Inverse design in Quantum Acoustics

Designing a Phononic Beamsplitter using Inverse Design with Adjoint Simulation

David Hambraeus

[DRAFT]

compiled 2023-05-10, 17:08

Inverse design in Quantum Acoustics:
Designing a Phononic Beamsplitter using Inverse Design with Adjoint Simulation

David Hambraeus
Department of Microtechnology and Nanoscience
Chalmers University of Technology

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Keywords: lorem, ipsum, dolor, sit, amet, consectetur, adipisicing, elit, sed, do.

Acknowledgements

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

David Hambraeus, Gothenburg, May 2023

Table of contents

List of figures

List of tables

Acronyms

ADAM Adaptive Moment Estimation. v, 13, 14

GD Gradient Descent. v, 14

PML Perfectly Matched Layer. v, 16, 18

List of Todos

Todonotes are organized as follows:

General comment / question

Things that could be done now, no further simulations/consultation needed

Things that could be done now but I am not sure if I should, or how to do it

Things that can't be done yet because they depend on other things, e.g. results

Citation needed

1. Introduction

After thesis is done, check that I've used cref and not ref

References before the dot or after? And space or no space?

Introduction to quantum acoustics...I need to read more literature I think

Restructure a little... I would like to talk about inverse design first and quantum acoustics second. Talk about how inverse design is a concept that has been applied to nanophotonics but not to quantum acoustics yet. Then talk about why we care about quantum acoustics. It feels a little bit forced to do it in that order though, talking about quantum acoustics first might be a better idea, since that would naturally lead one to introduce the problem of design.

Conventionally, when designing these components, the designer comes up with a design through intuition and parametrizes it with a couple of parameters. For example they may believe that a structure with periodically placed circular holes should yield a device that performs the desired function. The parameters that are unknown might then be the distance between neighbouring holes and the radius of the holes. To find the optimal device they would then systematically test parameter values to see which give the best performance in a simulation of the device. This brute force method of design limits the possible number of parameters to a very small number. If there are 10 different values to test for each parameter, even six parameters would require 1,000,000 simulations. One can of course use smarter optimization algorithms like bayesian optimization or particle swarm optimization[1] to decrease the number of simulations needed, but it will still be of the same order.

cite something, check Ida's thesis maybe

A different approach that has been gaining some popularity is *inverse design with adjoint simulation.*[2] The idea is that if the gradient of the figure of merit with respect to the parameters can be calculated, then we can use gradient based optimization methods, which converge much faster, even if the number of parameters is very large. With these methods, one hopes to be able to search among a much more general class of designs for the optimal one. Su, Vercruysse, Skarda, Sapra, Petykiewicz, and Vuckovic has developed software that successfully uses inverse design for nanophotonic devices [3].

With this thesis, we explore the possibility of extending this paradigm to acoustic devices. In order to do so, we attempt to design a phononic beam splitter.

Say something about the fact that I derived the acoustic case myself

1.1 Thesis outline

2. Acoustic waves and waveguides

In order to efficiently model the deformation and stresses in a solid material, a linear elasticity model is often assumed. For small deformations, solid materials obey Hooke's law which in it's full form looks like

$$\sigma = C : \epsilon$$

where σ is the stress tensor, C the elasticity tensor which is a four-tensor that is a property of the material, $\epsilon \coloneqq \nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^\intercal$ is the strain tensor, and : denotes double scalar product. This equation is linear in $\boldsymbol{u}$, hence the name *linear* elasticity. Using this and newtons equations of motion, the equation governing the dynamics is obtained:

$$\rho \ddot{\boldsymbol{u}} = \nabla \cdot \sigma + \boldsymbol{F}.$$

where ρ is the density, $\boldsymbol{u}$ is the displacement and $\boldsymbol{F}$ is the externally applied force. Assuming a time harmonic solution $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(x)e^{i\omega t}$ with angular frequency ω this becomes

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

To combine these into one equation that can be solved for $\boldsymbol{u}$ we first rewrite them in index notation to make calculations clearer:

$$\begin{aligned}
\epsilon_{ij} &= \frac{1}{2}(\partial_i u_j + \partial_j u_i) && (2.1)\\
\sigma_{ij} &= C_{ijkl}\epsilon_{kl} && (2.2)\\
&= \frac{1}{2}\left(C_{ijkl}\partial_k u_l + C_{ijkl}\partial_l u_k\right) && (2.3)\\
&= C_{ijkl}\partial_k u_l \text{ because of the symmetry of } C && (2.4)
\end{aligned}$$

