
The Emulated Ensemble

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

Ph.D. Dissertation
Silvin Willemsen

Dissertation submitted June 11, 2021

Thesis submitted: June 11, 2021

PhD Supervisor: Prof. Stefania Serafin
Aalborg University

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

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

Department: Department of Architecture, Design and Media Technology

ISSN: xxxx-xxxx

ISBN: xxx-xx-xxxx-xxx-x

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

© Copyright: Silvin Willemsen

Printed in Denmark by Rosendahls, 2021

Curriculum Vitae

Silvin Willemsen



Here is the CV text.

Curriculum Vitae

Acknowledgements

I would like to thank my mom..

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

Acknowledgements

List of Publications

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

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] —, “A physical model of the trombone using dynamic grids for finite-difference schemes,” in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21)*, 2021.

Publications with a Supervisory Role

- [S1] R. S. Alecu, S. Serafin, S. Willemsen, E. Parravicini, and S. Lucato, “Embouchure interaction model for brass instruments,” in *Proceedings of the 17th Sound and Music Computing (SMC) Conference*, 2020, pp. 153–160.
- [S2] T. Lasickas, S. Willemsen, and S. Serafin, “Real-time implementation of the shamisen using finite difference schemes,” in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021.
- [S3] M. G. Onofrei, S. Willemsen, and S. Serafin, “Implementing complex physical models in real-time using partitioned convolution: An adjustable spring reverb,” in *Proceedings of the 18th Sound and Music Computing (SMC) Conference*, 2021.
- [S4] —, “Real-time implementation of an elasto-plastic friction drum using finite difference schemes,” in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020in21)*, 2021.

Miscellaneous Publications

- [M1] 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.
- [M2] 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.
- [M3] M. Ducceschi, S. Bilbao, S. Willemsen, and S. Serafin, “Linearly-implicit schemes for collisions in musical acoustics based on energy quadratisation,” *Journal of the Acoustical Society of America (JASA)*, vol. 149, pp. 3502–3516, 2021.

Abstract

English abstract

Abstract

Resumé

Danish Abstract

Resumé

Contents

Curriculum Vitae	iii
Acknowledgements	v
List of Publications	vii
Abstract	ix
Resumé	xi
Contents	xiii
Preface	xxi
I Introduction	1
1 Physical Modelling of Musical Instruments	3
1.1 Physical Modelling Techniques	3
1.2 Real-Time Implementation	6
1.3 Why?	6
1.3.1 Samples vs. Physical Modelling	6
1.3.2 Resurrect Old or Rare Instruments	7
1.3.3 Go beyond what is physically possible	7
1.4 Project Objectives and Main Contributions	7
1.5 Thesis Outline	7
2 Introduction to FDTD Methods	9
2.1 Differential Equations	9
2.1.1 Dimensions and Degrees of Freedom	10
2.1.2 Ranges of Definition and Domains	12
2.2 Discretisation using FDTD methods	12

exactly as in
[3]

FULL DOC
SWEEP: check
capitalisation
of headings
throughout
document

FULL DOC
SWEEP: check
hyphen in ti-
tles

2.2.1	Grid Functions	13
2.2.2	Finite-Difference Operators	13
2.2.3	Identities	19
2.3	The Mass-Spring System	19
2.3.1	Continuous-time	20
2.3.2	Discrete-time	21
2.3.3	Implementation and Output	22
2.4	The 1D Wave Equation	23
2.4.1	Continuous time	23
2.4.2	Discrete time	25
2.4.3	Implementation: Excitation and Output	28
2.4.4	Stability and Simulation Quality	30
3	Analysis Techniques	33
3.1	Matrices	33
3.1.1	Operations	34
3.1.2	In a FDTD context	36
3.1.3	Matrix Inverse	38
3.1.4	Systems of Linear Equations	39
3.1.5	Eigenvalue Problems	40
3.2	Mathematical Tools and Product Identities	40
3.2.1	Inner product	40
3.2.2	Summation by Parts	41
3.2.3	Product identities	44
3.3	Frequency Domain Analysis	44
3.3.1	Mass-Spring System	47
3.3.2	1D Wave Equation	48
3.4	Energy Analysis	49
3.4.1	Energy Analysis: A 4-Step Tutorial	49
3.4.2	Mass-spring system	51
3.4.3	1D Wave Equation	53
3.4.4	Stability Analysis using Energy Analysis Techniques	57
3.5	Modal Analysis	59
3.5.1	One-Step Form	60
3.6	Dispersion analysis	61
II	Resonators	63
4	Stiff string	67
4.1	Continuous time	67
4.1.1	Adding Losses	68
4.2	Discrete Time	69

Contents

4.2.1	Boundary conditions	71
4.2.2	Implementation and Matrix Form	73
4.2.3	Parameters and output	75
4.3	von Neumann Analysis and Stability Condition	75
4.4	Energy Analysis	77
4.5	Modal Analysis	79
4.6	Implicit Scheme	80
4.6.1	von Neumann analysis	81
4.6.2	Modal analysis	81
5	Brass	83
5.1	Second-order system	83
5.2	First-order system	83
6	2D Systems	85
6.1	Analysis Techniques in 2D	85
6.1.1	Frequency Domain Analysis	85
6.1.2	Energy Analysis	86
6.1.3	Modal Analysis	86
6.2	2D Wave Equation	86
6.3	Thin plate	86
6.4	Stiff membrane	86
III	Exciters	87
7	Unmodelled Excitations	91
7.1	Initial conditions	91
7.2	Signals	91
7.2.1	Pulse train	91
7.2.2	Noise	91
8	Modelled Excitations	93
8.1	Preamble: Newton-Raphson	93
8.2	Hammer but maybe not as I didn't use it.	93
8.2.1	Mass-spring Systems Revisited: Adding Damping	93
8.3	The Bow	93
8.3.1	Static Friction Models	94
8.3.2	Dynamic Friction Models	95
8.4	Lip-reed	95

IV Interactions	97
9 Connections	101
9.1 Rigid connection	102
9.2 Spring-like connections	102
9.2.1 Connection with rigid barrier (scaled)	102
9.2.2 String-plate connection	103
9.2.3 Solving for f	104
9.2.4 Non-dimensional	104
10 Collisions	105
10.1 Classic models	105
10.2 Michele's tricks	105
V Dynamic Grid	107
11 Dynamic Grid	109
11.1 Background and Motivation	109
11.2 Method	110
11.2.1 Full-Grid Interpolation	111
11.2.2 Adding and removing Points at the Boundary	111
11.2.3 Cubic interpolation	113
11.2.4 Sinc interpolation	113
11.2.5 Displacement correction	113
11.3 Analysis and Experiments	114
11.3.1 Interpolation technique	114
11.3.2 Interpolation range	114
11.3.3 Location	114
11.4 Discussion and Conclusion	114
VI Real-Time Implementation and Control	115
12 Real-Time Implementation	119
12.1 MATLAB vs. C++	119
12.1.1 Speed	120
12.1.2 Syntax	120
12.2 Do's and don'ts in Real-Time FD schemes	120
12.3 Graphics	123
12.4 Matrices	125
12.4.1 Matrix Inversions in Real-Time	125

13 Control	127
13.1 Sensel Morph	127
13.2 Phantom OMNI	127
 VII Complete Instruments	 129
14 Large Scale Modular Physical models	133
14.1 Bowed Sitar	133
14.2 Dulcimer	133
14.3 Hurdy Gurdy	134
15 Tromba Marina	135
15.1 Introduction	135
15.2 Physical Model	135
15.2.1 Continuous	135
15.2.2 Discrete	135
15.3 Real-Time Implementation	135
15.3.1 Control using Sensel Morph	135
15.3.2 VR Application	135
16 Trombone	137
16.1 Introduction	137
16.2 Physical Model	137
16.2.1 Continuous	137
16.2.2 Discrete	138
16.3 Real-Time Implementation	138
16.4 Discussion	138
 VIII Conclusions and Perspectives	 139
17 Conclusions and Perspectives	141
References	147
 IX Appendix	 149
A List of Symbols	151
B List of Abbreviations	155

C	Code Snippets	157
C.1	Mass-Spring System (Section 2.3)	157
C.2	1D Wave Equation (Section 2.4)	158
X	Papers	161
A	Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph	165
A.1	Introduction	166
A.2	Conclusion	166
B	Physical Models and Real-Time Control with the Sensel Morph	167
C	Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite Difference Schemes	169
D	Real-time Implementation of a Physical Model of the Tromba Marina	171
E	Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling	173
F	DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study	175
G	Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations	177
H	A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes	179

Todo list

exactly as in [3]	xiii
FULL DOC SWEEP: check capitalisation of headings throughout document	xiv
FULL DOC SWEEP: check hyphen in titles	xiv
should I include this	5
exactly as in [3]	6
Put this work into perspective of the literature (higher level)	7
check whether this needs to be per point along a system	11
Figure and caption are not done yet	12
FULL DOC SWEEP: check capitalisation of headings throughout document	13
grid figure	13
this might be unnecessary, but I thought that it might be nice to have this in an equation for clarity	13
many figures for shift and FD operators	14
FULL DOC SWEEP: check centred instead of centered	14
these spacings are different in overleaf...	14
figure here visualising operators (with reference to grid figure)	14
in energy analysis, interleaved grids, etc.	15
see whether the negative version of identity (2.27c) is also used later on	19
figure out if this is valid as it falls outside of the defined temporal region of operation	22
maybe a bit short...	22
FULL DOC SWEEP: check hyphen in titles	23
unit?	23
different wording in caption	24
check whether still correct	25
FULL DOC SWEEP: check straightforward or straight-forward	25
more intuition on stability and gridspacing here	32
perhaps also dispersion analysis?	33
is this how you explain it?	37
ask stefan	40

Contents

check whether this is necessary	40
check if I should not refer to a subsection	45
check reference	45
check	45
FULL DOC SWEEP: nonlinear, non-linear or non linear	46
not talking about nonlinear systems though	46
only the denominator of the transfer function	46
shouldn't I just cite the first edition then?	49
check if the sum should indeed go until $n - 1$ and why	51
check	57
more explanation, perhaps refer to von neumann analysis in 3.3	59
not done	67
should I even include the lossless one? It's just so that we can slowly build up to the damped model...	67
citations here?	68
etc.	68
check wavespeed or wave speed (entire document)	68
insert figure showing virtual grid points somewhere in this section .	71
time-domain and fft	75
different wording	76
make figure	79
not done...	79
below is the exact same wording as above	81
check whether this is right..	86
Different title here?	91
check references here	94
FULL DOC SWEEP: check figure centering	94
check if is still true	109
These sections are taken from the JASA appendix	110
check whether all references are used	142
changed the template here, should check if it's ok like this	165

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."

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.

Some basic calculus knowledge is assumed.

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

Make physical modelling more accessible to the non-physicist.

Interested in physically impossible manipulations of now-virtual instruments.

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

Part I: Introduction introduces the field of physical modelling for musical instruments in Chapter 1 by giving a brief history of the field and provides and background for the project. At the end, the project objectives and contributions to the field will be 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.

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

Part III: Exciters shows two different ways that the

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

Silvin Willemsen
Aalborg University, June 11, 2021

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

The interest in digitising musical instruments

This is for various reasons...

Physical modelling is a technique to synthesise sound. According to many, physical modelling is the best way to realistically and naturally simulate real-world musical instruments [46, 42, 3]. As this technique simulates the instrument based on its physics rather than using pre-recorded samples, it is more flexible to player-interaction and thus more realistic when synthesising sound in performance. Although physical models could potentially sound indistinguishable from the instrument that they are simulating, it has been impossible, until recently, to make high-quality – defined here as non-linear and based on finite-difference (FD) schemes – physical models ‘playable’ in real-time [40]. With the computational power we currently possess, we can run the simulations in real-time and make them available for musicians in latency-less applications. These applications include digital instrument plug-ins that can be used by music producers, but also resurrect old or rare instruments that can not be played anymore due to damage, or because they are too valuable.

1.1 Physical Modelling Techniques

Nearly any musical instrument can be subdivided into a resonator component, an exciter component, and the interaction between them. This modular approach to musical instruments was first introduced by Borin, De Poli and Sarti in [8] and is used to structure this work.

The time-evolution of dynamic systems, including that of musical instruments, can be well described by partial differential equations (PDEs) [20, 3].

Examples of a dynamic systems are a guitar string, a drum-membrane, or air propagation in a concert hall; three very different concepts, but all based on the same types of equations of motion. Many of these equations and other knowledge currently available on the physics of musical instruments have been collected by Fletcher and Rossing in [20]. Though these equations are very powerful, only few have a closed-form solution, and in order for them to be implemented, they need to be approximated. In the past decades, much research has been done on implementing these PDEs to model and simulate different musical instruments. Great overviews of implementation techniques are given by, for example, Vesa Välimäki et al. in [46] and Julius O. Smith in [40, 42].

Some techniques that are described in this literature can be found below:

Modal Synthesis decomposes a system into a series of uncoupled ‘modes of vibration’. First used in a musical context by Morrison and Adrien in [35], 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 [5] with a small number of long-resonating modes [40]. When used to describe non-linear 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 [38].

Finite-Difference Time Domain methods (FDTD) aim to solve PDEs by approximating them with difference equations, discretising a continuous system into grid-points in space and time. In a musical context, this technique was first used for the case of string vibration in [39, 27, 28] and later in [13, 14]. Stefan Bilbao extensively describes this method in [3, 5]. Although computationally expensive, especially when working with higher-dimensional systems this technique can accurately resemble any system, whether it is linear or non-linear, time-invariant or time-variant.

Digital Waveguide Modelling (or Digital Waveguides (DWG)) is a modelling technique that discretises wave propagation and scattering. The technique was first presented in [41], 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 [46]. Some authors have combined DWGs with FD schemes (such as in [19, 30]) to accurately model non-linear behaviour while maintaining high-speed implementation.

1.1. Physical Modelling Techniques

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 [10, 11, 12] it is currently being further developed by Leonard and Villeneuve in a real-time, interactive environment [48, 29].

Other techniques include Functional Transformation Method [45], state-space modelling [31], [wave-domain modelling and energy-based port-Hamiltonian](#)

should I include this

- Modal Synthesis
 - modes of vibration
 - additive synthesis
 - independent vibration
- Finite-difference Time-domain methods
- Finite Element Methods
- Digital waveguides
- Mass-spring systems / networks
 - Cadoz with CORDIS-ANIMA
 - mi-Creative

Weird

- Functional transformation method
- State-space
- Wave-domain
- Energy-based (port-hamiltonian)

FDTD:

Energy techniques: [24] von Neumann analysis: [44]

The main advantage of FDTD methods is that they are extremely general and flexible in terms of the amount of systems they can model. Any set of PDEs can be directly numerically simulated.

