University of Nottingham
UK | CHINA | MALAYSIA

EEE Department of Electrical and Electronic Engineering

H54IOP

Final Year Project Thesis

Photonic Reservoir Computing for High Bandwidth Signal Processing Applications

AUTHOR: Mr Calum Bradbury

ID NUMBER: 14280588

SUPERVISOR: Dr. Sendy Phang

MODERATOR: Professor John Crowe

DATE: 13 May 2020

Acknowledgements

Throughout the formulation of this thesis I have received a vast amount of high-quality support and advice and thus I would like to place on record my sincere gratitude to both my project supervisor, Dr Sendy Phang, and my moderator, Professor John Crowe. From both I have learned a great deal.

Firstly, Dr Sendy Phang's generous advice and support throughout the project was invaluable and I could not have asked for a better supervisor. Even though he maintains a very busy schedule, he always made himself available to me and never rejected any request for a meeting. Even if that meant our meetings occurring outside of term time or spontaneously if I dropped by his office whilst he was eating his breakfast! Furthermore, he was always patient with me and never cut short meetings even if they ran over significantly longer than scheduled (as often they did).

Secondly, Professor John Crowe always offered an objective evaluation of my current work in our periodic meetings. Pushing me to think more critically about my research and thus allowing me to produce higher quality work.

Finally, I would like to thank my parents for their unwavering support throughout my entire university career.

List of Symbols

Abbreviation	Explanation	Units of Measurement
H	Magnetic Field	Ampere per metre (A$\cdot m^{-1}$)
E	Electric Field	Volt per metre ($V \cdot m^{-1}$)
$\vec{E}$	Electric Field Vector	Volt per metre ($V \cdot m^{-1}$)
ρ	Free Electric Charge Density	Coulomb per cubic metre (1A$\cdot s \cdot m^{-3}$)
ε	Electric Permittivity of Dielectric	Farad per metre ($F \cdot m^{-1}$)
μ	Magnetic Permeability	Henries per metre ($H \cdot m^{-1}$)
J	Free Current Density	Ampere per square metre ($A \cdot m^{-2}$)
∇	Divergence Operator	-
α	Lattice Constant	micron (μm)
r	Relative Radius of media	micron (μm)
$\vec{k}$	Wavevector	Reciprocal metre (m^{-1})
$\in$	Set Membership Operator	-
$\mathbb{R}$	Set of all Real Numbers	-
σ	Leaking Constant	-
im	Modulated Input Signal	-
r	Reservoir Impulse Response	-
S	Normalised Reservoir Impulse Response	-
i	Gaussian Impulse Signal	-
om	Modulated Output Signal	-

List of Abbreviations

Abbreviation	Explanation
ANN	Artificial Neural Network
AI	Artificial Intelligence
RC	Reservoir Computer
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
XOR	Exclusive-or
LSTM	Long Short-Term Memory
PC	Photonic Crystal
RF	Radio Frequency
MWP	Microwave-Photonics
ISI	Inter Symbol Interference
FDTD	Finite-difference method in the time domain
EM	Electromagnetic
FFT	Fast Fourier Transform
PBG	Photonic Band Gap
TE	Transverse Electric
TM	Transverse Magnetic
Ge-on-Si	Germanium on Silicon
NMSE	Normalised Mean Square Error
ASK	Amplitude Shift Keying
NARMA10	10^{th} Order Nonlinear Auto Regressive Moving Average

Abstract

The constant pace of technological advancement within the telecommunications sector, spearheaded by the arrival of 5G wireless communications, means that demands for faster, more stable and higher bandwidth mobile communications networks are intensifying. As the limitations of conventional RF infrastructure starts to become problematic, optical systems that boast far greater bandwidth, lower losses and the potential for real-time processing may hold the key to the future of communications technology.

The potential shown by reservoir computing, a relatively novel computational framework derived from the recurrent neural network, is attracting increasing attention in many fields of research. This thesis explores the theoretical implementation of a reservoir computer in an integrated photonic environment as a novel communications framework for the post-processing of optical signals. A high frequency digital communications signal is modulated with light and its transmission through a dense urban environment is simulated. The reconstruction and post-processing of the resulting transmitted signal that has been mixed and subject to interference is attempted by a reservoir computer.

Keywords: Reservoir Computer, Microwave-Photonics, Photonic Crystal, Finite-Difference-Time-Domain, Optical Devices

Table of contents

1.0 Introduction

Inspired by, and designed to resemble, the biological neural network structure of the human brain and replicate the way it learns, Artificial Neural Networks (ANNs) have proven to be a hugely effective and versatile tool in identifying unknown patterns and relationships between correlated data. Whilst most artificial intelligence (AI) methods look to imitate intelligent behaviour by replicating "what we do", ANNs seek to imitate "the way that we do it" [1]. Typically consisting of a network of connected neurons arranged in several layers in order to mimic biological neurons and synapses. With each neuron containing a non-linear activation function that uses the sum of all weighted inputs to produce an output that is passed to the next layer [2]. A similar function is used in an output layer to provide the final output. The network learns by going through "training" using feedback gained from comparisons between the previous outputs and what the expected output should actually be. In this way over time the gap between the intended output and actual output eventually match and the neural network learns what the correct output for a given input should be [3]. Whilst there exist many different types of ANN models, a relatively novel concept called a reservoir computer (RC) has shown itself to be a strong and computationally efficient candidate for signal processing, particularly in an optical format [4]. Due to the way training takes place in a RC, the number of neurons with fixed connection weights between each other can be significantly increased compared to conventional ANNs. This simulation-based project looks to harness the inherent advantages of photonics and the potential shown by reservoir computing to implement an all-optical RC and explore its capabilities for high frequency signal processing. With a particular focus on industrial telecommunication applications.

1.1 Thesis Structure

This thesis consists of eight chapters. Chapter 1 comprises a broad introduction including the aims and objectives of the project, a review of the background and history up to the present day of the project area and a discussion of the industrial relevance and project motivation. Chapter 2 contains an overview of the proposed system and a review of the simulation methods employed throughout the project. Chapter 3 contains relevant theory to, and the design of, an optical waveguide. Chapter 4 covers the design of an optical reservoir for a reservoir computer. Chapter 5 covers the generation of training data for the RC and relevant theory and chapter 6 displays benchmarking results and relevant discussion. Finally, chapter 7 consists of a conclusion and chapter 8 contains personal reflections and directions for future research and improvements.

1.2 Aims, Objectives & Project Deliverables

The aim of this project is to confirm the hypothesis that microwave-photonics in conjunction with artificial neural network technology can be used to increase wireless communications-capacity for information and communications processing and thus offer a new approach to physical infrastructure and technology.

The objectives planned out to achieve this are as follows:

- Gain an understanding and appreciation of photonic technology and artificial neural network (ANN) techniques.
    - Research and gain a familiarity with ANN techniques for reservoir computing.
    - Research on microwave-photonic technologies.
    - Understand core principles of photonic crystals such as the implementation of intentional defects.

- Design of optical resonant cavity for reservoir computer.
    - Design a photonic crystal structure.

    - Implement intentional defects to create a waveguide for verification of correct function.
    - Design RC reservoir as a resonant cavity within a photonic crystal structure.

- Development of photonic neural network algorithm.
    - Develop a reservoir computing algorithm using high-performance photonic simulation software (Lumerical) and MATLAB.

- Validation and benchmarking of RC code.
    - Solve a header recognition problem using the reservoir computing algorithm in order to ascertain the quality of its performance and develop an understanding of proper algorithm optimisation.

The proposed deliverables for this project are given below:

1. Validated Photonic crystal wafer design.
2. Validated optical waveguide design.
3. Optical resonant cavity model with simulation results.
4. A working and validated MATLAB reservoir computing algorithm.
5. Finalised thesis containing a background literature review, system design principles, methodologies used, RC simulation algorithm and test results, analysis of results and considerations for future improvement.

The original and revised Gannt charts used for planning this project can be found in Appendix A and B respectively. Appendix A also contains the entire original project plan.

1.3 Background and current state of Artificial Neural Networks

Throughout history countless philosophers, theologians and mathematicians have mused over the function of the brain and ideas of human thought in artificial capacities naturally followed. In fact, human research into the brain can be dated as far back as the 5th century BCE with Hippocrates who, contrary to his contemporaries of the time, theorised that the brain is the seat of thought and the interpreter of consciousness [5]. Nearly 1700 years later, the German mathematician and philosopher Gottfried Leibniz would publish his book *Dissertatio de arte Combinatoria* (1666). In this extended version of his first doctoral dissertation he argued that all human thought and ideas are combinations of smaller simple concepts [6], a fundamental idea that is reflected in the individual neurons that build a neural network. The first drawings of neurons were presented by Santiago Ramón y Cajal in his paper *Texture of the Nervous System of Man and the Vertebrates* (1904). By staining slices of brain matter he was able to ascertain that every neuron in the brain is separate and communicates through passing information across a gap, the synapse. This concept would come to be known as the *neurone doctrine* and would take half a century, and the advent of the electron microscope, to be conclusively proved correct. Figure 1.3.1 displays R. Cajal's depiction of neurons in a brain [7]. Such ideas and concepts were significant milestones in mankind's fledgling understanding of the brain's anatomy and function, and with the advent of modern electronics it became possible for humans to realistically consider how to mimic the brain in an artificial format. Thus, allowing for the creation of a system that had the ability to learn and model complex non-linear relationships.

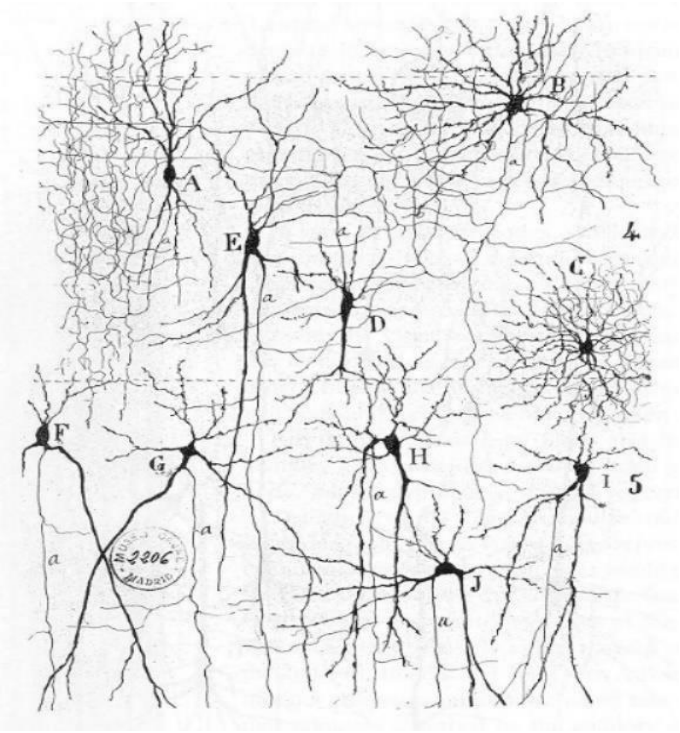

Figure 1.3.1: Ramón Cajal's drawing of neurons illustrating them as individual cells with synapses [7].

With the publication of Warren McCulloch's and Walter Pitts paper, *A Logical Calculus of The Ideas Immanent in Nervous Activity* (1943), significant steps towards the first artificial neural networks took place. Through combining mathematical logic with neurophysiology and using an electrical circuit they modelled each individual neuron in a network as a binary discrete-time element. The computations of their resulting neural networks were considered to be explanations of mental processes and human behaviour, thus connecting the study of biological neural networks to the idea of computation in a modern sense [8,9]. A few years later Donald Hebb published his book *The Organisation Of Behaviour: A Neuropsychological Theory* (1949) in which he ascertained that neural pathways are strengthened each time they are used [10], reinforcing knowledge of how neurons work and providing a fundamentally important idea of how the brain learns.

The concept of machine learning was later properly defined by Alan Turing in his book *Computing Machinery and Intelligence* (1950) in which he considers the question "Can machines think?" [11]. It was in this paper that Turing proposed his famous "Imitation game", otherwise known as the "Turing Test", that set the criterion for measuring of artificial intelligence as the ability of a machine to "fool a human into thinking it is also a human". A short time later in 1951 at Harvard University Marvin Minsky and Dean Edmonds built what would become to be known as the first artificial neural network, entitled a "Stochastic neural-analog reinforcement calculator" consisting of 40 interconnected artificial neurons made from electronic components [12]. Later on, Minsky would publish his book *Perceptrons* (1969) arguing that single neurons (named "perceptrons" in his book) were incapable of learning non-linear functions such as the exclusive-or (XOR) function [13], since they were only linear classifiers. Thus, he argued that no matter how complicated the system was, neural networks could only approximate linear functions. Whilst this argument was flawed and is in part blamed for a relative cooling of interest and freezing in funding and publications for AI and neural network research for about ten years [14], he made it clear that non-linearity must one day be introduced to neurons if the full potential of neural networks was to be realised. In modern times all ANNs take advantage of non-linearity including the reservoir computer studied here.

It was not until the early 1980's and several notable events in the field that interest in ANNs started to become renewed. Kunihiko Fukushima's paper, *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position* (1980), laid the basis for the first convolutional neural networks (CNNs). He proposed a "neocognitron" that through the use of convolutional operation was able to recognize stimulus patterns even when small amounts of distortion were introduced [15] excelling at image processing and pattern recognition. Up until this point the connections between neurons had only been one-way. This changed in 1982 when John Hopfield presented a paper to the National Academy of Sciences in which he argued that more powerful machines could be created using bidirectional lines [16]. The result was the publication of the first Recurrent Neural Networks (RNNs) that introduced feedback as a form of memory in order to capture information about the correlation between current and previous input data. This type of machine has since found success particularly in applications that involve working with sequential data such as word prediction and language processing. On the back of this success many new important concepts and algorithms started to appear. Including a learning procedure for ANNs, entitled "back propagation" [17] in 1986 that simplified neural network structure, the random forest algorithm [18] in 1995 that has since become one of the most widely used algorithms for classification and regression tasks and the Long Short-Term Memory (LSTM) RNN [19] in 1997 that has in particular found success in handwriting and speech recognition.

During this time rapidly improving computing speed and capacity alongside breakthroughs in research powered some notable successes for AI applications. In 1995 Richard Wallace developed his chatbot, named A.L.I.C.E (Artificial Linguistic Internet Computer Entity). Using natural language sample data collection enabled by the advent of the world wide web, A.L.I.C.E was the first of its kind and would go on to win the Loebner Prize in multiple years for being the most "human-like" computer program [20]. Then in 1997 IBM's chess playing computer program, named Deep Blue, became the first artificial system to defeat a reigning world chess champion. As time moved into the 21st century ANNs were further improved upon by newer concepts, including deep learning [21], that enabled their use for more advanced applications such as control of autonomous vehicles, financial forecasting and modelling of complex dynamic systems [22].

The progression and successes of ANN technology would go on to spur the developments of reservoir computing. Proposed independently by Herbert Jaeger, under the moniker "Echo State Networks" and Wolfgang Maass as "Liquid State Machines", the idea of reservoir computing was based on a new approach to RNNs that in particular provided solutions to existing RNN issues such as slow convergence, high computational costs and difficulty with training [23]. It was argued that as long as an RNN possesses certain generic properties, then supervised adaptation and training of all interconnected weights in the system is not required. The underlying theory being that a randomly constructed reservoir can transform an input signal in a complex non-linear fashion, allowing the readout layer to deduce the desired output using simple linear mapping [24]. Thus, by using an untrained RNN as a reservoir with nonlinear dynamics, only a supervised readout layer is required to be trained, see Figure 1.3.2(b). The output from which has been shown to be enough to obtain excellent performance in many tasks [25]. Because the input weights and the weights of the recurrent connections within the reservoir are not trained, the training process for a RC is far faster and requires much less computational power than standard RNNs. Furthermore, this feature of RC's offers a much greater degree of freedom in the design of the complicated kernel topology compared to other ANN's, a key characteristic that will be taken advantage of throughout this thesis. The difference in training methods between RNN's and RC's is illustrated in Figure 1.3.2.

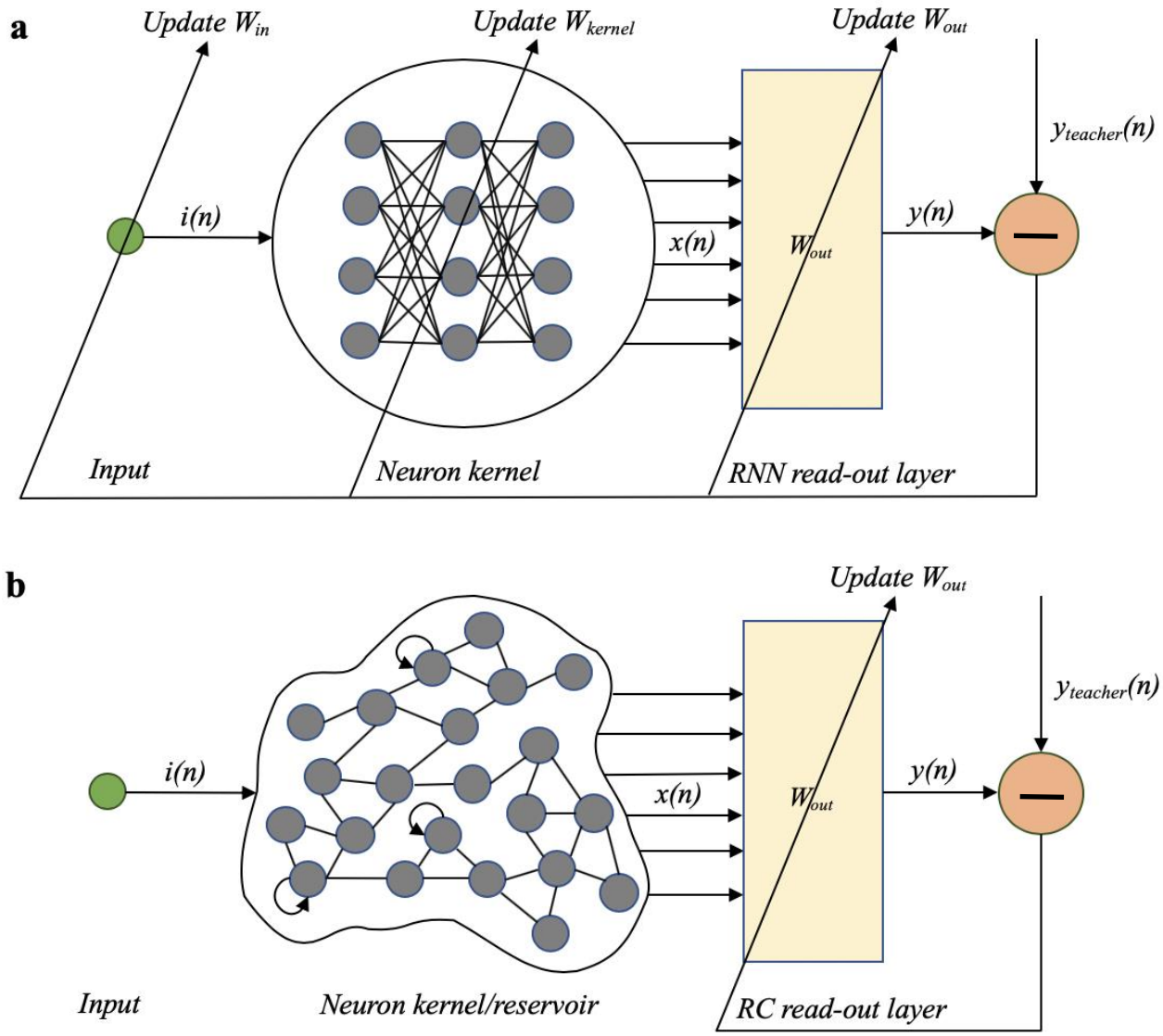

Figure 1.3.2: Illustration of training method for (a) Recurrent Neural Network and (b) Reservoir Computer.

As can be seen in Figure 1.3.2, the inspiration taken from RNN's in the design of the RC is clear. Both ANN approaches consist of three main sections, an input channel, neuron kernel and output channel. The RNN system is trained through finding appropriate weighting elements of the input, neuron kernel and read-out weights, however this method of training limits the freedom of the design of the kernel [26]. The RC system is fed with an input *i(n)* that is passed into a nonlinear recurrent reservoir with internal variables *x(n)*. The read-out layer W_{out} is the weighting of the output from the reservoir and *y(n)* is a linear combination of the internal variables. As can be seen, contrary to RNN's, the read-out layer is the only layer that is adjusted during the training phase. As discussed previously, this offers a much higher degree of freedom in the topology of the kernel, speeds up training times and decreases computational cost significantly [26]. In typical ANN's there may be many inputs, however for the purpose of simplicity due to time constraints, in this project only one input is used and thus this is reflected in Figure 1.3.2. See Section 5.0 for further discussion on the training of a reservoir computer.

RCs were initially utilized for simple tasks such as basic sine wave pattern generation and temporal XOR operation. As the technology advanced the applications of RCs became more complex and they have been proven to perform particularly well in applications such as pattern classification and prediction, and sequential information processing. Reservoir computing is in fact the most powerful approach known at present for some of these tasks and notably won a financial time series prediction competition in 2007 [27]. In 2018 the Institute of Electrical and Electronic Engineers (IEEE) established a group tasked with the purpose of promoting and stimulating the development of reservoir computing research under theoretical and application perspectives [28]. Table 1.3.1 [29] illustrates some of the most significant milestone applications in reservoir computing to date.

Table 1.3.1: Significant milestones in Reservoir Computing.

Application	Researcher(s)	Year
Chaotic time series prediction	H. Jaeger	2001
Sine-wave generation	H. Jaeger	2002
NARMA time series prediction	H. Jaeger	2003
Channel equalisation	H. Jaeger et al	2004
Temporal XOR tasks	N. Bertschinger et al.	2004
Temporal parity tasks	N. Bertschinger et al.	2005
Spoken digit recognition	D. Verstraeten et al.	2005
Waveform classification	Paquot et al.	2012
Human action recognition	H. Soh et al.	2012
Handwritten digit image recognition	H. Hauser et al.	2015

1.4 Artificial Neural Networks in an all-optical format

Within the photonics industry there are significant aspirations to replace electronic systems with photonic systems for the next generation of communication and computing systems. This is due to some of the major advantages offered, including signal processing at the speed of light and the massive parallelism inherent to optical structures. Advantages that it is argued make photonics the perfect choice for telecommunication applications in particular [30]. The first photonic implementation of a RC was proposed as a new approach to optical signal processing for large scale pattern recognition and signal processing problems, in a paper published by Kristof Vandoorne et al. entitled “Toward optical signal processing using Photonic Reservoir Computing” (2008) [31]. In this paper it was concluded through simulations that on a standard benchmark classification task, a photonic reservoir, by harnessing the inherent advantages of light, could outperform traditional reservoirs, offering the promise of massively parallel information processing at extremely high speed, high bandwidth and low power consumption.

Later, in 2014 Kristof Vandoorne et al. built an experimental prototype on a silicon chip using optical waveguides, splitters, and combiners with photodetectors to provide the non-linearity required. Demonstrations showed good performance across a range of tasks including 2-bit XOR logical operations and spoken digit classification [32]. Since then there have been several experimental implementations of optical RC’s tasked with complex tasks such as the detection of epileptic seizures [33], with most of them reporting error rates comparable to the best digital algorithms [34]. It is important to note however that the RC technique is still in its infancy and thus up to this point, training data has largely been confined to artificial data that only reflects ideal scenarios. Figure 1.4.1, from [35], illustrates a theoretical optical RC system proposed by Florian Denis-le Coarer et al in 2018, where each node in the photonic reservoir is a non-linear microring resonator connected by optical waveguides. Through simulations this system was shown to obtain state-of-the-art performance on a nonlinear Boolean task with very low power consumption [35].

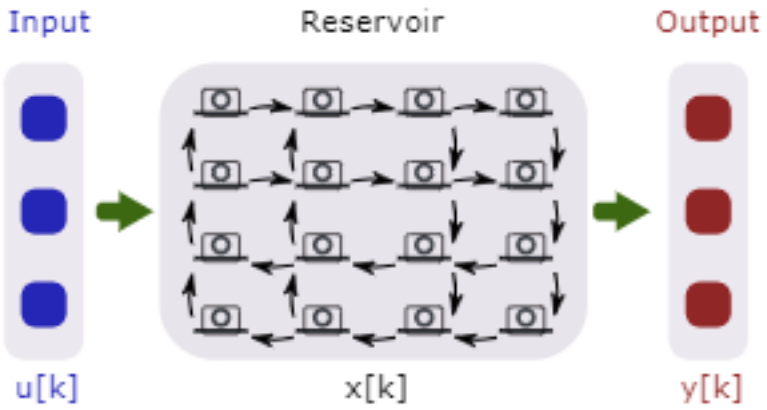

Figure 1.4.1: Theoretical photonic RC implementation using non-linear microring resonators and optical waveguides [35].

