Machine Learning Modelling, Optimisation and Thermal Compensation of Photonic Waveguides

Sunny Chugh

School of Mathematics, Computer Science & Engineering

City, University of London

This dissertation is submitted for the degree of

Doctor of Philosophy

March 2020

I would like to dedicate this thesis to my parents and siblings...

Declaration

I, hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except where specified in the text.

Sunny Chugh

March 2020

Acknowledgements

I would like to thanks my supervisor, Professor B. M. A. Rahman, for having me as a student and providing me with all the opportunities. I am also very grateful to my parents, siblings and friends for their never ending support and encouragement throughout my studies. I am also very thankful of the Erasmus Mundus Leaders scholarship program from the European Commission for awarding me PhD scholarship. I would also like to thank every member and alumni of the Photonics Modelling Research group especially, Dr. Souvik Ghosh for all the technical and moral support.

Abstract

Nanophotonic devices has led to many interesting applications in optical sensing, fibre lasers, fibre amplifiers, optical signal processing, and many others. Modelling and optimisation of such devices depends upon the numerical methods employed for modal analysis, such as finite difference method, finite element method, beam propagation method, and others to compute various optical properties including effective index, power confinement, mode effective area, dispersion, confinement loss, etc. One of the aims of this dissertation is to develop a finite element based time domain technique, similar to finite difference time domain method, that can have varying mesh resolutions for spatial discretisation of the computational domain. Parallel programming has been employed to speed up the simulations.

However, various design parameters of the optical devices are generally optimised before fabrication. This becomes an iterative process of trying and testing different design parameters which may require significant time and computer resources when dealing with complex optical structures. In this research work, the power of artificial intelligence techniques has been employed to quickly estimate the various properties of different photonics devices (slot, strip, and directional coupler waveguides) and photonic crystal fibre. An in-house code using a machine learning (an application of artificial intelligence) regression approach has been developed. Accuracy of these techniques are described by comparing their outputs with the actual outputs. PyTorch and Python programming languages are extensively used during the development of machine learning approach.

Additionally, a silicon directional coupler having a metallic heater for each waveguide for thermal compensation has been studied to tackle possible fabrication tolerances. The respective metal heater (either or both) can be activated or heated depending on whether there is a fabrication error in one or both of the waveguides. Fabrication errors change the coupling length of the directional coupler which is thermally compensated by placing heaters at different positions with respect to the waveguides.

Table of contents

List of figures

List of tables

Publications

Journals

1. **Sunny Chugh**, Souvik Ghosh, Aamir Gulistan and B. M. A. Rahman, “Machine learning regression approach to the nanophotonic waveguide analyses," *Journal of Lightwave Technology*, vol. 37, no. 24, pp. 6080–6089, 2019. https://doi.org/10.1109/JLT.2019.2946572

2. **Sunny Chugh**, Aamir Gulistan, Souvik Ghosh, and B. M. A. Rahman, “Machine learning approach for computing optical properties of a photonic crystal fiber,” *Optics Express*, vol. 27, no. 25, pp. 36414–36425, 2019. https://doi.org/10.1364/OE.27.036414 **(Editor’s Pick)**

3. **Sunny Chugh**, Souvik Ghosh, and B. M. A. Rahman, “Thermal compensation of phase mis-matching in a silicon directional coupler,” **(Under Review)**

Conferences

1. **Sunny Chugh**, Aamir Gulistan, Souvik Ghosh, Sneha Verma and B. M. A. Rahman, “Calculating photonic crystal fibre properties by using machine learning,” in *SPIE. Photonics Europe*, 2020 **(Accepted)**

2. **Sunny Chugh**, Souvik Ghosh, and B. M. A. Rahman, “Enhancing the Sensitivity by using Split Sections in a Ring Resonator”, in *Proceedings of the Photonics-2018*, 2018. Paper No. SP067, ISBN: 978-93-88653-41-1 **(Not a part of this thesis)**

3. Souvik Ghosh, Aamir Gulistan, **Sunny Chugh** and B. M. A. Rahman, "Modelling of Hybrid Plasmonic Asymmetric Mach-Zehnder Interferometer for Sensing Applications", in *Proceedings of the Photonics-2018*, 2018. Paper No. TP060, ISBN: 978-93-88653-41-1 **(Not a part of this thesis)**

Chapter 1

Introduction

1.1 Objectives and Motivations

Design, fabrication, characterisation, analysing error and possible redesigning can be time consuming steps in many industrial applications including the development of photonic devices. In the last decades, silicon photonics has become one of the most promising photonic integration platforms [1–4], as it provides a high index contrast device structure that leads to reduction in footprint of the waveguides. This smaller footprint helps in fabricating miniaturized sensors that can be extensively used for clinical analysis, healthcare, environmental monitoring, and biomedical sensing. Also, nanophotonic structures which depend on a large number of device parameters are being designed and used for novel photonics applications [3, 4].

To reduce both time and cost of development, many numerical models and software programs (open source and commercial) have been developed for various photonics applications. Accuracy for these modelling techniques has been increasing tremendously with the advancement in computational technology. Different models and simulators can be used to study two- and three- dimensional optical devices. While two-dimensional calculations are sufficient to explain some concepts and phenomena, full three-dimensional simulations

are necessary for determining the parameters of the devices intended to be used in real systems. Moreover, if the time response of the device needs to be studied, it can make the simulation process computationally expensive depending on the dimensions of the optical device. The Finite Difference Time Domain (FDTD) method is commonly used to study the three-dimensional time response, and it generally comprises a rectangular mesh with one grid resolution for the computational domain. This grid technique can become inefficient for complex devices with curved boundaries. To overcome this, an in-house Finite Element Time Domain (FETD) code has been developed which uses triangular mesh and can have different mesh resolutions in the computational domain as desired.

Inspite of having fast and advanced computers, a rigorous optimisation of the waveguide design parameters through parameter sweep often becomes a time consuming task. For example, the FDTD method may require between several minutes to hours to analyse the optical transmission response of a single photonic device, depending on its design. Recently, researchers started applying machine learning (ML) methods to optical communication systems to quickly optimise various parameters [5–9]. Technology companies like Google, Microsoft, Facebook, IBM, Baidu, Apple, Netflix, and others use machine learning models in their day-to-day applications. This becomes the motivation to explore the new domain of employing machine learning techniques to quickly and efficiently estimate outputs for various nanophotonics devices (slot, strip and directional coupler waveguides) and photonic crystal fibres.

After numerical modelling, the next step is to fabricate the device. Although, associated fabrication technologies have improved a lot in the past decades but these still incur some errors when fabricating optical devices. Thus, the major requirements include corrections of fabrication inaccuracies along with the reconfiguration of the system characteristics. Hence, a thorough study has been performed to understand how to compensate phase mis-matching

in a silicon directional coupler occurring due to fabrication inaccuracies using integrated metallic heaters close to each waveguide.

1.2 Thesis Outline

This thesis comprises eight chapters, including the current chapter on introduction and two Appendices. The chapter contents are briefly described as follows:

Chapter 2 presents the review of analytical and numerical methods used in the literature for design, optimisation, and performance analyses of the optical waveguides. These numerical methods generally use Maxwell's equations (in differential or integral form) when solving the computational domain for electromagnetic wave propagation. It is identified that the mesh size plays a crucial role in improving the accuracy of the numerical methods. A trade-off is required as decreasing the mesh element size might significantly increase the computation time and resources.

Chapter 3 focuses on the governing equations used for the two-dimensional structures for Finite Element Time Domain (FETD) method. Space discretisation of the two-dimensional computational domain is described. Next, it is explained how open source software, Gmsh is used to generate two different meshes which are used to store various electromagnetic field components at their nodes. In the next section, time discretisation of the problem/computational domain is achieved, followed by a brief introduction about the use of perfectly matched layers (PMLs) at the boundaries to reduce the reflection.

Chapter 4 discusses the proposed FETD code (generated using C++ programming language and OpenMP) for two-dimensional structures. First, the validity of FETD code is checked by comparing the obtained speed of light using FETD code with the actual normalised assumption. Next, a planar waveguide is considered for different types of input sources (point and mode sources), and normalised field profiles from FETD and an in-house FEM (already well developed and verified) codes are compared. The coupling length for

a directional coupler has been calculated by using the newly developed FETD code and compared with the FEM solutions for different mesh resolutions.

Chapter 5 discusses using machine learning based optimisation techniques for integrated silicon photonics devices. Machine learning is an application of artificial intelligence that focuses on the development of computer algorithms which learn automatically by extracting patterns from the data provided. Here, the finite element simulations and machine learning techniques are combined for the prediction of effective indices, power confinement and coupling length of different integrated photonics devices. Initially, a dataset using COMSOL Multiphysics is prepared and then this data is used for training while optimising various parameters of the machine learning model. Waveguide width, height, operating wavelength, and other device dimensions are varied to record different modal solution parameters. A detailed study has been carried out for a slot waveguide structure to evaluate different machine learning model parameters including the number of layers, number of nodes, choice of activation functions, and others. After training, this model is used to predict the outputs for new input device specifications. This method predicts the output for different device parameters faster than direct numerical simulation techniques. An absolute percentage error of less than 5% in predicting an output has been obtained for slot, strip and directional waveguide coupler designs.

Chapter 6 discusses using machine learning techniques to compute various optical properties including effective index, effective mode area, dispersion and confinement loss for a solid core photonic crystal fibre (PCF). These machine learning algorithms based on artificial neural networks, are able to make accurate predictions of the above mentioned optical properties for an usual parameter space of wavelength ranging from 0.5-1.8 μm, pitch from 0.8-2.0 μm, diameter by pitch from 0.6-0.9 and number of rings as 4 or 5 in a silica solid core PCF. The use of simple and fast-training feed forward artificial neural networks that predict the output for unknown device parameters faster than conventional numerical

simulation techniques has been demonstrated. Computation runtimes required with neural networks (for training and testing) and Lumerical Mode Solutions are also compared.

Chapter 7 describes a design of a Silicon-on-Insulator (SOI) directional coupler integrated with metallic heaters to compensate possible phase mismatching which may arise due to fabrication inaccuracies. A detailed study has been carried out to obtain the temperature profiles, thermally compensated field matching conditions, optical mode profiles, and time domain evolution of optical power transfer for different heater positions and heater power. It has been shown that heating the waveguides changes their isolated effective indices and hence power coupling between waveguides is also changed. Power transfer between the waveguides is examined using the time domain approach with and without fabrication defects for different heater powers. The heater power needed to obtain the field matching condition can be further reduced if the gap between the heater and waveguide is decreased, but this potentially increases the modal loss due to absorption in the metallic heater.

This thesis concludes with **Chapter 8** which summarises all the findings and explores possible directions of future work.

Chapter 2

Overview of Numerical Methods for Electromagnetics

2.1 Maxwell's Equations

Maxwell's equations are used to describe the propagation of electromagnetic waves, including light in optical waveguides. Maxwell's equations named after James Clerk Maxwell form the basis of the classical electromagnetism. These equations demonstrate the relationship between the electric and magnetic fields propagating in any medium. These equations form a set of four partial differential equations that can be presented in differential and integral forms.

2.1.1 Differential Form

The differential form of Maxwell's equations can be used to find time and space dependent electromagnetic field values in an optical waveguide. This form of the equations is more frequently used to solve analytical and numerical problems for calculation of the electromagnetic fields in more complicated (less symmetric) situations. The wave equation derived

using the differential form is more popular for various finite difference and finite element analyses [10].

The differential form of Maxwell's equations for the time varying electromagnetic fields is more widely used, and is defined as follows:

$$\nabla \cdot \mathbf{D} = \rho \tag{2.1}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.2}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.3}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{2.4}$$

2.1.2 Integral Form

The integral form of Maxwell's equations is used to describe the underlying physical laws. This form of the equations is needed to establish the boundary conditions. It is often used to describe electromagnetic fields having a higher degree of symmetry. These equations can be easily derived from the differential form by applying Stokes theorem [11]. They can also be used in some finite difference algorithms [12, 13] and finite integration methods [14]. The integral form of Maxwell's equations is as follows:

$$\oint_S \mathbf{D} \cdot d\mathbf{S} = Q_{enclosed} \tag{2.5}$$

$$\oint_S \mathbf{B} \cdot d\mathbf{S} = 0 \tag{2.6}$$

$$\oint \mathbf{E} \cdot d\mathbf{l} = -\frac{\partial}{\partial t} \int_S \mathbf{B} \cdot d\mathbf{S} \tag{2.7}$$

$$\oint \mathbf{H} \cdot d\mathbf{l} = \oint_S \mathbf{J} \cdot d\mathbf{S} + \frac{\partial}{\partial t} \int_S \mathbf{D} \cdot d\mathbf{S} \tag{2.8}$$

where d**S**, d**l** are the vectors denoting the change in the surface **S** and the change on the line **l**, respectively.

The quantities involved in Equations 2.1–2.8 and their respective units are listed below:

Table 2.1 Electromagnetic quantities and units

Quantity	Description	Units
E	Electric field amplitude	Volt/meter (V/m)
H	Magnetic field amplitude	Amp/meter (A/m)
D	Electric flux density	$Coulomb/meter^2 (C/m^2)$
B	Magnetic flux density	$Weber/meter^2 (Wb/m^2)$
J	Current density	$Amp/meter^2 (A/m^2)$
ρ	Charge density	$Coulomb/meter^3 (C/m^3)$
Q	Charge	Coulomb (C)

E and **H** denote the amplitudes describing the strength of a field at a given point in the space and time. **D** and **B** denote the fluxes.

2.1.3 Constitutive Relations

The electric and magnetic flux densities (**D** and **B**) are related to the electric and magnetic field amplitudes (**E** and **H**) by the constitutive relations. The functional form of the relationship depends upon the nature of the medium. For linear and isotropic media, the relations between electric flux, magnetic flux, electric field and magnetic field are given by:

$$\mathbf{B} = \mu\mathbf{H} \tag{2.9}$$

$$\mathbf{D} = \varepsilon\mathbf{E} \tag{2.10}$$

where μ is the magnetic permeability of the medium (in Henry/meter) and ε is the electrical permittivity of the medium (in Farad/meter). The values of μ and ε in vacuum are symbolically denoted by μ_0 and ε_0, respectively and are given as:

$$\mu_0 = 4\pi \times 10^{-7} \qquad Henry/m$$

$$\varepsilon_0 = 8.854 \times 10^{-12} \; Farad/m$$

2.1.4 The Wave Equation

The electromagnetic wave equation can be derived using Maxwell's equations. For a source free (ρ = 0, $\mathbf{J}$ = 0), linear (μ and ε are independent of $\mathbf{E}$ and $\mathbf{H}$), and isotropic medium conditions, Equations 2.1–2.4 become:

$$\nabla \cdot \mathbf{D} = 0 \tag{2.11}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{2.12}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.13}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} \tag{2.14}$$

Equations 2.11–2.14 are strongly coupled first-order differential equations and it is not easy to implement these equations in a computer program for the solution of many problems. Therefore, it is usual practice for many algorithms to use decoupled second-order differential equations that consist of only one field value (either **E** or **H**). It should be noted that the choice of a coordinate system is critical in obtaining the wave equation.

The wave equation in terms of the electric field amplitude is given by [11]:

$$\nabla^2\mathbf{E} - \mu\varepsilon\frac{\partial^2\mathbf{E}}{\partial t^2} = 0 \tag{2.15}$$

While, the wave equation in terms of the magnetic field amplitude is given by [11]:

$$\nabla^2\mathbf{H} - \mu\varepsilon\frac{\partial^2\mathbf{H}}{\partial t^2} = 0 \tag{2.16}$$

As both wave equations consist of only one field amplitude, discretisation of the computational domain for different numerical methods becomes easier in comparison to when using the coupled equations [15].

2.2 Analytical and Semi-Analytical Methods

Analytical methods are appropriate for the solution of the basic electromagnetic equations to describe the propagation of light through the optical devices and therefore only provide solutions for simple structures. It is only possible to use an analytical solution for planar/slab optical waveguides. Even for the two-dimensional optical waveguide structures some approximations are needed to obtain the analytical solutions [15]. In 1969, Marcatili [16] proposed a famous approximate analytical method that describes the propagation of

light through a low refractive index contrast rectangular dielectric optical waveguide. The Effective Index Method (EIM) for the analysis of optical waveguides proposed by Knox and Toulios in 1970 [17] is an extension of the Marcatili method. The EIM method was the most popular among the approximate methods, in which the rectangular structure is replaced by an equivalent slab structure having an effective index. The effective index is defined as the ratio of propagation constant in the waveguide to the free-space propagation constant. The drawback of the conventional EIM is that it only produces accurate results when the waveguide modes are operated in the far-from-cutoff region. Different variants of the EIM were developed, including the EIM based on linear combinations of solutions [18] and with a perturbation correction [19].

However, in the last few decades, high refractive index contrast waveguides have attracted strong interest for use in photonic integrated devices. High refractive index contrast provides strong light confinement in such devices that allows a small device footprint. In 2015, Westerveld *et al.* [20] described an approximate model of how light travels through a rectangular high refractive index contrast waveguide. Westerveld *et al.* improved Marcatili's model by adjusting the amplitudes of the electromagnetic field components. This improved model also shows good agreement with rigorous numerical simulations for the high index contrast waveguides.

2.3 Numerical Methods

Complex nanophotonic structures are being designed and fabricated to enable novel applications in optics and integrated photonics. Due to the large number of parameters used in the modern optical devices, analytical methods are not sufficient to understand the behaviour of such structures. Simulation and modelling using numerical methods play an important role in any scientific work. With continuous improvement of computational power at a reduced cost, modelling has become the key approach for developing improved photonic devices.

There exists a wide range of numerical methods that are constantly being improved for the design, development, and optimisation of the photonic components. Finite differences, finite elements, boundary integral, surface integral, volume integral, and hybrid methods are some of the widespread numerical approaches that are derived from the classical electrodynamics. For the accurate modelling of the optical devices a clear understanding of pros and cons of the available numerical methods is necessary when choosing an appropriate computational/numerical method for a photonics design problem. Most of the numerical modelling techniques use Maxwell's equations in some form as a starting point while solving the problem domain. The following includes some of the most widely used numerical analyses techniques for the design, development and characterisation of photonic devices.

2.3.1 Modal Analysis

The analysis of modal characteristics is an important subject in photonic waveguides and devices. The modal analysis method is used to find the propagation constant or effective index of the guided modes in a waveguide. It also provides the field profiles (of **E** and **H** fields) of all the modes supported by a waveguide. For the modal analysis method, it is assumed that the waveguide is uniform in the direction of propagation and a cross-sectional plane (for a 3D structure) or line (for a 2D structure) is considered. This method calculates the propagation constant or effective index for one frequency at a time. The solver can be used with both scalar and full-vectorial formulations. The vector **H**-field formulation can be written as [21, 22]:

$$\omega^2 = \frac{\int (\nabla \times \mathbf{H})^* \cdot \frac{1}{\varepsilon} \cdot (\nabla \times \mathbf{H}) d\Omega}{\int \mathbf{H}^* \cdot \mu \cdot \mathbf{H} d\Omega} \tag{2.17}$$

where ω and Ω are the angular frequency of the electromagnetic wave and the waveguide cross-section, respectively.

2.3.2 Beam Propagation Method

The Beam Propagation Method (BPM) can be used to simulate the propagation of light in slowly varying optical waveguides and is inaccurate for the modelling of discretely or fast varying structures. It is typically used to describe and study the evolution of total field propagating along the longitudinally varying designs, such as bent, tapered, Y-junctions, and terminated waveguide structures. The BPM algorithms take the initial field provided at the input and march it along the direction of propagation to produce the expected field at the output. This method can be inaccurate for the modelling of devices with high refractive index contrast, for example silicon-on-insulator structures.

Initially the BPM was reported in 1980 [23] based on the Fast Fourier Transformation (FFT). It is used to generate the mode related properties such as propagation constants, relative mode powers and group delays with high precision, which are required for the analysis of the optical fibre dispersion. Several numerical algorithms employing finite difference techniques for the vector BPM have been reported in the literature [24–26]. Subsequently in 1996 [27], Tsuji and Koshiba presented a unified finite element based BPM for both TE and TM waves propagating in a strongly guiding and longitudinally varying optical waveguides. Later in 2000 [28], Obayya *et al.* reported a finite element based full-vectorial BPM algorithm to accurately characterise 3D optical waveguide devices.

2.3.3 Finite Difference Method

The Finite Difference Method (FDM) is one of the simplest and oldest method used to solve differential equations that are difficult or impossible to solve analytically. The advent of finite difference techniques in numerical applications began in the early 1950s. In mathematics,

FDM solves the differential equations by approximating them with difference equations. It converts linear (non-linear) ordinary/partial differential equations into a system of linear (non-linear) equations. The resulting equations are then solved using matrix algebra. Modern computers can be efficiently used to solve these matrix equations. FDMs discretise the computational/problem's domain in space and time. Then approximations of the solution are computed at the space or time points. The computation domain is usually divided into a uniform grid. The error between the numerical solution and the exact solution is called discretisation error.

FDMs application to optical waveguide modelling are dated back to the early eighties. To restrict the computational domain for the numerical simulation of optical devices using FDM, a finite cross-section is defined which encloses the dielectric waveguide. Boundaries of this cross-section are either electric or magnetic walls and it is assumed that the field at these boundaries are very small. Perfectly matched layer (PML) at the boundaries can also be used which absorbs the fields at the edges without reflection [29]. The enclosed cross-section is divided into rectangular grids called as a mesh. This mesh consists of various nodal points which are used to store one or more field variables. A uniform mesh is most commonly used but can result in a large number of nodes depending upon the element size of the defined mesh. The accuracy of this method depends on the mesh element size. If the mesh element size is large, simulation is quick, but can result in inaccurate results. Decreasing the mesh element size too much may increase the accuracy, but it takes longer time to process and would be computationally inefficient.

2.3.4 Finite Element Method

The Finite Element Method (FEM) is a numerical technique used for solving a wide variety of engineering problems, including computational electromagnetics and is particularly used for problems involving irregular geometries and steep gradients. The FEM was initially

introduced for the modelling of mechanical applications related to aerospace and civil engineering structures. Later on, it was employed in different areas of interest including structural analysis, heat transfer, fluid flow, biomechanics, biomedical, electromagnetics, among others.

The main feature of FEM is to break the spatial domain (one-, two- or three-dimensional) into a number of simple geometric elements such as triangles or quadrilaterals. Each element may have different material properties in terms of its relative permeability and permittivity. These elements are assumed to be connected to one another, but only at interconnected joints, known as nodes. The complete arrangement of the elements is known as a mesh. The FEM formulation of the problem results in a system of algebraic equations. Each element represents a set of equations. All sets of element equations are then systematically recombined into a global system of equations that models the entire problem. FEM theory is well developed and offers great freedom in the selection of discretisation. The accuracy of this method also depends upon the mesh. A finer mesh across the whole domain may yield more accurate results but at the cost of increased computing time. In order to reduce this computation time different element sizes for the discretisation can also be considered. A finer mesh can be used in the areas of the problem domain where the field has a rapid variation. On the other hand, a coarser mesh can be used where there is a little variation or almost negligible field amplitudes.

2.3.5 Finite Difference Time Domain Method

Finite Difference Time Domain (FDTD) belongs to the class of finite difference numerical modelling method and involves stepping the system through discrete periods of time. FDTD is a time-domain approach and provides a solution for a wide range of frequencies with each simulation. FDTD method was first proposed in 1966 by Yee [30] for solving Maxwell's curl equations on staggered grids in space and time. In 1980, Taflove [31] validated the

application of FDTD method to sinusoidal steady-state electromagnetic fields penetrating an arbitrary dielectric or conducting body. In 1988, FDTD modelling of microstrips was introduced by Zhang *et al.* [32]. Since 1990, FDTD has been the most widely used method to computationally model many time-domain engineering problems dealing with electromagnetics. FDTD modelling applications include antennas, wireless communications, photonic crystals, nanoplasmonics, solitons, biophotonics, and others.

A computational domain must be established first to implement a FDTD solution that is simply a physical region over which the simulation is performed. The material of each cell (formed by the grids) within the computational domain must be specified in advance depending upon the device dimensions. Grid materials are defined in terms of permeability, permittivity, and conductivity. The source is then specified, for example the current on a wire or the applied electric field. FDTD method discretises the time-dependent Maxwell's equations to the space and time partial derivatives. The resulting finite-difference equations are then solved in a leapfrog manner. Electric field vector components for the computational domain are solved at a given time-step and then at the next time-step the magnetic field vector components are computed from the previously obtained electric field vector components for the same computational domain. This process is repeated and at each time-step the electric and magnetic field components are updated until steady-state or transient electromagnetic fields are obtained for the computational domain. The time taken and required computer memory is proportional to the size of the computational domain that depends on the dimensions of the photonic structure to be modelled.

Yee [30] proposed spatially staggering the electric and magnetic field vector components about rectangular unit cells of a computational grid, as shown in Fig. 2.1. Each electric field vector component is located midway between a pair of magnetic field vector components, and conversely. This is also known as a Yee lattice. Yee's proposed algorithm solves for both electric and magnetic fields using the coupled Maxwell's equations instead of using the wave

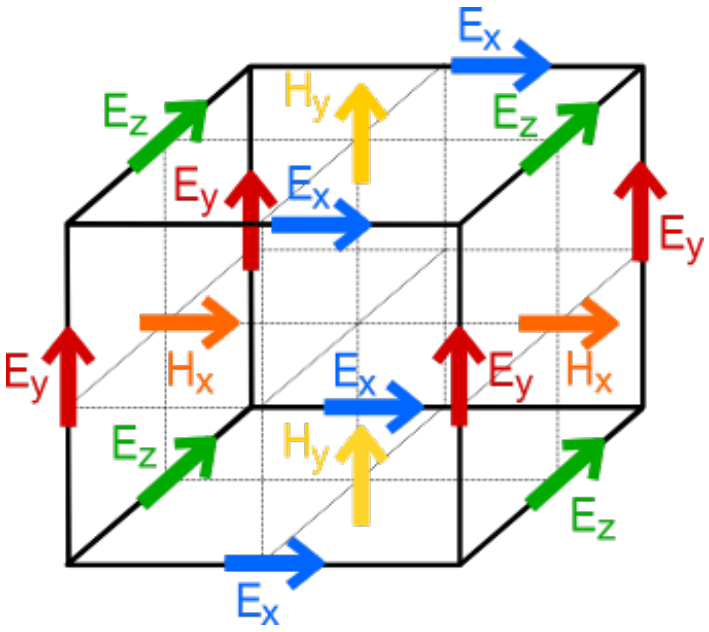


Fig. 2.1 3D cuboid representation for FDTD.

equation. The most commonly used grid truncation techniques in FDTD modelling are the perfectly matched layer (PML) formulations. PML can provide orders of magnitude lower reflections than other truncation techniques. There is a limit on the chosen time step value (Δt) to ensure the stability of the FDTD algorithm. This stability factor (S), also known as Courant–Friedrichs– factor for three dimensional geometry is given by [33]:

$$S = c \cdot \Delta t \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}} \tag{2.18}$$

where c is the wave propagation speed, Δt is the time step value, Δx, Δy and Δz are the space increments in the x, y and z directions, respectively, and the stability factor is $S < 1$.

2.3.5.1 Strengths of FDTD modelling

- FDTD is a very powerful numerical approach to obtain a full time response over a wide range of frequencies with a single simulation. This is particularly useful in

applications where resonant frequencies are not exactly known or when a broadband result is needed.

- The FDTD method does not create any global matrix to find the values at the next time-step. All calculations for the evolution of the fields are done locally to each Yee's lattice.
- It is suitable for any type of parallel computing as the governing equations are independent of each other, which further speeds up the simulations.

2.3.5.2 Weaknesses of FDTD modelling

- FDTD technique is computationally more expensive and normally requires relatively longer simulation times than other numerical methods. Modern computers have overcome this problem to some extent because they have large amounts of memory storage and fast processor speeds.
- FDTD method requires only one grid resolution throughout the computational domain, and hence the grid dimensions should be sufficiently fine to resolve the smallest geometrical feature in the model. Models with long, thin features (like wires) are difficult to model in FDTD. Eigenmode Expansion can be used as an alternative as it does not require a finer grid along the z-direction [34].
- Complex structures that do not conform to the rectangular grids introduce errors in the results. A stair case approximation is a better choice for the simulation of such designs.

2.3.6 Finite Element Time Domain Method

The problems with the FDTD methods are mostly associated with the grid. Therefore, a better grid or meshing technique is required to improve the overall transient response

performance. Finite Element Time Domain (FETD) methods are better alternatives based on a geometry-conforming mesh (structured or unstructured) [35]. In contrast to FDTD, the FETD method allows more accurate representations of arbitrarily irregular and complex structures by means of a flexible triangular and curvilinear mesh. FETD formulation has been widely investigated in the engineering computational electromagnetics in the literature [36]. These methods vary from each other depending upon the requirement of the implicit solution, the use of large matrices or the need for higher order Maxwell's equations solutions, etc. FETD also allows different mesh resolutions (dense and coarse) simultaneously in the same computational domain. For FETD to be an alternative of FDTD it should possess most of the advantages of FDTD and be able to discretise the computational domain efficiently, especially for the structures with curved geometry.

2.4 Summary

In summary, the Maxwell's equations in the differential and integral forms have been introduced. Maxwell's equations are generally used as a starting point for solving the computational domain when using different numerical approaches such as finite differences, finite elements, boundary integral, surface integral, and others. Finite difference time domain method has been extensively used by the researchers in the last decades for studying the propagation of the electromagnetic light. A rectangular mesh is used in the finite difference time domain method which makes the simulation computationally expensive. A finite element time domain method can be used as an alternative to the finite difference time domain method. A triangular or curvilinear mesh can be used with the finite element time domain method for more accurate representation of complex structures.

Chapter 3

Equations for Two-dimensional Structures

The differential form of Maxwell's equations was chosen for the derivation of the governing equations for two-dimensional structures as given below:

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu} \nabla \times \mathbf{E} \tag{3.1}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon} \nabla \times \mathbf{H} \tag{3.2}$$

where $\mathbf{H} = \hat{x}H_x + \hat{y}H_y + \hat{z}H_z$, $\mathbf{E} = \hat{x}E_x + \hat{y}E_y + \hat{z}E_z$, μ and ε are the vector magnetic field, vector electric field, permeability and permittivity of the medium, respectively. $\hat{x}$, $\hat{y}$, and $\hat{z}$ are the unit vectors in the x, y, and z directions, respectively.

The partial differential operator, ∇, is given by:

$$\nabla = \hat{x}\,\frac{\partial}{\partial x} + \hat{y}\,\frac{\partial}{\partial y} + \hat{z}\,\frac{\partial}{\partial z} \tag{3.3}$$

Equations for Two-dimensional Structures

For the two-dimensional structure, the propagation of light is considered in the x-y plane, hence $\frac{\partial}{\partial z}$ becomes equal to zero. Equations 3.1 and 3.2 can now be expanded as:

$$\frac{d\mathbf{H}}{dt} = -\frac{1}{\mu}\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ E_x & E_y & E_z \end{bmatrix} = -\frac{1}{\mu}\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \\ E_x & E_y & E_z \end{bmatrix} \tag{3.4}$$

$$\frac{d\mathbf{E}}{dt} = \frac{1}{\varepsilon}\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ H_x & H_y & H_z \end{bmatrix} = \frac{1}{\varepsilon}\begin{bmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \\ H_x & H_y & H_z \end{bmatrix} \tag{3.5}$$

Equations 3.4 and 3.5 are used to obtain equations for Transverse Electric (TE) and Transverse Magnetic (TM) propagation modes.

For TE propagation, there is no electric field in the direction of propagation. The TE propagation equations are given by:

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} \tag{3.6}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \tag{3.7}$$

$$\frac{dE_z}{dt} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) \tag{3.8}$$

For TM propagation, there is no magnetic field in the direction of propagation. The TM propagation equations are given by:

$$\frac{dE_x}{dt} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} \tag{3.9}$$

$$\frac{dE_y}{dt} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} \tag{3.10}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right) \tag{3.11}$$

The computational domain is initially discretised to solve the governing Eqs. 3.6–3.11. The electric (**E**) and magnetic (**H**) field components in these equations are functions of both space and time. The right hand side of these equations is used to calculate the evolution of field components in space, while the left hand side describes the time evolution of the fields. To discretise the computational domain in both space and time, nodal elements are used. Space evolution is calculated at different spatial nodes of the nodal elements at a fixed time, while time evolution is obtained at different time intervals at fixed spatial nodes.

3.1 Space Discretisation

Among the different numerical approaches reported in the literature, the finite element method (FEM) has established itself as a powerful method for electromagnetic problems [37, 38]. To employ FEM the cross-section of the optical waveguide was suitably divided into a number of subdomains or nodal elements [39]. One-, two-, or three-dimensional elements can be used for finite element analyses. For two-dimensional simulations the simplest case is to use triangular elements where the trial function within each of these elements was approximated using first order polynomials.

3.1.1 Shape Functions

The continuous field function $\phi\ (x,y)$ in the problem domain may be replaced by a set of discrete values ($\phi_p,\ p = 1,2,3,\ldots,m$), where m is the total number of nodes. The function is continuous across adjacent triangular elements. ϕ is interpolated continuously inside each first order triangle and can be achieved by introducing the nodal shape function, $N_i(x,y)$. The field inside an element, $\phi_e(x,y)$ can be written as [15]:

$$\phi_e(x,y) = \sum_{i=1}^{3} N_i(x,y)\ \cdot \phi_i \tag{3.12}$$

where ϕ_i are the nodal field values. The matrix form of Eq. 3.12 is defined as follows:

$$\phi_e(x,y) = \begin{bmatrix} N_1 & N_2 & N_3 \end{bmatrix} \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{Bmatrix} \tag{3.13}$$

where $[N]$ is the shape function matrix and $\{\phi_e\}$ corresponds to the element nodal field values. Linear shape functions with first order polynomials were considered to describe the spatial variation of the fields inside an element. First order triangular elements use a first-degree polynomial $(a+bx+cy)$ over each element. The element shape function $\{N\} \equiv [N]^T$ can be written as:

$$\{N\} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \frac{1}{2A_e} \begin{bmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \tag{3.14}$$

where T denotes the transpose, A_e denotes the area of the triangular element and x_1, x_2, x_3, y_1, y_2, and y_3 are the x,y coordinates of three nodes of the triangular nodal element, respectively. $\{N\}$ can also be written as:

$$\{N\} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} = \begin{bmatrix} a_1 & b_1 x & c_1 y \\ a_2 & b_2 x & c_2 y \\ a_3 & b_3 x & c_3 y \end{bmatrix} \tag{3.15}$$

Comparing Eqs. 3.14 and 3.15, the coefficients a_i, b_i, c_i are calculated as:

$$a_1 = \frac{x_2 y_3 - x_3 y_2}{2A_e} \tag{3.16}$$

$$b_1 = \frac{y_2 - y_3}{2A_e} \tag{3.17}$$

$$c_1 = \frac{x_3 - x_2}{2A_e} \tag{3.18}$$

Similarly, a_2, b_2, c_2, a_3, b_3, c_3 can be calculated using a cyclic exchange of $1 \rightarrow 2 \rightarrow 3$ in Eqs. 3.16 – 3.18.

Consider the typical point P in the triangular element in Fig. 3.1 with vertices 1, 2, and 3. N_1, N_2, and N_3 can be given by:

$$N_1 = \frac{area\ of\ the\ triangle\ P12}{area\ of\ the\ triangle\ 123}$$

$$N_2 = \frac{area\ of\ the\ triangle\ P23}{area\ of\ the\ triangle\ 123}$$

$$N_3 = \frac{area\ of\ the\ triangle\ P31}{area\ of\ the\ triangle\ 123}$$

and the area is given by:

$$N_1 + N_2 + N_3 = 1$$

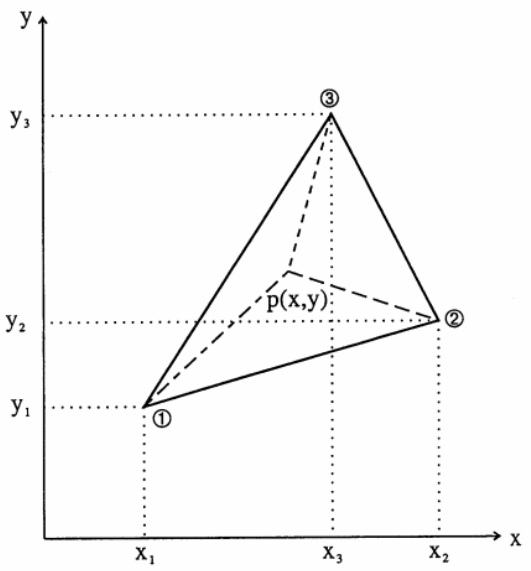


Fig. 3.1 Coordinates and node numbers of a typical first order triangular element.

It should be noted that the shape function of any order can be incorporated using the proposed method. However, linear elements are chosen for the space discretisation as they require the least computation time. More computer memory space would be required to store higher order elements in comparison to simple linear elements.

3.1.2 Two-dimensional Mesh

The accuracy of a finite element based code mostly depends on how efficiently the mesh discretises the computational domain. Many commercial and open source software packages are available for mesh generation including Pointwise, Gmsh [40], Netgen [41], Tetgen [42], Meshlab [43].

In this research work an open source software, Gmsh, is used to generate the mesh. Gmsh is a finite element mesh generator capable of generating 2D and 3D meshes using a built-in CAD engine. It is a fast and user-friendly interactive software tool that can easily create geometries and meshes with parametric input and advanced visualization capabilities. Gmsh's scripting language and its in-built graphical user interface can be used to generate

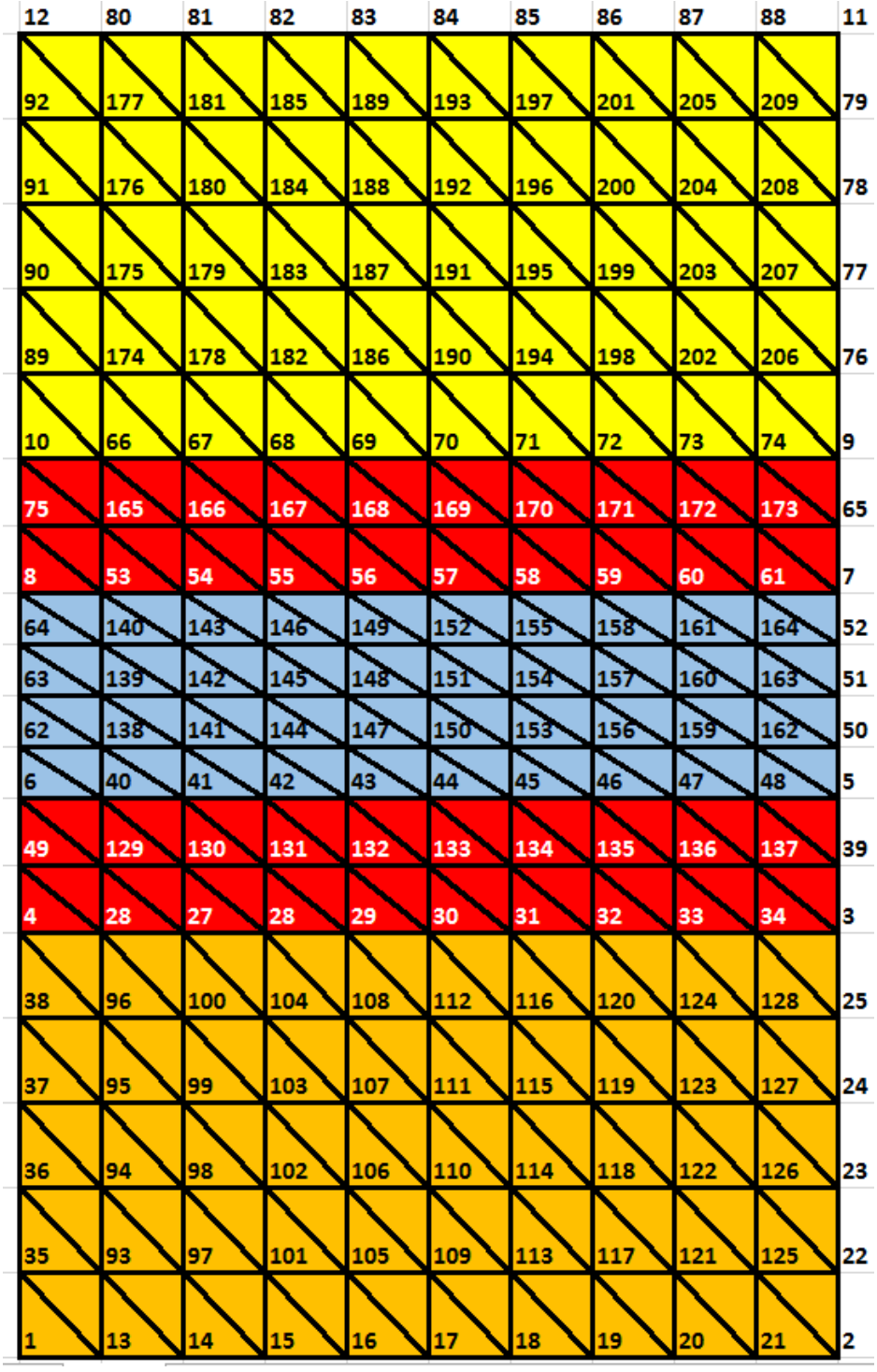


Fig. 3.2 Structured irregular mesh arrangement using GMSH software.

a mesh structure for any kind of design specifications. Mesh coordinates and elements generated using open source or commercial software can be taken as an initial point or input for the proposed FETD code.

