2019-04-01

Machine Learning Methods for Nanophotonic Design, Simulation, and Operation

Alec Michael Hammond
Brigham Young University

Machine Learning Methods for Nanophotonic Design, Simulation, and Operation

Alec Michael Hammond

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Ryan M. Camacho, Chair
Stephen M. Schultz
Daniel E. Smalley

Department of Electrical & Computer

Brigham Young University

ABSTRACT

Machine Learning Methods for Nanophotonic
Design, Simulation, and Operation

Alec Michael Hammond
Department of Electrical & Computer, BYU
Master of Science

Interest in nanophotonics continues to grow as integrated optics provides an affordable platform for areas like telecommunications, quantum information processing, and biosensing. Designing and characterizing integrated photonics components and circuits, however, remains a major bottleneck. This is especially true when complex circuits or devices are required to study a particular phenomena.

To address this challenge, this work develops and experimentally validates a novel machine learning design framework for nanophotonic devices that is both practical and intuitive. As case studies, artificial neural networks are trained to model strip waveguides, integrated chirped Bragg gratings, and microring resonators using a small number of simple input and output parameters relevant to designers. Once trained, the models significantly decrease the computational cost relative to traditional design methodologies.

To illustrate the power of the new design paradigm, both forward and inverse design tools enabled by the new design paradigm are demonstrated. These tools are directly used to design and fabricate several integrated Bragg grating devices and ring resonator filters. The method's predictions match the experimental measurements well and do not require any post-fabrication training adjustments.

Keywords: integrated optics, silicon photonics, nanophotonics, machine learning, artificial neural networks

ACKNOWLEDGMENTS

I am grateful for my undergraduate and graduate academic experiences at BYU. I am blessed to have been in such a positive and challenging environment where I could constantly push myself with the constant support of my friends.

I want to thank my graduate advisor, Dr. Ryan Camacho, for his unique guidance and friend-ship. He opened my eyes to the world of photonics and taught me how to approach engineering as a true scientist.

I'm also very grateful to my undergraduate research advisor, Dr. Stephen Schultz, for mentoring me and preparing me for what he knew was best for my academic career. In addition, I want to thank Dr. Daniel Smalley for his example and help throughout my studies.

I have had the privilege to work with so many amazing students in several different research labs and classes and am very grateful for their friendship.

Most importantly, I want to thank my wife, Jolene, for her never ending love and support. She has always been there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Motivation

Nanophotonic technologies have become powerful tools for processing classical and quantum information. In the era of big data, integrated photonics is increasingly used for optical interconnects [1], on-chip signaling [2], and on-chip photonic processing [3]. Processing quantum information using integrated photonics also shows great promise [4, 5], with known advantages in applications ranging from long-distance quantum communications [6] to on-chip quantum simulation [7, 8].

Designing and characterizing integrated photonics components and circuits, however, remains a major bottleneck [9]. This is especially true when complex circuits or devices are required to study a particular phenomena. Current design flows are complicated by computational tractability and the need for researchers with extensive experience [10]. Unlike their electronic counterparts, photonic integrated circuits require computationally expensive simulation routines to accurately predict their optical response functions. In fact, the typical time to design integrated photonic devices now often exceeds the time to manufacture and test them.

To address this challenge, this work proposes and experimentally validates a new design paradigm for integrated photonics that leverages machine learning methods in an intuitive way and is at least four orders of magnitude faster than traditional simulation methods.

This work builds on recent theoretical results showing that it is possible to model nanophotonic structures using machine learning. For example, Ferreira et al. [11] and Tahersima et al. [12] demonstrated that artificial neural networks (ANN) could assist with the numerical optimization of waveguide couplers and integrated photonic splitters respectively. In both cases the input parameter space was the entire 2D array of grid points, showing the power of ANNs in blind "black box" approach, though limiting the designer's ability to intuitively adjust input parameters. A re-

lated approach has also been applied to periodic photonic structures, which are often difficult to efficiently model [13, 14].

Two recent theoretical papers by Zhang et al [15], and Peurifoy et al. [16] go beyond simply optimizing over a large parameter space, and used ANNs to calculate complicated spectra using a smaller number of intuitive, smoothly varying input parameters. Even though in both cases each wavelength point in the calculated spectra required its own ANN output neuron, the usefulness of using ANNs to model systems with intuitive input parameters was demonstrated.

To illustrate the power of our new design paradigm, we demonstrate both forward and inverse design tools that use a chirped Bragg grating ANN as a computational backend. The forward design tool is interactive, and was used to design our fabricated circuits. The inverse design tool quickly constructs a temporal pulse compressing chirped Bragg grating within specified design constraints — a task typically too computationally expensive for traditional methods.

This new approach provides benefits such as rapid prototyping, inverse design, and efficient optimization. We demonstrate practical confidence in our method's accuracy by fabricating and measuring devices that experimentally validate the models' predictions.

1.2 Summary of Contributions

This thesis documents several original research contributions that not only establish a new simulation paradigm within the photonic community, but add value to BYU's CamachoLab.

- **Integrated photonic experimental lab setup.** Designed, assembled, and streamlined two interrogation workstations used to test photonic integrated circuits. Consists of swept laser source, photodiodes, microscope, polarization controller, DAQ, oscilloscope, various adjustable alignment stages, fiber arrays, lensed fibers, and control software written in python to automatically record and correlate data.

- **Design, fabrication, and testing of silicon photonic devices.** Designed various silicon photonic devices for several applications (microfludics, quantum transceivers, pulse compressors, cascaded ring filters, etc) for a variety of foundries. Fabricated some devices at BYU and interfaced with other foundries (Applied Nanotools inc and University of Washington). Tested devices in house.

- **BYU process development kit.** Generated a library of devices and components commonly used within the CamachoLab and generated a corresponding process development kit (PDK).

- **Open source nanophotonic simulation platform (SiP-ANN).** Generated datasets and trained various machine learning models for devices like ring resonators, waveguides, and Bragg gratings. These models are wrapped in a python interface and implemented in graphical design tools, inverse design solvers, parameter extraction scripts, and Monte Carlo routines [17].

- **Open source python mode solver (pyMode).** Upgraded the existing open source mode solver WGMS3D [18] (written in C and Fortran) to work with modern compilers and dependencies and developed a python interface. This new package generates the datasets necessary for the various machine learning models. The lightweight scripting nature enables deployment on high performance computing platforms and is capable of modeling several diverse waveguide conditions (e.g. waveguide bends, lossy materials, waveguide symmetries, etc.) [19].

- **Green Machine characterization algorithm.** Derived and demonstrated on hardware a characterization algorithm required to deploy structured optical receivers for photon-starved environments [20].

1.3 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 gives an overview of various machine learning methods, like artificial neural networks, linear regression, and novel optimization approaches. Chapter 3 applies these methods directly to nanophotonic device modeling. This section analyzes common nanophotonic devices, like waveguides, Bragg gratings, and ring resonators. Chapter 4 leverages these models to accelerate and enable various designer tasks, like Monte Carlo analysis, parameter extraction, and inverse design. Finally, chapter 5 extends these tools for quantum applications, specifically with structured optical receivers operating in photon-starved environments.

CHAPTER 2. MACHINE LEARNING METHODS

In this chapter, we explore a few machine learning methods used to train models for the different nanophotonic components. As is always the case with machine learning and optimization problems, no single algorithm or model performs well for all applications. It is up to the designer to test various algorithms and determine which one works best given the constraints of the problem (e.g. training time, dataset complexity, etc.).

We analyze three different methods used to model nanophotonic devices throughout this work. First, we look at multivariate polynomial regression. Polynomial regression is typically the easiest to implement, fastest to train, and works exceptionally well on smooth function mappings. We then turn to artificial neural networks, which are much more computationally expensive to train, but successfully model complicated device transfer functions like integrated chirped Bragg gratings. Finally, we explore how optimization can be used to learn device transfer functions on the fly and implement quantum routines.

2.1 Multivariate Polynomial Regression

Multivariate polynomial regression consists of determining a linear combination of basis terms that sufficiently represents the mapping from the inputs to the outputs. Specifically, this method relies on polynomial basis functions of the form

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \ldots, \tag{2.1}$$

where y corresponds to all the samples of one particular output, x_1, x_2, ..., x_m describe all of the samples for each of the input degrees of freedom, and a_i are the corresponding coefficients that minimize an error term e. Using all of the data samples, we relate the input, output, and error term with

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(n) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x_1(1) + a_2x_2(1) + ... \\ a_0 + a_1x_1(2) + a_2x_2(2) + ... \\ \vdots \\ a_0 + a_1x_1(n) + a_2x_2(n) + ... \end{bmatrix} + \begin{bmatrix} e(1) \\ e(2) \\ \vdots \\ e(n) \end{bmatrix} \tag{2.2}$$

and define the following vectors:

$$\mathbf{y} = \begin{bmatrix} y(1), & y(2), & ..., & y(3) \end{bmatrix}^T \mathbf{e} = \begin{bmatrix} e(1), & e(2), & ..., & e(3) \end{bmatrix}^T \mathbf{c} = \begin{bmatrix} a_0, & a_1, & ..., & a_m \end{bmatrix}^T. \tag{2.3}$$

We can then define the linear transformation A with each of the basis functions sampled similarly to the above vectors:

$$A = \begin{bmatrix} 1 & x_1(1) & x_2(1) & ... \\ 1 & x_1(2) & x_2(2) & ... \\ \vdots & & & \\ 1 & x_1(n) & x_2(n) & ... \end{bmatrix}. \tag{2.4}$$

We can now formulate the traditional least-squares problem with

$$\mathbf{y} = A\mathbf{c} + \mathbf{e} \tag{2.5}$$

and minimize the l_2 norm of the error vector $\mathbf{e}$ using

$$\mathbf{c} = (A^H A)^{-1} A^H \mathbf{y}, \tag{2.6}$$

where $\mathbf{c}$ is a vector of the corresponding basis function weights. Solving for the model's weights reduces to a (relatively simple) pseudo-inverse problem commonly found in various aspects of signal processing and computational mathematics. This method is guaranteed to find the minimum error (in the least squares sense).

As the complexity of the model increases (i.e. with higher order terms) then it is common for the condition number of matrix A, which indicates how ill-conditioned the matrix is, to drop exponentially. Consequently, the pseudoinverse is very sensitive to small perturbations and in some

cases cannot be computed numerically. To avoid such complications, orthogonal basis functions (like the Legendre polynomials) can be used.

While this method is easy to implement from scratch, several open source python packages already exist that interface well with other packages. For this work, scikit-learn was used [21].

As is the case with all machine learning methods and algorithms, the difficulty in implementing multivariate polynomial regression lies with selecting which basis functions to use. Fortunately, due to its computational simplicity, different model variations can be trained and tested relatively quickly. In addition to polynomials, other nonlinear basis functions, like Lorentzians, Gaussians, sines, cosines, and exponentials, can also be implemented. If parameters outside of the scaled weights are required (as is the case with Lorentzians and Gaussians) then more sophisticated minimization techniques are required. These more complex basis functions create networks that look more like other machine learning methods (like radial basis networks) rather than simple multivariate regression models.

2.2 Artificial Neural Networks

Artificial neural networks model the relationship between their inputs and outputs by cascading various nonlinear computational units, known as neurons [22]. ANNs learn the desired relationship between inputs and outputs by tuning each neuron in response to a training set. Training optimization algorithms, such as backpropagation, are used to strategically introduce these datasets into the neural network and gradually update the network's parameters until a convergence criteria is met [23]. If the network architecture is sufficiently large, any arbitrary relationship manifested in the dataset can be modeled [24].

Just like with multivariate regression, the designer must adequately choose an appropriate architecture over which to train the ANN. Network parameters like the number of layers, the number of neurons per layer, and the activation function, along with various training parameters, like the number of epochs, the batch size, and the learning rate all affect the ANN's final performance and expressivity.

The functional form of each neuron is

$$x_{n+1,k} = f(\Sigma_i w_{,i} x_{n,i} + b_{n,i}), \tag{2.7}$$

where $w_{n,i}$ is the weight for the n^{th} layer and the i^{th} node in that layer, $x_{n,i}$ is the output of the corresponding neuron, and $b_{n,i}$ is the corresponding bias. Various nonlinear functions f are used throughout the literature, including hyperbolic tangent and sigmoid. Once a particular architecture is chosen, the backpropogation algorithm formulates a cost function and together with an optimizer, determines the weights and biases that best relate the inputs to the outputs.

As with all deep learning applications, the network's utility is limited by biases introduced in the training set, the network architecture, or even the training process itself [25]. Fortunately, we can anticipate these biases by extracting the model's prediction uncertainty without modifying our network architecture. Dropout inference techniques leverage models that rely on dropout layers to mitigate over-fitting (a form of network bias) [26]. Even pre-trained networks can use dropout inference to extract prediction uncertainties without any modifications to the network. This particular network design methodology opens the door to many more applications, like training on fabricated device data. Foundry's that develop process design kits (PDKs), for example, can use this technique to model their fabrication processes while preserving their trade secrets.

2.3 Optimization

Optimization methods are required for almost all machine learning methods. However, optimization itself can be used to characterize and operate nanophotonic devices such that the algorithm "learns" the device's response. For example, if a particular device has a known analytic transfer function with tunable device parameters, then an optimization algorithm can learn these parameters by minimizing some user-defined cost function (also known as parameter extraction).

Chapter 4 illustrates a parameter extraction method for fabricated integrated chirped Bragg gratings (ICBG) and ring resonator filters. This is especially important for active devices controlled via feedback loops. Chapter 5 describes this type of feedback algorithm with a quantum photonic communications transceiver. A simple convex optimizer constantly learns the new codewords of the device in realtime.

CHAPTER 3. DEVICE MODELING[1]

To demonstrate this new modeling framework, we design, fabricate, and test simple integrated waveguides, integrated chirped Bragg gratings (ICBGs), and microring resonators (MRR). We pedagogically lead the reader by demonstrating each new model in that order. ICBGs have large parameter spaces and nonlinear responses and are generally representative of devices that are computationally prohibitive using other techniques. The experimental results show remarkable agreement with the ICBG ANN's predictions, and to the authors' knowledge represent the first experimental validation of photonic devices designed using ANN's. Ring resonators are also rather difficult because they demonstrate extreme sensitivity to fabrication defects.

3.1 Waveguide Modeling

We begin by describing the modeling process for simple silicon photonic strip waveguides. We explore three varieties of waveguides: straight, bent, and evanescently coupled. These are each used to simulate the much more complicated ICBG and MRR responses. As a case study, we train the straight model using polynomial regression and artificial neural networks.

Different dataset generation tools were used depending on the constraints of the problem. For example, the simple straight waveguides were typically generated using MEEP Photonic Bands (MPB) [29], a finite difference eigenmode solver that distributes well and solves quickly. This method takes Maxwell's equations

$$\nabla \cdot \mathbf{D} = \rho \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{3.1}$$

$$\nabla \cdot \mathbf{B} = 0 \quad \nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + J \tag{3.2}$$

[1] This chapter is a modified version of papers written by Alec M. Hammond, Easton Potokar, and Ryan M. Camacho [27, 28].

and formulates them into a curl-curl Helmholtz equation (also known as the master equation) of the form

$$\nabla \times \left(\frac{1}{\varepsilon(\mathbf{r})} \nabla \times \mathbf{H}(\mathbf{r}) \right) = (\frac{\omega}{c})^2 \mathbf{H}(\mathbf{r}), \tag{3.3}$$

which can be reformulated as an eigenvalue equation

$$\Theta \mathbf{H}(\mathbf{r}) = \lambda \mathbf{H}(\mathbf{r}), \tag{3.4}$$

where MPB handles the operator discretization and solves for the appropriate eigenvalues. For the more complicated bent waveguides where Maxwell's equations must be solved in cylindrical coordinates

$$\begin{aligned} (1+c\rho)^2 L h^\rho + c(1+c\rho)\left(3h_\rho^\rho + 2h_z^z\right) + c^2 h^\rho &= \beta^2 h^\rho \\ (1+c\rho)^2 L h^z + c(1+c\rho) h_\rho^z &= \beta^2 h^z \end{aligned} \tag{3.5}$$

and $c = 1/R$ (the waveguide curvature)

$$L = \frac{\partial^2}{\partial \rho^2} + \frac{\partial^2}{\partial z^2} + k^2 n^2 \tag{3.6}$$

the modified WGMS3D package in conjunction with pyMode was used [18, 19].

3.1.1 Multivariate Polynomial Regression Approach

We first report on a waveguide multivariate regression model capable of estimating the effective index of any arbitrary silicon photonic waveguide geometry for a variety of modes. Specifically, we modeled the relationship between the waveguide's width, thickness, and operating wavelength and the effective index for the first two TE and TM modes. We note that including wavelength as an input parameter is a unique and enabling strategy not previously adopted. The regression models trained with fourth order polynomials and converged to a mean squared error (MSE) between 10^{-6} and 10^{-7}.