Using FDTD methods can be quite computationally heavy. Moore's law [33]

1.2 Real-Time Implementation

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

Even though physical modelling has been a popular research field in the past few decades, relatively little research has been done on making the models work in real-time, i.e., 'playable' [?]. Several virtual string instruments and different electric pianos have been made real-time by Pfeifle and Bader in [?, ?, ?]. They used field programmable gate arrays (FPGAs) for implementing models based on FDTD methods. Furthermore, Roland's V-series use COSM (Composite Object Sound Modelling) technology [?] that implement real-time physical models in hardware instruments. In the NESS project [4, 7], Stefan Bilbao and his team focused on implementing systems using FDTD methods in real-time.

Real-time: no noticeable latency

1.3 Why?

1.3.1 Samples vs. Physical Modelling

Digital musical instruments based on samples, or recordings of an actual instrument, have an advantage of having an optimally realistic sound, as the output is exactly that of the original instrument. The digital version should thus sound indistinguishable from the instrument they try to digitise. Furthermore, the playback of samples is extremely computationally efficient as it simply requires loading data and playing this through a set of speakers.

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

exactly as in
[3]

1.4. Project Objectives and Main Contributions

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

1.3.2 Resurrect Old or Rare Instruments

In some cases, recording samples of an instrument is not even possible as they are too old, too rare or too vulnerable to be played.

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

1.3.3 Go beyond what is physically possible

Even though there are already existing real-time PMs (see Section ??), in my opinion, they do not make use of the full potential of using physical modelling for sound synthesis. PMs make it possible for parameters like shape, size, material properties, etc. to be changed, which is physically impossible or very hard to do. Furthermore, different instrument components can be combined to create hybrid instruments which can result in unique sounds that can only be created using PMs. The history of physical modelling of musical instruments

1.4 Project Objectives and Main Contributions

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

From [20] to [7]

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

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

As FDTD methods are quite rigid, changing parameters on the fly, i.e., while the instrument simulation is running, is a challenge. Other techniques, such as modal synthesis, are much more suitable for this, but come with the drawbacks mentioned in Section 1.1. Therefore, a novel method was devised to smoothly change parameters over time, introducing this to FDTD methods.

1.5 Thesis Outline

Introduction to finite-difference methods and analysis techniques

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

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

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

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

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

- Physical models
 - Resonators
 - Exciters
 - Interactions
- Dynamic Grids
- Real-Time Implementation and Control
- Complete instruments
 - Large-scale physical models
 - Tromba Marina
 - Trombone

Notes

- Think about how to define real-time.
- Create an intuition for different parts of the equation
- Talk about input and output locations and how that affects frequency content (modes).

One over number \rightarrow reciprocal of number

Example: When the waveform consists entirely of harmonically related frequencies, it will be periodic, with a period equal to the reciprocal of the fundamental frequency (from An Introduction to the Mathematics of Digital Signal Processing Pt 2 by F. R. Moore)

Chapter 2

Introduction to FDTD 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 the physical models presented later on in this document. By means of a simple mass-spring system and the 1D wave equation, the notation and terminology used throughout this document will be explained. Unless denoted otherwise, the notation has been taken from [3], most of which originates from [44].

2.1 Differential Equations

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

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

three dimensions [25] and potentially higher-dimensional systems! See Section 2.1.1 for more information on dimensions.

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

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

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

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

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

where the subscript notation could be seen as the partial derivative counterpart to Newton's notation due to its compactness. In the remainder of this document, Euler's notation will be used for PDEs, due to their similarity to operators in discrete time (introduced in Section 2.2.2) and as it allows for creation of bigger operators for more compactness when working with multiple (connected) systems (see e.g. Chapter 15). Also, state-of-the-art literature in the field of FDTD methods sound synthesis use this notation [5].

2.1.1 Dimensions and Degrees of Freedom

All objects in the physical world are three-dimensional (3D) as they have a non-zero width, length and depth. Moreover, these objects can move in these three dimensions and thus have three translational *degrees of freedom* (DoF) ([the three rotational DoF are ignored here](#)). To reduce the complexity of the model as well as computational complexity, 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 is orders of magnitude smaller than the others without affecting its behaviour much. A guitar string, for instance, has much greater length than its width or depth and can therefore be reduced to a one-dimensional (1D) system. If a 3D description were to be kept, the relative displacement between two locations on one cross-section along the length of the string would be taken into account. One could imagine that this displacement will always be orders of magnitude smaller than the relative displacement of two points along the string length and is thus negligible. Similarly, the thickness of a drum membrane is much smaller than its length and width and can therefore be simplified to a two-dimensional (2D) system.

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

check whether this needs to be per point along a system

Work has been done on strings with dual (transverse) polarisation by Desvages [18] and Desvages and Bilbao [17] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are couples can be found in [3, 6]. In [48], Villeneuve and Leonard present a mass-spring network where the state of every individual mass has three translational DoF. Due to these additional DoF, these networks do capture the aforementioned effects, but greatly increase the computational complexity of the models.

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

Notation

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

The translational DoF, on the other hand, determines the amount of coor-

ordinates 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 $u = u(x, y, t)$, 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. Finite-difference time-domain (FDTD) methods are the most straightforward approach to numerically approximate differential equations. These methods are considered of the most general and flexible techniques in terms of the systems they can model, and frankly, relatively simple to understand once some familiarity with them is obtained. The main concern with these methods is the numerical stability of the eventual approximation. Conditions for stability can be mathematically derived and will be introduced in Section 3.3.

FDTD methods essentially subdivide a continuous differential equation into discrete points in time and space, a process called *discretisation*. Once an ODE or PDE is discretised using these methods it is now called a *finite-difference (FD) scheme* which approximates the original differential equation. In the following, for generality and ease of explanation, a 1D system will be used. Unless denoted otherwise, the equations and theory used in this chapter has been taken from [3].

2.2. Discretisation using FDTD methods

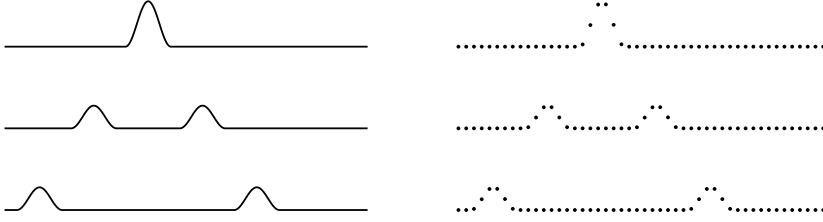


Fig. 2.1: A continuous PDE is discretised...

2.2.1 Grid Functions

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

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

To summarise, for a 1D system

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

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

FULL DOC SWEEP: check capitalisation of headings throughout document

grid figure

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

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

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

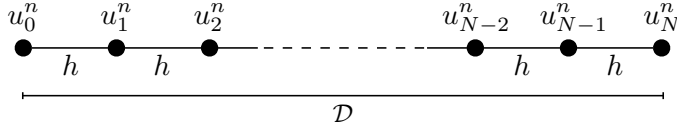


Fig. 2.2: When a 1D system $u(x, t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n , the spatial domain \mathcal{D} is divided into N intervals of length h and spatial range of interest $l = \{0, \dots, N\}$.

or spatially. Forward and backward shifts in time, together with the identity operation are

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

Similarly, forward and backward shifts in space are

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

These shift operators are rarely used in isolation, though they do appear in energy analysis techniques detailed in Section 3.4. The operators do, however, 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

$$\delta_{t+} \triangleq \frac{1}{k} (e_{t+} - 1), \quad (2.3a)$$

$$\delta_{t-} \triangleq \frac{1}{k} (1 - e_{t-}), \quad (2.3b)$$

$$\delta_{t\cdot} \triangleq \frac{1}{2k} (e_{t+} - e_{t-}), \quad (2.3c)$$

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

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

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

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

many figures for shift and FD operators

FULL DOC SWEEP: check centred instead of centered

these spacings are different in overleaf...

figure here visualising operators (with reference to grid figure)

2.2. Discretisation using FDTD methods

and when applied to u_l^n are

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

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

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

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

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

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

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

in energy analysis, interleaved grids, etc.

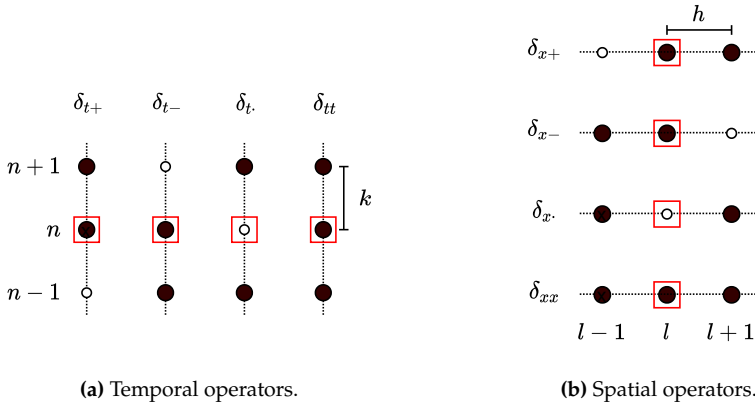


Fig. 2.3: The stencils of various FD operators applied to the grid point highlighted with a red square. Black grid points are used in the calculation, and white grid points are not. The averaging operators follow the same pattern.

are

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

$$\mu_{t\cdot} \triangleq \frac{1}{2} (e_{t+} + e_{t-}). \quad (2.9c)$$

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

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

Similarly, spatial averaging operators are

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

$$\mu_{x\cdot} \triangleq \frac{1}{2} (e_{x+} + e_{x-}), \quad (2.11c)$$

and when applied to u_l^n

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

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

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

and

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

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,

2.2. Discretisation using FDTD methods

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

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

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

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

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

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

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

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

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

and using Eq. (2.16) can be written to

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

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

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

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

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

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

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

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

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

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

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

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

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

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

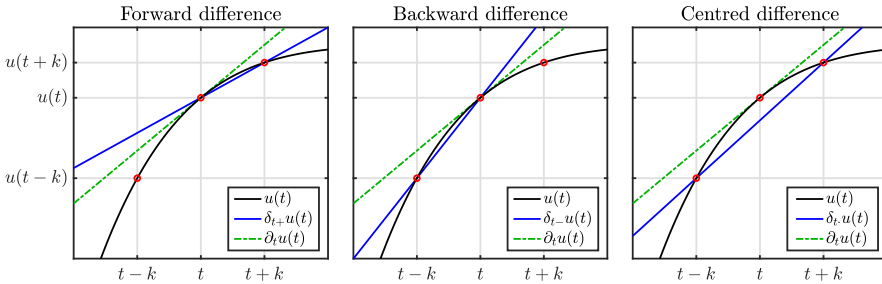


Fig. 2.4: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. One can observe that the centred difference operator much more closely approximates the derivative, or the slope, of u at t than the forward and backward difference operators.

2.3. The Mass-Spring System

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

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

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

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

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

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

2.2.3 Identities

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

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

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

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

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

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

2.3 The Mass-Spring System

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

2.3.1 Continuous-time

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

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

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

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

with angular frequency (in rad/s)

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

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

Apart from the choices of K and M , the behaviour of the mass-spring system is determined by its *initial conditions*, being $u(0)$ and $\partial_t u(0)$, i.e., the displacement and velocity of the mass at $t = 0$. If the initial conditions are non-zero, the path that the 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

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

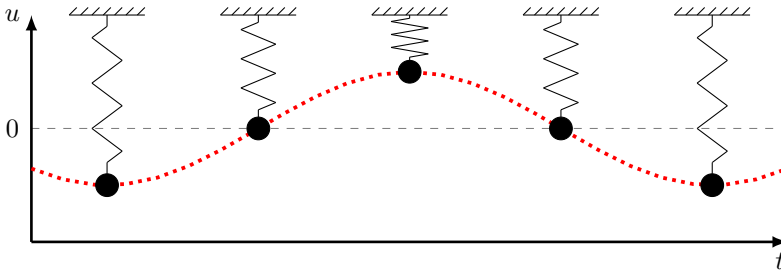


Fig. 2.5: Mass spring system over time. The system follows a harmonic (sinusoidal) motion.

2.3. The Mass-Spring System

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/s². Next, the force generated by the spring follows Hooke's law:

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

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

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

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

2.3.2 Discrete-time

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

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

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

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

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

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

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

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

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

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

2.3.3 Implementation and Output

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

To be able to calculate the scheme at $n = 0$ (the first time index of the simulation), values must be provided for u^0 and u^{-1} . These are determined by the initial conditions mentioned in Section 2.3.1. A simple way to obtain a sinusoidal motion with an amplitude of 1, is to set the initial conditions as follows (using the backwards time difference operator for discretising the first-order time derivative):

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

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

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

In short, setting $u^0 = u^{-1} \neq 0$ yields an oscillatory behaviour.

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

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

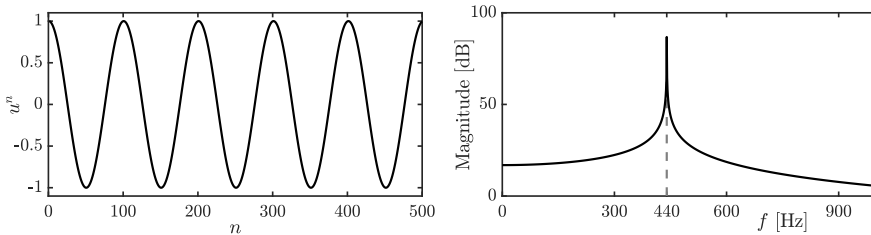


Fig. 2.6: The time-domain and frequency-domain output of a mass-spring system with $f_0 = 440$ Hz.

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

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

2.4 The 1D Wave Equation

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

2.4.1 Continuous time

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

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

where c is the wave speed (in 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.

FULL DOC
SWEEP: check
hyphen in ti-
tles

unit?

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 2.7 for a visualisation of the forces acting on u due to curvature.

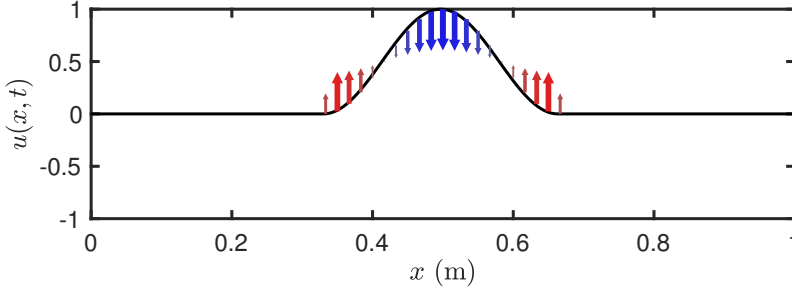


Fig. 2.7: The forces acting on the 1D wave equation described by $u(x, t)$ due to curvature. The arrows indicate the direction and magnitude of the force, and simultaneously the acceleration as these are connected through Eq. (2.38).

