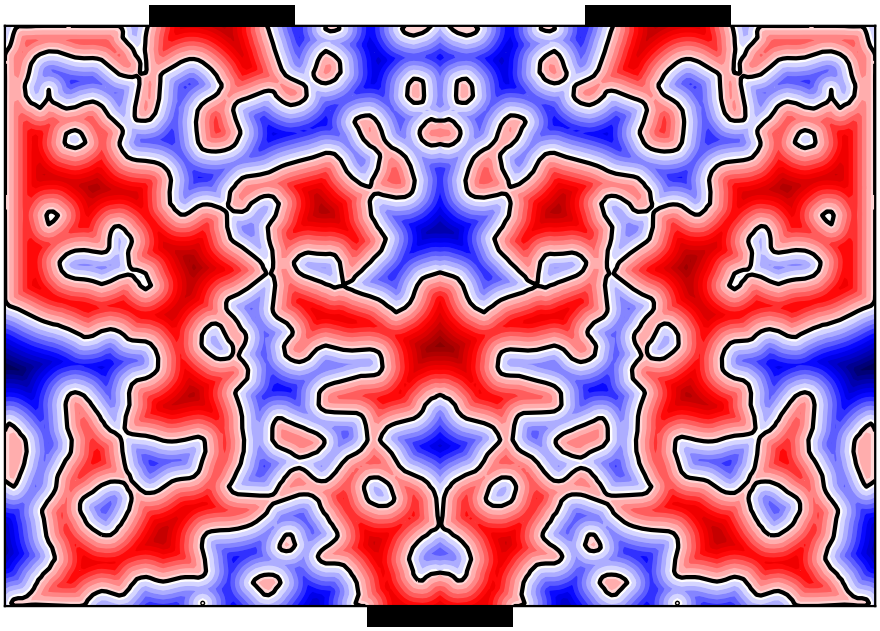


Inverse Design of Traveling-Wave Phononic Devices

Master's thesis in Physics

DAVID HAMBRAEUS

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Inverse Design of Traveling-Wave Phononic Devices

David Hambraeus

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Microtechnology and Nanoscience
Quantum Technology Laboratory
Quantum Photonics Laboratory
Chalmers University of Technology
Gothenburg, Sweden 2023

Inverse Design of Traveling-Wave Phononic Devices
David Hambraeus

Supervisor: Raphaël van Laer, Johan Kolvik, Paul Burger, QPL – Quantum technology division
Examiner: Raphaël van Laer, QPL – Quantum technology division

Master's Thesis 2023
Department of Microtechnology and Nanoscience
Quantum Technology Laboratory
Quantum Photonics Laboratory
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone: +46 31 772 1000

Cover: Signed distance field of optimized level-set beamsplitter design.

Typeset in LaTeX
Printed by Chalmers Digital Printing
Gothenburg, Sweden 2023

Inverse Design of Traveling-Wave Phononic Devices:

David Hambraeus
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

Phononic devices could enable and improve a broad range of functions in the realm of classical and quantum information processing. However, such devices are often designed through analytic methods, combined with brute-force parameter sweeps, which severely limits the designs that can be investigated. This work presents a method for inverse-design of traveling-wave phononic devices, allowing a vastly larger design space to be explored. At the heart of the method lies a fast calculation of gradients using the adjoint method, whereby the gradient computation costs no more than a single simulation. I show that this method is theoretically applicable to phononic devices, and demonstrate that it works in simulation. As a proof-of-concept, I attempt to design a phononic beamsplitter. The design process consists of two steps: one with continuously varying materials which is non-physical, but easier to optimize; and one with binary devices, accomplished through level-set methods. The first step yields near perfect performance, achieving less than one percent reflection and almost nothing scattering into other modes. The second step never reached as good performance, though still a 46/46 split was obtained with around 8 % of the power reflected. Though the resulting designs have some problems with small features and sensitive performance, this method looks promising for use in the design of future phononic devices.

Keywords: phononic devices, quantum acoustics, inverse design, adjoint method, level-set, gradient descent, solid mechanics.

Acknowledgements

First and foremost I would like to thank the members of the Quantum Photonics Laboratory for making this thesis project possible. I want to extend a special thank you to Paul and Johan, for their valuable comments and ideas, and to my supervisor and examiner Raphaël for fruitful discussions and

David Hambraeus, Gothenburg, June 2023

Table of contents

List of figures

List of tables

Acronyms

ADAM Adaptive Moment Estimation. ix, 16–18, 27

GD Gradient Descent. ix, 17, 18

PML Perfectly Matched Layer. ix, x, 22–25, 31–33

1. Introduction

In recent years, the research into quantum devices of different kinds has significantly intensified. Much of it is in one way or another connected to the construction or operation of a quantum computer. Though many people are focusing on superconducting circuits, where quanta of microwave-frequency light are manipulated, there has also been a growing interest in a different medium for quantum information: sound. More precisely, acoustic waves in solid materials. Just like how light, or electromagnetic waves, come in quantized packets of energy called photons, so too does acoustic waves and those packets are called phonons. Just recently, in 2019, researchers coupled an acoustic resonator to a transmon qubit and were able to directly resolve the number of phonons [1]. Acoustic or mechanical wave devices, also called phononic devices, have already found a multitude of applications in radio-signal processing and sensing, and research into new and better devices for these applications continues in parallel with devices for quantum applications [2].

Possible quantum applications of phononic devices are many. One of them is quantum memory [3]. Regular computers have both memory and a processing unit that retrieves data from memory, applies operations on it, and then returns it to memory. Keeping the data inside the processing unit would be inefficient, since the processing unit would then have to be very large. The same goes for quantum computers: storing the quantum information in the same place where the computation happens is not scalable. Instead, a quantum memory could be used to store information while it is not directly needed for the computation. The reason for using mechanical resonators as memory is that long lifetimes, on the order of seconds at 5 GHz, have been achieved in such resonators. Additionally, acoustic waves have a much smaller wavelength compared to electromagnetic waves at the same frequency, meaning that devices can be more compact [4]. Another application of quantum acoustics worth mentioning is coherent transduction between microwave and optical photons. This would enable communication between physically separated superconducting circuit based quantum computers [2]. A third possible application is doing quantum information processing purely in mechanical elements [5].

One important problem with such quantum acoustic devices is that they can be hard to design. Currently they are often designed by hand, through analytical derivations or analytically motivated guesswork, combined with brute force parameter sweeps. However, this severely limits the designs that can be investigated: sometimes analytical derivations are impossible, and brute force parameter sweeps quickly become computationally intractable. A parameter sweep of just 6 parameters with 10 different values for each requires 1 000 000 simulations. One can of course use smarter optimization algorithms like Bayesian optimization or particle swarm optimization [6],

[7] to decrease the number of simulations needed, but they will still struggle with high dimensional problems [8].

A different approach that has been gaining some popularity in the field of nano-photonics is *inverse design with adjoint simulation* [9]. The idea is that if the gradient of the figure of merit of the device with respect to the design parameters can be calculated, then gradient based optimization methods can be used. These converge much faster, even if the number of parameters is very large. Thus, one hopes to be able to search among a much more general class of designs for the optimal one. This has been successfully applied to photonic devices, where a wide variety of components have been designed [10]. In some cases, for example the waveguide bend, the inverse-designed device could be made much more compact than conventionally designed bends [11]. In other cases, for example the vertically incident wavelength-demultiplexing grating coupler, there are no other known conventional methods of designing the device [12]. Before their application in nano-photonics, closely related methods were used in structural mechanics under the name of topology optimization, for example to optimize the stiffness with limited material [13]. It has also been used for engineering some properties (e.g. bandgap) of phononic crystals [14]. However, to our knowledge, it has not yet been used for traveling-wave phononic devices.

Phononic devices have in general been studied less than photonic devices, and the library of known devices is smaller. With this thesis, we explore the possibility of extending the inverse-design paradigm to traveling-wave phononic devices. Since both acoustics and electromagnetics are wave phenomena, there are many analogies to be drawn, but there are also important differences. For example, the different boundary conditions mean that acoustics support surface waves while electromagnetics does not. We show in this work that the adjoint method is applicable to traveling-wave phononics, and derive the form of the equations. As a proof of concept, we attempt to design a phononic 50/50 beamsplitter. The beamsplitter is conceptually one of the simplest devices imaginable, and photonic beamsplitters have been studied in great detail for decades and found a multitude of uses in different experiments. However, there is still no standard implementation for phononic beamsplitters, though [5] has demonstrated a working beamsplitter for surface acoustic waves. There are two major differences between their design and the one attempted here. Firstly, they are using surface acoustic waves while we are using waves propagating through patterned silicon waveguides. Secondly, their beamsplitter reflects one of the arms of the beamsplitter back towards the source, whereas we have one input waveguide and two separate outputs.

1.1 Aim and Thesis Outline

The aims of this thesis are:

- Rederive the equations for inverse design with adjoint simulation in the case of phononics and confirm that the methods are theoretically applicable.

- Implement the methods and use them to design a phononic beamsplitter.

Chapter 2 presents some of the theory on solid mechanics and acoustic waves needed to understand the thesis. A band diagram over the modes of the waveguide used is also shown there. In chapter 3, the inverse design process is presented. First, the general adjoint method is showcased, followed by a derivation in the specific case of acoustics. Then, gradient based optimization algorithms are discussed and the ones used in this thesis are presented. Chapter 4 describes the specifics of the simulations performed in this work: both a specification of the skeleton of the devices, how the input wave was excited, and how the designs were parametrized. It also describes the specifics of the optimization algorithm, including parameter values. Chapter 5 then presents and analyzes the results of the simulations performed and Chapter 6 summarizes the conclusions drawn from the results and describes possible future research.

2. Acoustic Waves and Waveguides

This chapter presents the theory of the physics behind our simulations. Section 2.1 gives a review of solid mechanics in the frequency domain, culminating in equation (2.13), which is the equation that is solved each simulation. Section 2.2 goes on to show that the solutions to this equation with no external forces for infinite periodic structures can be obtained from simulating a single unit cell of the structure. Lastly, the solutions for the specific periodic structure used for the inputs and outputs in this thesis are shown with a band diagram as well as the shapes of the modes.