which gives

$$\begin{aligned}
F_i &= -\rho\omega^2 u_i - \partial_j \sigma_{ij} && (2.5)\\
&= -\rho\omega^2 \delta_{ik} u_k - \partial_j \left(C_{ijkl}\partial_l u_k\right) && (2.6)\\
&= -\left(\rho\omega^2 \delta_{ik} \cdot + \partial_j \left(C_{ijkl}\partial_l \cdot\right)\right) u_k && (2.7)
\end{aligned}$$

where the indices i, j, k, l go over the spacial dimensions x, y, z. All of the tensors in the equation above are really tensor fields, i.e. they are functions of $\boldsymbol{x}$. Defining the operator $\hat{A}_{ik}$ as $-\left(\rho\omega^2 \delta_{ik} \cdot + \partial_j \left(C_{ijkl}\partial_l \cdot\right)\right)$ we can write

$$\hat{A}_{ik} u_k = F_i \tag{2.8}$$

2.1 Bloch States and Band Diagrams

With no external forces, i.e. $F_i = 0$, (2.8) can be written as

$$\frac{1}{\rho}\partial_j \left(C_{ijkl}\partial_l u_k \right) = \omega^2 u_i \tag{2.9}$$

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 with some periodicity $\boldsymbol{a}$, meaning that $C_{ijkl}(\boldsymbol{x}) = C_{ijkl}(\boldsymbol{x} + n\boldsymbol{a})$ 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 space of simultaneous eigenstates to both of them. The eigenfunctions of the translation operator are $\exp(i\boldsymbol{k} \cdot \boldsymbol{x})$ with eigenvalues $\exp(ika)$, where $a = |a|$. Defining the reciprocal lattice constant $\boldsymbol{b} = 2\pi\boldsymbol{a}/a^2$, we see that the functions $\exp(i(\boldsymbol{k} + m\boldsymbol{b}) \cdot \boldsymbol{x})$ for integer values of m all have the same eigenvalue, which means that they form a degenerate subspace of eigenfunctions. Thus we can write the eigenfunctions of both operators as

$$\boldsymbol{u}_{\boldsymbol{k}}(\boldsymbol{x}) = \sum_m e^{i(\boldsymbol{k}+m\boldsymbol{b})\cdot\boldsymbol{x}} \boldsymbol{u}_{m,\boldsymbol{k}}(\boldsymbol{x}) = e^{i\boldsymbol{k}\cdot\boldsymbol{x}} \bar{\boldsymbol{u}}_{\boldsymbol{k}}(\boldsymbol{x}) \tag{2.10}$$

where $\bar{\boldsymbol{u}}_{\boldsymbol{k}}$ is a periodic function with periodicity $\boldsymbol{a}$.

Sakurai does basically the same derivation but for electronic states in a crystal lattice, but he also uses operator formalism. Can I cite that? Other option is citing Chan's PhD who do it the same way for photonic crystals and says that it can be done analogously for phononic crystals. I don't think Auld does anything with periodic crystals/bloch states?

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.

2.1.1 Concrete example of phononic crystal

In this work, a rectangular waveguide patterned with ellipses is used. The structure is clamped at the bottom (meaning that we use a fixed boundary condition there) while the other sides are free. A top down schematic of the unit cell can be seen in figure 2.1.

Enforcing a wave vector $\boldsymbol{k} = k\hat{\boldsymbol{y}}$ and running COMSOL to find the eigenmodes with their corresponding frequencies for this structure yields the band diagram in figure 2.2. Figure 2.3 shows the shapes of the eight lowest lying eigenmodes at $k = 0.9\pi/a$.

