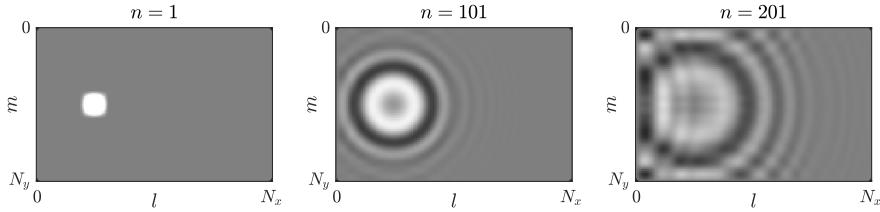


Fig. 6.10: Wave propagation of a thin plate with simply supported boundary conditions and parameters as shown in Table 6.1. The system is excited with a 2D raised cosine at $(x, y) = (0.25L_x, 0.5L_y)$ and dispersive effects are apparent.

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

$$(1 + \sigma_0 k)z + \left(16\mu^2(p_x + p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x + p_y) - 2 \right) + \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x + p_y) \right) z^{-1} = 0. \quad (6.47)$$

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

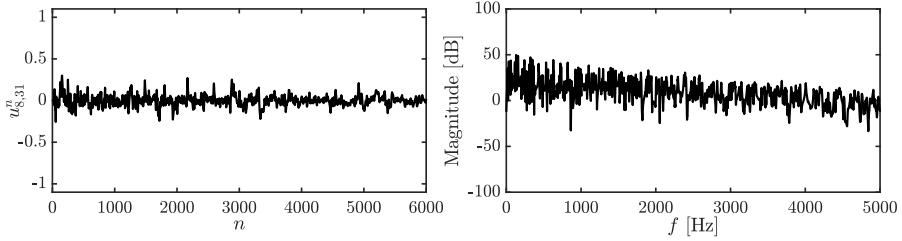


Fig. 6.11: The output of the thin plate at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.10.

are bounded by 1, the following can be written

$$4\mu^2(1+1)^2 + \frac{4\sigma_1 k}{h^2}(1+1) \leq 1$$

$$16\mu^2 + \frac{8\sigma_1 k}{h^2} \leq 1.$$

Finally, recalling the definition for κ in Eq. (6.41) solving for h yields

$$1 \geq \frac{16\kappa^2 k^2}{h^4} + \frac{8\sigma_1 k}{h^2},$$

$$h^4 - 8\sigma_1 k h^2 - 16\kappa^2 k^2 \geq 0,$$

$$h \geq \sqrt{\frac{8\sigma_1 k + \sqrt{(8\sigma_1 k)^2 + 64\kappa^2 k^2}}{2}},$$

$$h \geq \sqrt{\frac{8\sigma_1 k + 8\sqrt{\sigma_1^2 k^2 + \kappa^2 k^2}}{2}},$$

$$h \geq 2\sqrt{k \left(\sigma_1 + \sqrt{\sigma_1^2 + \kappa^2} \right)}, \quad (6.48)$$

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 $\delta_{t+\hbar}$

To obtain the rate of change of the total energy, one can take an inner product of the scheme in Eq. (6.40) with $(\delta_t u_{l,m}^n)$ over discrete (2D) domain $d =$

6.3. The thin plate

$\{0, \dots, N_x\} \times \{0, \dots, N_y\}$ to get

$$\begin{aligned}\delta_{t+}\mathfrak{h} &= \rho H \langle \delta_t u_{l,m}^n, \delta_{tt} u_{l,m}^n \rangle_d + D \langle \delta_t u_{l,m}^n, \delta_\Delta \delta_\Delta u_{l,m}^n \rangle_d \\ &\quad + 2\sigma_0 \rho H \langle \delta_t u_{l,m}^n, \delta_t u_{l,m}^n \rangle_d - 2\sigma_1 \rho H \langle \delta_t u_{l,m}^n, \delta_{t-} \delta_\Delta u_{l,m}^n \rangle_d = 0.\end{aligned}\quad (6.49)$$

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

$$\mathfrak{q} = 2\sigma_0 \rho H \|\delta_t u_{l,m}^n\|_d^2 - 2\sigma_1 \rho H \langle \delta_t u_{l,m}^n, \delta_{t-} \delta_\Delta u_{l,m}^n \rangle_d. \quad (6.50)$$

Expanding the stiffness term in Eq. (6.49) to

$$\begin{aligned}&D \langle \delta_t u_{l,m}^n, (\delta_{xx} + \delta_{yy}) \delta_\Delta u_{l,m}^n \rangle_d \\ \iff &D \left(\langle \delta_t u_{l,m}^n, \delta_{xx} \delta_\Delta u_{l,m}^n \rangle_d + \langle \delta_t u_{l,m}^n, \delta_{yy} \delta_\Delta u_{l,m}^n \rangle_d \right),\end{aligned}$$

one can perform summation by parts twice, using Eq. (3.16b) for both terms to get

$$D \left(\langle \delta_t \delta_{xx} u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{d}_x} + \langle \delta_t \delta_{yy} u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{d}_y} \right) + \mathfrak{b}.$$

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{d}_x and \underline{d}_y can be further reduced to \underline{d} and the terms can be combined as

$$D \langle \delta_t \delta_\Delta u_{l,m}^n, \delta_\Delta u_{l,m}^n \rangle_{\underline{d}} + \mathfrak{b},$$

FULL DOC
SWEEP: check
for SWcom-
ments

and using identities (3.17a) and (3.17b) a definition for the total energy can be found:

$$\begin{aligned}\mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho H}{2} \|\delta_{t-} u_{l,m}^n\|_d^2 \quad \text{and} \\ \mathfrak{v} &= \frac{D}{2} \langle \delta_\Delta u_{l,m}^n, e_{t-} \delta_\Delta u_{l,m}^n \rangle_{\underline{d}}.\end{aligned}\quad (6.51)$$

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 t is identical to its definition in Eq. (6.32), only the units for v will be checked here. Recalling that $D = EH^3/12(1 - \nu^2)$, which in units is $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$, yields

$$\begin{aligned} v &= \frac{D}{2} \langle \delta_{\Delta} u_{l,m}^n, e_t - \delta_{\Delta} u_{l,m}^n \rangle_{\underline{d}} \xrightarrow{\text{in units}} \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \cdot \text{m}^2 \cdot (\text{m}^{-2} \cdot \text{m}) \cdot (\text{m}^{-2} \cdot \text{m}) \\ &= \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \end{aligned}$$

and shows that v indeed has the correct units.

Step 4: Implementation

Figure 6.12 shows the energetic output of an implementation of the thin plate and shows that the energy is conserved.

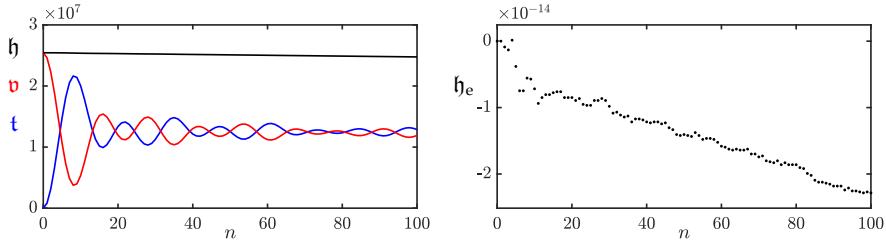


Fig. 6.12: The kinetic (blue), potential (red), and total (black) energy of an implementation of the thin plate are plotted in the left panel. Notice that the damping present in the system causes \mathfrak{h} to decrease. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

6.3.6 Modal analysis

Using the matrix form in Eq. (6.46), a modal analysis of the system can be performed using a one-step form described in Section 3.5.1.

Figure 6.13 shows the results of the analysis with parameter values as listed in Table 6.1. Although the modal frequencies follow a similar pattern to those of the 2D wave equation in Figure 6.8, the pattern is slightly more exponential like the stiff string in Figure 4.6.

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

look at this

6.4. The stiff membrane

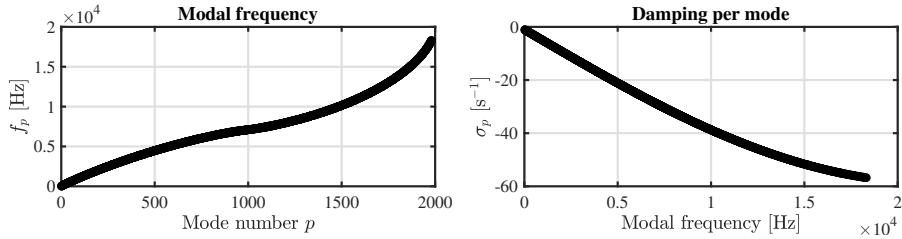


Fig. 6.13: The result of a modal analysis of the thin plate using the parameters in Table 6.1. Notice that the damping is plotted against modal frequency rather than mode number.

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 and modal analyses 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

$$\rho H \partial_t^2 u = T \Delta u - D \Delta \Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u, \quad (6.52)$$

where the parameters are identical to those in Eqs. (6.6) and (6.37).

6.4.2 Discrete time

Using familiar operators, Eq. (6.52) can be discretised to

$$\rho H \delta_{tt} u_{l,m}^n = T \delta_\Delta u_{l,m}^n - D \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_{t.} u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-} \delta_\Delta u_{l,m}^n, \quad (6.53)$$

or, using a more compact form after division by ρH , to

$$\delta_{tt} u_{l,m}^n = c^2 \delta_\Delta u_{l,m}^n - \kappa^2 \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_{t.} u_{l,m}^n + 2\sigma_1 \delta_{t-} \delta_\Delta u_{l,m}^n, \quad (6.54)$$

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

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{6.57}$$

with

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

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 3.5.1 and 6.2.4, the characteristic equation of the FD scheme in Eq. (6.54) can be obtained:

$$\begin{aligned}
 (1 + \sigma_0 k)z + & \left(4\lambda^2(p_x + p_y) + 16\mu^2(p_x + p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x + p_y) - 2 \right) \\
 & + \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x + p_y) \right) z^{-1} = 0.
 \end{aligned} \tag{6.58}$$

6.4. The stiff membrane

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

$$\begin{aligned} \lambda^2(1+1) + 4\mu^2(1+1)^2 + \frac{4\sigma_1 k}{h^2}(1+1) &\leq 1, \\ 2\lambda^2 + 16\mu^2 + \frac{8\sigma_1 k}{h^2} &\leq 1. \end{aligned}$$

Recalling the definitions for λ and μ from (6.56), one can solve for h

$$\begin{aligned} \frac{2c^2k^2}{h^2} + \frac{16\kappa^2k^2}{h^4} + \frac{8\sigma_1 k}{h^2} &\leq 1, \\ h^4 - (2c^2k^2 + 8\sigma_1 k)h^2 - 16\kappa^2k^2 &\geq 0, \\ h \geq \sqrt{\frac{2c^2k^2 + 8\sigma_1 k + \sqrt{(2c^2k^2 + 8\sigma_1 k)^2 + 64\kappa^2k^2}}{2}}, \\ h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \frac{1}{2}\sqrt{4(c^2k^2 + 4\sigma_1 k)^2 + 64\kappa^2k^2}}, \\ h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2k^2}}, \end{aligned} \tag{6.59}$$

and can be used as the stability condition for the stiff membrane.⁴

⁴This stability condition was wrong in paper [F]. It has been corrected here and included in Appendix A.

Chapter 6. 2D Systems

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 here. 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.

take an extra
look at these
intros

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)):

$$\partial_t^2 u = c^2 \partial_x^2 u \quad (7.1)$$

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))

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n. \quad (7.2)$$

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.

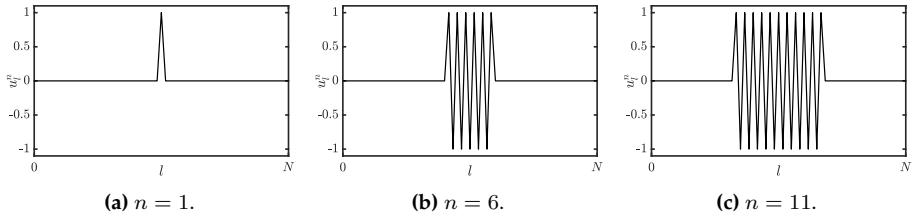


Fig. 7.1: The 1D wave equation initialised with an impulse at $l = \lfloor 0.5N \rfloor$.

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

$$e_{\text{rc}}(x) = \begin{cases} \frac{e_{\text{amp}}}{2} \left(1 - \cos \left(\frac{2\pi(x-x_s)}{x_w} \right) \right), & \text{if } x_s \leq x \leq x_e, \\ 0, & \text{otherwise,} \end{cases} \quad (7.3)$$

where

$$x_s = x_0 - \frac{x_w}{2}, \quad \text{and} \quad x_e = x_0 + \frac{x_w}{2} \quad (7.4)$$

are the start and end locations of the excitation. Furthermore, $x_s, x_e \in \mathcal{D}$, which puts a constraint on the width and location of the excitation.

In discrete time, 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 end locations of the raised cosine are

$$l_s = l_0 - \lfloor w/2 \rfloor \quad \text{and} \quad l_e = l_0 + \lfloor w/2 \rfloor, \quad (7.5)$$

7.1. Initial conditions

with $l_s, l_e \in d$ and, finally, $w = \lfloor x_w/h \rfloor$.¹ Equation (7.3) can then be discretised as

$$E_{l,rc} = \begin{cases} \frac{e_{amp}}{2} \left(1 - \cos \left(\frac{2\pi(l-l_s)}{w} \right) \right), & \text{if } l_s \leq l \leq l_e, \\ 0, & \text{otherwise.} \end{cases} \quad (7.6)$$

w does not have to be floored...

Figure 7.2 shows the 1D wave equation initialised with a raised cosine with $e_{amp} = 1$, $l_0 = \lfloor 0.5N \rfloor$ and $w = \lfloor 0.1N \rfloor$. Notice that the behaviour is much more smooth than the impulse shown in Figure 7.1.

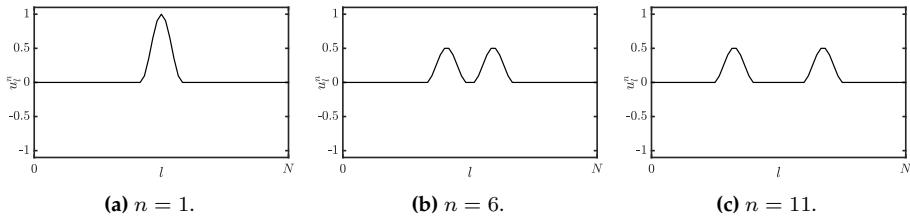


Fig. 7.2: The 1D wave equation initialised with a raised cosine at the center of the system.

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).

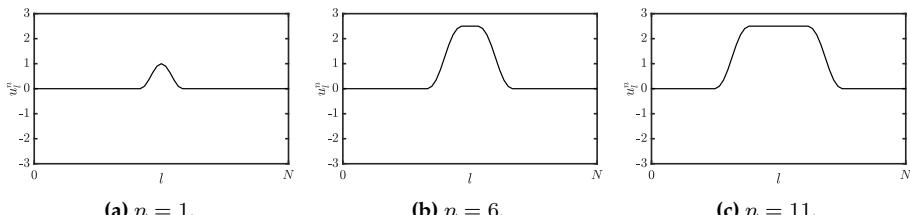


Fig. 7.3: The 1D wave equation initialised with a strike at the center of the system. Notice the scaling of the y-axis compared to Figure 7.2.

¹Notice that the width is given in ‘grid spacings’ and will thus affect $w + 1$ grid points. This is also why the range in Eq. (7.6) includes both end points.

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 triangular excitation can be defined as

$$e_{\text{tri}}(x) = \begin{cases} \frac{e_{\text{amp}}}{x_0} x, & \text{if } 0 \leq x \leq x_0, \\ \frac{e_{\text{amp}}}{x_0 - L}(x - L), & \text{if } x_0 < x \leq L. \end{cases} \quad (7.7)$$

In discrete time, Eq. (7.7) becomes

$$E_{l,\text{tri}} = \begin{cases} \frac{e_{\text{amp}}}{l_0} l, & \text{if } 0 \leq l \leq l_0, \\ \frac{e_{\text{amp}}}{l_0 - N}(l - N), & \text{if } l_0 < l \leq N. \end{cases} \quad (7.8)$$

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).

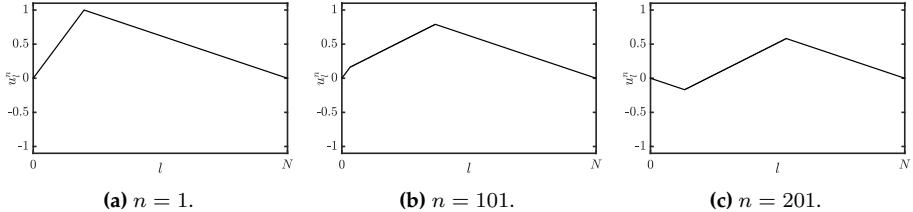


Fig. 7.4: The 1D wave equation initialised with a triangular pluck with $l_0 = \lfloor 0.2N \rfloor$. Note that sample rate has been set to $f_s = 441000$ Hz to show a more ideal triangular motion.

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

$$e_{\text{rc}}(x, y) = \begin{cases} \frac{e_{\text{amp}}}{2} \left(1 - \cos \left(\frac{2\pi(x - x_s)}{r_w} \right) \right) \left(1 - \cos \left(\frac{2\pi(y - y_s)}{r_w} \right) \right), & \text{if } x_s \leq x \leq x_e, \\ 0, & \text{and } y_s \leq y \leq y_e, \\ 0, & \text{otherwise,} \end{cases} \quad (7.9)$$

7.1. Initial conditions

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

$$x_s = x_0 - \frac{r_w}{2}, \quad x_e = x_0 + \frac{r_w}{2}, \quad y_s = y_0 - \frac{r_w}{2}, \quad \text{and} \quad y_e = y_0 + \frac{r_w}{2},$$

where (x_0, y_0) is the center coordinate and $x_s, x_e, y_s, y_e \in \mathcal{D}$ for a 2D domain \mathcal{D} .

In discrete time, Eq. (7.9) becomes

$$E_{(l,m),\text{rc}} = \begin{cases} \frac{e_{\text{amp}}}{2} \left(1 - \cos\left(\frac{2\pi(l-l_s)}{r}\right)\right) \left(1 - \cos\left(\frac{2\pi(m-m_s)}{r}\right)\right), & \text{if } l_s \leq l \leq l_e, \text{ and} \\ & m_s \leq m \leq m_e, \\ 0, & \text{otherwise,} \end{cases} \quad (7.10)$$

where $r = \lfloor r_w/h \rfloor$ is the discrete excitation radius (in ‘grid spacings’). Further- look at this

$$l_s = l_0 - \lfloor r/2 \rfloor, \quad l_e = l_0 + \lfloor r/2 \rfloor, \quad m_s = m_0 - \lfloor r/2 \rfloor, \quad \text{and} \quad m_e = m_0 + \lfloor r/2 \rfloor \quad (7.11)$$

for discrete center coordinate (l_0, m_0) and $l_s, l_e, m_s, m_e \in d$ 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.

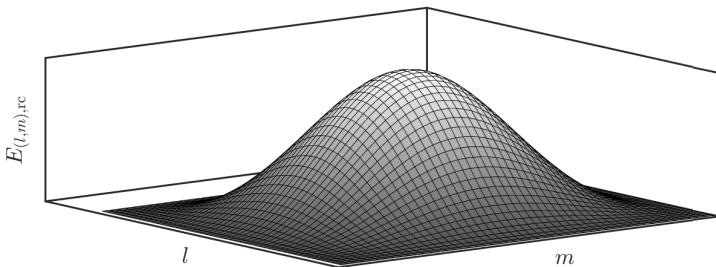


Fig. 7.5: A 2D raised cosine created using 7.10.

```
% Assuming Dirichlet boundary conditions and having initialised the
% following
% - center locations for the x and y-directions: 10 and m0
% - radius of the excitation r (in grid points)

ls = 10 - floor(r/2); % start location x-direction
le = 10 + 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))

$$\rho A \partial_t^2 u = T \partial_x^2 u + e(x) f(t) \quad (7.12)$$

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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n + E_l f^n \quad (7.13)$$

with discrete spatial distribution E_l and discrete time-varying external force f^n .

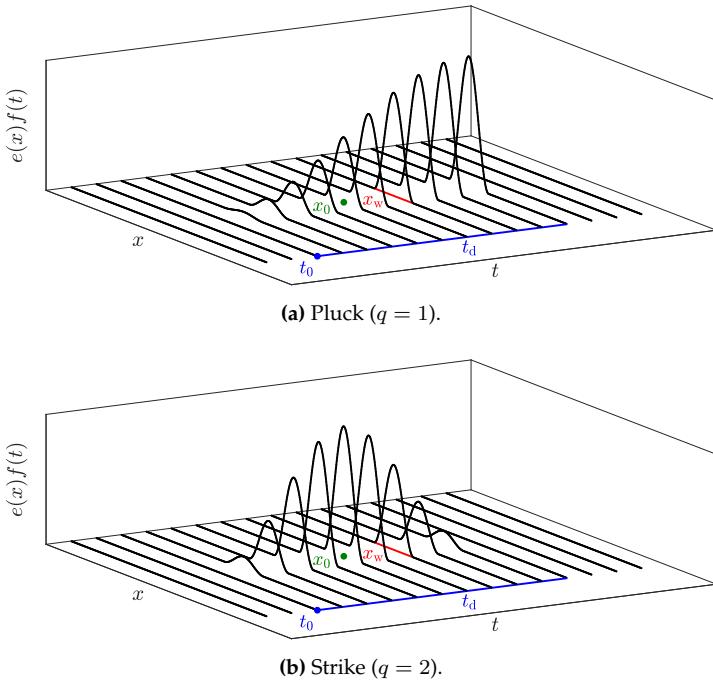


Fig. 7.6: The time-varying raised cosine showing a (a) pluck and a (b) strike. The location of excitation x_0 is shown in green, the width x_w in red and the excitation start t_0 and duration t_d in blue.

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 $t_d > 0$ (both in s), the temporal raised cosine can be used as a force function, as

$$f(t) = \begin{cases} \frac{f_{\text{amp}}}{2} \left(1 - \cos \left(\frac{q\pi(t-t_0)}{t_d} \right) \right), & t_s \leq t \leq t_0 + t_d \\ 0, & \text{otherwise.} \end{cases} \quad (7.14)$$

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 $e_{\text{amp}} = 1$ in Eq. (7.3).

In discrete time, the force function in Eq. (7.14) becomes

$$f^n = \begin{cases} \frac{f_{\text{amp}}}{2} \left(1 - \cos \left(\frac{q\pi(n-n_0)}{n_d} \right) \right), & n_0 \leq n \leq n_0 + n_d, \\ 0, & \text{otherwise.} \end{cases} \quad (7.15)$$

where $n_d = \lfloor t_d/k \rfloor$ 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 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.

```
%% 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.

pulse train figure placement

7.2. Time-varying excitations

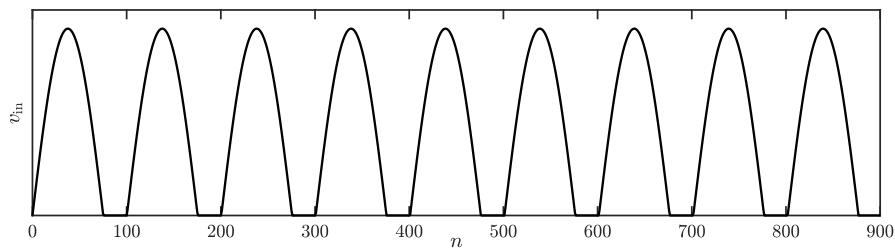


Fig. 7.7: The pulse train generated using Algorithm 7.2 ($f = 440$ Hz, duty cycle = 75%).

Chapter 7. Physically-Inspired Excitations

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 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

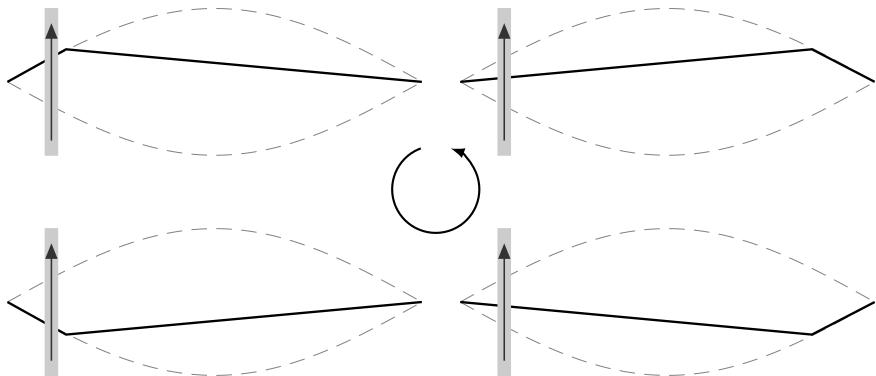


Fig. 8.1: Helmholtz motion. If the bow moves up on the left side of the string, the ‘Helmholtz corner’ travels anti-clockwise.

¹Also see <https://www.youtube.com/watch?v=6JeyiM0YNo4>

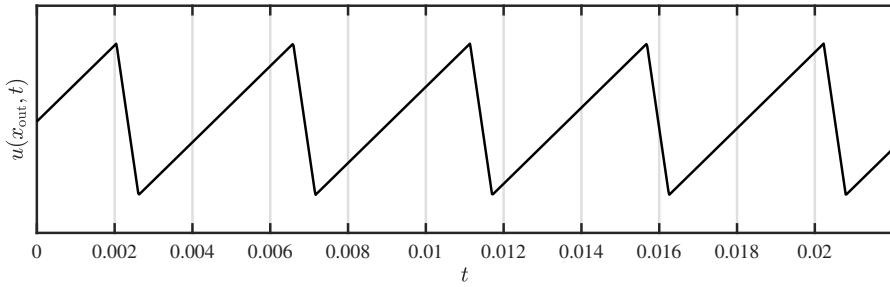


Fig. 8.2: An example of the (ideal) Helmholtz motion of a location x_{out} along a bowed string.

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* $I(x_i)$ can be introduced and can be 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:

$$I_{l,o}(x_i)u_l^n = \sum_{l \in d} I_{l,o}(x_i) \cdot u_l^n. \quad (8.1)$$

Here, o is the order of the operator, and $l \in d$ with discrete domain d , which needs to be the same for both $I_{l,o}(x_i)$ and u_l^n .

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

$$I_{l,0}(x_i) = \begin{cases} 1, & \text{if } l = l_i, \\ 0, & \text{otherwise,} \end{cases} \quad (8.2)$$

where the grid location of interest is defined as $l_i = \lfloor x_i/h \rfloor$ (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 neighbour, rather than using the flooring operation.

