The Emulated Ensemble

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

Ph.D. Dissertation

Silvin Willemsen

Dissertation submitted June 3, 2021

Thesis submitted:	June 3, 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

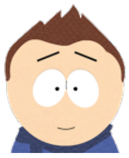
ISSN: xxxx-xxxx
ISBN: xxx-xx-xxxx-xxx-x

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

Printed in Denmark by Rosendahls, 2021

Curriculum Vitae

Silvin Willemsen



Here is the CV text.

Acknowledgements

I would like to thank my mom..
Both Stefan Bilbao and his seminal work *Numerical Sound Systhesis* have been invaluable to the result of this project...

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 V, 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

Resumé

Danish Abstract

Contents

FULL DOC SWEEP: check capitalisation of headings throughout document

FULL DOC SWEEP: check hyphen in titles

Todo list

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. 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).

All Newton's laws of motion will make their appearance

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.

Silvin Willemsen
Aalborg University, June 3, 2021

Part I

Introduction

Chapter 1

Physical Modelling of Musical Instruments

The history of physical modelling of musical instruments

Exciter-resonator approach.

The time-evolution of dynamic systems can be conveniently described by differential equations. Examples of a dynamic systems are a guitar string, a drum-membrane, or a concert hall; three very different concepts, but all based on the same types of equations of motion.

Though these equations are very powerful, only few have a closed-form solution. What this means is that in order for them to be implemented, they need to be approximated. There exist different approximation techniques to do this

1.1 Physical Modelling Techniques

Main:

- Modal Synthesis
- Finite-difference Time-domain methods
- Finite Element Methods
- Digital waveguides
- Mass-spring systems

Weird

- Functional transformation method

- State-space
- Wave-domain
- Energy-based (post-hamiltonian)

FDTD:
Energy techniques: [12] von Neumann analysis: [17]
Advantages of finite-difference methods
Using FDTD methods can be quite computationally heavy. Moore's law [14]

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
Real-time: no noticeable latency

exactly as in [3]

1.3 Why?

1.3.1 Audio plugins

Samples, or recordings, of real instruments are static and unable to adapt to changes in performance. 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.

Samples vs. Physical Modelling:
Trade off between storage and speed
Using musical instrument simulations, on the other hand, allows the sound to be generated on the spot based on physical parameters that the user can interact with.

1.3.2 Resurrect old or rare instruments

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

1.3.3 Go beyond what is physically possible

1.4 Thesis 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 [11] to [7]

The main objective of this thesis is to implement existing physical models in real time using FDTD methods. 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.

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

1.5 Thesis Outline

Introduction to finite-difference methods and analysis techniques

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

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

- 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 [17]. Before diving into the mathematics, let us go over some useful terminology.

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, among others, to describe 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 (nearly) 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 [13] and potentially higher-dimensional systems [15]! See Section 2.1.1 for more information on this.

maybe not such a relevant reference..

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

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

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 derivative. However, this notation can only be used for univariate functions which it will be used for in this document. Finally, Euler's notation uses an operator which can be applied to a function and indicates a derivative.

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

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

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, the operator notation will be used, due to their similarity to the discrete operators (introduced shortly) and as it allows for creation of bigger operators for more compactness when working with multiple (connected) systems (see e.g. Chapter **??**). Also, state-of-the-art literature in the field of FDTD methods sound synthesis use this notation [5].

maybe not yet as this is super general still

Often-used partial derivatives and their meanings are shown below

$$\begin{aligned} &\partial_t^2u \quad \text{(acceleration)} && \partial_x^2u \quad \text{(curvature)}\\ &\partial_tu \quad \text{(velocity)} && \partial_xu \quad \text{(slope)} \end{aligned}$$

2.1.1 Dimensions and Degrees of Freedom

All objects in the physical world are 3-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. 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 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 polarisation by Desvages [10] and Desvages and Bilbao [9] using FDTD methods. Models including longitudinal string vibration, where the longitudinal and transversal displacements are couples can be found in [3, 6]. In [19], 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 would capture the aforementioned effects, but greatly increase the computational complexity of the models.

Although the dimensionality reduction ignoring 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, whereas the translational DoF determines the amount of coordinates of the state variable describes. For example, the state of a 2D system, with 1 translational DoF is written as $u(x, y, t)$. A 1D system with 3 translational DoF can be written as $\mathbf{u}(x, t)$ where $\mathbf{u}$ is a vector containing the coordinates for all three translational DoF.

2.1.2 Domains

When describing physical systems using state variables, *domains* over which they are defined need to be provided. This means that for the state of a 1D system $u = u(x, t)$, ranges of definition must be given for x and t. Usually, time $t \geq 0$, meaning that the system is defined for non-negative time. For the spatial dimension, we may define a finite domain $\mathcal{D}$ over which x is defined, 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.

2.2 Discretisation using FDTD methods

Differential equations are powerful tools that describe the motion of physical systems. Despite this, only few of these have a closed-form or analytical solution (such as the simpler ones that will be presented in this chapter). More complex systems require methods that do not perfectly solve, but rather *approximate* the solutions to these equations. Finite-difference time-domain (FDTD) are considered one of the in terms of generality and flexibility...

Unless denoted otherwise, the equations and theory used in this chapter has been taken from [3].

It is important to note that a discrete FD scheme is an *approximation* to a continuous PDE, not a sampled version of it. This means that the resulting schemes are rarely an exact solution to the original continuous equation.

These 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.

Figure and caption are not done yet

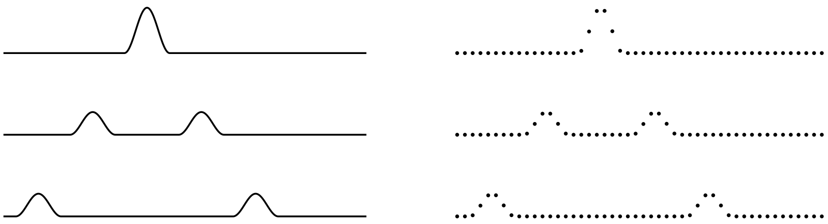


Fig. 2.1: A continuous PDE is discretised...

2.2.1 Grid Functions

FULL DOC SWEEP: check capitalisation throughout document

grid figure

We start by defining a discrete *grid* over time and space which we will use to approximate our continuous equations . 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 superscript 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 the temporal step k, or *time step* is the time (in s) between two consecutive temporal indices. The latter can be calculated $k = 1/f_\text{s}$ for a sample rate f_s (in Hz). In many audio applications $f_\text{s} = 44100$ 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 and one spatial domain. In discrete time, the temporal domain $t \geq 0$ 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 $N+1$, one more than the number of intervals.

To summarise, for a 1D system

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

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

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

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

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

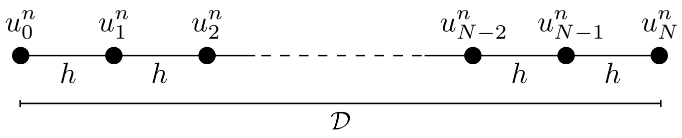


Fig. 2.2: When a 1D system $u(x,t)$ with $x \in \mathcal{D}$ is discretised to a grid function u_l^n, the spatial domain $\mathcal{D}$ is divided into N intervals of length h and spatial range of interest $l = \{0, \dots, N\}$.

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

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

Similarly, forward and backward shifts in space are

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

many figures for shift and FD operators

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

FULL DOC SWEEP: check centred instead of centered

these spacings are different in overleaf...

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

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

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

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

figure here visualising operators (with reference to grid figure)

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

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

and when applied to u_l^n are

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

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

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

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

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

both of which can be applied to a grid function u_l^n in a similar fashion. Further information on combining operators can be found in Section **??**.

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

in energy analysis, interleaved grids, etc.

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

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

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

Similarly, spatial averaging operators are

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

and when applied to u_l^n

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

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

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

and

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

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

Accuracy

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

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

where superscript (n) denotes the nth 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

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

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

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

Here, (following Newton's notation) 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

$$\begin{aligned}\frac{u(t+k)-u(t)}{k} &= \dot{u} + \frac{k}{2}\ddot{u} + \frac{k^2}{6}\dddot{u} + \mathcal{O}(k^3),\\ \delta_{t+}u(t) &= \dot{u} + \mathcal{O}(k),\end{aligned} \tag{2.18}$$

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

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

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

and performing a Taylor series expansion of u about t yields

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

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

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

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

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

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

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

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

As a first-order derivative indicates the *slope* of a function, the differences in accuracy between the above operators can be visualised as in Figure 2.3. It can be observed that the derivative approximation of the centred operator much more closely matches the the true derivative of u at t.

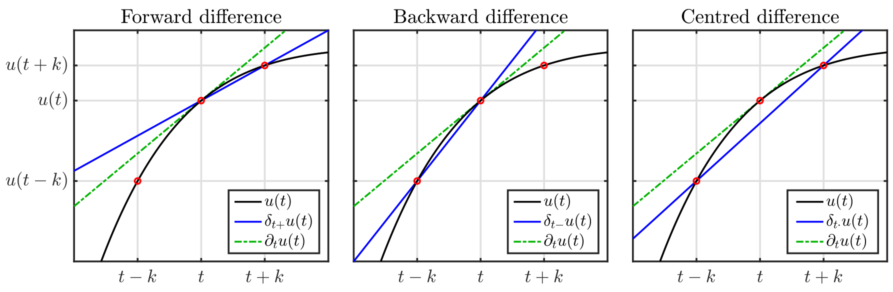


Fig. 2.3: The accuracy of the forward, backward and centred difference operators in (2.3) visualised. One can observe that the centred difference operator much more closely approximates the derivative, or the slope, of u at t than the forward and backward difference operators.