In 2017, Floris Laporte et al. proposed a quarter stadium shaped photonic crystal (PC) cavity for the structure of the reservoir as an alternative to the conventional node structure. This shape created interesting mixing dynamics and exhibited a form of memory as the signal remained in the cavity for some time before escaping out of an output. It was found that this form of memory combined with the non-linearity of the system were key properties for successful operation [30]. The non-linearity of the cavity caused reflections and enabled complex manipulation of the input signal. Floris Laporte concluded that photonic cavities on a silicon chip offer a promising candidate for an optical reservoir computer, with results showing good performance on non-linear XOR and header recognition tasks [30].

RC systems have been shown to be an effective strategy for implementing useful computations on dynamic systems, especially in instances where it is impractical to implement equivalents of memory cells and electronic logic gates. By treating the system as a reservoir in both simulations and physical concepts, functioning examples have been created using analogue electronics [36], opto-electronic and pure optical systems [31, 37] and even a bucket of water as the reservoir [38]. The readout layer of the RC is usually implemented in the conventional way, however. This project looks to capitalise on the benefits of a photonic crystal cavity structure on a silicon chip, first proposed by Floris Laporte et al, as the reservoir of an RC through design and simulation using high performance photonic simulation software.

1.5 Industrial Relevance & Project Motivation

The ever-quickening pace of technological advancement within the telecommunications sector spearheaded by the arrival of 5G wireless communications, means that the amount of data requiring processing at any given time is rising exponentially. In the year 1992 roughly 100GB of data was transferred over the internet per day. By 2022 this figure is estimated to have risen to 150,700GB of data transferred every second [39]. Moreover, the number of devices such as mobile phones, smartwatches and smart-home devices connected to IP networks is expected to rise to over three times the global population by 2022 [39] with 5G required to provide over 50 Exabytes/month of data traffic with a latency of less than 1ms[40]. This explosion in internet traffic, supported by the high data rates of 5G, will rapidly increase the pressures on mobile networks in the coming years as demands for faster, higher bandwidth and more stable mobile communications networks intensify [41]. In fact, it is predicted that communication speeds will reach 1-2TB per second in coming years, something that can only be realised through the use of integrated photonic circuit technology [42]. With this in mind, the establishment of a theoretical integrated photonic communications system is the primary motivation behind this project.

Currently established mobile communications infrastructure that operates using pure radio frequency (RF) technology has very limited bandwidth capabilities when compared to optical counterparts, see Table 1.5.1. Furthermore, the RF frequency spectrum has become congested with the provision of broadband services in new frequency bands becoming increasingly more problematic [43]. Optical wireless networking through the use of Microwave-photonics technology (MWP) in conjunction with an optical RC may offer a new approach to physical infrastructure and technology through exploiting the abundant processing bandwidth available within photonics technology. Whilst also opening up a much greater range of frequency bands for broadband services. Table 1.5.1 illustrates the significant difference in bandwidth between some typical photonic and RF components.

Table 1.5.1: Typical bandwidths of 1.55um photonic and conventional 2.4GHz RF components.

	Photonic	RF
Modulator	20GHz [44]	900MHz [45]
Filter	~Hundreds of GHz [46]	100MHz [47]

Although initially focusing on defence applications such as radar defence systems [48], MWP has now expanded into the commercial sector. Finding success in applications such as satellite communications and distributed antenna systems [49]. MWP is effectively a hybrid system that combines RF and photo-electronic technology, see Figure 1.5.1. A standard MWP system transmits and processes microwave signals in the optical part of the spectrum using photonic components such as modulators, filters and optical fibre. The motivations behind this type of system include significantly lower losses when compared to conventional pure RF transmission media, much greater bandwidth capacity and electromagnetic immunity [49]. All of which make this type of system well suited for use in antennas and communication infrastructure in general, especially given the increasing demands discussed previously. Furthermore, research has shown that photonic devices can satisfy signal processing niches where limited RF electronics no longer satisfy demands for bandwidth, such as mobile signal transmission and processing [50]. Figure 1.5.1 depicts a typical MWP system for the optical processing of microwave signals where RF signals are transmitted in the optical frequency domain. The optical input/output components allow for a variety of signal processing functions to be implemented in this domain.

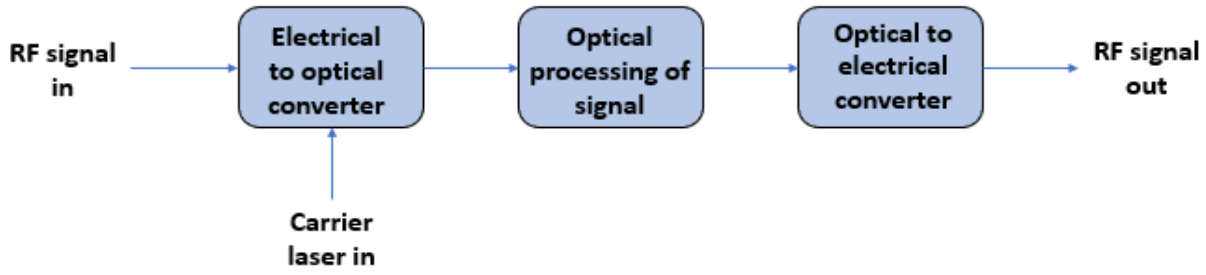

Figure 1.5.1: Block diagram of a typical MWP system.

Figure 1.5.2, from [49], illustrates the noteworthy difference in propagation loss between optical fibre and three commonly used RF transmission media. It is worth noting that the optical fibre in this figure is operating in single mode, as this will be discussed in more detail in Section 3.

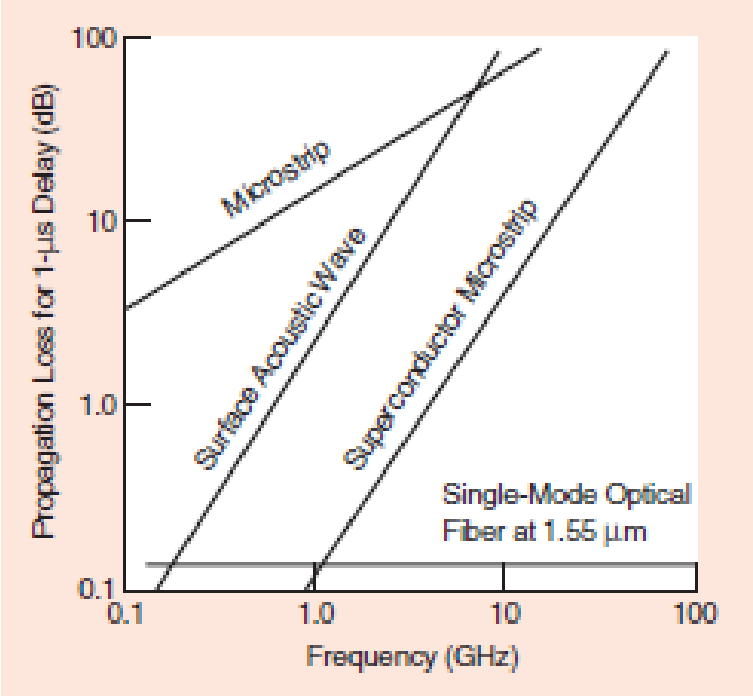

Figure 1.5.2: Propagation loss in optical fibre vs typical RF transmission media [49].

MWP offers not only performance enhancements in microwave and millimetre wave systems, but also new functionalities that are not possible using conventional electronic-based systems. Such as arbitrary waveform generation, frequency up/down conversion and instantaneous measurement, all of which are complex and, in some cases, not even possible using pure RF techniques [49]. The study of MWP also offers the potential to create new opportunities in communication and networking technology and is set to be an area of increasing importance going into the future.

During signal transmission, especially in more “chaotic” environments such as a major city or a location with many communication devices operating, signal loss and attenuation can occur. A problem that is becoming more and more prominent with the dawn of 5G and the many “internet of things” devices operating simultaneously that it will allow for. A major source of information loss in this way comes from inter-symbol interference (ISI). ISI is a type of signal distortion that occurs when noise and reflections are generated in a transmitted signal. Usually due to separate signals

mixing or a phenomenon called multipath propagation, where transmitted signals reach a receiver by multiple paths. Because the paths taken by separate parts of the transmitted signal can be of different lengths, some parts will arrive at the receiver at differing times. Thus, part or all of any given symbol will begin to spread and mix with subsequent transmission signals, interfering with the correct detection of those symbols at the receiver. Figure 1.5.3 illustrates multipath propagation that results in ISI in a 2-D city environment. Figure 1.5.4 depicts how ISI can affect a transmitted signal at the receiver.

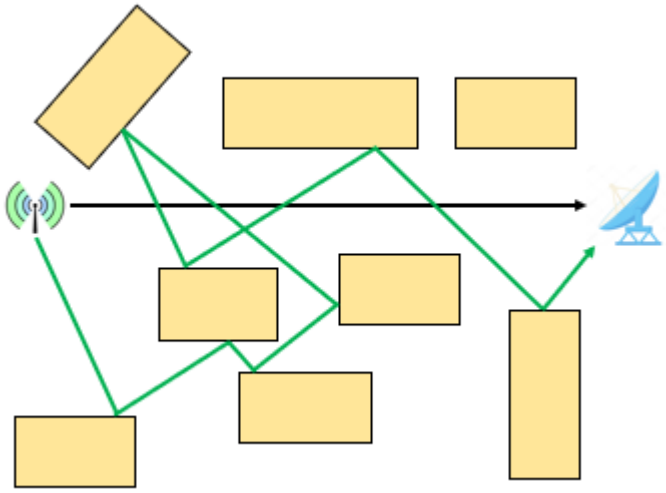

Figure 1.5.3: Multipath propagation in a city environment. The black arrow indicates the shortest signal path and the green arrow indicates a longer path part of the signal takes to the receiver.

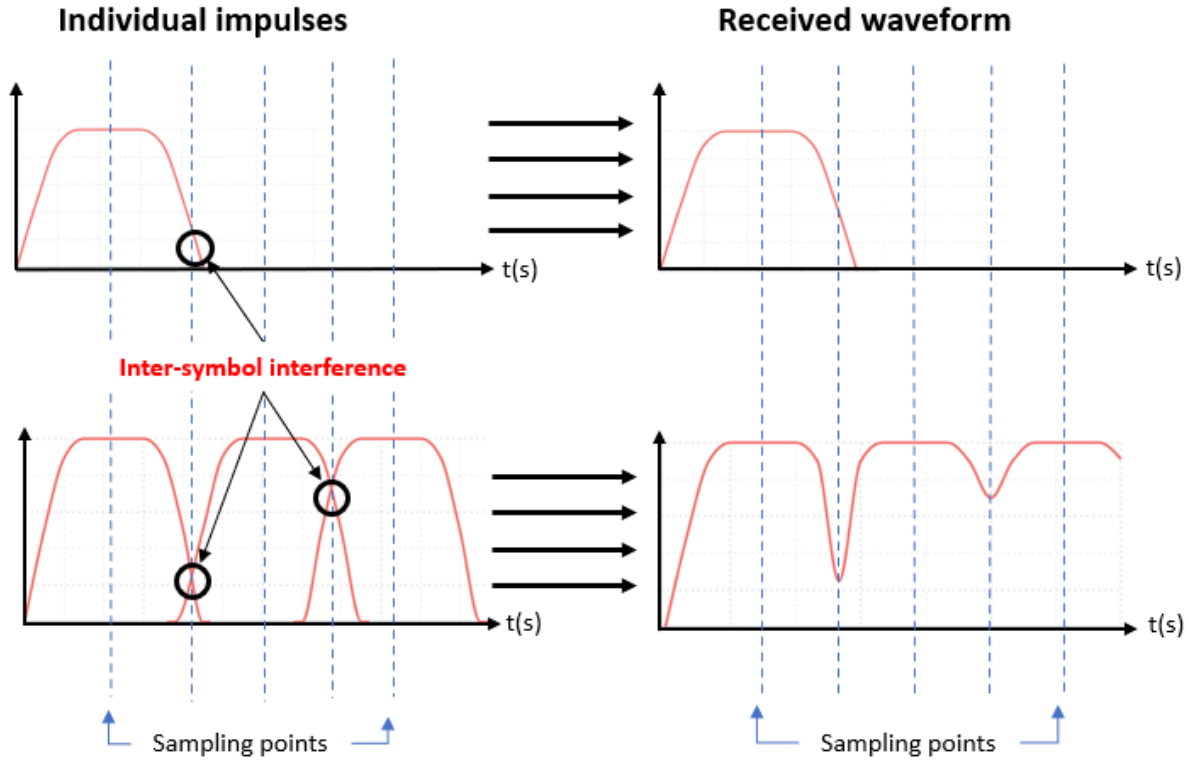

Figure 1.5.4: Individual impulses during transmission as ISI occurs, and the resulting mixed waveforms at the receiver.

The waveforms in Figure 1.5.4 were generated using a React.js component script (see Appendix C) and a modular graph component library for JavaScript called REAVIZ [51]. It is worth noting from Figure 1.5.4 that as the amount of data transmitted (bitrate) increases in a fixed time frame, ISI becomes more prominent in, and destructive to, the original signal. In this project the signal mixing that occurs due to ISI during signal transmission will be generated using an optical resonant cavity since, due to limitations on computational resources, it is highly impractical to model a physical city. As discussed in Section 1.4, a non-linear cavity, like Floris Laporte et al proposed in 2017, creates a chaotic environment that causes reflections and signal mixing, enabling complex manipulation of the input signal. Thus, simulating a dense urban environment effectively in a practical way. Figure 1.5.5 highlights the unpredictability and "chaotic" nature of a single ray of light in a "D" shaped resonant cavity with one input and two outputs using ray tracing software [52], see Section 4.0 for further justification of this particular cavity shape. Using multiple outputs provides more variation in timings when the signal exits the cavity and simulates the many paths the signal could take to the receiver in an urban environment.

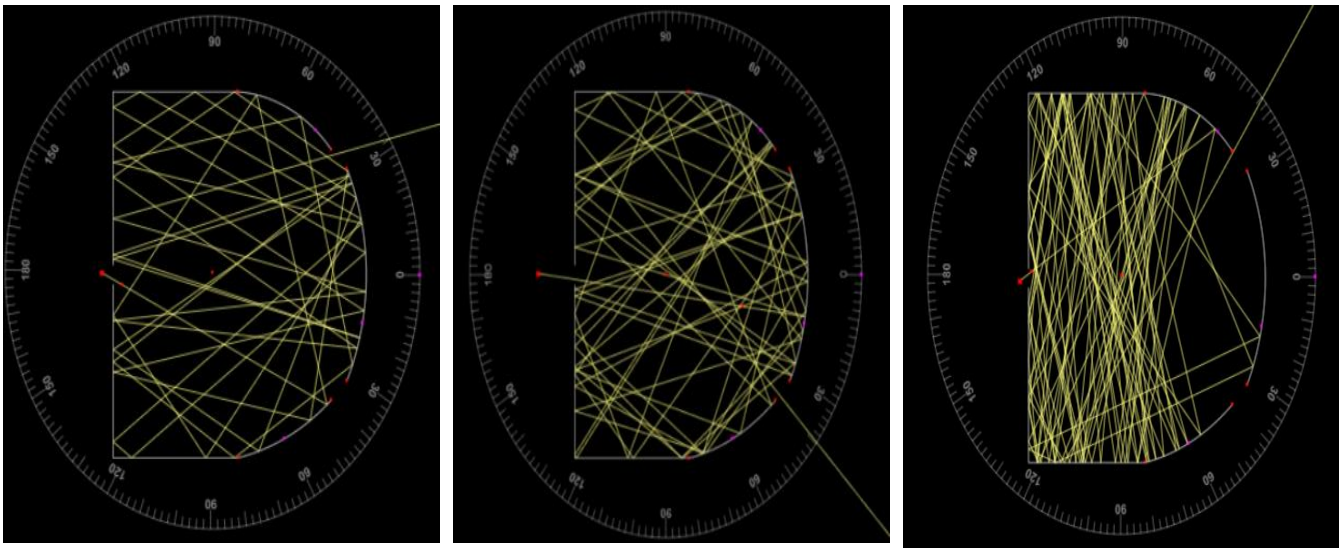

Figure 1.5.5: Ray tracing simulation in a "D" shaped resonant cavity.

In Figure 1.5.5, the chaotic nature of a "D" shaped resonant cavity is illustrated. It is also shown how a slight variation in the initial starting conditions, in this case the angle of the input ray, results in significantly different signal reflections and mixing before the ray exits the cavity. Thus, the unpredictable and chaotic nature of the resonant cavity makes it an ideal structure for mimicking a city environment in which communication signals such as 5G signals are transmitted. The task of the reservoir computing read-out layer is to then reconstruct the input signal once this mixing and attenuation has occurred, see Section 2.0.

Reservoir computing has started to find relevance and interest in a range of different industries, with the simplicity of the training method being particularly attractive over other types of ANNs. Table 1.5.2, adapted from [29], highlights some examples of industries that are starting to develop applications for RC systems. Most of the applications being studied in these industries are based in machine learning, such as time series forecasting and system approximation. With RC systems meeting demands for low training costs and real-time processing [29].

Table 1.5.2: RC applications in industry

Industry	Examples
Biomedical	EEG, fMRI, EMG
Visual	Image & video processing
Audio	Speech & music processing
Machinery	Robotics, sensors, controllers
Communication	Information & signal processing
Security	Cryptography
Financial	Stock index, stock price, exchange rate
Social	Grammar, syntax & language correction & prediction

The main motivation behind this project is to deliver an integrated MWP-RC telecommunications framework with far greater data processing capabilities than traditional RF technology can currently achieve. Thus, providing a viable solution for future communication infrastructure and the ever-increasing demands that will be required of it. Hence this project will primarily find relevance within the telecommunications industry. The outcomes of this project, however, may also find relevance amongst other groups such as within the artificial intelligence community and other industries, as in Table 1.5.2, that are looking to employ reservoir computing techniques. Finally, the nanofabrication and photonics industries may also find interest as the project provides new applications of combined MWP-RC technology on silicon chips.

2.0 System Overview

Figure 2.0.1 illustrates the proposed MWP-RC based telecommunications system. The overall system operates as follows. A high-frequency RF signal, such as a 5G signal, is received from a transmission source, for example, a mobile phone or other communications device. After which it is upconverted to an optical frequency (infrared) with a bandwidth between 75-150THz in order to take advantage of the previously discussed benefits of optical transmission, using an electro-optical modulator such as [44] mentioned in Table 1.5.1 previously. Infrared frequency is used because attenuation of the signal over long transmission distances is typically lower than other optical frequencies [53]. Next, the optical signal is fed into the photonic crystal resonant cavity, or reservoir, of the RC to manipulate and mix it. Simulating transmission through an urban environment and the interference that occurs in the process. Multiple exit optical waveguides in the cavity mimic the many paths the signal could take in this environment. After the signal exits the reservoir it is weighted using the RC read-out layer (acting as a receiver) in order to post-process and reconstruct the original RF signal. From which the error in the system and thus success of the RC system can be established. The entire RC system (resonant cavity and RC read-out layer) will be simulated in this project and thus all results are theoretical.

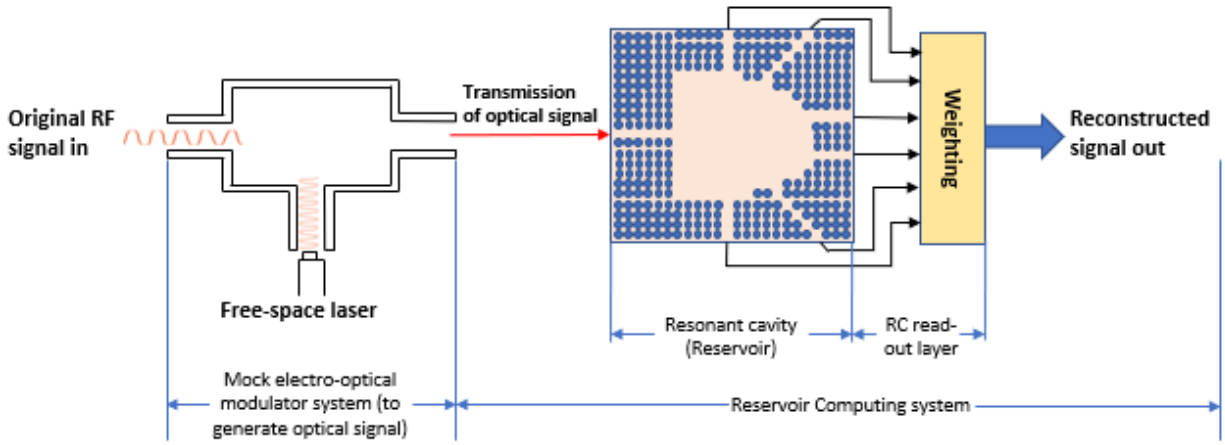

Figure 2.0.1: Proposed microwave-photonics system.

For the purpose of this project the optical input signal to the reservoir is generated mathematically through MATLAB for the simulations, rather than through the use of a physical electro-optical modulator. The resonant photonic crystal cavity is designed and tested using the Finite-Difference Time-Domain (FDTD) analysis method and the commercial software package Lumerical, see Section 2.1 for more details on these. The training data for the system is obtained through simulation of the optical resonant cavity using Lumerical and the RC read-out layer is simulated using MATLAB.

2.1 FDTD analysis and Lumerical Software overview

In this project electromagnetic simulations in the time domain are undertaken using the 3-dimensional FDTD method and Lumerical software. With the purpose of assessing the optical performance of photonic crystals (PC) for the resonant cavity of the RC system. First proposed in Kane Yee's paper "*Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media*" (1966) [54], the FDTD method is a numerical technique for solving Maxwell's equations for electromagnetic (EM) wave propagation in the time domain. Essentially the transient response of the system is acquired from which information about EM wave propagation is extracted using Fast Fourier Transform (FFT). The aforementioned Maxwell's equations used in FDTD analysis that describe the relationship between electric and magnetic fields are as follows:

$$\nabla \cdot H = 0 \tag{2.1.1}$$

$$\nabla \cdot E = \frac{\rho}{\varepsilon} \tag{2.1.2}$$

$$\mu \frac{\delta H}{\delta t} = -\nabla * E \tag{2.1.3}$$

$$\varepsilon \frac{\delta E}{\delta t} + \mathrm{J} = -\nabla * \mathrm{H} \tag{2.1.4}$$

Equation 2.1.1, Gauss's law for magnetic fields, dictates how magnetic fields relate to magnetic charges and states that magnetic monopoles (a hypothetical particle with only one magnetic pole) do not exist. Equation 2.1.2, Gauss's law for electric fields, dictates how electric fields (E) relate to electric charges (ρ). Equation 2.1.3 and 2.1.4, Faraday's and Ampere's laws respectively, describe how a change in an electric field affects a change in a magnetic field and vice-versa. The FDTD method solves these equations on a discrete spatial and temporal grid known as Yee's unit cell that is placed around the structure under simulation, illustrated in Figure 2.1.1 adapted from [55]. In Yee's grid each field component is solved at a slightly different position within the grid cell [56].

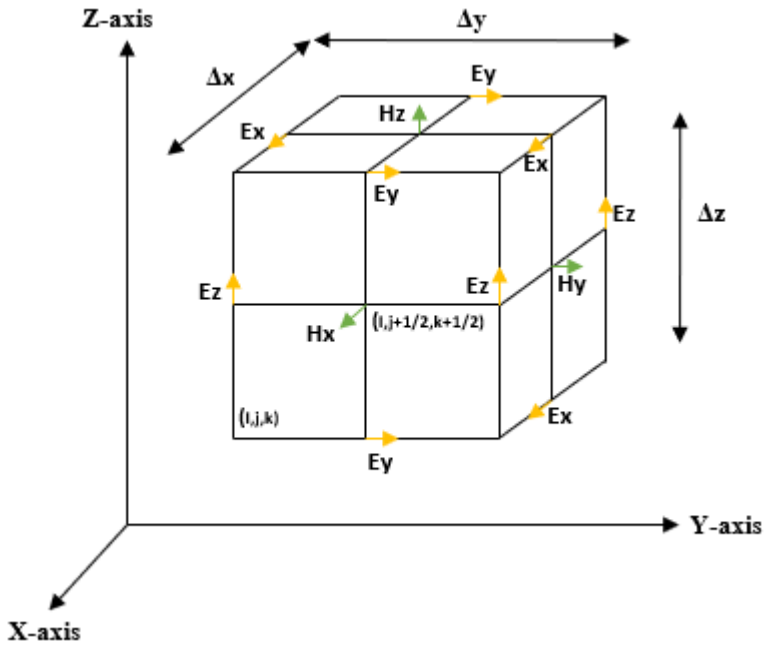

Figure 2.1.1: Yee's Unit Cell for the positions where E and H are analysed.

In the case of this project, this method is first employed to calculate the bandgap of a photonic crystal structure (Section 3.2), then to calculate EM wave propagation through an optical waveguide (Section 3.3) and finally to calculate EM wave propagation inside the resonant cavity (Section 4.0).

The FDTD algorithm repeatedly calculates both the electric and magnetic fields in incrementing periods of time within the structure until all the fields within the system decay to zero. Once the simulation is completed an FFT program is used to extract frequency information from the transient response of the system. The operation of Yee's FDTD algorithm can be summarised as follows:

- Replace all derivatives in equations 2.1.3 and 2.1.4 with finite differences and discretise space and time.
- Solve the resulting equations to attain equations that express the un-known future fields in terms of the previous fields.
- Evaluate the magnetic and electric fields one time-step into the future.
- Repeat previous steps until all EM fields over the desired time duration have been acquired or all fields within the system decay to zero.
- Extract frequency information from the system response using FFT.

Figure 2.1.2, adapted from [57], illustrates the main steps undertaken in a 3D-FDTD simulation.

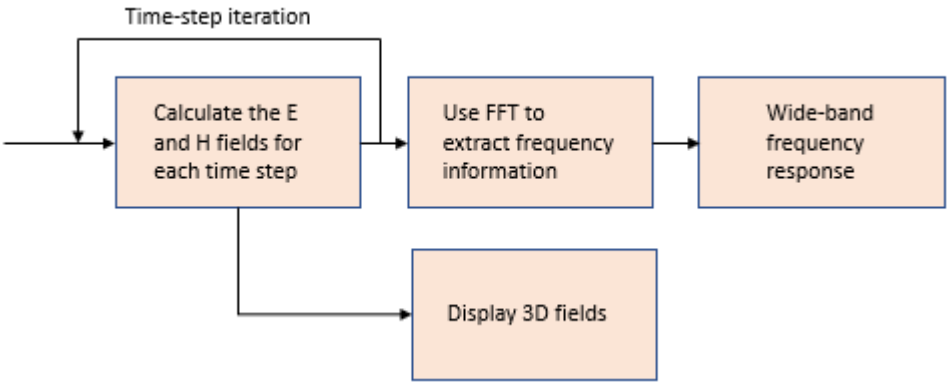

Figure 2.1.2: Main steps in the 3D-FDTD analysis method.

The specific formula derivations and calculations that are used in the 3D-FDTD method are very convoluted and thus they are not listed here. Furthermore, in this project they are all handled by Lumerical during the simulations. Lumerical is an industrial standard photonic simulation software package that allows for the modelling of optical, electrical and thermal effects at a physical level based on the 3D-FDTD method, offering a large range of analysis capabilities for the design and optimisation of photonic integrated circuits [58]. The graphical user interface of the software consists of four main elements, a script editor, toolbox, main window and a structured layout tree. Figure 2.1.3 illustrates the typical Lumerical software graphical user interface with a structure built in the main window. In this image, the FDTD region where the FDTD analysis would take place is the orange mesh in the centre of the structure in the main window. Table 2.1.1 details each main section of the interface with a brief description of its purpose.

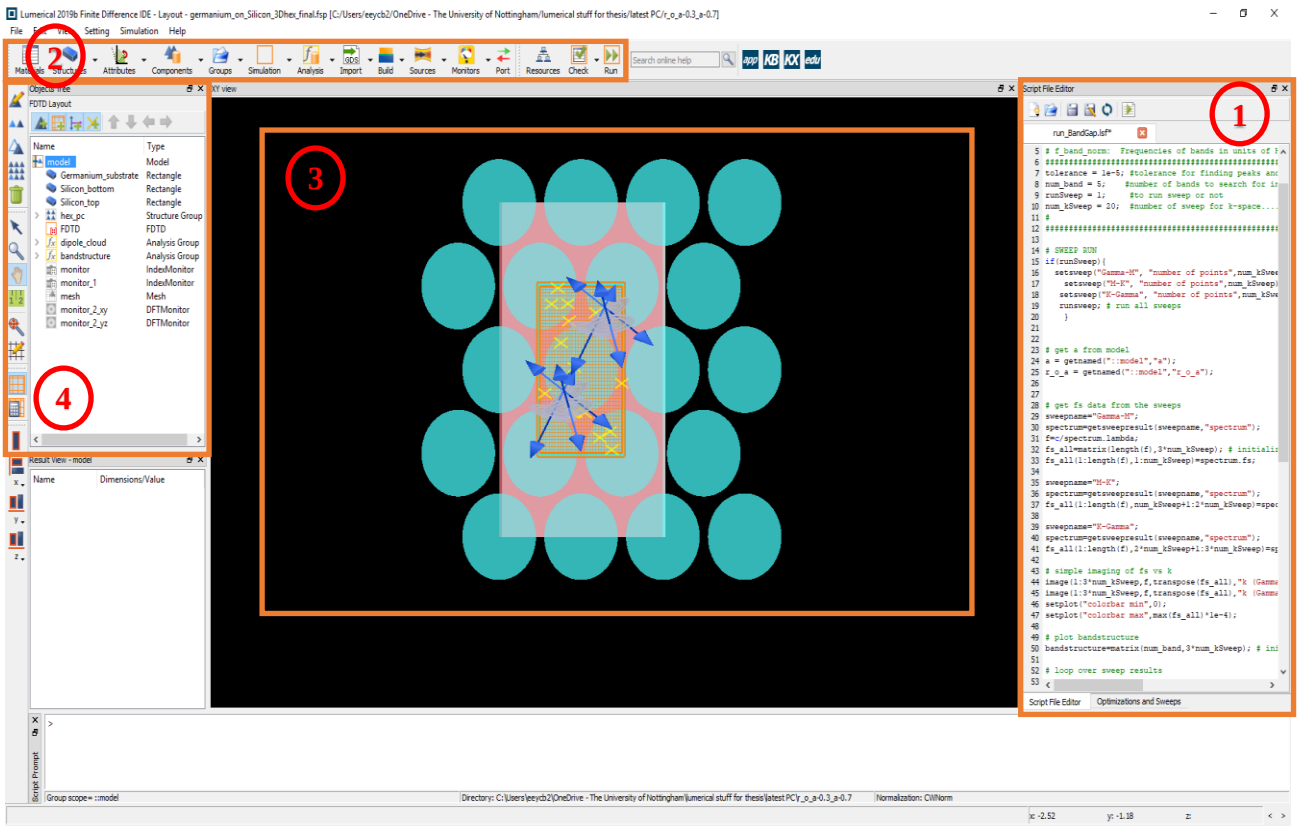

Figure 2.1.3: Lumerical software graphical user interface.

Table 2.1.1: Details of Lumerical graphical user interface in Figure 2.1.3.

Section number	Section name	Description
1	Script editor	Allows for the use of Lumerical's scripting language to automate tasks, launch simulations and analyse results.
2	Toolbox	Offers many relevant components, sources, monitors and functions for simulation.
3	Main window	Area where elements and structures are placed for simulations.
4	Layout tree	Details and organises the objects in the main window allowing the user to quickly locate and sort elements of a simulation.

3.0 Design of the optical waveguide

The input and outputs of the resonant cavity are effectively optical waveguides that guide the signal in and out of the structure. Therefore, before the resonant cavity could be designed, the design of an optical waveguide was necessary. Optical dielectric waveguides are critical to the confinement and transmission of optical signals between photonic components and devices as they offer immunity to electromagnetic interference, low-loss light propagation [59] and minimal loss of energy, especially when compared to optical transmission via free space components such as mirrors or lenses. A dielectric waveguide consists of an internal "core", constructed of a high-index optical medium that guides the flow of optical waves along its longitudinal direction [60]. The core is sandwiched between two low-index optical mediums, the substrate and cladding. Figure 3.0.1 illustrates a typical planar optical waveguide with light propagating through it. The refractive index, a measure of the speed of light in a given material, is notated as n_c, n_s and n_f for the cladding, substrate and core respectively.

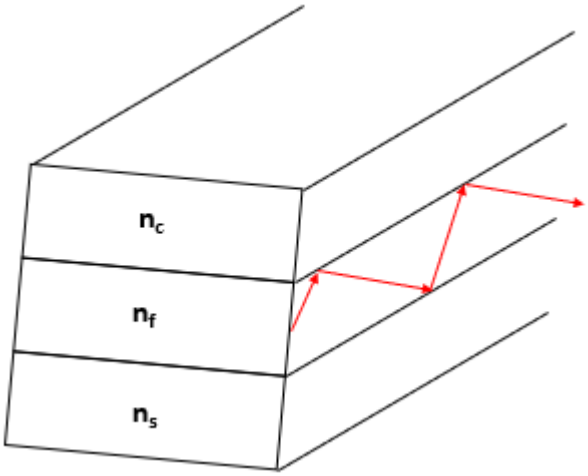

Figure 3.0.1: Typical structure of a planar waveguide.

Whilst EM waves are able to travel through waveguides using a multitude of different modes, in most applications, and indeed this one, it is generally preferable for optical waveguides to operate in single mode. This is obtained by reducing the dimensions of the waveguide until only the fundamental modes are radiating. The smaller dimensions mean that the number of reflections created as light travels through the waveguide core will decrease and thus attenuation is lowered and the signal can travel further. In order to assist with the design and determination of important qualities of the optical waveguide, a photonic crystal structure was first designed. This PC effectively acts like a "cage", trapping certain frequencies of light. The frequency range of which can be altered by changing certain qualities of the structure. Thus, by designing a PC that traps the desired frequencies inside, a waveguide can be created for these frequencies by introducing intentional defects to the structure. An intentional defect is a finite region of space within the PC structure where light propagates. Therefore, if a channel is created in the PC structure, the trapped frequencies of light are able to propagate through the structure and thus are "guided" from one location to another.

3.1 Photonic Crystal theory

Photonic crystals are periodically structured EM media through which a range of frequencies of light cannot propagate [61]. This range of frequencies is known as a photonic band gap (PBG) and is caused by variations in refractive index of the materials. There are two basic topologies for PC's, holes with a low refractive index surrounded by a high refractive index material, see Figure 3.1.1 (a). Or high refractive index rods surrounded by a low refractive index material, see Figure 3.1.1 (b). Low index holes are best suited to transverse electric (TE) modes of light, where E is restricted to the xy plane and conversely high index rods are best suited to transverse magnetic modes (TM) [61], where H is restricted to the xy plane, as each topology produces a strong bandgap for TE/TM modes respectively. To achieve a large PBG, the dielectric structure requires a complex topology with thin "veins" between the media that run in all directions for the electric field lines to pass through. This allows for the confinements of the lowest band(s) whilst forcing the upper band(s) to a far higher frequency since the thin veins cannot support multiple modes [62]. Figure 3.1.1 illustrates the two basic topologies for PCs, using a hexagonal-shaped lattice structure as this has been shown to generally yield the largest band gaps [61], as opposed to triangular, square or rectangular shaped lattices for example. Red and blue colours indicate areas of low and high refractive index respectively. By varying the values of the lattice constant α, and the relative radius of the holes/rods r, the frequency ranges the PBG appears at can be manipulated.

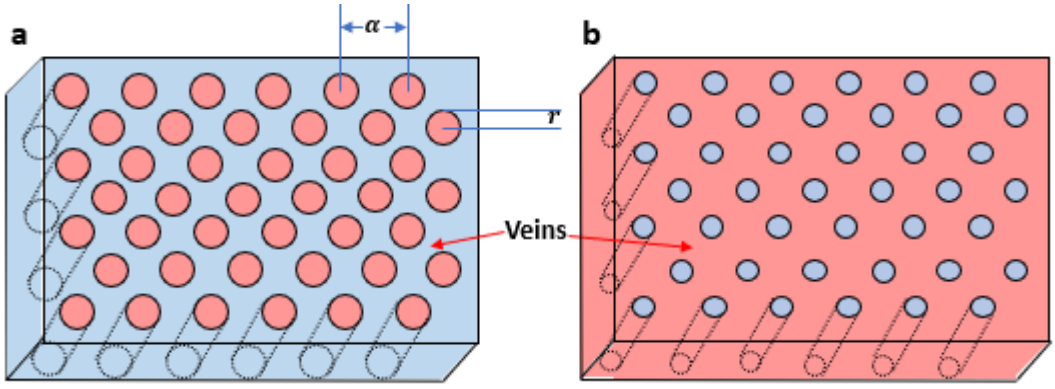

Figure 3.1.1: Basic topologies for PCs (a) Low index holes in high index material and (b) High index rods in low index material.

Figure 3.1.2 (a) illustrates how a PC structure creates a PBG by blocking the propagation of certain frequencies. When an EM wave produced by a source such as a dipole, as seen later in Figure 3.22 from Section 3.2, encounter unit cells of the PC structure, for certain frequencies depending on the structure of the PC, reflections will be generated. These refracted and reflected waves will combine and cancel out the original incoming wave and thus that range of frequencies will not pass through the structure. The Brillouin zone of a hexagonal lattice structure is illustrated in Figure 3.1.2 (b) with the irreducible Brillouin zone highlighted. The Brillouin zone is a region of space closer to the origin than to any other reciprocal lattice points and encloses the set of all wavevectors $\vec{k}$ [63]. The irreducible Brillouin zone is an area in which the extremes of the bands almost always occur at its boundaries, with the corners of this area being denoted by gamma (Γ), M and K. Γ is the origin where $\vec{k}$ = 0 and K and M are high-symmetry directions from this point. When plotting the PBG of a PC, it is conventional to only plot the bands along these high-symmetry boundaries to identify the band gap [60], see the PBG illustrated in Figure 3.2.4 in Section 3.2 for an example of this.

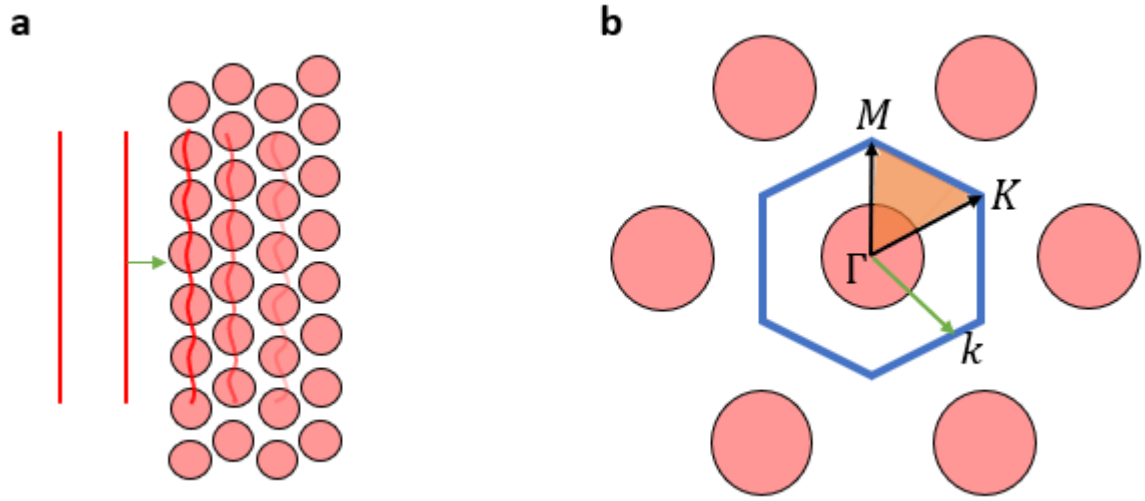

Figure 3.1.2: (a) An EM wave unable to pass through a hexagonal PC lattice, (b) The Brillouin zone of the hexagonal lattice centred at the origin (Γ). An arbitrary wave vector k is also shown. The irreducible Brillouin zone is shaded orange with high-symmetry points M and K.

As mentioned previously the range of frequencies for which a PBG exists in the photonic crystal can be manipulated and optimised by varying the ratio (r/a) between the relative radius of the holes/rods, (r), and the lattice constant, (α). As the value of r/a is increased/decreased the size of the holes/rods will likewise increase/decrease and thus the "veins" between the holes/rods will inversely increase or decrease in size. Varying size of the veins in the structure will vary the size of the PBG. If the veins between the media become too small or too large, then the veins and thus the PBG will cease to exist entirely. This effect is seen later in Figure 3.2.1 in Section 3.2. Other than the dimensions of the holes/rods, the existence of a PBG will also be affected by other criteria such as the materials used in the PC and the dimensions of the surrounding material. Because of this, the design of a PC is effectively a process of trial and error.

3.2 Design of the Photonic Crystal Slab

As previously noted, the optical infrared transmission signal has a bandwidth between 75-150THz. Thus, the PC structure in this project was aimed at obtaining a PBG in between 75THz – 150THz. In order to achieve a PBG for infrared frequencies the dimensions of the PC must be in the micron (µm) range [62]. As discussed previously in Section 3.0, using small dimensions such as this also allows for the use of single mode operation in the structure. From an experimental standpoint, TE modes of light are the easiest to couple from external excitation and the easiest to use when interfacing equipment such as within MWP applications. Thus, in this project the PC is built using low index holes in a high index material because this has been shown to be best suited to TE modes of light where $\vec{E}$ runs around the holes through the veins in the structure. Furthermore, the lattice of holes is shaped in a hexagonal structure because the largest band gaps have been shown to typically arise in this topology [61].

The photonic crystal structure consists of a germanium substrate with a thickness of 0.2µm, sandwiched between silicon cladding with a thickness of 0.8µm and patterned with a hexagonal lattice of drilled air holes with a thickness of 1.2µm. The refractive index of these materials is 4, 3.5 and 1 for the germanium, silicon and air respectively. A germanium on silicon (Ge-on-Si) structure was selected as this is currently available in the George Green Institute for Electromagnetics Research (Dr Phang) should this design be experimentally fabricated. Furthermore, this configuration has been shown to be most suitable for operation between 75THz – 150THz as germanium exhibits low losses in this range [63] and silicon has shown high potential for use in the creation of optical devices in previous research [64]. In the design of photonic crystals, typical r/a values are usually selected between 0.18 and 0.5 depending on the materials being used and the desired PBG [61,65,66, 67, 68,69]. Thus, this was a starting reference for optimising the structure. In this instance the final value of r/a was chosen based on the testing undertaken in Figure 3.2.1 where the widest bandgap was found to be at a value of r/a = 0.475 when a = 1.15µm and r = 0.55µm.

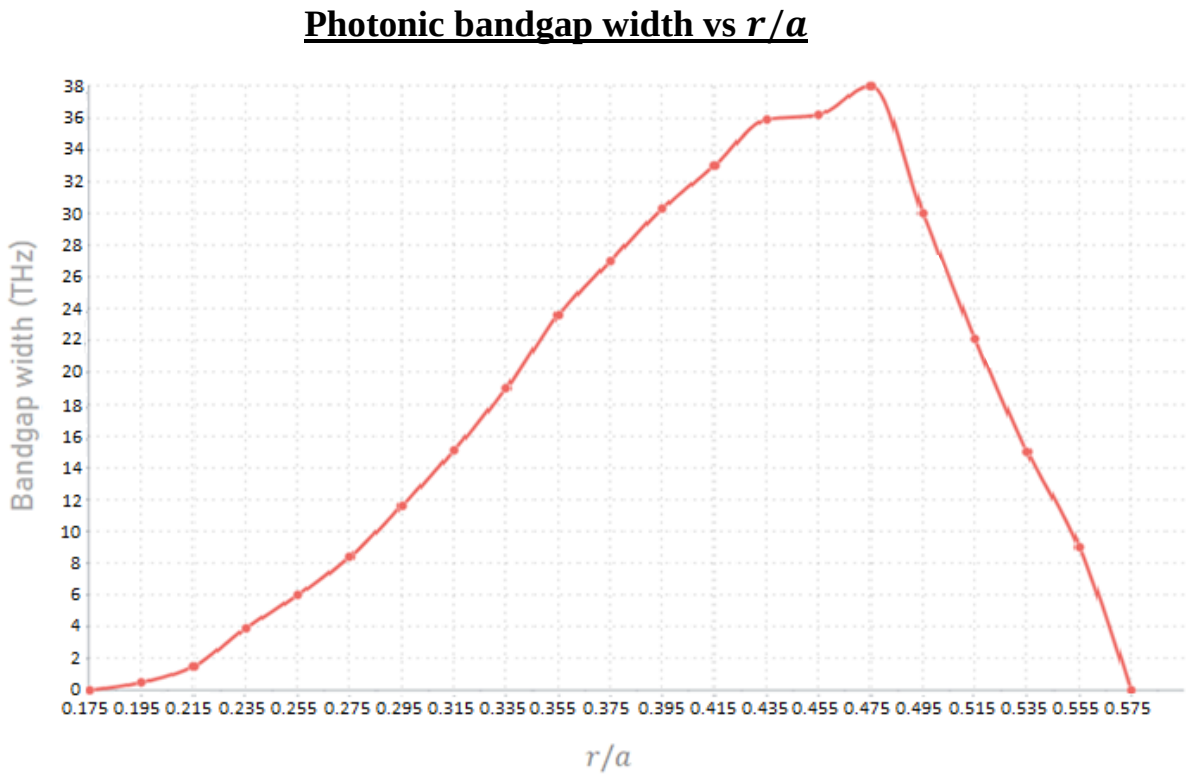

Figure 3.2.1: Width of PBG in the PC for varying values of r/a.

The graph from Figure 3.2.1 was generated using a React.js component (see Appendix D) and the REAVIZ modular chart component library for JavaScript [51]. The React.js component script that has been written here is highly customisable and will generate line charts for any given number of data points contained in an array of JavaScript and/or JSON objects that are passed into it. Thus, it may find use should this project be taken further in future work. As illustrated by Figure 3.2.1, the largest achieved bandgap was 38THz when r/a = 0.475. This PBG could potentially be optimised and widened through further experimentation with r/a, however due to time constraints on the project, in this instance an acceptable result was gained and taken forwards. It can also be seen how the width of the band gap decreases and eventually collapses when the size of the holes, as a function of r/a, become too large or too small, as discussed previously. Figures 3.2.2(a) and (b) show different perspectives of the designed photonic crystal in Lumerical software.

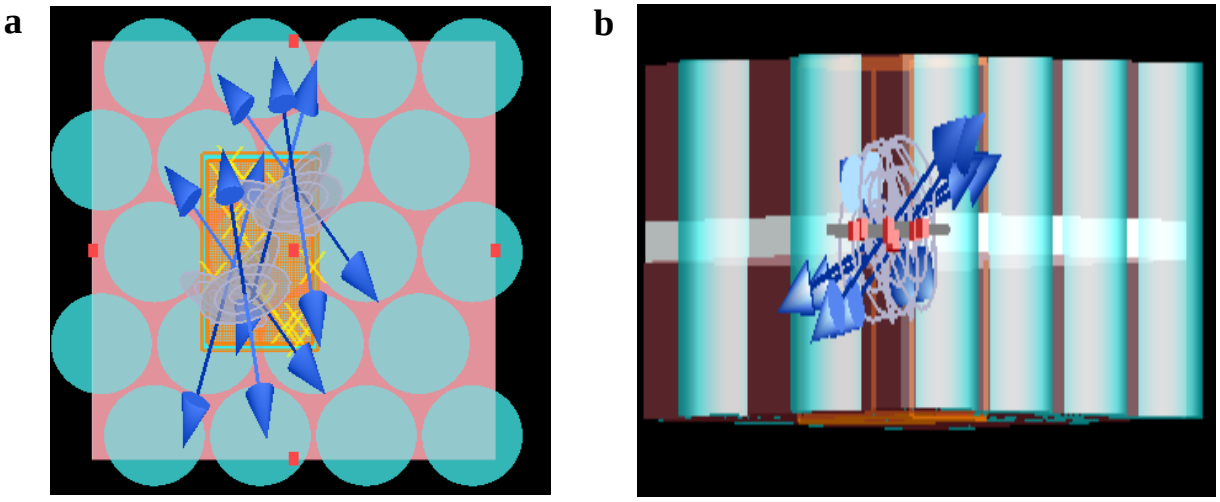

Figure 3.2.2: PC structure for (a) Top-down view and (b) side profile.

As can be seen from Figure 3.2.2 (b) the PC consists of a thin germanium substrate (the white material) surrounded by silicon cladding (the pink material) with a lattice of air holes running through the structure in a hexagonal topology. The circular objects with blue arrows protruding from the centre are dipoles that emit EM waves that propagate within the structure in order to calculate the PBG using the 3D-FDTD method. The yellow “cross” shaped elements are monitors for measuring EM wave activity within the structure. Finally, the orange rectangle structure is the 3D-FDTD region. As discussed previously in Section 2.1, the 3D-FDTD region is placed as a 3D grid around the structure or section of the structure that is being simulated. In this case it is placed around a single hole and its surrounding area in the photonic crystal. This is in order to ensure that the photonic crystal operates successfully as a cage for the desired optical frequencies, whilst not covering too much of the structure with the FTDT region such that the simulation time becomes too large due to the number of calculations that would need to take place. Furthermore, the number of cells within the grid were selected with consideration to the length of time the simulation would take as a finer grid yields more accurate results but also takes longer to simulate. In this case there were 46, 80 and 76 cells in the x, y and z-directions respectively. Figure 3.2.3 shows a closer view of this 3D-FDTD grid in both the xy and yz directions to illustrate its grid structure. Figure 3.2.4 displays the photonic band structure of the PC after 3D-FDTD simulations.

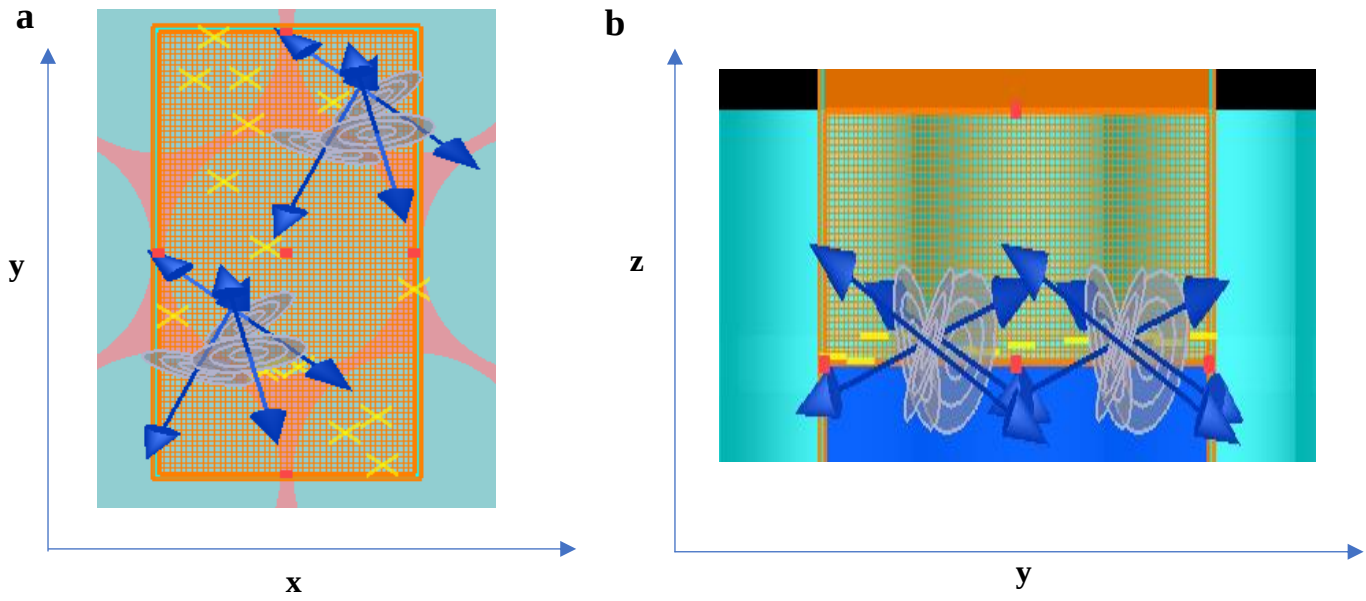

Figure 3.2.3: Illustration of 3D-FDTD region in the PC structure for (a) xy view and (b) yz view.

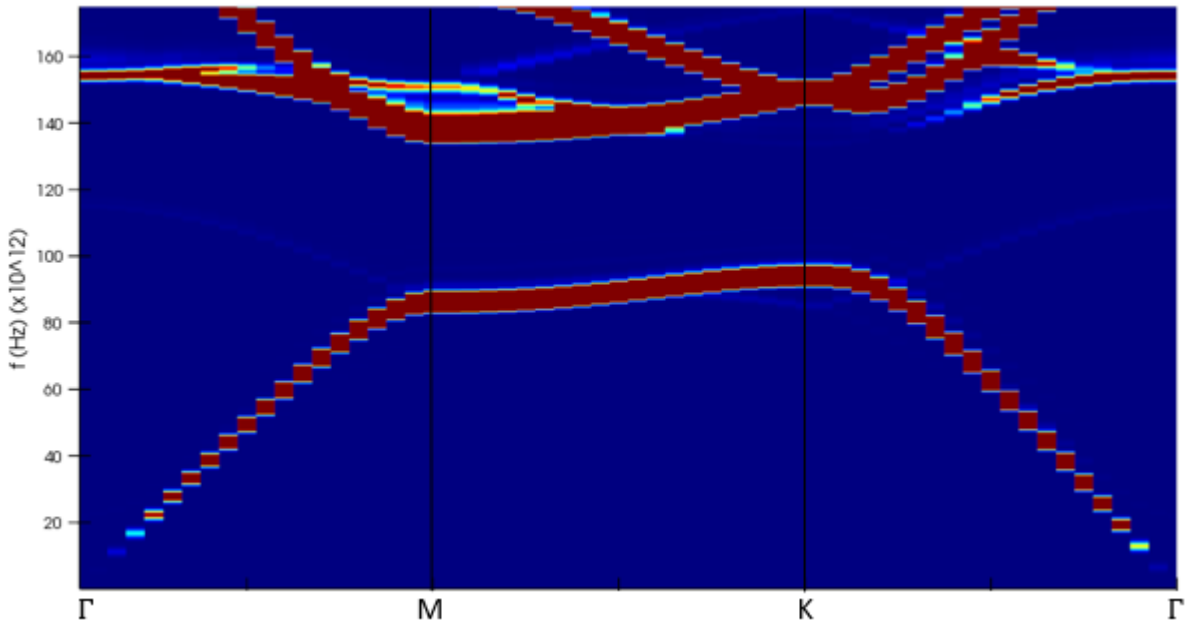

Figure 3.2.4: Photonic band structure for the Ge-on-Si PC along the Γ-M-K-Γ directions.

The red lines in Figure 3.2.4 above indicate TE bands that are propagating through and thus escaping from the PC structure. The blue region in-between the red lines is the photonic band gap and indicates the frequency range for which there are no waves propagating through the structure. This PBG can be clearly noted and sits in the 97-135THz frequency range. The results illustrated here clearly showcase the viability of this PC structure for containing a desired frequency range. Thus, it could be used for the construction of the optical waveguide.

3.3 Photonic Crystal Optical Waveguide

Once the PC structure was constructed and verified, a photonic crystal slab waveguide could be created using the designed PC structure with the optimum specification obtained previously, see Table 3.3.1. By removing a line of holes from the PC structure an intentional line-defect is created which allows for low loss containment and transmission of light due to the presence of the PC bandgap. Figure 3.3.1 illustrates the photonic crystal waveguide used for FDTD simulations. The purple arrow is the input signal source that injects a Gaussian pulse in single mode (see Figure 3.3.2) into the waveguide to test for its correct functionality. This signal has a bandwidth of 75THz, centred at 112.5THz, since the photonic bandgap of the structure is between 97-135THz, as calculated previously. This is known as impulse response testing where the system is given an impulse "kick" and the response is recorded [70]. The orange structure around the waveguide is the 3D-FDTD region boundary. The number of grid cells for the FDTD simulation were 1229, 413 and 80 in the x, y and z directions, respectively. They are not shown in Figure 3.3.1 for clarity.

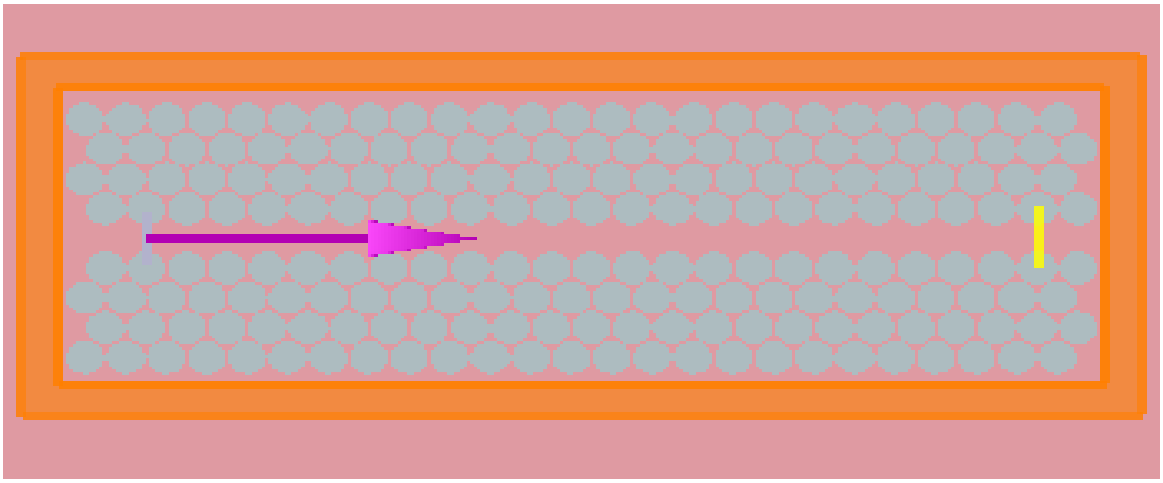

Figure 3.3.1: The designed photonic waveguide structure for FDTD simulation.

Table 3.3.1: Design parameters of the optical waveguide.

Parameters	Value
Length (μm)	33.8
Width (μm)	17.1
Thickness (μm)	1.2
Germanium substrate thickness (μm)	0.2
Silicon cladding thickness (μm)	0.8
Refractive index of air holes	1
Refractive index of Germanium substrate	4
Refractive index of Silicon cladding	3.5
Lattice constant α (μm)	1.15
Relative radius of air holes r (μm)	0.55
r/a	0.475

Figure 3.3.2 (a) displays the single mode optical profile of the Gaussian input signal that is transmitted through the waveguide from the source. Figure 3.3.2 (b) displays this impulse in the time domain. Figure 3.3.3 illustrates the electric field strength of the single mode gaussian impulse propagating through the waveguide during the 3D-FDTD simulation.

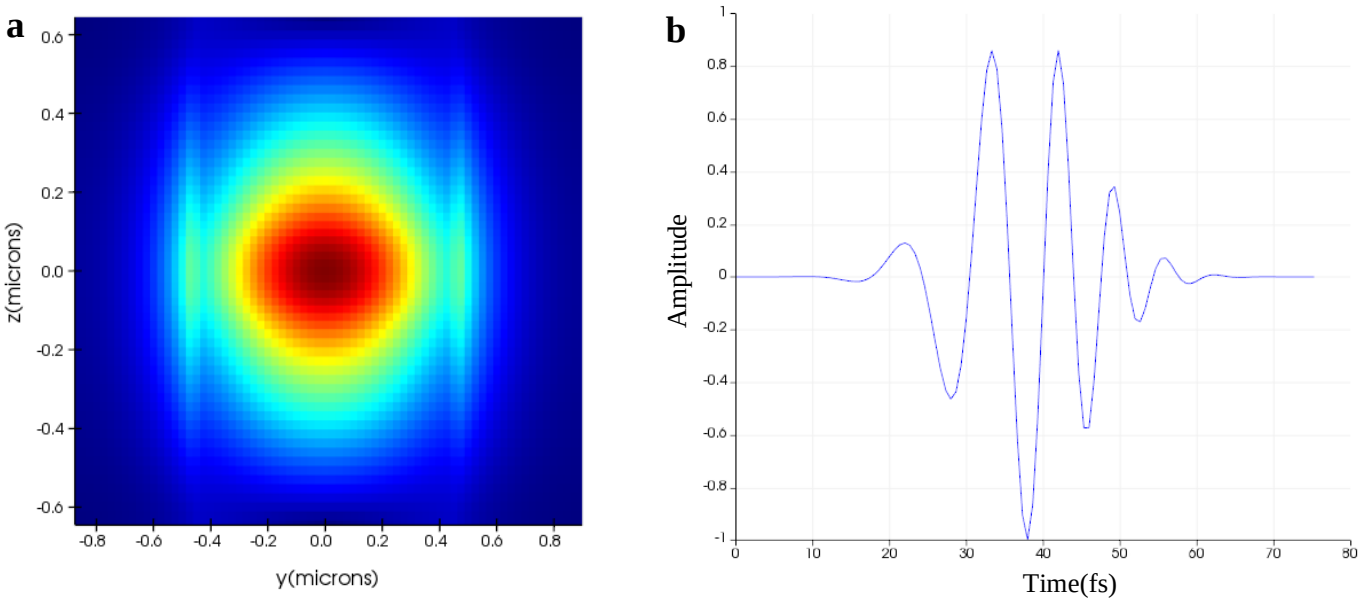

Figure 3.3.2: (a) Optical mode profile for single mode at the source port of the waveguide at 100THz and (b) Gaussian impulse input signal in the time domain.

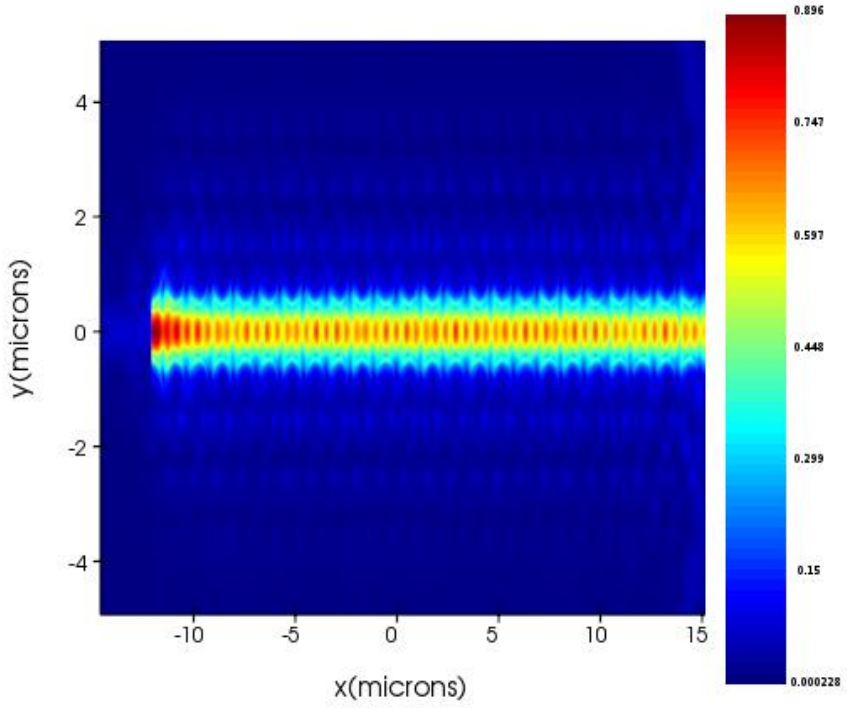

Figure 3.3.3: Electric field strength of the Gaussian pulse as it propagates through the optical waveguide at 100THz.

As seen in Figure 3.3.3, the pulse is contained within the waveguide and reaches the output without escaping or dissipating through the photonic crystal structure. Aside from initial transient losses at the input, the electric field strength remains constant throughout the structure. Figure 3.3.4 displays the single mode measured at the output at 100THz. Comparisons with Figure 3.3.2 (a) show that it has propagated through the waveguide successfully without being transformed or dissipating during transmission.

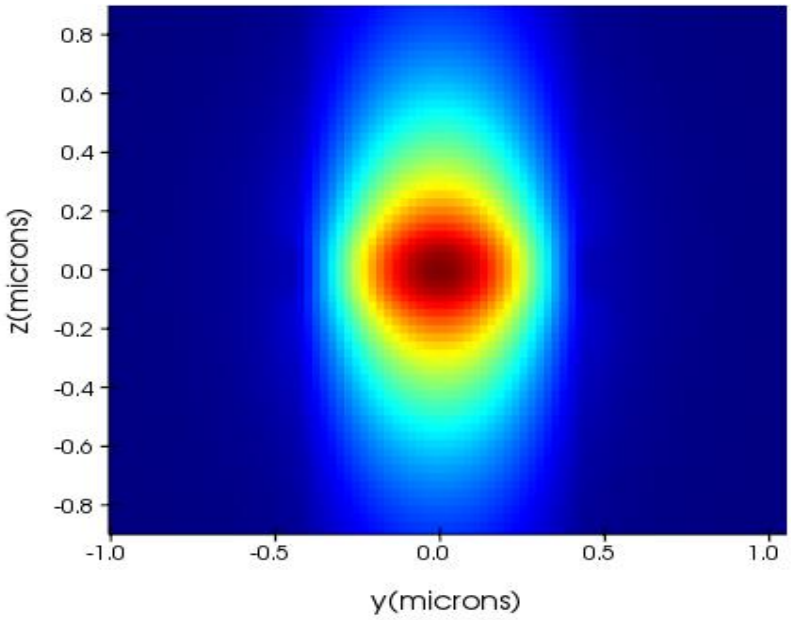

Figure 3.3.4: Optical mode profile at the output of the optical waveguide at 100THz.

Through performing 3D-FDTD simulations on the photonic crystal structure and subsequently the optical waveguide that was constructed using the previously designed PC, it has been shown that these structures have strong light confinement capabilities and thus are viable for the construction of the RC resonant cavity.

4.0 Design of the Optical Resonant Cavity

As the waveguide was generated by creating a line defect in the PC, the optical resonant cavity was produced by creating an intentional defect cavity in a larger PC wafer with the same specification as ascertained previously. Thus, the cavity only allows single mode wave propagation and provides the same light trapping qualities as the waveguide does. The resonant cavity was designed with a 'D' shape as this has been shown to exhibit fully chaotic ray dynamics [71]. The reader is also referred back to Figure 1.5.5 for a demonstration of this. The non-linearity and asymmetric shape of the cavity causes reflections and enables complex manipulation and mixing of the input signal, simulating an urban environment. Figure 4.0.1 displays the resonant cavity carved out of a PC wafer and Table 4.0.1 displays the design parameters of the resonant cavity that were used in the FDTD simulations.

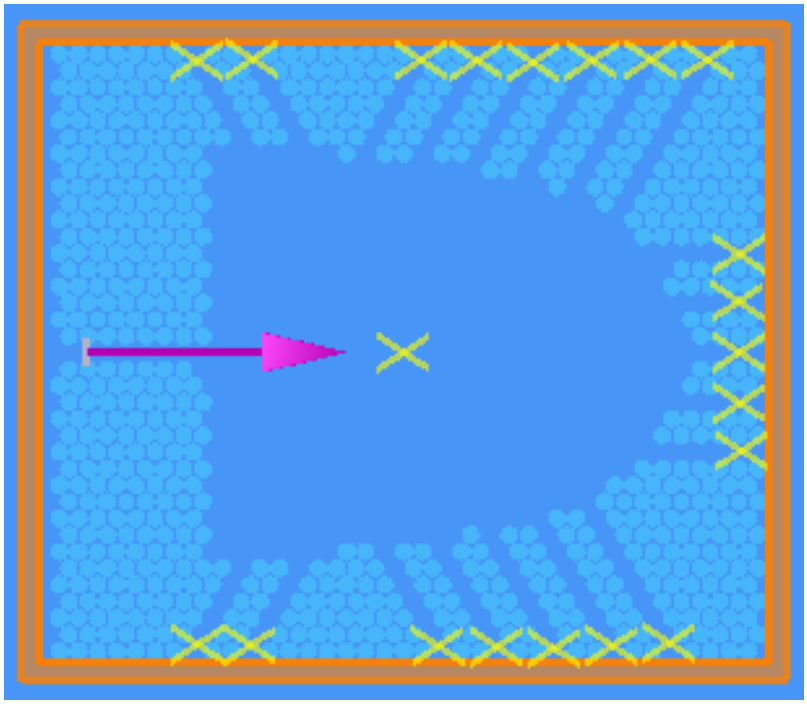

Figure 4.0.1: Optical Resonant Cavity carved out of a Ge-on-Si PC wafer.

Table 4.0.1: Design parameters of the optical resonant cavity.

Parameters	Value
Total length of wafer (μm)	43.2
Total width of wafer (μm)	37.1
Thickness (μm)	1.2
Length of cavity at longest point (μm)	30
Width of cavity at widest point(μm)	25
Length of cavity at shortest point (μm)	7.6
Width of cavity at finest point(μm)	3.55
Germanium substrate thickness (μm)	0.2
Silicon cladding thickness (μm)	0.8
Refractive index of air holes	1
Refractive index of Germanium substrate	4
Refractive index of Silicon cladding	3.5
Lattice constant α (μm)	1.15
Relative radius of air holes r (μm)	0.55
r/a	0.475

The optical resonant cavity has one input and 20 outputs, and the FDTD boundary conditions are symmetric on the x, y and z planes. Looking at Figure 4.0.1 it is clear that the input and outputs are effectively optical waveguides, guiding the signal in and out of the structure. As discussed previously, using a large number of outputs provides the signal, and the reflections that are generated, many paths to escape. Simulating the many paths that could be taken in a city environment. The yellow ‘x’ shapes in Figure 4.0.1 are monitors used to capture the result at each output, and one to capture the initial gaussian input pulse. The same impulse response testing as used with the optical waveguide was used here to acquire the system impulse response. The input signal operates in single mode and has a bandwidth of 75THz, centred at 112.5THz. The number of grid cells for the FDTD simulation were 1759, 1494 and 80 in the x, y and z directions, respectively. Figure 4.0.2 shows the gaussian input pulse and the respective output signals in the time domain at four arbitrary output ports after the FDTD simulation, where the optical cavity was excited through impulse response testing. Figure 4.03 shows these waveforms in the frequency domain. Each output signal is different for the same input signal depending on how it has been mixed and attenuated in the cavity.

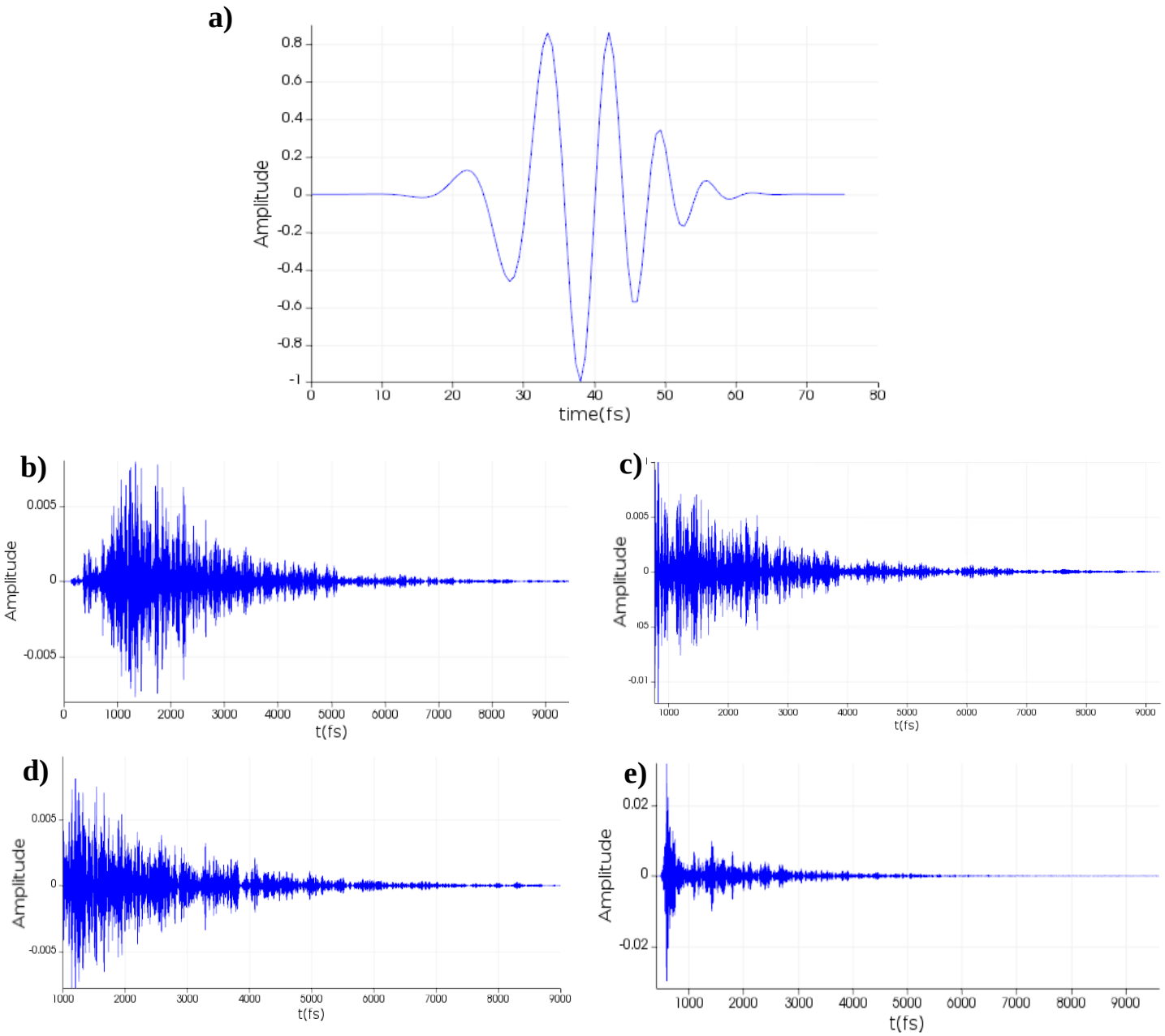

Figure 4.0.2: (a) The Gaussian input pulse signal in the time domain. (b)(c)(d)(e) System impulse response in the time domain at output ports 5,10,13 and 20 respectively.

Figure 4.0.3 clearly shows the frequency is centred at 112.5THz with a bandwidth of 75THz, as discussed previously. Furthermore, these results confirm that the reservoir is confining and mixing the desired frequencies.

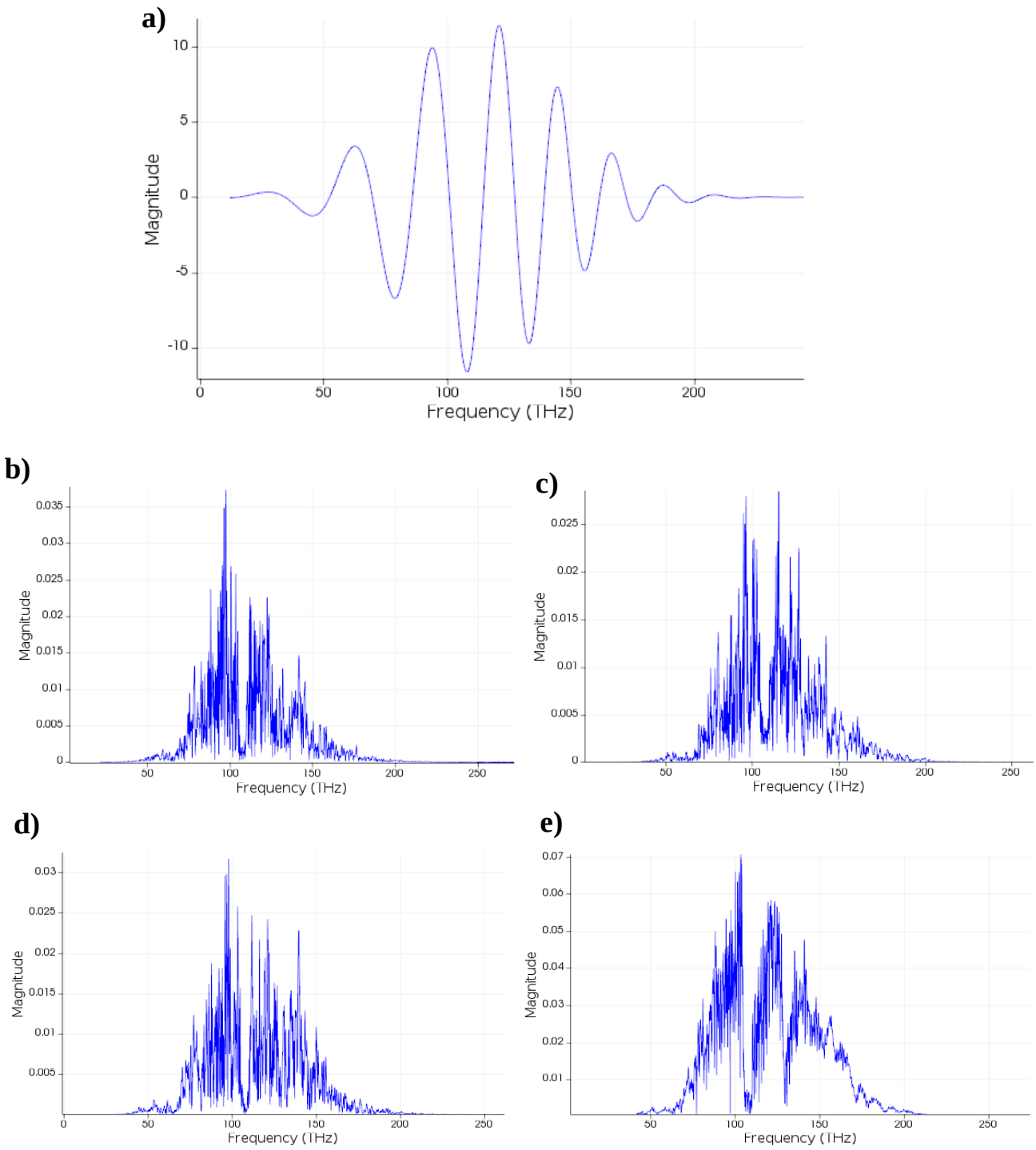

Figure 4.0.3: (a) The Gaussian input pulse signal in the frequency domain. (b)(c)(d)(e) System impulse response in the frequency domain at output ports 5,10,13 and 20 respectively.

The response data from the impulse testing was extracted from Lumerical and stored in files for use in generating the training data for the RC. Each file contained the time index and signal data captured at the input from the Gaussian impulse and all twenty output ports of the resonant cavity.

5.0 Reservoir Computer Training

The training method for a reservoir computer differs from traditional ANN training methods in a significant way. When training a reservoir computer there is a distinction made between the untrained dynamic reservoir (optical resonant cavity in this case) and a (usually) linear trained readout layer that produces the desired output, as illustrated in Figure 1.3.2 previously. During the standard RC training process, the reservoir is activated by a number of time varying input signals, represented by the column vector $\boldsymbol{u}(n) \in \mathbb{R}^{N_u}$. The update equation for neuron activation state $\boldsymbol{x}$ is a conventional ANN neuron equation that mimics a biological neural network,

$$\boldsymbol{x}(n) = (1 - \sigma) * \boldsymbol{x}(n-1) + \sigma\left[f\left(\boldsymbol{W}^{In}\boldsymbol{u}(n) + \boldsymbol{W}^{kernel}\boldsymbol{x}(n-1)\right)\right], n \geq 0 \qquad (5.0.1)$$

where n is a discrete time-step, f depicts the neuron activation function describing the type of neuron (linear nodes, threshold logic gates etc). Whilst it has been previously stated that the use of linear functions provides the best memory capacity, there exists no definitive rule for the most suitable choice of activation function [26]. $\sigma \in (0,1]$ is a leaking constant, however this model is frequently used without leaky integration by setting $\sigma = 1$. The weight of the input is described by $\boldsymbol{W}^{In} \in \mathbb{R}^{N_x * N_u}$ and the weight of the neuron kernel is $\boldsymbol{W}^{kernel} \in \mathbb{R}^{N_x * N_x}$. The notation $\mathbb{R}^{i*j}$ depicts a matrix of all real values with i rows and j columns. In the next step the neuron activation states are fed as inputs into the RC read-out layer. The output $\boldsymbol{y}(n) \in \mathbb{R}^{N_y}$ of which is given by equation 5.0.2.

$$\boldsymbol{y}(n) = \boldsymbol{W}^{out}\left[\boldsymbol{u}(n); \boldsymbol{x}(n); \boldsymbol{y}(n - T_{delay})\right] \qquad (5.0.2)$$

where the weighting of the read-out layer is notated by $w^{out} \in \mathbb{R}^{N_y * (N_u + N_x + N_y)}$ and T_{delay} is the delayed output signal fed back to the RC read-out layer. It will be seen in Section 6.1 how the value of T_{delay} has the potential to significantly affect the performance of the RC system. During training the vectors in $\boldsymbol{y}(n)$ and the desired teacher targets $\boldsymbol{y}_{teacher}(n)$ are stored into matrices $\mathbf{X}$ and $\mathbf{Y}_{\text{teacher}}$, respectively. Both contain a column for each training time-step n. The weights $\boldsymbol{W}^{out}$ of the neurons in the read-out layer are computed using ridge regression where I represents the identity matrix and γ denotes the regularisation factor, see Equation 5.0.3. There are many variations in the equations that are used for this, however, Equation 5.0.3 is highly recommended because the inclusion of γ improves the numerical stability and mitigates noise and overfitting [72].

$$\boldsymbol{W}^{out} = \boldsymbol{Y}_{teacher} * \boldsymbol{X}^T(\boldsymbol{X} * \boldsymbol{X}^T + \gamma^2 * I)^{-1} \qquad (5.0.3)$$

Figure 5.0.1 demonstrates the setup of the read-out layer and the relationship between the neuron activation states $\boldsymbol{x}$, their weightings $\boldsymbol{W}^{out}$, the delayed feedback signal $\boldsymbol{y}_{teacher}$ and the produced output signal $\boldsymbol{y}$, adapted from [26]. Note that the training of a reservoir computer only depends on the neuron activation states $\boldsymbol{x}$ (i.e. the reservoir outputs) and the delayed feedback output signal $\boldsymbol{y}_{teacher}$, that is the same signal as the original input signal to the reservoir.

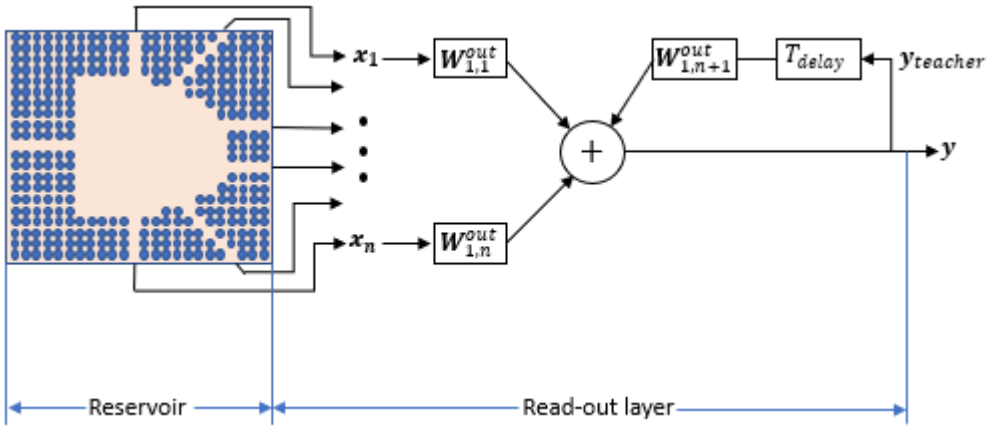

Figure 5.0.1: Reservoir Computer read-out layer training.

After training, to ascertain the accuracy and the % error of the system, the Normalised Mean Square Error (NMSE) is calculated using the desired output $\boldsymbol{y}_{teacher}$ and the actual output $\boldsymbol{y}$, as shown in Equation 5.0.4.

$$NMSE = \frac{(\boldsymbol{y} - \boldsymbol{y}_{teacher})^2}{\boldsymbol{y}_{teacher}^2} \tag{5.0.4}$$

The RC method of training, adapted from [26], can be summarised as follows:

1. Initialise $\boldsymbol{W}^{In}$ and $\boldsymbol{W}^{kernel}$ by generating a large, non-linear random reservoir.
2. Run the RC using the training input $\boldsymbol{u}(n)$ and harvest the corresponding reservoir neuron activation states $\boldsymbol{x}(n)$ using (5.0.1).
3. Compute the linear read-out weights $\boldsymbol{W}^{out}$.
4. Calculate NMSE to find %error of the system.
5. Use the trained network on new input data $\boldsymbol{u}(n)$ by computing $\boldsymbol{y}(n)$ after employing the trained output weights $\boldsymbol{W}^{out}$.

For this teaching approach to work, the reservoir must possess the *echo state* property, which is essentially a fading memory of the input, or in this project, reflections of the original input signal.

5.1 Generation of training data

To teach the RC read-out layer the training data must first be generated. Whilst this data could be generated using Lumerical, the computational resources required to generate the large dataset would have been impractical. Thus, MATLAB was used to achieve this. There were two signals that needed generating for the operation of the RC. These were the modulated input signal $im(t)$ to the reservoir that would in reality come from the use of MWP as discussed in Section 2.0, and the modulated output signal that is the actual output of the RC reservoir $om(t)$ (i.e. the neuron activation states $\boldsymbol{x}$) to feed into the RC read-out layer. $im(t)$ is also used as $\boldsymbol{y}_{teacher}$ since the signal the read-out layer should output is the reconstructed input signal. $om(t)$ is effectively a function of $im(t)$ and the normalised impulse response of the RC reservoir.

The modulated optical input signal was first generated, using a MATLAB function script that took the time period, number of points, carrier frequency, information bitstream and the information bitrate as arguments. These arguments were all passed to the function using a main script for generating all of the training data, shown in Appendix E. Figure 5.1.1 shows a snapshot of this function, see Appendix F for the full function script for generating the modulated input signal.

```
function [modulatedInputSignal] = generateModulatedInputSignal(dt, numOfPoints, carrierAmp, carrierFreq, informationBit, informationBitrate)
time = (0:numOfPoints-1)*dt; %time = vector from 0 to number of points-1 *dt

%mask
informationDuration = 1/informationBitrate; % 1ms
infoNumOfPoints = round(informationDuration/dt);%num of points per bit
mask = zeros(size(time));%Default mask values are zero

for eachBit = 1: length(informationBit) %set value of mask to envelope carrier signal

    if(informationBit(eachBit) == 1)
        lowerBound = ((eachBit-1)*infoNumOfPoints)+1;
        upperBound = (eachBit*infoNumOfPoints);
        if(upperBound > length(time))
            error("Information stream too long");
        end
        mask(lowerBound:upperBound) = 1;
    end
end

%Generate carrier signal
carrier = carrierAmp*sin((2*pi*carrierFreq)*time);
```

Figure 5.1.1: Snapshot of generateModulatedInputSignal MATLAB function.

This script generated a carrier signal of frequency = 150THz that modulated a digital bitstream representing a digital RF signal, up-converting it to the optical frequency required. In this case, Amplitude Shift Keying (ASK) modulation was used for its simplicity and ease for encoding and decoding data. In this method of modulation, the phase and frequency of the carrier signal is kept constant and only the amplitude is varied, as can be seen in Figure 5.1.3. Figure 5.1.2 shows the carrier signal and input bitstream waveforms that were generated in the time domain. Figure 5.1.3 shows the resulting ASK modulated input signal for the RC reservoir in the time domain that would in reality be generated using MWP.

a) **b)**

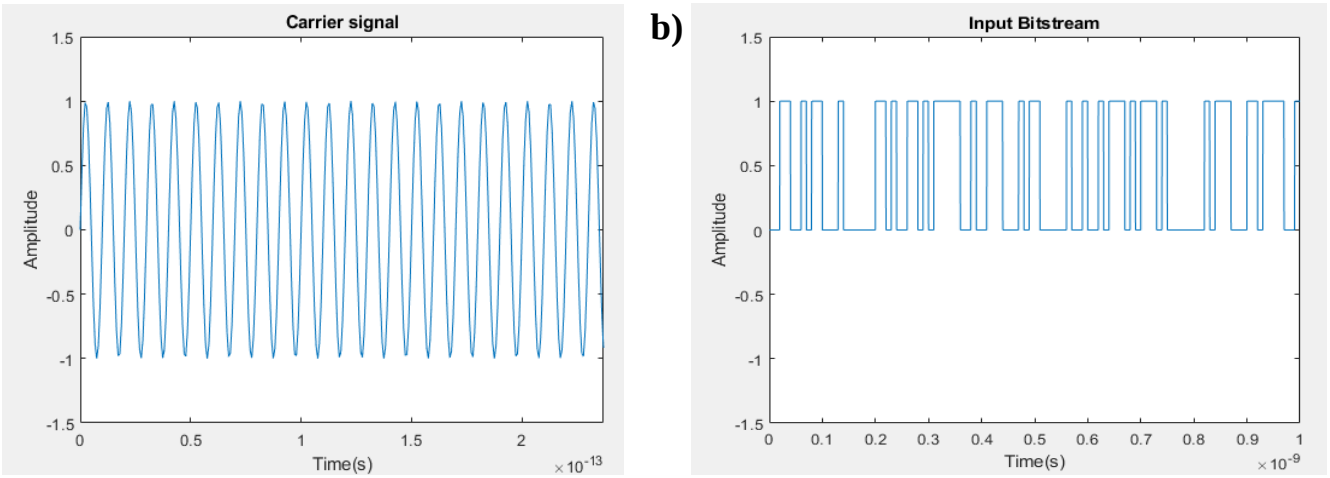

Figure 5.1.2: (a) 150THz carrier signal and (b) Digital Bitstream signal.

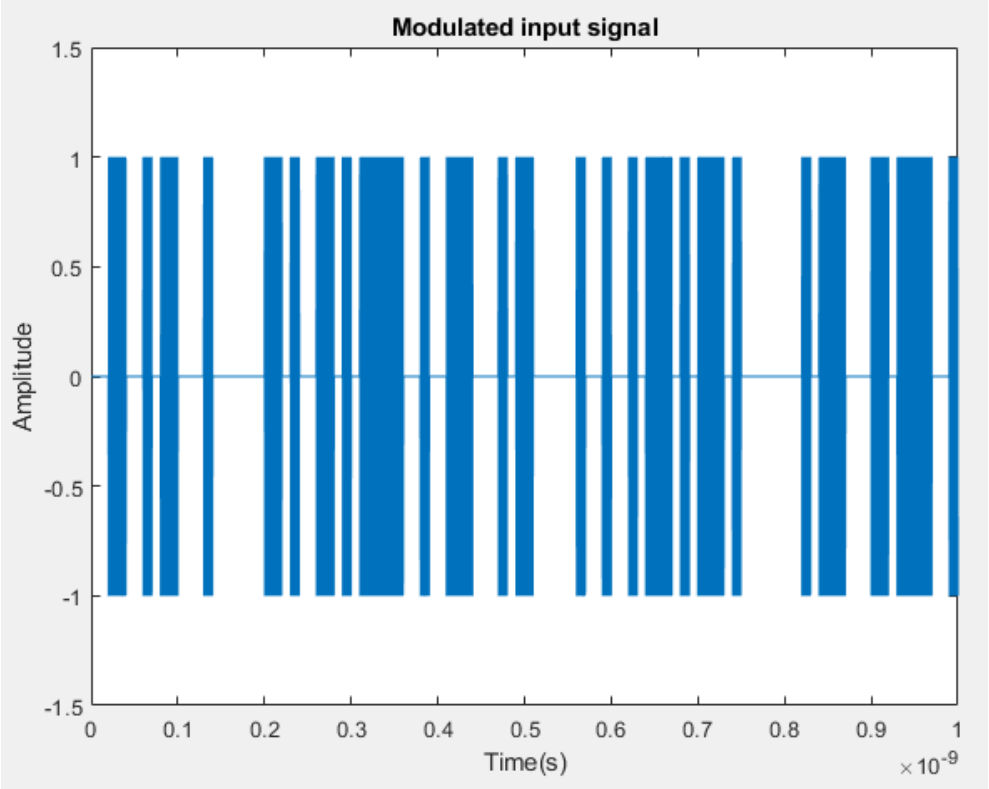

Figure 5.1.3: ASK modulated input signal $im(t)$ for RC system mimicking the output of the MWP system.

Once the modulated input signal $im(t)$ had been obtained the modulated output signal $om(t)$ could be generated. To acquire the desired output modulated signal, three other signals were required. These were: the signals from all twenty of the outputs of the resonant cavity $r_{1,\cdots,20}(t)$, the original gaussian impulse source signal $i(t)$ from the impulse response testing undertaken in section 4.0 and the modulated input signal generated previously $im(t)$. Firstly, the normalised system response of the reservoir at each output was needed. Equation 5.1.1 was used to calculate this.

$$S_{1,\cdots,20}(f) = \frac{r_{1,\cdots,20}(f)}{i(f)} \tag{5.1.1}$$

where $S_{1,\cdots,20}(f)$ denotes the normalised system response of the resonant cavity at all twenty outputs in the frequency domain. $r_{1,\cdots,20}(f)$ denotes the impulse response of all twenty output signals of the resonant cavity in the frequency domain and $i(f)$ denotes the original gaussian impulse signal in the frequency domain. Before this calculation could take place, $i(t)$ needed to be interpolated and padded with zeros as the impulse response data $r_{1,\cdots,20}(t)$ contained far more data points. This is because during the FDTD simulation, Lumerical stops monitoring the Gaussian input once it decays to zero, whilst the output signals are still reflecting within, and have not entirely exited, the cavity. Thus, in order to perform element wise division, as in Equation 5.1.1, both vectors required the same quantity of data. Figure 5.1.4 shows the impulse response of the cavity at output one (r_1) plotted against the gaussian impulse signal $i(t)$ in the time domain before interpolation and zero padding. Figure 5.1.5 shows the impulse response of the cavity at output one (r_1) plotted against the gaussian impulse signal $i(t)$ in the time domain after interpolation and zero padding to illustrate the need to do this.

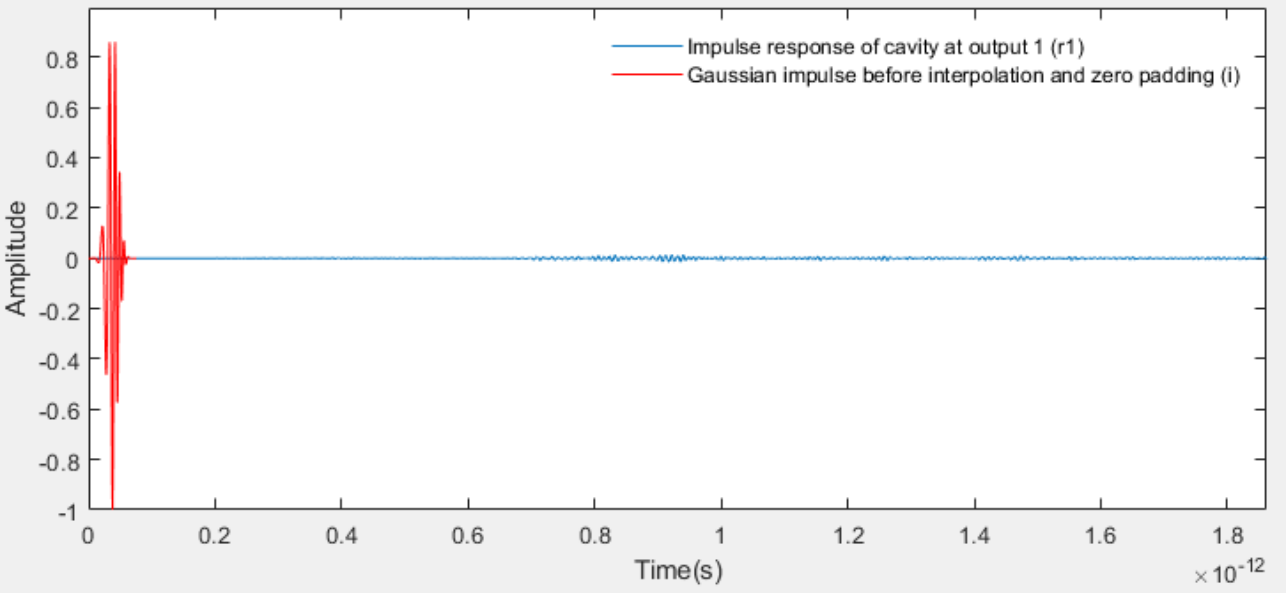

Figure 5.1.4: Gaussian impulse signal before interpolation and zero padding (i) against impulse response of cavity output 1 (r_1).

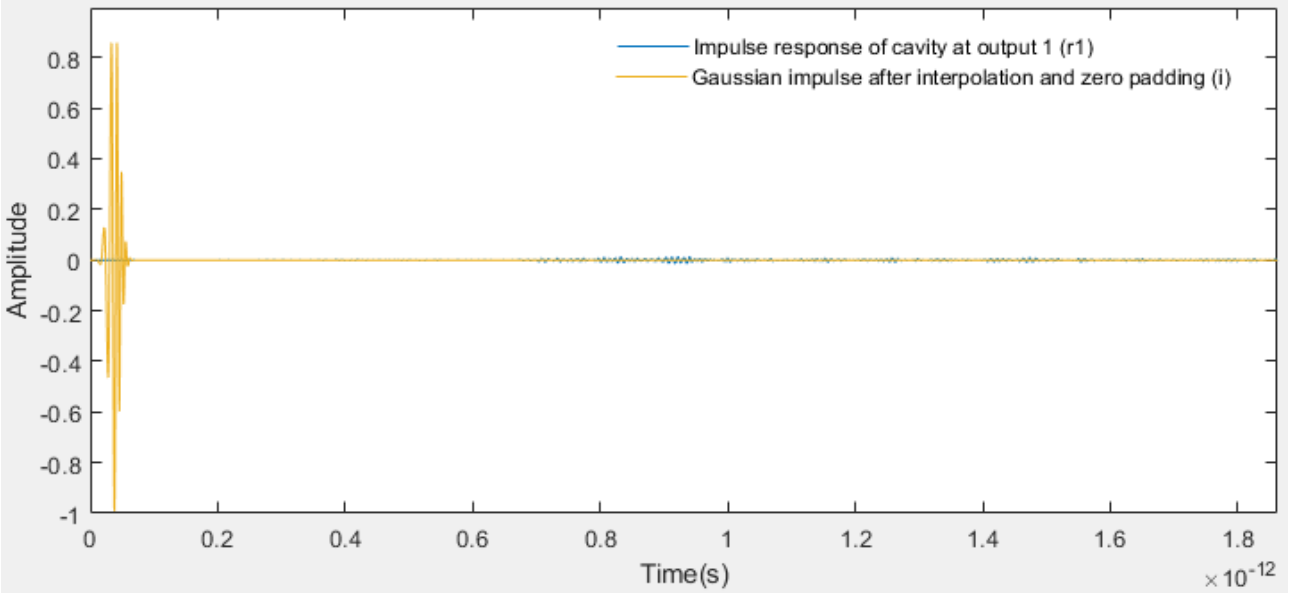

Figure 5.1.5: Gaussian impulse signal after interpolation and zero padding (i) against impulse response of cavity output 1 (r_1).

Once $i(t)$ had been interpolated and zero padded, in order to use Equation 5.1.1 to calculate the normalised system response of the cavity, both $i(t)$ and $r_{1,\cdots,20}(t)$ were required in the frequency domain. Figure 5.1.6 shows a snapshot of how this was done using FFT in a MATLAB function for generating the output modulated signal. This function took $r_{1,\cdots,20}(t)$, $i(t)$ and $im(t)$ as arguments and returned the modulated output signal in the time domain $om(t)$. See Appendix G for the full function script.

```
function [modulatedOutputSigInTimeDomain] = generateModulatedOutputSignal(rInTimeDomain, iInTimeDomain, imInTimeDomain)
%r = vector of resonant cavity outputs after gaussian pulse is injected
%i = Original Gaussian impulse source signal interpolated and padded with zeros
%im = Modulated input signal

%FT of rInTimeDomain to get rInFrequencyDomain
rInFreqDomain = fft(rInTimeDomain);

%FT of iInTimeDomain to get iInFrequencyDomain
iInFreqDomain = fft(iInTimeDomain);

%Get normalised resonant cavity system response
normalisedResponseInFreqDomain = rInFreqDomain ./ iInFreqDomain;
```

Figure 5.1.6: MATLAB code to calculate the normalised system response of the resonant cavity

With the normalised system response at each output of the cavity, the modulated output signal in the frequency domain at all twenty outputs of the cavity was calculated using Equation 5.1.2.

$$om_{1,\cdots,20}(f) = S_{1,\cdots,20}(f) * im(f) \tag{5.1.2}$$

Then using an inverse FFT function in MATLAB $om_{1,\cdots,20}(t)$ was generated. The modulated output signal that appears at output 1 of the reservoir $om_1(t)$ is plotted in Figure 5.1.7 for reference.

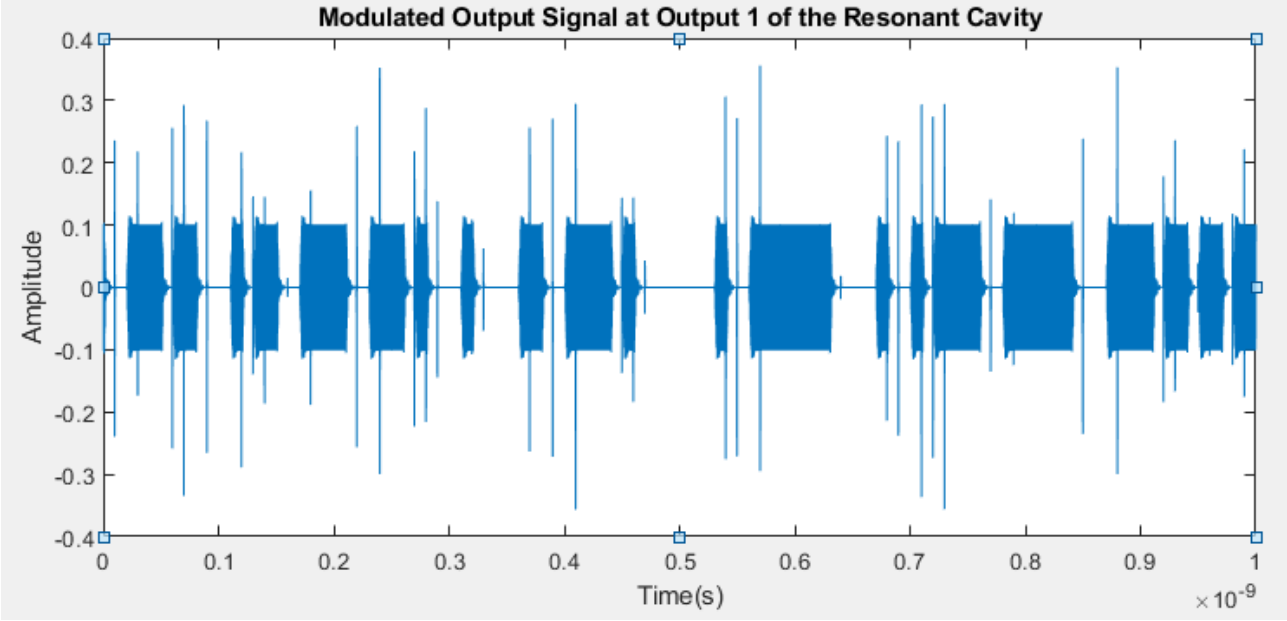

Figure 5.1.7: Modulated Output Signal at Output 1 of the reservoir (om1).

In Figure 5.1.7, and indeed for all reservoir output signals $om_{1,\cdots,20}(t)$ the data points are very fine and thus the RC read-out layer would have a difficult time detecting and differentiating between high (digital 1) and low (digital 0) bits. Thus, the MATLAB *abs* and *Hilbert transform* functions were used to pre-process $om_{1,\cdots,20}(t)$ before being passed into the RC read-out layer. The purpose

of which was to generate an envelope of the signal that more clearly differentiates between digital high and low and to shift the amplitude range of the signals to between 0-1. Figure 5.1.8 shows the waveform for $om_1(t)$ after this pre-processing and the ideal $\boldsymbol{y}_{teacher}$ that is desired at the output of the RC read-out layer.

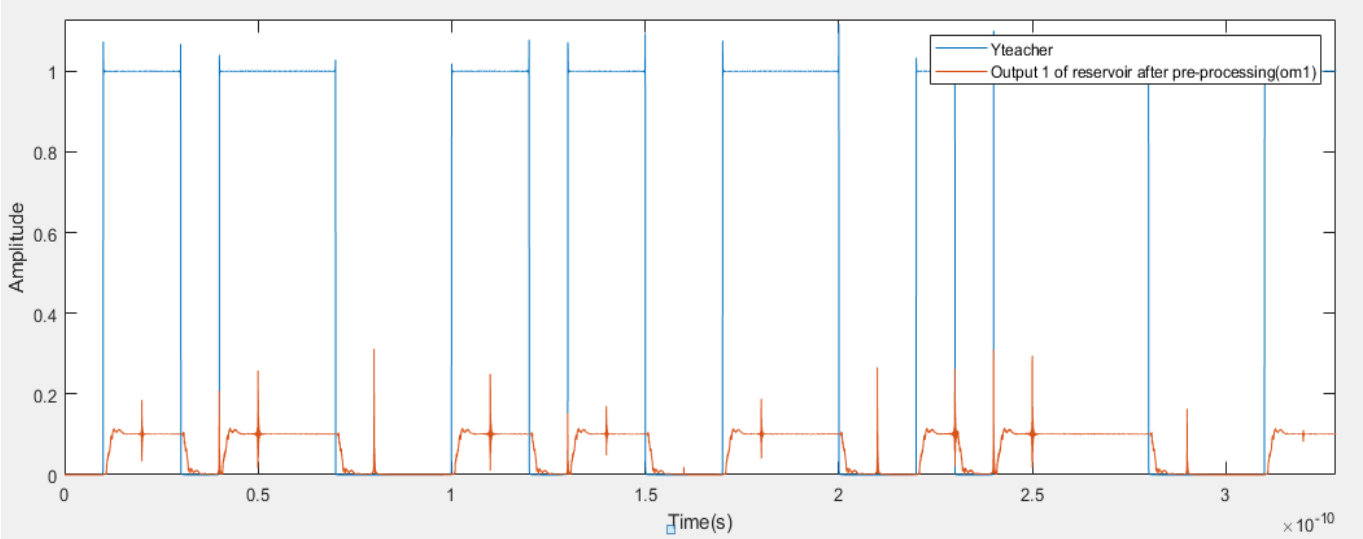

Figure 5.1.8: Modulated Output Signal envelope at Output 1 of the reservoir (om1) and $\boldsymbol{y}_{teacher}$ after pre-processing.

The overshooting that appears in Figure 5.1.8 is due to the Gibbs phenomenon that occurs during the signal processing discussed previously. Once $\boldsymbol{y}_{teacher}$ and $om_{1,\cdots,20}$ had been obtained, in order to pass the data to the RC read-out layer all signals $om_{1,\cdots,20}$ needed to be combined into one array of 20 signals, as illustrated in Figure 5.1.9.

```
%Combine modulated input signals to pass into RC read-out layer
observed_signal_envelope_combined = [observed_signal_envelope_1; ...    %Envelope of om1
                                     observed_signal_envelope_2; ...
                                     observed_signal_envelope_3; ...
                                     observed_signal_envelope_4; ...
                                     observed_signal_envelope_5;...
                                     observed_signal_envelope_6;...
                                     observed_signal_envelope_7;...
                                     observed_signal_envelope_8;...
                                     observed_signal_envelope_9;...
                                     observed_signal_envelope_10;...
                                     observed_signal_envelope_11;...
                                     observed_signal_envelope_12;...
                                     observed_signal_envelope_13;...
                                     observed_signal_envelope_14;...
                                     observed_signal_envelope_15;...
                                     observed_signal_envelope_16;...
                                     observed_signal_envelope_17;...
                                     observed_signal_envelope_18;...
                                     observed_signal_envelope_19;...
                                     observed_signal_envelope_20; ];
```

Figure 5.1.9: Create array containing all reservoir activation states/output signals om_{1-20}.

At this point the data could be passed into the RC read-out layer for training. However, first the RC read-out layer algorithm was benchmarked using the 10th order Nonlinear Auto Regressive Moving Average (NARMA10) task to verify its correct functionality.

6.0 Evaluation of the RC performance

NARMA10 is a discrete-time temporal task with 10th-order time lag that is commonly utilised for benchmarking the performance of RC's and RNNs [73]. In particular it is often used for the evaluation of memory capacity and computational power where a random input $u(n)$, drawn from a uniform distribution over the interval [0,0.05], is injected into the RC system [73]. Equation 6.0.1 defines the output from the RC that is targeted during the NARMA 10 benchmark task.

$$y(n+1) = 0.3y(n) + 0.05y(n)\left[\sum_{i=0}^{9} y(n-i)\right] + 1.5u(n-9) * u(n) + 0.1 \quad (6.0.1)$$

In this task the system is trained one hundred times over 100 time-steps, with its performance being measured using the NMSE, as in Equation 5.0.4. It should be noted that a reservoir that carries out no computation, i.e. produces a time independent output $y(n) = const$ has an NMSE result equal to 1 [4]. An NMSE results of 0 represents the perfect reconstruction of the original signal, however this is almost unobtainable. For reference, NMSE scores of reservoir computers are typically reported to be generally in the 0.01 – 0.5 range (1% - 50% error) [74, 75, 76]. Figure 6.0.1 shows the results of the NARMA 10 benchmark task where the input signal $u(n)$ is randomly generated as described above and the targeted output signal to be reconstructed by the RC system is described by Equation 6.0.1.

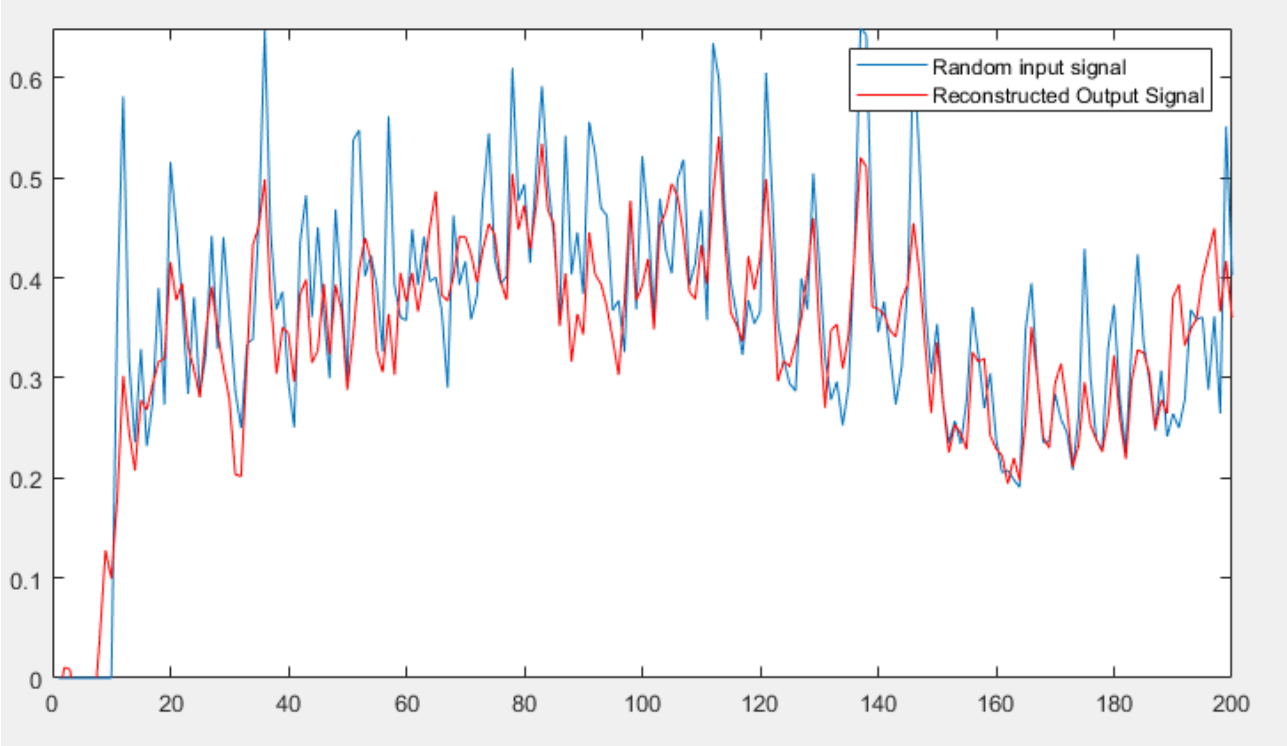

Figure 6.0.1: NARMA 10 benchmark result for the RC algorithm.

During this test, the RC algorithm reconstructed the original random input with an NMSE score of 0.028, giving an accuracy of 97.2% which corresponds to an error of 2.8%. Thus, the RC algorithm performed well and was able to reconstruct the input signal with a high degree of accuracy. Once the RC had been benchmarked and verified that it was functional, it could be trained. The training of the RC read-out layer is handled in MATLAB where the modulated output signals/neuron activation states from the reservoir $om_{1,\cdots,20}$ are stored in matrix $\mathbf{X}$ to be used as the input to the read-out layer, and the teaching/target data $\boldsymbol{y}_{teacher}$ is stored in matrix $\mathbf{Y}_{\text{teacher}}$. The RC system is trained as discussed in Section 5.0.

6.1 Reconstructing Original Signal

To operate the RC read-out layer, the necessary information is passed to the MATLAB implementation using the code displayed in Figure 6.1.1.

```
myRC = RVCObj('is_reservoir_enabled', false,...          %False as the reservoir has been created externally
              'is_input_weighted', true,...
              'is_readout_has_direct_access_to_input', true,...
              'is_readout_has_constant_pump_offset',true,...
              'input_signal',zeros(size(information_bit)),...
              'neuron_activation_signal',observed_signal_envelope_combined,...  %oml-20 reservoir activation states
              'teacher_signal',information_bit,...      %Y teacher signal
              'feedback_delay_timestep',500); %Feedback delay time-step (500dt)
```

Figure 6.1.1: Passing necessary information to the RC read-out layer MATLAB implementation.

The RC read-out layer required optimisation in order to better reconstruct the original modulated input signal to the reservoir. This was achieved by varying the feedback delay timestep T_{delay}. Table 6.1.1 showcases how varying the feedback delay timestep can affect the NMSE result and thus accuracy when reconstructing the original modulated reservoir input signal. Figure 6.1.2 shows a plot of feedback delay time-step vs accuracy of the reconstructed signal, generated using the React.js script in Appendix D. As can be seen, the most accurate NMSE result of 0.044, corresponding to an accuracy rate of 95.6%, was found when T_{delay} was set to 300dt, where dt = 7.47e-16.

Table 6.1.1: Accuracy of reconstructed signal for varying feedback delay time-step values.

Feedback Delay Time-Step (T_{delay})	NMSE	Accuracy of reconstructed signal (%)
0dt	1	0
200dt	0.88	12
250dt	0.23	77
300dt	0.044	95.6
350dt	0.060	94
400dt	0.055	94.5
450dt	0.051	94.9
500dt	0.05	95

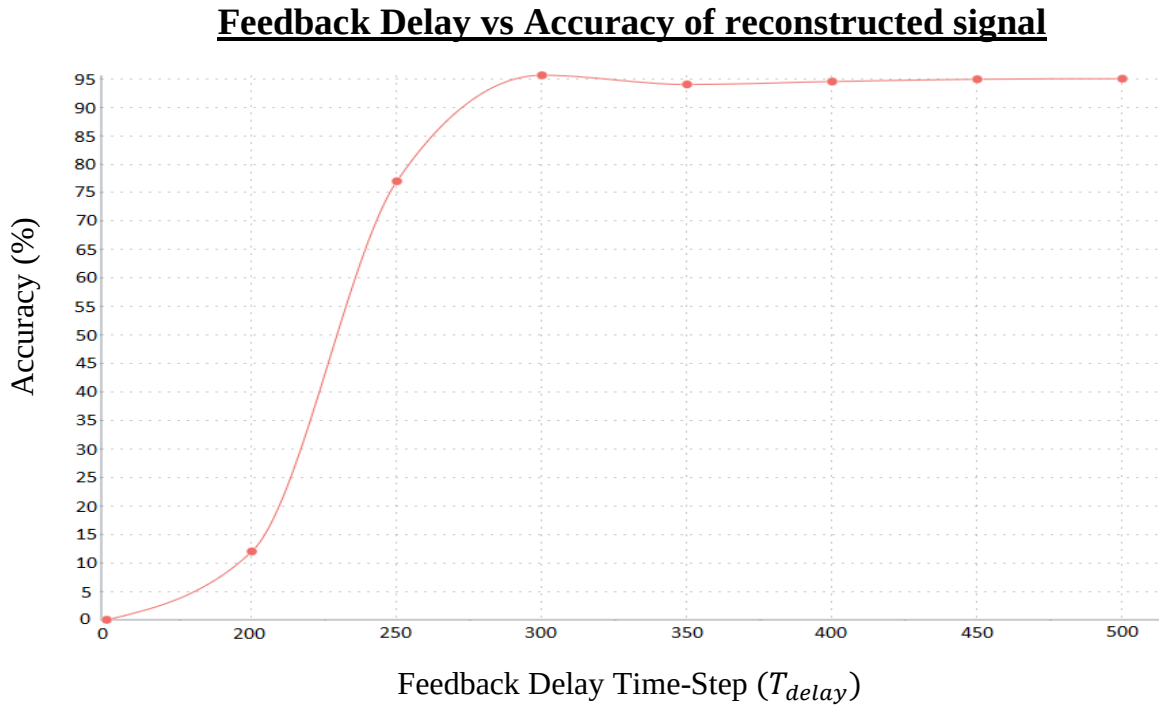

Figure 6.1.2: Value of feedback delay vs accuracy of reconstructed signal.

Figure 6.1.3 showcases the targeted output waveform (i.e. $\boldsymbol{y}_{teacher}$) and the reconstructed read-out layer output signal $\boldsymbol{y}(n)$ for cases where T_{delay} was 0dt and 200dt to illustrate the differences between the produced signals when the accuracy is poor. Figure 6.1.4 showcases the case when T_{delay} was set to 300dt and the original signal was reproduced with the optimum accuracy of 95.6%.

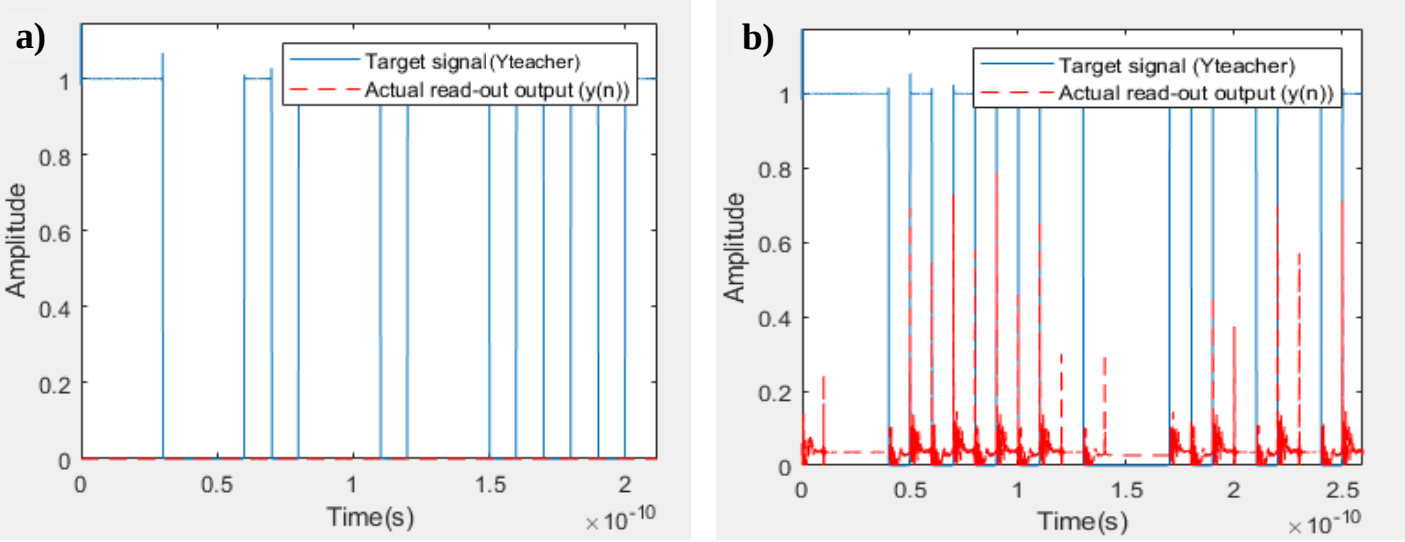

Figure 6.1.3: Reconstructed signal for (a) NMSE = 1 and (b) NMSE = 0.88.

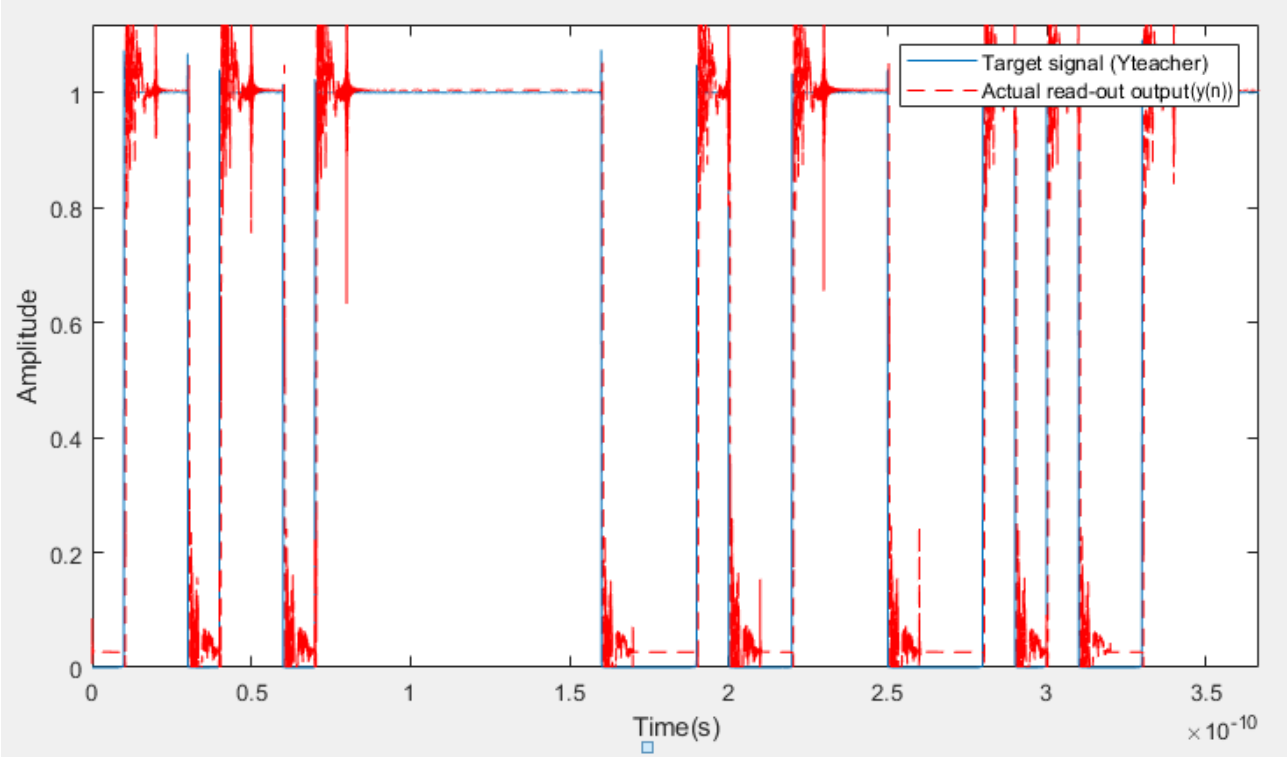

Figure 6.1.4: Successfully reconstructed reservoir input signal where NMSE = 0.044.

Figure 6.1.4 shows the teacher signal (the desired output) vs the actual output signal $\boldsymbol{y}(n)$ reconstructed with the optimum accuracy of 95.6%. When T_{delay} is small the shape of the resulting output signal tends to follow the shape of the target signal although the amplitude tends to be small. When T_{delay} is large the resulting output signal starts to resemble the target signal to a much higher degree, thus clearly the success of the RC system is dependent on a properly chosen feedback delay. However, this parameter is not easily calculated analytically and thus the trial and error approach is commonly used with this type of system.

The RC system has successfully re-constructed the original digital input signal to a high degree. As discussed previously it is almost, if not completely, impossible to reconstruct the signal with 100% accuracy (NMSE = 0) because no system is completely lossless. Thus, this RC system (and the MWP-RC system as a whole) has shown itself to be a good potential candidate for use in optical telecommunications applications as it has successfully decoded and reconstructed the original high bandwidth digital signal that was transmitted through a chaotic resonant cavity, simulating a dense urban environment, with low losses.

7.0 Conclusions

This thesis presents the theoretical design of an MWP-RC telecommunications framework for optical signal processing. Through harnessing the inherent advantages of photonic operation, this system offers a solution to ever-increasing demands for higher bandwidth and faster communications signal processing and transmission. An all-optical reservoir computer, based on the RC model originally independently proposed by Herbert Jaeger and Wolfgang Maass, is demonstrated using a photonic crystal cavity structure, similar to that proposed originally by Floris Laporte et al, as the reservoir and a MATLAB implementation of the read-out layer. A principle concept within reservoir computing, the use of an untrained neuron kernel/reservoir, is demonstrated with the reservoir being exhibited theoretically as an intentional defect cavity in a photonic crystal structure on a Ge-on-Si chip. The Ge-on-Si photonic crystal cavity has been shown to be a viable solution for real world applications of an RC reservoir. Furthermore, results show that the RC system was able to successfully reconstruct a digital input signal passed through the reservoir, acting to simulate an urban environment, with an optimum accuracy of 95.6%. Hence this project has confirmed the hypothesis that microwave-photonics in conjunction with artificial neural network technology could be used to increase wireless communications-capacity for information and communications processing. Offering a new approach to physical infrastructure and technology.

It should however be noted that physical optical hardware implementations of RC systems, and indeed RC techniques in general, are still in their infancy at this time. Furthermore, the read-out layer in this project has only been trained and optimised for the specific conditions of the designed reservoir. Thus, the system may not perform well if the operating conditions were to be changed such as using different values of feedback time-delay or different materials or geometric shapes for the photonic crystal Ge-on-Si chip. This issue could be solved however by training the system over a wide range of data for varying conditions.

7.1 Design of the Optical Waveguide

The optical waveguide has been successfully designed for the guiding of an optical signal into the input and out of the outputs of the optical reservoir. This section of the report covers photonic crystal theory and the design of a photonic crystal structure (deliverable 1). With the optical waveguide being generated as a line defect in the photonic crystal structure. The optical waveguide is initially aimed at single mode operation with a photonic band gap between 75THz – 150THz in the infrared frequency range. Impulse response testing using the 3D-FDTD method demonstrated successful single mode wave propagation in the waveguide with the widest bandgap of 38THz being gained between 97-135THz when r/a = 0.475, a = 1.15μm and r = 0.55μm (deliverable 2). However, this photonic bandgap could be optimised with further experimentation in future work to allow for the capture of a wider range of frequencies of optical signals in the waveguide and thus the reservoir.

7.2 Design of the Optical Resonant Cavity

As the optical waveguide is generated using an intentional line defect in a photonic crystal structure, the RC reservoir (deliverable 3) is designed as an asymmetric ‘D’ shaped resonant cavity by generating an intentional cavity defect in the same photonic crystal structure. A structure of this shape allows the reservoir to take full advantage of the chaotic ray dynamics inherent to it and thus the input signal undergoes complex manipulation and mixing in the cavity, simulating signal transmission and attenutation through a dense urban environment. Impulse response testing of the reservoir is undertaken and the results at four arbitrary outputs are shown, illustrating how differently output signals are affected depending on the route taken to the output. The results also confirm that the reservoir successfully confines and mixes the desired frequencies of light.

7.3 Generation of RC Training Data

The standard training process for a reservoir computer is outlined. Due to the nature of reservoir computing, only the read-out layer required training. Firstly, the modulated input signal to the reservoir, representing the resulting signal after light modulation using MWP, is generated. This is achieved through using a 150THz carrier signal and a digital bitstream signal representing a RF signal received from a transmission source such as a mobile phone. The bitstream is modulated with the carrier using ASK modulation for simplicity and ease of encoding and decoding data. The resulting signal is the modulated input signal which is also used as $\boldsymbol{y}_{teacher}$. Next, after normalisation of the impulse responses from the reservoir the output modulated signal for each reservoir output is calculated. These are the neuron activation states $\boldsymbol{x}$. The training of a reservoir computer only depends on the neuron activation states $\boldsymbol{x}$ and the delayed feedback output signal $\boldsymbol{y}_{teacher}$. Snapshots of relevant waveforms are used to show that the training data is generated successfully.

7.4 Evaluation of the Reservoir Computer performance

The RC algorithm (deliverable 4) is benchmarked using the NARMA 10 task, a commonly utilised benchmarking task for RC's and RNN's to validate its performance. Results from the NARMA 10 benchmark task suggested the RC algorithm worked well as it was able to reconstruct a random input signal with an accuracy of 97.2%. Next, the outputs of the reservoir (neuron activation states) were applied to the read-out layer and the original digital input signal was reconstructed with an optimum accuracy of 95.6% after optimisation of the feedback time-delay. However, it is worth noting that further optimisation of the feedback time-delay may result in even greater accuracy of the system. The results gained suggested that this MWP-RC system would be a good candidate for telecommunications applications, confirming the hypothesis stated at the start of this thesis, in Section 1.2.

8.0 Personal Reflections

8.1 Project Plan

The original and revised time plans for this project are seen in Appendix A and B respectively. For the most part this project ran smoothly, and the main objectives were achieved. However, ultimately the original plan of building a CNN for benchmarking the performance of the RC against was not completed due to time limitations. As is discussed in Section 3.1, the design of a photonic crystal is effectively a trial and error process of varying r/a and other elements until desired and/or acceptable results are gained. Even though this was foreseen in the time plan with a significant portion of time dedicated to establishing the model, in the end it took longer than this to achieve acceptable results. Furthermore, the impulse response testing of the reservoir took longer than expected due to an unforeseen issue that meant the simulation, that already took a significant amount of time to run, would diverge due to instabilities in the FDTD boundary conditions and so the results would be unusable. Once this was discovered and fixed, it was decided that building a CNN was not possible in the remaining time. Thus, some modifications were made to the aims of the project. However, the lack of CNN was not a major problem because the RC was benchmarked using the NARMA 10 task instead, which is a very commonly used benchmark for ANN performance in the RC and RNN world.

8.2 Future Improvements & Direction of Research

Because photonic reservoir computing is currently still in its infancy there is plenty of scope for further experimentation and research. Firstly, the performance of the photonic reservoir computer could be benchmarked against other types of ANNs such as CNNs (as originally planned in this project) or RNNs to ascertain whether there are other types of ANNs that would be better suited to this particular application. Secondly, since the design of the photonic crystal structure used for the

reservoir was a process of trial and error, further work could be put into varying r/a and the materials used for the PC structure in order to advance the design and thus build a reservoir capable of containing a greater range of frequencies and signal data for processing. Due to time constraints on the project, the feedback time-delay of the RC read-out layer was optimised until an acceptable result was gained. In future work more time could be put into the optimisation of this parameter as there is the potential to achieve even greater signal reconstruction accuracy than has been gained here.

As discussed in Section 7.0, the read-out layer in this project has only been trained for the specific conditions of the designed reservoir and the training data is ideal. Consequently, if presented with differing operating conditions the system may not perform well. In future work the system should be trained over a wide range of ideal data and eventually using real experimental data, perhaps gained from implementation of this model using real optical circuits. Which is another potential avenue for future work and research from this project. However, as discussed previously, physical optical hardware implementations of RC systems are currently very much in their early days.

Another route for further research could be directed into the impact of varying different reservoir cavity shapes and the use of different materials besides Ge-on-Si for its structure. The idea of using a photonic crystal resonant cavity for the reservoir structure is still quite novel in itself and so there is plenty of room for expansion here. Finally, in this project the reservoir is constructed with twenty outputs to provide many exit paths for the signal, simulating the many paths the signal could take in an urban environment. If this system were to be constructed in hardware, detecting the optical signals may prove difficult, especially considering that the light pulses are in the “fs” timescale. Furthermore, the equipment required to detect this number of optical signals may prove to be expensive. Thus, if this project was to be constructed in hardware it may be worth decreasing the number of outputs of the reservoir.

Although reservoir computing is still in its infancy, the technique is going from strength to strength with an increasing number of successes being reported in a range of applications and industries as time passes. Currently hardware implementations of (particularly optical) RC systems are challenging. Though these challenges will eventually be overcome, enabling reservoir computing to provide the promise of massively parallel information processing at extremely high speed, high bandwidth and low power consumption.

8.3 COVID-19 Statement

In light of the COVID-19 pandemic sweeping the globe at the time of writing, the University of Nottingham moved to online teaching as of 16th March 2020. This meant that students were unable to meet face to face with supervisors and those who worked in labs were unable to access them anymore. However, in relation to this project it is not felt that the current situation has had much of an effect on the work undertaken. Supervisor meetings were moved online, and the lab computer used was accessed remotely. Thus, there was no alteration to objectives due to COVID-19.

9.0 References

[1] D. Livingstone, *Artificial Neural Networks*. Totowa, NJ: Humana Press, 2009.
[2] Lei, Zhang. "Artificial neural networks model design of lorenz chaotic system for eeg pattern recognition and prediction." In *2017 IEEE Life Sciences Conference (LSC)*, pp. 39-42. IEEE, 2017.
[3] James Dacombe, “An introduction to Artificial Neural Networks”, *Medium*, Oct-23-2017. [Online]. Available: https://medium.com/@jamesdacombe/an-introduction-to-artificial-neural-networks-with-example-ad459bb6941b [Accessed: 10-Oct-2019].
[4] F. Duport et al, "Fully analogue photonic reservoir computer," *Scientific Reports*, 2016.

[5] S. Pandya, "Understanding Brain, Mind and Soul: Contributions from Neurology and Neurosurgery," *US National Library of Medicine*, vol. 9.1, pp. 129-149, 2011.
[6] G. Leibniz, Dissertatio De Arte Combinatoria, Leipzig: Leipzig University, 1666.
[7] S. R. y. Caja, Texture of the Nervous System of Man and the Vertebrates, vol. I, 1904.
[8] G. Piccinini, THE FIRST COMPUTATIONAL THEORY OF MIND AND BRAIN, Kluwer Academic Publishers, 2004.
[9] M. Arbib, The Handbook of Brain Theory and Neural Networks, The MIT Press, 1995.
[10] D. Hebb, The Organization of Behaviour: A Neuropsychological Theory, Montreal: Lawrence Erlbaum Associates, 1949.
[11] A. Turing, Computing Machinery And Intelligence, vol. 59, Oxford University Press, 1950, pp. 433-460.
[12] J. Akst, "Machine, Learning, 1951," 30 April 2019. [Online]. Available: https://www.the-scientist.com/foundations/machine--learning--1951-65792. [Accessed 03 January 2020].
[13] A. Ballantyne, "Minsky's "And / Or" Theorem: A Single Perceptron's Limitations," Medium, 17 November 2017. [Online]. Available: https://alan.do/minskys-and-or-theorem-a-single-perceptron-s-limitations-490c63a02e9f. [Accessed 03 January 2020].
[14] A. Kurenkov, "A 'Brief' History of Neural Nets and Deep Learning," 24 December 2015. [Online]. Available: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/. [Accessed 05 January 2020].
[15] K. Fukushima, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, Tokyo: NHK Broadcasting Science Research Laboratories, 1980.
[16] A. Karahoca et al, Data Mining Applications in Engineering and Medicine, IntechOpen, 2012.
[17] D. Rumelhart, G. Hinton & R. Williams, Learning representations by back-propagating errors, Nature, 1986.
[18] T. Ho, Random decision forests, Montreal: IEEE, 1995.
[19] S. Hochreiter, J. Schmidhuber LONG SHORT-TERM MEMORY, Munchen: University of Munchen, 1997.
[20] H. Shah, "A.L.I.C.E.: an ACE in Digitaland," *DOAJ*, 2008.
[21] I. Goodfellow, Y. Bengio, A. Courville, Deep learning. Cambridge (EE. UU.): MIT Press, 2016.
[22] J. Suykens, J. Vandewalle, B. De Moor. Artificial Neural Networks for Modelling and Control of Non-Linear System, Springer, 1996.
[23] F. Grezes, "Reservoir Computing," New York, 2014.
[24] B. Schrauwen, J. Defour, D. Verstraeten, J. Van Campenhout. "The Introduction of Time-Scales in Reservoir Computing, Applied to Isolated Digits Recognition," *Springer Link*, vol. 4668, pp. 471-479, 2007.
[25] M. Lukosevicius, H. Jaeger, B. Schrauwen. "Reservoir Computing Trends," *Springer Link*, pp. 365-371, 16 May 2012.
[26] S. Phang, P. Sewell, A. Vukovic, T. Benson; The Optical Reservoir Computer: A New Approach to a Programmable Integrated Optics System Based on an Artificial Neural Network, in "Integrated Optics: Recent Advances and Prospects" edited by Giancarlo C. Righini and Maurizio Ferrari, due to be published January 2019 by The Institution of Engineering and Technology (IET). In-press
[27] Y. Paquot et al. "Optoelectronic Reservoir Computing," *Scientific Reports*, 27 February 2012.
[28] IEEE, "IEEE Task Force on Reservoir Computing," [Online]. Available: https://sites.google.com/view/reservoir-computing-tf/home. [Accessed 23 January 2020].
[29] G. Tanaka et al, "Recent advances in physical reservoir computing: A review," *Science Direct*, vol. 115, pp. 100-123, July 2019.
[30] F. Laporte, J. Dambre, P. Bienstman , "Reservoir computing with signal-mixing cavities," IEEE, 2017

[31] K. Vandoorne et al, "Toward optical signal processing using Photonic Reservoir Computing," *Optics Express,* vol. 16, no. 15, pp. 11182-11192, 2008.
[32] K. Vandoorne et al, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Communications,* 2014.
[33] F. Duport et al, "Fully analogue photonic reservoir computer," *Scientific Reports,* 2016.
[34] P. Buteneers et al, "Automatic detection of epileptic seizures on the intra-cranial electroencephalogram of rats using reservoir computing," *Science Direct,* vol. 53, no. 3, pp. 215-233, 2011.
[35] F. D.-L. Coarer et al, "All-Optical Reservoir Computing on a Photonic Chip Using Silicon-Based Ring Resonators," IEEE Journal of Selected Topics in Quantum Electronics, p. 99, 2018.
[36] B. Schrauwen et al, "Compact hardware liquid state machines on FPGA for real-time speech recognition.," *NCBI,* 2007.
[37] L. Larger et al, "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing," *Optics Express,* vol. 20, no. 3, pp. 3241-3249, 2012.
[38] C. Fernando, S. Sojakka, "Pattern Recognition in a Bucket," *Schemantics Scholar,* 2003.
[39] "[White paper] Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper,". [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html. [Accessed: 07-Oct-2019].
[40] Marcus, M.J., 2015. 5G and" IMT for 2020 and beyond"[Spectrum Policy and Regulatory Issues]. IEEE Wireless Communications, 22(4), pp.2-3.
[41] "Next Generation Mobile Technologies: A 5G Strategy for the UK," HM Treasury, 2017.
[42] "How photonics will topple electronics," Eindhoven University Of Technology, 29 March 2018. [Online]. Available: https://www.tue.nl/en/tue-campus/news/29-03-2018-how-photonics-will-topple-electronics/. [Accessed 10 February 2020].
[43] Pooja et al, "Advantages and Limitation of Radio over Fiber System," *International Journal of Computer Science and Mobile Computing,* vol. 4, no. 5, pp. 506-511, 2015.
[44] "[Product Datasheet] Benchtop High-Speed LiNbO3 Electro-Optic Modulator Driver". [Online]. Available: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_ID=9949#ad-image-0 [Accessed: 08-Oct-2019].
[45] "[Product Datasheet] 1.5GHz to 2.4GHz High Linearity Direct Quadrature Modulator". [Online] Available: https://www.analog.com/media/en/technical-documentation/data-sheets/5528f.pdf [Accessed: 09- Oct-2019].
[46] Yiwei Xie, Zihan Geng, Leimeng Zhuang et al, "Programmable optical processor chips: toward photonic RF filters with DSP-level flexibility and MHz-band selectivity", vol. 7, issue 2, 2017-12-07.
[47] "[Product Datasheet] BPF.24.01 RF Filter". [Online]. Available: https://www.digikey.co.uk/product- detail/en/taoglas-limited/BPF.24.01/931-1467-ND/6362804 xml [Accessed: 09-Oct-2019].
[48] D. Novak, R. Waterhouse "Commercial and Defense Applications of Microwave Photonics," LLC, Hanover.
[49] S. Iezekiel et al, "RF Engineering Meets Optoelectronics," IEEE, vol 16, issue 8, 2015.
[50] Alexander N. Tait, Thomas Ferreira de Lima, Ellen Zhou et al, "Neuromorphic photonic networks using silicon photonic weight banks", *Scientific Reports,* 07-Aug-2017.
[51] "Reaviz Data Visualization using React and D3.js," 2019. [Online]. Available: https://openbase.io/js/reaviz. [Accessed 25 February 2020].
[52] R. Tu, "Ray Optics Simulation," [Online]. Available: https://ricktu288.github.io/ray-optics/simulator/. [Accessed 25 February 2020].
[53] "Understanding Wavelengths In Fiber Optics," [Online]. Available: https://www.thefoa.org/tech/wavelength.htm. [Accessed 02 March 2020].
[54] K. Yee, "Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media," *IEEE Xplore,* vol. 14, no. 3, pp. 302-307, May 1966.

[55] W. Chen et al, "An FPGA implementation of the two-dimensional Finite-Difference Time-Domain (FDTD) algorithm," in *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, Monterey, 2004.
[56] "Finite Difference Time Domain (FDTD) solver introduction," Lumerical, [Online]. Available: https://support.lumerical.com/hc/en-us/articles/360034914633-FDTD-solver. [Accessed 05 March 2020].
[57] W. Buchanan, "Analysis of Electromagnetic Wave-Propogation using the 3D Finite-Difference Time-Domain Method with Parallel Processing," Napier University, Edinburgh, 1996.
[58] "[Software manual] Lumerical's tools enable the design of photonic components, circuits, and systems".
[59] S. K. S. a. P. Sethi, "Review on Optical Waveguides," in *Emerging Waveguide Technology*, vol. I, 01-Oct-2018.
[60] J.-M. Liu, Photonic Devices, Cambridge: Cambridge University Press, 2009.
[61] S. Johnson, J. Joanopoulos, "Introduction to Photonic Crystals: Bloch's Theorem, Band Diagrams, and Gaps (But No Defects)," MIT, Massachussets, 2003.
[62] J. Joannopoulos et al, "Photonic Crystals: Molding the Flow of Light," Princeton University Press, Princeton, 2008.
[63] L. Shen et al, "All-optical modulation and nonlinear absorption in germanium-on-silicon waveguides near the 2μm wavelength regime," Optoelectronics Research Centre, Southampton.
[64] D. Miller, "Meshing optics with applications," *Nature Photonics,* pp. 403-404, 2017.
[65] T. Trifnov, "Photonic Bandgap Analysis and Fabrication of Macroporous Silicon by Electrochemical Etching," Rovira i Virgili University, Tarragona, 2010.
[66] M. Turduev, "Photonic Crystals Engineering For Ligh tManipulation: Low Symmetry, Graded Index and Parity Time Symmetry," TOBB University of Economics and Technology, Ankara, 2015.
[67] J. Escalante, S. Skipetrov "Level spacing statistics for light in two-dimensional disordered photonic crystals," *Scientific Reports,* 2018.
[68] Y. Kalra, R. Sinha, "Photonic band gap engineering in 2D photonic crystals," *Pramana journal of physics,* vol. 67, no. 6, pp. 1155-1164, 2006.
[69] M. Dood, E. Snoeks, A. Moroz, A. Polman, "Design and optimization of 2D photonic crystal waveguides based on silicon," *Research Gate,* pp. 145-159, 2002.
[70] A. Downey, *Think DSP: Digital Signal Processing in Python*. Needham, Massachusetts: Green Tea Press, 2014.
[71] S. Bittner et al, "Suppressing spatiotemporal lasing instabilities with wave-chaotic microcavities," *Science,* vol. 361, no. 6408, pp. 1225-1231, 2018.
[72] M. Lukosevicius, H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Science Direct,* vol. 3, no. 3, pp. 127-149, 2009.
[73] A. Goudarzi et al, "A Comparative Study of Reservoir Computing for Temporal Signal Processing," 2014.
[74] M. Dale, "A substrate-independent framework to characterize reservoir computers," *Mathematical, Physical and Engineering Sciences,* 2019.
[75] F. Wyffels, B. Schrauwen "A comparative study of Reservoir Computing strategies for monthly time series prediction," Ghent University, Ghent .
[76] Y. Kuriki et al, "Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers," *The Optical Society,* vol. 26, no. 5, pp. 5777-5788, 2018.

Appendices

Appendix A: Original Project Plan

The aim of this project is to confirm the hypothesis that microwave-photonics in conjunction with artificial neural network technology can be used to increase the wireless communications-capacity for information and communications processing and thus offer a new approach to physical infrastructure and technology.

The objectives planned out to achieve this are as follows:

- Gain an understanding and appreciation of photonic technology and artificial neural network (ANN) techniques.
    - Research and gain a familiarity with ANN techniques of Reservoir Computing and Convolutional Neural Networks.
    - Research on microwave-photonic technologies.
    - Understand core principles of photonic crystals such as the implementation of intentional defects.

- Design of optical resonant cavity for reservoir computer.
    - Design a photonic crystal.
    - Implement intentional defects to create a waveguide for verification of correct function.
    - Design RC reservoir as a resonant cavity within a photonic crystal.

- Development of photonic neural network algorithms.
    - Develop a reservoir computing algorithm using high-performance photonic simulation software (Lumerical) and MATLAB.
    - Develop a simulated Convolutional Neural Network using MATLAB.

- Validation and benchmarking of ANN codes.
    - Solve a header recognition problem using both algorithms in order to ascertain which performs better and develop an understanding of proper algorithm optimisation.
    - Benchmark the performance of a reservoir computing algorithm against a traditional Convolutional Neural Network to ascertain which approach processes large volumes of input data most effectively.

The proposed deliverables for this project are given below.

- Design of the Lumerical optical resonant cavity model with simulation results.
- A working and validated MATLAB reservoir computing algorithm.
- A working and validated MATLAB Convolutional Neural Network algorithm.
- Finalised thesis containing a background literature review, system design principles, methodologies used, RC and CNN simulation algorithms and test results, analysis of results and considerations for future improvement.

