The Emulated Ensemble

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

Ph.D. Dissertation
Silvin Willemsen

Dissertation submitted July 13, 2021

Curriculum Vitae

Silvin Willemsen

Here is the CV text.

Curriculum Vitae

Abstract

Digital musical instruments exist in large quantities and numerous strategies to virtualise traditional instruments exist. 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 make 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 and broaden 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.

Many physical modelling techniques exist, where finite-difference time-domain (FDTD) methods 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, the violin 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 parameters in

Abstract

FDTD-based musical instrument simulations allowing for physically impossible instrument manipulations. Finally, this work doubles as an aid for beginners in the field of musical instrument simulations based on FDTD methods, and aims to provide a low-entry-level explanation of the literature and theory that the physical models are based on.

Resumé

Digitale musikinstrumenter findes i store mængder, og der findes adskillige strategier til virtualisering af traditionelle instrumenter. Selvom man kunne oprette digitale musikinstrumenter ved hjælp af optagelser eller samples af deres virkelige kolleger, bliver instrumenternes spilbarhed og interaktion ikke fanget. I stedet kunne en simulering af instrumentets underliggende fysik oprettes og er meget mere fleksibel for spillerinteraktion. Denne *fysiske model* gør det muligt for en musiker at være meget mere udtryksfuld, når han spiller det digitale instrument, end hvis der skulle bruges statiske optagelser. Brug af ad hoc hardware til at kontrollere simuleringen kan muligvis få det simulerede instrument til at føles identisk med originalen.

Anvendelser af fysisk modellering af musikinstrumenter inkluderer at simulerer instrumenter der ikke kan afspilles, da de er for sjældne eller sårbare. En model af instrumentets underliggende fysik kan potentielt genoplive instrumentet og gøre det tilgængeligt for offentligheden igen. Desuden, da en simulering ikke er begrænset af fysikens love, kunne man udvide mulighederne for det originale instrument. Egenskaber som et instruments materiale eller geometri kunne ændres dynamisk og udvide musikerens udtryk. Man kunne endda forestille sig fysisk umulige musikinstrumenter, der stadig udviser en naturlig lyd på grund af de underliggende modeller.

Der findes mange fysiske modelleringsteknikker, hvor FDTD (finite-difference time-domain) metoder har en fordel med hensyn til generalitet og fleksibilitet med hensyn til de systemer, de kan modellere. En ulempe ved disse metoder er, at de er ret beregningsdygtige, og selvom der har eksisteret mange meget nøjagtige modeller baseret på disse metoder i årevis, er computerkraften til at køre dem i real time først for nylig blevet tilgængelig. Den største udfordring er således at køre simuleringerne i real time for at muliggøre korrekt spillerinteraktion.

Dette arbejde præsenterer udviklingen og realtidsimplementeringen af forskellige fysiske modeller af traditionelle musikinstrumenter baseret på FDTD metoder. Disse instrumenter inkluderer trombonen, violinen og mere uklare instrumenter såsom den hurdy gurdy og den tromba marina. Desuden præsenteres en ny metode, der baner vejen for dynamiske parametre i FDTD-

baserede musikinstrumentsimuleringer, der muliggør fysisk umulige instrumentmanipulationer. Endelig fungerer dette arbejde som et hjælpemiddel til begyndere inden for simuleringer af musikinstrumenter baseret på FDTD-metoder og sigter mod at give en low-entry-level forklaring af litteraturen og teorien, som de fysiske modeller er baseret på.

Preface

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

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

Many concepts that seemed impossible at the beginning

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

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

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

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

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

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

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

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

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

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

Make physical modelling more accessible to the non-physicist. Also supports reproducibility of science and lowers the entry level

Interested in physically impossible manipulations of now-virtual instruments.

Could be fun to include this :) : "I'm a musician. Will I be out of a job if you keep making physical models?" Physical modelling is not here to replace the original instruments and the musicians playing them. Instead, it can be used as a tool to understand the physics of existing instruments and possibly go beyond. Simulated instruments are not restricted by physics anymore and could provide new ways of expression for the musician.

Acknowledgements

I would like to thank my mom..

<div align="right">

Silvin Willemsen
Aalborg University, July 13, 2021

</div>

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, which are also included in Part III, papers with a supervisory role, and finally, other publications from various collaborative efforts.

Main Publications

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

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

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

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

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

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

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

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

Publications with a Supervisory Role

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

[S2] T. Lasickas, S. Willemsen, and S. Serafin, "Real-time implementation of the shamisen using finite difference schemes," in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021, pp. 100–107.

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

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

Other Publications

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

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

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

List of Symbols

The list of symbols found below contains often-used symbols in the thesis in the context that they are normally used. Depending on the context they might carry a different meaning (y being displacement of the lip-reed in Chapter **??** but the vertical spatial coordinate for 2D systems in fx. Chapter 6). Some might also be accompanied by a subscript in the main document

Symbol	Description	Unit
α	Fractional part of	
A	Cross-sectional area of string	m^2
c	Wave speed	m/s
$\frac{d^n}{dt^n}$	n^{th} order derivative with respect to t	-
∂_t^n	n^{th} order partial derivative with respect to t	-
δ_{t+}, δ_{t-}, $\delta_{t\cdot}$	Forward, backward and centred difference in time operator	-
δ_{x+}, δ_{x-}, $\delta_{x\cdot}$	Forward, backward and centred difference in space operator	-
δ_{tt}	Second order difference in time operator	-
δ_{xx}	Second order difference in space operator	-
δ_{xxxx}	Fourth order difference in space operator	-
μ_{t+}, μ_{t-}, $\mu_{t\cdot}$	Forward, backward and centred average in time operator	-
μ_{x+}, μ_{x-}, $\mu_{x\cdot}$	Forward, backward and centred average in space operator	-
μ_{tt}	Second order average in time operator	-
μ_{xx}	Second order average in space operator	-
E	Young's Modulus	Pa (kg·m^{-1}·s^{-2})
f	Force	N
f	Frequency	Hz

Other Publications

Symbol	Description	Unit
f_s	Sample rate	Hz
F	Scaled force	depends on system
h	Grid spacing	m
H	Membrane / Plate thickness	m
I	Area moment of inertia	m^4
l	Spatial index to grid function	-
L	Length	m
k	Time step $(= 1/f_\mathrm{s})$	s
K	Spring coefficient	N/m
κ	Stiffness coefficient	m^2/s (1D) $\mathrm{m}^4{\cdot}\mathrm{s}^{-2}$ (2D)
n	Sample index to grid function	-
N	Number of points string	-
$\mathbb{N}^0$	Set of non-negative integers	-
p	Pressure	Pa
r	Radius	m
S	Cross-sectional area (brass)	m^2
t	Time	s
T	Tension	N (1D) N/m (2D)
u	State variable	m
v	Particle velocity	m/s
x	Spatial dimension (horizontal for 2D systems)	m
y	Vertical spatial dimension	m
γ	Scaled wave speed	s^{-1}
λ	Courant number for 1D wave eq. $(= ck/h)$	-
μ	Stiffness free parameter	-
ν	Poisson's ratio	-
η	Relative displacement spring	m
ρ	Material density	$\mathrm{kg}{\cdot}\mathrm{m}^{-3}$

Operations

$\mathfrak{I}(\cdot)$	Imaginary part of	
$\mathfrak{R}(\cdot)$	Real part of	

Symbol	Description	Unit
$\lfloor \cdot \rfloor$	Flooring operation	
$\lceil \cdot \rceil$	Ceiling operation	

Subscripts

c	Connection	
p	Plate	
s	String	

Other Publications

List of Abbreviations

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

Other Publications

Contents

Contents

Contents

Todo list

Contents

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

At the time of writing, already an uncountable number of digital musical instruments exists. From digital keyboards that can create sounds from a various real (and non-real) instruments to digital instrument plug-ins that music producers use in their Digital Audio Workstations (DAWs) to create their music. Many of these digital instruments are based on samples, or recordings, of their real-life counterparts, while others use computationally cheap methods to generate sounds, some inspired by physical musical instruments. In the 1960s, for example, efficient sinusoidal-based and filter-based sound synthesis techniques such as additive synthesis, subtractive synthesis, or FM (frequency modulation) synthesis were invented [1, 2]. The latter became widely popular through the Yamaha DX7 synthesiser created in 1983 that synthesised sounds based solely on this technique [3]. Through a simple change of variables the same formula could generate sounds ranging from brass instruments to drums.

These techniques are referred to as *spectral modelling* methods, where the manipulation of sinusoids or filtering noise would produce (harmonic) sounds, that 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 [4].

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

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

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

1.1   A Brief History

Most likely the very first example of a physically modelled musical sound is the "Bicycle Built for Two" by Kelly, Lochbaum, and Matthews in 1961[2]. It uses what later got known as the Kelly-Lochbaum vocal-tract model to generate a voice and was published the year thereafter [5].

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 [6, 7, 8] and applied to the wave equation to simulate string sounds. The sound generation, however, was far from real-time and it took several minutes to generate only 1 second of sound. In 1983, Cadoz et al. introduced CORDIS, a real-time sound generating system based on *mass-spring networks* [9]. The first physical model of the bowed string was due to McIntyre et al. in their 1983 publication [10]. In the same year Karplus and Strong devised an extremely efficient way to generate a string sound in [11] later known as the Karplus-Strong algorithm. Based on these ideas, Smith coined the term *digital waveguides* around the late 1980s and early 1990s in [12, 13] and continued to develop the method [14]. Around the same time, Adrien in [15] and later Morrison and Adrien in [16] introduced *modal synthesis*, a way to generate an object's sound by decomposing it into its modes of vibration.

Although more techniques have been developed in the last 20-30 years, most of the developments in the field of physical modelling for musical in-

---

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

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

struments 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 [17] and and later developed by De Poli and Rocchesso in [18] and is used to structure this work. Examples or resonator-exciter combinations are the violin and the bow, or the trumpet and the lips of the player.

A resonator is a passive system, in this work mostly assumed to be linear, and does not emit sound unless triggered by an external source. Exciters can be seen as these external sources, and generally have a nonlinear element.[3] Exciters insert energy into a resonator and cause it to vibrate and emit sound and the method of excitation will greatly influence the sound of the resonator. In the real world, the interaction between the exciter and the resonator is bi-directional, hence called an interaction. In other words, the exciter not only affects the state of the resonator, but the resonator affects the exciter as well. For the most part, this is also what is attempted to model in this work.

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

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) [19, 20]. Examples of a dynamic systems are a guitar string, a drum-membrane, or air propagation in a concert hall; three very different concepts, but all based on the same types of equations of motion. Many of these equations and other knowledge currently available on the physics of musical instruments have been collected by Fletcher and Rossing in [19]. 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

---

[3]The difference between linear and non-linear 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 they can still be considered linear and their nonlinear effects can be ignored.

done on implementing these PDEs to model and simulate different musical instruments. Great overviews of implementation techniques are given by, for example, Vesa Välimäki et al. in [21] and Julius O. Smith in [4, 14].

The most popular physical modelling techniques, also mentioned in Section 1.1 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 [16], 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 [22, 4]. When used to describe nonlinear systems, however, the modes become 'coupled' and the system will quickly become more computationally expensive. Recent developments using the FAUST programming language allow a 3D-mesh model of any three-dimensional object to directly be decomposed into its modes of vibration and used as a sound-generating physical model [23].

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 [6, 7, 8] and later by Chaigne in [24, 25]. Bilbao extensively describes this method in [20, 22]. Although computationally expensive, especially when working with higher-dimensional systems, this technique could potentially accurately model any system, whether it is linear or nonlinear, time-invariant or time-variant.

Digital Waveguide Modelling (or Digital Waveguides (DWGs)) is a technique that discretises wave propagation and scattering. The technique was first presented by Smith in [12, 13], and is mostly used for one-dimensional systems, such as strings and acoustic tubes and decomposes their system into travelling wave components. This technique has also been used in higher-dimensional systems, but is superior in efficiency when used in the one-dimensional case [21]. Some authors have combined DWGs with FDTD methods (such as in [26, 27]) to accurately model nonlinear behaviour while maintaining high-speed implementation.

Mass-spring networks can be similar in nature to FDTD methods, but treat each grid point as an individual mass connected to other masses through springs in a network. Pioneered in a musical context by Cadoz in [28, 9, 29] it is currently being further developed by Leonard and Villeneuve in a real-time, interactive

environment [30, 31].

Discussion

This work focuses on physical modelling using FDTD methods. The main advantage of these methods is that they are extremely general and flexible in terms of the types and amount of systems they can model. They allow any set of PDEs to be directly numerically simulated without making any assumptions regarding travelling wave solutions or modes. Moreover, FDTD methods allow for various PDEs, fx. 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 to model a PDE, it requires the system to have closed-form or analytical solution. If this is not available, (finite-element) analysis of the system could be performed to obtain the modal shapes and frequencies of the system. This in itself is very computationally expensive and requires a lot of storage if the modal data needs to be saved.

check all of this

The main drawback of FDTD methods is the fact that they require great attention to numerical stability of the solution [20]. For a wrong choice of parameters, the implemented system could become unstable and "explode"[4]. Stability analysis as well as energy analysis techniques are invaluable in the process of ensuring a stable implementation and much attention to this will be given throughout this work.

A final drawback of using FDTD methods is that – especially for higher-dimensional systems – they are much more computationally heavy than other methods, such as DWGs or modal synthesis techniques. The bright side, if one believes in Moore's law [32], is that it can be assumed that computing power will continue to increase and that within several years, running high-quality simulations of musical instruments based on FDTD methods in real time, will not be an issue.

1.4 Applications of Physical Modelling

So why would we go through all this hassle of modelling musical instruments? Could we not use a recording of the original instrument and play that back at the right moment? Or taking another step back, why not buy a real instrument and learn to play that instead? This section aims to answer those questions, by providing some applications of physical modelling for musical instrument simulations. Very informal section, but I kinda like it :)

---

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

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 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.

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

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

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

1.4.2 Resurrect Old or Rare Instruments

Many instruments exist that are too old, too rare, or too valuable to be played. Some live behind museum glass only to be looked at by visitors, never to be played again. In these cases, it might even be hard to record samples of the musical instrument. If, however, the physics (geometry and material properties) of the instrument are available, a physical model of the instrument could be created and its sound brought back too life.

However, 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 [4]). This could potentially result in unique sounds that can only be created using physical models.

1.5 Project Objectives and Main Contributions

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

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

The biggest challenge in real-time audio applications, as opposed to those only involving graphics for example, is that the sample rate required is extremely high. As Nyquist's sampling theory states, a sampling rate of at least 40 kHz is necessary to produce frequencies up to the human hearing limit of 20 kHz [33]. Most graphics applications are made with a temporal sample rate (mostly referred to as frames per second (FPS)) of around 60 Hz [34], which is orders of magnitude smaller than the auditory sample rate.

Even though physical modelling has been a popular research field in the past few decades, relatively little research has been done on making the models work in real-time, i.e., 'playable' [35]. Several virtual string instruments and different electric pianos have been made real-time by Pfeifle and Bader in [36, 37, 38]. 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 [39] 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 [40, 41].

The main objective of this work is to implement physical models simulated using FDTD methods in real time without the need of special hardware, i.e., on a regular personal computer or laptop. The objective is not to renew the underlying models themselves, but novel combinations were be made to simulate relatively unknown instruments as test cases for this objective. The instruments modelled over the course of this project are the esraj (Bowed Sitar), hammered dulcimer 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 **??**.

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

As mentioned in Section 1.4.3, using physical modelling to simulate real-life instruments relieves the physical limitations that the real world imposes on them. As FDTD methods are quite rigid, dynamically changing parameters while the instrument simulation is running, is a challenge. Other techniques, such as modal synthesis, are much more suitable for this, but come with the drawbacks mentioned in Section 1.3. Therefore, one of the main objectives of this project was to devise a method to allow parameters in musical instrument simulations based on FDTD methods to be dynamically varied.

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 **??**.

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

A great challenge in musical instrument simulations is their control. In many physical instruments, one interacts immediately with the sound-creating object, such as a string on a guitar or a membrane on a drum. This allows the musician to be much more expressive than if they only used the keyboard and mouse. Expressivity, however, is not the only thing that makes an instrument

interesting and enjoyable to play. The interaction with a musical instrument simulations could feel very 'dry' or unnatural as there is no haptic feedback; something which is the case in (nearly) all physical musical instruments.

The last objective of this PhD project is thus to find ways to control the instrument simulations in a natural way. Over the course of this PhD project, the Sensel Morph, or Sensel for short, has been used extensively [42]. 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, described in paper [E], a virtual-reality implementation of the tromba marina is controlled by the PHANTOM Omni [43] which is a six-degrees-of-freedom haptic device. The controller contains a hand-held pen-like object that is attached to a robotic arm and can be linked to a virtual environment to provide haptic feedback through the arm based on this environment.

1.6 Thesis Outline

Although a collection of papers format has been used for the structure of this thesis, the introductory part to the papers has been written in a monographic style. This hybrid format

This thesis is divided into several parts which in their turn are divided in chapters. See Figure 1.1 for a visual overview of the thesis structure.

Part I: Introduction

This part introduced the field of physical modelling for musical instruments here in Chapter 1 by giving a brief history of the field and providing and background for the project. Furthermore, the project objectives and contributions to the field have been detailed. Chapter 2 will provide a thorough introduction to finite-difference time-domain methods using simple sound-generating systems as examples, after which Chapter 3 will introduce several analysis techniques in a tutorial-like fashion. This part has – as much as possible – been written in layman's terms and is perhaps more pedagogical than could be expected of a PhD thesis.

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 will present a model for the stiff string, Chapter 5 will introduce physical model for brass instruments and Chapter 6 will introduce two-dimensional systems such as membranes and plates which can be used to simulate simplified instrument bodies.

Part ??: Exciters

As stated in Section 1.2 stated, the excitation greatly determines the behaviour of the resonator. This pert presents various ways in which the resonators introduced in Part II can be excited. Chapter **??** introduces 'physically inspired' excitations, Chapter **??** introduces the bow and presents the contribution made in paper [C], and finally, Chapter **??** presents the lip reed used to excite brass instruments.

Part ??: Interactions

As mentioned before, most musical instruments consist of many individual resonators, and to properly model these, their interaction must be taken into account. This part two different ways that the resonators can interact with each other; collisions between various resonators are presented in Chapter **??** and connections between them in Chapter **??**.

The above parts are used as an introduction for the main contributions of the PhD project and – with the exception of Chapter **??** – 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 expected from a PhD thesis (specifically Part I). It is the hope of the PhD student that going this extra mile could make this work (and the above parts in particular) be a contribution in itself: to put this research field into reach of beginners in the field of physical modelling for sound synthesis using FDTD methods without the need of much experience in the fields of physics, mathematics or computer science.

Part ??: Contributions

This part contains extended summaries of the main contributions of the PhD project. Chapter **??** summarises paper [G] and extends it by providing some design considerations. Chapter **??** explains the considerations necessary for real-time implementation of physical models. Chapters **??**, **??** and **??**, summarise papers [A], [D] and [H] respectively and extend the papers mainly with more implementation details.

Part ??: Conclusions and Perspectives

This part 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 III: Papers** contains the main publications made over the course of this PhD project and an appendix appears in **Part IV: Appendix**.

Fig. 1.1: The outline of this thesis. The contributions made throughout the PhD project are marked in yellow. Most are collected in Part **??**, though the novel work done on the bow will already appear in Chapter **??**. Marked in green are the parts that describe the physical models which the contributions are based on. 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 III 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. Unless denoted otherwise, the theory presented in this chapter and the notation have been taken from [20].

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 [44] and potentially higher-dimensional systems. See Section 2.1.1 for more information on dimensions.

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

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

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

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

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 a dot above the function to denote a time-derivative. For this reason, Newton's notation will be used for ODEs in isolation. The drawback of this notation is that it only be used for univariate functions. Finally, Euler's notation indicates a derivative using an operator which can be applied to a function.

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

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

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

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

where the subscript notation could be seen as the partial derivative counterpart to Newton's notation due to its compactness. In the remainder of this document, Euler's notation will be used for PDEs, due to their similarity to operators in discrete time (introduced in Section 2.2.2) and as it allows for creation of bigger operators for more compactness when working with multiple (connected) systems (see e.g. Chapter **??**). Also, state-of-the-art literature in the field of FDTD methods for sound synthesis use this notation [22].

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) is ignored. In other words, every point along the string can only move up and down, not side-to-side and not forward and back. Although this greatly simplifies the system at hand and reduces computational complexity, this is not what happens in reality. Nonlinear effects such as pitch glides due to tension modulation caused by high-amplitude string vibration are not present in the simplified model and are not presented in this work.