2.1 The Frequency Domain Acoustic Equation

To efficiently model the deformation and stresses in a solid material, a linear elasticity model is often assumed. This means the force is assumed to be linear with displacement, which is a valid approximation for small displacements. The basic equation is Hooke's law, $F = kx$ for ideal springs. Since we are dealing with solid elements in three dimensions rather than ideal springs in one dimension, the equation changes slightly but the essence is still the same. Hooke's law in it's full form is

$$\sigma = C : \epsilon \tag{2.1}$$

where σ is the stress tensor, C the elasticity tensor, a rank four tensor that is a property of the material,

$$\epsilon := \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\intercal) \tag{2.2}$$

is the strain tensor, $\boldsymbol{u}$ is the displacement, and : denotes double scalar product[1]. Equation (2.1) is linear in $\boldsymbol{u}$, hence the name *linear* elasticity. Using this and Newton's equations of motion, the equation governing the dynamics is obtained:

$$\rho \ddot{\boldsymbol{u}} = \nabla \cdot \sigma + \boldsymbol{F}, \tag{2.3}$$

where ρ is the density and $\boldsymbol{F}$ is the externally applied force. If the force is sinusoidally varying with some frequency ω, $\boldsymbol{F}(\boldsymbol{x}, t) = \boldsymbol{F}(\boldsymbol{x})e^{i\omega t}$, then $\boldsymbol{u}$ is also expected to be oscillating with that same frequency: $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x})e^{i\omega t}$. Inserting this in equation (2.3) gives

$$-\rho\omega^2 \boldsymbol{u} = \nabla \cdot \sigma + \boldsymbol{F}. \tag{2.4}$$

To combine equations (2.1) to (2.4) into one equation that can be solved for $\boldsymbol{u}$, we first rewrite them in index notation to make calculations clearer. Note that

[1] See equation (2.6) for what the double scalar product means in this case. It is in general a product that contracts two indices, as opposed to the regular scalar product that contracts only one.

throughout this thesis, the Einstein summation convention is used, meaning that repeated indices are summed. In index notation the equations become

$$\epsilon_{ij} = \frac{1}{2}(\partial_i u_j + \partial_j u_i) \tag{2.5}$$
$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} \tag{2.6}$$
$$= \frac{1}{2}\left(C_{ijkl}\partial_k u_l + C_{ijkl}\partial_l u_k\right) \tag{2.7}$$
$$= C_{ijkl}\partial_k u_l \text{ because of the symmetry } C_{ijkl} = C_{ijlk} \tag{2.8}$$

which gives

$$F_i = -\rho\omega^2 u_i - \partial_j \sigma_{ij} \tag{2.9}$$
$$= -\rho\omega^2 \delta_{ik} u_k - \partial_j \left(C_{ijkl}\partial_l u_k\right) \tag{2.10}$$
$$= -\left(\rho\omega^2 \delta_{ik} \cdot + \partial_j \left(C_{ijkl}\partial_l \cdot\right)\right) u_k \tag{2.11}$$

where the indices i, j, k, l go over the spatial dimensions x, y, z. The tensors in the equation above are really tensor fields, i.e. they are functions of $\boldsymbol{x}$. Defining the operator

$$\hat{A}_{ik} = -\left(\rho\omega^2 \delta_{ik} \cdot + \partial_j \left(C_{ijkl}\partial_l \cdot\right)\right) \tag{2.12}$$

we can write

$$\hat{A}_{ik} u_k = F_i. \tag{2.13}$$

Solving this equation for $\boldsymbol{u}$ thus gives the response of the system to periodically varying force $\boldsymbol{F}(\boldsymbol{x}, t) = \boldsymbol{F}(\boldsymbol{x})e^{i\omega t}$.

2.2 Bloch States and Band Diagrams

With no external forces, i.e. $F_i = 0$, equation (2.13) can be written as

$$\frac{1}{\rho}\partial_j \left(C_{ijkl}\partial_l u_k\right) = \omega^2 u_i, \tag{2.14}$$

which is an eigenvalue equation for the operator $\hat{O}_{ik} = \frac{1}{\rho}\partial_j \left(C_{ijkl}\partial_l \cdot\right)$ where eigenvalues are the angular frequency squared. If furthermore the structure is periodic, then the eigenstates are so called *Bloch states*.

If the structure is periodic in the y direction with some periodicity $\boldsymbol{a} = a\hat{\boldsymbol{y}}$, meaning that $C_{ijkl}(\boldsymbol{x}) = C_{ijkl}(\boldsymbol{x} + n\boldsymbol{a})$ and $\rho(\boldsymbol{x}) = \rho(\boldsymbol{x} + n\boldsymbol{a})$ where n is an integer, then this operator commutes with the translation operator $\hat{T}_{\boldsymbol{a}}[\boldsymbol{u}(\boldsymbol{x})] = \boldsymbol{u}(\boldsymbol{x} + \boldsymbol{a})$. This means that there is a basis of simultaneous eigenstates of both operators. The eigenfunctions of the translation operator are $\boldsymbol{f}(x, z)e^{ik_y y}$, where $\boldsymbol{f}$ is an arbitrary function, and the eigenvalues are $e^{ik_y a}$. Defining the reciprocal lattice constant $b = 2\pi/a$, we see that the functions $\boldsymbol{f}(x, z)e^{i(k_y + mb)y}$ for integer values of m all have the same eigenvalue, which means that they form a degenerate subspace of eigenfunctions. This also

means that we can restrict ourselves to the first Brillouin zone: $k_y \in [-b/2, b/2]$, since any k_y outside this interval can be written as $k_y' + mb$ with $k_y' \in [-b/2, b/2]$. Thus, the simultaneous eigenstate $\boldsymbol{u}_{k_y,\omega^2}$ can be written

$$\boldsymbol{u}_{k_y,\omega^2}(\boldsymbol{x}) = \sum_m \boldsymbol{f}_{m,k_y,\omega^2}(x,z)e^{i(k_y+mb)y} = e^{ik_y y}\tilde{\boldsymbol{f}}_{k_y,\omega^2}(\boldsymbol{x}) \tag{2.15}$$

where $\tilde{\boldsymbol{f}}_{k_y,\omega^2}$ is a periodic function with periodicity $\boldsymbol{a}$ by construction [15].

The solutions to these eigenvalue equations are often called modes, and each mode has both a frequency and a wave vector. This gives rise to a band diagram, where the frequency is plotted as a function of the wave vector.

2.2.1 Concrete Example of a Phononic Crystal

In this work, a rectangular silicon waveguide patterned with elliptic holes is used. The structure is clamped at the bottom (meaning that a fixed boundary condition is used there, enforcing $\boldsymbol{u} = 0$) while the other sides are free. A top down schematic of the unit cell can be seen in figure 2.1. The reason for using this specific unit cell is that it has been shown to enable good coupling between optical and mechanical modes while still being clamped to a substrate, which potentially makes it more scalable for phononic circuitry than suspended designs [16]. For our proof-of-concept device, we do not model the substrate to which the waveguide would be clamped, instead using a fixed boundary condition on the bottom.

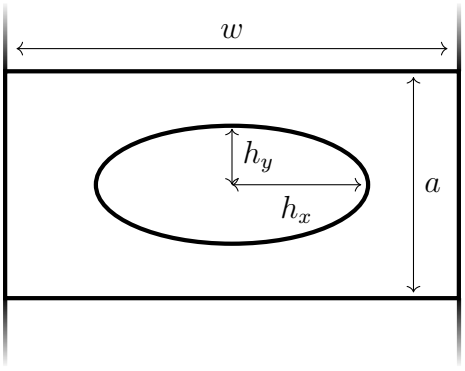


Figure 2.1: *Top down view of unit cell of a phononic crystal. The periodicity of the waveguide is a, the width is w and h_x and h_y are the major and minor radii of the elliptic hole.*

An infinite waveguide can be simulated with just one unit cell using periodic boundary conditions at the edge where the unit cell would be attached to the next one. As section 2.2 showed, waves with any $\boldsymbol{k}$ can be investigated with a single unit cell since $\boldsymbol{u}$ is a phase factor $e^{i\boldsymbol{k}\cdot\boldsymbol{x}}$ times some function with the same periodicity as the unit cell. Enforcing a wave vector $\boldsymbol{k} = k\hat{\boldsymbol{y}}$ thus entails using periodic boundary conditions with a specified phase shift over the unit cell. These are called *Floquet periodic boundary conditions* in COMSOL, which is the simulation software used throughout this thesis. Running simulations with different k to find the eigenmodes with their

corresponding frequencies for this structure yields the band diagram in figure 2.2. The parameter values used in the simulations can be found in table 4.1. Figure 2.3 shows the shapes of the eight lowest lying eigenmodes at $k = 0.9\pi/a$.

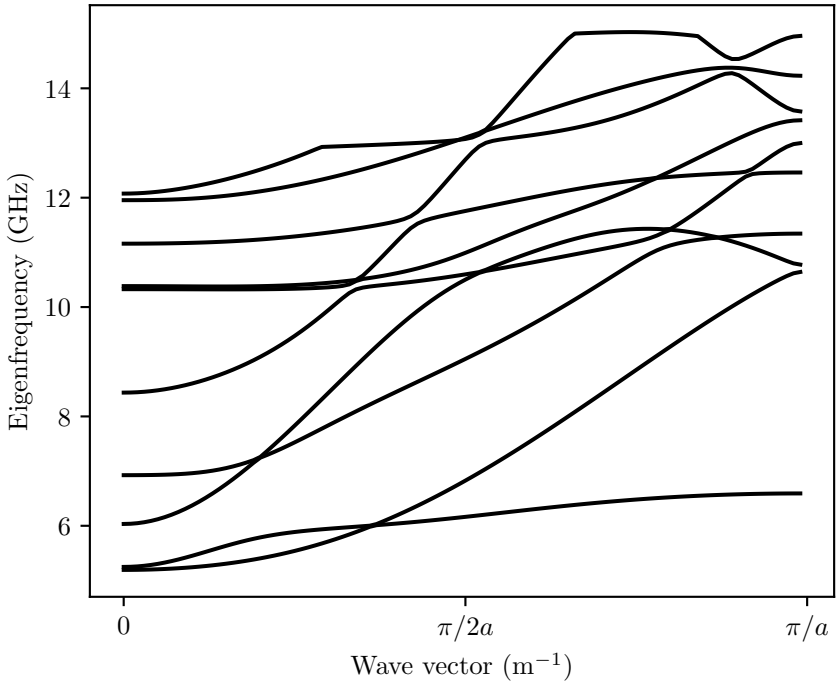
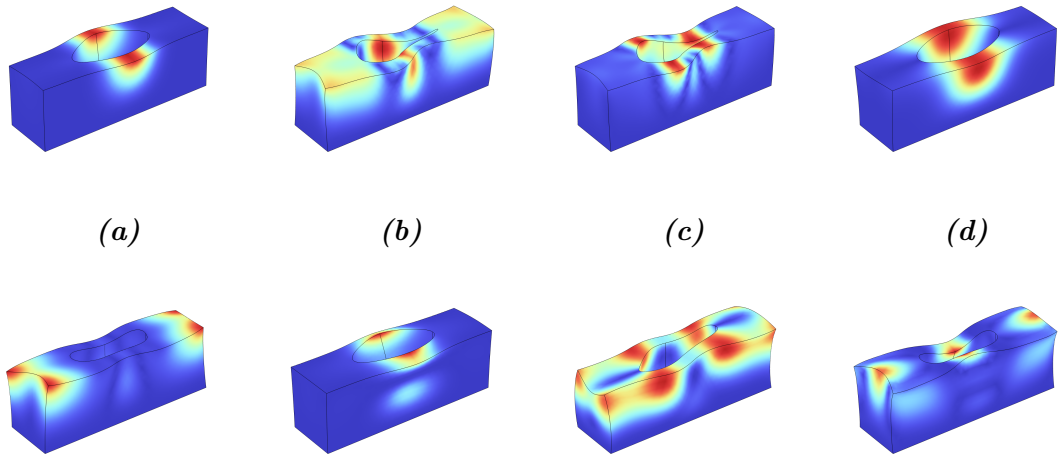


Figure 2.2: *Band diagram of phononic crystal defined in figure 2.1.*



(a) ***(b)*** ***(c)*** ***(d)***

(e) ***(f)*** ***(g)*** ***(h)***

Figure 2.3: *Mode shapes for the lowest eight modes at $k = 0.9\pi/a$. The color denotes the absolute value of the displacement, and the scale is normalized for each figure.*

3. Inverse Design in Acoustics

Inverse design is a design paradigm where the design of a device is guided fully by the desired characteristics. These desired characteristics are quantified through what is called an objective function[2], which we will denote f_{obj}, that should be maximized. When coupled with *adjoint simulation*, which is a clever way to compute gradients, and gradient based optimization algorithms, this becomes a powerful methodology.

An overview of the design process is as follows:

1. Initialize a random device design.
2. Calculate the gradient of the design through the adjoint method.
3. Update the device design using the gradient according to the optimization algorithm.
4. If the device performance is good enough, terminate optimization, else return to step 2.

3.1 Adjoint Simulation

Adjoint simulation is a way to compute the gradient of f_{obj} with respect to the design, which in our case means with respect to the material parameters. We will in this section first give a general derivation, following reference [17]. In section 3.1.2, we will then derive the specifics when applying this to acoustics.