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 for the boundary conditions are

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

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

The Dirichlet boundary condition says that at the end points of the system, the state is 0 at all times. The Neumann condition on the other hand, says that rather the slope of these points needs to be 0, but that the end points are free to move transversely. In the former case, incoming waves to invert after reaching the boundary whereas in the latter incoming waves are reflected un-inverted. See Figure 2.8.

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

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

2.4. The 1D Wave Equation

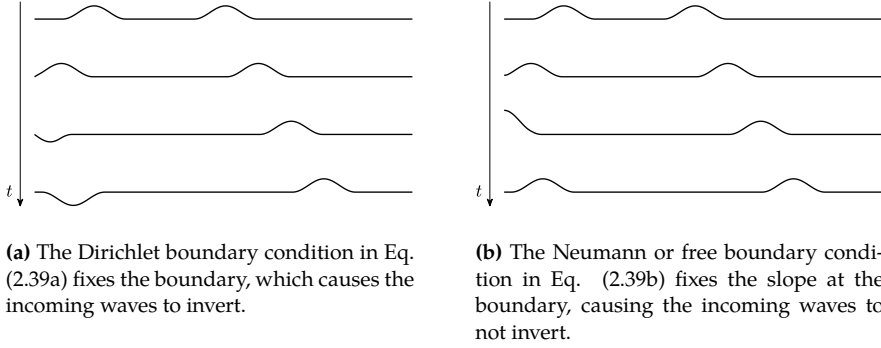


Fig. 2.8: The behaviour of the 1D wave equation with (a) Dirichlet or (b) Neumann boundary conditions.

Scaling

As this work follows much of Bilbao's *Numerical Sound Synthesis* [3], it might be good to talk about a major discrepancy between the PDEs and FD schemes that appear there and those used here. Non-dimensionalisation, or *scaling*, is extensively used in [3] and much of the literature published around that time (fx. [2, 6]) 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

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

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

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

check whether still correct

2.4.2 Discrete time

The most straightforward discretisation of Eq. (2.38) is the following FD scheme

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

FULL DOC SWEEP: check straightforward or straightforward

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

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

and solving for u_l^{n+1} yields

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

Here,

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

is called the *Courant number* and plays a big role in stability and quality of the FD scheme. More specifically, λ needs to abide the (famous) Courant-Friedrichs-Lewy or *CFL condition* for short [16]

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

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

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

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

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

Stencil

As was done for several FD operators in Figure 2.3, it can be useful to visualise the *stencil*, or region of operation, of a FD scheme. A stencil of a scheme visualises what grid values are necessary to calculate the state at the next time step u_l^{n+1} . Figure 2.9 shows the stencil for scheme (2.42) and – in essence – visualises the various shifts of the grid function in (2.44).

Boundary Conditions and Virtual Grid Points

The end points of the discrete domain are located at $l = 0$ and $l = N$. Substituting these locations into Eq. (2.44) seemingly shows that grid points outside of the defined domain are needed, namely u_{-1}^n and u_{N+1}^n . These can be referred

2.4. The 1D Wave Equation

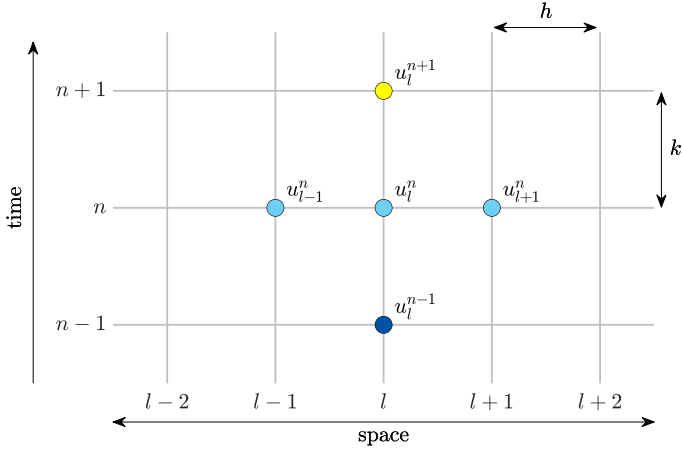


Fig. 2.9: The stencil, or region of operation, for the FD scheme in (2.42).

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

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

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

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

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

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

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

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

and

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

at the left and right boundary respectively.

2.4.3 Implementation: Excitation and Output

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

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

The discrete raised cosine can be parameterised 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

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

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

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

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

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

Output and Modes

After the system is excited, one can retrieve the output of the system by selecting a grid point l_{out} and listening to that at the given sample rate f_s . An example using the parameters in Table 2.1 and Dirichlet boundary conditions is shown in Figure 2.10.

As can be seen from Figure 2.10, 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 2.11). Although the PDE has not been discretised using modal synthesis (another physical modelling technique discussed in Chapter 1), the system can still be decomposed into different modes of vibration, each corresponding to a

2.4. The 1D Wave Equation

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 $\lambda = 1$.

harmonic frequency. These modes are assumed to vibrate independently, and their weighted sum yields the eventual behaviour of the system.

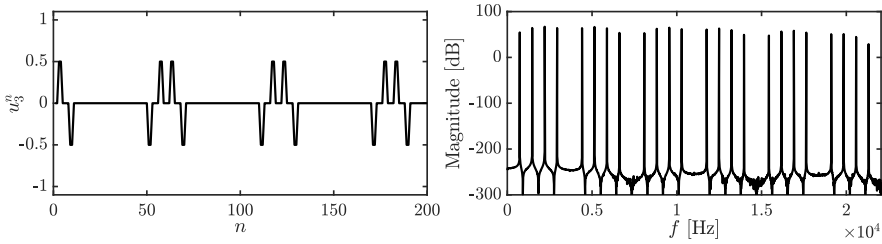


Fig. 2.10: The time-domain and frequency-domain output of the 1D wave equation with $f_0 = 735$ Hz and $f_s = 44100$ Hz ($N = 30$ and $\lambda = 1$) and Dirichlet boundary conditions. The system is initialised with a raised cosine described in Eq. (2.52) with $l_c = 0.2N$) and $w = 4$ and the output is retrieved at $l_{out} = 3$.

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 $m \in \{1, \dots, N - 1\}$ for Dirichlet and $m \in \{0, \dots, N\}$ for Neumann boundary conditions. The frequency of the

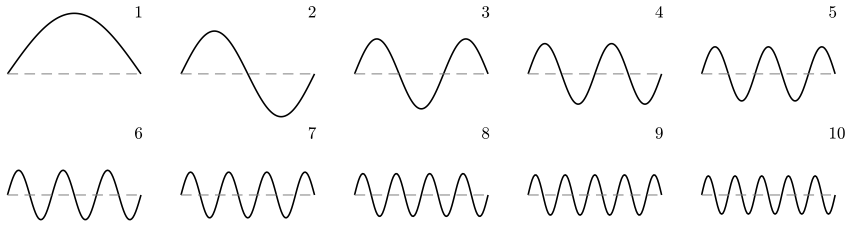


Fig. 2.11: The first 10 modal shapes of the 1D wave equation with Dirichlet boundary conditions. The number of the shape can be determined by the amount of antinodes present in the shape.

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

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 physical!²

Besides stability, the value of λ is closely related to the quality of the simulation. If $\lambda = 1$, Eq. (2.42) is actually an exact solution to Eq. (2.38), which is quite uncommon in the realm of differential equations! See Figure 2.12a. Identically, if Eq. (2.47) is satisfied with equality, the FD scheme is an exact solution to the PDE, and if h deviates from this condition, the quality of the simulation decreases.

If $\lambda < 1$, the quality of the simulation quality decreases in an effect called *numerical dispersion*. Dispersion is a phenomenon where some frequencies

²Imagine 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.

2.4. The 1D Wave Equation

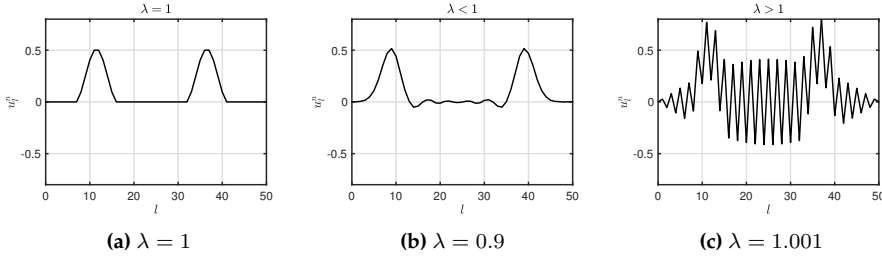


Fig. 2.12: Grid function u_l^n visualised ~ 100 samples after excitation. (a) If $\lambda = 1$, the solution is exact. (b) If $\lambda < 1$ dispersive behaviour shows. (c) If $\lambda > 1$ the CFL condition in Eq. (2.46) is not satisfied and the system is unstable.

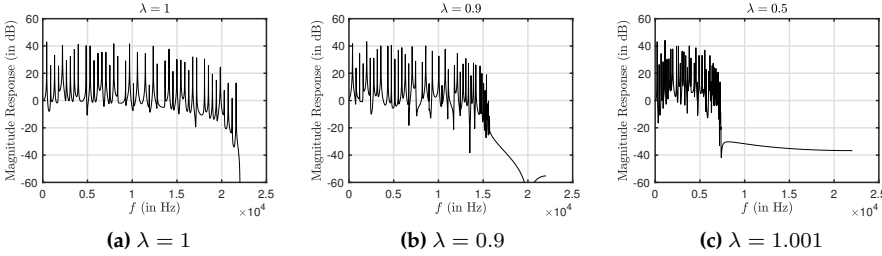


Fig. 2.13: Frequency spectra of the simulation output. The Courant number is set to (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$. One can observe that for lower values of λ the bandwidth of the output decreases drastically.

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, is usually not. Figure 2.12b shows an example when $\lambda = 0.9$, and one can observe that the wave propagation does not match the ideal case as Figure 2.12a shows. Moreover, bandlimiting effects occur, meaning that the highest frequency that the system can generate decreases. See Figure 2.13. 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 2.12c. Unstable behaviour usually comes in the form of high frequencies (around the Nyquist frequency of $f_s/2$) growing without bounds.

So in what situation would the stability condition not be satisfied with equality? As mentioned in Section 2.2.1, a continuous domain $\mathcal{D} = [0, L]$ for a system of length L needs to be divided into N equal sections of length h in the discretisation process. A logical step to calculate N would be to divide L by h calculated using Eq. (2.47) satisfied with equality to get the highest

possible simulation quality. However, this calculation might not result in an integer value, which N should be! To stay as close to the stability condition as possible, the following calculations are performed in order:

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

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. Take 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.

more intuition
on stability
and gridspacing
here

Chapter 3

Analysis Techniques

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

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

perhaps also
dispersion
analysis?

3.1 Matrices

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

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

3.1.1 Operations

Adding, subtracting, 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

$$p + \mathbf{A} = \mathbf{A} + p = \begin{bmatrix} p + a_{11} & p + a_{12} \\ p + a_{21} & p + a_{22} \end{bmatrix}, \quad p - \mathbf{A} = \mathbf{A} - p = \begin{bmatrix} p - a_{11} & p - a_{12} \\ p - a_{21} & p - a_{22} \end{bmatrix}$$

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

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

Matrix transpose

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

$$a_{ij} = a_{ji}. \quad (3.1)$$

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

Matrix Multiplication

Matrix multiplication (this includes matrix-vector multiplication) is different from regular multiplication in that it needs to abide several extra rules. In order for matrix multiplication to be valid, the number of columns of the first matrix needs to be equal to the number of rows in the second matrix. The

3.1. Matrices

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

(a) A 3×5 matrix \mathbf{A} .

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

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

result will then be a matrix with a number of rows equal to that of the first matrix and a number of columns equal to that of the second matrix. See Figure 3.2 for reference.

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

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

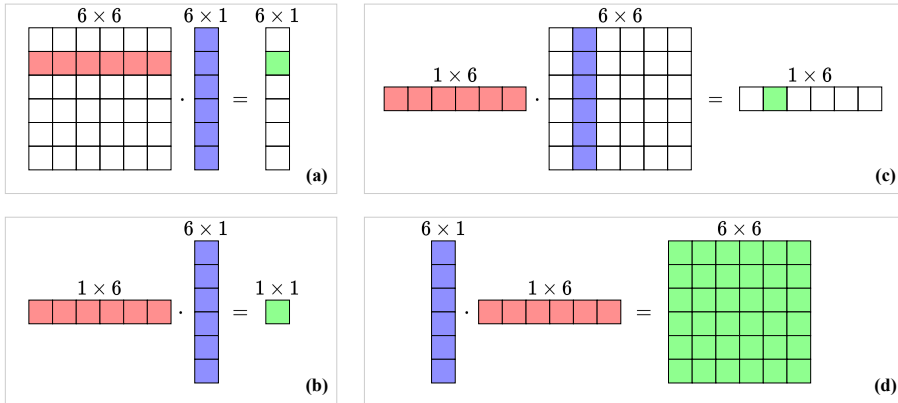


Fig. 3.2: Visualisation of valid matrix multiplications. The “inner” dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of “outer” dimensions (rows of the left matrix and columns of the right).

3.1.2 In a FDTD context

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

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \quad (3.3)$$

Expanding this operation results in

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

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

Operators in Matrix Form

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

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

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

3.1. Matrices

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 0s indicate that the rest of the values in the matrix are zeros.

is this how you explain it?

Averaging operators μ_{x+} , μ_{x-} and μ_x are defined in a similar way:

[illegible]

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

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

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

Schemes and Update Equations in Matrix Form

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

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

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

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

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

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

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

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

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

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

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

The identity matrix is necessary here for correct matrix addition.

3.1.3 Matrix Inverse

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

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \quad (3.9)$$

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

3.1. Matrices

$$\mathbf{A_inverted} = \text{inv}(\mathbf{A});$$

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

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

3.1.4 Systems of Linear Equations

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

For example, take the system of linear equations

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

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

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \quad (3.10)$$

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

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

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

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \quad (3.11)$$

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

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

$$\mathbf{u} = \text{inv}(\mathbf{A}) * \mathbf{w};$$

or more compactly, by using the \backslash operator:

$$\mathbf{u} = \mathbf{A} \backslash \mathbf{w};$$

3.1.5 Eigenvalue Problems

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

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

$$\mathbf{A}\phi = \lambda\phi. \quad (3.12)$$

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

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

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

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

The p^{th} eigenvector appears in the p^{th} column of $p \times p$ matrix `phi` and the corresponding eigenvalues are given in a $p \times 1$ column vector `lambda`. Note that the outcome is not necessarily sorted! To do this, do

```
[lambda, order] = sort(lambda);  
phi = phi(:, order);
```

3.2 Mathematical Tools and Product Identities

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

3.2.1 Inner product

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

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

ask stefan

check whether
this is neces-
sary

3.2. Mathematical Tools and Product Identities

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

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

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

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

$$\langle f_l^n, g_l^n \rangle'_d = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{h}{2} f_0^n g_0^n + \frac{h}{2} f_N^n g_N^n, \quad (3.16)$$

and the more general weighted inner product

$$\langle f_l^n, g_l^n \rangle_d^{\epsilon_l, \epsilon_r} = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{\epsilon_l}{2} h f_0^n g_0^n + \frac{\epsilon_r}{2} h f_N^n g_N^n, \quad (3.17)$$

which scale the boundary points of the regular inner product. Naturally, if $\epsilon_l = \epsilon_r = 1$, Eq. (3.17) reduces to Eq. (3.16), and if $\epsilon_l = \epsilon_r = 2$, (3.17) reduces to (3.15).