Work has been done on strings with dual (transverse) polarisation by Desvages [45] and Desvages and Bilbao [46] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are couples can be found in [20, 47]. In [30], 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.

check whether this needs to be per point along a system

---

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

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

Notation

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

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

2.1.2 Ranges of Definition and Domains

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

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

2.2 Discretisation using FDTD methods

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

concern with these methods is the numerical stability of the eventual approx-imation. 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. Unless denoted otherwise, the equations and theory used in this chapter has been taken from [20].
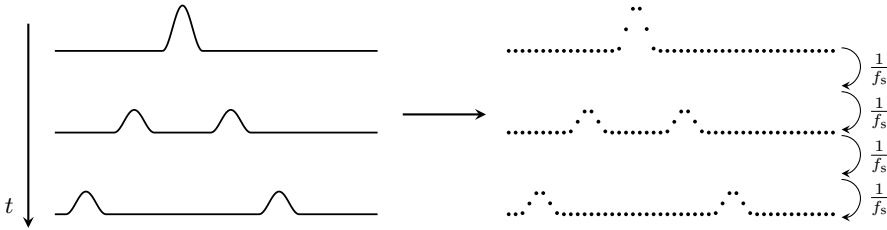
Fig. 2.1: A continuous PDE is discretised to a FD scheme. In a PDE, time passes continuously, whereas for a FD scheme, time passes in finite increments with a duration of the reciprocal of the sample rate f_s.

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 will be used in this work (unless denoted otherwise).

As mentioned in Section 2.1.2, a 1D system needs to be defined over a temporal range of definition and one spatial domain. In discrete time, $t \geq 0$ is discretised to $n \in \mathbb{N}^0$.[2] The spatial domain $\mathcal{D}$ can be subdivided into N equal sections, or intervals, of length h (see Figure 2.2). The grid points describing

---

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

the state of the system are placed at the edge of each interval, including the end points. The spatial range of interest then becomes $l \in \{0, \ldots, N\}$ and the total number of grid points is $N + 1$, which is one more than the number of intervals.

To summarise, for a 1D system

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

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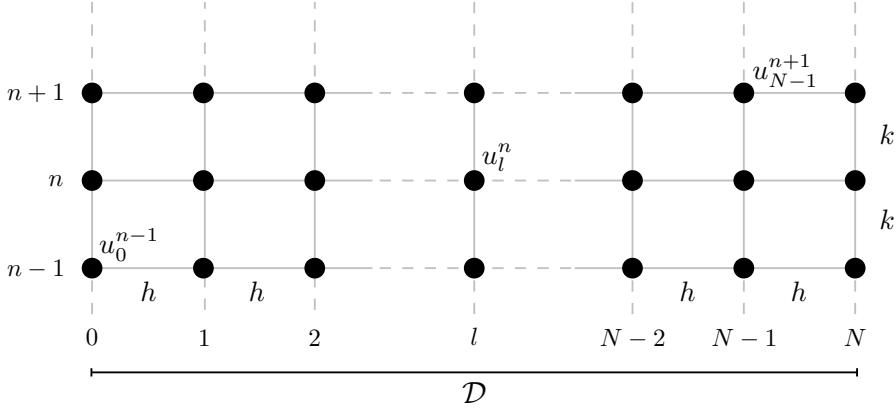
Fig. 2.2: The spatio-temporal grid that appears when a 1D system $u(x, t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n. The spatial domain $\mathcal{D}$ is divided into N intervals of length h and spatial range of interest $l = \{0, \ldots, N\}$. Time is subdivided into time steps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

2.2.2 Finite-Difference Operators

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

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

Similarly, forward and backward shifts in space are

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

many figures for shift and FD operators

These shift operators are rarely used in isolation, though they do appear in

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

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

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

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

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

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

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

and when applied to u_l^n are

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

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

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

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

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

---

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

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.

(a) Temporal operators.

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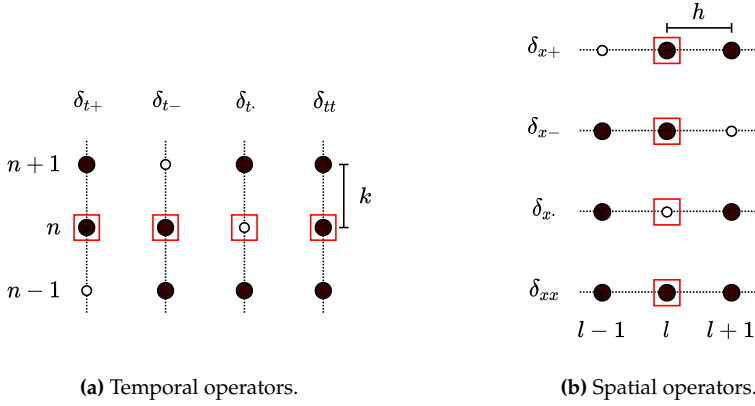
Fig. 2.3: The stencils of various FD operators applied to the grid point highlighted with a red square. Black grid points are used in the calculation, and white grid points are not. The averaging operators follow the same pattern.

<div style="border: 2px solid orange;">in energy analysis, interleaved grids, etc.</div>

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

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

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

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

Similarly, spatial averaging operators are

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

and when applied to u_l^n

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

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

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

and

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

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

Accuracy

As FDTD methods approximate continuous systems, the resulting solution is rarely 100% accurate. To determine the accuracy of the FD operators above, one can perform a *Taylor series analysis*. The Taylor series is an infinite sum and its expansion of a function f about a point a is defined as

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

where superscript (n) denotes the nth 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 [20], one may apply FD operators to continuous functions according to

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

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

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

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

the power of k the more accurate the approximation. Equation (2.17) can be rewritten to

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

and using Eq. (2.16) can be written to

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

As a first-order derivative indicates the *slope* of a function, the differences in accuracy between the above operators can be visualised as in Figure 2.4. It
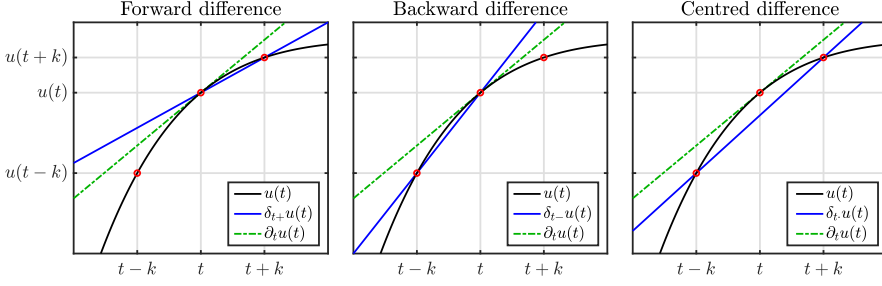
Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. One can observe that the centred difference operator much more closely approximates the derivative, or the slope, of u at t than the forward and backward difference operators.

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

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

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

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

$$u(t+k) + u(t-k) = 2u(t) + k^2 \ddot{u} + \mathcal{O}(k^4),$$

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

$$\delta_{tt} u(t) = \ddot{u} + \mathcal{O}(k^2). \tag{2.25}$$

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

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

2.2.3 Identities

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

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

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

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

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

2.3 The Mass-Spring System

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

2.3.1 Continuous-time

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

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

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

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

with angular frequency (in rad/s)

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

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*.

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

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

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

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

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

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

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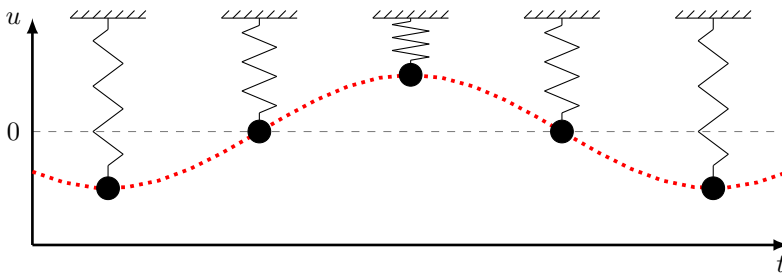

Fig. 2.5: Mass spring system over time. The system follows a harmonic (sinusoidal) motion.

this means that it remains stationary. If it is, at the exact moment that $u = 0$, the velocity is unchanged.

2.3.2   Discrete-time

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

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

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

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

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

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

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

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

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

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

2.3.3   Implementation and Output

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

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

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

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

$$\frac{1}{k}\left(u^1 - u^0\right) = 0,$$

$$\xleftrightarrow{u^0=1} \quad u^1 - 1 = 0,$$

$$u^1 = 1.$$

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

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

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

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

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

2.4 The 1D Wave Equation

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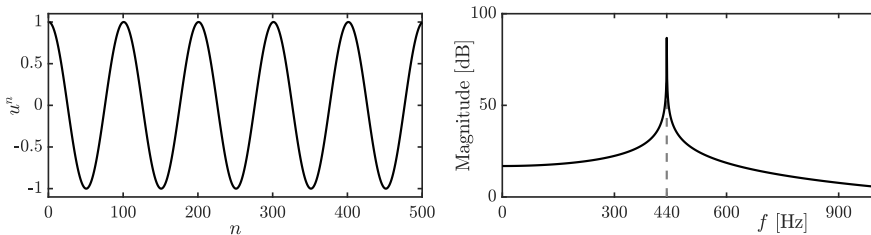

Fig. 2.6: The time-domain and frequency-domain output of a mass-spring system with $f_0 = 440$ Hz.

FULL DOC SWEEP: check hyphen in titles

unit?

2.4.1 Continuous time

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

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

where c is the wave speed of the system (in m/s). If the PDE is used to model an ideal string, the wave speed can be defined as $c = \sqrt{T/\rho A}$, with tension T (in N), material density ρ (in kg/m^3) and cross-sectional area A. If instead, it is used to model pressure in an acoustic tube c is the speed of sound in air. Figure 2.7 shows the wave propagation of the 1D wave equation excited using a raised cosine.

(a) $t = 0$ ms. **(b)** $t = 1$ ms. **(c)** $t = 2$ ms.
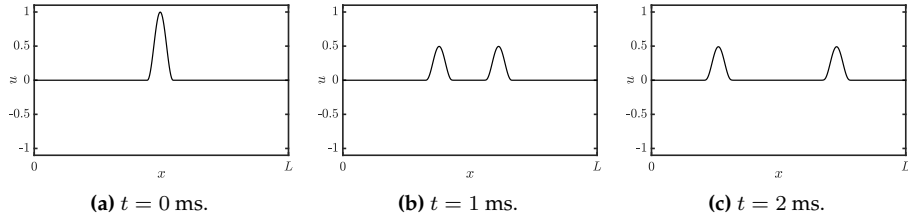
Fig. 2.7: Wave propagation in the 1D wave equation in Eq. (2.38) wtih $c \approx 127$ m/s.

Intuition

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

The 1D wave equation in (2.38) states that the acceleration of $u(x, t)$ at location x is determined by the second-order spatial derivative of u at that same location (scaled by a constant c^2). In the case that u describes the transverse displacement of an ideal string, this second-order derivative denotes the *curvature* of this string. As c^2 is always positive, the sign (or direction) of the acceleration is fully determined by the sign of the curvature. In other words, a 'positive' curvature at location x along the ideal string yields a 'positive' or upwards acceleration at that same location.

What a 'positive' or 'negative' curvature implies is more easily seen when 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 $\rho A dx$ is the mass of a (tiny) portion of the string of length dx (in m), $\partial_t^2 u$ is the acceleration of that portion and $T \partial_x^2 u dx$ describes the force acting on that portion, yielding Newton's second law.
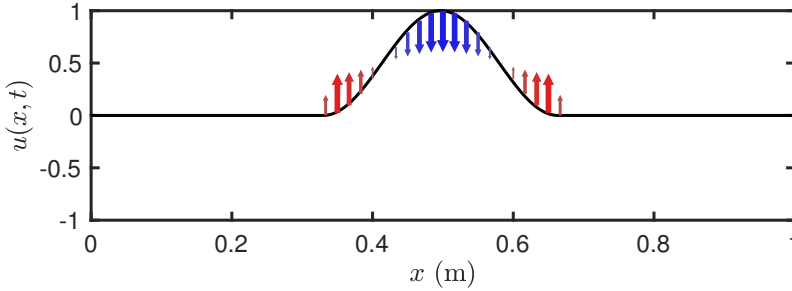
Fig. 2.8: The forces acting on the 1D wave equation described by $u(x, t)$ due to curvature. The arrows indicate the direction and magnitude of the force, and simultaneously the acceleration as these are connected through Eq. (2.38).

different wording in caption

Boundary Conditions

When a system is distributed in space, *boundary conditions* must be determined. Recalling that x is defined over domain $\mathcal{D} = [0, L]$, the boundaries, or end points of the system are located at $x = 0$ and $x = L$. Two often-used alternatives

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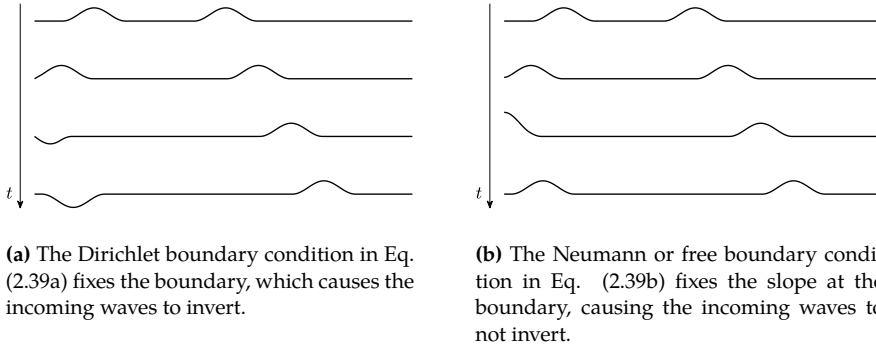
Fig. 2.9: The behaviour of the 1D wave equation with (a) Dirichlet or (b) Neumann boundary conditions.

for the boundary conditions are

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

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

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

add why this is relevant?

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

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

Scaling

As this work follows much of Bilbao's *Numerical Sound Synthesis* [20], 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 [20] and much of the literature published around that time (fx. [48, 47]) and can be useful to reduce the amount of parameters used to describe a system.

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

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

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

Although this parameter reduction might be useful for resonators in isolation, when multiple resonators interact with each other (see Part **??**), 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. The most straightforward discretisation of Eq. (2.38) is the following FD scheme

> FULL DOC SWEEP: check straightforward or straight-forward

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

with $l \in \{0, \dots, N\}$ and number of grid points $N+1$. Other schemes exist (see e.g. [20]), but are excluded as they have not been used in this work. Expanding the operators using the definitions given in Section 2.2.2 yields

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

and solving for u_l^{n+1} yields

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

Here,

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

is called the *Courant number* and plays a big role in stability and quality of the FD scheme. More specifically, λ needs to abide the (famous) Courant-Friedrichs-Lewy or *CFL condition* for short [49]

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

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

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

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

33

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

Stencil

As was done for several FD operators in Figure 2.3, it can be useful to visualise the *stencil*, or region of operation, of a FD scheme. A stencil of a scheme visualises what grid values are necessary to calculate the state at the next time step u_l^{n+1}. Figure 2.10 shows the stencil for scheme (2.42) and – in essence – visualises the various shifts of the grid function in Eq. (2.44). One could visualise this stencil to be placed on the left-most point sof the grid shown in Figure 2.2. The update equation then iterates this stencil over the entire domain and calculates all values of u_l^{n+1} based on known values of u_l^n and u_l^{n-1}.
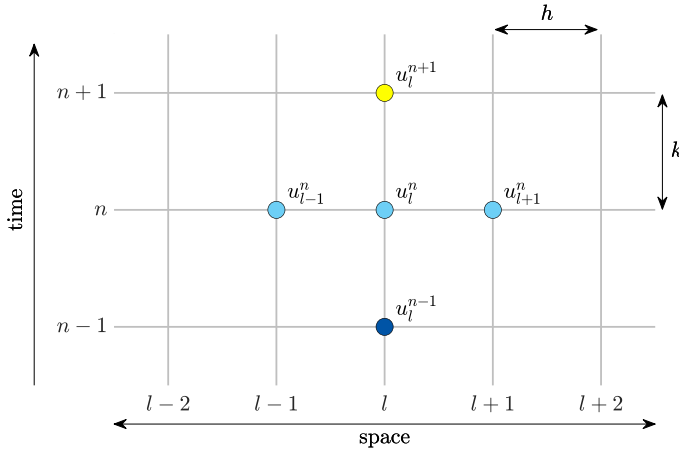
Fig. 2.10: The stencil, or region of operation, for the FD scheme in (2.42). The time steps of the various grid points are colour-coded by yellow ($n+1$), light blue (n) and dark blue ($n-1$).

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

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

If Dirichlet boundary conditions are used, the states of the boundary points will always be zero and can therefore can be excluded from the calculations. The range of calculation then simply becomes $l \in \{1, \ldots, N-1\}$ and no virtual grid points are needed when performing the update.

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

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

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

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

and

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

at the left and right boundary respectively.

2.4.3 Implementation: Excitation and Output

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

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

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

$$l_s = l_c - \lfloor w/2 \rfloor \quad \text{and} \quad l_e = l_c + \lfloor w/2 \rfloor , \tag{2.51}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation and needs to be used as all the above variables are integers. Furthermore, both l_s and l_e must fall into the defined

spatial range of calculation. Then, a raised cosine with an amplitude of 1 can be calculated and used as an initial condition for the system according to

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

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

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

Output and Modes

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

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

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

As can be seen from Figure 2.11, the output of the 1D wave equation contains many peaks in the frequency spectrum on top of the fundamental frequency. These are called *harmonic partials* or *harmonics* for short and arise from the various modes of vibration present in the system (see Figure 2.12). Although the PDE has not been 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.[4]
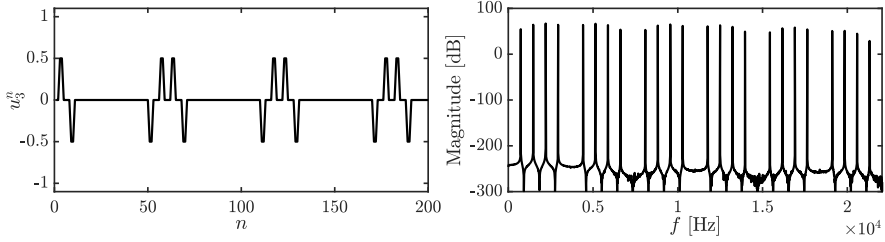

Fig. 2.11: The time-domain and frequency-domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_\mathrm{s} = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_\mathrm{c} = 0.2N$) and $w = 4$ and the output is retrieved at $l_\mathrm{out} = 3$.
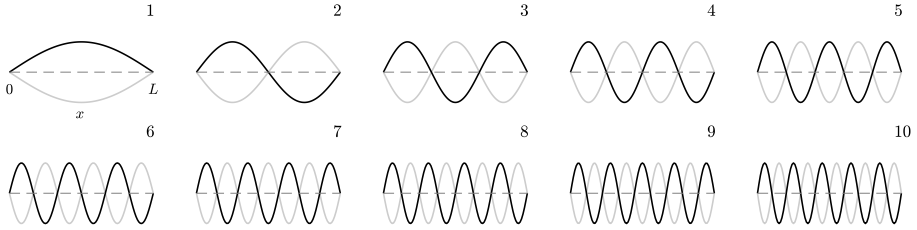
Fig. 2.12: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions defined for $x \in [0, L]$ (only shown for mode 1). The modes are normalised to have the same amplitude and vibrate at their respective modal frequencies with the extremes indicated by the black and the grey plot. The number of the shape can be determined by the amount of antinodes present in the shape.

The amount of modes present in the continuous PDE of the 1D wave equation is theoretically infinite. The amount present in the discrete FD scheme, however, is determined by the number of moving points in the system. If Dirichlet boundary conditions are used, this means that there are $N-1$ modes, and $N + 1$ modes for Neumann boundary conditions. If the CFL condition is satisfied with equality, the frequencies of these modes are integer multiples of the fundamental: $f_m = m f_0$ for mode number $m \in \{1, \ldots, N-1\}$ for Dirichlet and $m \in \{0, \ldots, N\}$ for Neumann boundary conditions. The frequency of the

---

[4]Modes of the vibrating string were first discovered by Sauveur in 1701 who said that "*especially at night*" he observed "*other small sounds*" on top of the fundamental frequency and coined the terms 'node' and 'harmonic' [50].

harmonics – and even the modal shapes – can be analytically derived using modal analysis as will be explained in Section 3.5.

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

2.4.4 Stability and Simulation Quality

As shown in Eq. (2.46), the Courant number needs to abide the CFL condition in order for the scheme to be stable. A system is regarded *unstable* if it exhibits (exponential) unbounded growth. If Neumann boundary conditions (free) are used, it is possible that the system drifts off over time. This does not mean that the system is unstable, and is actually entirely physically possible![5]

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 fx. Chapter 4). Numerical dispersion, however, which is due to numerical inaccuracy, never is! Figure 2.13b shows an example when $\lambda = 0.9$, and one can observe that the wave propagation does not match the ideal case as Figure 2.13a shows. Moreover, bandlimiting effects occur, meaning that the highest frequency that the system can generate decreases. See Figure 2.14. Higher modes get 'squished' together and are not exact multiples of the fundamental anymore. Section 3.5 elaborates on how to calculate the exact modal frequencies of a FD implementation of the 1D wave equation.