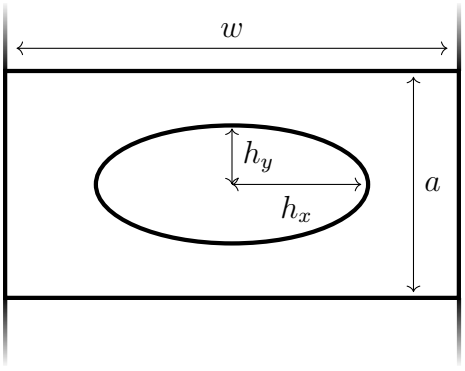

Figure 2.1: *Top down view of unit cell of a phononic crystal.*

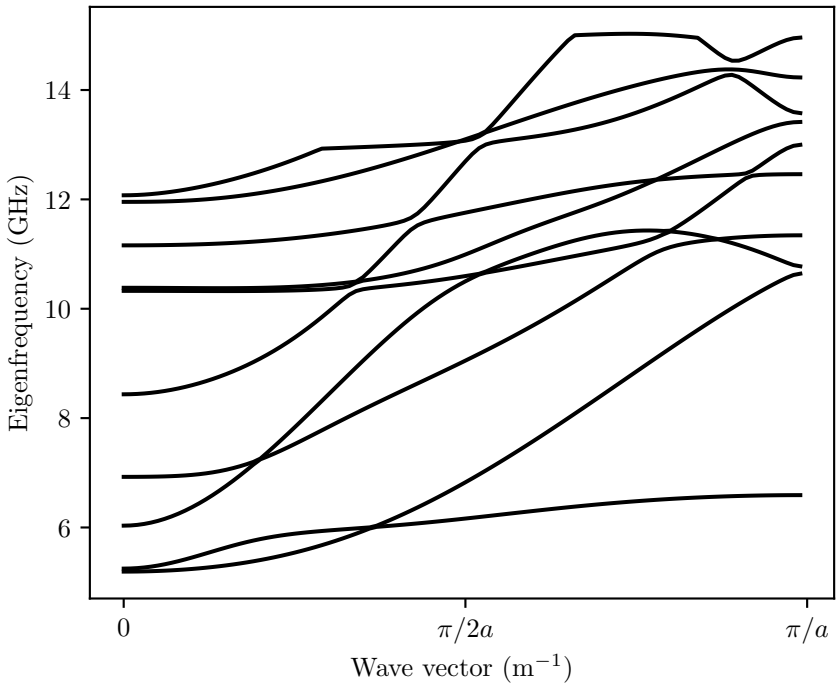

Figure 2.2: *Band diagram of phononic crystal defined in figure 2.1.*

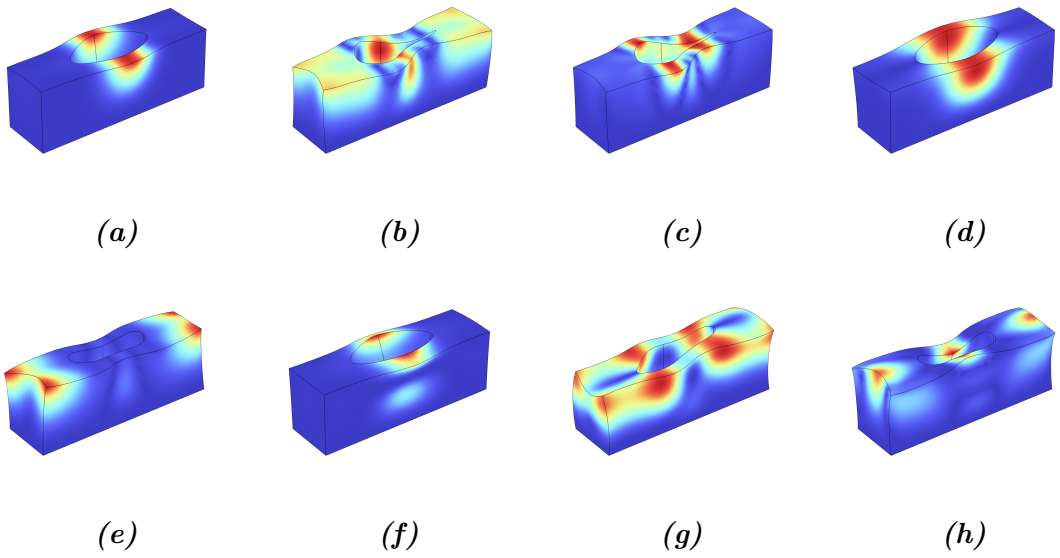

Figure 2.3: *Mode shapes for the lowest eight modes at $k = 0.9\pi/a$. The color denotes the absolute value of the displacement.*

At some point write about phonons? I haven't really had to care about the fact that excitations are discrete so if I talk about it it'd just for applications...

I originally thought to have something about PMLs here, but now it's been put in the method... It is more of a simulations topic, not really a physics one.

3. Inverse Design

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[1], which I 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 is a very 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. I will in this section first give a general derivation, followed by the case of inverse design in acoustics.

3.1.1 General Derivation

Let f_{obj} be a function which depends on some (large) 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. The overall goal is to find the parameters p that maximize the objective function f_{obj}. The goal of adjoint simulation is to find $\frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}p}$. This can be expanded through the chain rule as

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

[1] Also called *figure of merit (FoM)* by some.