3.1.1 General Derivation

Let f_{obj} be a function which depends on some high-dimensional vector v. The vector v can be calculated by solving the linear equation $Av = b$, where b is a fixed vector and A is a matrix that depends on a vector of design parameters p. This could be the acoustics equation, but it might also be the analogous equation for photonics, or something completely different like fluid dynamics. We will refer to the process of solving this equation as a simulation, since for this thesis it is solved through the simulation software COMSOL. The overall goal is to find the parameters p that maximize the objective function f_{obj}. The goal of adjoint simulation is to find $\frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}p}$[3].

[2] Also called *figure of merit (FoM)*.

[3] Note that this is a total derivative, not a partial one. In our case this is obvious since f_{obj} only depends on v but in general it could also depend directly on p.

This can be expanded through the chain rule as

$$\frac{\mathrm{d}f_{\mathrm{obj}}}{\mathrm{d}p} = \frac{\mathrm{d}f_{\mathrm{obj}}}{\mathrm{d}v}\frac{\mathrm{d}v}{\mathrm{d}p}. \tag{3.1}$$

To find the latter factor we do

$$\frac{\mathrm{d}}{\mathrm{d}p}[Av = b] \implies \frac{\mathrm{d}A}{\mathrm{d}p}v + A\frac{\mathrm{d}v}{\mathrm{d}p} = 0 \tag{3.2}$$

$$\implies A\frac{\mathrm{d}v}{\mathrm{d}p} = -\frac{\mathrm{d}A}{\mathrm{d}p}v. \tag{3.3}$$

This almost gives us the right thing, but we need to rewrite equation (3.1) a little. If we define a $\tilde{v}$ through the equation

$$\frac{\mathrm{d}f_{\mathrm{obj}}}{\mathrm{d}v} = \tilde{v}A \tag{3.4}$$

then plugging this into equation (3.1) and using equation (3.3) we obtain

$$\frac{\mathrm{d}f_{\mathrm{obj}}}{\mathrm{d}p} = \tilde{v}A\frac{\mathrm{d}v}{\mathrm{d}p} \tag{3.5}$$

$$= -\tilde{v}\frac{\mathrm{d}A}{\mathrm{d}p}v. \tag{3.6}$$

Finding $\tilde{v}$ from equation (3.4) amounts to solving the so called *adjoint problem*:

$$A^{\dagger}\tilde{v}^{\dagger} = \frac{\mathrm{d}f_{\mathrm{obj}}}{\mathrm{d}v}^{\dagger} \tag{3.7}$$

hence the name adjoint method. As it turns out, A is in many cases symmetric (or self-adjoint) which means that this is simply a normal simulation but with $\mathrm{d}f_{\mathrm{obj}}/\mathrm{d}v^{\dagger}$ as the source, and even if it isn't, solving the adjoint problem isn't more computationally expensive than running a normal simulation. Thus, the derivative can be obtained with just twice the simulation time compared to an ordinary simulation.

Now you might be wondering: what have we gained by this? Let n be the dimension of v, m the dimension of p and l the dimension of b. This means that A is a matrix with dimension $l \times n$ and $\mathrm{d}A/\mathrm{d}p$ is a rank three tensor with dimension $m \times l \times n$. Thus solving for $\mathrm{d}v/\mathrm{d}p$ from equation equation (3.3) means solving for an $n \times m$ matrix, which is much more computationally expensive than solving for just a vector of dimension n.

3.1.2 Specific Derivation with Acoustics

Now we turn to the specific case of acoustic devices. Here $Av = b$ is replaced by the acoustic field equation (equation (2.13)):

$$\hat{A}_{ik}u_k = F_i. \tag{3.8}$$

Instead of vectors, like we saw in section 3.1.1, these quantities are now functions[4] of $\boldsymbol{x}$. Analogously to the vector of design parameters we now have a *design field* $p(\boldsymbol{x})$, and analogously to f_{obj} being a function of a vector, this f_{obj} is a function of a function, i.e. a *functional*. For a quick overview of functionals and their derivatives, see box 3.1.

Box 3.1: On functionals and their derivatives

The f_{obj} is no longer a function, but rather a *functional*, and thus we need to use the functional derivative instead of the ordinary derivative. One can think of a functional as a function of a function, i.e. something that maps an element of a function space to a scalar number. There are also functionals which depend on both a function and a real number, or on multiple functions. Below we will give an overview of the notational conventions used, and give the definition of the functional derivative, as well as some useful properties of it.

Let $\mathcal{Y}$ be a function space of functions $\mathbb{R} \to \mathbb{R}$. A functional $F : \mathcal{Y} \to \mathbb{R}$ evaluated at the function $f \in \mathcal{Y}$ is denoted with the function in square brackets: $F[f]$. Note that in principle, F is the functional while $F[f]$ is just a number, analogously to how f is a function while $f(x)$ is a real number. If the functional additionally depends on a real number, $G : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, that is put in round brackets: $G[f](x)$.

The functional derivative of F with respect to its function argument is a functional $\mathcal{Y} \times \mathbb{R} \to \mathbb{R}$ denoted $\delta F[f]/\delta f$. In this expression, f is technically a dummy function, writing $\delta F[g]/\delta g$ is exactly the same functional. However, often the argument of F is omitted and the function in the denominator is named in accordance with the names in the definition of the functional. Furthermore, the same notation is also often used to denote the functional derivative evaluated at a certain function. For example, if we define a functional taking two function arguments $F[f_1, f_2] = \int f_1(x) + f_2(x)\,\mathrm{d}x$, one can write

$$\frac{\delta F}{\delta f_2}(x) \qquad \text{meaning} \qquad \frac{\delta F[g_1, g_2]}{\delta g_2}(x) \tag{3.9}$$

$$\frac{\delta F}{\delta f_2}(x) \qquad \text{meaning} \qquad \frac{\delta F[g_1, g_2]}{\delta g_2}[f_1, f_2](x) \tag{3.10}$$

where in the latter case, f_1 and f_2 are specific functions defined previously. Also note that the scalar argument to the functional derivative can be put either in the denominator or after, depending on what is more convenient. Oftentimes, when the functional being differentiated is a long expression and is placed after the fraction, the argument is placed in the denominator:

$$\frac{\delta F}{\delta f}(x) = \frac{\delta F}{\delta f(x)} = \frac{\delta}{\delta f(x)} F \tag{3.11}$$

[4]Vector-valued functions, but that is not the important part here.

The functional derivative is defined by

$$\int \frac{\delta F}{\delta f}(x)\varphi(x)\,\mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}\varepsilon}F[f+\varepsilon\varphi] \tag{3.12}$$

where F is a functional of f and φ is an arbitrary test function. We will use two properties of the functional derivative:

- If F is the functional $F[f](y) = f(y)$, then $\delta F(y)/\delta f(x) = \delta(y-x)$.
- The chain rule: if F is a functional with one function argument, G is a functional with one function and one real argument, and H is the functional defined as $H[f] = F[G[f](y)]$, then

$$\frac{\delta H}{\delta f}(x) = \int \frac{\delta F}{\delta G[f]}(y)\frac{\mathrm{d}G(y)}{\mathrm{d}f}(x)\,\mathrm{d}y \tag{3.13}$$

For simplicity we will limit ourselves to the case where the objective function is an overlap integral of the displacement field $u_k(\boldsymbol{x})$ with some function $\varphi_k^*(\boldsymbol{x})$:

$$f_{\mathrm{obj}}[\boldsymbol{u}] = \int_\Omega u_i(\boldsymbol{x})\varphi_i^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{3.14}$$

where Ω is the domain of $\boldsymbol{u}$. Such an integral is an inner product in the space of functions on Ω.

Analogously to the general derivation in section 3.1.1, the chain rule is used to expand $\delta f_{\mathrm{obj}}/\delta p(\boldsymbol{x})$, see box 3.1 for the form of the chain rule for the functional derivative. However, because $\boldsymbol{u}$ is in general complex, we will split it into its real and imaginary components: $\boldsymbol{u} = \boldsymbol{v} + i\boldsymbol{w}$.

$$\frac{\delta f_{\mathrm{obj}}}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y}\frac{\delta f_{\mathrm{obj}}}{\delta v_i}(\boldsymbol{y})\frac{\delta v_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) + \frac{\delta f_{\mathrm{obj}}}{\delta w_i}(\boldsymbol{y})\frac{\delta w_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.15}$$

The first factor of each of the two terms is easy enough to calculate:

$$\begin{aligned}
\frac{\delta f_{\mathrm{obj}}}{\delta v_i}(\boldsymbol{y}) &= \frac{\delta}{\delta v_i(\boldsymbol{y})}\int_\Omega u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.16)\\
&= \int_\Omega \frac{\delta}{\delta v_i(\boldsymbol{y})}u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.17)\\
&= \int_\Omega \delta(\boldsymbol{x}-\boldsymbol{y})\delta_{ij}\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.18)\\
&= \varphi_i^*(\boldsymbol{y}) && (3.19)
\end{aligned}$$

and

$$\frac{\delta f_{\text{obj}}}{\delta w_i}(\boldsymbol{y}) = \frac{\delta}{\delta w_i(\boldsymbol{y})} \int_\Omega u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{3.20}$$

$$= \int_\Omega \frac{\delta}{\delta w_i(\boldsymbol{y})} u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{3.21}$$

$$= \int_\Omega i\delta(\boldsymbol{x}-\boldsymbol{y})\delta_{ij}\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{3.22}$$

$$= i\varphi_i^*(\boldsymbol{y}) \tag{3.23}$$

which gives us

$$\frac{\delta f_{\text{obj}}}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\frac{\delta v_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) + i\varphi_i^*(\boldsymbol{y})\frac{\delta w_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.24}$$

$$= \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\mathrm{Re}\left(\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x})\right) + i\varphi_i^*(\boldsymbol{y})\mathrm{Im}\left(\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x})\right) \tag{3.25}$$

$$= \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.26}$$

To find $\delta u_i(\boldsymbol{y})/\delta p(\boldsymbol{x})$ we apply $\delta/\delta p(\boldsymbol{x})$ to equation (3.8), which gives us

$$0 = \frac{\delta \hat{A}_{ik}(\boldsymbol{y})}{\delta p}(\boldsymbol{x})u_k(\boldsymbol{y}) + \hat{A}_{ik}(\boldsymbol{y})\frac{\delta u_k(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.27}$$

Just like equation (3.5), we want an adjoint field $\tilde{\boldsymbol{u}}$ such that the integral in equation (3.26) is

$$\int_\Omega \mathrm{d}\boldsymbol{y}\,\varphi_i^*(\boldsymbol{y})\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y}\,\tilde{u}_i(\boldsymbol{y})\hat{A}_{ik}(\boldsymbol{y})\frac{\delta u_k(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.28}$$

$$= -\int_\Omega \mathrm{d}\boldsymbol{y}\,\tilde{u}_i(\boldsymbol{y})\frac{\delta \hat{A}_{ik}(\boldsymbol{y})}{\delta p}(\boldsymbol{x})u_k(\boldsymbol{y}). \tag{3.29}$$

The adjoint field is found by solving the adjoint problem:

$$\hat{A}_{ik}^\dagger \tilde{u}_k = \varphi_i^*, \tag{3.30}$$

where the adjoint operator $\hat{A}_{ik}^\dagger$ is defined through

$$\int_\Omega f(\boldsymbol{x})\left(\hat{A}_{ik}(\boldsymbol{x})g(\boldsymbol{x})\right)\mathrm{d}\boldsymbol{x} = \int_\Omega \left(\hat{A}_{ik}^\dagger(\boldsymbol{x})f(\boldsymbol{x})\right)g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{3.31}$$

To further simplify this expression, the dependence of $\hat{A}$ on p must be specified. A linear dependence is proposed here, though extending the formula to more complicated dependences is rather easy. Taking $\rho(\boldsymbol{y}) = \rho^0 p(\boldsymbol{y})$ and $C_{ijkl}(\boldsymbol{y}) = C_{ijkl}^0 p(\boldsymbol{y})$ in the definition of $\hat{A}_{ik}$ from equation (2.12) implies

$$\frac{\delta \hat{A}_{ik}(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) = -\rho^0\omega^2\delta_{ik}\delta(\boldsymbol{x}-\boldsymbol{y}) - \partial_j\left(C_{ijkl}^0\delta(\boldsymbol{x}-\boldsymbol{y})\partial_l\,\cdot\right) \tag{3.32}$$

$$= -\rho^0\omega^2\delta_{ik}\delta(\boldsymbol{x}-\boldsymbol{y}) - 2C_{ijkl}^0\delta(\boldsymbol{x}-\boldsymbol{y})\partial_j\partial_l. \tag{3.33}$$

Plugging this back into the integral in equation (3.29) gives

$$\omega^2 \rho^0 \tilde{u}_i(\boldsymbol{x}) u_i(\boldsymbol{x}) + 2C^0_{ijkl} \tilde{u}_i(\boldsymbol{x}) \partial_j \partial_l u_k(\boldsymbol{x}) \tag{3.34}$$

which is comparatively easily evaluated.