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

---

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

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

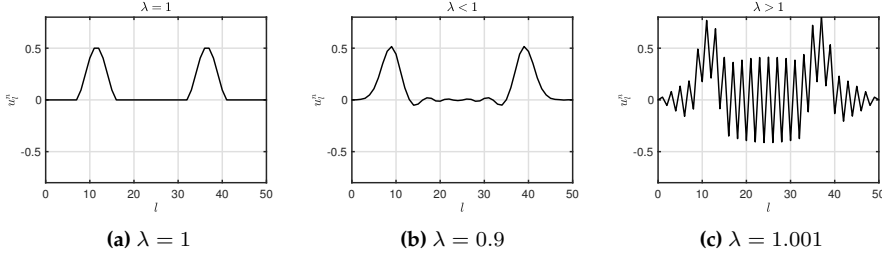**Fig. 2.13:** Grid function u_l^n visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (2.46) is not satisfied and the system is unstable.

(a) $\lambda = 1$ **(b)** $\lambda = 0.9$ **(c)** $\lambda = 1.001$
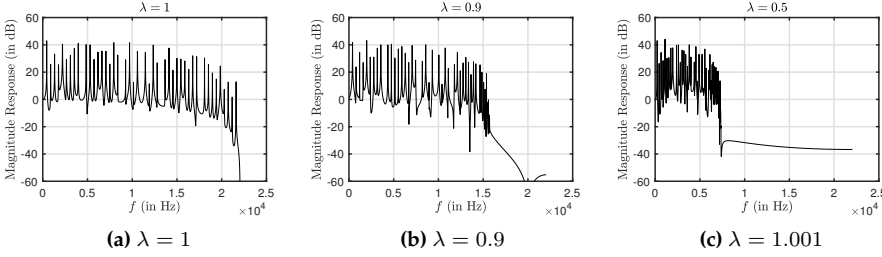
Fig. 2.14: Frequency spectra of the simulation output. The Courant number is set to (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$. One can observe that for lower values of λ the bandwidth of the output decreases drastically.

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

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

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

To understand why h needs to be recalculated, consider the following example. Consider the 1D wave equation defined over domain $\mathcal{D} = [0, L]$ where

$L = 1$. Furthermore, we say that the system should produce a fundamental frequency of $f_0 = 750$ Hz which requires a wave speed of $c = 1500$ 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 condition (2.46), and the decrease in quality will be perceptually irrelevant – or at the very least, less perceptually relevant than the change in f_0 if h is not recalculated.

Intuition

It might not be immediately clear why a too low value for h might cause instability. Some intuition is provided in [20, 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 that they can not lose to their neighbours, because

their effect should have reached their second neighbour over the course of one sample. The way to solve this would be to halve the time step k (or double the sample rate f_s), which would allow grid points to interact with their second neighbours over the course of once the the old time step (as this is now divided into two time steps). This also shows in the fact that $\lambda = 1$ again (as halving k cancels out halving h) and grid points transfer their energy to their neighbours proportionately again.

figure?

Possible solution

One of the main contributions of the PhD project is published in paper [G] and summarised in Chapter **??**, 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 used to debug implementations of FD schemes, as well as determine stability conditions in a more general fashion, and

- *Modal analysis* which can be used to determine the modal frequencies (and damping per mode) that a FD scheme exhibits.

3.1 Matrices in a FDTD context

For several purposes, such as implementation in `MATLAB` and several analysis techniques described shortly, it is useful to write a FD scheme in *matrix form*.[1] Matrix multiplication when working with FDTD methods usually involves multiplying a square matrix (with equal rows and columns) onto a column vector. Consider a $(N+1) \times (N+1)$ square matrix $\mathbf{A}$ and a $(N+1) \times 1$ column vector $\mathbf{u}$. Multiplying these results in a $(N+1) \times 1$ column vector $\mathbf{w}$:

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

---

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

43

Expanding this operation results in

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

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

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

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

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

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

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

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

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:

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

and has the following special property

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

3.1.2 Schemes and Update Equations in Matrix Form

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

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

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

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

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

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

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

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

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

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

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

The identity matrix is necessary here for correct matrix addition.

3.2 Mathematical Tools and Product Identities

Some useful mathematical tools used for the energy analysis techniques presented in Section 3.4 will be shown here. The tools shown here can be applied to 1D systems. These will be extended to 2D systems in Chapter 6. Unless denoted otherwise, the notation and theory will follow [20].

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

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

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

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

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

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

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

and the more general weighted inner product

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

which scale the boundary points of the regular inner product. Naturally, if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 1$, Eq. (3.10) reduces to Eq. (3.9), and if $\epsilon_\mathrm{l} = \epsilon_\mathrm{r} = 2$, (3.10) reduces to (3.8).

3.2.2 Summation by Parts

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

Here, the same functions as in the previous section, $f(x, t)$ and $g(x, t)$ and domain $\mathcal{D}$, will be used. Applying a spatial derivative to g, and using Eq. (3.7), integration by parts is defined as

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

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

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

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

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

A derivation of Eq. (3.12a) is given 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 the sign of the resulting inner product changes. Important to note, is that the sign (forward / backward) of the derivative operator has also changed. Lastly, discrete boundary terms have appeared and it can be seen that values outside of the defined domain are needed, i.e., g_{N+1}^n and f_{-1}^n. These can be accounted for by the boundary conditions imposed on the system (see Section 2.4.2 as an example).

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

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

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

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

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

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

all of which 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

$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \tag{3.15a}$$

$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_{\underline{\overline{d}}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0, \tag{3.15b}$$

$$\langle f, \delta_{xx} g \rangle_d' = \langle \delta_{xx} f, g \rangle_d' + f_N \delta_{x\cdot} g_N - g_N \delta_{x\cdot} f_N - f_0 \delta_{x\cdot} g_0 + g_0 \delta_{x\cdot} f_0 \tag{3.15c}$$

Derivations

To see why the above identities hold true, it is useful to briefly go through a derivation. As an example, we go through Eqs. (3.12a) and (3.13a) as they have the same inner product as a starting point, but yield different results. In the following, $d = \{0, \ldots, N\}$ and $N = 2$ are used.

Starting with Eq. (3.12a), suppressing the n superscript for brevity, and using the definition for the discrete inner product in Eq. (3.8), we get

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

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

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

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

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

$$(\delta_t . u_l^n)(\delta_{tt} u_l^n) = \delta_{t+} \left(\frac{1}{2} (\delta_{t-} u_l^n)^2 \right),$$ (3.16a)

$$(\delta_t . u_l^n) u_l^n = \delta_{t+} \left(\frac{1}{2} u_l^n e_{t-} u_l^n \right),$$ (3.16b)

$$(\delta_{t+} u_l^n)(\mu_{t+} u_l^n) = \delta_{t+} \left(\frac{1}{2} (u_l^n)^2 \right),$$ (3.16c)

$$(\delta_t . u_l^n)(\mu_t . u_l^n) = \delta_t . \left(\frac{1}{2} (u_l^n)^2 \right),$$ (3.16d)

$$u_l^n e_{t-} u_l^n = (\mu_{t-} u_l^n)^2 - \frac{k^2}{4} (\delta_{t-} u_l^n)^2$$ (3.16e)

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

$$\delta_{t+} (u_l^n w_l^n) = (\delta_{t+} u_l^n)(\mu_{t+} w_l^n) + (\mu_{t+} u_l^n)(\delta_{t+} w_l^n).$$ (3.17)

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 Eq. (3.17) 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 [20], 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 [51, 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 [52] co-authored by John von Neumann. Later, this technique got a more general treatment in [53] and is heavily used in [20]. The discrete-time z-transform and discrete spatial Fourier transform performed on a 1D grid function are defined as [20]

$$\hat{u} = \sum_{n=-\infty}^{\infty} u_l^n z^{-n} \quad \text{and} \quad \tilde{u} = \sum_{l=-\infty}^{\infty} u_l^n e^{-jl\beta h} \tag{3.18}$$

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) [53]

$$u_l^n \overset{\mathcal{A}}{\Longrightarrow} z^n e^{jl\beta h} \tag{3.19}$$

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

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

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 [54, p. 71]:

$$\sin(x) = \frac{e^{jx} - e^{-jx}}{2j} \quad \Rightarrow \quad \sin^2(x) = \frac{e^{j2x} + e^{-j2x}}{-4} + \frac{1}{2}, \tag{3.20a}$$

$$\cos(x) = \frac{e^{jx} + e^{-jx}}{2} \quad \Rightarrow \quad \cos^2(x) = \frac{e^{j2x} + e^{-j2x}}{4} + \frac{1}{2}. \tag{3.20b}$$

Take for example

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

Then, using $x = \beta h/2$, identity (3.20a) can be rewritten to

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

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 (3.19) with frequently appearing temporal and spatial shifts.

and substituted into the above to get

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

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

$$\delta_{t+} u_l^n \overset{\mathcal{A}}{\implies} \frac{1}{k}(z-1), \qquad \delta_{t-} u_l^n \overset{\mathcal{A}}{\implies} \frac{1}{k}\left(1 - z^{-1}\right),$$
$$\delta_{t\cdot} u_l^n \overset{\mathcal{A}}{\implies} \frac{1}{2k}\left(z - z^{-1}\right), \qquad \delta_{tt} u_l^n \overset{\mathcal{A}}{\implies} \frac{1}{k^2}\left(z - 2 + z^{-1}\right) \tag{3.21}$$

and for spatial operators identity (3.20a) can be used to obtain

$$\delta_{xx} u_l^n \overset{\mathcal{A}}{\implies} -\frac{4}{h^2} \sin^2(\beta h/2), \tag{3.22a}$$

$$\delta_{xxxx} u_l^n \overset{\mathcal{A}}{\implies} \frac{16}{h^4} \sin^4(\beta h/2). \tag{3.22b}$$

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

Proving stability

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

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

not talking about nonlinear systems though

only the denominator of the transfer function

$$|z| \leq 1. \tag{3.23}$$

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

$$z^2 + a^{(1)}z + a^{(2)} \tag{3.24}$$

its roots satisfy condition (3.23) when it abides the following condition [20]

$$|a^{(1)}| - 1 \leq a^{(2)} \leq 1. \tag{3.25}$$

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

$$|a^{(1)}| \leq 2, \tag{3.26}$$

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 (3.19) with $l = 0$. Using Table 3.1 and Eqs. (3.21) as a reference and substituting the definitions yields

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

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

$$z - \left(2 - \frac{Kk^2}{M}\right) + z^{-1} = 0. \tag{3.27}$$

To begin to prove stability, this equation needs to be written in the form found in (3.24). Multiplying all the terms by z, and noticing that $a^{(2)} = 1$, we could continue with condition (3.26). However, the scheme used here is a special case where the roots of the characteristic equation can not be identical [20]. 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 (3.26) becomes $|a^{(1)}| < 2$. Continuing with this conditions yields

$$\left|-2 + \frac{Kk^2}{M}\right| < 2,$$

$$-2 < -2 + \frac{Kk^2}{M} < 2,$$

$$0 < \frac{Kk^2}{M} < 4.$$

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

$$k < 2\sqrt{\frac{M}{K}}.\tag{3.28}$$

Recalling that $\omega_0 = \sqrt{K/M}$, Eq (3.28) can be more compactly written as

$$k < \frac{2}{\omega_0}.\tag{3.29}$$

3.3.2 1D Wave Equation

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

Recalling the FD scheme in (2.42):

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

its frequency-domain representation can be obtained using the definitions in Eqs. (3.21) and (3.22a):

$$\frac{1}{k^2}\left(z - 2 + z^{-1}\right) = -\frac{4c^2}{h^2}\sin^2\left(\beta h/2\right).\tag{3.30}$$

Also recalling that

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

the characteristic equation of the 1D wave equation is

$$z + \left(4\lambda^2\sin^2(\beta h/2) - 2\right) + z^{-1} = 0.\tag{3.31}$$

The scheme is then stable if the roots satisfy condition (3.23). As the characteristic equation is of the form in (3.24) (after multiplication with z) with $a^{(2)} = 1$, stability is shown by abiding condition (3.26) for all β and when applied to the characteristic equation (3.31), it can be seen that

$$|4\lambda^2\sin^2(\beta h/2) - 2| \leq 2,$$
$$|2\lambda^2\sin^2(\beta h/2) - 1| \leq 1,$$
$$-1 \leq 2\lambda^2\sin^2(\beta h/2) - 1 \leq 1,$$
$$0 \leq 2\lambda^2\sin^2(\beta h/2) \leq 2,$$
$$0 \leq \lambda^2\sin^2(\beta h/2) \leq 1.$$

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

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

$$\lambda \leq 1.$$

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

$$h \geq ck, \tag{3.32}$$

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

3.4 Energy Analysis

Of all analysis techniques described in this chapter, energy analysis is without a doubt the most important when working with FDTD methods. First of all, from a practical point of view, it is essential for debugging implementations of FD schemes. Especially when trying to model more complex systems, programming errors are unavoidable, and energy analysis can be extremely helpful in pinpointing where the error lies. Secondly, energy analysis techniques can be used to obtain stability conditions in a much more general sense than the frequency-domain analysis techniques presented in Section 3.3. Where frequency-domain analysis is restricted to LTI systems with infinite domains (for distributed systems), energy analysis can be applied to nonlinear systems and boundary conditions [20].

Gustafsson et al. in (the first edition of) [55] worked with energy to find stability conditions for FD schemes. This, they referred to as 'the energy method' and it effectively circumvented the need of a frequency domain representation to find stability conditions (as presented in Section 3.3). Later, energy, or more specifically 'energy as a conserved quantity', was used to determine stability and passivity of systems. Bilbao gives an extensive overview in [20] where this has been extensively used to show stability of the FD schemes used.

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

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

3.4.1 Energy Analysis: A 4-Step Tutorial

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

The first step to energy analysis is to take the appropriate *norm* of the scheme (see Eq. (3.7)), which yields and expression for the rate of change of the energy of the system: $\delta_{t+}\mathfrak{h}$. Usually, this means to take the inner product of the scheme with $(\delta_t . u_l^n)$. See Section 3.2.1 for more details on the inner product. Note that the forward time difference δ_{t+} is used (and not the backwards or centred) because of convention and preference [Bilbao, verbally].

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

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

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

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q} - \mathfrak{p}. \tag{3.33}$$

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

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.34}$$

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \tag{3.35}$$

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

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

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

An (1D) inner product (or norm) will 'add' one 'm' unit due to the h in its definition in (3.7). A first-order temporal difference operator will 'add' one 's^{-1}'-unit (because of the $1/k$) and a first-order spatial difference operator will 'add' one 'm^{-1}'-unit ($1/h$). Along these lines, a second-order time or difference operator will 'add' a 's^{-2}' ($1/k^2$) or 'm^{-2}'-unit ($1/h^2$) respectively. It is important to note that the time shift operator (e_{t-}) does not influence the units. Finally, the appearance of a grid function u_l^n 'adds' whatever it describes. Usually, as u_l^n describes a displacement in m, it will 'add' this to the equation. If it describes anything else, it will 'add' that.

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

In the end, the definition for the energy can be implemented and used as a check for whether the FD scheme has been implemented correctly. Usually, the energy of the system is calculated for every iteration in the for loop and plotted after the simulation. For a system without losses or energy inputs, the energy should be unchanged according to Eq. (3.34) and can be plotted according

$$\mathfrak{h}_e^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0} \, , \quad \text{if } \mathfrak{h}^0 \neq 0, \tag{3.36}$$

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 simulation, ultra slight fluctuations of the energy should be visible due to rounding errors. Plotting the Hamiltonian should show fluctuations within *machine precision*, which is usually in the range of 10^{-15}. Over time, the fluctuations can add up, and possibly end up out of this range, but generally, any fluctuations less than in the 10^{-10} range indicate that there is no programming error. See fx. Figures 3.1 and 3.2.

For a system with losses or energy inputs, a discrete integration, or summed form can be used (as done in fx. [56]):

$$\mathfrak{h}_e^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0 + k \sum_{m=0}^{n-1} (\mathfrak{q}^m + \mathfrak{p}^m)}{\mathfrak{h}^0} \, , \quad \text{if } \mathfrak{h}^0 \neq 0. \tag{3.37}$$

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

3.4.2 Mass-spring system

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

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

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

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

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

$$\delta_{t+}\mathfrak{h} = M(\delta_t.u^n)(\delta_{tt}u^n) + K(\delta_t.u^n)(u^n) = 0. \qquad (3.38)$$

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 (3.38), one can use identities (3.16a) and (3.16b) to get the following:

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n\right) = 0, \qquad (3.39)$$

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

$$\mathfrak{h} = \frac{M}{2}(\delta_{t-}u^n)^2 + \frac{K}{2}u^n e_{t-}u^n = 0. \qquad (3.40)$$

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{M}{2}(\delta_{t-}u^n)^2, \quad \text{and} \quad \mathfrak{v} = \frac{K}{2}u^n e_{t-}u^n. \qquad (3.41)$$

Step 3: Check units

As mentioned above, the energy $\mathfrak{h}$ needs to be in Joules, or $\text{kg}\cdot\text{m}^2\cdot\text{s}^{-2}$. Taking the terms in Eq. (3.41) one-by-one and writing them in their units results in

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

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

and indeed have the correct units.

Step 4: Implementation

Equation (3.46) can then be implemented in same for-loop recursion where the update is calculated.

```
1  %% Calculate the energy using Eq. (3.41)
2
3  % Kinetic energy
4  kinEnergy(n) = M / 2 * (1/k * (u - uPrev))^2;
5
6  % Potential energy
7  potEnergy(n) = K / 2 * u * uPrev;
8
9  % Total energy (Hamiltonian)
10 totEnergy(n) = kinEnergy(n) + potEnergy(n);
```

Figure 3.1 shows the normalised energy (according to Eq. (3.36)) of the mass-spring system and shows that the deviation is indeed within machine precision.
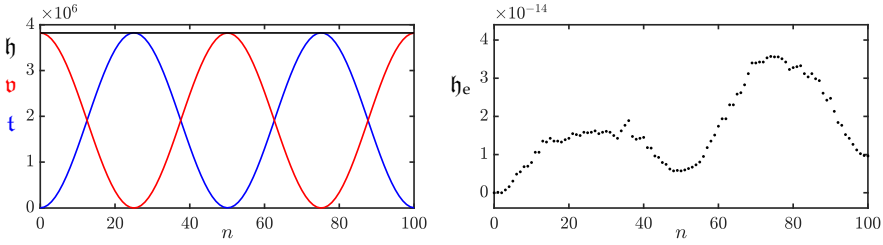

Fig. 3.1: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.36)). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within machine precision.

