NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

HENRIQUE DA FONSECA BORGES SOARES E SIMAS
BSc in Electrical and Computer Engineering

A DATA-DRIVEN METHODOLOGY FOR MODELLING LOSSES IN HTS POWER SYSTEMS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
August, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

A DATA-DRIVEN METHODOLOGY FOR MODELLING LOSSES IN HTS POWER SYSTEMS

HENRIQUE DA FONSECA BORGES SOARES E SIMAS

BSc in Electrical and Computer Engineering

Adviser: Roberto André Henrique Oliveira
Researcher, University of Bath

Co-adviser: João Miguel Murta Pina
Assistant Professor, NOVA University Lisbon

Examination Committee

Chair: José Manuel Matos Ribeiro da Fonseca
Full Professor, FCT-NOVA

Rapporteur: Nuno Manuel Ortega Amaro
Assistant Professor, FCT-NOVA

Member: João Miguel Murta Pina
Assistant Professor, FCT-NOVA

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
August, 2022

A Data-Driven Methodology for Modelling Losses in HTS Power Systems

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.5) Lourenço, 2021.

À minha mãe.
À Íris.
À minha família e amigos.

Acknowledgements

Gostaria de agradecer em primeiro lugar o meu orientador, o Doutor Roberto Oliveria, por toda a sua disponibilidade, simpatia, e partilha de conhecimentos. Independentemente do problema ajudou-me, e agradeço-lhe imenso por isso. Ao meu coorientador, Prof. João Murta Pina, agradeço por me abrir as portas para a supercondutividade durante as suas cadeiras, e pela oportunidade de participar neste projeto.

Ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa, pelos ensinamentos transmitidos ao longo do curso e pela formação. Um especial agradecimento à Prof. Isabel Catarino, pelo auxílio no tema da criogenia.

Aos meus colegas de laboratório de supercondutividade pela entreajuda durante todo o tipo de problemas.

Um agradecimento muito especial à minha mãe, por me ter dado a oportunidade de realizar um curso superior e por me ter sempre ajudado ao longo deste percurso, independentemente dos obstáculos.

Aos meus amigos e família, que direta ou indiretamente me ajudaram na realização desta dissertação.

À Íris, por todo o apoio e amor. Sem a tua ajuda tinha sido impossível.

Abstract

When designing a superconducting device one of the main obstacles is the AC losses. These losses created numerous difficulties, particularly in the design of the cryogenic system: the heat created from these losses must be removed in such a way that the cryogenic temperature is not affected, as to not change the materials state from superconductor to normal. Currently, most simulations of AC losses in superconductors are done using numerical methods, such as the finite element method. This type of simulation requires a significant amount of time and computational power.

A data-driven model is proposed in this work to make determining AC losses in a superconducting device easier. A lock-in amplifier method of AC loss measuring is applied to superconducting coils and transformers, as well as a direct V–I method. With these results, an artificial neural network is constructed, trained and optimized in order to accurately predict AC losses in such devices.

This approach is meant to determine AC losses quickly and without the requirement for significant computational power by using only a macro description of a device, such as the number of turns in a coil and the core size of the transformer.

This work was developed in the ambit of the tLOSS project "Transforming Losses Calculation in High Temperature Superconducting Power Systems" (reference PTDC/EEI-EEE/32508/2017_LISBOA-01-0145- FEDER-032508).

Keywords: AC Losses, Superconductor, Cryogenics, Neural Network, Data-Driven Model, Transformer

Resumo

Ao conceber um dispositivo supercondutor, um dos principais obstáculos são as perdas AC. Estas perdas criam numerosas dificuldades, particularmente na conceção do sistema criogénico: o calor devido a estas perdas deve ser removido de forma que as temperaturas criogénicas não sejam afetadas, para não alterar o estado do material de supercondutor para normal. Atualmente, a maioria das simulações de perdas de AC em supercondutores são feitas utilizando métodos numéricos, nomeadamente o método dos elementos finitos. Este tipo de simulação requer uma quantidade significativa de tempo e poder computacional.

Um modelo orientado por dados é proposto neste trabalho para facilitar a determinação de perdas AC num dispositivo supercondutor. Um método de amplificador *lock-in* para medição de perdas AC é aplicado a bobinas supercondutoras e transformadores, bem como um método direto V–I. Com estes resultados, uma rede neural artificial é construída, treinada e otimizada de modo a prever com precisão as perdas AC em tais dispositivos.

Esta abordagem destina-se a determinar perdas AC rapidamente e sem necessidade de poder computacional significativo, utilizando apenas uma descrição macro de um dispositivo, tal como o número de voltas numa bobina e o tamanho do núcleo do transformador.

Este trabalho foi desenvolvido no âmbito do projeto tLOSS "Transformando o Cálculo de Perdas em Sistemas de Potência com Supercondutores de Alta Temperatura" (referência PTDC/EEI-EEE/32508/2017_LISBOA-01-0145- FEDER-032508).