3.1.2 ANN Approach

We next report on a waveguide neural network. We generated our waveguide neural network's training set on a high performance computing cluster (HPC) using MEEP Photonic Bands (MPB) [29], a finite difference eigenmode solver. The solver simulated 31 different waveguide widths from 350 nm to 1500 nm and 31 waveguide thicknesses ranging from 150 nm to 400 nm resulting in 961 different geometries. The solver simulated 200 distinct wavelength points in the range of 1400 nm to 1700 nm. The total number of training samples fed into the neural network was 192,200. 70% of the data set was used as training samples and the remaining 30 % was used as validation samples. Each sample had three inputs (width, thickness, and wavelength) and four corresponding outputs (effective indices for the first two TE and TM modes). No postprocessing was performed on the waveguide training data.

The neural network was trained on the BYU office of supercomputing HPC cluster using Keras [30] and Tensorflow [31]. Several hundred different architectures were tested. To gauge the effectiveness of each architecture, the mean-squared-error and coefficient of determination (R^2) metrics were used. The waveguide neural network that worked best had 4 hidden layers with 128 neurons, 64, neurons, 32 neurons, and 16 neurons. Each neuron used a hyperbolic tangent activation function.

Figure 3.1 (f) compares the ANN's predicted effective index to a its corresponding simulation. The first TE and TM mode for any silicon photonic waveguide with a width between 350 nm and 1000 nm and a thickness between 150 nm and 350 nm are demonstrated. The network estimates a smooth response for both modes simultaneously, even for data points outside of its training set. The ANN's smooth output also produces smooth analytic derivatives, which are essential for calculating group index profiles and for gradient-based optimization routines.

We implemented various tests to validate the network's accuracy. First, we split the initial dataset into a training set and a validation set. While the network evaluated both sets after each epoch (i.e. training iteration) only the training set's results were used to update the network's weights. We monitored the validation set's results to assess overfitting. To better understand the network's performance after each iteration, we recorded each epoch's mean-square-error (MSE) and coefficient of determination (R^2). Figure 3.1 (a) and (b) illustrate the MSE and R^2 respectively after each epoch. To prevent overfitting, we stopped training at 100 epochs, where the MSE and R^2

appear to converge. At this point, the network demonstrated a MSE of 1.323×10^{-4} for the training set and 7.490×10^{-5} for the validation set. The final R^2 values for the training data and validation data were 0.9996 and 0.9997 respectively. The MSE and R^2 evolution for both the training set and validation set converge well, indicating little to no overfitting. Figure 3.1 (c) illustrates the relative error for both the training and validation sets after the final epoch. Both the training set errors and validation set errors are similarly distributed and tightly bounded between -1% and 1%, once again indicating little to no overfitting.

With confidence in the waveguide neural network's prediction accuracy, we benchmarked its speed and found that a single neural network evaluation was 10^4 times faster on average than the corresponding finite difference eigenmode simulation. Figure 3.1 (d) compares the computation speed for the ANN to the eigenmode solver, Meep Photonic Bands (MPB) [29]. This significant speedup enables many simulation techniques, like the layered dielectric media transfer matrix method (LDMTMM) [32] or the eigenmode expansion method (EMM) [33], where photonic components are discretized into individual waveguides. Using the ANN, a transfer matrix for each waveguide can be quickly generated and cascaded to formulate a fairly accurate response for the device. In addition, modeling fabrication variations is now much quicker since existing Monte Carlo sampling routines can leverage the ANN's speed.

We performed all benchmarks using a quad-core Intel(R) i5-2400 CPU clocked at 3.10 GHz with 12 GB of RAM. To evaluate the waveguide ANN's speed, we simulated various waveguide parameters in serial using both the ANN and MPB. To evaluate the BG ANN's speed, we simulated various grating's in serial using both the ANN and the LDMTMM. We linearly fit each method's results and compared the slopes to examine the speedup.

3.2 Bragg Grating Modeling

Modeling the relationship between a Bragg grating's physical design parameters and its corresponding responses is difficult since no one-to-one mapping exists. Consequently, many designers resort to black-box optimization routines that strategically search the parameter space for viable design options. As a result, inverse design problems — where a simulation needs to run each iteration — become intractable for even modest size gratings. If a full 3D FDTD simulation is performed, for example, each optimization iteration can take between 8-12 hours on typical

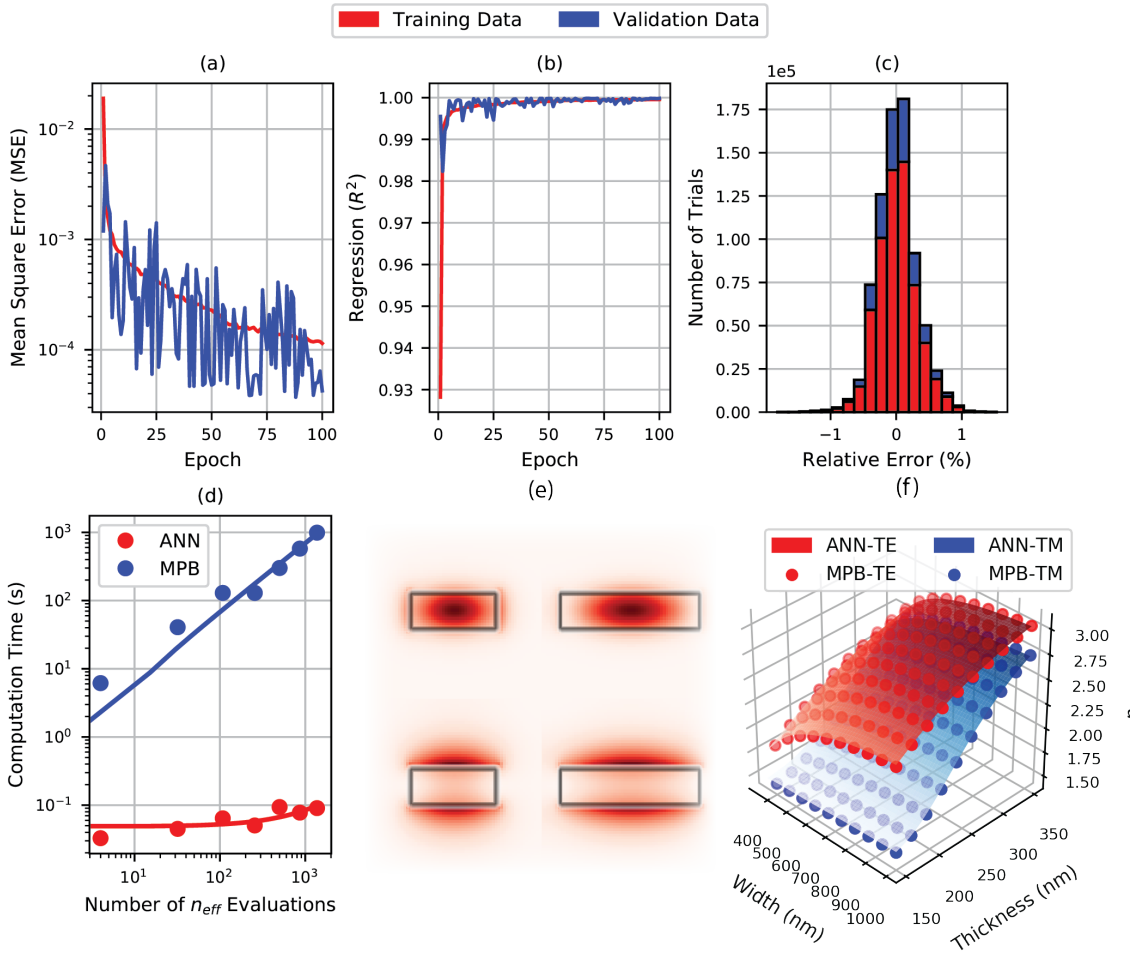

Figure 3.1: Waveguide artificial neural network training results demonstrated by the training convergence with reference to the mean square error (a), the coefficient of determination (b), and the residual errors after training (c). Panel (d) compares the computational cost for the ANN and the eigenmode solver that is used to simulate the mode profiles (e). Panel (f) exhibits the effective index profiles as a function of a waveguide geometry at 1550 nm for the first TE and TM modes.

desktop computing systems. In addition, the optimization routines tend to inefficiently simulate redundant test scenarios for different design problems. We train and demonstrate a Bragg ANN, however, that can predict a grating's response on the order of milliseconds on the same system, enabling much faster solutions to more complex design problems. We fabricate various test devices and validate our neural network's predictions.

3.2.1 Data Generation and Preprocessing

On the same HPC as before, we generated our Bragg training set by simulating 104,131 different gratings with the layered dielectric media transfer matrix method (LDMTMM) [32]. The LDMTMM method models each individual section of the Bragg grating as an ideal waveguide and cascades each sections's corresponding transfer matrix to estimate the grating's response for each wavelength point of interest. We calculated each individual waveguide's effective index using the waveguide neural network. Our simulations swept through 10 different corrugation widths from 10 nm to 100 nm, 11 different grating lengths from 100 periods to 2000 periods, and 32 different chirping patterns.

Bragg grating device responses are heavily influenced by their apodization [34]. For example, devices without apodization have much more ringing than devices with a raised cosine apodization. The effect is similar to finite impulse response (FIR) filtering windows [35]. Figure 3.2 illustrates various apodization windows and the resultant ringing. While the layered dielectric medium transfer matrix method (LDMTMM) can effectively simulate such ringing, fabrication inconsistencies make predicting the actual ringing close to impossible.

Consequently, designers focus more effort on shaping the Bragg grating's reflection, transmission, and group delay responses than mitigating ringing. This mentality encourages a design tool that temporarily ignores any ringing and simulates the basic grating response profiles.

To accomplish this, we filtered all apodization dependent ringing from the Bragg ANN's training set by fitting the reflection spectra and group delay responses. Through trial and error, we found that a generalized, skewed Gaussian of the form

$$f(\lambda, \lambda_0, \sigma, \beta, a, p, c) = \frac{a\sigma}{\gamma} e^{\frac{-\beta|\lambda-\lambda_0|^p}{\gamma}} + c, \tag{3.7}$$

where

$$\gamma = \frac{2\sigma}{1+e^{-\beta(\lambda-\lambda_0)}} \tag{3.8}$$

most comprehensively models both the reflection and group delay profiles for a wide array of tuning parameters. Each parameter strategically tunes a particular feature commonly found in both responses. For example, a simply scales the maximum reflectivity or maximum group delay, σ

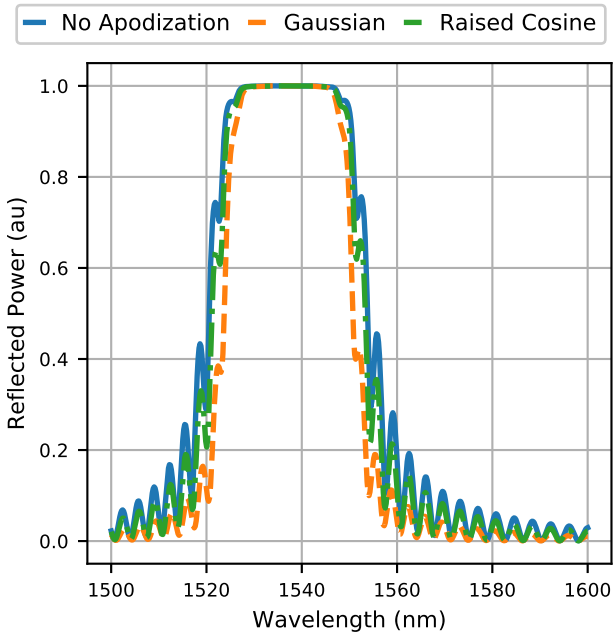

Figure 3.2: Reflection profile for an integrated chirped Bragg grating with no apodization (blue), a Gaussian apodization (orange), and a raised cosine apodization. Different apodization functions reduce the response's ringing.

shapes the reflection bandwidth or group delay spread, β induces skewness to one side or the other, p flattens the function's main lobe (i.e. for saturated reflection profiles), and c provides the necessary offsets for the group delay profiles. Figure 3.3 illustrates various LDMTMM simulations and the corresponding fit.

To find the optimal coefficients, we first ran a differential evolution algorithm that minimized the mean squared error between the LDMTMM simulation and the functional fit [36]. This global optimization routine provided a suitable starting point for a Levenberg-Marquardt algorithm to locate the local minima [37]. Using a two-staged optimization approach increased the robustness of the training processing and enabled batched parallelization on the high performance cluster (HPC).

We found that without fitting, the resulting oscillations significantly complicate the training process and restrict the network's domain to a single apodization. We fit both the reflection spectrum and group delay responses to generalized Gaussians and resampled the results with 250 wavelength points from 1.45 μm to 1.65 μm. Since the nonlinear fitting routine occasionally

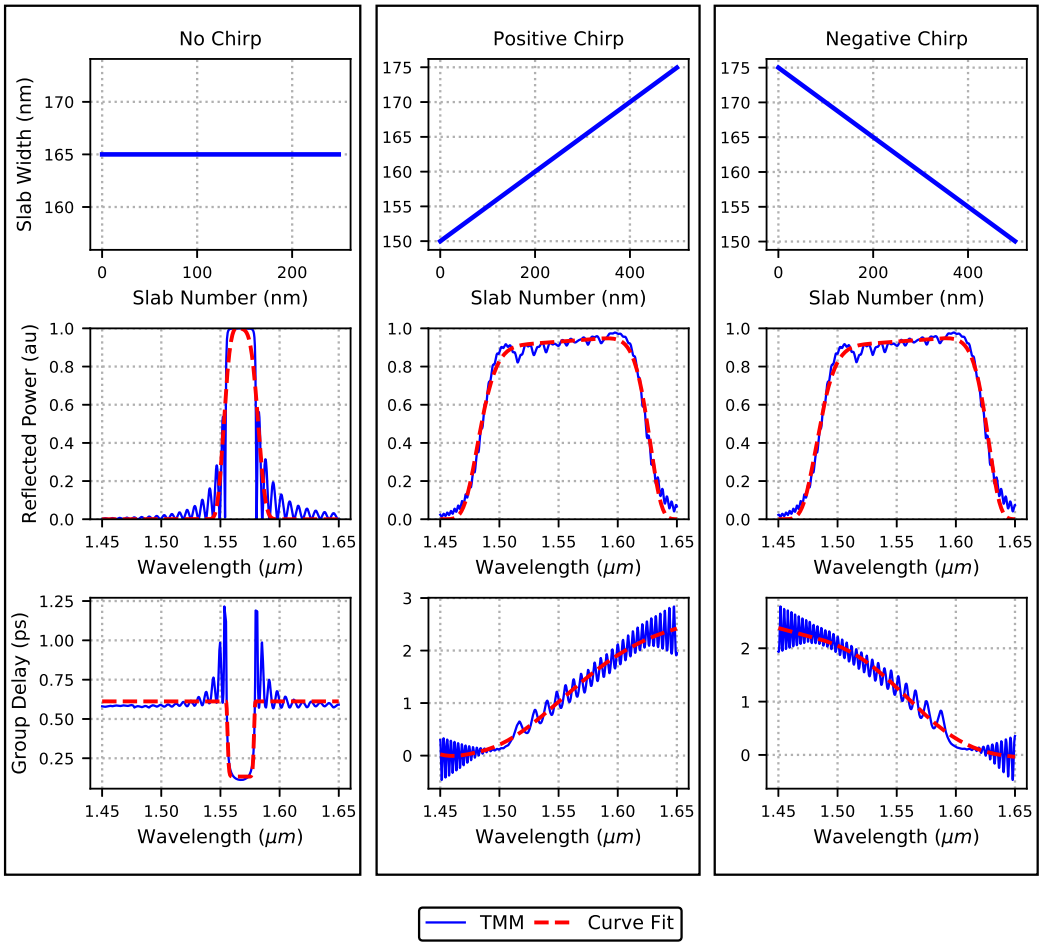

Figure 3.3: Demonstration of the fitting algorithm for a Bragg grating with no chirp (column one), a positive chirp (column two) and a negative chirp (column three). The chirp patterns themselves are illustrated in the first row, the reflection profiles are depicted in the second row, and the respective group delay responses are found in the third row. The modified Gaussian function accounts for the wider bandwidths, skewness, and overall shape of the responses will "filter" out the apodization dependent ringing.

failed, not all of the simulated gratings were appropriate for testing. After filtering through the results, we generated a database of 26,032,750 training samples.

3.2.2 ANN Training

Using the waveguide neural network, we generated a dataset to train our Bragg grating neural network to predict the reflection spectrum and group delay response of a silicon photonic, sidewall-corrugated, linearly chirped Bragg grating, as illustrated in Figure 3.4 (d). We note that

generating the training dataset was approximately 2 orders of magnitude faster using the waveguide ANN reported above rather than traditional methods. We parameterized the gratings by length of the first grating period (a_0), length of the last grating period (a_1), number of grating periods NG, and grating corrugation width difference($\Delta w = w_1 - w_0$). We designed the network to receive these four parameters along with a single wavelength point as inputs. The network has two outputs: reflected optical power and group delay.