To find the latter factor we do

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}p}[Av = b] &\implies \frac{\mathrm{d}A}{\mathrm{d}p}v + A\frac{\mathrm{d}v}{\mathrm{d}p} = 0 \\ &\implies \frac{\mathrm{d}v}{\mathrm{d}p} = -A^{-1}\frac{\mathrm{d}A}{\mathrm{d}p}v\end{aligned}$$

which gives

$$\frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}p} = -\frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}v}A^{-1}\frac{\mathrm{d}A}{\mathrm{d}p}v \tag{3.1}$$

$$= -\left(A^{-T}\frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}v}^{\intercal}\right)^{\intercal}\frac{\mathrm{d}A}{\mathrm{d}p}v \tag{3.2}$$

The first factor of this product is the solution to the *adjoint problem*

$$A^{\intercal}\tilde{v} = \frac{\mathrm{d}f_{\text{obj}}}{\mathrm{d}v}^{\intercal}, \tag{3.3}$$

hence the name adjoint simulation. As it turns out, A is often symmetric (or self-adjoint) which means that this is simply a normal simulation but with $\mathrm{d}f_{\text{obj}}/\mathrm{d}v^{\intercal}$ (or $\mathrm{d}f_{\text{obj}}/\mathrm{d}v^{\dagger}$) as the source. Thus, to obtain the derivative we just need to run an additional simulation with a different input.

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 three-tensor with dimension $m \times l \times n$. Thus calculating $A^{-1}\mathrm{d}A/\mathrm{d}p$ directly involves solving $Ax = w$ for a three-tensor, and calculating $A^{-1}\left(\mathrm{d}A/\mathrm{d}p\, v\right)$ involves solving for a matrix, both of which are orders of magnitude more computationally expensive than solving for a vector.

3.1.2 Specific derivation with acoustics

Now we turn to the specific case of acoustic devices. Here $Av = b$ is replaced by the dynamic equation of linear elasticity:

$$\hat{A}_{ik}u_k = F_i \tag{3.4}$$

dynamic equation? Better name

Instead of vectors, like we saw in section 3.1.1, these quantities are now functions[2] of $\boldsymbol{x}$. Analogously to the vector of design parameters we now have a *design field* $p(\boldsymbol{x})$.

[2]Vector-valued funcitons, but that is not the important part here.

On functionals and their derivatives

Big fat box on functionals and their derivatives. I think this should be included somewhere, since very few of my peers know what a functional derivative is... Not really sure how though. I kinda like the thought of putting it in a box like this. Alternatively, I could put it in an appendix.

Our 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 I will give an overview of the notational conventions I use, and then 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 notated 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 it's 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

$$\frac{\delta F}{\delta f_2}(x) \qquad \text{meaning} \qquad \frac{\delta F[g_1, g_2]}{\delta g_2}(x) \tag{3.5}$$

$$\frac{\delta F}{\delta f_2}(x) \qquad \text{meaning} \qquad \frac{\delta F[g_1, g_2]}{\delta g_2}[f_1, f_2](x) \tag{3.6}$$

where in the latter case, f_1 and f_2 are specific functions defined previously.

The functional derivative is defined by

$$\int \frac{\delta F}{\delta f}(x)\varphi(x)\,\mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}\varepsilon} F[f + \varepsilon\varphi] \tag{3.7}$$

where F is a functional of f and φ is an arbitrary test function.

I find the definition somewhat difficult to comprehend and really didn't understand it until I sat down and did some examples for myself. Maybe talk about analogies to vector functions...

I 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

$$\frac{\delta H}{\delta f}(x) = \int \frac{\delta F}{\delta G[f]}(y) \frac{\mathrm{d}G(y)}{\mathrm{d}f}(x)\,\mathrm{d}y \tag{3.8}$$

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

$$f_{\text{obj}}[\boldsymbol{u}] = \int_\Omega u_i(\boldsymbol{x})\varphi_i^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}. \tag{3.9}$$

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, we will use the chain rule to expand $\delta f_{\text{obj}}/\delta p(\boldsymbol{x})$. However, because $\boldsymbol{u}$ is in general complex, I will split it into its real and imaginary components: $u_i = v_i + iw_i$.

$$\frac{\delta f_{\text{obj}}}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y} \frac{\delta f_{\text{obj}}}{\delta v_i}(\boldsymbol{y}) \frac{\delta v_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) + \frac{\delta f_{\text{obj}}}{\delta w_i}(\boldsymbol{y}) \frac{\delta w_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.10}$$