Palavras-chave: Perdas AC, Supercondutor, Criogenia, Rede Neuronal, Modelo Orientado Por Dados, Transformador

Contents

Annexes

List of Figures

List of Tables

Acronyms

1G	First generation superconducting tapes
2G	Second generation superconducting tapes
AC	Alternating Current
ADALINE	ADAptive LInear NEuron
AI	Artificial Intelligence
ANFIS	Adaptive Neuro Fuzzy Interference System
ANN	Artificial Neural Network
BLCO	Superconductor with chemical composition $Ba_{0,15}La_{1,85}CuO_4$
BSCCO	Superconductor with chemical composition $Bi_2Sr_2Ca_nCu_{n+1}O_{6+2n}$ with $n = 0, 1, 2$
CI	Computer Intelligence
DC	Direct Current
DT	Decision Tree Model
emf	Electromotive force
FEM	Finite Element Method
FFNN	Feed Forward Neural Network
GD	Gradient Descent
GL	Generalized Linear Regression Model
HBCCO	Superconductor with chemical composition $HgBa_2Ca_2Cu_3O_{1+x}$
HTS	High-Temperature Superconductor
LIA	Lock-In Amplifier
LM	Levenberg-Marquardt
MgB2	Superconductor with chemical composition MgB_2

ML	Machine Learning
mmf	magnetomotive force
MSE	Mean Squared Error
Ni-W	Alloy composed by a mixture of nickel and tungsten
PLL	Phase-Locked Loop
PSD	Phase-Sensitive Detector
R-value	Regression value
RBFNN	Radial Basis Function Neural Network
RMSE	Root Mean Squared Error
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine Regression Model
TBCCO	Superconductor with chemical composition $TlBa_2Ca_3Cu_4O_{11}$
THD	Total Harmonic Distortion
YBCO	Superconductor with chemical composition $YBa_2Cu_3O_{7-\delta}$ where δ is the oxygen content

Symbols

B	Magnetic flux density	T
B_m	Maximum flux density	T
B_r	Residual flux density	T
$\vec{B}$	Magnetic flux density vector	T
λ_{11}	Leakage self-induction coefficient of the primary	H
λ_{22}	Leakage self-induction coefficient of the secondary	H
ℓ_{11}	Main self-induction coefficient of the primary	H
ℓ_{22}	Main self-induction coefficient of the secondary	H
ϕ_m	Mutual flux or, main flux	Wb
H	Magnetic field intensity	$\mathrm{A\,m^{-1}}$
H_C	Critical magnetic field	$\mathrm{A\,m^{-1}}$
H_{C1}	Inferior critical magnetic field	$\mathrm{A\,m^{-1}}$
H_{C2}	Superior critical magnetic field	$\mathrm{A\,m^{-1}}$
H_{coe}	Coercive force or, coercive magnetic field	$\mathrm{A\,m^{-1}}$
$\vec{H}$	Magnetic field intensity vector	$\mathrm{A\,m^{-1}}$
I_C	Critical current	A
J_C	Critical current density	$\mathrm{A\,m^{-2}}$
L_t	Total self-inductance coefficient	H
μ	Magnetic permeability	$\mathrm{H\,m^{-1}}$
$\vec{M}$	Magnetization vector	$\mathrm{A\,m^{-1}}$
Ψ_ℓ	Coupled leakage magnetic flux	Wb
$\Psi_{1\ell}$	Primary coupled leakage flux	Wb
$\Psi_{2\ell}$	Secondary coupled leakage flux	Wb
Ψ_m	Mutual flux linkage	Wb

Ψ_t	Total flux linkage	Wb
Q_{cl}	Classical Foucault's current loss	$\mathrm{W\,kg^{-1}}$
Q_{exc}	Excess Foucault's current loss	$\mathrm{W\,kg^{-1}}$
Q_h	Hysteresis loss	$\mathrm{W\,kg^{-1}}$
Q_t	Total magnetic loss	$\mathrm{W\,kg^{-1}}$
r_1	Electrical resistance of the primary's coil	Ω
r_2	Electrical resistance of the secondary's coil	Ω
T_C	Critical temperature of a superconductor	K

Chapter 1

Introduction

This chapter sets out the motivation behind this work and its objectives. The original contribution is also laid out, and the dissertation structure is provided in the end.

1.1 Motivation

AC losses have long been one of the biggest challenges in designing a superconducting device. The heat produced by said losses in the superconductor material must be removed in such a fashion that the cryogenic temperatures are not affected, as to reduce coolant evaporation and stop a transition from the superconducting state to normal state. This loss is of great importance when designing the cryogenic component of the device.

Simulations of AC loss in superconducting devices are generally made by using numerical solving methods, such as the Finite Element Method (FEM). When using such methods, both computational power and time are normally sizeable. Due to the crescent usage of superconductors, a faster way of obtaining AC loss is of great interest.

1.2 Objectives

In this work, a data driven methodology will be experimented to predict AC losses in superconducting devices. An artificial neural network will be fed experimental results of AC losses in superconducting coils, both alone and inserted in a ferromagnetic core, as to create a superconducting transformer. Using only a macro description of the components of the device, such as numbers of coils and the core dimensions, it will be attempted to predict AC losses in the superconductor, reducing the need for high computational power and cutting time in obtaining results.

1.3 Original Contribution

This work will contribute with a functional neural network capable of providing AC loss in superconducting coils and transformers with superconducting coils depending on only easily measured parameters, such as core size or number of turns in a coil. This work is carried out in the framework of the tLoss project (reference PTDC/EEI-EEE/32508/2017_LISBOA-01-0145-FEDER-032508), which intends to use different methodologies of calculating the losses of superconducting devices, making them available for the projection or optimization of these devices.

1.4 Structure

This work is divided in four chapters, an introduction, followed by the literature review, the currently developed work and the scheduling, in the following manner:

- **Literature Review**. A review of the concepts of a transformer are given, follow by a presentation of superconductivity in a macro view. Finally, a small chapter regarding cryogenics is followed by data modelling and neural networks.
- **Experimental Apparatus and Results**. The assembly of the experiment is presented, as well as the measuring methods used are firstly shown in this chapter. In the end, the experimental results are presented.
- **ANN Modelling**. The training and results for both the global network as well as for a series network are presented in this chapter.
- **Conclusions and Future Work**. A conclusion of the work is done, followed by possible future work.