First-order or linear interpolation uses the fractional part of the flooring operation as well, according to $\alpha_i = x_i/h - l_i$ and is defined as

$$I_{l,1}(x_i) = \begin{cases} (1 - \alpha_i), & \text{if } l = l_i, \\ \alpha_i, & \text{if } l = l_i + 1, \\ 0 & \text{otherwise.} \end{cases} \quad (8.3)$$

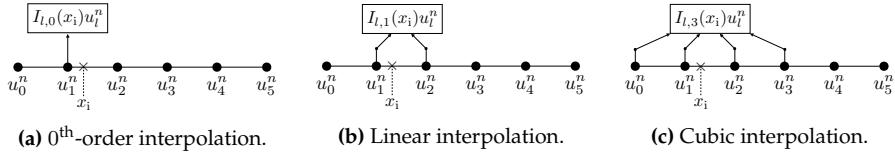


Fig. 8.3: Interpolation with varying orders of accuracy.

See Figure 8.3b.

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

$$I_{l,3}(x_i) = \begin{cases} -\alpha_i(\alpha_i - 1)(\alpha_i - 2)/6, & l = l_i - 1, \\ (\alpha_i - 1)(\alpha_i + 1)(\alpha_i - 2)/2, & l = l_i, \\ -\alpha_i(\alpha_i + 1)(\alpha_i - 2)/2, & l = l_i + 1, \\ \alpha_i(\alpha_i + 1)(\alpha_i - 1)/6, & l = l_i + 2, \\ 0, & \text{otherwise.} \end{cases} \quad (8.4)$$

See Figure 8.3c. Notice that the sum of all values of $I_{l,o}(x_i)$, regardless of the order or the value of α_i , add up to 1.

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 $\alpha_i = 0$, the higher-order interpolators reduce to the 0th-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 $J(x_i)$ 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 0th-order spreading operator

$$J_{l,0}(x_i) = \frac{1}{h} \begin{cases} 1, & \text{if } l = l_i, \\ 0, & \text{otherwise,} \end{cases} \quad (8.5)$$

a linear spreading operator

$$J_{l,1}(x_i) = \frac{1}{h} \begin{cases} (1 - \alpha_i), & \text{if } l = l_i, \\ \alpha_i, & \text{if } l = l_i + 1, \\ 0 & \text{otherwise,} \end{cases} \quad (8.6)$$

8.2. Interpolation and spreading operators

and a Lagrange cubic spreading operator

$$J_{l,3}(x_i) = \frac{1}{h} \begin{cases} -\alpha_i(\alpha_i - 1)(\alpha_i - 2)/6, & l = l_i - 1, \\ (\alpha_i - 1)(\alpha_i + 1)(\alpha_i - 2)/2, & l = l_i, \\ -\alpha_i(\alpha_i + 1)(\alpha_i - 2)/2, & l = l_i + 1, \\ \alpha_i(\alpha_i + 1)(\alpha_i - 1)/6, & l = l_i + 2, \\ 0, & \text{otherwise.} \end{cases} \quad (8.7)$$

Notice the scaling by $1/h$ which will be more elaborated on in Chapter 11. As is the case with the interpolation operators, higher-order spreading operators reduce to Eq. (8.5) if $\alpha_i = 0$.

Spreading operators approximate the spatial Dirac delta function $\delta(x - x_i)$ (in m^{-1}), which is a test function defined as

$$\delta(x) = \begin{cases} \infty, & x = 0, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(x) dx = 1, \quad (8.8)$$

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 :

$$\langle f_l, J_{l,o}(x_i) \rangle_d = I_{l,o}(x_i) f_l, \quad (8.9)$$

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):

$$\langle J_{l,o}(x_i), J_{l,o}(x_i) \rangle_d = \|J_{l,o}(x_i)\|_d^2 = I_{l,o}(x_i) J_{l,o}(x_i). \quad (8.10)$$

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 excitors 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

$$I_l = \frac{e_l}{\sum_d e_l} \quad \text{and} \quad J_l = \frac{1}{h} I_l. \quad (8.11)$$

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

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad (8.12)$$

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:

$$|x_{i+1} - x_i| < \epsilon, \quad (8.13)$$

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.