3.2.2 Summation by Parts

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

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

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

where $f g|_0^L$ 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 $d = \{0, \dots, N\}$. Then, using the discrete inner product in Eq. (3.15), two variants of summation by parts are defined as

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

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

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

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

$$\langle f_l^n, \delta_{x-} g_l^n \rangle_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle_{\underline{d}} + f_N^n g_N^n - f_0^n g_{-1}^n, \quad (3.20a)$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle_{\bar{d}} + f_N^n g_{N+1}^n - f_0^n g_0^n, \quad (3.20b)$$

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

$$\langle f_l^n, \delta_{x-} g_l^n \rangle'_d = -\langle \delta_{x+} f_l^n, g_l^n \rangle'_{\underline{d}} + f_N^n \mu_{x-} g_N^n - f_0^n \mu_{x-} g_0^n, \quad (3.21a)$$

$$\langle f_l^n, \delta_{x+} g_l^n \rangle'_d = -\langle \delta_{x-} f_l^n, g_l^n \rangle'_{\bar{d}} + f_N^n \mu_{x+} g_N^n - f_0^n \mu_{x+} g_0^n, \quad (3.21b)$$

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

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

$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \quad (3.22a)$$

$$\langle f, \delta_{xx} g \rangle_d = \langle \delta_{xx} f, g \rangle_{\bar{d}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0, \quad (3.22b)$$

$$\langle f, \delta_{xx} g \rangle'_d = \langle \delta_{xx} f, g \rangle'_d + f_N \delta_{x-} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0 \quad (3.22c)$$

Derivations

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

3.2. Mathematical Tools and Product Identities

Starting with Eq. (3.19a), suppressing the n superscript for brevity, and using the definition for the discrete inner product in Eq. (3.15), we get

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

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

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

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

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

3.2.3 Product identities

Some useful identities used in this work are

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

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

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

$$(\delta_t.u_l^n)(\mu_t.u_l^n) = \delta_t. \left(\frac{1}{2}(u_l^n)^2 \right), \quad (3.23d)$$

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

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

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

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

3.3 Frequency Domain Analysis

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

Frequency-domain representation and Ansatz

Frequency-domain analysis of FD schemes starts by performing a *z-transform* on the scheme. The *z-transform* converts a discrete signal into a frequency-domain representation, and is extensively used in the field of digital signal processing (DSP) to analyse the behaviour and especially stability of digital filters. To not go too much into detail here, the interested reader is referred to the very comprehensive explanation on the *z-transform* given in [36, 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 [44] and is heavily used

3.3. Frequency Domain Analysis

in [3]. The discrete-time z-transform and discrete spatial Fourier transform performed on a 1D grid function are defined as [3]

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

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

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) [44]

$$u_l^n \xrightarrow{A} z^n e^{jl\beta h} \quad (3.26)$$

where “ \xrightarrow{A} ” 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.

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

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

Using these definitions, the effect of various operators on a grid function can be written in their frequency-domain representation. For systems distributed in space, the following trigonometric identities are extremely useful when

check if I should not refer to a subsection

check reference

check

performing the analyses [51, p. 71]:

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

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

Take for example

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

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

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

and substituted into the above to get

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

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

$$\begin{aligned} \delta_{t+}u_l^n &\xrightarrow{A} \frac{1}{k} (z - 1), & \delta_{t-}u_l^n &\xrightarrow{A} \frac{1}{k} (1 - z^{-1}), \\ \delta_t u_l^n &\xrightarrow{A} \frac{1}{2k} (z - z^{-1}), & \delta_{tt}u_l^n &\xrightarrow{A} \frac{1}{k^2} (z - 2 + z^{-1}) \end{aligned} \quad (3.28)$$

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

$$\delta_{xx}u_l^n \xrightarrow{A} -\frac{4}{h^2} \sin^2(\beta h/2), \quad (3.29a)$$

$$\delta_{xxxx}u_l^n \xrightarrow{A} \frac{16}{h^4} \sin^4(\beta h/2). \quad (3.29b)$$

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

Proving stability

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

$$|z| \leq 1. \quad (3.30)$$

In the FDTD context, the frequency-domain representation of a FD scheme results in a *characteristic equation* – which is usually a second-order polynomial

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

not talking
about nonlin-
ear systems
though

only the de-
nominator of
the transfer
function

3.3. Frequency Domain Analysis

– in z and needs to satisfy condition (3.30) for all wave numbers β . It can be shown that for a polynomial of the form

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

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

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

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

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

suffices.

3.3.1 Mass-Spring System

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

$$M\delta_{tt}u_l^n = -Ku_l^n$$

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

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

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

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

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

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

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

$$k < 2\sqrt{\frac{M}{K}}. \quad (3.35)$$

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

$$k < \frac{2}{\omega_0}. \quad (3.36)$$

3.3.2 1D Wave Equation

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

Recalling the FD scheme in (2.42):

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

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

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

Also recalling that

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

the characteristic equation of the 1D wave equation is

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

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

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

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

3.4. Energy Analysis

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

$$h \geq ck, \quad (3.39)$$

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

3.4 Energy Analysis

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

Energy analysis techniques first appeared in (the first edition of) [24] and was back then referred to the energy method. In [3] they have been extensively used to show stability of the FD schemes used.

shouldn't I just cite the first edition then?

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

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

3.4.1 Energy Analysis: A 4-Step Tutorial

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

The first step to energy analysis is to take the appropriate *norm* of the scheme (see Eq. (3.14)), which yields an expression for the rate of change of the

energy of the system: $\delta_{t+} \mathfrak{h}$. Usually, this means to take the inner product of the scheme with $(\delta_{t+} u_l^n)$. 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

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

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

$$\delta_{t+} \mathfrak{h} = 0 \implies \mathfrak{h}^n = \mathfrak{h}^0. \quad (3.41)$$

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \quad (3.42)$$

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

Step 3: Check units

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

3.4. Energy Analysis

be in Joules per second or in SI units: $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$. Some information about operators and grid functions and how they ‘add’ units to the equation will be given below.

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

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

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

$$\mathfrak{h}_e^n = \frac{\mathfrak{h}^n - \mathfrak{h}^0}{\mathfrak{h}^0}, \quad \text{if } \mathfrak{h}^0 \neq 0, \quad (3.43)$$

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 3.3 and 3.4.

For a system with losses or energy inputs, a discrete integration, or summed form can be used (as done in fx. [26]):

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

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

3.4.2 Mass-spring system

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Step 3: Check units

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

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

and indeed have the correct units.

Step 4: Implementation

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

3.4. Energy Analysis

```

1 %% Calculate the energy using Eq. (3.48)
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 3.3 shows the normalised energy (according to Eq. (3.43)) of the mass-spring system and shows that the deviation is indeed within machine precision.

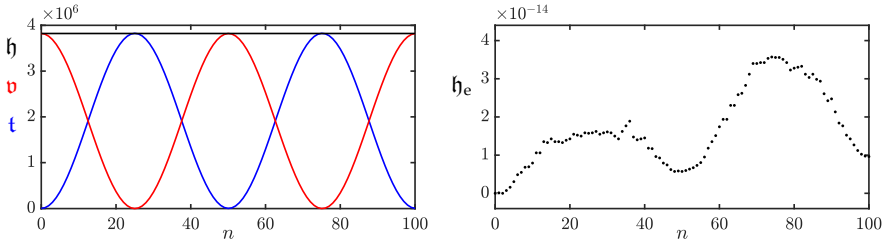


Fig. 3.3: The kinetic (blue), potential (red), and total (black) energy of an implementation of the mass-spring system are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.43)). Notice that the scaling of the y-axis is 10^{-14} and the energy is thus within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

3.4.3 1D Wave Equation

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

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

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

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

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

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

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

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

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

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

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

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

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. (3.15), Eq. (3.50) can be expanded to

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

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

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v}, \quad (3.53)$$

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

3.4. Energy Analysis

Step 3: Check units

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

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

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

Step 4: Implementation

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

```

1 %% Calculate the energy using Eq. (3.53)
2
3 % Kinetic energy
4 kinEnergy(n) = rho * A / 2 * h * sum((1/k * (u-uPrev)).^2);
5
6 % Potential energy
7 potEnergy(n) = T/(2*h) * sum(( [u; 0] - [0; u] ) ...
8     .* ([uPrev; 0] - [0; uPrev]));
9
10 % Total energy (Hamiltonian)
11 totEnergy(n) = kinEnergy(n) + potEnergy(n);

```

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

Neumann boundary conditions

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

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

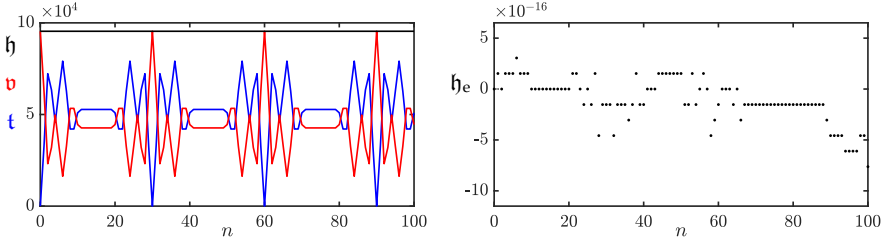


Fig. 3.4: The kinetic (blue), potential (red), and total (black) energy of an implementation of the 1D wave equation are plotted in the left panel. The right panel shows the normalised energy (according to Eq. (3.43)) and shows that the deviation of the energy is within machine precision.

where the boundary term

$$\begin{aligned} \mathbf{b} &= T(\delta_t \cdot u_N^n)(\mu_x - \delta_x \cdot u_N^n) - T(\delta_t \cdot u_0^n)(\mu_x - \delta_x \cdot u_0^n), \\ &\stackrel{\text{Eq. (2.27b)}}{\Longleftrightarrow} = T(\delta_t \cdot u_N^n)(\delta_x \cdot u_N^n) - T(\delta_t \cdot u_0^n)(\delta_x \cdot u_0^n). \end{aligned}$$

As the Neumann boundary condition states that

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

the boundary term vanishes and the energy balance results in

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

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