Chapter 2

Literature Review

This chapter will introduce and discuss various concepts and issues that are necessary for a better comprehension of the work offered in this dissertation.

Following a brief discussion of the transformer, an explanation and description of superconductivity and cryogenics are offered. Finally, data-driven models, particularly artificial neural networks, are introduced.

2.1 Transformer

A transformer is a fundamental component of any power system, enabling the transformation of electric tension. It's main components are a core, normally ferromagnetic with high magnetic permeability, and two or more coils around it.

In the case of a single-phase transformer only two coils are needed. The primary, which is connected to a source, will produce an alternating flux. A portion of this flux will link with the secondary, inducing a voltage that depends on the relationship between the number of turns of both coils.

The working principle of a transformer is Faraday's Law, as in the mutual induction between two magnetic circuits magnetically connected thru a simple mutual flux, ϕ_m, whose lines mainly run through the core. The coils of both the primary and the secondary are made of an electrical resistance r, a mutual self-induction coefficient ℓ and a leakage self-induction coefficient λ. The total coupled flux of each coil Ψ_t is composed by two parts (Equation 2.1): the mutual flux linkage Ψ_m, coupled to both coils, and the coupled leakage flux Ψ_ℓ, coupled to each of the coils and with field lines mainly thru the air.

$$\Psi_t = \Psi_m + \Psi_\ell \tag{2.1}$$

Consequently, the total self-inductance coefficient L_t is

$$L_t = \frac{\Psi_t}{i} = \frac{\Psi_m + \Psi_\ell}{i} = \frac{N\phi_m}{i} + \frac{\Psi_\ell}{i} = \ell + \lambda \tag{2.2}$$

with the self-induction coefficient given by

$$\ell = \frac{N^2}{\mathcal{R}} \tag{2.3}$$

where $\mathcal{R}$ is the magnetic reluctance. A schematic for a single-phase transformer is presented in Figure 2.1.

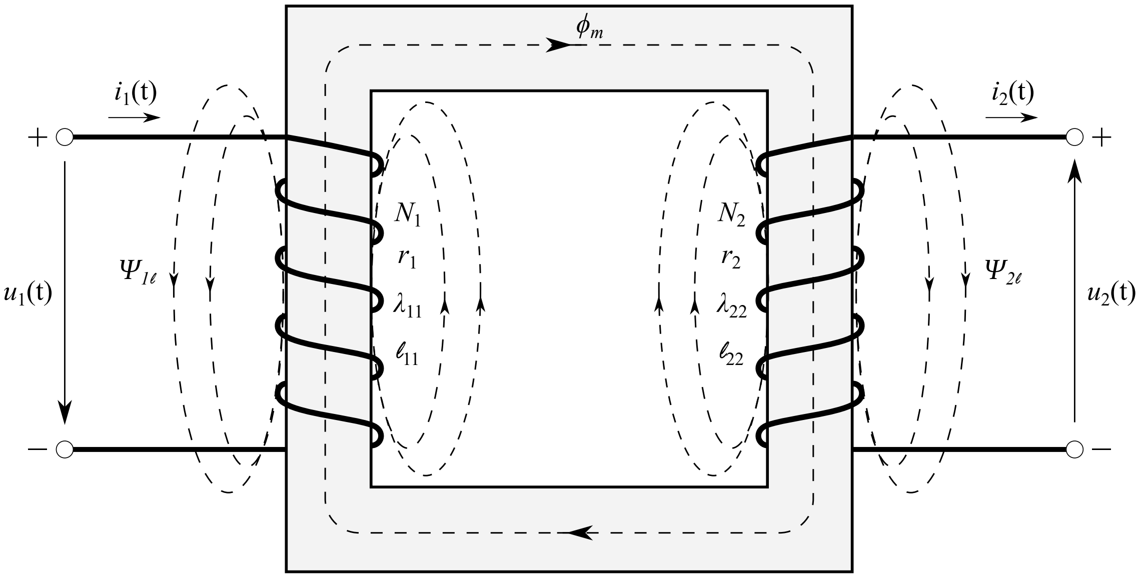


Figure 2.1: Schematic of a single-phase transformer where r_1 and r_2 are the electrical resistances of the primary and secondary, λ_{11} and λ_{22} the leakage self-inductance coefficients, ℓ_{11} and ℓ_{22} the main self-induction coefficients and $\Psi_{1\ell}$ and $\Psi_{2\ell}$ the leakage fluxes.

2.1.1 Hysteresis Loss

The relation between H and B, in most materials and within certain bounds, is given by

$$B = \mu H \tag{2.4}$$

where μ is the magnetic permeability of the material in $\mathrm{H\,m^{-2}}$. This relationship is however limited by the magnetic saturation in the ferromagnetic material, i.e. by the saturation of B, and also by the increase or decrease of H.

Consider a demagnetized magnetic material, in other words, with $H = 0$ and $B = 0$, such as in (1) of Figure 2.2. When first applying a magnetic field the relationship between H and B will be given by an ascending curve, saturating at B_m, called first magnetization curve. When reaching (2) the domains' magnetic moments align with the applied field and the material is said to be saturated, having its permeability close to the value of free space and losing its magnetic properties.