To summarize: in order to compute the gradient $\delta f_{\text{obj}}/\delta p(\boldsymbol{x})$ we first run a normal simulation, i.e. solve equation (3.8), to obtain $\boldsymbol{u}$. Then we run an adjoint simulation, i.e. we solve equation (3.30), to obtain $\tilde{\boldsymbol{u}}$. Finally, we calculate $\delta A(\boldsymbol{y})/\delta p(\boldsymbol{x})$, which is given in equation (3.33) in the case of a linear material parameter dependence on p, and obtain the gradient through equation (3.29). This gradient tells us *at every point in the design* if p should be increased or decreased in order to increase f_{obj}. The next section will describe more precisely how this is used to optimize the design.

3.2 Optimization Algorithms

In this section we motivate why the gradient is so useful and how it will be used in our algorithm. We begin by describing the advantages of gradient based optimization algorithms over those that don't use the gradient. Following that, we describe gradient descent, the most basic gradient based optimization method, as well as Adaptive Moment Estimation (ADAM), which is the algorithm that we have used.

An optimization algorithm is an algorithm for finding the optimum of a function. The function is often called the *objective function* or the *cost function*. A very naive optimization method would be to simply try some number of inputs and then choose the one with the highest function value. This requires a large number of points before a good value is found, meaning that it takes a long time. An improvement to this method is to use the information gained from the points already tried to decide which points to try next. If some point has a bad value, then try somewhere else; if some point has a good value, try another close by. Examples of algorithms that do this are bayesian optimization, particle swarm optimization, and various forms of genetic algorithms [6]. However, if the domain of the objective function is very high-dimensional, "close by" is a very large space. For such functions, it is essential to know in which direction the function increases. That is why gradient based optimization algorithms are so powerful; they enable us to quickly find the right direction to go in for best improvement.

3.2.1 Gradient Descent

The simplest gradient based optimization algorithm is called *gradient descent*.[5] Like all of the algorithms we will describe, it's an iterative algorithm, meaning that it

[5]The reason for calling it gradient descent, instead of gradient ascent is simply a convention. Any maximization problem can be made into a minimization problem by multiplying that objective function by -1, so which formulation one chooses is arbitrary. However, the convention of talking about gradient descent is so strong that I will stick to that terminology, even though I am solving a maximization problem.

generates a sequence of points that converges to an optimum, and the next point in the sequence is derived from the previous ones. In the case of gradient descent, the next point is gotten by

$$p_n = p_{n-1} + \eta g_{n-1} \tag{3.35}$$

where η is the so called *learning rate* and g_{n-1} is the gradient of the objective function at p_{n-1}. For ordinary gradient descent the learning rate would be fixed, and choosing an appropriate value for this parameter is one of the problems of this method. If a too high value is chosen, then the steps taken will be too large and the optimum might be missed entirely. A too low value results in too small steps which will yield a slow convergence. Most gradient descent implementations nowadays use a so called learning schedule, which means that η is not constant during the optimization. This introduces the additional problem of choosing how fast and between which values it should change.

3.2.2 Adaptive Moment Estimation (ADAM)

The ADAM algorithm is an improved version of gradient descent. It has three main differences:

1. Each dimension has a separate learning rate.

2. The learning rate is automatically set from the previously seen gradients.

3. The evolution carries some momentum.

Figure 3.1 shows the difference in performance between ADAM and Gradient Descent (GD). With a slightly too large learning rate, the GD algorithm gets stuck in an oscillation and makes little progress towards the minimum. If the learning rate is decreased, the oscillations disappear but the stepsize is now too small to make it all the way to the true minimum. Since the ADAM algorithm has some momentum, the motion along the valley gets compounded while the motion perpendicular gets dampened which means that the oscillations are not as much of a problem. The adaptive learning rate also means that the algorithm doesn't get stuck prematurely due to the small gradient at the bottom of the valley. Admittedly, this objective function is specifically chosen to showcase the advantages of ADAM, but it has been shown to outperform GD in almost all cases [18].

Listing 3.1 shows a pseudocode implementation of the ADAM algorithm, which has four important hyperparameters. The first is α which controls the order of the step size. The step will never be much larger than α in any dimension. β_1 and β_2 control how quickly the momentum decays. The algorithm basically keeps an exponentially decaying average of the previous gradients and previous squared gradients. The β:s set the decay rate for these averages. Finally, there is ϵ, which serves to make the algorithm more stable. If the average magnitude of the gradient becomes much smaller than ϵ, the step size of the algorithm will decrease.

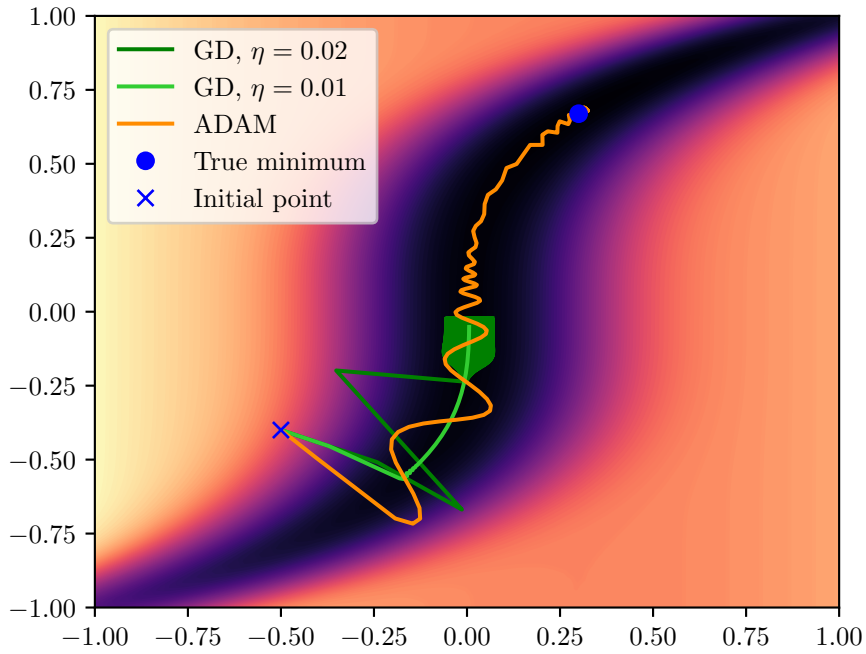


Figure 3.1: *A comparison between ADAM and GD in an optimization landscape with a narrow canyon. The two different GD algorithms are shown with 1000 steps, while 200 steps with the ADAM algorithm are shown.*

```
def adam(p_0, get_gradient, alpha,
                beta_1, beta_2, epsilon, n_iter):

        m = np.zeros(theta_0.shape)
        v = np.zeros(theta_0.shape)
        p = p_0

        for i in range(n_iter):
                g = get_gradient(theta)
                # Update biased first moment estimate
                m = beta_1 * m + (1-beta_1) * g
                # Update biased second moment estimate
                v = beta_2 * v + (1-beta_2) * g**2

                # Compute the bias-corrected estimates
                m_hat = m / (1 + beta_1**i)
                v_hat = v / (1 + beta_2**i)

                # Update p
                p += alpha * m_hat / (np.sqrt(v_hat) + epsilon)
```

Listing 3.1: *Pseudocode for the ADAM algorithm. See [18] for a more in depth discussion of the algorithm.*

4. Numerical Methods for Acoustic Inverse Design

The aim of this thesis was to use inverse design to find a phononic beamsplitter. In this section we describe how that was accomplished. First I specify the skeleton of the device, i.e. the parts that were fixed, as well as the objective function. We then go on to specify how the simulations and optimization was done.

4.1 Design

The skeleton of the device to be optimized can be seen in figure 4.1. The input and output waveguides consist of unit cells like the one in figure 2.1. The values for the parameters in the sketch are given in table 4.1. The reason for using this mode in this waveguide is that it has been shown to be interesting for avoiding mechanical leakage into the substrate on which it is clamped, as well as retaining a high optomechanical coupling [16].

The design placed in the design area was one of two types. The first was a *continuous design*, meaning that the material parameters ρ and C_{ijkl} varied continuously. The range of values that they can take are between the density and elasticity of pure silicon and that of air. realized, but was useful as a first step in the optimization. This was parametrized through the *design field*, p, which took values between 0, pure air, and 1, pure silicon. The second kind was a binary design, where each point either had silicon or not with no in-between values. This was accomplished using level-set methods, which will be explained in section 4.1.4. Because the device was completely symmetric, only one half of it needed to be modeled, and the other half extrapolated through a symmetry boundary condition.

4.1.1 Objective Function

The figure of merit of the device was the fraction of the input power transmitted into the output beams. Furthermore, the power in the output beams was to exit in the same mode as was excited at the input. Therefore, a mode overlap integral was used:

$$I = \int_{\Omega_1} \boldsymbol{u}(\boldsymbol{x}) \cdot \boldsymbol{u}_m^*(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{4.1}$$

where $\boldsymbol{u}_m$ is the shape of the mode (figure 2.3a). Because we were not interested in the phase of the output waves, the absolute value squared of the overlap integral was taken as the objective function,

$$f_{\text{obj}} = |I|^2 = II^*. \tag{4.2}$$

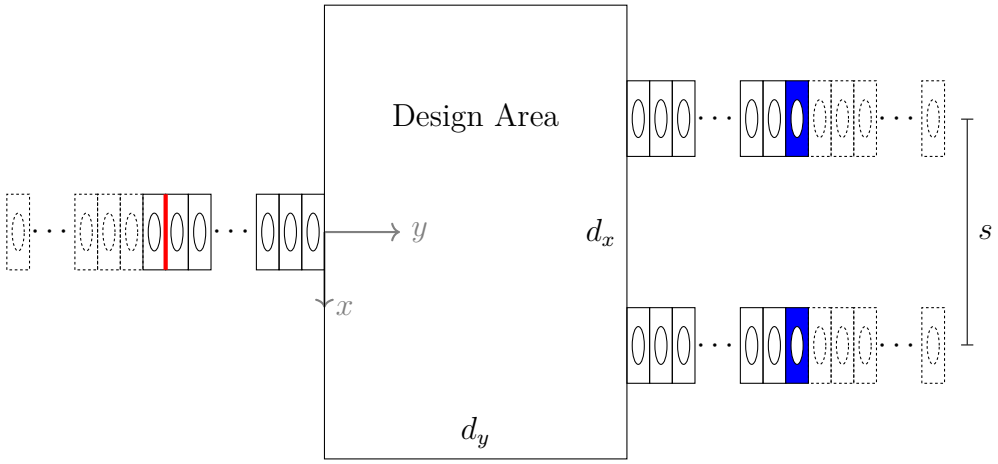


Figure 4.1: *Device design to be optimized. At the red line, a wave traveling right was excited. The output was measured over the blue unit cells. The dashed unit cells marks Perfectly Matched Layers (PMLs). The large, rectangular design area has dimensions $d_x \times d_y \times h$.*

Table 4.1: *Values for the geometric parameters of the device. Reference figures 2.1 and 4.1 for what the quantities mean.*

Parameter	value
a	187 nm
w	187 nm
h_x	153.5 nm
h_y	49.5 nm
h	220 nm
d_x	$6w$
d_y	$4w$
s	$3w$

This is maximal when the excitation of the mode m in the output waveguide is maximized, regardless of which phase it has. Note that because symmetry is enforced, the power through both outputs will be the same, and the outgoing waves will have the same phases.