```
1 %% Calculate the energy using Eq. (3.54)
2
3 % Scaling of the boundaries through weighted inner product
4 scaling = [0.5; ones(N-1, 1); 0.5];
5
6 % Kinetic energy
7 kinEnergy(n) = rho * A / 2 * h * sum(scaling .* (1/k * (u-uPrev)).^2);
8
9 % Potential energy
10 potEnergy(n) = T/(2*h) * sum(u(2:end) - u(1:end-1) ...
11     .* (uPrev(2:end) - uPrev(1:end-1)));
12
13 % Total energy (Hamiltonian)
14 totEnergy(n) = kinEnergy(n) + potEnergy(n);
```

3.4.4 Stability Analysis using Energy Analysis Techniques

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

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

Mass-spring system

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

$$x^2 + y^2 + 2axy \quad (3.55)$$

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

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

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

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

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

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

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

This can then be written as

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

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

$$k < 2\sqrt{\frac{M}{K}} \quad (3.56)$$

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

1D wave equation

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

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

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

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

to put a condition on \mathfrak{v}

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

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

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

3.5. Modal Analysis

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

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

and is non-negative for

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

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

3.5 Modal Analysis

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

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

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

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

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

more explanation, perhaps refer to von neumann analysis in 3.3

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

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

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

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

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

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

and can be rewritten to

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

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

3.5.1 One-Step Form

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

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

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

can be rewritten to

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

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

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

$$z\phi = \mathbf{Q}\phi, \quad (3.65)$$

which can be solved for the p th eigenvalue as

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

3.6. Dispersion analysis

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

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

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

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

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

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

3.6 Dispersion analysis

not sure whether to include..

Part II

Resonators

Resonators

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

This part presents the different resonators used over the course of the project and is structured as follows: Chapter 4 introduces the stiff string, a model which has been used a lot over the course of this project, Chapter 5 talks about brass instruments, or more generally, 1D systems of varying geometry along their spatial dimension. Finally, Chapter 6 will introduce 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 *inharmonic* due to stiffness in the material.

The restoring force of the 1D wave equation is only due to tension. In the real world, however, this force is also due to stiffness, dependent on the material and geometry of the string. This stiffness causes frequency dispersion and inharmonicity: the “harmonic” partials get exponentially further apart the higher their frequency is.

The stiff string played a prominent part in the following published works [A], [B], [C], [D] and [E].

This chapter presents the PDE of the stiff string, and goes through the discretisation process afterwards. The analysis techniques presented in the previous chapter will then be applied to the stiff string.

not done

4.1 Continuous time

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

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

parameterised 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 either E or I is 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.

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

Dispersion Analysis or: Frequency Dispersion

The 4th-order spatial derivative models *frequency dispersion*, a phenomenon that causes different frequencies to travel at different speeds. As opposed to the undesired numerical dispersion

not done

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 [1] and has been used extensively in the literature since. A damped stiff string can be modelled as

$$\rho A \partial_t^2 u = T \partial_x^2 u - EI \partial_x^4 u - 2\sigma_0 \rho A \partial_t u + 2\sigma_1 \rho A \partial_t \partial_x^2 u, \quad (4.2)$$

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. (4.2), and as is also found often in the literature [3], is to divide both sides by ρA to get

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa^2 \partial_x^4 u - 2\sigma_0 \partial_t u + 2\sigma_1 \partial_t \partial_x^2 u \quad (4.3)$$

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).

Intuition

Although Eq. (4.2) might look daunting at first, the principle of Newton's second law remains the same.

Something about the 4th spatial derivative and the loss terms here...

Boundary Conditions

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

$$u = \partial_x u = 0 \quad (\text{clamped}) \quad (4.4a)$$

$$u = \partial_x^2 u = 0 \quad (\text{simply supported}) \quad (4.4b)$$

$$\partial_x^2 u = \partial_x^3 u = 0 \quad (\text{free}) \quad (4.4c)$$

at $x = 0, L$. See Figure 4.1 for plots of the first modal shape for each respective boundary condition.

4.2. Discrete Time

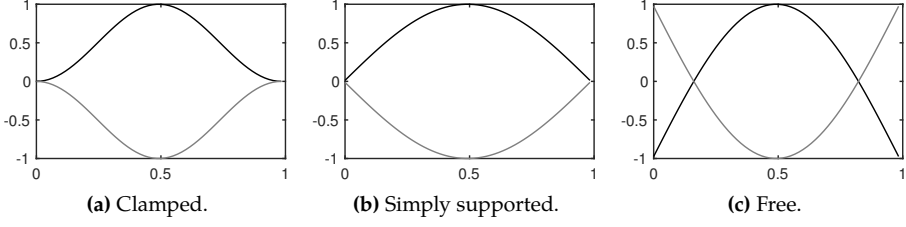


Fig. 4.1: Plots of the first (normalised) modal shape for the three boundary conditions in Eqs. (4.4). The extremes are indicated with black and grey lines respectively.

4.2 Discrete Time

For the sake of brevity we continue with Eq. (4.3), rather than Eq. (4.2). Naturally, same process can be followed for the latter, the only difference being a multiplication by ρA of all terms.

Equation (4.3) can be discretised as

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t-}u_l^n + 2\sigma_1\delta_{t-}\delta_{xx}u_l^n, \quad (4.5)$$

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

$$\delta_{xxxx} = \delta_{xx}\delta_{xx} = \frac{1}{h^4} (e_{x+}^2 - 4e_{x+} + 6 - 4e_{x-} + e_{x-}^2). \quad (4.6)$$

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

$$\delta_{xxxx}u_l^n = \frac{1}{h^4} (u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n). \quad (4.7)$$

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

$$\begin{aligned} \delta_{t-}\delta_{xx} &= \frac{1}{k} (1 - e_{t-}) \frac{1}{h^2} (e_{x+} - 2 + e_{x-}), \\ &= \frac{1}{kh^2} (e_{x+} - 2 + e_{x-} - e_{t-}(e_{x+} - 2 + e_{x-})). \end{aligned} \quad (4.8)$$

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

$$\delta_{t-}\delta_{xx}u_l^n = \frac{1}{hk^2} (u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} - u_{l-1}^{n-1}). \quad (4.9)$$

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

With these definitions, the operators in scheme (4.5) can be expanded to get

$$\begin{aligned} \frac{1}{k^2}(u_l^{n+1} - 2u_l^n + u_l^{n-1}) &= \frac{c^2}{h^2}(u_{l+1}^n - 2u_l^n + u_{l-1}^n) \\ &\quad - \frac{\kappa^2}{h^4}(u_{l+2}^n - 4u_{l+1}^n + 6u_l^n - 4u_{l-1}^n + u_{l-2}^n) \\ &\quad - \frac{\sigma_0}{k}(u_l^{n+1} - u_l^{n-1}) \\ &\quad + \frac{2\sigma_1}{kh^2}(u_{l+1}^n - 2u_l^n + u_{l-1}^n - u_{l+1}^{n-1} + 2u_l^{n-1} + u_{l-1}^{n-1}), \end{aligned} \quad (4.10)$$

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

$$\begin{aligned} (1 + \sigma_0 k)u_l^{n+1} &= \left(2 - 2\lambda^2 - 6\mu^2 - \frac{4\sigma_1 k}{h^2}\right)u_l^n \\ &\quad + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2}\right)(u_{l+1}^n + u_{l-1}^n) \\ &\quad - \mu^2(u_{l+2}^n + u_{l-2}^n) + \left(-1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2}\right)u_l^{n-1} \\ &\quad - \frac{2\sigma_1 k}{h^2}(u_{l+1}^{n-1} + u_{l-1}^{n-1}), \end{aligned} \quad (4.11)$$

with

$$\lambda = \frac{ck}{h} \quad \text{and} \quad \mu = \frac{\kappa k}{h^2}. \quad (4.12)$$

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

The stability condition for the FD scheme in (4.5) is defined as

$$h \geq \sqrt{\frac{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}, \quad (4.13)$$

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. (4.13) should be satisfied with equality, after which

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

which can then be used to calculate λ and μ in (4.12).

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 4.2. 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.

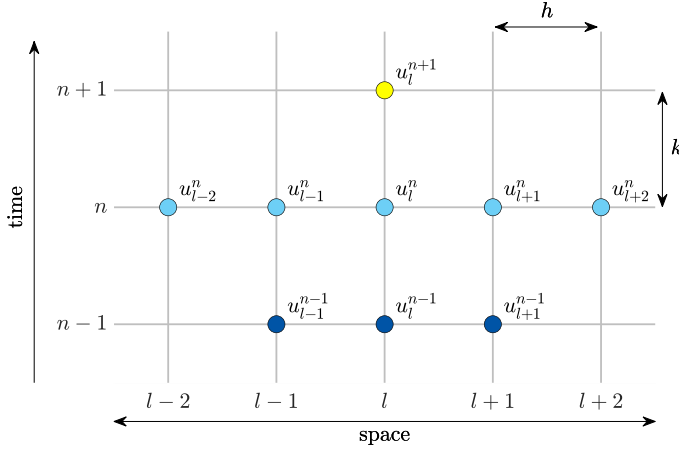


Fig. 4.2: The stencil for the damped stiff string scheme in Eq. (4.5).

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 conditions in (4.4) yields

$$u_l^n = \delta_{x\pm} u_l^n = 0 \quad (\text{clamped}) \quad (4.14a)$$

$$u_l^n = \delta_{xx} u_l^n = 0 \quad (\text{simply supported}) \quad (4.14b)$$

$$\delta_{xx} u_l^n = \delta_x \cdot \delta_{xx} u_l^n = 0 \quad (\text{free}) \quad (4.14c)$$

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. (4.4c), to discretise ∂_x^3 the more accurate $\delta_x \cdot \delta_{xx}$ operator has been chosen over the less accurate $\delta_{t-} \delta_{xx}$ and $\delta_{t+} \delta_{xx}$ 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

$$u_0^n = u_1^n = 0 \quad \text{and} \quad u_{N-1}^n = u_N^n = 0. \quad (4.15)$$

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

Simply supported

As the states of the end points of a system with simply supported boundary conditions are 0 at all times, the range of calculation can be reduced to $l \in \{1, \dots, N-1\}$. At $l = 1$ and $l = N-1$, definitions for the virtual grid points u_{-1}^n and u_{N+1}^n are needed. A definition for u_{-1}^n can be found by expanding Eq. (4.14b) at $l = 0$:

$$\begin{aligned} \frac{1}{h^2} (u_1^n - 2u_0^n + u_{-1}^n) &= 0, \\ \xLeftrightarrow{u_0^n=0} u_1^n + u_{-1}^n &= 0, \\ u_{-1}^n &= -u_1^n, \end{aligned} \quad (4.16)$$

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

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

Filling the first definition into the expanded scheme at $l = 1$ in (4.10) and collecting the terms yields

$$\begin{aligned} (1 + \sigma_0 k) u_1^{n+1} &= \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2} \right) u_1^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2} \right) u_2^n \\ &\quad - \mu^2 u_3^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2} \right) u_1^{n-1} - \frac{2\sigma_1 k}{h^2} u_2^{n-1}. \end{aligned} \quad (4.17)$$

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

$$\begin{aligned} (1 + \sigma_0 k) u_{N-1}^{n+1} &= \left(2 - 2\lambda^2 - 5\mu^2 - \frac{4\sigma_1 k}{h^2} \right) u_{N-1}^n + \left(\lambda^2 + 4\mu^2 + \frac{2\sigma_1 k}{h^2} \right) u_{N-2}^n \\ &\quad - \mu^2 u_{N-3}^n + \left(-1 + \sigma_0 k + \frac{4\sigma_0 k}{h^2} \right) u_{N-1}^{n-1} - \frac{2\sigma_1 k}{h^2} u_{N-2}^{n-1}. \end{aligned} \quad (4.18)$$

Free

Finally, the free boundary condition requires all points to be calculated and the range of calculation is $l \in \{0, \dots, N\}$. At each respective boundary, two virtual grid points are needed: u_{-1}^n and u_{-2}^n at the left and u_{N+1}^n and u_{N+2}^n

4.2. Discrete Time

at the right boundary respectively. The combined operator in Eq. (4.14c) is defined as:

$$\begin{aligned}\delta_x \cdot \delta_{xx} &= \frac{1}{2h^3} (e_{x+} - e_{x-}) (e_{x+} - 2 + e_{x-}), \\ &= \frac{1}{2h^3} (e_{x+}^2 - 2e_{x+} + 1 - (1 - 2e_{x-} + e_{x-}^2)), \\ &= \frac{1}{2h^3} (e_{x+}^2 - 2e_{x+} + 2e_{x-} - e_{x-}^2),\end{aligned}\tag{4.19}$$

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

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

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

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

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

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

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

In practice, the simply supported boundary condition is mostly chosen as this reflects reality the most. The clamped condition could be chosen for simplicity as this does not require an alternative update at the boundaries. The free boundary condition is more often used to model a (damped) ideal bar, (Eq. (4.2) with $T = 0$).

4.2.2 Implementation and Matrix Form

When using MATLAB, for a more compact implementation, it is useful to write the scheme in matrix form. The FD scheme of the stiff string in (4.5) can be written as

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}\tag{4.20}$$

where

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

Notice that A is a scalar rather than a matrix.

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

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

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

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

Finally for free boundary conditions as given in (4.14c), the state vectors and matrices are $(N+1) \times 1$ and $(N+1) \times (N+1)$ respectively. Now, the \mathbf{D}_{xx} matrix is of the form in (3.6) instead, and

$$\mathbf{D}_{xxxx} = \frac{1}{h^4} \begin{bmatrix} 2 & -4 & 2 & & \mathbf{0} \\ -2 & 5 & -4 & 1 & \\ 1 & -4 & 6 & -4 & 1 \\ & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & 1 & -4 & 6 & -4 & 1 \\ & & & 1 & -4 & 5 & -2 \\ \mathbf{0} & & & & 2 & -4 & 2 \end{bmatrix}. \quad (4.23)$$

4.2.3 Parameters and output

There exists a formula to calculate the loss coefficients σ_0 and σ_1 from T_{60} values at different frequencies (see [3, Eq (7.29)]). During this project, however, these values have been tuned by ear and are usually set to be in the range of $\sigma_0 = 1$ $\sigma_1 = 0.005$. See Table 4.1 for parameters most commonly used in this project.

Name	Symbol (unit)	Value
Length	L (m)	1
Material density	ρ (kg/m ³)	7850
Radius	r (m)	$5 \cdot 10^{-3}$
Tension	T (N)	$100 \leq T \leq 15^4$
Young's modulus	E (Pa)	$2 \cdot 10^{11}$
Freq.-independent damping	σ_1 (s ⁻¹)	1
Freq.-dependent damping	σ_0 (m ² /s)	0.005

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

Output

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

time-domain
and fft

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 (4.5).

Using the definitions found in Eq. (3.28) for the temporal operators and Eqs. (3.29a) and (3.29b) for the spatial operators, the frequency-domain representation of Eq. (4.5) can be obtained:

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

and after collecting the terms, the characteristic equation follows:

$$(1 + \sigma_0 k)z + \left(16\mu^2 \sin^4(\beta h/2) + \left(4\lambda^2 + \frac{8\sigma_1 k}{h^2}\right) \sin^2(\beta h/2) - 2\right) \\ + \left(1 - \sigma_0 k - \frac{8\sigma_1 k}{h^2} \sin^2(\beta h/2)\right) z^{-1} = 0. \quad (4.24)$$

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

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

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

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

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. (4.12) yields

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

different wording

4.4. Energy Analysis

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

$$h \geq \sqrt{\frac{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}}{2}}, \quad (4.25)$$

which is the stability condition for the damped stiff string also shown in Eq. (4.13).

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 (4.2) yields

$$\rho A \delta_{tt} u_l^n = T \delta_{xx} u_l^n - EI \delta_{xxxx} u_l^n - 2\sigma_0 \rho A \delta_t u_l^n + 2\sigma_1 \rho A \delta_{t-} \delta_{xx} u_l^n, \quad (4.26)$$

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. (3.15)) of the scheme with $(\delta_t u_l^n)$ over discrete domain d :

$$\begin{aligned} \delta_{t+} \mathfrak{h} &= \rho A \langle \delta_t u_l^n, \delta_{tt} u_l^n \rangle_d - T \langle \delta_t u_l^n, \delta_{xx} u_l^n \rangle_d + EI \langle \delta_t u_l^n, \delta_{xxxx} u_l^n \rangle_d \\ &\quad + 2\sigma_0 \rho A \langle \delta_t u_l^n, \delta_t u_l^n \rangle_d - 2\sigma_1 \rho A \langle \delta_t u_l^n, \delta_{t-} \delta_{xx} u_l^n \rangle_d = 0. \end{aligned} \quad (4.27)$$

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

$$\delta_{t+} \mathfrak{h} = \mathfrak{b} - \mathfrak{q}. \quad (4.28)$$

The damping term is defined as

$$\mathfrak{q} = 2\sigma_0 \rho A \|\delta_t u_l^n\|_d^2 - 2\sigma_1 \rho A \langle \delta_t u_l^n, \delta_{t-} \delta_{xx} u_l^n \rangle_d, \quad (4.29)$$

and the boundary term \mathfrak{b} appears after rewriting Equation (4.27) using summation by parts (see Section 3.2.2). Specifically, using Eq. (3.20a) for the second term and Eq. (3.22b) for the third, we get

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

where the boundary term becomes

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

For the clamped and simply supported boundary conditions in (4.14a) and (4.14b) it can easily be shown that $\mathfrak{b} = 0$. If free conditions as in Eq. (4.14c) are used, the boundary conditions will vanish when the primed inner product in Eq. (3.16) is used in Step 1 and identity (3.22c) when performing summation by parts. Here, we continue with the clamped / simply supported case.

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

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

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

$$\begin{aligned} \mathfrak{h} &= \mathfrak{t} + \mathfrak{v}, \quad \text{with} \quad \mathfrak{t} = \frac{\rho A}{2} \|\delta_{t-} u_l^n\|_d^2, \text{ and} \\ \mathfrak{v} &= \frac{T}{2} \langle \delta_{x+} u_l^n, e_{t-} \delta_{x+} u_l^n \rangle_{\underline{d}} + \frac{EI}{2} \langle \delta_{xx} u_l^n, e_{t-} \delta_{xx} u_l^n \rangle_{\underline{d}}. \end{aligned} \tag{4.30}$$

Step 3: Check units

Comparing the acquired energy balance in Eq. (4.30) to the energy balance for the 1D wave equation in Eq. (3.53), one can observe that the balances are nearly identical, the only difference being the second term in the definition for \mathfrak{v} in Eq. (4.30). Writing this term out in units, and recalling that Pa (the unit for E) in SI units is $\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}$, yields

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

and indeed has the correct units.

The damping terms in \mathfrak{q} need to be in Joules per second, or $\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-3}$.

4.5. Modal Analysis

Writing the terms in Eq. (4.29) out in their units yields

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

and also have the correct units.

Step 4: Implementation

Matrix form...

4.5 Modal Analysis

To be able to perform a modal analysis on the FD scheme in (4.5), it must be written in one-step form – introduced in Section 3.5.1 – due to the damping present in the system. Using with the matrix form of the damped stiff string in Eq. (4.20), it can be written as