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

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

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

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

The accuracy of averaging operators can also be found and follow a similar pattern.

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

2.2.3 Identities

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

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

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

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

That these equalities hold can easily be proven by expanding the operators de-

see whether the negative version of identity (2.27c) is also used later on fined in Section 2.2.2. Naturally, these identities also hold for spatial operators by simply substituting the 't' subscripts for 'x'.

2.3 The Mass-Spring System

Though a complete physical modelling field on its own (see Chapter 1), mass-spring systems are also sound-generating systems themselves 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

$$M\ddot{u} = -Ku, \tag{2.28}$$

where $u = u(t)$ is the distance from the equilibrium position (in m), $M > 0$ is the mass of the mass (in kg) and $K \geq 0$ is the spring constant (in N/m). In the literature [3, **?**], Eq. (2.28) is often written as

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

with

$$\omega_0 = \sqrt{K/M}. \tag{2.30}$$

This way of writing the mass-spring ODE is more compact and more straightforward to relate to a fundamental frequency $f_0 = \omega_0/2\pi$ (in Hz).

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 2.4), which is also why the mass-spring system is often referred to as the *simple harmonic oscillator*. The amplitude of the sinusoid is determined by the initial conditions, whereas the frequency is determined by M and K.

Intuition

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

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

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

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

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

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

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

Finally, if $u = 0$ there is no spring force present and the acceleration remains unchanged. 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.

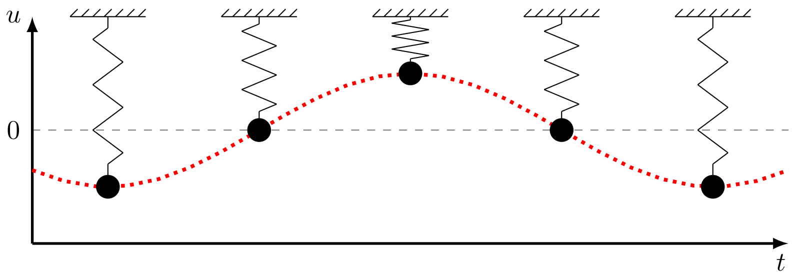


Fig. 2.4: Mass spring system over time. The system follows a harmonic (sinusoidal) motion.

2.3.2 Discrete-time

The displacement of the mass is approximated using

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

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

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

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

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

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

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

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

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

2.3.3 Implementation and Output

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

To be able to 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):

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

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

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

maybe a bit short...

To sum up, simply setting $u^0 = u^{-1} \neq 0$ yields an oscillatory behaviour.

The values for K and M are restricted by a stability condition which will be elaborated on in Section 3.3.

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 2.5 where the frequency of oscillation $f_0 = 440$ Hz.

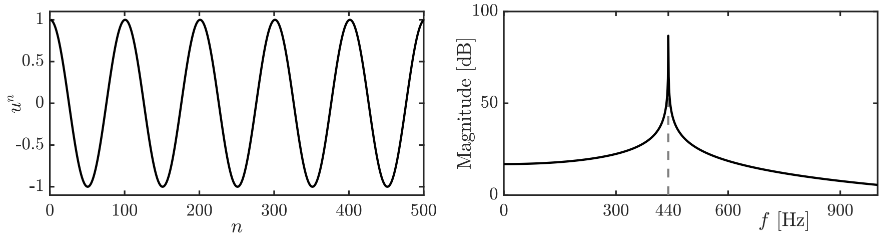


Fig. 2.5: The time-domain and frequency-domain output of a mass-spring system with $f_0 = 440$ Hz.

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.

FULL DOC SWEEP: check hyphen in titles

2.4.1 Continuous time

unit?

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

$$\partial_t^2 u = c^2 \partial_x^2 u, \tag{2.37}$$

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 (2.37) arises from Newton's second law, even though this connection might be less apparent.

The 1D wave equation in (2.37) states that the acceleration of $u = u(x,t)$ at location x is determined by the second-order spatial derivative of u at that same location (scaled by a constant c^2). In the case that u describes the transverse displacement of an ideal string, this second-order derivative denotes the *curvature* of this string. As c^2 is always positive, the sign (or direction) of the acceleration is fully determined by the sign of the curvature. In other words, a 'positive' curvature at location x along the ideal string yields a 'positive' or upwards acceleration at that same location.

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

So, what does this mean for the 1D wave equation? As a positive curvature implies a positive or upwards acceleration as per Eq. (2.37), 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 2.6.

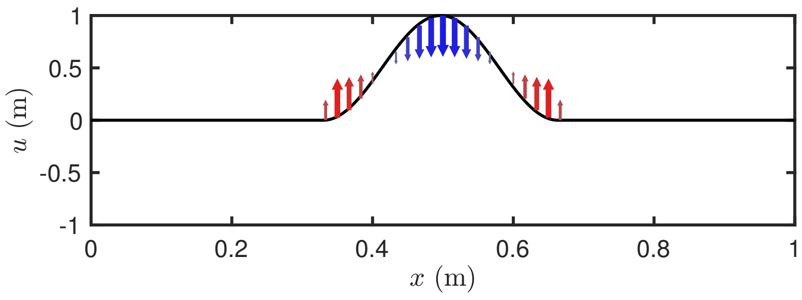


Fig. 2.6: The forces acting on the 1D wave equation due to curvature. The arrows indicate the direction and magnitude of the force, and simultaneously the acceleration as these are connected through Eq. (2.37).

different wording in caption

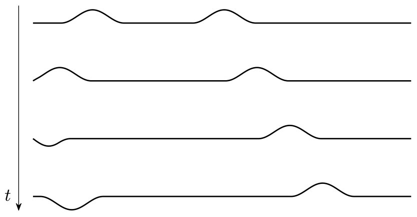
Boundary Conditions

When a system is distributed in space, *boundary conditions* must be determined. Consider a system of length L (in m) defined over space $x \in \mathcal{D}$ where domain $\mathcal{D} = [0, L]$. Two often-used alternatives are

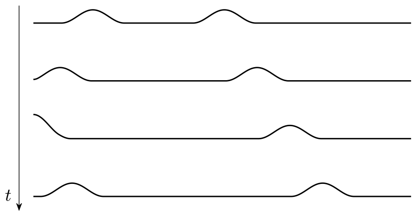
$$u(0,t) = u(L,t) = 0 \quad \text{(Dirichlet, fixed)}, \tag{2.38a}$$

$$\partial_x u(0,t) = \partial_x u(L,t) = 0 \quad \text{(Neumann, free)}. \tag{2.38b}$$

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



(a) The Dirichlet boundary condition in Eq. (2.38a) fixes the boundary, which causes the incoming waves to invert.



(b) The Neumann or free boundary condition in Eq. (2.38b) fixes the slope at the boundary, causing the incoming waves to not invert.

Fig. 2.7: The behaviour of the 1D wave equation with (a) Dirichlet or (b) Neumann boundary conditions.

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 2.7.

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

$$f_0 = \frac{c}{2L}\,. \tag{2.39}$$

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 $x' = x/L$ such that $x' \in [0, 1]$. The 1D wave equation in (2.37) can then be rewritten to

$$\partial_t^2 u = \gamma^2 \partial_{x'x'} u, \tag{2.40}$$

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 III), 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. (2.37) is the following FD scheme

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n, \tag{2.41}$$

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 yields

$$\frac{1}{k^2}\left(u_l^{n+1} - 2u_l^n + u_l^{n-1}\right) = \frac{c^2}{h^2}\left(u_{l+1}^n - 2u_l^n + u_{l-1}^n\right). \tag{2.42}$$

and solving for u_l^{n+1} yields

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

Here,

$$\lambda = \frac{ck}{h} \tag{2.44}$$

is called the *Courant number* and plays a big role in stability quality of the FD scheme. The Courant number needs to abide the (famous) Courant-Friedrichs-Lewy or *CFL condition* for short [8]

$$\lambda \leq 1, \tag{2.45}$$

which acts as a stability condition for scheme (2.41). If $\lambda = 1$, Eq. (2.41) is an exact solution to Eq. (2.37). Exact solutions are actually quite uncommon in the realm of differential equations! If $\lambda < 1$, the quality of the simulation quality decreases and dispersive and bandlimiting effects occur (see Section 2.4.3).

Stencil

It can be useful to visualise the *stencil*, or region of operation, of a FD scheme. A stencil visualises what grid values are necessary to calculate the state at the next time step u_l^{n+1}. Figure 2.8 shows the stencil for scheme (2.41) and in essence visualise the various appearances of the grid function in (2.43).

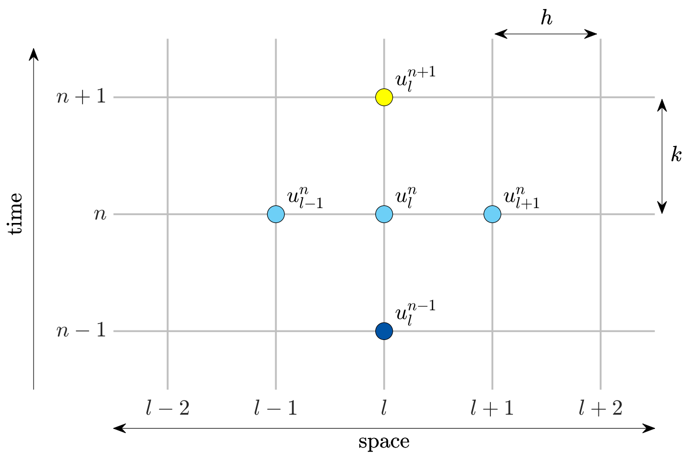


Fig. 2.8: The stencil, or region of operation, for the FD scheme in (2.41).