3.4.3 1D Wave Equation

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

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n, \qquad (3.42)$$

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

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

Taking an inner product using Eq. (3.42) 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:

$$\delta_{t+}\mathfrak{h} = \rho A \langle \delta_t.u_l^n, \delta_{tt}u_l^n \rangle_d - T \langle \delta_t.u_l^n, \delta_{xx}u_l^n \rangle_d = 0. \tag{3.43}$$

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

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

To isolate δ_{t+} in Eq. (3.43), the terms have to be rewritten in a way that fits the product identities in Section 3.2.3. Summation by parts as described in Section 3.2.2 can be used. Using identity (3.13a) 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

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

where the boundary term

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

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

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

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. (3.43) can be expanded to

$$\delta_{t+}\mathfrak{h} = \rho A \sum_{l=0}^{N} h(\delta_t.u_l^n)(\delta_{tt}u_l^n) + T \sum_{l=0}^{N} h(\delta_t.\delta_{x+}u_l^n)(\delta_{x+}u_l^n) \tag{3.44}$$

Then, using identities (3.16a) and (3.16b), δ_{t+} can be isolated

$$\delta_{t+}\mathfrak{h} = \delta_{t+}\left(\frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2 + \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n \rangle_{\underline{d}} \right), \tag{3.45}$$

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v},$$

$$\text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad \text{and} \quad \mathfrak{v} = \frac{T}{2}\langle \delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n \rangle_{\underline{d}}. \tag{3.46}$$

Step 3: Check units

Writing out the definitions for kinetic and potential energy in Eq. (3.46) respectively, yields

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

$$= \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2},$$

$$\mathfrak{v} = \frac{T}{2} \langle \delta_{x+} u_l^n, e_{t-} \delta_{x+} u_l^n \rangle_{\underline{d}} \xrightarrow{\text{in units}} \text{N} \cdot \text{m} \cdot (\text{m}^{-1} \cdot \text{m} \cdot \text{m}^{-1} \cdot \text{m}^{-1} \text{m})$$

$$= \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2},$$

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

```matlab
%% Calculate the energy using Eq. (3.46)

% 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, $\mathtt{u}$ is the vector $\mathbf{u} = [u_1^n, \ldots, u_{N-1}^n]^T$ (as Dirichlet boundary conditions are used) and need to be concatenated with 0 in the calculation of the potential energy as the boundaries need to be included in the calculation, despite them being 0![2] Figure 3.2 shows the plot of the normalised energy according to Eq. (3.36) and shows that the deviation of $\mathfrak{h}^n$ is within machine precision.

Neumann boundary conditions

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

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

---

[2] As can be seen from the definition of $\mathfrak{v}$ in Eq. (3.46), the domain used for the inner product is $\underline{d} = \{0, \ldots, N-1\}$ and $\mathfrak{v}$ contains a forward difference in its definition requiring u_N^n as well.
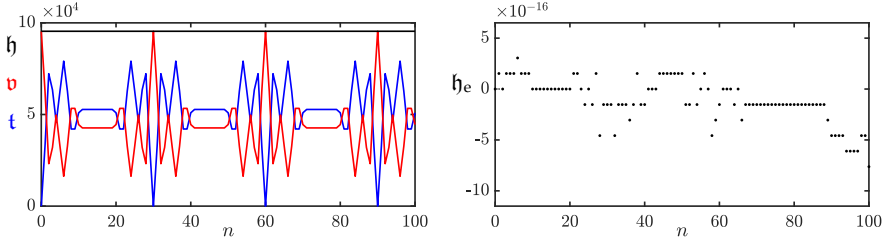
Fig. 3.2: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.36)) and shows that the deviation of the energy is within machine precision.

where the boundary term

$$\mathfrak{b} = T(\delta_t.u_N^n)(\mu_{x-}\delta_{x+}u_N^n) - T(\delta_t.u_0^n)(\mu_{x-}\delta_{x+}u_0^n),$$

$$\xleftarrow{\text{Eq. (2.27b)}} \quad = T(\delta_t.u_N^n)(\delta_x.u_N^n) - T(\delta_t.u_0^n)(\delta_x.u_0^n).$$

As the Neumann boundary condition states that

$$\delta_x.u_0^n = \delta_x.u_N^n = 0$$

the boundary term vanishes and the energy balance results in

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v},$$

$$\text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\left(\|\delta_{t-}u_l^n\|_d'\right)^2, \quad \text{and} \quad \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}. \tag{3.47}$$

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

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

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

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

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

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

3.4.4 Stability Analysis using Energy Analysis Techniques

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

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

Mass-spring system

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

$$x^2 + y^2 + 2axy \tag{3.48}$$

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

Equation (3.48) can be used to prove stability for the mass spring system using the energy balance in Eq. (3.41). One can easily conclude that t is non-negative due to the fact that $M > 0$ and $(\delta_{t-}u^n)$ is squared. The potential energy $\mathfrak{v}$, however, is of indefinite sign. Expanding the operators in Eq. (3.41) yields

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

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

Dividing all terms by $M/2k^2$ this equation is of the form in Eq. (3.48):

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

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

$$-1 < \frac{Kk^2}{2M} - 1 < 1$$

$$0 < \frac{Kk^2}{2M} < 2$$

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

$$k < 2\sqrt{\frac{M}{K}} \tag{3.49}$$

which is identical to the definition in Eq. (3.28).

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. (3.46) 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 (3.16e) 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 [20]

$$\|\delta_{x+}u_l^n\|_{\underline{d}} \leq \frac{2}{h}\|u_l^n\|_d' \leq \frac{2}{h}\|u_l^n\|_d, \tag{3.50}$$

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. (3.46) yields

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \geq \frac{1}{2}\left(1 - \lambda^2\right)\|\delta_{t-}u_l^n\|_d^2 + \frac{c^2}{2}\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2, \qquad (3.51)$$

and is non-negative for

$$1 - \lambda^2 \geq 0,$$
$$\lambda \leq 1.$$

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

3.5 Modal Analysis

Modes are the resonant frequencies of a system. The amount of modes that a discrete system contains depends on the number of moving points. A mass-spring system thus has one resonating mode, but – as briefly touched upon in Section 2.4.3 – a FD scheme of the 1D wave equation with $N = 30$ and Dirichlet boundary conditions will have 29 modes. Modal analysis can be used to obtain objective data on what modes a FD scheme should contain. This can then be used to determine whether this matches one's expectations or whether the output of the system matches what the analysis predicted. This section will show how to numerically obtain the modal frequencies of a FD implementation using the 1D wave equation as a test case.

[some citation here]

We start by using the matrix form of the 1D wave equation from Eq. (3.5)

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

Following [20] we can assume a test solution of the form $\mathbf{u}^n = z^n\boldsymbol{\phi}$. Substituting this into the above equation yields the characteristic equation

[more explanation, perhaps refer to von neumann analysis in 3.3]

$$(z - 2 + z^{-1})\boldsymbol{\phi} = c^2k^2\mathbf{D}_{xx}\boldsymbol{\phi}. \qquad (3.52)$$

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

$$z_p - 2 + z_p^{-1} = c^2k^2\text{eig}_p(\mathbf{D}_{xx}),$$
$$z_p + \left(-2 - c^2k^2\text{eig}_p(\mathbf{D}_{xx})\right) + z_p^{-1} = 0. \qquad (3.53)$$

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

$\sigma_p \leq 0$ of the pth mode.[3] 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

$$e^{j\omega_p k} + e^{-j\omega_p k} - 2 - c^2 k^2 \text{eig}_p(\mathbf{D}_{xx}) = 0,$$

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

Finally, using the trigonometric identity in Eq. (3.20a) we get

$$\sin^2(\omega_p k/2) + \frac{c^2 k^2}{4} \text{eig}_p(\mathbf{D}_{xx}) = 0,$$

$$\sin(\omega_p k/2) = \frac{ck}{2} \sqrt{-\text{eig}_p(\mathbf{D}_{xx})},$$

$$\omega_p = \frac{2}{k} \sin^{-1} \left(\frac{ck}{2} \sqrt{-\text{eig}_p(\mathbf{D}_{xx})} \right). \tag{3.54}$$

and can be rewritten to

$$f_p = \frac{1}{\pi k} \sin^{-1} \left(\frac{ck}{2} \sqrt{-\text{eig}_p(\mathbf{D}_{xx})} \right) \tag{3.55}$$

to get the modal frequency of the pth 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 an FD scheme, over 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 (3.20) can not be used directly. Although the eigenfrequency calculation needs to be done on a larger matrix, it allows for a more general and direct way to calculate the modal frequencies and damping coefficients per mode.

---

[3]Notice that regardless of the possible damping coefficient per mode, the eventual amplitude of each will mostly be determined by the locations of the excitation and output as discussed in Section 2.4.3.
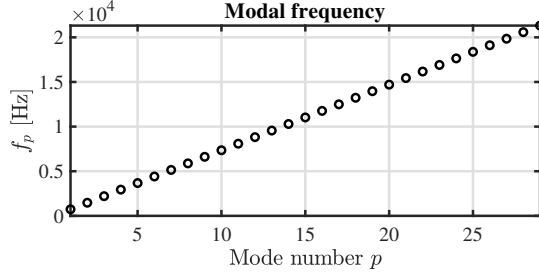
Fig. 3.3: Modal frequencies of the 1D wave equation with the parameters given in Table 2.1.

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

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{3.56}$$

can be rewritten to

$$\underbrace{\begin{bmatrix} \mathbf{u}^{n+1} \\ \mathbf{u}^n \end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix} \mathbf{A}^{-1}\mathbf{B} & \mathbf{A}^{-1}\mathbf{C} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} \mathbf{u}^n \\ \mathbf{u}^{n-1} \end{bmatrix}}_{\mathbf{w}^n} \tag{3.57}$$

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

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

$$z\phi = \mathbf{Q}\phi, \tag{3.58}$$

which can be solved for the pth eigenvalue as

$$z_p = \text{eig}_p(\mathbf{Q}). \tag{3.59}$$

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

$$e^{s_p k} = \text{eig}_p(\mathbf{Q}),$$
$$s_p = \frac{1}{k} \ln\left(\text{eig}_p(\mathbf{Q})\right). \tag{3.60}$$

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

$$\omega_p = \mathfrak{I}(s_p) \quad \text{and} \quad \sigma_p = \mathfrak{R}(s_p), \tag{3.61}$$

where $\mathfrak{I}(\cdot)$ and $\mathfrak{R}(\cdot)$ 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. (3.60) 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.

Part II

Resonators

Resonators

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

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

Chapter 4

Stiff string

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

This chapter presents the PDE of the stiff string in continuous time, and goes through the discretisation process. The analysis techniques presented in Chapter 3 will then be applied to the resulting FD scheme and derived in detail. Unless denoted otherwise, this chapter follows [20].

4.1 Continuous time

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

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

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

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

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

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

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

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

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

with inharmonicity coefficient

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

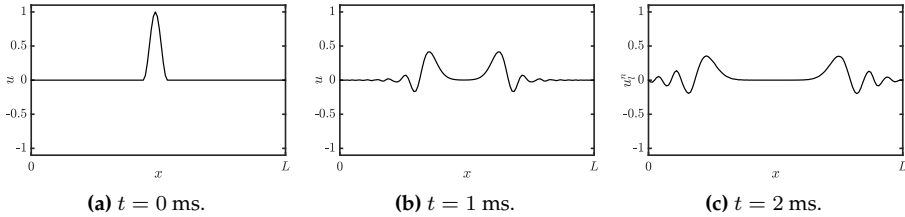Frequency dispersion and inharmonicity will be further discussed in Section 4.2.3.

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

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

4.1.1 Adding Losses

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

using a mixed derivative $\partial_t \partial_x^2$. This way of frequency-dependent damping first appeared in [57] and has been used extensively in the literature since. A damped stiff string can be modelled as

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

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

A more compact way to write Eq. (4.4), and as is also found often in the literature [20], is to divide both sides by ρA to get

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

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

Boundary Conditions

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

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

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

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

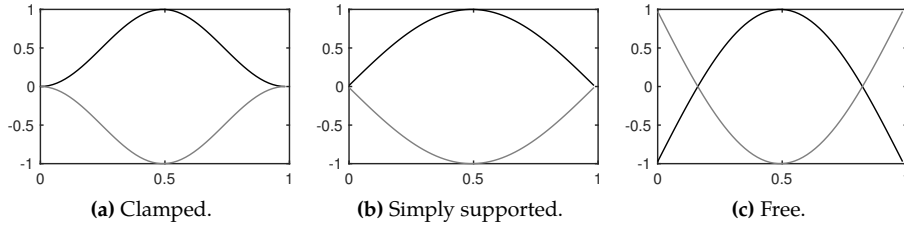at $x = 0, L$. See Figure 4.2 for plots of the first modal shape for each respective boundary condition.

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

4.2 Discrete Time

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

Equation (4.5) can be discretised as

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

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:

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

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

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

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

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

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

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

The reason a backwards difference is used here is to keep the system *explicit*. A scheme is explicit if the values of u_l^{n+1} can explicitly be calculated from known values. If this is not the case and values of fx. u_{l+1}^{n+1} and u_{l-1}^{n+1} are required to calculate u_l^{n+1}, the scheme is called an *implicit*. An example of an implicit scheme using the centred operator for the temporal derivative in the frequency-dependent damping term instead can be found in Section 4.6.

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

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

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

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

with

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

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

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

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

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

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

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

Stencil

As done in Section 2.4.2, a stencil for the FD scheme implementing the damped stiff string can be created. This is shown in Figure 4.3. In order to calculate u_l^{n+1}, 5 points at the current time step are needed due to the 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
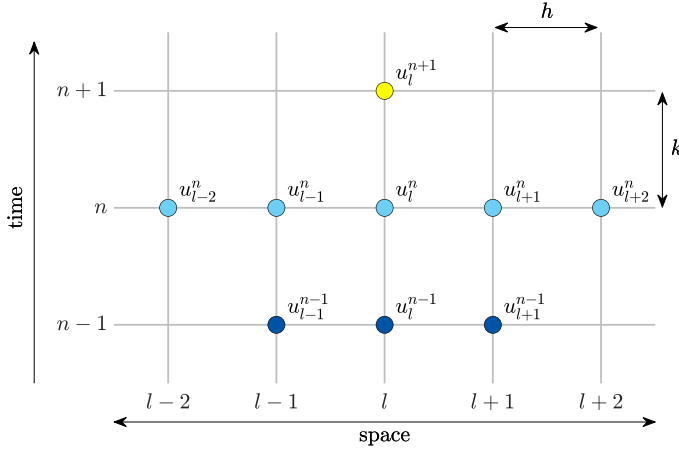
Fig. 4.3: The stencil for the damped stiff string scheme in Eq. (4.7). (Adapted from [A].)

conditions in (4.6) yields

$$u_l^n = \delta_{x\pm} u_l^n = 0 \quad \text{(clamped)} \tag{4.16a}$$
$$u_l^n = \delta_{xx} u_l^n = 0 \quad \text{(simply supported)} \tag{4.16b}$$
$$\delta_{xx} u_l^n = \delta_x. \delta_{xx} u_l^n = 0 \quad \text{(free)} \tag{4.16c}$$

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

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

insert figure showing virtual grid points somewhere in this section

Clamped

Expanding the operators for the clamped condition yields

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

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

Simply supported

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

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

$$\frac{1}{h^2} \left(u^n_1 - 2u^n_0 + u^n_{-1} \right) = 0,$$

$$\xleftrightarrow{u^n_0 = 0} \quad u^n_1 + u^n_{-1} = 0,$$

$$u^n_{-1} = -u^n_1, \tag{4.18}$$

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

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

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

$$(1 + \sigma_0 k)u^{n+1}_1 = \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2} \right) u^n_1 + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2} \right) u^n_2$$

$$- \mu^2 u^n_3 + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2} \right) u^{n-1}_1 - \frac{2\sigma_1 k}{h^2} u^{n-1}_2. \tag{4.19}$$

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

$$(1 + \sigma_0 k)u^{n+1}_{N-1} = \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2} \right) u^n_{N-1} + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2} \right) u^n_{N-2}$$

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

Free

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

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

$$= \frac{1}{2h^3} \left(e^2_{x+} - 2e_{x+} + 1 - (1 - 2e_{x-} + e^2_{x-}) \right),$$

$$= \frac{1}{2h^3} \left(e^2_{x+} - 2e_{x+} + 2e_{x-} - e^2_{x-} \right), \tag{4.21}$$

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

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

$$u^n_{-2} = u^n_2 - 2u^n_1 + 2u^n_{-1}.$$

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

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

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

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

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

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

4.2.2 Implementation and Matrix Form

When using MATLAB, for a more compact implementation, it is useful to write the scheme in matrix form (see Section 3.1.2). The FD scheme of the stiff string in (4.7) can be written as

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

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 boundary conditions. For clamped conditions, the state vectors ($\mathbf{u}^{n+1}$, $\mathbf{u}^n$ and $\mathbf{u}^{n-1}$) and matrices will be of size $(N-3) \times 1$ and $(N-3) \times (N-3)$ respectively. The $\mathbf{D}_{xx}$ matrix will be of the form given in (3.3) and the matrix form of the δ_{xxxx} operator is

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

For simply supported conditions, the state vectors and matrices will be of size $(N-1) \times 1$ and $(N-1) \times (N-1)$ respectively. Again, $\mathbf{D}_{xx}$ is as defined in (3.3) and $\mathbf{D}_{xxxx}$ can be obtained by multiplying two $\mathbf{D}_{xx}$ matrices according to

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

Finally for free boundary conditions as given in (4.16c), the state vectors and matrices are $(N+1) \times 1$ and $(N+1) \times (N+1)$ respectively. Now, the $\mathbf{D}_{xx}$ matrix is of the form in (3.4) instead, and

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

4.2.3 Parameters and output

The values of the parameters naturally determine the properties of the output sound. Where in the 1D wave equation, only the fundamental frequency f_0 could be affected, the stiff string has much more aspects that can be changed. See Table 4.1 for parameters most commonly used in this project. There exists a formula to calculate the loss coefficients σ_0 and σ_1 from T_{60} values at different frequencies (see [20, Eq (7.29)]). During this project, however, these values have been tuned by ear and are usually set to be approximately those found in Table 4.1.

Output

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

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

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

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

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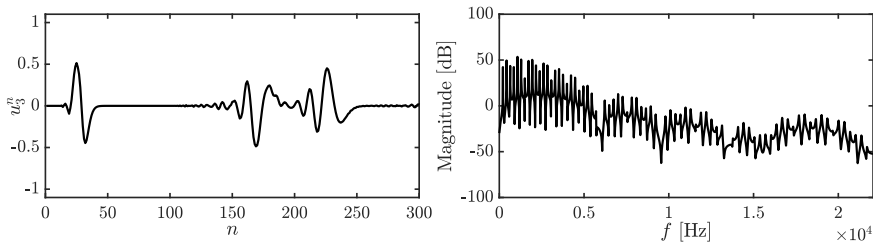

Fig. 4.4: The time-domain and frequency-domain output of the stiff string. The parameters are set as in Table 4.1 with $E = 7 \cdot 10^{11}$ to highlight dispersive effects and $T = 1129$. From the left panel, one can observe that over time, dispersive effects show due to the stiffness in the string.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

4.3 von Neumann Analysis and Stability Condition

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

Using the definitions found in Eq. (3.21) for the temporal operators and Eqs. (3.22a) and (3.22b) for the spatial operators, the frequency-domain representation of Eq. (4.7) can be obtained:

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

and after collecting the terms, the characteristic equation follows:

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

Rewriting this to the form in Eq. (3.24), and using $\mathcal{S} = \sin^2(\beta h/2)$ for brevity, yields

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

Stability of the system can then be proven using condition (3.25) and substituting the coefficients into this condition yields

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

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

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

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

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

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

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

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

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

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

> different wording

$$\frac{4\kappa^2 k^2}{h^4} + \frac{c^2 k^2 + 4\sigma_1 k}{h^2} \leq 1,$$

$$4\kappa^2 k^2 + (c^2 k^2 + 4\sigma_1 k)h^2 \leq h^4,$$

$$h^4 - (c^2 k^2 + 4\sigma_1 k)h^2 - 4\kappa^2 k^2 \geq 0,$$

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

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

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

4.4 Energy Analysis

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

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

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

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

The first step is to take the inner product (see Eq. (3.8)) of the scheme with $(\delta_{t\cdot}u_l^n)$ over discrete domain d:

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

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

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

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

The damping term is defined as

> virtual grid points needed for freq-dep damping term..

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

and the boundary term $\mathfrak{b}$ appears after rewriting Equation (4.29) using summation by parts (see Section 3.2.2). Specifically, using Eq. (3.13a) for the second term and Eq. (3.15b) for the third, we get

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

$$= \mathfrak{b} - \mathfrak{q}$$

where the boundary term becomes

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

$$+ EI\Big((\delta_{t\cdot} u_N^n)(\delta_{x+}\delta_{xx} u_N^n) - (\delta_{xx} u_N^n)(\delta_{x-}\delta_{t\cdot} u_N^n) \Big)$$

$$+ EI\Big(- (\delta_{t\cdot} u_0^n)(\delta_{x-}\delta_{xx} u_0^n) + (\delta_{xx} u_0^n)(\delta_{x+}\delta_{t\cdot} u_0^n) \Big).$$

For the clamped and simply supported boundary conditions in (4.16a) and (4.16b) it can easily be shown that $\mathfrak{b} = 0$. If free conditions as in Eq. (4.16c) are used, the boundary conditions will vanish when the primed inner product in Eq. (3.9) is used in Step 1 and identity (3.15c) is used when performing summation by parts. Here, we continue with the clamped / simply supported case.

Isolating δ_{t+} to obtain the total energy $\mathfrak{h}$ in the definition for $\delta_{t+}\mathfrak{h}$ above, requires identities (3.16a) and (3.16b) and yields

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

$$= -\mathfrak{q}.$$

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho A}{2}\|\delta_{t-} u_l^n\|_d^2, \quad \text{and}$$

$$\mathfrak{v} = \frac{T}{2}\langle \delta_{x+} u_l^n, e_{t-}\delta_{x+} u_l^n \rangle_{\underline{d}} + \frac{EI}{2}\langle \delta_{xx} u_l^n, e_{t-}\delta_{xx} u_l^n \rangle_{\overline{\underline{d}}}. \tag{4.32}$$

Step 3: Check units

Comparing the acquired energy balance in Eq. (4.32) to the energy balance for the 1D wave equation in Eq. (3.46), one can observe that the balances are nearly identical, the only difference being the second term in the definition for $\mathfrak{v}$ in Eq. (4.32). Writing this term out in units, and recalling that Pa (the unit for E) in SI units is $\mathrm{kg \cdot m^{-1} \cdot s^{-2}}$, yields

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

and indeed has the correct units.

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

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

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

and 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 damping is ignored but can be found in Appendix **??**. Figure 4.5 shows that the deviation of the total energy calculated using Eq. (3.37) is within machine precision.

> Add to appendix or refer to a gist