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

$$\begin{aligned}
\frac{\delta f_{\text{obj}}}{\delta v_i}(\boldsymbol{y}) &= \frac{\delta}{\delta v_i(\boldsymbol{y})} \int_\Omega u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.11)\\
&= \int_\Omega \frac{\delta}{\delta v_i(\boldsymbol{y})} u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.12)\\
&= \int_\Omega \delta(\boldsymbol{x} - \boldsymbol{y})\delta_{ij}\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.13)\\
&= \varphi_i^*(\boldsymbol{y}) && (3.14)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\delta f_{\text{obj}}}{\delta w_i}(\boldsymbol{y}) &= \frac{\delta}{\delta w_i(\boldsymbol{y})} \int_\Omega u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.15)\\
&= \int_\Omega \frac{\delta}{\delta w_i(\boldsymbol{y})} u_j(\boldsymbol{x})\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.16)\\
&= \int_\Omega i\delta(\boldsymbol{x} - \boldsymbol{y})\delta_{ij}\varphi_j^*(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} && (3.17)\\
&= i\varphi_i^*(\boldsymbol{y}) && (3.18)
\end{aligned}$$

which gives us

$$\frac{\delta f_{\text{obj}}}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\frac{\delta v_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) + i\varphi_i^*(\boldsymbol{y})\frac{\delta w_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.19}$$

$$= \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\mathrm{Re}\left(\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x})\right) + i\varphi_i^*(\boldsymbol{y})\mathrm{Im}\left(\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x})\right) \tag{3.20}$$

$$= \int_\Omega \mathrm{d}\boldsymbol{y}\varphi_i^*(\boldsymbol{y})\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.21}$$

To find $\delta u_i(\boldsymbol{y})/\delta p(\boldsymbol{x})$ we apply $\delta/\delta p(\boldsymbol{x})$ to equation (3.4), which gives us

$$0 = \frac{\delta \hat{A}_{ik}}{\delta p}(\boldsymbol{x})u_k(\boldsymbol{y}) + \hat{A}_{ik}\frac{\delta u_k(\boldsymbol{y})}{\delta p(\boldsymbol{x})} \tag{3.22}$$

The point of inverse design is that we now want to find an ajoint field $\tilde{u}_i(\boldsymbol{y})$ such that the integral in equation (3.21) is

$$\int_\Omega \mathrm{d}\boldsymbol{y}\,\varphi_i^*(\boldsymbol{y})\frac{\delta u_i(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) = \int_\Omega \mathrm{d}\boldsymbol{y}\,\tilde{u}_i(\boldsymbol{y})\hat{A}_{ik}\frac{\delta u_k(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) \tag{3.23}$$

which by equation (3.22) is equal to

$$\int_\Omega \mathrm{d}\boldsymbol{y}\,\tilde{u}_i(\boldsymbol{y})\frac{\delta \hat{A}_{ik}}{\delta p}(\boldsymbol{x})u_k(\boldsymbol{y}). \tag{3.24}$$

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 $C_{ijkl}(\boldsymbol{y}) = C^0_{ijkl}p(\boldsymbol{y})$ implies

$$\frac{\delta \hat{A}_{ik}(\boldsymbol{y})}{\delta p}(\boldsymbol{x}) = -\rho^0\delta_{ik}\delta(\boldsymbol{x}-\boldsymbol{y}) - \partial_j\left(C^0_{ijkl}\delta(\boldsymbol{x}-\boldsymbol{y})\partial_l\,\cdot\right) \tag{3.25}$$

$$= -\rho^0\delta_{ik}\delta(\boldsymbol{x}-\boldsymbol{y}) - C^0_{ijkl}\delta(\boldsymbol{x}-\boldsymbol{y})\partial_j\partial_l. \tag{3.26}$$

Plugging this back into the integral in (3.24) gives

$$-\rho^0\tilde{u}_i(\boldsymbol{x})u_i(\boldsymbol{x}) - C^0_{ijkl}\tilde{u}_i(\boldsymbol{x})\partial_j\partial_l u_k(\boldsymbol{x}) \tag{3.27}$$

which is comparatively easily evaluated. The only thing remaining is to calculate $\tilde{\boldsymbol{u}}$. The way to do this is through an *adjoint simulation*.

When solving these equations in practice, space is discretized and the fields are represented by vectors and the operators by matrices. Thus (3.23) becomes

$$\varphi_a^*\eta_{ab} = \tilde{u}_c A_{ca}\eta_{ab} \tag{3.28}$$

where $\delta\boldsymbol{u}/\delta p$ has been labeled η for notational simplicity, and the indices go over the discretized spacial meshpoints. Meaning that to find $\tilde{u}$ we solve

$$\varphi_a^* = \tilde{u}_c A_{ca} \tag{3.29}$$

or, taking the transpose of both sides

$$A^\intercal_{ac}\tilde{u}_c = \varphi_a^*. \tag{3.30}$$

3.2 Optimization Algorithms

In the last section I painstakingly derived how one can obtain the gradient, and in this section I will attempt to justify that by describing how one can use the gradient. I will begin by describing the advantages of gradient based optimization algorithms over those that don't use the gradient. Following that I describe the algorithm that I used, as well as some of it's predecessors.