The functional derivative of f_{obj} with respect to p then becomes

$$\frac{\delta f_{\text{obj}}}{\delta p}(x) = \frac{\delta I}{\delta p}(x) I^* + I \frac{\delta I^*}{\delta p}(x) \tag{4.3}$$

$$= \frac{\delta I}{\delta p}(x) I^* + \left(I^* \frac{\delta I}{\delta p}(x) \right)^* \tag{4.4}$$

$$= 2\text{Re}\left(\frac{\delta I}{\delta p}(x) I^* \right). \tag{4.5}$$

The derivative of I was derived in section 3.1.2.

4.1.2 Excitation

To excite the input waveguide in the desired mode, the stress on the boundary of a unit cell was exported from a unit cell eigenmode simulation with $k = 0.9\pi/a$ and applied to the boundary marked in red in figure 4.1. Since the frequency was perfectly controlled, this excites only the desired mode, since that is the only permitted mode close by as seen in the band diagram in figure 2.2.

In order to confirm that the excitation was indeed fully in the desired mode, a separate model with only a waveguide with 200 unit cells was created. After applying the excitation and running the simulation, the proportion of the excitation that ended up in the desired mode was calculated. This was done by first calculating the mode overlap integral $\int \boldsymbol{u}\boldsymbol{u}_m^* \, \mathrm{d}\boldsymbol{x}$ and comparing that to the norm of the displacement field $\int \boldsymbol{u}\boldsymbol{u}^* \, \mathrm{d}\boldsymbol{x}$. If $\boldsymbol{u}$ is written as $\boldsymbol{u} = a\boldsymbol{u}_m + b\boldsymbol{u}_r$ where a and b are scalars and $\boldsymbol{u}_r$ is orthogonal to $\boldsymbol{u}_m$, then $\int \boldsymbol{u}\boldsymbol{u}^* \, \mathrm{d}\boldsymbol{x} = a \int \boldsymbol{u}\boldsymbol{u}_m^* \, \mathrm{d}\boldsymbol{x} + b \int \boldsymbol{u}\boldsymbol{u}_r^* \, \mathrm{d}\boldsymbol{x}$. And since $\boldsymbol{u}_r$ is orthogonal to $\boldsymbol{u}_m$, $a = \int \boldsymbol{u}\boldsymbol{u}_m^* \, \mathrm{d}\boldsymbol{x} / \int \boldsymbol{u}_m \boldsymbol{u}_m^* \, \mathrm{d}\boldsymbol{x}$, which enables us to calculate b as well. The result was near perfect ($b < 0.03a$) excitation of only the desired mode. Since energy is proportional to the square of the amplitude, $b < 0.03a$ means that $> 99.9\%$ of the energy was in the correct mode. To obtain such high fidelity, it was important that the mesh used for the unit cells in the wave guide was the same as the mesh in the unit cell simulation. High fidelity was also achieved if both meshes were made very fine, but such fine meshes carries a prohibitively large computational cost.

4.1.3 Perfectly Matched Layers (PMLs)

Ideally, the input and outputs are infinite waveguides. Unfortunately, infinitely large models require infinite simulation time, which is problematic. Instead, PMLs were placed at the caps of the input and output waveguides. The purpose of the PML was to absorb any incoming waves without reflection, which would make it act as if there was an infinite waveguide on the other side into which the waves propagate

indefinitely. One way to accomplish this is to add an imaginary component to the density of the material. The imaginary part must be introduced smoothly, otherwise the abrupt change in material parameters would induce reflections anyway. Therefore, the imaginary part was taken to be exponentially increasing,

$$\rho_{\text{im}} = \rho_{\text{si}} \cdot s \cdot \frac{e^{-|y-y_0|/d} - e^{-n/d}}{1 - e^{-n/d}}. \tag{4.6}$$

In this equation, y_0 is the point where the waveguide ends, n is the length of the PML, d decides the steepness of the exponential and s the final value. Figure 4.2 shows the effect of changing these parameters on the shape of the profile of the imaginary component.

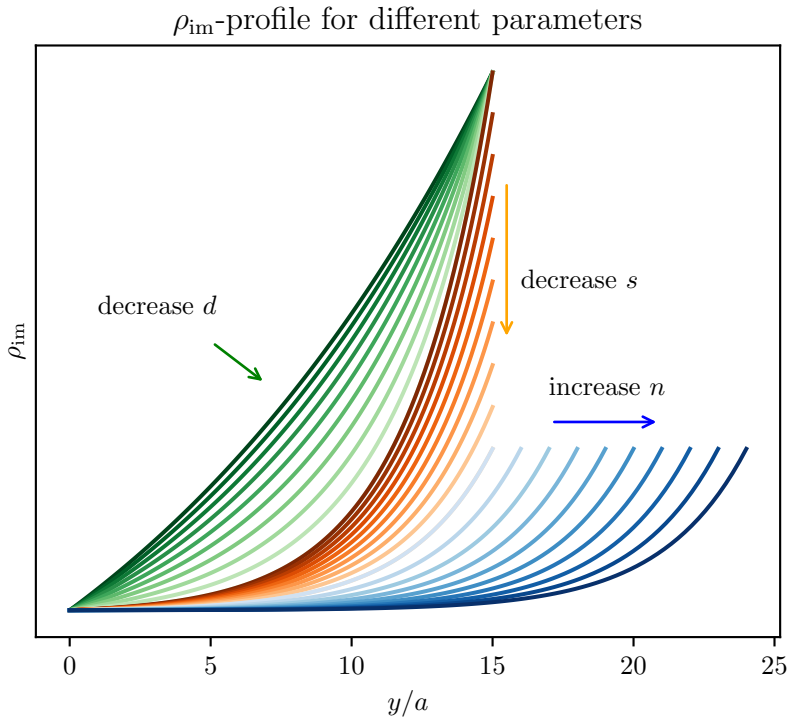


Figure 4.2: *This figure shows the effect of changing the different parameters in equation* (4.6)*. The green curves shows changing d while keeping the other parameters fixed, and the orange and blue show s and n respectively. Darker colour means higher value, and the last green curve coincides with the first orange, and the last orange with the first blue.*

There are three possible sources of reflections from the PML region. Firstly, if the transition from no imaginary component to some imaginary component is too abrupt, that causes reflections. Secondly, if the imaginary component is too small, the waves will not be completely dampened when they reach the end of the PML and thus reflect off of that. And lastly, if d is small then there can be reflections from the steep increase that happens some distance away from the beginning of the PML. See figure 2.2 for an illustration of where the different types of reflections occur.

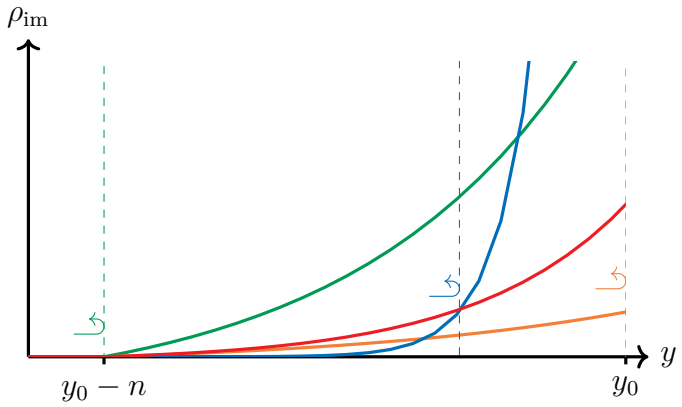


Figure 4.3: *For the green curve, the initial sudden increase of the imaginary component of the density at the beginning of the PML causes reflections. For the blue curve, the beginning of the PML is smooth but there is an increase partway through sharp enough to cause reflections. For the orange curve, the PML never becomes strong enough to completely dampen the waves, and they get reflected at the end. The red curve balances all these, yielding a perfect, reflectionless PML.*

It was desirable to make n as small as possible while still eliminating all reflections. This was because a smaller n meant a smaller model which resulted in shorter simulation times. In order to do so, a long waveguide with the same parameters as used for the input and output waveguides in the beamsplitter design was created. To discern where there was some component of the wave reflected, a fourier transform of the displacement field was made. The parameters controlling the shape of the ρ_{im} curve were then varied and an appropriate value was selected.

4.1.4 Level-Set

Ultimately, we wanted the device to consist of regions of material and regions of no material. Such a design is defined by the boundary between filled and empty regions. How to best implement a method of storing and evolving this boundary requires some thought. The first, and perhaps most intuitive implementation, would be to simply store the coordinates of evenly spaced points on the boundary. In addition to storing the coordinates, one must also store which points neighbour which. A different method, which is the one used in this report, is called the *level-set method*. With this method, the boundary is not directly stored, but is rather stored via an *implicit function*, $\phi(x)$, and the boundary is recovered as the 0-isocontour of ϕ, i.e. the points x where $\phi(x) = 0$.

There are two main advantages of using the level-set method rather than directly storing the boundary points. Firstly, when moving the boundary we would like to do so in the normal direction, as moving it along itself has no effect. Computing the normal direction of a directly stored boundary is slightly cumbersome, though certainly achievable. With level-set, moving the boundary in the normal direction is

as easy as adding a constant to the implicit function. Secondly, while the boundary is changing, the resolution in one part might need to be increased while the resolution in another needs to be decreased. Deciding where and when to add new points is non-trivial when directly storing the boundary. Furthermore, if two domains merge, or if one splits, points need to be removed and the connectivities changed, which is difficult. Figure 4.4 illustrates these problems with direct storage concretely. Both of these issues are automatically handled with the level-set method, as shown in figure 4.5.

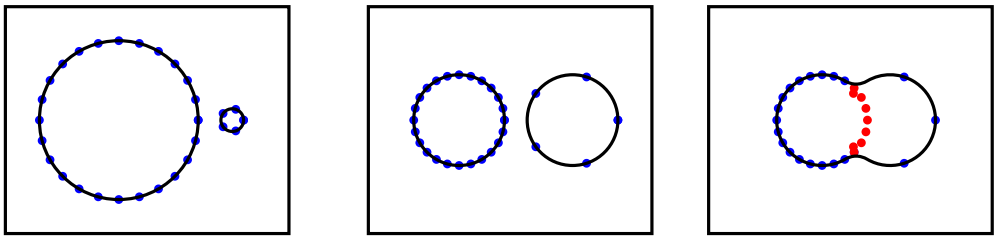


Figure 4.4: *Possible evolution of boundary. In the leftmost figure, the boundary is defined by pretty much evenly spaced points. In the center figure the boundaries have moved and the spacing is no longer even, and the right circle is very poorly resolved. The rightmost figure shows the boundary after the two circles moved closer together. Now there are multiple points that need to be removed, marked in red, and the connectivity of the points that remain must be changed such that the two boundaries are merged.*

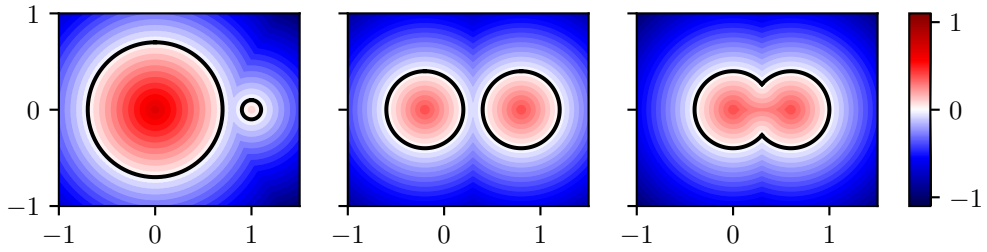


Figure 4.5: *Signed distance function for the boundaries in figure 4.4. The resolution of the boundary depends only on the resolution of the signed distance field, and not on how the boundary moves, so evolving boundaries will never become poorly resolved. Additionally, topological changes are also smoothly handled since there is no need to explicitly specify connectivities.*

There are of course a lot of possible functions $\phi(x)$ that have a given boundary as it's 0-isocontour. There is one choice that simplifies a lot of calculations though: the signed distance function. This function is defined as the distance from the closest point on the boundary, with a plus sign if it is inside and a minus sign if it is outside the boundary. See figure 4.5 for an example. It has the advantage that if one wishes to locally shift the boundary some length s in the normal direction, then simply add s to the function there. Figure 4.6 shows this effect in one dimension.