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. (2.43) 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 to as *virtual grid points* and can be accounted for by discretising the boundary conditions in Eq. (2.38). Discretising these (using the centred spatial difference operator for the Neumann condition) yields

$$u_0^n = u_N^n = 0, \quad \text{(Dirichlet)} \tag{2.46a}$$

$$\delta_{x\cdot} u_0^n = \delta_{x\cdot} u_N^n = 0. \quad \text{(Neumann)} \tag{2.46b}$$

Expanding the operators in Eq. (2.46b) and solving for u_{-1}^n and u_{N+1}^n provides the definitions for these virtual grid points based on values in the discrete domain:

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

If Dirichlet boundary conditions are used, the states of the boundary points will always be zero and can therefore can be excluded from the calculations. The range of calculation then simply becomes $l \in \{1, \ldots, N-1\}$. If, on the other hand, Neumann conditions are used, the update equation in (2.43) at the boundaries will have the the above definitions for the virtual grid points substituted at $l = 0$ and $l = N$ and will become

$$u_0^{n+1} = \left(2 - 2\lambda^2\right) u_0^n + 2\lambda^2 u_1^n - u_0^{n-1}, \tag{2.47}$$

and

$$u_N^{n+1} = \left(2 - 2\lambda^2\right) u_N^n + 2\lambda^2 u_{N-1}^n - u_N^{n-1}, \tag{2.48}$$

at the left and right boundary respectively.

2.4.3 Implementation and Output

As c, k and h are interdependent due to the CFL condition in (2.45), it is useful to rewrite this condition 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, Eq. (2.45) can be rewritten in terms of the grid spacing h:

$$h \geq ck, \tag{2.49}$$

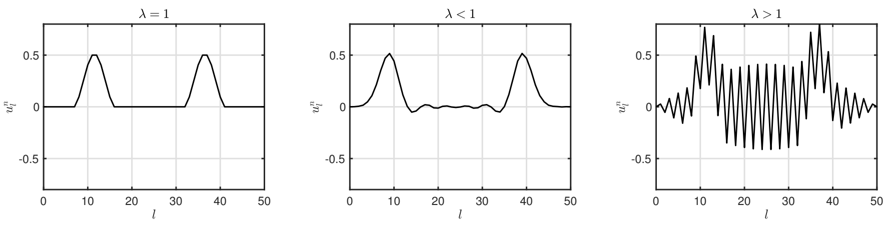
which, in implementation, is used as a stability condition for the scheme. See Section 3.3 for more information on how to derive the stability condition from a FD scheme. Again, if Eq. (2.49) is satisfied with equality, the FD scheme is an exact solution to the PDE, and if h deviates from this condition, the quality of the simulation decreases.

So why 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. A logical step to calculate N would be to divide L by h calculated using (2.49) satisfied with equality. However, this calculation might not result in an integer value, which N should be! To stay as close to the stability condition as possible, the following calculations are performed in order:

$$h := ck, \quad N := \left\lfloor \frac{L}{h} \right\rfloor, \quad h := \frac{L}{N}, \quad \lambda := \frac{ck}{h}, \tag{2.50}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. In other words, Eq. (2.49) 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 λ. This process assures that N is an integer and that the CFL condition is satisfied, though not necessarily with equality.

If this recalculation of h is not carried out, the behaviour of the system will not be correct. For example, we want our 1D wave equation to be 1 m long ($L = 1$) and produce a fundamental frequency of $f_0 = 750$ Hz according to (2.39). This requires a wave speed of $c = 1500\,\text{m/s}$. If we use a commonly-used sample rate of $f_\text{s} = 44100$ Hz, and recalling that $k = 1/f_\text{s}$, these values can be filled into (2.49) satisfied with equality yielding $h \approx 0.034$. If we divide the length by the grid spacing, we get $L/h = 29.4$, meaning that exactly 29.4



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

Fig. 2.9: State u_l^n with $N = 50$ and $f_s = 44100$ 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. (2.45) is not satisfied and the system is unstable.caption is straight from paper

intervals of size h fit in the domain $\mathcal{D}$. However, the number of intervals needs to be an integer and – using the above values – we get $N = 29$. If h is not recalculated according to (2.50), 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 tuned up: $f_0 \approx 760.34$ Hz. If h is recalculated based on N, L and f_0 will be unchanged, and $\lambda \approx 0.986$ which is still very close to satisfying condition (2.45).

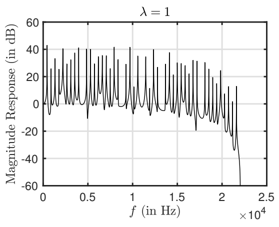
Output

After the system is excited (see **??**), one can retrieve the output of the system by selecting a grid point and listening to that at at the given sample rate f_s. The amount of modes 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.

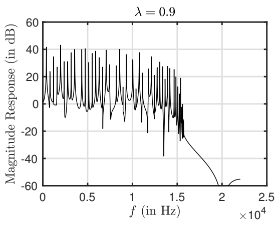
If the CFL condition is satisfied with equality, these modes are integer multiples of the fundamental: $f_m = mf_0$ for mode number $m \in \{1, \ldots, N - 1\}$ for Dirichlet and $m \in \{0, \ldots, N\}$ for Neumann boundary conditions. The frequency of the harmonic partials can also be analytically / numerically derived using modal analysis as will be explained in Section 3.5.

The amplitude of the different modes, on the other hand, depends on the location of excitation and output. modes

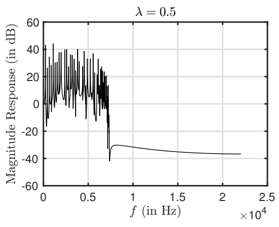
See Appendix C.2MATLAB implementation of the 1D wave equation



(a) $\lambda = 1$



(b) $\lambda = 0.9$



(c) $\lambda = 1.001$

Fig. 2.10: Bandwidths of the simulation output at $l = 16$ with $f_s = 44100$ Hz and $N = 50$ excited with a raised cosine with a width of 5 at center-location $N = 25$. The Courant number is set to (a) $\lambda = 1$, (b) $\lambda = 0.9$ and (c) $\lambda = 0.5$.caption is straight from paper

Chapter 3

Analysis Techniques

This chapter provides some techniques to analyses FD schemes .

Techniques to analyse PDEs also exist, but the focus here is on the discrete schemes.

The analysis techniques will be especially explained from a practical side, rather than providing all the theory that they are based on. Obviously, references to work substantiating the claims made in this chapter will be provided.

probably not necessary

References to the origins of these techniques will be provided for the interested reader.

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 "$row \times column$". A 3×5 matrix, for example, thus has 3 rows and 5 columns (see Figure 3.1a). Along those lines, a *row vector* is a matrix with 1 row and more than 1 column and a *column vector* is a matrix with 1 column and more than 1 row. If a matrix has only 1 row and 1 column, it can be used as a *scalar*.

In this document, matrices and vectors are written using bold symbols. Many notations exist, blabla $\bar{a}$ $\vec{a}$ 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}.

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

$$a_{ij} = a_{ji}. \tag{3.1}$$

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

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

(a) A 3×5 matrix $\mathbf{A}$.

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

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

Fig. 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).

Matrix Multiplication

In order for matrix multiplication (this includes matrix-vector multiplication) to be valid, the number of columns of the first matrix needs to be equal to the number of rows in the second matrix. The result will then be a matrix with a number of rows equal to that of the first matrix and a number of columns equal to that of the second matrix. See Figure 3.2 for reference.

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

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

A matrix can always be multiplied by a scalar (single number), which simply multiplies every element of the matrix by the scalar.

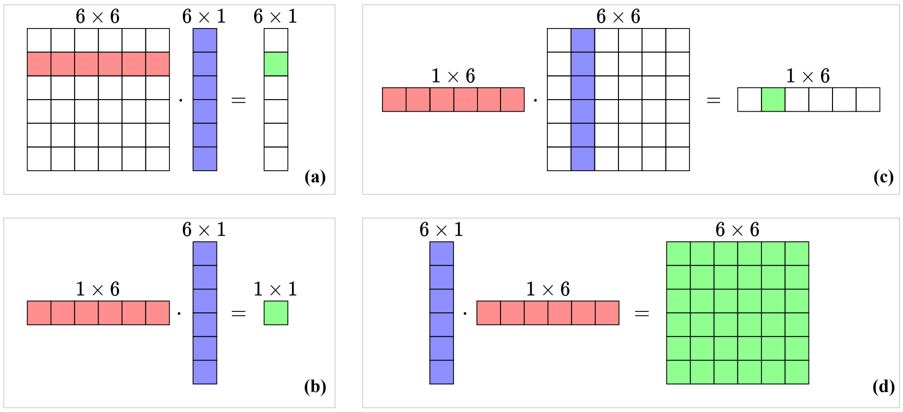


Fig. 3.2: Visualisation of valid matrix multiplications. The “inner” dimensions (columns of the left matrix and rows of the right) must match and result in a matrix with a size of “outer” dimensions (rows of the left matrix and columns of the right).

3.1.1 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 $\mathbf{w}$:

$$\mathbf{A}\mathbf{u} = \mathbf{w}. \tag{3.3}$$

Expanding this operation results in

$$\underbrace{\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0N} \\ a_{10} & a_{11} & \dots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N0} & a_{N1} & \dots & a_{NN} \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} a_{00}u_0 + a_{01}u_1 + \ldots + a_{0N}u_N \\ a_{10}u_0 + a_{11}u_1 + \ldots + a_{1N}u_N \\ \vdots \\ a_{N0}u_0 + a_{N1}u_1 + \ldots + a_{NN}u_N \end{bmatrix}}_{\mathbf{w}} \tag{3.4}$$

where the indexing starts at 0 rather than 1 here, as that relates better to a FDTD context.

Operators in Matrix Form

FD operators approximating spatial derivatives and averages can be written in matrix form and applied to a column vector $\mathbf{u}$ 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

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

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

is this how you explain it?

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

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

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

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

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

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

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

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 (2.41) can be written in matrix form.

If the Dirichlet boundary conditions in (2.46a) are used, the end points of the system do not have to be included in the calculation. The values of the grid function u_l^n for $l \in \{1, \ldots, N-1\}$ can then be stored in a column vector according to $\mathbf{u}^n = [u_1^n, \ldots, u_{N-1}^n]^T$. Furthermore, $(N-1) \times (N-1)$ matrices $\mathbf{D}_{x+}$ and $\mathbf{D}_{x-}$ can be multiplied to get a same-sized matrix $\mathbf{D}_{xx}$:

$$\mathbf{D}_{xx} = \mathbf{D}_{x+}\mathbf{D}_{x-} = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \mathbf{0} \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ \mathbf{0} & & & 1 & -2 \end{bmatrix}. \tag{3.5}$$

If instead, Neumann boundary conditions in Eq. (2.46a) are used, the values of u_l^n for the full range $l \in \{0, \ldots, N\}$ need be stored as $\mathbf{u}^n = [u_0^n, \ldots, u_N^n]^T$ and the $(N+1) \times (N+1)$ matrix $\mathbf{D}_{xx}$ will be

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

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 Eqs. (2.47) and (2.48) respectively.

Regardless of the boundary conditions, the FD scheme in (2.41) can be written in matrix form as

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

and rewritten to a matrix form of the update equation analogous to Eq. (2.43)

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

The identity matrix is necessary here for correct matrix addition.

3.1.2 Matrix Inverse

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

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}. \tag{3.9}$$

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 this case, the matrix is called *singular*. Rather than going through how to invert a matrix by hand, or determining whether it is singular, the following function in `MATLAB` will provide the inverse of a matrix `A`:

```
A_inverted = inv(A);
```

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

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

3.1.3 Systems of Linear Equations

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

For example, take the system of linear equations

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

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

$$\mathbf{Au} = \mathbf{w}. \tag{3.10}$$

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 3.1.2) and multiplying this onto $\mathbf{w}$

$$\mathbf{u} = \mathbf{A}^{-1}\mathbf{w}. \tag{3.11}$$

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 3.1.2

```
u = inv(A) * w;
```

or more compactly, by using the '\' operator:

```
u = A\w;
```

This method will come in handy in Section ...

3.1.4 Eigenvalue Problems

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

ask stefan

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

$$\mathbf{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi}. \tag{3.12}$$

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

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

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');
```

Check code here!

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

check whether this is necessary

```
[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. Tools for 2D systems will be introduced 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

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

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.

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

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

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

$$\langle f_l^n, g_l^n\rangle_d' = \sum_{l=1}^{N-1} h f_l^n g_l^n + \frac{h}{2} f_0^n g_0^n + \frac{h}{2} f_N^n g_N^n, \tag{3.16}$$

and the more general weighted inner product

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

which scale the boundary points of the regular inner product. Naturally, if $\epsilon_\text{l} = \epsilon_\text{r} = 1$, Eq. (3.17) reduces to (3.16), and if $\epsilon_\text{l} = \epsilon_\text{r} = 2$, (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 to integration by parts. Although its application will be only be apparent when actually performing an energy analysis (see fx. Section 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 the discrete inner product defined in Eq. (3.15),integration by parts is defined as

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

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

Defining two (1D) functions f_l^n and g_l defined for $l \in d$ with discrete domain $d = \{0, \dots, N\}$, two variants of summation by parts are defined as [3]

check whether this reference is necessary

$$\langle f_l^n, \delta_{x-} g_l^n\rangle_d = -\langle \delta_{x+} f_l^n, g_l^n\rangle_d + f_{N+1}^n g_N^n - f_0^n g_{-1}^n, \tag{3.19a}$$
$$\langle f_l^n, \delta_{x+} g_l^n\rangle_d = -\langle \delta_{x-} f_l^n, g_l^n\rangle_d + f_N^n g_{N+1}^n - f_{-1}^n g_0^n. \tag{3.19b}$$

A derivation of Eq. (3.19a) is given below. As in the continuous case in Eq. (3.18), the process 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 changed. Lastly, discrete boundary terms have appeared and it can be seen that values outside of the defined domain are needed, i.e., g_{N+1}^n and f_{-1}^n. These can be accounted for by the boundary conditions imposed on the system (see Section 2.4.2 as an example).

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

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

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

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

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

Finally, one can apply summation by parts twice to get:

$$\langle f, \delta_{xx} g\rangle = \langle \delta_{xx} f, g\rangle_d + f_N \delta_{x+} g_N - g_N \delta_{x+} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x-} f_0, \tag{3.22}$$

or, using the reduced domains,

$$\langle f, \delta_{xx} g\rangle = \langle \delta_{xx} f, g\rangle_{\underline{\overline{d}}} + f_N \delta_{x+} g_N - g_N \delta_{x-} f_N - f_0 \delta_{x-} g_0 + g_0 \delta_{x+} f_0. \tag{3.23}$$

Derivations

To see why the above identities hold true, it is useful to 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, $d = \{0, \ldots, N\}$ and $N = 2$ are used.

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

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

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

Using the same inner product as a starting point, identity (3.20a) can be proven to hold:

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

Similar processes can be used to prove the other identities presented in this section.

3.2.3 Product identities

Some useful identities used in this work are

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

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

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

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

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

Again, these can be used for spatial derivatives as well by substituting the 't' subscripts for 'x'. Also to

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

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

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. A mass-spring system finds a DSP equivalent in a resonator filter

Frequency domain representation and Ansatz

FD schemes can be analysed by performing a *z-transform*. 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 [16, 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 [17] and is heavily used in [3]. The discrete-time z-transform and discrete spatial Fourier transform performed on a 1D grid function are defined as [3]

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

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

check if I should not refer to a subsection

A shortcut to performing a full frequency domain analysis is to use a test solution, or *ansatz*, and replace the grid functions by their transforms. The grid function for a 1D system can be replaced by an ansatz of the form (1D) [17]

check reference

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

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

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

check

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

Table 3.1: Frequency-domain representation of a grid function using ansatz (3.27) with frequently appearing temporal and spatial shifts.

Using these definitions, the effect of various operators on a grid function can be written in their frequency-domain representation. For systems distributed in space, the following trigonometric identities are extremely useful when performing the analyses [20, p. 71]:

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

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

Take for example

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

Then, using $x = \beta h/2$, identity (3.28a) can be rewritten to

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

and substituted into the above to get

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

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

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

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

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

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

Frequency domain analysis only works on linear and time-invariant (LTI) systems. Nonlinear systems can be analysed using energy analysis (see Section 3.4).

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

not talking about nonlinear systems though

Proving stability

Just as with digital filters, the system is stable when the roots of the polynomial in z are bounded by 1 (unity)

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

In the FDTD context, the frequency-domain representation of a FD scheme results in a *characteristic equation* – usually a second-order polynomial- in z and needs to satisfy condition (3.31) for all wave numbers β. It can be shown that for a polynomial of the form

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

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

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

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

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

suffices.

3.3.1 Mass-Spring System

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

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

a frequency-domain representation can be obtained using the ansatz in (3.27) with $l = 0$. Using Table 3.1 and Eqs. (3.29) as a reference and substituting the definitions yields

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

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

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

To begin to prove stability, this equation needs to be written in the form found in (3.32). Multiplying all the terms by z, and noticing that $a^{(2)} = 1$, we could continue with condition (3.34). However, the scheme used here is a special case where the roots of the characteristic equation can not be identical [3]. When this happens, the output of the system will grow linearly and is called "marginally unstable". This means that $|a^{(1)}| \neq 1$ and the condition in (3.34) becomes $|a^{(1)}| < 2$:

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

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

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

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

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

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

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

3.3.2 1D Wave Equation

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

Recalling the FD scheme in (2.41),

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

its frequency-domain representation can be obtained using the definitions in Eqs. (3.29) and (3.30a):

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

Also recalling that

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

the characteristic equation of the 1D wave equation is

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

The scheme is then stable if the roots satisfy condition (3.31). As the characteristic equation is of the form in (3.32) (after multiplication with z) with $a^{(2)} = 1$, stability is shown by abiding condition (3.34) for all β and when applied to the characteristic equation (3.39), it can be seen that

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

Observing that all terms in $\lambda^2\sin^2(\beta h/2)$ are squared, this term will always be non-negative and therefore always satisfy the first condition. Continuing with the second condition, and knowing that the $\sin^2(\beta h/2)$-term is bounded by 1 for all β, we arrive at the following stability condition:

$$\lambda \leq 1.$$

This is the CFL condition given in (2.45). To obtain the stability condition in terms of the grid spacing, the definition for λ is substituted and written in terms of the grid spacing

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

which is the stability condition given in Eq. (2.49).

3.4 Energy Analysis

Energy steps for copying /////

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

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

Step 3: Check units

Step 4: Implementation

/////

shouldn't I just cite this one then?

Energy analysis techniques first appeared in (the first edition of) [12] called the energy method

(also see p. 54 in [3], specifically the section **Finite precision and round-off error for more references**)

For the 1D wave equation this means to multiply the scheme with $(\delta_{t\cdot}u_l^n)$

The total energy of the system, or Hamiltonian $\mathfrak{H}$

In implementation it can be very helpful to debug physical models... One can plot the energy of the system and in a lossless system without external energy input, the rate of change of the total energy should be 0, i.e.,

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.41}$$

.

Although the energy of a lossless system should be unchanged according to Eq. (3.41), but in a finite precision simulation, ultra slight fluctuations of the energy should be visible due to rounding errors.

Plotting energy should be within *machine precision*, which mostly is in the range of 10^{-15}

In this work, the focus of the energy analysis will be in discrete time...

The following steps can be followed to perform a full energy analysis of a FD scheme:

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 and expression for the rate of change of the energy of the system. Usually, this means to take the inner product of the scheme with $(\delta_{t\cdot}u_l^n)$. See Section 3.2.1 for more details on the inner product.

For the units of the resulting energy balance to add up (also see Step 4), 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 an FD scheme can generally be divided into three different types: the total energy contained within the system, or Hamiltonian $\mathfrak{h}$, energy losses through damping $\mathfrak{q}$ and energy input through external forces or excitations $\mathfrak{p}$. For distributed systems, an additional boundary term $\mathfrak{b}$ appears, but vanishes under 'regular' (lossless and not energy-storing) boundary conditions. Nearly any energy balance is thus of the form

$$\delta_{t+}\mathfrak{h} = \mathfrak{b} - \mathfrak{q} - \mathfrak{p} \tag{3.42}$$

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

$$\delta_{t+}\mathfrak{h} = 0 \quad \Longrightarrow \quad \mathfrak{h}^n = \mathfrak{h}^0. \tag{3.43}$$

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

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

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \tag{3.44}$$

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 in SI units: kg $\cdot$ m$^2\cdot$ s^{-2}. The other quantities such as energy losses $\mathfrak{q}$ and inputs $\mathfrak{p}$, should be in Joules per second or in SI units: kg $\cdot$ m$^2\cdot$ s^{-3}. 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, if the state u 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.

Subtract $\mathfrak{h}^0$

3.4.1 Mass-spring system

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

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

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

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

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

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

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 (3.24a) and (3.24b) to get the following:

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

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

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

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

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

Step 3: Check units

As mentioned above, the energy $\mathfrak{h}$ needs to be in Joules, or $\mathrm{kg}\cdot\mathrm{m}^2\cdot\mathrm{s}^{-2}$. Taking the terms in Eq. (3.47) one-by-one and writing them in their units results in

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

and indeed have the correct units.

Step 4: Implementation

todo

3.4.2 1D Wave Equation

Energy analysis could be directly performed on the FD scheme in (2.41). However, in order for the units of the scheme to add up to energy in Joules, it is better 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 scheme (2.41) by ρA yields

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

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. (2.46a) 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 of Eq. (3.49) with $(\delta_{t\cdot}u_l^n)$ and moving all terms to the left-hand side yields the definition for the rate of change of the Hamiltonian:

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

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. Recalling that $\delta_{xx} = \delta_{x+}\delta_{x-}$ we can use summation by parts as described in Eq. (3.20a) with $f_l^n \triangleq \delta_{t\cdot}u_l^n$ and $g_l^n \triangleq \delta_{x+}u_l^n$. The second term can then be rewritten to

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

where the boundary term

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

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

$$u_0^n = u_N^n = 0 \quad\Longrightarrow\quad \delta_{t\cdot}u_0^n = \delta_{t\cdot}u_N^n = 0 \quad\Longrightarrow\quad \mathfrak{b} = 0.$$

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

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

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

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

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

$$\begin{aligned}\mathfrak{h} &= \mathfrak{t} + \mathfrak{v},\\ \text{with}\quad \mathfrak{t} &= \frac{\rho A}{2}\|\delta_{t-}u_l^n\|_d^2, \quad\text{and}\quad \mathfrak{v} = \frac{T}{2}\langle\delta_{x+}u_l^n, e_{t-}\delta_{x+}u_l^n\rangle_{\underline{d}}.\end{aligned} \tag{3.53}$$

Step 3: Check units

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

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

Notice that an extra 'm' unit appears due to the 1D norm and inner product.

Step 4: Implementation

todo

Neumann boundary conditions

If Neumann boundary conditions – as per Eq. (2.46b) – 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

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

where the boundary term

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

As the Neumann boundary condition states that

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

the boundary term vanishes and the energy balance results in

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

3.4.3 Stability Analysis using Energy Analysis Techniques

One can perform stability analysis using the energy analysis techniques presented here. To arrive at a condition the energy must be *positive definite*

As opposed to the frequency domain analysis presented in Section 3.3, energy analysis techniques do not operate in the frequency domain. However, stability can be shown using the techniques presented here. Stability analysis using the energy method might even be considered more powerful than the frequency domain approach, as it can also (to some degree) be used to non-linear and time-varying systems!

Rather than the roots of a characteristic equation needing to be bound by unity, to arrive at a condition the energy must be *positive definite*. It can be shown that an equation of the form

$$x^2 + y^2 + 2axy \tag{3.55}$$

is positive definite if $|a| < 1$. If $|a| = 1$, the function is non-negative, but not positive definite.

Equation (3.55) can be used to prove stability for the mass spring system using the energy balance in Eq. (3.47). easily conclude that $\mathfrak{t}$ is positive definite due to non-negative M and the fact that $\delta_{t-}u^n$ is squared. The potential energy $\mathfrak{v}$, however, is of indefinite sign. Expanding the operators in Eq. (3.47) yields

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

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

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

and for $\mathfrak{h}$ to be positive definite, i.e., $\mathfrak{h} > 0$ the following condition must hold

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

This can then be solved for k according to

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

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

$$k < 2\sqrt{\frac{M}{K}} \tag{3.56}$$

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

For the 1D wave equation, one can take the energy balance in Eq. (3.53) and conclude that $\mathfrak{t}$ is positive definite due to the non-negativity of the parameters and $(\delta_{t-}u_l^n)$ being squared. The potential energy, however, is of indefinite sign. One can rewrite $\mathfrak{v}$ using identity (3.24e) as

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

One can then use the following bound [3]

$$\|\delta_{x+}u_l^n\|_{\underline{d}} \le \frac{2}{h}\|u_l^n\|'_d \le \frac{2}{h}\|u_l^n\|_d, \tag{3.57}$$

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

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

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

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

which, recalling that $c = \sqrt{T/\rho A}$ and $\lambda = ck/h$ yields

$$\mathfrak{h} = \mathfrak{t} + \mathfrak{v} \ge \frac{1}{2}\left(1 - \lambda^2\right)\|\delta_{t-}u_l^n\|_d^2 + \frac{c^2}{2}\|\mu_{t-}\delta_{x+}u_l^n\|_{\underline{d}}^2, \tag{3.58}$$

and is non-negative for

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

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 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 analytically obtain the modes of an FD scheme implementing the 1D wave equation.

We start by using the the matrix form of FD scheme (2.41) from Eq. (3.7)

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

more explanation, perhaps refer to von neumann analysis in 3.3

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

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

This is an eigenvalue problem (see Section 3.1.4) where the p^{th} solution $\boldsymbol{\phi}_p$ may be interpreted as the modal shape of mode p. The modal frequencies are the solutions to the following equations:

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

If the CFL condition for the scheme is satisfied, the roots will lie on the unit circle. Furthermore, we can substitute a test solution $z_p = e^{j\omega_p k}$ with angular frequency of the p^{th} mode ω_p and solve for these:

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

and using Eq. (3.28a) we get

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

3.5.1 One-Step Form

For more complicated systems, where the coefficients of z and z^{-1} in the characteristic equation are not identical for example, it is useful to rewrite the update in *one-step form*. Although the eigenfrequency calculation needs to be done on a larger matrix, it allows for a more general and direct way to calculate the modal frequencies and damping coefficients per mode.

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

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1}, \tag{3.62}$$

can be rewritten to

$$\underbrace{\begin{bmatrix}\mathbf{u}^{n+1}\\ \mathbf{u}^n\end{bmatrix}}_{\mathbf{w}^{n+1}} = \underbrace{\begin{bmatrix}\mathbf{A}^{-1}\mathbf{B} & \mathbf{A}^{-1}\mathbf{C}\\ \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{Q}}\underbrace{\begin{bmatrix}\mathbf{u}^n\\ \mathbf{u}^{n-1}\end{bmatrix}}_{\mathbf{w}^n} \tag{3.63}$$

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

Again we can assume solutions of the form $\mathbf{w} = z^n\boldsymbol{\phi}$ (where $\boldsymbol{\phi}$ is less-trivially connected to the modal shapes) and get

$$z\boldsymbol{\phi} = \mathbf{Q}\boldsymbol{\phi}, \tag{3.64}$$

which can be solved for the pth eigenvalue as

$$z_p = \text{eig}_p(\mathbf{Q}). \tag{3.65}$$

As the scheme could exhibit damping, the test solution needs to include this and will be $z_p = e^{s_pk}$ with complex frequency $s_p = j\omega_p + \sigma_p$ and damping of the pth eigenfrequency σ_p. Substituting the test solution and solving for s_p yields

$$\begin{aligned} e^{s_pk} &= \text{eig}_p(\mathbf{Q}),\\ s_p &= \frac{1}{k}\ln\left(\text{eig}_p(\mathbf{Q})\right). \end{aligned} \tag{3.66}$$

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

$$\omega_p = \Im(s_p) \quad \text{and} \quad \sigma_p = \Re(s_p), \tag{3.67}$$

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. (3.66) come in complex conjugates. For analysis, only the $\Im(s_p) \geq 0$ should be considered as these correspond to non-negative frequencies.