Figure 3.2 shows a structured irregular type mesh, generated using Gmsh, for a directional coupler design. Layers or meshes on extreme ends in the vertical direction represent the substrate (lower layer) and cladding (top layer). The centre layer separates the two cores of the directional coupler. The remaining two layers are the core layers that are used to propagate the electromagnetic wave. In an irregular mesh arrangement, element dimensions may vary as can be seen from Fig. 3.2 that different layers can have different vertical lengths of an element. If all the elements have similar dimensions, it would be a regular mesh arrangement.

Generally, more mesh elements having smaller element sizes are used in the core layers for a structured irregular mesh arrangement, while the lesser elements are reserved for the substrate and cladding layers. As most of the light (or power) travels through the core layers, it is logical to use dense mesh in the core layer in comparison to the substrate and cladding layers. This type of mesh arrangement helps by saving computation resources in comparison to the structured regular mesh arrangement.

3.1.3 Spatial Meshes - First and Second Mesh

Two-dimensional structures with right-angled triangular mesh elements arrangement have been considered to discretise the computational domain. This initial triangular arrangement of mesh, as shown in Fig. 3.3a [10] is termed as the First Mesh. The triangular element node numbers and their corresponding (x, y) coordinate values were generated using the open source mesh generator software, Gmsh. For TE propagation, the E_z field component values were stored at the nodes of the first mesh elements. Once the electric field component values were stored at each node, the values of magnetic field components (H_x and H_y) corresponding

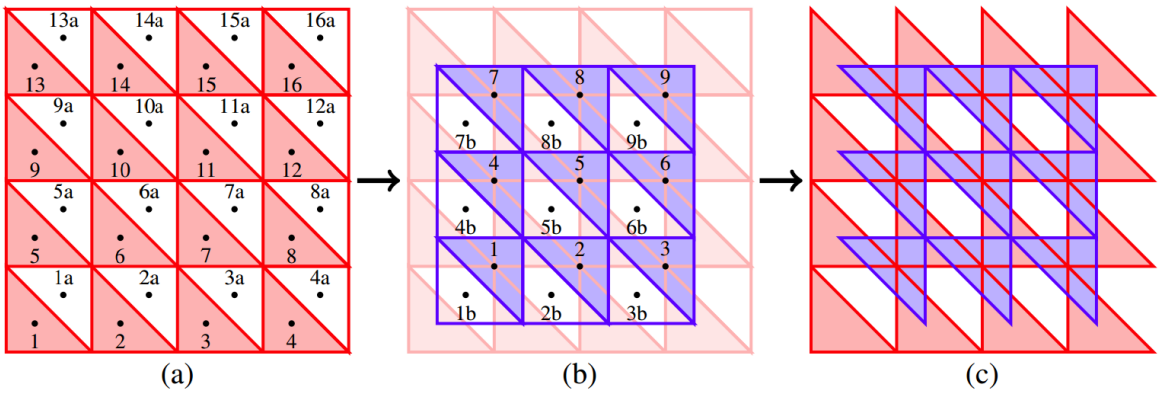


Fig. 3.3 (a) First mesh generation with triangular elements, (b) Generating second mesh by connecting the centroids of the first mesh, (c) Discarding unwanted elements from both meshes to save the computation time.

to each nodal element of the first mesh were calculated. Each triangular element of the first mesh (having E_z value stored at each of the three nodes) generates only one future value of both H_x and H_y field component. These generated H_x and H_y values were subsequently stored at the centroid of the same nodal element from which it was generated, shown by black dots in Fig. 3.3a. Another triangular mesh is required which is constructed using the centroids of the first mesh elements, as shown in Fig. 3.3b [10]. This new mesh generated from the centroids of the first mesh can be termed as the second mesh. Next value of the E_z field component were then generated using obtained H_x and H_y fields. This newly obtained value of E_z is then stored at the nodes of the first mesh by replacing the previously stored values provided that each element of the second mesh must surround one of the nodes of the first mesh.

For example, the centroids of the first mesh elements numbered 2, 5 and 6 (in Fig. 3.3a) become the three nodes of the element number 1 (in Fig. 3.3b) of the second mesh (shaded in light blue). It can be seen that the centroids of elements numbered 1, 2, 3, . . . , 9 (in Fig. 3.3b) of the second mesh coincide with the nodal coordinates of the first mesh. Half of the first and second mesh elements shown by white colours in Figs. 3.3a and 3.3b can be discarded to save the computation time. Although, half of the nodal elements are removed in

the generated code technique, none of the nodes are removed, as can be seen in Fig 3.3c [10]. All of the three nodes associated with those triangles are still updated during calculation at each time-step.

3.2 Time Discretisation

The computation domain has been spatially discretised and now the time axis discretisation is obtained as:

$$\Psi = \sum_{j=1}^{P} Q_j \ \psi^j \tag{3.19}$$

where Ψ can be any field component (H_x, H_y, H_z, E_x, E_y, E_z) inside the element, and ψ^j is the field component at the j^{th} time node, and P is the number of time nodes. For linear elements, P is 2. Here, Q_j is the shape function for the j^{th} time node, and for the linear shape function it is given as:

$$Q_j = p_j t + q_j \tag{3.20}$$

where p_j, q_j are the coefficients of the line passing through the nodes of the time element.

3.2.1 Time Meshing System

linear elements with two nodes has been considered to achieve the time discetisation. Higher order time elements can also be considered, but it increases the computation resources and the time required for the simulation.

The time discretisation is achieved by considering two different time meshes, **M** and **N**, associated with the first and second mesh, respectively, as shown in Fig. 3.4 [10]. At

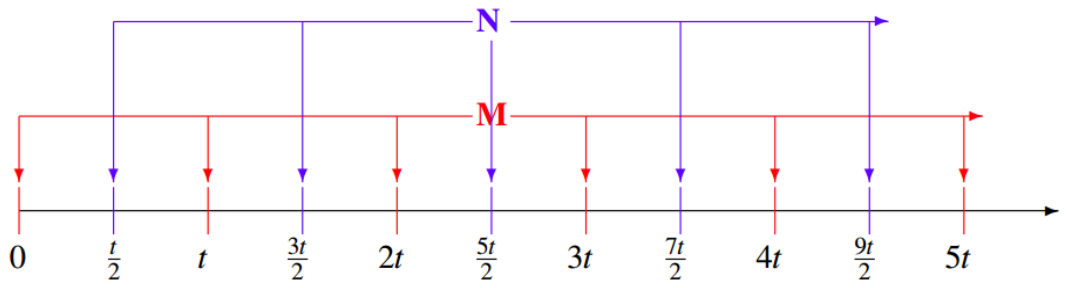


Fig. 3.4 Time mesh system with equal time spacing.

time, t = 0 (time mesh - **M**), electric field component values are stored at the nodes of the first mesh. From the stored electric field values, the magnetic field component values are calculated at the next time-step, i.e. time = t/2 (time mesh - **N**), and then again future values of electric field components are obtained at the next time-step, i.e. time = t (time mesh - **M**), etc. Therefore, the electric and magnetic field components cannot be calculated at the same time node. The time-step size for the calculation of the magnetic field components from electric field components and vice versa is considered to be of the same duration, i.e. t/2. For TE propagation, the E_z field components were calculated at 0, t, 2t, ..., while H_x and H_y field components were calculated at t/2, 3t/2, 5t/2, ... Similarly, time discretisation can be obtained for TM propagation.

Using Eqs. 3.12 and 3.19 discretised versions (employing spatial and time meshing systems) of Eqs. 3.6–3.11 for TE and TM propagation are as follows:

TE Propagation

$$h_x^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[-\frac{1}{\mu} \sum_{i=1}^{3} \frac{\partial N_i}{\partial y} e_{zi}^{(n)} - \frac{dQ_1}{dt} h_x^{(n-1)} \right] \tag{3.21}$$

$$h_y^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[\frac{1}{\mu} \sum_{i=1}^{3} \frac{\partial N_i}{\partial x} e_{zi}^{(n)} - \frac{dQ_1}{dt} h_y^{(n-1)} \right] \tag{3.22}$$

$$e_z^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[\frac{1}{\varepsilon} \left(\sum_{i=1}^{3} \frac{\partial N_i}{\partial x} h_{yi}^{(n)} - \sum_{i=1}^{3} \frac{\partial N_i}{\partial y} h_{xi}^{(n)} \right) - \frac{dQ_1}{dt} e_z^{(n-1)} \right] \tag{3.23}$$

TM Propagation

$$e_x^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[\frac{1}{\varepsilon} \sum_{i=1}^{3} \frac{\partial N_i}{\partial y} h_{zi}^{(n)} - \frac{dQ_1}{dt} e_x^{(n-1)} \right] \tag{3.24}$$

$$e_y^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[-\frac{1}{\varepsilon} \sum_{i=1}^{3} \frac{\partial N_i}{\partial x} h_{zi}^{(n)} - \frac{dQ_1}{dt} e_y^{(n-1)} \right] \tag{3.25}$$

$$h_z^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}} \left[-\frac{1}{\mu} \left(\sum_{i=1}^{3} \frac{\partial N_i}{\partial y} e_{yi}^{(n)} - \sum_{i=1}^{3} \frac{\partial N_i}{\partial y} e_{xi}^{(n)} \right) - \frac{dQ_1}{dt} h_z^{(n-1)} \right] \tag{3.26}$$

where $n+1, n$ and $n-1$ denotes the component's respective future, current and the past value.

3.3 Perfectly Matched Layer (PML) Boundary

A perfectly matched layer (PML) is an artificial absorbing layer that is most commonly used to truncate the computational domain in the numerical methods [33, 44]. This layer strongly absorbs outgoing waves from the interior of a computational region without reflecting them back into the interior [45].

PML was originally formulated by Berenger in 1994 [29] for solving unbounded electromagnetic problems with the finite-difference time-domain method. The PML boundary conditions have been effectively used in the literature for applications in optical wave propagation [46–50]. Several reformulations of PML has been reported in the past, and implemented in different ways including coordinate stretching, convolution or uniaxial PML, etc [51]. However, small numerical reflections may appear (which disappear with increased numerical resolution) when the wave equation is discretised for simulation using a computer.

The partial differential operator, ∇, has to be modified to implement any kind of PMLs. For a Cartesian coordinate system implementation, PML can be classified into three different categories: X axis PML, Y axis PML, and Corner PML, as shown in Fig. 3.5.

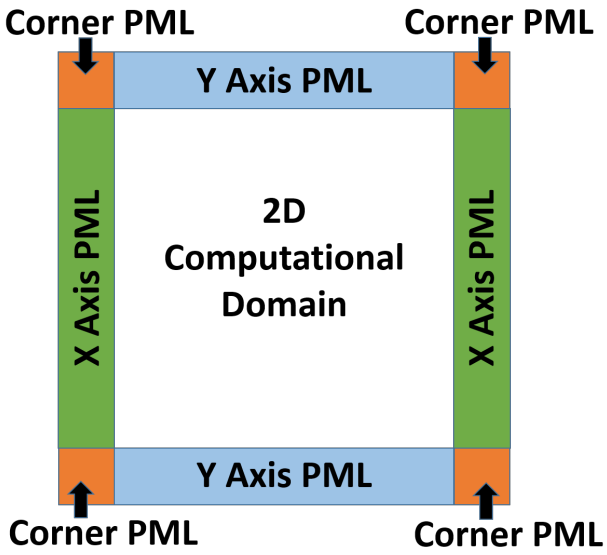


Fig. 3.5 X axis, Y axis, and corner PML in a general 2D computational domain.

3.3.1 X Axis PML

The X axis PML absorbs any wave having a X component value moving towards the computational boundary. The modified partial differential operator is given by:

$$\tilde{\nabla}^x = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{3.27}$$

where σ_x is the function of x. Any X component while using Eq. 3.27 is affected by the operator, but all other components remain unchanged. Various PML equations used are shown in Appendix A.1.

3.3.2 Y Axis PML

The Y axis PML absorbs any wave having a Y component value moving towards the computational boundary. Likewise, the modified differential operator for the Y axis PML is as follows:

$$\tilde{\nabla}^y = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z} \tag{3.28}$$

where σ_y is the function of y. Any Y component while using Eq. 3.28, is affected by the operator, but all other components remain unchanged. Various PML equations used are shown in Appendix A.2.

3.3.3 Corner PML

Corner PML is used only at the corners of the computational domain to absorb waves in both the directions. Likewise, the modified differential operator for the corner PML is as follows:

$$\tilde{\nabla}^{xy} = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z} \tag{3.29}$$

where σ_x, σ_y are the functions of x and y, respectively. Various PML equations used are shown in Appendix A.3.

3.4 Summary

In summary, mesh discretisation equations in the space and time for the two dimensional structures have been described. Open source software, Gmsh, is used to generate the mesh. A first mesh is generated using Gmsh and then a second mesh is generated from the centroids of the first mesh. Perfectly matched layers are used at the boundaries of the computational domain to reduce the unwanted reflections.

Chapter 4

FETD Method for Two-dimensional Structures

C++ programming language has been chosen to perform all the numerical simulations. C++ is a highly portable language with a rich function library. C++ is widely used for multi-device and multi-platform app development. It provides facilities for low-level memory manipulation along with object-oriented and generic programming features. OpenMP (Open Multi-Processing) was also employed for parallel programming. OpenMP is an application programming interface (API) which supports multi-platform shared memory multiprocessing programming in C++. It helps in reducing the simulation times. The output field values obtained during the FETD simulations were stored as a text file format. All the field plots presented in this current work were generated using MATLAB. The speed of light in free space is 3 X 10^8 ms^{-1}. However, in order to implement a dimensionless simulation, all speed values were normalised and hence lie in the range 0 to 1.

4.1 Free-Space Propagation

A continuous sinusoidal point E_z field source with a normalised wavelength of 1.55 μm (using the normalised light speed, c = 1) was placed at the centre of the 2D computational domain to perform the free-space propagation of the electromagnetic wave. For the free-space propagation, permeability (ε) was set 1. The computation domain is also surrounded by the perfectly matched layer (PML) before the boundaries. Presence of the PML helps in reducing the reflection of electromagnetic waves from the boundaries to the minimum or almost zero values.

The evolution of E_z, H_x and H_y field components in free-space are shown in Fig. 4.1. The mesh resolution was set to 20 per unit length. The red, blue and green colours correspond to the positive half, negative half, and zero amplitude of the corresponding field, respectively. E_z field is radially symmetric (expands uniformly in all directions) as shown in Fig. 4.1a. The amplitude of E_z field reduces with the distance from the point source in all directions. Figure 4.1b shows the decrease in E_z field amplitude along the horizontal (y) axis, as the electromagnetic wave from the point source travels away from the centre of the computation domain. The E_z point source subsequently generates other magnetic field components (H_x and H_y), as shown in Figs. 4.1c and 4.1e. Unlike E_z, H_x and H_y are not symmetric. H_x and H_y field extrema's are in the x and y directions, respectively with zero value along the y and x axes, as shown in Figs. 4.1c, and 4.1e. H_x and H_y field components also decay when the electromagnetic wave moves away from the centre of the computation domain towards the PML boundaries, as shown in Figs. 4.1d and 4.1f.

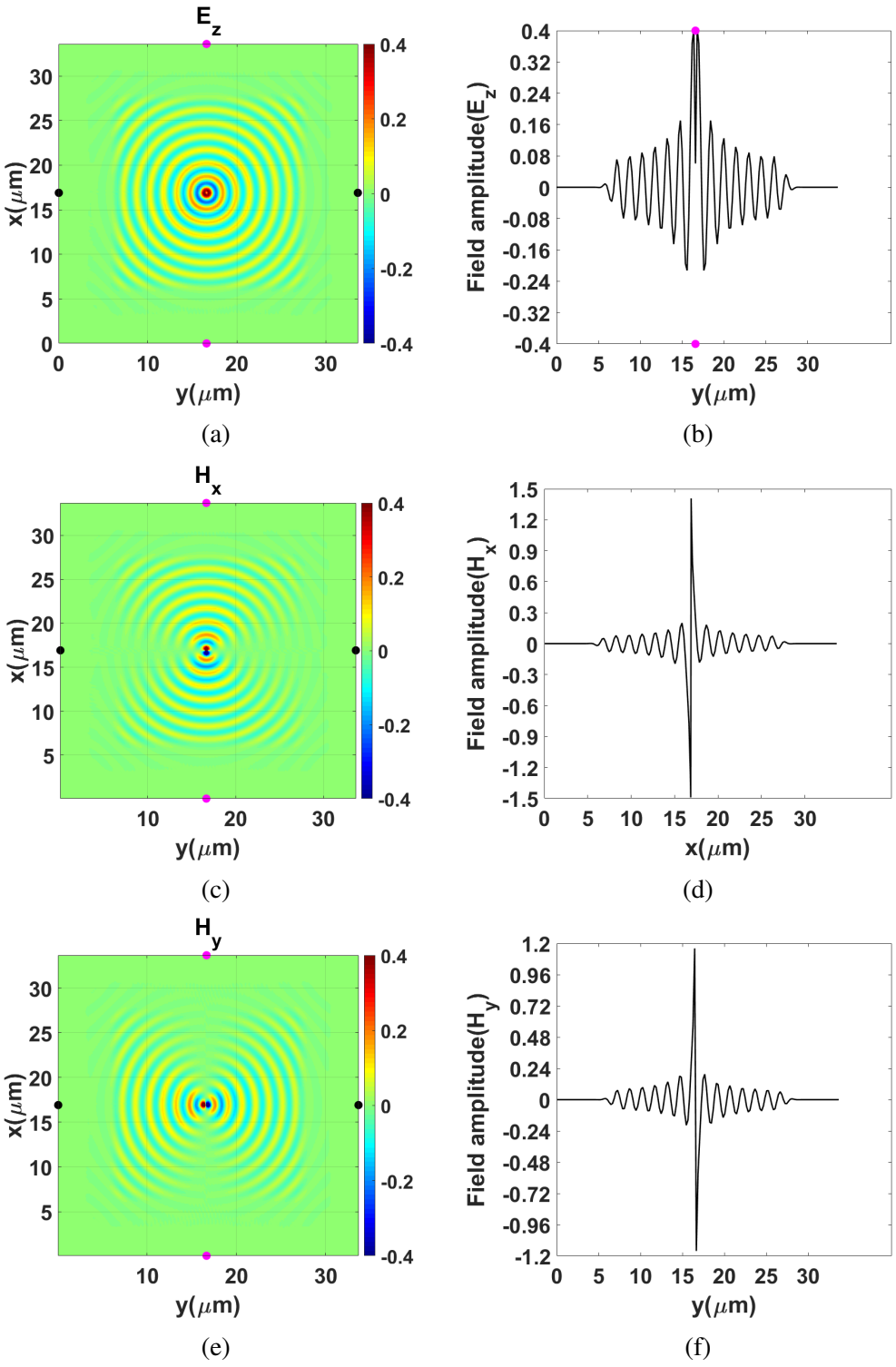


Fig. 4.1 (a) E_z field contour in free-space, (b) E_z field profile in free-space along horizontal axis at centre of computation domain, (c) H_x field contour in free-space, (d) H_x field profile in free-space along vertical axis at centre of computation domain, (e) H_y field contour in free-space, and (f) H_y field profile in free-space along horizontal axis at centre of computation domain.

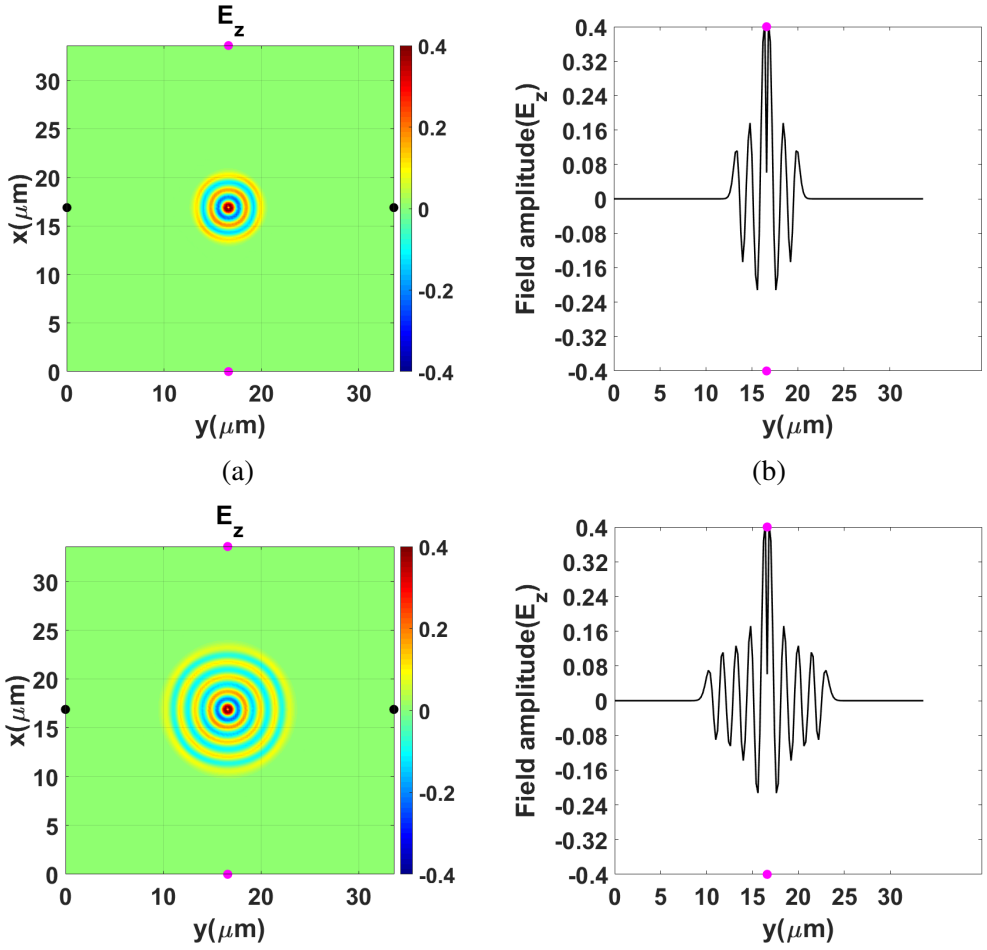


Fig. 4.2 (a) E_z field contour in free-space at time equals 3.86 μs, (b) E_z field profile in free-space along horizontal axis at centre of computation domain at time equals 3.86 μs, (c) E_z field contour in free-space at time equals 6.96 μs, and (d) E_z field profile in free-space along horizontal axis at centre of computation domain at time equals 6.96 μs.

From the FETD numerical simulations performed the speed of the electromagnetic wave was calculated as the ratio of the distance travelled by the wave to the time taken. Figure 4.2 shows the varying distances travelled by the wave at different times, considering the E_z point source in free-space at the centre of the computation domain. Figures 4.2a and 4.2c show the E_z field profile at time of 3.86 μs and 6.96 μs, respectively. It should be noted that at time of 6.96 μs field has expanded radially in comparison to when the time is 3.86 μs. Figures 4.2b

and 4.2d show the E_z field profile along the y-direction through the centre of the computation domain at time 3.86 μs and 6.96 μs, respectively.

From Figs. 4.2b and 4.2d the distance was calculated as the difference between the point values in the y-direction, where the wave terminates and the centre of the computation domain. This distance value was then divided by the respective time taken to obtain the speed of the electromagnetic wave. The ratio comes out to be 0.977, approximately 1, which is consistent with the initial assumption ($c = 1$) taken. Therefore, it justifies the effectiveness of the FETD solver.

4.2 Planar Waveguide

4.2.1 Point Source Waveguide

Here, a Silicon core planar waveguide excited using a E_z point source at a wavelength (λ) of 1.55 μm was considered. Cladding and substrate materials for this design were taken to be air having refractive index of 1.0 ($n_{air} = n_{cl}$ here). The refractive index of silicon ($n_{Si} = n_c$ here) was taken to be 3.44 at $\lambda = 1.55$ μm. The width of the silicon core layer (centre layer) was taken as 0.2 μm. The width of both substrate (the bottom layer) and the cladding (the top layer) were considered to be 4.2 μm each.

At first, a transverse-electric (TE) simulation was performed using a E_z point source. The E_z point source was placed at the coordinates (y, x) = (4.95, 4.3) μm, which is inside the silicon core layer. The placement of the E_z point source can also be visualized as the intersection point between an imaginary vertical line (formed by two pink colour dots) and horizontal line (formed by two black colour dots) in Figs. 4.3a, 4.3c, and 4.3e. The 2D computational domain is discretised with a mesh having resolution, $\Delta = 20$ per unit length and time resolution, $\Delta T = \Delta/2$. Regular mesh elements have been considered here, which means that the number of elements per unit length is same in all layers.

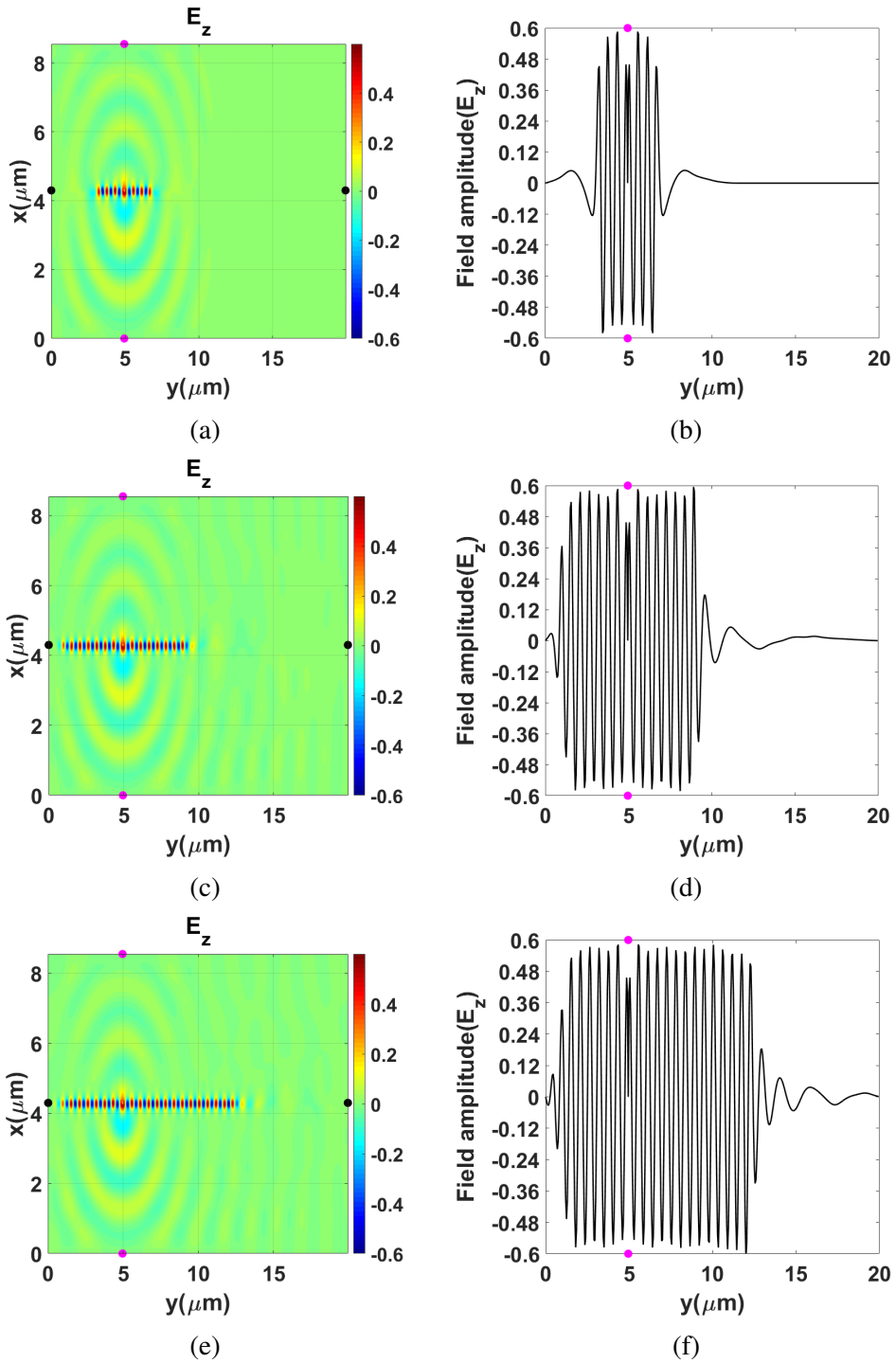


Fig. 4.3 E_z field profile for a dielectric planar waveguide (a) at time equals to 8.6025 μs, (b) along horizontal axis at centre of computation domain at time equals to 8.6025 μs, (c) at time equals to 17.205 μs, (d) along horizontal axis at centre of computation domain at time equals to 17.205 μs, (e) at time equals to 28.675 μs, and (f) along horizontal axis at centre of computation domain at time equals to 28.675 μs. Various parameters: λ is 1.55 μm, w_c is 0.2 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length.

Figures 4.3a, 4.3c, and 4.3e show the evaluation of the E_z field component at three different times, 8.6025 μs, 17.205 μs, and 28.675 μs, respectively. It can be observed that the E_z field is mostly confined inside the silicon core. This confinement of the electromagnetic wave in the dense silicon layer is based on the principle of total internal reflection (TIR) [52]. As the time increases from 8.6025 μs to 17.205 μs and then to 28.675 μs, the electromagnetic wave travels longer distances in the right-side direction. But the field amplitude decays to almost zero value in the left-side direction. This is due to the presence of PML layers around the boundaries, which makes the electromagnetic field die out more quickly on the left side in the simulations. However, some of the fields radiate away from the silicon waveguide. Therefore, besides its evolution to a propagating mode, other higher order modes were also generated and radiated subsequently. Figures 4.3b, 4.3d, and 4.3f show the E_z amplitude field profile along the y-direction through the centre of the computation domain at times 8.6025 μs, 17.205 μs, and 28.675 μs, respectively.

Table 4.1 Variations in the speed of wave (using FETD code) and effective index (n_{eff}) with respect to λ. The core width (w_c) is 0.2 μm, substrate width (w_s) and cladding width (w_{cl}) are 4.2 μm each. Resolution is taken as 20 i.e. number of elements per unit length.

Wavelength (μm)	**Resolution In Total**	**Effective Index** (n_{eff})	**Speed of Wave**
1.30	172	2.82402	0.553
1.40	172	2.76725	0.600
1.50	172	2.71157	0.653
1.55	172	2.68418	0.683
1.60	172	2.65712	0.700

The effect of operating wavelength on the electromagnetic wave propagation is summarised in Table 4.1. The speed of the electromagnetic wave through the silicon waveguide was calculated using the generated FETD code. The wavelength (λ) was increased from

1.30 μm to 1.60 μm, keeping the width of the core (w_c) fixed at 0.2 μm. The effective index (n_{eff}) at different λ for the design parameters was obtained using our in-house 1D-FEM code. As λ increased from 1.30 μm to 1.60 μm, n_{eff} decreases from 2.82402 to 2.65712, respectively. The speed of the electromagnetic wave increases with the decrease in n_{eff} and a similar trend is in evidence when the values obtained using the generated FETD code were used. The speed of the electromagnetic wave increases from 0.553 to 0.700 as n_{eff} decreases from 2.82402 to 2.65712 or as λ increases from 1.30 μm to 1.60 μm.

Table 4.2 Variations of the speed of the wave (using FETD code) and n_{eff} with respect to refractive index of core at λ = 1.55 μm. Core width (w_c) is 0.2 μm, substrate width (w_s) and cladding width (w_{cl}) are 4.2 μm each. Resolution is taken as 20.

Refractive Index	**Resolution In Total**	**Effective Index (n_{eff})**	**Speed of Wave**
2.00	172	1.38115	0.838
3.00	172	2.25056	0.709
3.44	172	2.68418	0.683
4.00	172	3.25277	0.567

Table 4.3 Variations of the speed of the wave (using FETD code) and n_{eff} with respect to w_c at λ = 1.55 μm. Substrate width (w_s) and cladding width (w_{cl}) are 4.2 μm each. Resolution is taken as 20.

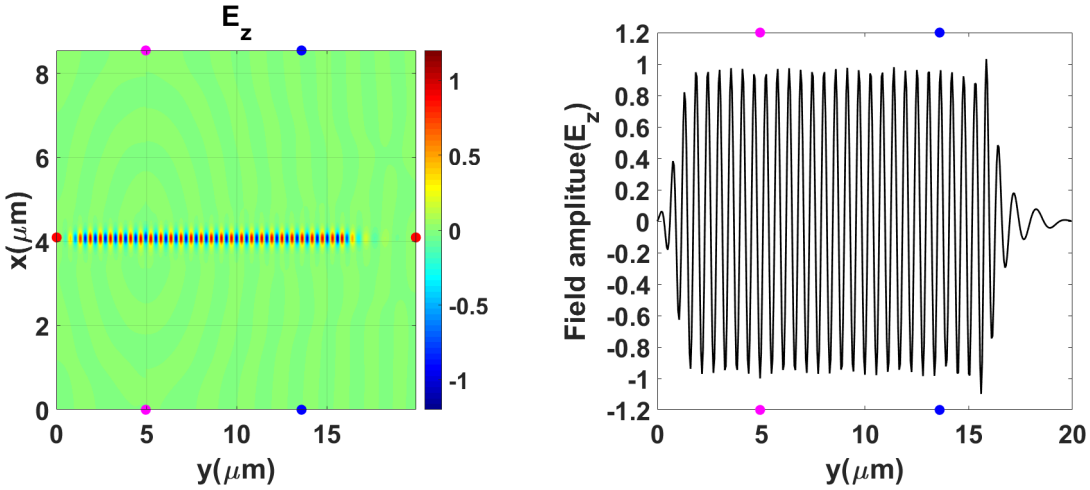
Core Width (μm)	**Resolution In Total**	**Effective Index (n_{eff})**	**Speed of Wave**
0.1	172	2.01518	0.787
0.2	172	2.68414	0.683
0.3	172	2.98192	0.503
0.4	172	3.13405	0.451

Increasing the refractive index of the core material, keeping λ and w_c fixed, increases the n_{eff}, as shown in Table 4.2. When the refractive index of the core was increased from 2.0 to 4.0, n_{eff} increases from 1.38115 to 3.25277, respectively. As n_{eff} increases the speed of the electromagnetic wave decreases through the waveguide from 0.838 to 0.567, as shown in Table 4.2.

Gradually increasing w_c while keeping λ and refractive index of core as fixed values, leads to an increase in the n_{eff} and subsequently decreases the propagation speed of the wave through the waveguide. A similar trend can be observed in Table 4.3 where n_{eff} increases from 2.01518 to 3.13405, and speed of the electromagnetic wave decreases from 0.787 to 0.451 when w_c increases from 0.1 μm to 0.4 μm, respectively.

4.2.2 Mode Source Waveguide

The E_z mode source (generated using in-house FEM code) was considered as an input source to the waveguide. The specifications of the waveguide dimensions were same as in the case of the point source considered in Section 4.2.1. Regular mesh type elements were considered in this case. The width of the silicon core layer (centre layer), width of substrate (bottom layer) and width of the cladding (top layer) were taken as 0.2 μm, 4.2 μm, and 4.2 μm, respectively. A normalised mode source (amplitude values of respective mode field was converted in the range 0–1) was used as an input at y of 4.95 μm along the vertical axis (in the x-direction). It can be observed from Fig. 4.4a that the propagating field is confined mostly to the core layer. In comparison to when the point source is taken as an input (in Section 4.2.1), there is less leakage field outside of the core when the mode source is taken as an input. Figure 4.4b shows the amplitude variation along the horizontal axis at x = 4.3 μm. The maximum amplitude values are close to 1 along the horizontal axis, which implies that there is very low loss of the propagating field in this case.



(a) (b)

Fig. 4.4 Using regular mesh arrangement: (a) E_z field profile for a dielectric planar waveguide with a E_z mode source, and (b) E_z field profile for a dielectric planar waveguide along the horizontal axis through the centre of the computation domain.

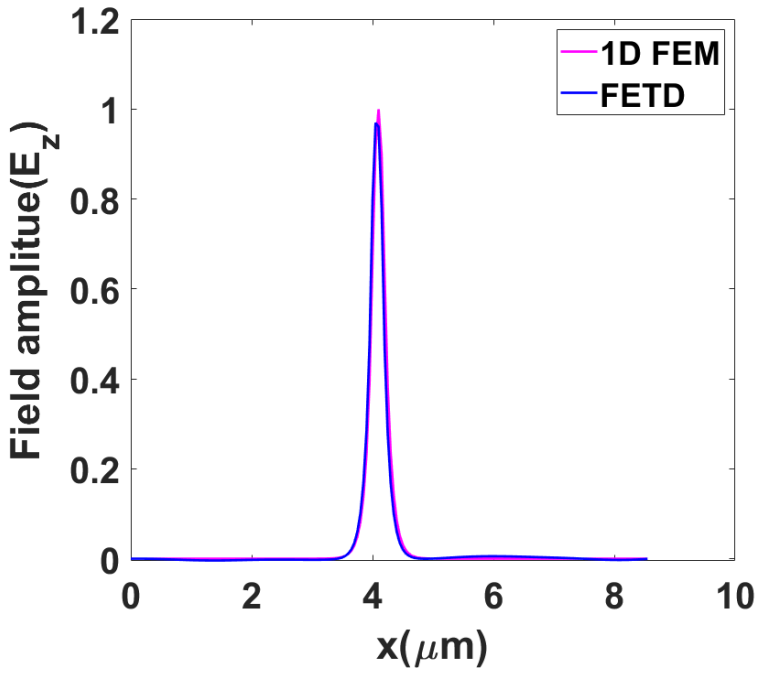


Fig. 4.5 Comparison of E_z field profile from the proposed FETD method and in-house FEM code having regular mesh arrangement.

A comparison of the normalised input mode source (generated from the FEM code) at y of 4.95 μm and the amplitude of the propagating electromagnetic wave (using FETD code output) along the x-axis at y = 14 μm (between the blue colour dots in Fig. 4.4a) is compared in Fig. 4.5. They both overlap each other which confirms that the wave is propagating efficiently (with very low loss) through the waveguide core.

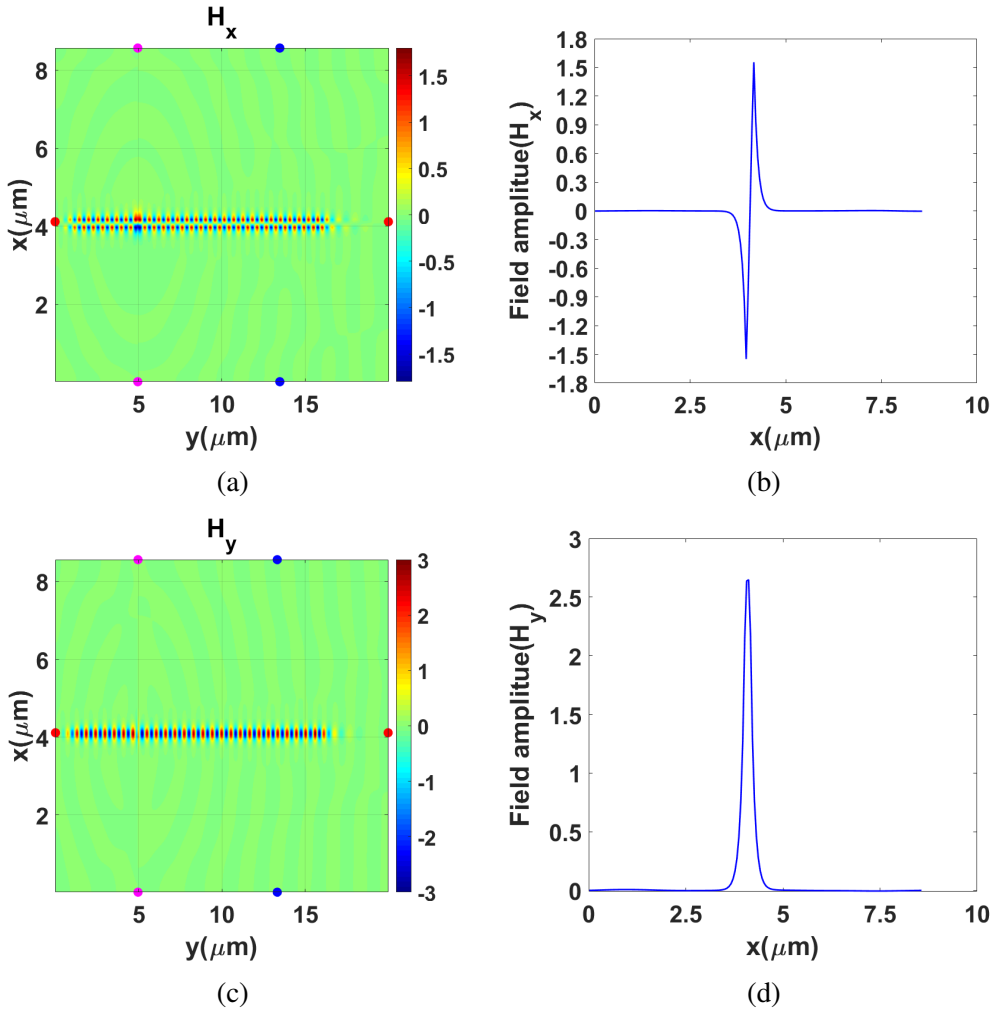
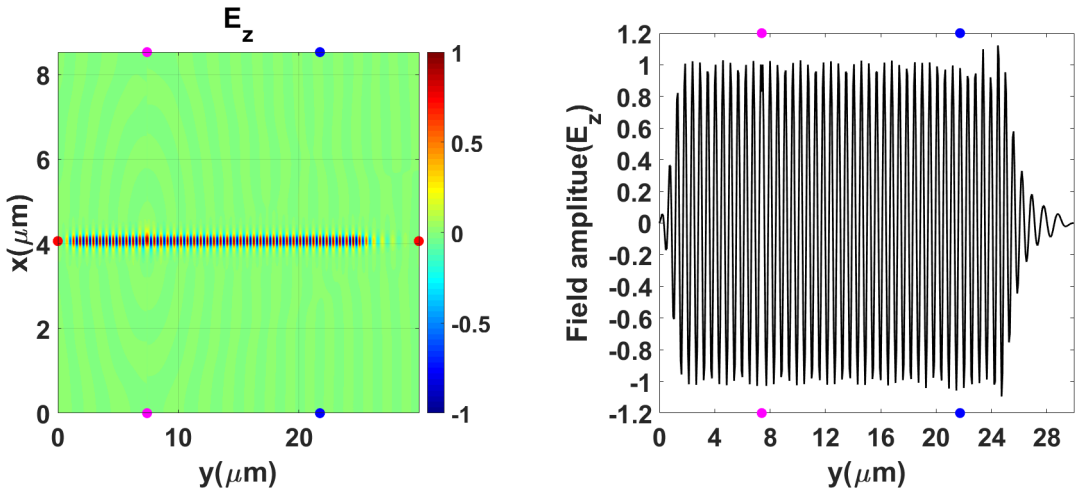


Fig. 4.6 Using regular mesh arrangement: (a) H_x field profile for a dielectric planar waveguide with a E_z mode source, (b) H_x field profile for a dielectric planar waveguide along vertical axis between blue colour dots, (c) H_y field profile for a dielectric planar waveguide with a E_z mode source, and (d) H_y field profile for a dielectric planar waveguide along vertical axis between blue colour dots.

The magnetic field components (H_x and H_y) generated from the input E_z mode source using Maxwell's equations are shown in Fig. 4.6.



(a) (b)

Fig. 4.7 Using irregular mesh arrangement: (a) E_z field profile for a dielectric planar waveguide with a E_z mode source, and (b) E_z field profile for a dielectric planar waveguide along the horizontal axis at the centre of the computation domain.

In Fig. 4.7, an irregular mesh has been considered with the mode source input for the same specifications of the design as used with the regular mesh arrangement. The purpose of using the irregular mesh elements arrangement is that a greater number of elements per unit length could be used in the dense core layer through which the wave propagates in comparison to the substrate and cladding layers. This may result in better accuracy of the model and it can be computationally more efficient in comparison to regular mesh arrangement.

The overlapping of the FEM input and FETD output at the far position (as shown in Fig. 4.8) confirms that the irregular mesh arrangement is also working accurately with the generated FETD code.

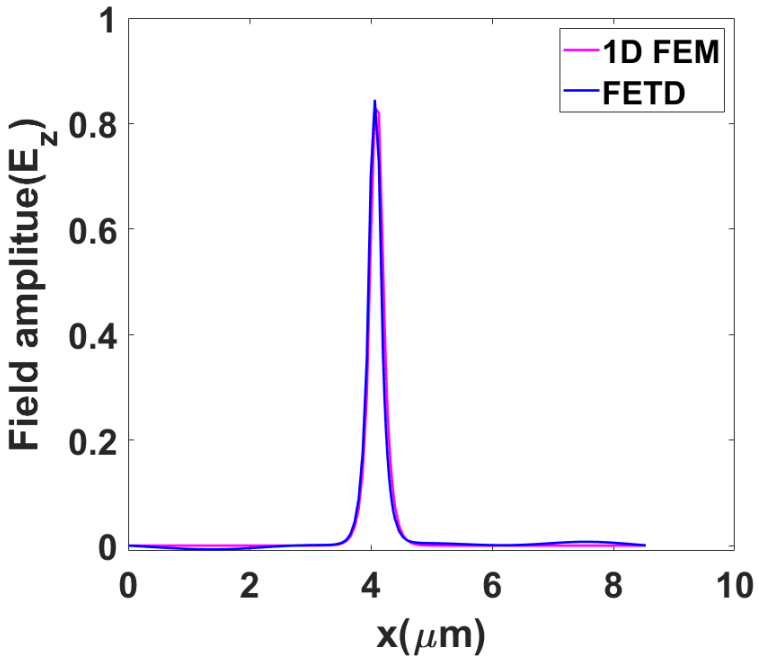
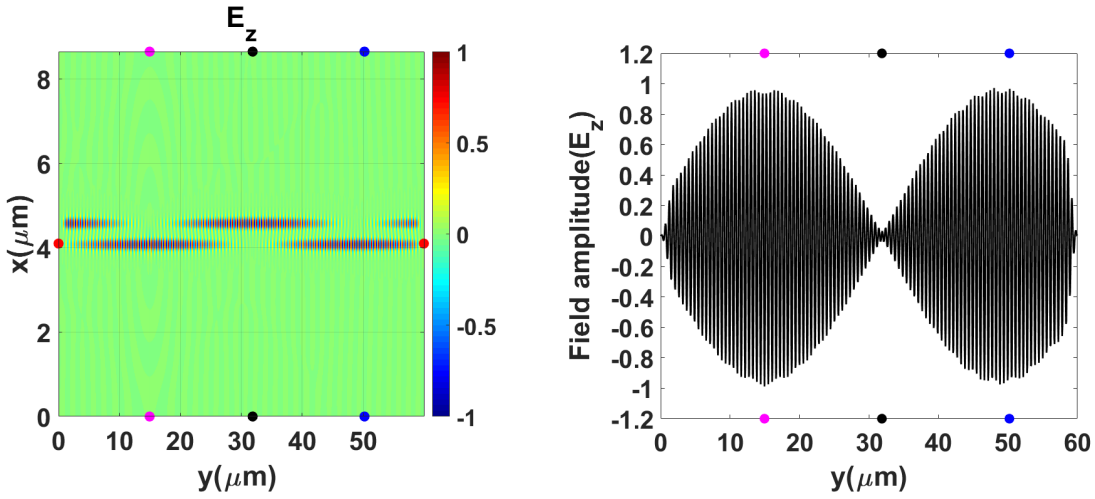


Fig. 4.8 Comparison of E_z field profile from the proposed FETD method and in-house FEM code having irregular mesh arrangement.

4.3 Directional Coupler

A directional coupler with two silicon cores separated by varying distances is considered with a mode source input. A mode source input generated from the FEM code was input to into the lower waveguide. A directional coupler couples light from one waveguide to another and then back to the original waveguide. The coupling length (L_c) of a directional coupler can be calculated as half the distance between two peak values of the field in any waveguide.

Figure 4.9a shows the coupling of E_z field from one waveguide to another. The widths of both of the silicon core layers were taken as 0.2 μm. Air is considered as the substrate and cladding material. The separation between the silicon waveguides was taken as 0.3 μm. The mode source input was generated using the FEM code which was fed at y = 15 μm along the vertical axis formed between two pink colour dots. Figure 4.9b shows the E_z field amplitude along the horizontal axis through the center of the lower waveguide. The



(a) (b)

Fig. 4.9 Using regular mesh arrangement with resolution of 20 per unit length: (a) E_z field profile for a directional coupler with a E_z mode source having waveguides separation equals to 0.3 μm, and (b) E_z field profile for a directional coupler along horizontal axis through the centre of lower waveguide.

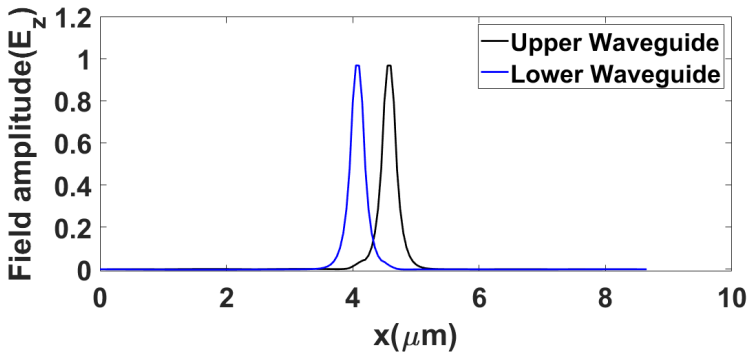
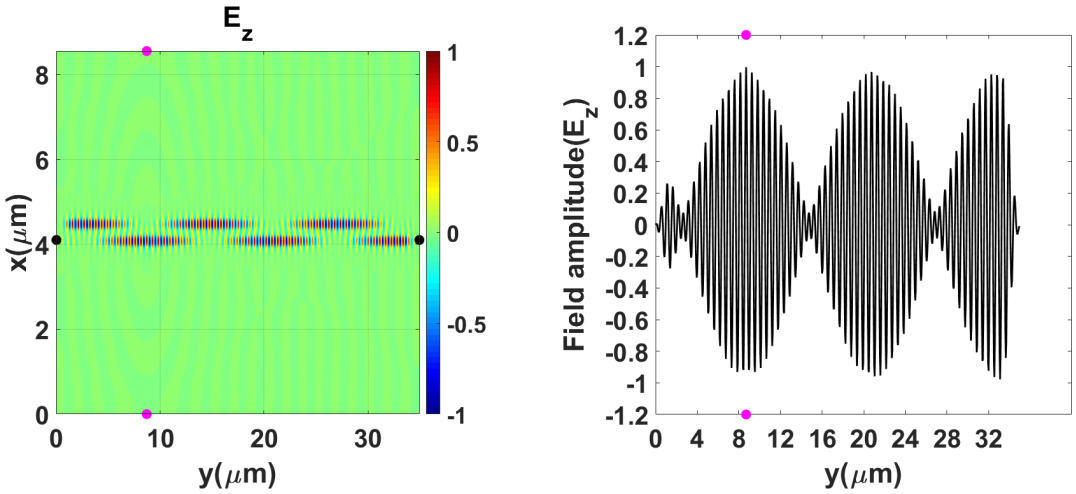


Fig. 4.10 Comparison of E_z field amplitude for a directional coupler along the vertical axis at two different y-positions (along vertical axes formed by two black colour dots and two blue colour dots in Fig. 4.9a), i.e. 32 μm and 50.25 μm.

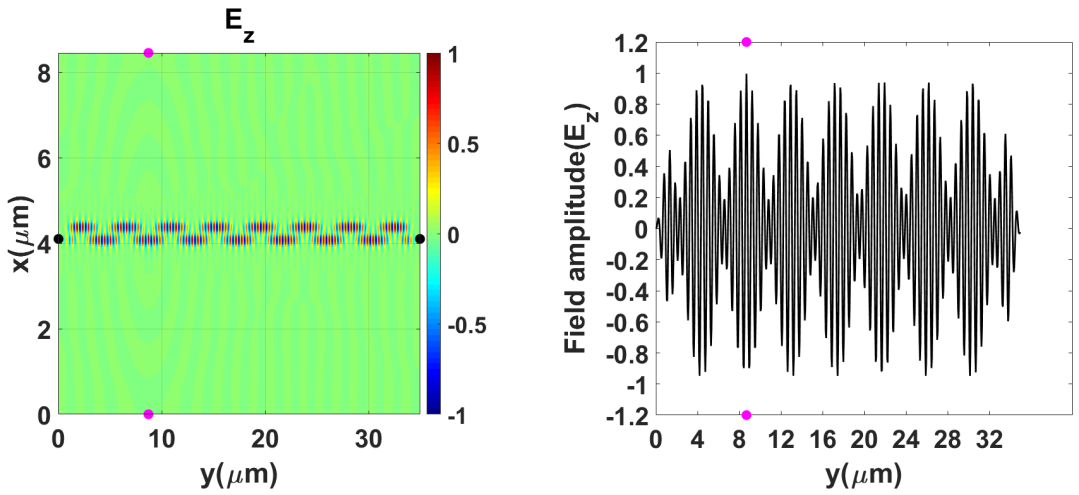
E_z field amplitude alternatively falls from peak to zero and rises to the peak again. The E_z field amplitude along the vertical axis at two different y-positions (along vertical axes formed by two black colour dots and two blue colour dots in Fig. 4.9a) is shown in Fig. 4.10. It shows that when the amplitude is maximum in one waveguide, it is approximately zero in the other waveguide and vice-versa. At y = 32 μm, most of the field is confined in the upper waveguide, while at y = 50.25 μm, most of the field has been transferred to the lower waveguide. The horizontal distance between the two waveguides when maximum field is transferred from one waveguide to another corresponds to the L_c of the directional coupler.



(a) (b)

Fig. 4.11 Using regular mesh arrangement with resolution of 20 per unit length: (a) E_z field profile for a directional coupler with a E_z mode source having waveguides separation equals to 0.2 μm, and (b) E_z field profile for a directional coupler along horizontal axis through the centre of lower waveguide.

Figures 4.11 and 4.12 show the coupling of an electromagnetic wave from one waveguide to another when the separation between the waveguides was 0.2 μm and 0.1 μm, respectively. It can be seen from Figs. 4.9b, 4.11b, and 4.12b that if the separation is reduced between the waveguides, the minimum field amplitude value does not reach zero and it still has some remaining amplitude value. This can be stated as the dip ratio. The dip ratio is defined as the ratio of maximum field amplitude near to the zero axis to the maximum field amplitude



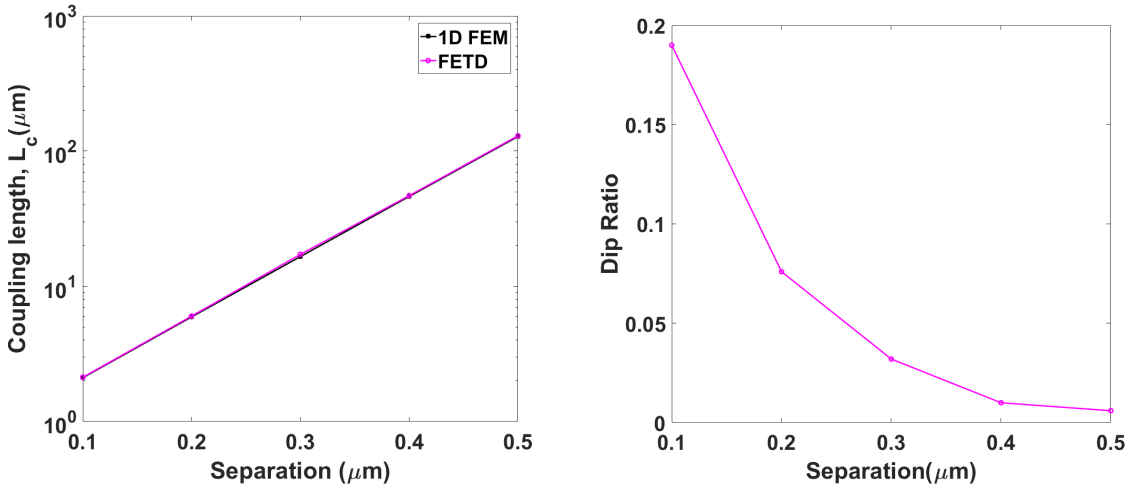
(a) (b)

Fig. 4.12 Using regular mesh arrangement with resolution of 20 per unit length: (a) E_z field profile for a directional coupler with a E_z mode source having waveguides separation equals to 0.1 μm, and (b) E_z field profile for a directional coupler along horizontal axis through the centre of lower waveguide.

near to the peak, as shown in Fig. 4.12b. It is well known that as the separation between the waveguides is reduced, complete power transfer between the waveguides is not possible, which increases the cross-talk.

Table 4.4 Comparison of coupling length (L_c) for TE propagation as calculated from FETD code and FEM code with varying separations between the waveguides cores. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length. Dip ratio variations are also shown with respect to separation between the waveguides cores.

Separation (μm)	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**% difference (L_c)**	**Dip Ratio**
0.1	2.10	2.12	0.943	0.190
0.2	5.94	6.02	1.328	0.076
0.3	16.59	17.25	3.826	0.032
0.4	46.23	46.9	1.428	0.010
0.5	128.32	129.95	1.025	0.006



(a) (b)

Fig. 4.13 (a) Comparison of coupling length (L_c) for TE propagation calculated from FETD code and FEM code on logarithmic scale, (b) Dip ratio variations shown with respect to separations between the waveguides cores. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length.

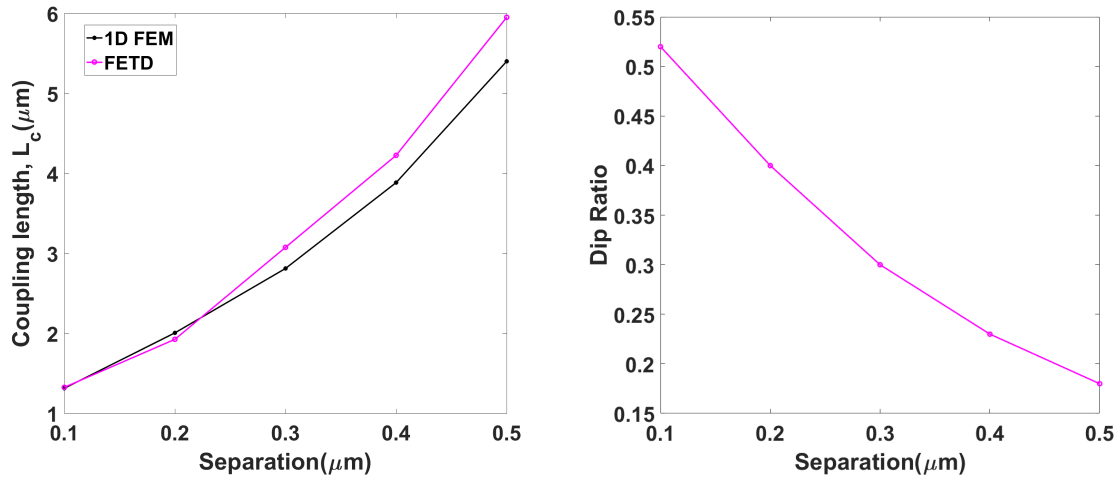
The L_c obtained from the FETD code was compared with the FEM code and summarised in Table 4.4. It can be seen that L_c matches for the FETD and FEM codes when separation was increased from 0.1 μm to 0.5 μm. The dip ratio value calculated using FETD code decreases from 0.190 to 0.006 when separation is increased from 0.1 μm to 0.5 μm. The resolution (number of mesh elements per unit length) can also be increased, which may lead to even closer results for L_c from FETD and FEM codes. However, increasing the resolution is at the cost of increased simulation times. Therefore trade-off is necessary between the accuracy of the results and the time taken to obtain a simulation result. The L_c shown on a logarithmic scale and the dip ratio variation with the separation for both FETD and FEM codes are plotted in Figs. 4.13a and 4.13b, respectively.

The L_c using FETD and FEM codes for TM propagation are compared in Table 4.5 and Fig. 4.14a. The L_c increases from 1.31 μm to 5.40 μm and 1.32 μm to 5.95 μm using FEM and FETD codes when separation increases from 0.1 μm to 0.5 μm, repectively. The L_c values obtained from both FETD and FEM codes are close. The dip ratio decreases with

separation for TM propagation, similar to TE propagation. Dip ratio falls from 0.52 to 0.18 when separation is increased from 0.1 μm to 0.5 μm, as shown in Fig. 4.14b.

Table 4.5 Comparison of coupling length (L_c) for TM propagation as calculated from FETD code and FEM code with varying separations between the waveguides cores. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length. Dip ratio variations are also shown with respect to separation between the waveguides cores.

Separation (μm)	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**% difference (L_c)**	**Dip Ratio**
0.1	1.31	1.32	0.757	0.52
0.2	2.00	1.92	4.166	0.40
0.3	2.81	3.07	8.469	0.30
0.4	3.88	4.22	8.056	0.23
0.5	5.40	5.95	9.243	0.18



(a) (b)

Fig. 4.14 (a) Comparison of coupling length (L_c) for TM propagation calculated from FETD code and FEM code, (b) Dip ratio variations shown with respect to separations between the cores. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length.

4.3.1 Regular Mesh - Varying Resolution

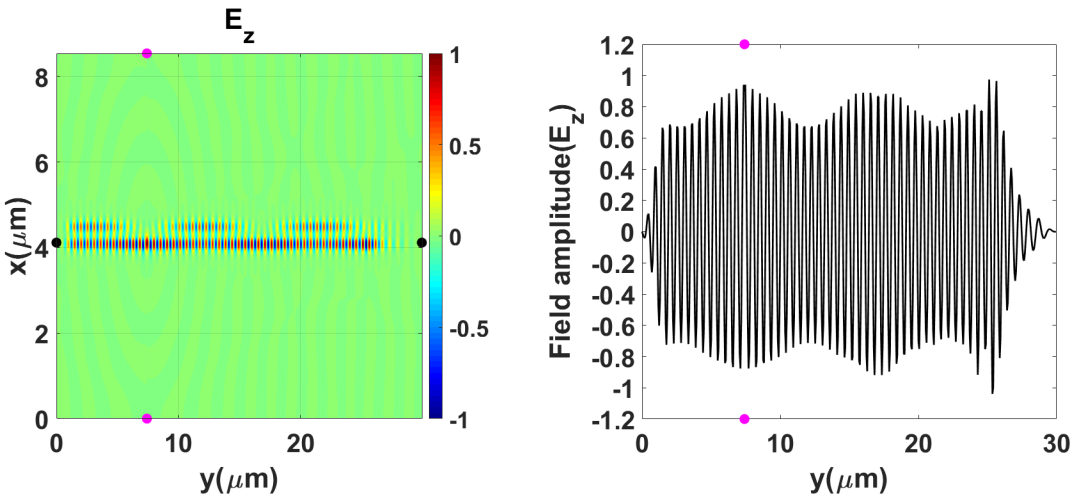
Table 4.6 Comparison of coupling length (L_c) for TE propagation calculated from FETD and FEM codes for varying resolutions. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0. Dip ratio variations are also shown.

Separation (μm)	**Resolution In Total**	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**% difference (L_c)**	**Dip Ratio**
0.2	172	5.94	6.02	1.328	0.076
0.2	344	6.01	6.13	1.957	0.058
0.2	688	6.03	6.17	2.269	0.052

The dip ratio variations for TE propagation with increased number of elements in all layers of the computation domain is compared in Table 4.6. A regular mesh arrangement is considered in this case which implies that the mesh elements in all layers have the same dimensions. It can be observed that the dip ratio decreases from 0.076 to 0.052 when the resolution is increased from 172 to 688. However, a denser mesh structure gives more accurate results, but also increases the simulation time.

4.3.2 Irregular Mesh - Varying Resolution

An irregular mesh arrangement was also employed for the directional coupler design. When considering irregular mesh, if the number of elements are nominal and aspect ratio (defined as the ratio of number of elements per unit length in core and substrate) is high, then the dip ratio does not fall to lower values in comparison to when the regular mesh arrangement is considered. It can be seen from Fig. 4.15a that the electromagnetic wave is not able to couple efficiently from the lower waveguide to upper waveguide. It can be observed that the lower waveguide still retains a large field amplitude at approximately y = 12 μm. However, ideally



(a) (b)

Fig. 4.15 Using irregular mesh arrangement: (a) E_z field profile for a directional coupler with a E_z mode source having waveguides separation equals to 0.2 μm, and (b) E_z field profile for a directional coupler along the horizontal axis through the centre of lower waveguide.

it should fall down to nearly zero at y = 12 μm. Consequently, the dip ratio is more if there is a high aspect ratio for irregular mesh type arrangement having nominal mesh elements.

The following mesh arrangements are proposed for the high aspect ratio condition:

1. Progressive mesh arrangement
2. Extra mesh elements arrangement

4.3.2.1 Progressive Mesh Arrangement

In a progressive mesh type arrangement, the element size is progressively increased (in geometric progression) towards the boundaries of the computational domain having minimum element dimension near to the interface of the core. It has been chosen to use a minimum element dimension of progressive mesh arrangement similar to the desired element size in the core layer.

The L_c and dip ratio are compared for the FETD code having a progressive type mesh arrangement with the FEM code in Table 4.7. The dip ratio decreases from 0.08 to 0.05 when

the total number of elements arranged in a progressive type mesh increases from 172 to 688, respectively.

Table 4.7 Comparison of coupling length (L_c) for TE propagation calculated from FETD and FEM codes for varying resolution using progressive type irregular mesh arrangement. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length. Dip ratio variations are also shown.

Separation (μm)	**Resolution In Total**	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**% difference (L_c)**	**Dip Ratio**
0.2	172	6.00	6.09	1.477	0.08
0.2	344	6.02	6.14	1.954	0.05
0.2	516	6.03	6.17	2.269	0.05
0.2	688	6.03	6.17	2.269	0.05

Table 4.8 Comparison of coupling length (L_c) for TE propagation calculated from FETD and FEM codes for varying resolution using progressive type irregular mesh arrangement for different combinations.

Separation (μm)	**Wavelength (μm)**	**Type**	**Resolution In Total**	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**Dip Ratio**
0.1	1.55	TE	320	2.15	2.16	0.12
0.1	1.25	TE	320	3.35	3.34	0.07
0.2	1.55	TE	344	6.02	6.14	0.05
0.3	1.25	TM	368	12.93	12.8	0.01

Other cases with random separations or wavelength or TE type, TM type or resolution are summarized in Table 4.8 for a progressive mesh arrangement in order to show the validity of the generated FETD code. L_c values compared from FETD and FEM codes are close for all the different cases, and dip ratio values are also nominal.

This shows that the problem of high dip ratio (when aspect ratio is high) can be overcome by using mesh elements whose dimensions increase progressively towards the boundaries.

4.3.2.2 Extra Mesh Elements Arrangement

In this type of arrangement, extra elements are introduced near to the interface. The dip ratio was compared with the increase in number of extra elements around the core-substrate and core-cladding interface. As shown in Table 4.9, the dip ratio falls from 0.670 to 0.075 to 0.050 when resolution was increased from 172 (0 extra element near to the interface) to 178 (6 extra elements near to the interface) to 220 (48 extra elements near to the interface), respectively. The more the number of extra elements, quicker the dip ratio converges to an optimum value.

Table 4.9 Comparison of coupling length (L_c) for TE propagation calculated from FETD and FEM codes for varying resolution using extra mesh elements type irregular mesh arrangement. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0, and mesh resolution is taken as 20 per unit length. Dip ratio variations are also shown.

Separation (μm)	**Resolution In Total**	**L_c (FEM) (μm)**	**L_c (FETD) (μm)**	**Dip Ratio**
0.2	172	6.00	5.95	0.670
0.2	174	6.00	5.95	0.140
0.2	176	6.00	5.95	0.100
0.2	178	6.00	5.95	0.075
0.2	184	6.00	5.95	0.050
0.2	196	6.00	5.95	0.050
0.2	220	6.00	5.95	0.050

Both the progressive and extra elements type irregular mesh arrangements were compared in terms of dip ratio and the results are shown in Table 4.10. The dip ratio falls from 0.08

to 0.05 and 0.67 to 0.05 when the resolution was increased from 172 to 196 for progressive and extra elements type mesh, respectively. It can be observed that the dip ratio reaches the optimum value quicker with progressive mesh arrangement type irregular mesh. Hence, it can be observed that progressive mesh arrangement is performing better than extra mesh elements arrangement.

Table 4.10 Comparison of dip ratio for TE propagation calculated from FETD and FEM codes for progressive mesh type and extra mesh elements type irregular mesh arrangement. Various parameters: λ is 1.55 μm, w_s and w_{cl} are 4.2 μm each, n_c is 3.44, n_{cl} is 1.0.

Separation (μm)	**Resolution In Total**	**Dip Ratio (Progressive Mesh)**	**Dip Ratio (Extra Elements Mesh)**
0.2	172	0.08	0.67
0.2	174	0.07	0.14
0.2	178	0.05	0.07
0.2	196	0.05	0.05

4.4 Summary

In summary, the proposed FETD has been validated by performing several benchmarking simulations, and comparing the results with in-house FEM code. The speed of the electromagnetic wave was calculated using the FETD code in the free-space propagation (which is equal to 0.977) which was found to be very close to the initial assumption value of 1.0. The normalised shapes and values of different field components e.g. E_z, H_x, H_y obtained using FETD code were compared with FEM code, which were also in a good agreement. Increasing the resolution of elements may increase the accuracy, but at the cost of increased simulation times. An irregular mesh with only increased resolution inside the core with generated FETD code can be used. This does not increase the simulation times significantly, unlike regular mesh arrangement. Different types of mesh arrangements including progressive type and

extra elements type mesh has been discussed. Hence, the FETD method has been used as an alternative of FDTD technique as it provides better meshing.

Chapter 5

Machine Learning Regression Approach to the Nanophotonic Waveguide Analyses

5.1 Introduction

Machine learning (ML) technology is being extensively used in many aspects of modern society: web searches, social networking, smartphones, bioinformatics, robotics, chatbots, and self-driving cars [53]. ML techniques are also used to classify or detect objects in images, speech to text conversion, pattern recognition, natural language processing, sentiment analysis and recommendations of products/movies for users based on their search preferences. ML algorithms can be trained to perform exceptionally well when it is difficult to analyze the underlying physics and mathematics of the problem [54]. ML algorithms extract patterns from the raw data provided during the training without being explicitly programmed. The learned patterns can be used to make predictions on some other data of interest. ML systems can be trained more efficiently when a large amount of data is present [55, 56].

Recently, research on the application of ML techniques for optical communication systems and nanophotonic devices has gained popularity. Several developments in ML over the past few years has motivated the researchers to explore its potential in the field of

photonics, including multimode fibres [5], power splitters [6], plasmonics [7], grating coupler [8], photonic crystals [9], metamaterials [57], label-free cell classification [58], molecular biosensing [59], optical communications [60, 61] and networking [62, 63].

Complex nanophotonic structures are being designed and fabricated to enable novel applications in optics and integrated photonics. Such nanostructures comprise a large number of parameters which needs to be optimised for efficient performance of the device and can be computationally expensive. For example, the finite-difference time-domain (FDTD) method may require several minutes to hours to analyze the optical transmission response of a single photonic device depending on its design. The ML approach offers a path for quick estimation of the optimised parameters for the design of complex nanostructures, which are critical for many sensing and integrated optics applications.

ML algorithms consider general function approximations to learn a complex mapping from the input to the output space. The most popular ML frameworks for building and training neural networks includes SciPy [64], Scikit-learn [65], Caffe [66], Keras [67], TensorFlow [68] and PyTorch [69]. PyTorch makes use of tensors for training neural networks along with strong graphical processing unit (GPU) acceleration. It provides separate modules to build a neural network and automatically calculates gradients for backpropagation [70] that are required during the training of a neural network. PyTorch appears to be more flexible with Python and NumPy/SciPy stack compared to TensorFlow and other frameworks, which allows easy usage of popular libraries and packages to write neural network layers in Python. Scikit-learn is another simple and efficient ML library used for data mining and data analysis. PyTorch and Scikit-learn numerical computing environment to handle the front-end modelling and COMSOL Multiphysics for the back-end data acquisition were used in the implementation. The commercial 2D FEM software such as COMSOL Multiphysics and Lumerical can provide the modal solution of any waveguide within a few minutes. However, a rigorous optimisation of the waveguide design parameters through parameter

sweep often becomes intensive for a modern workstation depending on the complexity of a design. In this chapter, an in-house developed ML-algorithm is reported as a stepping stone for the multi-parameter optimisation process where only the algorithm training (one time process) requires a few minutes of computational time to learn the features of similar types of waveguides.

Main concepts of ML related to integrated photonics applications are discussed in Section 5.2. In Section 5.3, results from ML algorithms using PyTorch and Scikit-learn with FEM results for commonly used silicon photonic waveguides and devices are compared, and finally the chapter is concluded in Section 5.4.

5.2 Neural Network Training

The most common form of machine learning is the supervised learning in which the training dataset consists of pairs of inputs and desired outputs, which are analyzed using ML algorithms to produce an inferred function. It is then used to obtain output values corresponding to any unknown input data samples. Supervised learning can be further categorised into a classification or regression problem, depending on whether the output variables have discrete or continuous values, respectively. In this chapter, the output predictions of different integrated photonics structures were considered as a regression problem.

5.2.1 Artificial Neural Network (ANN)

An ANN consists of a network of nodes, also called neurons. An ANN is a framework which is used to process complex data and it learns from the specific input data without being programmed using any task-specific rules. One of the commonly used ANN is the multilayer perceptron (MLP). An MLP consists of three or more layers. In Fig. 5.1, an MLP with four layers of nodes: an input layer, two hidden layers and an output layer is shown. These layers

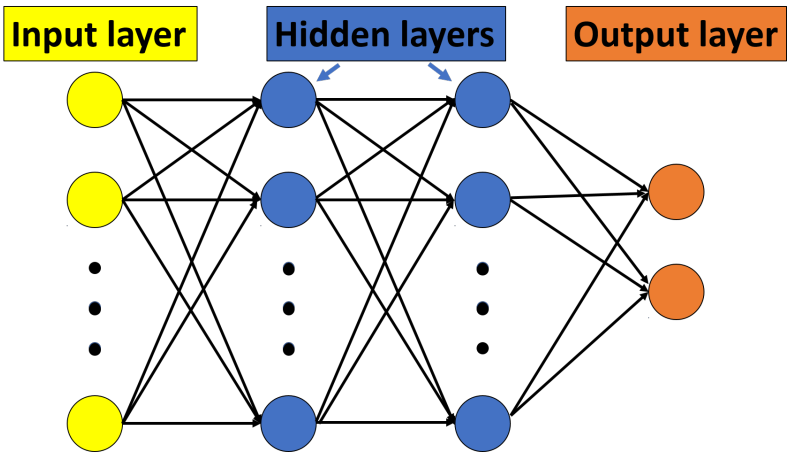


Fig. 5.1 General artificial neural network (ANN) representation, i.e. one input layer, two hidden layers, and one output layer.

operate as fully connected layers, which means that each node in one layer is connected to each node in the next layer. All the nodes have a variable weight assigned as an input which are linearly combined (or summed together) and passed through an activation function to obtain the output of that particular node.

5.2.2 Algorithm of ANN

The training procedure is illustrated in Fig. 5.2. Firstly, a sufficient number of randomly generated data samples are collected from the simulations using COMSOL Multiphysics for slot, strip and directional coupler structures. Each case has an array of inputs, called features, and an array of numerically solved outputs, called labels. Waveguide width, height, material, gap between the waveguides, and operating wavelength values can be taken as the input variables which are assigned to the nodes of the input layer. The effective index (n_{eff}), power confinement (P_{conf}), or coupling length (L_c) are taken as the output variables, which are assigned to nodes of the output layer depending on the specific design requirement. Next, preprocessing of the collected data is carried out by normalizing the input variables values between the range 0–1 to use a common scale. This is followed by shuffling of the normalized input data, otherwise the model can be biased towards particular input data values.

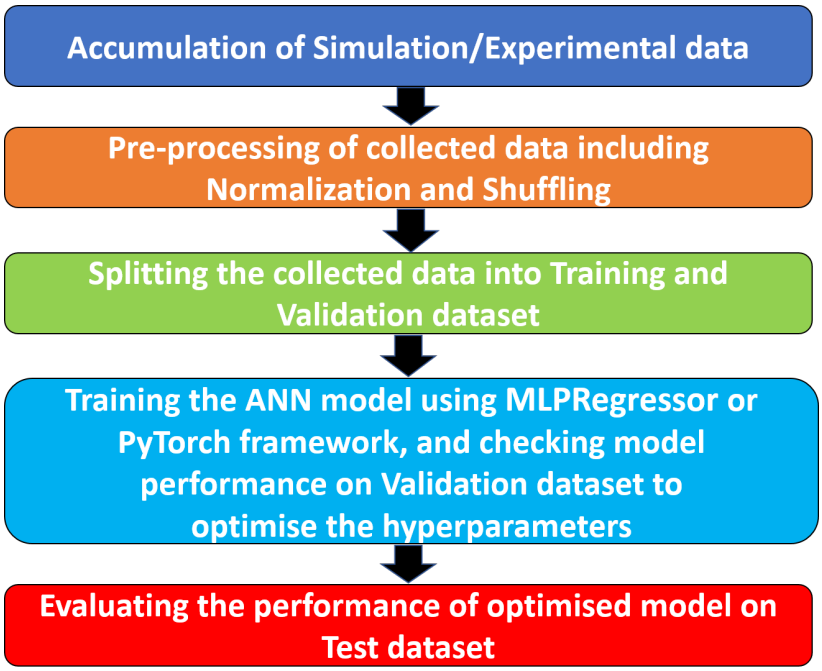


Fig. 5.2 The flow chart of ANN implementation.

The next step is to split the normalised input dataset into training and validation dataset. Validation dataset is used to provide an unbiased evaluation of a model fit on the training dataset while tuning various model parameters, also called hyperparameters. 5–25% of data has been allocated for the validation dataset in this work, while the rest was used for training the ANN model.

Neural networks have a tendency to closely or exactly fit a particular set of data during training, but may fail to predict future observations reliably, which is known as overfitting. During overfitting, the model learns both the real and noisy data, which negatively impacts on new data. Overfitting can be avoided through regularization such as dropout [71], while regularly monitoring the performance of the model during training on the generated validation dataset. Underfitting can be another cause of poor performance of an ANN in which the trained model neither closely fits the initial data nor generalizes to the new data. Hyperparameters need to be tuned to reduce the mean squared error (MSE) between the actual and predicted output values of the ANN model for a regression problem. During this

optimisation process, weights and biases of the model are repeatedly updated with each iteration or epoch using the backpropagation algorithm [70]. Various hyperparameters of choice includes activation functions, type of optimiser, number of hidden layers, number of nodes in each hidden layer, learning rate, number of epochs, and others.

5.2.2.1 Activation Functions

An ANN connects inputs and outputs through a set of non-linear functions, which is approximated using a non-linear activation function. Sigmoid, Tanh (hyperbolic tangent), and ReLU (rectified linear unit) are some of the commonly used activation functions [54].

$$\text{Sigmoid} : \sigma(z) = \frac{1}{1+e^{-z}} \tag{5.1}$$

$$\text{Hyperbolic Tangent (Tanh)} : \sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{5.2}$$

$$\text{Rectified Linear Unit (ReLU)} : \sigma(z) = max(0,z) \tag{5.3}$$

Among these, ReLU is used mostly as it trains the model several times faster in comparison to when using a Tanh function, as discussed in [72].

5.2.2.2 Optimisation Solvers

LBFGS, stochastic gradient descent (SGD), and Adam [73] solvers can be used to optimise the weights values during ML training process. Adam optimiser is a preferable choice as it works well on relatively large datasets.

5.2.2.3 Hidden Layers and Nodes

The number of layers or number of nodes in each layer of an ANN are decided by experimentation and from the prior experience of similar problems. There is no fixed rule to pre-decide their optimal values.

5.2.2.4 Learning Rate

Learning rate decides how much the weights of the network are adjusted with each epoch or iteration. Choosing the lower value of learning rate means the model needs more epochs and a longer time to converge. If the input dataset is big, it may take a very long time to optimise the ANN model. On the other hand, if the learning rate has a large value, then the model might fail to converge at all with gradient descent [74, 75] overshooting the global minima. Learning rate can be chosen to have constant or adaptive value when using Scikit-learn MLPRegressor.

5.2.2.5 Epochs

The number of epochs to train a model should be decided by the user when MSE value converges to an acceptable lower limit. Depending on the dataset size, model training can carried out using batches of inputs. In the case when using MLPRegressor, an automatic batch size was used, while all the inputs were trained in one batch with PyTorch.

Once the optimal hyperparameters are obtained, the final step is to evaluate the performance of the optimised trained model on the previously unseen test dataset (generated separately from the initially generated dataset) to observe the accuracy of the ANN model.

5.3 Numerical Results and Discussion

5.3.1 Slot Waveguide

Slot waveguide design structures are extensively used for optical sensing applications [76, 77], as the light is confined in low refractive index region, which allows strong interaction with the analyte leading to a large waveguide sensitivity.

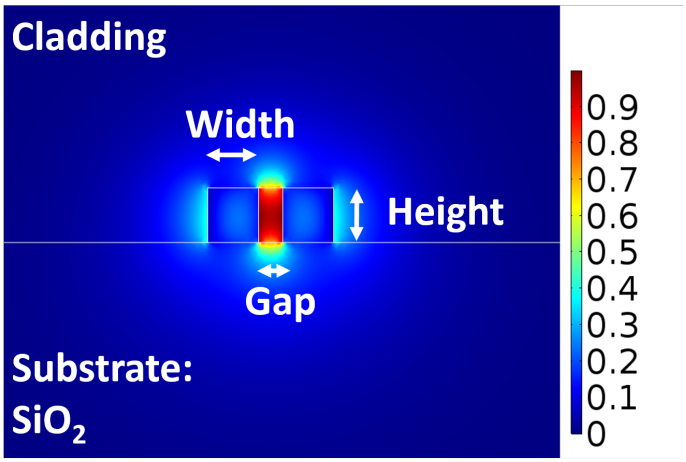


Fig. 5.3 An example of a slot waveguide showing E_x field profile.