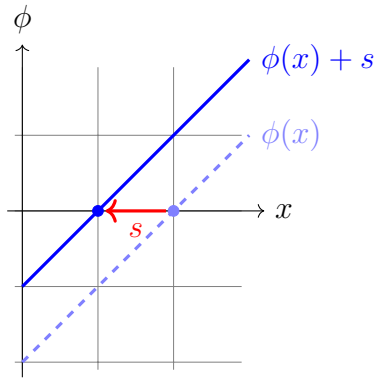


Figure 4.6: *Adding s to the signed distance function shifts boundary by s.*

Using a signed distance function means that a gradient descent step could be taken by simply adding the gradient field to the signed distance field. However, there are some pitfalls that had to be avoided. Firstly, the gradient needed to be rescaled so that the boundary moved an appropriate distance. This was done such that the boundary moves at most 1 nm. Secondly, since the gradient was occasionally sharply peaked somewhere far from the boundary, only the gradient near the boundary is actually added to the signed distance field. After performing this addition, what was previously a signed distance field would now no longer be that, and thus the signed distance field was recalculated from the new, shifted boundary. This recalculation came with a performance penalty, but since the COMSOL simulations were orders of magnitude slower than all other parts of the optimization, this was of little concern.

4.2 Optimization

For the optimization, we used the ADAM algorithm with one modification: a global learning rate was employed rather than separate learning rates for each dimension. The reasoning for this was that separate learning rates would give jagged contours, which might not be properly resolved by the meshing. Practically this modification means that `v` is a scalar and is set to `(1-beta_2)*np.mean(g**2) + beta_2 * v` in listing 3.1. The optimization was run and continually monitored, and once convergence was visually confirmed through looking at the plot of f_{obj} by iteration, it was terminated. A few different values for the β:s were tried, and in the end $\beta_1 = 0.9$ and $\beta_2 = 0.95$ were chosen. However, it seemed that the evolution were not too sensitive to this choice. α was taken to be $2 \cdot 10^{-3}$, because that is large enough that p could change from 0 to 1 in 500 iterations, which was deemed an appropriate timescale as the time per iteration was about 4 minutes. A too large α would lead to poor convergence, while a too low value would result in slow progress.

The final step of the optimization was to use level-set for the design. However, just using the $p = 0.5$ boundary of the final optimized continuous design as the initial point of the level-set optimization would be an abrupt change, and there is no reason to expect that the resulting level-set design would be close to a design with good performance. Therefore, once the continuous optimization had converged, a sigmoid

function was applied to the design field p before ρ and C_{ijkl} were set:

$$\rho(\boldsymbol{x}) = \rho_{\text{si}}\sigma_r(p(\boldsymbol{x})), \quad C_{ijkl}(\boldsymbol{x}) = C^{\text{si}}_{ijkl}\sigma_r(p(\boldsymbol{x})), \quad \sigma_r(p) = \frac{1}{1+e^{-(p-0.5)/r}}. \tag{4.7}$$

and then the optimization was restarted. The sigmoid results in fewer values close to 0.5, effectively making the device closer to being binary, which means that the step to level-set designs wasn't as abrupt. Note that the parameter r controls how much values are shifted, and with $r \to 0$, σ_r becomes a step function. Once that had been repeated a couple of times with decreasing r, the level-set design was initialized using the $p = 0.5$ boundary of the final optimized design, and the level-set optimization was run.

4.3 Simulations

In this section we detail some of the practicalities of performing the simulations. For finite element simulations we used COMSOL version 6.0. First, a unit cell with periodic boundary conditions was simulated, from which we obtained:

- the mode shape, used to calculate the component of the displacement field in the desired mode,
- the stress at the boundary, used as the force exciting the input waveguide,
- the frequency at which to excite in order to obtain a traveling wave with the desired wave vector.
- a mesh to be used when meshing the unit cells in the waveguides.

For the continuous optimization, the basic procedure went

1. Through the COMSOL-Matlab API, a beamsplitter model was made. The excitation force profile as well as the unit cell mesh and the mode shape were imported from the unit cell simulation.
2. A semi-random initial design field p was created. This was done by drawing a sample from a Gaussian process, which means that the characteristic length scale that the design varies on could be controlled.
3. If the sigmoid function was to be used in this optimization, it was applied to the design field, and the result was saved to a different variable, that we call the interpolation field.
4. The interpolation field was imported to the COMSOL model, and the material parameters adjusted proportional to said field.
5. Both forward and backward simulations were run and the gradient was calcu-

lated and exported to Matlab.

6. With the gradient, the design field was updated and the algorithm returned to step 3.

For the level-set optimization, the procedure was basically the same, with some minor differences:

1. Through the COMSOL-Matlab API, a beamsplitter model was made. The excitation force profile as well as the unit cell mesh and the mode shape was imported from the unit cell simulation.

2. An initial signed distance field s was created. This was done by finding the $p = 0.5$ isocontour from the final iteration of the continuous optimization, and initializing a signed distance field from that.

3. The zero isocontour of the signed distance field was imported into the COMSOL model and a geometry was built from that.

4. Both forward and backward simulations were run and the gradient was calculated and exported to Matlab.

5. With the gradient, the signed distance field was updated and the algorithm returns to step 3. Note that as part of the update, the signed distance field was reinitialized so that it would not lose its properties.

5. Results and Discussion

In this chapter I present and discuss the results of my work. The first section, section 5.1, presents the simulations needed to ascertain that the beamsplitter simulations would work as intended. This includes tuning the PML parameters and checking that the excitation excites the right mode. After that, I present the results of the continuous optimization followed by the results for the level-set optimization in sections 5.2 and 5.3 respectively. In summary, the continuous optimization yielded near perfect devices with full transmission and minimal reflection. The level-set optimizations never reached quite as good performance, reaching a 46/46 splitter with around 8 % of the power getting reflected.

5.1 Long Waveguide Simulations

As detailed in chapter 4, a long waveguide without any beamsplitter elements was simulated to ascertain that the excitation method correctly excited only the desired mode, and that the PML elements were reflectionless. In this section I present the results from those simulations. The waveguide used for these is pictured in figure 5.1.

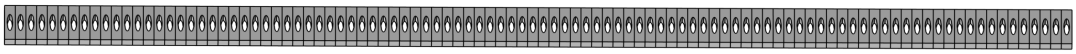


Figure 5.1: *This figure shows the long waveguide used for the PML simulations as wells as validating the excitation method.*

5.1.1 Excitation

The fact that the excitation was almost solely in the desired mode was verified in two ways. Firstly, as detailed in section 4.1.2, a and b in $u = au_m + bu_r$ was computed. The result was that $a = 1.166$ and $b = 0.0327$. Thus 99.92 % of the energy was in the desired mode. Secondly, a fourier transform of the y-component of the displacement field was made. This can be seen in figure 5.2. The other modes that could be excited at this frequency have very different wave vectors, which means that they would show up as separate peaks somewhere around $0.4\pi/a$ and $0.1\pi/a$. Since no other peaks can be seen, this further supports the result that only the desired mode is excited.

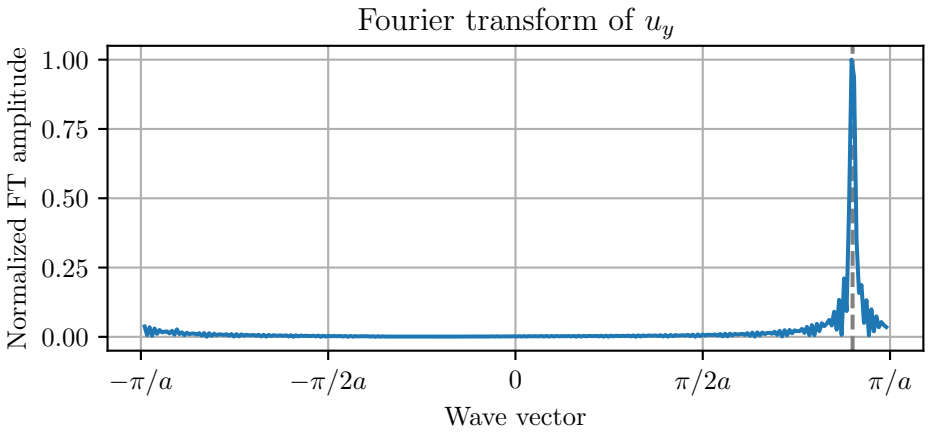


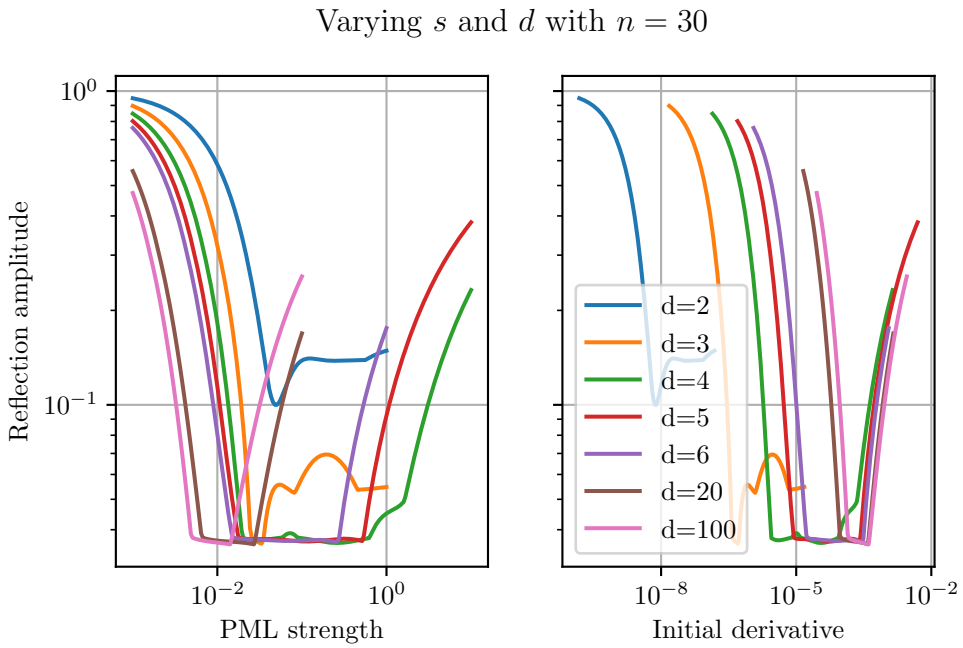
Figure 5.2: *Fourier transform of the y-component of the displacement field. It clearly shows one wave traveling forward with a k-vector of* $0.9\pi/a$*, where the dashed gray line is. Since the closest other mode at this frequency is at* $k_y \approx 0.4\pi/a$*, where no peak is visible, it is concluded that solely the desired mode is excited.*