$$\underbrace{\begin{bmatrix} \mathbf{u}^{n+1} \\ \mathbf{u}^n \end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix} \mathbf{B}/A & \mathbf{C}/A \\ \mathbf{I} & \mathbf{0} \end{bmatrix}}_{\mathbf{Q}} \underbrace{\begin{bmatrix} \mathbf{u}^n \\ \mathbf{u}^{n-1} \end{bmatrix}}_{\mathbf{w}^n} \quad (4.31)$$

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

$$z\phi = \mathbf{Q}\phi, \quad (4.32)$$

which has the following solutions

$$s_p = \frac{1}{k} \ln \left(\text{eig}_p(\mathbf{Q}) \right). \quad (4.33)$$

The (angular) frequency of the p^{th} mode can then be obtained using $\Im(s_p)$ and the damping per mode as $\Re(s_p)$. Only selecting the non-negative frequencies obtained from $\Im(s_p)$, these can be plotted and are shown in Figure ??.

The parameters used are the ones found in Table 4.1 with $T = 13000$ N, $f_s = 44100$.

make figure

not done...

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

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_t u_l^n + 2\sigma_1\delta_t\delta_{xx}u_l^n. \quad (4.34)$$

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 4.3 also shows this: in order to calculate u_l^{n+1} , neighbouring points at the next time step u_{l+1}^{n+1} and u_{l-1}^{n+1} are needed. The issue is that these values are unknown at the time of calculation.

Luckily, as the scheme is linear, it can be treated as a system of linear equations and solved following the technique described in Section 3.1.4. The drawback is that this requires one matrix inversion per iteration which can be extremely costly (see Section 12.4.1). However, both von Neumann and modal analysis (below) show that using the centred instead of the backwards operator has a positive effect on the stability and the modal behaviour of the scheme.

[Left off here!!](#) taking simply supported boundary conditions such that $l \in \{1, \dots, N-1\}$ are $N-1$ unknowns that can be calculated using $N-1$ equations. The column vector $\mathbf{u}^n = [u_1^n, u_2^n, \dots, u_{N-1}^n]$

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

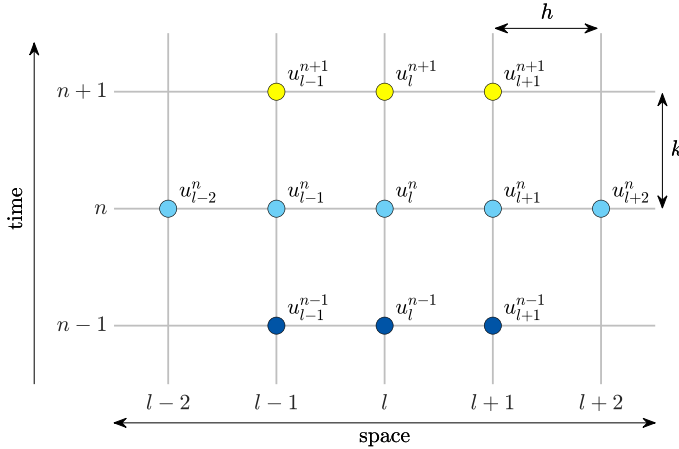


Fig. 4.3: The stencil for the damped stiff string scheme in (4.34).

4.6.1 von Neumann analysis

Following Section 3.3..

below is the exact same wording as above

$$\begin{aligned} \frac{1}{k^2} (z - 2 + z^{-1}) = & -\frac{4c^2}{h^2} \sin^2(\beta h/2) - \frac{16\kappa^2}{h^4} \sin^4(\beta h/2) - \frac{\sigma_0}{k} z + \frac{\sigma_0}{k} z^{-1} \\ & - \frac{4\sigma_1}{kh^2} \sin^2(\beta h/2) z + \frac{4\sigma_1}{kh^2} \sin^2(\beta h/2) z^{-1} \end{aligned} \quad (4.36)$$

and after collecting the terms, the characteristic equation follows

$$\begin{aligned} \left(1 + \sigma_0 k + \frac{4\sigma_1 k}{h^2} \sin^2(\beta h/2)\right) z + (16\mu^2 \sin^4(\beta h/2) + 4\lambda^2 \sin^2(\beta h/2) - 2) \\ + \left(1 - \sigma_0 k - \frac{4\sigma_1 k}{h^2} \sin^2(\beta h/2)\right) z^{-1} = 0. \end{aligned} \quad (4.37)$$

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

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

stability of the system can be proven using condition (3.32). Continuing with $\mathcal{S} = \sin^2(\beta h/2)$ and substituting the coefficients into this condition yields

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

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:

4.6.2 Modal analysis

As Eq. (4.35) matches the form in Eq. (3.63), one can perform a modal analysis by writing the scheme in one-step form as explained in Section 3.5.1

Chapter 4. Stiff string

Chapter 5

Brass

5.1 Second-order system

Acoustic potential...

5.2 First-order system

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

Chapter 6

2D Systems

In this work, it is mainly used to model a simplified body in papers...

State variable $u = u(x, y, t)$ where $t \geq 0$ and $(x, y) \in \mathcal{D}$ where \mathcal{D} is 2-dimensional. The state variable can then be discretised according to $u(x, y, t) \approx u_{l,m}^n$ with space $x = lh$ and $y = mh$ and time $t = nk$ and $k = 1/f_s$. For simplicity the grid spacing in both the x and y directions are set to be the same but could be different.

In continuous time the operators:

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

The same shift operators as defined in Chapter 2 can be applied to grid function $u_{l,m}^n$. Additional ones are

$$e_{y+} u_{l,m}^n = u_{l,m+1}^n, \quad \text{and} \quad e_{y-} u_{l,m}^n = u_{l,m-1}^n. \quad (6.2)$$

6.1 Analysis Techniques in 2D

Here, some of the differences between the analysis techniques presented in Chapter 3 and those in 2D will be elaborated on. Although, modal analysis remains the same

6.1.1 Frequency Domain Analysis

$p_x + p_y$
ansatz:

$$u_{l,m}^n = z^n e^{jh(l\beta_x + m\beta_y)} \quad (6.3)$$

6.1.2 Energy Analysis

Squared domain $d \in \{0, \dots, N_x\} \times \{0, \dots, N_y\}$

$$\langle f^n, g^n \rangle_d = \sum_{l=0}^{N_x} \sum_{m=0}^{N_y} h^2 f_{l,m}^n g_{l,m}^n \quad (6.4)$$

6.1.3 Modal Analysis

Stacked matrix form

6.2 2D Wave Equation

The 2D wave equation be used to model an ideal membrane such as done in

It has identical behaviour to the 2D waveguide mesh presented by van Duyn and Smith [47].

Consider a square ideal membrane with side lengths L_x and L_y (both in m) and its transverse displacement described by $u = u(x, y, t)$. The membrane is defined over $(x, y) \in \mathcal{D}$ with domain $\mathcal{D} = [0, L_x] \times [0, L_y]$ and its motion is described by the following PDE

$$\partial_t^2 u = c^2 \Delta u \quad (6.5)$$

with wave speed $c = \sqrt{T/\rho H}$ (in m/s), tension per unit length (applied to the boundary)/ T (in N/m), material density ρ (in kg/m³) and thickness H .

check whether this is right..

6.3 Thin plate

Used in [A], [B], [D] and [E] biharmonic operator, Laplacian in (6.1) applied to itself.

$$\rho H \partial_t^2 u = -D \Delta \Delta u \quad (6.6)$$

where $D = EH^3/12(1 - \nu^2)$

6.4 Stiff membrane

Combination between Eqs. (6.7) and (6.6)

$$\rho H \partial_t^2 u = T \Delta u - D \Delta \Delta u \quad (6.7)$$

[F]

Part III

Exciters

Exciters

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

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

Chapter 7

Unmodelled Excitations

Different title
here?

7.1 Initial conditions

Hammer

Full raised cosine

Pluck

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

7.2 Signals

7.2.1 Pulse train

For brass

7.2.2 Noise

Noise input

Chapter 8

Modelled Excitations

8.1 Preamble: Newton-Raphson

Iterative root-finding method that finds answers to nonlinear equations.

8.2 Hammer **but maybe not as I didn't use it..**

Hammer modelling

8.2.1 Mass-spring Systems Revisited: Adding Damping

Damping can be added to Eq. (2.28)

$$M\ddot{u} = -Ku - R\dot{u} \quad (8.1)$$

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

$$M\ddot{u} = -Ku - R\dot{u} + F \quad (8.2)$$

where

8.3 The Bow

The bow

Excites the system with a force due to friction, which has a nonlinear relationship to the relative velocity between the bow and the string.

Helmholtz motion..

coined as 'stick-slip' motion by Bowden and Leben in 1939 [9]

See fx. <https://www.youtube.com/watch?v=6JeyiM0YNo4>

Characteristic triangular motion (wave shapes?)

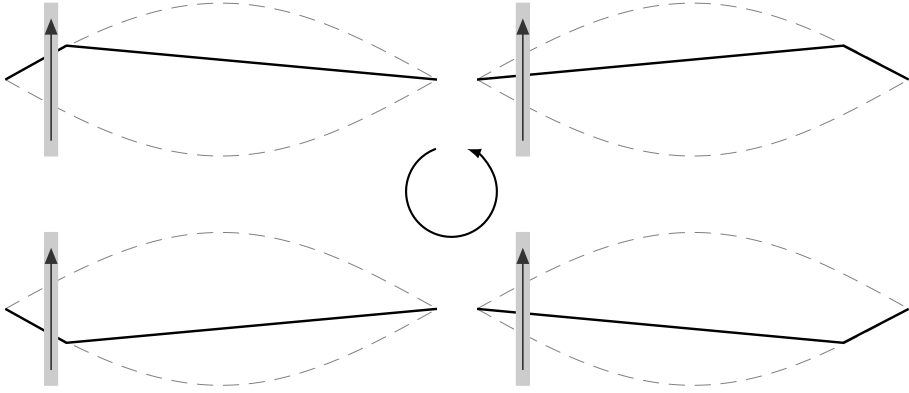


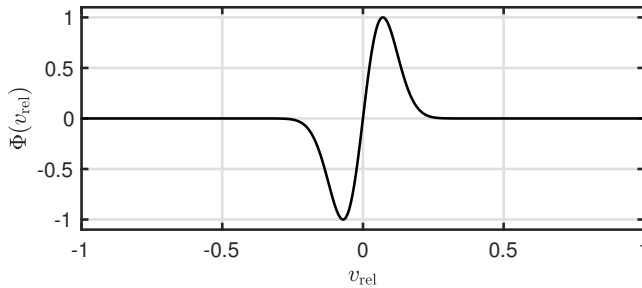
Fig. 8.1: Helmholtz motion.

8.3.1 Static Friction Models

In static bow-string-interaction models, the friction force is defined as a function of the relative velocity between the bow and the string only. The first mathematical description of friction was proposed by Coulomb in 1773 [15] to which static friction, or *stiction*, was added by Morin in 1833 [34] and viscous friction, or velocity-dependent friction, by Reynolds in 1886 [37]. In 1902, Stribeck found a smooth transition between the static and the coulomb part of the friction curve now referred to as the Stribeck effect [43]. The latter is still the standard for static friction models today.

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

$$\Phi(v_{\text{rel}}) = \sqrt{2a}v_{\text{rel}}e^{-av_{\text{rel}}^2+1/2} \quad (8.3)$$

Fig. 8.2: The friction characteristic in (8.3) with $a = 100$.

8.3.2 Dynamic Friction Models

As opposed to less complex bow models, such as the hyperbolic [source] and exponential [source] models, the elasto-plastic bow model assumes that the friction between the bow and the string is caused by a large quantity of bristles, each of which contributes to the total amount of friction.

Dynamic friction models exhibit / model *hysteresis*: the dependence of a system on its history.

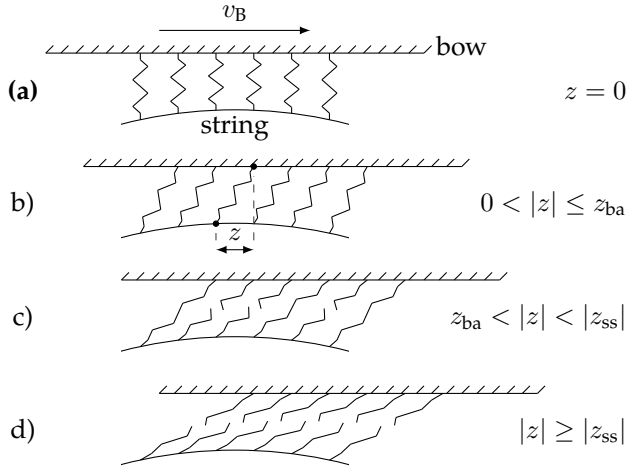


Fig. 8.3: Microscopic displacements of the bristles between the bow and the string. The bow moves right with a velocity of v_B . (a) The initial state is where the average bristle displacement $z = 0$. (b) The bow has moved right relative to the string. The purely elastic, or presliding regime is entered (stick). (c) After the break-away displacement z_{ba} , more and more bristles start to 'break'. This is defined as the elasto-plastic regime. (d) After all bristles have 'broken', the steady state (slip) is reached and the purely plastic regime is entered. (Taken from [C].) *exactly the same caption as paper*

8.4 Lip-reed

Lip-reed model

Coupling to Tube

Part IV

Interactions

Interactions

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

ε

Newtons third law (action reaction)

Subscripts are needed

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

Interpolation and spreading operators

Using $l_c = \lfloor x_c/h \rfloor$ and $\alpha_c = x_c/h - l_c$ is the fractional part the location of interest.

$$I_0(x_c) = \begin{cases} 1, & \text{if } l = l_c, \\ 0, & \text{otherwise} \end{cases} \quad (8.4)$$

$$I_1(x_c) = \begin{cases} (1 - \alpha_c), & \text{if } l = l_c \\ \alpha_c, & \text{if } l = l_c + 1 \\ 0 & \text{otherwise} \end{cases} \quad (8.5)$$

$$I_3(x_c) = \left\{ \dots \right. \quad (8.6)$$

The following identity is very useful when solving interactions between components:

$$\langle f, J_p(x_c) \rangle_{\mathcal{D}} = I_p(x_c) f \quad (8.7)$$

Chapter 9

Connections

Something about connections

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

Alternative interpretation of grid points

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

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

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

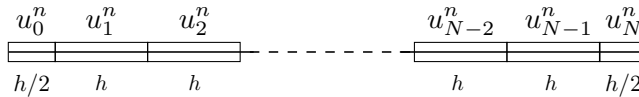


Fig. 9.1: Alternative interpretation of the discretisation of $u(x, t)$. The continuous system is divided into $N - 1$ sections of width h plus 2 sections of width $h/2$ at the boundaries. Through this interpretation, the ‘weight’ of a grid point can be calculated from its physical parameters.

9.1 Rigid connection

The simplest connection is Forces should be equal and opposite.

If component a is located ‘above’ component b , and their relative displacement is defined as $\eta = a - b$, then a positive η is going to have a negative effect on a and a positive effect on b and vice-versa. This is important for the signs when adding the force terms to the schemes.

9.2 Spring-like connections

9.2.1 Connection with rigid barrier (scaled)

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

$$\delta_{tt}u_l^n = \gamma^2 \delta_{xx}u_l^n + J(x_c)F^n \quad (9.1)$$

where

$$F^n = -\omega_0^2 \mu_t \eta^n - \omega_1^4 (\eta^n)^2 \mu_t \eta^n - 2\sigma_\times \delta_t \eta^n \quad (9.2)$$

and

$$\eta^n = I(x_c)u_l^n. \quad (9.3)$$

To obtain F^n , an inner product of scheme (9.1) needs to be taken with $J(x_c)$ over domain \mathcal{D} which, using identity (8.7) yields

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

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

$$\delta_{tt}\eta^n = \gamma^2 I(x_c)\delta_{xx}u_l^n + \|J(x_c)\|_{\mathcal{D}}^2 \left(-\omega_0^2 \mu_t \eta^n - \omega_1^4 (\eta^n)^2 \mu_t \eta^n - 2\sigma_\times \delta_t \eta^n \right). \quad (9.5)$$

and solved for η^{n+1} :

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

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

9.2.2 String-plate connection

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

Continuous

The systems in isolation are as in (4.2) and (6.6), but with an added force term:

$$\partial_t^2 u = c^2 \partial_x^2 u - \kappa_s^2 \partial_x^4 u - 2\sigma_{0,s} \partial_t u + 2\sigma_{1,s} \partial_t \partial_x^2 u - \delta(x - x_c) \frac{f}{\rho_s A} \quad (9.7a)$$

$$\partial_t^2 w = -\kappa_p^2 \Delta \Delta w - 2\sigma_{0,p} \partial_t w + 2\sigma_{1,p} \partial_t \partial_x^2 w + \delta(x - x_c, y - y_c) \frac{f}{\rho_p H} \quad (9.7b)$$

where

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

and

$$\eta = \eta(t) = u(x_c, t) - w(x_c, y_c, t) \quad (9.9)$$

Discrete

System (9.7) can then be discretised as

$$\delta_{tt} u_l^n = c^2 \delta_{xx} u_l^n - \kappa_s^2 \delta_{xxxx} u_l^n - 2\sigma_{0,s} \delta_t \cdot u_l^n + 2\sigma_{1,s} \delta_{t-} \delta_{xx} u_l^n - J_s(x_c) \frac{f^n}{\rho_s A}, \quad (9.10)$$

$$\delta_{tt} w_l^n = -\kappa_p^2 \delta_{\Delta} \delta_{\Delta} w_l^n - 2\sigma_{0,p} \delta_t \cdot w_l^n + 2\sigma_{1,p} \delta_{t-} \delta_{xx} w_l^n + J_p(x_c, y_c) \frac{f^n}{\rho_p H}, \quad (9.11)$$

where

$$f^n = K_1 \mu_{tt} \eta^n + K_3 (\eta^n)^2 \mu_t \cdot \eta^n + R \delta_t \cdot \eta^n, \quad (9.12)$$

and

$$\eta^n = I(x_c) u_l^n - I(x_c, y_c) w_l^n. \quad (9.13)$$

Expansion

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

9.2.3 Solving for f

9.2.4 Non-dimensional

The scaled system can be written as:

$$\partial_t^2 u = \gamma^2 \partial_x^2 u - \kappa_s^2 \partial_x^4 u - 2\sigma_{0,s} \partial_t u + 2\sigma_{1,s} \partial_t \partial_x^2 u - \delta(x - x_c) F \quad (9.14)$$

$$\partial_t w = -\kappa_p^2 \Delta \Delta w - 2\sigma_{0,p} \partial_t w + 2\sigma_{1,p} \partial_t \partial_x^2 w + \delta(x - x_c, y - y_c) F \quad (9.15)$$

where

$$F = F(t) = \omega_1^2 \eta + \omega_3^4 \eta^3 + \sigma_c \dot{\eta} \quad (9.16)$$

and

$$\eta = \eta(t) = u(x_c, t) - w(x_c, y_c, t) \quad (9.17)$$

$$I(x_c) \delta_{tt} u_l^n = c^2 (I(x_c) \delta_{xx} u_l^n) + I(x_c) J(x_c) F \quad (9.18)$$

Chapter 10

Collisions

Something about collisions

10.1 Classic models

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

10.2 Michele's tricks

Chapter 10. Collisions

Part V

Dynamic Grid

Chapter 11

Dynamic Grid

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

11.1 Background and Motivation

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

check if is still true

Tension in a string is changed when tuning it

Some artists even use this in their performances [22, 32]

The hammered dulcimer is another example where the strings are tensioned over a bridge where one can play the string at one side of the bridge, while pushing down on the same string on the other side [21].

1D:

- Trombone
- Slide whistle
- Guitar strings

- Fretting finger pitch bend
- Above the nut [32]
- Tuning pegs directly [22]
- Hammered dulcimer [21]
- Erhu?

2D:

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

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

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

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

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

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

11.2 Method

Iterations have been:

- Interpolated boundary conditions
- Linear interpolation

In this appendix, some iterations done over the course of this project will be shown in more detail. In the following, the 1D wave equation with a wave speed of $c = 1470$ m/s, a length of $L = 1$ m, Dirichlet boundary conditions and a sample rate of $f_s = 44100$ Hz is considered, and – through Eq. (??) – satisfies the CFL condition with equality. These values result in $N = 30$, or a

These sections are taken from the JASA appendix

grid of 31 points including the boundaries. Then, the wave speed is decreased to $c \approx 1422.6$ m/s, i.e., the wave speed that results in $N = 31$ and satisfies the stability condition with equality again.

11.2.1 Full-Grid Interpolation

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

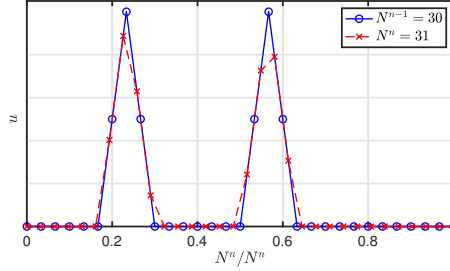


Fig. 11.1: Upsampling u (with an arbitrary state) using (linear) full-grid interpolation with $N^{n-1} = 30$ and $N^n = 31$. The horizontal axis is normalised with respect to N^n .

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

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

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

11.2.2 Adding and removing Points at the Boundary

To solve the issues exhibited by a full-grid interpolation, points can be added and removed at a single location and leave most points unaffected by the parameter changes. A good candidate for a location to do this is at a fixed

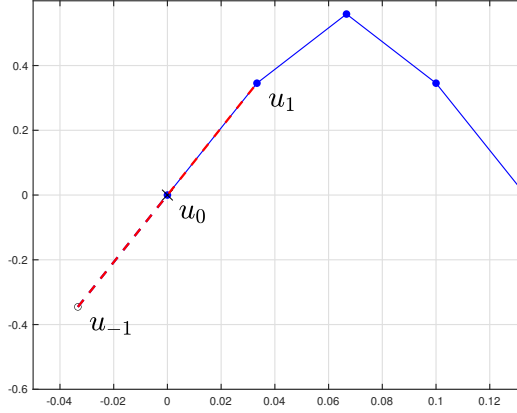


Fig. 11.2: The simply supported boundary condition: both the state and the curvature at the boundary – at $l = 0$ – should be 0.

(Dirichlet) boundary. The state u at this location is always 0 so points can be added smoothly.

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

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

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

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

or, when discretised,

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

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

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

This is visualised in Figure 11.2.

11.2. Method

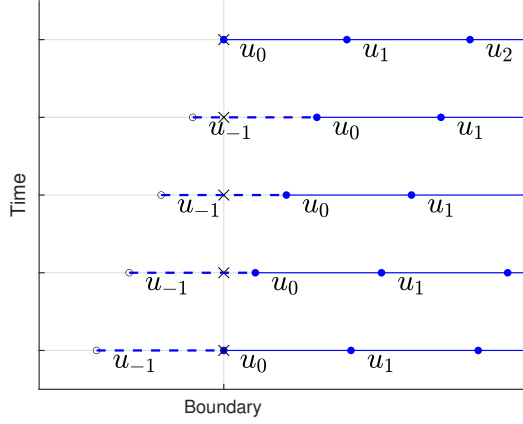


Fig. 11.3: The grid changing over time

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

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

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

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

11.2.3 Cubic interpolation

11.2.4 Sinc interpolation

11.2.5 Displacement correction

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

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

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

with $K = K(\alpha)$

$$K = (1 - \alpha)^\epsilon. \quad (11.6)$$

11.3 Analysis and Experiments

11.3.1 Interpolation technique

11.3.2 Interpolation range

11.3.3 Location

... where to add and remove points

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

11.4 Discussion and Conclusion

Part VI

Real-Time Implementation and Control

Real-Time Implementation and Control

It's all fun and dandy with all these physical models, but what use are they if you can't control them in real time?!

Chapter 12

Real-Time Implementation

JUCE Give overall structure of code

Implementation of the physical models using FDTD methods

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

Usually, MATLAB is used for simulating

Here, an interactive application is considered real-time when

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

For human-computer interaction, the task at hand greatly determines how much latency is acceptable. Wessel and Wright [50] place the upper limit of latency when interacting with computers for musical purposes at 10 ms. Moreover, they place the a limit on the *jitter*, or of variation of the latency, at 1 ms. *It is thus important to keep the CPU usage at a fixed level, and different ways of controlling and interacting with the application should not influence the amount of computations much...*

Also the application needs to be controlled continuously

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

Refer to [49] somewhere here

12.1 MATLAB vs. C++

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

- Easier to debug

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

12.1.1 Speed

Here is where the power of C++

12.1.2 Syntax

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

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

Some of the things I learned (the hard way)...

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

Create a limiter

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

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