A cross-sectional view of a slot waveguide with E_x field profile is shown in Fig. 5.3. A range of slot waveguides was simulated by changing the width, height, and gap between the silicon waveguides as the input parameters. In this case, use of ML algorithms is demonstrated to predict the effective index (n_{eff}) and power confinement (P_{conf}) in a slot waveguide design, but first various hyperparameters of the ANN model are optimised.

5.3.1.1 Histogram of Datasets

The training process requires a dataset of examples, which plays a crucial role in any ML algorithm. The accuracy of the trained model depends on the quality of the input data. A

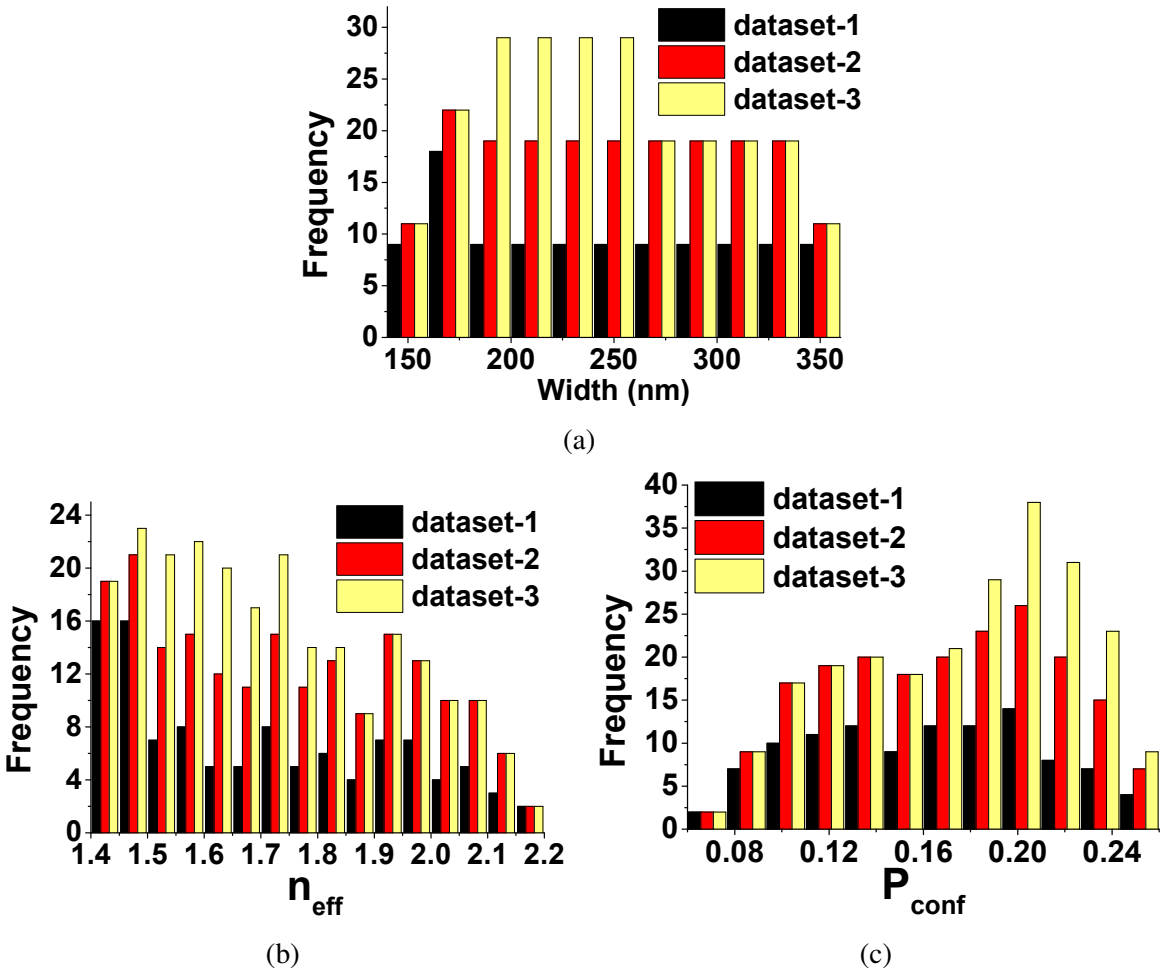


Fig. 5.4 Histogram of different datasets for slot waveguide with varying (a) width of waveguides, (b) n_{eff}, and (c) P_{conf}.

good training dataset which is well aligned with the problem to be solved is needed for the ML code to work properly.

Three different datasets to predict effective index (n_{eff}) and power confinement (P_{conf}) for a slot waveguide structure, shown in Fig. 5.4 were collected. Width, height, and gap between the waveguides in a slot waveguide design have been varied initially to record the n_{eff} and P_{conf} values for each dataset. n_{eff} and P_{conf} values recorded for a particular combination of width, height, and gap between the slot waveguides become one datapoint. Dataset-1 has 108 datapoints with nearly equal intervals between width, height, and gap. Dataset-2 also has points with nearly equal intervals but more values, 196 points. However,

dataset-3 has higher frequency of data points for waveguide width in the range 200–250 nm, with 236 values in total. Figure 5.4a shows the histogram of different datasets plotted with respect to different widths of the waveguides. It should be noted here that these datasets are not explicit, which implies that the dataset-1 is a subset of dataset-2 and similarly, dataset-2 is a subset of dataset-3. Figures 5.4b and 5.4c show the frequency of n_{eff} and P_{conf} for each of these 3 datasets. For the simulation setup, it took 2-3 minutes to record one datapoint, which means it took approximately 200, 400, 500 minutes to obtain dataset-1, dataset-2, and dataset-3, respectively. The time needed to collect one datapoint value may vary depending on the simulation/experimental setup.

5.3.1.2 Mean Squared Error

Mean squared error (MSE) is considered as the loss function in a regression problem, which is defined as the average squared difference between the estimated and true values, given as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\widehat{y_i} - y_i)^2 \tag{5.4}$$

where $\widehat{y_i}$ and y_i are the estimated and true data point values, respectively.

A smaller value of MSE means the predicted regression values are closer to the original values and hence the model is well trained. Next, the MSE values are compared to predict the n_{eff} for a slot waveguide design with different numbers of nodes or layers in an ANN model for dataset-3 using MLPRegressor from Scikit-learn. Dataset-3 is chosen as it has the maximum number of data points among the 3 datasets generated. Figure 5.5a shows that MSE decreases faster to a stable value when number of nodes is larger. MSE for nodes = 50 quickly reaches a stable low value of 0.0025 at epochs = 1500, shown by the orange line in comparison to 0.0192, 0.0820, and 0.3954 when nodes are taken as 25, 10 and 5, respectively. Random weights were assigned at the start of the algorithm. Hence, MSE for more number of nodes at first epoch can be larger than that for less number of nodes, as can be seen from blue

and red lines having values of 0.2112 and 0.1423 at first epoch, respectively. It can also be observed from the red and blue lines that the model with the greater number of nodes attains optimal updated weights quicker than those with the lower number of nodes, as the MSE for nodes = 25 (blue line) decreases quickly. Simulations were run upto 4000 epochs so as to be sure that MSE decreases to a lower value. At epochs = 4000, MSE values are 0.21279, 0.04685, 0.00109, and 0.00018 when number of nodes are 5, 10, 25, and 50, respectively. This shows that more neurons/nodes helps in achieving better accuracy for the ANN model by quickly decreasing the MSE value to the minimum, but the computational loading also increases.

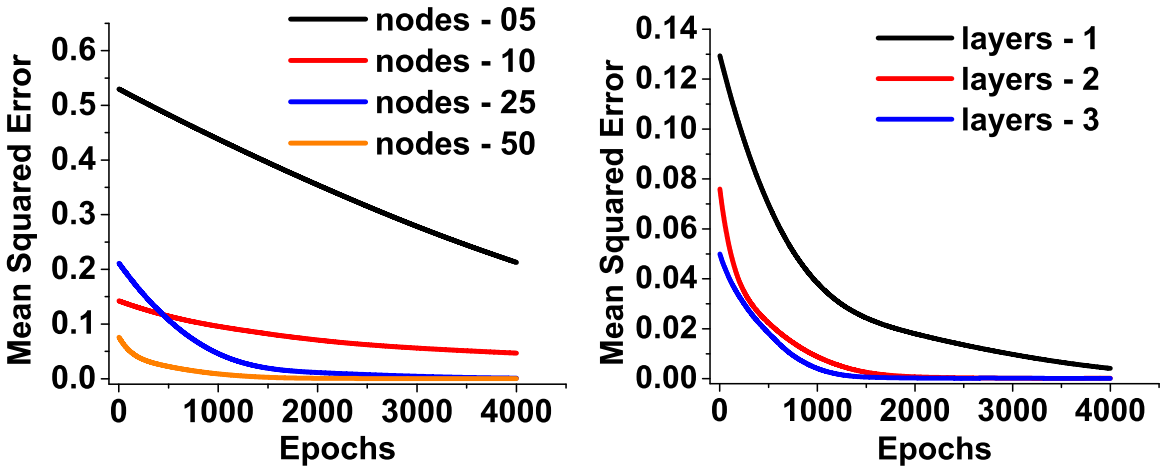
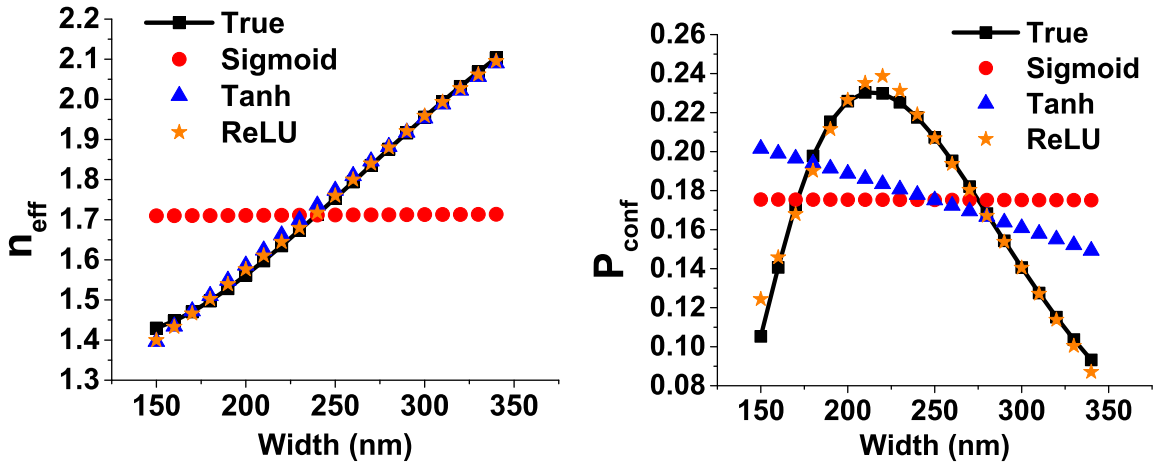


Fig. 5.5 Mean squared error (MSE) using training dataset-3 for (a) different number of nodes with 2 hidden layers, (b) different number of hidden layers with 50 nodes in each hidden layer.

The MSE variations when number of layers are varied in an ANN model having 50 nodes in each layer, as shown in Fig. 5.5b. The MSE values of 0.0025 and 0.0006 are obtained for models with 2 and 3 hidden layers, respectively at epochs = 1500 in comparison to 0.0243 when number of layers is equal to 1. Lower stable MSE values at epochs = 4000 are 0.00412, 0.00018, and 0.00009 when number of layers are 1, 2 and 3, respectively, with each layer having 50 nodes. Following this study, the number of layers as 2 with 50 nodes in each layer

are considered for future optimisation, to avoid more computational loading compared to when the number of layers was chosen as 3.

5.3.1.3 Activation Functions



(a) (b)

Fig. 5.6 Variation of (a) n_{eff} and (b) P_{conf} with waveguide width for different activation functions at waveguide height = 225 nm using training dataset-3.

Sigmoid, Tanh and ReLU activation functions were tested to predict the n_{eff} and P_{conf} using MLPRegressor trained model with 2 layers having 50 nodes in each layer. Dataset-3 is used during the training process. It can be seen from Fig. 5.6a that Tanh and ReLU closely predict the n_{eff} values compared to the true values at a waveguide height = 225 nm of the slot design. The data corresponding to a waveguide height = 225 nm has not been recorded or provided during the training of the model. However, data for other waveguide heights was used for the training. On the other hand, the Sigmoid function predicts an almost horizontal line as shown by the red circle symbols. Sigmoid function fails to predict the n_{eff} values accurately, as it might converges well for a classification problem. When tested to predict the P_{conf} of slot design, only ReLU activation function is able to predict the pattern much better, shown by orange star symbols and black rectangle symbols solid line in Fig. 5.6b, hence seems to be a better choice.

5.3.1.4 Comparing PyTorch Framework and MLPRegressor models

In-house codes using PyTorch framework and MLPRegressor from Scikit-learn were developed. 2 fully connected hidden layers each with 50 nodes were used. The ReLU activation function and Adam optimiser were employed for both of the generated codes. The learning rate was chosen to be 0.0001 or less. The dropout fraction value of 0.5 was used for regularization to prevent over-fitting when using the PyTorch framework, while dropout regularization was not available in the MLPRegressor. The number of epochs was decided based on when the MSE was reduced to a stable value for the considered photonics design structure.

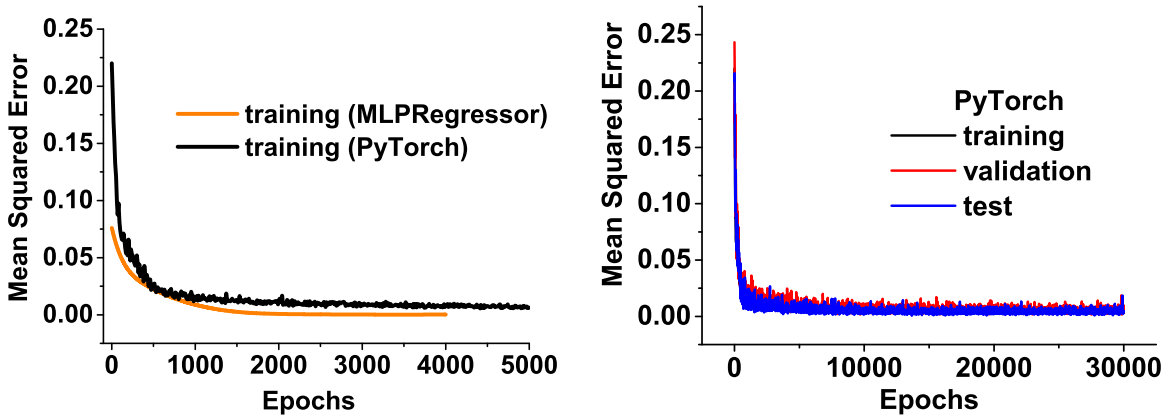


Fig. 5.7 Mean squared error (MSE) using (a) training dataset-3 for MLPRegressor and PyTorch (b) training, validation, and test dataset-3 for PyTorch, having 2 hidden layers with 50 nodes in each layer.

In Fig. 5.7, MSE values are compared using MLPRegressor and PyTorch for training, validation and test datasets. It should be noted here that the MSE or loss function for validation and test datasets are not readily available in MLPRegressor. The loss curve in MLPRegressor also depends on the initially defined tolerance for the optimisation ($1e^{-8}$, used in this case). MLPRegressor training automatically stops when the loss or score is not improving by at least the tolerance level in the consecutive iterations, which is shown by the orange solid line in Fig. 5.7a, where the MSE curve stops by itself at around 4000 epochs.

PyTorch can be used to visualise the MSE at very large epochs (>>4000) for training, validation, and test datasets, as shown in Fig. 5.7b. The fluctuations in the MSE curves with PyTorch are due to the use of the dropout fraction. The MSE for training, validation, and test datasets with PyTorch follow similar trend, and achieves a stable value at around 4000 epochs. The MSE for training dataset at epochs = 4000, 10000, and 30000 are 0.00808, 0.00451, and 0.00300, respectively. It can be observed here, that the MSE is almost similar and decreases slowly when the epochs is greater than 4000. Hence, it is good to fix the epochs when the MSE achieves a stable low value, rather than allowing the algorithm to run for very large epochs. MLPRegressor shows a lower MSE for the training dataset in comparison to PyTorch, as shown in Fig. 5.7a. Using the above mentioned parameters, the n_{eff} and P_{conf} are now predicted for a slot waveguide structure using PyTorch framework and MLPRegressor library for different datasets.

5.3.1.4.1 Effective Index (n_{eff}) Figure 5.8 shows the prediction of n_{eff} at waveguide height = 225 nm for a slot waveguide design. It should be noted that the datasets did not have any value corresponding to this particular waveguide height of 225 nm. True n_{eff} values of test dataset are compared with predicted values for PyTorch and MLPRegressor models at 500, 2000 and 5000 epochs using dataset-1 and dataset-2 during training. Figures 5.8a and 5.8b show the training prediction of n_{eff} at a waveguide height = 225 nm using PyTorch and MLPRegressor, respectively and using dataset-1. It can be observed when epochs = 500 (shown by red circle symbols), the predicted values are relatively far apart from the actual or true values obtained from the numerical simulations. When epochs were increased to 2000 (shown by blue triangle symbols) for both PyTorch and MLPRegressor, predicted values starts to follow the pattern of n_{eff} but the absolute values are still not sufficiently close. The epochs were further increased to 5000 (shown by orange star symbols) for both PyTorch and MLPRegressor. It can be seen that the predicted values are almost similar to the true values when dataset-1 was considered. Dataset-2 was also considered which has

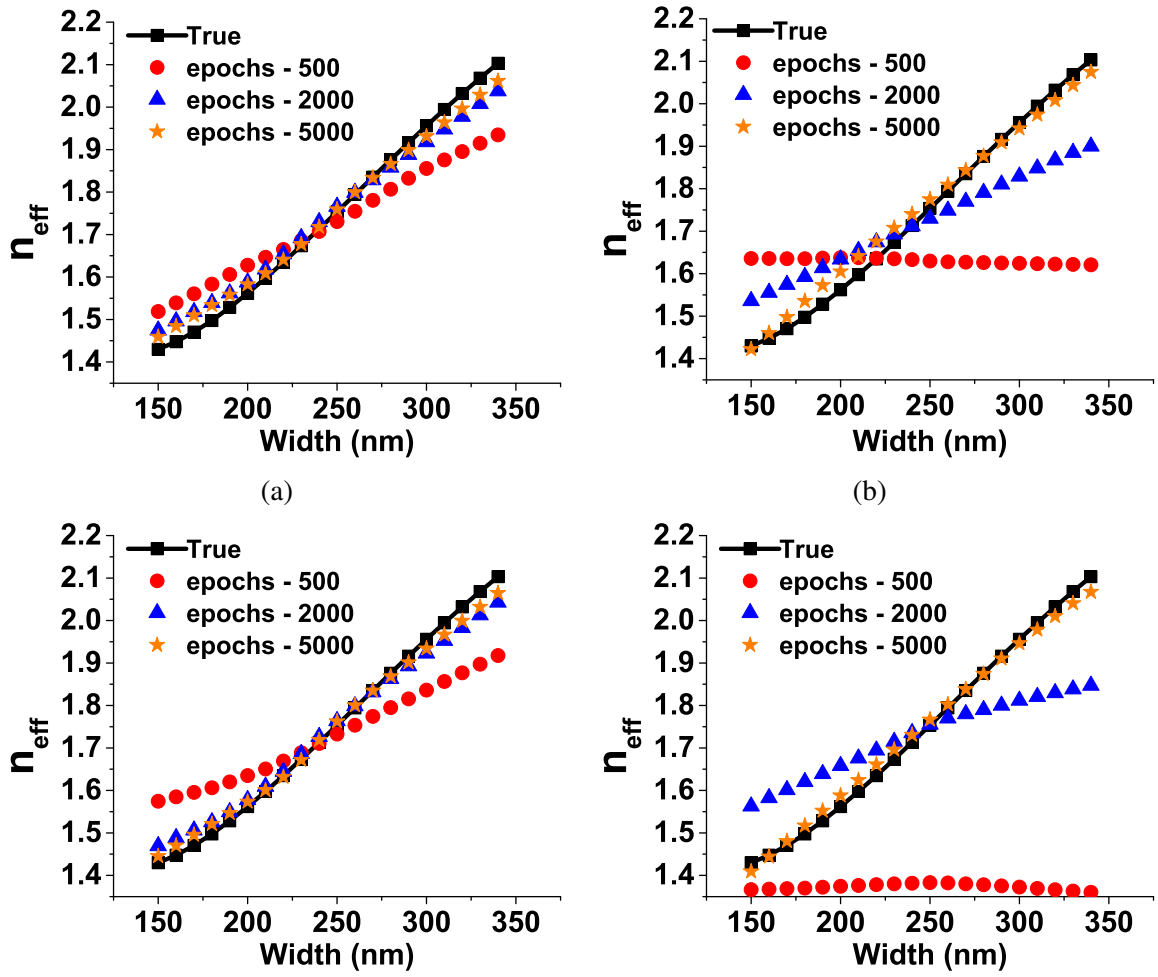


Fig. 5.8 Slot waveguide design predicting n_{eff} at waveguide height = 225 nm with (a) PyTorch using dataset-1, (b) MLPRegressor using dataset-1, (c) PyTorch using dataset-2, and (d) MLPRegressor using dataset-2.

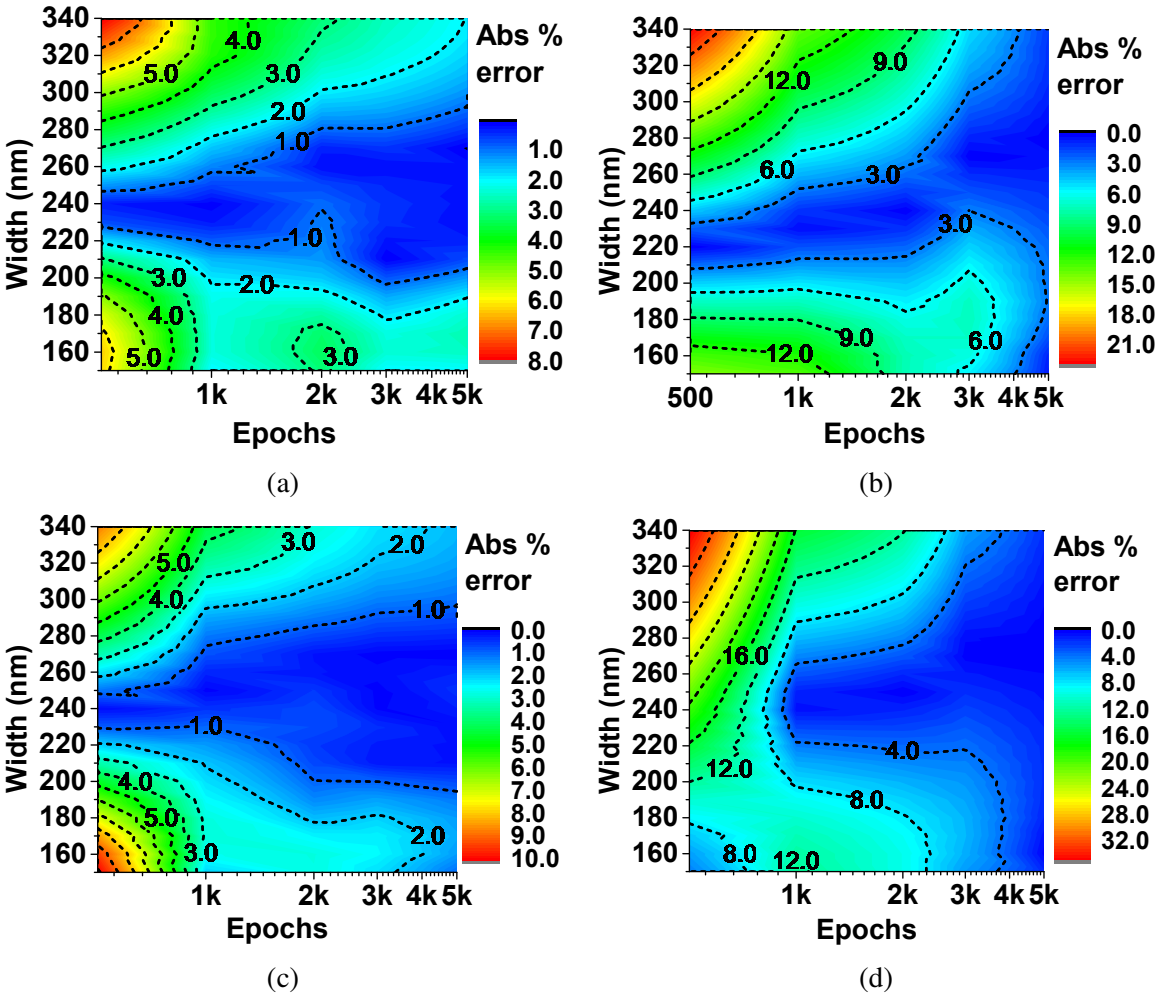


Fig. 5.9 Slot waveguide design showing contour of absolute percentage error for predicting n_{eff} at waveguide height = 225 nm with (a) PyTorch using dataset-1, (b) MLPRegressor using dataset-1, (c) PyTorch using dataset-2, and (d) MLPRegressor using dataset-2.

more datapoints that dataset-1. Again almost 5000 epochs are required using dataset-2 to efficiently predict the n_{eff}, as can be seen in Figs. 5.8c and 5.8d, respectively. This shows that epochs = 5000 are required to efficiently predict the n_{eff} values of test dataset for both PyTorch and MLPRegressor models using either dataset-1 or dataset-2. Predicted and true values are almost identical in all four cases, shown by the orange star symbols and black rectangle symbols with solid line, respectively.

Fig. 5.9 shows the percentage error in predicting the n_{eff} values using both PyTorch and MLPRegressor using dataset-1 and dataset-2. Here, percentage error has been calculated by comparing the predicted solutions with the numerically simulated results using COMSOL Multiphysics. The absolute percentage error at epochs = 500 for both datasets varies between 5–10% (Figs. 5.9a and 5.9c) using the PyTorch model. However, as the epochs are increased to 2000, this absolute error falls to 1–4% for both the datasets. Further increasing the epochs to 5000 decreases the absolute percentage error below 2% when PyTorch model was used. Similarly, when the epochs = 500 for both datasets with the MLPRegressor model, the absolute percentage error was between 6–20% (Figs. 5.9b and 5.9d). Increasing the epochs to 2000 decreases this percentage error to approximately 4–8%. This error further decreases to 1–3% when epochs were increased to 5000. This shows that the absolute percentage error at epochs = 500 for both datasets varies between 5–10% and 6-20% for PyTorch and MLPregressor models, respectively. As the epochs are increased to 5000, absolute percentage error reduces to only 1-3% for all the cases which may be acceptable in predicting n_{eff} values for a slot design.

5.3.1.4.2 Power Confinement (P_{conf}) Here, the model is trained using PyTorch and MLPRegressor for different datasets to predict the P_{conf}. Figures 5.10a and 5.10b show the training prediction for a waveguide height = 225 nm using PyTorch and MLPRegressor, respectively using dataset-1. It can be seen that even at epochs = 10000 (shown by the orange star symbols), the predicted value of P_{conf} is much different from the true values especially

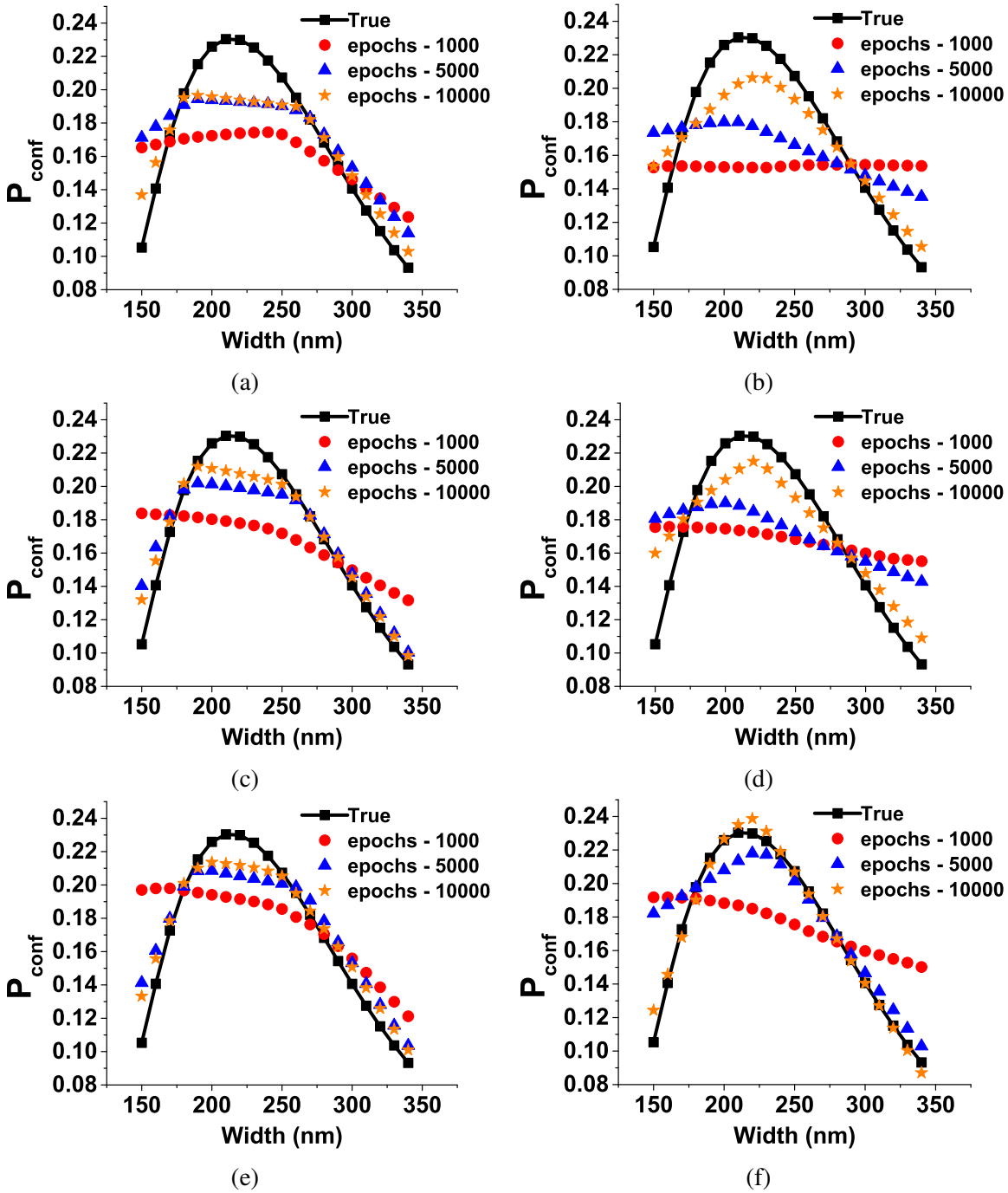


Fig. 5.10 Slot waveguide design predicting P_{conf} at waveguide height = 225 nm with (a) PyTorch using dataset-1, (b) MLPRegressor using dataset-1, (c) PyTorch using dataset-2, (d) MLPRegressor using dataset-2, (e) PyTorch using dataset-3, and (f) MLPRegressor using dataset-3.

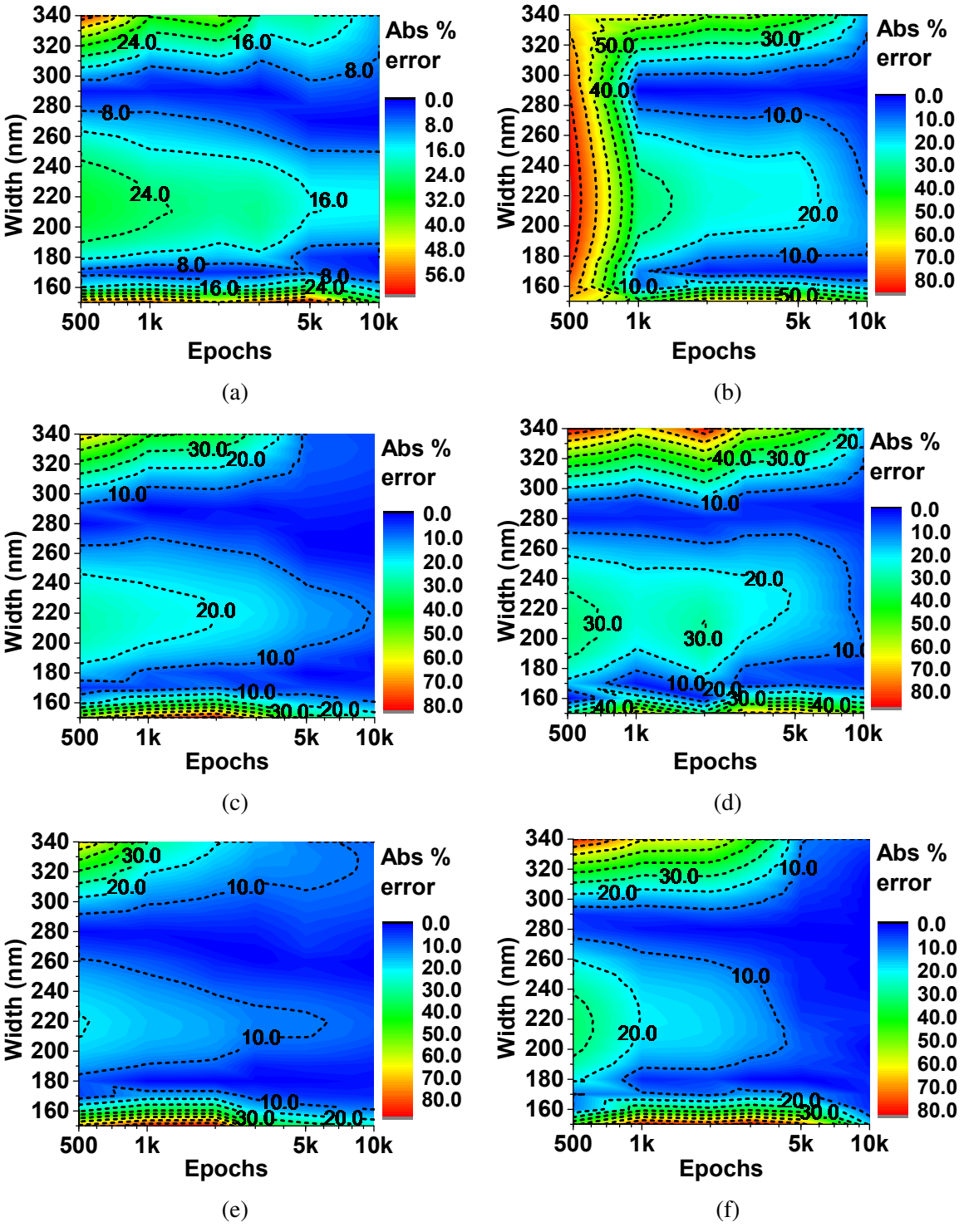


Fig. 5.11 Slot waveguide design showing contour of absolute percentage error for predicting P_{conf} at waveguide height = 225 nm with (a) PyTorch using dataset-1, (b) MLPRegressor using dataset-1, (c) PyTorch using dataset-2, (d) MLPRegressor using dataset-2, (e) PyTorch using dataset-3, and (f) MLPRegressor using dataset-3.

when the width is between 200–250 nm for both the algorithms. This error originated from the neural network modelling probably due to the insufficient data points in the parameter space of width between 200–250 nm in dataset-1, which may lead to underfitting of the trained model. Next, the neural network using dataset-2 was trained, which contains more data points compared to dataset-1, which is shown in Fig. 5.4a. Figures 5.10c and 5.10d show the predictions of the trained model using dataset-2 for PyTorch and MLPRegressor, respectively. It was observed that the trained model performed better than the dataset-1 case, but still not good enough for widths in the range 200–250 nm. This error can be further reduced by collecting more data points in the widths ranging from 200-250 nm during training. Figures 5.10e and 5.10f show the trained model performance using dataset-3, which has more data values in the range 200–250 nm, as shown in Fig. 5.4a. It can be observed that the PyTorch trained model (Fig. 5.10e) did not perform efficiently at epochs = 10000, predicting almost constant values in the waveguides width ranging between 200–250 nm. This error can be minimised if the dropout factor is reduced or taken as zero, but this may lead to over-fitting of the model. On the other hand, MLPRegressor (Fig. 5.10f) predicts the P_{conf} curve better at epochs = 10000, shown by the orange star symbols, which is similar to the true shape (black square symbols solid line). However, the predicted P_{conf} values at waveguide width = 150 nm deviate significantly, but this can be further improved by collecting more data points in this range of waveguides width. For respective PyTorch and MLPRegressor models, it can be observed that prediction of P_{conf} using dataset-3 is better than dataset-2, which in turn is better than dataset-1 at epochs = 10000, shown by the orange star symbols in Figs. 5.10a–5.10f. This shows that the quality of dataset plays an important role along with the choice of the algorithm and optimised values of hyperparameters.

The absolute percentage error was compared for predicting P_{conf} using dataset-1, dataset-2, and dataset-3. When using dataset-1, both PyTorch and MLPRegressor have error ranging between 10–40% when epochs vary from 1000 to 10000, as shown in Figs. 5.11a and 5.11b,

respectively. Figures 5.11c and 5.11d show that by taking more data values as in dataset-2, this error was reduced to between 7–30% with epochs. Dataset-3 shows a better performance with trained model using both PyTorch and MLPRegressor, as shown in Figs. 5.11e and 5.11f, respectively. The absolute percentage error ranges between approximately 7–10% and 1–4% for PyTorch and MLPRegressor at epochs = 10000, respectively. This shows that the MLPRegressor performed better than PyTorch for this particular set of design specifications of the slot waveguide with dataset-3. This performance difference between MLPRegressor and PyTorch is due to the different functionalities available in the algorithms. For example, dropout can only be implemented with PyTorch, while MLPRegressor uses exponential decay rates and numerical stability functions with Adam optimiser. Furthermore, the advantage of ANNs can be more pronounced if the sample space is also large.

5.3.1.5 Training Dataset Sizes

In the proposed model algorithms, the initially collected data was split into training and validation datasets depending on the percentage parameter. Figure 5.12 compares the true P_{conf} values with predicted values at a waveguide height = 225 nm using the MLPRegressor trained model with 25%, 50%, 75%, or 95% of initial collected data as training dataset. The data points were randomly selected for each case so that the final trained model was not biased towards any particular data points.

When only 25% or 50% of data was considered, the algorithm showed higher errors in predicting the values, especially in the waveguide width range of 150–225 nm, shown by red circles and blue triangles symbols, respectively. For epochs = 10000, true solutions are shown by the black line. This is understandable as ANN prediction error after training can be large if the sample space has limited dataset points. When 75% of the data was taken as input (shown by the green diamond symbols), it can observed that most of the data points were satisfactory, except in the width ranging from 150–175 nm that are not well predicted.

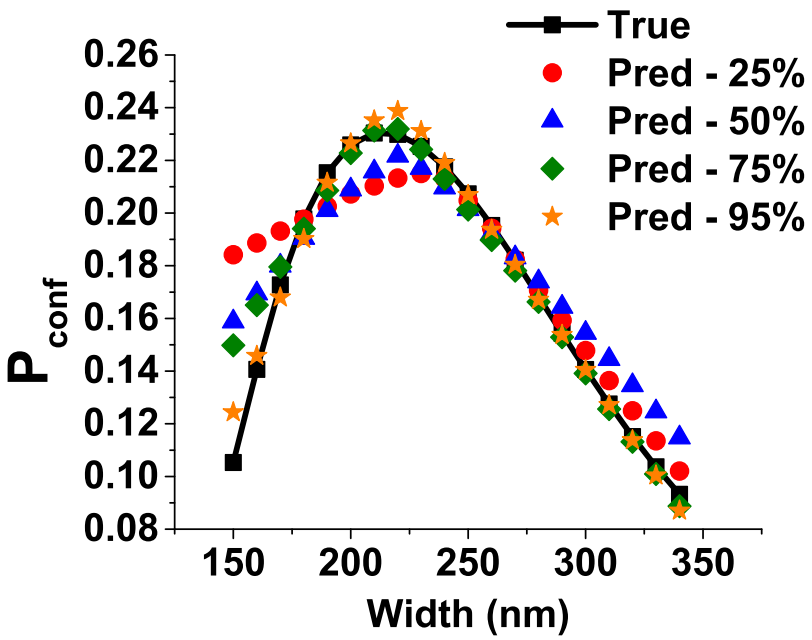


Fig. 5.12 Variation of P_{conf} with width at waveguide height = 225 nm for different data sizes of training dataset-3 for epochs = 10000 using MLPRegressor.

It can be seen that when 95% of data was used for training, the trained model predicted the true values more accurately, as shown by the orange star symbols. Therefore, if the dataset is overly reduced, error values of prediction increases significantly, as in the case of 25% in Fig. 5.12. Predicting an output only takes few milliseconds, once the model is trained by either MLPRegressor or PyTorch. On the other hand, it takes few minutes to get an output for particular waveguide dimensions with direct numerical simulation, which also depends on the density of the considered mesh.

5.3.2 Strip Waveguide

An ANN model was trained to obtain optimised hyperparameters for predicting the n_{eff} values of a strip waveguide. For a strip waveguide (Fig. 5.13), the width, height of the waveguides and wavelength were taken as input variables, while effective index was considered as the output variable. An MLPRegressor model was considered to train the initially recorded 225 data points with varying operating wavelength, height and width of the strip waveguide.

Hyperparameters were optimised while training the model to obtain low and stable MSE. The optimal weights of the model were then used or saved (to be used later) to predict the n_{eff} values on unseen test datasets.

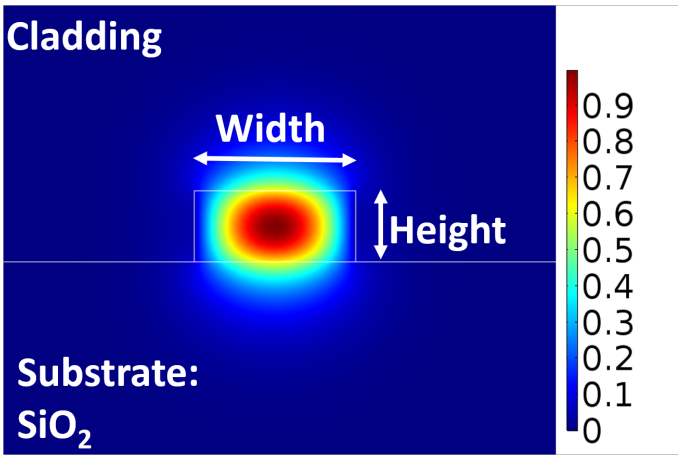


Fig. 5.13 An example of a strip waveguide showing H_y field profile.

Figures 5.14a, 5.14c, and 5.14e show the true and predicted n_{eff} values for different epochs with waveguide width, height and wavelength, respectively. Figures 5.14b, 5.14d, and 5.14f show the contour of absolute percentage error between true and predicted n_{eff} values with epochs (on logarithmic scale). It can be seen that the percentage error is approximately 16–20% when epochs = 250 and it decreases to approximately 2% when epochs = 2000 for all the above mentioned cases. When epochs = 4000, the percentage error is further reduced to less than 1%. Hence, the trained model is performing well at epochs = 4000. The trained model weights at epochs = 4000 can be saved for future testing.

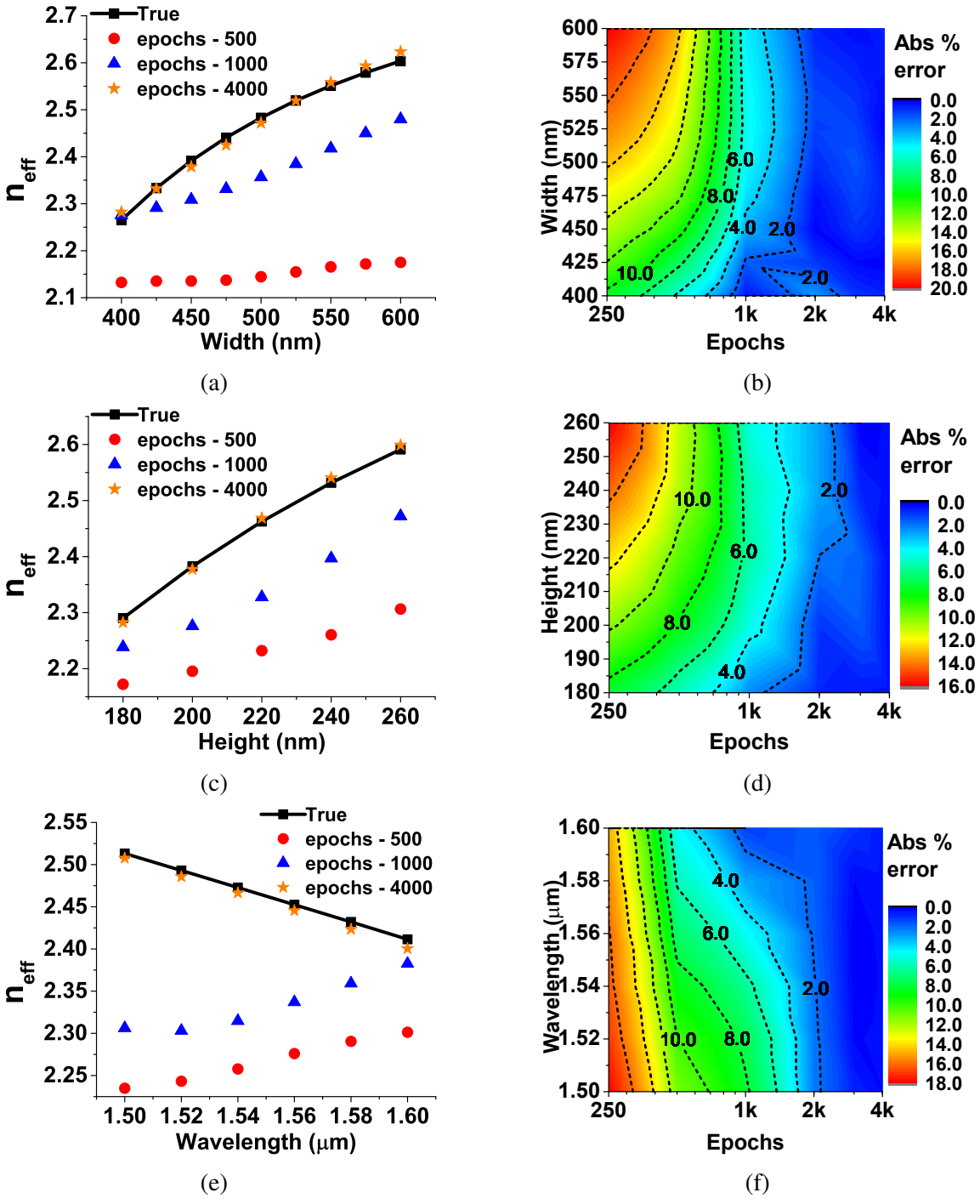


Fig. 5.14 Strip waveguide design (a) predicting n_{eff} at waveguide height = 230 nm and wavelength = 1.55 μm, (b) showing contour of absolute percentage error for predicting n_{eff} at waveguide height = 230 nm and wavelength = 1.55 μm, (c) predicting n_{eff} at waveguide width = 510 nm and wavelength = 1.55 μm, (d) showing contour of absolute percentage error for predicting n_{eff} at waveguide width = 510 nm and wavelength = 1.55 μm, (e) predicting n_{eff} with change in wavelength at waveguide width = 510 nm and height = 230 nm, and (f) showing contour of absolute percentage error for predicting n_{eff} at waveguide width = 510 nm and height = 230 nm.

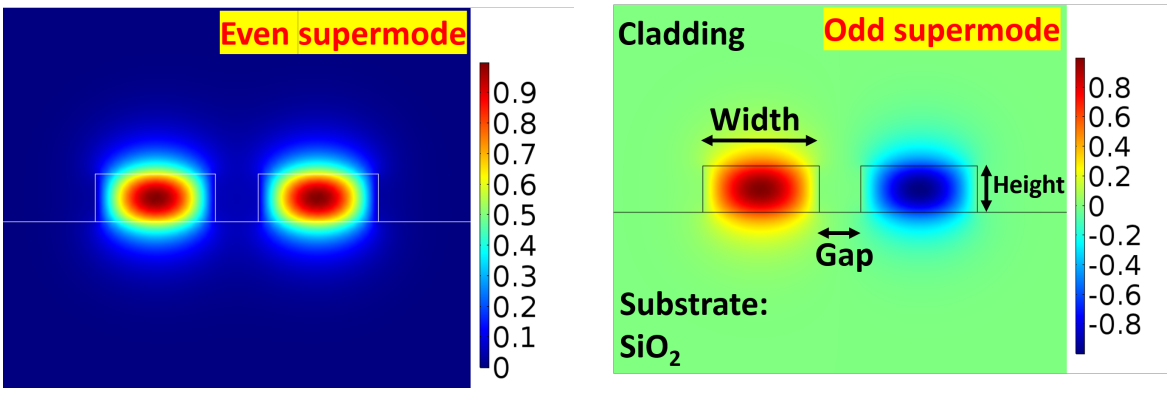
Table 5.1 Comparing predicted with true n_{eff} values and corresponding absolute percentage error for random wavelength, height, and width of strip waveguide design.

Wavelength (μm)	**Height (nm)**	**Width (nm)**	**Predicted - n_{eff}**		**True - n_{eff}**	**Absolute % error - n_{eff}**	
			epochs - 1000	**epochs - 4000**		**epochs - 1000**	**epochs - 4000**
1.52	210	490	2.323764	2.418138	2.424294	4.14	0.25
1.52	230	510	2.333022	2.541238	2.528034	7.71	0.52
1.54	210	490	2.32193	2.396881	2.402842	3.36	0.24
1.54	230	510	2.342372	2.520985	2.50819	6.61	0.51
1.56	210	490	2.328336	2.370507	2.38129	2.22	0.45
1.56	230	510	2.354483	2.489118	2.488223	5.37	0.03
1.58	210	490	2.330601	2.34713	2.359644	1.23	0.53
1.58	230	510	2.35926	2.45187	2.468138	4.41	0.65

The performance of the saved model was checked for some random strip waveguide design parameters which are not available in the training dataset. The resulting true and predicted n_{eff} values are compared in Table 5.1 when all the input parameters (operating wavelength, waveguides height and width) were unknown to the trained model, which implies that the outputs corresponding to these parameters have never been recorded during the initial data collection. It can be observed that when epochs = 4000, the absolute value of percentage error to predict n_{eff} is less than 1% for randomly chosen different input design parameters. This demonstrates that the model is performing well to predict n_{eff} for a strip waveguide design.

5.3.3 Directional Coupler

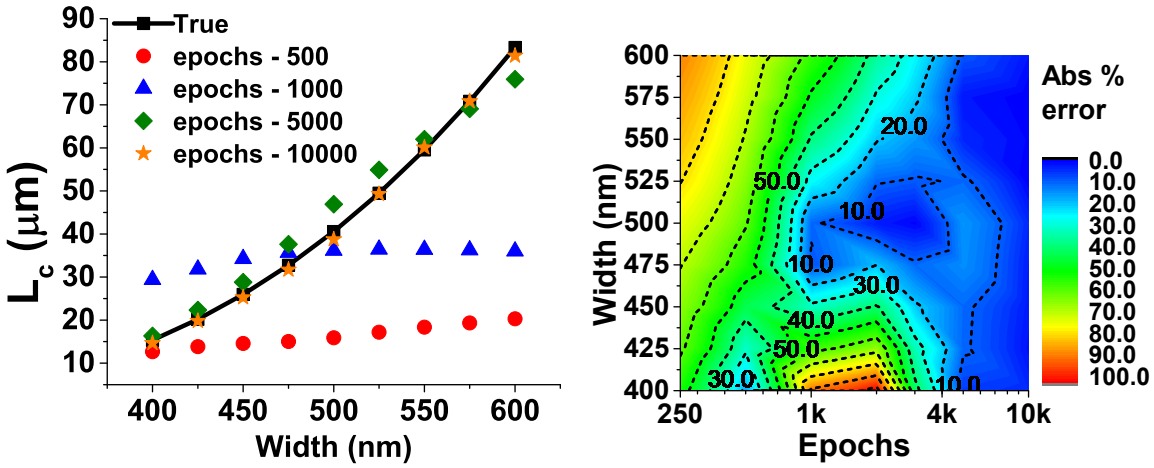
The directional coupler has been a major component in many photonic devices including spot-size converter [78], mode demultiplexer [79], polarization rotator [80], polarization splitter [81], etc. A directional coupler can couple light from one waveguide to another, and then couples back to the original/first waveguide, and so forth. A cross-sectional view of a directional coupler showing H_y field profile for even and odd supermodes is shown in Figs. 5.15a and 5.15b, respectively. Width, height, and gap between the silicon waveguides are changed to record the coupling lengths for a directional coupler design.



(a) (b)

Fig. 5.15 An example of a directional coupler showing H_y field profile for (a) even supermode, and (b) odd supermode.

The application of ML algorithm using MLPRegressor to predict the coupling length (L_c) of a directional coupler is presented. Again, The model parameters are optimised to obtain minimum stable MSE value using training dataset for a directional coupler design. Different height, width and gap between the waveguides are considered as the input parameters, and L_c is taken as the output parameter during the ANN training. Figure 5.16a shows the predicted L_c when epochs were taken as 500, 1000, 5000, and 10000. It can be observed that predicted L_c values are closer to the true values when epochs were 5000 or 10000 at a waveguide height = 230 nm. It should be noted that the training dataset did not have L_c values when the



(a) (b)

Fig. 5.16 Directional coupler design (a) predicting L_c at waveguide height = 230 nm and (b) showing contour of absolute percentage error for predicting L_c at waveguide height = 230 nm.

waveguide height = 230 nm. Figure 5.16b shows that there is approximately 6–10% absolute percentage error in L_c value at different widths when epochs = 5000. This error is reduced to 1–4% for different widths when the model was trained to 10000 epochs. Increasing the initially recorded data points to train the model can help in further reducing this absolute percentage error between predicted and true L_c values of a directional coupler. This model can be trained to calculate L_c for any given height, width, separation and operating wavelength of a directional coupler.

5.4 Summary

In summary, a machine learning model for predicting the effective index, power confinement and coupling length in a slot waveguide, strip waveguide and directional coupler design structure has been developed. Dataset-3 was the better choice in comparison to other datasets for the considered slot waveguide design, as it contained a greater number of input data points, which helps the machine learning model to be trained better. The absolute percentage error

in predicting the effective index for a slot waveguide design was lower than 2% using both PyTorch or MLPRegressor. PyTorch and MLPRegressor models give an absolute percentage error of approximately 7–10% and 1–4%, respectively for predicting the power confinement, which shows that MLPRegressor model performs better. A ReLU activation function was preferred as it better predicts the n_{eff} and P_{conf} better in comparison to when using Tanh or Sigmoid functions. MLPRegressor model has also been used to predict the effective index in a strip waveguide design with 99% accuracy. Similarly, hyperparameters have been optimised for a separate MLPRegressor model to predict the coupling length for a directional coupler design, giving only 1–4% of absolute percentage error. To the best of our knowledge, this is the first time machine learning is used in conjugation with rigorous finite element method for various nanophotonic waveguide analyses. The approach can accurately predict the waveguide parameters without extensive use of the computationally expensive time- and frequency-domain numerical methods.

Chapter 6

Machine learning approach for computing optical properties of a photonic crystal fibre

6.1 Introduction

Photonic crystal fibre (PCF) was first proposed by Knight *et al.* [82] in 1996, which consisted of a core with the periodic arrangement of air holes running along the length of the fibre. The core of the PCF can be solid or hollow. For a solid core PCF, there is a positive refractive index difference between the core and equivalent index of the cladding, and light is guided using the modified total internal reflection (TIR) phenomenon. On the other hand, hollow core PCF has a negative refractive index difference between the core and cladding, and light guidance is based on photonic band gap (PBG) mechanism [83]. Such structures exhibit the novel properties of being low loss and endlessly single mode propagation. Other specific fibre properties, including effective index (n_{eff}), propagation constant, effective mode area (A_{eff}), dispersion (D), non-linearity, birefringence and confinement loss (α_c) can be easily controlled by changing the holes size, the spacing between them and the number of air-hole

rings. The unique properties of PCF over standard optical fibres has motivated researchers to use PCF for supercontinuum generation [84, 85], Raman scattering [86, 87], fibre lasers [88], optical sensors [89, 90], spectroscopy [91], among other applications.

Accurate modelling and optimisation of photonic crystal fibres generally rely upon numerical methods such as finite difference method [92], finite element method (FEM) [93], block-iterative frequency-domain method [94], and the plane wave expansion method [95, 96]. However, these methods require significant computer resources when dealing with complex photonic crystal fibre designs which need to be simulated multiple times to obtain an optimised design. Such iterative analyses also depend upon the number of input design parameters that need to be optimised.

In 2018, extreme learning machine and deep learning have been used for computing dispersion relations [9] and optimisation of Q-factors [97] for photonic crystals. Here, it is proposed to use machine learning (ML) techniques for computing various optical properties of the PCF. The finite element simulations and artificial neural networks (ANN) were combined for the quick and accurate computation. The focus of this chapter is to design a simple feed forward multilayer perceptron (MLP) model which can be trained quickly to estimate the n_{eff}, A_{eff}, D, and α_c for a PCF structure. The computational platform used was a laptop with Intel Core i7 CPU @ 2.80 GHz, 16 GB RAM having the Windows 10 operating system.

This work is organized as follows. Section 6.2 describes the ANN/MLP concepts and modelling parameters. Section 6.3 presents the assessment of the modeled ANN on testing PCF by comparing their estimations with actual values and computing runtime, and finally the chapter is concluded in Section 6.4.

6.2 PCF modelling with ANN

The ANN/MLP architecture parameters are introduced in this section along with the PCF type that was used for generating the dataset. The first step of the training procedure of

an ANN model is to have a finite and appropriate labeled dataset. This initially generated dataset plays a crucial role for any ANN model. The accuracy of the model depends upon how well the dataset is aligned to the problem to be solved. Different PCFs were simulated by changing some geometric property values and their optical properties were calculated. In this case, the set of PCF geometric data including the diameter of the holes (d), separation between the centre of holes (pitch, Λ), refractive index of the core (n_c), wavelength (λ), and the number of rings (N_r) of a solid core PCF (as shown in Fig. 6.1) were taken as input variables of the labeled dataset. Subsequently, n_{eff}, A_{eff}, D, and α_c values were calculated using Lumerical Mode Solutions for more than 1000 samples, which are considered as the output variables of the labeled training dataset. A_{eff}, D, and α_c values can be defined by using the following equations [98]:

$$A_{eff} = \frac{\left(\iint_{\Omega} |\mathbf{H_t}|^2 \, dxdy \right)^2}{\iint_{\Omega} |\mathbf{H_t}|^4 \, dxdy} \tag{6.1}$$

$$D = -\frac{\lambda}{c} \frac{d^2 \, \mathrm{Re}(n_{eff})}{d \, \lambda^2} \tag{6.2}$$

$$\alpha_c = 8.686 \times 10^6 \, k_0 \, \mathrm{Im}(n_{eff}) \quad dB/m \tag{6.3}$$

where $\mathbf{H_t}$ is the transverse magnetic field vector, Ω is the area enclosed within the computational domain. Re and Im stand for the real and imaginary parts, respectively. c and k_0 are the free-space speed of light and wavenumber, respectively.

An ANN/MLP model with 3 hidden layers having 50 nodes/neurons in each layer was used throughout this chapteras shown in Fig. 6.2. These hidden layers are fully interconnected, which means that each node/neuron of a layer is connected to each node/neuron in the following layer. 10% of the data samples were randomly removed from the training dataset and allocated to the validation dataset to provide an unbiased evaluation for tuning the ANN model parameters (weights and biases). The rectified linear unit (ReLU) [54]

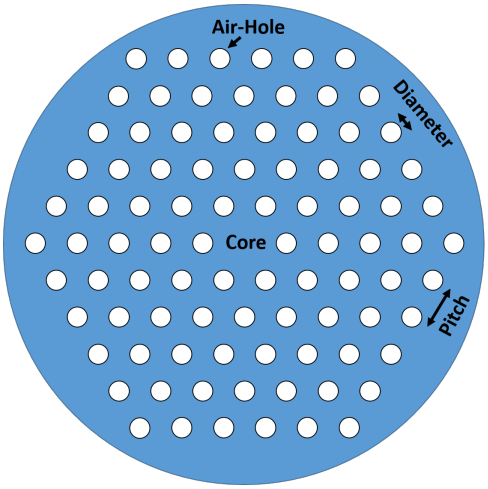


Fig. 6.1 Cross-section of a solid core hexagonal PCF with five rings of air holes.

activation function and Adam [73] optimiser were used for approximating the non-linear function and to optimise the weights during the training process, respectively. The ANN model predicts some outputs after each iteration/epoch. The mean squared error (MSE) value between this predicted and actual output is then calculated and back-propagation [70] was used repeatedly to update the weights of the hidden layers for each epoch. The number of epochs used was decided by the user when the MSE converges to an acceptable value. After optimising the model, having stable MSE value, suitable outputs were generated for the new input data which was not provided during the training process. Machine learning techniques are used to compute various optical properties including effective index, effective mode area, dispersion and confinement loss for a solid-core PCF. These machine learning algorithms, based on artificial neural networks are able to make accurate predictions of the above mentioned optical properties for new/unseen input data for usual parameter space of wavelength ranging from 0.5-1.8 μm, pitch from 0.8-2.0 μm, diameter by pitch from 0.6-0.9 and number of rings as 4 or 5 in a silica solid-core PCF.

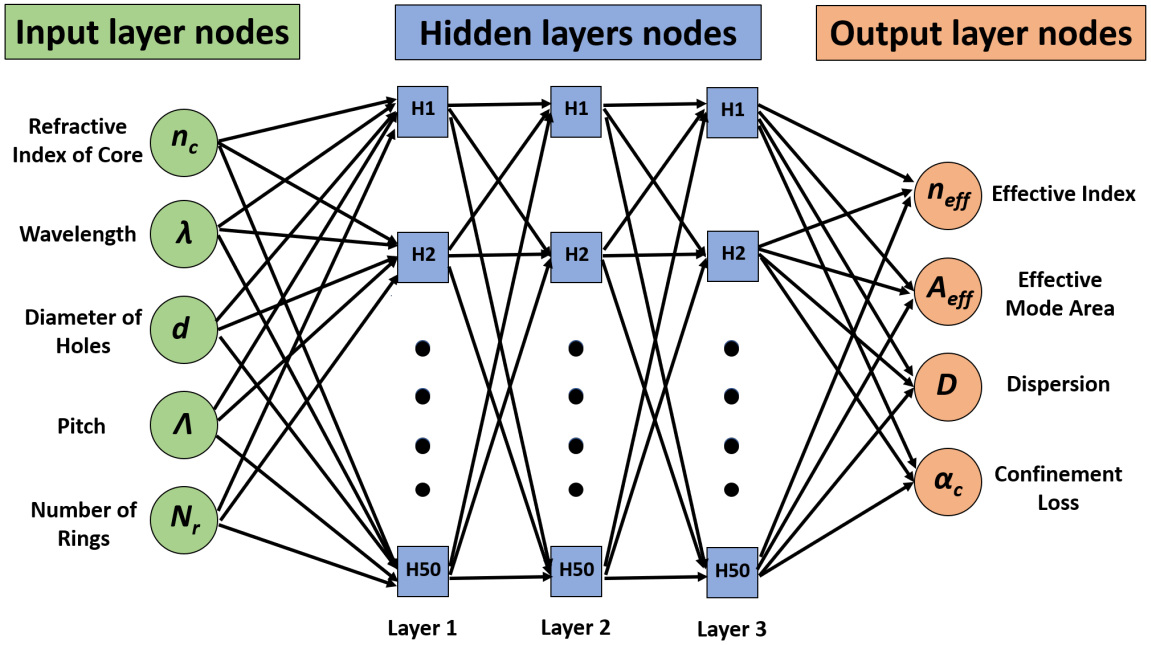


Fig. 6.2 Artificial neural network (ANN) representation with one input layer (5 input nodes), three hidden layers (50 nodes in each layer), and one output layer (4 output nodes).

6.3 Numerical Results and Computation Runtimes

In this section, the trained ANN model was validated by evaluating their outputs for a solid core PCF at an unknown design parameters, and finally the computational runtimes of the ANN model are compared with the numerical simulations.

6.3.1 Effective Index (n_{eff})

Figure 6.3 shows a scatter plot of the n_{eff} values of the dataset used for training of the ANN model. Predicted values of n_{eff} obtained from the ANN model were plotted against the actual values from FEM simulations. Each circle represents a single datapoint. For a well trained model, these values should be aligned closer to the y=x line (shown by the black solid line).

It can be stated that for epochs = 1000, the model is not well trained as the n_{eff} values (shown by blue circles) in the parameter space of 1.15-1.30 are not close to the y=x line.

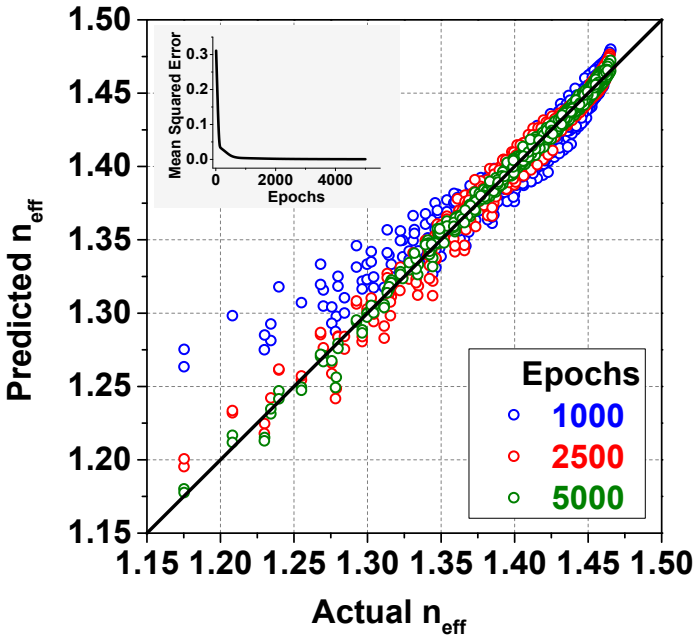


Fig. 6.3 The scatter plot of training dataset produced by ANN for different epochs, comparing n_{eff} values from the simulation (x-axis) and the ANN predictions (y-axis) along with the ideal linear model (y = x). Inset shows the mean squared error (MSE) obtained with epochs when training the ANN model.

This can also be explained from the MSE curve shown in the inset of Fig. 6.3. The MSE gives the average squared difference between the estimated and true values. Predictions were closer to the original values when the MSE values are smaller. It can be observed that the MSE decreases with epochs from 0.31113 for epoch = 1 to 0.00367 for epochs = 1000. For epochs = 2500, MSE reduces further to 0.00134. This implies that the n_{eff} scatter plot should be closer to the y=x line for epochs = 2500, as can be seen by the red circles data. The ANN model was ran until 5000 epochs at which MSE reached a stable value of 0.00065. For epochs = 5000, the trained ANN model for predicting n_{eff} agrees reasonably well with the actual n_{eff} values, being closest to the y=x line, as shown by green circles.

Next, in Fig. 6.4 actual and predicted n_{eff} were compared for different epochs using the trained ANN model at an unknown PCF parameters, Λ = 1.5 μm, d/Λ = 0.7, and N_r = 4. It should be noted that the generated input training dataset did not have any value corresponding to Λ = 1.5 μm. Generally, the n_{eff} of the fundamental mode of PCF decreases

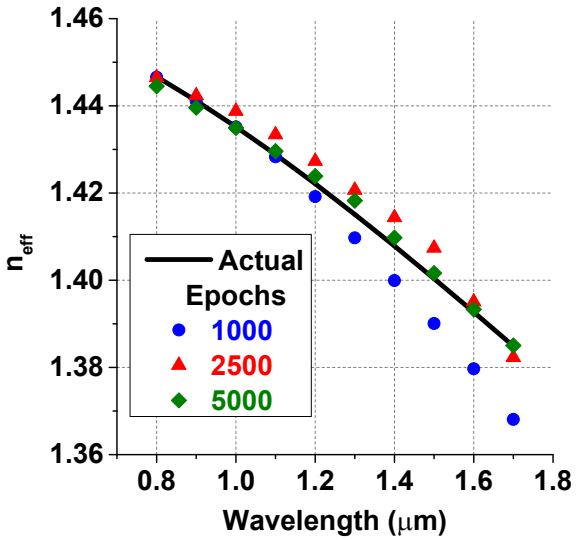


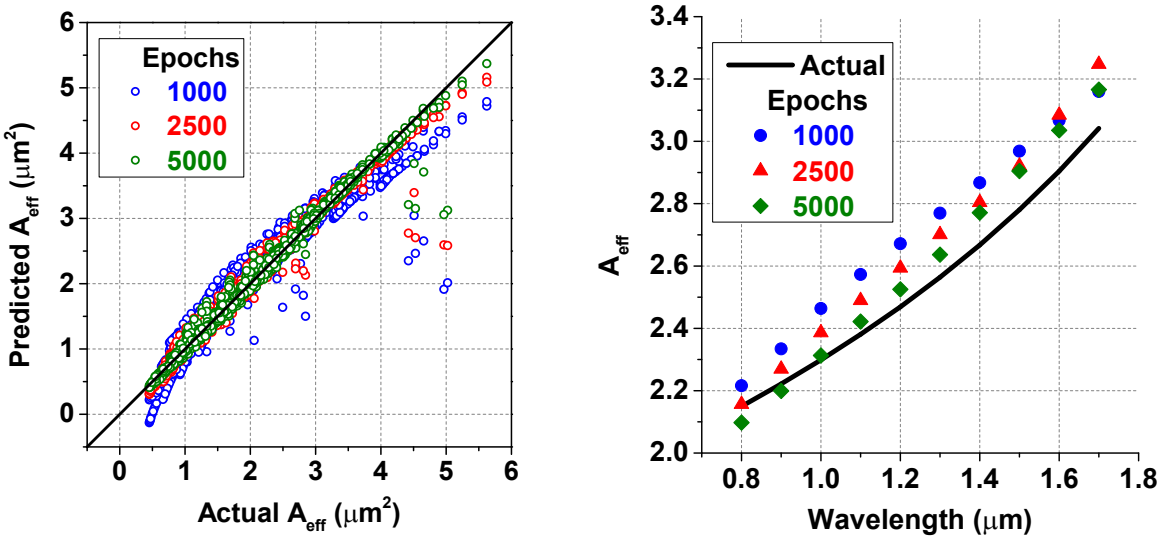
Fig. 6.4 Comparing actual (simulation) and predicted (ANN model) n_{eff} for different epochs at an unknown pitch, $\Lambda = 1.5$ μm, $d/\Lambda = 0.7$, and $N_r = 4$.

with the increase in λ. For epochs = 1000, ANN model is not able to accurately predict the actual pattern (shown by the black solid line) well, and the predicted values (shown by the blue filled circles) steered away from the actual values as λ increases from 1.4 μm to 1.8 μm. When the epochs were increased to 2500 or 5000, the MSE reaches to a more stable value of 0.00134 and 0.00065, respectively. Hence, the predicted and actual values are closer for both cases, having a lower error. This also shows that 5000 epochs are sufficient for the training.

6.3.2 Effective Mode Area (A_{eff})

A_{eff} plays an important role in the context of the waveguiding properties of PCF, and can be calculated using Eq. 6.1. A smaller A_{eff} is useful in applications with enhanced fibre nonlinearity, while a large A_{eff} can be useful in high power transmission applications.

Here, the previously trained ANN model is used to predict the A_{eff} values of the solid core PCF. Figure 6.5a shows that how well the dataset is trained for A_{eff} values for epochs equal to 1000, 2500, and 5000, shown by blue, red and green circles, respectively. It can be seen that some datapoints for epochs equal to 1000 and 2500 are not well trained, especially



(a) (b)

Fig. 6.5 (a) The scatter plot of training dataset produced by ANN for different epochs, comparing A_{eff} values from the simulation (x-axis) and the ANN predictions (y-axis) along with the ideal linear model (y = x), (b) Comparing actual (simulation) and predicted (ANN model) A_{eff} for different epochs at an unknown pitch, Λ = 1.5 μm, d/Λ = 0.7, and N_r = 4.

for A_{eff} values greater than 4 μm^2. When the epochs were increased to 5000, these datapoints become closer to the y=x line. The epochs can be further increased but this increases the simulation time, and may also lead to the overfitting problem. This trained ANN model was then used to predict the A_{eff} values for unknown PCF parameters, Λ = 1.5 μm, d/Λ = 0.7, and N_r = 4, as shown in Fig. 6.5b.

The A_{eff} data corresponding to these parameters was never recorded or provided during the training of the model. However, the ANN model was still able to predict them. Figure 6.5b shows the curve of actual and predicted A_{eff} values for epochs = 1000, 2500, and 5000. When epochs were increased from 1000 to 5000, the predicted A_{eff} values become closer to actual values, which is also justified by the fact that MSE is decreased with increase in epochs, as shown in inset of Fig. 6.3. However, it should be noted that even for epochs = 5000, the solutions were not very accurate especially in the wavelength range from 1.3-1.7 μm.

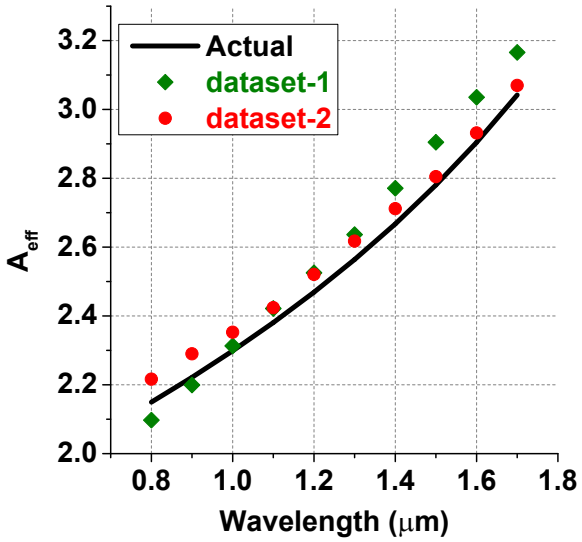


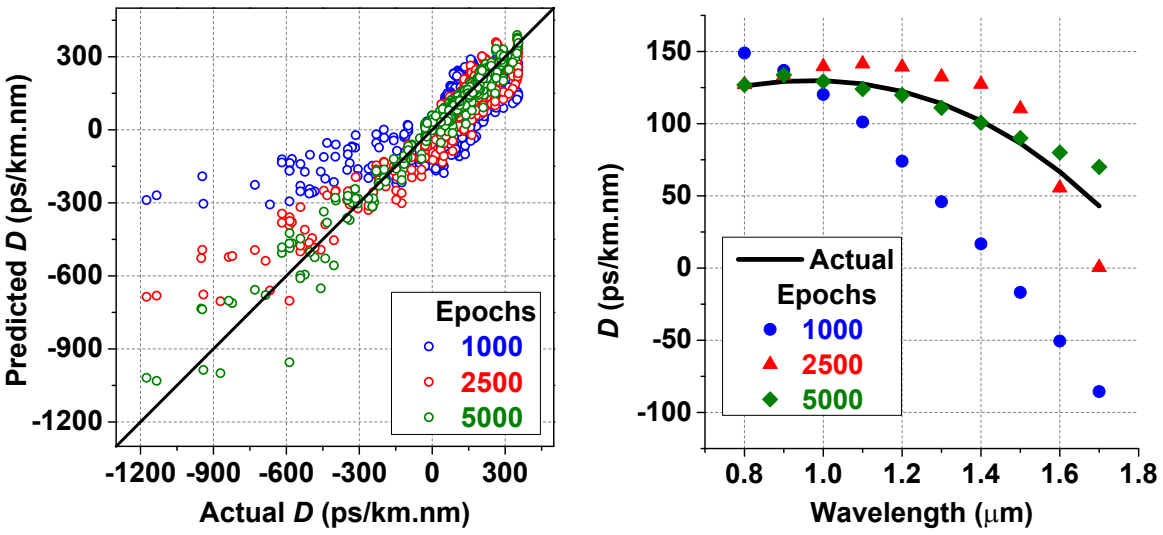
Fig. 6.6 Comparing actual (simulation) and predicted (ANN model) A_{eff} for different datasets at an unknown pitch, $\Lambda = 1.5$ μm, $d/\Lambda = 0.7$, and $N_r = 4$.

Figure 6.6 compared the actual and predicted A_{eff} values for two different datasets at epochs = 5000. Dataset-2 is composed of all the values of dataset-1 and has some additional datapoints especially in the wavelength range from 1.3-1.7 μm, where previous predictions were poor. It can be seen that the error between the actual and predicted values are further reduced when ANN model is trained using dataset-2 (shown by red circles) in comparison to initial dataset-1 (shown by green diamonds), which clearly shows improved predictions in the higher wavelength range. Figures 6.5b and 6.6 show that both number of epochs and dataset quality play an important role during the training of the ANN model.

6.3.3 Dispersion (D)

The chromatic dispersion (D) of a PCF is an important parameter for many applications, such as supercontinuum generation, which may be calculated using Eq. 6.2. The D depends on the second order derivative of n_{eff} with respect to λ.

The scatter plot showing training of the ANN model for D is shown in Fig. 6.7a. It can be observed that at epochs = 1000, the predicted D values significantly deviate from



(a) (b)

Fig. 6.7 (a) The scatter plot of training dataset produced by ANN for different epochs, comparing D values from the simulation (x-axis) and the ANN predictions (y-axis) along with the ideal linear model (y = x), (b) Comparing actual (simulation) and predicted (ANN model) D for different epochs at an unknown pitch, $\Lambda = 1.5\ \mu$m, $d/\Lambda = 0.7$, and $N_r = 4$.

the actual D values, especially when D is less than -400 ps/km.nm, as shown by the blue circles. This error is reduced if the model is trained until epochs = 2500 (red circles) or 5000 (green circles). However, at epochs = 5000, it can be observed that some datapoints are not close to the y=x line when D is less than -600 ps/km.nm. This error comes from the neural network modelling as there were insufficient number of datapoints in the parameter space of D between -1200 and -600 ps/km.nm. This leads to underfitting of the trained model for D values. The model is more biased to be trained towards D values greater than -300 ps/km.nm as there are more datapoints in that range. Increasing the datapoints in the lower D range further improves the training and accuracy of the ANN model for D calculations.

Actual and predicted D values for unknown $\Lambda = 1.5\ \mu$m, $d/\Lambda = 0.7$, and $N_r = 4$ are shown in Fig. 6.7b. At epochs = 1000, the error (or gap between the simulated and ANN values) is more because the ANN model is not able to learn the curve shape well and predicts it almost as a straight line. Increasing the epochs to higher values of 2500 (red triangles) or 5000

(green diamonds) reduces this error gap. It can be noted that epochs = 5000 were sufficient to predict the *D* pattern in this case.

6.3.4 Confinement Loss (α_c)

Confinement loss (α_c) in PCF depends on the structural parameters, that is, changing the lattice pitch and number of rings in the cladding. As the confinement loss can result in worse system performance, it is necessary to optimize its incidence. It depends on the imaginary part of the complex effective index, n_{eff}, and can be computed using Eq. 6.3.

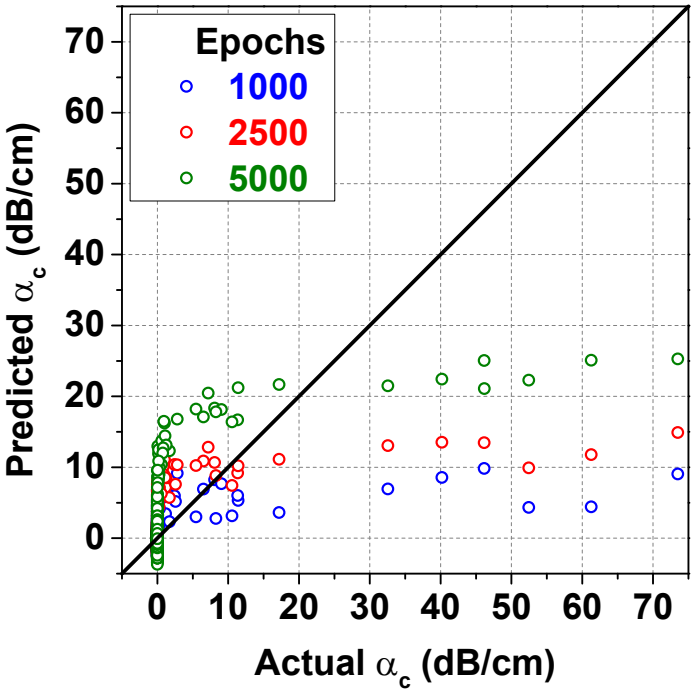


Fig. 6.8 The scatter plot of training dataset produced by ANN for different epochs, comparing α_c values from the simulation (x-axis) and the ANN predictions (y-axis) along with the ideal linear model (y = x).

Figure 6.8 displays the predictions of α_c of the fundamental mode with actual values when using the trained ANN model. It can be seen that for different epoch values of 1000, 2500 and 5000, the scatter plot looks significantly different. The majority of the scatter plot values lie in α_c ranging from 0-10 dB/cm for epochs equal to 1000, 2500 and 5000 as shown

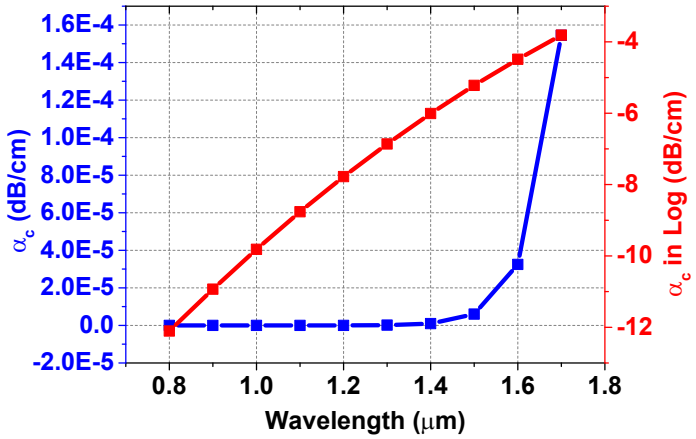
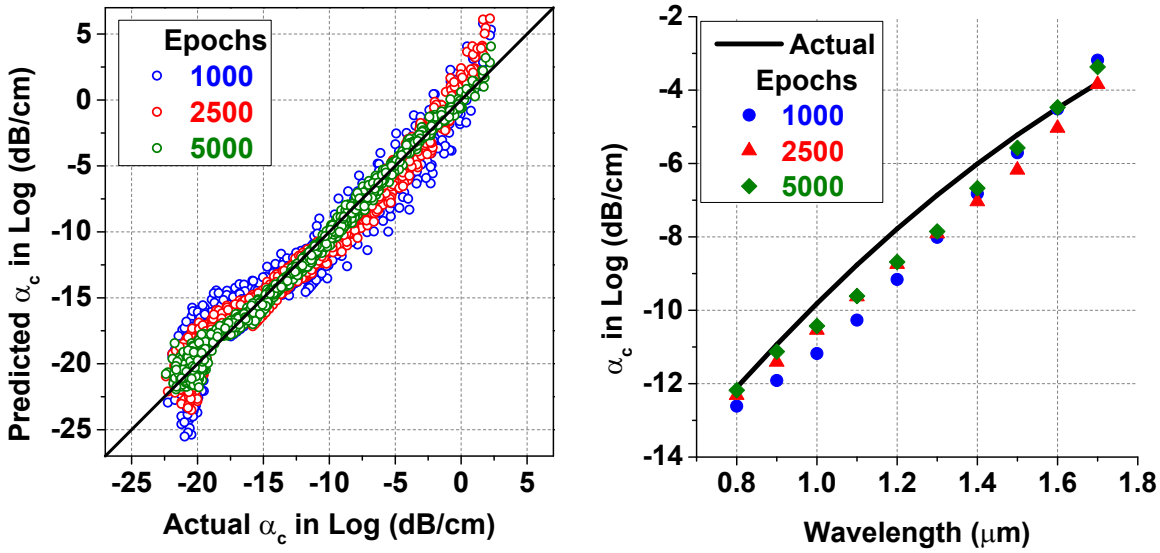


Fig. 6.9 Actual values of α_c from the simulation and in logarithm with wavelength for a general case.

by the blue, red and green circles, respectively. Only a few values are present for α_c greater than 20 dB/cm and these values are not close to the y=x line which means that the data corresponding to these α_c values is not well trained. To understand the reason behind this, the general trend of α_c with λ for a particular set of PCF parameters has to be understood, as shown by the blue line in Fig. 6.9. It can be clearly seen that for λ in the range 0.8-1.4 μm, α_c values are quite small and it rapidly increases up if λ is greater than 1.5 μm. This implies that when the dataset is recorded for different Λ, d/Λ, N_r, and λ, the majority of the output α_c values lie in one zone with only a small number of values in the another zone. Generally, the devices operate in narrow bandwidth around 1.55 μm, but a broad range between 0.8–1.7μm was considered to show the validity of the algorithm. When the majority of values lie in one zone, the ANN model becomes biased towards these values. Similar behaviour was observed in this case when training the ANN model for α_c values of the PCF.

The solution proposed to avoid this problem was to take the logarithm ($\log_{10}$) of the initially collected α_c values when the dataset was collected for different variations in Λ, d/Λ, N_r, and λ. Taking the logarithm converts the values to new values having regular variations, which can be efficiently trained using the ANN model. This can be seen by the red line in Fig. 6.9, which is the logarithm of the values of the blue line. The red line represents the

logarithm of the general trend (absolute values) of α_c with λ for the particular set of PCF parameters. The logarithmic values of α_c were trained by the ANN model as in the training case for the n_{eff}, A_{eff} and D, and α_c. During testing of the PCF with unknown parameters, first use the ANN model trained using the logarithm of α_c values and in the final step take the anti-logarithm to obtain the predicted absolute α_c values and compare their accuracy with the actual α_c values.



(a) (b)

Fig. 6.10 (a) The scatter plot of training dataset produced by ANN for different epochs, comparing α_c values in logarithm from the simulation (x-axis) and the ANN predictions (y-axis) along with the ideal linear model (y = x), (b) Comparing actual (simulation) and predicted (ANN model) α_c in logarithm for different epochs at an unknown pitch, Λ = 1.5 μm, d/Λ = 0.7, and N_r = 4.

Comparison of the predicted and actual α_c values using logarithmic conversion during training of the ANN model is shown in the Fig. 6.10a. For different epochs, it can be observed that the scatter plot values are closer to the y=x line, showing a well trained model. Without taking the logarithm of α_c values, the same ANN model was not trained well as shown earlier in Fig. 6.8. The α_c values obtained using logarithmic conversion from the simulation and ANN models for different epochs at an unknown Λ = 1.5 μm, d/Λ = 0.7, and

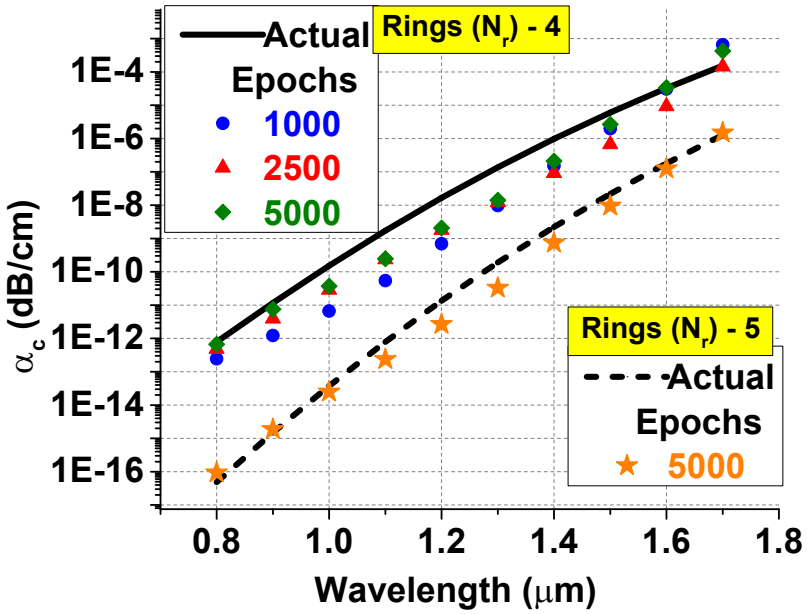


Fig. 6.11 Comparing actual values from the simulation and ANN model for different epochs at an unknown pitch, $\Lambda = 1.5$ μm, $d/\Lambda = 0.7$, and $N_r = 4$ or 5.

$N_r = 4$ are shown in Fig. 6.10b. As the epochs were increased from 1000 to 2500 to 5000, the error decreases for α_c.

Next, the trained ANN model was used to predict and compare the α_c values for different epochs with the actual α_c values. The same trained model was used to efficiently predict α_c for different $N_r = 4$ or 5 at an unknown $\Lambda = 1.5$ μm, $d/\Lambda = 0.7$, as shown in Fig. 6.11. But before comparing, anti-logarithm was taken as the ANN model was trained with α_c in logarithm values. For $N_r = 4$ and epochs = 1000, it can be seen that predicted α_c values (shown by blue filled circles) are still far from actual α_c values (shown by solid black line). As the epochs were increased to 5000 (shown by green diamond symbols), the estimations become closer to the actual α_c values. Similarly, the predicted and actual α_c values have been shown when $N_r = 5$ is taken for epochs = 5000. Predicted and actual values of α_c are closely matched as shown by orange star symbols and dotted black line, respectively. This implies that the ANN model performs better when logarithm of α_c values are taken to

train the model, and 5000 epochs are sufficient to closely predict the actual values for a PCF structure.

6.3.5 Computing Performance

The runtime to train the artificial neural network model depends on the training parameters including dataset size, number of hidden layers, number of neurons in each hidden layer and number of epochs, among others. For the particular model using 3 hidden layers with 50 nodes in each layer running for 5000 epochs, it takes around 20 seconds to train the model with the generated dataset. Once the training was completed, model weights and parameters were saved in the computer. The next step was to predict the output for unknown inputs, which typically takes only a few milliseconds to compute. This prediction was carried out using the already saved weights rather than first training the ANN again. On the other hand, the numerical computation using Lumerical Mode Solutions requires a few minutes for each point to be calculated, and can take even longer if a denser mesh is used. In this case, multiple outputs can also be predicted simultaneously within milliseconds with the ANN model when sets of input parameters are given, while it can take even longer in numerical simulations where multiple sweeps might be required.

6.4 Summary

In summary, machine learning techniques have been employed to accurately predict the important properties for a silica core photonic crystal fibre design. This chapter has shown that how to predict the effective index, effective mode area, dispersion and confinement loss of a photonic crystal fibre within milliseconds, in contrast of needing a few minutes with standard numerical simulations. Three hidden layers with 50 neurons in each layer were used throughout the investigation, which offer rapid convergence and sufficient accuracy in

predicting outputs for unknown geometric dimensions. The comparison between actual and predicted values in computing various optical properties are shown, and the errors between these values decrease when the number of epochs are increased, as the mean squared error value reduces with the number of epochs. The machine learning models are an efficient alternative and have potential to support computation solvers for both forward and inverse problems. In future, the code can be easily extended to different core materials as well as hollow core photonic crystal fibre.

Code and Data Availability

The open source machine learning framework - PyTorch [69] was used to build and test the artificial neural networks. The datasets and complete Python code used to generate the presented results during the current study are available via https://github.com/sunnychugh/ML_PCF and Appendix B. An additional dataset for testing purposes is also provided at the mentioned link.

Chapter 7

Thermal Compensation of Phase Mis-matching in a Silicon Directional Coupler

7.1 Introduction

The development of Si photonics can be traced back to the pioneering work carried out by Soref *et al.* [99] in mid-1980s. Following that, there has been a rapid increase of research interest for integrated silicon photonic technology [2–4, 100, 101], based on the silicon-on-insulator (SOI) platform. Strong modal confinement due to a high refractive index contrast allows a small bending radius which facilitates large-scale integration of photonic devices and thus enhanced functionalities with an unprecedented level of flexibility and scalability. It has the potential to reduce power consumption along with enhanced reliability of a Photonic Integrated Circuit (PIC). Potential applications of Si photonics include photonic interconnects, data communication, telecommunication, optical signal processing, optical storage, electro-optical logic, and others [99].

Although complementary metal-oxide semiconductor (CMOS) fabrication technology is continuously improving, it is never perfect and more advanced devices need strict fabrication tolerances. Heating elements are often used to tune the optical properties of Si photonic devices, to stabilize operating temperatures, and to compensate for fabrication variations. Reconfiguration of Si photonics devices can be actively achieved using free-carrier-plasma dispersion [102], all-optical tuning, electrooptic [103], active thermooptic techniques [104–106], but these approaches increase the power consumption and complicate the device with active control circuitry. On the other hand, post-fabrication trimming of the devices [107] has also been considered but this increases the fabrication cost. The thermooptic effect can be efficiently used to tune photonic devices with negligible insertion loss and has been employed for the implementation of reconfigurable filters and add-drop multiplexers (ROADMs), dispersion compensators, and switches [108]. The thermooptic effect can be utilized by heating the photonic devices, which can be achieved by integrating metallic heaters close to the Si device.

Integrating metal-strip heaters to reconfigure Si photonics devices has been widely used exploiting the large thermo-optic coefficient of bulk silicon crystal [109] that can be easily implemented using CMOS front-end technology. The waveguide core is mostly heated using metallic (Ti, TiN, CrAu, NiCr, graphene etc.) heater placed above the waveguide at a distance from the core [104, 108, 110]. Alternatively, metallic heaters can also be aligned parallel to the guiding axis at a safe lateral distance from the core [111]. Heating induces a transverse temperature gradient (ΔT) proportional to the dissipated electric power by the heater and hence, provides an extra effective index difference [112]. However, integrating metallic heater incurs additional optical absorption loss by the metal, but this can be reduced to a low value if the heater is placed at a distance from the waveguide core. It remains a challenging task to design a silicon photonics thermo-optic devices fulfilling the desired features, viz. compact design for large scale integration, lower optical insertion loss, low

power consumption, faster response time, etc. Therefore, a detailed study of the heat transport model is necessary to achieve an optimum design.

This chapter is organised as follows: In Section 7.2, related early work is reviewed. Proposed device design geometry with various heater positions are discussed in Section 7.3. Fabrication characterization details are presented in Section 7.4. Device optimisation for various separations, fabrication inaccuracies, heater power, and losses are presented in Section 7.5, and the chapter is concluded in Section 7.6.

7.2 Related Work

In 2010, Atabaki *et al.* [108] presented simulated and experimental results for the improvement of the reconfiguration speed of the thermally-tuned Si photonic devices, where the effects of device parameters such as width, thickness, and distance from the waveguide were studied. In 2013, Masood *et al.* [113] compared different resistive heaters compatible with CMOS processes: resistors in doped silicon, Ni-silicide, and Tungsten metallization. They demonstrated that the choice of the specific type of heater is determined by factors such as thermal speed (rise and fall times) instead of heating efficiency. In 2016, Kaushal and Das [114] presented a model to estimate the temperature profile along a single-mode SOI rib waveguide integrated with an active microheater in the slab region. The Fabry-Perot modulation technique has been used to extract the differential change in temperature, sensitivity, and response time of the microheater. Recently, Bahadori *et al.* [115] presented a modelling approach for characterizing the stationary and transient thermo-optic responses of microring resonators in Si photonics platform using integrated microheaters. They investigated pulse width modulation (PWM) drive scheme for microheaters and experimentally demonstrated its use in stabilizing microring resonators. Gupta *et al.* [110] presented a detailed theoretical and experimental study of metal-microheater integrated silicon waveguide phase-shifters. It has been also shown that temperature sensitivity can be improved significantly by proper

choices of waveguide design parameters, closer proximity of microheater to the waveguide core, and suitably folding the waveguide microheater phase-shifter system.

7.3 Device Geometry

Unwanted change in waveguide dimension deteriorates performance of all photonic devices, but more critically that of a directional coupler. Directional couplers are important components for many key photonic devices, such as mode demultiplexers [79], polarization splitters [81], polarization rotators [80], spot-size converters [78], etc. The light from a tunable laser source is input through one waveguide. The 3D schematic diagram of the proposed device design is shown in Fig. 7.1, which consists of two silicon cores with two Cu nanoheaters placed close to each waveguide. The light introduced to one port slowly transfers back and forth between these two waveguides because of the periodic power transfer between the adjacent waveguides. By choosing the distance between the waveguides and the length of the coupling region, it is possible to couple any desired fraction of the light from one waveguide to another. The cladding material is SiO_2. Silicon waveguide and the corresponding Cu nanoheater placed on the top position of each waveguide is separated by a distance, g_H. Here, two silicon waveguides are surrounded by SiO_2 as the cladding material. Cu nanoheaters are placed on top of each waveguide separated by a distance, g_H. A minimum separation between the waveguide and heater restricts the undesirable modal field overlap with the metal electrode, which can cause polarization dependent attenuation [116]. The silicon waveguides are separated from the silicon substrate by a silicon dioxide (SiO_2) buffer layer. The refractive indices of silicon and silicon dioxide are taken as n_{Si} = 3.478 and n_{SiO2} = 1.444, respectively at the 1550 nm operating wavelength (λ_0).

The cross-section of the directional coupler with Cu nanoheaters is shown in Fig. 7.2. It is assumed here that the desired rectangular cross-section for both of the Si waveguides are $w1 \times h = w2 \times h$ = 500 nm $\times$ 220 nm. Typically, a 1.5–2.0 μm thick SiO_2 buffer layer is used,

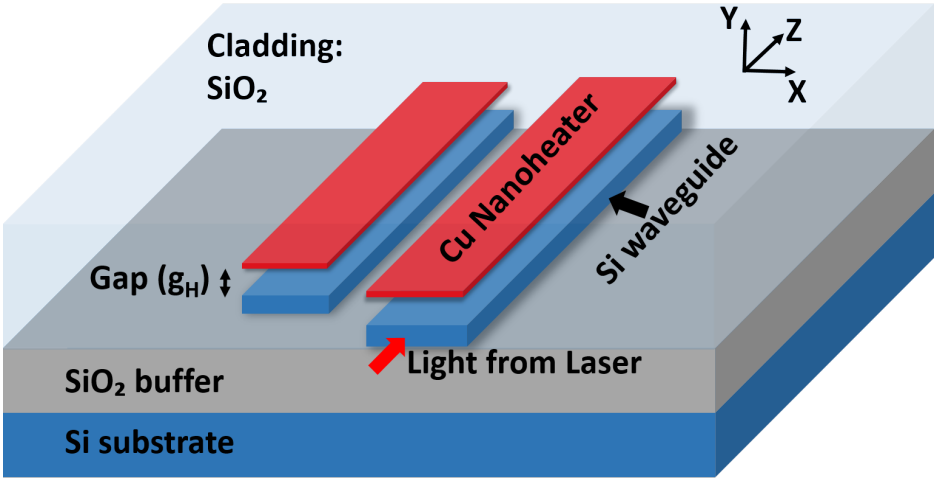


Fig. 7.1 3D schematic diagram of a directional coupler with Cu nanoheater at top position of each waveguide.

but as the mode does not extend beyond the SiO_2 layer, the lower Si substrate was ignored in all simulations. However, other combinations including an asymmetrical directional coupler for use as a polarization rotator and mode splitter can also be considered. The coupling strength of light from one waveguide to another is very sensitive to the separation between the guides, which is denoted by s. The coupling length of a directional coupler is calculated by [117]:

$$L_c = \frac{\pi}{(\beta_e - \beta_o)} = \frac{\lambda_0}{2(n_e - n_o)} \tag{7.1}$$

where β_e and β_o are the propagation constants of the even and odd supermodes, respectively. Here, n_e and n_o are the effective indices of these even and odd supermodes, respectively.

The separation between the Cu nanoheater (height = h_H) and Si waveguide is denoted by g_H. The plasmonic absorption loss is induced by the presence of a metal nanoheater, hence g_H should be carefully chosen to produce a low plasmonic absorption loss, which is given by [118]:

$$\alpha_p = 20\log_{10}[\exp(-\beta_{im} l_0)] \tag{7.2}$$

where β_{im} is the imaginary part of the propagation constant and l_0 is the propagation distance.

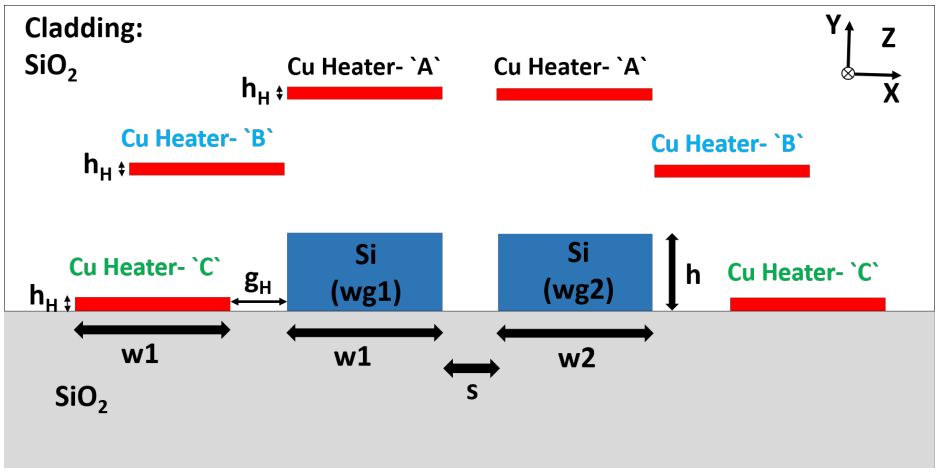


Fig. 7.2 Cross-section of the directional coupler with Cu nanoheater at three different possible positions: A (top), B (top-left), and C (bottom-left).

It is important to ultimately choose a design that can function acceptably for different types of imperfections expected from a given manufacturing process. Deviations in waveguide dimension results in degradation of the optical devices. Standard deviations (0.83–1.00 nm in width and 0.24–0.30 nm in height) in a waveguide cross-section of 440-nm-wide/220-nm-thick waveguide for a single 300-mm wafer has been reported in [119]. In this work, the focus was on the effects of deviations in width of the Si waveguide due to manufacturing/fabrication tolerances on the behavior of a directional coupler. The heaters can be placed either directly at the top of the waveguide, shown as 'A' in Fig. 7.2 or above the waveguide but laterally shifted, shown as 'B' or on the outside but at a lower height, shown as 'C'. As it is not possible to predict the likely fabrication error, so two sets of heating electrodes need to be incorporated in a more practical design. In the case of an asymmetric error, the heater located closer to the waveguide which is narrower is activated. In the case of an identical error, when both waveguides are slightly narrower, although they remain phase matched, but resulting reduced coupling length can be corrected by switching ON both the heaters. Commercially available simulation tools, COMSOL Multiphysics and Lumerical were used in this study to investigate the effects of thermal tuning of the Si waveguides directional coupler with

the metallic heaters at different positions. The thermal tuning and coupling efficiency were studied by using the 2D thermal and 3D FDTD simulations, respectively.

7.4 Fabrication Process

The waveguide structure shown in the Fig. 7.2 can be constructed by etching two straight silicon ridges into a commercially available SOI wafer. A commercially available SOI wafer typically has a 220 nm thick Si layer on top of a buried silica layer. First, a metal mask of similar waveguide width needs to be prepared by a plasma etch of the photoresist. After metal masking, the rest of the Si section can be etched away to the bottom SiO_2 layer using the Reactive Ion Etching (RIE) tool to form a Si ridge waveguide with vertical sidewalls. However, any change in its dimensions may violate the phase matching condition of a waveguide coupler. This fabrication imperfection could be rectified by incorporating micro-heaters. After etching and fabrication of the Si ridges, a SiO_2 layer could be deposited on top of the waveguides, to act as a cladding layer. A metal microheater can be deposited at a certain optimised position using the thermal evaporation technique.

7.5 Thermal Tuning of a Directional Coupler for Fabrication Inaccuracies

Various tuning approaches have been reported, including, the integrated micro-heater, electrooptic, and photothermal heating [113–115]. Among them, the integrated micro-heater is often preferred. However, a relatively thick upper cladding is required for the micro-heater configuration, which limits the tuning speed. Cu nanoheaters are used for thermal correction with uniform heating power along the length of the device. Various thermal and electrical parameters used for Cu, Si, and SiO_2 in calculating thermo-optic effects, which are taken

as default values from the library of the Lumerical Device Heat Transport simulator are included in Table 7.1.

Table 7.1 Thermal and Electrical parameters used for the simulation.

Property	**Cu**	**Si**	$\boldsymbol{SiO_2}$
Material Mass Density (kg/m^3)	8933	2330	2203
Specific Heat Capacity ($J/kg.K$)	385	711	709
Thermal Conductivity ($W/m.K$)	397	148	1.38

7.5.1 Temperature Profiles

The 2D steady state simulation was carried out using the Lumerical DEVICE with a total heating power (P_h) of 25 mW, which was uniformly distributed along the waveguide of 25 μm length. The gap between heater and waveguide, g_H, was initially taken as 400 nm and the separation between the waveguides was maintained constant as s = 200 nm. Heat generated by the metal heater flows to the sink (substrate) through the cladding and core materials. Such heat dissipation not only depends on the waveguide materials, but also on the size and shape of the waveguide and heater. Figure 7.3 shows the transverse temperature profile, T(x,y), around the Si waveguides and cladding when the Cu nanoheater was located at the top of the $wg1$ (position - A). The temperature decreases gradually towards the substrate, having a maximum value in close proximity to the heater. It can be observed that the temperature in $wg1$ is more than that in $wg2$ as the Cu heater is closer to $wg1$ than $wg2$. The heat dissipation not only depends on the waveguide materials, but also on the size and shape of the waveguide and heater. Analytically, the temperature increase of the heater can be

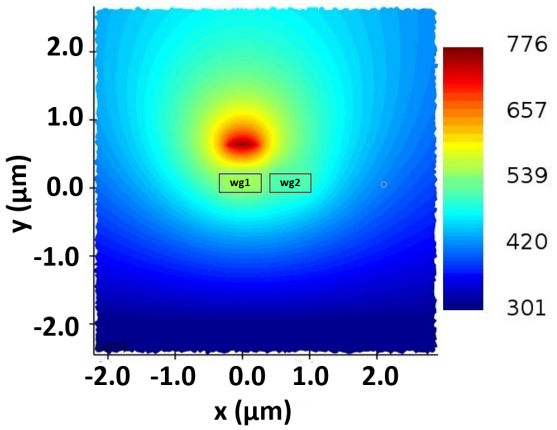


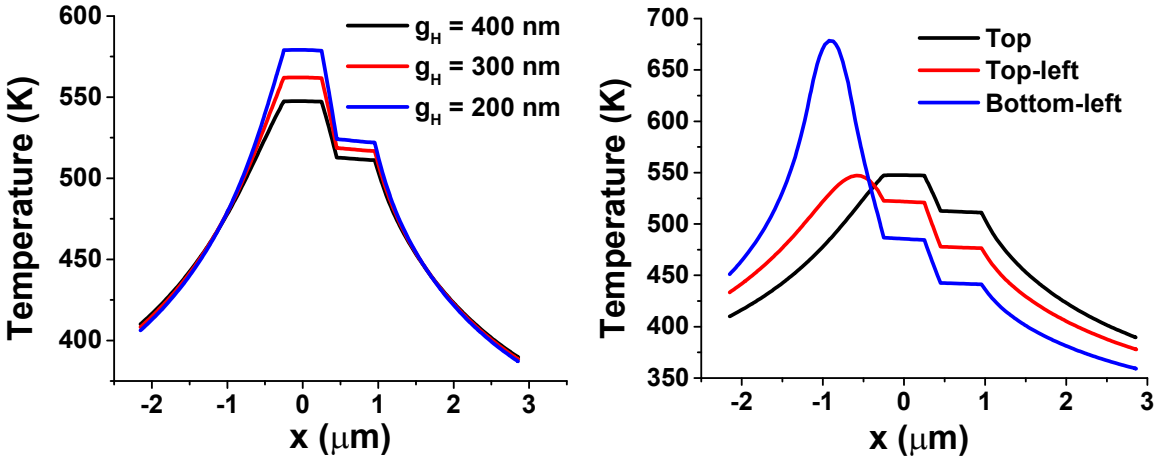
Fig. 7.3 Temperature distribution profile across the directional coupler cross-section with heating power (P_h) of 25 mW, g_H = 400 nm, dW = 5 nm, and s = 200 nm having Cu nanoheater at the top position.

approximately estimated using the following [120]:

$$\Delta T_h = \frac{0.5\ (P_h\ t_w)}{l_h\ w_h\ \kappa_{SiO_2}\ (1+0.88\ \frac{t_w}{w_h})} \tag{7.3}$$

where l_h and w_h are the length and width of the heater, respectively. κ_{SiO_2} is the thermal conductivity of SiO_2. $t_w = g_H + t_{SiO_2}$, where t_{SiO_2} is the thickness of substrate layer, and only half of the heater power contributes to the gradient of temperature in each direction. When P_h = 25 mW, l_h = 25 μm, w_h = 0.5 μm, κ_{SiO_2} = 1.38 $W/m.K$ and t_{SiO_2} is considered to be infinitely extended in comparison to the waveguide dimensions, then Eq. 7.3 results in ΔT_h = 411 K which leads to the heater temperature of approximately 711 K, which agrees reasonably well with the temperature profile shown in Fig. 7.3. However, to obtain the temperature profile T(x,y) more accurately and the resulting effect on the modal properties, a more rigorous numerical approach is needed, as presented below.

Figure 7.4a shows the temperature profile along the x-axis at y = 0 μm (through the centre of the Si waveguides) for three different values of g_H when the Cu nanoheater was placed at the top position with s = 200 nm. The solid black line (g_H = 400 nm) has a peak constant temperature value of approximately 548 K at x = 0 μm, at the centre of $wg1$, and



(a) (b)

Fig. 7.4 Temperature distributions along the x-axis through centre of Si waveguides with P_h = 25 mW, dW = 5 nm, and s = 200 nm having (a) Cu nanoheater at top position for different g_H, and (b) Cu nanoheater at three different positions with g_H = 400 nm.

approximately 512 K at the centre of $wg2$ (x = 0.7 μm) yielding a temperature difference, ΔT = 36 K. The red and blue solid lines show the temperature variations when g_H is 300 nm and 200 nm, respectively. The peak values of temperature in $wg1$ was increased to 579 K, and increased to 522 K in $wg2$, when g_H is reduced to 200 nm. It is clearly shown as the distance to the top heater is reduced, the local temperature inside both waveguides increased as the Si waveguides were heated more due to the reduced gap.

However, it is more critical to observe that the temperature difference between the two waveguides also increases as the heater distance is reduced. The difference in the temperatures (ΔT) between these two waveguide cores were 36 K, 45 K, and 57 K when g_H was 400 nm, 300 nm, and 200 nm, respectively at P_h = 25 mW and s = 200 nm. Therefore, ΔT increases when the gap between the heater and waveguide (g_H) is reduced and thus can be more effective for correcting defects due to fabrication tolerances.

The effect of heater position was also studied. Temperature variations along the x-axis at y = 0 μm for three different heater positions (top, top-left, and bottom-left) are shown in Fig. 7.4b for P_h = 25 mW, g_H = 400 nm, and s = 200 nm. It can be observed that for

the same g_H = 400 nm in all three heater positions, $wg1$ and $wg2$ have higher temperature values when the heater was located in the top position in comparison to the heaters at the top-left and bottom-left positions. However, the temperature difference (ΔT) between these two waveguide cores ($wg1$ and $wg2$) are approximately 36 K, 44 K, or 56 K when the heater was placed at the top, top-left or bottom-left position, respectively. In the case of the heater located at the top-left and bottom-left position, the peak temperature values are not at the centre of $wg1$ (i.e at x = 0), but at x = -0.5 μm and -0.9 μm, respectively. It should be noted that ΔT between the waveguides is higher for the lower left electrode position, and this may give a better option to correct phase mismatching.

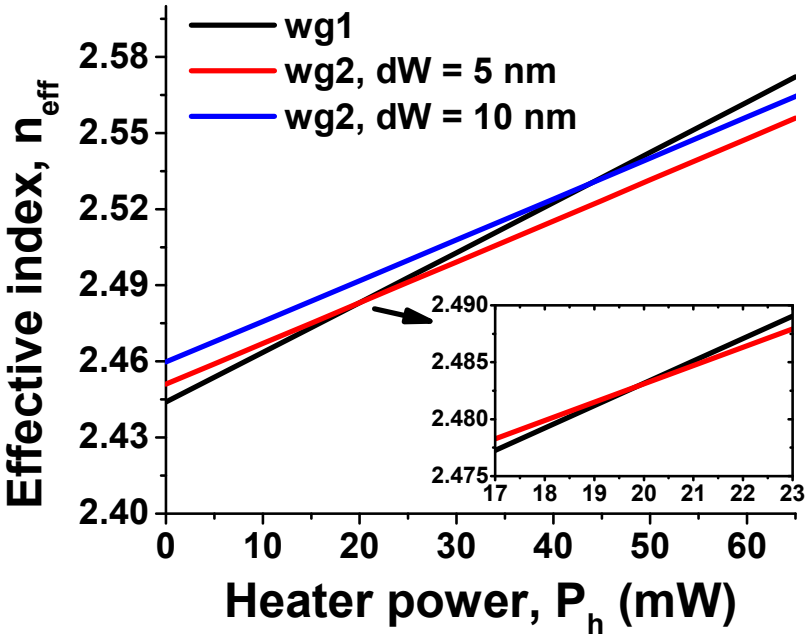
7.5.2 Phase Matching Conditions

Once the temperature profile, T(x,y) was obtained for a particular design, the local refractive index profile with the change in temperature was also obtained using:

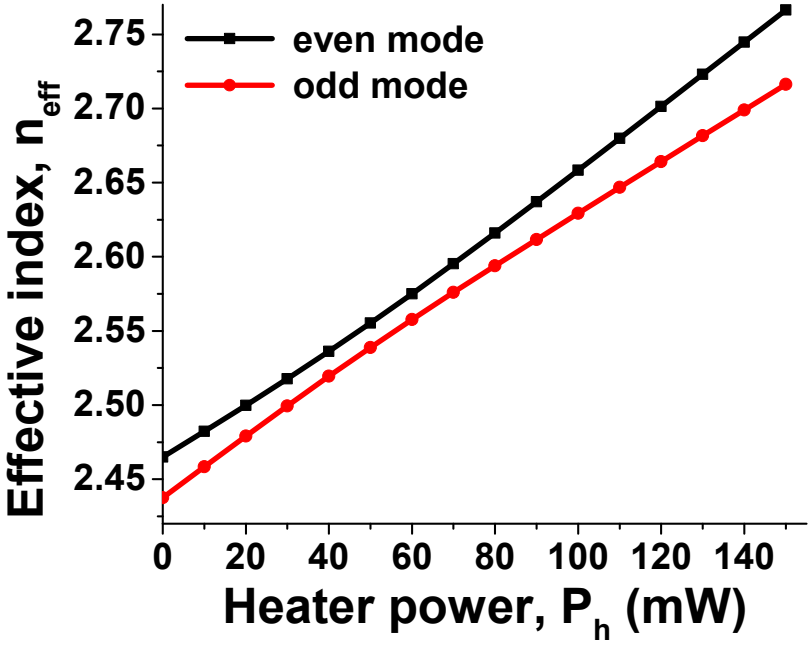
$$n_T = n_{T0} + (T - T_0)\frac{dn}{dT} \tag{7.4}$$

where n_{T0} and n_T are the refractive indices at ambient temperature (T_0) and given temperature (T), respectively [121]. Here, dn/dT for Si and SiO_2 were assumed to be $1.86 \times 10^{-4} K^{-1}$ and $1.10 \times 10^{-5} K^{-1}$, respectively at λ_0 = 1.55 μm.

For the desired performance, fabrication errors, such as deviations from the designed width and height in silicon waveguides should be controlled as accurately as possible. However, in reality it may deviate from the desired design. A case where the fabricated width of $wg2$ is more than its desired design width ($w2$) by dW was considered. As a result, $wg1$ and $wg2$ have different widths ($w1$ and $w2 + dW$, respectively), hence n_{eff} of the isolated $wg2$ is greater than the n_{eff} of $wg1$. Due to the difference in the n_{eff} of the isolated waveguides, the directional coupler is no longer phase matched and the power coupling from one waveguide to another is reduced. The coupling length, L_c is also different than its designed value as



(a)



(b)

Fig. 7.5 Effective index (n_{eff}) variations for (a) isolated waveguides, $wg1$ and $wg2$ with heater power (P_h) for Cu nanoheater at top position for different dW (5 nm and 10 nm), g_H = 400 nm and s = 200 nm, (b) even and odd supermodes with heater power (P_h) for g_H = 200 nm and dW = 10 nm.

the n_{eff} of the isolated waveguides is different due to this fabrication error. Two different cases were considered for dW (5 nm and 10 nm) and their n_{eff} variations with P_h for the fundamental quasi-TE (H_y^{11}) mode of each isolated strip waveguide, $wg1$ and $wg2$ as shown in Fig. 7.5a. The n_{eff} values without heating (P_h = 0) for $wg1$ is 2.44397, and $wg2$ with dW = 5 nm and 10 nm are 2.45101 and 2.45975, respectively. To obtain the phase matching, similar n_{eff} values for both of the isolated waveguides are needed, which can be achieved by activating the Cu nanoheater closer to $wg1$. In this case, initially it was considered that the heater is placed at the top of the waveguide. When the heater near to $wg1$ was activated, it increases the temperature of $wg1$ more than that of $wg2$, and hence the corresponding change in n_{eff} of $wg1$ is higher than that of $wg2$. The blue and red solid lines in Fig. 7.5a show the increase in n_{eff} for the isolated waveguide, $wg2$ with heating, P_h for dW = 5 nm and 10 nm, respectively. Both of these values increase linearly but have different starting values at P_h = 0 mW, as their widths are different. The n_{eff} of $wg1$ shown by the black solid line increases faster with P_h because the local temperature in $wg1$ is higher than $wg2$, $wg1$ being closer to the heater. The n_{eff} of $wg1$ crosses that of $wg2$ at P_h = 20 mW for dW = 5 nm. The zoomed image of this point is shown as an inset. Similarly, when dW = 10 nm, the n_{eff} values of the isolated waveguides $wg1$ and $wg2$ cross when P_h is 44 mW, as shown in Fig. 7.5a by the intersection point of the black and blue solid lines. This clearly shows that a higher P_h is required for phase matching of the two waveguides if the fabrication error (dW) is larger.

The phase matching condition shown in Fig. 7.5a is strictly valid only for the isolated guides. Therefore, the supermodes of the directional coupler were also studied. The variation of the effective indices for the even and odd supermodes are shown in Fig. 7.5b, when dW = 10 nm and g_H = 200 nm. However, these two effective index curves do not cross at any point in the range, similar as for any coupled structure. It can be noted that when phase matching is achieved, the effective indices difference between these two supermodes is a minimum. The n_{eff} values of even and odd supermodes at this phase matching condition

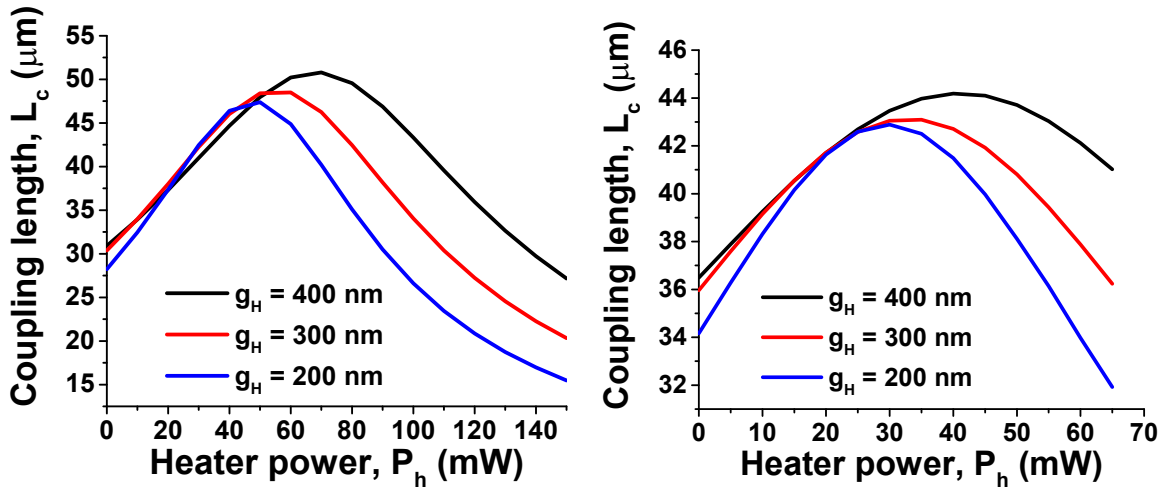


Fig. 7.6 Variations of coupling length (L_c) with heater power (P_h) having Cu nanoheater at top position for different g_H with fixed s = 200 nm, for (a) dW = 10 nm, and (b) dW = 5 nm.

(P_h = 50 mW) are 2.55522 and 2.53887, respectively. Variations of the coupling length with the heating power for different g_H values are shown in Fig. 7.6a for dW = 10 nm. At the phase matching condition, the coupling length, L_c shows a peak value when the effective index difference between the even and odd supermodes is the lowest. For g_H = 200 nm, L_c reached the maximum value of 47.4 μm when P_h was 50 mW, shown by the blue solid line. It can also be observed that the L_c is higher when the separation between waveguide and heater (g_H) is increased, and more heating power, P_h is required to obtain this phase matching condition. L_c values of 48.5 μm and 50.8 μm were obtained at the heating power, P_h = 60 mW and 70 mW for g_H = 300 nm and 400 nm, respectively.

Figure 7.6b shows the variations of L_c with heating power for different g_H when dW = 5 nm. Maximum values of L_c were 42.8 μm, 43.1 μm, and 44.2 μm when g_H are 200 nm, 300 nm, and 400 nm, respectively. The corresponding heater power values required to achieve the phase matching for dW = 5 nm are 30 mW, 35 mW, and 40 mW for g_H values of 200 nm, 300 nm, and 400 nm, respectively. The heater power needed to achieve phase matching for a given heater separation, g_H is smaller for lower dW. It can be noted that the heating

power required to match n_{eff} of isolated waveguides and for phase matching the directional couplers are similar but not the same.

7.5.3 Mode Profiles and Power Confinement

The dominant field profiles of the even and odd supermodes are strictly symmetrical and anti-symmetrical, respectively along the centre of the directional coupler when the coupled waveguides are identical. However, due to the fabrication error, when the two waveguides are no longer identical, the supermodes are no longer strictly symmetrical or anti-symmetrical. This causes unequal excitation of the supermodes and the resulting power transfer is not complete. Power confinements in $wg1$ and $wg2$ were studied as the waveguides were heated. The power confined in the left-half and right-half sections of the directional coupler were calculated. Power transfer from one waveguide to another is maximum only when both modes are equally excited. However, it is noted that the power confinement in the two waveguides by these supermodes is not always equal. Power confinement can be calculated by integrating the Poynting vector over a given region, using the formula:

$$\Gamma = \frac{\iint_{\Delta} Re\,(\mathbf{E} \times \mathbf{H}^*) \cdot \hat{z}\, dxdy}{\iint_{\infty} Re\,(\mathbf{E} \times \mathbf{H}^*) \cdot \hat{z}\, dxdy} \tag{7.5}$$

where Δ represents left-half or right-half region of the coupled structure. The full vectorial $\mathbf{E}$ and the complex conjugate of $\mathbf{H}$ ($\mathbf{H}^*$) fields were used to calculate the modal Poynting vector ($S_z = (\mathbf{E} \times \mathbf{H}^*) \cdot \hat{z}$).

The solid black and red lines in Fig. 7.7 show the power confinement in the left-half and right-half sections of the proposed thermally tuned directional coupler when P_h was increased from 0 mW to 40 mW. In this case the heater was located at the top position with g_H = 400 nm and s = 200 nm with a fabrication error in the $wg2$ width, dW = 5 nm. When P_h was 0, the power confinement in right-half section is greater than that in the left-half section as the first supermode (even-like) is better confined in the wider right side waveguide, $wg2$. The

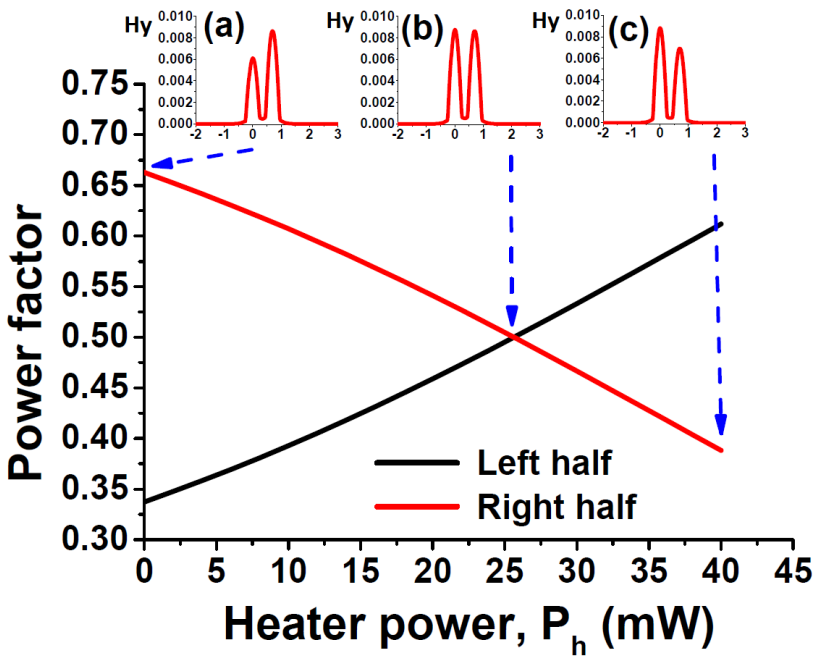


Fig. 7.7 Variation of power confinement in the left-half and right-half sections of the directional coupler for the even supermode with the heater power. Insets show the H_y field profile for even supermode at three different values of P_h, as indicated.

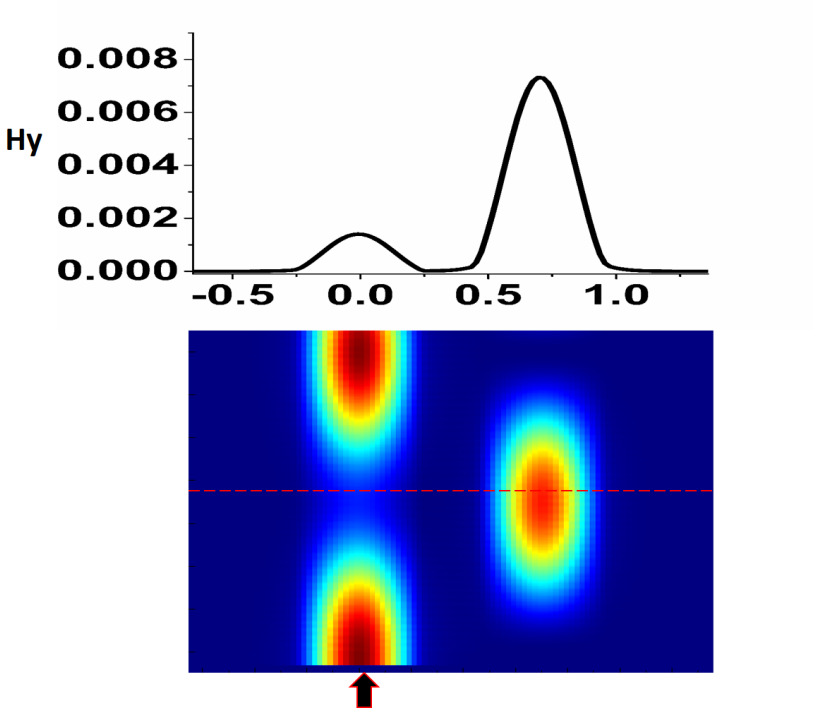
power fraction in the right-half section is 0.6627, while in the left-half section is 0.3372. This is also verified from the H_y field profile shown in the inset (a). At $P_h = 0$, the H_y magnitude in the left $wg1$ is smaller than that in the $wg2$ on the right. This confirms that more optical power is confined in $wg2$ compared to $wg1$ at $P_h = 0$. In this case, power confinement of the second supermode (odd-like) is more on the left side waveguide, $wg1$. When the heater power was increased, it starts heating both the $wg1$ and $wg2$, but as the heater is closer to $wg1$, the temperature in this waveguide is more than the right side waveguide, $wg2$. Hence, the increase in n_{eff} of the $wg1$ rises higher than the n_{eff} of $wg2$, which increases the optical power confinement in $wg1$ and decreases in $wg2$. At $P_h = 26$ mW, it can be observed that the power confinement values in the left and right half sections cross and their values are same and approximately equal to 0.50. At this value of P_h, it can also be observed from the inset (b) that H_y field has almost the same magnitude in both $wg1$ and $wg2$. This crossing point represents a case when the power confined in both the waveguides is equal. In this case, the

even and odd supermode profiles are symmetrical and asymmetrical, respectively. So there would be near ideal power transfer between the waveguides. Increasing P_h further, increases the power in the left-half and decreases in right-half, and the proposed design departs from the ideal condition. At P_h = 40 mW, it can be noted that the power in the right-half section is greater than the power in left-half section. The H_y field profiles at P_h = 40 mW are shown in the inset (c), where it can be seen that the H_y magnitude is smaller in $wg2$ compared to that in $wg1$.

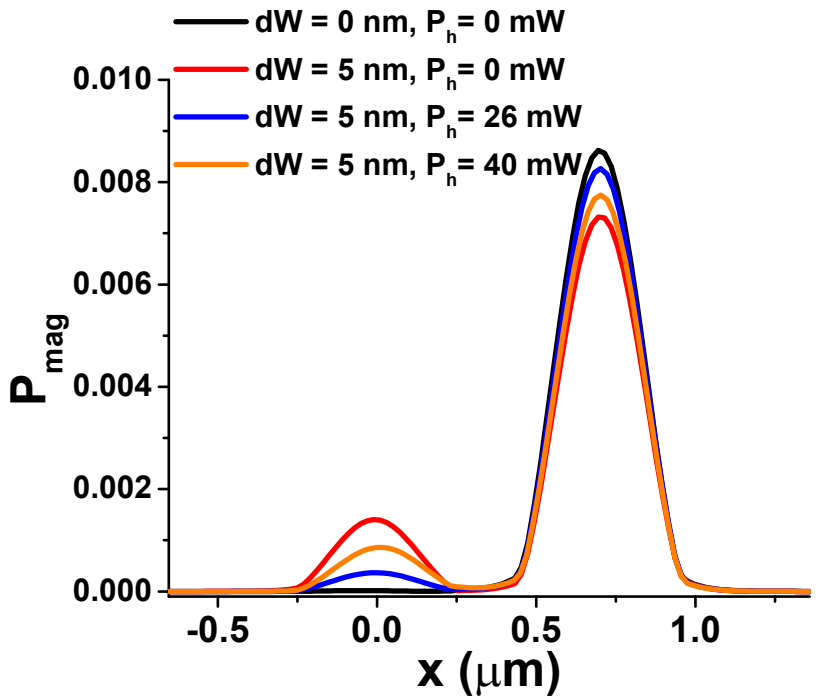
7.5.4 Time Domain Propagation

To verify the design, a time domain evolution of the optical power along the directional coupler has also been studied. For the ideal case, when dW = 0 nm, at the propagation distance, $z = L_{c0}$ = 37.5 μm, it was calculated that 99% of the power is transferred from $wg1$ to $wg2$. When dW = 5 nm and without any compensation by heating, this power transfer between the waveguides is incomplete, which is shown in Fig. 7.8a. The black arrow shows the position from where the light is fed in to $wg1$, which is coupled into $wg2$ and back again to $wg1$. The red dotted horizontal line shows the position of the coupling length for the ideal waveguides (L_{c0}) and remnant optical power is observed across this position without any heating. The black solid line in Fig. 7.8a shows that there is some residual H_y field in $wg1$ when dW = 5 nm and P_h = 0, which implies that power is not fully coupled from $wg1$ to $wg2$.

The H_y field magnitudes for different heater power (P_h) values across the red dotted line (at L_{c0} = 37.5 μm) are shown in Fig. 7.8b. When dW = 0 and P_h = 0, shown by the black line, the H_y field is mostly confined to $wg2$, as this was the ideal intended design. However, when dW = 5 nm with P_h = 0, the H_y field magnitude (shown by a red line) in $wg2$ is reduced in comparison to the ideal case with dW = 0 due to a lack of phase matching. The ratio of H_y peak values in $wg1$ and $wg2$ at z = L_{c0} is approximately 0.192 and this represents only 84% power being transferred to $wg2$. At the optimised heating level, P_h = 26 mW, this ratio is



(a)



(b)

Fig. 7.8 (a) Time domain field profile along with H_y across red dotted horizontal line for dW = 5 nm, P_h = 0 mW (b) H_y field profiles across x-axis at z = L_{c0} for different values of dW and P_h.

reduced to 0.0443, shown by the blue solid line, which represents an increased power transfer of

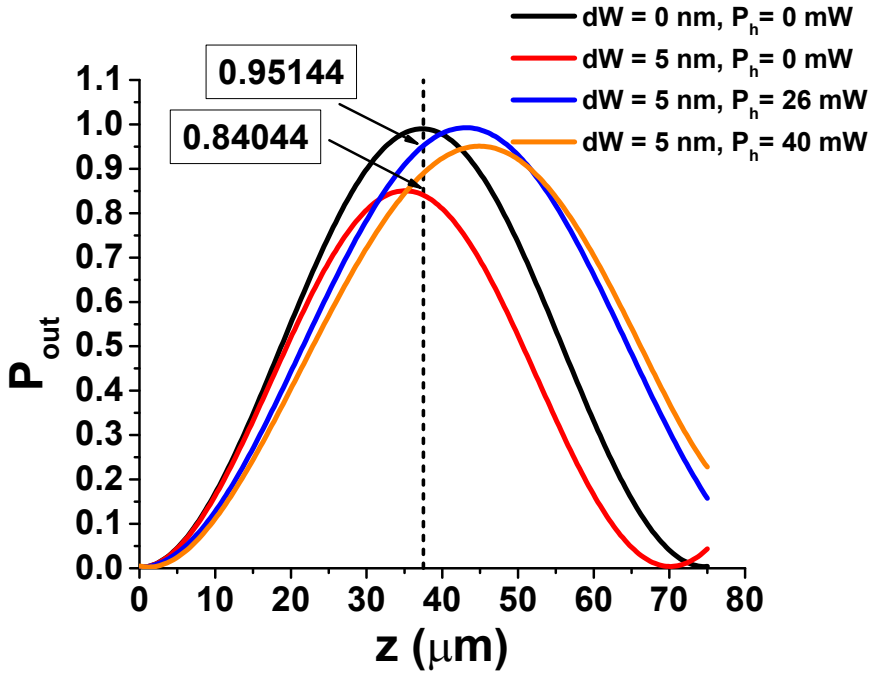
P_h = 40 mW, as shown by the orange line, where the maximum H_y ratio is approximately 0.111, and therefore should be avoided.

The power transfer from one waveguide to another along the propagation direction (z) was also studied. In this case, the heater was located placed at the top position with g_H = 400 nm and s = 200 nm. The power transfer is shown for four different cases in Fig. 7.9a: (a) ideal case without any fabrication error (dW = 0), when $w1$ = $w2$ = 500 nm and heater was deactivated throughout with P_h = 0, shown by the black solid line. In this case, 99% of power was transferred from $wg1$ to $wg2$ at $z = L_{c0}$ = 37.5 μm. This z position is the desired length of the device, being equal to the coupling length, L_{c0} of the ideal design, shown by the vertical black dotted line. When the fabrication error, dW = 5 nm, without heating compensation, case (b), is shown by the red solid line. At this condition, the waveguide widths are different, and hence phase matching condition is not satisfied, which leads to a reduced power transfer from $wg1$ to $wg2$. It is shown here that 84% power is transferred at L_{c0}, which represents a 15% drop in the power due to this fabrication error of 5 nm in $wg2$. In case (c), when heating was introduced, 26 mW of optimised heating power was required to attain the field matching condition for dW = 5 nm, shown by the blue solid line. It can be observed that again 99% of the power can be transferred from $wg1$ to $wg2$ inspite of the fabrication errors provided the heater is activated to the correct heating power. However, this maximum power is attained at a new coupling length (denoted by L_c, which in this case is equal to 43 μm), and not at L_{c0}. In this case, the coupling length is increased as the n_{eff} of the isolated waveguides also increases due to an increase in the temperature. An enhanced mode confinement with higher n_{eff} of the even and odd supermodes represents less evanescent field outside core and thus leads to a longer coupling length value. Although, like in any other design, the power transfer at the intended device length, L_{c0} is the main concern, and not at the new coupling length, L_c.

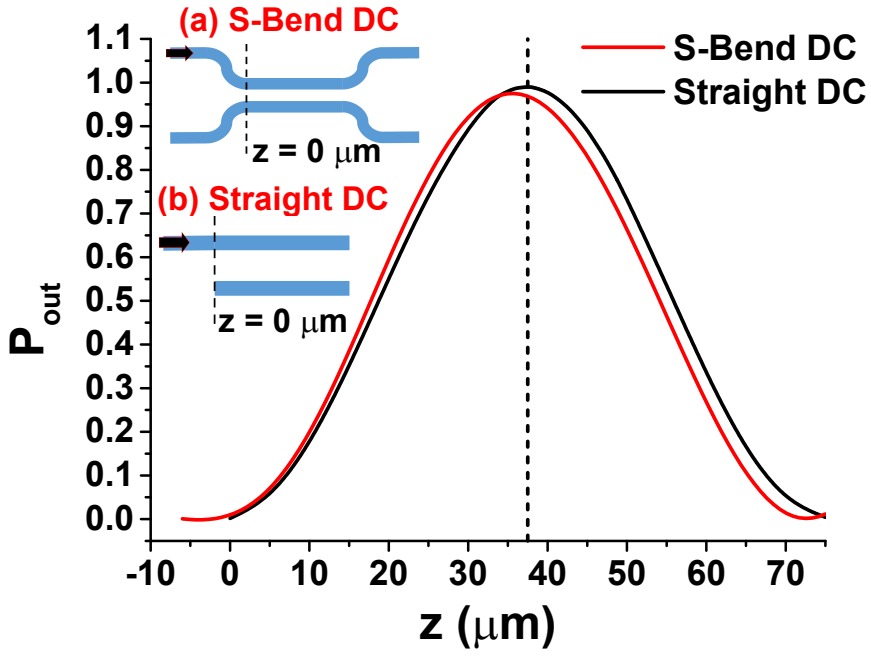
As can be seen from the intersection point of the black dotted line with the blue solid line in Fig. 7.9a, approximately 95% of power is transferred at z = L_{c0}. This can be further increased by chosing different positions of the heater with respect to the waveguides. A heater power of 40 mW was used for case (d), shown by the orange solid line, which shows that power transfer at L_{c0} and L_c are approximately 89% and 95%, respectively, which are less than the values when P_h is 26 mW. Therefore, it was observed that by providing the optimum heater power of 26 mW, power transfer (P_{out}) between the waveguides of directional coupler can be increased from 84% to 95% at the ideal coupling length, L_{c0}, for the fabrication error dW = 5 nm with the heater located at the top position.

In practice, directional couplers are fabricated with S-bends which induce extra coupling. The effect of this coupling in the S-bend section can be accurately calculated using a modal solution approach [122]. For this structure, it has been calculated that the length of the straight section needs to be reduced by 1.5 μm when the radius of the bend section, R_{bend} was 5 μm to compensate for the effect of coupling in the S-bend. Also, maximum power transfer with S-bends is reduced to 98% compared to 99% when only the straight waveguides section was considered as shown in Fig. 7.9b. The concept of compensation for fabrication tolerance is similar, and thus it is sufficient that thermal simulation has been shown for a simpler case with only a straight waveguides section.

Variations of transferred power to adjacent waveguide, P_{out} with the heater power (P_h) are shown in Fig. 7.10 at two different positions along the propagation direction. It shows the maximum power transfer (shown by the red solid line) at the corresponding coupling length, L_c which also changes with heating. Although, the maximum power transfer can be obtained at L_c, in real devices the power transfer at L_{c0} = 37.5 μm is more important, which is shown by the black solid line. This location is fixed at the design stage and cannot be changed following the fabrication, as possible error could not have been anticipated. It should be noted that the L_c values depend on the heater power. It can be observed that the



(a)



(b)

Fig. 7.9 Power transfer, P_{out} along the propagation direction (a) with heater at top position for different values of dW and P_h, (b) for S-bend and straight directional couplers, having parameters values: g_H = 400 nm, dW = 5 nm, and s = 200 nm.

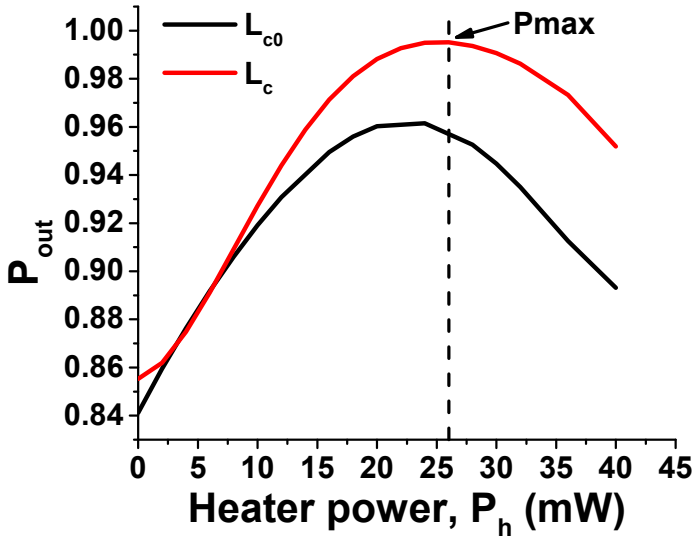
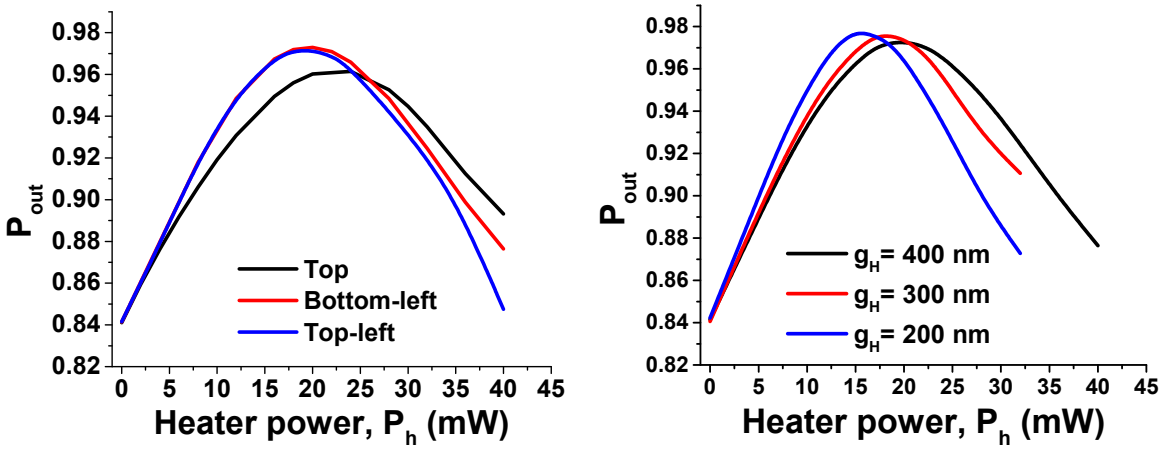


Fig. 7.10 Power transfer, P_{out} with heater at top position at L_{c0} and L_c, having parameters values: g_H = 400 nm, dW = 5 nm, and s = 200 nm.

P_{out} values at L_{c0} and L_c are approximately 84% and 86%, respectively for $P_h = 0$. As the P_h is increased, P_{out} increases until it achieves a maximum value at the field matching condition. The maximum value of P_{out} at L_c is always more than the maximum value of P_{out} at L_{c0}, because when the device was heated, the isolated n_{eff} and the coupling length of the heated device was different than in the case of the original design, and the optimised parameters were obtained at this new coupling length, L_c. If P_h is increased further, the field matching condition is destroyed, and P_{out} again starts decreasing at both distances, L_{c0} and L_c. As P_{out} at L_{c0} is the main concern, it can be seen that introducing the heater improves the P_{out} at L_{c0} from 84% to 96% at the top heater position when P_h = 24 mW.

As mentioned earlier, the position of the electrode can influence the performance of these designs. Figure 7.11a shows the power transfer (P_{out}) at L_{c0} with the heaters located at different positions, i.e top, top-left and bottom-left positions for fabrication error, dW = 5 nm. It can be observed that for all three heater positions initially P_{out} is low at L_{c0} (approximately 84% at $P_h = 0$), but as the P_h is increased, the design reaches the optimum condition, and P_{out} achieves respective maximum values at different P_h for different heater positions. The



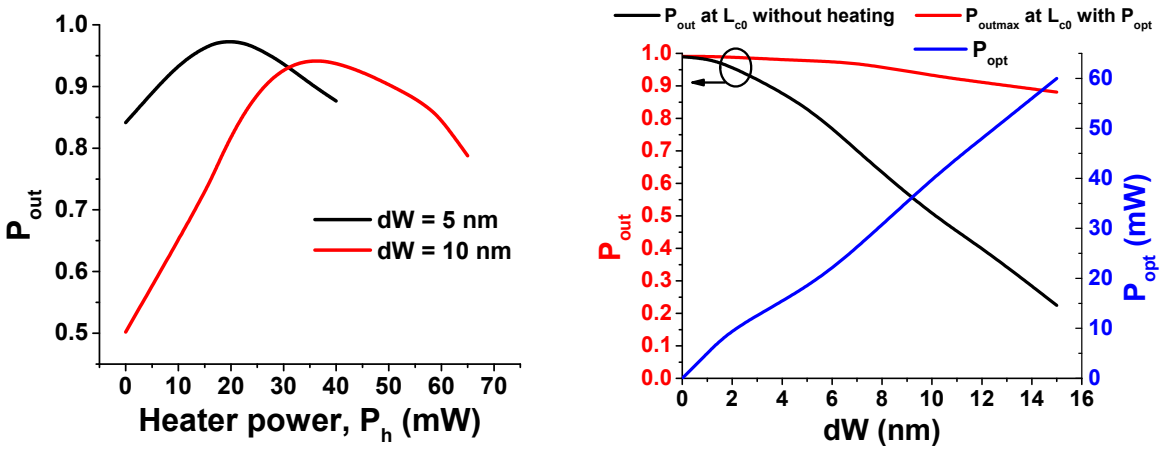
(a) (b)

Fig. 7.11 Power transfer, P_{out} with heater power (P_h) at L_{c0} (a) for different heater positions having g_H = 400 nm, dW = 5 nm, and s = 200 nm, (b) when heater is at bottom-left position for different g_H having dW = 5 nm and s = 200 nm.

bottom-left, top-left, and top heater positions yield a field matching condition at P_h of 20 mW, 20 mW, 24 mW with maximum P_{out} value of 97%, 97%, 96%, respectively. It can be observed that heaters located at the bottom-left and top-left positions attain field matching at a lower P_h than the heater at the top position and power transfer is slightly better than the top electrode placement.

On the other hand, if the fabrication error (dW) is 10 nm, P_{out} is only 50% when no heating was used. This confirms that the power transfer efficiency rapidly deteriorates as the fabrication error increases. It can be noted that bottom-left, top-left, and top heater positions have field matching condition at P_h values of 35 mW, 38 mW, 46 mW and provide maximum P_{out} values of 94%, 92%, 88%, respectively at z = L_{c0}. The heater at the top-left and top positions attain a smaller maximum value than the bottom-left heater and require more heater power to achieve maximum P_{out}. It can be noted that the bottom-left electrode would therefore be the best choice, giving a higher P_{out} and it requires a relatively small power to achieve this condition.

Next, the power transfer from one waveguide to another (P_{out}) for different gaps between the heater and waveguide was compared. When the electrode was located above $wg1$, then



(a) (b)

Fig. 7.12 Power transfer (P_{out}) at L_{c0} with heater at bottom-left position having g_H = 400 nm and s = 200 nm (a) for different dW, (b) when heater is deactivated (black line) and activated (red line). The blue line depicts the heater power (P_h) needed to attain field match condition at respective fabrication error, dW.

it is shown in Fig. 7.4a that the differential temperature, ΔT, between the waveguides is increased when the distance to the heater, g_H, is reduced. Similarly, for the heater at the bottom-left position, these ΔT values were calculated as approximately 44 K, 48 K, and 54 K for g_H of 400 nm, 300 nm, and 200 nm, respectively. As a result it can be seen from Fig. 7.11b that a smaller value of P_h is needed when g_H is 200 nm in comparison to when g_H is 300 nm or 400 nm in order to obtain the maximum value of P_{out} at L_{c0}. Black, red and blue solid lines show that P_h of 20 mW, 18 mW, and 16 mW are required to obtain a maximum P_{out} at L_{c0} with the heater located at the bottom-left position when g_H was 400 nm, 300 nm and 200 nm, respectively. At these conditions, the calculated maximum P_{out} values were 97.3%, 97.6%, and 97.7%. When the heater was located closer to waveguide, less heating power was required to obtain the field matching condition and at this field matching condition, the maximum value of P_{out} was obtained.

It has been clearly shown that when the fabrication error dW = 5 nm, it is possible to improve power coupling to 97% using a 20 mW heater power. However, when the dW became larger, more heater power was needed to correct this shortcoming due to the

fabrication error. Figure 7.12a compares the P_{out} at L_{c0} when the fabrication error, dW was 5 nm and 10 nm for the bottom-left heater position. When the heater was deactivated with P_h = 0, P_{out} for the case with dW = 10 nm was only 50%, much smaller than the P_{out} with dW = 5 nm, that was obtained as 84%. As the P_h is increased, the temperature and n_{eff} of the isolated waveguides increase and P_{out} reaches maximum at a particular value of P_h. Also, the P_h required to attain maximum P_{out} is different for designs having a different dW. As can be seen from Fig. 7.12a, when dW = 5 nm, at optimum heating power, $P_h = P_{opt}$ = 20 mW, a maximum P_{out} is 97%. On the other hand, for dW = 10 nm, a maximum P_{out} = 94% can be achieved when P_{opt} = 35 mW. A higher value of P_h is required when the fabrication error in width (dW) is greater in order to achieve the field matching condition.

The optimum heater power needed (P_{opt}) and maximum power transfer possible (P_{outmax}) were studied for a range of possible fabrication errors. Variations of P_{out} at L_{c0} when the heater was deactivated (P_h = 0) and activated, plotted versus varying dW for the heater at the bottom-left position are shown by black and red solid lines, respectively in Fig. 7.12b. When dW = 0 (ideal waveguides with same width), there is no need to activate the heater and P_{out} is approximately 99 % with P_h = 0. As the fabrication error, dW was increased keeping the heater deactivated (P_h = 0), P_{out} at L_{c0} rapidly decreases to only 22% when dW = 15 nm, shown by the black solid line. On the other hand, if the heater is activated to provide the optimised power value (P_{opt}) in order to the attain field matching condition for different fabrication errors (dW), it is found that the drop in P_{out} is far less (shown by the red solid line) in comparison to when the heater is not used (shown by the black solid line). P_{outmax} at L_{c0} dropped from approximately 99% to only 88%, even when the dW was increased from 0 to 15 nm. This demonstrates that P_{out} at L_{c0} can be significantly increased by choosing an optimised heater power, P_{opt} and the variations of optimised P_{opt} required with dW is shown using the blue solid line. As expected, the required P_{opt} increases almost linearly as dW is increased.

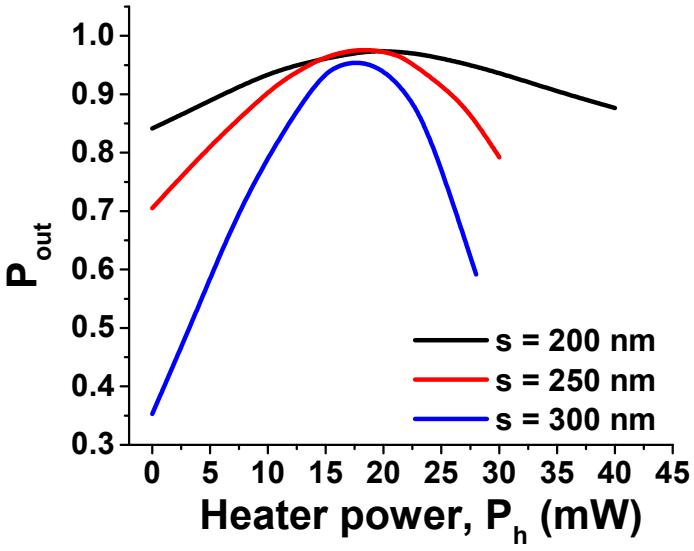


Fig. 7.13 Power transfer (P_{out}) at L_{c0} for different waveguide separation, s with heater at the bottom-left position having parameters values: g_H = 400 nm and dW = 5 nm.

When the separation between the waveguides, s was reduced, then the coupling length (or device length) also reduced, but in this scenario cross-talk is likely to increase. A smaller separation also makes the device more resilient with the fabrication tolerance due to a stronger coupling. The effect of separation between the waveguides on the thermal compensation was studied. As the separation (s) increased, ideal coupling length (L_{c0}) increases as the light needs to propagate to a longer distance before coupling from one waveguide to another. The P_{out} variations at the respective L_{c0} of the design with the heater power (P_h) is shown in Fig. 7.13 for separation values of 200 nm, 250 nm, and 300 nm. It can be observed that keeping $P_h = 0$, P_{out} drops from approximately 84% to 70% when s was increased from 200 nm to 250 nm, as shown by the black and red lines, respectively. For s = 300nm, P_{out} reduces further to only 35%, as shown by the blue line. As the P_h increased, P_{out} also increases for all the three cases. It can be noted that P_{out} is more sensitive to the varying heating power when the separation is larger. In all 3 cases, P_{outmax} is in the range between 96–97% when the P_h value is in the range of 18–20 mW. Further increasing the P_h again decreases the P_{out}, as field

matching is destroyed. An approximately similar P_h is required to attain the field matching condition for all three values of waveguide separation, s.

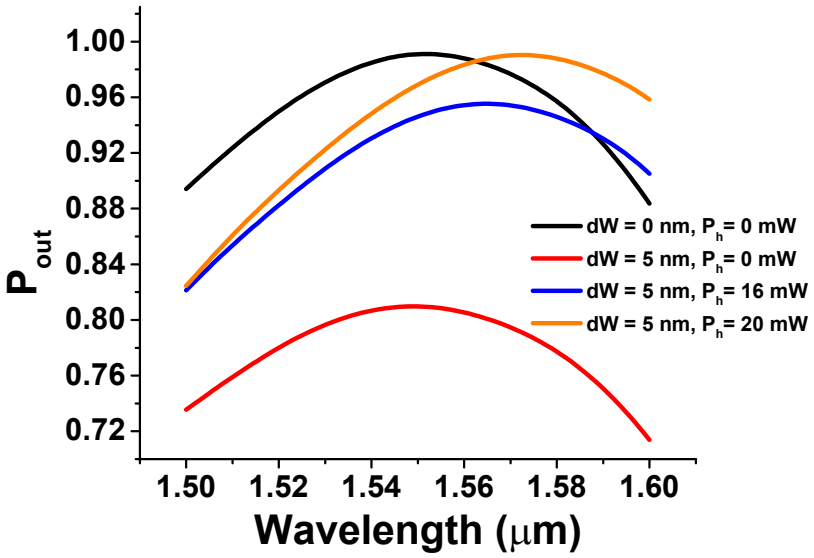


Fig. 7.14 Power transfer (P_{out}) at L_{c0} with operating wavelength (λ) for heater at bottom-left position having parameters values: g_H = 400 nm, dW = 5 nm, and s = 200 nm.

When all the parameters were optimised including dW, g_H, and s, the effect of the variation of operating wavelength (λ) variation could be studied when the heater was located at the bottom-left position. P_{out} with the operating wavelength is shown by the black solid line in Fig. 7.14, when waveguides are assumed to be ideal without any fabrication error (dW = 0) and keeping the heater deactivated. P_{out} is maximum (approximately 99%) at λ = 1.55 μm, as the directional coupler parameters (especially length, L_{c0}) were optimised at 1.55 μm with dW = 0, g_H = 400 nm, and s = 200 nm. P_{out} was reduced when λ changed from 1.55 μm, as the wavelength dependent coupling length does not match the device length. When dW = 5 nm with P_h = 0, shown by the red solid curve, P_{out} is reduced when the wavelength deviates from its design value of 1.55 μm, but these values are much smaller than the ideal case without any fabrication error shown by the black solid curve. The effect of heater power values of 16 mW and 20 mW was investigated to obtain the field matching condition dependence on the wavelength when there was a fabrication error (dW) of 5 nm. A

heating power, P_h = 20 mW is needed to increase P_{out} to approximately 97% at λ = 1.55 μm. The maximum value of P_{out} is 99% when λ = 1.57 μm, as shown by the orange solid curve. This larger output at λ = 1.57 μm is due to the fact that smaller L_c at higher wavelength partially compensates the increased L_c due to the fabrication error.

7.5.5 Absorption Loss

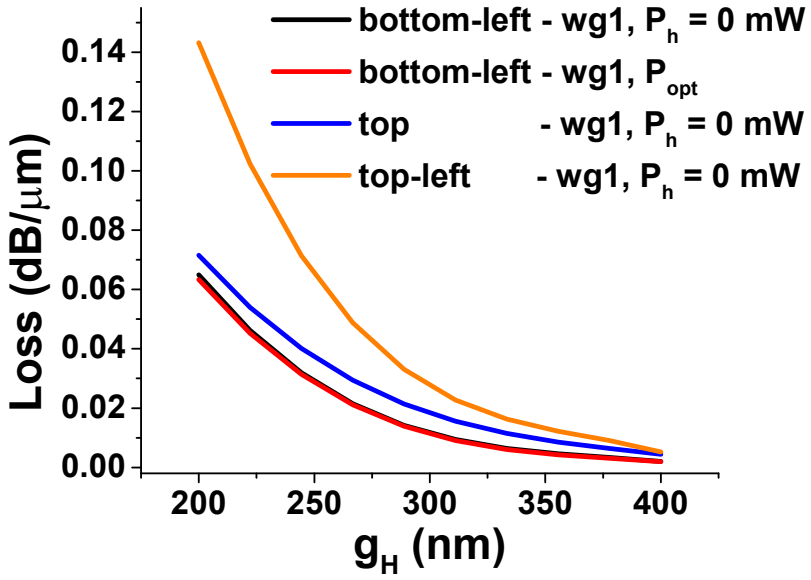


Fig. 7.15 Absorption loss variation with g_H for heater at bottom-left, top or top-left position.

The introduction of a metal Cu nanoheater to the directional coupler also introduces the absorption loss (α_p). α_p was calculated for isolated waveguides when the heater was placed at the bottom-left, top or top-left positions for different separations (g_H), as shown in Fig. 7.15. α_p falls from 0.0654 dB/μm to 0.0020 dB/μm for the isolated waveguide, $wg1$ (shown by the black solid line) when g_H was increased from 200 nm to 400 nm keeping the heater power at 0. Using the optimum heater power (P_{opt}), the absorption loss, α_p decreases from 0.0632 dB/μm to 0.0018 dB/μm when g_H changes from 200 nm to 400 nm, as shown by the red solid line in Fig. 7.15. Modal loss is slightly reduced with heating, as the confinement in the Si core increases with heating, which reduces the overlap of the optical mode field to the

lossy metal electrode. When the heater was placed at the top position, α_p decreases from 0.0715 dB/μm to 0.0044 dB/μm as g_H increases from 200 nm to 400 nm, shown by the blue line. When the heater is placed at the top-left position, modal loss decreases from 0.1432 dB/μm to 0.0052 dB/μm as g_H increases from 200 nm to 400 nm, shown by an orange line. It can be observed that the bottom-left position of heater has smaller waveguide modal loss in comparison to the heater at the top or top-left position.

7.6 Summary

In summary, a rigorous study on the performance analysis of metallic heaters integrated with a silicon directional coupler has been presented to identify the optimum design. The analyses considers the optimisation of various parameters including the position of the heater (top, top-left, or bottom-left), the gap between heater and waveguide (g_H), the separation between waveguides (s), and heater power (P_h) to obtain the maximum power transfer (P_{out}) between the waveguides of a directional coupler. A fabrication error (dW) in the range 0–15 nm is assumed in the width of the Si waveguide which is thermally compensated. P_{out} drops to approximately 84% when there is fabrication error (dW) of 5 nm. Therefore, the use of metallic heater is proposed for thermal compensation achieving the field matching condition when the fabrication inaccuracy is present. It is shown here that for dW = 5 nm, P_{outmax} can be increased from 84% to 97.3% when heater was placed at the bottom-left position by using only 20 mW of heating power when g_H = 400 nm and s = 200 nm. In this case the modal loss due to the metallic electrode is 0.0020 dB/μm. When g_H was reduced to 200 nm, P_{outmax} increased to 97.7% at a slightly lower heater power, P_h = 16 mW, but modal loss slightly increased to 0.0654 dB/μm. Bottom-left heater performs better as it requires less heating power (P_h), yet gives higher P_{out} and also incurs slightly lower plasmonic modal loss. For dW = 10 nm, P_{out} increased from approximately 50% (when P_h = 0 mW) to 94% (when P_{opt} = 35 mW) with electrode placed at the bottom-left position. It can be noted that absorption

loss, α_p decreased from 0.0632 dB/μm to 0.0018 dB/μm at optimum heater power (P_{opt}) when g_H was increased from 200 nm to 400 nm.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this dissertation, consideration was given to modelling, optimisation, and thermal compensation using metallic heaters for nanophotonic waveguides. One of the main objectives was to develop a finite element time domain approach as an alternative to finite difference time domain method. The finite element method is more advantageous than other numerical methods which was briefly discussed in Chapter 2. The formulation of FETD equations for the two-dimensional propagation and mesh discretisation was presented in Chapter 3. Space discretisation of the computational domain was obtained using Gmsh, an open source mesh generator, while linear elements were used for time discretisation. The computation domain was terminated by a PML boundary to reduce the reflections in the simulations. Several benchmarking results have been shown in Chapter 4 which shows the validity of the in-house FETD code written in C++ programming language and using multithreading to reduce the simulation times. Free space propagation of the electromagnetic wave has also been shown and the speed obtained with the in-house code was compared with the normalised value. The propagation of an electromagnetic wave through planar waveguide has been shown for two different input sources: a point source and mode source (generated using

in-house FEM code). Furthermore, a directional coupler has been used to show the coupling of an electromagnetic wave from one waveguide to another. Regular and irregular mesh arrangements were considered and compared in terms of accuracy and dip ratio. Irregular meshes can use dense mesh inside the core, while coarse mesh can be used outside the core to further reduce the computational times in comparison to when using a regular mesh arrangement.

Time domain simulation can take minutes to hours to obtain results for each set of device dimensions. Testing of various device dimensions for a particular device through a parameter sweep can take even longer to obtain an optimised design. Another objective of this research was to use the artificial intelligence or machine learning techniques to hasten this optimisation process. An in-house code has been developed using machine learning regression approach to quickly predict effective index, power confinement for different nanophotonic waveguides, as presented in Chapter 5. A detailed discussion about the architecture of the used artificial neural network along with various activation functions and optimisation solvers has been presented. Slot waveguide design has been used to initially optimise various parameters of the machine learning models using three different datasets. The greater the number of datapoints in a dataset, the better is the accuracy of the model. Two different models (using MLPRegressor and PyTorch framework) have been shown and compared in terms of their accuracy. The MLPRegressor model was performing better. Hence, MLPRegressor has been used for strip waveguide and directional coupler designs, which shows approximately 1–4% of absolute percentage error between the actual values from the simulations and predictions from the machine learning model for unknown device dimensions. Furthermore, the work was expanded to use machine learning techniques for more complicated design of photonic crystal fibre (PCF), as discussed in Chapter 6. The effective index, effective mode area, dispersion, and confinement loss for a PCF from actual numerical simulations were compared with the predicted values obtained from machine learning models. Machine learning models

accurately predict outputs in a few milliseconds in comparison to few minutes needed for each numerical simulation, which again depends on the mesh density considered.

Metallic heaters have been used to compensate the phase mis-matching occurring due to the variations in waveguides width of a directional coupler during fabrication as presented in Chapter 7. These heaters can be aligned at different positions: top, top-left, and bottom-left. Temperature increases in the waveguide when heater is activated. Power confinements in the left-half and right-half sections were compared when there was fabrication error of 5 nm, and it was shown that approximately 26 mW of heater power was needed to obtain similar power in both sections. Moreover, time domain evolution of optical power along the directional coupler has also been studied. It has been shown that the heater at the bottom-left position requires less power in comparison to heater at the top or top-left position to compensate same fabrication error. The presence of metallic heaters also introduces an absorption loss. Also, the bottom-left heater has less absorption loss when compared with the heater at the top or top-left positions. This use of heater significantly increases output coupling power from 84% to 97% and from 50% to 94% for fabrication error of 5 nm and 10 nm, respectively.

8.2 Future Work

The machine learning methods presented in this thesis has been shown to complement finite element simulations to quickly predict the outputs for different device parameters faster than direct numerical simulation techniques. The generated machine learning code can be easily extended to more exotic directional coupler designs such as asymmetric, tapered, and slot waveguide based devices. The simulation of the propagation in a solid-core PCF design using a machine learning based algorithm has been presented in this work. Furthermore, a model can be created that can be used to predict outputs simultaneously for solid-core, hollow-core or more complicated PCF designs for different core materials. The range of parameters for prediction can be quickly extended by collecting more datapoints. This work can also be

extended to use transfer learning, which is a research problem in machine learning. Transfer learning uses pre-trained models from one task and re-purposed them on a different but related task. Transfer learning is an optimisation that allows rapid progress or improved performance when modelling the second task.

The use of two metallic heaters to compensate phase mis-matching in a silicon directional coupler for variations in width of waveguides shows promising results. This work can be extended to consider the variations in the height of the waveguides as well. Practical S-bends section has been considered briefly in this research, which can be studied for more thorough results. Adiabatic couplers that generally have different dimensions of the waveguides can be utilised to observe the impact of heater power when the heater is placed parallel to the guiding axis. Asymmetric directional couplers can be used for polarisation rotator or mode splitter. Mode splitter can be very sensitive to the waveguides width variations during fabrication. The initial results presented in this report will lead to the further exploration of using metallic heaters for mode splitter.

References

[1] G. T. Reed, "Device physics: the optical age of silicon," *Nature*, vol. 427, no. 6975, pp. 595–596, 2004.

[2] B. Jalali and S. Fathpour, "Silicon photonics," *Journal of Lightwave Technology*, vol. 24, no. 12, pp. 4600–4615, 2006.

[3] X. Chen, C. Li, and H. K. Tsang, "Device engineering for silicon photonics," *NPG Asia Materials*, vol. 3, no. 1, pp. 34–40, 2011.

[4] Z. Fang and C. Z. Zhao, "Recent progress in silicon photonics: a review," *ISRN Optics*, vol. 2012, 2012.

[5] N. Borhani, E. Kakkava, C. Moser, and D. Psaltis, "Learning to see through multimode fibers," *Optica*, vol. 5, no. 8, pp. 960–966, 2018.

[6] M. H. Tahersima, K. Kojima, T. Koike-Akino, D. Jha, B. Wang, C. Lin, and K. Parsons, "Deep neural network inverse design of integrated photonic power splitters," *Scientific Reports*, vol. 9, no. 1, p. 1368, 2019.

[7] J. Baxter, A. C. Lesina, J.-M. Guay, A. Weck, P. Berini, and L. Ramunno, "Plasmonic colours predicted by deep learning," *Scientific Reports*, vol. 9, no. 1, p. 8074, 2019.

[8] D. Gostimirovic and W. N. Ye, "Automating photonic design with machine learning," in *2018 IEEE 15th International Conference on Group IV Photonics (GFP)*. IEEE, 2018, pp. 1–2.

[9] A. da Silva Ferreira, G. N. Malheiros-Silveira, and H. E. Hernández-Figueroa, "Computing optical properties of photonic crystals by using multilayer perceptron and extreme learning machine," *Journal of Lightwave Technology*, vol. 36, no. 18, pp. 4066–4073, 2018.

[10] S. M. Raiyan Kabir, B. M. A. Rahman, A. Agrawal, and K. T. V. Grattan, "Elimination of numerical dispersion from electromagnetic time domain analysis by using resource efficient finite element technique," *Progress In Electromagnetics Research*, vol. 137, pp. 487–512, 2013.

[11] C. R. Pollock, *Fundamentals of Optoelectronics*. Irwin, 1995.

[12] G. R. Werner and J. R. Cary, "A stable FDTD algorithm for non-diagonal, anisotropic dielectrics," *Journal of Computational Physics*, vol. 226, no. 1, pp. 1085–1101, 2007.

[13] G. R. Werner, C. A. Bauer, and J. R. Cary, "A more accurate, stable, FDTD algorithm for electromagnetics in anisotropic dielectrics," *Journal of Computational Physics*, vol. 255, pp. 436–455, 2013.

[14] M. Clemens and T. Weiland, "Discrete electromagnetism with the finite integration technique," *Progress In Electromagnetics Research*, vol. 32, pp. 65–87, 2001.

[15] B. M. A. Rahman and A. Agrawal, *Finite element modeling methods for photonics*. Artech house, 2013.

[16] E. A. J. Marcatili, "Dielectric rectangular waveguide and directional coupler for integrated optics," *Bell System Technical Journal*, vol. 48, no. 7, pp. 2071–2102, 1969.

[17] R. M. Knox and P. P. Toulios, "Integrated circuits for the millimeter through optical frequency range," in *Proc. Symp. Submillimeter Waves*, vol. 20. Brooklyn, NY, 1970, pp. 497–515.

[18] K. S. Chiang, "Analysis of optical fibers by the effective-index method," *Applied Optics*, vol. 25, no. 3, pp. 348–354, 1986.

[19] K. S. Chiang, K. M. Lo, and K. S. Kwok, "Effective-index method with built-in perturbation correction for integrated optical waveguides," *Journal of Lightwave Technology*, vol. 14, no. 2, pp. 223–228, 1996.

[20] W. J. Westerveld, S. M. Leinders, K. W. A. Van Dongen, H. P. Urbach, and M. Yousefi, "Extension of Marcatili's analytical approach for rectangular silicon optical waveguides," *Journal of Lightwave Technology*, vol. 30, no. 14, pp. 2388–2401, 2012.

[21] B. M. A. Rahman and J. B. Davies, "Finite-element solution of integrated optical waveguides," *Journal of Lightwave Technology*, vol. 2, no. 5, pp. 682–688, 1984.

[22] A. Berk, "Variational principles for electromagnetic resonators and waveguides," *IRE Transactions on Antennas and Propagation*, vol. 4, no. 2, pp. 104–111, 1956.

[23] M. D. Feit and J. A. Fleck, "Computation of mode properties in optical fiber waveguides by a propagating beam method," *Applied Optics*, vol. 19, no. 7, pp. 1154–1164, 1980.

[24] Y. Chung, N. Dagli, and L. Thylen, "Explicit finite difference vectorial beam propagation method," *Electronics Letters*, vol. 27, no. 23, pp. 2119–2121, 1991.

[25] W. P. Huang, C. L. Xu, and S. K. Chaudhuri, "A finite-difference vector beam propagation method for three-dimensional waveguide structures," *IEEE Photonics Technology Letters*, vol. 4, no. 2, pp. 148–151, 1992.

[26] W. P. Huang and C. L. Xu, "A wide-angle vector beam propagation method," *IEEE Photonics Technology Letters*, vol. 4, no. 10, pp. 1118–1120, 1992.

[27] Y. Tsuji and M. Koshiba, "A finite element beam propagation method for strongly guiding and longitudinally varying optical waveguides," *Journal of Lightwave Technology*, vol. 14, no. 2, pp. 217–222, 1996.

[28] S. S. A. Obayya, B. M. A. Rahman, and H. A. El-Mikati, "New full-vectorial numerically efficient propagation algorithm based on the finite element method," *Journal of Lightwave Technology*, vol. 18, no. 3, pp. 409–415, 2000.

[29] J.-P. Berenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, vol. 114, no. 2, pp. 185–200, 1994.

[30] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[31] A. Taflove, "Application of the finite-difference time-domain method to sinusoidal steady-state electromagnetic-penetration problems," *IEEE Transactions on Electromagnetic Compatibility*, no. 3, pp. 191–202, 1980.

[32] X. Zhang, J. Fang, K. K. Mei, and Y. Liu, "Calculations of the dispersive characteristics of microstrips by the time-domain finite difference method," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 2, pp. 263–267, 1988.

[33] A. Taflove and S. C. Hagness, *Computational electrodynamics: the finite-difference time-domain method*. Artech house, 2005.

[34] D. Gallagher and P. Design, "Photonic CAD matures," *IEEE LEOS NewsLetter*, pp. 8–14, 2008.

[35] E. S. Um, *Three-dimensional finite-element time-domain modeling of the marine controlled-source electromagnetic method*. Ph.D. Thesis Stanford University, 2011.

[36] J.-F. Lee, R. Lee, and A. Cangellaris, "Time-domain finite-element methods," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 3, pp. 430–442, 1997.

[37] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, and J. Zhu, *The finite element method*. McGraw-hill London, 1977, vol. 3.

[38] B. M. A. Rahman, F. A. Fernandez, and J. B. Davies, "Review of finite element methods for microwave and optical waveguides," *Proceedings of the IEEE*, vol. 79, no. 10, pp. 1442–1448, 1991.

[39] B. M. A. Rahman, "Finite element analysis of optical waveguides," *Progress In Electromagnetics Research*, vol. 10, pp. 187–216, 1995.

[40] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[41] J. Schoberl, "Netgen an advancing front 2d/3d-mesh generator based on abstract rules," *Computing and Visualization in Science*, vol. 1, no. 1, pp. 41–52, 1997.

[42] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Transactions on Mathematical Software (TOMS)*, vol. 41, no. 2, pp. 11:1–11:36, 2015.

[43] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool." in *Eurographics Italian Chapter Conference*, vol. 2008, 2008, pp. 129–136.

[44] S. G. Johnson, "Notes on perfectly matched layers (PMLs)," *Lecture notes, Massachusetts Institute of Technology, Massachusetts*, vol. 29, 2008.

[45] A. Agrawal and A. Sharma, "Perfectly matched layer in numerical wave propagation: factors that affect its performance," *Applied Optics*, vol. 43, no. 21, pp. 4225–4231, 2004.

[46] C. Vassallo and F. Collino, "Highly efficient absorbing boundary conditions for the beam propagation method," *Journal of Lightwave Technology*, vol. 14, no. 6, pp. 1570–1577, 1996.

[47] W. P. Huang, C. L. Xu, W. Lui, and K. Yokoyama, "The perfectly matched layer (PML) boundary condition for the beam propagation method," *IEEE Photonics Technology Letters*, vol. 8, no. 5, pp. 649–651, 1996.

[48] D. Zhou, W. P. Huang, C. L. Xu, D. G. Fang, and B. Chen, "The perfectly matched layer boundary condition for scalar finite-difference time-domain method," *IEEE Photonics Technology Letters*, vol. 13, no. 5, pp. 454–456, 2001.

[49] W. C. Chew and W. H. Weedon, "A 3d perfectly matched medium from modified Maxwell's equations with stretched coordinates," *Microwave and Optical Technology Letters*, vol. 7, no. 13, pp. 599–604, 1994.

[50] B. Chen, D. G. Fang, and B. H. Zhou, "Modified berenger PML absorbing boundary condition for FD-TD meshes," *IEEE Microwave and Guided Wave Letters*, vol. 5, no. 11, pp. 399–401, 1995.

[51] S. D. Gedney, "An anisotropic perfectly matched layer-absorbing medium for the truncation of FDTD lattices," *IEEE Transactions on Antennas and Propagation*, vol. 44, no. 12, pp. 1630–1639, 1996.

[52] J. M. Senior and M. Y. Jamro, *Optical fiber communications: principles and practice*. Pearson Education, 2009.

[53] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[54] F. N. Khan, Q. Fan, C. Lu, and A. P. T. Lau, "An optical communication's perspective on machine learning and its applications," *Journal of Lightwave Technology*, vol. 37, no. 2, pp. 493–516, 2019.

[55] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[56] D. Cote, "Using machine learning in communication networks," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D100–D109, 2018.

[57] W. Ma, F. Cheng, and Y. Liu, "Deep-learning-enabled on-demand design of chiral metamaterials," *ACS Nano*, vol. 12, no. 6, pp. 6326–6334, 2018.

[58] C. L. Chen, A. Mahjoubfar, L.-C. Tai, I. K. Blaby, A. Huang, K. R. Niazi, and B. Jalali, "Deep learning in label-free cell classification," *Scientific Reports*, vol. 6, p. 21471, 2016.

[59] A. Tittl, A. John-Herpin, A. Leitis, E. R. Arvelo, and H. Altug, "Metasurface-based molecular biosensing aided by artificial intelligence," *Angewandte Chemie International Edition*, 2019.

[60] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-end deep learning of optical fiber communications," *Journal of Lightwave Technology*, vol. 36, no. 20, pp. 4843–4855, 2018.

[61] T. A. Eriksson, H. Bülow, and A. Leven, "Applying neural networks in optical communication systems: possible pitfalls," *IEEE Photonics Technology Letters*, vol. 29, no. 23, pp. 2091–2094, 2017.

[62] F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore, "An overview on application of machine learning techniques in optical networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1383–1408, 2018.

[63] R. M. Morais and J. Pedro, "Machine learning models for estimating quality of transmission in DWDM networks," *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D84–D99, 2018.

[64] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

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

[66] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[67] F. Chollet, "Keras," https://keras.io, 2015.

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

[69] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[70] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 Connectionist Models Summer School*, vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[71] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[73] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[74] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[75] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[76] C. A. Barrios, "Optical slot-waveguide based biochemical sensors," *Sensors*, vol. 9, no. 6, pp. 4751–4765, 2009.

[77] F. Dell'Olio and V. M. N. Passaro, "Optical sensing by optimized silicon slot waveguides," *Optics Express*, vol. 15, no. 8, pp. 4977–4993, 2007.

[78] W. Jiang, N. Kohli, X. Sun, and B. M. A. Rahman, "Multi-poly-silicon-layer-based spot-size converter for efficient coupling between silicon waveguide and standard single-mode fiber," *IEEE Photonics Journal*, vol. 8, no. 3, pp. 1–12, 2016.

[79] C. Pan and B. M. A. Rahman, "Accurate analysis of the mode (de) multiplexer using asymmetric directional coupler," *Journal of Lightwave Technology*, vol. 34, no. 9, pp. 2288–2296, 2016.

[80] A. Barh, B. M. A. Rahman, R. K. Varshney, and B. P. Pal, "Design and performance study of a compact SOI polarization rotator at 1.55 μm," *Journal of Lightwave Technology*, vol. 31, no. 23, pp. 3687–3693, 2013.

[81] W. Jiang, X. Sun, and B. M. A. Rahman, "Compact and fabrication-tolerant polarization splitter based on horizontal triple-slot waveguide," *Applied Optics*, vol. 56, no. 8, pp. 2119–2126, 2017.

[82] J. C. Knight, T. A. Birks, P. St. J. Russell, and D. M. Atkin, "All-silica single-mode optical fiber with photonic crystal cladding," *Optics Letters*, vol. 21, no. 19, pp. 1547–1549, 1996.

[83] P. St. J. Russell, "Photonic crystal fibers: Basics and applications," in *Optical Fiber Telecommunications VA*. Elsevier, 2008, pp. 485–522.

[84] W. J. Wadsworth, A. Ortigosa-Blanch, J. C. Knight, T. A. Birks, T.-P. M. Man, and P. St. J. Russell, "Supercontinuum generation in photonic crystal fibers and optical fiber tapers: a novel light source," *JOSA B*, vol. 19, no. 9, pp. 2148–2155, 2002.

[85] M. R. Karim, H. Ahmad, and B. M. A. Rahman, "All-normal dispersion chalcogenide PCF for ultraflat mid-infrared supercontinuum generation," *IEEE Photonics Technology Letters*, vol. 29, no. 21, pp. 1792–1795, 2017.

[86] F. Benabid, J. C. Knight, G. Antonopoulos, and P. St. J. Russell, "Stimulated Raman scattering in hydrogen-filled hollow-core photonic crystal fiber," *Science*, vol. 298, no. 5592, pp. 399–402, 2002.

[87] F. Benabid, G. Bouwmans, J. C. Knight, P. St. J. Russell, and F. Couny, "Ultrahigh efficiency laser wavelength conversion in a gas-filled hollow core photonic crystal fiber by pure stimulated rotational Raman scattering in molecular hydrogen," *Physical Review Letters*, vol. 93, no. 12, pp. 123 903–(1–4), 2004.

[88] P. K. Cheo, A. Liu, and G. G. King, "A high-brightness laser beam from a phase-locked multicore Yb-doped fiber laser array," *IEEE Photonics Technology Letters*, vol. 13, no. 5, pp. 439–441, 2001.

[89] M. De, T. K. Gangopadhyay, and V. K. Singh, "Prospects of photonic crystal fiber as physical sensor: An overview," *Sensors*, vol. 19, no. 3, p. 464, 2019.

[90] T. M. Monro, W. Belardi, K. Furusawa, J. C. Baggett, N. G. R. Broderick, and D. J. Richardson, "Sensing with microstructured optical fibres," *Measurement Science and Technology*, vol. 12, no. 7, p. 854, 2001.

[91] R. Holzwarth, Th. Udem, T. W. Hänsch, J. C. Knight, W. J. Wadsworth, and P. St. J. Russell, "Optical frequency synthesizer for precision spectroscopy," *Physical Review Letters*, vol. 85, no. 11, p. 2264, 2000.

[92] C.-P. Yu and H.-C. Chang, "Applications of the finite difference mode solution method to photonic crystal structures," *Optical and Quantum Electronics*, vol. 36, no. 1-3, pp. 145–163, 2004.

[93] A. Cucinotta, S. Selleri, L. Vincetti, and M. Zoboli, "Holey fiber analysis through the finite-element method," *IEEE Photonics Technology Letters*, vol. 14, no. 11, pp. 1530–1532, 2002.

[94] S. G. Johnson and J. D. Joannopoulos, "Block-iterative frequency-domain methods for Maxwell's equations in a planewave basis," *Optics Express*, vol. 8, no. 3, pp. 173–190, 2001.

[95] S. Shi, C. Chen, and D. W. Prather, "Plane-wave expansion method for calculating band structure of photonic crystal slabs with perfectly matched layers," *JOSA A*, vol. 21, no. 9, pp. 1769–1775, 2004.

[96] R. A. Norton and R. Scheichl, "Planewave expansion methods for photonic crystal fibres," *Applied Numerical Mathematics*, vol. 63, pp. 88–104, 2013.

[97] T. Asano and S. Noda, "Optimization of photonic crystal nanocavities based on deep learning," *Optics Express*, vol. 26, no. 25, pp. 32 704–32 717, 2018.

[98] S. S. A. Obayya, B. M. A. Rahman, and K. T. V. Grattan, "Accurate finite element modal solution of photonic crystal fibres," *IEE Proceedings-Optoelectronics*, vol. 152, no. 5, pp. 241–246, 2005.

[99] R. Soref, "The past, present, and future of silicon photonics," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 12, no. 6, pp. 1678–1687, 2006.

[100] M. Lipson, "Guiding, modulating, and emitting light on silicon-challenges and opportunities," *Journal of Lightwave Technology*, vol. 23, no. 12, pp. 4222–4238, 2005.

[101] X. Chen, M. M. Milosevic, S. Stankovic, S. Reynolds, T. D. Bucio, K. Li, D. J. Thomson, F. Gardes, and G. T. Reed, "The emergence of silicon photonics as a flexible technology platform," *Proceedings of the IEEE*, vol. 106, no. 12, pp. 2101–2116, 2018.

[102] C. Li, L. Zhou, and A. W. Poon, "Silicon microring carrier-injection-based modulators/switches with tunable extinction ratios and or-logic switching by using waveguide cross-coupling," *Optics Express*, vol. 15, no. 8, pp. 5069–5076, 2007.

[103] J. Takayesu, M. Hochberg, T. Baehr-Jones, E. Chan, G. Wang, P. Sullivan, Y. Liao, J. Davies, L. Dalton, A. Scherer, and W. Krug, "A hybrid electrooptic microring resonator-based $1 \times 4 \times 1$ ROADM for wafer scale optical interconnects," *Journal of Lightwave Technology*, vol. 27, no. 4, pp. 440–448, 2009.

[104] X. Wang, J. A. Martinez, M. S. Nawrocka, and R. R. Panepucci, "Compact thermally tunable silicon wavelength switch: modeling and characterization," *IEEE Photonics Technology Letters*, vol. 20, no. 11, pp. 936–938, 2008.

[105] S. Dwivedi, H. D'heer, and W. Bogaerts, "Maximizing fabrication and thermal tolerances of all-silicon FIR wavelength filters," *IEEE Photonics Technology Letters*, vol. 27, no. 8, pp. 871–874, 2015.

[106] X. Han and Y. Yu, "Optimization of a thermally tuned silicon-based reconfigurable optical power splitter with thermal isolations," *Optical Engineering*, vol. 56, no. 1, p. 017106, 2017.

[107] J. Schrauwen, D. Van Thourhout, and R. Baets, "Trimming of silicon ring resonator by electron beam induced compaction and strain," *Optics Express*, vol. 16, no. 6, pp. 3738–3743, 2008.

[108] A. H. Atabaki, E. S. Hosseini, A. A. Eftekhar, S. Yegnanarayanan, and A. Adibi, "Optimization of metallic microheaters for high-speed reconfigurable silicon photonics," *Optics Express*, vol. 18, no. 17, pp. 18 312–18 323, 2010.

[109] G. Cocorullo and I. Rendina, "Thermo-optical modulation at 1.5 μm in silicon etalon," *Electronics Letters*, vol. 28, no. 1, pp. 83–85, 1992.

[110] R. K. Gupta and B. K. Das, "Performance analysis of metal-microheater integrated silicon waveguide phase-shifters," *OSA Continuum*, vol. 1, no. 2, pp. 703–714, 2018.

[111] J. V. Campenhout, W. M. J. Green, S. Assefa, and Y. A. Vlasov, "Integrated NiSi waveguide heaters for CMOS-compatible silicon thermo-optic devices," *Optics Letters*, vol. 35, no. 7, pp. 1013–1015, 2010.

[112] P. Orlandi, F. Morichetti, M. J. Strain, M. Sorel, A. Melloni, and P. Bassi, "Tunable silicon photonics directional coupler driven by a transverse temperature gradient," *Optics Letters*, vol. 38, no. 6, pp. 863–865, 2013.

[113] A. Masood, M. Pantouvaki, G. Lepage, P. Verheyen, J. V. Campenhout, P. Absil, D. V. Thourhout, and W. Bogaerts, "Comparison of heater architectures for thermal control of silicon photonic circuits," in *10th International Conference on Group IV Photonics*. IEEE, 2013, pp. 83–84.

[114] S. Kaushal and B. K. Das, "Modeling and experimental investigation of an integrated optical microheater in silicon-on-insulator," *Applied Optics*, vol. 55, no. 11, pp. 2837–2842, 2016.

[115] M. Bahadori, A. Gazman, N. Janosik, S. Rumley, Z. Zhu, R. Polster, Q. Cheng, and K. Bergman, "Thermal rectification of integrated microheaters for microring resonators in silicon photonics platform," *Journal of Lightwave Technology*, vol. 36, no. 3, pp. 773–788, 2018.

[116] G. R. Bhatt and B. K. Das, "Improvement of polarization extinction in silicon waveguide devices," *Optics Communications*, vol. 285, no. 8, pp. 2067–2070, 2012.

[117] C. Ye, K. Liu, R. A. Soref, and V. J. Sorger, "A compact plasmonic MOS-based 2×2 electro-optic switch," *Nanophotonics*, vol. 4, no. 3, pp. 261–268, 2015.

[118] W. Jiang and X. Sun, "TE-pass polarizer with cylinder array inserted in buffer layer of silica-on-silicon waveguide," *IEEE Photonics Technology Letters*, vol. 26, no. 9, pp. 937–940, 2014.

[119] T. Horikawa, D. Shimura, S.-H. Jeong, M. Tokushima, K. Kinoshita, and T. Mogami, "The impacts of fabrication error in Si wire-waveguides on spectral variation of coupled resonator optical waveguides," *Microelectronic Engineering*, vol. 156, pp. 46–49, 2016.

[120] Y. Hida, H. Onose, and S. Imamura, "Polymer waveguide thermooptic switch with low electric power consumption at 1.3 μm," *IEEE Photonics Technology Letters*, vol. 5, no. 7, pp. 782–784, 1993.

[121] S. Ghosh and B. M. A. Rahman, "Design of on-chip hybrid plasmonic Mach-Zehnder interferometer for temperature and concentration detection of chemical solution," *Sensors and Actuators B: Chemical*, vol. 279, pp. 490–502, 2019.

[122] B. M. A. Rahman, T. Wongcharoen, and K. T. V. Grattan, "Finite element analysis of nonsynchronous directional couplers," *Fiber & Integrated Optics*, vol. 13, no. 3, pp. 331–336, 1994.

Appendix A

PML Equations

A.1 X Axis PML

The affected equations would be Eqs. 3.7, 3.8 3.10 3.11. If $\widetilde{\nabla}^x$ is used instead of ∇, then Eqs. 3.7 and 3.8 can be replaced by the following equations (TE propagation):

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} - \sigma_x H_y \tag{A.1}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \Psi_{x[x]}\right) - \sigma_x E_z \tag{A.2}$$

$$\frac{\partial \Psi_{x[x]}}{\partial t} = \sigma_x \frac{\partial H_x}{\partial y} \tag{A.3}$$

where $\Psi_{x[x]}$ is the auxiliary field generated due to X PML.

For TM propagation, Eqs. 3.10 and 3.11 can be replaced by the following:

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} - \sigma_x E_y \tag{A.4}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} - \Phi_{x[x]}\right) - \sigma_x H_z \tag{A.5}$$

$$\frac{\partial \Phi_{x[x]}}{\partial t} = \sigma_x \frac{\partial E_x}{\partial y} \tag{A.6}$$

where $\Phi_{x[x]}$ is the auxiliary field generated due to X PML.

A.2 Y Axis PML

As a result, replaced Eqs. 3.6 and 3.8 for TE propagation are as follows:

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} - \sigma_y H_x \tag{A.7}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} + \Psi_{y[y]}\right) - \sigma_y E_z \tag{A.8}$$

$$\frac{\partial \Psi_{y[y]}}{\partial t} = \sigma_y \frac{\partial H_y}{\partial x} \tag{A.9}$$

where $\Psi_{y[y]}$ is the auxiliary field generated due to Y PML.

Similarly, replaced Eqs. 3.9 and 3.11 for TM propagation are as follows:

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} - \sigma_y E_x \tag{A.10}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + \Phi_{y[y]}\right) - \sigma_y H_z \tag{A.11}$$

$$\frac{\partial \Phi_{y[y]}}{\partial t} = \sigma_y \frac{\partial E_y}{\partial x} \tag{A.12}$$

where $\Phi_{y[y]}$ is the auxiliary field generated due to Y PML.

A.3 Corner PML

Replaced Eqs. 3.6, 3.7 and 3.8 for TE propagation are as follows:

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} - \sigma_y H_x \tag{A.13}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} - \sigma_x H_y \tag{A.14}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} + \Psi_{y[y]} - \Psi_{x[x]}\right) - \sigma_y E_z - \sigma_x E_z - \Theta_{z[xy]} \tag{A.15}$$

$$\frac{\partial \Psi_{x[x]}}{\partial t} = \sigma_x \frac{\partial H_x}{\partial y} \tag{A.16}$$

$$\frac{\partial \Psi_{y[y]}}{\partial t} = \sigma_y \frac{\partial H_y}{\partial x} \tag{A.17}$$

$$\frac{\partial \Theta_{z[xy]}}{\partial t} = \sigma_x \sigma_y E_z \tag{A.18}$$

where $\Psi_{x[x]}$, $\Psi_{y[y]}$ and $\Theta_{z[xy]}$ are the auxiliary fields generated due to corner PML.

Replaced Eqs. 3.9, 3.10 and 3.11 for TM propagation are as follows:

$$\frac{\partial E_x}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_z}{\partial y} - \sigma_y E_x \tag{A.19}$$

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\varepsilon}\frac{\partial H_z}{\partial x} - \sigma_x E_y \tag{A.20}$$

$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + \Phi_{y[y]} - \Phi_{x[x]}\right) - \sigma_x H_z - \sigma_y H_z - \Omega_{z[xy]} \tag{A.21}$$

$$\frac{\partial \Phi_{x[x]}}{\partial t} = \sigma_x \frac{\partial E_x}{\partial y} \tag{A.22}$$

$$\frac{\partial \Phi_{y[y]}}{\partial t} = \sigma_y \frac{\partial E_y}{\partial x} \tag{A.23}$$

$$\frac{\partial \Omega_{z[xy]}}{\partial t} = \sigma_x \sigma_y H_z \tag{A.24}$$

where $\Phi_{x[x]}$, $\Phi_{y[y]}$ and $\Omega_{z[xy]}$ are the auxiliary fields generated due to corner PML.

Appendix B

Machine Learning Code

```
import time
start_time = time.time()
# print('start_time: ', start_time)

import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import sys
import pickle
import torch
from torch import nn, optim
from torchvision import transforms
from collections import OrderedDict
```

```
# Use GPU if it's available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
scaler1 = MinMaxScaler()
scaler2 = MinMaxScaler()
no_of_output_nodes = 5
df_1 = pd.read_excel('pcf_modeSoln_data_2.xlsx',
                                sheetname='SiO2-air-rings-4-dBYp-0.6')
datafile_1 = df_1.values                ## stored data from xlsx file
##########    taking data from other sheets    #############
sheets_names = ['SiO2-air-rings-4-dBYp-0.8',
            'SiO2-air-rings-4-dBYp-0.9', 'SiO2-air-rings-4-dBYp-0.7',
            'SiO2-air-rings-5-dBYp-0.6', 'SiO2-air-rings-5-dBYp-0.8',
            'SiO2-air-rings-5-dBYp-0.9', 'SiO2-air-rings-5-dBYp-0.7']
# sheets_names = []
for sheet_name in sheets_names:
print(sheet_name)
df_sheet_name = pd.read_excel('pcf_modeSoln_data_2.xlsx',
                                                sheetname=sheet_name)
datafile_sheet_name = df_sheet_name.values  ## stored data from xlsx
######    combining data from all sheets of excel file    #####
datafile_1 = np.concatenate((datafile_1, datafile_sheet_name),axis=0)

########   just to see output variable values   ##########
out_var_datafile_1 = datafile_1[:,range(6,11)]
out_var_datafile_1 = out_var_datafile_1.reshape(
     (-1,no_of_output_nodes))  ## one column with unknown no. of rows
print(out_var_datafile_1)
print('no. of training points: ', len(out_var_datafile_1))

scaler1.fit(datafile_1)
```

```
scaler2.fit(out_var_datafile_1)
scaler_datafile_1 = scaler1.transform(datafile_1)
X = scaler_datafile_1[:,range(0,6)]         ## input variables columns
y = scaler_datafile_1[:,range(6,11)]       ## output variables columns
print(X); print(); print(y)

X, y = shuffle(X, y)
X_train, X_validation, y_train, y_validation = train_test_split(
                                                X, y, test_size = 0.1)
X_train = X_train.reshape(-1, 6)
y_train = y_train.reshape(-1, no_of_output_nodes)
X_validation = X_validation.reshape(-1, 6)
y_validation = y_validation.reshape(-1, no_of_output_nodes)
print('no. of training points: ', len(X_train))
print('no. of validation points: ', len(X_validation))

############      manual testing     #########
df_2 = pd.read_excel('pcf_modeSoln_data_manual_2.xlsx',
                                                  sheetname='Sheet1')
datafile_2 = df_2.values                  ## stored data from xlsx file
print(datafile_2)
scaler_datafile_2 = scaler1.transform(datafile_2)
X_test = scaler_datafile_2[:,range(0,6)]    ## input variables columns
y_test = scaler_datafile_2[:,range(6,11)] ## output variables columns
print(X_test); print(); print(y_test)
print('no. of test points: ', len(X_test))
X_test = X_test.reshape(-1, 6)
y_test = y_test.reshape(-1, no_of_output_nodes)
#########################################

input_dim = 6                        ## = no. of input variables columns
output_dim = no_of_output_nodes   ## = no. of output variables columns
```

```
####     model with dropout - 3 layers    ########
dropout_prob = 0.0
nodes_hidden_1 = 50
nodes_hidden_2 = 50
nodes_hidden_3 = 50
## nn.Linear() is a fully connected layer
model = nn.Sequential(OrderedDict([
                  ('fc1', nn.Linear(input_dim, nodes_hidden_1)),
                  ('relu', nn.ReLU()),
                  ('dropout', nn.Dropout(dropout_prob)),
                  ('fc2', nn.Linear(nodes_hidden_1, nodes_hidden_2)),
                  ('relu', nn.ReLU()),
                  ('dropout', nn.Dropout(dropout_prob)),
                  ('fc3', nn.Linear(nodes_hidden_2, nodes_hidden_3)),
                  ('relu', nn.ReLU()),
                  ('dropout', nn.Dropout(dropout_prob)),
                  ('fc4', nn.Linear(nodes_hidden_3, output_dim)), ]))
print(model)

###########   parameters   ###############
criterion = nn.MSELoss()
learning_rate = 0.0001
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

epochs = 5000
# Convert numpy array to torch Variable
# inputs = torch.from_numpy(X_train).requires_grad_()
# labels = torch.from_numpy(y_train)
inputs = torch.Tensor((X_train))
labels = torch.Tensor((y_train))
inputs_validation = torch.Tensor((X_validation))
```

```
labels_validation = torch.Tensor((y_validation))
running_loss = []; running_loss_validation = []
for epoch in range(epochs):
    epoch += 1

    ########   train the model   #########
    model.train()     # prep model for training
    # Clear gradients w.r.t. parameters, else gradients will be added
    # up with every previous pass
    optimizer.zero_grad()
    # Forward to get output
    outputs = model(inputs)
    # Calculate Loss
    loss = criterion(outputs, labels)           ## mean squared error
    # Getting gradients w.r.t. parameters
    loss.backward()
    # Updating parameters
    optimizer.step()  ## take a step with optimizer to update weights
    running_loss.append(loss.item())

    ###  validate the model (showing fluctuations) ####
    outputs_validation = model(inputs_validation)
    loss_validation = criterion(outputs_validation,labels_validation)
    running_loss_validation.append(loss_validation.item())
    print('epoch: {}, mse_loss: {:.6f}, mse_loss_validation: {:.6f}'
                .format(epoch, loss.item(), loss_validation.item()))

#### save the model weights & parameters
torch.save(model.state_dict(), 'checkpoint.pth')
#### load the saved model
state_dict = torch.load('checkpoint.pth')
model.load_state_dict(state_dict)
```

```
# Purely inference
model.eval()
predicted_on_X_train = model(torch.Tensor(X_train)).data.numpy()
predicted_on_X_validation = model(torch.Tensor(X_validation))
                                                        .data.numpy()
predicted_on_X_test = model(torch.Tensor(X_test)).data.numpy()

end_time = time.time()
print('end_time: ', end_time)
print('time taken to train in sec: ', (end_time - start_time))

#### make axis bold for plotting
plt.rcParams.update({'font.size': 10})
plt.rcParams["font.weight"] = "bold"
plt.rcParams["axes.labelweight"] = "bold"

mse_training_interval = 10
mse_validation_interval = 10
running_loss = running_loss[::mse_training_interval]
running_loss_index = [i for i in range(1, epochs,
                                       mse_training_interval)]
running_loss_validation = running_loss_validation
                                          [::mse_validation_interval]
running_loss_validation_index = [i for i in range(1, epochs,
                                         mse_validation_interval)]
print('mse lengths: ',len(running_loss),len(running_loss_validation))
print('running_loss_index: ', running_loss_index)
print('running_loss_validation_index: ',
                                      running_loss_validation_index)
```

```
#############################################################
############   plotting graphs together - neff  ##########
#############################################################
plt.figure()
#### giving title on top of all subplots
plt.suptitle('pcf - neff - (epochs-{}) - pyTorch'.format(epochs),
                            fontsize=25, color='r', fontweight='bold')

plt.subplot(231)
plt.plot(running_loss_index, running_loss, 'r-', linewidth=3,
                                                label='mse_loss_train')
plt.plot(running_loss_validation_index, running_loss_validation, 'b-'
                            ,linewidth=3, label='mse_loss_validation')
plt.legend(loc='best', fontsize=10)
plt.xlabel('epochs#', fontsize=15)

plt.subplot(232)
# Plot true data
plt.plot(scaler2.inverse_transform(y_train)[:,0], 'ro', markersize=12
                                                    ,label='y_train')
# Plot predictions
plt.plot(scaler2.inverse_transform(predicted_on_X_train)[:,0], 'b*',
                           markersize=12, label='predicted_on_X_train')
# Legend and plot
plt.legend(loc='best', fontsize=10)

plt.subplot(233)
# Plot true data
plt.plot(scaler2.inverse_transform(y_validation)[:,0], 'ro',
                                   markersize=12, label='y_validation')
# Plot predictions
plt.plot(scaler2.inverse_transform(predicted_on_X_validation)[:,0],
```

```
                'b*', markersize=12, label='predicted_on_X_validation')
# Legend and plot
plt.legend(loc='best', fontsize=10)


plt.subplot(234)
# Plot true data
plt.plot(scaler2.inverse_transform(y_test)[:,0], 'ro', markersize=12
                                                    ,label='y_test')
# Plot predictions
plt.plot(scaler2.inverse_transform(predicted_on_X_test)[:,0], 'b*',
                            markersize=12, label='predicted_on_X_test')
# Legend and plot
plt.legend(loc='best', fontsize=10)


plt.subplot(235)
xx = scaler2.inverse_transform(y_train)[:,0]
yy = scaler2.inverse_transform(predicted_on_X_train)[:,0]
xx_validation = scaler2.inverse_transform(y_validation)[:,0]
yy_validation = scaler2.inverse_transform(predicted_on_X_validation)
                                                                [:,0]
xx_test = scaler2.inverse_transform(y_test)[:,0]
yy_test = scaler2.inverse_transform(predicted_on_X_test)[:,0]
bubble_plot_line_x1y1=[min(np.minimum(xx,yy)),max(np.maximum(xx,yy))]
bubble_plot_line_x2y2=[min(np.minimum(xx,yy)),max(np.maximum(xx,yy))]
plt.xlim(bubble_plot_line_x1y1[0], bubble_plot_line_x1y1[1])
plt.ylim(bubble_plot_line_x1y1[0], bubble_plot_line_x1y1[1])
plt.plot(bubble_plot_line_x1y1, bubble_plot_line_x2y2, 'k-',
                                                    linewidth=2)
plt.grid(linestyle='--', linewidth=1)
plt.scatter(xx, yy, label='train', marker='o', facecolors='',
                                                edgecolors='red', s=50)
plt.scatter(xx_validation, yy_validation, label='validation',
```

```
                    marker='o', facecolors='', edgecolors='blue', s=50)
plt.scatter(xx_test, yy_test, label='test', marker='o', facecolors=''
                                            , edgecolors='black', s=50)
plt.legend(loc='best', fontsize=10)
plt.xlabel('true-values', fontsize=15)
plt.ylabel('predicted', fontsize=15)

plt.subplot(236)
true_values = scaler2.inverse_transform(y_test)[:,0]
predicted_values = scaler2.inverse_transform(predicted_on_X_test)
                                                                  [:,0]
x_index = [i for i in range(len(true_values))]
error_values = predicted_values - true_values
plt.errorbar(x=x_index, y=true_values, yerr=error_values, fmt='o',
             color='black', ecolor='black', elinewidth=2, capsize=10)
plt.grid(linestyle='--', linewidth=1)

print()
print("o/p of test set:\n", (scaler2.inverse_transform(y_test)[:,0]))
print("predicted o/p of test set: \n", (scaler2.inverse_transform(
                                        predicted_on_X_test)[:,0]))
print("mse_test_set: ", mean_squared_error(y_test,
                                           predicted_on_X_test))


plt.show()
```