5.1.2 PML Investigation

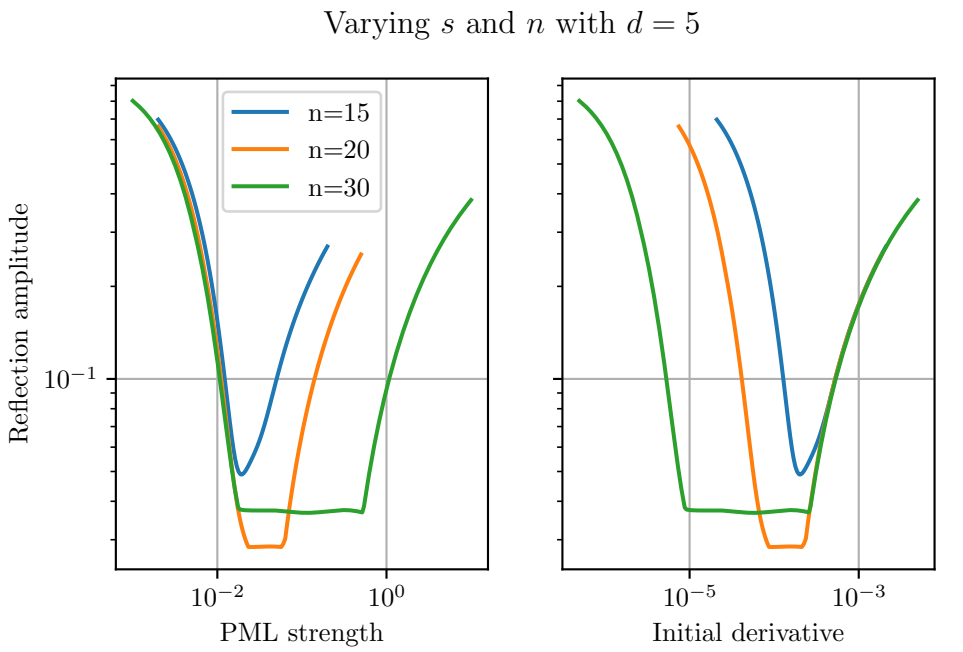
Figure 5.3 shows the amplitude of the reflection, quantified as the peak height relative to the forward propagating wave, for different profiles. These figures fit well with the three types of reflection mentioned previously. The leftmost plot in figure 5.3a shows that all of the different shapes yield reflections when the derivative at the beginning of the PML is around $3 \cdot 10^{-4}$, which indicates that the reflections are dominated by effects from the sudden start of the PML. On the other end, the increase in reflection amplitude come from waves reaching the end of the PML and getting reflected there. The third type of reflection seems to only be noticeable for $d \leq 4$. It is also worth noticing that with lower d, the plateau where neither of the first two kinds of reflections are significant is broader. Since $d = 5$ was the lowest d to show no signs of the third type of reflection, that was chosen for the shape of the PML profile. After that, I investigated how short I could make the it while retaining low reflections. Figure 5.3b shows s sweeps for different n. To achieve a short yet functional PML, $d = 5$, $s = 0.03$ and $n = 20$ was chosen. For $n = 15$, reflections from the beginning start to become significant before the reflections from the end have waned.

5.2 Continuous Optimization

There was a lot of problems with the optimization of the continuous design. Many optimization runs would end up looking like figure 5.4, reaching a somewhat performant beamsplitter but invariably declining after some point. The figure shows the evolution of the transmitted power in one of the output arms, as well as the reflected power back into the input. This has been normalized such that 1 is the



(a) *On the left is the reflection amplitude plotted as a function of the PML strength s for different d. For $d > 4$ there are two sources of reflection: for small s, reflection at the end of the PML occurs, and for large s, there is reflection at the beginning. The right figure makes it clear that it is the slope at the beginning of the PML that matters, since the point at which it becomes significant is the same for all d.*



(b) *This is the same as figure 5.3a but with varying n. For $n = 15$, the reflections from the beginning do not subside before the reflections from the end become significant, so at least $n = 20$ is necessary.*

Figure 5.3

power in through the input waveguide. Thus, an optimal beamsplitter would reach a transmission of 0.5. If the powers do not add up, it is because energy is flowing out in a different mode.

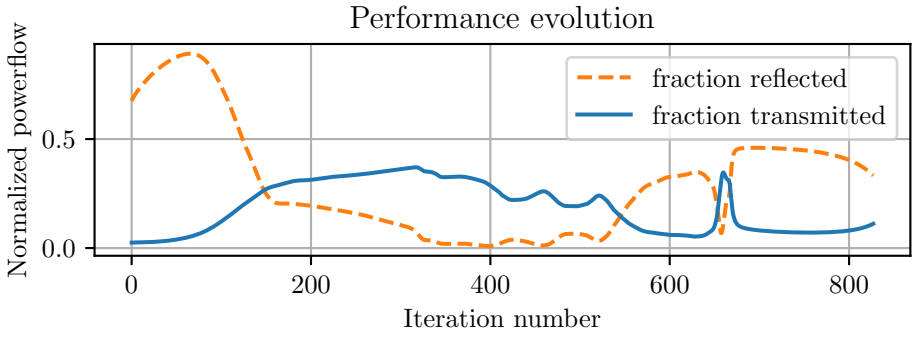


Figure 5.4: *Non-converging optimization example. The blue curve shows the transmitted power in one of the output arms as a function of iteration, while the yellow shows the power reflected back out through the input. The cases where two times the blue plus the yellow isn't 1.0 is explained by power exiting through a different mode.*

I tested a few different theories on why the algorithm might fail to converge. The first one was that perhaps the simulations were unstable for the case when $p \approx 0$. To test this I interpolated between ρ^{si} and $1000\,\text{kg/m}^3$ instead. However, convergence was still not reached. The second was that maybe the meshing was too coarse to resolve the design field properly. However, this was not the case as increasing the meshing did not change the objective function more than a percent or so. The last thing I thought of was that comsol was using quadratic shape functions for the fields. This means that on each finite element of the mesh, the fields are approximated with quadratic polynomials. Thus, if one computes any third order spatial derivative, it will be 0 everywhere, and the second order derivatives will not be very accurate. In figure 5.5 I show how quadratic splines can give a very good approximation to a function, while still being a terrible approximation to its second derivative. Since I used second derivates in computing my gradient, that also wasn't very reliable. One way of solving this is to use higher order shape functions, however this greatly increases the degrees of freedom, making the problem too computationally expensive, at least with our hardware and geometry. Another way is to make the mesh much finer, but this is also too computationally expensive.

However, there was another way: using the built in COMSOL function `fsens` for computing the gradient, convergence was obtained. Figure 5.6 shows the performance evolution of this optimization run. At iteration 467, the model was deemed to have converged and as described in section 4.2, a sigmoid function with $r = 0.1$ was applied and optimization continued. At iteration 615, the model again seemed converged and r was set to 0.05. Finally, at iteration 830, near perfect performance was reached and optimization was terminated. Figure 5.7 shows the design field after convergence for each of the three stages.

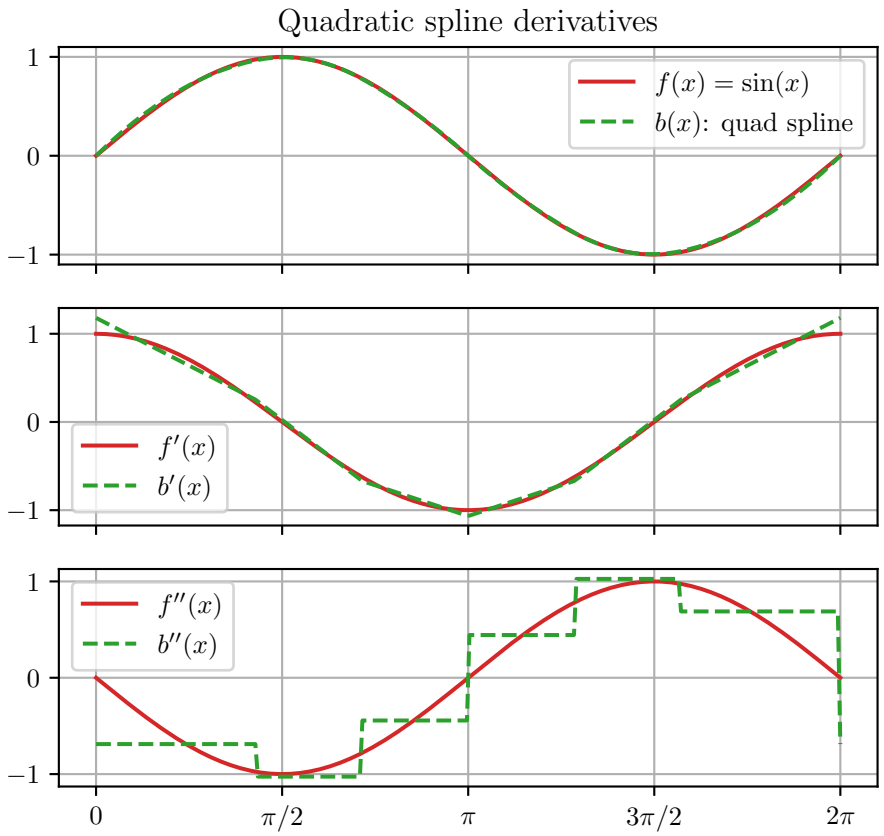


Figure 5.5: *This figure shows a quadratic spline approximation to a sine function, given eight points on the function. Even though the spline approximates the sine function very well, the second derivative approximation is terrible.*

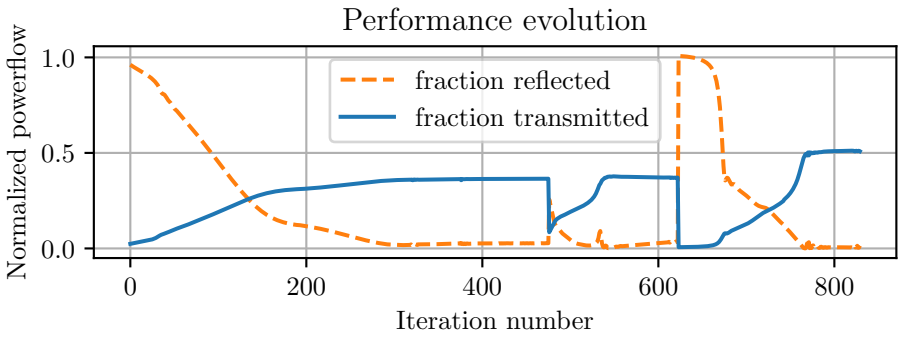


Figure 5.6: *Similar to figure 5.4, but at iteration 467 and 615, a sigmoid function was abruptly applied to the design field, which causes the dips. The final device was a near perfect beamsplitter, with less than 1 % of the power begin reflected, and nothing being scattered into other modes.*

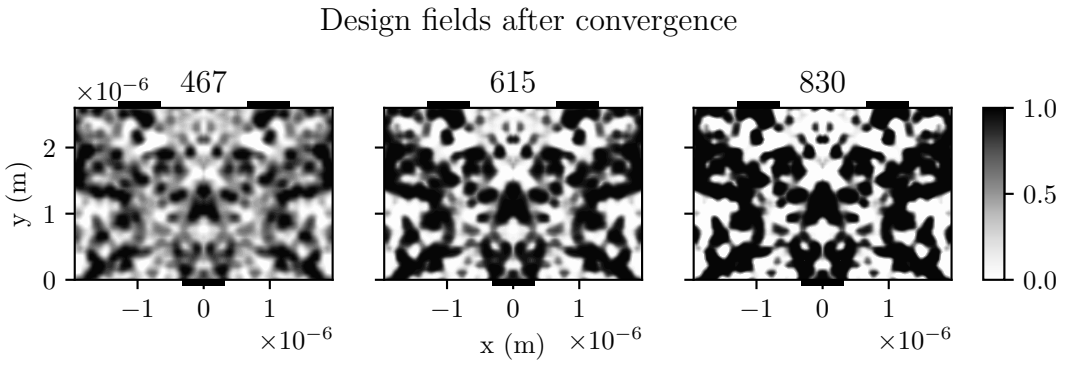


Figure 5.7: *This figure shows the interpolation field at iteration 467, 615, and 830. It is clearly seen that the device becomes closer and closer to being binary.*

5.3 Level-Set Optimization

The initial contour of the device was taken as the 0.5-isocontour of the final design field from the continuous optimization. Figure 5.8 shows the convergence plot for the level-set optimization, and figure 5.9 shows the final device. The performance did reach quite high values: splitting the input power roughly 46/46, though it never achieved the perfect 50/50 that was seen after the continuous optimization. Something to note is that the device does seem to be rather sensitive; a shift of only a nanometre can greatly change the device performance. Therefore the α chosen for the algorithm had to be set as low as 0.5 nm. Another important note is that the final design is rather close to the initial, so the starting point is very important for how high performance can be reached.