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 would require a large number of points before a good value is found, which means that it would take 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, However, if the input to your function is very high-dimensional, choosing a random direction step in for your next try is very unlikely to yield a much better value. For such functions, it is essential that you know in which direction the function increases the fastest, so that you can choose your next point in that direction. That is why we want to use gradient based optimization algorithms; they enable us to quickly find the right direction to go in for best improvement.

more examples

3.2.1 Gradient Descent

The simplest gradient based optimization algorithm is called *gradient descent*. Like all of the algorithms I will describe it is an iterative algorithm, meaning that it 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

$$p_n = p_{n-1} + \eta g_{n-1} \tag{3.31}$$

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 optimium might be missed entirely. A too low value results in too small steps which will yield a slow convergence. Almost all gradient descent implementations use a so called learning schedule, which means that η is not constant during the learning. This introduces the additional problem of how to choose 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.

CFigure 3.1 shows the difference in performance between ADAM and GD. With a slightly too large learning rate, the GD algorithm gets stuck in an oscillation and makes very 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.

citations

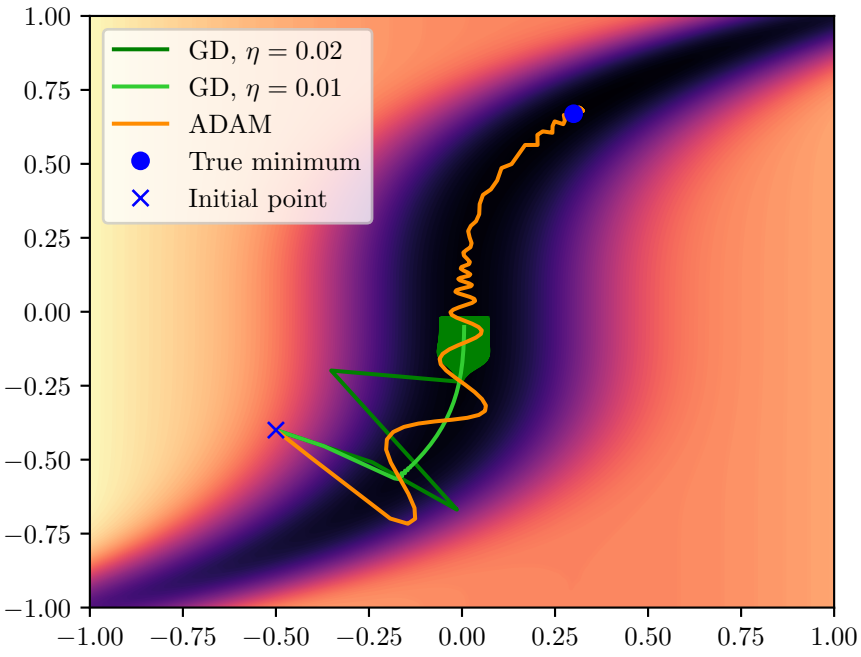

Figure 3.1: *A comparison between ADAM and GD in an optimization landscape with a narrow canyon. The two different GD algorithms are shown with 1000 steps, while 200 steps with the ADAM algorithm are shown.*

Pseudocode for the algorithm

Somewhere I should write about epsilon, and alpha and how I set them, but that probably needs to come later, in the method

4. Methods

The aim of this thesis is to use inverse design to find a phononic beamsplitter, a task that can be divided into three parts: First, some definitions of what should be designed and what constitutes a "good" design needs to be made. Second, we need a way to calculate the gradient of the "goodness" with respect to the design. And lastly, we need a gradient based optimization algorithm to find the optimal design. All of this will be described in this chapter.

4.1 Design

The device design to be optimized can be seen in figure 4.1. The input and output waveguides consists 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.[4]

Inside the design area, there can be one of two kinds of designs. The first is a *continuous design*, meaning that the material parameters ρ and C_{ijkl} are continuously varying. The range of values that they can take are between the density and elasticity of pure silicon and that of air. Any in-between values are obviously not something that can be physically realized, but it is useful as a first step in the optimization. The second kind is a binary design, where each point either has silicon or not and there are no in-between values. This is accomplished using level-set methods, which will be explained in section 4.1.4.

Because the device is completely symmetric, only one half of it needs to be modeled, and the other half is extrapolated with a symmetry boundary condition.

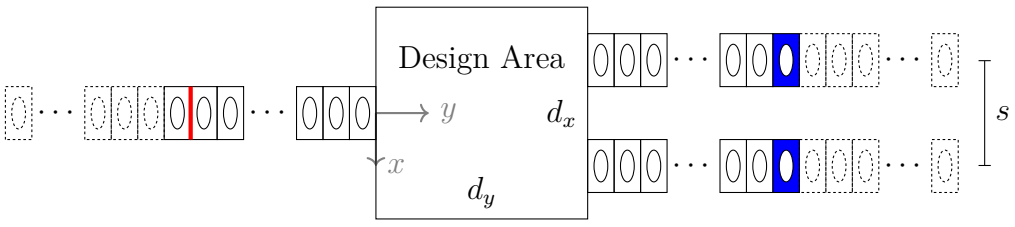

Figure 4.1: *Device design to be optimized. At the red line, a wave traveling right is excited. On the blue unit cells is where the output is measured. The dashed unit cells are PMLs*

Are the dx and dy in figure 4.1 clear? I thought of having arrows but it feels like the image gets quite messy then

Table 4.1: *Values for the geometric parameters of the device. Reference figures 2.1 and 4.1 for what the quantities mean. The only parameter not mentioned there is h, which is the height of the structure.*

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	$3w$

Write about why we use the mode we use, and why I clamp the bottom. Can reference Johan and Pauls paper

4.1.1 Objective function

The figure of merit of the device is how much of the input excitation gets transmitted into the output beams. Furthermore, all of the excitation of the output waveguide should be in the same mode that was excited at the input. Therefore, a mode overlap integral is used:

$$I = \int_{\Omega_1} \boldsymbol{m}^*(\boldsymbol{x})\boldsymbol{u}(\boldsymbol{x}) \tag{4.1}$$

where $\boldsymbol{m}$ is the shape of the mode (figure 2.3a). Because the phase of the output waves does not matter, the absolute value squared of the overlap integral is taken to be the objective function,

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

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.

Paragraph about pure part of objective function, enforcing the minimum feature size

4.1.2 Excitation

In order 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 and applied to the boundary marked in red in figure 4.1. Since the frequency is perfectly controlled, this should excite 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 unitcells was created. Applying the excitation and running the simulation, the mode overlap integral was then calculated, as well as the integral $\int u_i^* u_i$ from which one can calculate the proportion of energy in the mode compared to the energy in other modes. The result was near perfect (>99.9%) excitation of only the desired mode.

4.1.3 Perfectly Matched Layers (PMLs)

Ideally, the input and outputs are infinite waveguides. Unfortunately, it has been discovered that infinity is big; simulating infinite waveguides would take infinite time, and the author would like to be done by June. Instead, PMLs are placed at the caps of the input and output waveguides. The purpose of the PML is 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. The way to accomplish this is to add an imaginary component to the density of the material. This needs to be done smoothly, otherwise the abrupt change in material parameters would induce reflections anyway. Therefore, the imaginary part is taken to be exponentially increasing. Furthermore, the curve is shifted such that it starts at 0, and rescaled so that the endpoint is at s.

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

Figure 4.2 shows the effect of changing these parameters on the shape of the profile of the imaginary component.

Write about the long waveguide simulation and fourier transform to show that the pml was indeed reflectionless.

Simulations currently running... There will be a graph of relfection as a function of strength for a bunch of different d.