Similar to the waveguide network, we divided the dataset into a training set and validation set. We tracked both the MSE and the R^2 metrics after each epoch. The Bragg training set was much larger than the waveguide training set, owing to the larger parameter space. Consequently, the MSE converged within the first few epochs and we stopped training after just five epochs to prevent overfitting. The final MSE for the training and validation sets was 1.845×10^{-4} and 1.677×10^{-4} respectively. The final R^2 was 0.9975 and 0.9977 respectively. Once again, the MSE and R^2 evolution for both the training set and validation set converge well, indicating little to no overfitting. Figure 3.4 (a-b) illustrates the network's MSE and R^2 evolution. Figure 3.4 (c) illustrates the absolute error for both the training sets and validation sets. We calculated the absolute error because several training samples were at or near zero and skewed the relative error.

We note that calculating Bragg grating response with the ANN is much more computationally efficient than previously demonstrated methods. This is because the Bragg ANN linearly increases in computation complexity with added grating parameters, while LDMTMM and all other methods known to these authors increase at least quadratically.

To validate the Bragg ANN, we fabricated and measured several integrated photonic Bragg gratings with different chirping patterns and compared their transmission, reflection, and group delay spectra to the neural network's predictions. The gratings were arranged in one of two configurations: (1) a simple circuit that only measured the Bragg grating's transmission spectra and (2) a more complicated interrogation circuit capable of measuring the reflection, transmission, and group delay profiles from the same device simultaneously. Figure 3.4 illustrates the interrogation circuit used to measure all three responses simultaneously. In both configurations, grating couplers were used to route light on and off the chip. While the simpler circuit required less de-embedding, the full interrogator circuit allowed for a more comprehensive device characterization.

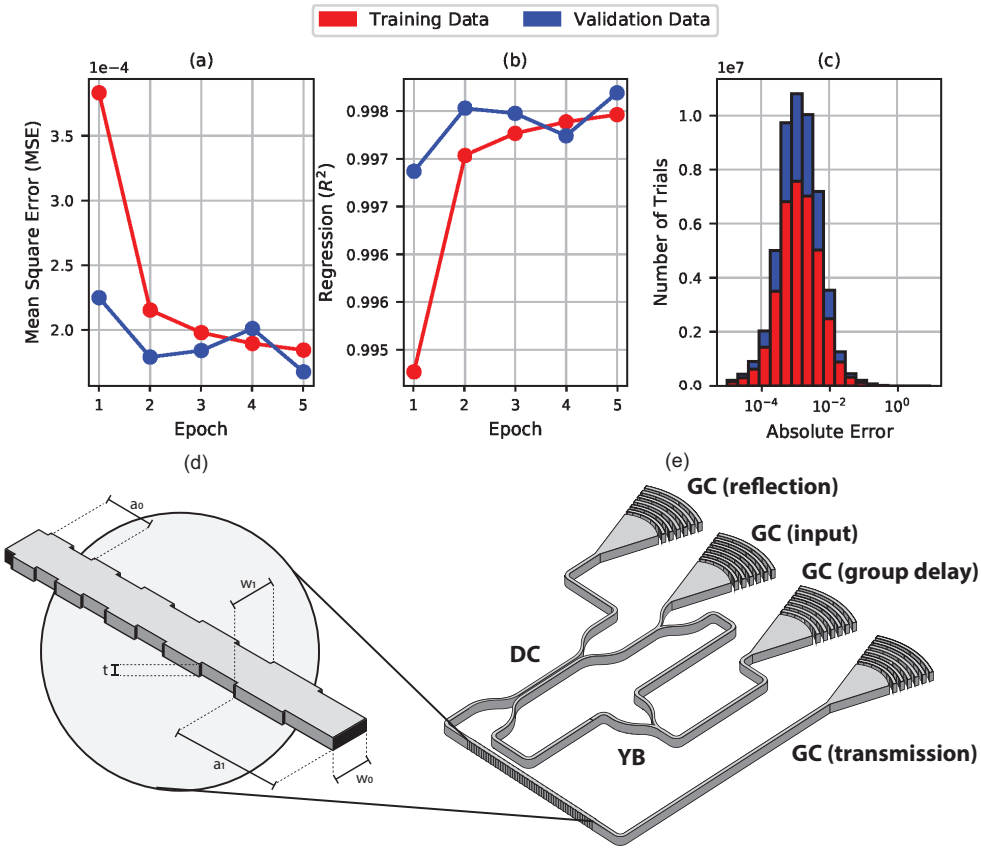

Figure 3.4: Bragg grating artificial neural network training results demonstrated by the training convergence with reference to the mean square error (a), the coefficient of determination (b), and the absolute error after training (c). (d) illustrates the different adjustable grating parameters and (e) illustrates the interrogation circuit used to extract the reflection, transmission, and group delay profiles simultaneously from the chirped Bragg grating. A grating coupler (GC) feeds light into various Y-branches (YB) and directional couplers (DC) such that the transmission and reflection spectra can both be extracted from the chirped Bragg grating (BG). Half of the reflected signal is sent through a Mach-Zehner Interferometer (MZI). The output of which is used to extract the group delay. .

The transmission-only gratings were designed with various grating period bandwidths from 5 nm to 20 nm, each with 600 periods and a corrugation width of 50 nm. The initial design parameters produced ANN predictions that match the measured data extremely well. Small discrepancies in the grating responses are largely attributed to the grating's apodization profile and detector noise. Figure 3.5 illustrates the comparison between the ANN's predictions and the measured data.

We designed the remaining gratings using a much smaller chirp bandwidth of 3 nm with 750 grating periods and a 30 nm corrugation width. We mirrored the orientation of half the gratings

in order to measure both positive and negative sloped group delay profiles. Once measured, we normalized the data by de-embedding the responses from the various Y-branches, directional couplers, and grating couplers that complicate the measurement data. Even with the rather complex transfer function, the transmission, reflection, and group delay profiles match the ANN's corresponding predictions well except for occasional resonant features caused by fabrication defects. These defects are expected since the narrow bandwidth devices have a grating pitch with a fine discretization that approaches the e-beam raster grid resolution. Small changes in grating pitch that don't align with the raster grid occasionally produce weak Fabry-Perot resonance conditions visible in the data. These raster-induced defects also account for a small lateral shift (˜1 nm) in the responses. Even with these fabrication challenges, it is notable that the ANN successfully predicts the transmission, reflection, and group delay profiles simultaneously. In fact, the ability to do so in noisy fabrication environments is one of the key advantages of the ANN and may allow for efficient parameter extraction where other methods fail.

3.2.3 Measurement Data Normalization

The photonic Bragg gratings were fabricated by Applied Nanotools Inc (Edmonton, Canada) using a direct-write 100 keV electron beam lithographic process. Silicon-on-Insulator wafers with 220 nm device thickness and 2 μm thick insulator layer were used. The devices were patterned with a raster step of 5 μm and etched with a ICP-RIE etch process. A 2.2 μm oxide cladding was deposited using a plasma-enhanced chemical vapor deposition (PECVD) process.

Each device was measured using an automated process at the University of British Colombia (UBC). An Agilent 81600B tunable laser was used as the input source and Agilent 81635A optical power sensors as the output detectors. The wavelength was swept from 1500 to 1600 nm in 10 pm steps. A polarization maintaining (PM) fibre was used to maintain the polarization state of the light, to couple the TE polarization in and out of the grating couplers. Several dembedded test structures were used to normalize out the coupler profiles.

The circuit used to simultaneously extract the transmission, reflection, and group delay profiles of the integrated Bragg gratings incorporated several additional devices that skewed the Bragg grating's transfer function. The grating couplers, Y-branches, and directional couplers used

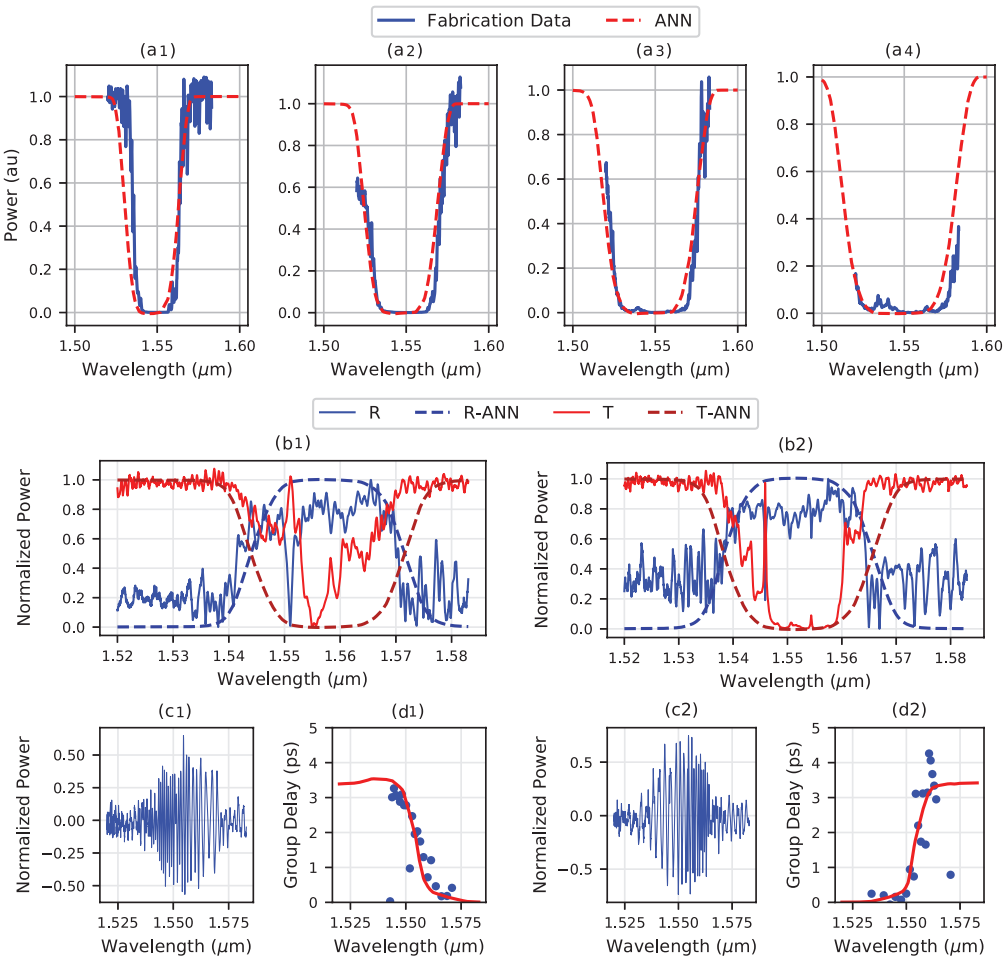

Figure 3.5: Fabrication data compared to corresponding ANN predictions. (a1-a4) Measured transmission responses for gratings with a period chirp of 5 nm (a1), 10 nm (a2), 15 nm (a3), and 20 nm (a4). (b1-b2) Transmission and reflection responses for two different Bragg gratings. Both gratings share the same design parameters, and have an identical but opposite linear chirp. The result of the mirrored chirping is seen in both the normalized MZI interference patterns (c1 , c2) and the extracted group delay responses (d1 , c2).

to route the signal all have non uniform frequency responses, and must be normalized out in order to accurately measure the Bragg grating's response.

Many designers fabricate de-embedded devices where each individual transfer function can be extracted and subsequently eliminated from the larger circuit [9]. Fabrication variability, however, prevents consistency from one device to another. The circuit's grating couplers, for example, sometimes showed 3 dB of variation across the band from one de-embedded structure to another. Consequently, this method cannot be reliably used to calibrate the Bragg grating responses.

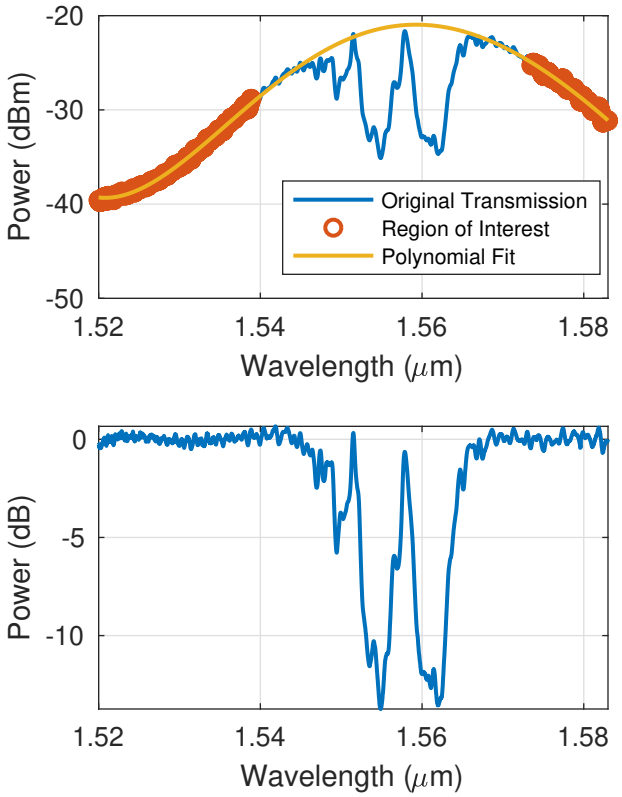

Figure 3.6: The normalization process for the integrated Bragg gratings. Data points outside of the stop band are fitted to a fifth degree polynomial to capture the response of the grating couplers and other devices within the circuit (top). The polynomial is then normalized from the data (bottom).

Instead, we fit the signal outside of the Bragg grating's reflection/transmission band using a fifth order polynomial that much more accurately describes the other devices' spectral influences. Figure 3.6 illustrates this procedure. Since the Bragg grating's group delay and magnitude responses are band-limited, we know what the rest of the spectrum should look like.

To extract the group delay from the interference pattern, we first normalize out the low frequency carrier by fitting the entire signal to a fifth degree polynomial. Then, we estimate the free spectral range (FSR) by tracking each oscillation peak. Using the transfer function of the MZI [9] along with the FSR, the resulting group index and group delay is estimated. Figure 3.7 illustrates this process.

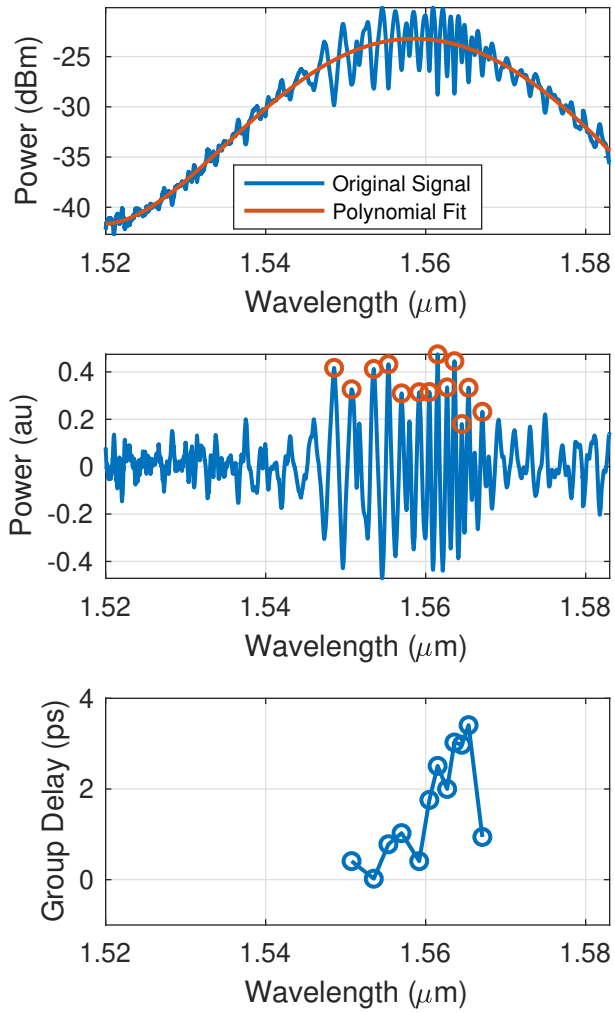

Figure 3.7: The group delay extraction process. First, the low frequency carrier is removed by using a fifth order polynomial fit (top). Next, the oscillation peaks are tracked and the FSR is estimated (middle). Finally, the group delay is calculated using the transfer function of the MZI (bottom).

3.3 Ring Resonator Modeling

To model the transmission responses of the racetrack resonators, we use a hybrid approach involving both machine learning and analytic models. Specifically, we use various multivariate regression models to relate design parameters like the waveguide thickness, width, or bend radius to the corresponding eigenmode effective indices. We modeled straight waveguides, bent waveguides, and evanescently coupled waveguides independently. We subsequently feed the output of these models into an analytic model commonly used to describe ring resonators. Figure 3.8 illustrates this process.

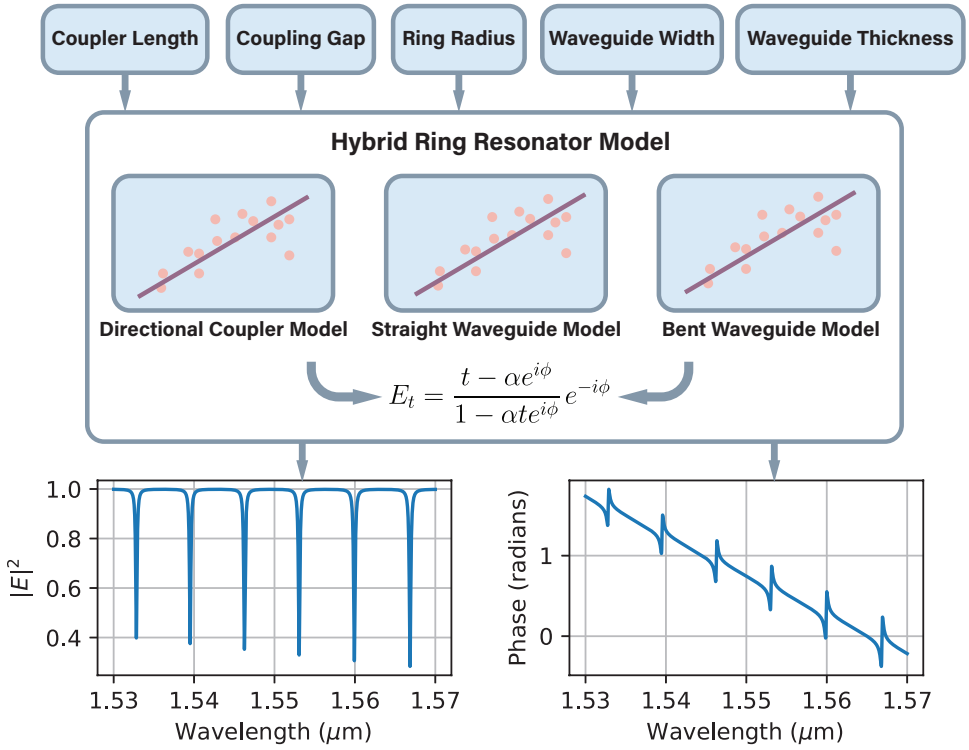

Figure 3.8: Hybrid model used to describe the racetrack ring resonator transmission response. First, individual ring components like bent waveguides, straight waveguides, and directional couplers are modeled using multivariate regression. The models relate common design parameters, like waveguide width, thickness, and coupler gap to the corresponding effective index of a particular mode. These effective indices are the fed into various analytic models that describe the field evolution for the racetrack resonator.

For this analysis, we tested over 2000 fabricated racetrack ring resonators with a nominal ring radius of 12 μm, coupler length of 4.5 μm, waveguide width of 0.5 μm, waveguide thickness of 0.22 μm, and separation gap in the directional coupler region of 0.2 μm. Figure 3.9 illustrates the designed geometry of the racetrack resonators.

The complex electric field propagation for the bent (b) and straight (s) waveguide sections of the resonator is modeled by

$$E_{b,s} = e^{i\beta_{b,s} l_{b,s}}, \tag{3.9}$$

where l is the length of the bent or straight section β is the wavenumber of that particular section described by

$$B = \frac{2\pi n_{eff}}{\lambda}, \tag{3.10}$$

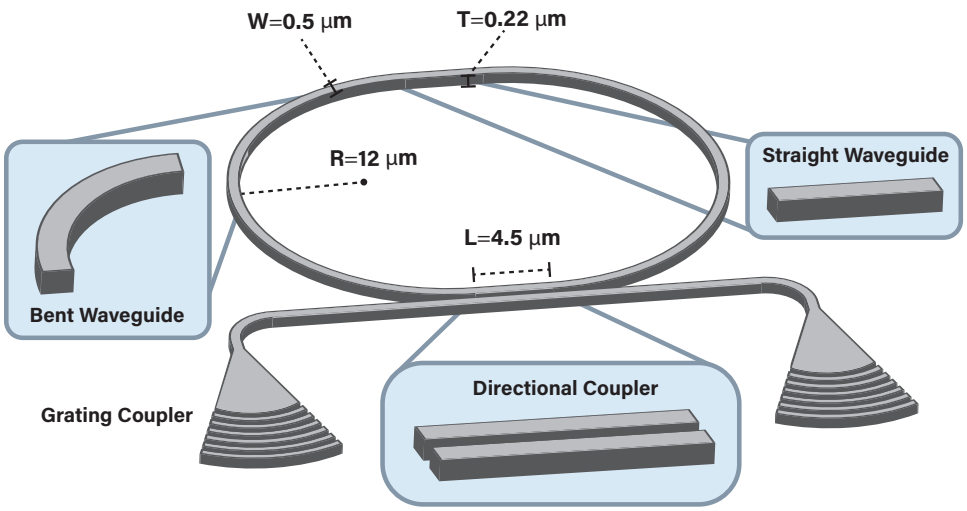

Figure 3.9: Hybrid model used to describe the racetrack ring resonator transmission response. The racetrack ring resonators rely on grating couplers to route light on and off the chip. The racetrack section of the ring is 4.5 μm and the bent sections have a radius of 12 μm. First, individual ring components like bent waveguides, straight waveguides, and directional couplers are modeled using multivariate regression. Next, the output of these models is fed into

where λ is the free-space operating wavelength and n_{eff} is the corresponding effective index pulled from the corresponding machine learning model.

We assume that the dominate source of coupling in the resonator is from the parallel waveguide section and that the bent waveguide sections are negligible in comparison. Consequently, we model the directional coupler as an ideal evanescent waveguide coupler with self-coupling coefficients (t) and cross-coupling coefficients (κ) described by

$$t = \frac{1}{2}(e^{i\beta_1 l_c} + e^{i\beta_2 l_c}) \tag{3.11}$$

$$\kappa = \frac{1}{2}(e^{i\beta_1 l_c} + e^{i\beta_2 l_c - i\pi}), \tag{3.12}$$

where l_c is the length of the directional coupler and β_1 and β_2 are the even and odd supermode wavenumbers for the coupler system. The effective indices for each of the supermodes are extracted from another machine learning model.

With the fields adequately described in each section of the racetrack resonator, the total accumulated phase shift, ϕ, around the ring can be calculated by

$$\begin{aligned}\phi &= \phi_b + \phi_s + \phi_c \\ &= \angle E_s + \angle E_b + \angle t \\ &= \beta_s l_s + \beta_b l_b + -i \log\left(\frac{(e^{i\beta_1 l_c} + e^{i\beta_2 l_c})}{|(e^{i\beta_1 l_c} + e^{i\beta_2 l_c})|}\right).\end{aligned} \tag{3.13}$$

Finally, the transfer function of the resonator is

$$E_t = \frac{t - \alpha e^{i\phi}}{1 - \alpha t e^{i\phi}} \alpha_c^2 e^{-i\phi}, \tag{3.14}$$

where α describes the loss of the resonator structure. From this transfer function, the transmission function is described by

$$T = \left(\frac{t^2 + \alpha^2 - 2\alpha t \cos\phi}{t^2 + \alpha^2 t^2 - 2\alpha t \cos\phi}\right). \tag{3.15}$$

3.4 Summary

In this chapter, we trained machine learning models for integrated strip waveguides, integrated chirped Bragg gratings, and microring resonators. The models show excellent agreement with the training set, other simulation methods, and even fabrication data. Hybrid models are also possible thanks to the flexibility of each machine learning model.

CHAPTER 4. CIRCUIT MODELING FOR CLASSICAL APPLICATIONS[1]

In this chapter, we apply the machine learning models developed in chapter 3 toward much more complicated designer tasks. We focus exclusively here on classical applications (e.g. filter design).

First, we demonstrate a graphical user interface (GUI) forward design tool used to design the ICBGs tested in chapter 3. This tool is the first of its kind and made possible thanks to the speed of the ICBG ANN. Next, we wrap the same ICBG ANN in an optimization algorithm to design a pulse compressor for communications applications.

Then, we use the hybrid ring resonator model to perform a deep parameter extraction on a coupled ring resonator filter to anticipate potential fabrication defects that may significantly alter performance. Finally, we perform a parameter extraction using the ANN and regression models of the waveguides, ICBGs, and ring resonators on fabricated devices.

4.1 Forward Design

The neural network's speed and flexibility enable forward design exploration. For example, Figure 4.1 illustrates a graphical user interface (GUI) built with slider bars to adjust the Bragg grating's design parameters (i.e. corrugation widths, grating length, chirp pattern, etc). The plots dynamically update, calling the neural network every time the user modifies the input, and display the corresponding reflection and group delay profiles. Because wavelength is included as an input to the ANN rather than an output, arbitrary wavelength sampling within the domain is allowed. Computing these responses in real time is not possible using traditional techniques. This capability is valuable and allows even novice designers the ability to rapidly gain device intuition without necessarily understanding the underlying numerical techniques.

[1] This chapter is a modified version of papers written by Alec M. Hammond, Easton Potokar, and Ryan M. Camacho [27, 28].

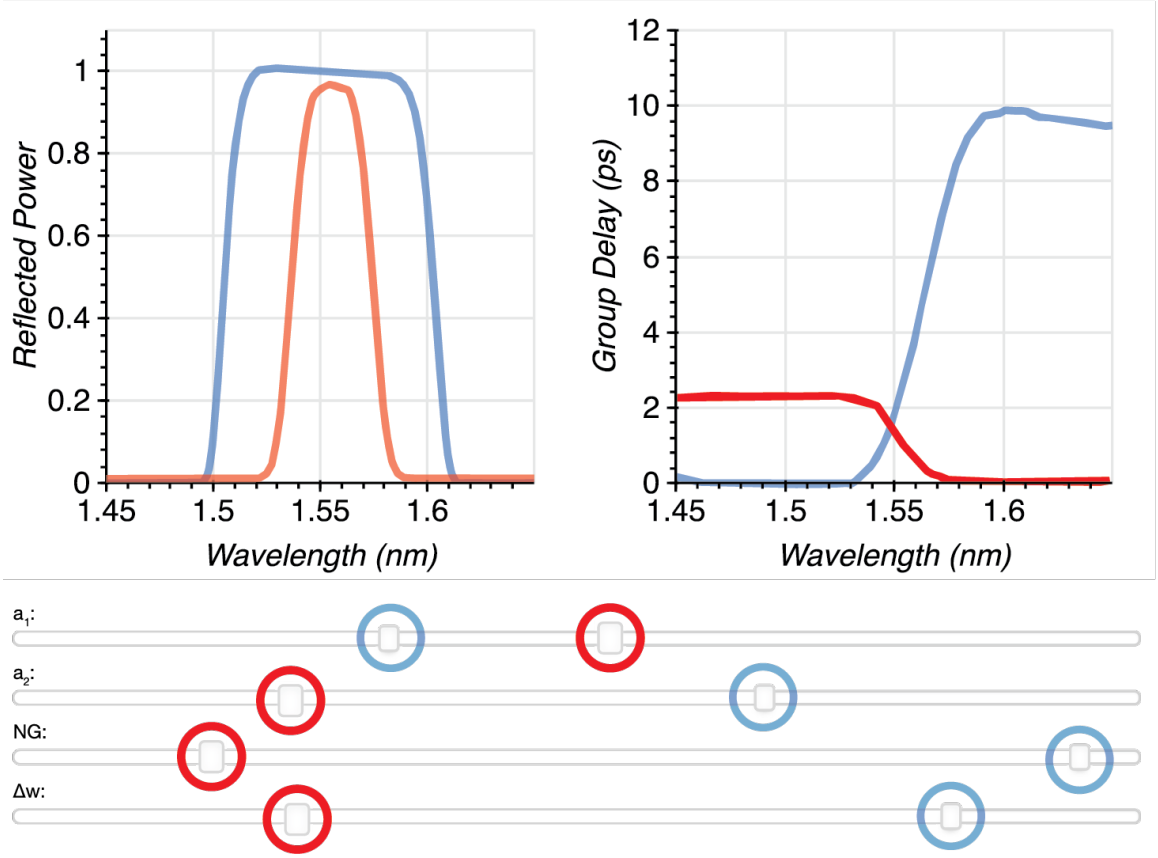

Figure 4.1: Graphical user interface used to explore the design space of a chirped Bragg grating. The slider bars on the left control physical parameters like grating length (NG), grating corrugation (dw), and the grating chirp (a1 and a2). Any time the user adjusts these parameters, the program calls the ANN and reproduces the expected reflection and group delay profiles for that particular grating. Due to the ANN's speed, the program is extremely responsive.

4.2 Inverse Design: Chirped Bragg Grating Pulse Compressor

This new approach also enables an entirely new set of inverse design problems. For example, we used the neural network in conjunction with a truncated Newtonian optimization algorithm to design a temporal pulse compressor. Designers often rely on dispersive Bragg gratings to generate short, optical pulses for high-capacity communications [38]. In this particular case study, we assumed an arbitrary source generates a 20 ps wide chirped pulse with 4 nm of bandwidth. Figure 4.2 illustrates the optimization routine's evolution, the resulting grating response, and the pulse shape before and after the Bragg grating. Such optimization algorithms run much quicker than previously known methods, owing to the accelerated cost function. The agnostic nature of the neural network interface works well with typical optimization routines, especially since any arbi-

trary wavelength sampling is allowed. Depending on the cost function formulation, gradient-based methods could directly evaluate the Jacobian and Hessian tensors from the ANN without any extra sampling or discretization.

4.3 Parameter Extraction

Silicon photonic devices typically contain features with sub-micron dimensions, owing to the platform's high index contrast and years of complementary metal-oxide-semiconductor (CMOS) process refinement.

While small features enable several innovative and scalable designs, they also induce an increased sensitivity to fabrication defects [10]. A fabrication defect of just one nanometer, for example, can cause a nanometer shift in the output spectrum of the silicon photonic device [39]. Understanding and characterizing these process defects is essential for device modeling and variability analysis [40, 41]. Predicting and compensating for such sensitivity in the optical domain is difficult because typical simulation routines are computationally expensive and in many cases, prohibitive.

To overcome these challenges, we demonstrate a new parameter extraction method using artificial neural networks (ANN). We train an ANN to model the complex relationships between integrated chirped Bragg gratings (ICBG) [42, 43] and their corresponding reflection and group delay profiles. We use the trained ANN to extract the physical parameters of various fabricated ICBGs using a nonlinear least squares fitting algorithm — a task that is computationally prohibitive using traditional simulation routines. We find that the proposed routine produces spectra that matches well the experimental reflection and group delay profiles for the ICBGs.

Our work builds upon previous efforts that extract integrated photonic device parameters using analytic models. Chrostowski *et al*., for example, extract the group index across a wafer with 371 identical microring resonators (MRR) using an analytic formula describing the free spectral range (FSR) [44]. Similarly, Chen *et al*. derive both the effective and group indices from MRRs by fitting the full, analytic spectral transfer function to the experimental data [45]. Melati *et al*. extract phase and group index information from small lumped reflectors knows as point reflector optical waveguides (PROW) using various analytic formulas [46].

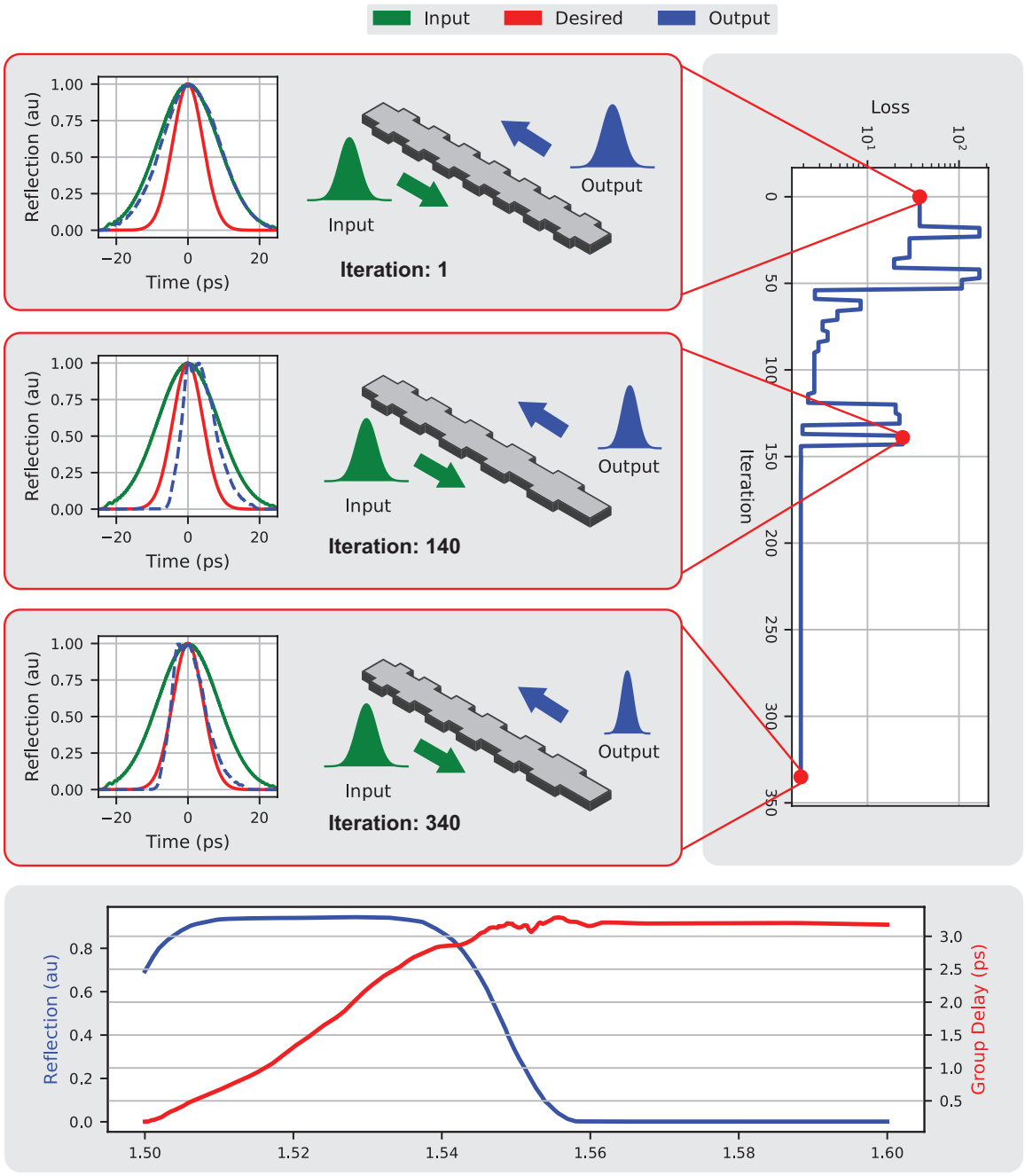

Figure 4.2: ANN-assisted design of a monolithic temporal pulse compressor using an integrated photonic chirped Bragg grating. A truncated Newton algorithm was tasked with constructing a grating that compressed an arbitrary chirped pulse by a factor of 2. After 340 grating simulations, the optimizer sufficiently minimized a cost function (right) that compared the new pulse's width to the old pulse. The resulting grating is demonstrated below and the input, output, and desired pulses for iterations 1, 140, and 340 are demonstrated on the left.

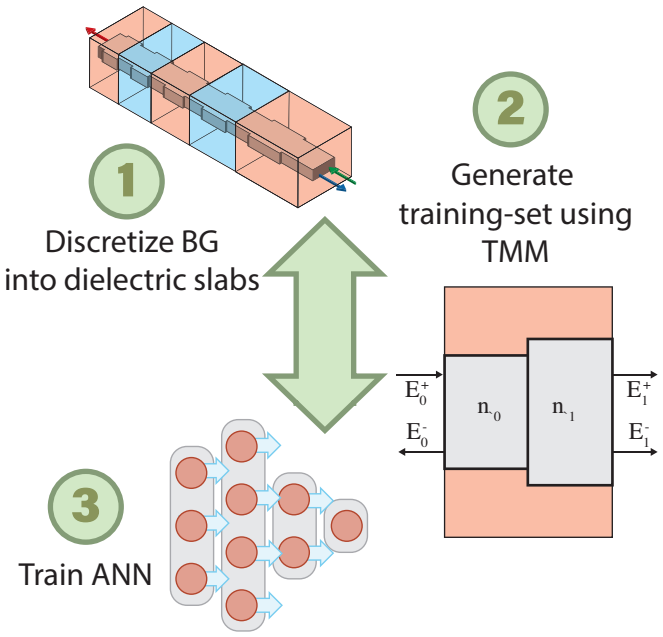

Figure 4.3: ANN modeling process. First, several different ICBGs are discretized into individual dielectric layers (1). Then, the reflection and group delay profiles are simulated using the transfer matrix method (2). This dataset is then fed into a ANN training algorithm (3). Often, this process must be repeated until the ANN can suitably express a large enough ICBG design space.

Perhaps most similar to this work, Xing et al. build a regression model from data generated by an eigenmode solver that relates waveguide design parameters (e.g. width and thickness) to their corresponding effective indices [47]. They subsequently use this regression model in addition to an analytic transfer matrix and a fitting routine to extract the average waveguide width and thickness from various Mach-Zhender Interferometer (MZI) devices. Our ANN parameter extraction method is fast, just like the analytic and regression models, but capable of modeling much more complicated devices, like ICBGs.

The rest of the paper is outlined as follows: first, we describe the data generation process necessary to train the ANN. Next, we describe the process of training the ANN. We then discuss the ICBG device design, fabrication, testing, and data calibration. Finally, we describe our nonlinear fitting algorithm and present our experimental results.

To estimate the actual fabrication parameters of the ICBGs, we used the ANN in conjunction with a nonlinear least squares fitting routine within the SciPy package [48]. The routine initializes by calling the ANN using the original design parameters. The simulated reflection and group delay profiles are directly compared to the measurement data. From the residuals, the al-

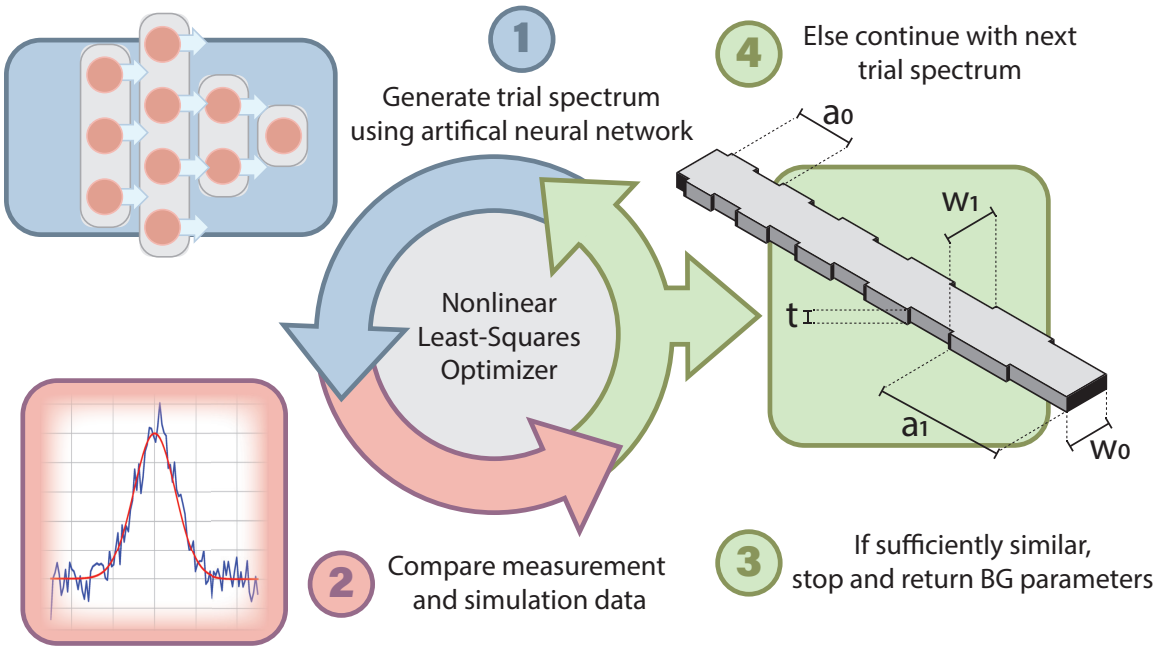

Figure 4.4: Efficient and robust method to extract fabricated ICBG device parameters using ANNs and a nonlinear least-squares optimizer. First, the ANN simulates reflection and group delay spectra for the device's initial design parameters (1). Then, the simulations are compared directly to the measured data (2). If the results are sufficiently similar, the optimizer returns the device parameters (3). If not, the optimizer strategically simulates a new set of device parameters based on the residual error (4).

gorithm decides whether the current design is sufficiently similar to the measurement data or if further simulation is needed. Figure 4.4 illustrates this procedure.

Since the ICBG has fabrication limits that can be cast as parameter bounds, we chose to run a Trust Region Reflective (TRF) optimization algorithm within the nonlinear solver [49]. Specifically, we bounded the first and last ICBG periods (a_0 & a_1) between 312 nm and 328 nm and the corrugation width between 1 nm and 50 nm. The number of periods was fixed.

We chose to extract parameters of three different ICBGs. After just 5 minutes of optimization on a MacBook Air 2012 (1.8 GHz Intel Core i5, 4 GB 1600 MHz DDR3 RAM), the solver converged on new parameters for all three devices that more reasonably reflect the measurement data. Figure 4.5 illustrates the algorithm's results compared to the fabrication data and the original design spectra. Table 4.1

Not only do the algorithm's profiles match the data much better, but the extracted parameter differences are expected from the processes used to fabricate the devices. For example, the

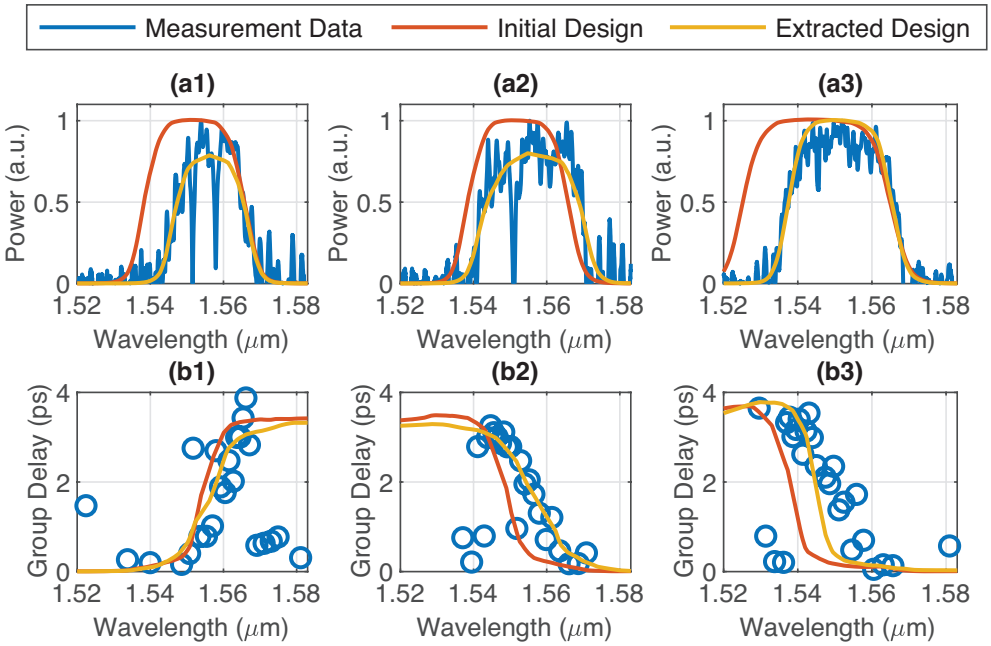

Figure 4.5: The extracted reflection (a1, a2, a3) and group delay (b1, b2, b3) profiles (yellow) compared to the initial design profiles (red) and the calibrated measurement data (blue).

Table 4.1: Parameter extraction results for three separate devices. The design parameters are compared directly to the algorithm's extracted parameters for each device..

	Device 1		Device 2		Device 3	
Design Parameters	Design	Extracted	Design	Extracted	Design	Extracted
a_0 (nm)	324	324.3	318	315.8	318	320.1
a_1 (nm)	318	317.0	324	325.6	324	323.7
NG	750	750	750	750	750	750
Δw (nm)	30	13.7	30	15.7	50	39.3

algorithm predicts a slightly wider chirping bandwidth and smaller corrugation width for all three devices. The E-beam raster grid's resolution approaches the chirping resolution of the ICBG (1 nm), so "snapping" from one grid point to the next results in slightly wider chirping bandwidths. The E-beam's resolution, along with the etch process, also tend to round the sharp ICBG corners, resulting in lower net corrugation width.

Other small differences between the extracted parameter sets and the fabricated data, like the Fabry-Perot resonances, are difficult to model with the current ANN abstraction. It would require a much more sophisticated, and possibly impractical, parameterization to capture these

defects. Despite these small discrepancies, the fitting algorithm and ANN demonstrate a strong ability to extract parameters for complex silicon photonic devices.

4.4 Summary

In this chapter, we reviewed several novel applications that leverage machine learning nanophotonics device models. These applications were previously unavailable to the user (or computationally impractical.) Fortunately, these models can continue to persist after training and accelerate many other classical design flows.

CHAPTER 5. CIRCUIT MODELING FOR QUANTUM APPLICATIONS[1]

Recent developments in structured optical receivers demonstrate potential to exceed classical information transmission limits in photon-starved environments [50]. Rather than reading codewords symbol by symbol, these networks perform joint-detection over multiple symbols simultaneously. Consequently, the Shannon capacity of this device approaches the Holevo limit — an upper bound for communication transceivers relying on quantum state alphabets.

One particular architecture, known as the "Green Machine" (GM) in the literature, matches the topology of the fast Fourier transform (FFT) by mapping the inputs and outputs of various beamsplitter stages in a butterfly fashion. It performs a Hadamard transform on a binary phase-shift keyed (BPSK) codebook to generate a pulse-position modulation (PPM) based codebook. So long as we choose a BPSK codebook with columns of a Hadamard matrix as its codewords, the GM architecture will efficiently route each codeword to a single output channel, enabling deep space applications with strict signal power constraints.

Leveraging years of process refinement within the semiconductor industry allows silicon photonics to provide an ideal platform for fabricating the GM [9]. These small, solid-state systems can transition from prototype to large-scale, cost-effective fabrication almost instantly, paving the way for rapid innovation. Constructing this receiver is difficult, however, because preserving the Hadamard transform requires phase matching across the entire network.

In this chapter, we propose and demonstrate a solution to this challenge that allows for complete phase compensation *after* fabrication and applies to both integrated and bulk optical systems. It is known that in the case of quantum-limited detection, the GM architecture is capable of distinguishing between a maximally orthogonal set of phase-encoded codewords with near ideal performance. Here we demonstrate that the maximal orthogonality of the codewords is robust to internal phase errors so long as a new codebook is defined that compensates for the internal

[1] This chapter is a modified version of a paper written by Alec M. Hammond, Ian W. Frank, and Ryan M. Camacho [20]

phase errors. In other words, the GM does not require a strict Hadamard transform for maximum codeword distinction. This is the key observation of this work, and transfers the experimental task to learning the new codewords rather than phase-matching. We present a feedback algorithm that learns the codewords of an arbitrary GM and demonstrate the algorithm's success on a free-space setup.

We generalize the GM transformation requirements by relating them to Optical Butler Matrices (OBM). Butler matrices are a class of passive beamforming networks that take a single beam as an input and coherently generate multiple output beams with a well-defined phase relationship [51]. First invented over 50 years ago, they are now commonly used at radio frequencies (RF) for applications in beam steering, astronomy, satellite communications, and remote sensing [52]. These devices are a subclass of the GM architecture, where "phase errors" are strategically placed to induce the proper "codewords", or phase profiles.

The outline of the rest of the paper is as follows: In section 2 we give a theoretical overview of the Green Machine and fast Fourier transform. In Section 3 we present our feedback algorithm to find new codewords for a non phase-matched Green Machine and in Section 4 we present our experimental results using the algorithm in a 4×4 system. Section 5 then ends with concluding remarks and an outlook for future work.

5.1 Green Machine and Fast Fourier Transform

By examining the Green Machine's corresponding transformation matrix and how it matches the FFT topology, we show that any arbitrary device maintains an orthogonal book of codewords. Due to fabrication constraints, we first analyze a GM built with symmetric beamsplitters (i.e. directional couplers). We then extend this analysis to the original Hadamard GM and to Butler Matrices. In the process, we show that any arbitrary device composed of symmetric beamsplitters, regardless of the induced phase errors between stages, has a unique codebook.

The original FFT as proposed by Cooley and Tukey is a "divide and conquer" algorithm that recursively simplifies the discrete Fourier transform into more computationally attractive operations. The simplest method, known as radix-2, forms 2-input, 2-output "butterflies" as the base of its recursive routine. These butterflies mix the inputs in a fashion similar to that of a beamsplitter. Each butterfly also applies various twiddle factors (phase shifts) to the inputs and/or outputs to

shape the desired Fourier transform response. These twiddle factors are analogous to the "phase errors" that are present in GM devices after fabrication.