3.6 Dispersion analysis

Part II

Resonators

Resonators

Though the physical models described in the previous part 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 includes 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 introduces 2D systems which, in this project, have been used to simulate (simplified) instrument bodies.

Chapter 4

Stiff string

Earlier, the case of the ideal string was presented (Section 2.4) modelled using the 1D wave equation. As shown, the model generates an output with harmonic partials that are integer multiples of the fundamental frequency. In the real world, however, strings exhibit a phenomenon called *inharmonicity* 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 partials get exponentially further apart with frequency.

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

This chapter

4.1 Continuous time

Consider a lossless stiff string with a circular cross-section of length L. 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

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

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

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 (2.37) where $c = \sqrt{T/\rho A}$. If instead $T = 0$, Eq. (4.1) reduces to the *ideal bar* equation.

Dispersion Analysis

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

4.1.1 Adding Losses

Before moving on to the discretisation of the PDE in (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 die out (dampen) 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. The model of a damped stiff string is

$$\rho A\partial_t^2 u = T\partial_x^2 u - EI\partial_x^4 u - 2\sigma_0\rho A\partial_t u + 2\sigma_1\rho A\partial_t\partial_x^2 u \tag{4.2}$$

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

etc.

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa^2\partial_x^4 u - 2\sigma_0\partial_t u + 2\sigma_1\partial_t\partial_x^2 u \tag{4.3}$$

where $c = \sqrt{T/\rho A}$ is the wave speed (in m/s) as in the 1D wave equation in (2.37) and $\kappa = \sqrt{EI/\rho A}$ is a *stiffness coefficient* (in m^2/s).

check wavespeed or wave speed (entire document)

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

The boundary conditions found in Eq. (2.38) can be extended to

$$u = \partial_x u = 0 \quad \text{(clamped)} \tag{4.4a}$$
$$u = \partial_x^2 u = 0 \quad \text{(simply supported)} \tag{4.4b}$$
$$\partial_x^2 u = \partial_x^3 u = 0 \quad \text{(free)} \tag{4.4c}$$

at $x = 0, L$.

insert figure showing virtual grid points

4.2 Discrete Time

For the sake of brevity, we continue with Eq. (4.3) (rather than Eq. (4.2)) which can be discretised as

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

for domain $l \in \{0, \dots, N\}$ and number of grid points $N+1$.

The δ_{xxxx} operator is the second-order spatial difference in Eq. (2.8) applied to itself

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

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

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

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

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

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

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

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

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

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

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

with

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

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

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

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

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

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

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 4.1. In order to calculate u_l^{n+1}, 5 points at the current time step are needed due to the 4th -order spatial derivative. Due to the mixed derivative in the frequency-dependent damping term neighbouring points at the previous time step are also required.

4.2.1 Boundary conditions

Due to the 4th -order spatial derivative, two virtual grid points need to be accounted for at the boundaries of the system. Discretising the boundary

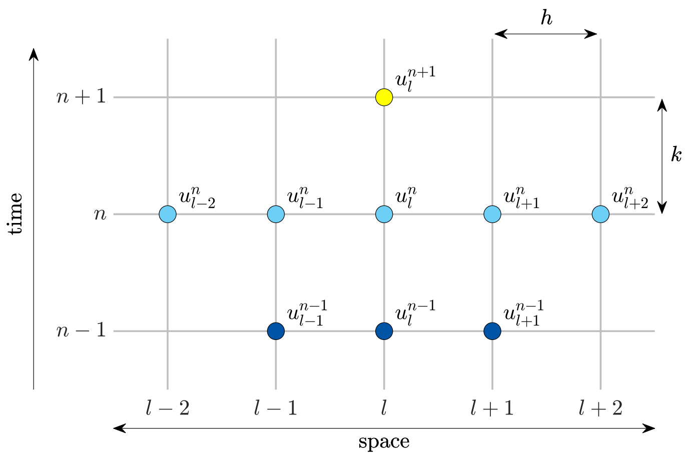


Fig. 4.1: The stencil for the damped stiff string scheme in (4.5).

conditions in (4.4) yields

$$u_l^n = \delta_{x\pm}u_l^n = 0 \quad \text{(clamped)} \tag{4.14a}$$
$$u_l^n = \delta_{xx}u_l^n = 0 \quad \text{(simply supported)} \tag{4.14b}$$
$$\delta_{xx}u_l^n = \delta_{x\cdot}\delta_{xx}u_l^n = 0 \quad \text{(free)} \tag{4.14c}$$

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 ($l = N$). Below, the boundary conditions are solved and update equations at the boundaries are given.

Clamped

Expanding the operators for the clamped condition yields

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

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 $l = N-1$ 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 $l = 0$:

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

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

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

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

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

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

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

Free

Finally, the free boundary condition requires all points to be calculated and $l \in \{0, \dots, N\}$. The combined operator in Eq. (4.14c) is defined as:

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

This can be used to solve for the virtual grid points in the free condition at $l = 0$:

$$\begin{aligned}&\frac{1}{2h^3}\left(u_2^n - 2u_1^n + u_{-1}^n - u_{-2}^n\right) = 0\\ &u_{-2}^n = u_2^n - 2u_1^n + u_{-1}^n\\ \xLeftrightarrow{\delta_{xx}u_l^n=0 \;\Rightarrow\; \text{Eq. (4.16)}}\quad &u_{-2}^n = u_2^n - 3u_1^n\end{aligned}$$

The update equation then becomes....

$$\dots \tag{4.20}$$

In practice, the simply supported boundary condition is mostly chosen as this reflects reality most.

The other boundary conditions occur more often for a (damped) ideal bar, (Eq. (4.2) with $T = 0$.)

4.2.2 Implementation and Matrix Form

When using `MATLAB`, to prevent clutter in the code, it is useful to write the scheme in matrix form. The FD scheme of the stiff string in (4.5) can be written as

$$A\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{4.21}$$

where

$$\begin{aligned} A = (1+\sigma_0 k), \quad & \mathbf{B} = c^2k^2\mathbf{D}_{xx} - \kappa^2k^2\mathbf{D}_{xxxx} + 2\sigma_1 k\mathbf{D}_{xx} \\ \text{and} \quad & \mathbf{C} = -(1-\sigma_0 k)\mathbf{I} - 2\sigma_1 k\mathbf{D}_{xx} \end{aligned} \tag{4.22}$$

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.

The matrix form of the δ_{xxxx} operator in (4.6) with simply supported boundary conditions can be obtained by multiplying two $\mathbf{D}_{xx}$ (eq. (**??**)) matrices according to

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

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 scheme (4.5).

Using the definitions found in Eq. (3.29) for the temporal operators and Eqs. (3.30a) and (3.30b) for the spatial operators, the frequency-domain representation of Eq. (4.5) can be obtained:

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

and after collecting the terms, the characteristic equation follows

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

Rewriting this to the form (3.32), and using $\mathcal{S}=\sin^2(\beta h/2)$ for brevity, yields

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

Stability of the system can then be proven using condition (3.33). Substituting the coefficients into this condition yields

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

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

$$\begin{aligned}-2+\frac{8\sigma_1 k}{h^2}\mathcal{S}\le 16\mu^2\mathcal{S}^2+\left(4\lambda^2+\frac{8\sigma_1 k}{h^2}\right)\mathcal{S}-2\le 2-\frac{8\sigma_1 k}{h^2}\mathcal{S},\\ 0\le 16\mu^2\mathcal{S}^2+4\lambda^2\mathcal{S}\le 4-\frac{16\sigma_1 k}{h^2}\mathcal{S}.\end{aligned}$$

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

$$\begin{aligned}16\mu^2\mathcal{S}^2+\left(4\lambda^2+\frac{16\sigma_1 k}{h^2}\right)\mathcal{S}\le 4,\\ 4\mu^2\mathcal{S}^2+\left(\lambda^2+\frac{4\sigma_1 k}{h^2}\right)\mathcal{S}\le 1.\end{aligned}$$

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

different wording

$$\begin{aligned}\frac{4\kappa^2k^2}{h^4}+\frac{c^2k^2+4\sigma_1k}{h^2}&\leq 1,\\ 4\kappa^2k^2+(c^2k^2+4\sigma_1k)h^2&\leq h^4,\\ h^4-(c^2k^2+4\sigma_1k)h^2-4\kappa^2k^2&\geq 0,\end{aligned}$$

which is a quadratic equation in h^2 with h bounded by

$$h\geq\sqrt{\frac{c^2k^2+4\sigma_1k+\sqrt{(c^2k^2+4\sigma_1k)^2+16\kappa^2k^2}}{2}}. \tag{4.25}$$