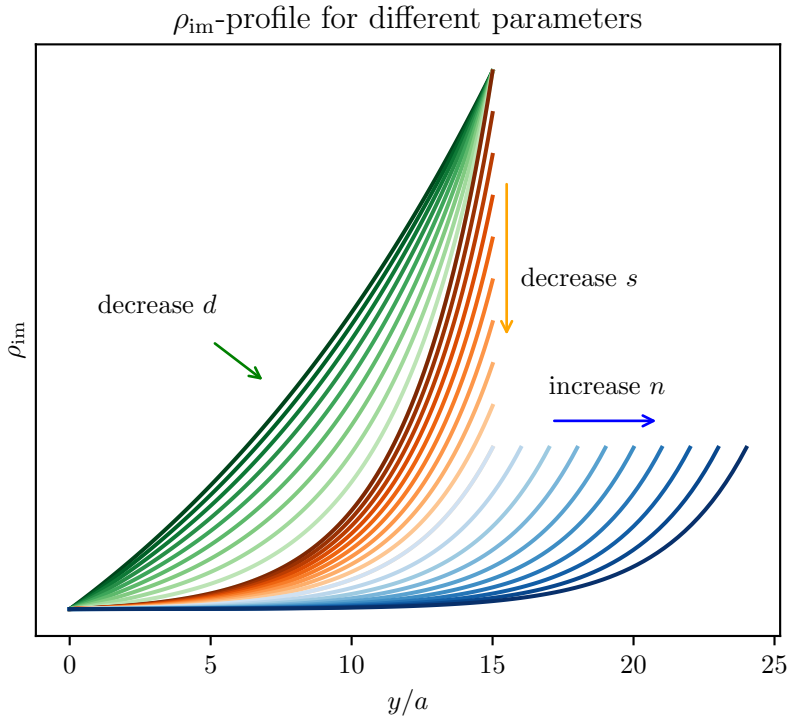

Figure 4.2: *This figure shows the effect of changing different parameters. The green curves shows changing d while keeping the other parameters fixed, and the orange and blue show s and n respectively. Darker colour means higher value, and the last green curve coincides with the first orange, and the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

4.1.4 Level-set

Ultimately, we want our device to consist of regions of material and regions of no material. There are basically two ways of doing this. The first, and perhaps most intuitive, is to simply store the coordinates of the boundary between the filled and empty regions. In addition to storing the coordinates, one must also store which points neighbour which. The second method, which is the one used in this report, is called the *level-set method*. In this method, the boundary is not directly stored, but rather is stored via an *implicit function*, $\phi(x)$, defined such that the boundary is the 0-isocontour of ϕ.

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 boundaries merge, or if one splits in two, points need to be removed and the connectivities changed, which is quite complex. Figure 4.3 illustrates these problems with direct storage concretely. Both of these issues are automatically handled with the level-set method.

How? (Isn't it obvious?)

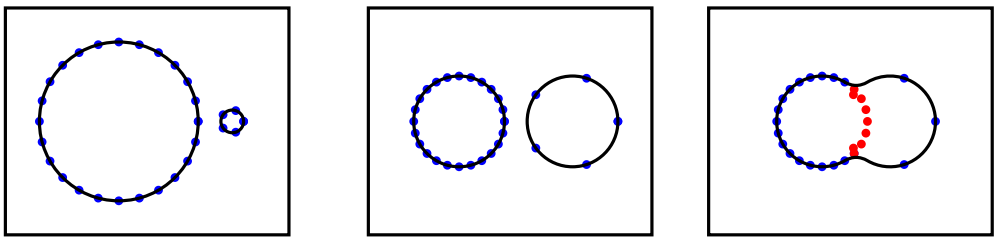

Figure 4.3: *Possible evolution of boundary. In the leftmost figure, the boundary is defined by pretty much evenly spaced points. In the center figure the boundaries have moved and the spacing is no longer even, and the right circle is very poorly resolved. The rightmost figure shows the boundary after 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.*

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. 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 4.4 shows this effect in one dimension.

Create another figure that shows it in two dimensions. I'm thinking a circular boundary, and adding s in the left half and subtracting s in the right half. Alternatively adding $s \cdot x$ (unit circle centered on 0) so that it will be smooth

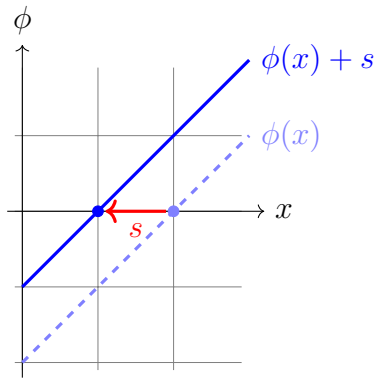

Figure 4.4: *Adding s to the signed distance function shifts boundary by s.*

As the update algorithm when optimizing level-set designs, the gradient is simply added to the signed distance field. Some care needs to be taken in how this is done though. Firstly, the gradient obviously needs to be rescaled so that the boundary moves an appropriate distance. This has been done such that the boundary moves maximally 5 nm. Secondly, since the gradient is occasionally sharply peaked somewhere which may not lie near 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 will now no longer be that, and thus the signed distance field is recalculated from the new boundary. This recalculation comes with a performance penalty, but since the COMSOL simulations are orders of magnitude slower than all other parts of the optimization, this is of little concern.

check this number before finalizing, I change it every now and then

4.2 Simulations

How much specifics should I have? Should I write about exporting the gradient to a file and how I calculate the 2D gradient in my matlab scripts from that

Mesh export / import and why that is done should probably be mentioned since it's quite important to get the excitation in the right mode.

4.3 Optimization

Describe what optimization algorithm was used, as well as how this changed during the simulation. E.g. first 200 iterations ADAM; next ADAM but with sigmoid function application; sigmoid + feature size; and finally level-set.

5. Results

6. Conclusion

References

[1] 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).

[2] 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).

[3] 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].

[4] 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).

A. First appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

B. Second appendix

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.