Each butterfly can be modeled by a transformation matrix. To find the transformation matrix and butterfly representation of a GM using ideal directional couplers and no phase errors, we recursively cascade matrices that characterize its various components. For example, each two-port directional coupler stage is modeled by

$$A_2 = \begin{bmatrix} t & ir \\ ir & t \end{bmatrix}, \tag{5.1}$$

where i is $\sqrt{-1}$, t is the Fresnel transmission coefficient, and r is the Fresnel reflection coefficient. We model larger networks by cascading a circulant matrix, C_n, with a combined beamsplitter stage:

$$A_n = C_n(A_{\frac{n}{2}} \bigotimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}), \tag{5.2}$$

so that an ideal 4 port device where $t = r = \frac{\sqrt{2}}{2}$ has the following transformation matrix:

$$\begin{aligned} A_4 &= \frac{1}{2} \begin{bmatrix} 1 & 0 & i & 0 \\ 0 & i & 0 & 1 \\ i & 0 & 1 & 0 \\ 0 & 1 & 0 & i \end{bmatrix} \begin{bmatrix} 1 & i & 0 & 0 \\ i & 1 & 0 & 1 \\ 0 & 0 & 1 & i \\ 0 & 0 & i & 1 \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} 1 & i & i & -1 \\ i & 1 & -1 & i \\ i & -1 & 1 & i \\ -1 & i & i & 1 \end{bmatrix}. \end{aligned} \tag{5.3}$$

A codeword is the list n phases of the coherent inputs to the n channels of the GM that result in all of the beams combined into a single output. For an n-channel GM there are n codewords, each one corresponding to combining the beam to a unique output. We extract a device's codebook, H

mathematically by inverting its corresponding transfer matrix:

$$
\begin{aligned}
H_4 &= \left\{ \frac{1}{2} \begin{bmatrix} 1 & i & i & -1 \\ i & 1 & -1 & i \\ i & -1 & 1 & i \\ -1 & i & i & 1 \end{bmatrix} \right\}^{-1} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \\
&= \begin{bmatrix} 1 & -i & -i & -1 \\ -i & 1 & -1 & -i \\ -i & -1 & 1 & -i \\ -1 & -i & -i & 1 \end{bmatrix}.
\end{aligned}
\tag{5.4}
$$

The cascaded diagonal matrix corresponds to the expected output field magnitude for each codeword. The codewords are then the columns of H. In this case, all of the codewords are mutually orthogonal (since the codebook and transformation matrix are unitary) and symbols are separated by $\frac{\pi}{2}$ in phase space.

We can follow a similar procedure to formulate the transformation matrix and corresponding codebook of a Hadamard GM. The device is composed of asymmetric beamsplitter stages, which are each modeled by

$$
A_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{5.5}
$$

Upon closer examination, we realize we can build an asymmetric beamsplitter by simply applying phase shifts to the inputs and outputs of the ideal symmetric beamsplitter:

$$
\frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} -i & 0 \\ 0 & -1 \end{bmatrix} \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} \begin{bmatrix} i & 0 \\ 0 & 1 \end{bmatrix}. \tag{5.6}
$$

Similarly, we model a four-port Hadamard GM by cascading appropriate phasor matrices (P) in between stages:

$$
A_n = C_n P_n (A_{\frac{n}{2}} \bigotimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}), \tag{5.7}
$$

so that resulting transformation matrix is simply a Hadamard matrix:

$$A_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{5.8}$$

We determine the corresponding codebook to be

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \tag{5.9}$$

As we expect, all codewords are mutually orthogonal and each symbol is separated by π in phase space.

We model a Butler Matrix the same way we modeled our ideal and Hadamard Green Machines. We note that the fundamental stage also relies on an asymmetric beamsplitter, but the induced phase shifts in between stages are more involved [53]. The final transformation matrix and codebook are given by

$$\frac{\sqrt{2}}{2}\begin{bmatrix} 1 & i \\ 1 & -i \end{bmatrix} = \begin{bmatrix} -i & 1 \\ 0 & -1 \end{bmatrix} \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix} \begin{bmatrix} i & 0 \\ 0 & i \end{bmatrix} \tag{5.10}$$

and

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & 1 \\ 1 & -i & -1 & i \end{bmatrix}. \tag{5.11}$$

Here we see that codewords continue to be mutually orthogonal and each symbol is separated by $\frac{\pi}{2}$ in phase space.

After exploring all three cases, we realize that the "phase errors" within the device simply rotate the codeword symbols in phase space, but each codeword retains maximum orthogonality. For a more rigorous analysis, we inject arbitrary phase errors between two stages of a 4 port device in Equation 5.12 and solve for its accompanying codebook:

$$\begin{aligned} A_4 &= C_4 P_4 \begin{bmatrix} A_2 & 0 \\ 0 & A_2 \end{bmatrix} \\ &= \begin{bmatrix} t & 0 & ir & 0 \\ 0 & ir & 0 & t \\ ir & 0 & t & 0 \\ 0 & t & 0 & ir \end{bmatrix} \begin{bmatrix} e^{i\phi_1} & 0 & 0 & 0 \\ 0 & e^{i\phi_2} & 0 & 0 \\ 0 & 0 & e^{i\phi_3} & 0 \\ 0 & 0 & 0 & e^{i\phi_4} \end{bmatrix} \begin{bmatrix} t & ir & 0 & 0 \\ ir & t & 0 & t \\ 0 & 0 & t & ir \\ 0 & 0 & ir & t \end{bmatrix} \end{aligned} \tag{5.12}$$

$$\begin{aligned} H &= A_4^{-1} X_0 \\ &= \frac{1}{2} \begin{bmatrix} e^{i\theta_1} & -ie^{i\theta_1} & -ie^{i\theta_2} & -e^{i\theta_2} \\ -ie^{i\theta_1} & -e^{i\theta_1} & e^{i\theta_2} & -ie^{i\theta_2} \\ -ie^{i\theta_3} & e^{i\theta_3} & -e^{i\theta_4} & -ie^{i\theta_4} \\ -e^{i\theta_3} & -ie^{i\theta_3} & -ie^{i\theta_4} & e^{i\theta_4} \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \\ &= \begin{bmatrix} e^{i\theta_1} & -ie^{i\theta_1} & -ie^{i\theta_2} & -e^{i\theta_2} \\ -ie^{i\theta_1} & -e^{i\theta_1} & e^{i\theta_2} & -ie^{i\theta_2} \\ -ie^{i\theta_3} & e^{i\theta_3} & -e^{i\theta_4} & -ie^{i\theta_4} \\ -e^{i\theta_3} & -ie^{i\theta_3} & -ie^{i\theta_4} & e^{i\theta_4} \end{bmatrix}. \end{aligned} \tag{5.13}$$

We note that even in the presence of arbitrary phase errors, there still exists a set of codewords that lead to maximally orthogonal outputs. One might suspect that the new codewords have become less distinguishable from those without errors. However, we note that each codeword can always be grouped such that the top and bottom halves of the old codewords get rotated in phase space while maintaining a constant distance in phase space. In other words, thanks to the nature of the GM's beamsplitters, phase errors simply shift the device's transfer function from one stable

state to another stable state, guaranteeing a new unique set of codewords. Figure 5.1 illustrates the equivalent butterfly circuits for each of the above cases, along with the corresponding beamsplitter topology and codeword after exciting the third input port.

We generalize this conclusion to any size Green Machine by examining its transformation matrix composition. Since each component of the Green Machine, including the phase errors, is modeled by a unitary matrix, the subsequent transformation matrix must also be unitary. The inverse of that transformation matrix (i.e. the codebook) must have orthonormal columns, indicating that each codeword is indeed maximally orthogonal.

As an added benefit, we can also model amplitude errors between the stages and find pairings in the elements comprising the codewords such that the losses can also be compensated. Imperfect splitting ratios, for example, are modeled with the t and r coefficients and simply scale the codewords away from unit amplitude. The resulting transformation matrix is composed of orthogonal columns, but is no longer unitary. While it is possible to correct for such errors by adjusting the amplitudes of the inputs, this would not improve the system performance. Using a quantum-limited phase-sensitive optical amplifier, for example, would require a knowledge of the phase information in the signal before detection, defeating the very purpose of the receiver. Thus amplitude errors, unlike phase errors, result in decreased mutual information and are therefore detrimental to applications for quantum receivers.

5.2 Feedback Algorithm

Experimentally, we would like an algorithm that allows us to inject light into the input ports and find the codewords only by measuring the intensity, but not the phase, at each output. We modulate the phase angle of each input, but not the amplitude, which corresponds to a vector $\mathbf{x}$ of complex values. These inputs are then modified by the GM, modeled by the transformation matrix A. The photodiodes measure the optical power at each output. With these parameters in mind, we designed a feedback loop using a convex objective function and optimization routine.

Before we present our convex optimization algorithm, we first present various other phase-learning algorithms to help establish some intuition and perspective on the nature of the system. Consider first an algorithm which leverages the recursive structure of the GM and is shown graphically in Figure 5.2. By systematically adjusting the input phases so that light at each beamsplitter

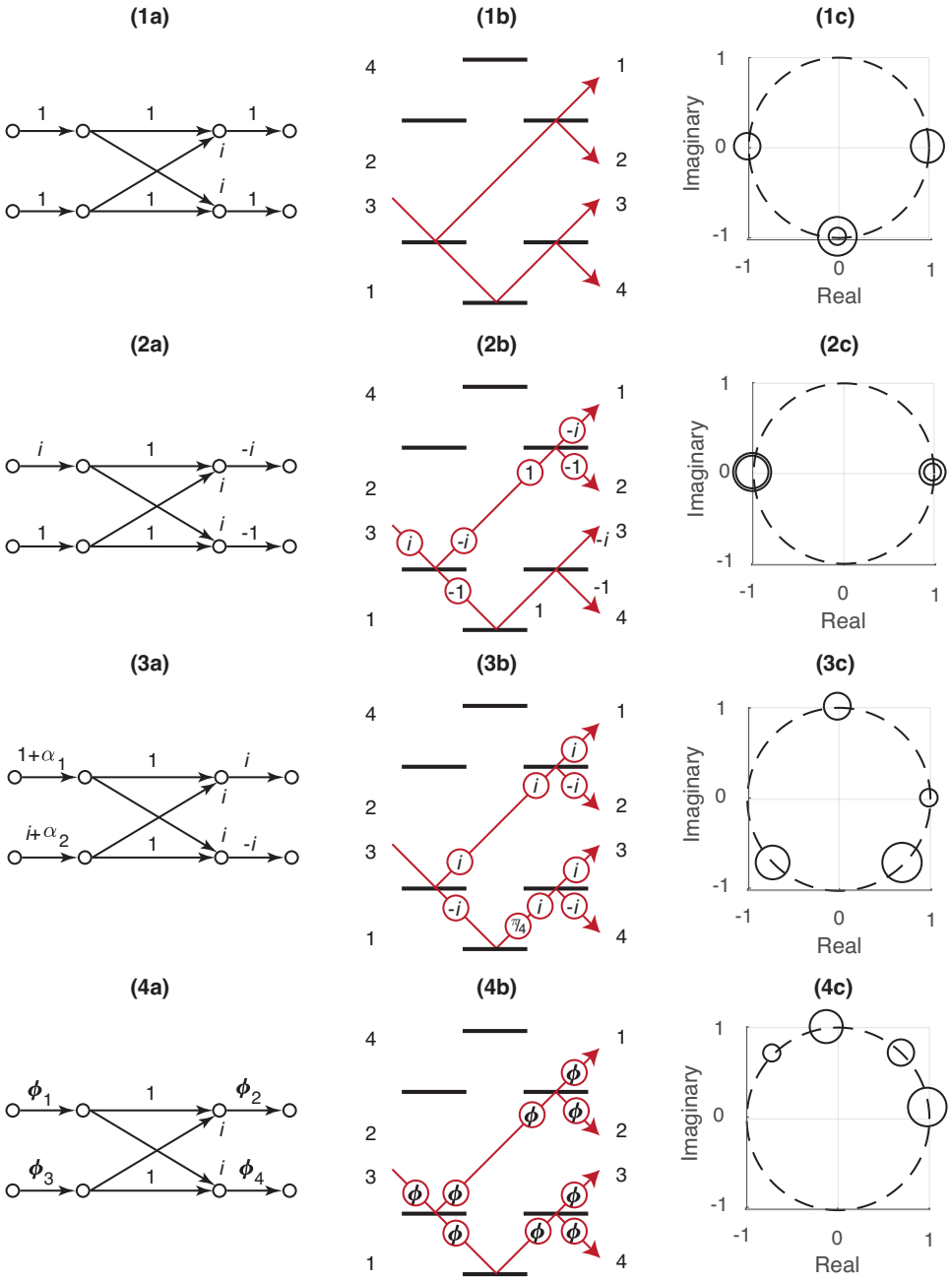

Figure 5.1: A comparison between the equivalent FFT butterfly circuit using directional couplers (column 1), the corresponding GM architecture with effective phase shifts (column 2), and the resulting codewords (column 3) for the ideal GM (row a), the Hadamard GM (row b), the Butler Matrix (row c), and a GM with arbitrary phase errors (row d). The symbol spread for each codeword depends on the embedded phase shifts/errors. For example, the Hadamard GM contains phase errors where all of the symbols are π apart, forming BPSK codewords. All other schemes rotate the codewords, but manage to maintain at least $\frac{\pi}{2}$ distance between critical stages.

exits through a single output, we can can find the input phases for a given codeword. Repeating this process for each port produces the codebook for the GM. While intuitive, this algorithm scales poorly with size and is difficult to implement in hardware. In addition, this algorithm requires prior knowledge of the channel mappings and internal binary combinations within the GM, which can be problematic since several equivalent choices are possible. We therefore seek an algorithm that overcomes these limitations.

A stochastic excitement technique — where each channel is randomly modulated and the statistics of the output are used to solve for the elements of the corresponding A-matrix using an analytically derived system of equations — would be ideal except for the nonlinearities induced by the higher order statistics. Such nonlinearities reduce the system of equations to a non-convex optimization problem that scales poorly with size. When simulating this approach, we found the algorithm often did not converge, but became trapped in a local minimum. Likewise, a Maximum Likelihood Estimator (MLE) also reduced to a non-convex optimization problem. Even under ideal conditions, we determined that both these statistics-based approach are not sufficiently robust.

Fortunately, we can derive a convex objective function, which encourages a brute force optimization approach. We identify the first codeword of a 4×4 Green Machine, for example, by iterating through various phase profiles that maximize the intensity of output port 1 and minimize the intensity of ports 2, 3, and 4. The output $\mathbf{y}$ for a given input $\mathbf{x}$ is given by

$$A\mathbf{x} = \mathbf{y}, \tag{5.14}$$

where $\mathbf{x}$ and $\mathbf{y}$ are complex. The total intensity measured at all the photodetectors is subsequently modeled as an induced inner product between the output phase profile:

$$I = \mathbf{x}^H A^H A\mathbf{x}. \tag{5.15}$$

The intensity measured at a single photodiode can be modeled with a weighting matrix:

$$I_1 = \mathbf{x}^H A^H W_1 A\mathbf{x}. \tag{5.16}$$

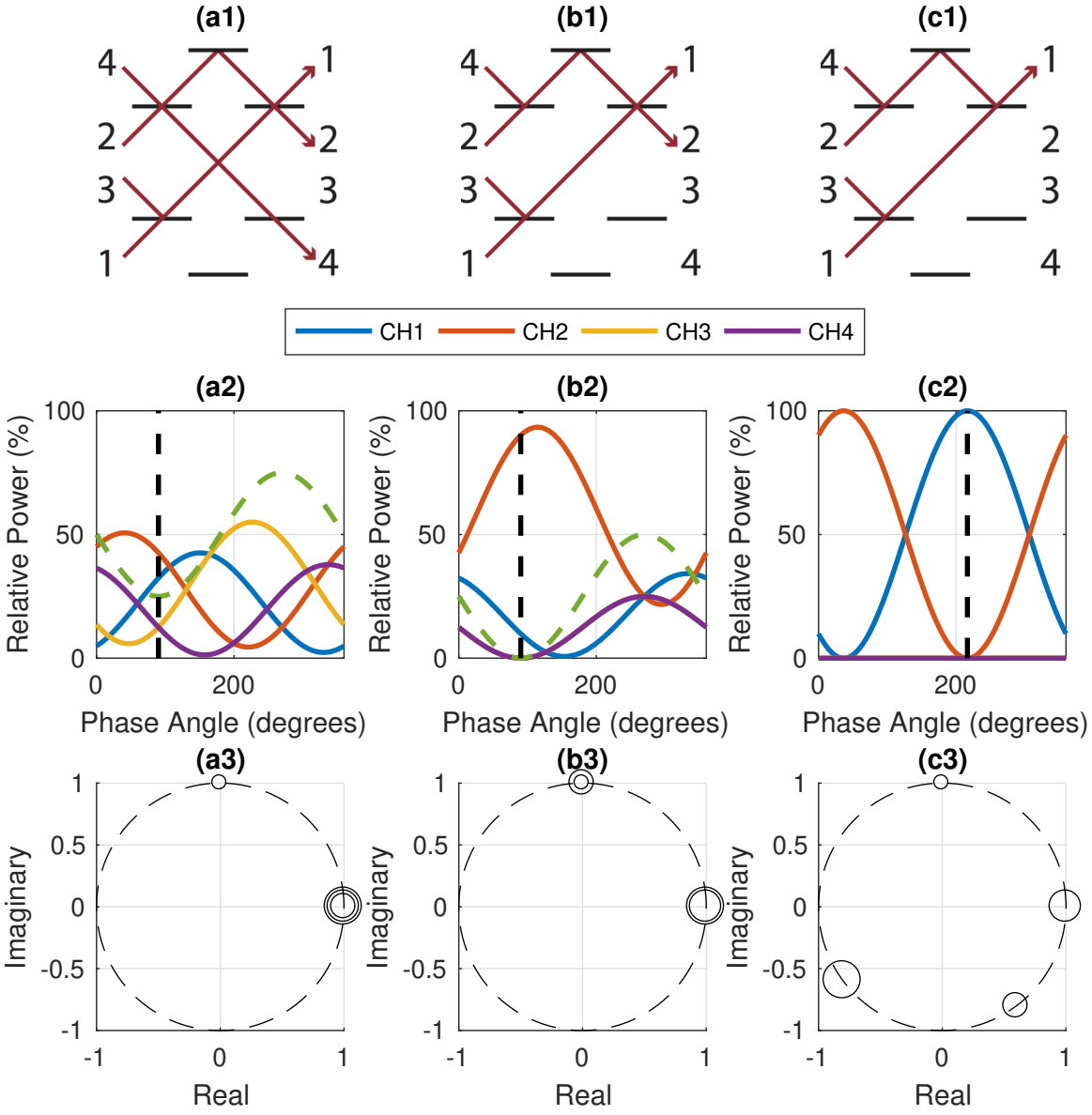

Figure 5.2: A systematic algorithm for determining the first codeword for an arbitrary 4×4 GM (i.e. all the light exits port 1). Step 1 is to interfere channels 1 and 3 such that transmission toward ports 3 and 4 is eliminated (a1). This is done by phase modulating channel 1 from 0 to 2π (a2) until the combined light of ports 3 and 4 is minimized (shown by dashed green line). The phase that minimizes this quantity is saved and applied to the codeword (a3). The next step is to interfere channels 2 and 4 such that, once again, the transmission towards ports 3 and 4 is minimized (b1). This is similarly done by phase modulating channel 2 until the combined light in ports 3 and 4 is minimized. The corresponding phase is then applied to the codeword (b3). At this point, light only exits ports 1 and 2. We solve for the final codeword by sweeping channels 1 and 3 in concert (to preserve the previously determined necessary phase relationship for interference) until all the light exits port 1. (c1,c2). The resulting phase profile describes the first codeword(c3).

If we were interested in channel 1, for example, we would insert the vector $[1,0,0,0]$ along the diagonal of W. When optimizing for the second codeword, the second element along the diagonal of the weighting matrix is 1. So long as we restrict our search space to the complex unit sphere (i.e. we don't modulate the amplitude) the transformation is convex [54]. The optimization problem for the first codeword is described by:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \quad J_1(\mathbf{x}) \\ &\text{subject to} \quad |\mathbf{x}| = 1. \end{aligned} \tag{5.17}$$

The cost function for the first codeword is then defined as the weighted inner product

$$J_1(\mathbf{x}) = -\mathbf{x}^H A^H W_1 A \mathbf{x} \tag{5.18}$$

and the negative sign is necessary to transform the maximization criteria into a minimization problem.

Figure 5.3 illustrates the error space for an GM with random phase errors. Since the phase profiles are all relative to each other, there are infinitely many local minima, each of which are valid solutions. Theoretically, the control algorithm can begin descending wherever it initializes and will always converge to a valid solution. Equipment constraints, however, along with environmental noise will limit the phase angle range over which the algorithm can iterate.

5.3 Experiment and Results

To test the control algorithm, we implemented a free-space, four port Green Machine using beamsplitters and mirrors arranged in a Michelson-like interferometric configuration. While the algorithm is intended for eventual use in integrated GM's, our free space implementation allows for a more complete analysis and debugging, since each path can be directly and independently manipulated.

Figures 5.4 and 5.5 illustrate the experimental setup. A 1550 nm (C band) laser feeds light into a circulator, and the light then exits via a fiber launch (FL) and enters the first beamsplitter (BS1). After passing through the Green Machine, four distinct beams eventually retroflect from

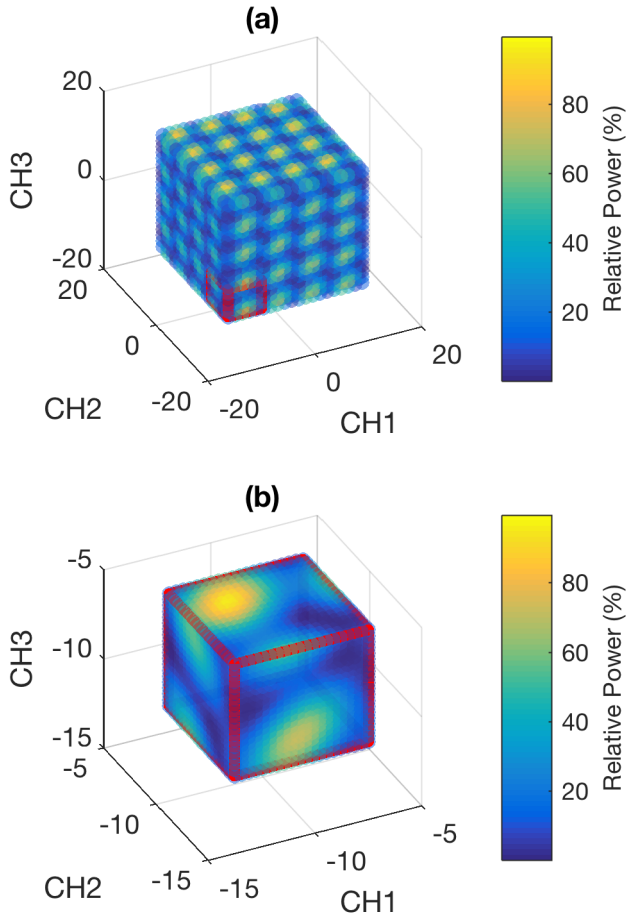

Figure 5.3: (a) The error space for an arbitrary GM with induced phase errors. In this simulation, we phase modulate each channel from -4π to 4π. Each "cell" with length of 2π contains a single valid solution. (b) A single cell where each channel is phase modulated from -4π to -2π.

their corresponding mirrors (M1, M2, etc.), which are piezo driven and used to modulate the phase of each channel. Consequently, we consider this the input of the Green Machine, with the initial pass through the GM simply functioning as a convenient method for splitting light into four channels and providing alignment beams for back-reflection. The modulated beams return through the network, interfering as expected. Photodetectors are placed at the outputs to measure the resulting interference pattern. Channel 1 returns through the fiber launch and is rerouted through the circulator to another photodetector.

We used non-polarizing beamsplitter cubes manufactured by Lambda Research Optics (Costa Mesa, CA) and anti-reflection coated for operation at 1550 nm. Each cube was speci-

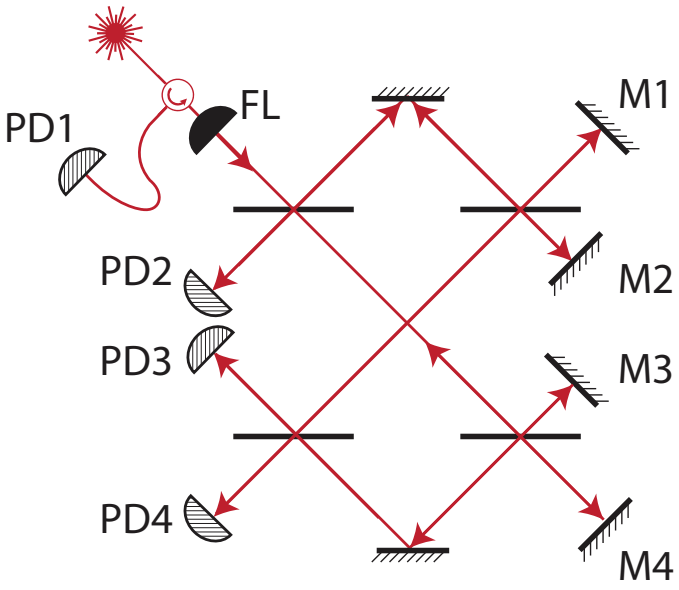

Figure 5.4: Diagram of the optical paths from the fiber launch (FL) to photodiodes (PD's) in experimental 4×4 GM.

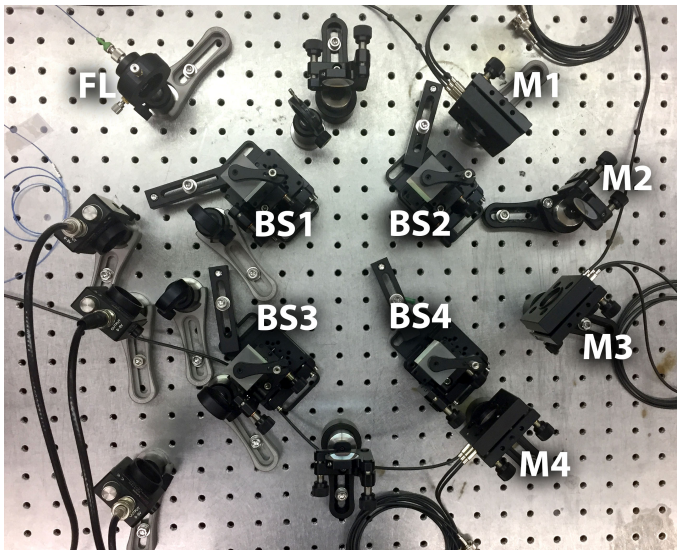

Figure 5.5: Top view of the experimental setup showing fiber launch (FL), photodiodes (PD's) and piezo mirrors (M's).

fied by the manufacturer with a T and R value of $50\% \pm 3\%$ [55]. After alignment, we achieved a fringe visibility between 95% and 99% for all $\binom{4}{2} = 6$ binary interferometric combinations.

We first attempted solving our system with an interior point algorithm [56]. Small amounts of hysteresis in the phase modulation and temperature gradients across the experimental apparatus induce phase noise, which degraded the algorithm's performance. By extension, any algorithm relying on numerical gradient approximations will naturally be sub-optimal as well since the error space is constantly changing.

To combat the stochastic nature of the device we implemented a constrained, globalized, and bounded Nelder-Mead method (GBNM) [57]. The traditional Nelder-Mead Simplex algorithm is gradient-free and tends to perform better than other descent algorithms over nonsmooth cost functions [58]. The GBNM variation allows for constraints and restarts if the algorithm spends too many iterations around a particular point. This restart feature is key for overcoming noise-induced local minima.

Figure 5.6 details the evolution of the GBNM algorithm for each channel. Optimal values for channels 1, 2, 3, and 4 result in 93.7%, 94.7%, 95.4% and 96.0% relative intensity respectively. The restart nature of the GBNM algorithm is evident by the repeated jumps from high to low channel intensity. In this particular configuration, 12 points were randomly chosen at initialization. If the algorithm didn't converge after 100 iterations, it moved onto the next starting point.

Given the imperfect beamplitting ratios and fringe visibility, error analysis indicates that deeper convergence (i.e channel intensities of 100%) requires beamsplitters with tolerances closer to the ideal 50% splitting ratio to achieve better fringe visibility. Increasing the size of the system will only exaggerate the effect of these errors, pushing the codewords further from unity and decreasing the maximum achievable transmission. Despite these imperfections, the method appears to accurately characterize the codebook of the device.

Apart from the short-term noise sources that bias the system's cost function, we observed a much slower system drift on the order of minutes to hours. Factors like temperature drift, air currents, and building vibrations all transformed the system's codebook, requiring a renewed characterization. Despite these system fluctuations, the algorithm runs quick enough to accurately learn the new codebook. We anticipate that the system performance will further improve on chip, where

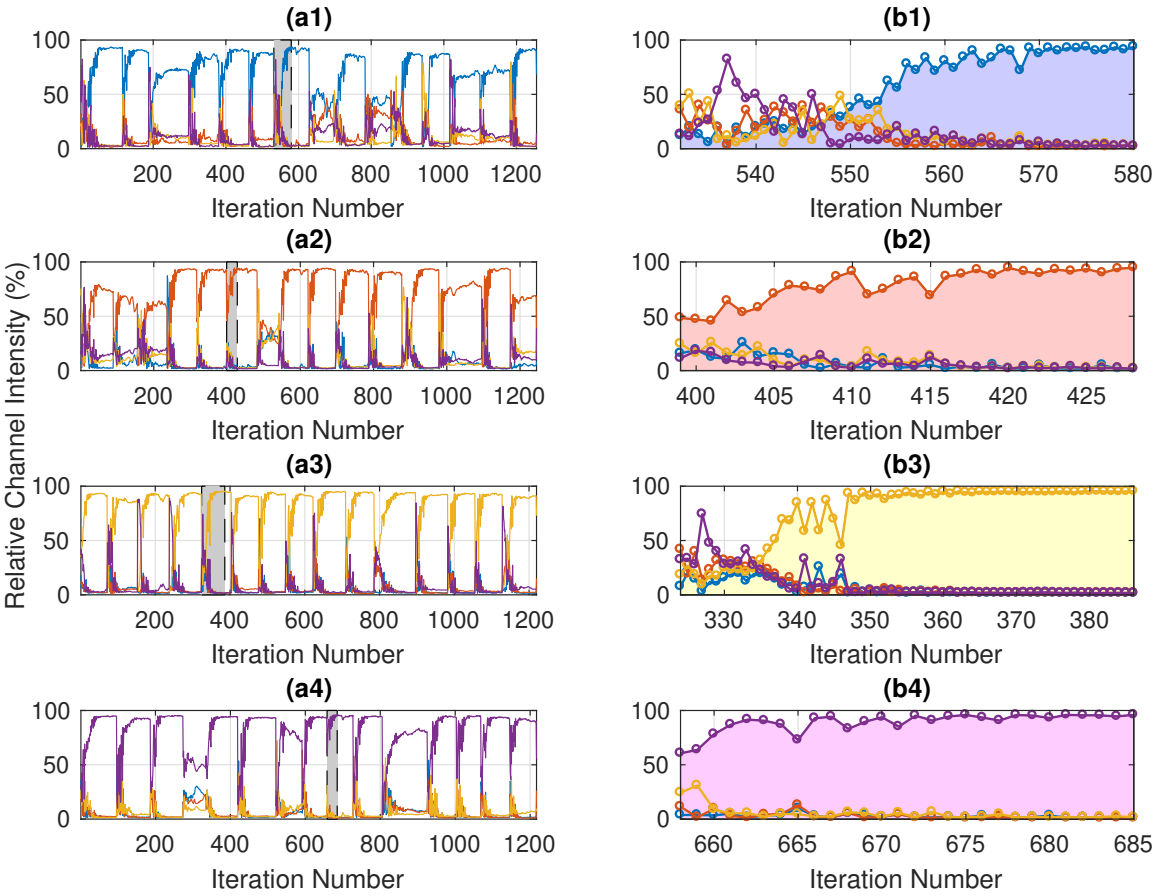

Figure 5.6: Evolution of relative channel intensities while maximizing channels 1 (a1), 2 (a2), 3 (a3), and 4 (a4) using the GBNM algorithm. Before running, the algorithm chose 12 random initialization points inside of the constraint sphere (defined by the piezoelectric voltage driver). It began descending from each point for 100 iterations, after which it moved on to the next point (unless it converged). Certain initializations converged much deeper than others, depending on how close they were to a noise induced local minimum. The relative channel intensities were saved at each iteration. The best initial point, along with its evolution for a particular channel are shown in column (b). Final channel intensities were 93.7% (b1), 94.7% (b2), 95.4% (b3), and 96.0% (b4).

the feedback algorithms will not only run faster, but techniques like temperature stabilization can be used to better preserve the system's corresponding codebook.

5.4 Summary

Designing and operating Green Machines is difficult because of stringent phase matching requirements. We derived and demonstrated a simple method to compensate for phase inconsistencies and characterize a device after fabrication using a feedback algorithm.

Our feedback algorithm scales well with size and does not require previous knowledge of the system architecture, since it relies on a convex objective function as defined by the inherent nature of the GM. Several optimization routines are readily available to minimize the system's objective function. We demonstrated the practicality of using the GBNM algorithm to compensate for time-varying system fluctuations and avoid noise-induce local minima. Future work may explore other routines that better handle these system fluctuations.

While our current configuration is in free space, future work will implement this algorithm on an integrated device and explore scaling with the size of the Green Machine.

CHAPTER 6. CONCLUSION AND FUTURE WORK

This thesis demonstrates a new, viable platform for integrated photonic circuit design. We note that all integrated photonic applications, quantum and classical, may benefit from this methodology. Quantum photonic processors, for example, often require bandwidth-engineered Bragg filters to separate pump photons from the signal and idler photons [59]. Similarly, this design framework can engineer filters for wavelength division multiplexing (WDM) communications transceivers [60].

With a single global parameter fit, we successfully modeled integrated photonic waveguides and chirped Bragg gratings with arbitrary bandwidths, chirping patterns, lengths, and corrugation widths and allowed arbitrary wavelength sampling. Future work could explore new network architectures (e.g. different activation functions, layer connections, etc.) and training algorithms. Several other devices, like ring resonators [61], can also be modeled. One could subsequently cascade several ANNs that model the scattering parameters of different devices, opening the door to large-scale optimization problems.

An important feature of this work is the choice to model the wavelength as a continuous input parameter rather than fix each output neuron at a specific wavelength point as done in all previous work known to the authors. The waveguide ANN, for example, outputs effective index values and the Bragg ANN outputs reflection and group delay values across the entire input spectrum. This approach, while more difficult to train, is more convenient for the designer. For example, an optimization routine tasked with designing a Bragg filter can focus more on parameters like the bandwidth and shape, rather than an arbitrarily sampled wavelength profile. Furthermore, this method doesn't require the training spectra to have the same sampling. Training sets for structures like ring resonators, whose features may require finer wavelength resolution than other devices, can now be strategically simulated to highlight these features. Assuming the network

is trained correctly across a suitable domain, the ANN will seamlessly interpolate between both design parameters and wavelength points without any additional routines.

Unlike traditional simulation methods, training an arbitrary device ANN requires large datasets that are too time-intensive for most typical computers. With the growing availability of vast cloud-based computational resources, however, several million training simulations can now be run in hours or days. [62]. Once trained, a neural network can reliably interpolate between training data, is compact and easily shared with the community, and can even continue to learn on new datasets via transfer learning [63]. Thus, the computational complexity inherent in designing integrated photonic devices can be moved to the front end of the design process, allowing individual designers to work with abstracted components whose optical response can be rapidly calculated.

6.1 Future Work

The new simulation paradigm presented in this work enables several new classical and quantum design applications. Many new implementations, algorithms, and paths are yet to be tried before this method can gain widespread traction in the community:

- A more thorough comparison of the various machine learning algorithms for each device.
- The effectiveness of this method for other devices with more complicated transfer functions, like circulators and branch couplers.
- Application of this method to other modeling frameworks, like eigenmode expansion (EE) and coupled mode theory (CMT).
- Better stochastic analysis with fabrication defects and perhaps more robust methods to account for the variance.
- More quantum applications, specifically with entangled photons and how their correlations propagate through a system.
- Modeling nonlinear effects and processes for both classical and quantum applications.

REFERENCES

[1] Heck, M. J., Chen, H., Fang, A. W., Koch, B. R., Liang, D., Park, H., Sysak, M. N., and Bowers, J. E., 2011. "Hybrid silicon photonics for optical interconnects." *IEEE Journal of Selected Topics in Quantum Electronics,* **17**(2), pp. 333–346. 1

[2] Barwicz, T., Byun, H., Gan, F., Holzwarth, C. W., Popovic, M. A., Rakich, P. T., Watts, M. R., Ippen, E. P., Kärtner, F. X., Smith, H. I., Orcutt, J. S., Ram, R. J., Stojanovic, V., Olubuyide, O. O., Hoyt, J. L., Spector, S., Geis, M., Grein, M., Lyszczarz, T., and Yoon, J. U., 2007. "Silicon photonics for compact, energy-efficient interconnects (invited)." *J. Opt. Netw.,* **6**(1), Jan, pp. 63–73. 1

[3] Wang, J., 2016. "Chip-scale optical interconnects and optical data processing using silicon photonic devices." *Photonic Network Communications,* **31**(2), Apr, pp. 353–372. 1

[4] Orieux, A., and Diamanti, E., 2016. "Recent advances on integrated quantum communications." *Journal of Optics,* **18**(8), jul, p. 083002. 1

[5] Flamini, F., Spagnolo, N., and Sciarrino, F., 2018. "Photonic quantum information processing: a review." *Reports on Progress in Physics,* **82**(1), nov, p. 016001. 1

[6] Bunandar, D., Lentine, A., Lee, C., Cai, H., Long, C. M., Boynton, N., Martinez, N., DeRose, C., Chen, C., Grein, M., Trotter, D., Starbuck, A., Pomerene, A., Hamilton, S., Wong, F. N. C., Camacho, R., Davids, P., Urayama, J., and Englund, D., 2018. "Metropolitan quantum key distribution with silicon photonics." *Phys. Rev. X,* **8**, Apr, p. 021009. 1

[7] Harris, N. C., Steinbrecher, G. R., Prabhu, M., Lahini, Y., Mower, J., Bunandar, D., Chen, C., Wong, F. N. C., Baehr-Jones, T., Hochberg, M., Lloyd, S., and Englund, D., 2017. "Quantum transport simulations in a programmable nanophotonic processor." *Nature Photonics,* **11**, 06, pp. 447 EP –. 1

[8] Qiang, X., Zhou, X., Wang, J., Wilkes, C. M., Loke, T., O'Gara, S., Kling, L., Marshall, G. D., Santagati, R., Ralph, T. C., Wang, J. B., O'Brien, J. L., Thompson, M. G., and Matthews, J. C. F., 2018. "Large-scale silicon quantum photonics implementing arbitrary two-qubit processing." *Nature Photonics,* **12**(9), pp. 534–539. 1

[9] Chrostowski, L., and Hochberg, M., 2015. *Silicon Photonics Design: From Devices to Systems*. Cambridge University Press. 1, 19, 20, 33

[10] Bogaerts, W., and Chrostowski, L., 2018. "Silicon Photonics Circuit Design: Methods, Tools and Challenges." *Laser & Photonics Reviews,* **12**(4), Apr., p. 1700237. 1, 27

[11] da Silva Ferreira, A., da Silva Santos, C. H., Gonçalves, M. S., and Hernández Figueroa, H. E., 2018. "Towards an integrated evolutionary strategy and artificial neural network computational tool for designing photonic coupler devices." *Applied Soft Computing,* **65**, Apr., pp. 1–11. 1

[12] Tahersima, M. H., Kojima, K., Koike-Akino, T., Jha, D., Wang, B., Lin, C., and Parsons, K., 2018. "Deep Neural Network Inverse Design of Integrated Nanophotonic Devices." *arXiv:1809.03555 [physics.app-ph]*, Sept. 1

[13] Inampudi, S., and Mosallaei, H., 2018. "Neural network based design of metagratings." *Applied Physics Letters,* **112**(24), June, p. 241102. 2

[14] Silva Ferreira, A. D., Malheiros-Silveira, G. N., and Hernandez-Figueroa, H. E., 2018. "Computing Optical Properties of Photonic Crystals by Using Multilayer Perceptron and Extreme Learning Machine." *Journal of Lightwave Technology,* **36**(18), Sept., pp. 4066–4073. 2

[15] Zhang, T., Liu, Q., Dai, J., Han, X., Li, J., Zhou, Y., and Xu, K., 2018. "Spectrum prediction and inverse design for plasmonic waveguide system based on artificial neural networks." *arXiv:1805.06410 [physics]*, May. 2

[16] Peurifoy, J., Shen, Y., Jing, L., Yang, Y., Cano-Renteria, F., DeLacy, B. G., Joannopoulos, J. D., Tegmark, M., and Soljacic, M., 2018. "Nanophotonic particle simulation and inverse design using artificial neural networks." *Science Advances,* **4**(6), June. 2

[17] Hammond, A. M., Potokar, E., and Camacho, R. M., 2019. SiP-ANN https://github.com/contagon/SiP-ANN. 3

[18] Krause, M., 2011. "Finite-difference mode solver for curved waveguides with angled and curved dielectric interfaces." *Journal of Lightwave Technology,* **29**(5), pp. 691–699. 3, 9

[19] Hammond, A. M., 2019. pyMode https://github.com/smartalecH/pyMode. 3, 9

[20] Hammond, A. M., Frank, I. W., and Camacho, R. M., 2018. "Error correction in structured optical receivers." *IEEE Journal of Selected Topics in Quantum Electronics,* **24**(6), pp. 1–8. 3, 33

[21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., 2011. "Scikit-learn: Machine Learning in Python ." *Journal of Machine Learning Research,* **12**, pp. 2825–2830. 6

[22] Schmidhuber, J., 2015. "Deep learning in neural networks: An overview." *Neural Networks,* **61**, Jan., pp. 85–117. 6

[23] le Cun, Y., 1988. "A theoretical framework for back-propagation." *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, pp. 21–28. 6

[24] Hornik, K., 1991. "Approximation Capabilities of Multilayer Feedforward Networks." *Neural Networks,* **4**(2), pp. 251–257. 6

[25] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., 2014. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research,* **15**, June, pp. 1929–1958. 7

[26] Gal, Y., and Ghahramani, Z., 2015. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." *arXiv:1506.02142 [cs, stat]*, June. 7

[27] Hammond, A. M., and Camacho, R. M., 2018. "Designing Silicon Photonic Devices using Artificial Neural Networks." *arXiv:1812.03816 [physics]*, Nov. arXiv: 1812.03816. 8, 25

[28] Hammond, A. M., Potokar, E., and Camacho, R. M., 2019. "Accelerating silicon photonic parameter extraction using artificial neural networks." *arXiv preprint arXiv:1901.08176*. 8, 25

[29] Johnson, S. G., and Joannopoulos, J. D., 2001. "Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis." *Optics Express,* **8**(3), Jan., pp. 173–190. 8, 10, 11

[30] Chollet, F., 2015. Keras https://github.com/fchollet/keras. 10

[31] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems Software available from tensorflow.org. 10

[32] Helan, R., 2006. "Comparison of methods for fiber bragg gratings simulation." In *2006 29th International Spring Seminar on Electronics Technology*, IEEE, pp. 161–166. 11, 13

[33] Gallagher, D. F. G., and Felici, T. P., 2003. "Eigenmode expansion methods for simulation of optical propagation in photonics - Pros and cons." In *Integrated Optics: Devices, Materials, and Technologies Vii*, Y. S. Sidorin and A. Tervonen, eds., Vol. 4987. Spie-Int Soc Optical Engineering, Bellingham, pp. 69–82. 11

[34] Hill, K. O., and Meltz, G., 1997. "Fiber Bragg grating technology fundamentals and overview." *Journal of Lightwave Technology,* **15**(8), Aug., pp. 1263–1276. 13

[35] Harris, F. J., 1978. "On the use of windows for harmonic analysis with the discrete Fourier transform." *Proceedings of the IEEE,* **66**(1), Jan., pp. 51–83. 13

[36] Storn, R., and Price, K., 1997. "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces." *Journal of Global Optimization,* **11**(4), Dec, pp. 341–359. 14

[37] Marquardt, D., 1963. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." *Journal of the Society for Industrial and Applied Mathematics,* **11**(2), June, pp. 431–441. 14

[38] Tan, D. T. H., Sun, P. C., and Fainman, Y., 2010. "Monolithic nonlinear pulse compressor on a silicon chip." *Nature Communications,* **1**, Nov., p. 116. 26

[39] Selvaraja, S. K., 2011. *Wafer-scale fabrication technology for silicon photonic integrated circuits*. Ph.D. thesis (Ghent University, Ghent, Belgium). 27

[40] Lu, Z., Jhoja, J., Klein, J., Wang, X., Liu, A., Flueckiger, J., Pond, J., and Chrostowski, L., 2017. "Performance prediction for silicon photonics integrated circuits with layout-dependent correlated manufacturing variability." *Optics Express,* **25**(9), May, p. 9712. 27

[41] Zortman, W. A., Trotter, D. C., and Watts, M. R., 2010. "Silicon photonics manufacturing." *Optics Express,* **18**(23), Nov., pp. 23598–23607. 27

[42] Wang, X., Shi, W., Yun, H., Grist, S., Jaeger, N. A. F., and Chrostowski, L., 2012. "Narrow-band waveguide Bragg gratings on SOI wafers with CMOS-compatible fabrication process." *Optics Express,* **20**(14), July, pp. 15547–15558. 27

[43] Strain, M. J., and Sorel, M., 2010. "Design and Fabrication of Integrated Chirped Bragg Gratings for On-Chip Dispersion Control." *IEEE Journal of Quantum Electronics,* **46**(5), May, pp. 774–782. 27

[44] Chrostowski, L., Wang, X., Flueckiger, J., Wu, Y., Wang, Y., and Fard, S. T., 2014. "Impact of Fabrication Non-Uniformity on Chip-Scale Silicon Photonic Integrated Circuits." In *Optical Fiber Communication Conference*, OSA, p. Th2A.37. 27

[45] Chen, X., Li, Z., Mohamed, M., Shang, L., and Mickelson, A. R., 2014. "Parameter extraction from fabricated silicon photonic devices." *Applied Optics,* **53**(7), Mar., p. 1396. 27

[46] Melati, D., Alippi, A., and Melloni, A., 2016. "Waveguide-Based Technique for Wafer-Level Measurement of Phase and Group Effective Refractive Indices." *Journal of Lightwave Technology,* **34**(4), Feb., pp. 1293–1299. 27

[47] Xing, Y., Dong, J., Dwivedi, S., Khan, U., and Bogaerts, W., 2018. "Accurate extraction of fabricated geometry using optical measurement." *Photonics Research,* **6**(11), Nov., p. 1008. 29

[48] Jones, E., Oliphant, T., and Peterson, P., 2001–. SciPy: Open source scientific tools for Python. 29

[49] Branch, M. A., Coleman, T. F., and Li, Y., 1999. "A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems." *SIAM Journal on Scientific Computing,* **21**(1), pp. 1–23. 30

[50] Guha, S., 2011. "Structured Optical Receivers to Attain Superadditive Capacity and the Holevo Limit." *Physical Review Letters,* **106**(24), June, p. 240502. 33

[51] Butler, J., 1961. "Beam-forming Matrix Simplifies Design of Electronically Scanned Antennas." *Electronics Design,* **9**, Apr., pp. 170–173. 34

[52] Fenn, A. J., Temme, D. H., Delaney, W. P., and Courtney, W. E., 2000. "The development of phased-array radar technology." *Lincoln Laboratory Journal,* **12**(2), pp. 321–340. 34

[53] Ueno, M., 1981. "A systematic design formulation for Butler matrix applied FFT algorithm." *IEEE Transactions on Antennas and Propagation,* **29**(3), pp. 496–501. 37

[54] Polyak, B. T., 1998. "Convexity of quadratic transformations and its use in control and optimization." *Journal of Optimization Theory and Applications,* **99**(3), pp. 553–583. 43

[55] Broadband Non-Polarizing Cube Beamsplitters (BNPB) – Lambda CC. 46

[56] Boyd, S. P., and Vandenberghe, L., 2004. *Convex optimization*. Cambridge University Press, Cambridge, UK ; New York. 46

[57] Luersen, M., Le Riche, R., and Guyon, F., 2004. "A constrained, globalized, and bounded Nelder?Mead method for engineering optimization." *Structural and Multidisciplinary Optimization,* **27**(1-2), May, pp. 43–54. 46

[58] Nelder, J. A., and Mead, R., 1965. "A Simplex Method for Function Minimization." *The Computer Journal,* **7**(4), Jan., pp. 308–313. 46

[59] Silverstone, J. W., Bonneau, D., O'Brien, J. L., and Thompson, M. G., 2016. "Silicon quantum photonics." *IEEE Journal of Selected Topics in Quantum Electronics,* **22**(6), pp. 390–402. 49

[60] Brouckaert, J., Bogaerts, W., Selvaraja, S., Dumon, P., Baets, R., and Van Thourhout, D., 2008. "Planar concave grating demultiplexer with high reflective bragg reflector facets." *IEEE Photonics technology letters,* **20**(4), pp. 309–311. 49

[61] Bogaerts, W., De Heyn, P., Van Vaerenbergh, T., De Vos, K., Selvaraja, S. K., Claes, T., Dumon, P., Bienstman, P., Van Thourhout, D., and Baets, R., 2012. "Silicon microring resonators." *Laser & Photonics Reviews,* **6**(1), Jan., pp. 47–73. 49

[62] Vecchiola, C., Pandey, S., and Buyya, R., 2009. *High-Performance Cloud Computing: A View of Scientific Applications*. IEEE, New York. 50

[63] Pan, S. J., and Yang, Q., 2010. "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering,* **22**(10), Oct., pp. 1345–1359. 50