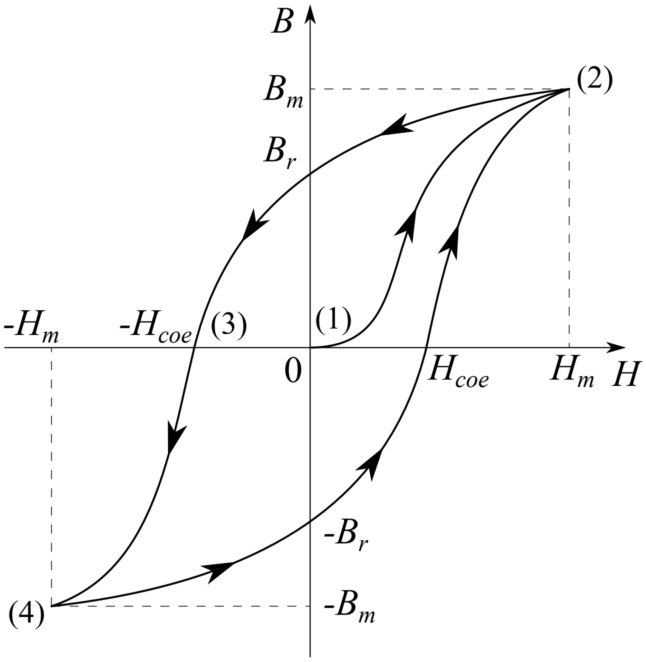


Figure 2.2: Representation of an hysteresis loop

By reducing the external field the curve will not decrease with the same slope as it increased, as shown from (2) to (3). While reducing the external field to zero will relax the magnetic dipoles towards their initial orientation, they will retain a net magnetization component along the applied field direction (Guru & Hiziroglu, 2001). This effect is called magnetic hysteresis, and it's the reason for the nonlinear and multivalued nature of the relationship between H and B. When $H = 0$ the magnetic flux density is not null: this value is called residual flux density B_r and the lower it is the lower the hysteresis losses will be. To bring B to zero, H is reduced to H_{coe}, known as coercive force. Increasing any more H in the reverse direction causes magnetization with reversed polarity, shown along the section from (3) to (4), reaching saturation once again. Reducing the applied field to zero and then applying the original direction, from (4) back to (2), a cycle appears and is referred to as an hysteresis loop.

For each hysteresis cycle a power loss exists, referred to as hysteresis loss and is the first of two loss mechanisms associated with magnetic loss. This loss stems from a molecular friction when the magnetic domains reverse their directions, due to the applied magnetomotive force (mmf) (Guru & Hiziroglu, 2001). Hysteresis losses are proportional to the area of the hysteresis loop in a ferromagnetic material. Equation 2.5 shows the hysteresis losses Q_h for a certain volume Vol and for the frequency of the hysteresis loops.

$$Q_h = Vol \cdot \oint H dB \cdot f \tag{2.5}$$

An empirical formula was proposed by Charles Steinmetz in (Steinmetz, 1892) that allowed to obtain hysteresis losses based on the maximum flux density B_m, given by

$$Q_h = k_h f B_m^{\alpha} \tag{2.6}$$

where k_h is the hysteresis loss constant, dependent on the material, and α is the Steinmetz exponent.

2.1.2 Classical Foucault's Current Loss

The second magnetic loss mechanism is due to Foucault's currents. From the time-varying magnetic field applied to a coil an induced electromotive force (emf) results, complying with Faraday's law. If the coil is wrapped around a magnetic material, as is the case with the transformer, the induced emf will generate an electrical current inside that material, such that the magnetic flux producing it will be enclosed. Commonly referred to as eddy currents, they circulate in the core of a transformer, forming a swirling pattern and opposing changes in flux density in the material.

Due to this currents, energy is converted into heat, and by summing the power loss in each loop within the magnetic material, one can obtain the total power loss causes by Foucault's currents (Guru & Hiziroglu, 2001).

To reduce its effects, the magnetic material is built with multiple sheets insulated from each other and aligned with the direction of the field lines. The insulation of each lamination interrupts the current path, and smaller losses appear with thinner sheets and with increased resistivity of the ferromagnetic material (Pronto, 2010).

For a sinusoidal flux density, the average classical Foucault's current loss is

$$Q_{cl} = k_f f^2 \delta^2 B_m^2 Vol \tag{2.7}$$

where Q_{cl} if given in $\mathrm{W\,kg^{-1}}$, k_f is a constant dependent on the material's conductivity, δ is the lamination's thickness in m, B_m is the maximum flux density in T and Vol is the volume of the magnetic material in m^3. From this equation it is also possible to see the relationship with frequency, increasing the losses derived from Foucault's currents with the increase of operating frequencies.

2.1.3 Excess Foucault's Current Loss

The sum of the hysteresis losses and classical Foucault's losses doesn't match the real loss value measured. Another phenomena must be causing it, and is referred to as excess Foucault's current loss Q_{exc}, with its value being

$$Q_{exc} = Q_t - Q_h - Q_{cl} \tag{2.8}$$

where Q_t are the total magnetic losses. When applying an external magnetic field to a ferromagnetic material, small areas referred to as magnetic domains, reorient themselves to the same direction until magnetization is achieved. The transition between two domains unaligned is gradual, taking place along a border referred to as wall of the domain. Foucault's excess losses are associated with the large scale movement of this walls and its respective rotation (Pronto, 2010).

2.2 Superconductivity

In the year of 1908, physicist Heike Kamerlingh Onnes was able to liquefy helium, opening the door for research in the field of low temperature physics. With continuous experiments, Onnes and his team produced the plot shown in Figure 2.3, demonstrating the abrupt change in the resistance of mercury at 4.20K (Rogalla & Kes, 2012). With this experiment, the state of superconductivity was discovered.