```

    return val;
}

```

```

1 for i = 0:lengthSound
2     uNext = ...
3 end

```

Use pointer switches

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

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

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

```

1 for n = 1:lengthSound
2     ...
3     uPrev = u;
4     u = uNext;
5 end

```

In C++ this is done using

```

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

```

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

Grouping terms and Precalculating coefficients

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

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

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

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

The undamped stiff string FD scheme

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxx}u_l^n, \quad (12.1)$$

can be expanded to an update equation as

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

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

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

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

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

```

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

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

```


}

Algorithm 12.2: Precalculation

12.3 Graphics

```

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

    Path stringState = visualiseState (g, 100);

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

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

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

    // initialise path
    Path stringPath;

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

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

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

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

        stringPath.lineTo (x, newY);
    }
}

```

Chapter 12. Real-Time Implementation

```
        x += spacing;
    }

    return stringPath;
}
```

```

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

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

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

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

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

```

12.4 Matrices

Library called *Eigen* [23]

12.4.1 Matrix Inversions in Real-Time

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

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

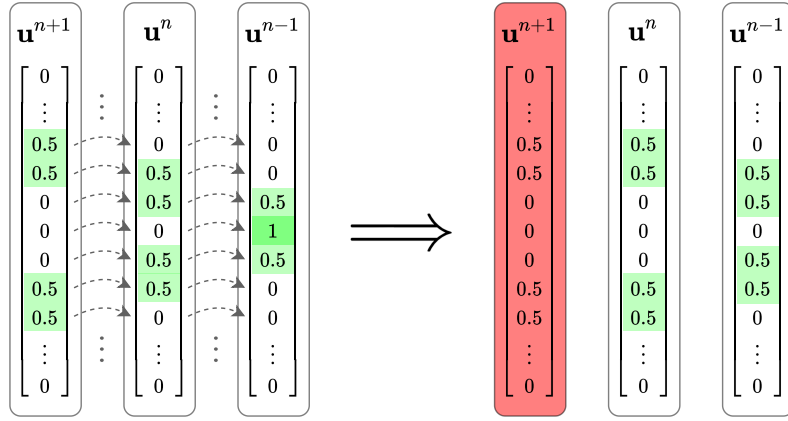
where

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

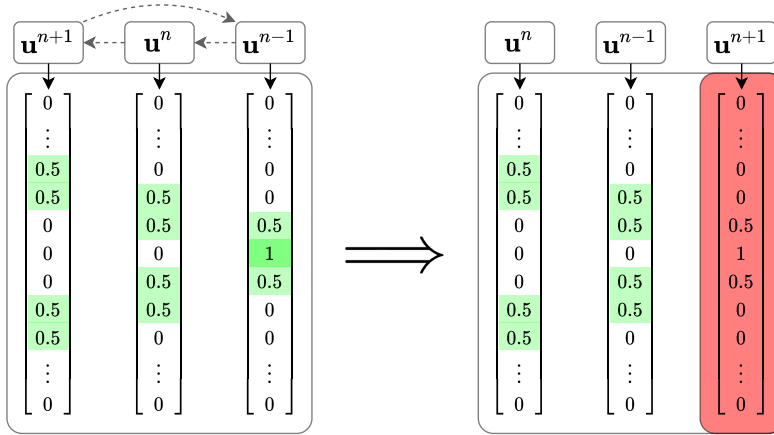
```

