A Neural Network based Surrogate Model for Single Stator Double Rotor Axial Flux Permanent Magnet Machine

Jorawar Singh Dham,
Department of Electrical and Computer Engineering
McGill University, Montreal
August, 2022

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Sciences in Electrical Engineering

August 13, 2022

Abstract

Axial flux machines have a different geometry of construction when compared to their traditional radial flux machines where the air gap between the rotor and stator is aligned parallel to the axis of rotation. Their unique geometry provides advantages when compared to radial flux machines in many applications. Research into axial flux machines has become popular in recent years with their uses being investigated in many different applications. With the aid of computer design and simulations, many different designs of axial flux machines can be studied to optimise the geometry of the machine for a given design problem. However, simulating axial flux machines has proven to be very computationally expensive. As such, designers need a way to reduce the computational cost of simulations to efficiently search the design space where simulating hundreds or thousands of designs may be required. To solve this problem of computational cost, surrogate models have been proposed in the literature but not well investigated for axial flux machines. In this thesis, we design a type of axial flux machine known as an axial flux permanent magnet machine and create a neural network based surrogate model for the same. We show that the surrogate model reduces the computational cost of simulating a high fidelity model by a significant margin. This neural network surrogate can then be used as a design tool by designers to explore the design space.

Abrégé

Les machines à flux axial ont une géométrie de construction différente de celle des machines à flux radial traditionnelles où l'entrefer entre le rotor et le stator est aligné parallèlement à l'axe de rotation. Leur géométrie unique offre des avantages par rapport aux machines à flux radial dans de nombreuses applications. Les recherches sur les machines à flux axial sont devenues populaires ces dernières années et leurs utilisations ont été étudiées dans de nombreuses applications différentes. Grâce à la conception et aux simulations par ordinateur, de nombreuses conceptions différentes de machines à flux axial peuvent être étudiées afin d'optimiser la géométrie de la machine pour un problème de conception donné. Cependant, la simulation des machines à flux axial s'est avérée très coûteuse en termes de calcul. Les concepteurs ont donc besoin d'un moyen de réduire le coût de calcul des simulations pour rechercher efficacement dans l'espace de conception où la simulation de centaines ou de milliers de conceptions peut être nécessaire. Pour résoudre ce problème de coût de calcul, des modèles de substitution ont été proposés dans la littérature mais n'ont pas été bien étudiés pour les machines à flux axial. Dans cette thèse, nous concevons un type de machine à flux axial connu sous le nom de machine à aimant permanent à flux axial et nous créons un modèle de substitution basé sur un réseau neuronal pour cette machine. Nous montrons que le modèle de substitution réduit le coût de calcul de la simulation d'un modèle haute fidélité de manière significative. Ce modèle de substitution à base de réseau neuronal peut ensuite être utilisé comme outil de conception par les concepteurs pour explorer l'espace de conception.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Prof. David A. Lowther for his patience and guidance throughout the process of research work and completing this thesis.

I am also thankful to my parents. I could not have undertaken this journey without their love and support. Lastly, I'd like to acknowledge my fellow colleagues at the McGill Computational Electromagnetics lab for their support throughout. In particular, i would like to thank Hayagrish Balaji for aiding me with the design software and showing me the ropes.

Contribution of Authors

The thesis was entirely written by me as the only author. I worked as the sole contributor to the work described in this thesis with input from Prof. David A Lowther to guide me in the right direction and my colleagues in discussing the topics and aiding in any technical issues with the software used. I was the sole contributor and researcher on the literature review in chapter 1. The design and parameterisation of the model in chapter 2 were done by me with some aid from my lab mates in getting the hang of the softwares used. The neural network and machine learning aspects of the research described in chapter 3 were executed solely by me. The results and discussion in chapter 4 are my observations on the data with inputs from my supervisor on where more explanation or discussion is needed. The conclusion and notes on the future work in chapter 5 are my thoughts and observations on the work conducted over the overall project and where this work can be extended.

Table of Contents

List of Abbreviations

AFM: Axial flux machine

AFPM: Axial flux permanent magnet machine

SSSR: Single stator single rotor

SSDR: Single stator double rotor

DSSR: Double stator single rotor

MSMR: Multi stator multi-rotor

SMC: Soft magnetic composite

AMM: Amorphous magnetic material

PM: Permanent magnets

NS: North-south

NN: North-north

FEA: Finite element analysis

LMMA: linear machine

IRMA: Inner rotor machine

ORMA: Outer rotor machine

CAD: Computer aided design

FEM: Finite element method

MEC: Magnetic equivalent circuit

SM: Space mapping

SPRP: Shape preserving response prediction

CFD: Computational fluid dynamics

RBF: Radial basis function

MARS: Multivariate adaptive regression splines

SVR: Support vector regression

ANN: Artificial neural network

MLP: Multi Layer perceptron

ReLU: Rectified linear unit

GPU: Graphical processing unit

RMSE: Root Mean Squared Error

MAE: Mean absolute error

MAPE: Mean absolute percentage error

MAX: Max absolute error

RMSPE: Root mean square percentage error

MSE: Mean squared error

List of Figures

List of Tables

Chapter 1

Introduction And Background

While Axial flux machines(AFMs) have been theoretically understood for a long time, it is only in recent decades that technological advancements in materials and manufacturing have enabled them to have a resurgence and become viable in a variety of applications where their unique advantages of axial compactness, high torque density and high-efficiency [1,2] are desired. The literature reports many axial flux machine variants. Of these, the most popular is the axial flux permanent magnet machine (AFPM). Even though AFPMs are increasingly popular, in situations where the costs of rare earth magnets are a concern, other variants such as axial flux induction and reluctance machines have also been studied [3–5]. However, for the purposes of this thesis, we will focus solely on AFPMs and the following sections will serve as an introduction to the field of AFPMs and where the current research stands.

1.1 Early Concepts

In 1831 Michael Faraday made the first simplistic axial flux machine in the form of the homopolar generator [6]. The homopolar generator consisted of a rotating copper disk in an axially directed magnetic field.

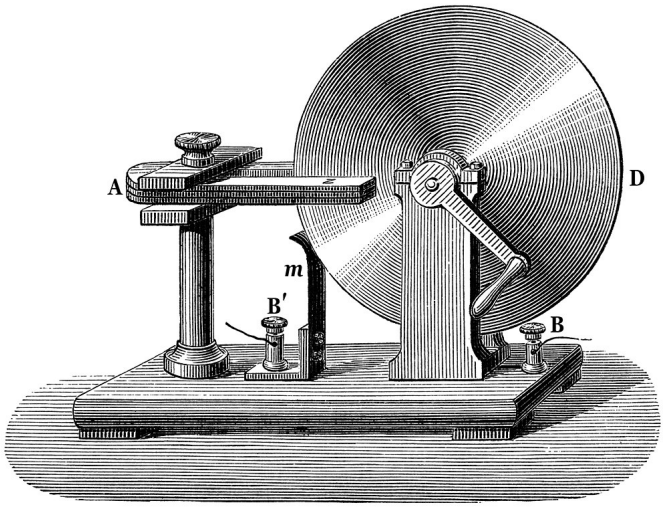


Figure 1.1: Homopolar Generator [6]

However further research into this concept was abandoned as AFMs were burdened with some significant problems. In early single-sided AFMs, large unbalanced attraction forces between the rotor and stator caused deformation and reduced the durability of the machine. Also, AFMs had high costs and difficulty associated with manufacturing owing to the geometry. [7,8].

1.2 AFPM Topologies

Axial flux PM machines have several different possible topologies that can be made through a combination of several hierarchical factors as shown in Fig. 1.2

1.2.1 Structure

The highest level is defined by the number of stators and rotors used in the machine. The simplest structure uses a single stator and a single rotor(SSSR). However, this configuration suffers from a large unbalanced axial force between the rotor and stator which may cause deformations [8,9]. Several techniques have been developed to manage the axial force: slotless stators, current shifting angle, thicker rotor yoke, several bearing config-

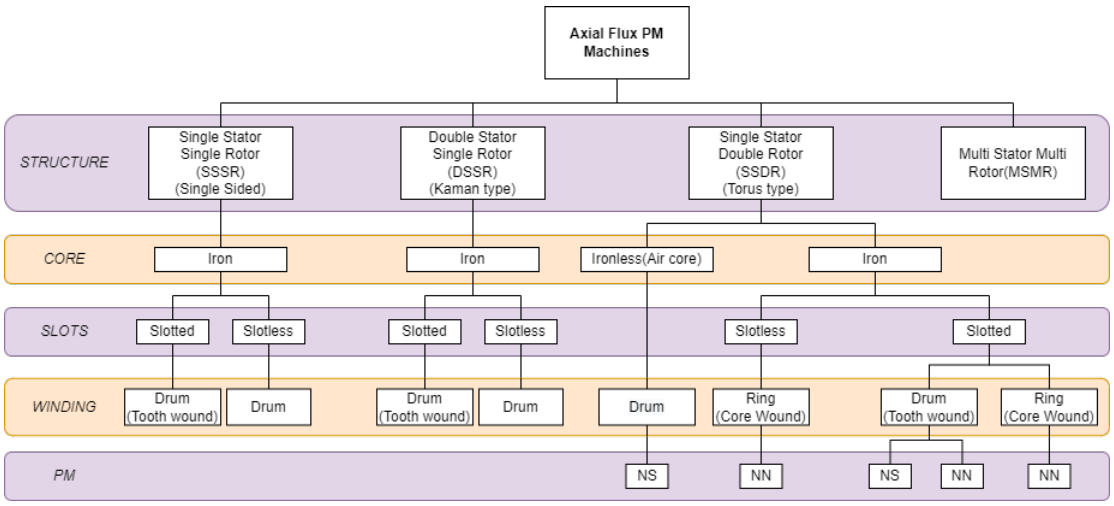
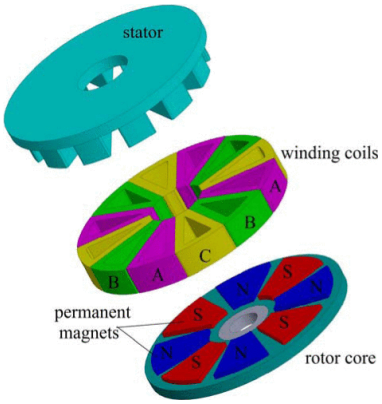


Figure 1.2: Types of AFPM Topologies

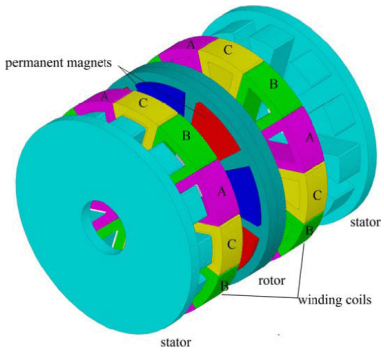
urations, etc. [10, 11]. In double stator single rotor(DSSR) and single stator double rotor(SSDR) configurations, the axial forces are balanced due to symmetry and cancel each other out. Multi stator multi-rotor (MSMR) designs are made by stacking multiple stators and rotors using either SSDR or DSSR configurations. In general, MSMR designs have N stators and N+1 rotors. [12] This enables increasing torque and power density without increasing the diameter of the machine. Fig.1.3 shows the various different AFPM structures.

1.2.2 Core

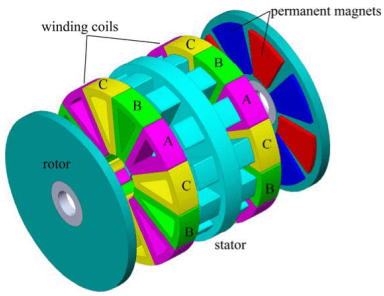
This refers to the stator core material. The category of iron cores also includes other ferromagnetic material such as soft magnetic composite (SMC), amorphous magnetic material (AMM), charged polymers, etc. [14–16]. Ironless(Air) cores can be used in the SSDR configuration where the flux can be made to travel completely axially from rotor to rotor and so the yoke and even the stator core can be omitted. [17, 18]. Doing so completely removes the attraction forces between the stator and rotor, and the stator iron losses, while also significantly reducing the overall weight of the machine. However, this comes with the disadvantage of increasing the effective air gap.



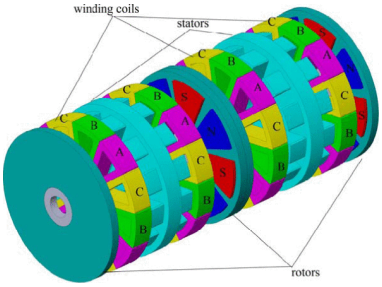
(a) 3-D view of a SSSR AFPM machine.



(b) 3-D view of a DSSR AFPM machine.



(c) 3-D view of a SSDR AFPM machine.



(d) 3-D view of a MSMR AFPM machine (N = 2 stator; N + 1 = 3 rotors).

Figure 1.3: Different AFPM structures [13]

1.2.3 Slots

When the stator is not coreless, it can be made with or without slots. Each option has its trade-offs. Slots create a smaller effective airgap while also providing mechanical strength to the machine. Also, slots allow for variable winding structures which enable control over the flux distribution [19]. However, they also introduce cogging torque, losses in the teeth and flux ripple. Going slotless shortens end windings and lowers mutual and leakage inductance. This eliminates the aforementioned problems but sacrifices the mechanical strength of the machine in exchange [20]. One tertiary advantage is that removing stator teeth also significantly reduces the deforming attraction forces between the stator and rotor in SSSR configurations [21].

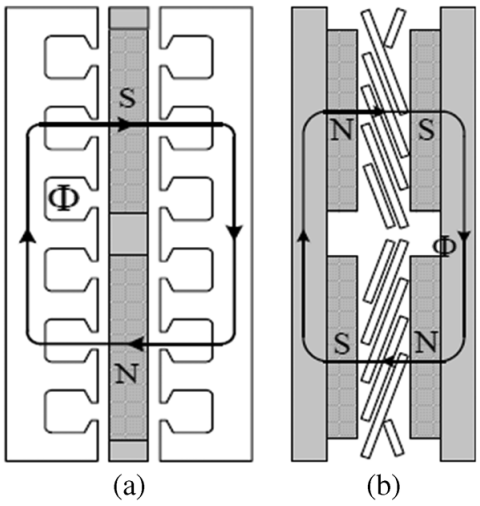


Figure 1.4: Flux paths for: (a)DSSR slotted (b)SSDR coreless [22]

1.2.4 Winding

Stator slots allow for some flexibility in the type of winding used. There are two options here, namely the ring and the drum winding. Each scheme has a different end connection. A Drum winding can be overlapping or non-overlapping and has circumferential end connections along the inner and outer radii. A Drum winding can also be called a tooth wound winding, specifically in the case of slotted stators. A Ring winding(also called core wound, toroidal or back-to-back winding) is always non-overlapping and has axial end connections on the outer and inner radii. Fig.1.5 shows the two types of winding discussed. The Winding is a significant factor in DSSR configurations where the presence of multiple stators increases the overall length of the end windings leading to increased copper losses.

1.2.5 Permanent Magnets(PM)

SSDR(Torus) type machines can have two arrangements of permanent magnets(PM). If the opposing permanent magnets on the rotors have the same polarity it's referred to as the NN configuration and if the magnets have the opposite polarity it is called the NS configuration. This determines whether the flux travels through the stator axially from

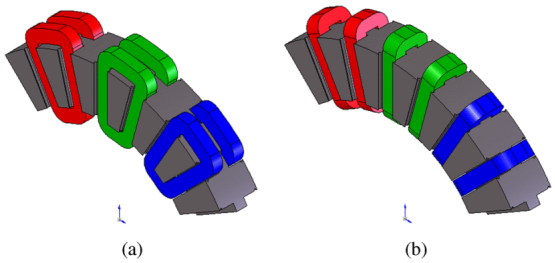


Figure 1.5: Types of AFPM Winding schemes : (a) Non overlapping Drum (tooth) winding (b) Ring Winding [23]

one rotor to another(NS) or the flux travels in the stator yoke circumferentially(NN). This leads to differences in the structure of both configurations. In NN machines, a thicker stator yoke is required as more total flux enters the stator from the two rotors. This flux travels through the yoke, which increases the iron loss.

In NS machines, the stator yoke is redundant as the magnetic flux travels axially through the stator. This reduces iron loss. Even the stator core can be removed completely as discussed before. However, NS machines require drum windings which increase the external diameter of the machine and lengthen the end windings [24].

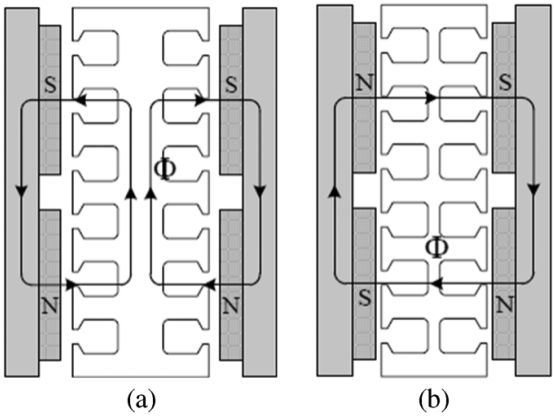


Figure 1.6: Flux paths for: (a)SSDR NN (b)SSDR NS [22]

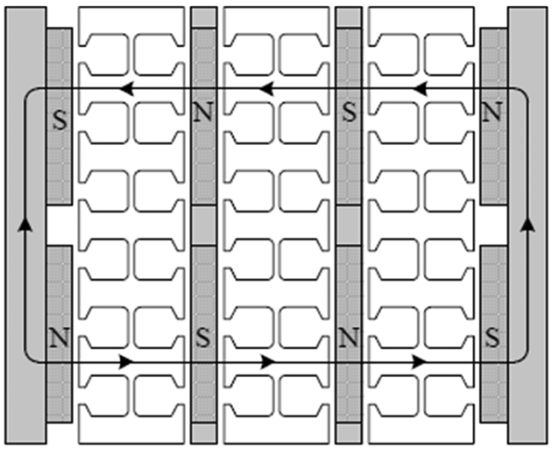


Figure 1.7: Flux paths for: MSMR NS [22]

1.3 Sizing Equations

Machine design is an iterative process and the first step is estimating the required dimensions and other geometrical parameters such as PM volumes, number of phases, stator slots, etc. In terms of dimensions, the outer and inner diameter of the stator and rotor and active axial length are important factors [25]. All of this must be determined while subject to the constraints for the required application such as maximum allowable size, rated/maximum torque, rated/maximum speed, the maximum line-to-line voltage, operating temperature and cooling etc. Sizing equations are a way to determine how these parameters relate to the machine output.

The first widely used sizing equation was proposed by Huang et al. [26] to relate the output power of the machine to the dimensions and the electrical and magnetic parameters.This is known as Essen's Rule and a general compact form is given as:

$$P_{out} = C_{mec} f D_0^2 L_e \tag{1.1}$$

C_{mec} is known as Essen's coefficient and in it's most general form it is given by:

$$C_{mec} = \frac{1}{1+K_\phi}\frac{m}{m_1}\frac{\pi}{2}K_e K_i K_L K_p \eta B_g A_{avg}\frac{1}{p}(1-\lambda^2)\frac{1+\lambda}{2} \tag{1.2}$$

P_{mec}	Mechanical power
C_{mec}	Essen's coefficient
f	Supply frequency
D_0	Machine outer diameter
L_e	Effective Machine axial length
K_ϕ	Ratio of electrical loading on the rotor and stator
m	Number of phases of the machine
m_1	Number of phases of each stator
K_e	Back-emf waveform factor
K_i	Current waveform factor(the ratio between the peak current and the rms current)
K_L	Aspect ratio of the machine(ratio between the outer diameter and the axial length)
K_p	Electrical power waveform factor.
η	Machine efficiency
B_g	Maximum flux density in air gap
A_{avg}	Electrical loading
p	Machine pole pairs
λ	ratio of outer to inner diameter of machine
P_{den}	Power density of machine (by volume)
D_{tot}	Total Outer Diameter
L_{tot}	Total length of machine
A_{in}	Electrical loading at the inner radius

Table 1.1: List of symbols

Equation (1.1) is often used to calculate power density over the volume of the machine:

$$P_{den} = \frac{P_{out}}{\frac{\pi}{4} D_{tot}^2 L_{tot}} \tag{1.3}$$

In Huang et al [26] they determine that the only actual independent variable here is λ(ratio between the outer and inner diameters of the machine) since all of the other parameters are either dependent on it or don't show a lot of range in terms of variation.

The other type of sizing equation relates output torque to the various parameters and was first discussed in [27]:

$$T_{em} = \frac{\pi}{4} B_{ave} A_{in} K_d \lambda (1 - \lambda^2) D_0^3 \tag{1.4}$$

Dividing (1.4) with volume or total weight will give the torque density and dividing by total rotor surface area will give the magnetic sheer stress. Even with this equation, the independent term is still λ.

1.4 Analysis and optimisation

To get a general idea of the machine performance and to aid designers, analysis of the various machine topologies and optimisation is necessary. Unlike radial machines, axial machines cannot be completely converted to a two-dimensional (2D) computational problem because the axial flux paths produce significant three-dimensional effects that must be considered:

1. Flux fringing at outer and inner radii.
2. Curvature Effect: The flux density distribution is radially dependent because geometric features such as PM pole pitches, slots and teeth are themselves radially dependent.

These effects mean that 3D models using finite element analysis (FEA) are the most accurate way to analyse these machines. 2D or quasi-3D models will always lose accuracy but in return for lesser computation time. This is not a big issue because, in the pre/initial design phase, a less accurate but faster simulation which provides an acceptable projection of machine performance is more desirable. Hence most modern techniques try to reduce 3D FEA solves or produce 2D models with acceptable reductions in accuracy.

The simplest 2D models convert the AFPM to a 2D linear machine which ignores the aforementioned effects. In [28] such a model is used for solving the magnetic field in the air gap. The flux fringing is accounted for by defining an "effective length ratio". The other effect is not taken into account. In [29], the authors use a quasi-3D model by using a 2D model along with a 3D FEA derived analytical function to account for the curvature effect.

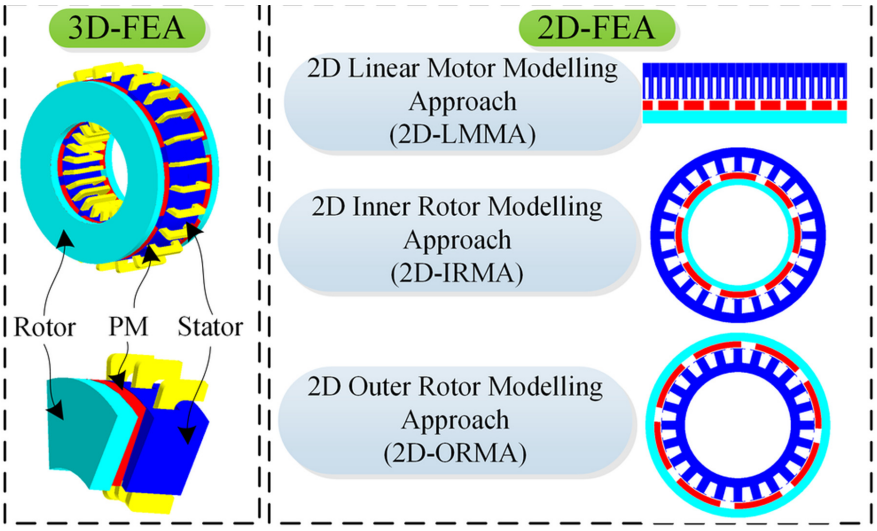


Figure 1.8: Different 2D approaches [30]

This method evolved into the multi-slice approach where the AFPM is divided into several 2D computational planes of different radii. These computational planes can then be solved individually using 2D FEA by considering each plane as a linear machine as before or other 2D topologies. The final output such as flux linkage or torque is then calculated by accumulating the solutions of each computational plane. This accounts for the curvature effect. In [31] the multi-slice linear machine approach is used to calculate the magnetic field in the air gap of a slotted machine. The multi-slice method still has some drawbacks as shown in [32] where a comparison with 3D FEA calculations showed that the approximations can influence cogging torque calculations. [30] studies various multi-slice approaches where each slice can be considered a linear machine (LMMA), an inner rotor machine (IRMA), or an outer rotor machine (ORMA). It observed that the linear machine has the least computation time and that the 2D models were more accurate for coreless designs when compared with experimental values, due to the absence of the non-linearity of iron in coreless designs.

Pure three dimensional models have also been investigated in literature in [33, 34]. These take both aforementioned effects into account but increase the computation time of the model by a significant amount. Even with increasing computational power as tech-

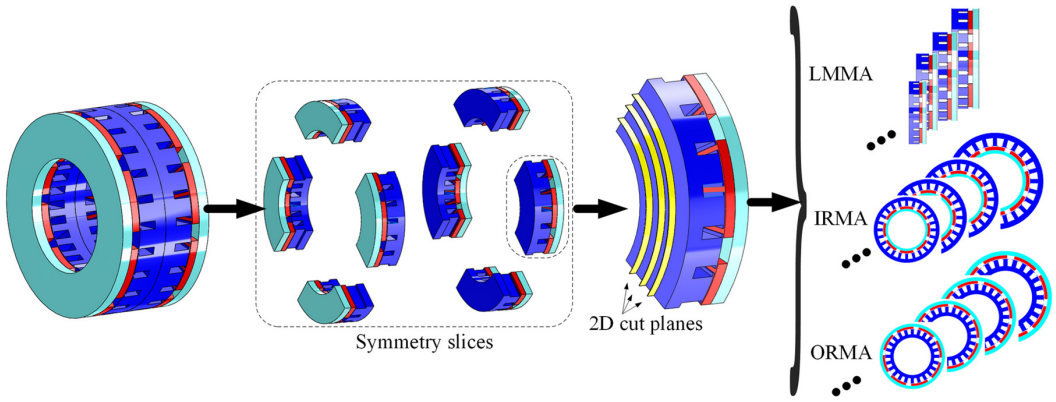


Figure 1.9: Reducing 3D models to 2D planes [30]

nology evolves, a more viable and attractive approach where 3D models are used is to minimise the number of FEA solves using surrogate models [35]. Surrogate or meta models are in the simplest terms, a model of a model. The original high fidelity model 3D model that computes the complex electromagnetic solutions is replaced by a simpler system that can produce analogous results in much lesser time. While surrogate models have been used for other machines for a long time, there is a dearth of research on surrogate models for AFPMs. The aim of this thesis lies within this context. The objective of this thesis is to design an axial flux permanent magnet machine and create a neural network based surrogate model for the same. Hopefully, this will lead to the production of better and more extensive surrogate models for axial flux permanent magnet machines and axial flux machines in general.

1.5 Thesis Outline

The remainder of this thesis is organized in the following manner:

- Chapter 2 discusses computer aided design and the process of designing and modelling an AFPM.
- Chapter 3 is dedicated to defining surrogate models and describes the process of creating a neural network based surrogate for the AFPM designed in chapter 2.

- Chapter 4 presents the results and explains the various performance metrics used to quantify the performance of the surrogate model.
- Finally, Chapter 5 concludes the thesis with an overall conclusion and suggestions for future work to further extend this study.

Chapter 2

Computational Design And Analysis of a Single stator Double Rotor AFPM

2.1 Introduction to Modelling

Within the realm of science, the term ”model” is applied in various contexts to many distinct systems or structures. Definitions of what a model can be range from simple analogies like Bohr's solar system model of the atom to a set of mathematical equations like Maxwell's [36] or even more complex systems. Within the context of this thesis, a definition of modelling that fits well was given by Rose [37] and illustrated in Fig.2.1

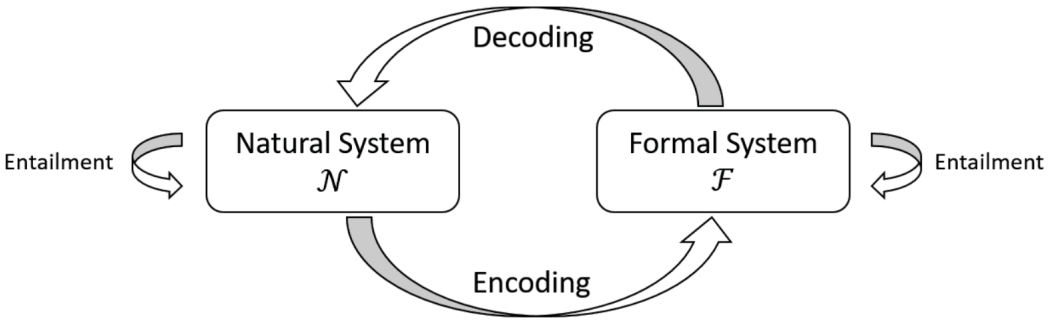


Figure 2.1: The process of modelling [37]

In general, we have a natural system that we want to emulate and we assume that this natural system is governed by a certain set of rules. These rules can be encoded into a new set of rules to be used by a formal system or model in software or other applications. For example, the electric and magnetic field distributions in an electric machine can be encoded into a set of differential equations in a formal system or model. This enables us to map the fields in the natural system of the machine to a set of mathematical equations.

However, this is just one example and there are no rules regarding how to map the rules of the natural system to the rules of the formal system or model. This means that the rules of the natural system and the model do not 'entail' each other and the natural system and the model are 'entailed' only by their own set of internal rules. Therefore, it is possible to have models which have outputs that are representative of observations of the natural system but have completely different structures from the natural system. Since this definition of a model means multiple models can describe a natural system we need some way to evaluate which model is the best. Depending on the requirements there are various considerations to take into account while evaluating a model:

- A model must consistently explain past observations and predict future observations of the natural system. If it is inconsistent in doing so, it must either be modified or rejected.

- The cost of using the model must also be taken into account. Sometimes a less accurate model that produces results quickly is preferable to a perfectly accurate model that takes very long to do the same.

- If informativeness and intelligibility of the model are not important then the simplest model is best given all the prospective models are equally acceptable.

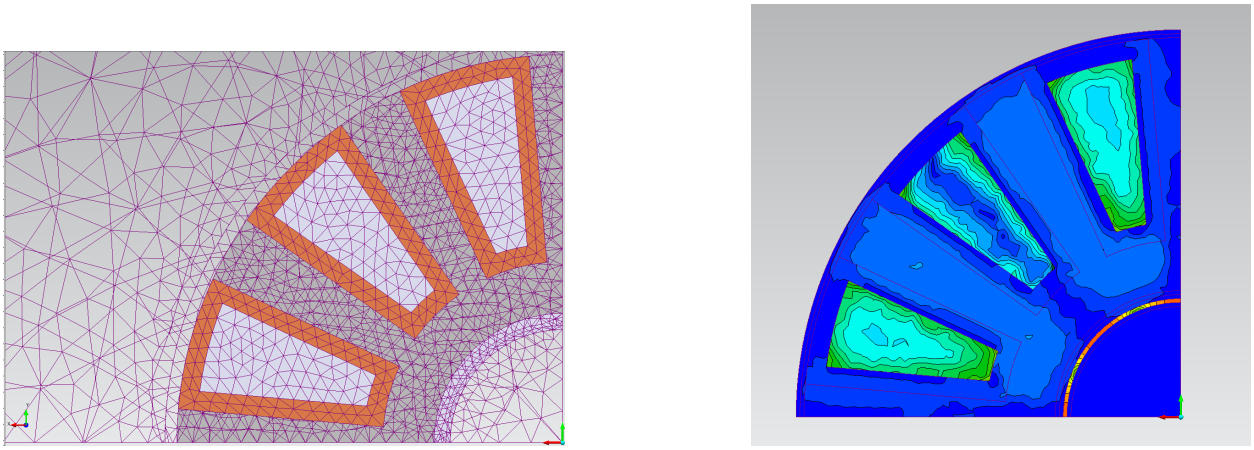
2.2 Computer Aided Design(CAD)

There are many mathematical models in science that represent natural phenomena to an extremely high degree of accuracy. For example, In the realm of electromagnetics, the set of partial differential equations known as Maxwell's equations [36] is the model of choice. The equations describe the behaviour of electric and magnetic fields and how they propagate and interact with objects.

By combining these mathematical models with computers and numerical methods it becomes possible to recreate and simulate many different machines and systems on our computers. This forms the basis of computer-aided design(CAD). Additionally, CAD-based modelling allows us to change various parameters of the model to allow experimentation and explore the design space. This process is known as parameterisation and is very important in analysing and optimising designs.

Since this thesis deals with only electromagnetic modelling of machines we use a software application that combines maxwell's equations with a numerical method known as the finite element method(FEM) to model our machine. This process of using FEM to analyze some natural phenomena by combining it with a set of equations is also called finite element analysis (FEA).

To solve a problem, the FEA uses mesh generation to divide the model into smaller, simpler parts that are called finite elements. The equations defining each element are simple equations that locally approximate the original complex equations(in our case, Maxwell's equations). The equations that model these finite elements are then assembled into a larger system of equations that models the entire problem. The FEM then approximates a solution by minimizing an associated error function using an appropriate method such as numerical linear algebra approaches, Range Kutta etc. Fig2.2 illustrates the mesh and solution fields using FEA.



(a) Generated mesh over a model

(b) Magnetic Field solution using FEA

Figure 2.2: FEA analysis

2.3 Initial Design and Specifications

In this thesis, we model a double rotor single stator(DSSR) model with stator slots, drum windings and a North-South(NS) polarity. Table 2.1 gives the specifications of the AFPM. Fig. 2.3 gives the 3D view of the model. This initial design of the model was done in the Simcenter 3D® software using its inbuilt CAD-based modelling system.

Outer Diameter	240 mm
Inner Diameter	140 mm
Poles	8
Phases	3
slots	18
Rotor yoke thickness	10 mm
Stator yoke thickness	0 mm
PM thickness	9 mm
Pole Pitch	45°(mechanical)
PM skew	0°
PM material	Neodymium Iron Boron: 32/31
PM swept angle	45°
Number of turns in windings	25
Air gap	5mm

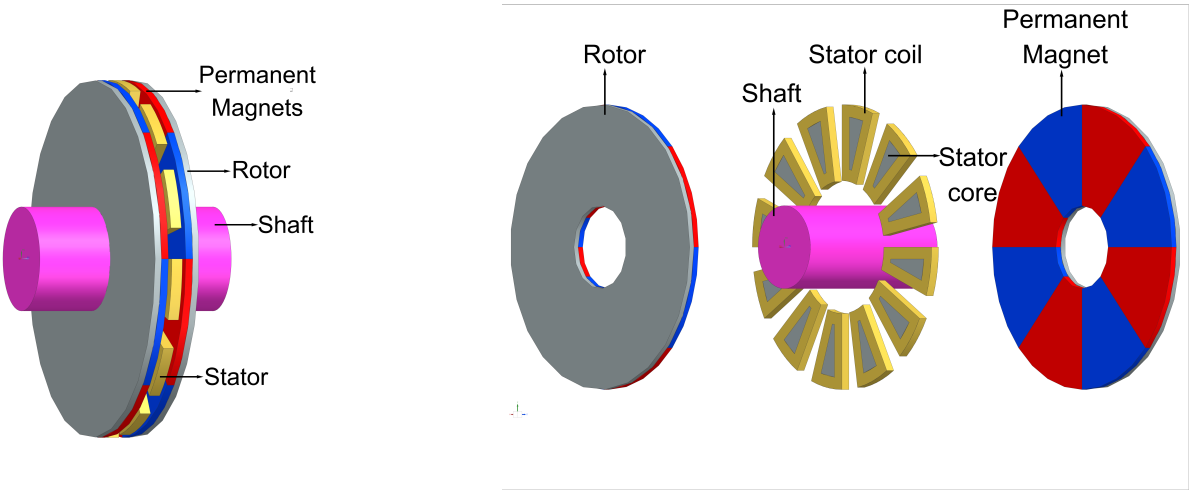
Table 2.1: Specifications of the AFPM

2.4 Simulating the Model using 3D FEA

To run FEA simulations, the model was then recreated in Simcenter MAGNET®. Simcenter 3D® does not have a dedicated electromagnetic FEA solver. The initial design was made in Simcenter 3D® since it has a better and more robust CAD-based modelling system. The model geometry was then imported to Simcenter MAGNET® from Simcenter 3D®.

To ensure that the simulations are computationally efficient and physically accurate, we made several assumptions to reduce our model to a simpler form:

- The stator yoke can be ignored since the flux paths are axial and hence the model is being considered yokeless.
- The motor is an eight pole machine. Since the outputs repeat after each pole pair we can reduce the model to a one-eighth equivalent using the symmetry in geometry and boundary conditions. Fig. 2.4 visualises the process of reducing the model. Fig. 2.5 shows the boundary conditions imposed on the model.
  - The boundary plane across the face of the stator and perpendicular to the axis has a flux normal boundary condition.
  - The two boundary planes perpendicular to the direction of motion have the even periodic boundary condition. i.e, they represent a 360° electrical segment of the machine
  - The other remaining surfaces of the outer airbox have flux tangential boundary conditions to terminate the field appropriately.
- Since the relative motion between the stator and rotor is simulated we can make the stator our component in motion while the rotors stay stationary.
- The amount of magnetic flux passing through the shaft is negligible and the shaft is redundant in terms of the magnetic field and can be ignored in our simulations.



(a) 3D view of the AFPM

(b) Exploded view of the AFPM

Figure 2.3: A double rotor single stator Axial flux machine with NS polarity

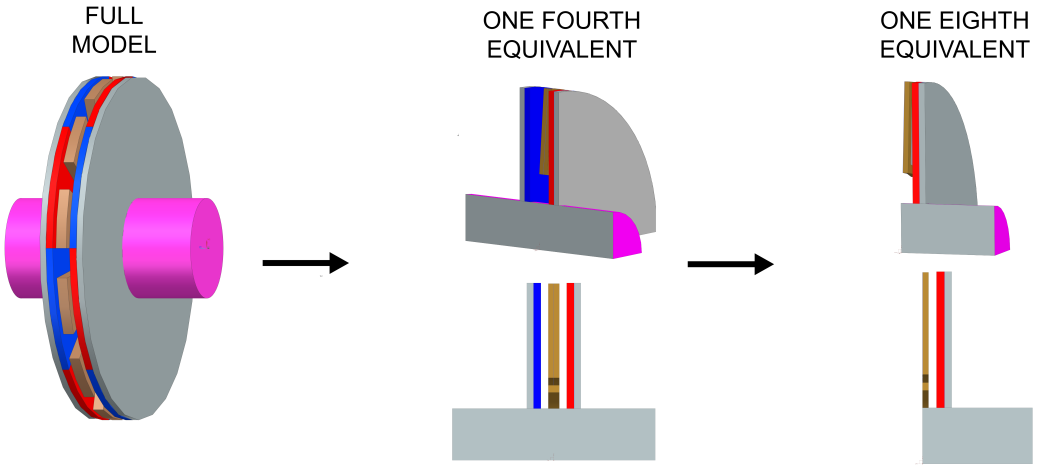


Figure 2.4: Reducing the model

After the model was finished in MAGNET we connected the stator coils to a 100V 40Hz three phase supply with no load on the motor and ran a transient 3D simulation with motion over 10 sec with a time step of 100ms. Simcenter MAGNET's 3D transient solver finds the time-varying magnetic field in and around current-carrying conductors, in the presence of materials that may be conducting, magnetic, or both. These are then used by the solver to calculate the force and torque on the different bodies that constitute the model. Our desired output is the torque on the rotor. However, the simulations were

found to still take too much computational time with each time step taking more than a minute which means the complete model took more than 100 minutes.

To reduce computation time we instead run 3D static simulations over different positions of the rotor. The 3D static simulations can act as a replacement for transient solutions as there are no eddy currents induced in the components of the machine. Two of the stator coils are supplied with 5A DC current and the third is given a 10A DC current in reverse. This creates an instantaneous snapshot of the current in the system, since in a three phase system with no or linear load, the sum of the instantaneous currents of the three conductors is zero. The magnitude of current in each conductor is equal to the sum of the currents in the other two, with the opposite sign. 3D static simulations are run with the rotor at the 0°, 5°, 10° and 15° positions. The simulations take about 10 sec for each position thus reducing the computation time for each model to 30-40 sec.

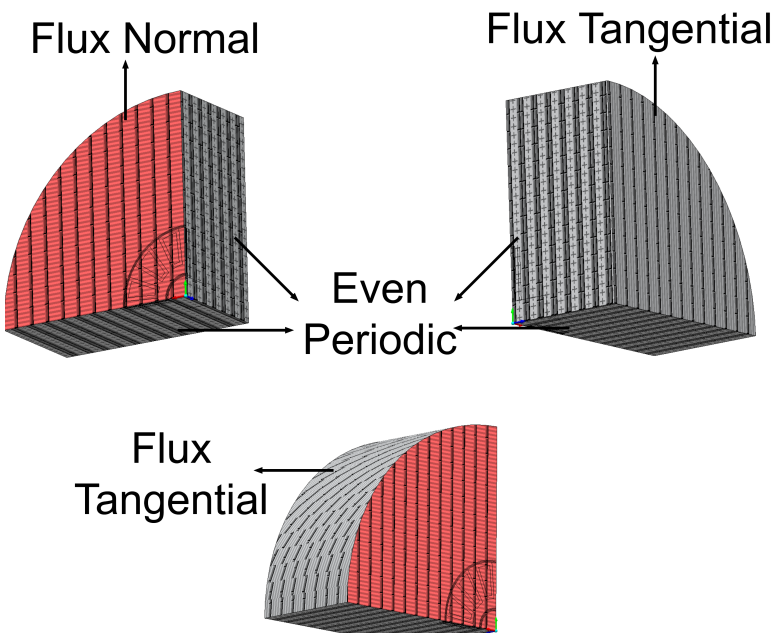


Figure 2.5: Model with outer airbox and boundary conditions

2.5 Parameterisation

To parameterise the model, we vary a few specifications of the model over a range of values and create a model for every combination of the different values. The other specifications from Table 2.1 are kept the same. Table 2.2 shows the parameters that are being varied across the different models. This leads to 14145 distinct combinations.

$Air\ gap(mm)$	10, 9.5, 8, 7.5 , 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1
$PM\ Thickness(mm)$	10, 9.5, 9, 8.5, 8, 7.5, 7, 6.5, 6, 5.5, 5, 4.5, 4, 3.5, 3, 2.5, 2
$PM\ material$	Neodymium Iron Boron: 32/31, AL9N: Alnico-9NB, Ceramic 8A, Samarium Cobalt: 26/10
$PM\ swept\ angle$	45°, 40°, 35°, 30°, 25°, 20°, 15°, 10°

Table 2.2: Design variables being varied in the parameterised model

All the possible model specifications were compiled in an excel sheet. To automate the creation of each model, visual basic scripting is used. Simcenter MAGNET allows the user to recreate every action that can be done on the software using a visual basic script. A script was created that if given the specifications, can create the model automatically, run the FEA simulations and export the output torque values. Another script iterates through the excel sheet and pulls the specifications of each model, feeds it to the first script and writes the torque values to the same excel sheet. This creates our final database that we will use to create our surrogate model.

2.6 Summary

In this chapter, we described the process of modelling, parameterising and simulating a double rotor single stator axial flux permanent magnet machine using computer-aided design. First, we described what a 'model' is and the process that fits our methodology in this thesis. Next, we decided on the specifications and geometry of the machine we need to model. Then we used computer-aided design tools to create and parameterise the

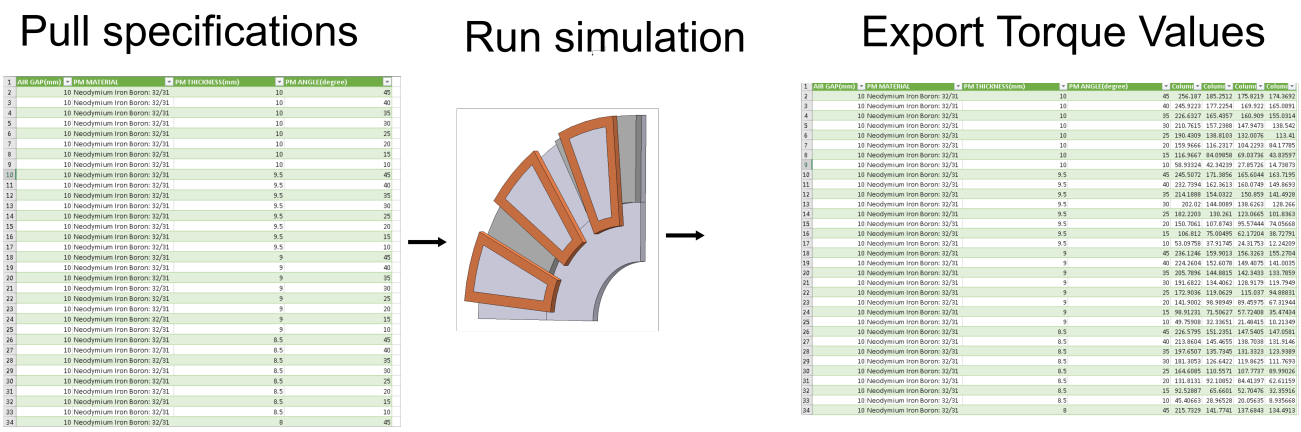


Figure 2.6: Creating the database

model equivalent of our machine. This model was then simulated using FEA to create a database of results. Fig 2.7 illustrates and summarises our design process. However, this design process can be optimised further using surrogate modelling. In the next chapter, we will describe surrogate modelling and use our database of results to create a surrogate model of our own.

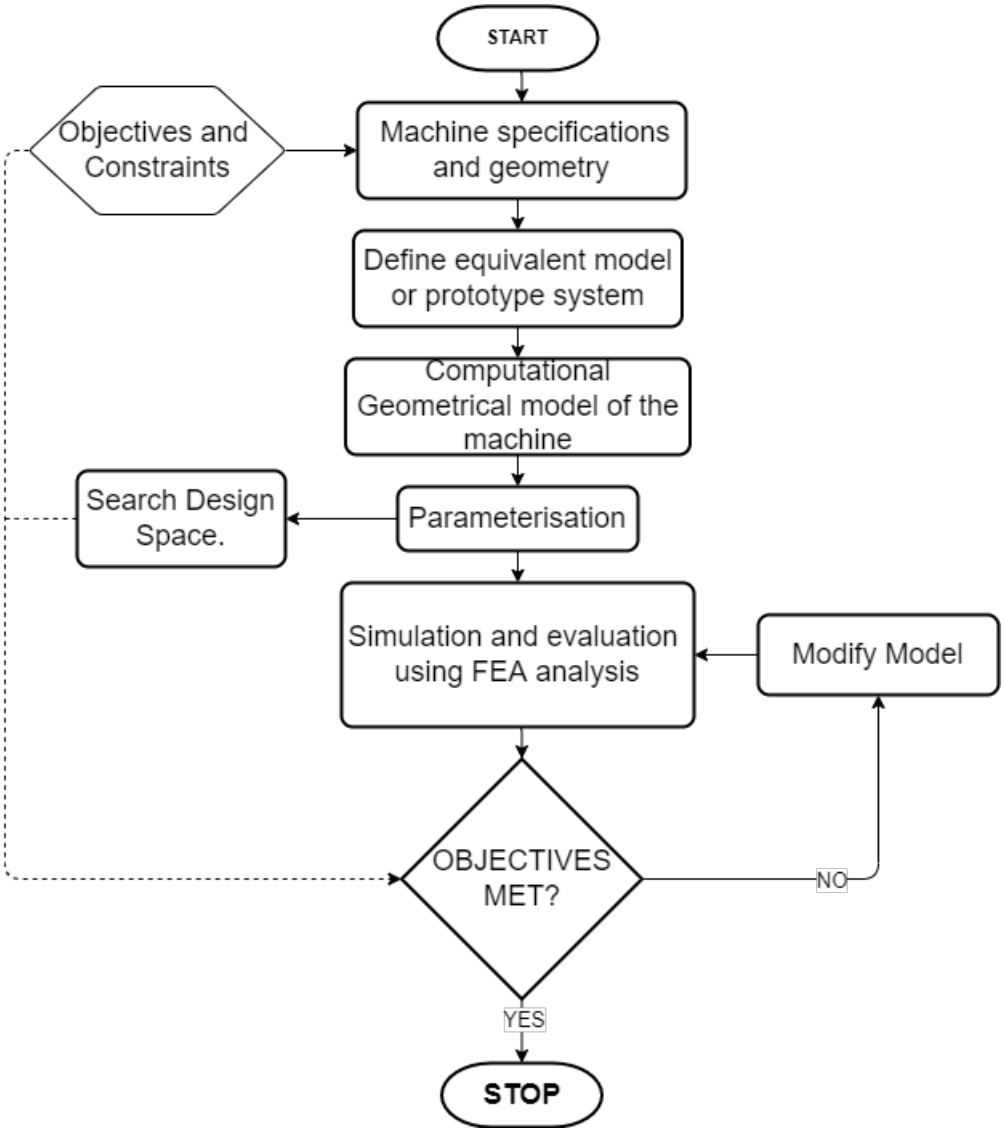


Figure 2.7: Flowchart of the design process

Chapter 3

Neural Network Based Surrogate Model

3.1 Optimisation in Modelling

Based on the design process to create an equivalent model of a machine and its simulation, we can create a mathematical definition of the design problem consisting of two basic elements [38]: (i) The synthesis problem and the (ii)The analysis problem.

3.1.1 The synthesis problem

The synthesis problem deals with taking the requirements and specifications produced by the client or designer and translating them to appropriate constraint and objective functions that our computational model can use. It also involves parameterising the model by taking some of the parameters of the machine and making them design variables so that we can explore the design space. The mathematical definition of the synthesis problem is a multi-objective optimisation problem of the form:

$$\min_{x} f(x) \tag{3.1}$$

$$subject\,to : x \in \Omega$$

where $x = [x_1, x_2, ..., x_n]$ is the vector containing our design variables within the feasible space Ω defined by the constraints, and $f(\cdot) = [f_1(\cdot), f_2(\cdot), ..., f_m(\cdot)]$ is the vector of objective functions.

3.1.2 The Analysis Problem

The analysis problem deals with the simulation of the model and consists of using numerical methods to solve the mathematical equations which model the natural laws that dictate the behaviour of the machine. The analysis problem must be solved for each possible design solution given by the parameterised model. For example, in our case, this means solving Maxwell's equations for each possible design solution.

3.1.3 Optimisation Methods

In many design problems, the client may require a large number of objectives to be optimised simultaneously. Moreover, the objective functions themselves can be harder to optimise having multi-modal, discontinuous and even non-convex functions. This can make the synthesis problem (3.1) very complex to optimise and significantly increase computational costs. In these cases, traditional optimisation algorithms such as trust region [39] and gradient-based methods [40] do not significantly reduce computational costs and are not adequate [41]. More advanced algorithms, such as genetic algorithms, memetic algorithms, swarm intelligence etc., have been developed over the years to better address these problems and achieve more flexibility. [38, 42–44]

One of the simplest ways to make the process more efficient is to make the optimisation algorithms better so that they converge faster for the given problem. However, here we run the risk of losing adaptability and making the algorithms too specific to a given problem. Moreover, there is no way to evaluate beforehand what algorithm will work best for what specific problem.

Even with advanced optimisation algorithms for the synthesis problem, if there are higher accuracy demands, the model has nonlinearities and the complex system requires a large number of simulations, the analysis problem can become very computationally expensive to solve. The simulation time may run into hours or even days [41].

There may be instances where time is a constraint and we need to stop searching the design space for solutions after a specified amount of time. The optimal solution found by the model at this time may not meet the standards of the client. We can try to use better optimisation algorithms to reduce the computational cost but if that's not an option, we need to go back to the model and make some changes to make the model more efficient to solve without changing the objectives or the design specifications.

In the worst-case scenario, despite all efforts, the model may completely run out of computational resources and no optimal solution is found. The model would be considered a failure and we need to go back to the drawing board and suggest to the client that the original objectives are not feasible or practical to solve. In this case, depending on how flexible the client's demands are we may need to change or reduce the objective functions that we are solving for, and/or change the design specifications to reduce the search space.

From this discussion, we can conclude that in CAD-based modelling, the computational cost of designing and simulating a model can be a major factor, especially if the designer does not have a lot of computational resources. We need ways to reduce the computational costs to make design automation practical. In the following sections, we will discuss methods of reducing the computational cost of our model when running the original model becomes impractical.

3.2 Model Order Reduction

One of the best ways to reduce the computational complexity of a model is to replace the original synthesis problem with another problem that is analogous to the original prob-

lem but easier to solve. This problem is called a reduced problem and can be described as:

$$\min_{a} f_s(a) \tag{3.2}$$
$$subject\,to : a \in \Gamma$$

where $a = [a_1, a_2, ..., a_p]$ is the vector containing our design variables within the feasible space Γ defined by the reduced problem, and $f_s(\cdot) = [f_{s1}(\cdot), f_{s2}(\cdot), ..., f_{sq}(\cdot)]$ is the vector of surrogate objective functions. This can then be solved using any of the available optimisation algorithms. The process of creating the reduced problem is called model order reduction and involves either reducing the number of design variables or objective functions from the original problem. In the first approach we solve the optimisation problem for a set of variables $a = [a_1, a_2, ..., a_p]$ which is a subset of the original set of variables $x = [x_1, x_2, ..., x_n]$. This subset is created by removing irrelevant and redundant variables to reduce the dimension of the search space. Removing these variables does not cause any significant loss of information. However this approach is not very effective in reducing the computational cost since any good designer will make sure that the design variables in the original problem are all relevant to the problem. For example, designers use methods such as sensitivity analysis before creating the synthesis problem to find the dependence of the objective functions on the design variables [45, 46]. Another salient point to mention is that removing design variables also means restricting our design space which might not be desirable to the client.

The other, and more useful approach is to reduce the number of objective functions instead [47–49], i.e. we solve for the set of functions $f_s(\cdot) = [f_{s1}(\cdot), f_{s2}(\cdot), ..., f_{sq}(\cdot)]$ instead of $f(\cdot) = [f_1(\cdot), f_2(\cdot), ..., f_m(\cdot)]$ where $f_s \subset f$. When solving for multiple objective functions, we want to operate in a completely independent objective space, i.e. we want the objective functions to be linearly independent of each other. Removing the irrelevant and

redundant objective functions significantly reduces the complexity of the synthesis problem, increases search efficiency and lowers computational cost. Unlike the first approach, we are not restricting the search space and by removing dependent objectives their information is not lost as we optimise them simultaneously through the other objectives in the reduced problem.

3.3 Surrogate Models

If we think about it, the software responsible for solving the computationally expensive analysis problem can be treated as a black box. As long as this black box gives us consistent outputs within our acceptable range of accuracy, it does not matter what is inside it. The objectives, constraints and optimisation algorithms do not depend on how it works. If an analogous black box can do the analysis problem more efficiently then we can use that instead of our original model. This analog is called a surrogate model. A surrogate model is essentially a model of a model or a further abstraction of the natural system. Fig 3.1 illustrates the concept of a surrogate model. Surrogate models usually have higher errors than the original model but can produce the outputs much quicker. If the errors are within a tolerable range, a surrogate model is a much better alternative.

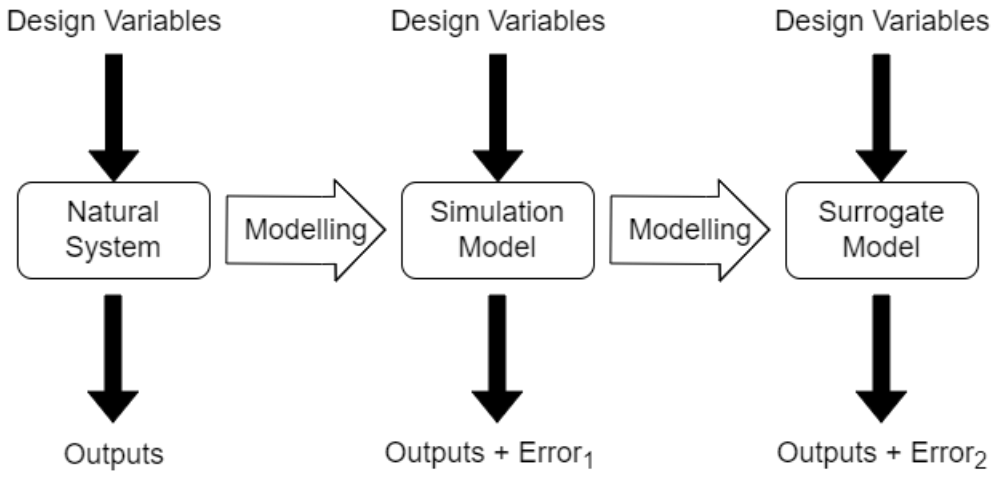


Figure 3.1: Surrogate Model

3.3.1 Physics based Surrogate Models

One type of surrogate model is where the mathematical model of the original is simplified. This type of surrogate model is still based on the physical laws and it can describe the physics of the machine. A great example of this is the Magnetic Equivalent Circuit (MEC). A MEC is similar to an electric circuit but where the electromotive force (emf) is replaced by magnetomotive force (mmf), the current is replaced by magnetic flux and resistance is replaced by reluctance. This allows a designer to translate the geometrical description of a magnetic device such as a motor with its coils, iron and permanent magnets into a simpler reluctance circuit [50, 51]. The magnetic fields of the original device can then be solved much faster using techniques used for simple electric circuits. Other examples of physics based surrogate models include space mapping (SM) [52, 53] and shape preserving response prediction (SPRP) [54] used in microwave engineering.

3.3.2 Reduced Order Models

Reduced Order Models are used to reduce the computational complexity of the model where numerical methods such as FEM and computational fluid dynamics(CFD) are used. Various methods such as orthogonal projection [55] and reduced basis methods [56] are used to reduce the number of nodes produced during the discretisation step of numerical methods. This lowers the number of local solutions that need to be obtained to model the entire model thereby reducing the computational costs.

Reduced order models are very sensitive to geometrical design. This is why they are not used for geometry optimisation, as when the shape of the model varies with the design variables then the mesh generated will not be the same for all the possible design solutions [57]. Another subtle distinction to make here is that the overlying model that is reduced is usually a physics based model but doesn't necessarily have to be.

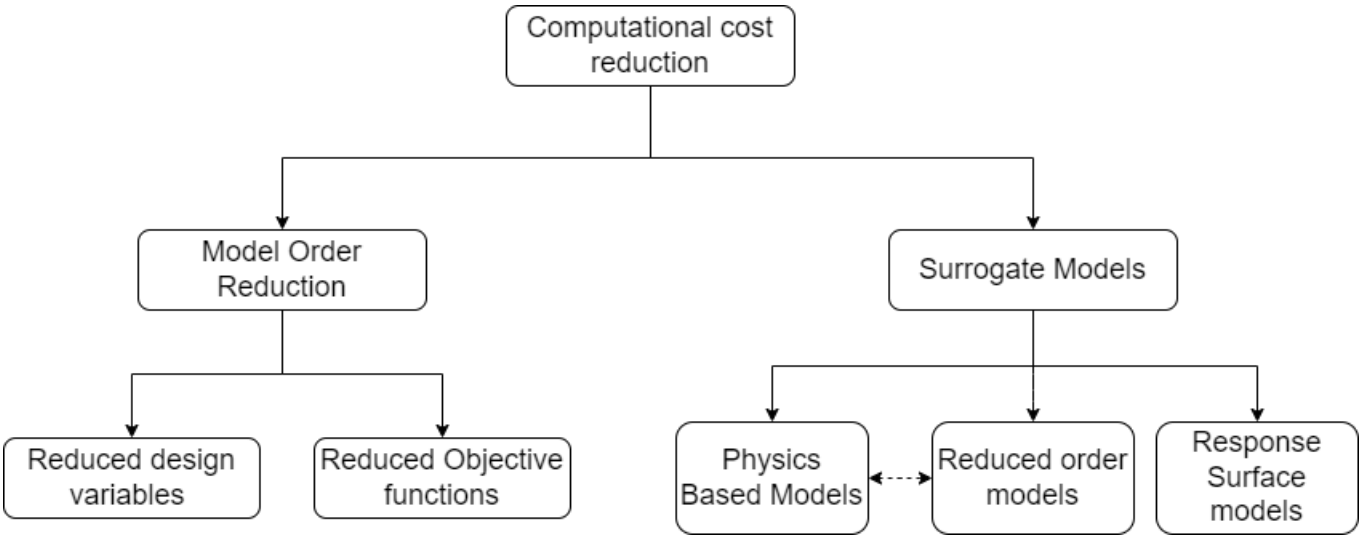


Figure 3.2: Ways to reduce computational complexity

3.3.3 Response Surface Models

A Surrogate model can be created using a response surface. This type of surrogate model maps the design variables to the outputs and only describes the relationship between them. This type of surrogate model does away with the underlying mathematical model of the original and does not describe the physics of the machine and how the design variables interact with each other. Simple response surface models that assume the nature of the relationship between the inputs and the outputs prior to modelling are the most common types of this surrogate approach [58]. One of the examples of this is using least squares estimation to fit polynomial models to the given data [59]. Polynomial models such as these were popular up to the late 1990s [60, 61]. However, it was soon observed that these polynomial models could only be used for unconstrained low order problems [59]. Also, assuming the nature of the relationship between the inputs and the outputs before modelling meant that the models did not work with nonlinear systems [62].

This led to the development of more flexible response surface models that did not assume any relationship between the inputs and outputs before modelling. These include kriging [63], radial basis functions (RBF) [64], Genetic algorithms [65], multivariate adaptive regression splines (MARS) [66], support vector regression (SVR) [67] and artificial

neural networks(ANNs) [68,69] etc. In this thesis, we will be focusing on artificial neural network based surrogate modelling. In the next few sections, we discuss neural networks in more detail and how they were used to create a surrogate model for the single stator double rotor AFPM we designed in chapter 2.

3.4 Artificial neurons and neural networks

An artificial neuron is a mathematical model of a biological neuron that mimics the functioning of its biological counterpart. The first model for an artificial neuron was proposed by McCulloch and Pitts in 1943. [70]. This artificial neuron model is also called a MCP neuron and can be mathematically described as:

$$y = \sigma(x) = \begin{cases} 1 & \text{if } \sum_{k=1}^{n} x_k > \Theta \text{ and } i = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where $x = [x_1, x_2, x_3, ..., x_n]$ and $x_k \in \{0, 1\}$ are the inputs to the MCP neuron. Looking at the equation, a MCP neuron can be described using the following rules:

- The output is binary, i.e. $y \in \{0, 1\}$, where $y = 0$ means the neuron stays at rest and $y = 1$ and means the neuron fires.
- There are n number of binary inputs.($x_k \in \{0, 1\}$)
- The neuron can fire only if $i \neq 0$ and i is called the inhibitory input.
- The neuron fires only if the sum of inputs is greater than a threshold value Θ

A MCP neuron could model any linear boolean function such as AND, OR , NOT. However, the MCP neuron had several limitations:

- A MCP neuron can only model linear boundaries and cannot represent any nonlinear functions such as XOR.

- There is no learning algorithm, i.e. the function cannot be learned from data and must be hard coded beforehand.
- All the inputs have the same weight equal to one, thus implying that all the inputs to the neuron make equal contributions to the output.
- The inputs cannot be non boolean.

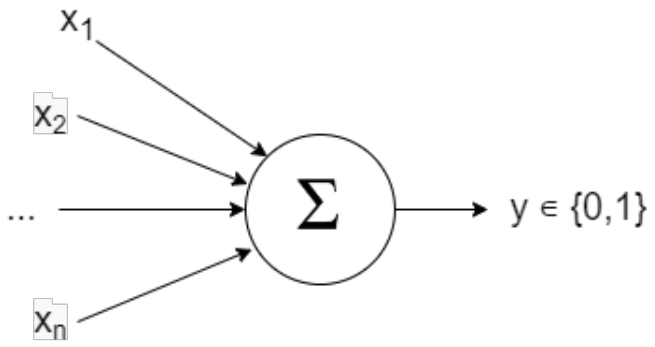


Figure 3.3: MCP neuron

3.4.1 Perceptron Model

The MCP neuron was improved upon by Rosenblatt in 1957 when he came up with the 'perceptron' [71]. Mathematically, the perceptron model of the neuron can be defined as a classifier of the form:

$$y = sgn(w^T x + b) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.4}$$

where $x = [x_1, x_2, x_3, ..., x_n]$ is the vector of inputs, $w = [w_1, w_2, w_3, ..., w_n]^T$is the vector of synaptic weights and b is the bias term or offset from origin. Some important differences from the MCP neuron were:

- The neuron takes an extra constant input of the bias or synaptic weight b. Compared with the MCP neuron, the bias term b is the negative of the activation threshold.
- The perceptron allows some inputs to have more influence on the output than others by having the synaptic weights w not restricted to unity.
- The inputs are not restricted to boolean or even strictly positive numbers.

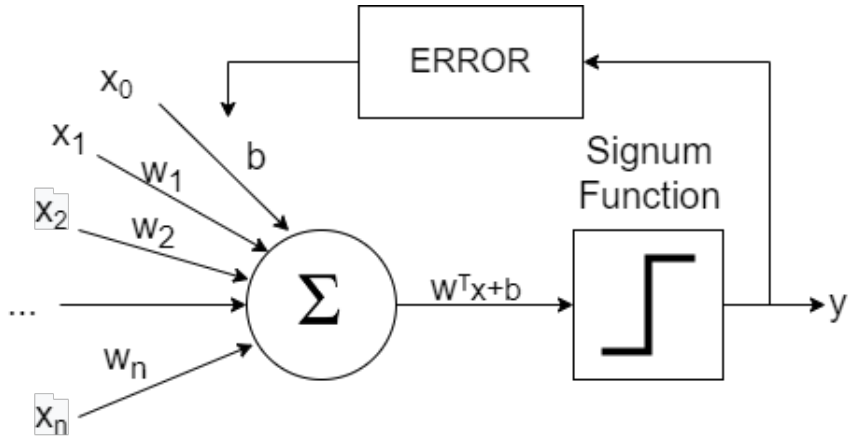


Figure 3.4: Perceptron model of the neuron

The decision boundary for the perceptron is given by $b+w_1x_1+w_2x_2+w_3x_3, ..., w_nx_n = 0$ or more simply $b + w^Tx = 0$. A single perceptron is a binary classifier that represents a linear boundary since the decision function depends linearly on the inputs.

A major improvement of Rosenblatt over the MCP neuron was not just its ability to perform flexible binary classification but also to come up with an algorithm enabling the perceptron to learn from data and adjust the synaptic weights, similar to how the biological neurons adapt during the learning process. We could give the model a set of classifiable inputs and expected outputs and the perceptron could learn the linear boundary that separated the inputs. This type of learning algorithm is also called a supervised learning algorithm. Algorithm 1 illustrates the simple learning capabilities of a perceptron.

Algorithm 1: Perceptron Learning Algorithm

input : Training Examples: x_i, y_i where $i = 1\ to\ m$
Initialise w and b randomly
while *not converged* **do**
  for $j = 1, m$ **do**
    $error = y_j - sgn(w^T x_j + b)$
    if $error \neq 0$ **then**
      $w = w + error \times x_j$
      $b = b + error$
  Test for convergence
output: Set weights w and bias b of perceptron

One major issue with the perceptron model that came up was that of convergence. Since the decision boundary was linear it would converge only if the two classes to be predicted were linearly separable and would not work for non-linear boundaries [72].

The next step was to go away from the signum function to other activation functions. Activation functions are functions that decide whether a neuron fires or not. Instead of using the signum function as the activation function, we can use other activation functions such as linear [73] or sigmoid [74] functions that are continuous and differentiable. This property allows us to define a cost function that we can minimize in order to update our weights. The cost function is then minimised using the well-known conjugate gradient method [40].

For a convex cost function for each weight, the principle behind gradient descent can be described as climbing down a hill until a global or local minimum is reached. After each iteration, we take a step in the opposite direction of the gradient, and the step size is determined by the slope of the gradient and the value of the learning rate:

$$\Delta w = -\eta \nabla J(w) \tag{3.5}$$

where η is the learning rate and $J(w)$ is the cost function. Thus, we have to compute the partial derivative of the cost function for each weight in the weight vector: $\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$. This reiterates our point that we need differentiable activation functions in order

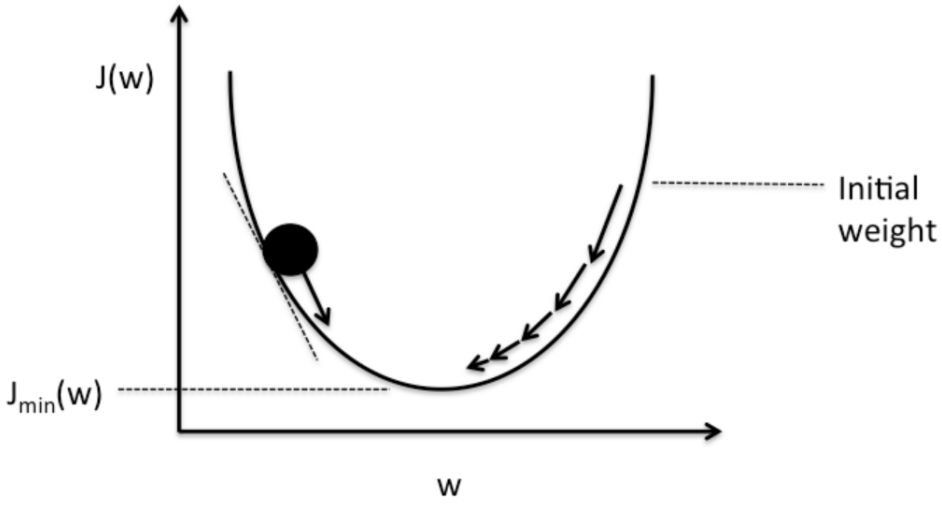
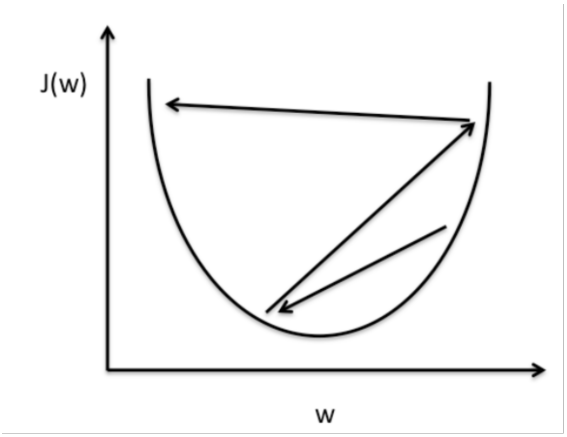
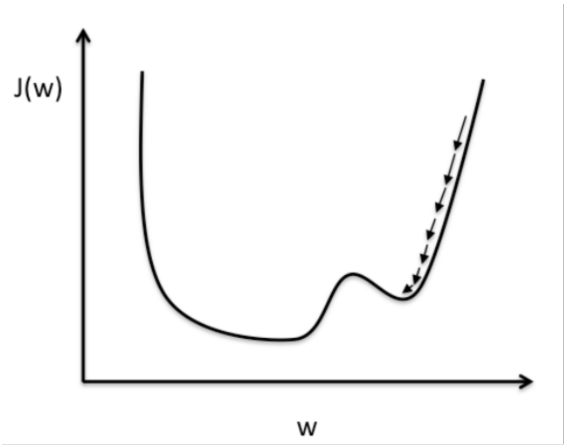


Figure 3.5: Gradient descent [75]

to use this method. The learning rate η is also an important parameter that controls the step size of the algorithm. If the learning rate is too small then the algorithm may take too many iterations to converge or become trapped in a local minimum. Conversely, if the learning rate is too large the algorithm will overshoot the minimum and never converge.



(a) Overshooting due to large learning rate



(b) Excessive iterations or trapping in local minima due to small learning rate

Figure 3.6: Effects of Learning rate on conjugate gradient method [75]

3.4.2 Multi Layer perceptron (MLP)

The next significant advancement came in the form of multi layer perceptron networks [76] that combine several layers of perceptrons with sigmoid activation functions to create a multi layered neural network. Fig 3.7 illustrates such a neural network.

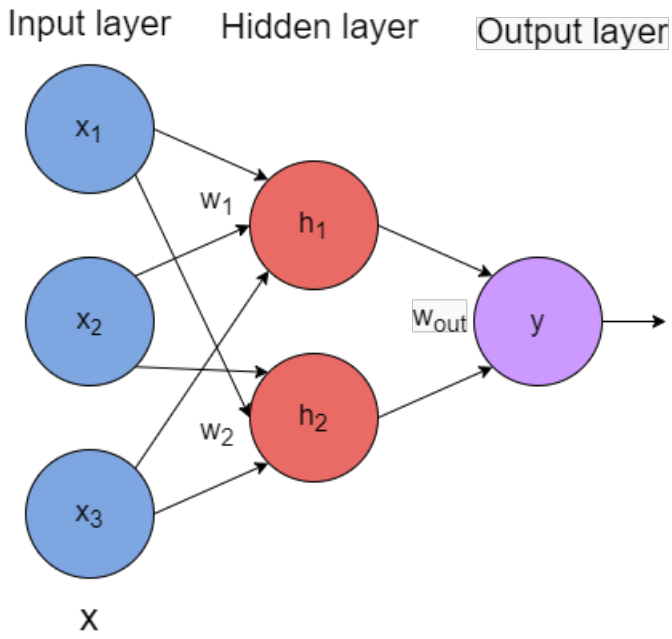


Figure 3.7: A multi layer perceptron model

- All the neurons have sigmoid activation functions except the neurons in the output layer.
- The output layer can have the identity activation function so that the network predicts a continuous value and becomes a regression model or it can have the sigmoid activation function to act as a classifier.
- There can be multiple hidden layers. The number of hidden layers plus the output layer is defined as the depth of the network.
- The output layer can have multiple neurons for multi label classifications and multiple regression.

- The output of neurons in layer j becomes the input to the neurons in layers j+1. This is why these networks are also called feed forward neural networks.
- There are no backward/recurrent connections.
- No cross connection between neurons in the same layer.
- If all neurons in layer j provide input to all neurons in layer j+1 then the network is called a fully connected network.

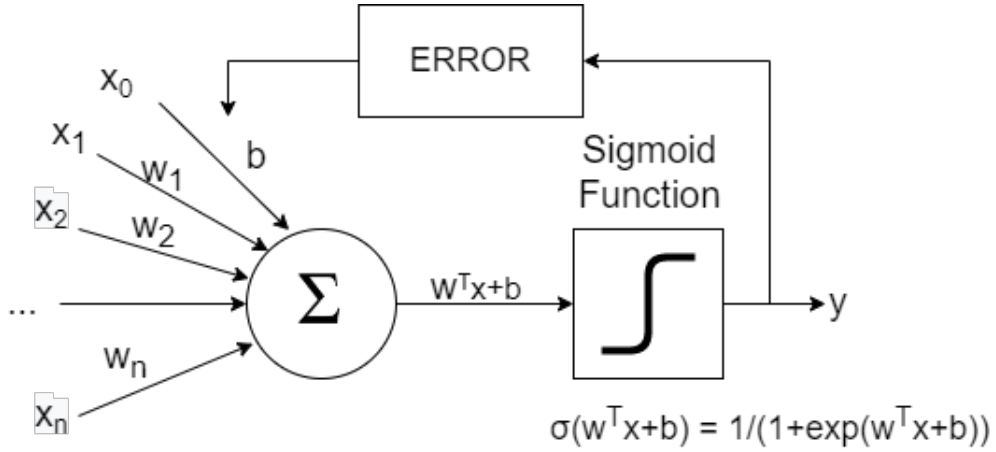


Figure 3.8: Neuron with sigmoid activation

Similar to the perceptron, here we need to train the entire network using supervised learning so that we can find a good set of weights to minimize the error at the output of the network. [76] also derives a method known as backpropagation to jointly train the entire network using gradient descent. Again gradient descent is preferred here since the sigmoid function (Fig 3.8) can be easily differentiated since $\frac{d\sigma(z)}{d(z)} = \sigma(z)(1 - \sigma(z))$. The backpropagation algorithm initializes the weights biases to random values and then does a forward pass by calculating the output by sequentially computing each layer. It then works by using the chain rule to compute the gradient of the loss function for each weight, one layer at a time and iterating backwards from the last layer. One complete forward and backwards pass through the network is called an epoch. Algorithm 2 gives the pseudo-code for the backpropagation algorithm during each epoch. Generally while

training neural networks we can also specify the number of epochs to train the network after which the training is stopped.

Algorithm 2: Backpropagation Algorithm with stochastic gradient descent

input : Training Examples: x_i, y_i where $i = 1\ to\ m$
Initialise w and b randomly, choose loss function J
while *not converged* **do**
  for $i = 1, m$ **do**
    Feed x_i through network to compute output y
    For the output unit, compute: $\frac{\partial J}{\partial w_{out}}$
    For each hidden unit j, compute the gradient using chain rule: $\frac{\partial J}{\partial w_j} = \frac{\partial J}{\partial y}\frac{\partial y}{\partial w_j}$
    Update each network weight: $w_j = w_j - \eta\frac{\partial J}{\partial w_j}\forall j$, $w_{out} = w_{out} - \eta\frac{\partial J}{\partial w_{out}}$
  Test for convergence

The multi layer perceptron model can be further generalised by using activation functions other than the sigmoid function. The most popular alternatives are the hyperbolic tan function, $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ and the rectified linear unit or ReLU $ReLU(z) = max(z, 0)$. Both the sigmoid and tanh functions have a problem with saturation [77]. This means that for large values of input they snap to 1 and for very small values they snap to -1 or 0 for tanh and sigmoid respectively. Once saturated, the algorithm cannot adapt the weights to improve the performance of the model. The other problem with the sigmoid and tanh functions arises as the depth of the network increases. In very large networks, these activation functions fail to update the gradient. Error decreases dramatically with each additional layer through which it is propagated, as the derivative of the chosen activation function vanishes. This is called the vanishing gradient problem. For this reason, modern feed forward networks use ReLU and it has become the standard for deeper networks as it never saturates due to its unbounded range and forces negative inputs to 0 thereby preventing the vanishing gradients. [78].

Nowadays, open source software libraries such as Tensorflow [79] have automated differentiation frameworks where the derivative computations are handled by the software library itself and the only things we need to specify are the network architecture, the training and testing data sets and the hyperparameters for the optimisation algorithm.

The training and test data sets are self-explanatory. The training data is used to train the network and adjust the weights while the test data is used to evaluate the performance of the network after training is complete by comparing the predictions of the network with the expected outputs.

In terms of hyperparameters we need to decide on the learning rate η and the batch size of the training data that is sent into the network per epoch. Both of these influence the step size of the algorithm and the overall runtime of training the network. Depending on the batch size, the gradient methods are classified as:

- Stochastic gradient descent: The error is computed on a single training example before updating the weights. This method is very noisy which reduces generalisation error, increases runtime and may even prevent the algorithm from converging if the steps start going back and forth.

- Batch gradient descent: Passes all of the training data before computing the error. All of the training data is fed to the network and gradients are calculated. The average of the gradients of all the training examples is used to update the weights. It has a straight trajectory towards the minimum and it is guaranteed to converge in theory to the global minimum if the loss function is convex and to a local minimum if the loss function is not convex.

- Mini Batch Gradient: Error is computed on a random small subset of the training data before updating the weights. The batch sizes are usually chosen as a power of 2 because some hardware such as GPUs achieve better run times with batch sizes that are a power of 2 such as 32, 64, 128 etc. This method tries to combine the advantages of both, batch and stochastic methods.

The learning rate η is very important since backpropagation is very sensitive to the learning rate. Making it too large will cause the algorithm to diverge while making it too small will make the learning algorithm very slow to converge. All the modern software libraries for neural networks have adaptive optimisation algorithms built into them.

These algorithms modify the learning rate dynamically, depending on how the training is progressing. This is unlike the gradient methods where the learning rate is fixed for the entire duration of training. Adam [80], RMSProp [81] , and AdaGrad [82] are popular approaches, with Adam now being the standard method in deep learning. Intuitively, these methods try to increase the update strength for parameters that have had smaller updates in the past by looking at statistics of the history of gradient updates for that parameter.

Another problem that models like MLPs can have is underfitting or overfitting. These have to do with how well the model approximates our target function. If the network cannot fit to the training data well then the network is said to be underfitting. If the network fits to the training data too well it will also end up modelling the noise in the training data and this is called overfitting. In both cases, the networks will not generalise well to new data and the predictions will have high error. Underfitting is easy to detect with suitable evaluation metrics and will have poor performance on the training data. Overfitting is harder to detect as training the network for too long may cause the model to learn the noise in the training data. To detect overfitting, neural networks use a method known as validation. In addition to the test data, we can keep another set of data known as validation data separated from the model. After each training step, the loss is calculated on the validation data in addition to the training data, but the validation loss is not used to change the weights of the network. By observing how the model is performing on unseen data after each training step we can keep track of how well the model is generalising. If the training loss decreases and the validation loss increases over the course of training we can say that the network is overfitting.

Finally, If a single layer of neurons can represent any linearly separable region then the multi layer perceptron can be used to represent convex regions. In fact, it has been shown that an MLP with two hidden layers is sufficient for creating classification regions of any desired shape [83] and even that MLPs are universal approximators and with one hidden layer, an MLP can approximate any function that contains a continuous mapping

from one finite space to another [77]. This makes neural networks an excellent tool to create a response surface model using our AFPM model database from chapter 2.

3.5 Creating a Neural Network Surrogate Model

In this section, we will discuss how the single stator double rotor AFPM model and database of results from chapter 2 were used to create a surrogate model using a simple feed forward artificial neural network.

AIR GAP(mm)	PM MATERIAL	PM THICKNESS(mm)	PM ANGLE(degree)	Torque(Nm) at 0°	Torque(Nm) at 5°	Torque(Nm) at 10°	Torque(Nm) at 15°
7.5	Samarium Cobalt: 26/10	3.0	25.0	51.395524	40.668993	35.002521	24.975975
3.5	AL9N: Alnico-9NB	8.5	20.0	58.196534	47.751059	36.787559	18.282537
1.5	AL9N: Alnico-9NB	7.0	45.0	236.906900	104.198767	92.808676	99.363091
2.0	Ceramic 8A	2.0	20.0	12.111604	9.175845	6.069192	4.071529
4.5	Samarium Cobalt: 26/10	7.0	45.0	286.662844	195.246598	176.030693	171.241482
1.5	Samarium Cobalt: 26/10	9.5	30.0	418.104345	346.323157	340.546287	311.786728
7.0	Ceramic 8A	4.5	30.0	16.276199	13.491713	11.718570	9.769500
6.0	Neodymium Iron Boron: 32/31	9.0	45.0	348.638995	243.527205	231.822986	224.456587
6.5	Samarium Cobalt: 26/10	7.5	15.0	111.111509	85.419852	67.317134	43.570838
3.0	Ceramic 8A	2.5	10.0	5.189733	2.782050	2.476323	2.439047

Table 3.1: Format of raw data

3.5.1 Feature Engineering

The first step was to organise the data into a format that can be accepted by the neural network and can be used in supervised learning. Table 3.1 shows the format of our raw data. The air gap, PM material and PM angle are our input features while the torque value on the rotor at different positions will be our output features. All of our features are in an acceptable format except the PM material. The values of PM material are stored as a string data type. However, Neural Networks only accept data in a numerical format. This means that we need to replace this feature with dummy variables. A dummy variable is a variable that takes values of 0 and 1, where the values indicate the presence or absence of a categorical feature.

Table 3.2 illustrates how the PM material feature is encoded into three dummy variables. Even though the PM material can take four possible values (see Table 2.2), we

Ceramic 8A	Neodymium Iron Boron: 32/31	Samarium Cobalt: 26/10
0	0	0
0	0	0
0	1	0
1	0	0
0	0	1
0	0	0
1	0	0
0	1	0

Table 3.2: Dummy Variables for PM material feature

created only three dummy variables since the absence of all three of them indicates the presence of the fourth and so, adding a fourth dummy variable would be redundant. The original PM material feature is then dropped and the dummy variables are added to the data.

Next, we split our data into training, test and validation data sets. We took 50 percent of the data at random to be our training set and the remaining 50 percent was divided into two sets of 25 percent each as our test and validation data. This leads to 7072 training samples and 3536 test and validation samples. The test data will not be shown to the neural network at all and will only be used to evaluate the neural network after training is complete.

Since our input features have different scales, our data points are far from each other and we need to scale/normalize the data. Large differences between the data points make the model assign larger weights which is often unstable and makes it harder for our model to understand the data. This can make the model perform poorly during training and produce poor results with lowered accuracy. Scaling reduces the distance between the data points and improves the overall accuracy of the model. Our input features for all three sets of data were scaled using min-max normalization which scales all the values between 0 and 1. The general formula is given by:

$$x_n = \frac{x - min(x)}{max(x) - min(x)} \tag{3.6}$$

where x_n is the normalised value and x is the original value. An important point to note here is that while scaling the input features, the maximum and minimum values used are taken from only the training set to prevent the leakage of information about the test and validation data to the neural network.

3.5.2 Creating and training the Neural Network

An open source software package called Tensorflow [79] is used to create a simple feed-forward neural network.

- Input layer: 6 neurons, activation function: Rectified Linear Unit(ReLu)
- Hidden layer: 5 neurons, activation function: Rectified Linear Unit(ReLu)
- Output layer: 4 neurons, activation function: Identity function.
- Optimiser: Adam [80]
- Output Loss function: Mean squared Error

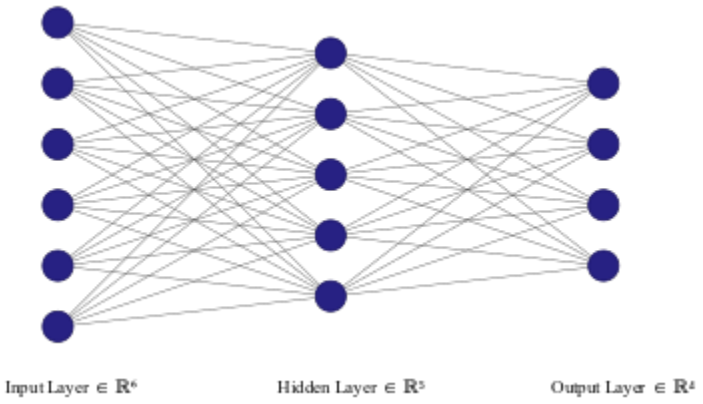


Figure 3.9: Architecture of our neural network

The computer system used was a free cloud platform provided by Google through their "Colaboratory, or Colab" service [84]. The specifications of the system provided through the cloud are:

- CPU: 2 $\times$ Intel(R) Xeon(R) CPU @ 2.20GHz
- Total Physical Memory(RAM): 13298580 KB or 13.29858 GB
- GPU: NVIDIA Tesla T4
- GPU memory: GDDR6 with 16 GB

Fig 3.9 illustrates the architecture of our neural network. The input and output layers have neurons equal to the number of features in one data point of the input(i.e. air gap, PM angle, PM thickness and the three dummy variables) and output respectively (the torque on the rotor at different positions). Several general guidelines can be found in the literature when trying to decide the number of hidden neurons: the size of the hidden layer has to be somewhere between the size of the input layer and the output layer [85], it should be less than twice the size of the input layer [86] and we need as many hidden neurons as the dimensions needed to capture 70-90% of the variance of the input data [87]. Here we try to follow these guidelines and start with 5 hidden neurons. The loss function for the final output is chosen to be the mean squared error since our network is a regression model where predictions are continuous values. Similar to the sigmoid function used for networks that do classification, the means squared function is easily differentiable which allows the network to use gradient descent for error minimisation.

The network was then trained with the training samples for 400 epochs with a batch size of 128 to create our final surrogate model for the single stator double rotor AFPM.

3.6 Summary

In this chapter we discussed optimisation in the modelling process and how the computational cost of designing and simulating the model can be reduced. We discussed surrogate models as a way to reduce the computational burden of simulating designs. We further explained the different types of surrogate models and how neural networks can be used as a response surface surrogate model. Finally, neural networks were explained in more

detail and how a neural network was created to model the AFPM from chapter 2. In the next chapter, we will discuss how to evaluate neural networks and evaluate our network and discuss any other observations on the results and data.

Chapter 4

Results and Analysis

4.1 Evaluation of Models

Most quality metrics for surrogate models are the same ones used for all models in general. The basic rationale is to observe if the responses produced by a model match the observations from the natural system. However, simply computing the difference between the prediction and observation values does not provide an accurate overview of a model's performance since models may be accurate for a given set of inputs but perform poorly for another set. In 1981 scientists working on air quality models compiled and suggested the following metrics for evaluating models [88]:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- Pearson's product moment correlation (r) and coefficient of determination (r^2)
- Error variance
- Average error

These metrics were then generalised to be used with any model and other metrics such as Mean Absolute Percentage Error (MAPE), Maximum Absolute Error (MAX) and Root

Mean Squared Percentage Error (RMSPE) were also later discussed [89–91]. Average error and error variance are not very popular now since they do not tell us anything about model performance and only characterise the distribution of the error. For example, if the average error is zero it could mean the model has no error or that the model has an equal amount of positive and negative error. For regression models, the error based metrics such as RMSE, MAE,MAPE etc. and the coefficient of determination r^2 have remained popular.

Coming to surrogate models, the only difference in using these metrics is the frame of reference. The predictions of the surrogate model are compared with the predictions of the CAD based simulation model. [92] discusses using MAE and r^2 for a linear regression model. In addition to these error based metrics other criteria for evaluating surrogate models were also suggested in [93]:

- Simplicity: How easy is it to implement the model.
- Robustness: How much is the modelling process problem dependent.
- Transparency: How much information can the model provide about the relationships between design variables and objectives.
- Efficiency: How computationally expensive is it to create the model and simulate it.

4.1.1 Error based metrics

Let $y = [y_1, y_2, ..., y_n]$ be the vector of outputs of the original model and $y = [y_{s_1}, y_{s_2}, ..., y_{s_n}]$ be the outputs of the surrogate model. Then the various error based metrics can be defined as:

- Mean Absolute Error (MAE):

$$E_{MAE}(y, y_s) = \frac{\sum_{i=1}^{n} |y_i - y_{s_i}|}{n} \tag{4.1}$$

- Mean Squared Error (MSE):

$$E_{MSE}(y, y_s) = \frac{\sum_{i=1}^{n}(y_i - y_{s_i})^2}{n} \tag{4.2}$$

- Root Mean Squared Error (RMSE):

$$E_{RMSE}(y, y_s) = \sqrt{E_{MSE}(y, y_s)} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - y_{s_i})^2}{n}} \tag{4.3}$$

- Mean Absolute Percentage Error (MAPE):

$$E_{MAPE}(y, y_s) = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(y_i - y_{s_i})}{y_i}\right| \times 100 \tag{4.4}$$

- Maximum Absolute Error (MAX):

$$E_{MAX}(y, y_s) = \max_{i \in 1....n} |y_i - y_{s_i}| \tag{4.5}$$

RMSE and MAE are generally preferred over MSE since the error has the same units as the outputs of the model. MSE is useful as a loss function while training models since it's a convex, symmetric and differentiable function [94]. RMSE and MAE differ in the type of distribution they try to describe. MAE is good for uniform distributions while RMSE is suitable for normal distributions [95]. RMSE also gives more weight to larger absolute errors while MAE gives equal weight to all errors. [96]

MAPE is independent of the scale of the units and is a more intuitive metric when evaluating models that predict outputs with large variations in range. MAX gives us the upper bound of the prediction error but is sensitive to outliers since even a single outlier in the data may give a very high value of MAX.

4.1.2 Coefficient of Determination (r^2)

The coefficient of determination is a measure of accuracy that can be defined as:

$$r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_{s_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{4.6}$$

where $\overline{y}$ is the mean of the vector y. It represents the proportion of variance in the output y that can be explained by the model. It gives a measure of the goodness of fit and hence, a measure of how well the model predicts unseen samples [97]. r^2 has a range of $(-\infty, 1]$ with a value of 1 meaning that the model explains a hundred percent of the variance in the data. It is also insensitive to scaling, the value of r^2 does not change if y and y_s are scaled by the same factor.

4.2 Evaluating the Neural Network Surrogate Model

4.2.1 Overfitting

To detect if the model was overfitting, we tracked the validation loss along with the training loss per epoch during the training period. Fig 4.1 shows the change in validation and training loss over the training period. The validation and training loss follow the same trajectory until the end of training and in our model and we can see that the network is not overfitting to the training data.

4.2.2 Error

	r^2 score	MAPE	MAE	RMSE
Torque(Nm) at 0 °	0.967504	0.240850	21.750740	36.900560
Torque(Nm) at 5 °	0.951535	0.344496	21.056612	34.720123
Torque(Nm) at 10 °	0.945729	0.341104	21.287058	35.724224
Torque(Nm) at 15 °	0.941349	0.483233	21.202178	36.213670

Table 4.1: Performance Metrics for the Neural Network

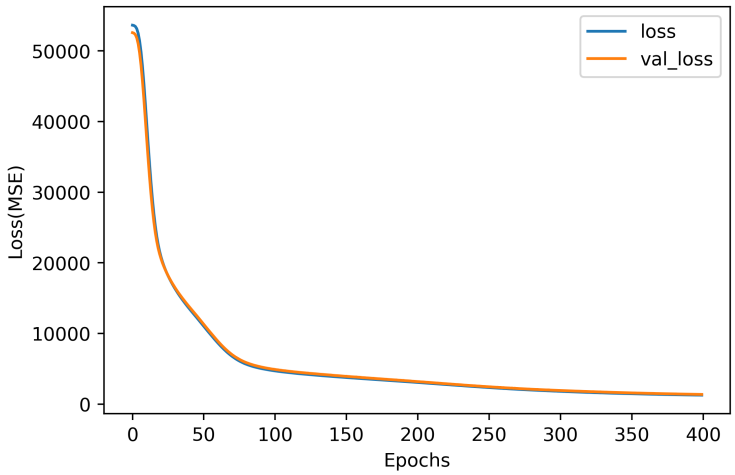


Figure 4.1: Training and Validation loss over the training period

MAE, RMSE, MAPE and r^2 scores were determined for the four outputs of the neural network. The values are shown in Table 4.1. Fig 4.2 illustrates the distribution of the torque values in the whole data set that was produced by the high-fidelity model. Since the values to be predicted by the neural network vary over a large range, MAE and RMSE are not as useful to determine the performance of the network.

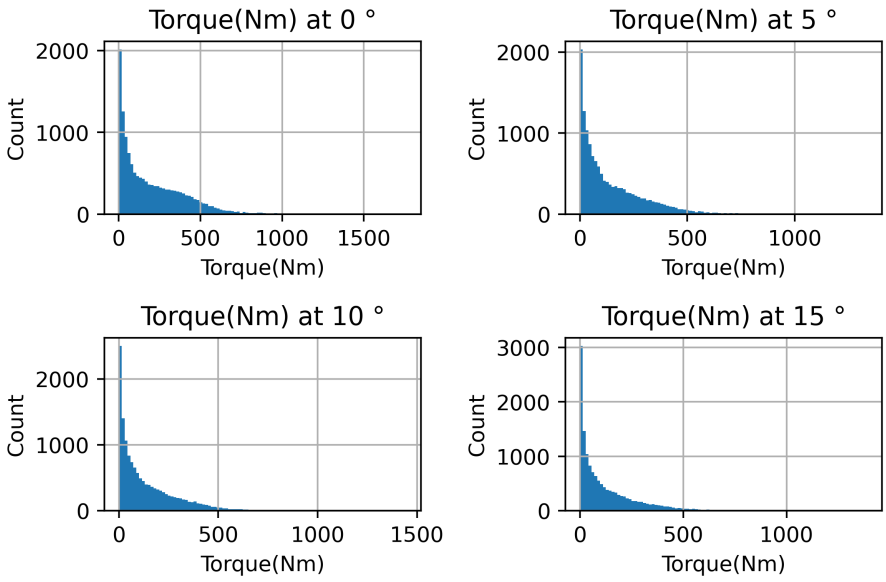


Figure 4.2: Distribution of Output Torque Values produced by the high fidelity model

MAPE and r^2 scores are scale independent and give a better measure of the performance of the network. Fig 4.3 illustrates the distribution of the predictions of the neural network about the output predicted by the high fidelity model. The network fits well to most of the data but does not fit well when the values of output torque are at the higher end of the range. This is because the data set does not have enough examples with very high torque values to sufficiently train the network.

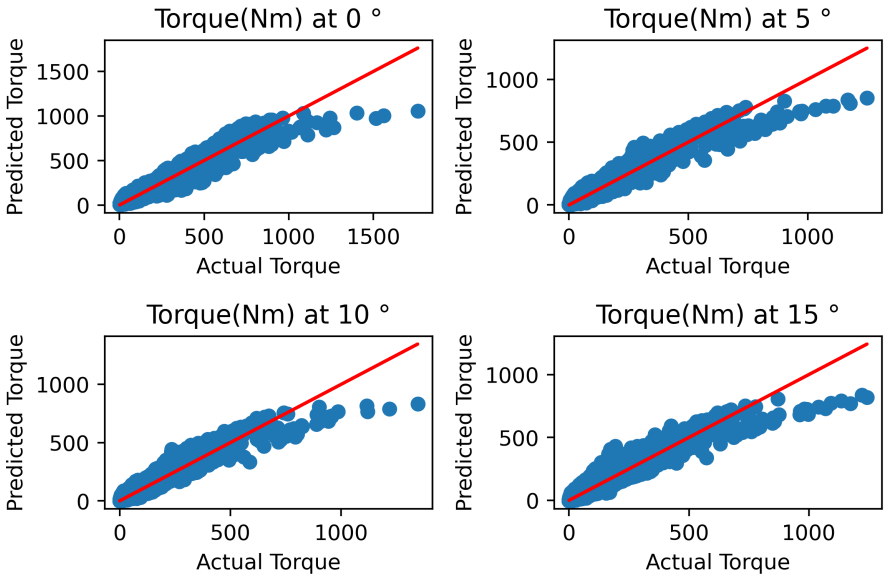


Figure 4.3: Scatter plot of the Predictions versus True Output

Fig 4.4 gives the corresponding distribution of the percentage errors for all the predictions. From the distribution and low MAPE values, we can conclude that the neural network approximates the original high fidelity model to a high degree of accuracy. Moreover, the values of r^2 score are very close to one which means the neural network fits the data well, predicts well on unseen data and explains a large proportion of the variance in the data.

However, it is important to note that there are a lot of designs in the data set that have very low values of output torque (see Fig 4.2). These designs with torque values close to zero are not very useful. Their presence might also skew the performance of the surrogate model where the network predicts well on these designs but might not be

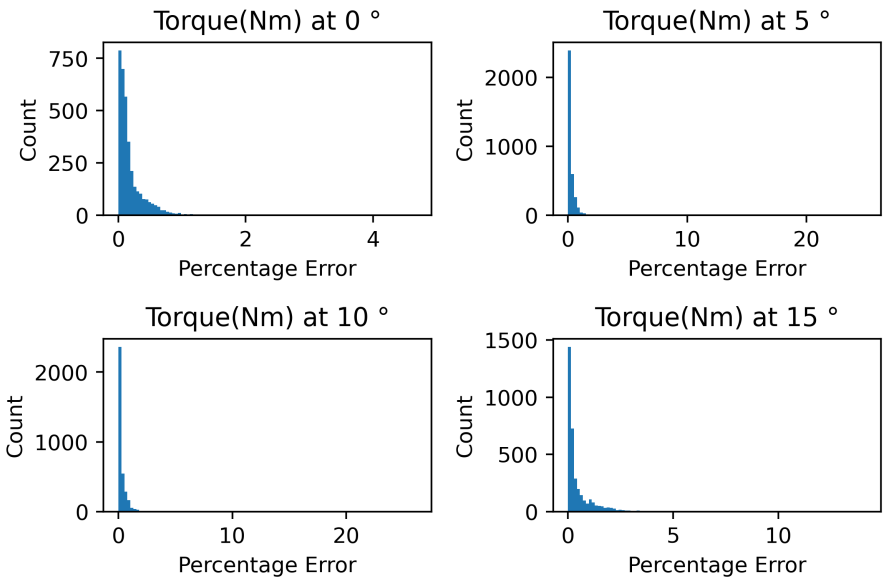


Figure 4.4: Distribution of Percentage Error

performing well on the other data. To take this into account, designs with torque values less than 50 Nm at the 0° position were removed. This reduced the number of designs in the data set to 10103 samples. Using the same 50:25:25 split between the training, test and validation data sets , the network was retrained and all the metrics were calculated again.

	r^2 score	MAPE	MAE	RMSE
Torque(Nm) at 0 °	0.894807	0.213017	44.180711	63.345560
Torque(Nm) at 5 °	0.890807	0.268578	36.365526	51.844716
Torque(Nm) at 10 °	0.887601	0.309928	37.031849	51.440558
Torque(Nm) at 15 °	0.873618	0.412601	38.134065	54.843100

Table 4.2: Performance Metrics with Modified Data Set

Table 4.2 gives the error metrics of the surrogate with the modified data set. The values of MAE and RMSE increase since the network is predicting on higher values of torque. Fig 4.5 illustrates the distribution of the predictions with this new data set. The r^2 scores reduce a little due to the increase in variance in the overall data caused by the removal of the low torque samples. The values of MAPE remain comparable to the values obtained with the original data set and the network still predicts with a high level of accuracy. Fig

4.6 shows the distribution of percentage error for all the predictions with the modified data set.

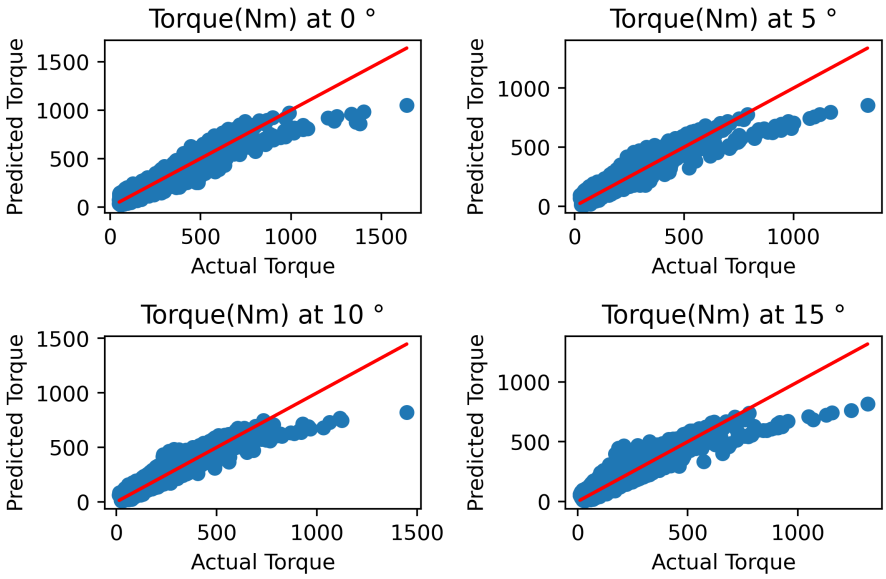


Figure 4.5: Scatter plot of the Predictions versus True Output with Modified Data Set

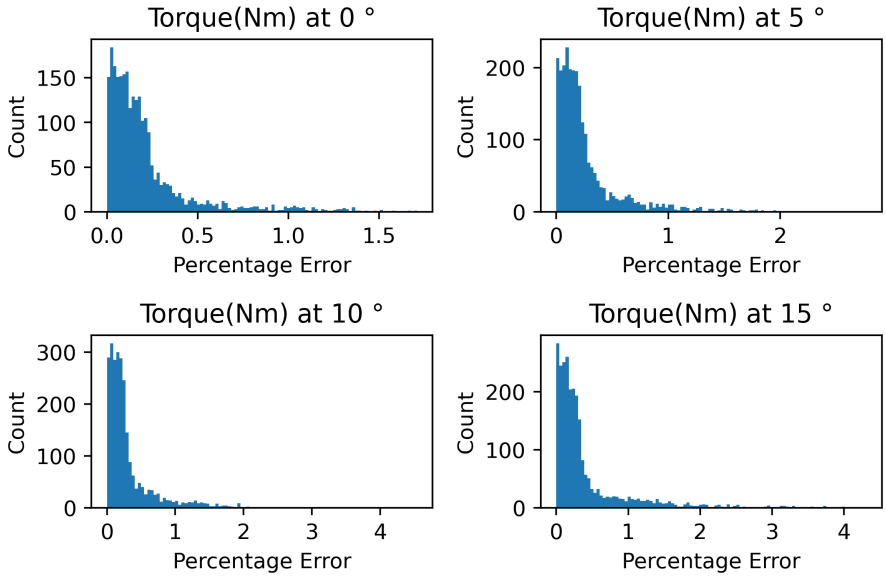


Figure 4.6: Distribution of Percentage Error with Modified Data Set

4.2.3 Computational Cost

The neural network surrogate is very computationally efficient when compared to the high fidelity model. The neural network architecture was set up and trained over 400 epochs in around 80 seconds. Each prediction on the neural network surrogate takes only 0.05 seconds which is a big cost reduction compared to the high fidelity model, where setting up the model takes several minutes and the simulation of a design takes 30-40 seconds.

4.3 Summary

We discussed the history and nuances of the various metrics used to assess the quality of surrogate models. Based on our dataset we chose MAPE and r^2 score as the best metrics to assess the performance of our neural network surrogate model. Low values of MAPE and r^2 scores of ≈ 1 show that the neural network creates a response surface with high accuracy. Combining this with the computational efficiency of training and predicting on a neural network, we have a fast and accurate surrogate model for our AFPM that can be used as an excellent design tool to predict the performance of possible designs in the design space.

Chapter 5

Conclusion

Axial flux machines(AFMs) are making a resurgence with advancements in manufacturing and material technology. Research is being done on various aspects of their design and various applications have been found where they have a huge advantage mainly in terms of compactness of the design and torque density for their size. Axial flux permanent magnet machines (AFPMs) are one of the most popular subsets of AFMs and have received extensive research into their design and performance. The computer aided design process has also become an indispensable part of the design process. Using various numerical methods, it is possible to computationally simulate machines to a high degree of accuracy. Thus, a large part of the design process is now computer aided and AFMs/AFPMs are no exception. Research into computationally simulating AFPMs is necessary for the end goal of optimising their design and topology.

Unfortunately, high fidelity models of AFPMs tend to be computationally expensive to simulate. Moreover, if accuracy is paramount then AFPMs cannot be simply reduced to two-dimensional models without some loss in accuracy due to various three-dimensional effects owing to their unique geometry. There is a significant research gap in terms of methods to reduce the computational burden of simulating these models. Looking into the various methods of optimising the modelling process, surrogate models are a popular method to reduce the cost of simulating the model by replacing the high fidelity model

with a cheaper to compute alternative that gives analogous results. Response surface models are a type of surrogate model that do not model the underlying physics of the machine but map the design variables to the outputs. This thesis describes the creation of a neural network response surface surrogate model to emulate the electromagnetic simulation of a AFPM. A high fidelity 3D model of a AFPM was created, parameterised and simulated to create a data set of observations. This data set was used to train and test the performance of a feed forward neural network.

Various metrics to assess the quality of surrogate models are discussed. Evaluating the neural network surrogate shows that the model can emulate the high fidelity model to a high degree of accuracy while making predictions in a fraction of the time. This neural network can then be used as a design tool to assess the performance of various possible design solutions within the design space defined by the parameters of the original high fidelity model but without actually running a physics based simulation.

5.1 Further studies

One limitation to creating the data set was running a full motion solver over the parameterised model. We had to compromise by instead running a static solver to reduce the computational cost of simulating all the possible combinations. Running a full motion solver will allow us to graph the torque speed curve for each design and better match design requirements.

We can also try to create a 2D model of the AFPM to reduce the cost of the transient motion simulation, provided that the two-dimensional model is accurate enough compared to the 3D model and within the tolerance set by the designer.

Reducing the computational cost will also allow for expanding the parameter set. In this thesis we only explored a subset of axial flux machines. By introducing more design variables such as number of poles, magnet shapes and structures, number and type of rotors and stators etc., a surrogate that encompasses other types of AFPMs or even different

types of axial flux machines such as axial induction machines, can be created. Another extension of this work is setting up a multi objective design problem instead of just looking at values of torque and optimising using the various algorithms in the literature.

Bibliography

[1] Amin Mahmoudi, Solmaz Kahourzade, Nasrudin Abd Rahim, Hew Wooi Ping, and Mohammad Nasir Uddin. Design and prototyping of an optimised axial-flux permanent-magnet synchronous machine. *IET Electric Power Applications*, 7(5):338–349, 2013.

[2] Surong Huang, Jian Luo, F. Leonardi, and T.A. Lipo. A comparison of power density for axial flux machines based on general purpose sizing equations. *IEEE Transactions on Energy Conversion*, 14(2):185–192, 1999.

[3] Jie Mei, Christopher H. T. Lee, and James L. Kirtley. Design of axial flux induction motor with reduced back iron for electric vehicles. *IEEE Transactions on Vehicular Technology*, 69(1):293–301, 2020.

[4] Babak Dianati, Solmaz Kahourzade, and Amin Mahmoudi. Optimization of axial-flux induction motors for the application of electric vehicles considering driving cycles. *IEEE Transactions on Energy Conversion*, 35(3):1522–1533, 2020.

[5] Zhi Cao, Amin Mahmoudi, Solmaz Kahourzade, and Wen L. Soong. An overview of axial-flux induction machine. In *2021 31st Australasian Universities Power Engineering Conference (AUPEC)*, pages 1–6, 2021.

[6] Émile Alglave and J. Boulard. *The Electric Light: Its History, Production, and Applications, translated by T. O'Conor Sloan, D. Appleton and Co.* 1884.

[7] V. B. Honsinger. Performance of polyphase permanent magnet machines. *IEEE Transactions on Power Apparatus and Systems*, PAS-99(4):1510–1518, 1980.

[8] Yong Juan Cao, Yun Kai Huang, and Long Jin. Research on axial magnetic force and rotor mechanical stress of an air-cored axial-flux permanent magnet machine based on 3d fem. In *Vibration, Structural Engineering and Measurement I*, volume 105 of *Applied Mechanics and Materials*, pages 160–163. Trans Tech Publications Ltd, 1 2012.

[9] T. Chan and L. L. Lai. Permanent-magnet machines for distributed power generation: A review. In *2007 IEEE Power Engineering Society General Meeting*, pages 1–6, 2007.

[10] Cheng-Tsung Liu, T.-S. Chiang, J.F.D. Zamora, and S.-C. Lin. Field-oriented control evaluations of a single-sided permanent magnet axial-flux motor for an electric vehicle. *IEEE Transactions on Magnetics*, 39(5):3280–3282, 2003.

[11] J. Colton, D. Patterson, and J. Hudgins. Design of a low-cost and efficient integrated starter-alternator. In *2008 4th IET Conference on Power Electronics, Machines and Drives*, pages 357–361, 2008.

[12] F. Caricchi, F. Crescimbini, F. Mezzetti, and E. Santini. Multistage axial-flux pm machine for wheel direct drive. *IEEE Transactions on Industry Applications*, 32(4):882–888, 1996.

[13] Solmaz Kahourzade, Amin Mahmoudi, Hew Wooi Ping, and Mohammad Nasir Uddin. A comprehensive review of axial-flux permanent-magnet machines. *Canadian Journal of Electrical and Computer Engineering*, 37(1):19–33, 2014.

[14] Z Wang, Y Enomoto, R Masaki, K Souma, H Itabashi, and S Tanigawa. Development of a high speed motor using amorphous metal cores. In *8th International Conference on Power Electronics - ECCE Asia*, pages 1940–1945, 2011.

[15] Takahiko Miura, Shinji Chino, Masatsugu Takemoto, Satoshi Ogasawara, Akira Chiba, and Nobukazu Hoshi. A ferrite permanent magnet axial gap motor with segmented rotor structure for the next generation hybrid vehicle. In *The XIX International Conference on Electrical Machines - ICEM 2010*, pages 1–6, 2010.

[16] Roberto Di Stefano and Fabrizio Marignetti. An axial flux permanent magnet machine with charged polymer stator core. In *2011 IEEE International Symposium on Industrial Electronics*, pages 520–524, 2011.

[17] Maarten J. Kamper, Rong-Jie Wang, and Francois G. Rossouw. Analysis and performance of axial flux permanent-magnet machine with air-cored nonoverlapping concentrated stator windings. *IEEE Transactions on Industry Applications*, 44(5):1495–1504, 2008.

[18] N.F. Lombard and M.J. Kamper. Analysis and performance of an ironless stator axial flux pm machine. *IEEE Transactions on Energy Conversion*, 14(4):1051–1056, 1999.

[19] Yee-Pien Yang and Fu-Xuan Ding. Driving-scenario oriented design of an axial-flux permanent-magnet synchronous motor for a pedal electric cycle. *IET Electric Power Applications*, 9(6):420–428, 2015.

[20] F. Caricchi, F.G. Capponi, F. Crescimbini, and L. Solero. Experimental study on reducing cogging torque and no-load power loss in axial-flux permanent-magnet machines with slotted winding. *IEEE Transactions on Industry Applications*, 40(4):1066–1075, 2004.

[21] JR Bumby, L Haydock, and NL Brown. Axial flux machines and their use in hybrid electric vehicles. In *uk magnetics society seminar on "motors and actuators for automotive applications*, 2002.

[22] Metin Aydin, Shouming Huang, and Thomas A Lipo. Axial flux permanent magnet disc machines: A review. In *Conf. Record of SPEEDAM*, volume 8, pages 61–71, 2004.

[23] J Colton, D Patterson, and J Hudgins. Design of a low-cost and efficient integrated starter-alternator. In *2008 4th IET Conference on Power Electronics, Machines and Drives*, pages 357–361. IET, 2008.

[24] F. Locment, E. Semail, and F. Piriou. Design and study of a multiphase axial-flux machine. *IEEE Transactions on Magnetics*, 42(4):1427–1430, 2006.

[25] Osman Kalender, Yavuz Ege, and Sedat Nazlibilek. Design and determination of stator geometry for axial flux permanent magnet free rod rotor synchronous motor. *Measurement*, 44(9):1753–1760, 2011.

[26] Surong Huang, Jian Luo, Franco Leonardi, and Thomas A Lipo. A comparison of power density for axial flux machines based on general purpose sizing equations. *IEEE Transactions on energy conversion*, 14(2):185–192, 1999.

[27] E Spooner and BJ Chalmers. 'torus': a slotless, toroidal-stator, permanent-magnet generator. In *IEE Proceedings B (Electric Power Applications)*, volume 139, pages 497–506. IET, 1992.

[28] Jim R Bumby, R Martin, MA Mueller, E Spooner, NL Brown, and BJ Chalmers. Electromagnetic design of axial-flux permanent magnet machines. *IEE Proceedings-Electric Power Applications*, 151(2):151–160, 2004.

[29] Jaouad Azzouzi, Georges Barakat, and Brayima Dakyo. Quasi-3-d analytical modeling of the magnetic field of an axial flux permanent-magnet synchronous machine. *IEEE Transactions on Energy Conversion*, 20(4):746–752, 2005.

[30] Mehmet Gulec and Metin Aydin. Implementation of different 2d finite element modelling approaches in axial flux permanent magnet disc machines. *IET Electric Power Applications*, 12(2):195–202, 2018.

[31] Asko Parviainen, Markku Niemela, and Juha Pyrhonen. Modeling of axial flux permanent-magnet machines. *IEEE Transactions on Industry Applications*, 40(5):1333–1340, 2004.

[32] Huguette Tiegna, Adel Bellara, Yacine Amara, and Georges Barakat. Analytical modeling of the open-circuit magnetic field in axial flux permanent-magnet machines with semi-closed slots. *IEEE Transactions on Magnetics*, 48(3):1212–1226, 2011.

[33] Yu N Zhilichev. Three-dimensional analytic model of permanent magnet axial flux machine. *IEEE transactions on magnetics*, 34(6):3897–3901, 1998.

[34] O De la Barriere, S Hlioui, H Ben Ahmed, M Gabsi, and M LoBue. 3-d formal resolution of maxwell equations for the computation of the no-load flux in an axial flux permanent-magnet synchronous machine. *IEEE transactions on magnetics*, 48(1):128–136, 2011.

[35] Narges Taran, Dan M Ionel, and David G Dorrell. Two-level surrogate-assisted differential evolution multi-objective optimization of electric machines using 3-d fea. *IEEE Transactions on Magnetics*, 54(11):1–5, 2018.

[36] James Clerk Maxwell. *A treatise on electricity and magnetism*, volume 1. Clarendon press, 1873.

[37] Robert Rosen. *Life itself: a comprehensive inquiry into the nature, origin, and fabrication of life*. Columbia University Press, 1991.

[38] Frederico Gadelha Guimaraes. Aprendizagem e busca local em algoritmos meméticos para projetoassistido por computador. 2008.

[39] Michael JD Powell. A new algorithm for unconstrained optimization. In *Nonlinear programming*, pages 31–65. Elsevier, 1970.

[40] Reeves Fletcher and Colin M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[41] Slawomir Koziel and Leifur Leifsson. *Surrogate-based modeling and optimization*. Springer, 2013.

[42] Paolo Di Barba. *Multiobjective shape design in electricity and magnetism*, volume 47. Springer, 2010.

[43] Dipankar Dasgupta and Zbigniew Michalewicz. *Evolutionary algorithms in engineering applications*. Springer Science & Business Media, 2013.

[44] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[45] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[46] F. Mach. Reduction of optimization problem by combination of optimization algorithm and sensitivity analysis. *IEEE Transactions on Magnetics*, 52(3):1–4, 2016.

[47] Ankur Sinha, Dhish Kumar Saxena, Kalyanmoy Deb, and Ashutosh Tiwari. Using objective reduction and interactive procedure to handle many-objective optimization problems. *Applied Soft Computing*, 13(1):415–427, 2013.

[48] Dhish Kumar Saxena, Joao A Duro, Ashutosh Tiwari, Kalyanmoy Deb, and Qingfu Zhang. Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):77–99, 2012.

[49] Dimo Brockhoff and Eckart Zitzler. Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary computation*, 17(2):135–166, 2009.

[50] Marco Amrhein and Philip T Krein. Magnetic equivalent circuit modeling of induction machines design-oriented approach with extension to 3-d. In *2007 IEEE*

International Electric Machines & Drives Conference, volume 2, pages 1557–1563. IEEE, 2007.

[51] Romain-Bernard Mignot, Frédéric Dubas, Christophe Espanet, Cécile Cuchet, and Didier Chamagne. Original design of axial flux pm motor and modeling of the magnetic leakage using a magnetic equivalent circuit. In *2012 IEEE Vehicle Power and Propulsion Conference*, pages 138–141, 2012.

[52] John W Bandler, Natalia Georgieva, Mostafa A Ismail, José E Rayas-Sánchez, and Q-J Zhang. A generalized space-mapping tableau approach to device modeling. *IEEE Transactions on Microwave Theory and Techniques*, 49(1):67–79, 2001.

[53] Mark Dorica and Dennis D Giannacopoulos. Response surface space mapping for electromagnetic optimization. *IEEE Transactions on magnetics*, 42(4):1123–1126, 2006.

[54] S Koziel and S Szczepanski. Accurate modelling of microwave structures using shape-preserving response prediction. *IET microwaves, antennas & propagation*, 5(9):1116–1122, 2011.

[55] Gaetan Kerschen, Jean-claude Golinval, Alexander F Vakakis, and Lawrence A Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics*, 41(1):147–169, 2005.

[56] Sébastien Boyaval, Claude Le Bris, Tony Lelievre, Yvon Maday, Ngoc Cuong Nguyen, and Anthony T Patera. Reduced basis techniques for stochastic problems. *Archives of Computational methods in Engineering*, 17(4):435–454, 2010.

[57] Emiliano Iuliano and Domenico Quagliarella. Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *Computers & Fluids*, 84:327–350, 2013.

[58] George EP Box and Kenneth B Wilson. On the experimental attainment of optimum conditions. In *Breakthroughs in statistics*, pages 270–310. Springer, 1992.

[59] Raymond H Myers, Douglas C Montgomery, and Christine M Anderson-Cook. *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 2016.

[60] Linda Weiser Friedman and Israel Pressman. The metamodel in simulation analysis: Can it be trusted? *Journal of the Operational Research Society*, 39(10):939–948, 1988.

[61] B Yu and K Popplewell. Metamodels in manufacturing: a review. *The International Journal of Production Research*, 32(4):787–796, 1994.

[62] András Sobester, Alexander Forrester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.

[63] Luiz Lebensztajn, Carina Alexandra Rondini Marretto, M Caldora Costa, and J-L Coulomb. Kriging: A useful tool for electromagnetic device optimization. *IEEE Transactions on Magnetics*, 40(2):1196–1199, 2004.

[64] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[65] Marcus HS Mendes, Gustavo L Soares, Jean-Louis Coulomb, and João A Vasconcelos. A surrogate genetic programming based model to facilitate robust multiobjective optimization: A case study in magnetostatics. *IEEE transactions on magnetics*, 49(5):2065–2068, 2013.

[66] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[67] Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: a survey. *IEEE computational intelligence magazine*, 4(2):24–38, 2009.

[68] Manolis Papadrakakis, Nikos D Lagaros, and Yiannis Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer methods in applied mechanics and engineering*, 156(1-4):309–333, 1998.

[69] P Hajela and L Berke. Neurobiological computational models in structural analysis and design. *Computers & Structures*, 41(4):657–667, 1991.

[70] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[71] F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[72] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 479:480, 1969.

[73] Bernard Widrow. *Adaptive" adaline" Neuron Using Chemical" memistors.".* 1960.

[74] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE New York, 1988.

[75] Sebastian Raschka. Single-layer neural networks and gradient descent, Mar 2015. https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.

[76] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[77] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning (adaptive computation and machine learning series)*, pages 195,198. The MIT Press Cambridge, 2016.

[78] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*, 2016.

[79] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[80] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[81] T. Tieleman and G. Hinton. Lecture 6, RMSProp, COURSERA: Neural networks for machine learning, 2012.

[82] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[83] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22, 1987.

[84] Ekaba Bisong. *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019.

[85] Adam Blum. *Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems*. John Wiley & Sons, Inc., USA, 1st edition, 1992.

[86] Michael JA Berry and Gordon S Linoff. *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons, 2004.

[87] Zvi Boger and Hugo Guterman. Knowledge extraction from artificial neural network models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 4, pages 3030–3035. IEEE, 1997.

[88] Douglas G Fox. Judging air quality model performance: a summary of the ams workshop on dispersion model performance, woods hole, mass., 8–11 september 1980. *Bulletin of the American Meteorological Society*, 62(5):599–609, 1981.

[89] Cort J Willmott. On the validation of models. *Physical geography*, 2(2):184–194, 1981.

[90] Cort J Willmott. Some comments on the evaluation of model performance. *Bulletin of the American Meteorological Society*, 63(11):1309–1313, 1982.

[91] Cort J Willmott. On the evaluation of model performance in physical geography. In *Spatial statistics and models*, pages 443–460. Springer, 1984.

[92] Linda Weiser Friedman and Hershey H Friedman. Validating the simulation metamodel: Some practical approaches. *Simulation*, 45(3):144–146, 1985.

[93] Ruichen Jin, Wei Chen, and Timothy W Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and multidisciplinary optimization*, 23(1):1–13, 2001.

[94] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[95] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[96] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[97] Nico JD Nagelkerke et al. A note on a general definition of the coefficient of determination. *biometrika*, 78(3):691–692, 1991.