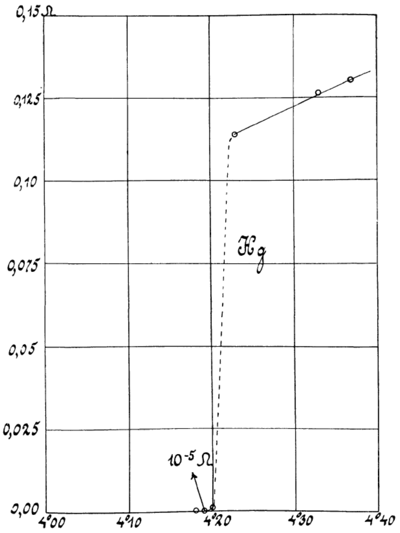


Figure 2.3: Historic plot of resistance (Ω) versus temperature (K) from Onnes' 1911 experiment (Onnes, Gavroglu, & Goudaroulis, 1991)

Throughout the years, many pieces of research were dedicated to superconductors, intending to understand its unusual behaviours, such as Walter Meissner and Robert Ochsenfeld, with their discovery of perfect diamagnetism in superconductor materials in 1933; the brothers Fritz and Heinz London, with their macroscopic description of the superconducting state and the London equations (London & London, 1935); John Bardeen, Leon Cooper and John Schrieffer with their microscopic theory on superconductivity and Cooper pairs; and Georg Bednorz and Alexander Müller with the discovery of High-Temperature Superconductor (HTS) (Bednorz & Müller, 1986). This last discovery will be of great interest in this report and will be further elaborated in this section.

Three critical values can be defined for the superconducting state[1]:

[1] In reality, pressure can also influence superconductivity (Pina, 2010), but it's out of the scope of this report.

- **Critical temperature**. Under a certain critical temperature T_C, superconductors show virtually zero resistance (Onnes, Gavroglu, & Goudaroulis, 1991).
- **Critical magnetic field**. Applying a magnetic field superior to the critical magnetic field H_C will destroy the superconductivity (Tuyn & Onnes, 1926).
- **Critical current density**. Only under a critical current density J_C can the superconducting state exist. This value depends on the temperature and the magnetic field (Silsbee, 1916).

Those three values are not independent from each other, forming a well defined boundary where superconductivity exists, normally represented in a $T-J-H$ diagram, as shown in Figure 2.4. A material is superconductor when inside those boundaries.

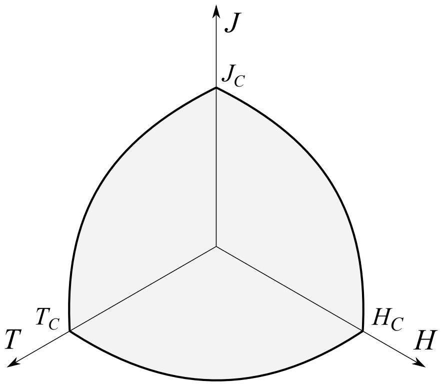


Figure 2.4: Typical $T-J-H$ diagram of a superconductor

2.2.1 Perfect Conductivity

When a material is in a superconductivity state, complying with the critical limits, and in a DC regime, the resistivity will be null and, consequently, the losses will be nonexistent. In the AC regime, such phenomena will not happen. The reasons for dissipation with AC currents will be exposed in Section 2.2.3.

2.2.2 Meissner Effect

The expulsion of all magnetic field by a superconductor is known as the Meissner effect, or perfect diamagnetism. This effect is based on the expulsion of the magnetic flux lines from inside the superconductor when $T < T_C$. With an applied field smaller than H_C, the magnetic induction inside the superconductor is null (Meissner & Ochsenfeld, 1933). Inside a superconductor the flux density is given by

$$\vec{B} = \mu_0(\vec{H} + \vec{M}) \tag{2.9}$$

where $\vec{M}$ is the materials magnetization vector. With $\vec{B} = 0$ one gets $\vec{M} = -\vec{H}$. From this observation, the Meissner effect results in a magnetization with equal modulus and opposite direction to the applied field, or, in other words, to a perfect diamagnetism.

2.2.3 Types of Superconductors

Superconductors are classified into two types based on their behavior: Type I and Type II. The magnetic response of a Type I material is presented in Figure 2.5(a), where one can see the compliance with the Meissner effect and Equation 2.9. This type of superconductor has little practical use in power applications due to the low value of critical flux density, typically under 0.1T.

A Type II superconductor has two critical magnetic fields, H_{C1}, and a much higher H_{C2}. Up until H_{C1} perfect diamagnetism is present, but between H_{C1} and H_{C2} it is said to be in a mixed state, where the material is still a diamagnet but not perfect with some magnetic flux penetrating. A general magnetic response for Type II superconductor is presented in Figure 2.5(b).

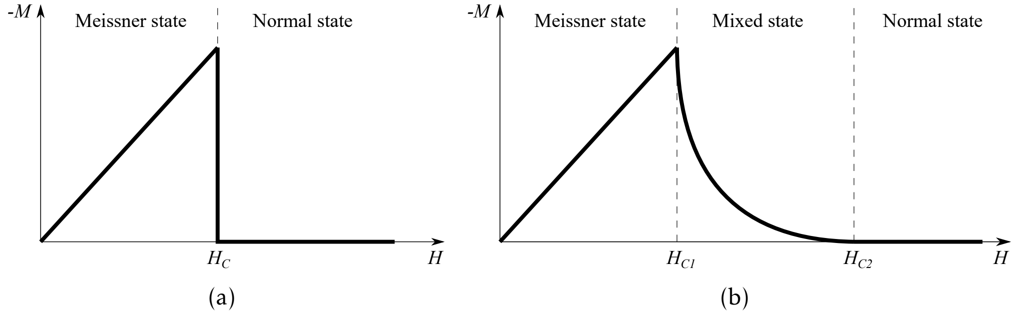


Figure 2.5: Magnetization curve in function of the applied field for a (a) Type I superconductor, and (b) Type II superconductor

In the mixed state flux penetrates the material trough fluxons, which are threaded trough vortices. With an applied current, a resultant emf exists and is perpendicular to the vortices, moving them around the material and causing dissipation (Melhem, 2012).

2.2.4 High-Temperature Superconductors

High-temperature superconductors (HTS), have a much higher critical temperature than other materials. In this report, a critical temperature over 77K, the boiling point of nitrogen, will be the cutoff for a material to be considered an HTS. This decision is based on the low cost of cooling a superconductor with the use of liquid nitrogen. Some HTS and its corresponding critical temperature are presented in Table 2.1, as well as two non-HTS, namelly BLCO and MgB2.

Material	Acronym	T_C [K]
$Ba_xLa_{5-x}Cu_5O_{5(3-y)}$	BLCO	30
MgB_2	MgB2	40
$Y_{1,2}Ba_{0,8}CuO_{4-\delta}$	YBCO	93
$BiSrCaCu_2O_x$	BSCCO	105
$TlBa_2Ca_3Cu_4O_{11}$	TBCCO	128
$HgBa_2Ca_2Cu_3O_{1+x}$	HBCCO	133

Table 2.1: Critical temperatures of superconducting materials (adapted from (Pina, 2010) and (Buzea & Yamashita, 2001))

The most common commercially available compounds are YBCO with chemical formula $YBa_2Cu_3O_{7-\delta}$ where δ is the oxygen content, defining the superconducting properties of the material, and BSCCO with chemical formula $Bi_2Sr_2Ca_nCu_{n+1}O_{6+2n}$ where $n = 0,1,2$. Both of this materials are available in bulk, cable or tape forms. Only the tape form will be discussed, since other forms are out of the scope of this work.

2.2.5 Superconducting Tapes

Superconducting tapes are divided in first and second generation tapes, 1G and 2G, respectively.

First generation tapes use BSCCO, with critical temperatures of 90K or 110K, depending on the chemical constitution of the BSCCO. The BSCCO is encompassed in silver, to assure flexibility and mechanical robustness. Even with the silver, the tape has a minimum bending radius before the superconducting properties are lost. This type of tapes are sensitive to applied magnetic fields, mainly perpendicular ones (Pina, 2010).

In a 2G tape, a small layer of YBCO is laid on a flexible metal substrate, such as Ni-W, as shown in Figure 2.6. With this technology its possible to operate in higher temperatures and fields, and with higher critical current density, in relation to 1G tapes (Pina, 2010). Besides that, 2G tapes present better mechanical properties and material homogeneity. Despite the higher price of 2G tapes at this time, the advantages outweigh the costs.

2.2.6 Superconductor Modelling

Even though there is not a consensus relatively to the model that should be used to describe a Type II superconductor, some theories, with the right assumptions, can characterize a material's behaviour to an acceptable degree.

2.2.6.1 Bean's Critical State Model

One of the most widely accepted models is Bean's Critical State Model, introduced in 1962 (Bean, 1962) by Charles P. Bean, and updated in 1964 (Bean, 1964). A critical state

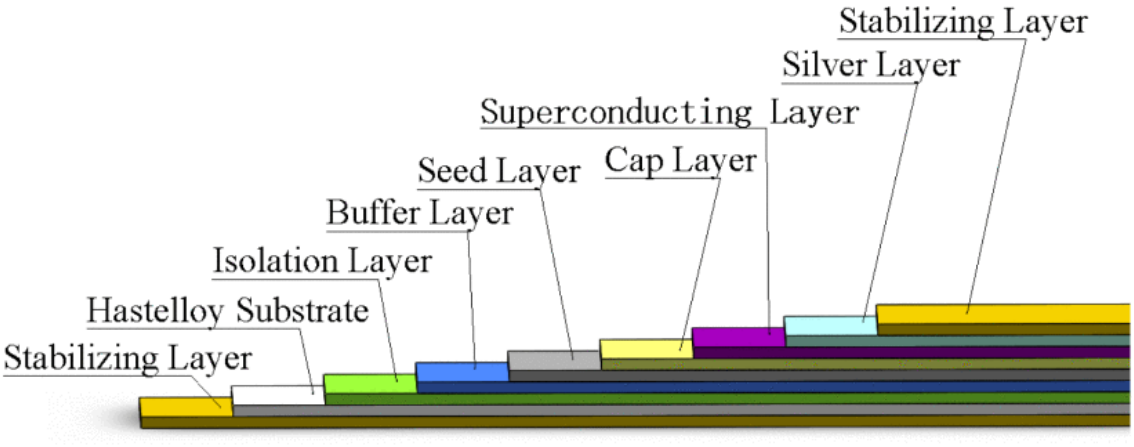


Figure 2.6: Structure of a 2G superconducting tape (Xu, Dai, Ma, & Wang, 2021)

model makes assumptions based on Ampère's Law shown in Equation 2.10, where **B** is the magnetic field and **J** the current density (Poole, 1999).

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} \tag{2.10}$$

Bean's Critical State Model makes the assumptions that $J = \text{const} = J_C$, and that the current density can only take the values of $\pm J_C$ or 0. It's proposal is the penetration of the magnetic field in a linear fashion. Even though it does not correspond to reality, this simplification yields valid approximations for certain conditions, such as in AC loss calculation in a tape (Oomen, 2000).

2.2.6.2 *E–J* Power Law

In a superconductor, the assumption that the current density will always be its critical value or 0 is not true: an abrupt transition between the superconducting state and the normal state does not exist. The E–J Power Law takes in to account this reality, and is translated as

$$\rho(J) = \frac{E_C}{J_C} \left| \frac{\mathbf{J}}{J_C} \right|^{n-1} \tag{2.11}$$

where ρ is a non-linear resistance, and the critical current density J_C is the current that produces a predefined value of the electrical field E_C (Brambilla, Grilli, & Martini, 2006; Seidel, 2015). The value of E_C is usually $1\,\mu\text{V}\,\text{cm}^{-1}$ (Oomen, 2000). The n parameter is used to define the curves shape where the higher the value of n, the more pronounced the transition to the normal state is. Two specific cases are of note: $n = 1$ is Ohm's Law and $n = \infty$ is Bean's Model. For HTS a value of n between 5 and 20 is the norm (Pina, 2010).

2.2.7 AC Losses in Superconducting Materials

In a superconducting material with an AC current applied, losses will appear. This is in contrast to the case where DC current is passing trough the superconductor, where virtually zero losses exist. Two types of losses exist in a Type II HTS:

- **Magnetization loss** Q_{mag}, due to variations of the magnetic field. The magnetic field variation inside the material induces an electrical field according to Faraday's law $\nabla \times \mathbf{E} = -d\mathbf{B}/dt$ (Oomen, 2000). They are composed by hysteresis loss in both the superconductor and the ferromagnetic substrate, and Foucault's currents.

- **Transport current loss** Q_{trs}, composed of self-field loss, flux-flow loss and resistive loss (Martínez, 2010; Oomen, 2000).

The total loss is therefore given by $Q_{total} = Q_{mag} + Q_{trs}$.

2.2.7.1 Superconducting Hysteresis Loss

Applying a variable external field to a superconductor leads to an hysteresis loop as shown in Figure 2.7.

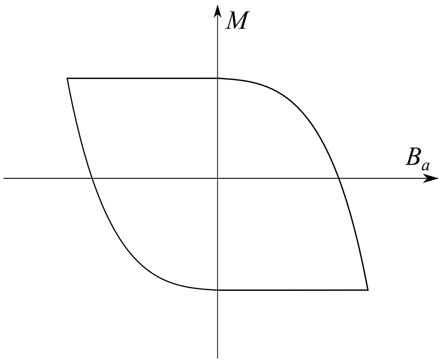


Figure 2.7: Representation of a hysteresis loop of a superconductor

For each cycle, the material will cover the loop and return to its initial state, thus, the loss of energy of this process is given by

$$Q_H = \oint M dB \tag{2.12}$$

and is referred to as superconducting hysteresis loss.

2.2.7.2 Ferromagnetic Loss in Superconducting Tapes

As referred in Section 2.2.5, the substrate of a 2G tape is typically made of a ferromagnetic material, such as Alloy composed by a mixture of nickel and tungsten (Ni-W). The losses present in this materials are generally the same as in any ferromagnetic materials, namely hysteresis loss (see Section 2.1.1) and Foucault's current loss (see Sections 2.1.2 and 2.1.3).

Using a Ni-W alloy for the substrate in a YBCO tape, it's shown in (Ijaduola, Thompson, Goyal, Thieme, & Marken, 2004) that the dissipated power is larger with larger operating frequencies, and with cut, deformed or damaged materials. It's also described that, for a realistic level of operating currents, ferromagnetic losses are relatively smaller when compared with the loss in the superconductor itself.

2.2.7.3 Self-field Loss

Associated with an AC current flowing in the superconductor is a magnetic field, called self-field, which produces superconducting hysteresis losses (Martínez, 2010). Due to this field, even with zero external field, the critical current density is limited.

2.2.7.4 Flux-flow Loss

As talked about in Section 2.2.3, vortices form in the mixed state. Those vortices are anchored by defects, such as impurities, in pinning centres and are distributed in an Abrikosov's network. When a current passes these flux, a Lorentz force is produced in each vortex, and, with a sufficiently high current, the vortices are de-anchored from their positions and move through the superconductor (Sjöström, 2001). The loss associated with this effect is called flux-flow loss.

2.2.7.5 Resistive Loss

When the current exceeds I_C of the superconductor, a new path is taken by the excess current via the conducting layers (Martínez, 2010), resulting in a resistive loss.

2.2.8 AC Loss Measuring Methods

AC losses can be measured based on two main categories: calorimetric or electromagnetic. The calorimetric category has a main method, the boil-off method, while the electromagnetic category has two, the lock-in amplifier method and the pick-up coil method. All three are briefly presented below.

2.2.8.1 Boil-Off Method

The boil-off method is based on measuring the amount of cryogenic liquid evaporating owing to heat created by the superconductor (J.-H. Kim, Kim, Iyyani, Kvitkovic, & Pamidi, 2011; Okamoto, Sumiyoshi, Miyoshi, & Suzuki, 2006). Since there is a need to measure the emitted gas flow, the cryostat must be a closed one. This method does not have a high sensibility, but performs well for large superconducting systems, since the amount of heat generated by the AC losses is high.

2.2.8.2 Pick-Up Coil Method

The first method of the electromagnetic category, the pick-up coil method uses the variation in magnetic field to obtain the AC loss. By placing coils that are sensible to magnetic field variations in certain locations (which depend based on the device or application), the variation can be measured, and thus the AC losses can be calculated (Kajikawa, Iwakuma, Funaki, Wada, & Takenaka, 1999; Yang, Martinez, & Norris, 2004).

2.2.8.3 Lock-In Amplifier Method

The most common AC loss measuring method is LIA method. Using a LIA it is possible to obtain both the resistive and inductive components of the loss, where the inductive part corresponds to the self-inductance. A more in-depth description of this method is given in Section 3.3.2.