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

```



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



(b) Pointer switch: 4 operations per iteration.

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

Chapter 13

Control

13.1 Sensel Morph

150 Hz

13.2 Phantom OMNI

Part VII

Complete Instruments

Complete Instruments

This part will give several examples of full instrument models that have been developed during the PhD. Chapter 14 shows a three instrument-inspired case-studies using a large-scale modular environment, Chapter 15 describes the implementation of the tromba marina and Chapter 16 that of the trombone.

Chapter 14

Large Scale Modular Physical models

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

14.1 Bowed Sitar

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

14.2 Dulcimer

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

14.3 Hurdy Gurdy

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

Chapter 15

Tromba Marina

15.1 Introduction

15.2 Physical Model

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

$$\mathcal{L}q = 0 \quad (15.1)$$

where $q(x, t)$

15.2.1 Continuous

$$\mathcal{L}_s = \rho_s A \partial_t^2 - T \partial_x^2 + E_s I \partial_x^4 + 2\rho_s A \sigma_{0s} \partial_t - 2\rho_s A \sigma_{1s} \partial_t \partial_x^2. \quad (15.2)$$

Complete system

Test

15.2.2 Discrete

15.3 Real-Time Implementation

15.3.1 Control using Sensel Morph

15.3.2 VR Application

Chapter 15. Tromba Marina

Chapter 16

Trombone

Published in [H]

16.1 Introduction

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

16.2 Physical Model

Most has been described in Chapter 5

16.2.1 Continuous

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

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

$$\frac{S}{\rho_0 c^2} \partial_t p = -\partial_x (Sv) \quad (16.1a)$$

$$\rho_0 \partial_t v = -\partial_x p \quad (16.1b)$$

with acoustic pressure $p = p(x, t)$ (in N/m^2), particle velocity $v = v(x, t)$ (in m/s) and (circular) cross-sectional area $S(x)$ (in m^2). Furthermore, ρ_0 is the density of air (in kg/m^3) and c is the speed of sound in air (in m/s). System (16.1) can be condensed into a second-order equation in p alone, often referred to as Webster's equation [?]. Interesting! In NSS it is the acoustic potential right? Can you go from that to a second-order PDE in p ? There is a time-derivative hidden there somewhere right? (Just wondering :) Yes, the form

in p alone is the one you usually see. You get it by differentiating the first equation, giving you a \dot{v} on the RHS, and then you can substitute the second equation in...I used the velocity potential one because it has direct energy balance properties. Right. So Webster's eq. in p and Ψ are identical (will exhibit identical behaviour), except for the unit of the state variable..?yes that's right...using the velocity potential allows you to do all the energy analysis easily, in terms of physical impedances. But the scheme you get to in the end is the same, just one derivative down. Alright cool! Thanks for the explanation :) For simplicity, effects of viscothermal losses have been neglected in (16.1). For a full time domain model of such effects in an acoustic tube, see, e.g. [?].

16.2.2 Discrete

16.3 Real-Time Implementation

Unity??

16.4 Discussion

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

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

Part VIII

Conclusions and Perspectives

Chapter 17

Conclusions and Perspectives

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

There is always more work to be done

Parameter design

Many parameters that can always be improved


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

Dynamic grid

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

Chapter 17. Conclusions and Perspectives

check whether
all references
are used



References

- [1] 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.
- [2] S. Bilbao, "A modular percussion synthesis environment," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, 2009.
- [3] —, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. John Wiley & Sons, 2009.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] G. Borin, G. De Poli, and A. Sarti, "A modular approach to excitor-resonator interaction in physical models syntheses," *Proceedings of the International Computer Music Conference*, 1989.
- [9] F. P. Bowden and L. Leben, "The nature of sliding and the analysis of friction," in *Proceedings of the Royal Society of London*, vol. 169, 1939.

References

- [10] C. Cadoz, "Synthèse sonore par simulation de mécanismes vibratoires," 1979, thèse de Docteur Ingénieur, I.N.P.G. Grenoble, France.
- [11] C. Cadoz, A. Luciani, and J.-L. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1983.
- [12] —, "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.
- [13] 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.
- [14] 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.
- [15] C. A. Coulomb, "Sur une application des règles de maximis et minimis à quelques problèmes de statique, relatifs à l'architecture [On maximums and minimums of rules to some static problems relating to architecture]," *Mémoires de Mathématique de l'Académie Royale de Science*, vol. 7, pp. 343–382, 1773.
- [16] R. Courant, K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen de mathematischen physik," *Mathematische Annalen*, vol. 100, pp. 32–74, 1928.
- [17] 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.
- [18] C. G. M. Desvages, "Physical modelling of the bowed string and applications to sound synthesis," Ph.D. dissertation, The University of Edinburgh, 2018.
- [19] C. Erkut and M. Karjalainen, "Finite difference method vs. digital waveguide method in string instrument modeling and synthesis," *International Symposium on Musical Acoustics*, 2002.
- [20] N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments*. Springer, 1998.
- [21] T. L. Glenn, "Amazing hammered dulcimer musician - joshua messick," 2014. [Online]. Available: <https://youtu.be/veuGTnzgNRU?t=215>

- [22] J. Gomm, "Passionflower," 2011. [Online]. Available: <https://www.youtube.com/watch?v=nY7GnAq6Znw>
- [23] G. Guennebaud, B. Jacob *et al.*, "Eigen," 2021. [Online]. Available: <http://eigen.tuxfamily.org>
- [24] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time-Dependent Problems and Difference Methods*, 2nd ed. John Wiley & Sons, 2013.
- [25] B. Hamilton, "Finite difference and finite volume methods for wave-based modelling of room acoustics," Ph.D. dissertation, The University of Edinburgh, 2016.
- [26] R. L. Harrison-Harsley, "Physical modelling of brass instruments using finite-difference time-domain methods," Ph.D. dissertation, University of Edinburgh, 2018.
- [27] 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.
- [28] —, "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.
- [29] 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.
- [30] E. Maestre, C. Spa, and J. Smith, "A bowed string physical model including finite-width thermal friction and hair dynamics," *Proceedings ICMC | SMC | 2014*, pp. 1305–1311, 2014.
- [31] D. Matignon, P. Depalle, and X. Rodet, "State space models for wind-instrument synthesis," *ICMC*, 1992.
- [32] J. Mayer, "Gravity (Live in L.A.)," 2008. [Online]. Available: <https://youtu.be/dBFW8OvciIU?t=284>
- [33] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.
- [34] A. Morin, "New friction experiments carried out at metz in 1831-1833," in *Proceedings of the French Royal Academy of Sciences*, 1833, pp. 1–128.
- [35] J. Morrison and J.-M. Adrien, "Mosaic: A framework for modal synthesis," *Computer Music Journal*, vol. 17, no. 1, pp. 45–56, 1993.

References

- [36] T. H. Park, *Introduction To Digital Signal Processing: Computer Musically Speaking*. World Scientific Publishing Co. Pte. Ltd, 2010.
- [37] O. Reynolds, "On the theory of lubrication and its application to Mr. Beauchamp tower's experiments, including an experimental determination of the viscosity of olive oil," in *Proceedings of the Royal Society of London*, vol. 40, 1886, pp. 191–203.
- [38] J. O. S. Romain Michon, Sara Martin, "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.
- [39] P. Ruiz, "A technique for simulating the vibrations of strings with a digital computer," Master's thesis, University of Illinois, 1969.
- [40] J. O. Smith, "Virtual acoustic musical instruments: Review and update," *Center for computer research in music and acoustics (CCRMA)*, 2010.
- [41] J. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.
- [42] J. O. Smith, "Physical audio signal processing (online book)," 2010. [Online]. Available: <http://ccrma.stanford.edu/~jos/pasp/>
- [43] R. Stribeck, "Die wesentlichen eigenschaften der gleit- und rollenlager [The essential properties of sliding and roller bearings]," *Zeitschrift des Vereins Deutscher Ingenieure*, vol. 46, no. (38,39), pp. 1342–48, 1432–37, 1902.
- [44] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Pacific Grove, California: Wadsworth and Brooks/Cole Advanced Books and Software, 1989.
- [45] L. Trautmann and R. Rabenstein, *Digital Sound Synthesis by Physical Modeling Using the Functional Transformation Method*. Springer-Science + Business Media LLC, 2003.
- [46] V. Välimäki, J. Pakarinen, C. Erku, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Institute of Physics Publishing*, 2006.
- [47] S. A. van Duyne and J. O. Smith, "Physical modeling with the 2-D digital waveguide mesh," in *ICMC Proceedings*, 1993.
- [48] 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.

References

- [49] C. Webb and S. Bilbao, "On the limits of real-time physical modelling synthesis with a modular environment," in *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx)*, 2015.
- [50] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, vol. 26, no. 3, 2002.
- [51] R. Zucker, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards Applied Mathematics Series 55, 1972, ch. 4: Elementary Transcendental Functions, pp. 65–226, Tenth Printing.

References

Part IX

Appendix

Appendix A

List of Symbols

The list of symbols found below contains often-used symbols in the thesis in the context that they are normally used. Depending on the context they might carry a different meaning (y being displacement of the lip-reed in Chapter 16 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	-
$\delta_{t+}, \delta_{t-}, \delta_t.$	Forward, backward and centred difference in time operator	-
$\delta_{x+}, \delta_{x-}, \delta_x.$	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	-
$\mu_{t+}, \mu_{t-}, \mu_t.$	Forward, backward and centred average in time operator	-
$\mu_{x+}, \mu_{x-}, \mu_x.$	Forward, backward and centred average in space operator	-
μ_{tt}	Second order average in time operator	-
μ_{xx}	Second order average in space operator	-

Appendix A. List of Symbols

Symbol	Description	Unit
E	Young's Modulus	Pa ($\text{kg}\cdot\text{m}^{-1}\cdot\text{s}^{-2}$)
f	Force	N
f	Frequency	Hz
f_s	Sample rate	Hz
F	Scaled force	depends on system
h	Grid spacing	m
H	Membrane / Plate thickness	m
I	Area moment of inertia	m^4
l	Spatial index to grid function	-
L	Length	m
k	Time step ($= 1/f_s$)	s
K	Spring coefficient	N/m
κ	Stiffness coefficient	m^2/s (1D) $\text{m}^4\cdot\text{s}^{-2}$ (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	$\text{kg}\cdot\text{m}^{-3}$
Operations		

Symbol	Description	Unit
$\Im(\cdot)$	Imaginary part of	
$\Re(\cdot)$	Real part of	
$\lfloor \cdot \rfloor$	Flooring operation	
$\lceil \cdot \rceil$	Ceiling operation	

Subscripts

c	Connection
p	Plate
s	String

Appendix A. List of Symbols

Appendix B

List of Abbreviations

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

Appendix B. List of Abbreviations

Appendix C

Code Snippets

C.1 Mass-Spring System (Section 2.3)

```
1 %% Initialise variables
2 fs = 44100;           % sample rate [Hz]
3 k = 1 / fs;           % time step [s]
4 lengthSound = fs;     % length of the simulation (1 second) [samples]
5
6 f0 = 440;             % fundamental frequency [Hz]
7 omega0 = 2 * pi * f0; % angular (fundamental) frequency [Hz]
8 M = 1;               % mass [kg]
9 K = omega0^2 * M;     % spring constant [N/m]
10
11 %% initial conditions (u0 = 1, d/dt u0 = 0)
12 u = 1;
13 uPrev = 1;
14
15 % initialise output vector
16 out = zeros(lengthSound, 1);
17
18 %% Simulation loop
19 for n = 1:lengthSound
20
21     % Update equation Eq. (2.35)
22     uNext = (2 - K * k^2 / M) * u - uPrev;
23
24     out(n) = u;
25
26     % Update system states
27     uPrev = u;
28     u = uNext;
29 end
```

C.2 1D Wave Equation (Section 2.4)

```

1 %% Initialise variables
2 fs = 44100;           % Sample rate [Hz]
3 k = 1 / fs;           % Time step [s]
4 lengthSound = fs;     % Length of the simulation (1 second) [samples]
5
6 c = 300;              % Wave speed [m/s]
7 L = 1;                % Length [m]
8 h = c * k;            % Grid spacing [m] (from CFL condition)
9 N = floor(L/h);       % Number of intervals between grid points
10 h = L / N;           % Recalculation of grid spacing based on integer N
11
12 lambdaSq = c^2 * k^2 / h^2; % Courant number squared
13
14 % Boundary conditions ([D]irichlet or [N]eumann)
15 bcLeft = "D";
16 bcRight = "D";
17
18 %% Initialise state vectors (one more grid point than the number of
   intervals)
19 uNext = zeros(N+1, 1);
20 u = zeros(N+1, 1);
21
22 %% Initial conditions (raised cosine)
23 loc = round(0.8 * N); % Center location
24 halfWidth = round(N/10); % Half-width of raised cosine
25 width = 2 * halfWidth; % Full width
26 rcX = 0:width;         % x-locations for raised cosine
27
28 rc = 0.5 - 0.5 * cos(2 * pi * rcX / width); % raised cosine
29 u(loc-halfWidth : loc+halfWidth) = rc; % initialise current state
30
31 % Set initial velocity to zero
32 uPrev = u;
33
34 % Range of calculation
35 range = 2:N;
36
37 % Output location
38 outLoc = round(0.3 * N);
39
40 %% Simulation loop
41 for n = 1:lengthSound
42
43     % Update equation Eq. (2.44)
44     uNext(range) = (2 - 2 * lambdaSq) * u(range) ...
45         + lambdaSq * (u(range+1) + u(range-1)) - uPrev(range);
46
47     % boundary updates Eq. (2.49)
48     if bcLeft == "N"
49         uNext(1) = (2 - 2 * lambdaSq) * u(1) - uPrev(1) ...
50             + 2 * lambdaSq * u(2);

```

C.2. 1D Wave Equation (Section 2.4)

```
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
```

Appendix C. Code Snippets

Part X

Papers

Paper Errata

Here, some errors in the published papers will be listed:

Real-Time Tromba [D]

- The minus sign in Eq. (28) (and thus Eqs. (31) and (35)) should be a plus sign.
- $\sigma_{1,s}$ in Eq. (21) should obviously be $\sigma_{1,p}$
- 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:

$$h \geq \sqrt{c^2 k^2 + 4\sigma_1 k + \sqrt{(c^2 k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}} \quad (1)$$

- Unit for membrane tension is N/m.

Dynamic grids [G]

- Reference in intro for ‘recently gained popularity’ should go to [4]
Note: not really an error, but should be changed before resubmission

Paper A

Real-Time Control of Large-Scale Modular Physical Models using the Sensel Morph

Silvin Willemsen, Nikolaj Andersson, Stefania Serafin
and Stefan Bilbao

The paper has been published in the
Proceedings of the 16th Sound and Music Computing (SMC) Conference, pp.
275–280, 2019.

©2019 Silvin Willemsen et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

changed the
template here,
should check if
it's ok like this

Abstract

Here is an abstract.

A.1 Introduction

Here is an introduction.

A.2 Conclusion

Here is the conclusion.

Paper B

Physical Models and Real-Time Control with the Sensel Morph

Paper C

Real-Time Implementation of an Elasto-Plastic Friction Model applied to Stiff Strings using Finite Difference Schemes

Paper D

Real-time Implementation of a Physical Model of the Tromba Marina

Paper E

Resurrecting the Tromba Marina: A Bowed Virtual Reality Instrument using Haptic Feedback and Accurate Physical Modelling

Paper F

DigiDrum: A Haptic-based Virtual Reality Musical Instrument and a Case Study

Paper G

Dynamic Grids for Finite-Difference Schemes in Musical Instrument Simulations

Paper H

A Physical Model of the Trombone using Dynamic Grids for Finite-Difference Schemes