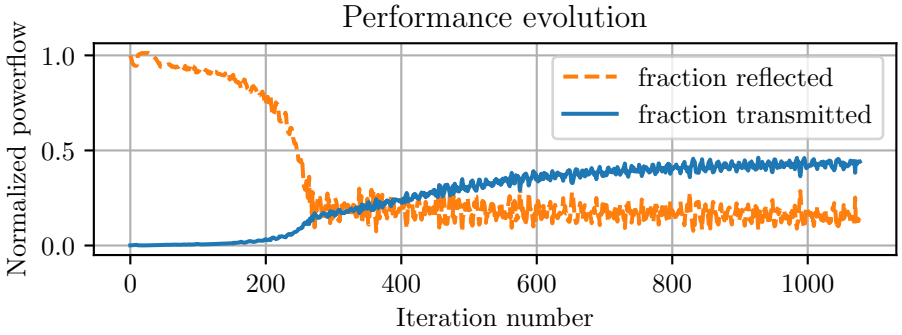


Figure 5.8: *Performance evolution of the level-set device during optimization. The curve is not as smooth as the curve from the continuous optimization, which indicates that the step size is closer to being too large.*

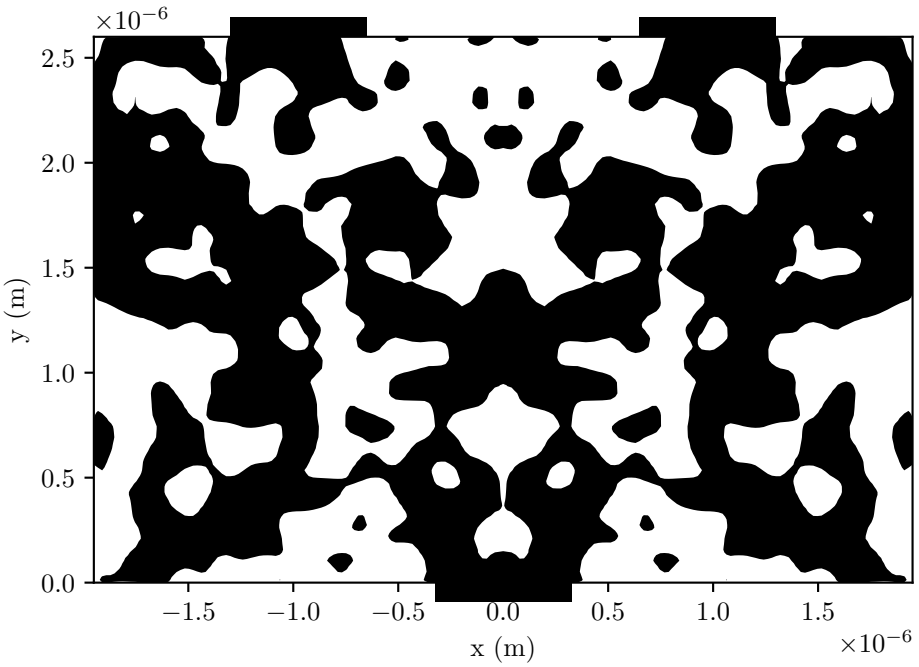


Figure 5.9: *This figure shows the final optimized design for the level-set optimization. The device performs a roughly 45/45 split of the incoming power, with the rest of the power being reflected.*

6. Concluding Remarks

Phononics have the potential to become a vital part of both quantum and classical information processing architecture, but devices are still limited to those that can be investigated through analytical means and/or be optimized with brute force methods. This study set out to investigate whether inverse design with adjoint simulation could be used to design so far unrealized devices and enable little explored applications.

I first examined theoretically the applicability of adjoint simulation on phononics, from which I concluded that the methods should work in theory. The second aim of this thesis was to demonstrate the utility of this method by designing a phononic beamsplitter. To do that, the process was split into two stages, a first where the material was allowed to vary continuously, and a second where a binary design was enforced using level-set methods. The continuous optimization yielded near perfect performance, validating the theoretical derivations. The level-set optimization never did achieve perfect performance, but good performance was still reached. These results indicate that the inverse design concept can be very useful for designing phononic devices going forward. We remain hopeful that the problems with the latter stage can be solved though, which would be a major step forward.

6.1 Future Research

There are a great number of potential paths that can be explored from this point, and the positive results presented here warrant further efforts in this area. One important improvement that should be investigated is the sensitivity of the device to small changes in the design. If it is very sensitive, imperfections in fabrication could be detrimental to the devices performance. There may be some ways of mitigating this however, for example by running multiple simulations with small perturbations in the design and averaging them to obtain the objective function. Another potential path of exploration is the limiting of the feature size. This could be done by augmenting the objective function, adding a pure part that punishes small features.

In addition to method improvements, another path is to apply this to designing other types of devices. For example, waveguide bends seem to be well suited for this type of design, and would be useful if one wishes to use phononic waveguides for routing excitations around on a chip. It may also be possible to inverse design hybrid devices that use both optics and mechanics for example, though that would likely require a significant effort.

References

[1] P. Arrangoiz-Arriola, E. A. Wollack, Z. Wang, *et al.*, "Resolving the energy levels of a nanomechanical oscillator," *Nature*, vol. 571, no. 7766, pp. 537–540, Jul. 2019, Number: 7766 Publisher: Nature Publishing Group. DOI: 10.1038/s41586-019-1386-x. [Online]. Available: https://www.nature.com/articles/s41586-019-1386-x (visited on 2023-04-14).

[2] A. H. Safavi-Naeini, D. V. Thourhout, R. Baets, and R. V. Laer, "Controlling phonons and photons at the wavelength scale: Integrated photonics meets integrated phononics," *Optica*, vol. 6, no. 2, pp. 213–232, Feb. 20, 2019, Publisher: Optica Publishing Group. DOI: 10.1364/OPTICA.6.000213. [Online]. Available: https://opg.optica.org/optica/abstract.cfm?uri=optica-6-2-213 (visited on 2023-04-19).

[3] E. A. Sete and H. Eleuch, "High-efficiency quantum state transfer and quantum memory using a mechanical oscillator," *Physical Review A*, vol. 91, no. 3, p. 032 309, Mar. 17, 2015, Publisher: American Physical Society. DOI: 10.1103/PhysRevA.91.032309. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.91.032309 (visited on 2023-06-02).

[4] G. S. MacCabe, H. Ren, J. Luo, *et al.*, "Phononic bandgap nano-acoustic cavity with ultralong phonon lifetime," *Science*, vol. 370, no. 6518, pp. 840–843, Nov. 13, 2020. DOI: 10.1126/science.abc7312. arXiv: 1901.04129[cond-mat,physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/1901.04129 (visited on 2023-06-06).

[5] H. Qiao, E. Dumur, G. Andersson, *et al.*, *Developing a platform for linear mechanical quantum computing*, 2023. arXiv: 2302.07791 [quant-ph].

[6] P.-I. Schneider, X. Garcia Santiago, V. Soltwisch, M. Hammerschmidt, S. Burger, and C. Rockstuhl, "Benchmarking five global optimization approaches for nano-optical shape optimization and parameter reconstruction," *ACS Photonics*, vol. 6, no. 11, pp. 2726–2733, 2019.

[7] Y. Zhang, S. Yang, A. E.-J. Lim, *et al.*, "A compact and low loss y-junction for submicron silicon waveguide," *Optics Express*, vol. 21, no. 1, pp. 1310–1316, Jan. 14, 2013, Publisher: Optica Publishing Group. DOI: 10.1364/OE.21.001310. [Online]. Available: https://opg.optica.org/oe/abstract.cfm?uri=oe-21-1-1310 (visited on 2023-04-19).

[8] S. Chen, J. Montgomery, and A. Bolufé-Röhler, "Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution," *Applied Intelligence*, vol. 42, no. 3, pp. 514–526, Apr. 1, 2015. DOI: 10.1007/s10489-014-0613-2. [Online]. Available: https://doi.org/10.1007/s10489-014-0613-2 (visited on 2023-06-17).

[9] S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez, "Inverse design in nanophotonics," *Nature Photonics*, vol. 12, no. 11, pp. 659–670, Nov. 2018, Number: 11 Publisher: Nature Publishing Group. DOI: 10.1038/s41566-018-0246-9. [Online]. Available: https://www.nature.com/articles/s41566-018-0246-9 (visited on 2023-04-14).

[10] L. Su, D. Vercruysse, J. Skarda, N. V. Sapra, J. A. Petykiewicz, and J. Vuckovic, "Nanophotonic inverse design with spins: Software architecture and practical considerations," 2019. arXiv: 1910.04829 [physics.app-ph].

[11] J. S. Jensen and O. Sigmund, "Systematic design of photonic crystal structures using topology optimization: Low-loss waveguide bends," *Applied Physics Letters*, vol. 84, no. 12, pp. 2022–2024, Mar. 16, 2004. DOI: 10.1063/1.1688450. [Online]. Available: https://doi.org/10.1063/1.1688450 (visited on 2023-05-28).

[12] A. Y. Piggott, J. Lu, T. M. Babinec, K. G. Lagoudakis, J. Petykiewicz, and J. Vučković, "Inverse design and implementation of a wavelength demultiplexing grating coupler," *Scientific Reports*, vol. 4, no. 1, p. 7210, Nov. 27, 2014. DOI: 10.1038/srep07210. arXiv: 1406.6185[physics]. [Online]. Available: http://arxiv.org/abs/1406.6185 (visited on 2023-05-28).

[13] O. Sigmund and K. Maute, "Topology optimization approaches," *Structural and Multidisciplinary Optimization*, vol. 48, no. 6, pp. 1031–1055, Dec. 1, 2013. DOI: 10.1007/s00158-013-0978-6. [Online]. Available: https://doi.org/10.1007/s00158-013-0978-6 (visited on 2023-06-16).

[14] G. Yi and B. D. Youn, "A comprehensive survey on topology optimization of phononic crystals," *Structural and Multidisciplinary Optimization*, vol. 54, no. 5, pp. 1315–1344, Nov. 1, 2016. DOI: 10.1007/s00158-016-1520-4. [Online]. Available: https://doi.org/10.1007/s00158-016-1520-4 (visited on 2023-06-17).

[15] J. Joannopoulos, S. Johnson, J. Winn, and R. Meade, *Photonic Crystals: Molding the Flow of Light 2nd edn Princeton Univ.* Princeton University Press, 2008.

[16] J. Kolvik, P. Burger, J. Frey, and R. Van Laer, *Clamped and sideband-resolved silicon optomechanical crystals*, Mar. 31, 2023. DOI: 10.48550/arXiv.2303.18091. arXiv: 2303.18091[physics,physics:quant-ph]. [Online]. Available: http://arxiv.org/abs/2303.18091 (visited on 2023-04-14).

[17] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, Turbulence and Combustion*, vol. 65, no. 3, pp. 393–415, Dec. 1, 2000. DOI: 10.1023/A:1011430410075. [Online]. Available: https://doi.org/10.1023/A:1011430410075 (visited on 2023-05-27).

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. arXiv: 1412.6980 [cs.LG].

DEPARTMENT OF MICROTECHNOLOGY AND NANOSCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY