MASTER

Neural network regression for the warm plasma dispersion relation in electron cyclotron ray tracing

Sebastian, Nicholas

Award date:
2018

Technische Universiteit
Eindhoven
University of Technology

Department of Applied Physics

Science and Technology of Nuclear Fusion

Neural Network Regression for the Warm Plasma Dispersion Relation in Electron Cyclotron Ray Tracing

Master Thesis

Nicholas Sebastian

Supervisors:
Dr. Egbert Westerhof
Dr. Jonathan Citrin

Eindhoven, November 2018

Abstract

Electron Cyclotron Resonance Heating (ECRH) is one of the external heating methods employed in almost all of present day Tokamaks. Its main advantages include localized power deposition, high power absorption efficiency, the ability to suppress MHD instabilities, and the ability to induce current drive. All of which are desirable for plasma control. Computational tools modelling the propagation and absorption of the injected EC waves have been developed to better optimize ECRH performance. However, most of these computational tools are not yet real-time capable. A real-time capable ECRH model is desirable for real-time Tokamak plasma control which in turn is desirable for plasma performance optimization. Here, we use neural network regression to develop a faster surrogate model of the warm plasma routine of a 3D ECRH ray tracing code called TORAYFOM as part of the effort to make it real-time capable. We show that it is possible to obtain sufficiently good neural network regression for the O-mode heating and to implement a faster surrogate model in TORAYFOM based on the aforementioned regression.

Acknowledgements

I would like to thank my supervisors Egbert Westerhof and Jonathan Citrin for their support and counseling throughout my master thesis project. Their guidance is crucial for ensuring the project went in the correct direction. I would also like to thank Karel van de Plassche for helping me with the neural network aspect of this project especially the technical side of it. His help allowed me to quickly gain basic understanding on how to train a neural network. A process which is previously very opaque to me. Lastly, I would like to thank my family and friends for their support during this project be it directly or indirectly.

The LaTeXtemplate for this master thesis is adapted from the template created by Thijs Nugteren and Joos Buijs.

Contents

Chapter 1

Introduction and Research Questions

1.1 Introduction

The world is undergoing a combination of energy and climate crisis [23]. Population growth and the increase in the standard of living especially in the developing world [37] are contributing to the continuous rise of global energy demand. On the other hand, mankind's dependency on fossil fuels for energy generation [1] has damaged the environment and caused a climate crisis. In order to successfully overcome the energy and climate crisis, sustainable energy sources have to be developed. One of these sustainable energy sources is nuclear fusion [18].

Nuclear fusion is the process where two (or more) atoms fuse together to form a heavier nucleus. This process is accompanied by the release of energy via the conversion of mass. This is the very process which powers the stars. The development of nuclear fusion for energy generation has been ongoing since the 1950s. Magnetically confined thermonuclear fusion remains the most promising concept for energy generation. Examples of devices belonging to this concept are the Tokamak and the Stellarator. The word "thermonuclear" refers to the fact that the fuel are heated up to plasma temperatures (usually tens of keV) where the fusion rates will be maximized. There are several heating methods commonly employed in the Tokamak or the Stellarator. The main topic for this project is one of the heating methods: Electron Cyclotron Resonance Heating (ECRH). More specifically, the focus is on the computational modelling for ECRH waves propagation and absorption. Note that computational modelling of ECRH is almost always accompanied by Electron Cyclotron Current Drive (ECCD). For the purpose of this project however, the theoretical discussion is solely on ECRH. Current drive profiles will still be shown for comparison but they will not be accompanied by theoretical description.

ECRH is performed by injecting electromagnetic waves into the plasma. These waves will be absorbed by the available electron resonances in the plasma and will exclusively heat the electrons. The heating done by ECRH is localized which also makes ECRH a suitable tool for plasma control due to the possibility of precise tailoring of the heat and current distributions. To better optimize ECRH performance in the Tokamak, computational tools have been developed to model the waves propagation and absorption [16] [36]. One such tool is the TORAYFOM (referred interchangeably with TORAY throughout this work) which is a 3D ray tracing code for ECRH (and ECCD) in the Tokamak [34] [7]. For the purpose of real-time control of the tokamak plasma, we require a real-time capable ECRH code. Unfortunately, TORAYFOM is not yet real-time capable. In particular, the warm plasma routine (see Chapter 2 for a description of warm plasma) of TORAYFOM is computationally relatively expensive. In this work, we use neural network regression to develop a faster surrogate model that emulates the functionality of the original warm plasma routine.

A brief qualitative description of neural network will be given here. A neural network can be regarded as a single nonlinear function which is able to make an approximate mapping between an input parameter space and an output parameter space [2]. This dataset of input and output mapping needs to be provided beforehand. The advantage of using neural networks then is the ability to predict outputs from known inputs without actually performing the actual calculations themselves. This often results in a significant decrease in computational time since the function of a neural network is much faster to compute than the actual calculations. As one example from the field of Fusion, neural networks have been used to speed up turbulent transport modelling [31] [13] [5].

1.2 Research Question

ECRH code that is able to perform real-time simulations is desirable for the purpose of real-time control of the Tokamak plasma. To date, the only known ECRH code to demonstrate real-time capability is TORBEAM [8] [9]. TORBEAM is actually an ECRH beam tracing code. A beam tracing code is able to include diffraction effects not accounted for in the ray tracing code. Diffraction effects can lead to a broadening of the power deposition profile. The real-time version of TORBEAM however reduces to a ray-tracing code computing a single ray. Furthermore, TORBEAM also limit itself to using a cold plasma dispersion relation at the moment. For certain cases near the electron cyclotron resonance, the cold plasma ray tracing could deviate significantly from the warm plasma ray tracing [35]. Thus, it will be beneficial to also develop a real-time version of an ECRH code that can account for warm plasma effects such as TORAYFOM. In this project, we are researching the possibilities of speeding up the warm plasma routines of TORAYFOM using the help of neural networks. The end goal is to have a real-time capable ECRH ray tracing

code which has the option of using the warm plasma dispersion relation. Note however that TORAYFOM itself is currently not optimized for speed. Additional work other than the speed up of the warm plasma routine also needs to be done in order to achieve real-time capability. This is however outside the scope of this project.

Neural networks have been successfully used in the field of Fusion for computationally heavy endeavor such as turbulent transport modelling. Here, they will be used on a ECRH ray tracing code called TORAYFOM that models the waves propagation and absorption in the fusion plasma. This ECRH modelling is computationally cheaper than turbulent transport modelling. Nevertheless, it is still necessary to speed up the process for the ultimate end goal of having real-time simulation. Thus, the research question for this project is the following:

"Is it computationally feasible - with reasonable accuracy [1] - to obtain a neural network approximation (mapping 4D [2] input parameters space onto 2D [3] output parameters space) of the warm plasma ray tracing routine of TORAYFOM?

In order to answer the research question, the following steps were taken:

1. Studied the TORAY subroutine which is responsible for the warm plasma dispersion relation calculation and make a code that calls this subroutine for the purpose of generating the required dataset. Also, determine whether the subroutine needs to be modified or simplified.

2. Checked the validity of the outputs produced by TORAY's warm plasma dispersion calculation.

3. Determined the final size of the dataset considering the requirements and limitations (or problems).

4. Generated the database required for the training of the neural network.

5. Trained a neural network on the database.

6. Checked the output from the neural network against the dataset.

7. Compared ray tracing results made from the neural network outputs against the original results from TORAY's subroutines.

The project was carried out at the Dutch Institute For Fundamental Energy Research (DIFFER). The supervisors for this project are Egbert Westerhof and Jonathan Citrin. TORAYFOM

[1] Determined by the quality of the ray tracing

[2] Normalized magnetic field ($Y = \omega_{ce}/\omega$), Normalized electron density ($X = (\omega_{pe}/\omega)^2$), Electron temperature (T_e), Parallel refractive index ($N_\parallel$)

[3] Real and imaginary parts of the perpendicular refractive index ($N_\perp$)

is written in Fortran and is provided by Egbert Westerhof via the European Integrated Tokamak Modelling (ITM) framework. The work on TORAYFOM was done through the Eurofusion Gateway. The work on the neural network was done using the open-source TensorFlow [10] library in Python.

Chapter 2

Waves in Plasmas

As Electron Cyclotron Resonance Heating (ECRH) is performed by injecting electromagnetic waves into the fusion plasma, knowledge on waves propagation and absorption in plasma is required in order to utilize ECRH effectively. This chapter briefly reviews basic and relevant concept of waves propagation and absorption in plasma. Reference texts include Stix [28], Swanson [29], Freidberg [18], and Westerhof [33].

2.1 Dielectric Tensor

The dielectric tensor of a plasma is able to summarize the plasma behavior and response towards the propagating electromagnetic waves. Plasma behavior is in general anisotropic thus necessitating the use of a tensor. The starting point for the definition of the dielectric tensor of a plasma is Maxwell's equations (temperature and collision effects are neglected):

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{2.1a}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \epsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t} \tag{2.1b}$$

The variables $\mathbf{E}$, $\mathbf{B}$, and $\mathbf{J}$ denotes the electric field, magnetic field, and current density respectively as per usual. The constants μ_0 and ϵ_0 are the vacuum permeability and vacuum permitivitty. We can combine both equations (2.1a) and (2.1b) as follows:

$$\nabla \times (\nabla \times \mathbf{E}) = -\frac{\partial}{\partial t}\left(\mu_0 \mathbf{J} + \epsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t}\right) \tag{2.2}$$

Ohm's law relates $\mathbf{J}$ and $\mathbf{E}$ as:

$$\mathbf{J} = \overleftrightarrow{\sigma} \cdot \mathbf{E} \tag{2.3}$$

where $\overleftrightarrow{\sigma}$ is the conductivity tensor which can only be determined after we have assumed a certain plasma model. Combining equations (2.2) and (2.3) and assuming harmonic perturbation for the $\left(\sim e^{i(kx-\omega t)}\right)$ to the electric field, we obtain:

$$\mathbf{k} \times (\mathbf{k} \times \mathbf{E}) + i\omega \left(\mu_0 \overleftrightarrow{\sigma} \cdot \mathbf{E} - i\omega\epsilon_0\mu_0 \mathbf{E} \right) = 0 \tag{2.4}$$

where $\mathbf{k}$ and ω are the wave vector and the radial frequency respectively. We take the dielectric tensor to be:

$$\overleftrightarrow{\epsilon} = \mathbf{1} + \frac{i}{\omega\epsilon_0} \overleftrightarrow{\sigma} \tag{2.5}$$

leading equation (2.4) to become:

$$\mathbf{k} \times (\mathbf{k} \times \mathbf{E}) + \frac{\omega^2}{c^2} \overleftrightarrow{\epsilon} \cdot \mathbf{E} = 0 \tag{2.6}$$

where c is the speed of light in vacuum and $c^2 = 1/\epsilon_0\mu_0$. Equation (2.5) gives us the dielectric tensor provided we know the conductivity tensor of the plasma model we are using.

2.2 Dispersion Relation (General Formulation)

The dispersion relation of a certain medium is the relationship between ω and $\mathbf{k}$ in that medium. One can easily obtain the phase velocity and the group velocity of the propagating electromagnetic waves from the dispersion relation. In plasma, the dispersion relation can be obtained from equation (2.6). More specifically in order to have a non-trivial solutions (i.e. non-vanishing electric field), the determinant of equation (2.6) must be zero:

$$det \left| \mathbf{kk} - k^2\mathbf{1} + \frac{\omega^2}{c^2} \overleftrightarrow{\epsilon} \right| = 0 \tag{2.7}$$

The determination of the dispersion relation can only proceed when the dielectric tensor $\overleftrightarrow{\epsilon}$ is known. In turn, we can only determine the dielectric tensor if we have already assume a certain plasma model. Two plasma models commonly used are the cold plasma model and the warm plasma model. As their names suggested, the cold plasma model neglects all temperature effects in contrast to the warm plasma model. Analysis of the cold plasma model is significantly simpler than the more realistic warm plasma model. Nevertheless, it offers good physical insight and can be seen as the lowest order approximation of the warm plasma model. Notable deficiency of the cold plasma is its inability to account for wave absorption. We will begin with a description of the cold plasma model followed by that of the warm plasma model.

2.3 Cold Plasma

We restrict the current discussion to a cold two-fluid magnetized plasma. Two-fluid simply means we only take into account a single ion species and the electrons. The starting point is the

momentum equations:

$$m_e \left(\frac{\partial \mathbf{v_e}}{\partial t} + \mathbf{v_e} \cdot \nabla \mathbf{v_e} \right) = -e \left(\mathbf{E} + \mathbf{v_e} \times \mathbf{B} \right) \tag{2.8a}$$

$$m_i \left(\frac{\partial \mathbf{v_i}}{\partial t} + \mathbf{v_i} \cdot \nabla \mathbf{v_i} \right) = e \left(\mathbf{E} + \mathbf{v_i} \times \mathbf{B} \right) \tag{2.8b}$$

where m, $\mathbf{v}$, e, and $\mathbf{B}$ are the species' mass, species' velocity, electron charge, and the external magnetic field respectively. The subscript i and e denotes the ions and the electrons respectively. The next step is to linearize the equations by considering small perturbations around equilibrium values. Additionally, we take the static or equilibrium values of the velocity and electric field to be zero. This means:

$$n = n_0 + n_1 \tag{2.9a}$$

$$\mathbf{v} = \mathbf{v_1} \tag{2.9b}$$

$$\mathbf{E} = \mathbf{E_1} \tag{2.9c}$$

$$\mathbf{B} = \mathbf{B_0} + \mathbf{B_1} \tag{2.9d}$$

where the subscript 0 and 1 denotes the equilibrium and perturbation values respectively. We have also dropped the species subscript temporarily for convenience. We also take both the ion density and electron density to be equal to n_0. After dropping all terms higher than the first order, we obtain the following:

$$m \frac{\partial \mathbf{v_1}}{\partial t} = q \left(\mathbf{E_1} + \mathbf{v_1} \times \mathbf{B_0} \right) \tag{2.10a}$$

$$\mathbf{J_1} = e n_0 \left(\mathbf{v_{i1}} - \mathbf{v_{e1}} \right) \tag{2.10b}$$

where equation 2.10b follows from the usual definition of current density. The charge q is either $-e$ for electrons or $+e$ for ions. We take a coordinate system where the external magnetic field $\mathbf{B_0}$ is along the direction of the z-axis while the perpendicular wave vector is along the x-axis. After assuming a harmonic perturbation to equation (2.10a), we obtain the relationship between the velocity and the electric field as follows:

$$\begin{pmatrix} -i\omega & -\omega_c & 0 \\ \omega_c & -i\omega & 0 \\ 0 & 0 & -i\omega \end{pmatrix} \mathbf{v_1} = \frac{\pm e}{m} \mathbf{E_1} \tag{2.11}$$

where $\left(\omega_c = \frac{\pm e \mathbf{B_0}}{m} \right)$ is the cyclotron frequency. Combining equation (2.11) with equation (2.10b) and Ohm's law (2.3), we can solve for the conductivity tensor giving us:

$$\overleftrightarrow{\sigma} = \epsilon_0 \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_{yy} & 0 \\ 0 & 0 & \sigma_{zz} \end{pmatrix} \tag{2.12}$$

with

$$\sigma_{xx} = \sigma_{yy} = i \sum_s \frac{\omega \omega_{ps}^2}{\omega^2 - \omega_{cs}^2} \tag{2.13a}$$

$$\sigma_{xy} = -\sigma_{yx} = -\sum_s \frac{\omega_{cs} \omega_{ps}^2}{\omega^2 - \omega_{cs}^2} \tag{2.13b}$$

$$\sigma_{zz} = i \frac{\omega_{ps}^2}{\omega} \tag{2.13c}$$

where $\omega_{ps} = \sqrt{\frac{n_s q_s^2}{\epsilon_0 m_s}}$ is the plasma frequency of the species. Here, the end result for the conductivity tensor is given for a multi-fluid plasma in general. The summation is over all plasma species s. For the case of a two-fluid plasma, the summation is simply over the single ion species and the electron. The dielectric tensor can be obtained through equation (2.5) and is usually casted in the form:

$$\overleftrightarrow{\epsilon} = \begin{pmatrix} S & -iD & 0 \\ iD & S & 0 \\ 0 & 0 & P \end{pmatrix} \tag{2.14}$$

with

$$S = \frac{1}{2}(R + L) = 1 - \sum_s \frac{\omega_{ps}^2}{\omega^2 - \omega_{cs}^2} \tag{2.15a}$$

$$D = \frac{1}{2}(R - L) = \sum_s \frac{\omega_{cs}}{\omega} \frac{\omega_{ps}^2}{\omega^2 - \omega_{cs}^2} \tag{2.15b}$$

$$R = 1 - \sum_s \frac{\omega_{ps}^2}{\omega(\omega + \omega_{cs})} \tag{2.15c}$$

$$L = 1 - \sum_s \frac{\omega_{ps}^2}{\omega(\omega - \omega_{cs})} \tag{2.15d}$$

$$P = 1 - \sum_s \frac{\omega_{ps}^2}{\omega^2} \tag{2.15e}$$

Here, R and L represent the plasma response to a right-handed and left-handed polarized electromagnetic waves respectively (i.e. when the electric field component of the wave is perpendicular to the external magnetic field). Notice that since ω_{ce} is negative and ω_{ci} is positive, R becomes

singular near ω_{ce} whereas L becomes singular near ω_{ci}. In other words, the wave polarization needs to match with the particle polarization in order for wave absorption to occur. P represents the plasma response - which is the same as an unmagnetized plasma - towards electromagnetic waves whose electric field component is parallel to the external magnetic field.

We can then obtain the dispersion relation by substituting the dielectric tensor into equation (2.7). We first obtain a dispersion tensor:

$$\overleftrightarrow{\Lambda} = \begin{pmatrix} S - N^2\cos^2\theta & -iD & N^2\sin^2\theta\cos^2\theta \\ iD & S - N^2 & 0 \\ N^2\sin^2\theta\cos^2\theta & 0 & P - N^2\sin^2\theta \end{pmatrix} \tag{2.16}$$

where the refractive index N is related to the wave number/vector as $N = \frac{ck}{\omega}$ and θ is the angle between the wave vector and the external magnetic field. The determinant of this dispersion tensor must be set to zero in order to obtain the dispersion relation. The resulting dispersion relation can be casted in many forms. Here, we cast the dispersion relation in the form of a biquadratic equation:

$$AN^4 + BN^2 + C = 0 \tag{2.17}$$

where the coefficients are as follows:

$$A = S\sin^2\theta + P\cos^2\theta \tag{2.18a}$$

$$B = RL\sin^2\theta + PS(1 + \cos^2\theta) \tag{2.18b}$$

$$C = PRL \tag{2.18c}$$

From this point onwards, we take the high frequency limit of the dispersion relation. This means that we neglect all ion contributions and only consider the electron contributions. We can then obtain a solution known as the Appleton-Hartree dispersion relation:

$$N^2 = 1 - \frac{X(1-X)}{1 - X - \frac{1}{2}Y^2\sin^2\theta \pm \sqrt{\left(\frac{1}{2}Y^2\sin^2\theta\right)^2 + (1-X)^2Y^2\cos^2\theta}} \tag{2.19}$$

where $X = \left(\frac{\omega_{pe}}{\omega}\right)^2$ and $Y = \frac{\omega_{ce}}{\omega}$.

Often, the parallel component - with respect to the equilibrium magnetic field - of the refractive index is taken to be a conserved quantity. In other words, only the perpendicular component of the refractive index contributes to the inhomogeneity of the plasma. The reason is that in a toroidal symmetry the quantity of major radius times the toroidal component of the refractive index RN_ϕ is conserved. Since the dominant magnetic field in a tokamak is the toroidal field, this also means that $N_\parallel \approx N_\phi$ is also approximately conserved. This allows us to recast the biquadratic equations in terms of the perpendicular component only:

$$A'N_\perp^4 + B'N_\perp^2 + C' = 0 \tag{2.20}$$

where the coefficients are now as follows:

$$A' = S \tag{2.21a}$$

$$B' = (S + P)(S - N_{\parallel}^2) - D^2 \tag{2.21b}$$

$$C' = P((S - N_{\parallel}^2)^2 - D^2) \tag{2.21c}$$

This means that we can obtain the solution $N_{\perp}^2$ by specifying four input parameters: magnetic field (B), parallel refractive index ($N_{\parallel}$), electron density (n_e), and the injected waves' frequency (f or ω). More succinctly, $N_{\perp}^2(B, N_{\parallel}, n_e, \omega)$ for the cold plasma case. Using normalized quantities: $N_{\perp}^2(X, Y, N_{\parallel})$. As X and Y are spatially varying, we could also state that $N_{\perp}^2(r, N_{\parallel})$. $N_{\perp}^2$ is generally considered to be a time independent quantity as the waves propagation is assumed to be fast.

We now turn the discussion towards wave cutoffs and resonances. Cutoffs occur when $N^2 = 0$ while resonances occur when $N^2 = \infty$. At the cutoff, the waves stop propagating through the plasma and are reflected instead. N^2 changes sign from positive to negative at the cutoff resulting in a purely imaginary value for the refractive index (i.e. the waves become evanescent). As for resonances, in addition to the wave-particle or cyclotron resonances ($\omega = \omega_{ce}$) there are hybrid resonances. We need to keep in mind that term "resonances" does not necessarily imply that energy absorption occurs. Even for the cyclotron resonances, it is necessary for the wave field to have the same polarization as the particle gyro motion in order for wave absorption to occur. In fact in the framework of the cold plasma analysis, we cannot account for any wave absorption at all. The cutoff and resonances will be illustrated more clearly when we consider the dispersion diagram for the waves. From now on, we will restrict the discussion to a perpendicular propagating wave ($\theta = \frac{\pi}{2}$) and oblique propagating wave as they are the more relevant one with regard to ECRH in a fusion plasma. An oblique propagating wave refers to a wave with finite parallel refractive index ($N_{\parallel}$). For these cases, there are two modes: the O-mode and the X-mode. We consider perpendicular propagation to illustrate these modes. In this case, the O-mode corresponds to the case where the electric field of the wave is parallel to the external magnetic field whereas the X-mode corresponds to the case where the electric field is perpendicular to the external magnetic field. For oblique propagation, these polarizations no longer hold.

The O-mode is a simple one. It only has a cutoff at the plasma frequency ω_{pe}. Depending on the electron density and magnetic field, this cutoff could be lower or higher than the electron cyclotron frequency. For ECRH to work, it is imperative that the cyclotron frequency is above the cutoff. The X-mode is somewhat more complex than the O-mode. It has two cutoffs: the right-handed (+) and left-handed (-) cutoffs. They are given as follows:

$$\omega_{\pm} = \pm\frac{1}{2}|\omega_{ce}| + \sqrt{\left(\frac{1}{2}\omega_{ce}\right)^2 + \omega_{pe}^2/(1 - N_{\parallel}^2)} \tag{2.22}$$

There are also two hybrid resonances: the upper hybrid and the lower hybrid. For the high frequency limit, only the upper hybrid is of interest. The upper hybrid is obtained by setting $S = 0$. The result is:

$$\omega_{uh} = \sqrt{\omega_{ce}^2 + \omega_{pe}^2} \tag{2.23}$$

There is an evanescent region between the right-handed cutoff and the upper hybrid. This evanescent region will separate the X-mode into the Fast X-mode (FX) branches and the Slow X-mode (SX) branches. No X-mode wave can propagate in the evanescent region. Additionally, the O-mode and X-mode do not propagate below the plasma frequency cutoff and the left-handed cutoff respectively. In summary, the dispersion diagram for a perpendicular or oblique propagating wave in a cold magnetized plasma and in the high frequency limit is illustrated by Figure 2.1.

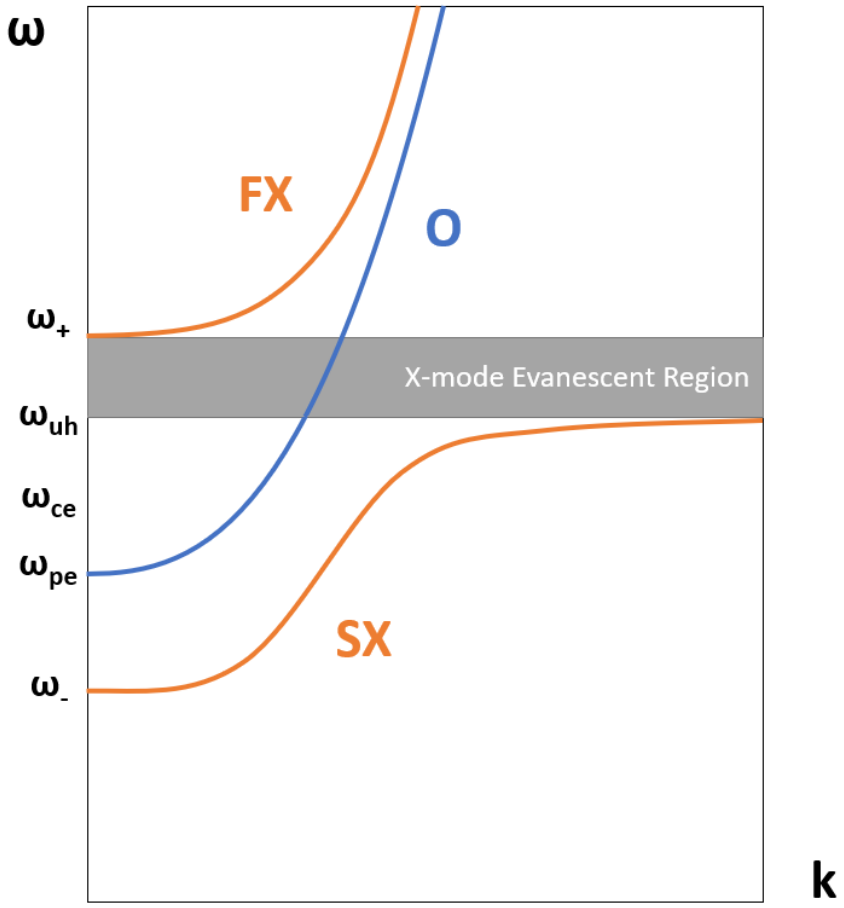

Figure 2.1: A typical dispersion diagram for a perpendicular or oblique propagating wave in a cold magnetized plasma. Only modes relevant to the high frequency limit are shown. The O-mode is represented by the blue curve while the X-mode is represented by the orange curves. The X-mode consists of two branches: Fast (FX) and Slow (SX). Between the right-handed cutoff and the upper hybrid resonance, there exist a evanescent region for the X-mode where wave propagation is prohibited. In addition, the O-mode and X-mode do not propagate below the plasma frequency cutoff and left-handed cutoff respectively. Note that the electron cyclotron frequency is chosen such that it is above the plasma frequency. Also note that $\omega_{\pm}$, ω_{uh}, and ω_{pe} are all dependent on the density and as such their positions relative to the resonances ($n\omega_{ce}$) can be different than what is shown here.

2.4 Warm Plasma

The cold plasma analysis described in the previous section is unable to account for waves absorption. Kinetic theory is needed in order to properly describe waves absorption. We introduce a general particle distribution function $f_s(\mathbf{r}, \mathbf{p})$ that exists in a six dimensional phase space where $\mathbf{r}$ and $\mathbf{p}$ denote the position and momentum vector respectively. In the absence of collisions, the particle phase space density is conserved when we follow a trajectory in phase space. Hence, we can write:

$$\frac{df_s(\mathbf{r}, \mathbf{v}, t)}{dt} \equiv \frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla f_s + \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_v f_s = 0 \tag{2.24}$$

which can be recognized as a collisionless Boltzmann equation. This equation is also frequently referred to as the Vlasov equation. Combining the Vlasov equation with Maxwell equations creates a set of equations called the Vlasov-Maxwell equations. The set of Vlasov-Maxwell equations forms the foundation for the analysis of a warm plasma. The full analysis and derivation for the description of a warm magnetized plasma are lengthy and complicated. They are beyond the scope of this master thesis. Interested readers are referred to more detailed texts for a full analysis and derivation [28] [29]. Here, we will merely present the resulting dielectric tensor for a warm magnetized plasma. We will also briefly describe some differences between a cold plasma and a warm plasma.

The dielectric tensor for a warm magnetized plasma (in the high frequency limit) is given by [11]:

$$\epsilon_{ij} = \delta_{ij} - \left(\frac{\omega_{pe}}{\omega}\right)^2 \sum_{n=-\infty}^{+\infty} \int d^3\bar{p} \, \frac{S_{ij}^{(n)}}{\frac{n\omega_{ce}}{\omega} + N_{\parallel}\bar{p}_{\parallel} - \gamma} \tag{2.25}$$

where $\bar{p} \equiv p/m_e c$, $\gamma = \sqrt{1 + \bar{p}^2}$ is the relativistic factor, and the integer number n denotes the electron cyclotron harmonics. Note that the summation is over all cyclotron harmonics but in practical implementation there will be a limit on the order of harmonics included. The tensor $S_{ij}^{(n)}$ is a Hermitian matrix and is given by:

$$S_{ij}^{(n)} = \begin{pmatrix} \bar{p}_\perp U \left(\frac{nJ_n}{b}\right)^2 & -i\bar{p}_\perp \frac{nJ_n J_n'}{b} & \bar{p}_\parallel U \frac{nJ_n^2}{b} \\ i\bar{p}_\perp \frac{nJ_n J_n'}{b} & \bar{p}_\perp U \left(J_n'\right)^2 & i\bar{p}_\parallel U J_n J_n' \\ \bar{p}_\parallel U \frac{nJ_n^2}{b} & -i\bar{p}_\parallel U J_n J_n' & \bar{p}_\parallel W J_n^2 \end{pmatrix} \tag{2.26}$$

where J_n is a shorthand for $J_n(b)$ which is a Bessel function of order n, $J_n' = \frac{dJ_n}{db}$, and $b = k_\perp \rho = N_\perp \bar{p}_\perp \frac{\omega}{\omega_{ce}}$ is the ratio of the Larmor radius ($\rho = v_\perp/\omega_{ce}$) to the perpendicular wavelength. The variable b is a measure of what is called as the finite Larmor radius (FLR) effects. The finite

value of b causes additional effects not found in the cold plasma analysis. Consequences of the FLR effects include the contribution of higher cyclotron harmonics to the dielectric tensor and the existence of a new set of electrostatic waves called Bernstein waves not found in cold plasma. The variables U and W are given by:

$$U = \frac{\partial f}{\partial \bar{p}_\perp} + \frac{N_\parallel}{\gamma}\left(\bar{p}_\perp \frac{\partial f}{\partial \bar{p}_\parallel} - \bar{p}_\parallel \frac{\partial f}{\partial \bar{p}_\perp}\right) \tag{2.27}$$

$$W = \frac{\partial f}{\partial \bar{p}_\parallel} - \frac{n\omega_{ce}}{\omega\gamma\bar{p}_\perp}\left(\bar{p}_\perp \frac{\partial f}{\partial \bar{p}_\parallel} - \bar{p}_\parallel \frac{\partial f}{\partial \bar{p}_\perp}\right) \tag{2.28}$$

where f refers to the electron distribution function which is still being kept general for now.

The warm plasma dispersion relation can still be cast in a biquadratic equation as in equation (2.20) for the cold plasma case. However, except when considering the lowest order approximation for the FLR effects, in general the coefficients themselves become a function of $N_\perp$. The solution then needs to be solved computationally using iterative methods. The computational method used to solve the warm plasma dispersion relation will be discussed in greater detail in later section dedicated to the TORAY code.

There are several notable differences that differentiate waves propagation in a warm plasma from a cold plasma. These are in addition to ability of the warm plasma analysis to account for wave absorption. First of all, we can notice from equation (2.25) that the condition for wave-particle or cyclotron resonance for a warm plasma is given by:

$$\omega = k_\parallel v_\parallel + \frac{n\omega_{ce}}{\gamma} \tag{2.29}$$

instead of only $\omega = \omega_{ce}$ from the cold plasma analysis. The most important thing to note is that higher order harmonics are also included in the equation. Next, the equation reveals two dominant broadening mechanism that will affect ECRH: Doppler broadening represented by the first term and relativistic broadening represented by the second term. Increase in temperature will increase both types of broadening while increase in parallel component of the wave number will increase the Doppler broadening. The cyclotron resonance interactions will mainly increase the energy of the resonant particles in the perpendicular direction. The second notable difference from a cold plasma is the existence of a group of electrostatic waves called the Bernstein waves due to additional solutions to the dispersion relation. Refer to Figure 2.2 for an illustration of these Bernstein waves adopted from [17]. It is possible for mode conversion to occur from the cold plasma waves into these Bernstein waves. One example that will be somewhat relevant for this project is the conversion from the X-mode into the electron Bernstein waves near the second harmonic. The third difference - which is the most relevant to this project - is the modification of the waves propagation near the cyclotron resonances. How significant this deviation is from the cold plasma depends on the plasma parameters involved. For present day fusion machines, this

deviation is unlikely to play an important role. More exact description on the waves propagation trajectory in a warm plasma requires the usage of ray tracing or beam tracing method. Figure 2.3 shows how the wave propagation can be affected [35].

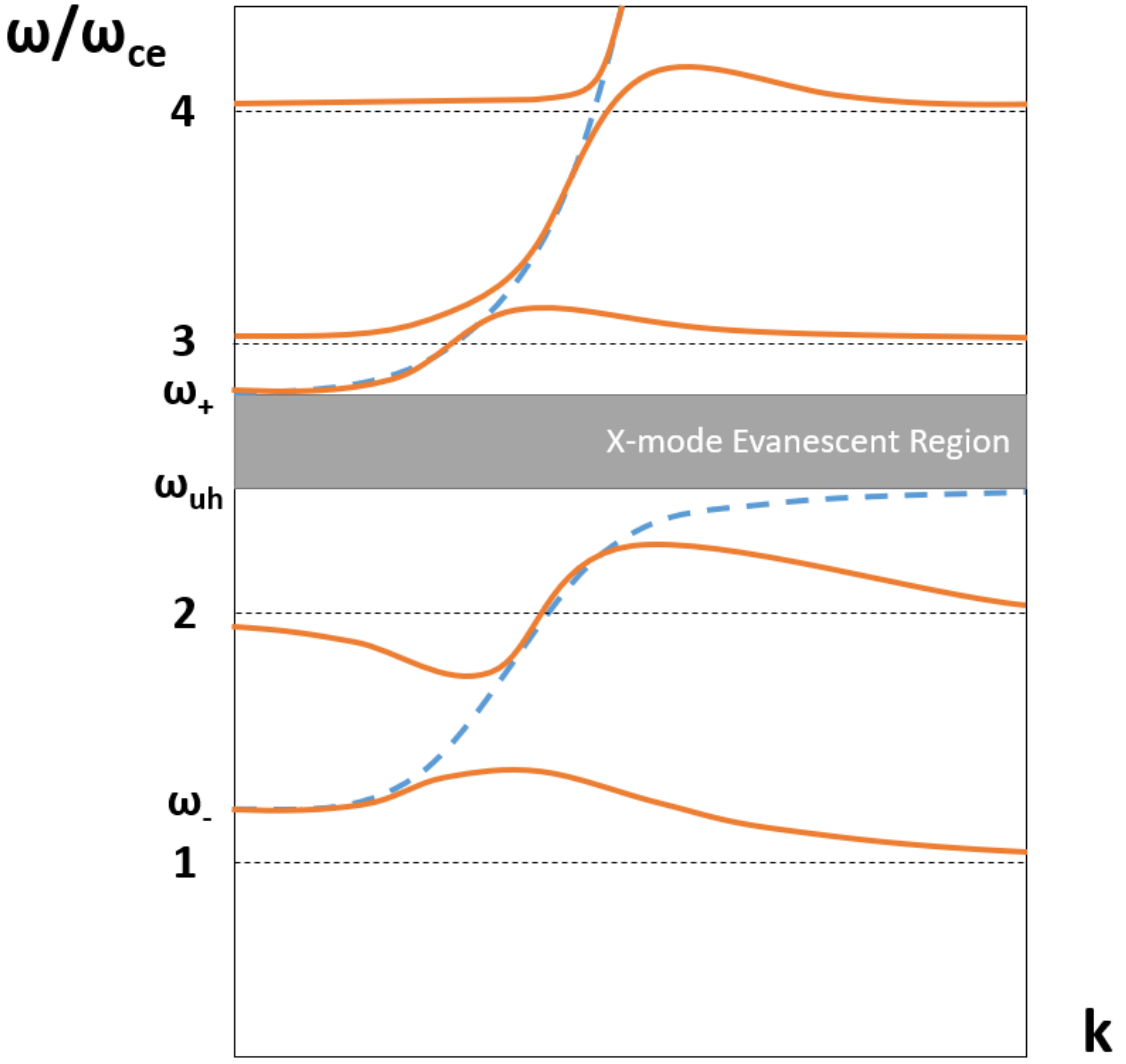

Figure 2.2: An illustration depicting the existence of warm plasma electron Bernstein modes in the electrostatic approximation (Orange) in addition to the cold plasma X-mode (Dashed blue). Adapted from [17].

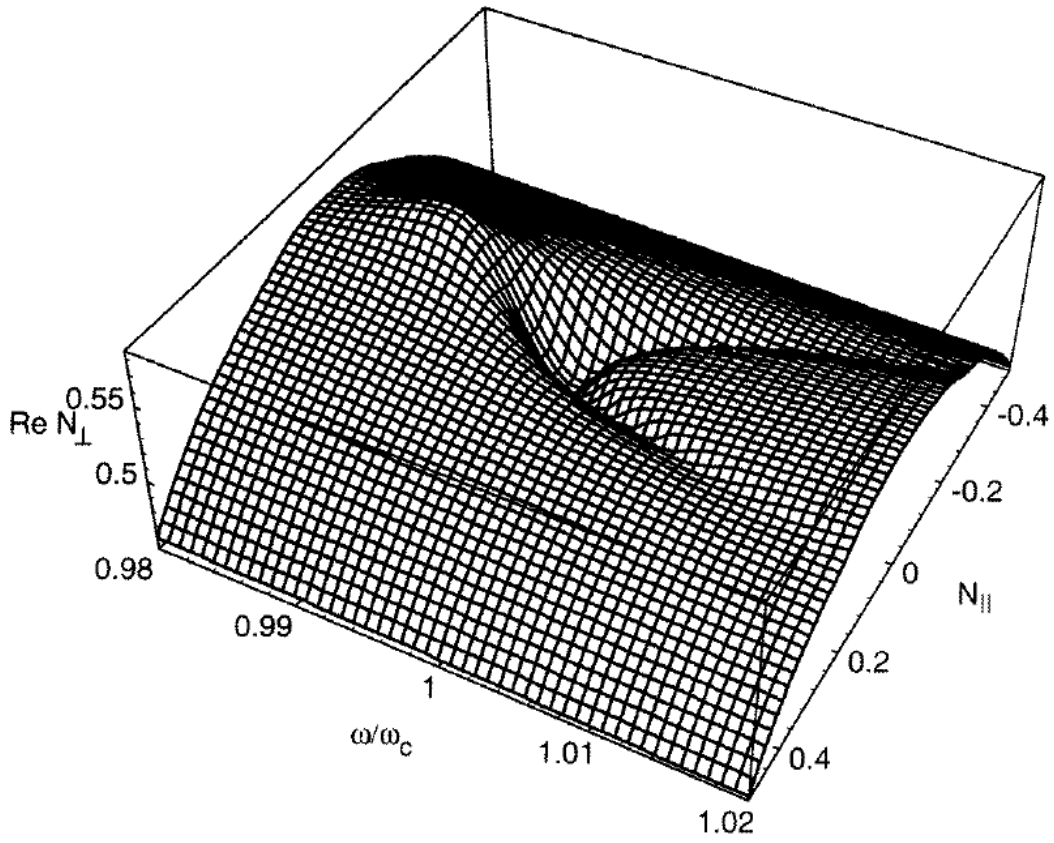

Figure 2.3: An illustration depicting how the (Real) $N_\perp$ varies as a function of $N_\parallel$ and ω/ω_{ce} for an O-mode wave propagating near the first harmonic cyclotron resonance of a warm plasma. The plasma parameters are $T_e = 1$ keV and $\omega_{pe}^2/\omega^2 = 0.7$. This illustration was generated using results from the ray tracing code TORAY. Source: [35].

Chapter 3

Electron Cyclotron Resonance Heating

In this chapter, we will describe in more detail - based on knowledge of waves propagation in plasma from the previous chapter - how the injected waves from the ECRH system are going to propagate in a tokamak plasma. Perhaps the most important question is of the accessibility of the EC waves. This is because we know that there are evanescent regions that prevent the EC waves from propagating further. We will briefly describe the basics of ECRH in a tokamak plasma in the first section of this chapter. The second section briefly describes the ray tracing method that is commonly used to describe the ECRH waves trajectory in a more exact manner. The third section describes the relevant routines of the ECRH ray tracing code TORAY/TORAYFOM.

3.1 Basics

ECRH is one of the external heating method applied to tokamak plasma. It allows localized heating and current drive owing to its small wavelength and its relatively narrow absorption region. It can also be used for perturbative heat experiments and for suppressing plasma instabilities. The most important drawback is that it requires a high frequency source that can deliver high power in a sufficiently long pulse. Another drawback is the fact that most of these sources (the Gyrotrons) only have a single fixed frequency. On present day tokamaks, the frequencies used are around the 100 GHz mark. In ITER, they will use a frequency of 170 GHz. It is convenient to know that a 28 GHz electron cyclotron wave will approximately have its first harmonic or fundamental resonance at 1 T. This means that in ITER, the fundamental resonance is around 6 T.

Evanescent regions determine the accessibility of the EC waves. To illustrate the accessibility aspect, it is sufficient to simply use the cold plasma analysis. We will also set the $N_\perp$ equals to zero

for this purpose. We start with the O-mode which only has one cutoff at the plasma frequency. In a tokamak, the magnetic field is inversely proportional to the major radius ($B \propto 1/R$). The density is higher towards the center of the magnetic axis. This lead to a picture of a poloidal cross-section as shown in Figure 3.1. Once the density is high enough to cause the plasma frequency to be equal to the injected waves frequency, the EC wave will be reflected. For O-mode, there is good accessibility when $\omega_{pe}^2/\omega_{ce}^2 < 1$. We can also easily calculate for the O-mode the density limit for a certain value of wave frequency:

$$n_{\text{limit}} = 1.24 \times 10^{20} \times f_{100\ \text{GHz}}^2\ \text{m}^{-3} \tag{3.1}$$

where the frequency is in units of 100 GHz. O-mode ECRH is almost always done at the fundamental resonance. There is not much reason to heat at higher harmonics since accessibility is reasonably good and absorption efficiency will decrease at higher harmonics. In ITER, the ECRH will be done at the fundamental O-mode. For the X-mode, the picture is a bit different than the O-mode. Recall that there is an evanescent region between the right-handed cutoff and the upper hybrid. If we heat at the fundamental resonance, the result will be similar to that of the left diagram of Figure 3.2. There is a crescent-like evanescent region blocking access to the absorption region if we inject waves from the low field side. As a result, we are only able to heat if we inject waves from the high field side (i.e. from the center of the tokamak). This is undesirable from a design point of view and thus X-mode heating is typically done at the second harmonic instead. The second harmonic heating is illustrated by the right diagram of Figure 3.2. Notice that it looks similar to the fundamental O-mode heating. In order to perform second harmonic X-mode heating at the same absorption region, the wave frequency must be doubled. This typically means a frequency above 200 GHz for tokamak operating at 4 T (on the magnetic axis) or above. Developing a frequency source that high with sufficient power and pulse duration will be a challenge in itself. This means that second harmonic X-mode heating is more practical to perform in smaller tokamak that operate at lower magnetic fields than in big tokamak like ITER. The advantage compared to the fundamental O-mode is the higher density limit of $\omega_{pe}^2/\omega_{ce}^2 < 2$. The absorption efficiency is also stronger than the fundamental O-mode.

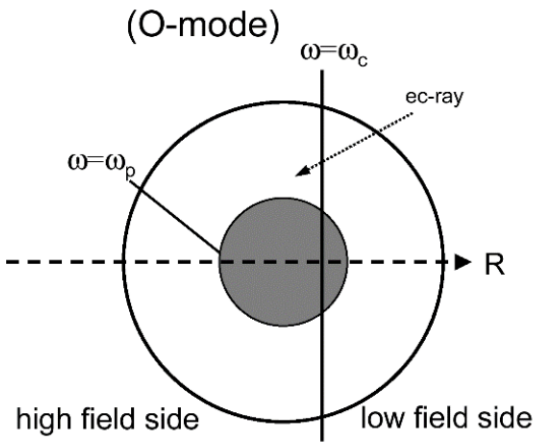

Figure 3.1: An illustration of the O-mode accessibility in a tokamak plasma. The shaded region represents the evanescent region where wave propagation is prohibited. The density is higher towards the center of the magnetic axis to the point that the plasma frequency becomes equal to the injected waves frequency. The vertical line denotes the absorption region. Source: [32].

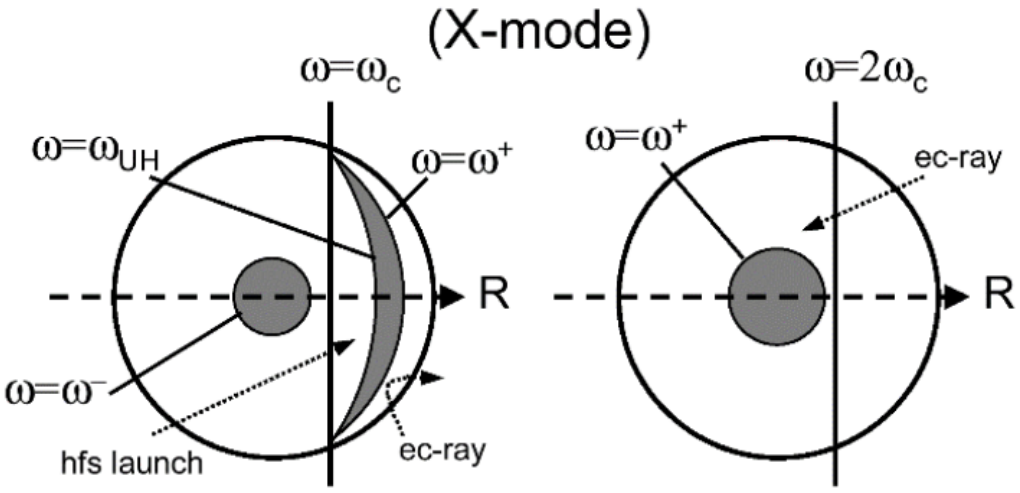

Figure 3.2: An illustration of the X-mode accessibility in a tokamak plasma for heating at the fundamental (Left) and second harmonic (Right) resonance. The shaded region represents the evanescent region where wave propagation is prohibited. Heating at the fundamental resonance is only possible from the high field side (i.e. from the center of the tokamak) because there is a crescent-like evanescent region blocking access from the low field side. Heating at the second harmonic is similar to the fundamental O-mode. The vertical line denotes the absorption region. Source: [32].

3.2 Ray Tracing

A more exact description of the EC waves trajectory in the tokamak plasma is possible through the ray tracing method. Ray tracing method is a method that approximates the propagating waves as a number of independent rays. This is often called the geometric optics approximation. The rays are advanced in small steps considering the local parameters of the medium. The local parameters determine the new direction of the rays at each steps. This allows us to approximately trace the trajectory of the actual waves.

The basis for ray tracing is a set of Hamiltonian equations:

$$\frac{\partial \mathbf{r}}{\partial s} = \frac{\partial D}{\partial \mathbf{k}} \tag{3.2a}$$

$$\frac{\partial \mathbf{k}}{\partial s} = -\frac{\partial D}{\partial \mathbf{r}} \tag{3.2b}$$

where $D(\omega, \mathbf{r}, \mathbf{k})$ is the local dispersion relation, $\mathbf{r}$ is the position of the ray, $\mathbf{k}$ is the local wave vector, and ∂s is an infinitesimal step of the ray. After obtaining the rays trajectory, power absorption along the trajectory can be calculated according to:

$$p_{abs}(s)ds = P_{ray}\ \alpha(s)\ \exp\left(-\int_{s_0}^{s} \alpha(s)ds'\right) ds \tag{3.3}$$

where $p_{abs}(s)$ is the power absorbed along the path, P_{ray} is power carried by a particular ray, and $\alpha(s)$ is the absorption coefficient which is proportional to the imaginary part of the perpendicular refractive index ($N_{\perp}^{''}$). In summary, for a three dimensional ray tracing, each ray is governed by a total of seven ordinary first order differential equations. Note that ray tracing is unable to account for diffraction effects and breaks downs at the focus of the wave beam. Beam tracing method is able to solve these deficiencies but requires solving second order differential equations.

Regarding the dispersion relation, for a cold plasma this is simply provided by solving the bi-quadratic equation. For the warm plasma however, Westerhof [35] discovered that for the purpose of ray tracing we should use the following (real) dispersion relation:

$$D' = k^2 - k_{\perp}^{'2}(\mathbf{r}, k_{\parallel}) - k_{\parallel}^2(\mathbf{r}, \mathbf{k}) = 0 \tag{3.4}$$

where $k_{\perp}^{'}$ and $k_{\parallel}$ are functions of spatial coordinates $\mathbf{r}$ and wave vector $\mathbf{k}$. $k_{\perp}^{'}$ is the real part of the solution of the full warm plasma dispersion relation which is obtained by solving the biquadratic equation. This has the effect of including the anti-hermitian part of the dispersion tensor which is not negligible compared to the hermitian part near the resonances. Using Equation (3.4) for warm plasma ray tracing is necessary in order to avoid anomalous behavior near the cyclotron resonances. Another method is to diagonalize the dispersion tensor and use the real part of the eigenvalue for ray tracing [30].

3.3 TORAYFOM

TORAYFOM is a 3D ray tracing code for solving the EC waves propagation and absorption in a tokamak. It can used for the purpose of both ECRH or ECCD. In this project, the focus is solely on ray tracing for the purpose of ECRH. A brief history of the code is as follows. The original code is called RAYS by Batchelor and Goldfinger. The version called TORAY is used by Kritz to treat ECRH in tokamak. TORAYFOM refers specifically to the version of TORAY maintained at the Dutch Institute for Fundamental Energy Research (DIFFER). TORAYFOM is the version used for this project but for convenience we often refer to it simply as TORAY.

Over the years, many different authors have contributed to different parts of the code. The current version of TORAY is a quite extensive code consisting of many subroutines. We will not describe all the subroutines in this report. Figure 3.3 shows the relevant subroutines for this project. The ODE subroutine solves the set of ordinary differential equations for the ray tracing (Equation3.2). It requires spatial coordinates, wave numbers, energy, and their derivatives with respect to the ray position as input. These inputs are provided by the subroutine DERV. DERV in turn requires the values of the real and imaginary part of the perpendicular refractive index. These values are provided by the subroutines TWOCMPC and TWOCMPW depending if we choose the cold plasma or the warm plasma respectively. These two subroutines are actually where the plasma dispersion relation is evaluated. Therefore, they require the dielectric tensor of the plasma in order to function. Whereas the calculation of the dielectric tensor is relatively straightforward for TWOCMPC, the calculation for the TWOCMPW is significantly more involving. There are two subroutines options to calculate the (fully relativistic) dielectric tensor for TWOCMPW: through the default subroutine FREPSLN (NLRELA=1) or through the subroutine by Farina [12] (NLRELA=3). We also show the subroutine TWOCMPWNN developed in this project. This subroutine houses the neural network regression of the results by TWOCMPW and is expected to be able to take the role of TWOCMPW but with greater speed. As a rough approximation, the TWOCMPW subroutine time for a single execution is on the order of 10^{-4} s on average when using FREPSLN to compute the dielectric tensor. When using the Farina method to compute the dielectric tensor, the time for a single execution of TWOCMPW is on the order of 10^{-5} s on average. A more appropriate comparison however would be the total time needed for TORAY to successfully complete its run. This computational time is harder to approximate beforehand as it is dependent on more factors including the system architecture. We will provide these total computational time later in the results section.

As TWOCMPW is needed to generate the dataset for a neural network regression, we will look into it in more details. We first reproduce the expressions for the warm dielectric tensor from the previous chapter for convenience. The dielectric tensor for a warm magnetized plasma (in the

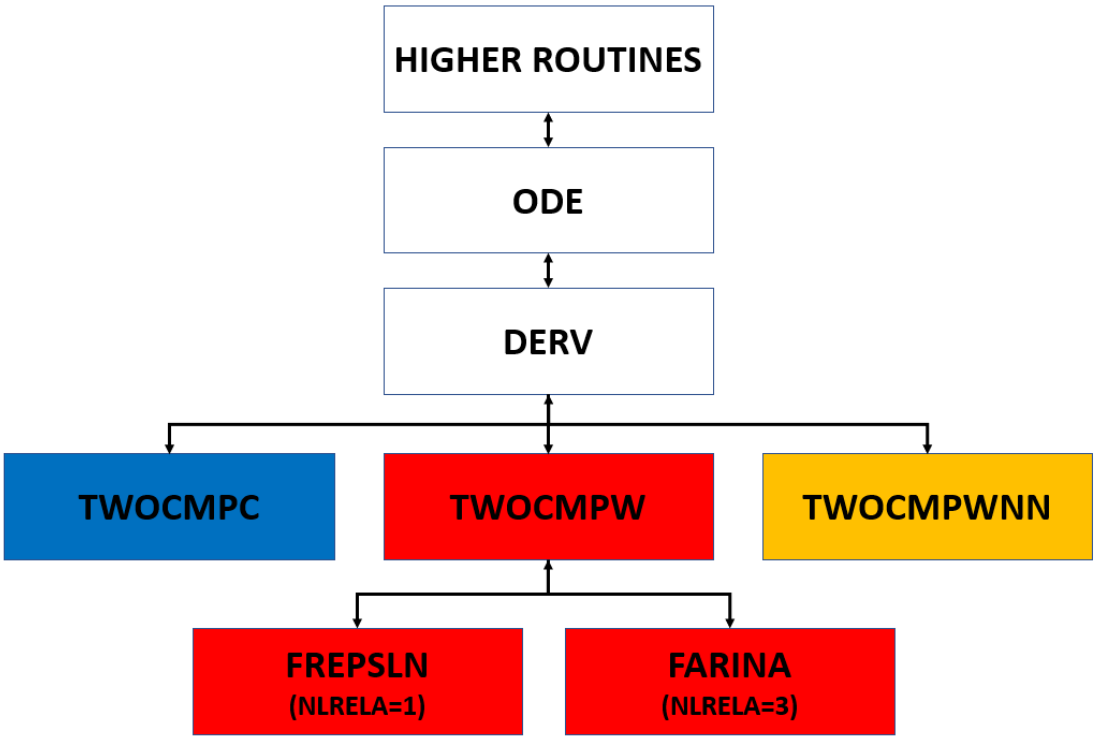

Figure 3.3: A flowchart of the relevant TORAY subroutines.

high frequency limit) is given by:

$$\epsilon_{ij} = \delta_{ij} - \left(\frac{\omega_{pe}}{\omega}\right)^2 \sum_{n=-\infty}^{+\infty} \int d^3\bar{p}\, \frac{S_{ij}^{(n)}}{\frac{n\omega_{ce}}{\omega} + N_{\|}\bar{p}_{\|} - \gamma} \tag{3.5}$$

The tensor $S_{ij}^{(n)}$ is given by:

$$S_{ij}^{(n)} = \begin{pmatrix} \bar{p}_{\perp} U \left(\frac{nJ_n}{b}\right)^2 & -i\bar{p}_{\perp} \frac{nJ_n J_n^{'}}{b} & \bar{p}_{\|} U \frac{nJ_n^2}{b} \\ i\bar{p}_{\perp} \frac{nJ_n J_n^{'}}{b} & \bar{p}_{\perp} U \left(J_n^{'}\right)^2 & i\bar{p}_{\|} U J_n J_n^{'} \\ \bar{p}_{\|} U \frac{nJ_n^2}{b} & -i\bar{p}_{\|} U J_n J_n^{'} & \bar{p}_{\|} W J_n^2 \end{pmatrix} \tag{3.6}$$

The variables U and W are given by:

$$U = \frac{\partial f}{\partial \bar{p}_{\perp}} + \frac{N_{\|}}{\gamma}\left(\bar{p}_{\perp}\frac{\partial f}{\partial \bar{p}_{\|}} - \bar{p}_{\|}\frac{\partial f}{\partial \bar{p}_{\perp}}\right) \tag{3.7}$$

$$W = \frac{\partial f}{\partial \bar{p}_\parallel} - \frac{n\omega_{ce}}{\omega\gamma\bar{p}_\perp}\left(\bar{p}_\perp \frac{\partial f}{\partial \bar{p}_\parallel} - \bar{p}_\parallel \frac{\partial f}{\partial \bar{p}_\perp}\right) \tag{3.8}$$

This rather complex expressions for the warm dielectric tensor will be computed by the subroutine FREPSLN [36]. First of all, we assume the electron distribution function to behave as $f \sim e^{-\mu\gamma}$ (also known as the Maxwell-Juttner distribution) where $\mu = m_e c^2/k_B T_e$ with k_B as the Boltzmann constant and T_e as the electron temperature. In other words, for large momenta we assume the distribution falls off to zero. We then perform a series expansion to evaluate the Bessel functions $J_n(b)$ and their derivatives as follow:

$$J_n(b) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m(m+n)}\left(\frac{b}{2}\right)^{2m+n} \tag{3.9}$$

where m denotes the order of the expansion. The number of terms to keep is set by the code parameter NTERM in TORAY. For the dataset generation, we took NTERM = 7 which is a rather high order in order to ensure accuracy. For ray tracing runs, NTERM is set to 3. The integral in Equation 3.5 can be separated into two as follows (we exploited toroidal symmetry):

$$2\pi \int_{-\infty}^{+\infty} d\bar{p}_\parallel \int_{\gamma^-}^{+\infty} d\gamma \frac{\gamma S_{ij}^{(n)}(\bar{p}_\parallel, \gamma)}{\gamma^{(n)} - \gamma} \tag{3.10}$$

The γ integral has a singularity at $\gamma = \gamma^{(n)}$ and thus is needed to be evaluated by its principal value plus a resonant contribution according to Landau integration contour. This resulted in the dielectric tensor having two parts: the Hermitian part and the anti-Hermitian part which determines wave absorption. The principal value of γ integral can be written as:

$$\begin{aligned} PV \int_{\gamma^-}^{+\infty} d\gamma \frac{\gamma S_{ij}^{(n)}(\bar{p}_\parallel, \gamma)}{\gamma^{(n)} - \gamma} &= \int_{\gamma^-}^{+\infty} d\gamma \frac{\gamma S_{ij}^{(n)}(\bar{p}_\parallel, \gamma) - \gamma^{(n)} S_{ij}^{(n)}(\bar{p}_\parallel, \gamma^{(n)}) e^{-\mu(\gamma-\gamma^{(n)})}}{\gamma^{(n)} - \gamma} \\ &+ PV \int_{\gamma^-}^{+\infty} d\gamma \frac{\gamma^{(n)} S_{ij}^{(n)}(\bar{p}_\parallel, \gamma^{(n)}) e^{-\mu(\gamma-\gamma^{(n)})}}{\gamma^{(n)} - \gamma} \end{aligned} \tag{3.11}$$

The second term has the analytical solution of:

$$PV \int_{\gamma^-}^{+\infty} d\gamma \frac{e^{-\mu(\gamma-\gamma^{(n)})}}{\gamma^{(n)} - \gamma} = \mathrm{Ei}(-\mu(\gamma - \gamma^{(n)})) \tag{3.12}$$

where Ei means an exponential integral. The exponential integral is numerically solved using the subroutine CALCEI. The first term of Equation 3.11 can be calculated using the Gaussian quadrature method. More specifically, since the distribution function behaves as $\sim e^{-\mu\gamma}$, we use the Gauss-Laguerre form with weights w_i and nodes γ_i as follows:

$$\int_{-\gamma}^{+\infty} d\gamma f(\gamma) e^{-\mu\gamma} = \sum_i w_i f(\gamma_i) \tag{3.13}$$

The number of nodes is set by the parameter NGM. In the code, it is able to take values in the range of 1 to 64.

In order to evaluate the $\bar{p}_\parallel$ integral, we first determine the range of $\bar{p}_\parallel$ that fulfills the resonance condition ($\gamma = \sqrt{1+\bar{p}_\parallel^2} = \gamma^{(n)}$). This is given by:

$$\bar{p}_\parallel^{\pm} = \frac{n\omega_{ce}N_\parallel}{\omega(1-N_\parallel^2)} \pm \sqrt{\left(\frac{n\omega_{ce}N_\parallel}{\omega(1-N_\parallel^2)}\right)^2 - \frac{\omega^2 - n^2\omega_{ce}^2}{\omega^2(1-N_\parallel^2)}} \tag{3.14}$$

There are two cases depending on the value of $N_\parallel$: one where $|N_\parallel| < 1$ and one where $|N_\parallel| > 1$. In the case of $|N_\parallel| < 1$, real solutions only exist when:

$$\omega < \frac{n\omega_{ce}}{\sqrt{1-N_\parallel^2}} \tag{3.15}$$

For the case of $|N_\parallel| \geq 1$ or $N_\parallel \leq -1$ and $N_\parallel \geq 1$, there will always be real solutions. We can now define up to three integration regions based on the value of $N_\parallel$ and the existence of real solutions. When $|N_\parallel| < 1$ and real solutions exist, we can define a non-resonant region I $(-\infty, \bar{p}_\parallel^-]$, a resonant region II $[\bar{p}_\parallel^-, \bar{p}_\parallel^+]$, and a non-resonant III $[\bar{p}_\parallel^+, +\infty)$. When $|N_\parallel| < 1$ and real solutions do not exist, then we only have one non-resonant region I over the entire domain $(-\infty, +\infty)$. For the case of $|N_\parallel| > 1$, we only need to consider for $N_\parallel \leq -1$ and $N_\parallel \geq 1$ since there will always be real solutions. For $N_\parallel \leq -1$, we divide into a resonant region II $(-\infty, \bar{p}_\parallel^-]$ and a non-resonant region III $[\bar{p}_\parallel^-, +\infty)$. For $N_\parallel \geq 1$, we divide into a non-resonant region I $(-\infty, \bar{p}_\parallel^+]$ and a resonant region II $[\bar{p}_\parallel^+, +\infty)$. The $\bar{p}_\parallel$ integral can also be evaluated using the Gaussian quadrature method. The code parameter which set the number of nodes is NPM with a range of value from 1 to 32. Different cases will use different forms of the Gaussian quadrature. For cases with high values of μ, the integrand will have a factor of $\sim e^{-\frac{1}{2}\mu\bar{p}_\parallel^2}$ and is appropriately solved using the Gauss-Hermite:

$$\int_{-\infty}^{+\infty} d\bar{p}_{\parallel} f(\bar{p}_{\parallel}) e^{-\frac{1}{2}\mu\bar{p}_{\parallel}^2} = \sum_i w_i f(\bar{p}_{\parallel,i}) \tag{3.16}$$

It is sufficient to use only Gauss-Hermite if all of its nodes fall inside the resonant region II. If not and in the case of $N_{\parallel} \leq 1/\mu$, then the integration for region II must be done using Gauss-Legendre:

$$\int_{\bar{p}_{\parallel}^-}^{\bar{p}_{\parallel}^+} d\bar{p}_{\parallel} f(\bar{p}_{\parallel}) = \sum_i w_i f(\bar{p}_{\parallel,i}) \tag{3.17}$$

while region I and III is done using Gauss-Hermite. For cases with large values of $|N_{\parallel}|$, the integrand factor behaves as $\sim e^{-\mu N_{\parallel}\bar{p}_{\parallel}}$. If the nodes of the Gauss-Laguerre quadrature falls inside region II, we can use it for integration region I and II or II and III depending on the value of $N_{\parallel}$. If not, Gauss-Legendre is used for region II while Gaussian-Laguerre is used for region I or III.

The final integral concerns the anti-Hermitian part of the dielectric tensor and is given by:

$$\epsilon_{ij}^{aH} = -i\pi \left(\frac{\omega_{pe}}{\omega}\right)^2 \sum_{n=-\infty}^{+\infty} 2\pi \int d\bar{p}_{\parallel} \gamma^{(n)} S_{ij}^{(n)}(\bar{p}_{\parallel}, \gamma^{(n)}) \tag{3.18}$$

where the integration is carried out according to the resonant regions defined previously. The integration factor behaves as $\sim e^{-\mu N_{\parallel}\bar{p}_{\parallel}}$ if the resonant region is in a large enough range of $\bar{p}_{\parallel}$. Thus, the Gauss-Laguerre quadrature is used if all of the nodes are inside the resonant region. If not, the Gauss-Legendre quadrature is used instead. The number of nodes for the anti-Hermitian integral is set through the parameter NAM and has a range of 1 to 64.

The subroutine FREPSLN computes the warm plasma dielectric tensor but needs $N_{\perp}$ as one of its input. Since the coefficients for the biquadratic equations are dependent on the dielectric elements, they are also dependent on $N_{\perp}$ in general. Since $N_{\perp}$ is also the output that we want to obtain, we must solve the problem iteratively. We provide an initial guess of $N_{\perp}$ and solve iteratively for:

$$A(N_{\perp}^{(i)})N_{\perp}^{(i+1)^4} + B(N_{\perp}^{(i)})N_{\perp}^{(i+1)^2} + C(N_{\perp}^{(i)}) = 0 \tag{3.19}$$

where i is the step of the iteration. Convergence is achieved when the difference between the new and old value is less or equal than a certain value of accuracy. This accuracy value is set using the parameter ACCUR. The maximum number of iterations is set through the parameter MAXIT. This iterative method could fail if there are two nearby solution roots for the dispersion relation. As an example, this could happen near the second harmonic X-mode for certain values of density because of the existence of Bernstein waves nearby.

The computation of the elements of dielectric tensor can be done in a more compact form as illustrated by Farina [12]. This resulted in a faster computational time. In this method, the harmonic number is taken to be equal to the number of terms in the FLR expansion. For the Hermitian part, there is an additional constraint of taking the FLR expansion only up to the third order as a compromise between accuracy and speed.

Last but not least, the wave absorption coefficient α is computed using the imaginary part of the obtained perpendicular refractive index ($N_{\perp}^{''}$). It is given by:

$$\alpha = 2\ \sin(\beta) N_{\perp}^{''} \frac{\omega}{c} \tag{3.20}$$

where β is the angle between the ray step ($d\mathbf{s}$) and the external magnetic field ($\mathbf{B}$).

Chapter 4

Neural Network

Before we discuss about neural network, it is perhaps worthwhile to say several words on machine learning in general. At its core, machine learning algorithms can be considered as algorithms for fitting an arbitrary non-linear function. This might seem unrelated to what we often heard machine learning does: classifying images, translating languages, playing chess, etc. But most processes can be thought of as an input-output function/mapping. As a result, machine learning's ability to approximate arbitrary non-linear function is very powerful and can be used to solved multitude of problems (problems that most often are unfeasible to solve using traditional computer algorithms). Effectively, machine learning algorithms can make predictions based on data that we gave to them similar to how humans learn from examples. Hence, machine learning is a subset of the field of artificial intelligence.

Artificial Neural Network (NN) or simply Neural Network (NN) is a class of machine learning tools inspired by the biological neurons in our brain. Much like the biological neural network in the brain, the artificial neural network consists of nodes or neurons interconnected with each other. There are different types of neural network with different level of complexity. In this project, we are using a Feedforward Neural Network (FFNN) which is arguably the simplest type of neural network. FFNN is sufficient for solving the regression problem encountered in this project. In this chapter, we will briefly describe the basics of neural network. Example of reference texts for a primer on neural network and machine learning are: [24], [21], [3], and [19] .

4.1 The Universal Approximation Theorem

Before delving into the details of neural network, it is worthwhile to briefly mention a theorem that is crucial for explaining why neural networks work: The Universal Approximation Theorem. The Universal Approximation Theorem ensures that for any continuous function there exists a

neural network that is able to approximate the function up to an arbitrary accuracy. Without this universality theorem, there is no reason to think that a particular function can be approximated using neural networks. The mathematical proof for this theorem is outside the scope of this thesis. Readers interested in the proof should consult the relevant papers such as: [6] and [20]. Note that this theorem only ensures existence. It does not say how to obtain such network or how feasible it is to obtain such network. Therefore, solving practical problems using neural networks often becomes a problem of finding the right tools and techniques to obtain a sufficiently good network in a reasonable amount of time.

4.2 Neurons

Neurons are the basic building blocks of neural network. Mathematically, a single neuron is expressed as:

$$y = f\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{4.1}$$

where w_i are the weights, x_i are the inputs, b is the bias, f is the activation function, and y is the output of the neuron. We can consider the weights as a measure of the importance of an input. If the weight of an input is zero, then that input is simply removed from the equation. Biases can be considered as thresholds that determine how easy it is for the neuron to transmit its signal. A high bias will be more likely to cause the neuron to transmit regardless of the weighted sum. The weighted sum plus bias is inserted into a non-linear activation function. Non-linearity is needed in order for the neural network to be able to approximate non-linear functions. A graphical illustration of a single neuron is depicted in Figure 4.1.

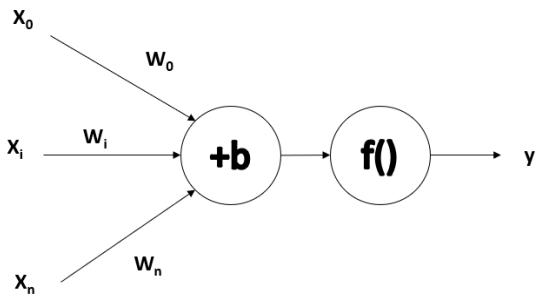

Figure 4.1: A graphical illustration of a single neuron.

4.3 Activation Function

The choice of activation functions are crucial for the ability of neural networks to approximate non-linear continuous functions. A non-linear activation function is required if we want to approximate non-linear continuous functions which is the case for most of practical problems. Traditionally, the non-linear activation function has to be bounded (i.e. saturates as its input approaches $+\infty$ and $-\infty$) and monotonically increasing [31]. However, recent research has shown that they need not be bounded [27]. Many non-linear activation functions have been developed over the years. For this thesis, we will only consider two: Tanh and ReLU.

4.3.1 Tanh

This activation function as its name suggested is simply the function $\tanh(x)$:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{4.2}$$

It has an "S" shape and is centered at the origin (see Figure 4.2). Note that it fulfills the bounded and monotonically increasing requirements. Another function similar to the tanh is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

In fact, tanh is merely a scaled sigmoid function centered at the origin. In practice, tanh is almost always used over sigmoid because a zero-centered function is found to be better for the purpose of training a neural network. Until relatively recently, tanh is the go-to activation function for training neural networks.

4.3.2 ReLU

ReLU stands for Rectified Linear Unit and is given by:

$$f(x) = \max(0, x) \tag{4.4}$$

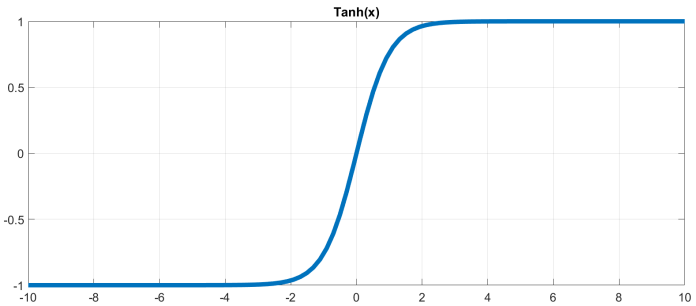

Figure 4.2: A tanh function.

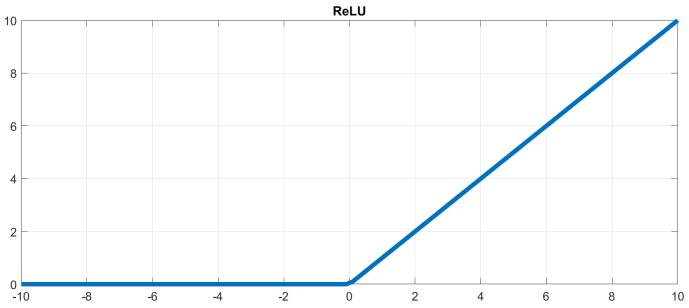

Figure 4.3: A relu function.

As its name suggested, it is a linear function that rectifies all its negative values to zero (see Figure 4.3). Note that this is an unbounded function and thus does not fulfill the traditional requirements. Relu has recently replaced tanh as the go-to activation function for training deep neural networks. Deep neural networks are typically used for image and speech recognition. Its effectiveness compared to tanh for shallow neural networks is less clear however.

4.4 Feedforward Neural Network

A single neuron by itself is not able to accomplish much. The next step of building a neural network is to connect multiple neurons with each other in a particular way. Figure 4.4 illustrates a typical FFNN. There are 2 inputs that ended up producing 1 output. We map them using a FFNN with 2 hidden layers consisting of 4 neurons or nodes each (2x4) for a total of 8 nodes. The outputs of the first hidden layer are taken as the inputs for the second hidden layer. The connecting lines can be considered as the weights. Each of the hidden layer nodes and the output node have their own biases. Typically the activation function is the same for all the hidden nodes. Note that the entire network "flows" forward as there is no connections between nodes in the same layer and no looping connections to previous layers. Hence the name feedforward neural network. The number of inputs and outputs are usually determined by the problem. The size of the hidden layers on the other hand is fully determined by the user.

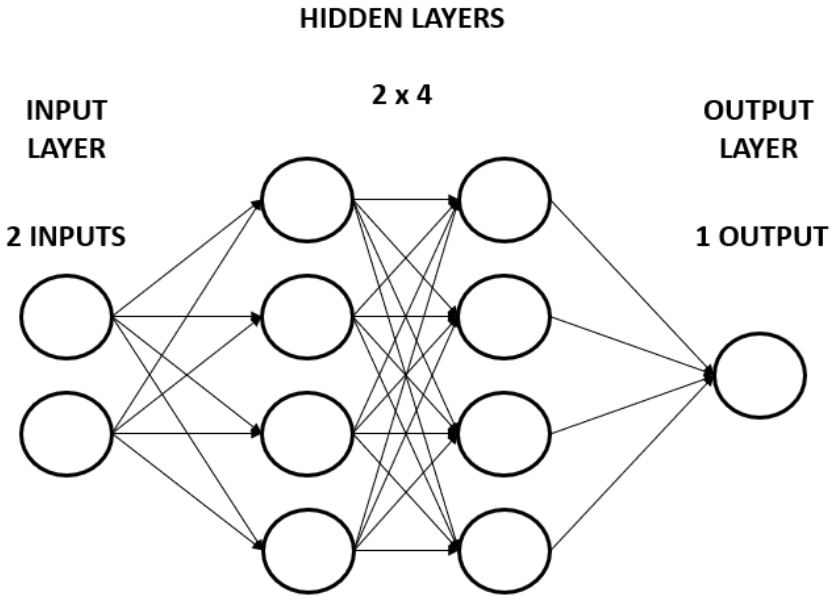

Figure 4.4: A typical pictorial representation of a feedforward neural network. Here, it maps 2 inputs onto 1 output. There are 2 hidden layers with 4 neurons or nodes each (2x4) for a total of 8 nodes. The connecting lines can be considered as the weights. Each of the hidden layers nodes and the output node have their own biases. Typically the activation function is the same for all the hidden nodes.

4.5 Training

Training a neural network means optimizing the weights and biases in order to obtain the best possible fit of the function we are approximating. The weights and biases are the free parameters that will be "learned" by the neural network. In order to optimize the weights and biases, we need an algorithm that is able to determine the effects of changing the weights and biases on the quality of the regression. We also need an algorithm that can iteratively converge on a set of weights and biases that will provide the best regression. Generally speaking, the two algorithms are always combined together. In more details, we have:

1. **Backpropagation algorithm**
   This algorithm computes the partial derivatives of a defined loss function with respect to the weights and biases. The loss function represents the general quality of the fit. Smaller value of the loss function means a better quality fit. Backpropagation got its name by the way it computes the derivatives: by first computing the final output given certain inputs (forward pass) and then computing all the derivatives backwards using the chain rule (backward pass). We will not describe the mechanism in detail here. Readers are referred to relevant texts such as [26] or [24] for more details. We will simply briefly describe the loss functions considered for this work in a short while (subsection 4.5.1).

2. **Optimizer/Learning Algorithm**
   The optimizer or the learning algorithm determines how the weights and biases will be updated for the next iteration. They aim to minimize the loss function. They are dependent on the derivatives computed by the backpropagation algorithm. Various optimizers have been developed over the years. We will briefly describe only two (subsection 4.5.2): the simplest algorithm called gradient descent [24] and a presently popular one called Adam [22].

Training a neural network requires the user to provide a dataset or examples. In general, the full dataset is divided into a training set and a test set (sometimes there is also a validation set). Training is only done on the training set. The test set is used to gauge the ability of the neural network to generalize outside of the training set.

4.5.1 Loss Function

For a regression problem, a common loss function to use is the Mean Squared Error (MSE):

$$L_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f_i - y_i)^2 \tag{4.5}$$

where f_i is the neural network output and y_i is the actual true output. It is also common to add a regularization term to the actual loss function. Regularization is added in order to tackle a problem called overfitting. When overfitting occurs, the neural network loses its ability to generalize beyond the dataset it was trained upon. This is generally due to the high number of free parameters in a neural network. In this work, we only consider the use of the L2 regularization where all of the squared-weights are summed:

$$L_{L2} = \frac{\lambda}{2} \sum_{w} w^2 \tag{4.6}$$

with λ the regularization strength. This regularization will heavily penalize large weights. Large weights will generally add to the complexities of the regression. The analogy would be like performing a high-order polynomial regression on a noisy experimental data instead of a linear regression. The total loss function we will be using is the sum of the MSE and the L2 regularization:

$$L_{Total} = L_{MSE} + L_{L2} \tag{4.7}$$

4.5.2 Optimizer

Gradient Descent

The simplest optimizer algorithm is the gradient descent. Suppose we have a vector of free parameters $\mathbf{x}$ (either weight or bias) that will be learned by the neural network. The update rule for the next iteration step according to gradient descent will be:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta \nabla L_i \tag{4.8}$$

where ∇L_i is a vector containing the corresponding partial derivatives of the loss function with respect to the free parameters and η is the learning rate tunable by the user (a hyperparameter). We can imagine the optimization process as trying to roll a ball into the deepest valley (i.e. the global minimum) of the loss function surface. Since derivatives point to direction of increasing values, we put a negative sign to the second term of Equation 4.8 to guide the ball to a general direction of decreasing values. However, in general the loss function surface is non-convex [4] which can cause difficulties for the gradient descent to converge to a solution. For example, at saddle points the derivatives may be too small causing the update to get "stuck" if the learning rate is unable to compensate.

Note that because the loss function is an average over all training examples (Equation 4.6), the derivative ∇L also need to be averaged over the individual derivatives of each training examples. In practice, this is too time consuming to perform. As an alternative, people instead approximate the true value of ∇L by averaging over a random smaller subset batch (mini-batch) of the full training examples. This method is called stochastic gradient descent (SGD) and is not dissimilar to how

survey or poll sampling works. If the size of the mini-batch is large enough, we can expect that the mini-batch ∇L will be a good approximation of the full training set ∇L. There is no general rule regarding the appropriate size of the mini-batch yet. For now, it needs to be determined in a case-by-case basis. But in general, smaller size leads to quicker training while larger size leads to better approximation of the true gradient. Note that SGD is in general applicable to other optimizers not only gradient descent. We will also employ training by mini-batches in this work.

Adam

Adam (Adaptive moment estimation) [22] was conceived relatively recently in 2014 by Kingma et al and has since become widely used for training neural networks. Interested readers are referred to the original paper for the full description. Here, we will merely cite the update rules and briefly explain some of the features. The update rules of Adam are:

$$\mathbf{m}_{i+1} = \beta_1 \mathbf{m}_i + (1 - \beta_1)\nabla L_i \tag{4.9a}$$

$$\mathbf{mt}_{i+1} = \mathbf{m}_{i+1} / \left(1 - \beta_1^{i+1}\right) \tag{4.9b}$$

$$\mathbf{v}_{i+1} = \beta_2 \mathbf{v}_i + (1 - \beta_2)(\nabla L_i)^2 \tag{4.9c}$$

$$\mathbf{vt}_{i+1} = \mathbf{v}_{i+1} / \left(1 - \beta_2^{i+1}\right) \tag{4.9d}$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta\ \mathbf{mt}_{i+1} / (\sqrt{\mathbf{vt}_{i+1}} + \epsilon) \tag{4.9e}$$

where η, β_1, β_2, and ϵ are all hyperparameters tunable by the user which have default values of 0.001, 0.9, 0.999, and 10^{-8} respectively. The hyperparameter η is the same learning rate as in gradient descent. The hyperparameters β_1 and β_2 control the exponential decay rates of the moving averages of the gradient (m) and the squared gradient (v). The hyperparameter ϵ serves to avoid division by zero. Note that $(\nabla L_i)^2$ and $\sqrt{\mathbf{vt}_{i+1}}$ are element-wise operations.

The first thing to notice is that unlike gradient descent, Adam does not directly update the trainable parameters using the derivatives. It updates the parameters using the variable $\mathbf{m}$ which is analogous to velocity. The parameters ($\mathbf{x}$) then are analogous to position. To resume our analogy of a ball rolling down a surface, this means that in Adam the virtual ball behaves more like a real physical ball. Second, the effective learning rate in Adam is modified by the denominator in Equation 4.9e. If the derivatives become small, then the effective learning rate is increased and vice versa. As a result, Adam has less risk of becoming "stuck" in saddle points or "bad" local minima compared to gradient descent. Furthermore, this modification of the learning rate is applied per parameters. Third, Equations 4.9b and 4.9d are bias correction mechanisms. Since the vectors of $\mathbf{m}$ and $\mathbf{v}$ are initialized at zero, they will be biased at zero especially during initial iterations.

In this work, we used the Adam optimizer with the default values of hyperparameters for

all neural network training. We used Adam because it is presently considered to be the default optimizer to use for training neural networks [25].

Chapter 5

Methodology

5.1 Dataset Generation

We have decided to perform neural network regression only on the O-mode for this work. The reason is that the O-mode with a single branch is simpler to fit than the X-mode with two branches. Furthermore, there are numerical problems near the X-mode second harmonic due to nearby Bernstein modes. The iterative algorithm used to compute the perpendicular refractive index could fail due to the presence of multiple solutions. We leave the X-mode regression for future work.

The dataset generation are done using TORAY warm plasma routine (TWOCMPW) with the dielectric tensor determined by the Farina routine. We found that the the Farina routine is more numerically stable than the FREPSLN routine. The warm plasma routine dataset consists of 4 inputs and 2 outputs. The inputs are $X = \left(\frac{\omega_{pe}}{\omega}\right)^2$, $Y = \frac{\omega_{ce}}{\omega}$, electron temperature ($T_e$), and the parallel refractive index ($N_\parallel$). X and Y can be seen as normalized electron density and normalized magnetic field respectively. The range of these inputs are:

1. X

   From 0.01 to 0.90 with all multiples of 0.05 in between as the interval points.

2. Y

   From 0.3 to 2.0. Standard interval is 0.01 but is increased to 0.001 (or 0.0001 for $T_e = 1$ keV) near the first and second harmonic if the features are sharp enough. More specifically, higher resolution is used if the approximate broadening width is less than a certain value:

$$\Delta w_{Total} \approx \Delta w_{Doppler} + \Delta w_{Relativistic} = N_\parallel \sqrt{\frac{T_e[keV]}{500}} + \frac{T_e[keV]}{500} < \Delta w_{Threshold} \quad (5.1)$$

where T_e is in keV. A value of $\Delta w_{Threshold} = 0.05$ is found to be sufficient for limiting high resolution only to cases with small enough broadening. The input Y has the most points compared to the other inputs because it is imperative to capture the features around the cyclotron resonances well.

3. T_e

   From 1 to 40 keV with an interval of 3 keV starting from 1 keV (i.e. 4 keV, 7 keV, and so forth). This range of temperature should cover most present day fusion machines, ITER, and DEMO.

4. $N_{\parallel}$

   From 0.00 to 0.99 with all multiples of 0.05 in between as the interval points.

The outputs of the dataset are the real and imaginary part of the perpendicular refractive index ($N_{\perp}$).

Not all possible permutations are included in the dataset. If $N_{\perp}^2 \leq 0$, then the propagating wave has transformed into an evanescent wave and these points are excluded from the dataset. Data points with a combination of $T_e > 16$ keV and $X > 0.80$ are also excluded as they produced erroneous outputs. Simply speaking the code fails to produce valid outputs due to reasons that is not yet known. In general, one should be wary for errors when approaching cutoff and when the values of T_e, X, and $N_{\parallel}$ are all simultaneously high. The final dataset consists of about 2.5×10^6 data points.

5.2 Neural Network Training

All neural network trainings were done using TensorFlow 1.8 [10] in the Python language. The dataset is divided randomly into 80% training set and 20% test set. Training is only performed with the training set. The test set is used to test the network ability to generalize beyond the training set. Training is also performed separately for the real and imaginary part of the refractive index. Since they are visually different function-wise, we reason that training separately would allow for more flexible choices of network topology and hyperparameters.

Several pre-treatments or pre-processings are performed on the raw dataset before actual training. These were all done to aid the network in the training process. First, each of the four inputs in the training set are normalized to a distribution with a mean of 0 and a standard deviation of 1. This is done by first subtracting the mean of an input and then dividing by the standard deviation on every elements of that input:

$$x_{i,norm} = \frac{x_i - Mean_x}{Std_x} \tag{5.2}$$

This operation is also applied on the test set. The only difference is that the means and standard deviations used must be the same one from the training set. Subtracting by the mean serves to center the data on the origin while dividing by the standard deviation serves to make each of the inputs equal in range. The purpose of this normalization is to make every inputs to be roughly equal in importance for the training.

The second pre-treatments are on the outputs. The raw real part of the perpendicular refractive index is processed as follows:

$$N^{'}_{\perp,train} = \left(N^{'2}_{\perp,raw} - N^{'2}_{\perp,cold}\right)\left(X\left(1-X\right)\right)^{-1} \tag{5.3}$$

where $N^{'}_{\perp,raw}$ is the raw warm plasma value and $N^{'}_{\perp,cold}$ is the cold plasma value. We computed $N^{'}_{\perp,cold}$ using TORAY cold plasma routine (TWOCMPC). We subtract the cold plasma value from the warm plasma value in order to make it easier for the neural network to perform regression on the warm plasma effects. The X scaling factor is performed in order to make the absolute magnitude for varying X values roughly the same. This scaling is introduced since the magnitude of $N^{'}_{\perp,raw}$ for higher values of X can be smaller by about two orders of magnitude compared to lower values of X. This resulted in worse regression for cases where $N^{'}_{\perp,raw}$ is small since their features are negligible compared to cases where $N^{'}_{\perp,raw}$ is large. The X scaling can be considered as a magnitude equalizer. The raw imaginary part of the perpendicular refractive index is processed as follows:

$$N^{''}_{\perp,train} = N^{''}_{\perp,raw}\ \left(X\left(1-X\right)\right)^{-1}\ max\left(\frac{N_{\parallel}}{\sqrt{T_e[keV]/500}}, 1\right) \tag{5.4}$$

The last scaling (based on the scaling of the absorption coefficient for large angle on page 1202 of [11]) is applied because at high enough $N_{\parallel}$ the absolute magnitude of $N^{''}_{\perp,raw}$ becomes small. The reasoning is the same as in the case of $N^{'}_{\perp,raw}$. Lastly, both the processed outputs are also subjected to the normalization done on the inputs.

The neural networks were trained using a mini-batch size of 1024 (see subsection 4.5.2), the Adam optimizer, and the mean squared error as the loss function. The Adam hyperparameters are fixed at the default settings: $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We considered hidden layers of 2 and 3 with the same number of nodes in each layers. Deeper layers are likely to be impractical for our purpose due to the higher computational time in computing the matrices. A single hidden layer however was found to lead to very slow convergence. The number of nodes considered were 32, 64, and 128. Number of nodes lower than 32 were also found to lead to very slow convergence. The activation functions considered are tanh and relu. We considered only L2 regularization with different regularization strength. We also implemented an early stopping mechanism with a patience (how many epochs to keep training after the loss function stopped improving) set to 30 epochs. One epoch means the entire training set has been used once. Early

stopping can be considered as another form of regularization. One thing that was not done is training repetition for each of the neural networks. Because of randomness in the weights initialization, there will be variations in each training run even if all settings are fixed. There are possibilities that some runs will found a better local minima in the loss functions surface than others.

5.3 Validation

Similar to [31], we also found that the mean squared error is an insufficient metric for determining the quality of the regression. Problems that occur in several regimes of the parameter space cannot be known simply by observing the mean squared error. Therefore in this work, we simply determine the quality of the regression through visual inspection. A plot of the refractive index against the the normalized magnetic field Y - with the rest of the inputs fixed - can give a rough idea of the quality of regression. Several different cases can be chosen to represent different regimes in the parameter space. In this work however, we only show the detailed plot comparison for two different cases (both with $X = 0.30$):

1. A case where both T_e and $N_\parallel$ are low ($T_e = 1$ keV, $N_\parallel = 0.00$) resulting in sharp features at the resonances.

2. A case where $N_\parallel$ is high ($N_\parallel = 0.90$) and T_e is moderate ($T_e = 16$ keV). This leads to an evanescent region appearing for $N_\perp^{'}$ and a broad peak for the $N_\perp^{''}$.

From experience, problems are more likely to occur for these two cases. The quality of the regression for most other explored cases are generally better than these two cases. Thus, they are not shown here. However, there is still a possibility of bad regression in some unexplored region of the parameters space.

We also performed another step of validation using the ray tracing results. This is simply done by comparing the ray tracing results when we use the original warm plasma routine (TWOCMPW) and when we use the neural network warm plasma routine (TWOCMPWNN).

Chapter 6

Results and Discussions

The main results of this project are presented in four parts in the following order: the regression plots of the perpendicular refractive index versus the normalized magnetic field, the graphs showing the loss function of a neural network as a function of training epoch, the ray tracing results, and a brief analysis on the computational time of each TWOCMP routines including the newly created TWOCMPWNN routine.

6.1 Regression Plots

Here, we visually compare regression plots of different neural networks. We first compare regression plots of neural networks with different topologies and activation functions. We then picked one neural network that seems to produce the best regression. We then add L2 regularization to the chosen network to see if we can improve the regression. The method of comparison employed here is by no means robust or objective. However, we will see later that the chosen neural networks can produce sufficiently good ray tracing results. Since this work serves as a proof-of-principle, we reason that a more robust optimization of the neural networks can be further explored in future work. Also, keep in mind that regression points that are negative are physically invalid (for $N_{\perp}^{'}$ these indicate an evanescent region) and will be set to zero in the implementation.

6.1.1 Topology and Activation Function

We show the regression plots for several neural networks with different topologies and activation functions. We show two different cases: low T_e low $N_{\parallel}$ and moderate T_e high $N_{\parallel}$. We focused only on the first harmonic as it is the most relevant for O-mode heating. Keep in mind that we also have separate neural networks for the real and imaginary refractive index respectively. Also, we expect the plots to contain 80% points from the training dataset and 20% points from the test

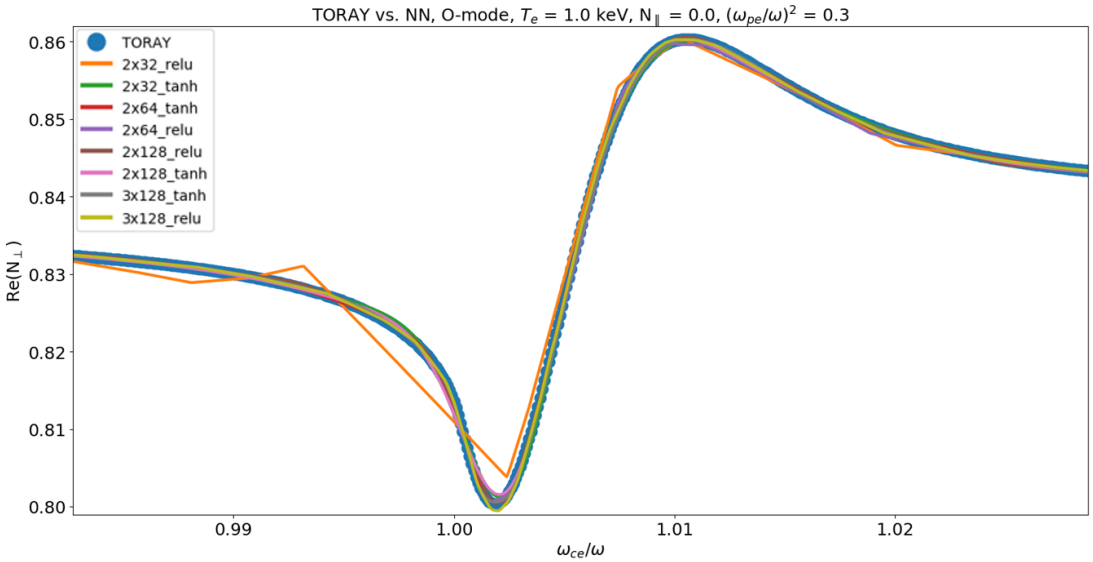

Figure 6.1: A regression plot for the real perpendicular refractive index for several neural networks with different topologies and activation functions. This shows a zoomed-in view (notice the small interval of the horizontal axis) of the first harmonic for a case where both T_e and $N_\parallel$ are low. With the exception of 2x32_relu, the neural networks show good regression with TORAY.

dataset.

Figure 6.1 and 6.2 show the regression for the real perpendicular refractive index. Except for the neural network with 2x32_relu, the regressions are generally good. We have problems in the high $N_\parallel$ case near the evanescent region for most of the networks. The reason for this is not understood but one possibility might be because of the relatively high gradient when you approach the evanescent region. The networks with 3 hidden layers seem to have an advantage near the evanescent region. However, we prefer networks with fewer layers as we are aiming to minimize the computational time of the matrix operations. After considerations, we decided to use the network with 2x64_tanh. This is a trade off between better regression near the evanescent region with computational speed. The effects of the number of nodes are hard to discern from the graphs but we settled for 64. Using 32 nodes will make the computational time faster but it was found that the speed up is minor (about 2 seconds for a single TORAY run) and the ray tracing results are slightly worse. Lastly, we used tanh as it produces a smoother (i.e. less kinks) regression compared to relu. A smoother regression of $N_\perp^{'}$ means it will be easier for TORAY to compute the derivatives necessary for the ray tracing.

Figure 6.3 and 6.4 show the regression for the imaginary perpendicular refractive index. Compared to the real part, the regressions for the imaginary part are generally worse. For the case of low T_e and low $N_\parallel$ shown in Figure 6.3, networks with 3 layers generally perform better than those

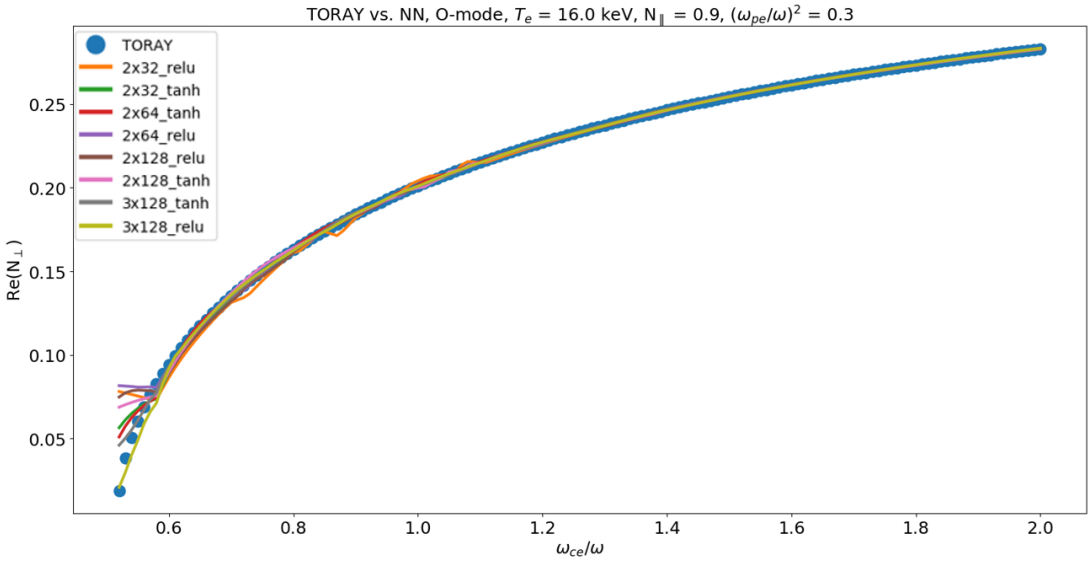

Figure 6.2: A regression plot for the real perpendicular refractive index for several neural networks with different topologies and activation functions. This shows a case where $N_\parallel$ is high resulting in low values of (real) $N_\perp$ and the appearance of an evanescent region (points not included) where $N_\perp^2$ becomes smaller than zero. The neural networks show good regression (2x32_relu still performs worse than the others) except near the evanescent region. In this case, neural networks with 3 hidden layers seems to be better near the evanescent region.

with 2 layers. Networks with the relu activation function are also generally better than those with the tanh activation function. The quality of the regressions degrade significantly for the case of high $N_{\parallel}$ as shown in Figure 6.4. The cause of this problem is not properly understood but might be related to the relatively small magnitude of the imaginary perpendicular refractive index. The values might be just too small for the neural networks to register as an absorption peak. The pre-processings done before training are supposed to alleviate this problem in principle. However, it was discovered that the $N_{\parallel}$ scaling (see Equation 5.4) did not manage to properly equalize the order of magnitude for the different cases. Due to time constraint, a better scaling factor has not been conceived. Despite this, we nevertheless proceed in choosing 3x64_relu as the neural network for our imaginary refractive index. It is one of the networks with smaller noise oscillations in the high $N_{\parallel}$ case while still having relatively moderate network size. Unlike for the real refractive perpendicular index, we are not willing to sacrifice the better performance of the 3 layers for the speed of 2 layers. Somewhat surprisingly, even with the problem in the high $N_{\parallel}$ case we can still obtain decent current drive and power absorption profiles as we will see later. Note that we do not require the derivatives of $N_{\perp}^{''}$. As a result, we can afford to use the choppier relu regression. This might also be why we can still produce decent current drive and power absorption profiles even though we have bad regression in the high $N_{\parallel}$ case. Another reason might be that the absorption for the high $N_{\parallel}$ cases are simply too weak to affect the current drive and power absorption profile (i.e. $N_{\perp}^{''}$ is too small).

We also show the regression for a neural network trained on a reduced dataset that only contains $N_{\parallel} \geq 0.90$ in Figure 6.5. The idea here is to train different neural networks on different regimes of the parameters space. We can see that the regression is better compared to the regression on the full dataset (Figure 6.4). However, we will see later that the ray tracing results are not improved. This lends further support to the idea that the bad regression in the high $N_{\parallel}$ case is not a significant problem. Therefore, this method of improving the regression by having multiple neural networks is not pursued further as it is deemed a lower priority. Nevertheless, it remains a possible path for improvement that can be explored in later works.

6.1.2 L2 Regularization

We show the effects of adding L2 regularization to regression plots of the chosen networks from the previous section: 2x64_tanh for $N_{\perp}^{'}$ and 3x64_relu for the $N_{\perp}^{''}$. Several plots with different regularization strength are shown. The two different cases used for comparison are the same as in the previous section. The purpose of adding regularization is to prevent overfitting. Overfitting leads to a worse regression for points outside of the training set.

Figure 6.6 and 6.7 show the results for $N_{\perp}^{'}$ while Figure 6.8 and 6.9 show the results for $N_{\perp}^{''}$. The regularizations do not seem to improve the original regression. In fact for the case of low T_e

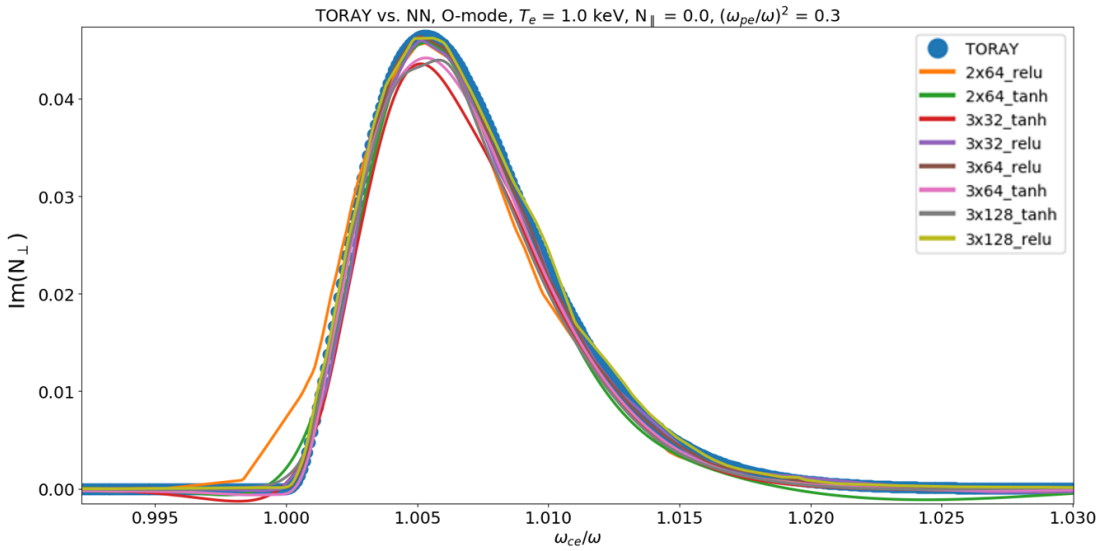

Figure 6.3: A regression plot for the imaginary perpendicular refractive index for several neural networks with different topologies and activation functions. This shows a zoomed-in view (notice the small interval of the horizontal axis) of the first harmonic for a case where both T_e and $N_\parallel$ are low. Generally, networks with 3 hidden layers seems to perform better than those with 2 layers. Networks with the relu activation function also seems to perform better than those with the tanh activation function.

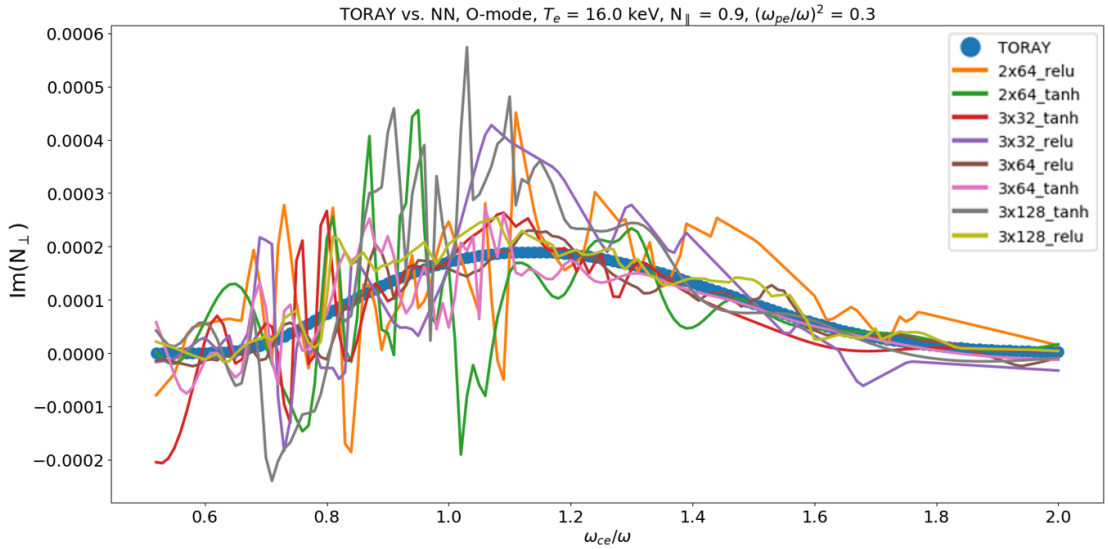

Figure 6.4: A regression plot for the imaginary perpendicular refractive index for several neural networks with different topologies and activation functions. This shows a case where $N_\parallel$ is high resulting in low values of (imaginary) $N_\perp$. The regression for this case is significantly worse compared to other cases. This may be caused by the relatively small magnitude of the imaginary perpendicular refractive index. The difference in absolute magnitude might be too minute for the neural networks to capture.

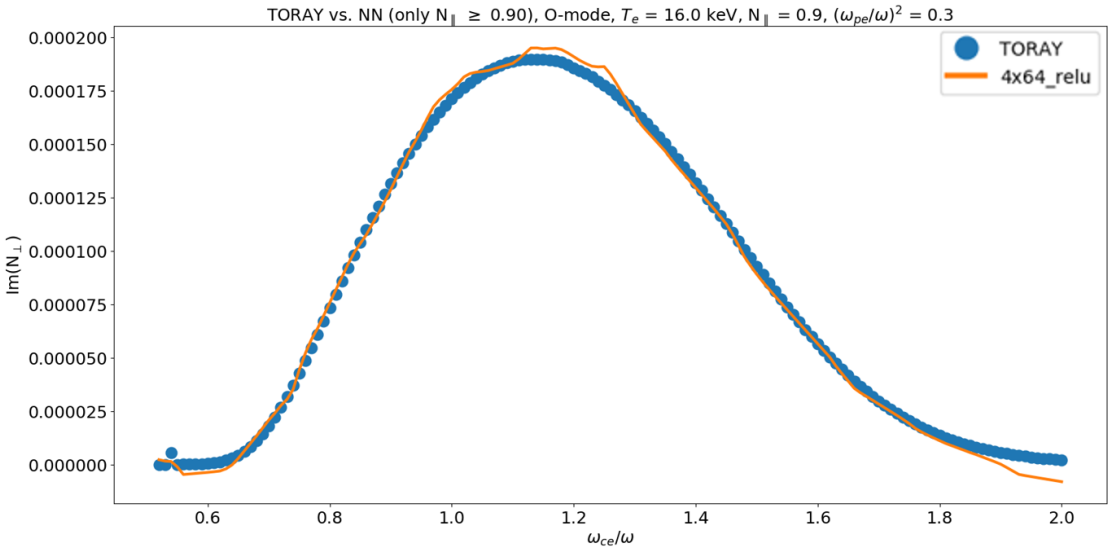

Figure 6.5: A regression plot for the imaginary perpendicular refractive index for a 4x64_relu neural network trained on a reduced dataset that only contains $N_{\parallel} \geq 0.90$. There are differences in the training settings compared to the values described in Chapter 5: Batch size is 512 and the last term of the $N_{\perp}^{''}$ pre-scaling (Equation 5.4) is not applied. We can see that the regression here is better compared to the regression on the full dataset (Figure 6.4).

and low $N_{\parallel}$, too strong regularization will only degrade the quality of the regression. We conclude that the early stopping mechanism is enough to prevent overfitting in this work. We therefore use the unregularized networks for implementation in the TWOCMPWNN routine.

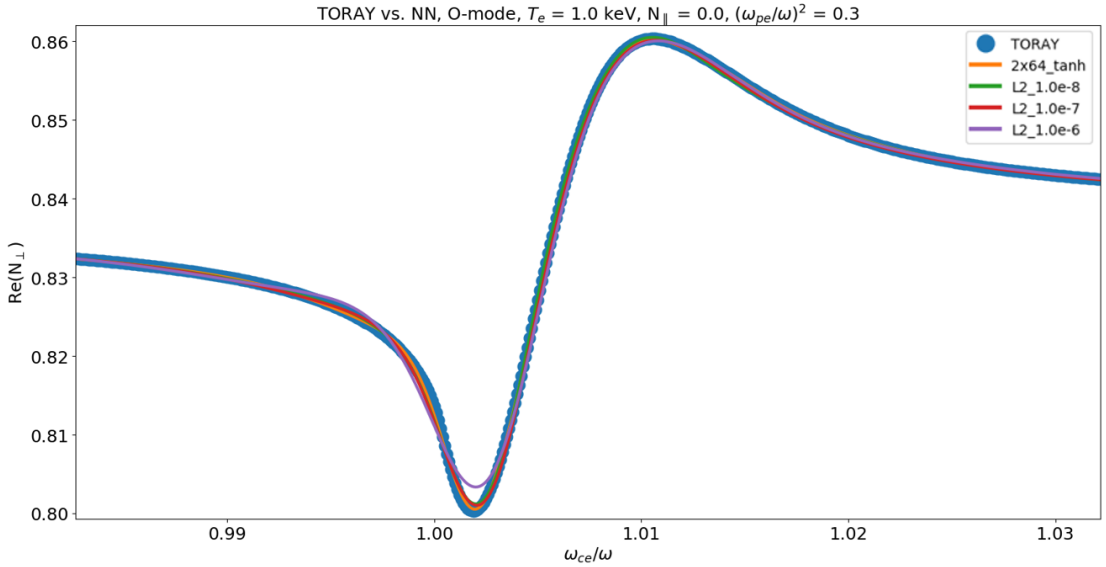

Figure 6.6: A regression plot for the real perpendicular refractive index for several neural networks with different L2 regularization strength. This shows a zoomed-in view (notice the small interval of the horizontal axis) of the first harmonic for a case where both T_e and $N_\parallel$ are low. There appears to be no noticeable difference for L2_1.0e-8 and L2_1.0e-7. L2_1.0e-6 degrades the quality of the regression.

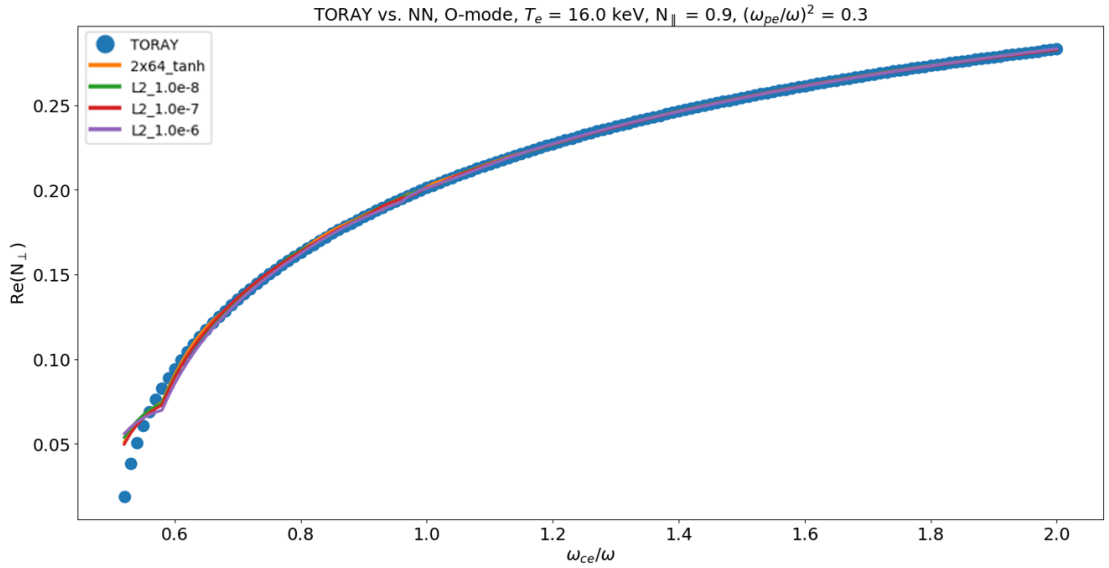

Figure 6.7: A regression plot for the real perpendicular refractive index for several neural networks with different L2 regularization strength. This shows a case where $N_\parallel$ is high resulting in low values of (imaginary) $N_\perp$. There does not appear to be much effect in adding regularization.

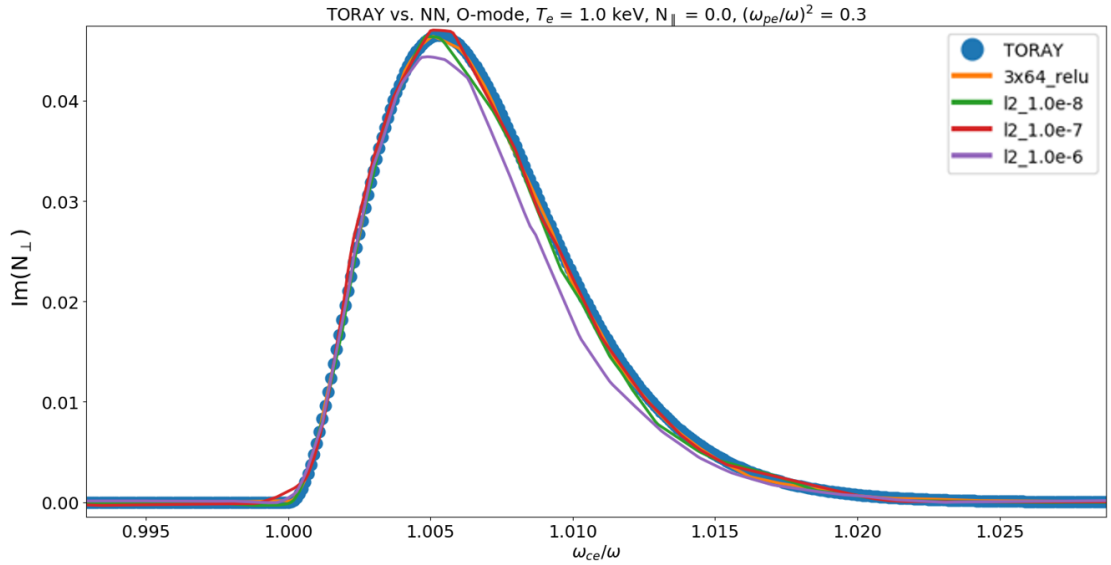

Figure 6.8: A regression plot for the imaginary perpendicular refractive index for several neural networks with different L2 regularization strength. This shows a zoomed-in view (notice the small interval of the horizontal axis) of the first harmonic for a case where both T_e and $N_\parallel$ are low. As we increase the regularization strength, the regression appears to only become worse.

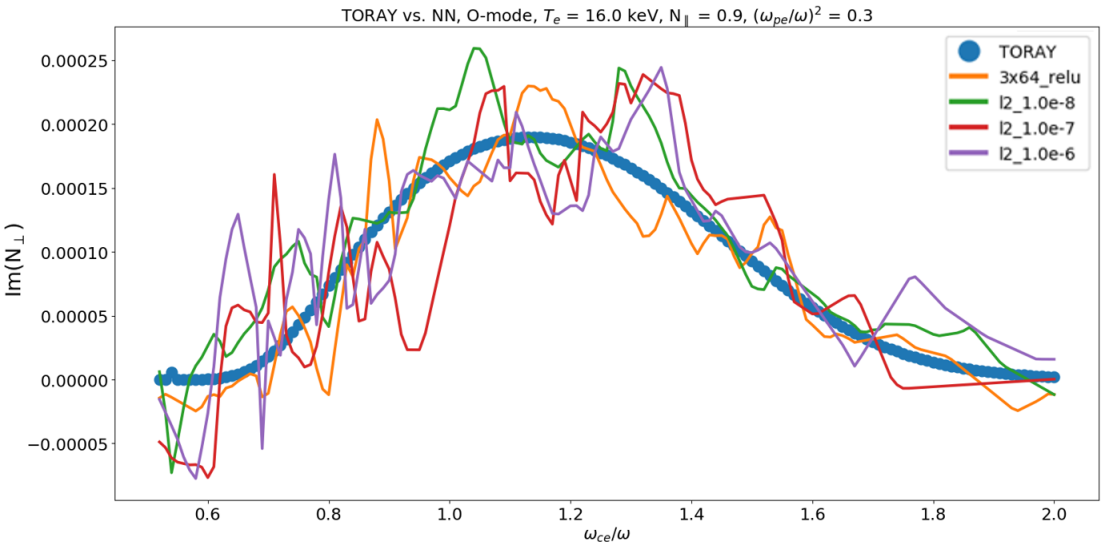

Figure 6.9: A regression plot for the imaginary perpendicular refractive index for several neural networks with different L2 regularization strength. This shows a case where $N_\parallel$ is high resulting in low values of (imaginary) $N_\perp$. The regularization appears to have no effect on the already bad unregularized fit. We prefer not to increase the strength further as the regression for the case of low T_e low $N_\parallel$ is already degraded even with current strength levels.

6.2 Loss Function

We show the graph of loss function versus training epoch for our final choice of neural networks. There are two plots for each neural network: the loss function of the training set (orange) and the loss function of the test set (blue).

Figure 6.10 shows the loss function for the 2x64_tanh $N_{\perp}^{'}$ network. As expected, the loss function decreases as we train. There does not seem to be any sign of overfitting which is usually indicated by increasing test lost function after we have trained long enough. It is however unexpected that the test loss function is always lower than the training loss function. This is believed due to a mistake in how the average loss function is computed in the author's coding. Due to time constraint, this problem has not been rectified.

Figure 6.11 shows the loss function for the 3x64_relu $N_{\perp}^{''}$ network. The loss function is noisier than the case for $N_{\perp}^{'}$ especially the training loss function. This could indicate that the regression for $N_{\perp}^{''}$ is relatively bad compared to $N_{\perp}^{'}$. Another reason might be because of too small mini-batch size. The MSE value also converge to a larger magnitude (by about one order of magnitude) compared to $N_{\perp}^{'}$ which could also indicate a worse regression. In any case, the early stopping mechanism ensures that the epoch with the lowest test loss function is chosen. Same as the case for $N_{\perp}^{'}$, the test loss function is surprisingly always lower than the training loss function.

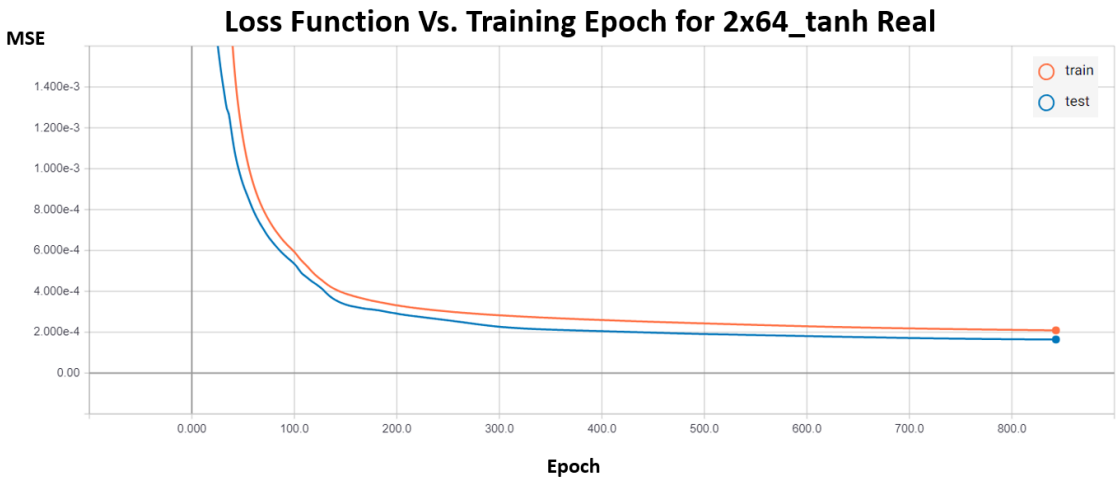

Figure 6.10: A plot of the mean squared error loss function against the training epoch for the 2x64_tanh $N_{\perp}^{'}$ network. The loss function decreases as expected. There does not seem to be any sign of overfitting. It is however unexpected that the test loss function is always lower than the training loss function. This is believed due to a mistake in how the average loss function is computed in the author's coding.

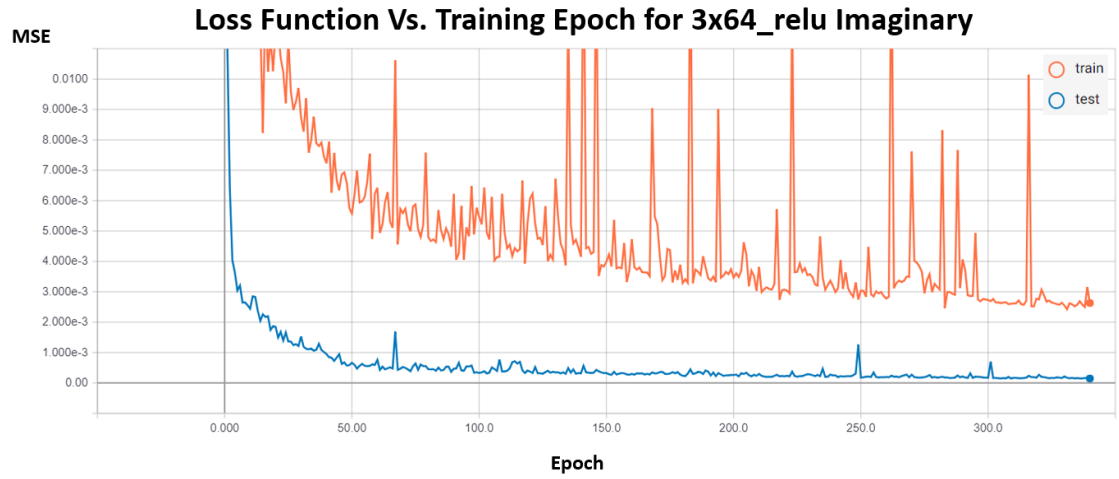

Figure 6.11: A plot of the mean squared error loss function against the training epoch for the 3x64_relu $N_{\perp}^{''}$ network. The loss function decreases as expected but it is noisier than the case for $N_{\perp}^{'}$ especially the training loss function. This could indicate that the regression for $N_{\perp}^{''}$ is relatively bad compared to $N_{\perp}^{'}$. The MSE value also converge to a larger magnitude (by about one order of magnitude) compared to $N_{\perp}^{'}$ which could also indicate a worse regression. The early stopping mechanism ensures that the epoch with the lowest test loss function is chosen. Same as the case for $N_{\perp}^{'}$, the test loss function is always lower than the training loss function. This is believed due to a mistake in how the average loss function is computed in the author's coding.

6.3 Ray Tracing Results

We show the ray tracing results from using the outputs computed by our chosen neural networks. We compare the neural network results with the cold and warm plasma results from TORAY original routines. The EC benchmarking used in this work is the EC-17 benchmark for ITER [16]. In this benchmark, three EC beams each carrying 1 MW heating power are launched: two from the equatorial port and one from the upper port. Beam 1 is launched from the equatorial port and has a 25^o toroidal launching angle. Beam 2 is launched from the equatorial port and has a 40^o toroidal launching angle. Beam 3 is launched from the upper port and has a 18^o toroidal launching angle and 48^o poloidal launching angle. There are four type of plots (see Figure 6.12, 6.13, 6.14, 6.17, 6.20, 6.21, 6.22, 6.23): the ray trajectories projected on the tokamak equatorial mid-plane, the ray trajectories projected on the poloidal cross-section of the tokamak, the current drive profile, and the power deposition profile. We first show the results for ray tracing using only one ray per beams. Afterwards, we will show the results for a more realistic ray tracing using 81 rays per beams. The one ray per beams run was performed only on a single thread. The 81 rays per beams run was parallelized using 48 threads.

Figure 6.12, 6.13, 6.14, and 6.17 show the results for ray tracing with 1 ray per beam. The ray trajectories produced by using the neural network regression (green) are in good agreement with the original warm plasma dispersion relation (red). In the current drive and power deposition profile, we can see discrepancies between the neural network and original warm plasma especially for beam 1 and beam 2. For the current drive, the neural network produced some positive current drive as can be seen from Figure 6.15. This is because there are finite values of $N_\perp^{''}$ between the first harmonic and the second harmonic in the neural network regression. These values are supposed to be zero or too small to be significant. TORAY mistakenly attributed these finite values as absorption at the second harmonic leading to a positive current drive. Figure 6.16 shows more details regarding the discrepancies in the current drive. In any case, the discrepancies in the current drive and power deposition profile are expected because we have a relatively bad $N_\perp^{''}$ regression. We simply need a better $N_\perp^{''}$ regression to rectify the discrepancies. In particular, the greatest problem seems to be when the beam first start to approach the absorption region (i.e. from the low field side). This is consistent with the behavior observed in the $N_\perp^{''}$ regression plots such as Figure 6.3 and 6.8. In these plots, we can see that in general the regression has some trouble at the area where the peak starts to rise from the low field side. One explanation could be that there is a relatively steep increase in gradient that makes the neural network regression more difficult when approaching the first harmonic from the low field side. This explanation is the same one used to explain the worse regression for $N_\perp^{'}$ with high $N_\parallel$ when approaching the evanescent region (see Figure 6.2 or 6.7). Over there, there is also a relatively steep increase in gradient. In

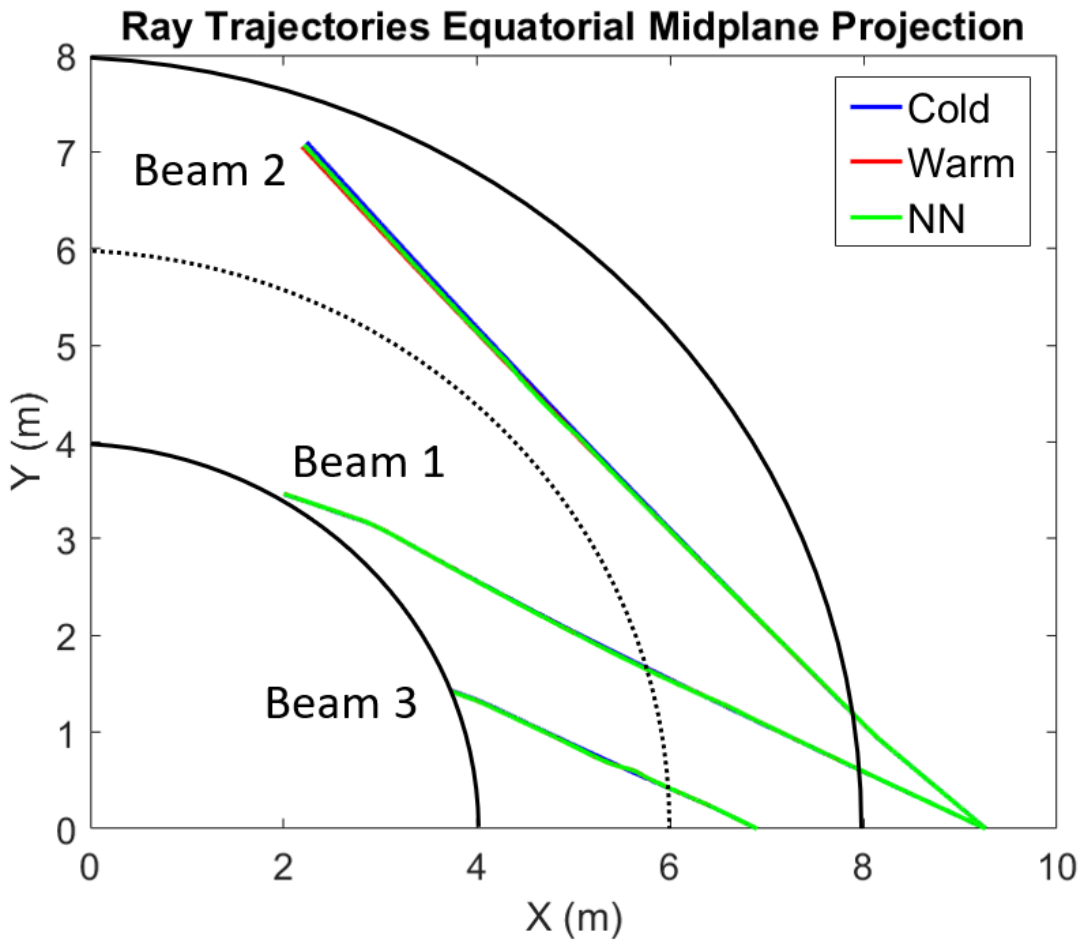

Figure 6.12: The ray trajectories projected on the equatorial midplane for ray tracing with 1 ray per beams. The blue lines, red lines, and green lines refer to the cold plasma, warm plasma, and neural network ray tracing respectively. Ray tracing using the neural network regression produced results in good agreement with the original warm plasma ray tracing.

any case, the area at the foot of the peak from the low field side is likely to have relatively worse regression resulting in too early or too late rise of the peak. For the oblique beam 2, the problem could be exacerbated because the regression for the $N_{\perp}^{''}$ with high $N_{\parallel}$ is particularly bad. But we suspect from the relatively good results of beam 2 that the bad regression for high $N_{\parallel}$ is less significant than previously thought. In Figure 6.18 and 6.19 we plotted the power deposited as the ray was traced for beam 1 and beam 2 respectively. We see that indeed problems occur near the start of the absorption.

Figure 6.20, 6.21, 6.22, and 6.23 show the results for ray tracing with 81 rays per beam. The results are similar to the case of ray tracing with 1 ray per beam. One exception is the appearance of anomalous spikes near the core for the original warm plasma ray tracing (red). These spikes

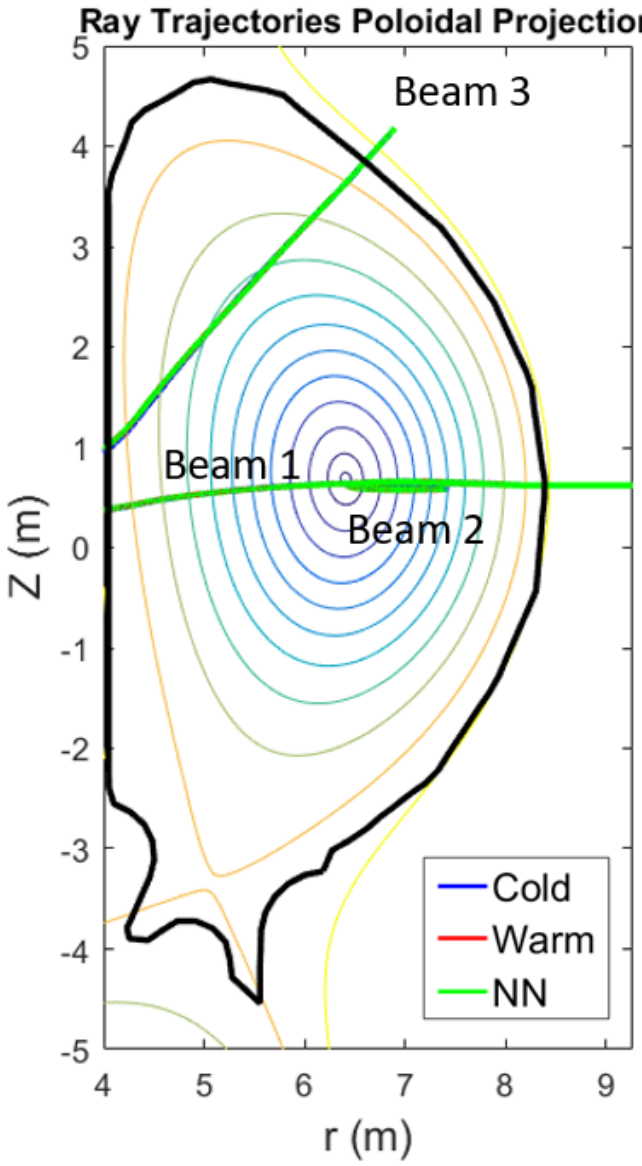

Figure 6.13: The ray trajectories projected on the poloidal cross-section for ray tracing with 1 ray per beam. The blue lines, red lines, and green lines refer to the cold plasma, warm plasma, and neural network ray tracing respectively. Ray tracing using the neural network regression produced results in good agreement with the original warm plasma ray tracing.

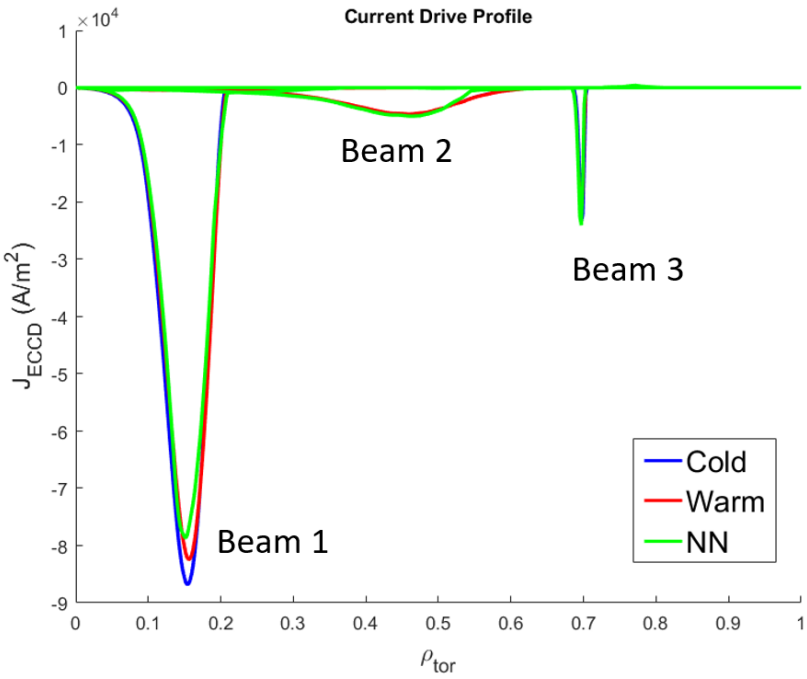

Figure 6.14: The current drive profile for ray tracing with 1 ray per beam. The blue lines, red lines, and green lines refer to the cold plasma, warm plasma, and neural network ray tracing respectively. There is a current drive deficiency from beam 1 and 2 in the neural network ray tracing compared to the original warm plasma ray tracing. As a result, the total current drive in the neural network ray tracing is less than the original warm plasma ray tracing. The discrepancies are due to the relatively worse $N_{\perp}^{''}$ regression compared to the $N_{\perp}^{'}$.

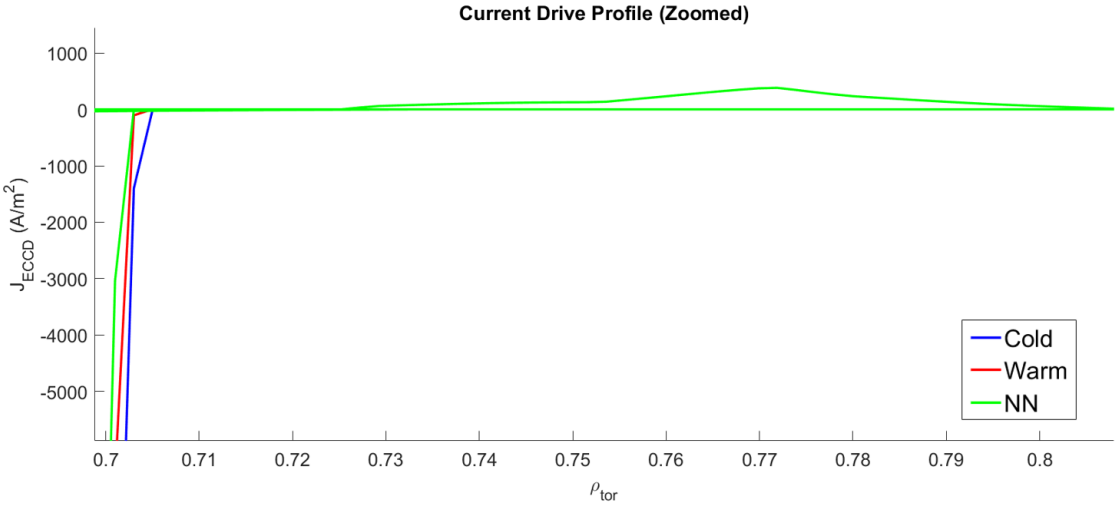

Figure 6.15: Zoomed-in view of Figure 6.14 near Beam 3. There are some positive current drive for the neural network. This is because there are finite values of $N_{\perp}^{''}$ between the first harmonic and the second harmonic in the neural network regression. These values are supposed to be zero or too small to be significant. TORAY mistakenly attributed these finite values as absorption at the second harmonic leading to a positive current drive.

Dispersion Relation	Total Current Drive (kA)	Location of Peak Tip (ρ_{tor})		
		Beam 1	Beam 2	Beam 3
Cold (with FARINA Damping)	- 59.75	0.1538	0.4594	0.6970
Warm (FARINA)	-59.77	0.1558	0.4594	0.6970
Warm (NN)	-56.45	0.1517	0.4614	0.6970

Figure 6.16: A table showing the total current drive and the location of the peak tip of each beams for the cold, warm, and neural network ray tracing. The neural network has a discrepancy of about 5% for the total current drive compared to the cold and warm likely in part due to the existence of erroneous positive current drive. The differences in the location of the peak tip are minute (less than 1%).

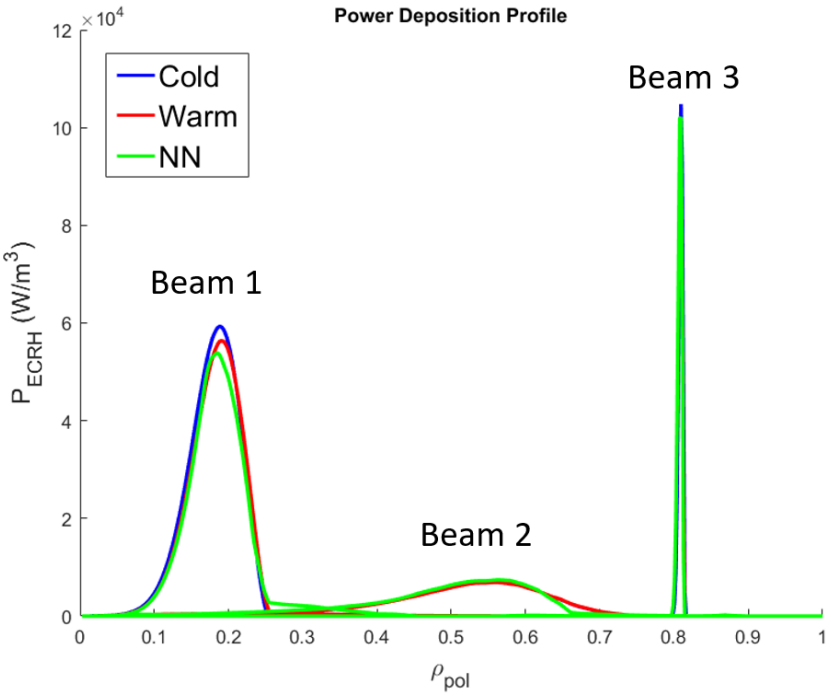

Figure 6.17: The power deposition profile for ray tracing with 1 ray per beam. The blue lines, red lines, and green lines refer to the cold plasma, warm plasma, and neural network ray tracing respectively. In beam 1, too early absorption occurs for the neural network compared to the original warm plasma. There is also a discrepancy near the tip of the peak of beam 1. In beam 2, too late absorption occurs for the neural network compared to the original warm plasma. However despite these discrepancies in the power deposition profile, the total absorbed power remains the same. The discrepancies are due to the relatively worse $N_{\perp}^{''}$ regression compared to the $N_{\perp}^{'}$.

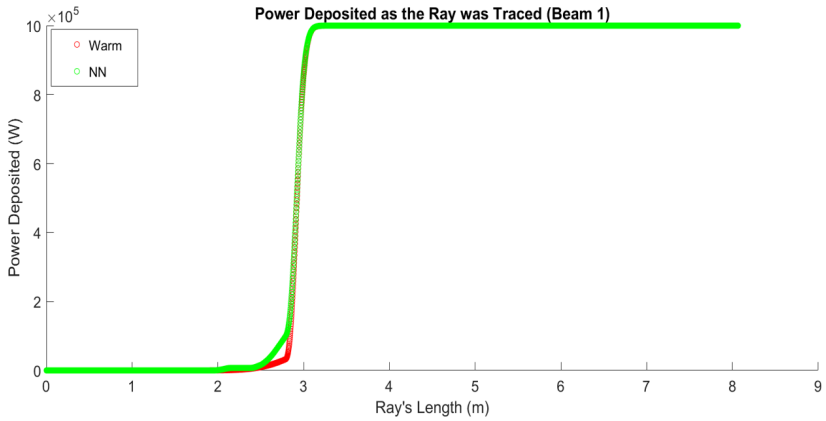

Figure 6.18: The power deposited as the ray was traced for the case of beam 1. Discrepancy occurs near the area where absorption starts to occur.

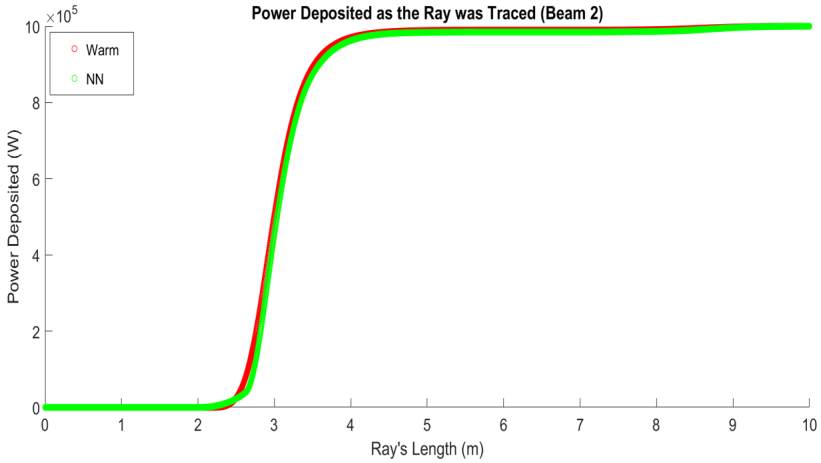

Figure 6.19: The power deposited as the ray was traced for the case of beam 2. Discrepancy occurs near the area where absorption starts to occur.

appeared because of there are some anomalous ray trajectories for the warm ray tracing near the core. Presumably, this problem occurred when the ray becomes tangent with the magnetic flux surface near the core. The reason why this problem does not occur in the neural network ray tracing is not known. One possible explanation is that there are sudden large change in the $N_{\perp}^{'}$ value when this problem occurred. This may lead to a discontinuous function that is naturally smoothened by the neural network regression. Thus, eliminating the anomalous spikes. Lastly, we can notice that the peaks in the current drive and power deposition profile are broader and have smaller magnitudes than the case with 1 ray per beam. This is expected since with 81 rays per beam we have a better representation of an actual beam which has a broader absorption profile than a single ray.

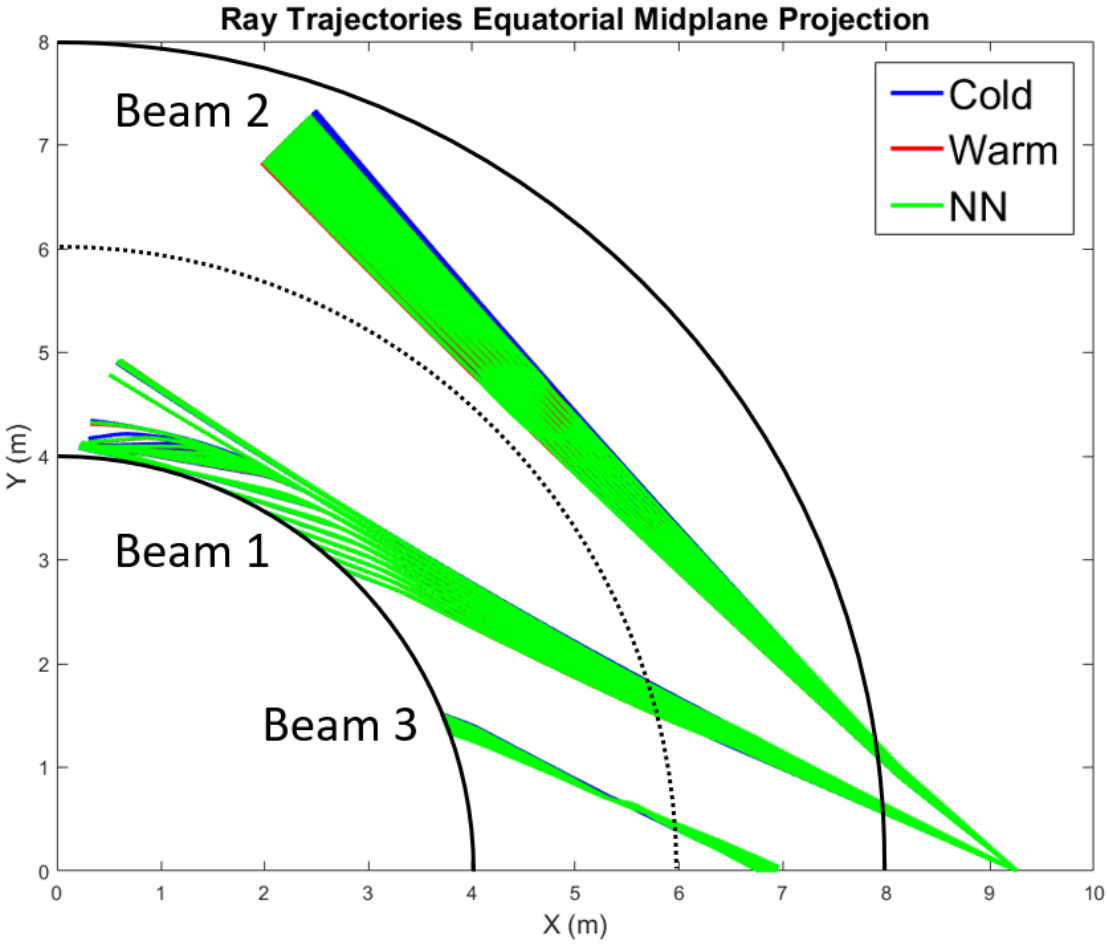

Figure 6.20: The ray trajectories projected on the equatorial midplane for ray tracing with 81 rays per beam. The results are similar to the case of ray tracing with 1 ray per beam.

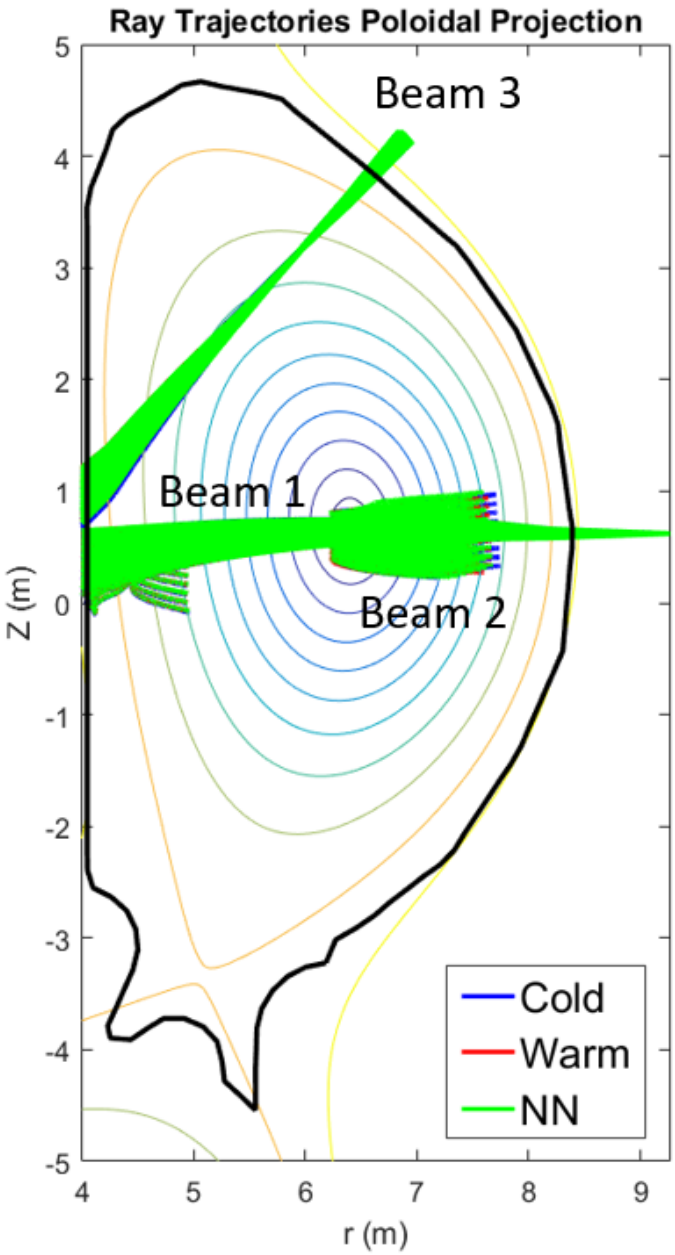

Figure 6.21: The ray trajectories projected on the poloidal cross-section for ray tracing with 81 rays per beam. The results are similar to the case of ray tracing with 1 ray per beam.

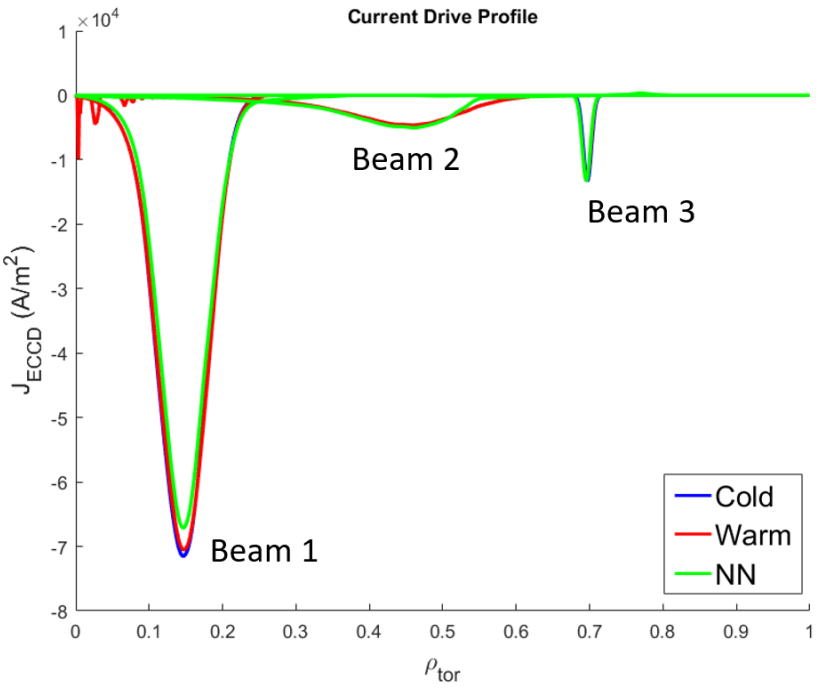

Figure 6.22: The current drive profile for ray tracing with 81 rays per beam. Except for some anomalous spikes occurring near the core for the original warm plasma (red), the results are similar to the case of ray tracing with 1 ray per beam. Notice that the peaks are also broader and have smaller magnitudes since the absorptions are more spread out.

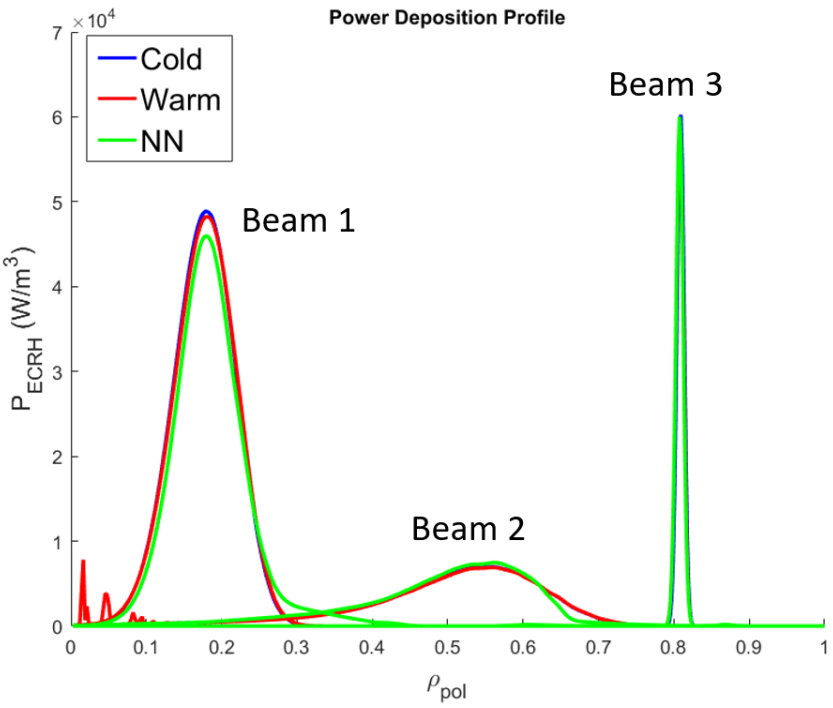

Figure 6.23: The power deposition profile for ray tracing with 81 rays per beam. Except for some anomalous spikes occurring near the core for the original warm plasma (red), the results are similar to the case of ray tracing with 1 ray per beam. Notice that the peaks are also broader and have smaller magnitudes since the absorptions are more spread out.

6.4 Computational Time

We briefly analyze the computational time required by all of the TWOCMP dispersion routines: TWOCMPC (Cold plasma with damping), TWOCMPW (Warm plasma using dielectric tensor computed by Farina and FREPSLN), and TWOCMPWNN (Warm plasma with Neural Network). The analysis presented here is based on the one ray per beams run on a single thread. Furthermore, the values presented here are obtained using the Ifort compiler option: -profile-functions. Note that this compiler option only works for single thread. After adding this option to the makefile, a dump file will automatically be generated after each execution. This file gives a detailed breakdown on the computational usage of every routines and functions in TORAY. Note that there are irreproducible variations between repeated runs but the results presented here are of typical values.

Figure 6.24 shows a table comparing the routines in terms of computational time and number of calls. We can see that the newly created neural network routine TWOCMPWNN managed to be the fastest (time per call on the order of 1 microsecond) warm plasma routine. However, all of the warm plasma routines are called about 9 times more than the cold plasma routine. The code needs higher number of calls when performing warm plasma ray tracing because it needs to compute the numerical derivatives. Derivatives for the cold plasma ray tracing on the other hand is computed analytically without requiring additional calls to the dispersion routine. This tells us that reducing the number of calls to the warm plasma routine is equally as important as reducing the computational time of the warm plasma routine itself. Fortunately, since neural networks are analytical functions, we can directly compute the derivatives analytically from them. This is a natural next step to pursue for future work.

Dispersion Relation	Subroutines	% of Total Time	Time (s)	# of Calls	Time per Call (s)
Cold with Damping	TWOCMPC	0.01	2.357×10^{-3}	137176	1.72×10^{-8}
	DAMPBQ (FARINA)	20.26	4.78	137173	3.48×10^{-5}
Warm (FREPSLN)	TWOCMPW (FREPSLN)	92.38	910.78	1264629	7.20×10^{-4}
Warm (FARINA)	TWOCMPW (FARINA)	64.46	75.29	1234380	6.10×10^{-5}
Warm (NN)	TWOCMPWNN	17.77	6.81	1233804	5.52×10^{-6}

Figure 6.24: A table showing comparisons between the TWOCMP routines in terms of computational time and number of calls. The newly created neural network routine TWOCMPWNN is the fastest (time per call on the order of 1 microsecond) of the warm plasma routines. However, all of the warm plasma routines are called about 9 times more than the cold plasma routine. The code needs higher number of calls when performing warm plasma ray tracing because it needs to compute the numerical derivatives.

Chapter 7

Outlook and Conclusions

7.1 Conclusions

This project aims to research the feasibility of performing a neural network regression on the warm plasma dispersion relation for the purpose of electron cyclotron ray tracing. The purpose of the neural network regression is to create a faster surrogate model of the warm plasma routine of TORAY. A faster computational speed of the warm plasma routine is necessary as a stepping stone towards a real-time version of TORAY.

The input parameters space is 4D with the following quantities: normalized magnetic field ($Y = \omega_{ce}/\omega$), normalized electron density ($X = (\omega_{pe}/\omega)^2$), electron temperature (T_e), and parallel refractive index ($N_\parallel$). The outputs consist of the real part of the perpendicular refractive index ($N_\perp^{'}$) and the imaginary part of the refractive index ($N_\perp^{''}$) corresponding to the wave propagation and absorption respectively. We decided to only perform neural network regression for the O-mode as it is simpler in behavior than the X-mode. A dataset containing about 2.5×10^6 data points was generated for this purpose.

The neural network training was done using Tensorflow 1.8 in Python. We used the mean squared error as the loss function and Adam as the optimizer. The full dataset is randomly split into 80 % training set and 20 % test set. We also decided to use separate neural networks for $N_\perp^{'}$ and $N_\perp^{''}$. Next, we experimented with different hyperparameters for the neural networks: network topology, activation function, and L2 regularization. We also employed an early stopping with a patience of 30 epochs. The final decision is to use a 2x64 tanh network for $N_\perp^{'}$ and a 3x64 relu network for $N_\perp^{''}$ both without L2 regularization.

We validate the neural network regression by visually inspecting the quality of regression and by comparing ray tracing results produced by the neural network routine (TWOCMPWNN) with the original routine. The regression results are generally good for $N_\perp^{'}$ but worse for the $N_\perp^{''}$. This

translates as expected to the ray tracing results: the ray trajectories are good but discrepancies occur for the current drive and power absorption profile.

The computational time for one call of the warm plasma routine is reduced by about one order of magnitude (from 10^{-5} to 10^{-6}) by using the TWOCMPWNN routine. However, we discovered that all the warm plasma routines are called about 9 times more than the cold plasma routine during ray tracing. This is because the need for the warm plasma routines to compute the derivatives numerically.

In conclusion, this work serves as a proof-of-principle that demonstrated the feasibility of using neural network regression to speed up the computational time of the electron cyclotron warm plasma dispersion routine. Future works or improvements are discussed in the Outlook section (7.2).

7.2 Outlook

Below we list possible future works or improvements that can be taken as continuation of this work:

1. Perform the neural network regression for the X-mode. The greatest problem will likely be the existence of multiple solutions near the second harmonic. One would need to devise a way to identify and separate the Bernstein solutions from the X-mode solutions. Another problem is the possible existence of an evanescent region between the upper hybrid resonance and the right-handed cutoff that separates the slow and fast mode branch of the X-mode for $N_\perp^{'}$. This should be less of problem however since we can simply train on ${N_\perp^{'}}^2$ where the evanescent region will correspond to negative values and set the negative values to zero in the regression. It is also possible to split the $N_\perp^{'}$ regression into two to avoid the evanescent region: one for the slow branch and one for the fast branch.

2. Obtain the derivatives of the outputs with respect to the inputs directly from the neural network. Since neural networks are analytical functions, we can directly compute the derivatives analytically from them. Doing this will reduce the number of calls to the dispersion routine. This is the natural next step to reduce the computational time of TORAY warm plasma ray tracing.

3. Optimization of the data generation or the neural network training. The dataset and neural networks used for this work is far from optimized. The current dataset needs to be further investigated to remove possible erroneous points or physically unrealistic points. Better pre-processing of the raw dataset is also beneficial for the training process.

For the neural network itself, the regression for $N_{\perp}^{''}$ is relatively bad. We currently have a problem when we approach the first harmonic from the low field side. The regression seems to have a problem determining the start of the absorption perhaps because there is a very steep increase in gradient. Furthermore, if the regression fails to keep the values between the first harmonic and the second harmonic low enough, TORAY could mistakenly think that there is absorption at the second harmonic which leads to positive current drive. We could turn off the contribution from the second or higher harmonics to remove the positive current drive. However, doing so might have a bad effect on the first harmonic current drive since it is also affected by higher harmonics contribution except in the lowest order approximation. For the case of $N_{\perp}^{'}$, the regression is generally good except near the evanescent region that occurs at high $N_{\parallel}$. All of the mentioned problems could be solved by simply having a better neural network regression however it is not immediately clear how to obtain them. One would likely have to experiment with all the possible hyperparameters. One option that may improve the neural network regression is to have different neural networks trained on different regimes of the parameters space. This method is able to improve the $N_{\perp}^{''}$ regression in the high $N_{\parallel}$ case as shown in Figure 6.5. Another goal is perhaps to find smaller networks that work as good as (or even better than) the current one to further reduce computational time.

In addition, the training pipeline itself has much room for improvement. In particular, the hyperparameters scan should be automatic as opposed to manual as is the case presently. A robust measure of the quality of the regression must also be developed to replace the current visual inspection method. Lastly, a bug in the code that caused the average test mean squared error to be always lower than the training needs to be identified and fixed.

4. Use neural network regression to speed up beam tracing code such as TORBEAM [8] [9] for example. Beam tracing has additional second order differential equations to solve but the warm plasma dispersion relation is the same. Beam tracing can take into account diffraction effects that may lead to a broader absorption profile. In terms of number of equations to solve, beam tracing could have a lower amount than ray tracing depending on the number of rays used to approximate a beam [9].

5. Further validate the neural network regression on other ECRH benchmark cases. The three beams used in the EC-17 benchmark do not represent all possible cases. It is possible that the regression will give poorer results for certain cases. In any case, more validation is always preferred.

6. TORAY - once real-time capable - can provide the current drive and power deposition profile to plasma transport codes such as RAPTOR [15] [14]. Currently, RAPTOR uses

ad-hoc models for the ECRH and ECCD. A more physically-robust model can be provided by TORAY.

7. Intel MKL can be implemented to possibly speed up the neural network matrices computation even further.

Bibliography

[1] International Energy Agency. World Energy Balances. https://www.iea.org/statistics/balances/. Accessed: 25 October 2018. 1

[2] C. M. Bishop. *Neural Networks for Pattern Recognition (Advanced Texts in Econometrics)*. Clarendon Press, 1st edition, 1996. 2

[3] N. Buduma and N. Lacascio. *Fundamentals of Deep Learning*. O'Reilly Media, 1st edition, 2017. 28

[4] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surface of multilayer networks. *Journal of Machine Learning Research*, 38:192–204, 2015. 34

[5] J. Citrin, S. Breton, F. Felici, F. Imbeaux, T. Aniel, J.F. Artaud, B. Baiocchi, C. Bourdelle, Y. Camenen, and J. Garcia. Real-time capable first principle based modelling of tokamak turbulent transport. *Nuclear Fusion*, 55(9), 2015. 2

[6] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. 29

[7] A. H. Kritz, et al. In *Proceedings of the 3rd International Symposium on Heating in Toroidal Plasmas*, volume 22, page 707, Grenoble, France, 22-26 March 1982. Centre d'Etudes Nucleaires de Grenoble. 2

[8] E. Poli, et al. TORBEAM, a beam tracing code for electron-cyclotron waves in tokamak plasmas. *Computer Physics Communications*, 136:90–104, 2001. 2, 67

[9] E. Poli, et al. TORBEAM 2.0, a paraxial beam tracing code for electron-cyclotron beams in fusion plasmas for extended physics applications. *Computer Physics Communications*, 225:36–46, 2018. 2, 67

[10] M. Abadi, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems, 2015. Available from tensorflow.org. 4, 38

[11] M. Bornatici, et al. Electron cyclotron emission and absorption in fusion plasmas. *Nuclear Fusion*, 23(9):1153–1257, 1983. 13, 39

[12] D. Farina. Relativistic Dispersion Relation of Electron Cyclotron Waves. *Fusion Science and Technology*, 53(1):130–138, 2008. 22, 27

[13] F. Felici, J. Citrin, A.A. Teplukhina, J. Redondo, C. Bourdelle, F. Imbeaux, O. Sauter, JET Contributors, and the EUROfusion MST1 Team. Real-time-capable prediction of temperature and density profiles in a tokamak using raptor and a first-principle-based transport model. *Nuclear Fusion*, 58(9), 2018. 2

[14] F. Felici and O. Sauter. Non-linear model-based optimization of actuator trajectories for tokamak plasma profile control. *Plasma Physics and Controlled Fusion*, 54(2), 2012. 67

[15] F. Felici, O. Sauter, S. Coda, B.P. Duval, T.P. Goodman, J-M. Moret, J.I. Paley, and the TCV Team. Real-time physics-model-based simulation of the current density profile in tokamak plasmas. *Nuclear Fusion*, 51(8), 2011. 67

[16] Figini, L., Decker, J., Farina, D., Marushchenko, N. B., Peysson, Y., Poli, E., Westerhof, E., and ITM-TF contributors. Benchmarking of electron cyclotron heating and current drive codes on iter scenarios within the european integrated tokamak modelling framework. *EPJ Web of Conferences*, 32:01011, 2012. 2, 52

[17] R. Fitzpatrick. Perpendicular Wave Propagation. `http://farside.ph.utexas.edu/teaching/plasma/Plasmahtml/node91.html`. Accessed: 22 August 2018. 14, 16

[18] J. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, New York, 2007. 1, 5

[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 28

[20] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 29

[21] A. Karpathy. CS231n: Convolutional Neural Networks for Visual Recognition. `http://cs231n.github.io/`. Accessed: 25 October 2018. 28

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 33, 35

[23] D. J. MacKay. *Sustainable Energy - Without the Hot Air*. UIT Cambridge, 2009. 1

[24] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. 28, 33

[25] S. Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. 36

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. 33

[27] S. Sonoda and N. Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017. 30

[28] T. H. Stix. *Waves in Plasmas*. Springer Science & Business Media, 1992. 5, 13

[29] D. G. Swanson. *Plasma Waves*. Institute of Physics Publishing, Bristol and Philadelphia, 2nd edition, 2003. 5, 13

[30] M. D. Tokman, E. Westerhof, and M. A. Gavrilova. Wave power flux and ray-tracing in regions of resonant absorption. *Plasma Physics and Controlled Fusion*, 42(2):91, 2000. 21

[31] K. van de Plassche. Realtime capable turbulent transport modelling using neural networks. Master's thesis, Eindhoven University of Technology, 2017. 2, 30, 40

[32] E. Westerhof. RF Heating and Current Drive. Egbert Westerhof's lecture notes for the 2016 TU/e course: 3MF130 Heating and Diagnostics. 20

[33] E. Westerhof. Waves in Plasma. Egbert Westerhof's lecture notes for the 2016 TU/e course: 3MF130 Heating and Diagnostics. 5

[34] E. Westerhof. *Implementation of TORAY at JET*. Rijnhuizen Report 89-183, 1989. 2

[35] E. Westerhof. Wave propagation through an electron cyclotron resonance layer. *Plasma Physics and Controlled Fusion*, 39(6):1015–1029, 1997. 2, 15, 17, 21

[36] E. Westerhof. *TORAYFOM manual: a 3D ray-tracing code for Electron Cyclotron Resonance Heating and Current Drive in tokamaks*. Dutch Institute For Fundamental Energy Research, Eindhoven, The Netherlands, 2017. 2, 24

[37] C. Wolfram, O. Shelef, and P. Gertler. How will energy demand develop in the developing world? *Journal of Economic Perspectives*, 26(1):119–38, February 2012. 1

Appendix A

Files

In this appendix, we describe the various files used in this work which are mainly grouped into 3 types: scripts, datasets, neural networks, and ids files. Ids files described the rays traced and were used to generate the ray tracing results in this work. The datasets are in csv. The neural networks use the Matlab variable extension (.mat). We provide a table which states the name of the file, the location of the file, and a brief description of the file. For datasets, neural networks, and ids files, we will not list individual files but merely state the location of the files and describe their naming format. For the purpose of locating files, we assume that the user starts in the 'neuralnet' branch of the TORAY svn.

Name (Type)	Location in neuralnet/	Description
(Neural Network)	In nicholas_files/ neural_networks/. Neural networks trained on full dataset as described in Section 5.1 are located in fullest_data14/. Those trained on reduced dataset that only include datapoints with $N_{\parallel} \geq 0.90$ are located in fullest_data14c/. Folders 're' refer to the real part regression while folders 'im' refer to the imaginary part regression.	The naming format for the neural networks is as follow: 'hiddenlayers'x'nodes' _'activationfunction'_'batchsize' _'regularization'. All the weights and biases for a particular network are included in a single mat file.
(Dataset)	In nicholas_files/datasets/. Full dataset as described in Section 5.1 are located in fullest_data14/. Reduced dataset that only include datapoints with $N_{\parallel} \geq 0.90$ are located in fullest_data14c/.	The dataset files used the csv extension. A text file named 'dataset_division' describes how the dataset is divided into the train and test datasets.
(Ids)	In ids_files/. Files for ray tracing with 1 ray per beam are in 1ray/ while those for ray tracing with 81 rays per beam are in 81rays/.	Ids files described the rays traced and were used to generate the ray tracing results in this work. For 1 ray per beam: 705 (Cold), 704 (Warm), and 810 (NN). For 81 rays per beam: 1112 (Cold), 1115 (Warm), and 1116 (NN).

Name (Type)	Location in neuralnet/	Description
regressor_nn_re.py & regressor_nn_im.py (Script)	In nicholas_files/scripts/	The Python script used for the neural network training. 're' refers to the real part regression and 'im' refers to the imaginary part regression.
makennf90.py & makennf90_im_hnpar.py (Script)	In nicholas_files/scripts/	The Python script used to create 'nn.f90' file which contains the hard coded neural networks in Fortran. The script 'makennf90_im_hnpar.py' is used specially for the network trained on the reduced set only. Note, that the user still have to manually load the neural network matrices first before running the script.
call_twocmpw.f90 (Script)	In src/	The Fortran script used to generate the dataset. Needs the namelist file 'parameters.f90' for inputs.
parameters.f90 (Script)	In obj/	A namelist file containing inputs for 'call_twocmpw.f90'.
ray_equations.f90 (Script)	In src/	Modified TORAY source file that now contains the TWOCMPWNN subroutine.
nn.f90 (Script)	In src/	A Fortran file containing the hard coded neural network matrices.