This 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 PDE (4.2) yields

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

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 of the scheme with $(\delta_{t\cdot}u_l^n)$ over discrete domain d

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

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

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

$$\delta_{t+}\mathfrak{h}=\mathfrak{b}-\mathfrak{q}, \tag{4.28}$$

where the boundary term $\mathfrak{b}$ appears after summation by parts shown shortly, and

$$\mathfrak{q}=2\sigma_0\rho A\|\delta_{t\cdot}u_l^n\|_d^2-2so\rho A \tag{4.29}$$

which, after summation by parts (see Section 3.2.2) and re-arranging the terms becomes

$$\delta_{t+}\mathfrak{h} = -2\sigma_0\rho A\|\delta_{t\cdot}u_l^n\| + 2\sigma_1\rho A\langle\delta_{t\cdot}u_l^n, \delta_{t-}\delta_{xx}u_l^n\rangle_d \tag{4.30}$$

Step 3: Check units

Step 4: Implementation

4.5 Modal analysis

To perform a modal analysis on the FD scheme of the damped stiff string in (4.5)

4.6 Implicit scheme

Although not used in the published work of this project, I would like 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:

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa^2\delta_{xxxx}u_l^n - 2\sigma_0\delta_{t\cdot}u_l^n + 2\sigma_1\delta_{t\cdot}\delta_{xx}u_l^n. \tag{4.31}$$

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. The stencil in Figure 4.2 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 3.1.3. The drawback is that this requires one matrix inversion per iteration which can be extremely costly (see Section **??**). However, both von Neumann and modal analysis (below) show that using the centred instead of the backwards operator has a positive effect on the stability and the modal behaviour of the scheme.

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

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{B}\mathbf{u}^n + \mathbf{C}\mathbf{u}^{n-1} \tag{4.32}$$

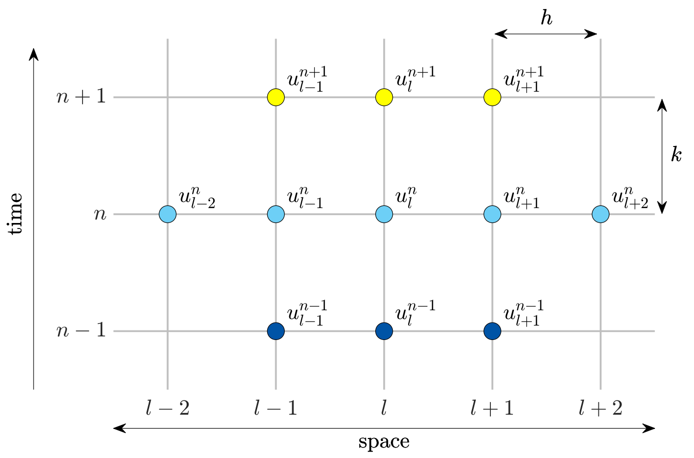


Fig. 4.2: The stencil for the damped stiff string scheme in (4.31).

4.6.1 von Neumann analysis

Following Section 3.3..

below is the exact same wording as above

$$\frac{1}{k^2}\left(z-2+z^{-1}\right) = -\frac{4c^2}{h^2}\sin^2(\beta h/2) - \frac{16\kappa^2}{h^4}\sin^4(\beta h/2) - \sigma_0 kz + \sigma_0 kz^{-1}$$
$$- \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)z + \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)z^{-1} \tag{4.33}$$

and after collecting the terms, the characteristic equation follows

$$\left(1+\sigma_0 k + \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z + \left(16\mu^2\sin^4(\beta h/2) + 4\lambda^2\sin^2(\beta h/2) - 2\right)$$
$$+\left(1-\sigma_0 k - \frac{4\sigma_1 k}{h^2}\sin^2(\beta h/2)\right)z^{-1} = 0. \tag{4.34}$$

Rewriting this to the form (3.32) and, again, using $\mathcal{S} = \sin^2(\beta h/2)$ yields:

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

stability of the system can be proven using condition (3.33). Continuing with $\mathcal{S} = \sin^2(\beta h/2)$ and substituting the coefficients into this condition yields

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

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

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

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

4.6.2 Modal analysis

As Eq. (4.32) matches the form in Eq. (3.62), one can perform a modal analysis by writing the scheme in one-step form as explained in Section 3.5.1

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 $k = 1/f_\text{s}$. 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:

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

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

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

6.1 Analysis Techniques in 2D

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

6.1.1 Frequency Domain Analysis

$p_x + p_y$

ansatz:

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

6.1.2 Energy Analysis

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

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

6.1.3 Modal Analysis

Stacked matrix form

6.2 2D Wave Equation

The 2D wave equation be used to model an ideal membrane such as done in
It has identical behaviour to the 2D waveguide mesh presented by van Duyne and Smith [18].

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

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

with wave speed $c = \sqrt{T/\rho H}$ (in m/s), tension per unit length (applied to the boundary)T (in N/m), material density ρ (in kg/m^3) and thickness H.

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.

$$\rho H \partial_t^2 u = -D \Delta\Delta u \tag{6.6}$$

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

6.4 Stiff membrane

Combination between Eqs. (6.7) and (6.6)

$$\rho H \partial_t^2 u = T \Delta u - D \Delta\Delta u \tag{6.7}$$

[F]

Part III

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 7 describes ways to connect different systems and Chapter 8 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 $l_\text{c} = \lfloor x_\text{c}/h \rfloor$ and $\alpha_\text{c} = x_\text{c}/h - l_\text{c}$ is the fractional part the location of interest.

$$I_0(x_\text{c}) = \begin{cases} 1, & \text{if } l = l_\text{c}, \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

$$I_1(x_\text{c}) = \begin{cases} (1-\alpha_\text{c}), & \text{if } l = l_\text{c} \\ \alpha_\text{c}, & \text{if } l = l_\text{c}+1 \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

$$I_3(x_\text{c}) = \begin{cases} ... \end{cases} \tag{6.10}$$

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

$$\langle f, J_p(x_\text{c}) \rangle_\mathcal{D} = I_p(x_\text{c}) f \tag{6.11}$$

Chapter 7

Connections

Something about connections

Pointlike $\delta(x - x_{\mathrm{c}})$ 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 7.1. 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 $\rho A h$. The weight of one grid point of a 2D plane can be calculated as $\rho H h^2$. 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.

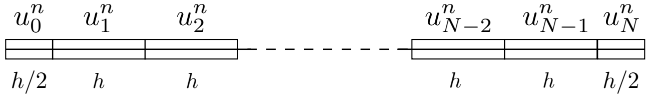


Fig. 7.1: Alternative interpretation of the discretisation of $u(x,t)$. The continuous system is divided into $N-1$ sections of width h plus 2 sections of width $h/2$ at the boundaries. Through this interpretation, the 'weight' of a grid point can be calculated from its physical parameters.

7.1 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.

7.2 Spring-like connections

7.2.1 Connection with rigid barrier (scaled)

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

$$\delta_{tt} u_l^n = \gamma^2 \delta_{xx} u_l^n + J(x_\text{c}) F^n \tag{7.1}$$

where

$$F^n = -\omega_0^2 \mu_{t\cdot} \eta^n - \omega_1^4 (\eta^n)^2 \mu_{t\cdot} \eta^n - 2\sigma_\times \delta_{t\cdot} \eta^n \tag{7.2}$$

and

$$\eta^n = I(x_\text{c}) u_l^n. \tag{7.3}$$

To obtain F^n, an inner product of scheme (7.1) needs to be taken with $J(x_\text{c})$ over domain $\mathcal{D}$ which, using identity (6.11) yields

$$\delta_{tt} I(x_\text{c}) u_l^n = \gamma^2 I(x_\text{c}) \delta_{xx} u_l^n + \underbrace{I(x_\text{c}) J(x_\text{c})}_{\|J(x_\text{c})\|_\mathcal{D}^2} F^n. \tag{7.4}$$

As u is connected to a rigid barrier according to (7.3), a shortcut can be taken and Eqs. (7.2) and (7.3) can be directly substituted into Eq. (7.4) to get

$$\delta_{tt} \eta^n = \gamma^2 I(x_\text{c}) \delta_{xx} u_l^n + \|J(x_\text{c})\|_\mathcal{D}^2 \left(-\omega_0^2 \mu_{t\cdot} \eta^n - \omega_1^4 (\eta^n)^2 \mu_{t\cdot} \eta^n - 2\sigma_\times \delta_{t\cdot} \eta^n \right). \tag{7.5}$$

and solved for η^{n+1}:

$$\begin{aligned} &\Big(1 + \|J(x_\text{c})\|_\mathcal{D}^2 k^2 [\omega_0^2/2 + \omega_1^4 (\eta^n)^2/2 + \sigma_\times/k]\Big) \eta^{n+1} \\ &\quad = 2\eta^n - \Big(1 + \|J(x_\text{c})\|_\mathcal{D}^2 k^2 [\omega_0^2/2 + \omega_1^4 (\eta^n)^2/2 - \sigma_\times/k]\Big) \eta^{n-1} \\ &\qquad + \gamma^2 k^2 I(x_\text{c}) \delta_{xx} u_l^n \end{aligned} \tag{7.6}$$

This can then be used to calculate F^n in (7.2) and can in turn be used to calculate u_l^{n+1} in (7.1).