8.3. The Newton-Raphson method

```
% 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 $f(x) = e^x - 1$.

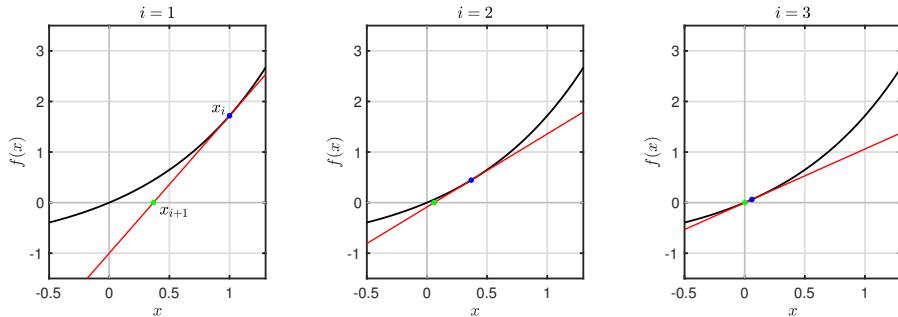


Fig. 8.4: The Newton-Raphson method. The x -value of the root of the tangent line at $f(x_i)$ is used to evaluate the next iteration.

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

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \underbrace{\begin{bmatrix} \frac{\partial f_1(\mathbf{x}_i)}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x}_i)}{\partial x_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x}_i)}{\partial x_1} & \dots & \frac{\partial f_M(\mathbf{x}_i)}{\partial x_M} \end{bmatrix}}_{\mathbf{J}(\mathbf{x}_i)}^{-1} \begin{bmatrix} f_1(\mathbf{x}_i) \\ \vdots \\ f_M(\mathbf{x}_i) \end{bmatrix}, \quad (8.14)$$

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 equations²:

$$f_1(\mathbf{x}) = 3x_1 - \cos(x_2 x_3) - 3/2 = 0, \quad (8.15a)$$

$$f_2(\mathbf{x}) = 4x_1^2 - 625x_2^2 + 2x_3 - 1 = 0, \quad (8.15b)$$

$$f_3(\mathbf{x}) = 20x_3 + e^{-x_1 x_2} + 9 = 0, \quad (8.15c)$$

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.

²Taken 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 $\text{sgn}(v_{\text{rel}})$ in their definition. In this project, only the following static friction model has been used [21]

$$\Phi(v_{\text{rel}}) = \sqrt{2av_{\text{rel}}} e^{-av_{\text{rel}}^2 + 1/2}, \quad (8.16)$$

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.

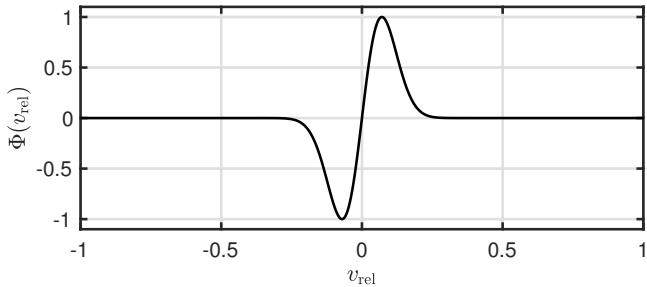


Fig. 8.5: The friction model in Eq. (8.16) with $a = 100$.

FULL DOC
SWEEP: check
figure center-
ing

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 $x_B = x_B(t) \in \mathcal{D}$ and the bow can then be described as

$$v_{\text{rel}} = \partial_t u(x_B, t) - v_B(t) \quad (8.17)$$

(in m/s) with bow velocity $v_B = v_B(t)$ (in m/s).

Recalling the PDE of the stiff string in Eq. (4.3)

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \quad (8.18)$$

one can add the bow force to the equation according to

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u - \delta(x - x_B) f_B \Phi(v_{\text{rel}}) \quad (8.19)$$

where spatial Dirac delta function $\delta(x - x_B)$ (in m^{-1}) (see Section 8.2) positions the bow along the string and $f_B = f_B(t) \geq 0$ is the bow force (in N).³

³If 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 friction⁴. 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 $\text{sgn}(\Phi(v_{\text{rel}})) = \text{sgn}(v_{\text{rel}})$ through Eq. (8.16) and $f_B \geq 0$, the term $\delta(x - x_B)f_B\Phi(v_{\text{rel}})$ 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

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_t.u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n - J_l(x_B^n)F_B^n\Phi(v_{\text{rel}}^n), \quad (8.20)$$

with $F_B^n = f_B^n/\rho A$ and spreading operator $J_l(x_B^n) = J_{l,o}(x_B^n)$ (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

$$v_{\text{rel}}^n = I_l(x_B^n)\delta_t.u_l^n - v_B^n, \quad (8.21)$$

with interpolation operator $I_l(x_B^n) = I_{l,o}(x_B^n)$ and is of the same order as $J_l(x_B^n)$. 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.

⁴The 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).

8.4. Static friction models

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 $J_l(x_B^n)$ 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

$$I_l(x_B^n)\delta_{tt}u_l^n = c^2 I_l(x_B^n)\delta_{xx}u_l^n - \kappa^2 I_l(x_B^n)\delta_{xxxx}u_l^n - 2\sigma_0 I_l(x_B^n)\delta_{t\cdot}u_l^n + 2\sigma_1 I_l(x_B^n)\delta_{t-\delta_{xx}}u_l^n - \|J_l(x_B^n)\|_d^2 F_B^n \Phi(v_{\text{rel}}^n). \quad (8.22)$$

One can rewrite Eq. (8.21) to

$$I_l(x_B^n)\delta_{t\cdot}u_l^n = v_{\text{rel}}^n + v_B^n, \quad (8.23)$$

and using identity (2.27a), Eq. (8.22) can be rewritten and assigned to a function $g(v_{\text{rel}}^n)$ according to

$$g(v_{\text{rel}}^n) = \left(\frac{2}{k} + 2\sigma_0 \right) v_{\text{rel}}^n + \|J_l(x_B^n)\|_d^2 F_B^n \Phi(v_{\text{rel}}^n) + b^n = 0, \quad (8.24)$$

check time varying x_B^n

where the terms not dependent on v_{rel} are collected in⁵

$$b^n = -\frac{2}{k} I_l(x_B^n)\delta_{t\cdot}u_l^n - c^2 I_l(x_B^n)\delta_{xx}u_l^n + \kappa^2 I_l(x_B^n)\delta_{xxxx}u_l^n + \left(\frac{2}{k} + 2\sigma_0 \right) v_B^n - 2\sigma_1 I_l(x_B^n)\delta_{t-\delta_{xx}}u_l^n. \quad (8.25)$$

One can then perform the Newton-Raphson method detailed in Section 8.3 to iteratively solve for v_{rel}

$$(v_{\text{rel}}^n)_{i+1} = (v_{\text{rel}}^n)_i - \frac{g((v_{\text{rel}}^n)_i)}{g'((v_{\text{rel}}^n)_i)}, \quad (8.26)$$

where

$$g'(v_{\text{rel}}^n) = \frac{2}{k} + 2\sigma_0 + \|J_l(x_B^n)\|_d^2 F_B^n \Phi'(v_{\text{rel}}^n),$$

and

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

⁵An 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 $I_{l,1}(x_i)\delta_{xx}u_l^n = (1 - \alpha_i)\delta_{xx}u_{l_i}^n + \alpha\delta_{xx}u_{l_i+1}^n$.

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

$$x_B^n = 1/8, \quad f_B^n = 1 \text{ N}, \quad v_B^n = 0.2 \text{ m/s}, \quad \text{and} \quad a = 100.$$

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.

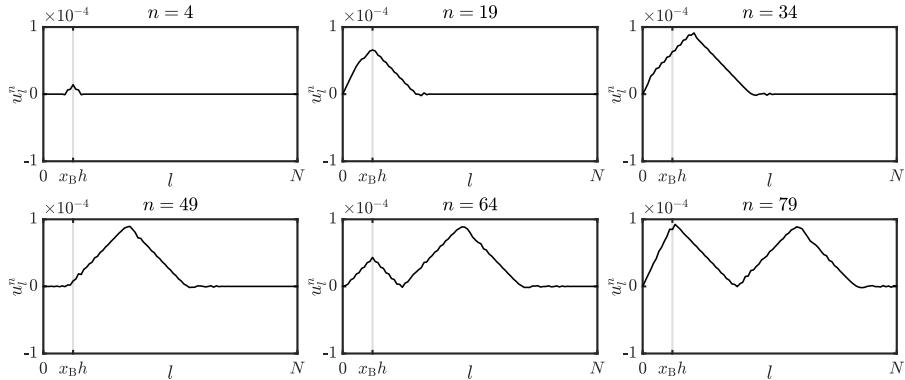


Fig. 8.6: Behaviour of a bowed stiff string at the start of the simulation.

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 $(\delta_t u_l^n)$ over

8.4. Static friction models

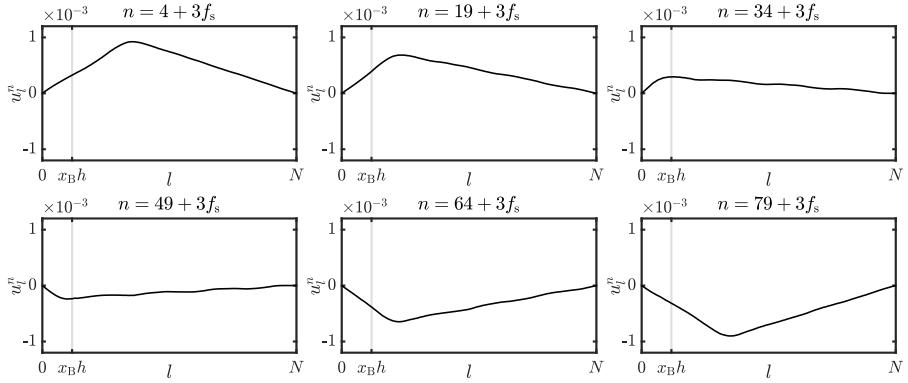


Fig. 8.7: Behaviour of a bowed stiff string simulation after 3 seconds. The string exhibits a Helmholtz motion as presented in Figure 8.1.

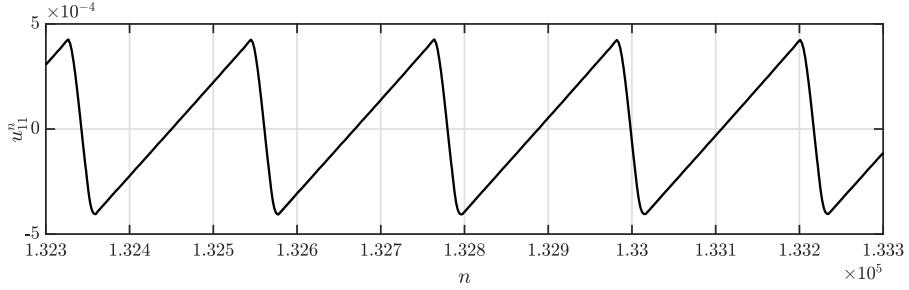


Fig. 8.8: Time domain output at the bowing location of a bowed stiff string using the static friction model in Eq. (8.16).

discrete domain d one arrives at the following

$$\begin{aligned}
 \delta_{t+} \mathfrak{h} + \mathfrak{q} &= -\langle (\delta_t u_l^n), J_l(x_B^n) f_B^n \Phi(v_{\text{rel}}^n) \rangle_d \\
 &\stackrel{\text{Eq. (8.9)}}{\iff} -I_l(x_B^n) (\delta_t u_l^n) f_B^n \Phi(v_{\text{rel}}^n) \\
 &\stackrel{\text{Eq. (8.23)}}{\iff} \underbrace{-f_B^n \Phi(v_{\text{rel}}^n) v_{\text{rel}}^n}_{\text{loss}} - \underbrace{f_B^n \Phi(v_{\text{rel}}^n) v_B^n}_{\text{power}},
 \end{aligned}$$

where \mathfrak{h} and \mathfrak{q} are as defined in Eqs. (4.29) and (4.28) respectively. As $\text{sgn}(\Phi(v_{\text{rel}}^n)) = \text{sgn}(v_{\text{rel}}^n)$ 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

$$\delta_{t+} \mathfrak{h} = -\mathfrak{q} - \mathfrak{q}_B - \mathfrak{p} \quad (8.27)$$

where

$$\mathfrak{q}_B = f_B^n \Phi(v_{\text{rel}}^n) v_{\text{rel}}^n \quad \text{and} \quad \mathfrak{p} = f_B^n \Phi(v_{\text{rel}}^n) v_B^n.$$

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.

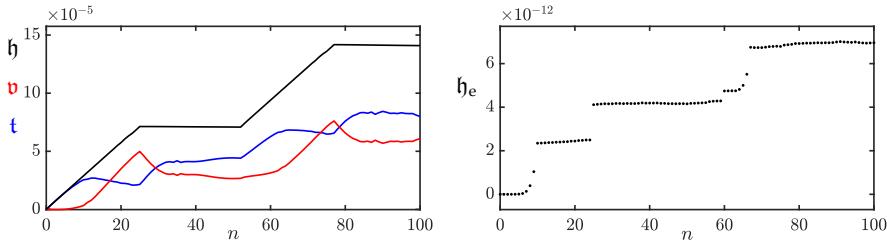


Fig. 8.9: The kinetic (blue), potential (red), and total (black) energy of the bowed stiff string. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

look at implementation

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.

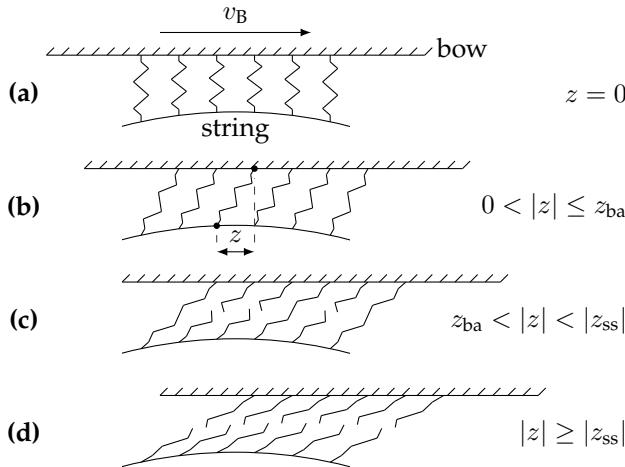


Fig. 8.10: A visualisation of the microscopic displacements of the bristles between the bow and the string assumed by the elasto-plastic friction model. The bow moves right with a velocity of v_B . (a) The average bristle displacement $z = 0$. (b) The bow moves right relative to the string and the purely 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 $|z| \geq |z_{ss}|$ 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 by Dupont et al. 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

$$f(v, z) = s_0 z + s_1 \dot{z} + s_2 v + s_3 w, \quad (8.28)$$

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 $x_B = x_B(t) \in \mathcal{D}$ and the bow is (similar to Eq. (8.17))

$$v = \partial_t u(x_B, t) - v_B, \quad (8.29)$$

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

$$\dot{z} = r(v, z) = v \left[1 - \alpha(v, z) \frac{z}{z_{ss}(v)} \right]. \quad (8.30)$$

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

$$z_{ss}(v) = \frac{\text{sgn}(v)}{s_0} \left[f_C + (f_S - f_C) e^{-(v/v_S)^2} \right], \quad (8.31)$$

where the v_S is the Stribeck velocity (in m/s). Furthermore, using the normal force $f_N = f_N(t)$ (in N), the Coulomb force and stiction force can be calculated according to $f_C = f_N \mu_C$ and $f_S = f_N \mu_S$ 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.

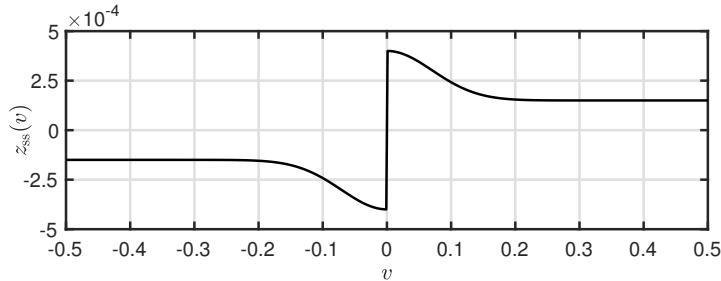


Fig. 8.11: The steady-state function $z_{ss}(v)$ plotted against relative velocity v with $s_0 = 10^4$, $\mu_C = 0.3$, $\mu_S = 0.8$, $v_S = 0.1$ and $f_N = 5$.

Finally, $\alpha(v, z)$ in Eq. (8.30) is an adhesion map between the bow and the string and is defined as

$$\alpha(v, z) = \begin{cases} 0 & |z| \leq z_{ba} \\ \alpha_m(v, z) & z_{ba} < |z| < |z_{ss}(v)| \\ 1 & |z| \geq |z_{ss}(v)| \end{cases} \quad \begin{array}{l} \text{if } \text{sgn}(v) = \text{sgn}(z) \\ \text{if } \text{sgn}(v) \neq \text{sgn}(z), \end{array} \quad (8.32)$$

8.5. Dynamic friction models

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

$$\alpha_m = \frac{1}{2} \left[1 + \text{sgn}(z) \sin \left(\pi \frac{z - \text{sgn}(z) \frac{1}{2} (|z_{ss}(v)| + z_{ba})}{|z_{ss}(v)| - z_{ba}} \right) \right], \quad (8.33)$$

where the break-away displacement $z_{ba} = z_{ba}(t) = 0.7f_C/s_0$ 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.

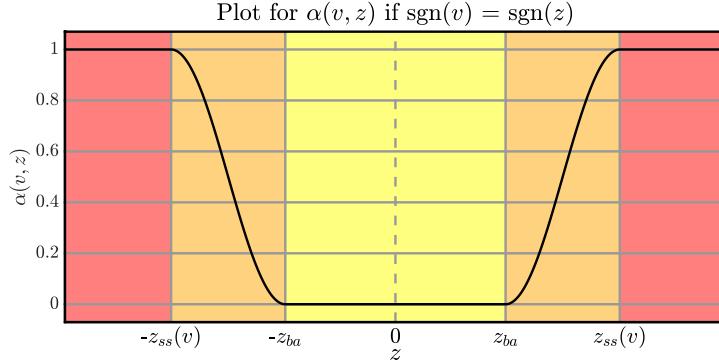


Fig. 8.12: A plot of the adhesion map $\alpha(v, z)$ in Eq. (8.32) plotted against z when $\text{sgn}(v) = \text{sgn}(z)$. The different coloured regions correspond to Figure 8.10 according to: yellow - a) & b), orange - c) and red - d). (Adapted from paper [C].)

Discrete time

Equation (8.28) can be discretised to

$$f(v^n, z^n) = s_0 z^n + s_1 r^n + s_2 v^n + s_3 w^n, \quad (8.34)$$

where Eq. (8.29) can be discretised to

$$v^n = I_l(x_B^n) \delta_t \cdot u_l^n - v_B^n, \quad (8.35)$$

with interpolation operator $I_l(x_B^n) = I_{l,o}(x_B^n)$ (of unspecified order o) and

$$r^n = r(v^n, z^n) = v^n \left[1 - \alpha(v^n, z^n) \frac{z^n}{z_{ss}(v^n)} \right] \quad (8.36)$$

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:

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa^2 \delta_{xxxx} u_l^n - 2\sigma_0 \delta_{t-} u_l^n + 2\sigma_1 \delta_{t-} \delta_{xx} u_l^n - J_l(x_B^n) \frac{f(v^n, z^n)}{\rho A}, \quad (8.37)$$

where spreading operator $J_l(x_B^n) = J_{l,o}(x_B^n)$ and is of the same order as $I_l(x_B^n)$ in Eq. (8.35). Following the same procedure as for the static friction model in Section 8.4, one takes an inner product with $J_l(x_B^n)$ 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

$$g_1(v^n, z^n) = \left(\frac{2}{k} + 2\sigma_0 \right) v^n + \|J_l(x_B^n)\|_d^2 \frac{f(v^n, z^n)}{\rho A} + b^n = 0, \quad (8.38)$$

where

$$\begin{aligned} b^n = & -\frac{2}{k} I_l(x_B^n) \delta_{t-} u_l^n - c^2 I_l(x_B^n) \delta_{xx} u_l^n + \kappa^2 I_l(x_B^n) \delta_{xxxx} u_l^n \\ & + \left(\frac{2}{k} + 2\sigma_0 \right) v_B^n - 2\sigma_1 I_l(x_B^n) \delta_{t-} \delta_{xx} u_l^n. \end{aligned}$$

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 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]

$$a^n = (\mu_{t-})^{-1} \delta_{t-} z^n \implies a^n = \frac{2}{k} (z^n - z^{n-1}) + a^{n-1}. \quad (8.39)$$

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:

$$g_2(v^n, z^n) = r^n - a^n = 0. \quad (8.40)$$

Finally, one can use the multivariate Newton-Raphson method described in Section 8.3.1, which results in

$$\begin{bmatrix} v^n \\ z^n \end{bmatrix}_{i+1} = \begin{bmatrix} v^n \\ z^n \end{bmatrix}_i - \begin{bmatrix} \frac{\partial g_1}{\partial v} & \frac{\partial g_1}{\partial z} \\ \frac{\partial g_2}{\partial v} & \frac{\partial g_2}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \quad (8.41)$$

The derivatives in the Jacobian matrix are given in Appendix F.6.

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 $f_N = f_B$). 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 elasticity of the bristles.

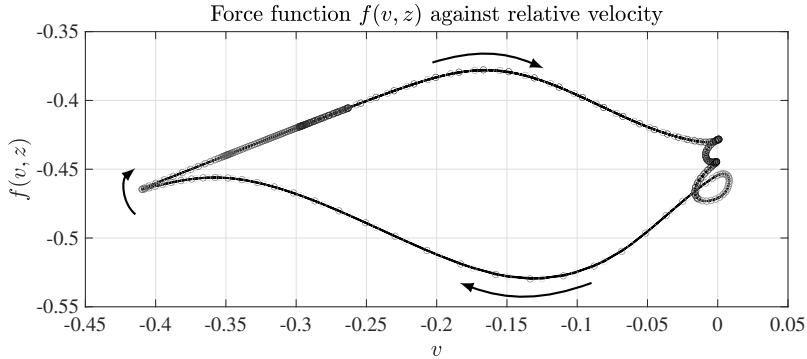


Fig. 8.13: Hysteresis loop showing 500 values up to $n = 3f_s$. (Adapted from paper [C].)

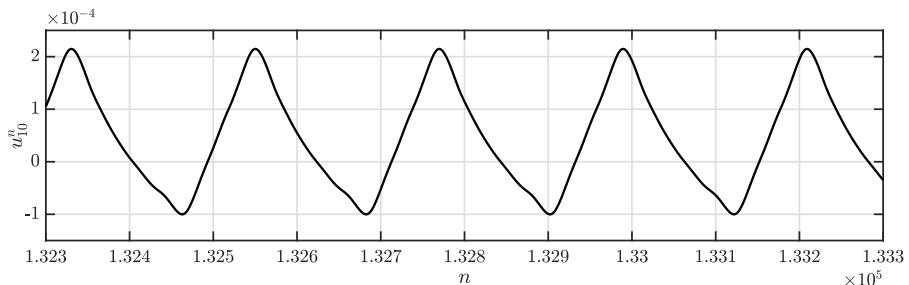


Fig. 8.14: Time domain output at the bow location of a stiff string bowed using an elasto-plastic friction model.

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 $\delta_t u_l^n$ over the string domain \mathcal{D} is taken to get

$$\delta_t + \mathfrak{h}_s + \mathfrak{q}_s = -\mathfrak{p}_B \quad (8.42)$$

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

$$\mathfrak{p}_B = \langle (\delta_t u_l^n), J(x_B^n) f^n \rangle_{\mathcal{D}}$$

which, using identity (8.9), can be written as

$$\mathfrak{p}_B = I_l(x_B^n) \delta_t u_l^n f^n.$$

Finally, using Eq. (8.35) yields

$$\mathfrak{p}_B = f^n v^n + f^n v_B^n. \quad (8.43)$$

The term $f^n v^n$ is the important one as $f^n v_B^n$ 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 $f v$ and ignoring the noise term $s_3 w^n$ for now, yields

$$\mathfrak{p}_B = f v = \sigma_0 z v + \sigma_1 r v + \sigma_2 v^2. \quad (8.44)$$

The definition for r^n in Eq. (8.36) may be rewritten as

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

8.5. Dynamic friction models

and (following Olsson) may be substituted in Eq. (8.44) as

$$\mathfrak{p}_B = s_0 z \left(r + \frac{v\alpha z}{z_{ss}(v)} \right) + s_1 r \left(r + \frac{v\alpha z}{z_{ss}(v)} \right) + s_2 v^2, \quad (8.45)$$

or

$$\mathfrak{p}_B = s_0 z r + s_1 \left(r + \frac{v\alpha z}{2z_{ss}(v)} \right)^2 + \frac{v\alpha z^2}{z_{ss}(v)} \left(s_0 - \frac{s_1 v\alpha}{4z_{ss}(v)} \right) + s_2 v^2. \quad (8.46)$$

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

$$\mathfrak{q}_{\text{brist}} = s_1 \left(r + \frac{v\alpha z}{2z_{ss}(v)} \right)^2 + \frac{v\alpha z^2}{z_{ss}(v)} \left(s_0 - \frac{s_1 v\alpha}{4z_{ss}(v)} \right) + s_2 v^2, \quad (8.47)$$

one can show that, as all coefficients are non-negative and $\text{sgn}(v) = \text{sgn}(z_{ss}(v))$ (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:

$$s_1 \leq \frac{4s_0 z_{ss}(v)}{v}. \quad (8.48)$$

This 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

$$\mathfrak{h}_{\text{brist}} = \frac{s_0}{2} z^n e_{t-} z^n, \quad (8.49)$$

which is not necessarily non-negative. In [98], Olsson performs the analysis in continuous time, where the energy in the bristles is defined as

$$\mathfrak{H}_{\text{brist}} = \frac{s_0}{2} z^2, \quad (8.50)$$

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

⁶This could possibly include a different discretisation of $s_0 z$ in Eq. (8.28) or adding a mass to the bristles.

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 apply an adjusted version 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.

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] 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:

$$M\ddot{u} = -Ku - R\dot{u}, \quad (9.1)$$

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:

$$M\delta_{tt}u^n = -Ku^n - R\delta_t.u^n. \quad (9.2)$$

Expanding and solving for u^{n+1} yields the following update equation:

$$\left(1 + \frac{Rk}{2M}\right)u^{n+1} = 2u^n - u^{n-1} - \frac{Kk^2}{M}u^n + \frac{Rk}{2M}u^{n-1}. \quad (9.3)$$

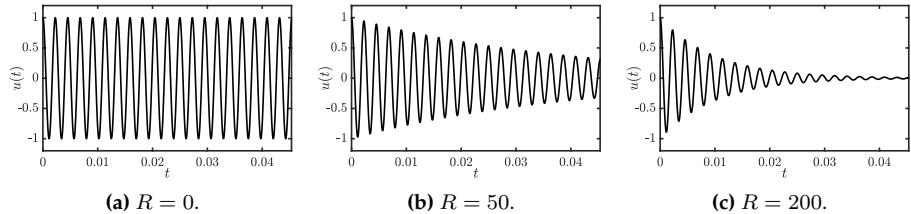


Fig. 9.1: The mass-spring-damper system in Eq. (9.1) with $f_0 = 440$ Hz for different values of R .

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 $(\delta_t.u^n)$ to get

$$M(\delta_{tt}u^n)(\delta_t.u^n) = -K(\delta_t.u^n)u^n - R(\delta_t.u^n)^2. \quad (9.4)$$

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

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

9.2. Continuous time

Using identities (3.17a) and (3.17b), \mathfrak{h} and \mathfrak{q} can be obtained from Eq. (9.4)

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-} u^n)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2} u^n e_{t-} u^n, \quad (9.5)$$

and

$$\mathfrak{q} = R(\delta_{t-} u^n)^2. \quad (9.6)$$

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).

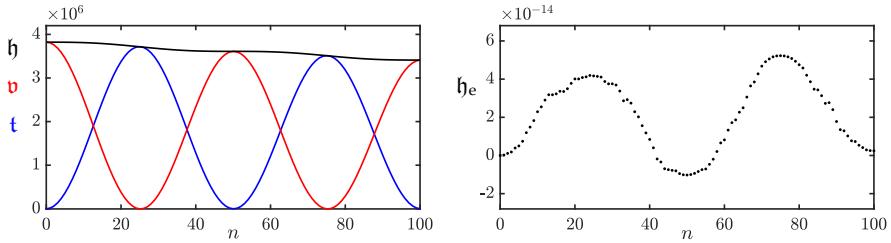


Fig. 9.2: The potential (red), kinetic (blue), and total (black) energy of the mass-spring-damper system. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

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

$$M\ddot{y} = -Ky - R\dot{y} + S_r\Delta p, \quad (9.7)$$

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 $S_r \geq 0$ (in m²). Furthermore,

$$\Delta p = \Delta p(t) = P_m - p(0, t) \quad (9.8)$$

is the difference between the pressure in the mouth $P_m = P_m(t)$ and the pressure at the left boundary of the acoustic tube $p(0, t)$ (all in Pa). The

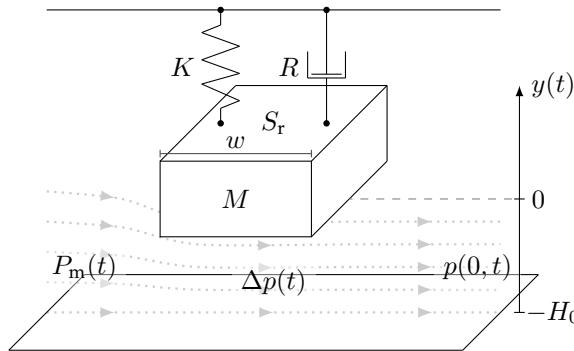


Fig. 9.3: Lip-reed system with parameters as appear in Section 9.2. (Adapted from paper [H].)

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 m³/s) and follows the Bernoulli equation

$$U_B = U_B(t) = w[y + H_0]_+ \operatorname{sgn}(\Delta p) \sqrt{\frac{2|\Delta p|}{\rho_0}}, \quad (9.9)$$

with effective lip-reed width w (in m), density of air ρ_0 (in kg/m³), 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 m³/s) is generated by the lip reed itself according to

$$U_r = U_r(t) = S_r \dot{y}, \quad (9.10)$$

and assuming that the volume flow velocity is conserved, the total air volume entering the acoustic tube at the left boundary is defined as

$$S(0)v(0, t) = U_B(t) + U_r(t). \quad (9.11)$$

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

$$\ddot{y} = -\omega_0^2 y - \sigma_r \dot{y} + \frac{S_r}{M} \Delta p, \quad (9.12)$$

9.3. Discrete time

with angular frequency of the lip reed $\omega_0 = \sqrt{K/M}$ (in rad/s) and loss parameter $\sigma_r = R/M$ (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 grid¹, and the equations presented above can be discretised to the following system:

$$\delta_{tt}y^{n+1/2} = -\omega_0^2\mu_t.y^{n+1/2} - \sigma_r\delta_t.y^{n+1/2} + \frac{S_r}{M}\Delta p^{n+1/2}, \quad (9.13a)$$

$$\Delta p^{n+1/2} = P_m - \mu_{t+}p_0^n, \quad (9.13b)$$

$$U_B^{n+1/2} = w[y^{n+1/2} + H_0]_+ \operatorname{sgn}(\Delta p^{n+1/2}) \sqrt{\frac{2|\Delta p^{n+1/2}|}{\rho_0}}, \quad (9.13c)$$

$$U_r^{n+1/2} = S_r\delta_t.y^{n+1/2}, \quad (9.13d)$$

$$\mu_{x-}(S_{1/2}v_{1/2}^{n+1/2}) = U_B^{n+1/2} + U_r^{n+1/2}. \quad (9.13e)$$

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

$$\alpha_r y^{n+3/2} = 4y^{n+1/2} + \beta_r y^{n-1/2} + \xi_r \Delta p^{n+1/2}, \quad (9.14)$$

where²

$$\alpha_r = 2 + \omega_0^2 k^2 + \sigma_r k, \quad \beta_r = \sigma_r k - 2 - \omega_0^2 k^2, \quad \text{and} \quad \xi_r = \frac{2S_r k^2}{M}. \quad (9.15)$$

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.

¹The variables are placed on the non-interleaved spatial grid, as the lip reed interacts with the boundary of the tube ($x = 0$).

²Notice 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

$$\frac{2}{k}(\delta_{t+} - \delta_{t-})y = -\omega_0^2(k\delta_{t+} + e_{t-})y - \sigma_r\delta_{t+}y + \frac{S_r}{M}\Delta p,$$

and, after grouping the terms,

$$a_1\delta_{t+}y - a_2\Delta p - a_3^n = 0, \quad (9.16)$$

where

$$a_1 = \frac{2}{k} + \omega_0^2 k + \sigma_r \geq 0, \quad a_2 = \frac{S_r}{M} \geq 0, \quad \text{and} \quad a_3^n = \left(\frac{2}{k}\delta_{t-} - \omega_0^2 e_{t-} \right) y. \quad (9.17)$$

Note that the non-negative 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)

$$\frac{a_1}{S_r}U_r - a_2\Delta p - a_3^n = 0,$$

and consequently Eq. (9.13e), to get

$$\frac{a_1}{S_r} \left(\mu_{x-}(S_{1/2}v_{1/2}) - U_B \right) - a_2\Delta p - a_3^n = 0. \quad (9.18)$$

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

$$\frac{\bar{S}_0}{\rho_0 c^2} \delta_{t+} p_0^n = -\delta_{x-}(S_{1/2}v_{1/2}). \quad (9.19)$$

Using identity (2.27d) for δ_{x-} and δ_{t+} , this can be rewritten to

$$\frac{2\bar{S}_0}{\rho_0 c^2 k} (\mu_{t+} p_0^n - p_0^n) = \frac{2}{h} \left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2} \right). \quad (9.20)$$

and substituting Eq. (9.13b) yields

$$\begin{aligned} \frac{2\bar{S}_0}{\rho_0 c^2 k} (P_m - \Delta p - p_0^n) &= \frac{2}{h} \left(\mu_{x-}(S_{1/2}v_{1/2}) - S_{1/2}v_{1/2} \right). \\ \mu_{x-}(S_{1/2}v_{1/2}) &= b_1^n - b_2 \Delta p \end{aligned} \quad (9.21)$$

9.3. Discrete time

where

$$b_1^n = S_{1/2}v_{1/2} + \frac{\bar{S}_0 h}{\rho_0 c^2 k} (P_m - p_0^n), \quad \text{and} \quad b_2 = \frac{\bar{S}_0 h}{\rho_0 c^2 k} \geq 0. \quad (9.22)$$

Final steps

Equations (9.21) and (9.13c) can be substituted into Eq. (9.18) to get

$$\begin{aligned} \frac{a_1}{S_r} \left(b_1^n - b_2 \Delta p - w[y + H_0]_+ \operatorname{sgn}(\Delta p) \sqrt{\frac{2|\Delta p|}{\rho_0}} \right) - a_2 \Delta p - a_3^n &= 0, \\ -w[y + H_0]_+ \operatorname{sgn}(\Delta p) \sqrt{\frac{2|\Delta p|}{\rho_0}} - b_2 \Delta p - \frac{a_2 S_r}{a_1} \Delta p + b_1^n - \frac{a_3^n S_r}{a_1} &= 0, \\ -c_1^n \operatorname{sgn}(\Delta p) \sqrt{|\Delta p|} - c_2 \Delta p + c_3^n &= 0, \end{aligned} \quad (9.23)$$

where

$$c_1^n = w[y + H_0]_+ \sqrt{\frac{2}{\rho_0}} \geq 0, \quad c_2 = b_2 + \frac{a_2 S_r}{a_1} \geq 0, \quad \text{and} \quad c_3^n = b_1^n - \frac{a_3^n S_r}{a_1}. \quad (9.24)$$

Equation (9.23) can be divided by $-\operatorname{sgn}(\Delta p)$ to get a quadratic equation in $\sqrt{|\Delta p|}$:

$$c_2 |\Delta p| + c_1^n \sqrt{|\Delta p|} - \frac{c_3^n}{\operatorname{sgn}(\Delta p)} = 0. \quad (9.25)$$

As $c_1^n, c_2 \geq 0$, the following must be true for any real solutions to exist

$$\operatorname{sgn}(c_3^n) = \operatorname{sgn}(\Delta p) \implies \frac{c_3^n}{\operatorname{sgn}(\Delta p)} = |c_3^n|, \quad (9.26)$$

and one can solve for $\sqrt{|\Delta p|}$:

$$\sqrt{|\Delta p|} = \frac{-c_1^n \pm \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2}. \quad (9.27)$$

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. (9.26), the definition for the pressure difference can be found:

$$\Delta p = \operatorname{sgn}(c_3^n) \left(\frac{-c_1^n + \sqrt{(c_1^n)^2 + 4c_2|c_3^n|}}{2c_2} \right)^2, \quad (9.28)$$

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

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0} \left(-2\mu_{x-}(S_{1/2}v_{1/2}) + 2S_{1/2}v_{1/2} \right). \quad (9.29)$$

Equation (9.13e) can then be substituted to get

$$p_0^{n+1} = p_0^n - \frac{\rho_0 c \lambda}{\bar{S}_0} \left(-2(U_B + U_r) + 2S_{1/2}v_{1/2} \right). \quad (9.30)$$

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 $P_m = 2000$ 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 $S(x) = 5 \cdot 10^{-5}$ m². 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

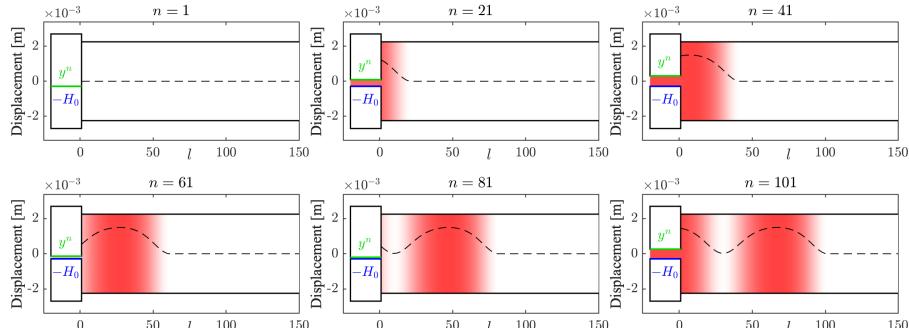


Fig. 9.4: A lip reed (shown at the left side of the plots) exciting a cylindrical acoustic tube. The y-axis refers to the displacement of the lips and the pressure in the tube p_l^n is shown in red and highlighted with a dashed line (not related to the y-axis).

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].

³This is similar behaviour to what the pulse train in Section 7.2.2 attempts to model.

9.4. Energy analysis

As all physical parameters need to be written out to obtain the correct units, Eq. (9.7) is discretised to get

$$M\delta_{tt}y^{n+1/2} = -K\mu_t.y^{n+1/2} - R\delta_t.y^{n+1/2} + S_r\Delta p^{n+1/2}, \quad (9.31)$$

and will be used in this analysis. Again, the superscript $n + 1/2$ will be suppressed for $y, \Delta p, U_B, U_r, S_{1/2}$ and $v_{1/2}$ for brevity.

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

Multiplying Eq. (9.31) by $(\delta_t.y)$, and moving all terms to the left-hand side, yields the rate of change of the energy in the lip reed \mathfrak{h}_r :

$$\delta_{t+}\mathfrak{h}_r = M(\delta_t.y)(\delta_{tt}y) + K(\delta_t.y)(\mu_t.y) + R(\delta_t.y)^2 - S_r(\delta_t.y)\Delta p = 0.$$

One can substitute Eqs (9.13d), and (9.13e) thereafter, to get

$$\begin{aligned} \delta_{t+}\mathfrak{h}_r &= M(\delta_t.y)(\delta_{tt}y) + K(\delta_t.y)(\mu_t.y) + R(\delta_t.y)^2 \\ &\quad - (\mu_{x-}(S_{1/2}v_{1/2}) - U_B)\Delta p = 0. \end{aligned}$$

Finally, substituting Eq. (9.13b), yields

$$\begin{aligned} \delta_{t+}\mathfrak{h}_r &= M(\delta_t.y)(\delta_{tt}y) + K(\delta_t.y)(\mu_t.y) + R(\delta_t.y)^2 \\ &\quad + U_B\Delta p - \mu_{x-}(S_{1/2}v_{1/2})(P_m - \mu_{t+}p_0^n) = 0. \end{aligned}$$

One can then include the tube by recalling that $\delta_{t+}\mathfrak{h}_t = -\mathfrak{b}_r + \mathfrak{b}_l$, and that the left boundary term is defined as (Eq. (5.53))

$$\mathfrak{b}_l = (\mu_{t+}p_0)\mu_{x-}(S_{1/2}v_{1/2}),$$

and substituting this (ignoring the right boundary term, i.e., $\mathfrak{b}_r = 0$) to get

$$\begin{aligned} \delta_{t+}(\mathfrak{h}_r + \mathfrak{h}_t) &= M(\delta_t.y)(\delta_{tt}y) + K(\delta_t.y)(\mu_t.y) + R(\delta_t.y)^2 \\ &\quad + U_B\Delta p - \mu_{x-}(S_{1/2}v_{1/2})P_m = 0. \end{aligned}$$

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

Using identities (3.17a) and (3.17d), the energy balance can be shown to be

$$\delta_{t+}(\mathfrak{h}_t + \mathfrak{h}_r) = -\mathfrak{q}_r - \mathfrak{p}_r, \quad (9.32)$$

where the energy of the tube \mathfrak{h}_t is as defined in Eq. (5.56) and the energy of the mass is

$$\mathfrak{h}_r = \mathfrak{t}_r + \mathfrak{v}_r, \quad \text{with} \quad \mathfrak{t}_r = \frac{M}{2}(\delta_{t-}y)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}\mu_{t-}(y^2). \quad (9.33)$$

Furthermore, the damping term is defined as

$$q_r = R(\delta_{t-}y)^2 + U_B \Delta p, \quad (9.34)$$

and the input power as

$$p_r = -(U_B + U_r)P_m. \quad (9.35)$$

It is interesting to note that due to the choice of discretisation of the lip reed, t_r , v_r and q_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

$$\begin{aligned} t_r &= \frac{M}{2}(\delta_{t-}y)^2 \xrightarrow{\text{in units}} \text{kg} \cdot (\text{s}^{-1} \cdot \text{m})^2 = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}, \\ v_r &= \frac{K}{2}\mu_{t-}(y^2) \xrightarrow{\text{in units}} \text{N} \cdot \text{m}^{-1} \cdot \text{m}^2 = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}, \end{aligned}$$

and have the correct units. Recalling that the damping and input power terms need to have units of $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$, writing the individual components of these terms in their respective units, yields

$$\begin{aligned} R(\delta_{t-}y)^2 &\xrightarrow{\text{in units}} \text{kg} \cdot \text{s}^{-1} \cdot (\text{s}^{-1} \cdot \text{m})^2 = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}, \\ U_B \Delta p &\xrightarrow{\text{in units}} \text{m}^3 \cdot \text{s}^{-1} \cdot \text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2} = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}, \\ -(U_B + U_r)P_m &\xrightarrow{\text{in units}} \text{m}^3 \cdot \text{s}^{-1} \cdot \text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2} = \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}, \end{aligned}$$

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 h^1 instead of h^0 in Eq. (3.38).

figure layout

9.4. Energy analysis

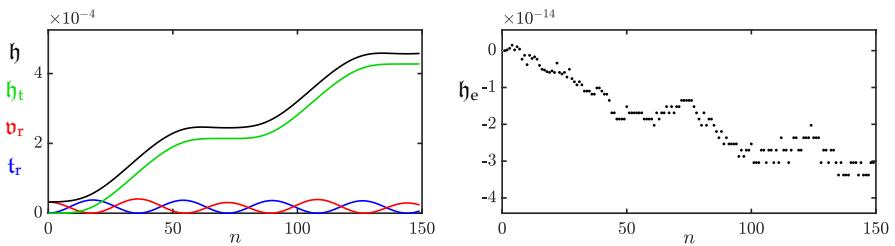


Fig. 9.5: The energy of the acoustic tube (green), the potential energy (red) the kinetic energy of the lip reed (blue), and the total energy (black) of the system corresponding to Figure 9.4. The right panel shows the normalised energy (according to Eq. (3.38), starting at $n = 1$) and shows that the deviation of the energy is within machine precision.

Chapter 9. The Lip Reed

Part IV

Interactions

Interactions

Part II introduced various resonators in isolation. Most musical instruments are composed of several individual systems which all interact with one another. The following chapters will describe different ways of interaction between individual resonators.

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 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 inter-penetrate, 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]), Duccheschi 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 Duccheschi'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 Duccheschi 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]

$$\phi(\eta) = \frac{K_c}{\alpha_c + 1} [\eta]_+^{\alpha_c + 1}, \quad (10.1)$$

with collision stiffness $K_c \geq 0$ (in N/m $^{\alpha_c}$) and dimensionless nonlinear collision coefficient $\alpha_c \geq 1$. Here $\eta = \eta(t)$ describes the relative displacement between the two colliding bodies (in m). The $[\cdot]_+$ operator, defined as

$$[\cdot]_+ = \frac{\cdot + |\cdot|}{2}, \quad (10.2)$$

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

$$\phi'(\eta) = K_c [\eta]_+^{\alpha_c} \quad (10.3)$$

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

$$\phi'(\eta) = \psi \psi' \quad \text{where} \quad \psi = \sqrt{2\phi} \quad \text{and} \quad \psi' = \frac{\dot{\psi}}{\dot{\eta}}, \quad (10.4)$$

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

$$M\ddot{u} = -\psi\psi', \quad (10.5)$$

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

$$M\ddot{u} = -\psi g, \quad (10.6a)$$

$$\dot{\psi} = g\dot{\eta}, \quad (10.6b)$$

$$\eta(t) = u(t) - b, \quad (10.6c)$$

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

$$\psi^{n-1/2} = \mu_{t-}\psi^n, \quad (10.7)$$

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

$$\psi \approx \mu_{t+}\psi^{n-1/2} \quad (10.8)$$

and

$$g \approx g^n = \frac{\delta_{t+}\psi^{n-1/2}}{\delta_{t-}\eta^n}, \quad (10.9)$$

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:

$$\delta_{t+}\psi^{n-1/2} = g^n \delta_{t-}\eta^n. \quad (10.10)$$

With the above approximations in place, system (10.6) can be discretised and yields the following system of equations:

$$M\delta_{tt}u^n = -\left(\mu_{t+}\psi^{n-1/2}\right)g^n, \quad (10.11a)$$

$$\delta_{t+}\psi^{n-1/2} = g^n \delta_{t-}\eta^n, \quad (10.11b)$$

$$\eta^n = u^n - b. \quad (10.11c)$$

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 $g^n \approx \psi'$ as per Eq. (10.9), the derivative can be computed analytically according to

$$g^n = \psi' \Big|_{\eta=\eta^n} \stackrel{\text{Eq. (10.4)}}{=} \frac{\phi'}{\sqrt{2\phi}} \Big|_{\eta=\eta^n}. \quad (10.12)$$

¹ $\delta_{t+}\psi^{n-1/2} \stackrel{\text{Eq. (10.9)}}{=} \delta_{t+}\mu_{t-}\psi^n \stackrel{\text{Eq. (2.27b)}}{=} \delta_{t-}\psi^n$ which is second-order accurate (see Section 2.2.2).

10.1. The mass – rigid barrier collision

Recalling (10.3) and (10.1), this can conveniently be rewritten to

$$g^n = \frac{K_c [\eta^n]_+^{\alpha_c}}{\sqrt{\frac{2K_c}{\alpha_c+1} [\eta^n]_+^{\alpha_c+1}}} = K_c \sqrt{\frac{\alpha_c+1}{2K_c}} [\eta^n]_+^{\alpha_c} [\eta^n]_+^{-\frac{(\alpha_c+1)}{2}} = \sqrt{\frac{K_c(\alpha_c+1)}{2}} [\eta^n]_+^{\frac{\alpha_c-1}{2}}. \quad (10.13)$$

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)

$$g_{\text{imp}}^n = 2 \frac{\psi^{n+1/2} - \psi^{n-1/2}}{\eta^{n+1} - \eta^{n-1}}. \quad (10.14)$$

If there is, however, no collision at $n + 1/2$, $\psi^{n+1/2} = 0$ and Eq. (10.14) reduces to

$$g_{\text{imp}}^n = -2 \frac{\psi^{n-1/2}}{\eta^{n+1} - \eta^{n-1}}.$$

Furthermore, due to the fact that there is no collision, η^{n+1} can be calculated according to $\eta^{n+1} = \eta^* = u^* - b$, where u^* 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

$$\frac{M}{k^2} (u^* - 2u^n + u^{n-1}) = 0 \implies u^* = 2u^n - u^{n-1}.$$

Thus, if there is no collision, g_{imp}^n can now be explicitly calculated from known values and be used in the definition for g^n according to [103]

$$g^n = \begin{cases} \kappa \sqrt{\frac{K_c(\alpha_c+1)}{2}} \cdot (\eta^n)^{\frac{\alpha_c-1}{2}}, & \text{if } \eta^n \geq 0, \\ -2 \frac{\psi^{n-1/2}}{\eta^* - \eta^{n-1}}, & \text{if } \eta^n < 0 \text{ and } \eta^* \neq \eta^{n-1}, \\ 0, & \text{if } \eta^n < 0 \text{ and } \eta^* = \eta^{n-1}. \end{cases} \quad (10.15a)$$

$$(10.15b) \quad (10.15c)$$

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

$$\mu_{t+}\psi^{n-1/2} = \frac{k}{2}g^n\delta_t.\eta^n + \psi^{n-1/2},$$

and inserting this into (10.11a), yields

$$M\delta_{tt}u^n = -\left(\frac{k}{2}g^n\delta_t.\eta^n + \psi^{n-1/2}\right)g^n. \quad (10.16)$$

As the position of barrier b is static, the following is true:

$$\frac{d\eta}{dt} = \frac{d}{dt}(u - b) \implies \delta_t.\eta^n = \delta_t.u^n, \quad (10.17)$$

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

$$\left(\frac{M}{k^2} + \frac{(g^n)^2}{4}\right)u^{n+1} = \frac{M}{k^2}(2u^n - u^{n-1}) + \frac{(g^n)^2}{4}u^{n-1} - \psi^{n-1/2}g^n, \quad (10.18)$$

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 $n + 1$:

$$\eta^{n+1} = u^{n+1} - b, \quad (10.19)$$

which is used to calculate $\psi^{n+1/2}$ by expanding and rewriting (10.11b) to

$$\psi^{n+1/2} = \psi^{n-1/2} + \frac{\eta^{n+1} - \eta^{n-1}}{2}. \quad (10.20)$$

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.

²Note that if the barrier was placed underneath the mass, making (10.11c) $\eta^n = b - u^n$, this would result in $\delta_t.\eta^n = -\delta_t.u^n$.

10.1. The mass – rigid barrier collision

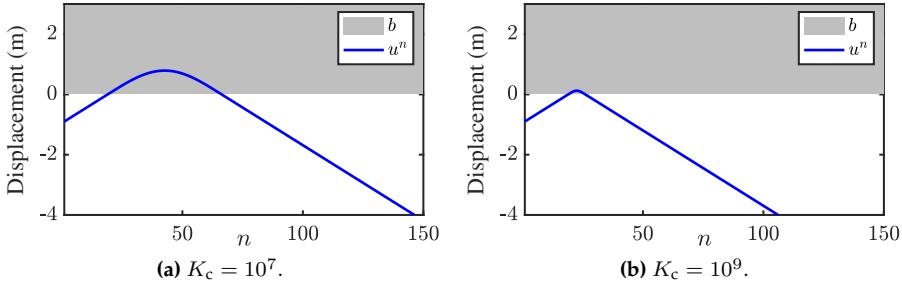


Fig. 10.1: The mass – rigid barrier collision over time with $\alpha_c = 1.3$ for different values of K_c .

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

$$\delta_{t+} h_m = - \left(\mu_{t+} \psi^{n-1/2} \right) g^n(\delta_t \cdot u^n)$$

where the energy of the mass is defined as (see Eq. (3.42))

$$h_m = \frac{M}{2} (\delta_{t-} u^n)^2. \quad (10.21)$$

Expanding g^n yields

$$\begin{aligned} \delta_{t+} h_m &= - \left(\mu_{t+} \psi^{n-1/2} \right) \frac{\delta_{t+} \psi^{n-1/2}}{\delta_t \eta^n} (\delta_t \cdot u^n) \\ &\stackrel{\text{Eq. (10.17)}}{=} - \left(\mu_{t+} \psi^{n-1/2} \right) \delta_{t+} \psi^{n-1/2}, \end{aligned}$$

which, using identity (3.17c), can be rewritten to

$$\delta_{t+} (h_m + h_c) = 0, \quad (10.22)$$

with collision energy

$$h_c = \frac{(\psi^{n-1/2})^2}{2}. \quad (10.23)$$

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.

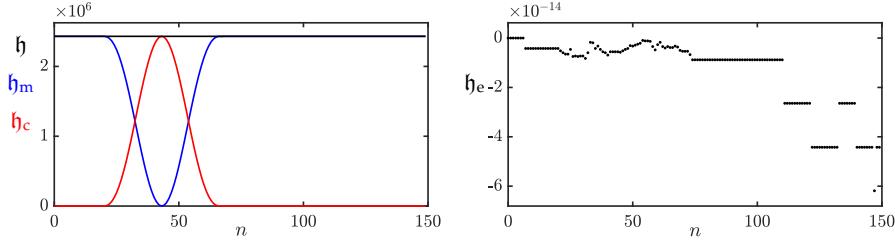


Fig. 10.2: The energy of the mass (blue), the collision (green) and the total energy (black) of the mass – rigid barrier collision. The energy corresponds to the behaviour in Figure 10.1a. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy 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:

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u + \delta(x - x_m) \psi g \quad (10.24a)$$

$$M \ddot{w} = -K w - \psi g \quad (10.24b)$$

$$\dot{\psi} = g \dot{\eta}, \quad (10.24c)$$

$$\eta(t) = w(t) - u(x_m, t), \quad (10.24d)$$

where spatial Dirac delta function $\delta(x - x_m)$ localises the mass (finger) along the string at location $x_m \in \mathcal{D}$ (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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n - EI \delta_{xxxx} u_l^n + J_l(x_m) (\mu_{t+} \psi^{n-1/2}) g^n, \quad (10.25a)$$

$$M \delta_{tt} w^n = -K w^n - (\mu_{t+} \psi^{n-1/2}) g^n, \quad (10.25b)$$

$$\delta_{t+} \psi^{n-1/2} = g^n \delta_t. \eta^n, \quad (10.25c)$$

$$\eta^n = w^n - I_l(x_m) u_l^n, \quad (10.25d)$$

where $l \in d$ with discrete domain $d = \{0, \dots, N\}$ and number of grid points along the string $N + 1$. Furthermore, spreading and interpolation operators $I_l(x_m) = I_{l,o}(x_m)$ and $J_l(x_m) = J_{l,o}(x_m)$ 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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n - EI \delta_{xxxx} u_l^n + J_l(x_m) \left(\frac{k}{2} g^n \delta_t. \eta^n + \psi^{n-1/2} \right) g^n, \quad (10.26a)$$

$$M \delta_{tt} w^n = -K w^n - \left(\frac{k}{2} g^n \delta_t. \eta^n + \psi^{n-1/2} \right) g^n, \quad (10.26b)$$

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 $J_l(x_m)$ over discrete domain d . Using identity (8.9) and dividing all terms by ρA yields

$$\begin{aligned} \delta_{tt} I_l(x_m) u_l^n &= c^2 I_l(x_m) \delta_{xx} u_l^n - \kappa^2 I_l(x_m) \delta_{xxxx} u_l^n \\ &\quad + \frac{\|J_l(x_m)\|_d^2}{\rho A} \left(\frac{k}{2} g^n \delta_t. \eta^n + \psi^{n-1/2} \right) g^n. \end{aligned}$$

with $c = \sqrt{T/\rho A}$ and $\kappa = \sqrt{EI/\rho A}$. Expanding the temporal FD operators yields

$$I_l(x_m) u_l^{n+1} = u^\star + \underbrace{\frac{\|J_l(x_m)\|_d^2 k^2}{\rho A}}_{\mathfrak{J}_l} \left(\frac{(g^n)^2}{4} (\eta^{n+1} - \eta^{n-1}) + \psi^{n-1/2} g^n \right), \quad (10.27)$$

where

$$u^\star = I_l(x_m) (2u_l^n - u_l^{n-1}) + c^2 k^2 I_l(x_m) \delta_{xx} u_l^n - \kappa^2 k^2 I_l(x_m) \delta_{xxxx} u_l^n$$

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 $\eta^{n+1} = w^{n+1} - I_l(x_m)u_l^{n+1}$, can be substituted into Eq. (10.27), which results in

$$\begin{aligned} \left(1 + \mathfrak{J}_l \frac{(g^n)^2}{4}\right) I_l(x_m)u_l^{n+1} - \mathfrak{J}_l \frac{(g^n)^2}{4} w^{n+1} &= u^* \\ &+ \mathfrak{J}_l \left(-\frac{(g^n)^2}{4} \eta^{n-1} + \psi^{n-1/2} g^n\right). \end{aligned} \quad (10.28)$$

Performing this same process on the FD scheme of the mass in Eq. (10.26b) yields

$$\begin{aligned} -\frac{(g^n)^2 k^2}{4M} I_l(x_m)u_l^{n+1} + \left(1 + \frac{(g^n)^2 k^2}{4M}\right) w^{n+1} &= w^* \\ -\frac{k^2}{M} \left(-\frac{(g^n)^2}{4} \eta^{n-1} + \psi^{n-1/2} g^n\right), \end{aligned} \quad (10.29)$$

where

$$w^* = 2w^n - w^{n-1} - \frac{Kk^2}{M} w^n$$

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 $I_l(x_m)u_l^{n+1}$ and w^{n+1} . Writing the aforementioned equations in matrix form yields

$$\begin{bmatrix} I_l(x_m)u_l^{n+1} \\ w^{n+1} \end{bmatrix} = \mathbf{A}^{-1} \mathbf{v} \quad (10.30)$$

where

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \left(1 + \mathfrak{J}_l \frac{(g^n)^2}{4}\right) & -\mathfrak{J}_l \frac{(g^n)^2}{4} \\ -\frac{(g^n)^2 k^2}{4M} & \left(1 + \frac{(g^n)^2 k^2}{4M}\right) \end{bmatrix} \quad \text{and} \\ \mathbf{v} &= \begin{bmatrix} u^* + \mathfrak{J}_l \left(-\frac{(g^n)^2}{4} \eta^{n-1} + \psi^{n-1/2} g^n\right) \\ w^* - \frac{k^2}{M} \left(-\frac{(g^n)^2}{4} \eta^{n-1} + \psi^{n-1/2} g^n\right) \end{bmatrix}. \end{aligned}$$

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.

10.2. Mass-spring – string collision

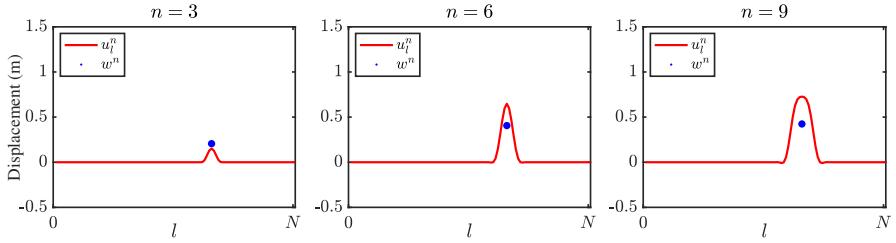


Fig. 10.3: The collision of the mass (blue) and the string (red). The collision model allows for interpenetration of the objects as shown in the left panel.

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

$$\delta_{t+} \mathfrak{h}_s = \left\langle (\delta_t \cdot u_l^n), J_l(x_m) \left(\mu_{t+} \psi^{n-1/2} \right) g^n \right\rangle_d \quad (10.31)$$

where the energy of the string is (see Eq. (4.29))

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

Energy analysis for the mass in Eq. (10.25b) can be done by multiplying the scheme by $(\delta_t \cdot w^n)$ to get

$$\delta_{t+} \mathfrak{h}_m = -(\delta_t \cdot w^n) \left(\mu_{t+} \psi^{n-1/2} \right) g^n, \quad (10.32)$$

where (see Eq. (3.42))

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

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:

$$\begin{aligned} \delta_{t+} (\mathfrak{h}_s + \mathfrak{h}_m) &= \left(I_l(x_m) (\delta_t \cdot u_l^n) - (\delta_t \cdot w^n) \right) \left(\mu_{t+} \psi^{n-1/2} \right) g^n, \\ &= \underbrace{\delta_t \cdot (I_l(x_m) u_l^n - w^n)}_{-\delta_t \cdot \eta^n} \left(\mu_{t+} \psi^{n-1/2} \right) g^n. \end{aligned}$$

Then, expanding g^n according to Eq. (10.9) yields

$$\begin{aligned}\delta_{t+}(\mathfrak{h}_s + \mathfrak{h}_m) &= -\delta_t \cdot \eta^n \left(\mu_{t+} \psi^{n-1/2} \right) \frac{\delta_{t+} \psi^{n-1/2}}{\delta_t \cdot \eta^n} \\ &= -\left(\mu_{t+} \psi^{n-1/2} \right) \delta_{t+} \psi^{n-1/2}\end{aligned}$$

which, using identity (3.17c), can be rewritten as

$$\delta_{t+}(\mathfrak{h}_s + \mathfrak{h}_m + \mathfrak{h}_c) = 0, \quad (10.33)$$

with collision energy

$$\mathfrak{h}_c = \frac{(\psi^{n-1/2})^2}{2}.$$

Again, the fact that ψ is squared here, means that \mathfrak{h}_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}_c at these times.

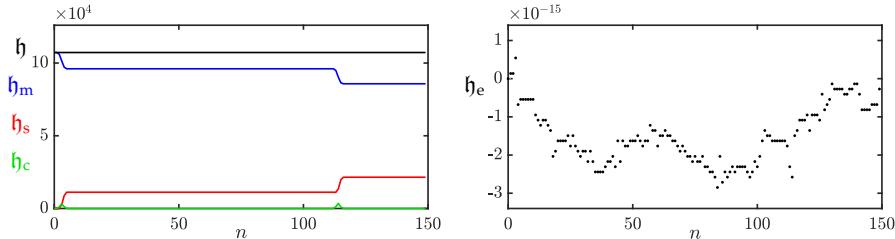


Fig. 10.4: The energy of the mass (blue), the string (red), the collision (green) and the total energy (black) of the mass-string collision. The energy corresponds to the system in Figure 10.3. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy 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

$$\phi(\eta) = \frac{K}{\alpha_c + 1} |\eta|^{\alpha_c + 1}, \quad (10.34)$$

and taking its derivative with respect to η yields

$$\phi'(\eta) = \text{sgn}(\eta) K |\eta|^{\alpha_c}. \quad (10.35)$$

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.

³The equation proposed in [65] contains a ‘dead zone’ that has been set to 0 here.

Chapter 10. Collisions

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 $l_i = \lfloor x_i/h \rfloor$ and $m_i = \lfloor y_i/h \rfloor$, a 0th-order interpolation operator $I_0(x_i, y_i) = I_{(l,m),0}(x_i, y_i)$ is defined as

$$I_0(x_i, y_i) = \begin{cases} 1, & \text{if } l = l_i \text{ and } m = m_i, \\ 0, & \text{otherwise.} \end{cases} \quad (11.1)$$

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 $\alpha_x = x_i/h - l_i$ and $\alpha_y = y_i/h - m_i$, a 2D linear interpolator $I_1(x_i, y_i) = I_{(l,m),1}(x_i, y_i)$ can then be composed as

$$I_1(x_i, y_i) = \begin{cases} (1 - \alpha_x)(1 - \alpha_y) & \text{if } l = l_i \text{ and } m = m_i, \\ (1 - \alpha_x)\alpha_y & \text{if } l = l_i \text{ and } m = m_i + 1, \\ \alpha_x(1 - \alpha_y) & \text{if } l = l_i + 1 \text{ and } m = m_i, \\ \alpha_x\alpha_y & \text{if } l = l_i + 1 \text{ and } m = m_i + 1, \\ 0, & \text{otherwise.} \end{cases} \quad (11.2)$$

Spreading operators are defined in the same way as in Section 8.2. A 0th-order spreading operator $J_0(x_i, y_i) = J_{(l,m),0}(x_i, y_i)$ can be defined as

$$J_0(x_i, y_i) = \frac{1}{h^2} \begin{cases} 1, & \text{if } l = l_i \text{ and } m = m_i, \\ 0, & \text{otherwise,} \end{cases} \quad (11.3)$$

as well as a linear spreading operator $J_1(x_i, y_i) = J_{(l,m),1}(x_i, y_i)$ as

$$J_1(x_i, y_i) = \frac{1}{h^2} \begin{cases} (1 - \alpha_x)(1 - \alpha_y) & \text{if } l = l_i \text{ and } m = m_i, \\ (1 - \alpha_x)\alpha_y & \text{if } l = l_i \text{ and } m = m_i + 1, \\ \alpha_x(1 - \alpha_y) & \text{if } l = l_i + 1 \text{ and } m = m_i, \\ \alpha_x\alpha_y & \text{if } l = l_i + 1 \text{ and } m = m_i + 1, \\ 0, & \text{otherwise.} \end{cases} \quad (11.4)$$

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

$$\delta(x, y) = \begin{cases} \infty, & x = y = 0, \\ 0, & \text{otherwise,} \end{cases} \quad \text{and} \quad \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \delta(x, y) dx dy = 1. \quad (11.5)$$

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.

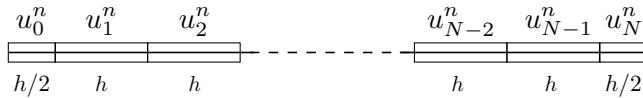


Fig. 11.1: Alternative interpretation of the discretisation of $u(x, t)$ to a grid function u_l^n . The continuous system is divided into $N - 1$ sections of length h plus 2 sections of length $h/2$ at the boundaries. Through this interpretation, the ‘weight’ of a grid point can be calculated from its 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 strings¹ 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 $x_c \in \mathcal{D}_u$ and $\chi_c \in \mathcal{D}_w$ yields the following system of PDEs:

$$\rho_u A_u \partial_t^2 u = T_u \partial_x^2 u - \delta(x - x_c)f, \quad (11.6a)$$

$$\rho_w A_w \partial_t^2 w = T_w \partial_\chi^2 w + \delta(\chi - \chi_c)f, \quad (11.6b)$$

¹Recall 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

$$\eta(t) = u(x_c, t) - w(\chi_c, t), \quad (11.7)$$

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

$$\delta_{tt} u_l^n = c_u^2 \delta_{xx} u_l^n - J_{l,u}(x_c) \frac{f^n}{\rho_u A_u}, \quad (11.8a)$$

$$\delta_{tt} w_m^n = c_w^2 \delta_{\chi\chi} w_m^n + J_{m,w}(\chi_c) \frac{f^n}{\rho_w A_w}, \quad (11.8b)$$

where $c_u = \sqrt{T_u / \rho_u A_u}$ and $c_w = \sqrt{T_w / \rho_w A_w}$. The spreading operators $J_{l,u}(x_c) = J_{l,o_u,u}(x_c)$ and $J_{l,w}(\chi_c) = J_{l,o_w,w}(\chi_c)$ are as defined in Section 8.2, and their orders o_u and o_w are left unspecified.

²Here, m is used for the spatial index of w_m^n to avoid double subscripts l_u and l_w .

11.3. Rigid connection

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 $J_{l,u}(x_c)$ and $J_{l,w}(\chi_c)$ over discrete domains $d_u = \{0, \dots, N_u\}$ and $d_w = \{0, \dots, N_w\}$ respectively. Using identity (8.9) one can write

$$I_{l,u}(x_c)\delta_{tt}u_l^n = c_u^2 I_{l,u}(x_c)\delta_{xx}u_l^n - \|J_{l,u}(x_c)\|_{d_u}^2 \frac{f^n}{\rho_u A_u}, \quad (11.9a)$$

$$I_{m,w}(\chi_c)\delta_{tt}w_m^n = c_w^2 I_{m,w}(\chi_c)\delta_{\chi\chi}w_m^n + \|J_{m,w}(\chi_c)\|_{d_w}^2 \frac{f^n}{\rho_w A_w}. \quad (11.9b)$$

Here, interpolation operators $I_{l,u}(x_c) = I_{l,o_u,u}(x_c)$ and $I_{m,w}(\chi_c) = I_{l,o_w,w}(\chi_c)$ 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:

$$u(x_c, t) = w(\chi_c, t), \quad (11.10)$$

which in discrete time becomes

$$I_{l,u}(x_c)u_l^n = I_{m,w}(\chi_c)w_m^n. \quad (11.11)$$

For a rigid connection, the following must also hold:

$$I_{l,u}(x_c)\delta_{tt}u_l^n = I_{m,w}(\chi_c)\delta_{tt}w_m^n. \quad (11.12)$$

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

$$c_u^2 I_{l,u}(x_c)\delta_{xx}u_l^n - \|J_{l,u}(x_c)\|_{d_u}^2 \frac{f^n}{\rho_u A_u} = c_w^2 I_{m,w}(\chi_c)\delta_{\chi\chi}w_m^n + \|J_{m,w}(\chi_c)\|_{d_w}^2 \frac{f^n}{\rho_w A_w},$$

which can be explicitly solved for f^n according to

$$f^n = \frac{c_u^2 I_{l,u}(x_c) \delta_{xx} u_l^n - c_w^2 I_{m,w}(\chi_c) \delta_{\chi\chi} w_m^n}{\frac{\|J_{l,u}(x_c)\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_c)\|_{d_w}^2}{\rho_w A_w}}. \quad (11.13)$$

This value can then be used in the update equation obtained after expanding system (11.8) as

$$u_l^{n+1} = (2 - 2\lambda_u^2) u_l^n + \lambda_u^2 (u_{l+1}^n + u_{l-1}^n) - u_l^{n-1} - J_{l,u}(x_c) \frac{k^2 f^n}{\rho_u A_u}, \quad (11.14a)$$

$$w_m^{n+1} = (2 - 2\lambda_w^2) w_m^n + \lambda_w^2 (w_{m+1}^n + w_{m-1}^n) - w_m^{n-1} + J_{m,w}(\chi_c) \frac{k^2 f^n}{\rho_w A_w}, \quad (11.14b)$$

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 $x_c = 0.25$ m and $\chi_c = 0.75$ 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.

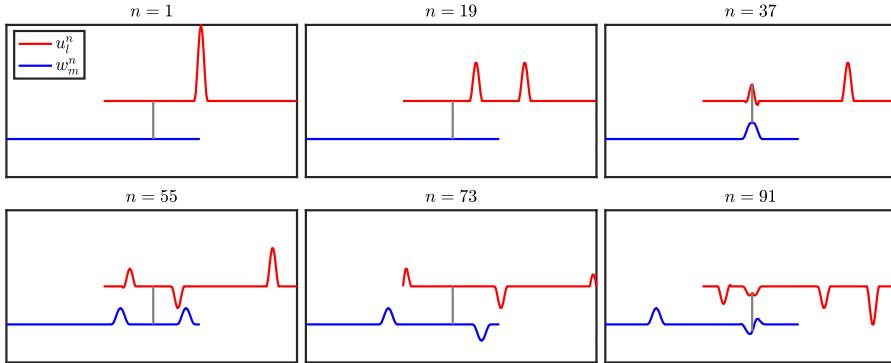


Fig. 11.2: The behaviour of two connected 1D wave equations. The systems are offset for clarity, but the relative displacement at the connection location is 0.

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.

11.3. Rigid connection

Recalling that $I_{l,u}(x_c) = I_{l,o_u,u}(x_c)$ and $I_{m,w}(\chi_c) = I_{l,o_w,w}(\chi_c)$, one can set the interpolation orders to 0, i.e., $o_u = o_w = 0$, to yield the following short-hand notations

$$I_{l,0,u}(x_c)u_l^n = u_{l_c}^n, \quad \text{and} \quad I_{m,0,w}(\chi_c)w_m^n = w_{m_c}^n, \quad (11.15)$$

where $l_c = \lfloor x_c/h_u \rfloor$ and $m_c = \lfloor \chi_c/h_w \rfloor$,

$$\|J_{l,0,u}(x_c)\|_{d_u}^2 = \frac{1}{h_u} \quad \text{and} \quad \|J_{m,0,w}(\chi_c)\|_{d_w}^2 = \frac{1}{h_w}. \quad (11.16)$$

This simplifies Eqs. (11.9) to

$$\delta_{tt}u_{l_c}^n = c_u^2\delta_{xx}u_{l_c}^n - \frac{f^n}{\rho_u A_u h_u}, \quad (11.17a)$$

$$\delta_{tt}w_{m_c}^n = c_w^2\delta_{\chi\chi}w_{m_c}^n + \frac{f^n}{\rho_w A_w h_w}, \quad (11.17b)$$

which, after rewriting Eq. (11.12) to

$$\delta_{tt}u_{l_c}^n = \delta_{tt}w_{m_c}^n, \quad (11.18)$$

one can solve for f^n , yielding the following simplified form of Eq. (11.13)

$$f^n = \frac{c_u^2\delta_{xx}u_{l_c}^n - c_w^2\delta_{\chi\chi}w_{m_c}^n}{\frac{1}{\rho_u A_u h_u} + \frac{1}{\rho_w A_w h_w}}. \quad (11.19)$$

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 $(\delta_t u_l^n)$ over discrete domain d_u , to get

$$\delta_{t+}\mathfrak{h}_u = \langle \delta_t u_l^n, -J_{l,u}(x_c) f^n \rangle_{d_u}, \quad (11.20)$$

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 $(\delta_t w_m^n)$ over discrete domain d_w to get

$$\delta_{t+}\mathfrak{h}_w = \langle \delta_t w_m^n, J_{m,w}(\chi_c) f^n \rangle_{d_w}, \quad (11.21)$$

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

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -I_{l,u}(x_c)\delta_t.u_l^n f^n + I_{m,w}(\chi_c)\delta_t.w_m^n f^n. \quad (11.22)$$

Finally, due to the rigid connection in Eq. (11.11), $I_{l,u}(x_c)\delta_t.u_l^n = I_{m,w}(\chi_c)\delta_t.w_m^n$ (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

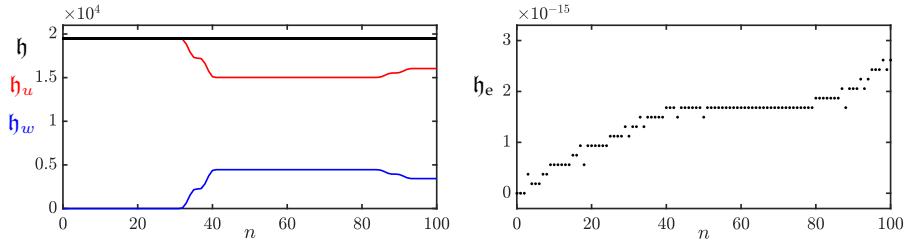


Fig. 11.3: The energy of u (red), the energy of w (blue), and the total (black) energy of connected 1D wave equations in (11.8). The energy corresponds to Figure 11.2. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

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

$$u_l^{n+1} = 2u_l^n - u_l^{n-1} + c^2 k^2 \delta_{xx} u_l^n - J_{l,u}(x_c) \frac{k^2}{\rho_u A_u} \left(\frac{c_u^2 I_{l,u}(x_c) \delta_{xx} u_l^n - c_w^2 I_{m,w}(\chi_c) \delta_{\chi\chi} w_m^n}{\frac{\|J_{l,u}(x_c)\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_c)\|_{d_w}^2}{\rho_w A_w}} \right), \quad (11.23a)$$

$$w_m^{n+1} = 2w_m^n - w_m^{n-1} + c^2 k^2 \delta_{xx} w_m^n + J_{m,w}(\chi_c) \frac{k^2}{\rho_w A_w} \left(\frac{c_w^2 I_{l,u}(x_c) \delta_{xx} u_l^n - c_w^2 I_{m,w}(\chi_c) \delta_{\chi\chi} w_m^n}{\frac{\|J_{l,u}(x_c)\|_{d_u}^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_c)\|_{d_w}^2}{\rho_w A_w}} \right). \quad (11.23b)$$

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):

$$\begin{bmatrix} \mathbf{u}^{n+1} \\ \mathbf{w}^{n+1} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix} - \begin{bmatrix} \mathbf{u}^{n-1} \\ \mathbf{w}^{n-1} \end{bmatrix}, \quad (11.24)$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_w \end{bmatrix} + \begin{bmatrix} -\mathbf{j}_u \\ \mathbf{j}_w \end{bmatrix} [\mathbf{f}_u \quad -\mathbf{f}_w].$$

The matrix in the definition of \mathbf{B} contains the operations of the 1D wave equation (also see Eq. (3.5)),

$$\mathbf{B}_u = 2\mathbf{I}_{N_u-1} + c_u^2 k^2 (\mathbf{D}_{xx})_u, \quad \text{and} \quad \mathbf{B}_w = 2\mathbf{I}_{N_w-1} + c_w^2 k^2 (\mathbf{D}_{xx})_w, \quad (11.25)$$

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 $J_{l,u}(x_c)$ and $J_{m,w}(\chi_c)$ 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 $I_{l,u}(x_c)$ and $I_{m,w}(\chi_c)$ respectively. Here, $\mathbf{i}_u \mathbf{j}_u$ and $\mathbf{i}_w \mathbf{j}_w$ are matrix-vector forms of $\|J_{l,u}(x_c)\|_d^2$ and $\|J_{m,w}(\chi_c)\|_d^2$

check with stefan if this is a correct notation for this

respectively (see Eq. (8.10)) and reduce to a scalar.

Equation (11.24) can then easily be rewritten in 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

$$f = f(t) = K\eta, \quad (11.26)$$

where $K \geq 0$ is the spring constant (in N/m) and

$$\eta = \eta(t) = u(x_c, t) - w(\chi_c, t) \quad (11.27)$$

is the relative displacement between the two systems at their respective connection locations (in m).⁴

In discrete time, Eq. (11.26) becomes

$$f^n = K\mu_t \cdot \eta^n, \quad (11.28)$$

where

$$\eta^n = I_{l,u}(x_c)u_l^n - I_{m,w}(\chi_c)w_m^n. \quad (11.29)$$

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.

³As the system does not exhibit damping, it could be analysed directly, not using a one-step form.

⁴Note 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 $u(x_c, t)$ would have been subtracted from $w(\chi_c, t)$ 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 $n + 1$:

$$I_{l,u}(x_c)u_l^{n+1} = u^* - \|J_{l,u}(x_c)\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u}, \quad (11.30a)$$

$$I_{m,w}(\chi_c)w_m^{n+1} = w^* + \|J_{m,w}(\chi_c)\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w}, \quad (11.30b)$$

where

$$u^* = I_{l,u}(x_c)(2u_l^n - u_l^{n-1}) + c_u^2 k^2 I_{l,u}(x_c) \delta_{xx} u_l^n,$$

and

$$w^* = I_{m,w}(\chi_c)(2w_m^n - w_m^{n-1}) + c_w^2 k^2 I_{m,w}(\chi_c) \delta_{xx} w_m^n,$$

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

$$\eta^{n+1} = I_{l,u}(x_c)u_l^{n+1} - I_{m,w}(\chi_c)w_m^{n+1},$$

into which Eqs. (11.30) can be substituted, as

$$\eta^{n+1} = u^* - \|J_{l,u}(x_c)\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u} - \left(w^* + \|J_{m,w}(\chi_c)\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w} \right). \quad (11.31)$$

A second definition for η^{n+1} can be obtained after expanding Eq. (11.28):

$$\eta^{n+1} = \frac{2f^n}{K} - \eta^{n-1} \quad (11.32)$$

and can be substituted into Eq. (11.31) to get

$$\frac{2f^n}{K} - \eta^{n-1} = u^* - \|J_{l,u}(x_c)\|_{d_u}^2 \frac{k^2 f^n}{\rho_u A_u} - \left(w^* + \|J_{m,w}(\chi_c)\|_{d_w}^2 \frac{k^2 f^n}{\rho_w A_w} \right). \quad (11.33)$$

Finally, one can group the terms for f^n

$$\left(\frac{2}{K} + \frac{\|J_{l,u}(x_c)\|_{d_u}^2 k^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_c)\|_{d_w}^2 k^2}{\rho_w A_w} \right) f^n = u^* - w^* + \eta^{n-1}$$

and solve for the force, solely based on known values of the system

$$f^n = \frac{u^* - w^* + \eta^{n-1}}{\frac{2}{K} + \frac{\|J_{l,u}(x_c)\|_{d_u}^2 k^2}{\rho_u A_u} + \frac{\|J_{m,w}(\chi_c)\|_{d_w}^2 k^2}{\rho_w A_w}}. \quad (11.34)$$

Figure 11.4 shows the behaviour of system Eq. (11.8), connected with a spring with spring constant $K = 5 \cdot 10^4 \text{ N/m}$. 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.

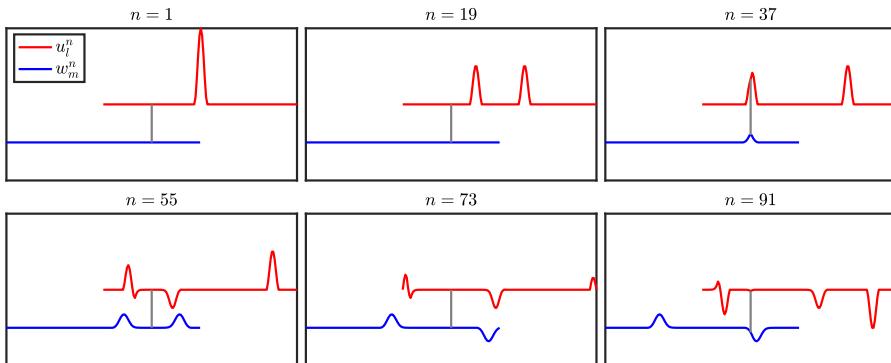


Fig. 11.4: The behaviour of two ideal strings connected using a spring. The systems are offset for clarity, but the relative displacement at the connection location at the start of the simulation is 0.

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):

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -I_{l,u}(x_c)\delta_t.u_l^n f^n + I_{m,w}(\chi_c)\delta_t.w_m^n f^n,$$

which can be rewritten to

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -\delta_t. (I_{l,u}(x_c)u_l^n - I_{m,w}(\chi_c)w_m^n) f^n.$$

One can then substitute the definitions for f^n and η^n from Eqs. (11.28) and

11.4. Spring connection

(11.29) to get

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w) = -K(\delta_t \cdot \eta^n)(\mu_t \cdot \eta^n), \quad (11.35)$$

which, using identity (3.17d), can be rewritten to

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w + \mathfrak{h}_c) = 0, \quad (11.36)$$

where

$$\mathfrak{h}_c = \frac{K}{2} (\mu_{t-}(\eta^n)^2) \quad (11.37)$$

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.

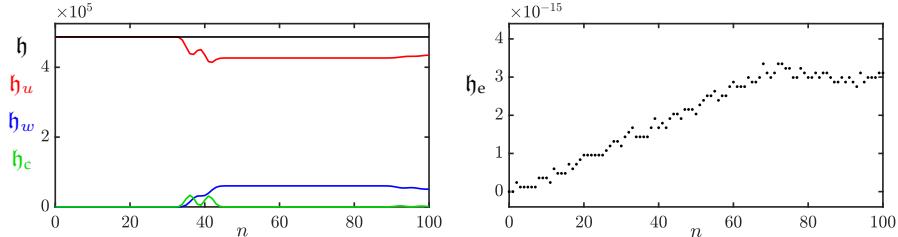


Fig. 11.5: The energy of u (red), the energy of w (blue), the energy of the spring connection (green) and the total energy (black) of the system of connected 1D wave equations in (11.8). The energy corresponds to the system in Figure 11.4. The right panel shows the normalised energy (according to Eq. (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

$$\delta_{t+}(\mathfrak{h}_u + \mathfrak{h}_w + \mathfrak{h}_c) = 0,$$

where

$$\mathfrak{h}_c = \frac{K}{2} (\eta^n e_{t-} - \eta^n).$$

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 $\chi \in \mathcal{D}_s$ where domain $\mathcal{D}_s = [0, L]$. Its PDE is described by

$$\rho_s A \partial_t^2 u = T \partial_{\chi\chi} u - E_s I \partial_{\chi\chi\chi} u - 2\sigma_{0,s} \rho_s A \partial_t u + 2\sigma_{1,s} \rho_s A \partial_t \partial_{\chi\chi} u, \quad (11.38)$$

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

$$\rho_p H \partial_t^2 w = -D \Delta \Delta w - 2\sigma_{0,p} \rho_p H \partial_t w + 2\sigma_{1,p} \rho_p H \partial_t \partial_x^2 w, \quad (11.39)$$

where parameters are as in Eq. (6.37).

One can connect the above PDEs, by adding a localised connection force. After a division by $\rho_s A$ and $\rho_p H$ respectively the connected string-plate system

11.5. String-plate connection

becomes

$$\partial_t^2 u = c^2 \partial_{\chi\chi} u - \kappa_s^2 \partial_{\chi\chi\chi\chi} u - 2\sigma_{0,s} \partial_t u + 2\sigma_{1,s} \partial_t \partial_{\chi\chi} u - \delta(\chi - \chi_c) \frac{f}{\rho_s A}, \quad (11.40a)$$

$$\partial_t^2 w = -\kappa_p^2 \Delta \Delta w - 2\sigma_{0,p} \partial_t w + 2\sigma_{1,p} \partial_t \partial_x^2 w + \delta(x - x_c, y - y_c) \frac{f}{\rho_p H}, \quad (11.40b)$$

where $\delta(\chi - \chi_c)$ (in m^{-1}) and $\delta(x - x_c, y - y_c)$ (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 $\chi_c \in \mathcal{D}_s$ (in m) along the string and $(x_c, y_c) \in \mathcal{D}_p$ (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])

$$f = f(t) = K_1 \eta + K_3 \eta^3 + R \dot{\eta}, \quad (11.41)$$

with linear and nonlinear spring coefficients K_1 (in N/m) and K_3 (in N/m³) 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

$$\eta = \eta(t) = u(\chi_c, t) - w(x_c, y_c, t). \quad (11.42)$$

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

$$\begin{aligned} \delta_{tt} u_q^n &= c^2 \delta_{\chi\chi} u_q^n - \kappa_s^2 \delta_{\chi\chi\chi\chi} u_q^n - 2\sigma_{0,s} \delta_t u_q^n + 2\sigma_{1,s} \delta_t \delta_{\chi\chi} u_q^n \\ &\quad - J_q(\chi_c) \frac{f^n}{\rho_s A}, \end{aligned} \quad (11.43)$$

$$\begin{aligned} \delta_{tt} w_{l,m}^n &= -\kappa_p^2 \delta_\Delta \delta_\Delta w_{l,m}^n - 2\sigma_{0,p} \delta_t w_{l,m}^n + 2\sigma_{1,p} \delta_t \delta_{xx} w_{l,m}^n \\ &\quad + J_{l,m}(x_c, y_c) \frac{f^n}{\rho_p H}, \end{aligned} \quad (11.44)$$

where $J_q(\chi_c) = J_{q,o_s}(\chi_c)$ is a 1D spreading operator of order o_s as defined in Section 8.2 and $J_{l,m}(x_c, y_c) = J_{(l,m),o_p}(x_c, y_c)$ 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 as⁵

$$f^n = K_1 \mu_{tt} \eta^n + K_3 (\eta^n)^2 \mu_t \cdot \eta^n + R \delta_t \cdot \eta^n, \quad (11.45)$$

where

$$\eta^n = I_q(x_c) u_q^n - I_{l,m}(x_c, y_c) w_m^n. \quad (11.46)$$

Here, $I_q(\chi_c) = I_{q,o_s}(\chi_c)$ and $I_{l,m}(x_c, y_c) = I_{(l,m),o_p}(x_c, y_c)$ are interpolation operators of the same order as $J_q(\chi_c)$ and $J_{l,m}(x_c)$ 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 $d_u = \{0, \dots, N\}$ and $d_w = \{0, \dots, N_x\} \times \{0, \dots, N_y\}$ respectively. Taking these inner products, expanding the δ_{tt} and δ_t operators (as these contain u_q^{n+1}) and solving for the states at $n + 1$ yields

$$I_q(\chi_c) u_q^{n+1} = u^* - \|J_q(\chi_c)\|_{d_u}^2 \frac{k^2 f^n}{\rho_s A (1 + \sigma_{0,s} k)}, \quad (11.47a)$$

$$I_{l,m}(x_c) w_{l,m}^n = w^* + \|J_{l,m}(x_c, y_c)\|_{d_w}^2 \frac{k^2 f^n}{\rho_p H (1 + \sigma_{0,p} k)}, \quad (11.47b)$$

where

$$\begin{aligned} u^* &= \left(I_q(\chi_c) (2u_q^n - u_q^{n-1}) + c^2 k^2 I_q(\chi_c) \delta_{\chi\chi} u_q^n - \kappa_s^2 k^2 I_q(\chi_c) \delta_{\chi\chi\chi\chi} u_q^n \right. \\ &\quad \left. + \sigma_{0,s} k I_q(\chi_c) u_q^{n-1} + 2\sigma_{1,s} k^2 I_q(\chi_c) \delta_{t-} \delta_{\chi\chi} u_q^n \right) / (1 + \sigma_{0,s} k), \end{aligned} \quad (11.48a)$$

$$\begin{aligned} w^* &= \left(-\kappa_p^2 I_{l,m}(x_c) \delta_{\Delta} \delta_{\Delta} w_{l,m}^n + \sigma_{0,p} k I_{l,m}(x_c) w_{l,m}^{n-1} \right. \\ &\quad \left. + 2\sigma_{1,p} I_{l,m}(x_c) \delta_{t-} \delta_{xx} w_{l,m}^n \right) / (1 + \sigma_{0,p} k), \end{aligned} \quad (11.48b)$$

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

$$\begin{aligned} \eta^{n+1} &= u^* - \|J_q(\chi_c)\|_{d_u}^2 \frac{k^2 f^n}{\rho_s A (1 + \sigma_{0,s} k)} \\ &\quad - \left(w^* + \|J_{l,m}(x_c, y_c)\|_{d_w}^2 \frac{k^2 f^n}{\rho_p H (1 + \sigma_{0,p} k)} \right). \end{aligned} \quad (11.49)$$

⁵The 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

$$\eta^{n+1} = \frac{f^n}{r_+^n} - \frac{K_1}{2r_+^n} \eta^n - \frac{r_-^n}{r_+^n} \eta^{n-1}. \quad (11.50)$$

Substituting this into Eq. (11.49), one can find a definition for the connecting force

$$f^n = \frac{u^* - w^* + \frac{K_1}{2r_+^n} \eta^n + \frac{r_-^n}{r_+^n} \eta^{n-1}}{\frac{1}{r_+^n} + \frac{\|J_d(x_c)\|_{d_u}^2 k^2}{\rho_s A(1+\sigma_{0,s}k)} + \frac{\|J_{l,m}(x_c, y_c)\|_{d_w}^2 k^2}{\rho_p H(1+\sigma_{0,p}k)}}. \quad (11.51)$$

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 \mathbf{i} and \mathbf{j} vectors are used as 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:

$$\mathbf{u}^* = \frac{\mathbf{B}_s \mathbf{u}^n + \mathbf{C}_s \mathbf{u}^{n-1}}{A_s}, \quad \text{and} \quad \mathbf{w}^* = \frac{\mathbf{B}_p \mathbf{w}^n + \mathbf{C}_p \mathbf{w}^{n-1}}{A_p}. \quad (11.52)$$

2. Obtain u^* and w^* :

$$u^* = \mathbf{i}_u \mathbf{u}^* \quad \text{and} \quad w^* = \mathbf{i}_w \mathbf{w}^*. \quad (11.53)$$

⁶Note that in the online code, L , L_x and L_y have been halved to reduce computations and avoid crashes on some machines.

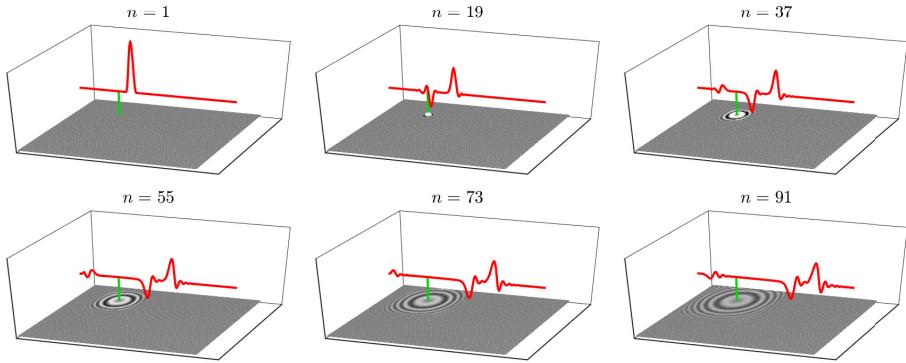


Fig. 11.6: The behaviour of the string-plate system connected with a nonlinear damped spring. The string is shown in red, the plate in gray and the connection in green.

3. Calculate the connection force f^n (Eq. (11.51)).
4. Add force terms to the schemes in Eq. (11.52):

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \mathbf{j}_u \frac{f^n k^2}{\rho_s A (1 + \sigma_{0,s})}, \quad \text{and} \quad \mathbf{w}^{n+1} = \mathbf{w}^* + \mathbf{j}_w \frac{f^n k^2}{\rho_p H (1 + \sigma_{0,p})}. \quad (11.54)$$

Calculating \mathbf{u}^* and \mathbf{w}^* beforehand reduces computations, and allows u^* and w^* 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:

$$\delta_{t+}(\mathfrak{h}_s + \mathfrak{h}_p) + q_s + q_p = -I_q(\chi_c)(\delta_t \cdot u_q^n) f^n + I_{l,m}(x_c)(\delta_t \cdot w_{l,m}^n) f^n, \quad (11.55)$$

which can be rewritten to

$$\delta_{t+}(\mathfrak{h}_s + \mathfrak{h}_p) + q_s + q_p = -\delta_t \cdot (I_q(\chi_c)u_q^n - I_{l,m}(x_c)w_{l,m}^n) f^n.$$

Substituting the definitions for f^n and η^n from Eqs. (11.45) and (11.46) respectively, yields

$$\delta_{t+}(\mathfrak{h}_s + \mathfrak{h}_p) + q_s + q_p = -(\delta_t \cdot \eta^n)(K_1 \mu_{tt} \eta^n + K_3(\eta^n)^2 \mu_t \cdot \eta^n + R \delta_t \cdot \eta^n).$$

11.5. String-plate connection

Due to the nonlinear dependency on η^n one must isolate δ_{t+} from the cubic term manually, according to

$$\begin{aligned}
& K_3(\eta^n)^2 (\delta_t \cdot \eta^n) (\mu_t \cdot \eta^n) \\
&= \frac{K_3(\eta^2)}{2k} (\eta^{n+1} - \eta^{n-1}) \frac{1}{2} (\eta^{n+1} + \eta^{n-1}) \\
&= \frac{K_3(\eta^n)^2}{4k} ((\eta^{n+1})^2 - (\eta^{n-1})^2) \\
&= \frac{K_3}{4k} ((\eta^{n+1} \eta^n)^2 - (\eta^n \eta^{n-1})^2) \\
&= \delta_{t+} \left(\frac{K_3}{4} (\eta^n \eta^{n-1})^2 \right).
\end{aligned}$$

Finally, using identity (3.17e) for the linear term, the following balance follows

$$\delta_{t+} (\mathfrak{h}_s + \mathfrak{h}_p + \mathfrak{h}_c) = -\mathfrak{q}_s - \mathfrak{q}_p - \mathfrak{q}_c, \quad (11.56)$$

where the energy stored by the spring connection is

$$\mathfrak{h}_c = \frac{K_1}{8} (\eta^n + \eta^{n-1})^2 + \frac{K_3}{4} (\eta^n \eta^{n-1})^2,$$

and the damping term of the connection is

$$\mathfrak{q}_c = R(\delta_t \cdot \eta^n)^2.$$

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.

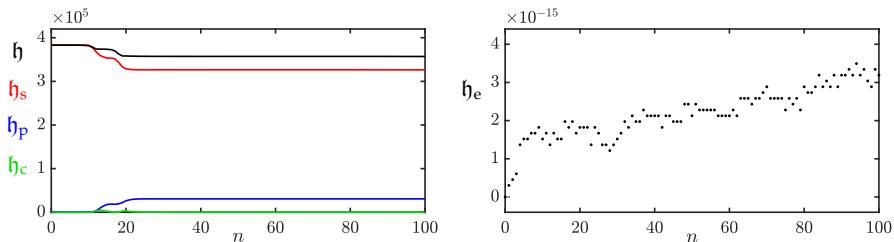


Fig. 11.7: The energy of the string (red), the plate (blue), the spring connection (green) and the total energy (black) of the connected string-plate system (11.43). The energy corresponds to the system in Figure 11.6. The right panel shows the normalised energy (according to Eq. (3.38)) and shows that the deviation of the energy is within machine precision.

Chapter 11. Connections

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 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 and has been 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 presents 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 implementation details and design considerations.

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 $l \in \{0, \dots, N\}$ 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

$$\delta_{tt} q_l^n = c^2 \delta_{xx} q_l^n. \quad (12.1)$$

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)):

$$h := ck, \quad N := \left\lfloor \frac{L}{h} \right\rfloor, \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}, \quad (12.2)$$

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

$$h := ck, \quad \mathcal{N} := \frac{L}{h}, \quad \lambda := \frac{ck}{h}, \quad (12.3)$$

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

$$\delta_{tt} u_{l_u}^n = (c^n)^2 \delta_{xx} u_{l_u}^n, \quad (12.4a)$$

$$\delta_{tt} w_{l_w}^n = (c^n)^2 \delta_{xx} w_{l_w}^n, \quad (12.4b)$$

where $l_u \in \{0, \dots, M^n\}$ and $l_w \in \{0, \dots, M_w^n\}$ and integers

$$M^n = \lceil 0.5N^n \rceil \quad \text{and} \quad M_w^n = \lfloor 0.5N^n \rfloor \quad (12.5)$$

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:

$$x_{u_0}^n = 0, \quad x_{w_{M_w^n}}^n = L. \quad (12.6)$$

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

$$x_{u_{M^n}}^n = M^n h^n, \quad x_{w_0}^n = L - M_w^n h^n. \quad (12.7)$$

If 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 to¹

$$x_{u_{l_u}}^n = l_u h^n, \quad x_{w_{l_w}}^n = L - (M_w^n - l_w) h^n. \quad (12.8)$$

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

¹Notice 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).

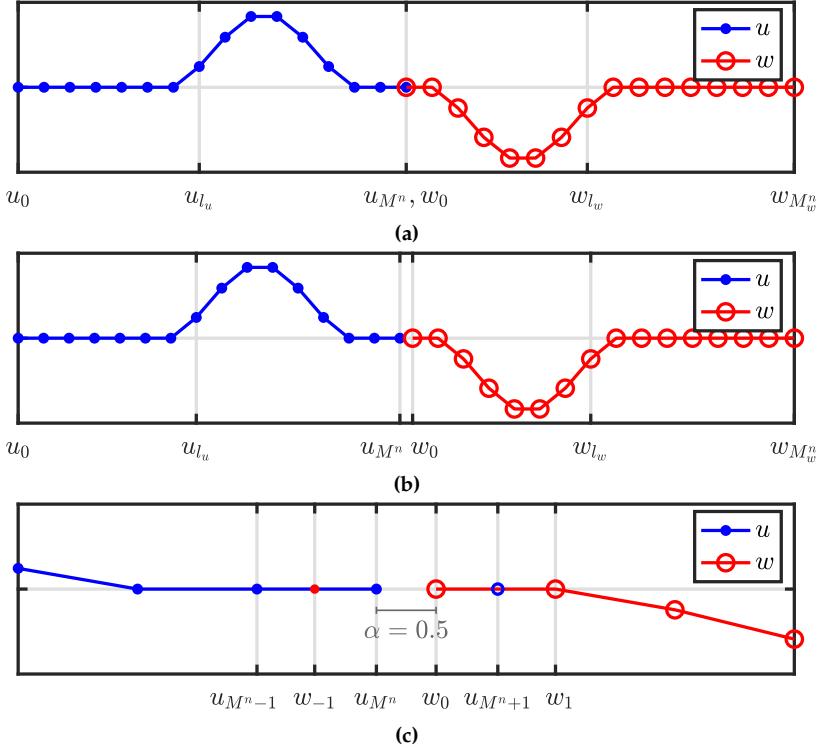


Fig. 12.1: Illustration of the proposed method. In all figures, the x-axis shows the location of the respective grid points, but ‘\$x^n\$’ is omitted for brevity. (a) Locations of the states of two (1D wave) systems connected at the inner boundaries ($N^n = 30$, $x_{u,M^n}^n = x_{w,0}^n$). (b) When c^n – and consequently h^n – are decreased and the positions of the grid points change ($N^n = 30.5$, $x_{u,M^n}^n \neq x_{w,0}^n$). (c) Figure 12.1b zoomed-in around the inner boundaries. The virtual grid points u_{M^n+1} and w_{-1}^n 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 N^n

$$\alpha = \alpha^n = N^n - N^n \quad (12.9)$$

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,

12.2. Extended summary

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.,

$$u_{M^n}^n = w_0^n, \quad \text{if } \alpha = 0, \quad (12.10)$$

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. (11.3) 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

$$u_{M^n+1}^n = \frac{\alpha - 1}{\alpha + 1} u_{M^n}^n + w_0^n - \frac{\alpha - 1}{\alpha + 1} w_1^n, \quad (12.11a)$$

$$w_{-1}^n = -\frac{\alpha - 1}{\alpha + 1} u_{M^n-1}^n + u_{M^n}^n + \frac{\alpha - 1}{\alpha + 1} w_0^n. \quad (12.11b)$$

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

$$I_i(x) = \prod_{j=0, j \neq i}^o \frac{x - x_j}{x_i - x_j}, \quad (12.12)$$

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)

$$\begin{aligned} x_0 &= x_{u_{M^n}} = 0, \\ x_1 &= x_{w_0} = \alpha, \\ x_2 &= x_{w_1} = \alpha + 1. \end{aligned} \quad (12.13)$$

The virtual grid point is the point calculated by the interpolation and will be at

$$x = x_{u_{M^n}+1} = 1. \quad (12.14)$$

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:

$$\mathbf{u}^n = [u_1^n, \dots, u_{M^n}^n]^T, \text{ and } \mathbf{w}^n = [w_0^n, \dots, w_{M_w^n-1}^n]^T, \quad (12.15)$$

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 $-h^n$ 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 and will be done according to the following:

$$\text{if } N^n > N^{n-1} \begin{cases} \mathbf{u}^n = [(\mathbf{u}^n)^T, I_3 \mathbf{v}^n]^T & \text{if } N^n \text{ is odd,} \\ \mathbf{w}^n = [I_3^\leftarrow \mathbf{v}^n, (\mathbf{w}^n)^T]^T & \text{if } N^n \text{ is even.} \end{cases} \quad (12.16)$$

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))

$$I_3 = \begin{bmatrix} -\frac{\alpha(\alpha+1)}{(\alpha+2)(\alpha+3)} & \frac{2\alpha}{\alpha+2} & \frac{2}{\alpha+2} & -\frac{2\alpha}{(\alpha+3)(\alpha+2)} \end{bmatrix}, \quad (12.17)$$

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:

$$\text{if } N^n < N^{n-1} \begin{cases} \mathbf{u}^n = [u_0^n, u_1^n \dots, u_{M^n-1}^n]^T & \text{if } N^n \text{ is even,} \\ \mathbf{w}^n = [w_1^n, w_2^n \dots, w_{M_w^n}^n]^T & \text{if } N^n \text{ is odd.} \end{cases} \quad (12.18)$$

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

$$M^n = N^n - 1, \quad \text{and} \quad M_w^n = 1, \quad (12.19)$$

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

$$\mathbf{u}^n = [(\mathbf{u}^n)^T, I_3 \mathbf{v}^n]^T, \quad \text{if } N^n > N^{n-1} \quad (12.20)$$

and

$$\mathbf{u}^n = [u_0^n, u_1^n \dots, u_{M^n-1}^n]^T, \quad \text{if } N^n < N^{n-1}, \quad (12.21)$$

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 and 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 as³

$$\delta_{tt} u_{l_u}^n = (c^n)^2 \delta_{xx} u_{l_u}^n + J_{l_u}(x_{u_{M^n}}^n) F_c^n, \quad (12.22a)$$

$$\delta_{tt} w_{l_w}^n = (c^n)^2 \delta_{xx} w_{l_w}^n - J_{l_w}(x_{w_0}^n) F_c^n, \quad (12.22b)$$

where the correction effect is determined by a linear damped spring (see Chapter 11)

$$F_c^n = \beta (\mu_t \eta^n + \sigma_0 \delta_t \eta^n). \quad (12.23)$$

Here, σ_0 is a damping coefficient and the difference between the system states

²The 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’.

³Paper [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.

at the inner boundaries is defined as

$$\eta^n \triangleq w_0^n - u_{M^n}^n. \quad (12.24)$$

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 $\alpha = 0$, $\beta \rightarrow \infty$ and the correction effect acts like a rigid connection. If, on the other hand $\alpha \rightarrow 1$, the correction effect should vanish according to $\beta \rightarrow 0$. The following function that satisfies these conditions was proposed in paper [G]:

$$\beta = \frac{1 - \alpha}{\alpha + \varepsilon}, \quad (12.25)$$

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

$$u_{M^n}^{n+1} = 2u_{M^n}^n - u_{M^n}^{n-1} + (c^n)^2 k^2 \delta_{xx} u_{M^n}^n + \frac{k^2}{h^n} F_c^n, \quad (12.26a)$$

$$w_0^{n+1} = 2w_0^n - w_0^{n-1} + (c^n)^2 k^2 \delta_{xx} w_{l_w}^n - \frac{k^2}{h^n} F_c^n, \quad (12.26b)$$

and substituting this into Eq. (12.24) evaluated at $n + 1$, yields

$$\eta^{n+1} = w^* - \frac{k^2}{h^n} F_c^n - \left(u^* + \frac{k^2}{h^n} F_c^n \right), \quad (12.27)$$

where

$$u^* = 2u_{M^n}^n - u_{M^n}^{n-1} + (c^n)^2 k^2 \delta_{xx} u_{M^n}^n,$$

and

$$w^* = 2w_0^n - w_0^{n-1} + (c^n)^2 k^2 \delta_{xx} w_0^n,$$

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

$$\begin{aligned} F_c^n &= \beta \left(\frac{1}{2} (\eta^{n+1} + \eta^{n-1}) + \frac{\sigma_0}{2k} (\eta^{n+1} - \eta^{n-1}) \right), \\ F_c^n &= \left(\frac{\beta(1 + \sigma_0/k)}{2} \right) \eta^{n+1} + \left(\frac{\beta(1 - \sigma_0/k)}{2} \right) \eta^{n-1}, \\ \xrightarrow{\text{Eq. (12.25)}} \quad \eta^{n+1} &= \left(\frac{2(\alpha + \varepsilon)}{(1 + \sigma_0/k)(1 - \alpha)} \right) F_c^n - \underbrace{\frac{1 - \sigma_0/k}{1 + \sigma_0/k}}_r \eta^{n-1}. \end{aligned} \quad (12.28)$$

This definition can be substituted into Eq. (12.27) and solved for F_c^n according to

$$\begin{aligned} \left(\frac{2(\alpha + \varepsilon)}{(1 + \sigma_0/k)(1 - \alpha)} \right) F_c^n - r\eta^{n-1} &= w^* - u^* - \frac{2k^2}{h^n} F, \\ \left(\frac{2h^n(\alpha + \varepsilon) + 2k^2(1 + \sigma_0/k)(1 - \alpha)}{h^n(1 + \sigma_0/k)(1 - \alpha)} \right) F_c^n &= w^* - u^* + r\eta^{n-1}, \end{aligned}$$

and finally

$$F_c^n = (w^* - u^* + r\eta^{n-1}) \left(\frac{h^n(1 + \sigma_0/k)(1 - \alpha)}{2h^n(\alpha + \varepsilon) + 2k^2(1 + \sigma_0/k)(1 - \alpha)} \right).$$

It is clear now that no matter the value of α , no division by 0 will occur, proving that $\varepsilon = 0$ in Eq. (12.25) will still yield a defined solution. The final equation for F_c^n can thus be written as

$$F_c^n = (w^* - u^* + r\eta^{n-1}) \left(\frac{h^n(1 + \sigma_0/k)(1 - \alpha)}{2h^n\alpha + 2k^2(1 + \sigma_0/k)(1 - \alpha)} \right). \quad (12.29)$$

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 as⁴

$$\mathcal{U}^n = \begin{bmatrix} \mathbf{u}^n \\ \mathbf{w}^n \end{bmatrix}, \quad (12.30)$$

⁴Note that, although \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

$$\begin{aligned}\mathbf{I}_{\mathcal{M}^n} \mathbf{U}^{n+1} &= \mathbf{B}^n \mathbf{U}^n - \mathbf{I}_{\mathcal{M}^n} \mathbf{U}^{n-1} \\ &\quad + (\mathbf{j}\boldsymbol{\eta}) \frac{\beta k^2}{2} ((1 + \sigma_0/k) \mathbf{U}^{n+1} + (1 - \sigma_0/k) \mathbf{U}^{n-1})\end{aligned}\quad (12.31)$$

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 \mathbf{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):

$$\mathbf{B}^n = 2\mathbf{I}_{\mathcal{M}^n} + \lambda^2 \left[\begin{array}{ccc|cc} \ddots & \ddots & \ddots & & \mathbf{0} \\ 1 & -2 & 1 & & \\ \hline & \frac{1}{\alpha+1} & \frac{\alpha-1}{\alpha+1}-2 & 1 & -\frac{\alpha-1}{\alpha+1} \\ -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1}-2 & 1 & 1 \\ \hline \mathbf{0} & & 1 & -2 & 1 \\ & & & \ddots & \ddots \end{array} \right]$$

which, as the method allows $\lambda = 1$ at all times, can be simplified to

$$\mathbf{B}^n = \left[\begin{array}{ccc|cc} \ddots & \ddots & \ddots & & \mathbf{0} \\ 1 & 0 & 1 & & \\ \hline & \frac{1}{\alpha+1} & \frac{\alpha-1}{\alpha+1} & 1 & -\frac{\alpha-1}{\alpha+1} \\ -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1} & 1 & 1 \\ \hline \mathbf{0} & & 1 & 0 & 1 \\ & & & \ddots & \ddots \end{array} \right]. \quad (12.32)$$

Equation (12.31) can then be rewritten to

$$\mathbf{A}^n \mathbf{U}^{n+1} = \mathbf{B}^n \mathbf{U}^n + \mathbf{C}^n \mathbf{U}^{n-1}, \quad (12.33)$$

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

$$\mathbf{B}^n = \left[\begin{array}{ccc|c} \ddots & \ddots & \ddots & \\ & 1 & 0 & 1 \\ & 1 & \frac{\alpha-1}{\alpha+1} & 1 \\ \hline -\frac{\alpha-1}{\alpha+1} & 1 & \frac{\alpha-1}{\alpha+1} & \end{array} \right], \quad (12.34)$$

due to the Dirichlet boundary condition.

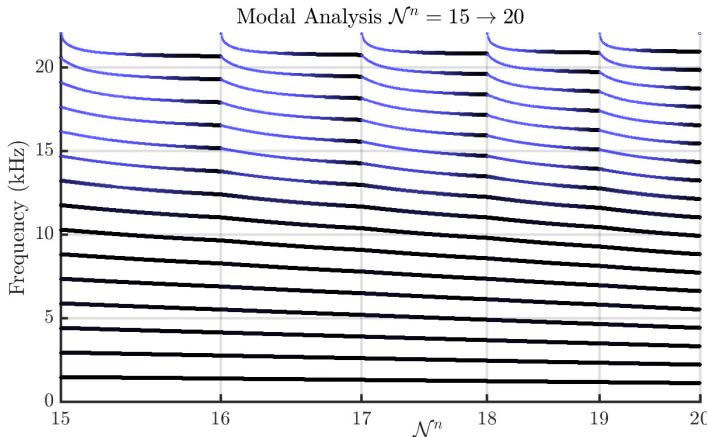


Fig. 12.2: Results of a modal analysis performed on the one-step form of the dynamic grid system in (12.35). Thinner and bluer lines indicate a higher amount of damping.

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, 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

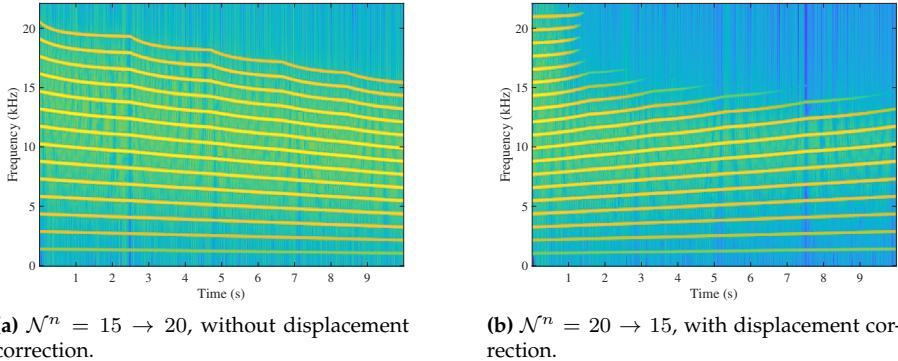


Fig. 12.3: Output spectra of an implementation of the dynamic grid.

one-step form (see Section 3.5.1),

$$\begin{bmatrix} \mathbf{U}^{n+1} \\ \mathbf{U}^n \end{bmatrix} = \underbrace{\begin{bmatrix} (\mathbf{A}^n)^{-1} \mathbf{B}^n & (\mathbf{A}^n)^{-1} \mathbf{C}^n \\ \mathbf{I}_{\mathcal{M}^n} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}^n} \begin{bmatrix} \mathbf{U}^n \\ \mathbf{U}^{n-1} \end{bmatrix}, \quad (12.35)$$

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 $\mathcal{N} = 20 \rightarrow 15$ without displacement correction, and shows that the modes follow the pattern predicted by the analysis. Figure 12.3b presents the spectral output where $\mathcal{N} = 15 \rightarrow 20$ 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).

⁵Notice 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

$$\mathcal{U}^{n+1} = \mathbf{B}^n \mathcal{U}^n - \mathcal{U}^{n-1} \quad (12.36)$$

and, using a test solution $\mathcal{U}^n = z^n \phi$ (following Section 3.5), the eigenfrequencies can be calculated according to (using trigonometric identity (3.21b))

$$f_p^n = \frac{1}{2\pi k} \cos^{-1} \left(\frac{1}{2} \text{eig}_p(\mathbf{B}^n) \right). \quad (12.37)$$

In the following, results will be shown for a varying wave speed corresponding $N^n = 15 \rightarrow 20$, 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 is 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

$$u_{M^n+1}^n = \alpha w_0^n + (1 - \alpha) w_1^n, \quad (12.38a)$$

$$w_{-1}^n = (1 - \alpha) u_{M^n-1}^n + \alpha u_{M^n}^n, \quad (12.38b)$$

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

$$\mathbf{B}^n = \left[\begin{array}{ccc|cc} \ddots & \ddots & & & 0 \\ 1 & 0 & 1 & \alpha & (1 - \alpha) \\ & 1 & 0 & 0 & 1 \\ \hline (1 - \alpha) & \alpha & & 1 & 0 & 1 \\ 0 & & & & \ddots & \ddots \end{array} \right] \quad (12.39)$$

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.

12.3. Interpolation experiments

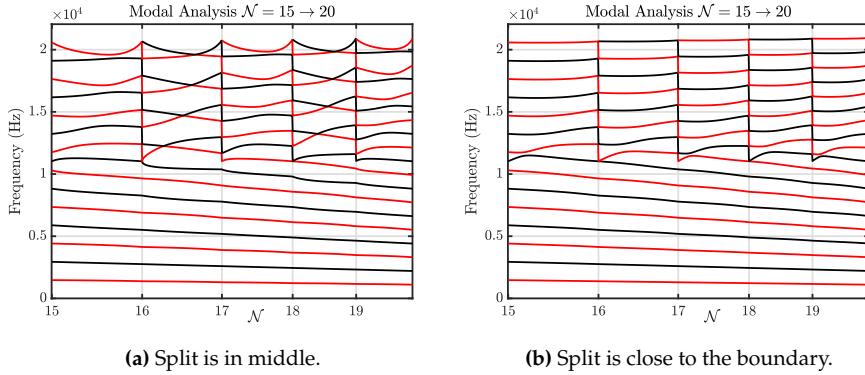


Fig. 12.4: Results of modal analysis for linear interpolation.

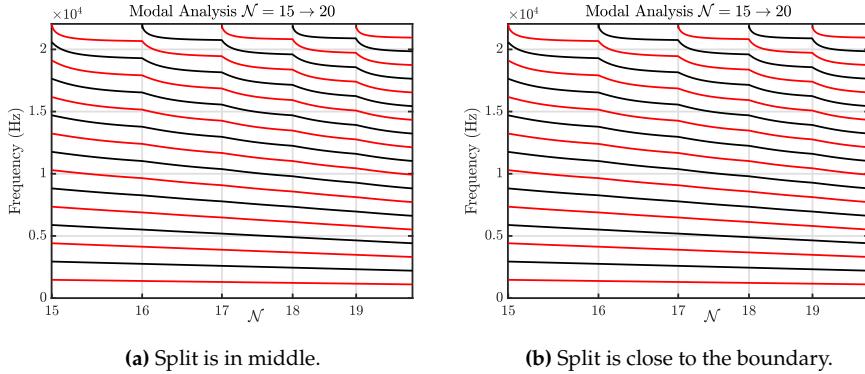


Fig. 12.5: Results of modal analysis for quadratic interpolation.

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

$$u_{M^n+1}^n = \frac{\alpha - 1}{\alpha + 2} u_{M^n}^n + \frac{\alpha + 1}{2} w_0^n + (1 - \alpha) w_1^n + \frac{\alpha(\alpha - 1)}{2(\alpha + 2)} w_2^n, \quad (12.42a)$$

$$w_{-1}^n = \frac{\alpha(\alpha - 1)}{2(\alpha + 2)} u_{M^n-2}^n + (1 - \alpha) u_{M^n-1}^n + \frac{\alpha + 1}{2} u_{M^n}^n + \frac{\alpha - 1}{\alpha + 2} w_0^n, \quad (12.42b)$$

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 $w_2^n = -w_0^n$. As the value at the boundary remains 0, i.e., $w_1 = 0$, this alters Eqs. (12.42) to be

$$u_{M^n+1}^n = \frac{\alpha - 1}{\alpha + 2} u_{M^n}^n + \left(\frac{\alpha + 1}{2} - \frac{\alpha(\alpha - 1)}{2(\alpha + 2)} \right) w_0^n$$

$$w_{-1}^n = \frac{\alpha(\alpha - 1)}{2(\alpha + 2)} u_{M^n-2}^n + (1 - \alpha) u_{M^n-1}^n + \frac{\alpha + 1}{2} u_{M^n}^n.$$

The results are shown in Figure 12.6.

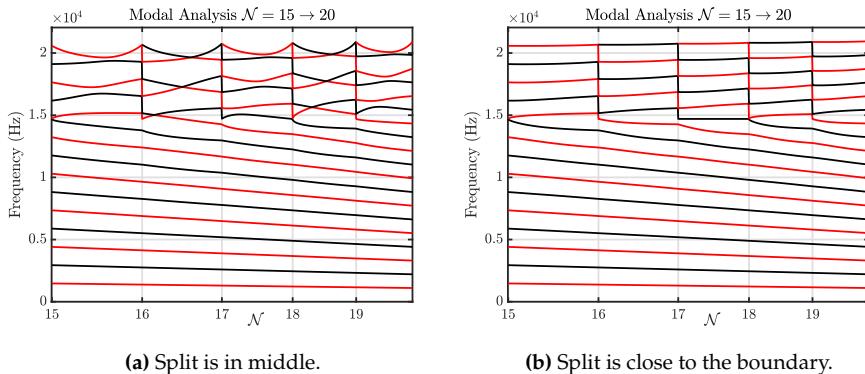


Fig. 12.6: Results of modal analysis for cubic interpolation.

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.

12.3. Interpolation experiments

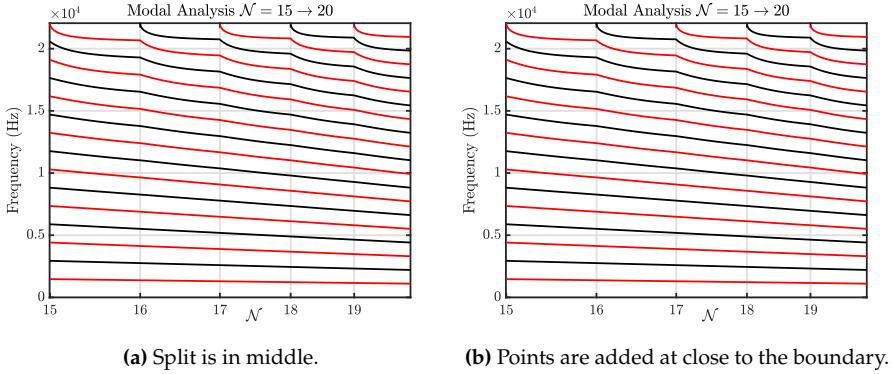


Fig. 12.7: Results of modal analysis for quartic (fourth-order) interpolation.

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 in the middle, ‘modal crossings’ appear for even values of $\lfloor \mathcal{N}^n \rfloor$, but 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 $3/8f_s$ 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 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 nut⁶, 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], which uses the method presented in this chapter. 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 by 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

⁶John Mayer - Gravity (Live in L.A.): <https://youtu.be/dBFW8OvcIu?t=284>

⁷Jon Gomm - Passionflower: <https://youtu.be/nY7GnAq6Znw?t=49>

⁸gr4yhound - servo bender: <https://youtu.be/fSQ9Dg65EFo>

⁹Amazing Hammered Dulcimer Musician - Joshua Messick: <https://youtu.be/veuGTnzgNRU?t=215>

¹⁰John Joe Bodhran Solo - Christ church Dublin 2012: <https://youtu.be/b9HyB5yNS1A?t=146>

¹¹Ayan Bisi Adeleke - Master talking drummer - drum talks: <https://youtu.be/B4oQJZ2TEVI?t=9>

¹²Musical 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 as 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:

$$u[0] \rightarrow \mathbf{u}^{n+1}, \quad u[1] \rightarrow \mathbf{u}^n, \quad \text{and} \quad u[2] \rightarrow \mathbf{u}^{n-1}.$$

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).

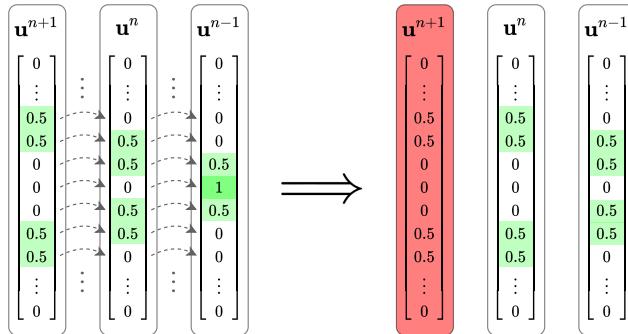
The state vectors themselves are stored in a matrix (which is a ‘vector of vectors’ in C++). This matrix will have 3 columns related to the 3 time steps required to calculate the FD scheme, and $N + 1$ rows, which corresponds to the number of grid points.³ The matrix is initialised as follows