```
1  %%%% Before the main loop: %%%%
2
3  % Initialise Dx+ operator to calculate potential energy due to tension
4  % As the domain is reduced by one, the matrix needs to be of size N x N
5  Dxp = sparse(1:N, 1:N, -ones(1, N), N, N) + ...
6          sparse(1:N-1, 2:N, ones(1, N-1), N, N);
7
8  %%%% In the main loop: %%%%
9
10 % energy in the system
11 kinEnergy(n) = rho * A * h / 2 * sum((1/k * (u - uPrev)).^2);
12 potEnergy(n) = T / 2 * h * sum((Dxp * [0; u]) .* (Dxp * [0; uPrev]))
       ... + E * I * h / 2 * sum((Dxx * u) .* (Dxx * uPrev));
```

Algorithm 4.1: Calculating $\mathfrak{h}$ for the simply supported boundary condition.
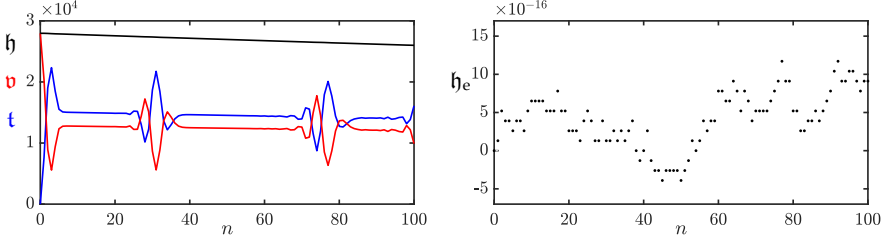
Fig. 4.5: The kinetic (blue), potential (red), and total (black) energy of an implementation of the stiff string are plotted in the left panel. Notice that the damping present in the system causes ℏ to decrease. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.

4.5 Modal Analysis

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

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

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

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

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

which has the following solutions

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

The (angular) frequency of the pth mode can then be obtained using $\mathfrak{I}(s_p)$ and the damping per mode as $\mathfrak{R}(s_p)$. Only selecting the non-negative frequencies obtained from $\mathfrak{I}(s_p)$, these can be plotted and are shown in Figure 4.6. The parameters used are the ones found in Table 4.1 with $T = 1885$ N, and $E = 2 \cdot 10^{14}$ to highlight inharmonic behaviour. The left panel shows that the system is indeed inharmonic, i.e., modal frequencies increase more as the modal number increases. The right panel shows that higher modes exhibit a

higher amount of damping. This is due to the frequency-dependent damping term. If $\sigma_1 = 0$ in (4.7), it can be shown that $\sigma_p = \sigma_0$ for every mode p (in this case -1).
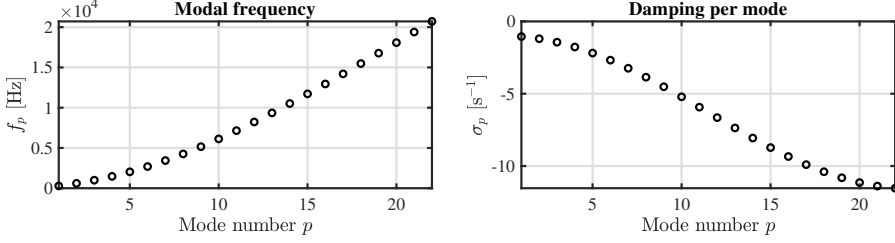

Fig. 4.6: The modal frequencies and damping per mode for the stiff string using the values in Table 4.1 and $T = 1885$ and $E = 2 \cdot 10^{14}$ to highlight effects of stiffness. Notice form the left panel that the frequency increases exponentially with the mode number. The right panel shows that higher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

4.6 Implicit Scheme

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

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

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

Luckily, as the scheme is linear, it can be treated as a system of linear equations and solved following the technique described in Section B.3. The drawback is that this requires one matrix inversion per iteration which can be extremely costly (see Section **??**). However, both von Neumann and modal analysis (below) show that using the centred instead of the backwards operator has a positive effect on the stability and the modal behaviour of the scheme.

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

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

where

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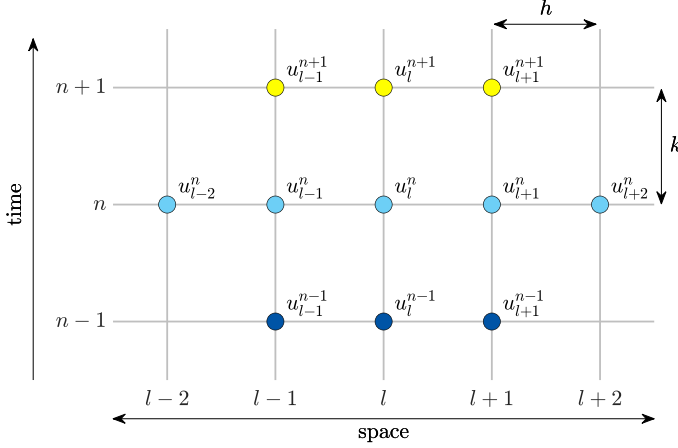

Fig. 4.7: The stencil for the damped stiff string scheme in (4.36).

4.6.1   von Neumann analysis

Using the same process as in Section 4.3, the definitions in Section 3.3 can be used to obtain a frequency-domain representation of the FD scheme in Eq. (4.36) can be obtained:

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

and collecting the terms, yields the following characteristic equation:

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

Rewriting this to the form found in Eq. (3.24) and, again, using $\mathcal{S} = \sin^2(\beta h/2)$ yields:

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

Stability of the system can then be proven using condition (3.25), and after substitution of the coefficients yields

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

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

$$\leq 1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}\mathcal{S},$$

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

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

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

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

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

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

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

and yields the following stability condition:

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

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

4.6.2 Modal analysis

As the matrix form of the implicit FD scheme in Eq. (4.37) matches the form in Eq. (3.56), one can perform a modal analysis by writing the scheme in one-step form as explained in Section 3.5.1. The results of the analysis are shown in Figure 4.8. To highlight the difference between using the backwards and centred difference for the frequency-dependent damping term, σ_1 has been set to 1, which is much higher than one would normally use.

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

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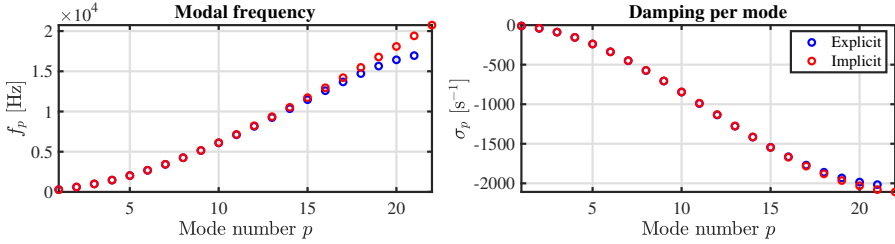

Fig. 4.8: A comparison between the modal frequencies and damping per mode of the explicit (blue) and implicit (red) scheme. Here, $T = 1885$, $E = 2 \cdot 10^{14}$ and $\sigma_1 = 1$ to highlight differences between the two schemes. One can observe that the modes of the implicit scheme follow the expected exponential pattern for the stiff string, where the explicit scheme shows numerically dispersive effects. Furthermore, due to the absence of σ_1 in the stability condition in Eq. (4.40) and allows for one more grid point

4.6.3 Conclusion

This section presented an implicit discretisation of the stiff string where the centred operator has been used to discretise the temporal derivative in the frequency-dependent damping term. By means of stability analysis and modal

analysis some advantages that the implicit scheme has over its explicit counterpart (presented in Section 4.2) have been shown.

As these advantages only show for higher values of σ_1, much higher than the ones used in this project, it has been chosen to use the explicit scheme for all further implementation. The decrease in accuracy is negligible for lower values of σ_1 and the calculation of the scheme becomes orders of magnitude more computationally expensive if the implicit scheme is used.

Chapter 5

Brass

In this work, the wave propagation in tubes is approximated using a 1D system

5.1 Webster's Equation

The main difference between the PDE modelling a the 1D brass PDE and the 1D wave equation is the presence of a variable cross-section. For low-amplitude vibrations in an (axially symmetric) acoustic tube and for which the wavelengths are much larger than the radius of this tube, one can describe its dynamics using *Webster*'s equation [58]

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

with *acoustic potential* $\Psi = \Psi(x,t)$ (in m^2/s), $S = S(x)$ is the cross sectional area (in m^2) and the speed of sound in air c (in m/s). If $S(x)$ is constant, Eq. (5.1) reduces to the 1D wave equation in Eq. (2.38). The acoustic potential can be related to pressure $p = p(x,t)$ (in Pa) and particle velocity $v = v(x,t)$ (in m/s) according to

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

5.1.1 Discretisation

Introducing interleaved gridpoints at $n - 1/2$ and $n + 1/2$ for S, a we can discretise Eq. (5.1) (following [22]) to

$$\bar{S}\delta_{tt}\Psi_l^n = c^2\delta_{x+}(S_{l-1/2}(\delta_{x-}\Psi_l^n)), \tag{5.3}$$

where

$$\bar{S} = \mu_{x+}S_{l-1/2} = \frac{S_{l+1/2} + S_{l-1/2}}{2}. \tag{5.4}$$

The right side of the equation in (5.3) contains an operator applied to two grid functions (S and Ψ) multiplied onto each other. In order to expand this, we need to use the product rule in Eq. (3.17) which is

$$\delta_{x+}(u_l w_l) = (\delta_{x+}u_l)(\mu_{x+}w_l) + (\mu_{x+}u_l)(\delta_{x+}w_l). \tag{5.5}$$

In the case of (5.3), $u_l \triangleq S_{l-1/2}$ and $w_l \triangleq \delta_{x-}\Psi_l^n$. Expanding (retaining the notation for $\bar{S}$) and solving for Ψ_l^{n+1} yields (Appendix **??**)

$$\Psi_l^{n+1} = 2(1-\lambda^2)\Psi_l^n - \Psi_l^{n-1} + \frac{\lambda^2 S_{l+1/2}}{\bar{S}}\Psi_{l+1}^n + \frac{\lambda^2 S_{l-1/2}}{\bar{S}}\Psi_{l-1}^n, \tag{5.6}$$

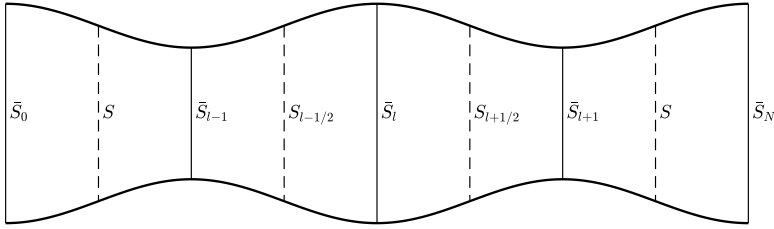which is identical to Eq. (19.51) in [22]. Also see Figure 5.1

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

5.1.2 Boundary Conditions

The choices for boundary conditions in an acoustic tube are open and closed, defined as [22]

$$\partial_t \Psi = 0 \quad \text{(open, Dirichlet)} \tag{5.7a}$$
$$\partial_x \Psi = 0 \quad \text{(closed, Neumann)}, \tag{5.7b}$$

at the ends of the tube. This might be slightly counter-intuitive as in the case of a string "closed" might imply the "clamped" or Dirichlet boundary condition. The opposite can be intuitively shown imagining a wave front with a positive acoustic potential moving through a tube and hitting a closed end. What comes back is also a wave front with a positive acoustic potential, i.e., the sign of the potential does not flip, which also happens using the free or Neumann condition for the string.

In this case we follow [20, Chapter 9] and use the following

$$\partial_x \Psi(0,t) = 0 \quad \text{and} \quad \partial_t \Psi(L,t) = 0 \tag{5.8}$$

i.e. closed at the left end and open at the right end. In discrete time we have two choices for the closed condition

$$\delta_{x\cdot}\Psi_0^n = 0 \;\Rightarrow\; \Psi_{-1}^n = \Psi_1^n \quad \text{(centered)} \tag{5.9}$$

At the left boundary we can now solve Eq. (5.6) for the centered case:

$$\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\Psi_{-1}^n$$

$$\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2(S_{1/2} + S_{-1/2})}{\bar{S}_0}\Psi_1^n$$

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

and the non-centered case

$$\Psi_0^{n+1} = 2(1-\lambda^2)\Psi_0^n - \Psi_0^{n-1} + \frac{\lambda^2 S_{1/2}}{\bar{S}_0}\Psi_1^n + \frac{\lambda^2 S_{-1/2}}{\bar{S}_0}\Psi_0^n. \tag{5.10}$$

As can be seen from the equations above, we need undefined points $\bar{S}_0$ and $S_{-1/2}$. At the left boundary, we set $\bar{S}_0 = S_0$ from which, we can calculate $S_{-1/2}$:

$$S_0 = \frac{1}{2}(S_{1/2} + S_{-1/2}) \;\Rightarrow\; S_{-1/2} = 2S_0 - S_{1/2} \tag{5.11}$$

The same can be done for the right boundary ($\bar{S}_N = S_N$) if this is chosen to be anything else but open (e.g., closed or radiating – see Section 5.1.2):

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

For now though, we follow the conditions given in (5.8) and we can simply set the right boundary to its initial state

$$\Psi_N^n = \Psi_N^0 \tag{5.13}$$

which is normally 0. A more realistic open end is a radiating one, which can be found below.

Radiating end

We can change the condition presented in Eq. (5.8) to a radiating end,

$$\partial_x \Psi(L,t) = -a_1\partial_t \Psi(L,t) - a_2\Psi(L,t) \tag{5.14}$$

where [20]

$$a_1 = \frac{1}{2(0.8216)^2 c} \quad \text{and} \quad a_2 = \frac{L}{0.8216\sqrt{S_0 S(1)/\pi}}. \tag{5.15}$$

taken from [?] and are valid for a tube terminating on an infinite plane. The terms in Eq. (5.14) are a damping and an inertia term where a_1 is a loss coefficient (in s/m) and a_2 is the **inertia coefficient** (in m^{-1}). The centered and non-centered case are defined as

$$
\begin{aligned}
\delta_{x\cdot}\Psi_N^n = 0 &\Rightarrow \Psi_{N+1}^n = \Psi_{N-1}^n \quad \text{(centered)} \\
\delta_{x+}\Psi_N^n = 0 &\Rightarrow \Psi_{N+1}^n = \Psi_N^n \quad \text{(non-centered)}
\end{aligned}
\tag{5.16}
$$

First, we solve Eq. (5.14) for the centered (Eq. (9.16) in [20])

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

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

$$
\frac{1}{2h}(\Psi_{N+1}^n - \Psi_{N-1}^n) = -\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1})
$$

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

which can be substituted into Eq. (5.6) (Appendix **??**)

$$
\Psi_N^{n+1} = \frac{2(1-\lambda^2)\Psi_N^n - \Psi_N^{n-1} + \frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\left(\frac{a_1}{k} - a_2\right)\Psi_N^{n-1} + 2\lambda^2\Psi_{N-1}^n}{\left(1 + \left(\frac{a_1}{k} + a_2\right)\frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\right)}.
\tag{5.19}
$$

The same can be done for the non-centered case (Eq. (9.15) in [20])

$$
\delta_{x+}\Psi_N^n = -a_1\delta_{t\cdot}\Psi_N^n - a_2\mu_{t\cdot}\Psi_N^n
\tag{5.20}
$$

which when solved for Ψ_{N+1}^n yields

$$
\frac{1}{h}(\Psi_{N+1}^n - \Psi_N^n) = -\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1})
$$

$$
\Psi_{N+1}^n = h\left(-\frac{a_1}{2k}(\Psi_N^{n+1} - \Psi_N^{n-1}) - \frac{a_2}{2}(\Psi_N^{n+1} + \Psi_N^{n-1})\right) + \Psi_N^n.
\tag{5.21}
$$

Substituted into Eq. (5.6) yields (Appendix **??**)

$$
\Psi_N^{n+1} = \frac{2(1-\lambda^2)\Psi_N^n - \Psi_N^{n-1} + \frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\left(\frac{a_1}{2k} - \frac{a_2}{2}\right)\Psi_N^{n-1} + \frac{\lambda^2 S_{N+1/2}}{\bar{S}_N}\Psi_N^n + \frac{\lambda^2 S_{N-1/2}}{\bar{S}_N}\Psi_{N-1}^n}{\left(1 + \left(\frac{a_1}{2k} + \frac{a_2}{2}\right)\frac{h\lambda^2 S_{N+1/2}}{\bar{S}_N}\right)}.
\tag{5.22}
$$

5.2 First-order system

This will be the first appearance of a first-order system.

5.2.1 Adding Radiation

Following [56] we can add a radiation to the schemes using a circuit representation of the Levine and Schwinger radiation model (See Figure \ref{fig:radiation}). The system can be described as

$$\bar{v} = \mu_{t+}v_{(1)} + \frac{1}{R_2}\mu_{t+}p_{(1)} + C_{\mathrm{r}}\delta_{t+}p_{(1)}, \tag{5.23a}$$

$$\bar{p} = L_{\mathrm{r}}\delta_{t+}v_{(1)}, \tag{5.23b}$$

$$\bar{p} = \left(1 + \frac{R_1}{R_2}\right)\mu_{t+}p_{(1)} + R_1 C_{\mathrm{r}}\delta_{t+}p_{(1)}, \tag{5.23c}$$

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

$$\bar{p} = \mu_{t+}p_N^n, \quad \bar{S}_N\bar{v} = \mu_{x-}\left(S_{N+1/2}v_{N+1/2}^{n+1/2}\right). \tag{5.24}$$

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

$$p_N^{n+1} = p_N^n - \frac{\rho_0 c\lambda}{\bar{S}_N}\left(S_{N+1/2}v_{N+1/2}^{n+1/2} - S_{N-1/2}v_{N-1/2}^{n+1/2}\right), \tag{5.25}$$

and, similar to (**??**), rewriting this to

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

$$p_N^{n+1} = p_N^n - \frac{2\rho_0 c\lambda}{\bar{S}_N}\left(\bar{S}_N\bar{v} - S_{N-1/2}v_{N-1/2}^{n+1/2}\right). \tag{5.26}$$

We can then find a definition for $\bar{v}$ by expanding system (5.23) and make Eq. (5.23a) solely dependent on known values of $v_{(1)}$, $p_{(1)}$ and p_N^n and the unknown p_N^{n+1} (as we can solve for the latter using (5.26)).

$$\bar{v} = \frac{1}{2}\left(v_{(1)}^{n+1} + v_{(1)}^n\right) + \left(\frac{1}{2R_2} + \frac{C_{\mathrm{r}}}{k}\right)p_{(1)}^{n+1} + \left(\frac{1}{2R_2} - \frac{C_{\mathrm{r}}}{k}\right)p_{(1)}^n \tag{5.27}$$

where, after expanding Eq. (5.23b)

$$v_{(1)}^{n+1} = \frac{k}{L_{\mathrm{r}}}\bar{p} + v_{(1)}^n, \tag{5.28}$$

and Eq. (5.23c)

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

$$\bar{p} = \frac{1}{2}\left(1 + \frac{R_1}{R_2}\right)\left(p_{(1)}^{n+1} + p_{(1)}^n\right) + \frac{R_1 C_r}{k}\left(p_{(1)}^{n+1} - p_{(1)}^n\right)$$

$$\left(\frac{1}{2} + \frac{R_1}{2R_2} + \frac{R_1 C_r}{k}\right)p_{(1)}^{n+1} = \bar{p} + \left(\frac{R_1 C_r}{k} - \frac{1}{2} - \frac{R_1}{2R_2}\right)p_{(1)}^n$$

$$p_{(1)}^{n+1} = \underbrace{\left(\frac{2R_2 k}{2R_1 R_2 C_r + k(R_1 + R_2)}\right)}_{\zeta_1}\bar{p} + \underbrace{\left(\frac{2R_1 R_2 C_r - k(R_1 + R_2)}{2R_1 R_2 C_r + k(R_1 + R_2)}\right)}_{\zeta_2}p_{(1)}^n.$$

$$(5.29)$$

Filling these into Eq. (5.27) and using the definition of $\bar{p}$ from Eq. (5.24) yields

$$\bar{v} = \frac{1}{2}\left(\frac{k}{L_r}(\mu_{t+}p_N^n) + 2v_{(1)}^n\right) + \left(\frac{1}{2R_2} + \frac{C_r}{k}\right)\zeta_1\mu_{t+}p_N^n + \left(\frac{1}{2R_2} + \frac{C_r}{k}\right)\zeta_2 p_{(1)}^n + \left(\frac{1}{2R_2} - \frac{C_r}{k}\right)p$$

$$\bar{v} = \underbrace{\left(\frac{k}{2L_r} + \frac{\zeta_1}{2R_2} + \frac{C_r\zeta_1}{k}\right)}_{\zeta_3}\mu_{t+}p_N^n + v_{(1)}^n + \underbrace{\left(\frac{\zeta_2 + 1}{2R_2} + \frac{C_r\zeta_2 - C_r}{k}\right)}_{\zeta_4}p_{(1)}^n. \quad (5.30)$$

Finally, filling in this definition for $\bar{v}$ into Eq. (5.26)

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

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

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

$$(5.31)$$

5.2.2 Energy

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

$$\mathfrak{b}_r = (\mu_{t+}p_N)\underbrace{\mu_{x+}(S_{N-1/2}v_{N-1/2})}_{\mu_{x-}S_{N+1/2}v_{N+1/2}}, \quad (5.32)$$

using Eq. (5.24) we can rewrite this to

$$\mathfrak{b}_r = \bar{p}\bar{S}_N\bar{v}. \quad (5.33)$$

then we can

Chapter 6

2D Systems

The previous chapters considered systems distributed over (maximally) 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. 2D PDEs can be used to model drum membranes, plate reverbs or simplified instrument bodies, which is what it is mainly used for in this work.

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 **??**.

This chapter starts by providing some additional information about 2D grid functions and operators. Then, the 2D wave equation is presented, and can be interpreted as the 2D-equivalent of the 1D wave equation presented in Section 2.4. Using this model, the analysis techniques presented in Chapter 3 will be extended to 2D and the differences between this and the 1D case will be highlighted.[1] Afterwards, two 2D models used in this work, the thin plate model and the stiff membrane will be described in a similar fashion. The systems modelled in this work are simplified to be rectangular and are defined on a Cartesian coordinate system. Section 6.5 briefly elaborates on radial coordinate systems and their shortcomings. As in previous chapters, unless denoted otherwise, the theory follows [20].

---

[1]The abbreviation 2D will also be used for 'two dimensions'.

6.1 PDEs and FD schemes in 2D

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 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 $x = lh$ and $y = mh$ and time $t = nk$ and $k = 1/f_s$. For simplicity, in this work the grid spacing h is set to be the same in both the x and y directions.

Additional operators

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

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

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

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

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

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

Using these shift operators, a discrete approximation of the Laplacian in Eq. (6.1) can be made[2]

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

and when applied to a grid function yields

$$\Delta u \cong \delta_\Delta u_{l,m}^n = \frac{1}{h^2}\left(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n - 4u_{l,m}^n\right). \tag{6.5}$$

See Figure 6.1a for the stencil of the discrete Laplacian. Similarly, an approxi-

---

[2]Notice that the δ_Δ operator is identical to $\delta_{\Delta\boxplus}$ in [20], but will not be used here as Eq. (6.4) is the only discretisation to the Δ operator used in this work.

mation of the biharmonic operator in Eq. (6.2) can be made as

$$\Delta\Delta \approx \delta_\Delta\delta_\Delta \triangleq \frac{1}{h^4}\Bigg[\left(e_{x+}^2 + e_{x-}^2 + e_{y+}^2 + e_{y-}^2\right) \tag{6.6}$$
$$+ 2\left(e_{x+}e_{y+} + e_{x+}e_{y-} + e_{x-}e_{y+} + e_{x-}e_{y-}\right)$$
$$- 8\left(e_{x+} + e_{x-} + e_{y+} + e_{y-}\right) + 20\Bigg],$$

and when applied to a grid function yields

$$\Delta\Delta u \approx \delta_\Delta\delta_\Delta u_{l,m}^n = \frac{1}{h^4}\Big[\left(u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n\right)$$
$$+ 2\left(u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n\right)$$
$$- 8\left(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n\right) + 20u_{l,m}^n\Big].$$

See Figure 6.1b for the stencil of the discrete biharmonic operator.

(a) The δ_Δ operator in Eq. (6.4).   (b) The $\delta_\Delta\delta_\Delta$ operator in Eq. (6.6).
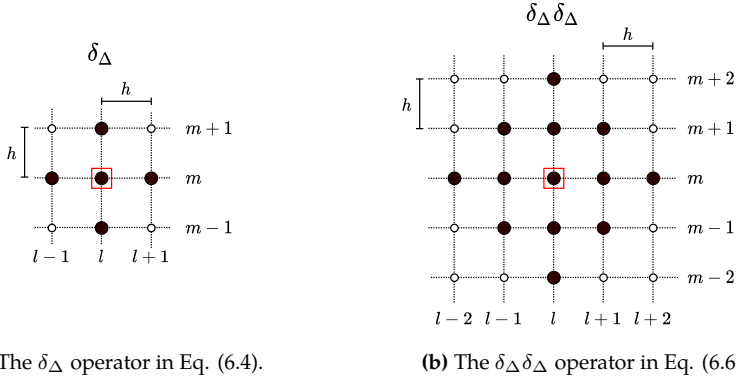
Fig. 6.1: The stencils of the 2D spatial FD operators in Eqs. (6.4) and (6.6) respectively. The red square denotes what grid point the operator is applied to. The stencils follow the same layout as Figure 2.3, but the vertical axis denotes a second spatial dimension rather than time.

6.2 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 due to van Duyne and Smith who used digital waveguides, or more specifically a waveguide mesh, to discretise it [59]. The implementation is identical to the FD scheme that will be presented here.though it is more general here due to λ^2 instead of hard-coded 0.5

This section will present the 2D wave equation in continuous time and its discretisation afterwards. Then it will be used as a test-case to extend the various analysis techniques presented in Chapter 3 to 2D.

6.2.1 Continuous time

Consider a system modelling the 2D wave equation with side lengths L_x and L_y (both in m) and its state described by $u = u(x, y, t)$. The system is defined over $(x, y) \in \mathcal{D}$ with domain $\mathcal{D} = [0, L_x] \times [0, L_y]$ and its motion is described by the following PDE

$$\partial_t^2 u = c^2 \Delta u, \tag{6.7}$$

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 (applied to the boundary) T (in N/m), material density ρ (in kg/m^3) and thickness H (in m).

check whether this wording is right..

Boundary conditions

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

$$\left.\begin{aligned} u(0, y, t) = u(L_x, y, t) = 0 \quad \forall y, \\ u(x, 0, t) = u(x, L_y, t) = 0 \quad \forall x, \end{aligned}\right\} \quad \text{(Dirichlet, fixed),} \tag{6.8a}$$

$$\left.\begin{aligned} \partial_x u(0, y, t) = \partial_x u(L_x, y, t) = 0 \quad \forall y, \\ \partial_y u(x, 0, t) = \partial_y u(x, L_y, t) = 0 \quad \forall x, \end{aligned}\right\} \quad \text{(Neumann, free),} \tag{6.8b}$$

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

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. (6.7) can be discretised to

$$\delta_{tt} u_{l,m}^n = c^2 \delta_\Delta u_{l,m}^n, \tag{6.9}$$

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. (6.5)) and solving for $u_{l,m}^n$ yields the following update equation

$$u_{l,m}^{n+1} = 2u_{l,m}^n - u_{l,m}^{n-1} + \lambda^2 \left(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n - 4u_{l,m}^n \right), \quad (6.10)$$

where the Courant number

$$\lambda = \frac{ck}{h}, \quad (6.11)$$

and needs to abide

$$\lambda \leq \frac{1}{\sqrt{2}} \quad (6.12)$$

for the scheme to be stable (see Section 6.2.4). Writing this condition in terms of the grid spacing places the following limit on h

$$h \geq \sqrt{2}ck . \quad (6.13)$$

Discrete boundary conditions

The continuous-time boundary conditions in Eqs. (6.8) can be discretised to

$$\left.\begin{array}{ll} u_{0,m}^n = u_{N_x,m}^n = 0 & \forall m, \\ u_{l,0}^n = u_{l,N_y}^n = 0 & \forall l, \end{array}\right\} \quad \text{(Dirichlet, fixed)}, \quad (6.14a)$$

$$\left.\begin{array}{ll} \delta_x . u_{0,m}^n = \delta_x . u_{N_x,m}^n = 0 & \forall m, \\ \delta_y . u_{l,0}^n = \delta_y . u_{l,N_y}^n = 0 & \forall l, \end{array}\right\} \quad \text{(Neumann, free)}. \quad (6.14b)$$

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

Stencil

Figure 6.2 shows the stencil of the 2D wave equation FD scheme in Eq. (6.9). Due to the extra dimension, layout of the stencil is made to account for this. The colour-coding for grid points at the various time steps is unchanged.

6.2.3   Implementation and Matrix Form

Similar to how the number of intervals between grid points are calculated 1D in Eq. (2.53), these can be calculated using the following operations:

$$h := \sqrt{2}ck, \;\; N_x := \left\lfloor \frac{L_x}{h} \right\rfloor, \;\; N_y := \left\lfloor \frac{L_y}{h} \right\rfloor, \;\; h := \min\left(\frac{L_x}{N_x}, \frac{L_y}{N_y} \right), \;\; \lambda := \frac{ck}{h}, \quad (6.15)$$

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
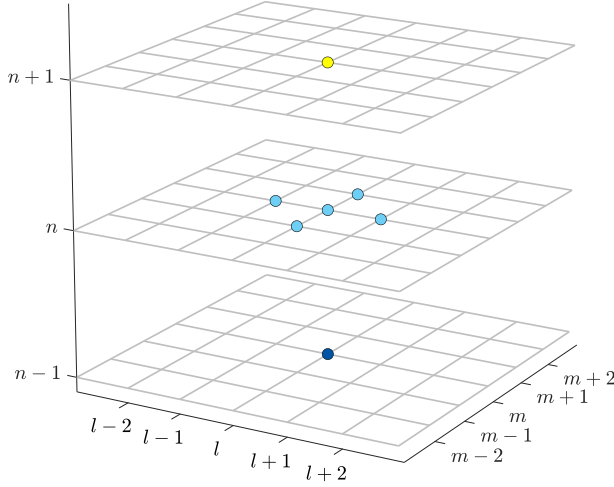
Fig. 6.2: The stencil for the 2D wave equation FD scheme in Eq. (6.9). The grid points use the same colour-coding as previous stencils (see e.g. Figure 2.10).

equation in Eq. (6.10), one could save the states of the system in matrices (as opposed to vectors in the 1D case such as in Section 4.2.2) and directly work with these. Using Dirichlet boundary conditions the $(N_x - 1) \times (N_y - 1)$ state matrix at time index n would be

$$\mathbf{u}^n = \begin{bmatrix} u_{1,1}^n & \cdots & u_{1,N_x-1}^n \\ \vdots & \ddots & \vdots \\ u_{N_y-1,1}^n & \cdots & u_{N_y-1,N_x-1}^n \end{bmatrix}, \tag{6.16}$$

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 **??**). For a more compact and faster 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, e.g., the stiff string in Section 4.2.2. Again using Dirichlet boundary conditions, the stacked state vector of size will be structured as

$$\mathbf{U}^n = [(\mathbf{u}_1^n)^T, \ldots, (\mathbf{u}_{N_x-1}^n)^T]^T, \quad \text{with} \quad \mathbf{u}_l^n = [u_{l,1}^n, \ldots, u_{l,N_y-1}^n]^T, \tag{6.17}$$

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, the *kronecker product* and *kronecker sum* must be introduced [60]. The kronecker product between two arbitrarily-
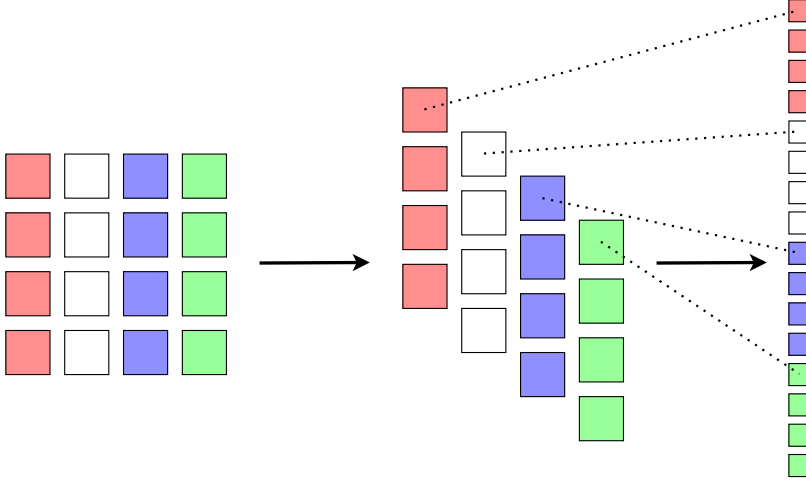
Fig. 6.3: Stacking, or 'flattening' a 4×4 matrix to a 16-element vector.

sized matrices (using their dimensions as a subscript) is

$$\mathbf{A}_{M \times N} \otimes \mathbf{B}_{K \times L} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1N}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{M1}\mathbf{B} & \dots & a_{MN}\mathbf{B} \end{bmatrix}_{MK \times NL} . \quad (6.18)$$

The kronecker sum between two square matrices is

$$\mathbf{A}_{M \times M} \oplus \mathbf{B}_{N \times N} = \mathbf{I}_N \otimes \mathbf{A} + \mathbf{B} \otimes \mathbf{I}_M, \quad (6.19)$$

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

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

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

$$(6.20)$$

Following [44], 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

$$
\mathbf{D}_\Delta = \mathbf{D}_{yy} \oplus \mathbf{D}_{xx} =
\begin{bmatrix}
\ddots & & & & 0 \\
& \mathbf{D}_{yy} & & & \\
& & \mathbf{D}_{yy} & & \\
& & & \mathbf{D}_{yy} & \\
0 & & & & \ddots
\end{bmatrix}
+ \frac{1}{h^2}
\begin{bmatrix}
\ddots & \ddots & & & 0 \\
\ddots & -2\mathbf{I} & \mathbf{I} & & \\
& \mathbf{I} & -2\mathbf{I} & \mathbf{I} & \\
& & \mathbf{I} & -2\mathbf{I} & \ddots \\
0 & & & \ddots & \ddots
\end{bmatrix},
$$
(6.21)

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

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

$$
\mathbf{U}^{n+1} = \left(2\mathbf{I} + c^2 k^2 \mathbf{D}_\Delta\right) \mathbf{U}^n - \mathbf{U}^{n-1},
$$
(6.22)

where the identity matrix is of the same size as $\mathbf{D}_\Delta$. See Appendix C.3 for a MATLAB implementation of the 2D wave equation. As the matrices are extremely sparse (many 0-entries), it is useful to utilise MATLABs optimisation for sparse matrices and use the sparse() function. One can use speye() for sparse identity matrices.

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 lines 40–52 in Appendix C.3).

check if this is still true

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 frequency. 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 the reflections. The right panel shows that the output contains many close-together partials. 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 these reflections and is thus highly inharmonic.
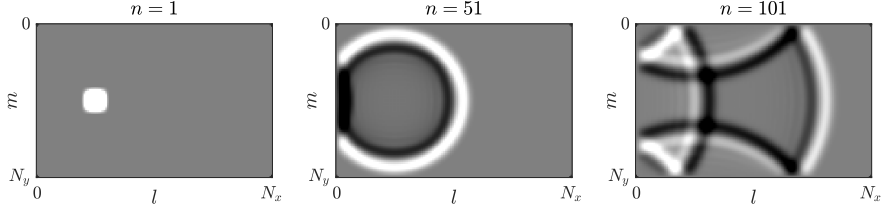
Fig. 6.4: Wave propagation of an implementation of the 2D wave equation with $L_x = 1.5$ m, $L_y = 1$ m and $c = 360$ m/s. The system is excited with a 2D raised cosine at $(0.25L_x, 0.5L_y)$.
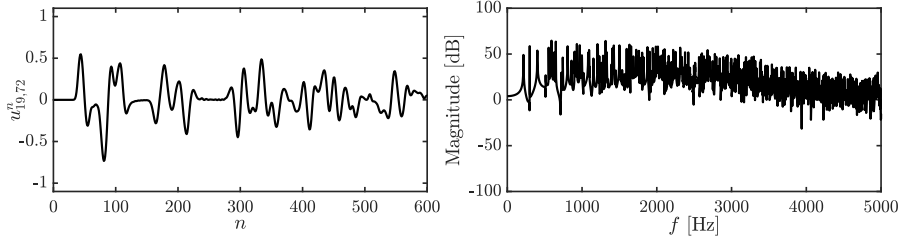

Fig. 6.5: The output of the 2D wave equation at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.4. The partials are extremely close together (notice that only frequency up to 5000 Hz are shown) and is related to the a-periodic nature of the system behaviour.

6.2.4 Frequency Domain Analysis in 2D

Section 3.3 showed how to perform frequency domain analysis to obtain stability conditions for an FD scheme. This section shows extensions to this in 2D and follows [20, Ch. 10].

In 2D, the ansatz in Eq. (3.19) can be extended to

$$u_{l,m}^n \xRightarrow{\mathcal{A}} z^n e^{jh(l\beta_x + m\beta_y)} \tag{6.23}$$

where β_x and β_y are components of a 2D wavenumber $\boldsymbol{\beta}$ in the x and y directions respectively. Frequency-domain representations of temporal operators shown in Eq. (3.21) do not change in the 2D case. Using

$$p_x = \sin^2(\beta_x h/2) \quad \text{and} \quad p_y = \sin^2(\beta_y h/2) \tag{6.24}$$

for brevity, the following frequency-domain representation of spatial operators can be obtained

$$\delta_{xx} u_{l,m}^n \xRightarrow{\mathcal{A}} -\frac{4}{h^2} p_x u_{l,m}^n \quad \text{and} \quad \delta_{yy} u_{l,m}^n \xRightarrow{\mathcal{A}} -\frac{4}{h^2} p_y u_{l,m}^n, \tag{6.25}$$

from which it follows that

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

$$\delta_\Delta \delta_\Delta u_{l,m}^n \xrightarrow{\mathcal{A}} \frac{16}{h^4}(p_x + p_y)^2 u_{l,m}^n. \tag{6.27}$$

Using these definitions, a frequency-domain interpretation of the 2D wave FD scheme in Eq. (6.9) can be obtained

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

Recalling λ in Eq. (6.11), this can be rewritten to the following characteristic equation

$$z + \left(4\lambda^2(p_x + p_y) - 2\right) + z^{-1} = 0. \tag{6.28}$$

As (after multiplication by z) the characteristic equation is of the form in Eq. (3.24) and $a^{(2)} = 1$, its roots are bounded by condition (3.26)

$$\left|4\lambda^2(p_x + p_y) - 2\right| \le 2.$$

Further derivation yields

$$-2 \le 4\lambda^2(p_x + p_y) - 2 \le 2,$$
$$0 \le 4\lambda^2(p_x + p_y) \le 4,$$

and as middle term is non-negative the first condition is always satisfied, yields

$$\lambda^2(p_x + p_y) \le 1.$$

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

$$2\lambda^2 \le 1,$$
$$\lambda \le \frac{1}{\sqrt{2}} \tag{6.29}$$

which is the stability condition given in Eq. (6.12).

6.2.5 Energy Analysis in 2D

Energy analysis for the 1D case is introduced in Section 3.4. Extensions for the analysis in 2D will be given here.

Analogous to the 1D inner product presented in Section 3.2.1, one can define a 2D inner product. For two functions $f = f(x, y, t)$ and $g(x, y, t)$ defined for a 2D domain $\mathcal{D}$ their inner product over this domain is defined as

$$\langle f, g \rangle_{\mathcal{D}} = \iint_{\mathcal{D}} fg\,dxdy. \tag{6.30}$$

Like in the 1D case, these functions do not have to be a function of time, but they are for coherence.

For two (grid) functions $f_{l,m}^n$ and $g_{l,m}^n$ defined over a discrete domain $d \in \{0, \ldots, N_x\} \times \{0, \ldots, N_y\}$ their discrete inner product is defined as

$$\langle f_{l,m}^n, g_{l,m}^n \rangle_d = \sum_{l=0}^{N_x} \sum_{m=0}^{N_y} h^2 f_{l,m}^n g_{l,m}^n. \tag{6.31}$$

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

$$\underline{d_x} = \{0, \ldots, N_x - 1\} \times \{0, \ldots, N_y\}, \tag{6.32a}$$

$$\overline{d_x} = \{1, \ldots, N_x - 1\} \times \{0, \ldots, N_y\}, \tag{6.32b}$$

$$\underline{d_y} = \{0, \ldots, N_x\} \times \{0, \ldots, N_y - 1\}, \tag{6.32c}$$

$$\overline{d_y} = \{0, \ldots, N_x\} \times \{1, \ldots, N_y - 1\} \tag{6.32d}$$

$$\overline{d} = \{1, \ldots, N_x - 1\} \times \{1, \ldots, N_y - 1\} \tag{6.32e}$$

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. (6.9) can be multiplied by ρH and a 2D inner product (see Eq. (6.31)) with $(\delta_t \cdot u_{l,m}^n)$ over discrete domain d can be taken to yield a definition for $\delta_{t+}\mathfrak{h}$:

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

which can be rewritten to

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

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

Summation by parts as described in Section 3.2.2 can also be applied to δ_{yy} and the following energy balance follows

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

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

where

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \quad \text{with} \quad \mathfrak{t} = \frac{\rho H}{2} \|\delta_{t-} u_{l,m}^n\|_d^2 \quad \text{and}$$

$$\mathfrak{v} = \frac{T}{2} \left(\langle \delta_{x+} u_{l,m}^n, e_{t-} \delta_{x+} u_{l,m}^n \rangle_{\underline{d_x}} + \langle \delta_{y+} u_{l,m}^n, e_{t-} \delta_{y+} u_{l,m}^n \rangle_{\underline{d_y}} \right). \tag{6.33}$$

Here, the reduced domains $\underline{d_x}$ and $\underline{d_y}$ are as defined in (6.32). The boundary term is

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

$$\left. + \langle \delta_t . u_{l,N_y}^n, \delta_{y+} u_{l,N_y}^n \rangle_{(x,N_y)} - \langle \delta_t . u_{l,0}^n, \delta_{y-} u_{l,0}^n \rangle_{(x,0)} \right],$$

and uses 1D inner products at the boundaries. Here, $(l,y) = \{l\} \times \{0, \ldots, N_y\}$ and $(x,m) = \{0, \ldots, N_x\} \times \{m\}$ are slices of domain d. The boundary term can be shown to vanish under Dirichlet boundary conditions in Eq. (6.14a). 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. (6.33) 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 in N/m, one can write the terms in Eq. (6.33) in their units:

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

$$= \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$$

$$\frac{T}{2} \langle \delta_{x+} u_{l,m}^n, e_{t-} \delta_{x+} u_{l,m}^n \rangle_{\underline{d_x}} \xrightarrow{\text{in units}} \text{N} \cdot \text{m}^{-1} \cdot \text{m}^2 \cdot (\text{m}^{-1} \cdot \text{m}) \cdot (\text{m}^{-1} \cdot \text{m})$$

$$= \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}$$

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 Section 6.2.3 and the update equation is written in matrix form as in Eq. (6.22), performing a modal analysis
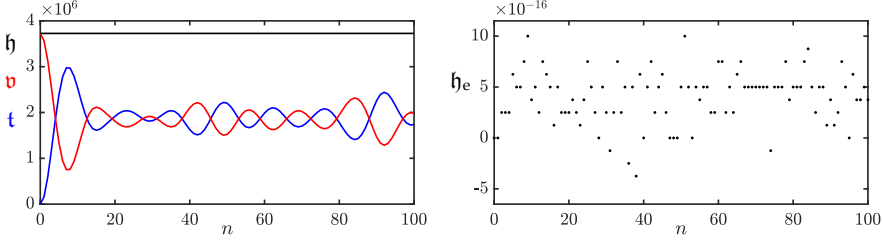
Fig. 6.6: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 2D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.36)) and shows that the deviation of the energy is within machine precision.

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 $\mathbf{U}^n = z^n \boldsymbol{\phi}$ into the matrix form of the 2D wave equation in Eq. (6.22) yields the following characteristic equation

$$\left(z - 2 + z^{-1}\right) \boldsymbol{\phi} = c^2 k^2 \mathbf{D}_\Delta \boldsymbol{\phi}. \tag{6.34}$$

The pth modal frequency can then be obtained by finding the roots of

$$z_p + \left(-2 - c^2 k^2 \text{eig}_p(\mathbf{D}_\Delta)\right) + z_p^{-1} = 0, \tag{6.35}$$

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

$$f_p = \frac{1}{\pi k} \sin^{-1} \left(\frac{ck}{2} \sqrt{-\text{eig}_p(\mathbf{D}_\Delta)}\right). \tag{6.36}$$

Notice the similarity to the equation calculating the modal frequencies of the 1D wave equation in Eq. (3.55). 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.

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.
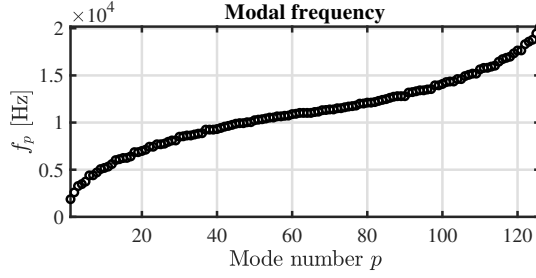
Fig. 6.7: Modal frequencies of the 2D wave equation with $L_x = 1.5$ m, $L_y = 1$ m and $c \approx 3118$ m/s, such that $\lambda = 1/\sqrt{2}$ according to Eq. (6.12).
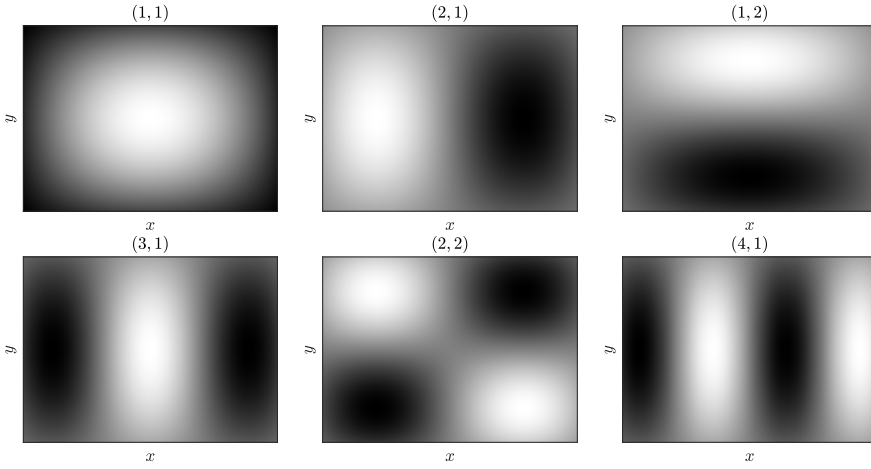

Fig. 6.8: The first (lowest-frequency) six modal shapes of 2D wave equation with $L_x = 1.5$ m and $L_y = 1$ m.

6.3 Thin plate

The thin plate 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 yields extremely interesting sounds.

The plate model is versatile and can be used to model a plate reverb [61] 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 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 $u = u(x, y, t)$. 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, also known as the Kirchhoff model, can be defined as [62]

$$\rho H \partial_t^2 u = -D \Delta \Delta u, \tag{6.37}$$

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. (6.37) only holds for thin 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.

> look up what this actually is

Adding losses to Eq. (6.37) yields

$$\rho H \partial_t^2 u = -D \Delta \Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u \tag{6.38}$$

where, as in the case of the stiff string in Eq. (4.4), σ_0 and σ_1 are the frequency independent (in s^{-1}) and frequency dependent damping coefficient (in m^2/s) respectively.

Boundary conditions

Similar to the stiff string, clamped and simply supported boundary conditions exist where

$$\left. \begin{array}{ll} u = \partial_x u = 0 & \text{if } y = \{0, L_y\} \quad \forall x \\ u = \partial_y u = 0 & \text{if } x = \{0, L_x\} \quad \forall y \end{array} \right\} \quad \text{(Clamped)}, \tag{6.39a}$$

$$\left. \begin{array}{ll} u = \partial_x^2 u = 0 & \text{if } y = \{0, L_y\} \quad \forall x \\ u = \partial_y^2 u = 0 & \text{if } x = \{0, L_x\} \quad \forall y \end{array} \right\} \quad \text{(Simply supported)}. \tag{6.39b}$$

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 [20, Ch. 12].

6.3.2 Discrete time

Equation (6.38) can be discretised to the following FD scheme:

$$\rho H \delta_{tt} u_{l,m}^n = -D \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_{t \cdot} u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-} \delta_\Delta u_{l,m}^n \tag{6.40}$$

where $l \in \{0, \ldots, N_x\}$ and $m \in \{0, \ldots, N_y\}$. Like for the stiff string FD scheme in Eq. (4.7), the backwards difference operator is used for the frequency-dependent damping term to yield an explicit scheme. A more compact way to

write this scheme is after a division by ρH which yields

$$\delta_{tt} u_{l,m}^n = -\kappa^2 \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \delta_{t-} \delta_\Delta u_{l,m}^n \qquad (6.41)$$

with

$$\kappa = \sqrt{\frac{D}{\rho H}} \; . \qquad (6.42)$$

Using the expansion of the discrete biharmonic operator in Eq. (6.6), Eq. (6.41) can be expanded and solved for $u_{l,m}^{n+1}$ according to

$$
\begin{aligned}
u_{l,m}^{n+1} = {} & (2 - 20\mu^2 - 4S) u_{l,m}^n \\
& + (8\mu^2 + S)(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n) \\
& - 2\mu^2 (u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n) \\
& - \mu^2 (u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n), \\
& + (\sigma_0 k - 1 + 4S) u_{l,m}^{n-1} \\
& - S(u_{l+1,m}^{n-1} + u_{l-1,m}^{n-1} + u_{l,m+1}^{n-1} + u_{l,m-1}^{n-1})
\end{aligned}
\qquad (6.43)
$$

where

$$\mu = \frac{\kappa k}{h^2} \qquad (6.44)$$

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

$$h \geq 2\sqrt{k\left(\sigma_1^2 + \sqrt{\kappa^2 + \sigma_1^2}\right)}, \qquad (6.45)$$

and will be derived in Section 6.3.4.

Discrete boundary conditions

The boundary conditions shown in Eq. (6.39) can be discretised to

$$
\left.
\begin{aligned}
u_{l,m}^n &= \delta_{x+} u_{l,m}^n = 0 && \text{if } m = 0 && \forall l \\
u_{l,m}^n &= \delta_{x-} u_{l,m}^n = 0 && \text{if } m = N_y && \forall l \\
u_{l,m}^n &= \delta_{y+} u_{l,m}^n = 0 && \text{if } l = 0 && \forall m \\
u_{l,m}^n &= \delta_{y-} u_{l,m}^n = 0 && \text{if } l = N_x && \forall m
\end{aligned}
\right\} \quad \text{(Clamped)}, \qquad (6.46a)
$$

$$
\left.
\begin{aligned}
u_{l,m}^n &= \delta_{xx} u_{l,m}^n = 0 && \text{if } m = \{0, N_y\} && \forall l \\
u_{l,m}^n &= \delta_{yy} u_{l,m}^n = 0 && \text{if } l = \{0, N_x\} && \forall m
\end{aligned}
\right\} \quad \text{(Simply supported).} \qquad (6.46b)
$$

The clamped condition can be implemented by simply reducing the discrete range of operation to $l = \{2, \ldots, N_x - 2\}$ and $m = \{2, \ldots, N_y - 2\}$. For the

(a) Full stencil.

(b) Stencil of $u_{l,m}^n$.

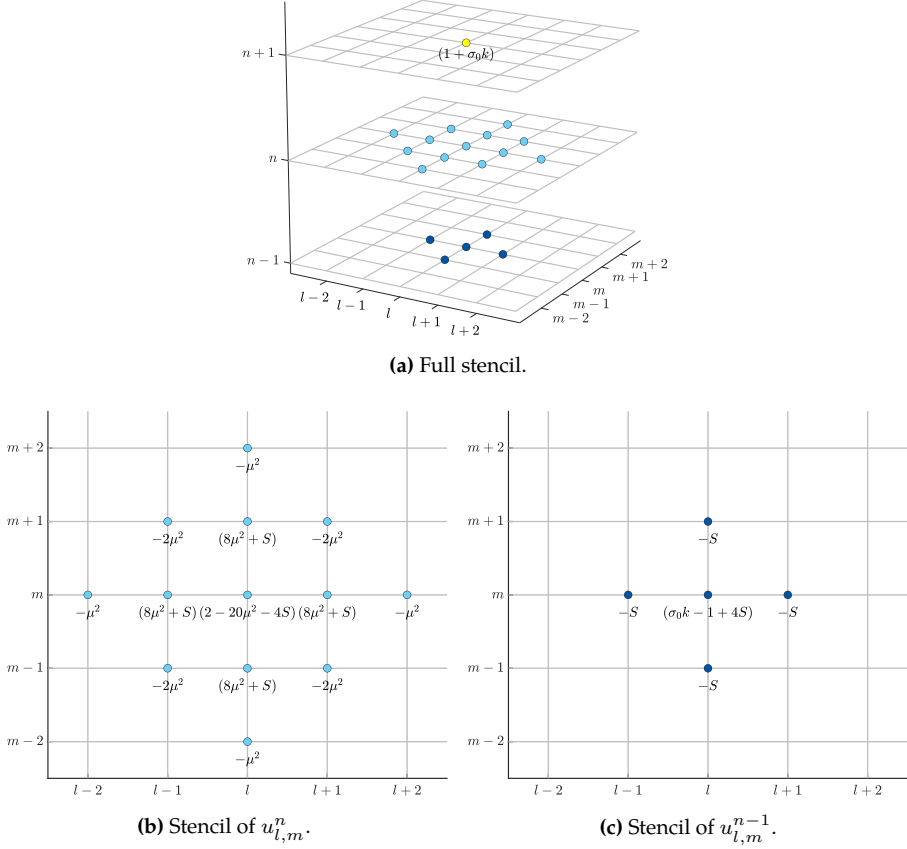**(c)** Stencil of $u_{l,m}^{n-1}$.

Fig. 6.9: The stencil of the plate with coefficients corresponding to those in update equation (6.43). (a) A full overview of the stencil. (b) The current time-step n highlighted. (c) The previous time-step $n-1$ highlighted.

simply supported case, the range of operation reduces to $l = \{1, \ldots, N_x - 1\}$ and $m = \{1, \ldots, N_y - 1\}$, and similar to the simply supported stiff string described in Section 4.2.1, the virtual grid points needed for this condition become

$$u_{-1,m}^n = -u_{1,m}^n \quad \text{and} \quad u_{N_x+1,m}^n = -u_{N_x-1,m}^n \quad \forall m,$$

$$u_{l,-1}^n = -u_{l,-1}^n \quad \text{and} \quad u_{l,N_y+1}^n = -u_{l,N_y-1}^n \quad \forall l.$$

115

6.3.3 Implementation and Output

Similar to the implementation of the 2D wave equation in Section 6.2.3, one can use a stacked state vector. If simply supported boundary conditions are used, one can easily obtain a matrix form of the $\delta_\Delta\delta_\Delta$ operator by multiplying two $\mathbf{D}_\Delta$ matrices presented in Eq. (6.21) to get $\mathbf{D}_{\Delta\Delta} = \mathbf{D}_\Delta\mathbf{D}_\Delta$.

Using a stacked form of the state as described in Eq. (6.17) the scheme in Eq. (6.41) in matrix form is

$$A\mathbf{U}^{n+1} = \mathbf{B}\mathbf{U}^n + \mathbf{C}\mathbf{U}^{n-1}, \tag{6.47}$$

where

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

and the identity matrix $\mathbf{I}$ is of the same size as $\mathbf{D}_{\Delta\Delta}$ and $\mathbf{D}_\Delta$.

As a starting point for implementation, possible parameters are given in Table 6.1.

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.

Figure 6.10 shows the wave propagation of a thin plate excited with a 2D raised cosine at $(x, y) = (0.25L_x, 0.5L_y)$ and uses the parameters given in Table 6.1. When compared to Figure 6.4, dispersive effects – where higher-frequency components travel faster than lower-frequency ones – due to stiffness are apparent. Figure 6.11 shows the time-domain and frequency domain output of the thin plate at $(x, y) = (0.15L_x, 0.85L_y)$

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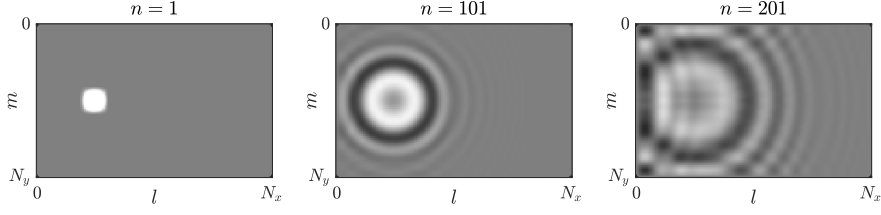
Fig. 6.10: Wave propagation of a thin plate with simply supported boundary conditions and parameters as shown in Table 6.1. The system is excited with a 2D raised cosine at $(x, y) = (0.25L_x, 0.5L_y)$ and dispersive effects are apparent.
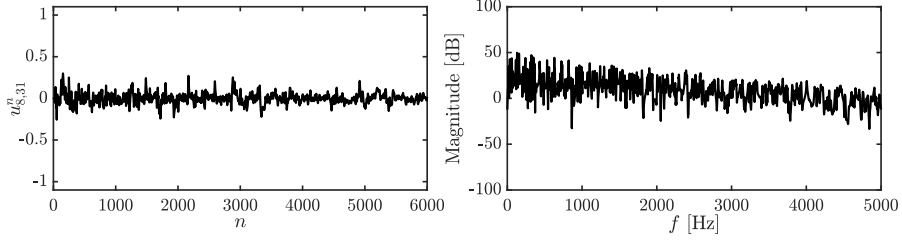

Fig. 6.11: The output of the thin plate at $(x, y) = (0.15, 0.85)$ corresponding to Figure 6.10. Compared to the output of the 2D wave equation in Figure 6.5, the amplitude is much lower and waves are closer together due to the dispersion.

Using Eqs. (6.26) and (6.27) one can obtain a frequency-domain representation of the FD scheme in Eq. (6.41) and obtain the following characteristic equation

$$(1 + \sigma_0 k)z + \left(16\mu^2 (p_x + p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x + p_y) - 2 \right)$$
$$+ \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x + p_y) \right) z^{-1} = 0. \tag{6.48}$$

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. (6.24), and given the fact that these are bounded by 1, the following can be written

$$4\mu^2 (1 + 1)^2 + \frac{4\sigma_1 k}{h^2}(1 + 1) \leq 1$$
$$16\mu^2 + \frac{8\sigma_1 k}{h^2} \leq 1.$$

Finally, recalling the definition for κ in Eq. (6.42) solving for h then yields

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

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

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

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

$$h \geq 2\sqrt{k\left(\sigma_1 + \sqrt{\sigma_1^2 + \kappa^2}\right)}, \tag{6.49}$$

which is the stability condition given in Eq. (6.45).

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. (6.41).

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

To obtain the rate of change of energy, one can take an inner product of the scheme in Eq. (6.41) with $(\delta_t . u_{l,m}^n)$ over discrete (2D) domain d to get

$$\delta_{t+}\mathfrak{h} = \rho H \langle \delta_t . u_{l,m}^n, \delta_{tt} u_{l,m}^n \rangle_d + D \langle \delta_t . u_{l,m}^n, \delta_\Delta \delta_\Delta u_{l,m}^n \rangle_d$$
$$+ 2\sigma_0 \rho H \langle \delta_t . u_{l,m}^n, \delta_t . u_{l,m}^n \rangle_d - 2\sigma_1 \rho H \langle \delta_t . u_{l,m}^n, \delta_{t-} \delta_\Delta u_{l,m}^n \rangle_d = 0. \tag{6.50}$$

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 damping term

$$\mathfrak{q} = 2\sigma_0 \rho A \|\delta_t . u_{l,m}^n\|_d^2 - 2\sigma_1 \rho A \langle \delta_t . u_{l,m}^n, \delta_{t-} \delta_\Delta u_{l,m}^n \rangle_d. \tag{6.51}$$

Expanding the stiffness term in Eq. (6.50) to

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

one can perform summation by parts twice using Eq. (3.15b) for both terms to get

$$D\left(\langle\delta_{t\cdot}\delta_{xx}u^n_{l,m},\delta_\Delta u^n_{l,m}\rangle_{\overline{\underline{d_x}}}+\langle\delta_{t\cdot}\delta_{yy}u^n_{l,m},\delta_\Delta u^n_{l,m}\rangle_{\overline{\underline{d_y}}}\right)+\mathfrak{b}.$$

The definitions for the reduced domains can be found in Eqs. (6.32). Finally, as the boundaries are always 0 due to the boundary conditions in Eq. (6.46), $\overline{\underline{d_x}}$ and $\overline{\underline{d_y}}$ can be further reduced to $\underline{\overline{d}}$ and the terms can be combined as check with stefan

$$D\langle\delta_{t\cdot}\delta_\Delta u^n_{l,m},\delta_\Delta u^n_{l,m}\rangle_{\underline{\overline{d}}}+\mathfrak{b},$$

and using identities (3.16a) and (3.16b) a definition for the total energy can be found:

$$\mathfrak{h}=\mathfrak{t}+\mathfrak{v},\quad\text{with}\quad\mathfrak{t}=\frac{\rho H}{2}\left\|\delta_{t-}u^n_{l,m}\right\|^2_d\quad\text{and}$$
$$\mathfrak{v}=\frac{D}{2}\langle\delta_\Delta u^n_{l,m},e_{t-}\delta_\Delta u^n_{l,m}\rangle_{\underline{\overline{d}}}\,. \tag{6.52}$$

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. (6.46) [20].

Step 3: Check units

As $\mathfrak{t}$ is identical to its definition in Eq. (6.33), only the units for $\mathfrak{v}$ will be checked here. Recalling that $D=EH^3/12(1-\nu^2)$, which in units is kg· m^2·s^{-2}, yields

$$\frac{D}{2}\langle\delta_\Delta u^n_{l,m},e_{t-}\delta_\Delta u^n_{l,m}\rangle_{\underline{\overline{d}}}\xrightarrow{\text{in units}}\text{kg}\cdot\text{m}^2\cdot\text{s}^{-2}\cdot\text{m}^2\cdot(\text{m}^{-2}\cdot\text{m})\cdot(\text{m}^{-2}\cdot\text{m})$$
$$=\text{kg}\cdot\text{m}^2\cdot\text{s}^{-2}$$

and shows that $\mathfrak{v}$ indeed has the correct units.

Step 4: Implementation

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

6.3.6 Modal Analysis

Using the matrix form in Eq. (6.47), a modal analysis of the system can be performed using a one-step form described in Section 3.57.

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 6.36, the pattern is slightly more exponential like the stiff string in Figure 4.6.not really sure what to say here honestly
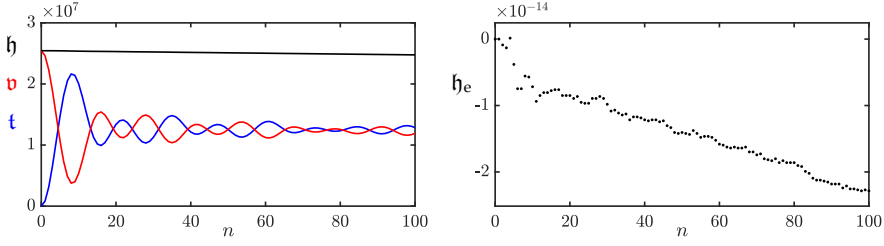
Fig. 6.12: The kinetic (blue), potential (red), and total (black) energy of an implementation of the thin plate are plotted in the left panel. Notice that the damping present in the system causes ħ to decrease. The right panel shows the normalised energy (according to Eq. (3.37)) and shows that the deviation of the energy is within machine precision.
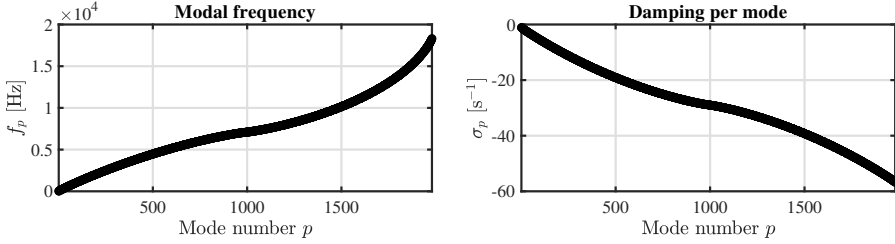

Fig. 6.13: The result of a modal analysis of the thin plate using the parameters in Table 6.1. Notice that the damping is plotted against modal frequency rather than mode number.

6.4 Stiff membrane

The term stiff membrane appears in [19] and is essentially a 2D version of the stiff string. It can be used to model membranes with dispersive effects or provide tension control for thin platex. 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 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. (6.7) and the thin plate in Eq. (6.37). Adding losses as in Eq. (6.38) yields the following equation of motion

$$\rho H \partial_t^2 u = T\Delta u - D\Delta\Delta u - 2\sigma_0 \rho H \partial_t u + 2\sigma_1 \rho H \partial_t \Delta u, \qquad (6.53)$$

where the parameters are identical to those in Eqs. (6.7) and (6.38).

6.4.2 Discrete time

Using familiar operators, the (6.53) can be discretised to

$$\rho H \delta_{tt} u_{l,m}^n = T \delta_\Delta u_{l,m}^n - D \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \rho H \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \rho H \delta_{t-} \delta_\Delta u_{l,m}^n, \quad (6.54)$$

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

$$\delta_{tt} u_{l,m}^n = c^2 \delta_\Delta u_{l,m}^n - \kappa^2 \delta_\Delta \delta_\Delta u_{l,m}^n - 2\sigma_0 \delta_{t\cdot} u_{l,m}^n + 2\sigma_1 \delta_{t-} \delta_\Delta u_{l,m}^n, \quad (6.55)$$

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 (6.6) respectively to get

$$\begin{aligned} u_{l,m}^{n+1} = {}& (2 - 4\lambda^2 - 20\mu^2 - 4S)u_{l,m}^n \\ & + (\lambda^2 + 8\mu^2 + S)(u_{l+1,m}^n + u_{l-1,m}^n + u_{l,m+1}^n + u_{l,m-1}^n) \\ & - 2\mu^2(u_{l+1,m+1}^n + u_{l-1,m+1}^n + u_{l+1,m-1}^n + u_{l-1,m-1}^n) \\ & - \mu^2(u_{l+2,m}^n + u_{l-2,m}^n + u_{l,m+2}^n + u_{l,m-2}^n), \\ & + (\sigma_0 k - 1 + 4S)u_{l,m}^{n-1} \\ & - S(u_{l+1,m}^{n-1} + u_{l-1,m}^{n-1} + u_{l,m+1}^{n-1} + u_{l,m-1}^{n-1}) \end{aligned} \quad (6.56)$$

where

$$\lambda = \frac{ck}{h} \quad \text{and} \quad \mu = \frac{\kappa k}{h^2} \quad (6.57)$$

and again, $S = 2\sigma_1 k/h^2$ for compactness. The stability condition for this scheme will be shown in Section 6.4.4.

6.4.3 Implementation

Writing Eq. (6.55) in matrix form yields

$$A\mathbf{U}^{n+1} = \mathbf{B}\mathbf{U}^n + \mathbf{C}\mathbf{U}^{n-1}, \quad (6.58)$$

with

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

where the only difference with Eq. (6.47) 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. (6.55) can be obtained:

$$
(1 + \sigma_0 k)z + \left(4\lambda^2(p_x + p_y) + 16\mu^2(p_x + p_y)^2 + \frac{8\sigma_1 k}{h^2}(p_x + p_y) - 2 \right)
$$
$$
+ \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2}(p_x + p_y) \right) z^{-1} = 0.
\tag{6.59}
$$

Similar to the stiff string in Section 4.3 and the thin plate in Section 6.3.4, this can be solved to

$$
\lambda^2(p_x + p_y) + 4\mu^2(p_x + p_y)^2 + \frac{4\sigma_1 k}{h^2}(p_x + p_y) \le 1,
$$

and recalling that p_x and p_y are bounded by 1 yields

$$
\lambda^2(1 + 1) + 4\mu^2(1 + 1)^2 + \frac{4\sigma_1 k}{h^2}(1 + 1) \le 1,
$$
$$
2\lambda^2 + 16\mu^2 + \frac{8\sigma_1 k}{h^2} \le 1.
$$

Recalling the definitions for λ and μ from 6.57, one can solve for h

$$
\frac{2c^2 k^2}{h^2} + \frac{16\kappa^2 k^2}{h^4} + \frac{8\sigma_1 k}{h^2} \le 1,
$$
$$
h^4 - (2c^2 k^2 + 8\sigma_1 k)h^2 - 16\kappa^2 k^2 \ge 0,
$$
$$
h \ge \sqrt{\frac{2c^2 k^2 + 8\sigma_1 k + \sqrt{(2c^2 k^2 + 8\sigma_1 k)^2 + 64\kappa^2 k^2}}{2}},
$$
$$
h \ge \sqrt{c^2 k^2 + 4\sigma_1 k + \frac{1}{2}\sqrt{4(c^2 k^2 + 4\sigma_1 k)^2 + 64\kappa^2 k^2}},
$$
$$
h \ge \sqrt{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}},
\tag{6.60}
$$

and is the stability condition for the stiff membrane.

6.5 Radial Coordinates

This chapter presented various models using a Cartesian coordinate system. Circular or elliptical systems, such as membranes or gongs, could be modelled using a radial coordinate system [20, Ch. 10]. However, using explicit methods to discretise the systems cause the schemes to exhibit high amount much numerical dispersion and reduction of bandwidth [20, Ch. 11]. For better behaviour, one could resort to an implicit scheme, but comes with the drawbacks

mentioned in Section 4.6. A better alternative is to retain the cartesian coordinate system and set the boundary according to a staircase approximation as done in [S4] (see fx. [44, 56]).

check whether
all references
are used

References

[1] C. Roads, *The Computer Music Tutorial*. MIT Press, Cambridge, Massachusetts, 1996.

[2] 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.

[3] 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.

[4] J. O. Smith, "Virtual acoustic musical instruments: Review and update," *Center for computer research in music and acoustics (CCRMA)*, 2010.

[5] J. Kelly and C. Lochbaum., "Speech synthesis," in *Proceedings of the Fourth International Congress on Acoustics*, 1962, pp. 1–4.

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

[7] L. Hiller and P. Ruiz, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part I," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 6, pp. 462–470, 1971.

[8] ——, "Synthesizing musical sounds by solving the wave equation for vibrating objects: Part II," *Journal of the Audio Engineering Society (JASA)*, vol. 19, no. 7, pp. 542–550, 1971.

[9] 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.

[10] 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.

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

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

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

[14] ——, "Physical audio signal processing (online book)," 2010. [Online]. Available: http://ccrma.stanford.edu/~jos/pasp/

[15] 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.

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

[17] 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.

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

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

[20] S. Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley & Sons, 2009.

[21] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Institute of Physics Publishing*, 2006.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

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

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

[34] S. Yantis and R. A. Abrams, *Sensation and Perception*, 2nd ed.   Worth Publishers, 2016.

[35] 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.

[36] 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.

[37] ——, "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.

[38] 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.

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

[40] 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.

[41] 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.

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

[43] 3D Systems, Inc, "3D Systems Touch Haptic Device," 2021. [Online]. Available: https://www.3dsystems.com/haptics-devices/touch

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

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

[46] 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.

[47] 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.

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

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

[50] 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.

References

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

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

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

[54] 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.

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

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

[57] 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.

[58] 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.

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

[60] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[61] 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.

[62] P. Morse and U. Ingard., *Theoretical Acoustics*. Princeton University Press, 1968.

check whether to sort references or not

References

Part III

Papers

Paper A

Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Silvin Willemsen, Nikolaj Andersson, Stefania Serafin
and Stefan Bilbao

Paper B

Physical Models and Real-Time Control with the
Sensel Morph

Silvin Willemsen, Stefan Bilbao, Nikolaj Andersson
and Stefania Serafin

Paper C

Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao and Stefania Serafin

Paper D

Real-time Implementation of a Physical Model of the Tromba Marina

Silvin Willemsen, Stefania Serafin, Stefan Bilbao and Michele Ducceschi

Paper E

Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling

Silvin Willemsen, Razvan Paisa and Stefania Serafin

Paper F

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study

Silvin Willemsen, Anca-Simona Horvath and Mauro Nascimben

Paper G

Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

Paper H

A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes

Silvin Willemsen, Stefan Bilbao, Michele Ducceschi and Stefania Serafin

Paper H.

Part IV

Appendix

Appendix A

Paper Errata

Here, some errors in the published papers will be listed:

Elasto-plastic [C]

- The authors in reference [15] are a bit shuffled.

Real-Time Tromba [D]

- The minus sign in Eq. (28) (and thus Eqs. (31) and (35)) should be a plus sign.

- $\sigma_{1,\mathrm{s}}$ in Eq. (21) should obviously be $\sigma_{1,\mathrm{p}}$

- the unit of the spatial Dirac delta function δ should be m^{-1}

DigiDrum [F]

- σ_0 and σ_1 should be multiplied by ρH in order for the stability condition to hold.

- stability condition is wrong. Should be:

$$h \geq \sqrt{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}} \qquad \text{(A.1)}$$

- 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 dissertation.

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,* and is indicated with the T operator. Transposing a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^T$. This means that the elements in the ith row and the jth column of the original matrix become the elements in the jth row and the ith column of the transposed matrix. Essentially the row and column indices of the elements inside the matrix get switched according to

$$a_{ij} = a_{ji}. \tag{B.1}$$

Also see Figure B.1. For a row vector, the transpose operation simply changes it to a column vector and vice versa. Another way of seeing a transpose is that all the elements get flipped over the *main diagonal* of the matrix. The main diagonal comprises the elements a_{ij} where $i = j$ and a transpose does not affect the location of these elements.

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

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

Fig. B.1: A matrix $\mathbf{A}$ and its transpose $\mathbf{A}^T$. The elements get flipped along the main diagonal of the matrix according to Eq. (B.1).

Matrix Multiplication

Matrix multiplication (this includes matrix-vector multiplication) is different from regular multiplication in that it needs to abide several extra rules. The multiplication of two matrices $\mathbf{A}$ and $\mathbf{B}$ to a resulting matrix $\mathbf{C}$ is defined as

$$c_{ij} = \sum_{k=1}^{K} a_{ik} b_{kj}, \tag{B.2}$$

where K is both the number columns of matrix $\mathbf{A}$ and the number of rows in matrix $\mathbf{B}$. It thus follows that, order for matrix multiplication to be valid, the number of columns of the first matrix needs to be equal to the number of rows in the second matrix. The result will then be a matrix with a number of rows equal to that of the first matrix and a number of columns equal to that of the second matrix. See Figure B.2 for reference.

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

$$\overbrace{(L \times M)}^{\mathbf{A}} \cdot \overbrace{(M \times N)}^{\mathbf{B}} = \overbrace{(L \times N)}^{\mathbf{C}}. \tag{B.3}$$
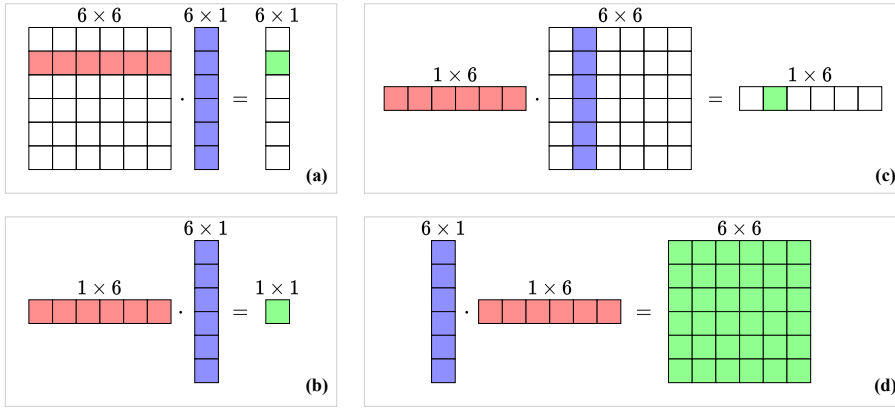

Fig. B.2: Visualisation of valid matrix multiplications (see Eq. (B.2)). The "inner" dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of "outer" dimensions (rows of the left matrix and columns of the right).

B.2 Matrix Inverse

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

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \tag{B.4}$$

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

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

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

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

B.3   Systems of Linear Equations

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

For example, take the system of linear equations

$$x + z = 6,$$
$$z - 3y = 7,$$
$$2x + y + 3z = 15,$$

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

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \tag{B.5}$$

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:

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

We can then solve for $\mathbf{u}$ by taking the inverse of $\mathbf{A}$ (see Section B.2) and multiplying this onto $\mathbf{w}$

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \tag{B.6}$$

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

Solving a system of linear equations can be implemented in MATLAB by using the code given in Section B.2 and multiplying this onto a vector `w`

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

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

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

B.4 Eigenvalue Problems

A square matrix $\mathbf{A}$ 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

$$\mathbf{A}\phi = \lambda\phi. \tag{B.7}$$

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

$$\lambda_p = \text{eig}_p(\mathbf{A}), \tag{B.8}$$

where $\text{eig}_p(\cdot)$ denotes the pth 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 pth eigenvector appears in the pth 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)