7.2.2 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:

$$\partial_t^2 u = c^2\partial_x^2 u - \kappa_\text{s}^2\partial_x^4 u - 2\sigma_{0,\text{s}}\partial_t u + 2\sigma_{1,\text{s}}\partial_t\partial_x^2 u - \delta(x - x_\text{c})\frac{f}{\rho_\text{s}A} \tag{7.7a}$$

$$\partial_t^2 w = -\kappa_\text{p}^2\Delta\Delta w - 2\sigma_{0,\text{p}}\partial_t w + 2\sigma_{1,\text{p}}\partial_t\partial_x^2 w + \delta(x - x_\text{c}, y - y_\text{c})\frac{f}{\rho_\text{p}H} \tag{7.7b}$$

where

$$f = f(t) = K_1\eta + K_3\eta^3 + R\dot{\eta} \tag{7.8}$$

and

$$\eta = \eta(t) = u(x_\text{c}, t) - w(x_\text{c}, y_\text{c}, t) \tag{7.9}$$

Discrete

System (7.7) can then be discretised as

$$\delta_{tt}u_l^n = c^2\delta_{xx}u_l^n - \kappa_\text{s}^2\delta_{xxxx}u_l^n - 2\sigma_{0,\text{s}}\delta_{t\cdot}u_l^n + 2\sigma_{1,\text{s}}\delta_{t-}\delta_{xx}u_l^n - J_\text{s}(x_\text{c})\frac{f^n}{\rho_\text{s}A}, \tag{7.10}$$

$$\delta_{tt}w_l^n = -\kappa_\text{p}^2\delta_\Delta\delta_\Delta w_l^n - 2\sigma_{0,\text{p}}\delta_{t\cdot}w_l^n + 2\sigma_{1,\text{p}}\delta_{t-}\delta_{xx}w_l^n + J_\text{p}(x_\text{c}, y_\text{c})\frac{f^n}{\rho_\text{p}H}, \tag{7.11}$$

where

$$f^n = K_1\mu_{tt}\eta^n + K_3(\eta^n)^2\mu_{t\cdot}\eta^n + R\delta_{t\cdot}\eta^n, \tag{7.12}$$

and

$$\eta^n = I(x_\text{c})u_l^n - I(x_\text{c}, y_\text{c})w_l^n. \tag{7.13}$$

Expansion

System (7.10) can be expanded at the connection location x_c by taking an inner product of the schemes with their respective spreading operators.

7.2.3 Solving for f

7.2.4 Non-dimensional

The scaled system can be written as:

$$\partial_t^2 u = \gamma^2 \partial_x^2 u - \kappa_\mathrm{s}^2 \partial_x^4 u - 2\sigma_{0,\mathrm{s}} \partial_t u + 2\sigma_{1,\mathrm{s}} \partial_t \partial_x^2 u - \delta(x - x_\mathrm{c}) F \tag{7.14}$$

$$\partial_t w = -\kappa_\mathrm{p}^2 \Delta\Delta w - 2\sigma_{0,\mathrm{p}} \partial_t w + 2\sigma_{1,\mathrm{p}} \partial_t \partial_x^2 w + \delta(x - x_\mathrm{c}, y - y_\mathrm{c}) F \tag{7.15}$$

where

$$F = F(t) = \omega_1^2 \eta + \omega_3^4 \eta^3 + \sigma_\mathrm{c} \dot{\eta} \tag{7.16}$$

and

$$\eta = \eta(t) = u(x_\mathrm{c}, t) - w(x_\mathrm{c}, y_\mathrm{c}, t) \tag{7.17}$$

$$I(x_\mathrm{c}) \delta_{tt} u_l^n = c^2 \left(I(x_\mathrm{c}) \delta_{xx} u_l^n \right) + I(x_\mathrm{c}) J(x_\mathrm{c}) F \tag{7.18}$$

Chapter 8

Collisions

Something about collisions

8.1 Classic models

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

8.2 Michele's tricks

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

[9] 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.

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

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

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

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

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

[15] F. Orduña-Bustamente, "Auralization in space and in rooms of arbitrary d dimensions," in *Proceedings of the 14th Sound and Music Computing Conference*, 2017, pp. 250–253.

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

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

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

[19] 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.

[20] 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.

Part IV

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 **??** but the vertical spatial coordinate for 2D systems in fx. Chapter 6). Some might also be accompanied by a subscript in the main document

Symbol	Description	Unit
α	Fractional part of	
A	Cross-sectional area of string	m^2
c	Wave speed	m/s
$\frac{d^n}{dt^n}$	n^{th} order derivative with respect to t	-
∂_t^n	n^{th} order partial derivative with respect to t	-
δ_{t+}, δ_{t-}, $\delta_{t\cdot}$	Forward, backward and centred difference in time operator	-
δ_{x+}, δ_{x-}, $\delta_{x\cdot}$	Forward, backward and centred difference in space operator	-
δ_{tt}	Second order difference in time operator	-
δ_{xx}	Second order difference in space operator	-
δ_{xxxx}	Fourth order difference in space operator	-
μ_{t+}, μ_{t-}, $\mu_{t\cdot}$	Forward, backward and centred average in time operator	-
μ_{x+}, μ_{x-}, $\mu_{x\cdot}$	Forward, backward and centred average in space operator	-
μ_{tt}	Second order average in time operator	-
μ_{xx}	Second order average in space operator	-

Symbol	Description	Unit
E	Young's Modulus	Pa (kg·m^{-1}·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 ($= 1/f_\text{s}$)	s
K	Spring coefficient	N/m
κ	Stiffness coefficient	m^2/s (1D) m^4·s^{-2} (2D)
n	Sample index to grid function	-
N	Number of points string	-
$\mathbb{N}^0$	Set of non-negative integers	-
t	Time	s
T	Tension	N (1D) N/m (2D)
u	State variable	m
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	kg·m^{-3}
$\lfloor\cdot\rfloor$	Flooring operation	-
$\lceil\cdot\rceil$	Ceiling operation	-

Subscripts

c	Connection	

Symbol	Description	Unit
p	Plate	
s	String	

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
ODE	Ordinary differential equation
PDE	Partial differential equation

Appendix C

Code Snippets

C.1 Mass-Spring System (Section 2.3)

```matlab
%% Initialise variables
fs = 44100;                % sample rate [Hz]
k = 1 / fs;                % time step [s]
lengthSound = fs;          % length of the simulation (1 second) [samples]

f0 = 440;                  % fundamental frequency [Hz]
omega0 = 2 * pi * f0;      % angular (fundamental) frequency [Hz]
M = 1;                     % mass [kg]
K = omega0^2 * M;          % spring constant [N/m]

%% initial conditions (u0 = 1, d/dt u0 = 0)
u = 1;
uPrev = 1;

% initialise output vector
out = zeros(lengthSound, 1);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.35)
    uNext = (2 - K * k^2 / M) * u - uPrev;

    out(n) = u;

    % Update system states
    uPrev = u;
    u = uNext;
end
```

C.2 1D Wave Equation (Section 2.4)

```
%% Initialise variables
fs = 44100;          % Sample rate [Hz]
k = 1 / fs;          % Time step [s]
lengthSound = fs;    % Length of the simulation (1 second) [samples]

c = 300;             % Wave speed [m/s]
L = 1;               % Length [m]
h = c * k;           % Grid spacing [m] (from CFL condition)
N = floor(L/h);      % Number of intervals between grid points
h = L / N;           % Recalculation of grid spacing based on integer N

lambdaSq = c^2 * k^2 / h^2; % Courant number squared

% Boundary conditions ([D]irichlet or [N]eumann)
bcLeft = "D";
bcRight = "D";

%% Initialise state vectors (one more grid point than the number of
    intervals)
uNext = zeros(N+1, 1);
u = zeros(N+1, 1);

%% Initial conditions (raised cosine)
loc = round(0.8 * N);       % Center location
halfWidth = round(N/10);    % Half-width of raised cosine
width = 2 * halfWidth;      % Full width
rcX = 0:width;              % x-locations for raised cosine

rc = 0.5 - 0.5 * cos(2 * pi * rcX / width); % raised cosine
u(loc-halfWidth : loc+halfWidth) = rc; % initialise current state

% Set initial velocity to zero
uPrev = u;

% Range of calculation
range = 2:N;

% Output location
outLoc = round(0.3 * N);

%% Simulation loop
for n = 1:lengthSound

    % Update equation Eq. (2.43)
    uNext(range) = (2 - 2 * lambdaSq) * u(range) ...
        + lambdaSq * (u(range+1) + u(range-1)) - uPrev(range);

    % boundary updates Eq. (2.47)
    if bcLeft == "N"
        uNext(1) = (2 - 2 * lambdaSq) * u(1) - uPrev(1) ...
        + 2 * lambdaSq * u(2);
    end

    % Eq. (2.48)
    if bcRight == "N"
        uNext(N+1) = (2 - 2 * lambdaSq) * u(N+1) - uPrev(N+1) ...
        + 2 * lambdaSq * u(N);
    end

    out(n) = u(outLoc);

    % Update system states
    uPrev = u;
    u = uNext;
end
```

Part V

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.
- $\sigma_{1,\text{s}}$ in Eq. (21) should obviously be $\sigma_{1,\text{p}}$
- the unit of the spatial Dirac delta function δ should be m^{-1}

DigiDrum [F]

- σ_0 and σ_1 should be multiplied by ρH in order for the stability condition to hold.
- stability condition is wrong. Should be:

$$h \geq \sqrt{c^2k^2 + 4\sigma_1 k + \sqrt{(c^2k^2 + 4\sigma_1 k)^2 + 16\kappa^2 k^2}} \tag{1}$$

- 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.

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