```
//// In the constructor of SimpleString /////

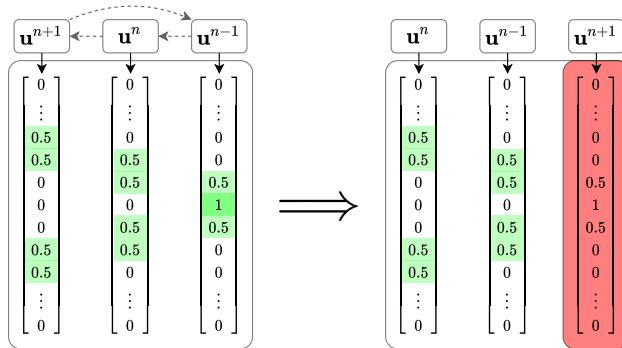
// initialise vectors
uStates = std::vector<std::vector<double>> (3,
                                                 std::vector<double> (N+1, 0));
```

Next, the aforementioned pointers are initialised such that they contain the memory addresses of the start of the three state vectors in the matrix.

³These 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

13.1. Real-time FD schemes

vary over time). Recalling the update equation for a stiff string in Eq. (4.10), one can write this as

$$A_{\text{div}} u_l^{n+1} = B_1 u_l^n + B_2(u_{l+1}^n + u_{l-1}^n) + B_3(u_{l+2}^n + u_{l-2}^n) + C_1 u_l^{n-1} + C_2(u_{l+1}^{n-1} + u_{l-1}^{n-1}), \quad (13.1)$$

where

$$\begin{aligned} A_{\text{div}} &= 1 + \sigma_0 k, & B_0 &= 2 - 2\lambda^2 - 6\mu^2 - \frac{4\sigma_1 k}{h^2}, & B_1 &= \lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}, \\ B_2 &= -\mu^2, & C_0 &= -1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}, & \text{and} & C_1 = -\frac{2\sigma_1 k}{h^2}. \end{aligned}$$

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 * sigmal * 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 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

of instruction / dataflow and the colour of the box denotes at what rate this happens.

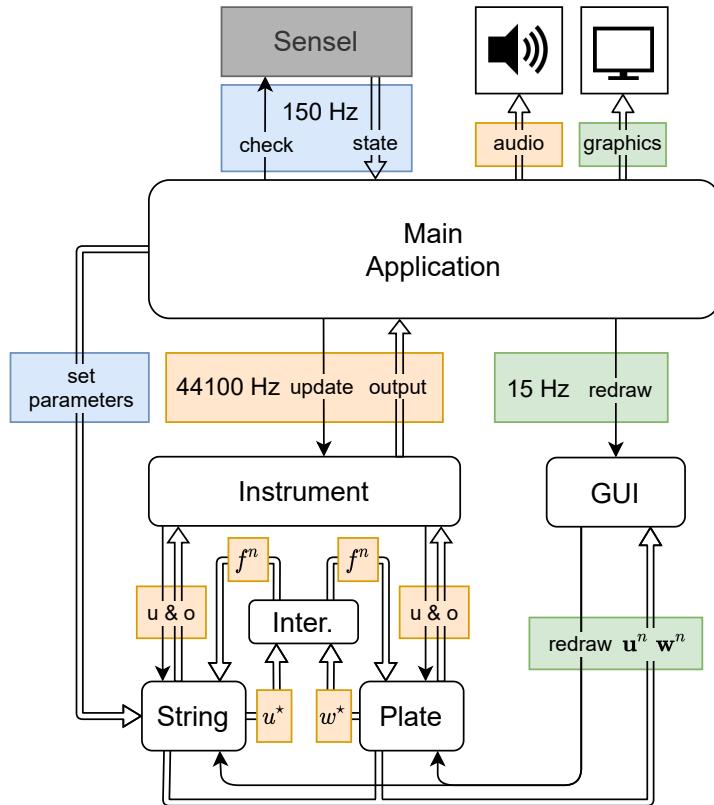


Fig. 13.2: The structure of a real-time implementation of a physical model. Boxes with 'u & o' refer to 'update and output' and 'Inter.' is short for 'Interaction'. A detailed description of the figure is given in Section 13.2.

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^* and w^* , 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^* and w^* , 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)).

⁴Note 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

⁵This 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 14.3.1). 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/BkxLji2aplw>

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 . The PDE of string s with an external connection force is defined as

$$\begin{aligned}\partial_t^2 u_s = & c_s^2 \partial_{\chi_s \chi_s} u_s - \kappa_s^2 \partial_{\chi_s \chi_s \chi_s \chi_s} u_s - 2\sigma_{0,s} \partial_t u_s \\ & + 2\sigma_{1,s} \partial_{\chi_s \chi_s} u_s - \delta(\chi_s - \chi_{s,c}) \frac{f_s}{\rho_s A_s},\end{aligned}\quad (14.1)$$

where spatial Dirac delta function $\delta(\chi_s - \chi_{s,c})$ (in m^{-1}) localises the connection force between the string s and the plate to connection location $\chi_{s,c}$. Other parameters are as defined in Eq. (4.3) but have a subscript s to denote that they can be different for each strings.

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

$$\partial_t^2 w = -\kappa_p^2 \Delta \Delta w - 2\sigma_{0,p} \partial_t w + 2\sigma_{1,p} \partial_t \partial_x^2 w + \sum_s \delta(x - x_{c,s}, y - y_{c,s}) \frac{f_s}{\rho_p H}, \quad (14.2)$$

where 2D spatial Dirac delta function $\delta(x_s - x_{s,c}, y_s - y_{s,c})$ (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

$$\eta_s = u_s(\chi_{c,s}, t) - w(x_{c,s}, y_{c,s}, t) \quad (14.4)$$

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.

14.2 Implementation

This section provides considerations for implementing the above models. Details on discretisation of the models and how to solve for 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.

14.3 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.

14.3.1 Instruments

Using the setup presented in Section 14.1, various configurations inspired by real instruments have been made. The choices of simulated instruments were aimed at those containing many (sympathetic) strings.³ Another condition was that no FDTD-based physical models 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 are implemented, and one in each pair is connected to the plate. This causes a slight detuning between the strings, resulting in a characteristic

³Sympathetic strings – apart from being friendly – are strings that add resonances to the instrument without being excited directly.

'chorus' effect exhibited by the instrument. To excite the strings, the time-varying strike presented in Section 7.2.1 is used.

14.3.2 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 8.16 was used to implement the wheel.

The instrument simulation consists of 5 bowed strings and 13 sympathetic strings, all connected to a plate.

14.3.3 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 natural and 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 goes through 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>

²youtu.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=S1HqvaaPCyU>

where \mathcal{L} is a linear (partial) differential operator and $q(\mathbf{x}, t)$ describes the state of the component in isolation. q is defined for time $t \geq 0$ and spatial coordinate $\mathbf{x} \in \mathcal{D}$ 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 $\chi \in \mathcal{D}_s$ where domain $\mathcal{D}_s = [0, L]$.⁴ Using partial differential operator $\mathcal{L} = \mathcal{L}_s$, the PDE describing its motion is defined as

$$\mathcal{L}_s u = 0, \quad (15.2)$$

where (see Eq. (4.3))

$$\mathcal{L}_s = \rho_s A \partial_t^2 - T \partial_\chi^2 + E_s I \partial_\chi^4 + 2\rho_s A \sigma_{0,s} \partial_t - 2\rho_s A \sigma_{1,s} \partial_t \partial_\chi^2.$$

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 $\rho_s A \partial_t^2$ 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

$$\mathcal{L}_s u = -\delta(\chi - \chi_B) F_B \Phi(v_{\text{rel}}), \quad (15.3)$$

with bow position $\chi_B = \chi_B(t) \in \mathcal{D}_s$ (in m), bow force $F_B = F_B(t)$ 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 $\mathcal{L} = \mathcal{L}_m$, its PDE

⁴Notice 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

$$\mathcal{L}_m w = 0, \quad (15.4)$$

where (see Eq. (9.1))

$$\mathcal{L}_m = M \frac{d^2}{dt^2} + M\omega_0^2 + M\sigma_m \frac{d}{dt}.$$

Here, $\omega_0 = \sqrt{K/M}$ (in s^{-1}) and $\sigma_m = R/M$ (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 $(x, y) \in \mathcal{D}_p$ where domain $\mathcal{D}_p = [0, L_x] \times [0, L_y]$. Using partial differential operator $\mathcal{L} = \mathcal{L}_p$, the PDE describing the motion of the body is

$$\mathcal{L}_p z = 0, \quad (15.5)$$

where (see Eq. (6.37))

$$\mathcal{L}_p = \rho_p H \partial_t^2 + D \Delta \Delta + 2\rho_p H \sigma_{0,p} \partial_t - 2\rho_p H \sigma_{1,p} \partial_t \Delta,$$

with $D = E_p H^3 / 12(1 - \nu^2)$. 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):

$$\phi(\eta) = \frac{K_{mp}}{\alpha_{mp} + 1} [\eta_{mp}]_+^{\alpha_{mp} + 1}, \quad (15.6)$$

⁵In paper [D], the symbol R is used for the damping coefficient, but for coherence in this work, σ_m is used instead.

15.2. Physical model

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):

$$\phi(\eta_{\text{sm}}) = \frac{K_{\text{sm}}}{\alpha_{\text{sm}} + 1} |\eta_{\text{sm}}|^{\alpha_{\text{sm}} + 1}, \quad (15.7)$$

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)

$$\phi'(\eta) = \psi\psi' \quad \text{where} \quad \psi = \sqrt{2\phi} \quad \text{and} \quad \psi' = \frac{\dot{\psi}}{\dot{\eta}}, \quad (15.8)$$

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

$$\mathcal{L}_{\text{s}}u = -\delta(\chi - \chi_{\text{B}})F_{\text{B}} + \delta(\chi - \chi_{\text{sm}})\psi_{\text{sm}}g_{\text{sm}}, \quad (15.9a)$$

$$\mathcal{L}_{\text{m}}w = -\psi_{\text{sm}}g_{\text{sm}} + \psi_{\text{mp}}g_{\text{mp}}, \quad (15.9b)$$

$$\mathcal{L}_{\text{p}}z = -\delta(x - x_{\text{mp}}, y - y_{\text{mp}})\psi_{\text{mp}}g_{\text{mp}}, \quad (15.9c)$$

$$\dot{\psi}_{\text{sm}} = g_{\text{sm}}\dot{\eta}_{\text{sm}}, \quad (15.9d)$$

$$\dot{\psi}_{\text{mp}} = g_{\text{mp}}\dot{\eta}_{\text{mp}}, \quad (15.9e)$$

$$\eta_{\text{sm}}(t) = w(t) - u(\chi_{\text{sm}}, t), \quad (15.9f)$$

$$\eta_{\text{mp}}(t) = z(x_{\text{mp}}, y_{\text{mp}}, t) - w(t). \quad (15.9g)$$

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 introducing⁶

$$\xi^n = \frac{k}{2}g^n\delta_t.\eta^n + \psi^{n-1/2}, \quad (15.10)$$

⁶The 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.

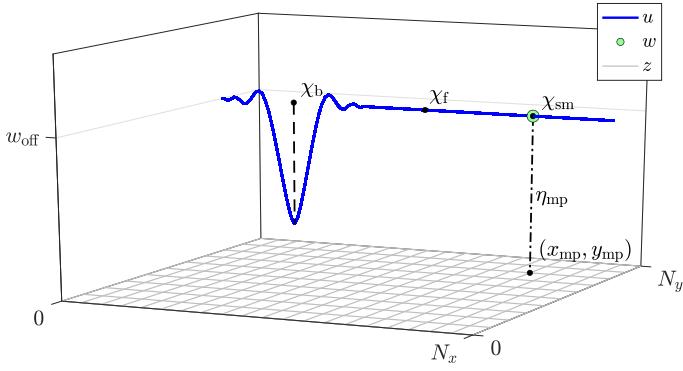


Fig. 15.1: A visualisation of system (15.9) with various important coordinates highlighted. The figure includes the offset and the damping finger described in paper [D]. Note that η_{sm} (Eq. (15.9f)) is not shown as it is close to 0 at all times. (Figure taken from [D].)

for brevity, system (15.9) can be discretised as follows:⁷

$$\ell_s u_l^n = -J_{l,3}(\chi_B) F_B + J_{l,0}(\chi_{sm}) \xi_{sm}^n g_{sm}^n, \quad (15.11a)$$

$$\ell_m w^n = -\xi_{sm}^n g_{sm}^n + \xi_{mp}^n g_{mp}^n, \quad (15.11b)$$

$$\ell_p z_{l,m}^n = -J_{(l,m),0}(x_{mp}, y_{mp}) \xi_{mp}^n g_{mp}^n, \quad (15.11c)$$

$$\delta_t \psi_{sm}^{n-1/2} = g_{sm}^n \delta_t \eta_{sm}^n, \quad (15.11d)$$

$$\delta_t \psi_{mp}^{n-1/2} = g_{mp}^n \delta_t \eta_{mp}^n, \quad (15.11e)$$

$$\eta_{sm}^n = w^n - u_{br}^n, \quad (15.11f)$$

$$\eta_{mp}^n = z_{br}^n - w^n, \quad (15.11g)$$

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

$$\ell_s = \rho_s A \delta_{tt} - T \delta_{xx} + EI \delta_{xxxx} + 2\rho_s A \sigma_{0,s} \delta_t - 2\rho_s A \sigma_{1,s} \delta_{t-} \delta_{xx}, \quad (15.12a)$$

$$\ell_b = M \delta_{tt} + M \omega_0^2 + M \sigma_m \delta_t, \quad (15.12b)$$

$$\ell_p = \rho_p H \delta_{tt} + D \delta_\Delta \delta_\Delta + 2\rho_p H \sigma_{0,p} \delta_t - 2\rho_p H \sigma_{1,p} \delta_{t-} \delta_\Delta. \quad (15.12c)$$

Furthermore, the definitions for g_{sm}^n and g_{mp}^n can be found in Eq. (10.15)⁸ and $J_{l,0}(\chi_{sm})$ and $J_{l,3}(\chi_B)$ are the 0th-order and cubic spreading operators respectively, as defined in Section 8.2 and $J_{(l,m),0}(x_{mp}, y_{mp})$ is a 0th-order 2D

⁷Notice 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.

⁸Paper [D] still uses the old definition of g^n in Eq. (10.13).

15.3. Implementation details

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 $l_{\text{sm}} = \lfloor \chi_{\text{sm}} / h_s \rfloor$ and $z_{\text{br}}^n = z_{(l_{\text{mp}}, m_{\text{mp}})}^n$ with $(l_{\text{mp}}, m_{\text{mp}}) = (\lfloor x_{\text{mp}} / h_p \rfloor, \lfloor y_{\text{mp}} / h_p \rfloor)$ 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. $\chi_B \neq \chi_{\text{sm}}$, 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)): $\|J_{l,0}(\chi_{\text{sm}})\|_{d_s}^2 = 1/h_s$

and $\|J_{(l,m),0}(x_{\text{mp}}, y_{\text{mp}})\|_{d_p}^2 = 1/h_p^2$. This yields the following updates for Eqs. (15.11a), (15.11b) and (15.11c) at the locations of interaction

$$u_{\text{br}}^{n+1} = u_{\text{br}}^* + \frac{k^2}{h_s \rho_s A (1 + \sigma_{0,s} k)} \left(\frac{(g_{\text{sm}}^n)^2 k}{2} \delta_t \cdot \eta_{\text{sm}}^n + \psi_{\text{sm}}^{n-1/2} g_{\text{sm}}^n \right), \quad (15.14a)$$

$$\begin{aligned} w^{n+1} &= w^* - \frac{k^2}{M (1 + \frac{\sigma_m k}{2})} \left(\frac{(g_{\text{sm}}^n)^2 k}{2} \delta_t \cdot \eta_{\text{sm}}^n + \psi_{\text{sm}}^{n-1/2} g_{\text{sm}}^n \right) \\ &\quad + \frac{k^2}{M (1 + \frac{\sigma_m k}{2})} \left(\frac{(g_{\text{mp}}^n)^2 k}{2} \delta_t \cdot \eta_{\text{mp}}^n + \psi_{\text{mp}}^{n-1/2} g_{\text{mp}}^n \right), \end{aligned} \quad (15.14b)$$

$$z_{\text{br}}^{n+1} = z_{\text{br}}^* - \frac{k^2}{h_p^2 \rho_p H (1 + \sigma_{0,p} k)} \left(\frac{(g_{\text{mp}}^n)^2 k}{2} \delta_t \cdot \eta_{\text{mp}}^n + \psi_{\text{mp}}^{n-1/2} g_{\text{mp}}^n \right), \quad (15.14c)$$

where the update equations of the components in isolation (without the collision terms) at their respective interaction locations are

$$\begin{aligned} u_{\text{br}}^* &= \frac{2u_{\text{br}}^n - u_{\text{br}}^{n-1} + c^2 k^2 \delta_{xx} u_{\text{br}}^n - \kappa_s^2 k^2 \delta_{xxx} u_{\text{br}}^n + \sigma_{0,s} k u_{\text{br}}^{n-1} + 2\sigma_{1,s} k^2 \delta_t - \delta_{xx} u_{\text{br}}^n}{1 + \sigma_{0,s} k}, \\ w^* &= \frac{2w^n - w^{n-1} - k^2 \omega_0^2 w^n + \frac{\sigma_m k}{2} w^{n-1}}{1 + \frac{\sigma_m k}{2}}, \\ z_{\text{br}}^* &= \frac{2z_{\text{br}}^n - z_{\text{br}}^{n-1} - \kappa_p^2 k^2 \delta_\Delta \delta_{\Delta} z_{\text{br}}^n + \sigma_{0,p} k z_{\text{br}}^{n-1} + 2\sigma_{1,p} k^2 \delta_t - \delta_\Delta z_{\text{br}}^n}{1 + \sigma_{0,p} k}, \end{aligned}$$

with $c = \sqrt{T/\rho_s A}$, $\kappa_s = \sqrt{E_s I/\rho_s A}$ and $\kappa_p = \sqrt{D/\rho_p H}$.

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

$$\begin{bmatrix} \delta_t \cdot \eta_{\text{sm}}^n \\ \delta_t \cdot \eta_{\text{mp}}^n \end{bmatrix} = \mathbf{A}^{-1} \mathbf{v}, \quad (15.15)$$

where

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} 1 + \frac{(g_{\text{sm}}^n)^2 k^2}{2M(2+\sigma_m k)} + \frac{(g_{\text{sm}}^n)^2 k^2}{4\rho_s A h_s (1+\sigma_{0,s} k)} & -\frac{(g_{\text{mp}}^n)^2 k^2}{2M(2+\sigma_m k)} \\ -\frac{(g_{\text{mp}}^n)^2 k^2}{2M(2+\sigma_m k)} & 1 + \frac{(g_{\text{mp}}^n)^2 k^2}{2M(2+\sigma_m k)} + \frac{(g_{\text{mp}}^n)^2 k^2}{4h_p^2 \rho_p H (1+\sigma_{0,p} k)} \end{bmatrix}, \\ \mathbf{v} &= \begin{bmatrix} \frac{w^* - w^{n-1} - u_{\text{br}}^* + u_{\text{br}}^{n-1}}{2k} - \frac{k(\psi_{\text{sm}}^{n-1/2} g_{\text{sm}}^n - \psi_{\text{mp}}^{n-1/2} g_{\text{mp}}^n)}{M(2+\sigma_m k)} - \frac{\psi_{\text{sm}}^{n-1/2} g_{\text{sm}}^n k}{2\rho_s A h_s (1+\sigma_{0,s} k)} \\ \frac{z_{\text{br}}^* - z_{\text{br}}^{n-1} - w^* + w^{n-1}}{2k} + \frac{k(\psi_{\text{sm}}^{n-1/2} g_{\text{sm}}^n - \psi_{\text{mp}}^{n-1/2} g_{\text{mp}}^n)}{M(2+\sigma_m k)} - \frac{\psi_{\text{mp}}^{n-1/2} g_{\text{mp}}^n k}{2h_p^2 \rho_p H (1+\sigma_{0,p} k)} \end{bmatrix}. \end{aligned}$$

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 $z_{(l,m)}^{n+1}$

15.3. Implementation details

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. One can observe that the string forces the mass downwards through their connection, causing the latter to collide with the plate.

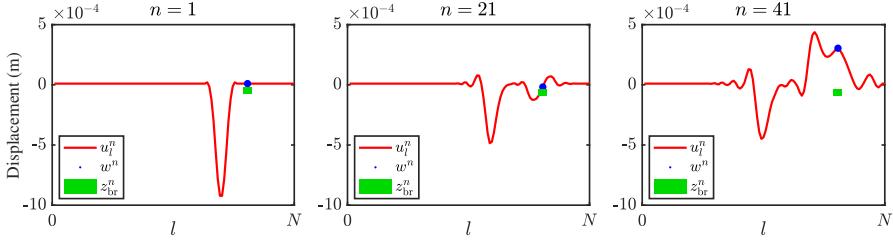


Fig. 15.2: The string of the tromba marina (red) excited using a raised cosine. The string-mass interaction forces the mass (blue) downwards, which collides with the plate (green).

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:

$$\delta_{t+} (\mathfrak{h}_s + \mathfrak{h}_m + \mathfrak{h}_p + \mathfrak{h}_{\text{sm}} + \mathfrak{h}_{\text{mp}}) = -q_s - q_m - q_p - q_B - p_B. \quad (15.16)$$

Here, the total energy of the string is (Eq. (4.29)),

$$\mathfrak{h}_s = \frac{\rho_s A}{2} \|\delta_{t-} u_l^n\|_{d_s}^2 + \frac{T}{2} \langle \delta_{x+} u_l^n, e_{t-} \delta_{x+} u_l^n \rangle_{\underline{d_s}} + \frac{E_s I}{2} \langle \delta_{xx} u_l^n, e_{t-} \delta_{xx} u_l^n \rangle_{\underline{d_s}},$$

the total energy of the mass is (Eq. (9.5))

$$\mathfrak{h}_m = \frac{M}{2} (\delta_{t-} w^n)^2 + \frac{K}{2} w^n e_{t-} w^n,$$

and the total energy of the plate is (Eq. (6.51))

$$\mathfrak{h}_p = \frac{\rho_p H}{2} \|\delta_{t-} z_{l,m}^n\|_{d_p}^2 + \frac{D}{2} \langle \delta_{\Delta} z_{l,m}^n, e_{t-} \delta_{\Delta} z_{l,m}^n \rangle_{\underline{d_p}}.$$

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))

$$q_s = 2\sigma_{0,s}\rho_s A \|\delta_t \cdot u_l^n\|_{d_s}^2 - 2\sigma_{1,s}\rho_s A \langle \delta_t \cdot u_l^n, \delta_t - \delta_{xx} u_l^n \rangle_{d_s},$$

for the mass as (Eq. (9.6))

$$q_m = R (\delta_t \cdot w^n)^2$$

and for the plate as (Eq. (6.50))

$$q_p = 2\sigma_0 \rho_p H \|\delta_t \cdot z_{l,m}^n\|_{d_p}^2 - 2\sigma_1 \rho_p H \langle \delta_t \cdot z_{l,m}^n, \delta_t - \delta_\Delta z_{l,m}^n \rangle_{d_p}.$$

Finally, the power dissipated and supplied by the bow are defined as (Eq. (8.27))

$$q_B = f_B^n \Phi(v_{rel}^n) v_{rel}^n \quad \text{and} \quad p_B = f_B^n \Phi(v_{rel}^n) v_B^n,$$

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.

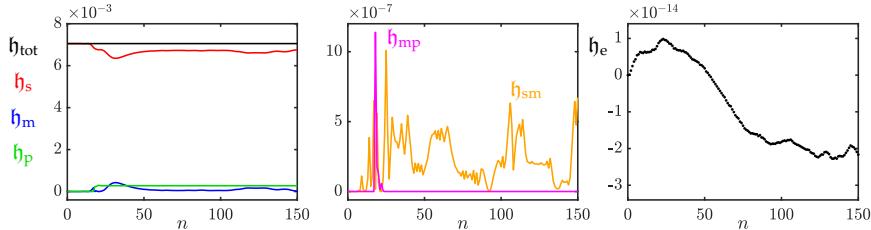


Fig. 15.3: The energy of the tromba marina excited with a raised cosine (see Figure 15.2). The left panel shows the total energy (black) present in the system as well as the energy of the string (red), mass (blue) and plate (green) respectively. The second panel shows the energy of the mass-plate (magenta) 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 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 first-order equations as

described in Section 5.2, and the lip reed has been modelled using the 1-DoF mass spring system presented in Chapter 9. The lip reed has been extended using the collision method described in Chapter 10, details of which will be given in Section 16.3.

An application was made implementing the trombone in real time in C++ using the JUCE framework (see Chapter 13). The application and 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. 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 was split at a location of constant cross-section, the method should continue to work.

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

¹<https://github.com/SilvinWillemsen/cppBrass/releases/>

²youtu.be/Ht5gVNrshYo

16.3. Lip-reed with collision

the pressure grid, rather than that of the particle velocity. That said, if the constraint on the 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. Therefore, the method could easily be applied to a system with a time-varying length rather than a time-varying wave speed. Naturally, in an acoustic tube with a variable cross-section, a change in c would definitely be different than a change in L , which is also why a varying wave speed was not used.

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

$$M\ddot{y} = -Ky - R\dot{y} + S_r\Delta p + \psi g \quad (16.1)$$

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).

³Notice 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 $\sigma_r = R/M$, yields⁴

$$\delta_{tt}y^{n+1/2} = -\omega_0^2\mu_t.y^{n+1/2} - \sigma_r\delta_t.y^{n+1/2} + \frac{S_r}{M}\Delta p^{n+1/2} + \frac{\psi^{n+1/2}g^{n+1/2}}{M}. \quad (16.2)$$

Here,

$$g^{n+1/2} = \frac{\delta_t + \psi^n}{\delta_t.\eta^{n+1/2}}, \quad (16.3)$$

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

$$\delta_{tt}y^{n+1/2} = -\omega_0^2\mu_t.y^{n+1/2} - \sigma_r\delta_t.y^{n+1/2} + \frac{S_r}{M}\Delta p^{n+1/2} + \frac{(\mu_t + \psi^n)g^{n+1/2}}{M}.$$

In the following, the superscript $n + 1/2$ is suppressed for y , Δp , g and η for brevity. Rewriting Eq. (16.3) to

$$\delta_t + \psi^n = g\delta_t.\eta \quad (16.5)$$

and using identity (2.27c), one arrives at

$$\delta_{tt}y = -\omega_0^2\mu_t.y - \sigma_r\delta_t.y + \frac{S_r}{M}\Delta p + \left(\frac{k}{2}g\delta_t.\eta + \psi^n \right) \frac{g}{M}. \quad (16.6)$$

As the barrier is static and placed below y , this implies that

$$\delta_t.\eta = -\delta_t.y, \quad (16.7)$$

and a solution for $y^{n+3/2}$ can be obtained:

$$\alpha_r y^{n+3/2} = 4y^{n+1/2} + \beta_r y^{n-1/2} + \xi_r \Delta p + 4\psi^n \gamma_r, \quad (16.8)$$

⁴Note that the paper uses ω_r and M_r instead of ω_0 and M .

16.3. Lip-reed with collision

with

$$\alpha_r = 2 + \omega_0^2 k^2 + \sigma_r k + g\gamma_r, \quad \beta_r = \sigma_r k - 2 - \omega_0^2 k^2 + g\gamma_r,$$

$$\xi_r = \frac{2S_r k^2}{M}, \quad \text{and} \quad \gamma_r = \frac{gk^2}{2M}.$$

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

$$g^{n+1/2} = \begin{cases} \kappa \sqrt{\frac{K_c(\alpha_c + 1)}{2}} \cdot (\eta^{n+1/2})^{\frac{\alpha_c - 1}{2}}, & \text{if } \eta^{n+1/2} \geq 0, \\ -2 \frac{\psi^n}{\eta^* - \eta^{n-1/2}}, & \text{if } \eta^{n+1/2} < 0 \text{ and } \eta^* \neq \eta^{n-1/2}, \\ 0, & \text{if } \eta^{n+1/2} < 0 \text{ and } \eta^* = \eta^{n-1/2}, \end{cases}$$

where parameters are as in Eq. (10.15). Furthermore, $\eta^* = -H_0 - y^*$ where

$$y^* = \frac{4}{\alpha_r^*} y^{n+1/2} + \frac{\beta_r^*}{\alpha_r^*} y^{n-1/2} + \frac{\xi_r}{\alpha_r^*} \Delta p^*, \quad (16.10)$$

is the update equation of the system without the effect of the collision and

$$\alpha_r^* = 2 + \omega_0^2 k^2 + \sigma_r k, \quad \text{and} \quad \beta_r^* = \sigma_r k - 2 - \omega_0^2 k^2,$$

are the coefficients in Eq. (16.8) without the collision terms (as found in Eq. (9.15)). Notice that ξ_r is unchanged. Finally, Δp^* is the pressure difference calculated using Eq. (9.28), 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

$$\begin{aligned} \frac{2}{k} (\delta_{t-} - \delta_{t-}) y &= -\omega_0^2 (k\delta_{t-} + e_{t-}) y - \sigma_r \delta_{t-} y + \frac{S_r}{M} \Delta p + \left(-\frac{k}{2} g \delta_{t-} y + \psi^n \right) \frac{g}{M}, \\ a_1^n \delta_{t-} y - a_2 \Delta p - a_3^n &= 0, \end{aligned}$$

with

$$a_1^n = \frac{2}{k} + \omega_0^2 k + \sigma_r + \frac{g^2 k}{2M} \geq 0, \quad a_2 = \frac{S_r}{M} \geq 0,$$

and $a_3^n = \left(\frac{2}{k} \delta_{t-} - \omega_0^2 e_{t-} \right) y + \frac{g}{M} \psi^n.$

Note that a_1^n 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 Δp^* , 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

$$\psi^{n+1} = \psi^n - \frac{g}{2} \left(y^{n+3/2} - y^{n-1/2} \right). \quad (16.11)$$

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 (9.32) can be extended to

$$\delta_{t+}(\mathfrak{h}_t + \mathfrak{h}_r) = -\mathfrak{q}_r - \mathfrak{p}_r + (\mu_{t+} \psi^n) \frac{\delta_{t+} \psi^n}{\delta_t \eta^{n+1/2}} (\delta_t y^{n+1/2})$$

which, using Eq. (16.7), yields

$$\delta_{t+}(\mathfrak{h}_t + \mathfrak{h}_r) = -\mathfrak{q}_r - \mathfrak{p}_r - (\mu_{t+} \psi^n) (\delta_{t+} \psi^n),$$

with

$$\mathfrak{h}_c = \frac{(\psi^n)^2}{2}.$$

Part VI

Conclusions and Perspectives

Chapter 17

Conclusions and Perspectives

This chapter concludes this thesis by providing a summary of the presented work. Finally, some perspectives 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. Parts II, III and IV provided detailed information about the physical models used for the contributions of this PhD project. Finally, 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.

shortly describe

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 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

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

¹One could always set the states of the system to 0 when a parameter is changed, and re-initialise 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.

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.

17.2. Perspectives and future work

check whether
all references
are used

Chapter 17. Conclusions and Perspectives

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.

References

- [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.

References

- [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/>

References

- [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.

References

- [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, "Syst'eme général des intervalles des sons, et son application 'a tous les syst'emes et 'a 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.

References

- [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.

References

- [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," *J. Acoust. Soc. Am.*, 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.

References

- [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.

References

- [100] S. Bilbao, "Direct simulation of reed wind instruments," *Computer Music Journal*, vol. 33, no. 4, 2009.
- [101] V. Chatzioannou 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. Chatzioannou, "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 quadratization," *Journal of the Acoustical Society of America (JASA)*, vol. 149, pp. 3502–3516, 2021.
- [104] N. Lopes, T. Hélie, 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/91d10f6279af7fd0da7fb65fb7647da>
- [111] G. F. Carrier, "On the non-linear vibration problem of the elastic string," *Quarterly of Applied Mathematics*, vol. 3, pp. 157–165, 1945.

References

- [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.

References

- [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/>

check whether
to sort refer-
ences or not

References

Part VII

Papers

Paper A

Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Silvin Willemse, Nikolaj Andersson, Stefania Serafin
and Stefan Bilbao

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp.
275–280, 2019.

Paper B

Physical Models and Real-Time Control with the Sensel Morph

Silvin Willemse, Stefan Bilbao, Nikolaj Andersson
and Stefania Serafin

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp.
95–96, 2019.

Paper C

Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes

Silvin Willemse, Stefan Bilbao and Stefania Serafin

The paper has been published in the
Proceedings of the 22nd International Conference on Digital Audio Effects
(DAFx-19), pp. 40–46, 2019.

Paper D

Real-time Implementation of a Physical Model of the Tromba Marina

Silvin Willemsen, Stefania Serafin, Stefan Bilbao and Michele
Ducceschi

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp.
161–168, 2020.

Paper E

Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling

Silvin Willemse, Razvan Paisa and Stefania Serafin

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp.
300–307, 2020.

Paper F

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study

Silvin Willemse, Anca-Simona Horvath and Mauro Nascimben

The paper has been published in the
Proceedings of the 17th Sound and Music Computing (SMC) Conference, pp.
292–299, 2020.

Paper G

Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations

Silvin Willemse, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects
(DAFx2020in21), 2021.

Paper H

A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes

Silvin Willemse, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

The paper has been published in the
Proceedings of the 23rd International Conference on Digital Audio Effects
(DAFx2020in21), 2021.

Paper H.

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.
- $\sigma_{1,s}$ in Eq. (21) should be $\sigma_{1,p}$.
- 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 should be multiplied by ρH in order for the stability condition to hold.
- The stability condition is wrong. It should be:

$$h \geq \sqrt{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}. \quad (\text{A.1})$$

- The unit for membrane tension is N/m.

Appendix A. Paper Errata

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* × *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

$$a_{ij} = a_{ji}. \quad (\text{B.1})$$

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.

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

(a) A 3×5 matrix \mathbf{A} .

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

Fig. 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

$$c_{ij} = \sum_{k=1}^K a_{ik} b_{kj}, \quad (\text{B.2})$$

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 **A** and a $M \times N$ matrix **B** with $L \neq N$. The multiplication **AB** is defined as the number of columns of matrix **A** (M) is equal to the number of rows of matrix **B** (also M). The result, **C**, is a $L \times N$ matrix. The multiplication **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

$$\overbrace{(L \times M)}^{\mathbf{A}} \cdot \overbrace{(M \times N)}^{\mathbf{B}} = \overbrace{(L \times N)}^{\mathbf{C}}. \quad (\text{B.3})$$

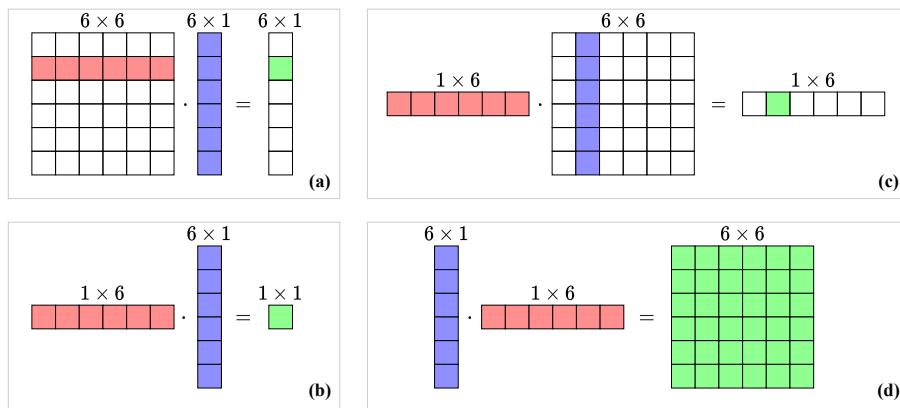


Fig. B.2: Visualisation of valid matrix multiplications (see Eq. (B.2)). The “inner” dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of “outer” dimensions (rows of the left matrix and columns of the right).

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 **A** is invertable if there exists a matrix **B** such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \quad (\text{B.4})$$

This matrix **B** is then called the *inverse* of **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:

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \quad (\text{B.5})$$

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}

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \quad (\text{B.6})$$

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 \mathbf{w}

$$\mathbf{u} = \text{inv}(\mathbf{A}) * \mathbf{w};$$

or more compactly, by using the ‘\’ operator:

$$\mathbf{u} = \mathbf{A} \backslash \mathbf{w};$$

B.4 Eigenvalue problems

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

$$\mathbf{A}\phi = \lambda\phi. \quad (\text{B.7})$$

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

$$\lambda_p = \text{eig}_p(\mathbf{A}), \quad (\text{B.8})$$

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 corresponsding eigenvalues are given in a $p \times 1$ column vector `lambda`.

Appendix B. Matrices

Appendix C

Code Snippets

C.1 Mass-spring system (Section 2.3)