```matlab
%% 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)

```matlab
%% 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);
```

```
51      end
52
53      % Eq. (2.50)
54      if bcRight == "N"
55          uNext(N+1) = (2 - 2 * lambdaSq) * u(N+1) - uPrev(N+1) ...
56              + 2 * lambdaSq * u(N);
57      end
58
59      out(n) = u(outLoc);
60
61      % Update system states
62      uPrev = u;
63      u = uNext;
64  end
```

C.3 2D Wave Equation (Section 6.2)

```
1
2   %% Initialise variables
3   fs = 44100;            % Sample rate [Hz]
4   k = 1 / fs;            % Time step [s]
5   lengthSound = fs;      % Length of the simulation (1 second) [samples]
6
7   rho = 7850;                    % Material density [kg/m^3]
8   H =  0.0005;                   % Thickness [m]
9   T = 1000000;                   % Tension per unit length [N/m]
10  c = sqrt(T / (rho * H));       % Wave speed [m/s]
11
12  Lx = 1;                        % Length in x direction [m]
13  Ly = 2;                        % Length in y direction [m]
14
15  h = sqrt(2) * c * k;           % Grid spacing [m]
16  Nx = floor(Lx/h);              % Number of intervals in x direction
17  Ny = floor(Ly/h);              % Number of intervals in y direction
18  h = min(Lx/Nx, Ly/Ny);         % Recalculation of grid spacing
19
20  lambdaSq = c^2 * k^2 / h^2;    % Courant number squared
21  h = max(Lx/Nx, Ly/Ny);         % Recalculation of grid spacing
22
23  %% Create scheme matrices with Dirichlet boundary conditions
24  Nxu = Nx - 1;
25  Nyu = Ny - 1;
26  Dxx = toeplitz([-2, 1, zeros(1, Nxu-2)]);
27  Dyy = toeplitz([-2, 1, zeros(1, Nyu-2)]);
28
29  % Kronecker sum
30  D = kron(speye(Nxu), Dyy) + kron(Dxx, speye(Nyu));
31  D = D / h^2;
32
```

```matlab
33 % Total amount of grid points
34 Nu = Nxu * Nyu;
35
36 %% Initialise state vectors (one more grid point than the number of
        intervals)
37 uNext = zeros(Nu, 1);
38 u = zeros(Nu, 1);
39
40 %% Initial conditions (2D raised cosine)
41 halfWidth = floor(min(Nx, Ny) / 5);
42 width = 2 * halfWidth + 1;
43 xLoc = floor(0.3 * Nx);
44 yLoc = floor(0.6 * Ny);
45 xRange = xLoc-halfWidth : xLoc+halfWidth;
46 yRange = yLoc-halfWidth : yLoc+halfWidth;
47
48 rcMat = zeros(Nyu, Nxu);
49 rcMat(yRange, xRange) = hann(width) * hann(width)';
50
51 % initialise current state
52 u = reshape(rcMat, Nu, 1);
53
54 % Set initial velocity to zero
55 uPrev = u;
56
57 % Output location
58 xOut = 0.45;
59 yOut = 0.25;
60 outLoc = round((xOut + yOut * Nyu) * Nxu);
61 out = zeros(lengthSound, 1);
62
63 %% Simulation loop
64 for n = 1:lengthSound
65
66     %% Update equation Eq. (6.22)
67     uNext = (2 * eye(Nu) + c^2 * k^2 * D) * u - uPrev;
68
69     % Update system states
70     uPrev = u;
71     u = uNext;
72
73 end
```