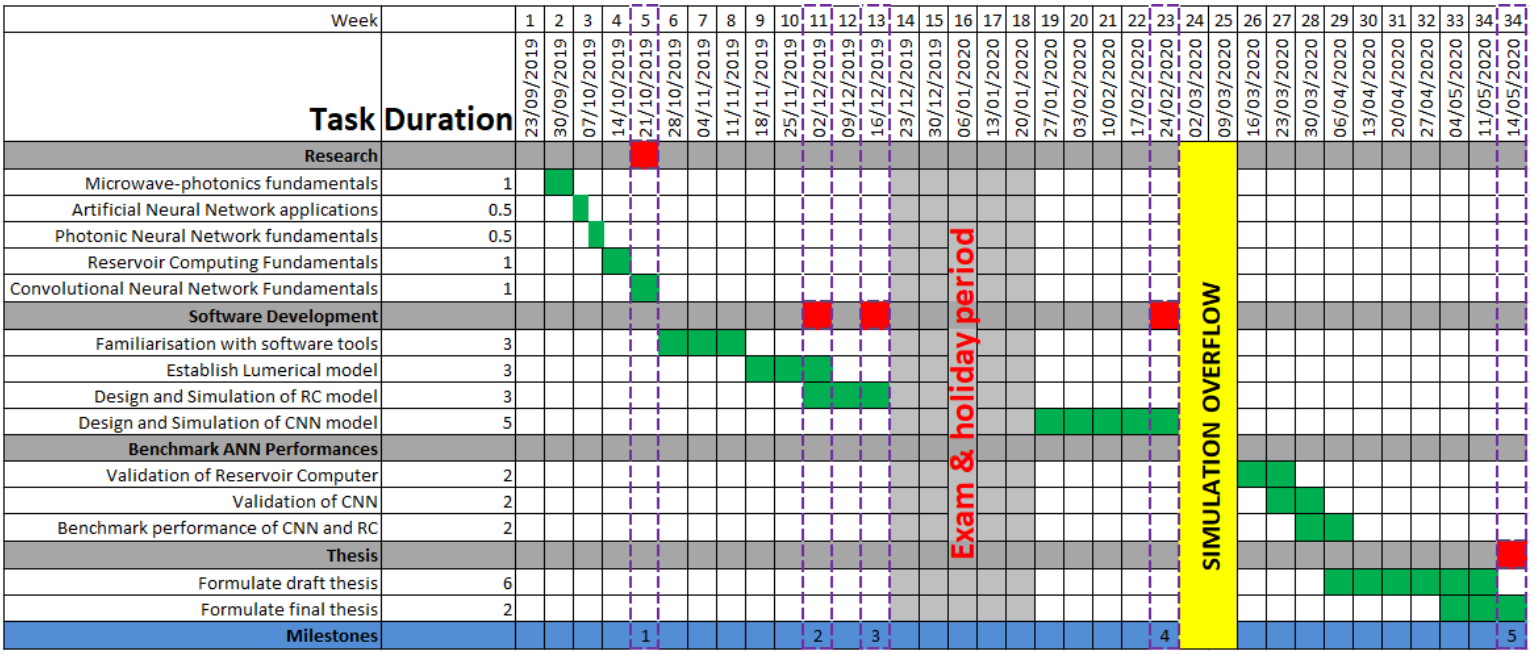

Figure A.1: Original Gantt Chart time plan

Appendix B: Revised Project Plan Gantt Chart

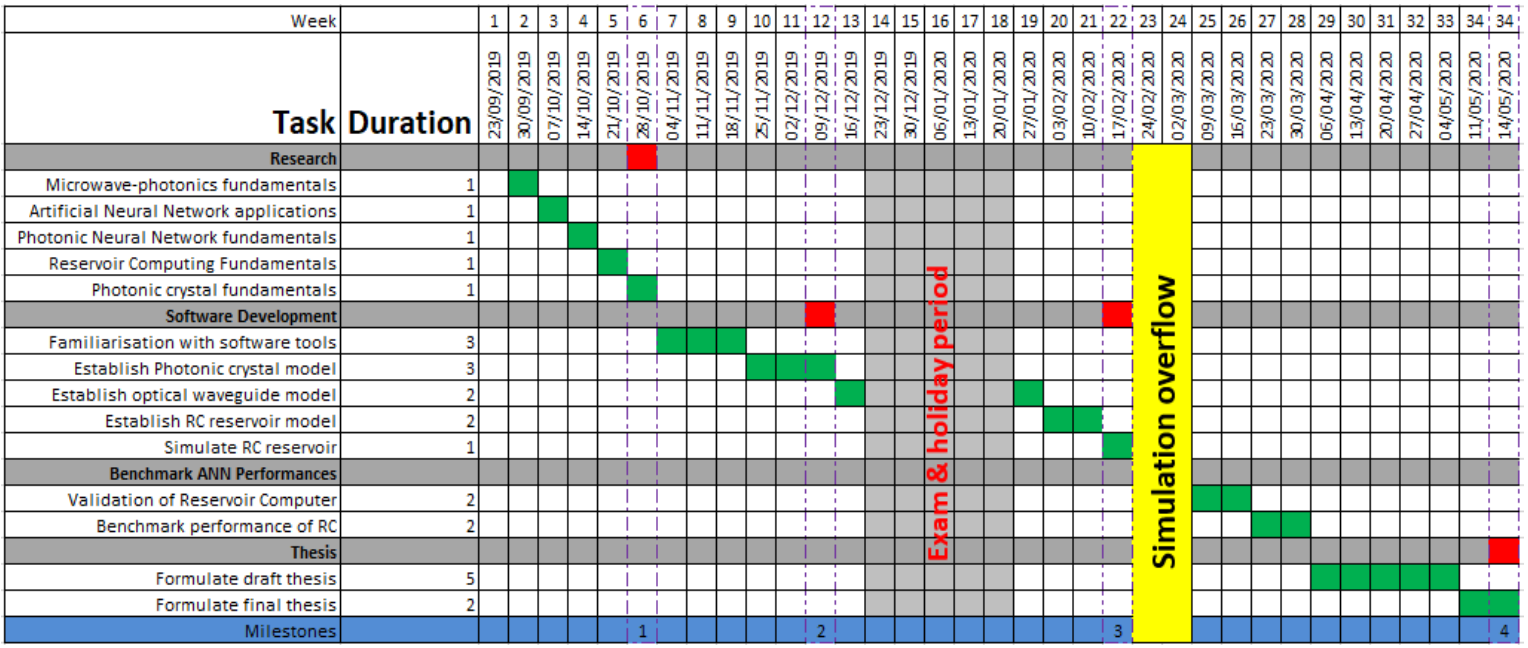

Figure B.1: Revised project Gantt Chart time plan

Appendix C: generateISIWaveforms React.js component

```
import React from "react"; //Import necessary libraries and components
import {LineChart, Line, LineSeries, PointSeries, LinearXAxis, LinearXAxisTickSeries, LinearYAxis, LinearYAxisTickSeries} from "reaviz";
import "./App.css"; //Import style sheet for this application
import individualSingalSig from "./graphData.js"; //Get chart data
import ISIdata from "./ISIdata.js"; //Get chart data
let data = [        //JSON object of waveform data···
];
export default class generateISIWaveforms extends React.Component {  //Class for ISI visualisation

  singleSignal = () => {  //Single symbol waveform
    return (
      <LineChart
        width={350}
        height={250}
        data={individualSingalSig}
        xAxis={
          <LinearXAxis
            type="Number"
            scaled={true}
            tickSeries={<LinearXAxisTickSeries />}
          />
        }
        yAxis={
          <LinearYAxis scaled={true} tickSeries={<LinearYAxisTickSeries />} />
        }
        series={
          <LineSeries
            line={<Line strokeWidth={3} />}
            symbols={<PointSeries show={false} />}
            interpolation={"smooth"}
            colorScheme={"#EF6C67"}
          />
        }
      />
    );
  };
imposedSignals = () => {     //Imposed single waveforms
    return (
      <div>
        <div>
          <LineChart
            width={600}
            height={250}
            data={data}
            series={
              <LineSeries
                type="grouped"
                scaled={true}
                interpolation={"smooth"}
                line={<Line strokeWidth={3} />}
                colorScheme={"#EF6C67"}
                symbols={<PointSeries show={false} />}
              />
            }
          />
        </div>
      </div>
    );
};
  interSymbolInterferenceSignal = () => { //Signal displaying ISI
    return (
      <LineChart
        width={600}
        height={250}
        data={ISIdata}
        xAxis={
          <LinearXAxis
            type="Number"
            scaled={true}
            tickSeries={<LinearXAxisTickSeries />}
          />
        }
        yAxis={
          <LinearYAxis scaled={true} tickSeries={<LinearYAxisTickSeries />} />
        }
        series={
          <LineSeries
            line={<Line strokeWidth={3} />}
            symbols={<PointSeries show={false} />}
            interpolation={"smooth"}
            colorScheme={"#EF6C67"}
          />
        }
      />
    );
  };
  render() {       //Render waveforms
    return (
      <div className="grid">          {/*Use grid styling*/}
        <div classNname="single">{this.singleSignal()}</div>
        <div>{this.imposedSignals()}</div>
        <div>{this.interSymbolInterferenceSignal()}</div>
      </div>
    );
  }
}
```

Appendix D: generateGraph React.js component for graphical data visualisation

```
import React from "react"; //Import necessary libraries and components
import {
  LineChart, Line,
  LineSeries, PointSeries,
  LinearXAxis,LinearXAxisTickSeries,
  LinearYAxis,LinearYAxisTickSeries,
} from "reaviz";
import "./App.css"; //Import style sheet for the application
import bandGapData from "./graphData.js"; //Get chart data
export default class generateGraph extends React.Component {//Generate graph class
  renderChart = () => { //Render graph
    return (
      <div>
        {/*Set parameters of graph*/}
        <LineChart
          width={900}
          height={550}
          data={bandGapData}
          xAxis={
            <LinearXAxis
              type="Number"
              scaled={true}
              tickSeries={<LinearXAxisTickSeries />}
            />
          }
          yAxis={
            <LinearYAxis scaled={true} tickSeries={<LinearYAxisTickSeries />} />
          }
          series={
            <LineSeries
              line={<Line strokeWidth={3} />}
              symbols={<PointSeries show={true} />}
              interpolation={"smooth"}
              colorScheme={"#EF6C67"}
            />
          }
        />
      </div>
    );
  };
  render() {
    //Render graph
    return (
      <div className="flex">{/*Use flexbox styling*/}
        {this.renderChart()}
      </div>
    )
  }
}
```

Appendix E: Main MATLAB script to generate RC training data

```
%% Loading and preprocessing data from Lumerical
load('RC_reservoir_data.mat')
Ex_1 = squeeze(Ex_1(1:1:1,:));Ex_2 = squeeze(Ex_2(1:1:1,:));Ex_3 = squeeze(Ex_3(1:1:1,:));Ex_4 = squeeze(Ex_4(1:1:1,:));Ex_5 = squeeze(Ex_5(1:1:1,:));
Ex_6 = squeeze(Ex_6(1:1:1,:));Ex_7 = squeeze(Ex_7(1:1:1,:));Ex_8 = squeeze(Ex_8(1:1:1,:));Ex_9 = squeeze(Ex_9(1:1:1,:));Ex_10 = squeeze(Ex_10(1:1:1,:));
Ex_11 = squeeze(Ex_11(1:1:1,:));Ex_12 = squeeze(Ex_12(1:1:1,:));Ex_13 = squeeze(Ex_13(1:1:1,:));Ex_14 = squeeze(Ex_14(1:1:1,:));
Ex_15 = squeeze(Ex_15(1:1:1,:));Ex_16 = squeeze(Ex_16(1:1:1,:));Ex_17 = squeeze(Ex_17(1:1:1,:));Ex_18 = squeeze(Ex_18(1:1:1,:));
Ex_19 = squeeze(Ex_19(1:1:1,:));Ex_20 = squeeze(Ex_20(1:1:1,:)); %r1 - r20
%% Defining some parameters
carrierFreq = 100e12; %int %Frequency of carrier signal
informationBitrate = 0.001*carrierFreq;%int %How fast any bits transferred
carrierAmp = 1; %int %Amplitude of carrier signal
informationBit = hardlim(rand(1,100)-0.5);%Information bit stream
dt = 2*t(2);% dt time period (comes from r1..r20)
informationDuration = 1/informationBitrate; % 1ms
infoNumOfPoints = round(informationDuration/dt);%num of points per bit
num_time_step_required = length(informationBit) * infoNumOfPoints;% number of time steps needed
time = (0:num_time_step_required-1)*dt; %time = vector from 0 to number of points-1 *dt
%% Zeropadding and interpolate for gaussian impulse and all outputs (responses) of resonant cavity
[Ex_1] = interpZeroPad(t(2), Ex_1, dt, num_time_step_required);[Ex_2] = interpZeroPad(t(2), Ex_2, dt, num_time_step_required);
[Ex_3] = interpZeroPad(t(2), Ex_3, dt, num_time_step_required);[Ex_4] = interpZeroPad(t(2), Ex_4, dt, num_time_step_required);
[Ex_5] = interpZeroPad(t(2), Ex_5, dt, num_time_step_required);[Ex_6] = interpZeroPad(t(2), Ex_6, dt, num_time_step_required);
[Ex_7] = interpZeroPad(t(2), Ex_7, dt, num_time_step_required);[Ex_8] = interpZeroPad(t(2), Ex_8, dt, num_time_step_required);
[Ex_9] = interpZeroPad(t(2), Ex_9, dt, num_time_step_required);[Ex_10] = interpZeroPad(t(2), Ex_10, dt, num_time_step_required);
[Ex_11] = interpZeroPad(t(2), Ex_11, dt, num_time_step_required);[Ex_12] = interpZeroPad(t(2), Ex_12, dt, num_time_step_required);
[Ex_13] = interpZeroPad(t(2), Ex_13, dt, num_time_step_required);[Ex_14] = interpZeroPad(t(2), Ex_14, dt, num_time_step_required);
[Ex_15] = interpZeroPad(t(2), Ex_15, dt, num_time_step_required);[Ex_16] = interpZeroPad(t(2), Ex_16, dt, num_time_step_required);
[Ex_17] = interpZeroPad(t(2), Ex_17, dt, num_time_step_required);[Ex_18] = interpZeroPad(t(2), Ex_18, dt, num_time_step_required);
[Ex_19] = interpZeroPad(t(2), Ex_19, dt, num_time_step_required);[Ex_20] = interpZeroPad(t(2), Ex_20, dt, num_time_step_required);%r20
[Ex_source] = interpZeroPad(time_source(2), Ex_source, dt, num_time_step_required); %source (Gaussian impulse) after padding and interpolation
%%Generate input signal
[inputSig] = generateModulatedInputSignal(dt, num_time_step_required, carrierAmp, carrierFreq, informationBit, informationBitrate);
%%Generate output signal
[Ex_1] = generateModulatedOutputSignal(Ex_1, Ex_source ,inputSig);[Ex_2] = generateModulatedOutputSignal(Ex_2, Ex_source ,inputSig);
[Ex_3] = generateModulatedOutputSignal(Ex_3, Ex_source ,inputSig);[Ex_4] = generateModulatedOutputSignal(Ex_4, Ex_source ,inputSig);
[Ex_5] = generateModulatedOutputSignal(Ex_5, Ex_source ,inputSig);[Ex_6] = generateModulatedOutputSignal(Ex_6, Ex_source ,inputSig);
[Ex_7] = generateModulatedOutputSignal(Ex_7, Ex_source ,inputSig);[Ex_8] = generateModulatedOutputSignal(Ex_8, Ex_source ,inputSig);
[Ex_9] = generateModulatedOutputSignal(Ex_9, Ex_source ,inputSig);[Ex_10] = generateModulatedOutputSignal(Ex_10, Ex_source ,inputSig);
[Ex_11] = generateModulatedOutputSignal(Ex_11, Ex_source ,inputSig);[Ex_12] = generateModulatedOutputSignal(Ex_12, Ex_source ,inputSig);
[Ex_13] = generateModulatedOutputSignal(Ex_13, Ex_source ,inputSig);[Ex_14] = generateModulatedOutputSignal(Ex_14, Ex_source ,inputSig);
[Ex_15] = generateModulatedOutputSignal(Ex_15, Ex_source ,inputSig);[Ex_16] = generateModulatedOutputSignal(Ex_16, Ex_source ,inputSig);
[Ex_17] = generateModulatedOutputSignal(Ex_17, Ex_source ,inputSig);[Ex_18] = generateModulatedOutputSignal(Ex_18, Ex_source ,inputSig);
[Ex_19] = generateModulatedOutputSignal(Ex_19, Ex_source ,inputSig);[Ex_20] = generateModulatedOutputSignal(Ex_20, Ex_source ,inputSig);
%figure(4);
%plot(time, Ex_1); % output modulated signal at output 1 of reservoir
%% retrieve the envelope
information_bit = abs(hilbert(inputSig));
observed_signal_envelope_1 = abs(hilbert(Ex_1));observed_signal_envelope_2 = abs(hilbert(Ex_2));observed_signal_envelope_3 = abs(hilbert(Ex_3));
observed_signal_envelope_4 = abs(hilbert(Ex_4));observed_signal_envelope_5 = abs(hilbert(Ex_5));observed_signal_envelope_6 = abs(hilbert(Ex_6));
observed_signal_envelope_7 = abs(hilbert(Ex_7));observed_signal_envelope_8 = abs(hilbert(Ex_8));observed_signal_envelope_9 = abs(hilbert(Ex_9));
observed_signal_envelope_10 = abs(hilbert(Ex_10));observed_signal_envelope_11 = abs(hilbert(Ex_11));observed_signal_envelope_12 = abs(hilbert(Ex_12));
observed_signal_envelope_13 = abs(hilbert(Ex_13));observed_signal_envelope_14 = abs(hilbert(Ex_14));observed_signal_envelope_15 = abs(hilbert(Ex_15));
observed_signal_envelope_16 = abs(hilbert(Ex_16));observed_signal_envelope_17 = abs(hilbert(Ex_17));observed_signal_envelope_18 = abs(hilbert(Ex_18));
observed_signal_envelope_19 = abs(hilbert(Ex_19));observed_signal_envelope_20 = abs(hilbert(Ex_20));
clearvars Ex_1 Ex_2 Ex_3 Ex_4 Ex_5 Ex_6 Ex_7 Ex_8 Ex_9 Ex_10 Ex_11 Ex_12 Ex_13 Ex_14 Ex_15 Ex_16 Ex_17 Ex_18 Ex_19 Ex_20 %Memory management
%% downsampling
downsampling_rate = 3;
timeDownSampled = time(1:downsampling_rate:end);
information_bit = information_bit(1:downsampling_rate:end);
observed_signal_envelope_1 = observed_signal_envelope_1(1:downsampling_rate:end);observed_signal_envelope_2 = observed_signal_envelope_2(1:downsampling_rate:end);
observed_signal_envelope_3 = observed_signal_envelope_3(1:downsampling_rate:end);observed_signal_envelope_4 = observed_signal_envelope_4(1:downsampling_rate:end);
observed_signal_envelope_5 = observed_signal_envelope_5(1:downsampling_rate:end);observed_signal_envelope_6 = observed_signal_envelope_6(1:downsampling_rate:end);
observed_signal_envelope_7 = observed_signal_envelope_7(1:downsampling_rate:end);observed_signal_envelope_8 = observed_signal_envelope_8(1:downsampling_rate:end);
observed_signal_envelope_9 = observed_signal_envelope_9(1:downsampling_rate:end);observed_signal_envelope_10 = observed_signal_envelope_10(1:downsampling_rate:end);
observed_signal_envelope_11 = observed_signal_envelope_11(1:downsampling_rate:end);observed_signal_envelope_12 = observed_signal_envelope_12(1:downsampling_rate:end);
observed_signal_envelope_13 = observed_signal_envelope_13(1:downsampling_rate:end);observed_signal_envelope_14 = observed_signal_envelope_14(1:downsampling_rate:end);
observed_signal_envelope_15 = observed_signal_envelope_15(1:downsampling_rate:end);observed_signal_envelope_16 = observed_signal_envelope_16(1:downsampling_rate:end);
observed_signal_envelope_17 = observed_signal_envelope_17(1:downsampling_rate:end);observed_signal_envelope_18 = observed_signal_envelope_18(1:downsampling_rate:end);
observed_signal_envelope_19 = observed_signal_envelope_19(1:downsampling_rate:end);observed_signal_envelope_20 = observed_signal_envelope_20(1:downsampling_rate:end);
%Combine modulated input signals to pass into RC read-out layer
observed_signal_envelope_combined = [observed_signal_envelope_1;observed_signal_envelope_2;observed_signal_envelope_3; observed_signal_envelope_4; ...
observed_signal_envelope_5;observed_signal_envelope_6;observed_signal_envelope_7;observed_signal_envelope_8;observed_signal_envelope_9;observed_signal_envelope_10;...
observed_signal_envelope_11;observed_signal_envelope_12;observed_signal_envelope_13;observed_signal_envelope_14;observed_signal_envelope_15;
observed_signal_envelope_16;observed_signal_envelope_17;observed_signal_envelope_18;observed_signal_envelope_19;observed_signal_envelope_20;];
%% plotting
%figure(5);
%plot(timeDownSampled, information_bit,'DisplayName','information_bit');hold on; %What we want the read-out layer to produce - Yteacher
%plot(timeDownSampled, observed_signal_envelope_1,'DisplayName','observed_signal_envelope_1');hold on;
%plot(timeDownSampled, observed_signal_envelope_dwn,'DisplayName','observed_signal_envelope_dwn');%enveloped om is passed to read-out layer
myRC = RVCObj('is_reservoir_enabled', false,...          %False as the reservoir has been created externally
              'is_input_weighted', true,...
              'is_readout_has_direct_access_to_input', true,...
              'is_readout_has_constant_pump_offset',true,...
              'input_signal',zeros(size(information_bit)),...
              'neuron_activation_signal',observed_signal_envelope_combined,...  %oml-20 reservoir activation states
              'teacher_signal',information_bit,...      %Y teacher signal
              'feedback_delay_timestep',300); %Feedback delay time-step (500dt)
myRC.doTraining(); % perform first training
myRC.do_validate('testing_input_signal',zeros(size(information_bit)),...
                 'neuron_activation_signal',observed_signal_envelope_combined);
%Calculate the Normalised Mean Square Error
NMSE = 0;
temp_mean1 = sum((abs(myRC.out_y - information_bit)).^2,1);
temp_mean1 = sum(temp_mean1,2)/length(information_bit);
temp_mean2 = sum(information_bit.^2,1);
temp_mean2 = sum(temp_mean2,2)/length(information_bit);
NMSE = NMSE + temp_mean1/temp_mean2;  %need to display this
fprintf('%f', NMSE);
%% plot the last validation
figure(1)
    plot(timeDownSampled, information_bit,'DisplayName',"Target signal") ; hold on
    plot(timeDownSampled, myRC.out_y,'--','DisplayName',"What we get") ; hold on
    ylim([0 max(information_bit)])
    legend
```

Appendix F: MATLAB script to generate the modulated input signal

```
function [modulatedInputSignal] = generateModulatedInputSignal(dt, numOfPoints, carrierAmp, carrierFreq, informationBit, informationBitrate)
time = (0:numOfPoints-1)*dt; %time = vector from 0 to number of points-1 *dt

%mask
informationDuration = 1/informationBitrate; % 1ms
infoNumOfPoints = round(informationDuration/dt);%num of points per bit
mask = zeros(size(time));%Default mask values are zero

for eachBit = 1: length(informationBit) %set value of mask to envelope carrier signal

    if(informationBit(eachBit) == 1)
        lowerBound = ((eachBit-1)*infoNumOfPoints)+1;
        upperBound = (eachBit*infoNumOfPoints);
        if(upperBound > length(time))
            error("Information stream too long");
        end
        mask(lowerBound:upperBound) = 1;
    end
end

%Generate carrier signal
carrier = carrierAmp*sin((2*pi*carrierFreq)*time);
figure(1);
plot(time, carrier); %plot carrier wave against time
title('Carrier signal');
xlabel('Time(s)');
ylabel('Amplitude');
%xlim([])
ylim([-1.5 1.5]);

%Plot modulated signal
modulatedInputSignal = mask.*carrier;
figure(2);
plot(time, modulatedInputSignal); %Plot modulated input signal against time
title('Modulated input signal');
xlabel('Time(s)');
ylabel('Amplitude');
%xlim([])
ylim([-1.5 1.5]);
```

Appendix G: MATLAB script to generate the modulated output signal

```
function [modulatedOutputSigInTimeDomain] = generateModulatedOutputSignal(rInTimeDomain, iInTimeDomain, imInTimeDomain)
%r = vector of resonant cavity outputs after gaussian pulse is injected
%i = Original Gaussian impulse source signal interpolated and padded with zeros
%im = Modulated input signal

%FT of aInTimeDomain to get aInFrequencyDomain
rInFreqDomain = fft(rInTimeDomain);

%FT of iInTimeDomain to get iInFrequencyDomain
iInFreqDomain = fft(iInTimeDomain);

%Get normalised resonant cavity system response
normalisedResponseInFreqDomain = rInFreqDomain ./ iInFreqDomain;
%normalisedResponseInFreqDomain = fftshift(normalisedResponseInFreqDomain);

%FT of modulatedInputSignalInTimeDomain to get modulatedInputSignalInFrequencyDomain
imInFreqDomain = fft(imInTimeDomain);

%Get modulatedOutputSignalInFrequencyDomain
modulatedOutputSigInFreqDomain = normalisedResponseInFreqDomain .* imInFreqDomain;
%modulatedOutputSigInFreqDomain = fftshift(modulatedOutputSigInFreqDomain);

%Inverse FT to get modulatedOutputSignalInTimeDomain
modulatedOutputSigInTimeDomain = ifft(modulatedOutputSigInFreqDomain);
```