```
%% Initialise variables
fs = 44100; % sample rate [Hz]
k = 1 / fs; % time step [s]
lengthSound = fs; % length of the simulation (1 second) [samples]

f0 = 440; % fundamental frequency [Hz]
omega0 = 2 * pi * f0; % angular (fundamental) frequency [Hz]
M = 1; % mass [kg]
K = omega0^2 * M; % spring constant [N/m]

%% initial conditions (u0 = 1, d/dt u0 = 0)
u = 1;
uPrev = 1;

% initialise output vector
out = zeros(lengthSound, 1);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.35)
    uNext = (2 - K * k^2 / M) * u - uPrev;

    out(n) = u;

    % Update system states
    uPrev = u;
    u = uNext;
end
```

C.2 1D wave equation (Section 2.4)

```

%% Initialise variables
fs = 44100; % Sample rate [Hz]
k = 1 / fs; % Time step [s]
lengthSound = fs; % Length of the simulation (1 second) [samples]

c = 300; % Wave speed [m/s]
L = 1; % Length [m]
h = c * k; % Grid spacing [m] (from CFL condition)
N = floor(L/h); % Number of intervals between grid points
h = L / N; % Recalculation of grid spacing based on integer N

lambdaSq = c^2 * k^2 / h^2; % Courant number squared

% Boundary conditions ([D]irichlet or [N]eumann)
bcLeft = "D";
bcRight = "D";

%% Initialise state vectors (one more grid point than the number of
% intervals)
uNext = zeros(N+1, 1);
u = zeros(N+1, 1);

% Initial conditions (raised cosine)
loc = round(0.8 * N); % Center location
halfWidth = round(N/10); % Half-width of raised cosine
width = 2 * halfWidth; % Full width
rcX = 0:width; % x-locations for raised cosine

rc = 0.5 - 0.5 * cos(2 * pi * rcX / width); % raised cosine
u(loc-halfWidth : loc+halfWidth) = rc; % initialise current state

% Set initial velocity to zero
uPrev = u;

% Range of calculation
range = 2:N;

% Output location
outLoc = round(0.3 * N);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.44)
    uNext(range) = (2 - 2 * lambdaSq) * u(range) ...
        + lambdaSq * (u(range+1) + u(range-1)) - uPrev(range);

    % boundary updates Eq. (2.49)
    if bcLeft == "N"
        uNext(1) = (2 - 2 * lambdaSq) * u(1) - uPrev(1) ...
            + 2 * lambdaSq * u(2);
    end
end

```

C.3. 2D wave equation (Section 6.2)

```

    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;
Nyv = Ny - 1;
Dxx = toeplitz([-2, 1, zeros(1, Nxu-2)]);
Dyy = toeplitz([-2, 1, zeros(1, Nyv-2)]);

% Kronecker sum
D = kron(speye(Nxu), Dyy) + kron(Dxx, speye(Nyv));
D = D / h^2;

```

Appendix C. Code Snippets

```
% 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 $2\sigma_1\partial_t\partial_x^2u$. Recalling the compact version of the PDE of the stiff string in Eq. (4.4):

$$\partial_t^2u = c^2\partial_x^2u - \kappa^2\partial_x^4u - 2\sigma_0\partial_tu + 2\sigma_1\partial_t\partial_x^2u, \quad (\text{D.1})$$

consider first the frequency-independent damping term $-2\sigma_0\partial_tu$. The more positive the velocity ∂_tu 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 $\partial_t\partial_x^2u$, which literally describes the ‘rate of change of the curvature’ of the string.

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

$$\begin{aligned} \delta_{xx}f(x) &= \frac{1}{h^2} (f(-1) - 2f(0) + f(1)), \\ &= \frac{1}{1^2} ((-1)^2 - 2 \cdot 0^2 + 1^2), \\ &= (1 - 0 + 1) = 2. \end{aligned} \quad (\text{D.2})$$

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:

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

So the rate of change of the curvature is positive, i.e., the already positively curved function x^2 gets more curved over time.

If the curvature around a point along a string gets more positive (or less negative) over time, the force applied to that point will be positive, effectively ‘trying’ to reduce the curvature. Vice versa, if the curvature around a point along a string gets more negative (or less positive) over time, the force applied will be negative, again ‘trying’ to reduce the curvature.

From an auditory point of view, higher curvature generally means higher frequency. As the frequency-dependent damping term reduces 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 $\sim 10^{-307}$! 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 E. Considerations in real-time FD schemes

Appendix F

Derivations

This appendix contains derivations of several equations used in this thesis.

F.1 von Neumann analysis damped stiff string (Section 4.3)

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):

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

one can rewrite this to the form in Eq. (3.25), and using $\mathcal{S} = \sin^2(\beta h/2)$ for brevity, yields

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

Stability of the system can then be proven using condition (3.26), and substi-

Appendix F. Derivations

tuting the coefficients into this condition yields

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

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

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

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

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

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

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

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

$$h \geq \sqrt{\frac{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}, \quad (\text{F.1})$$

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

F.2 von Neumann analysis implicit damped stiff string (Section 4.6.1)

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):

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

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

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

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:

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

Again, the first condition is always satisfied due to the non-negativity of all

Appendix F. Derivations

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:

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

and yields the following stability condition:

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

F.3 Webster's update equation (Eq. (5.9))

This section derives the update equation for Webster's equation in Eq. (5.9):

$$\begin{aligned} \frac{\bar{S}}{k^2} (\Psi_l^{n+1} - 2\Psi_l^n + \Psi_l^{n-1}) &= c^2 \left((\delta_{x-} S_{l+1/2})(\mu_{x-} \delta_{x+} \Psi_l^n) \right. \\ &\quad \left. + (\mu_{x-} S_{l+1/2})(\delta_{x-} \delta_{x+} \Psi_l^n) \right), \\ \Psi_l^{n+1} - 2\Psi_l^n + \Psi_l^{n-1} &= \frac{c^2 k^2}{\bar{S}} \left(\underbrace{\frac{1}{h} (S_{l+1/2} - S_{l-1/2})}_{\lambda = \frac{ck}{h}} \underbrace{\frac{1}{2h} (\overbrace{\Psi_{l+1}^n - \Psi_{l-1}^n}^{\mu_{x+} \delta_{x-} \Psi_l^n = \delta_x \cdot \Psi_l^n})}_{\frac{\lambda^2}{2\bar{S}}} \right. \\ &\quad \left. + \frac{1}{2} (S_{l+1/2} + S_{l-1/2}) \frac{1}{h^2} (\Psi_{l+1}^n - 2\Psi_l^n + \Psi_{l-1}^n) \right), \\ \Psi_l^{n+1} &= 2\Psi_l^n - \Psi_l^{n-1} + \underbrace{\frac{\lambda^2}{2\bar{S}}}_{\lambda = \frac{ck}{h}} \left(S_{l+1/2} \Psi_{l+1}^n - S_{l+1/2} \Psi_{l-1}^n \right. \\ &\quad \left. - S_{l-1/2} \Psi_{l+1}^n + S_{l-1/2} \Psi_{l-1}^n + S_{l+1/2} \Psi_{l+1}^n + S_{l+1/2} \Psi_{l-1}^n \right. \\ &\quad \left. + S_{l-1/2} \Psi_{l+1}^n + S_{l-1/2} \Psi_{l-1}^n - 2(S_{l+1/2} + S_{l-1/2}) \Psi_l^n \right), \\ \Psi_l^{n+1} &= 2\Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2}{2\bar{S}} \left(2S_{l+1/2} \Psi_{l+1}^n + 2S_{l-1/2} \Psi_{l-1}^n - 4\bar{S} \Psi_l^n \right), \\ \Psi_l^{n+1} &= 2\Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}} \Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}} \Psi_{l-1}^n - 2\lambda^2 \Psi_l^n, \\ \Psi_l^{n+1} &= 2(1 - \lambda^2) \Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}} \Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}} \Psi_{l-1}^n. \end{aligned}$$

F.4 Boundary terms Webster's equation (Eq. (5.31))

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.

Starting with the second term in the energy balance in Eq. (5.27):

$$-c^2 \langle \delta_t \cdot \Psi_l^n, \delta_{x-} (S_{l+1/2}(\delta_{x+} \Psi_l^n)) \rangle_d^{\epsilon_l, \epsilon_r}, \quad (\text{F.2})$$

and using identity (3.15a):

$$\begin{aligned} \langle f_l^n, \delta_{x-} g_l^n \rangle_d^{\epsilon_l, \epsilon_r} &= -\langle \delta_{x+} f_l^n, g_l^n \rangle_d + f_N^n g_{N-1}^n - f_0^n g_{-1}^n \\ &\quad + \frac{\epsilon_r}{2} f_N^n (g_N^n - g_{N-1}^n) + \frac{\epsilon_l}{2} f_0^n (g_0^n - g_{-1}^n), \end{aligned}$$

this can be rewritten to (with $f = \delta_t \cdot \Psi$ and $g = S_{l+1/2}(\delta_{x+} \Psi)$)

$$-c^2 \langle \delta_t \cdot \Psi_l^n, \delta_{x-} (S_{l+1/2}(\delta_{x+} \Psi_l^n)) \rangle_d^{\epsilon_l, \epsilon_r} = c^2 \langle \delta_t \cdot \delta_{x+} \Psi_l^n, (S_{l+1/2}(\delta_{x+} \Psi_l^n)) \rangle_d - b,$$

where

$$b = b_r - b_l, \quad (\text{F.3})$$

with

$$\begin{aligned} b_r &= c^2 (\delta_t \cdot \Psi_N^n) \left(S_{N-1/2} \underbrace{(\delta_{x+} \Psi_{N-1}^n)}_{(\delta_{x-} \Psi_N^n)} \right) \\ &\quad + \frac{\epsilon_r}{2} (\delta_t \cdot \Psi_N^n) \left(S_{N+1/2}(\delta_{x+} \Psi_N^n) - S_{N-1/2} \underbrace{(\delta_{x+} \Psi_{N-1}^n)}_{(\delta_{x-} \Psi_N^n)} \right), \end{aligned}$$

and

$$\begin{aligned} b_l &= c^2 (\delta_t \cdot \Psi_0^n) \left(S_{1/2}(\delta_{x+} \Psi_0^n) \right) \\ &\quad - \frac{\epsilon_l}{2} (\delta_t \cdot \Psi_0^n) \left(S_{1/2}(\delta_{x+} \Psi_0^n) - S_{-1/2} \underbrace{(\delta_{x+} \Psi_{-1}^n)}_{(\delta_{x-} \Psi_0^n)} \right). \end{aligned}$$

This can be rewritten to

$$b_r = c^2 (\delta_t \cdot \Psi_N^n) \left(\frac{\epsilon_r}{2} S_{N+1/2}(\delta_{x+} \Psi_N^n) + \left(1 - \frac{\epsilon_r}{2} \right) S_{N-1/2}(\delta_{x-} \Psi_N^n) \right), \quad (\text{F.4})$$

$$b_l = c^2 (\delta_t \cdot \Psi_0^n) \left(\frac{\epsilon_l}{2} S_{-1/2}(\delta_{x-} \Psi_0^n) + \left(1 - \frac{\epsilon_l}{2} \right) S_{1/2}(\delta_{x+} \Psi_0^n) \right). \quad (\text{F.5})$$

Then, for the centred radiating boundary condition in Eq. (5.17) to be strictly dissipative, i.e., $\delta_x \cdot \Psi_l^n = 0 \Rightarrow b_r = 0$, the special choice for $\epsilon_r =$

Appendix F. Derivations

$S_{N-1/2}/\mu_{xx}S_N$ needs to be made:

$$\begin{aligned}
\mathfrak{b}_r &= c^2(\delta_t \cdot \Psi_N^n) \left(\frac{S_{N-1/2}}{2\mu_{xx}S_N} S_{N+1/2}(\delta_{x+} \Psi_N^n) + \left(1 - \frac{S_{N-1/2}}{2\mu_{xx}S_N}\right) S_{N-1/2}(\delta_{x-} \Psi_N^n) \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(\frac{S_{N+1/2}}{2\mu_{xx}S_N} (\delta_{x+} \Psi_N^n) + \left(1 - \frac{S_{N-1/2}}{2\mu_{xx}S_N}\right) (\delta_{x-} \Psi_N^n) \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{S_{N-1/2}}{2\mu_{xx}S_N} \right) \left(\frac{\frac{S_{N+1/2}(\delta_{x+} \Psi_N^n)}{2\mu_{xx}S_N}}{\left(1 - \frac{S_{N-1/2}}{2\mu_{xx}S_N}\right)} + \delta_{x-} \Psi_N^n \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) \left(\frac{\frac{S_{N+1/2}(\delta_{x+} \Psi_N^n)}{2\mu_{xx}S_N}}{\left(\frac{2\mu_{xx}S_N - S_{N-1/2}}{2\mu_{xx}S_N} \right)} + \delta_{x-} \Psi_N^n \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) \left(\frac{S_{N+1/2}(\delta_{x+} \Psi_N^n)}{2\mu_{xx}S_N - S_{N-1/2}} + \delta_{x-} \Psi_N^n \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) \left(\frac{S_{N+1/2}(\delta_{x+} \Psi_N^n)}{S_{N+1/2} + S_{N-1/2} - S_{N-1/2}} + \delta_{x-} \Psi_N^n \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) (\delta_{x+} \Psi_N^n + \delta_{x-} \Psi_N^n), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) \left(\frac{1}{h} (\Psi_{N+1}^n - \Psi_N^n + \Psi_N^n - \Psi_{N-1}^n) \right), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} \left(1 - \frac{\epsilon_r}{2} \right) 2(\delta_x \cdot \Psi_N^n), \\
&= c^2(\delta_t \cdot \Psi_N^n) S_{N-1/2} (2 - \epsilon_r) (\delta_x \cdot \Psi_N^n).
\end{aligned} \tag{F.6}$$

The same can be done for \mathfrak{b}_l with $\epsilon_l = S_{1/2}/\mu_{xx}S_0$ to get

$$\mathfrak{b}_l = c^2(\delta_t \cdot \Psi_0^n) S_{1/2} (2 - \epsilon_l) (\delta_x \cdot \Psi_0^n). \tag{F.7}$$

F.5 Levine and Schwinger radiation model update (Eq. (5.59))

This section provides a derivation for the update equation in Eq. (5.59) and follows [62, Sec. 4.1.3, pp. 109–111]. Recalling system (5.57)

$$\bar{v} = \mu_{t+} v_{(1)} + \frac{1}{R_2} \mu_{t+} p_{(1)} + C_r \delta_{t+} p_{(1)}, \tag{F.8a}$$

$$\bar{p} = L_r \delta_{t+} v_{(1)}, \tag{F.8b}$$

$$\bar{p} = \left(1 + \frac{R_1}{R_2} \right) \mu_{t+} p_{(1)} + R_1 C_r \delta_{t+} p_{(1)}, \tag{F.8c}$$

F.5. Levine and Schwinger radiation model update (Eq. (5.59))

where $\bar{p}^{n+1/2}$ and $\bar{v}^{n+1/2}$ are related to the tube by (see Eq. (5.58))

$$\bar{p} = \mu_{t+} p_N^n, \quad (\text{F.9})$$

$$\bar{S}_N \bar{v} = \mu_{x-} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} \right), \quad (\text{F.10})$$

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 $l = N$

$$p_N^{n+1} = p_N^n - \frac{\rho_0 c \lambda}{\bar{S}_N} \left(S_{N+1/2} v_{N+1/2}^{n+1/2} - S_{N-1/2} v_{N-1/2}^{n+1/2} \right), \quad (\text{F.11})$$

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

$$p_N^{n+1} = p_N^n - \frac{2\rho_0 c \lambda}{\bar{S}_N} \left(\bar{S}_N \bar{v} - S_{N-1/2} v_{N-1/2}^{n+1/2} \right). \quad (\text{F.12})$$

A definition for \bar{v} can then be found by first expanding Eq. (F.8a) to

$$\bar{v} = \frac{1}{2} \left(v_{(1)}^{n+1} + v_{(1)}^n \right) + \left(\frac{1}{2R_2} + \frac{C_r}{k} \right) p_{(1)}^{n+1} + \left(\frac{1}{2R_2} - \frac{C_r}{k} \right) p_{(1)}^n, \quad (\text{F.13})$$

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

$$v_{(1)}^{n+1} = \frac{k}{L_r} \bar{p} + v_{(1)}^n, \quad (\text{F.14})$$

and Eq. (F.8c) to

$$\begin{aligned} \bar{p} &= \left(1 + \frac{R_1}{R_2} \right) \mu_{t+} p_{(1)} + R_1 C_r \delta_{t+} p_{(1)}, \\ \bar{p} &= \frac{1}{2} \left(1 + \frac{R_1}{R_2} \right) \left(p_{(1)}^{n+1} + p_{(1)}^n \right) + \frac{R_1 C_r}{k} \left(p_{(1)}^{n+1} - p_{(1)}^n \right), \\ \left(\frac{1}{2} + \frac{R_1}{2R_2} + \frac{R_1 C_r}{k} \right) p_{(1)}^{n+1} &= \bar{p} + \left(\frac{R_1 C_r}{k} - \frac{1}{2} - \frac{R_1}{2R_2} \right) p_{(1)}^n, \end{aligned}$$

Appendix F. Derivations

and finally solved for $p_{(1)}^{n+1}$ as

$$p_{(1)}^{n+1} = \underbrace{\left(\frac{2R_2k}{2R_1R_2C_r + k(R_1 + R_2)} \right) \bar{p}}_{\zeta_1} + \underbrace{\left(\frac{2R_1R_2C_r - k(R_1 + R_2)}{2R_1R_2C_r + k(R_1 + R_2)} \right) p_{(1)}^n}_{\zeta_2}. \quad (\text{F.15})$$

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

$$\begin{aligned} \bar{v} &= \frac{1}{2} \left(\frac{k}{L_r} (\mu_t + p_N^n) + 2v_{(1)}^n \right) + \left(\frac{1}{2R_2} + \frac{C_r}{k} \right) \zeta_1 \mu_t + p_N^n \\ &\quad + \left(\frac{1}{2R_2} + \frac{C_r}{k} \right) \zeta_2 p_{(1)}^n + \left(\frac{1}{2R_2} - \frac{C_r}{k} \right) p_{(1)}^n, \\ \bar{v} &= \underbrace{\left(\frac{k}{2L_r} + \frac{\zeta_1}{2R_2} + \frac{C_r \zeta_1}{k} \right) \mu_t + p_N^n}_{\zeta_3} + v_{(1)}^n + \underbrace{\left(\frac{\zeta_2 + 1}{2R_2} + \frac{C_r \zeta_2 - C_r}{k} \right) p_{(1)}^n}_{\zeta_4}. \end{aligned} \quad (\text{F.16})$$

Finally, substituting this definition for \bar{v} into Eq. (F.12), yields

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

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. (5.59). 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.6 Derivatives for Newton-Raphson for the elasto-plastic friction model (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)

$$g_1(v^n, z^n) = \left(\frac{2}{k} + 2\sigma_0 \right) v^n + \|J_l(x_B^n)\|_d^2 \frac{f(v^n, z^n)}{\rho A} + b^n = 0,$$

F.6. Derivatives for Newton-Raphson for the elasto-plastic friction model (Eq. (8.41))

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

$$\begin{aligned} \frac{\partial g_1}{\partial v} &= \frac{2}{k} + 2\sigma_0 + \frac{s_1 \|J_l(x_B^n)\|_d^2}{\rho A} \frac{\partial r}{\partial v} + \frac{s_2 \|J_l(x_B^n)\|_d^2}{\rho A}, \\ \frac{\partial g_1}{\partial z} &= \frac{s_0 \|J_l(x_B^n)\|_d^2}{\rho A} + \frac{s_1 \|J_l(x_B^n)\|_d^2}{\rho A} \frac{\partial r}{\partial z}, \\ \frac{\partial g_2}{\partial v} &= \frac{\partial r}{\partial v} \\ \frac{\partial g_2}{\partial z} &= \frac{\partial r}{\partial z} - \frac{2}{k}. \end{aligned}$$

Recalling from Eq. (8.36) that

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

its derivatives can be computed as

$$\begin{aligned} \frac{\partial r}{\partial v} &= 1 - z^n \left(\frac{(\alpha^n + \frac{\partial \alpha^n}{\partial v} v^n) z_{ss}^n - \frac{\partial z_{ss}^n}{\partial v} \alpha^n v^n}{(z_{ss}^n)^2} \right), \\ \frac{\partial r}{\partial z} &= -\frac{v^n}{z_{ss}^n} \left(\frac{\partial \alpha^n}{\partial z} z^n + \alpha^n \right), \end{aligned}$$

with

$$\begin{aligned} \frac{\partial \alpha^n}{\partial v} &= \text{sgn}(z_{ss}^n) \frac{\partial z_{ss}^n}{\partial v} \frac{z_{ba} - |z^n|}{(|z_{ss}^n| - z_{ba})^2} \frac{\pi}{2} \cos(\text{sgn}(z^n) \Phi), \\ \frac{\partial \alpha^n}{\partial z} &= \frac{\text{sgn}(z^n) \pi \cos(\text{sgn}(z^n) \Phi)}{2(|z_{ss}^n| - z_{ba})}, \\ \frac{\partial z_{ss}^n}{\partial v} &= -\frac{2|v^n|}{v_S^2 s_0} (f_S - f_C) e^{-(v^n/v_S)^2}. \end{aligned}$$