162

Appendix D

Intuition for the Damping Terms in the Stiff String PDE

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

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa^2\partial_x^4 u - 2\sigma_0\partial_t u + 2\sigma_1\partial_t\partial_x^2 u \tag{D.1}$$

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

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

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

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

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

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

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

$$\delta_{t-}\delta_{xx} f(x,t) = \frac{1}{kh^2} \Big(\ f(-1,2) - 2f(0,2) + f(1,2)$$

$$- \Big(f(-1,1) - 2f(0,1) + f(1,1) \Big) \Big),$$

$$= \frac{1}{1 \cdot 1^2} \Big(2 \cdot (-1)^2 - 2 \cdot 2 \cdot (0)^2 + 2 \cdot 1^2$$

$$- \Big(1 \cdot (-1)^2 - 2 \cdot 1 \cdot (0) + 1 \cdot (1^2) \Big) \Big),$$

$$= 2 + 2 - (1 + 1) = 2.$$

So the rate of change of the curvature is positive, i.e., the already positively curved function x^2 gets more curved over time.

If the curvature around a point along a string gets more positive (or less negative) over time, the force applied to that point will be positive, effectively 'trying' to reduce the curvature. Vice versa, if the curvature around a point along a string gets more negative (or less positive) over time, the force applied will be negative, again 'trying' to reduce the curvature.

From an auditory point of view, higher curvature generally means higher frequency. As the frequency-dependent damping term reduces curvature along the string it effectively damps higher frequencies.

Digital versions of musical instruments have been created for several decades, and for good reason! They are more compact, more easy to maintain, and less difficult to play than their real-life counterparts. One way to digitise an instrument is to record it and play back the samples, but this does not capture the entire range of expression of the real instrument. Simulating an instrument based on its physics, including its geometry and material properties, is much more flexible to player control. Although it requires more computational power to generate the sound in real time, the simulation could possibly go beyond what is physically possible. A violin growing into a cello, bowing your trumpet, your imagination is the limit...