2.2.9 Example of AC Loss Measurement Experiment

In (Friedman, Wolfus, Kopansky, & Yeshurun, 2012), a YBCO pancake coil with the parameters shown in Table 2.2 was tested, using both calorimetric and electromagnetic methods. The schematic of the device used to measure the I-V curves of a single pancake coil is shown in Figure 2.8.

Parameter	Value	Unit
Inner diameter	10	mm
Outer diameter	11	mm
Number of turns	20	–
Length of wire	6	m
Inductance	76 - 81	μH
Critical current @ 77K	70	A

Table 2.2: Parameters of the YBCO pancake coil used in (Friedman, Wolfus, Kopansky, & Yeshurun, 2012)

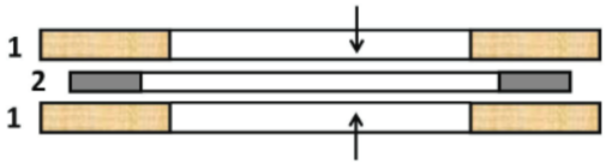
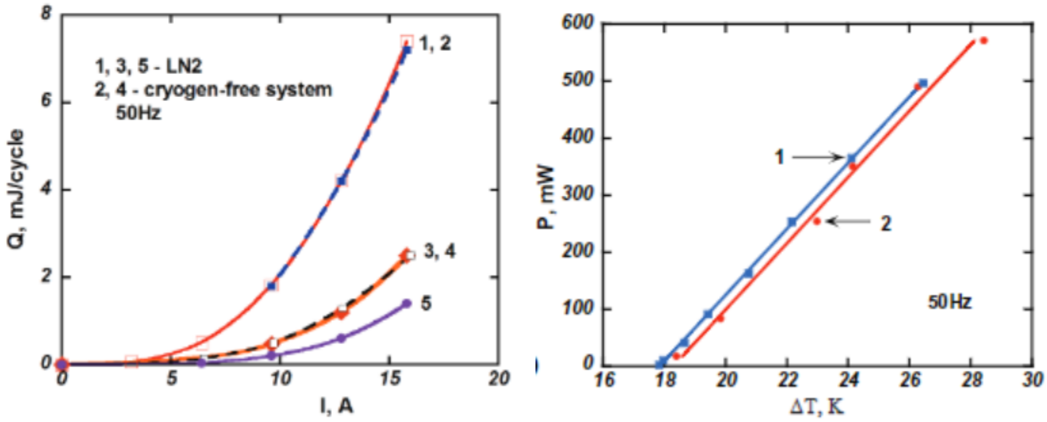


Figure 2.8: I-V measuring setup used in (Friedman, Wolfus, Kopansky, & Yeshurun, 2012) for single pancake coils in external magnetic field, where 1 are copper coils and 2 is a HTS pancake coil

Using a power analyzer, the AC losses were measured in the coils using an electromagnetic method. To confirm the values, the authors also used a calorimetric method simultaneously, and the results are presented in Figure 2.9.



(a) (b)

Figure 2.9: (a) AC losses measured by electric method in liquid nitrogen and in the cryogen-free system: 1, 2 - double pancake; 3,4 - single pancake with adjacent pancake; 5 - single pancake. (b) Results of AC losses measurements by: 1 - electric method; 2 - calorimetric method.

The expected function form in a linear scale can be seen in Figure 2.9(a). In most cases, a logarithmic scale is employed to illustrate these data, and a logarithm function-like form is expected. Also of note is the units used to represent the AC loss, J/cycle.

2.3 Cryogenics

Cryogenics is the application and study of materials at low temperatures, normally under 120K (Melhem, 2012). For HTS applications liquid nitrogen is typically used, so a temperature under 77K, the boiling point of nitrogen, is of most interest.

2.3.1 Fluid Cooling

Fluid cooling uses a coolant, normally either helium or nitrogen, to cool the superconducting device. Three broad categories of fluid cooling methods can be described (Seidel, 2015):

- **Direct cooling**. The coolant comes into direct touch with the superconductor, and it fills or flows through channels built within the system. Convection is the most common mode of heat transfer. It has two different modes: *bath cooling* and *internal cooling*.

- **Fluid-mediated cooling**. A fluid loop transports heat from a cold source to a thermal anchor that transmits heat cools the superconducting system. Conduction and convection in the fluid loop are the primary heat mechanisms.

- **Conduction cooling**. Thermal linkages connect the cold supply to the system, and heat is eliminated through pure conduction.

In this work, only direct cooling will be considered. It is also possible to differentiate between coolant conditions. There are three different conditions which can be applied to both methods of direct cooling:

- **Saturated liquid**. The fluid is at equilibrium conditions of pressure and temperature, with evaporation and/or boiling being a part of the heat transfer.
- **Single-phase subcooled liquid**. The pressure is lower than the critical pressure, and the temperature is lower than the value on the saturation line. With increased temperature evaporation or boiling may take place.
- **Single-phase supercritical fluid**. The pressure is higher than the critical pressure, with the fluid remaining in a single-phase, and no evaporation or boiling can happen.

2.3.2 Bath Cooling

Bath cooling is a method of direct cooling in which the superconducting device is immersed in, or permeated by, a pool of fluid (Seidel, 2015). The heat generated by the device or entering the cryostat is collected in the bath. The bath may be a saturated bath, where there is a free surface, liquid and vapour phases are in equilibrium, and the heat is removed by evaporation at the free surface; or a subcooled bath, where the liquid is not close to saturation and fills the cooling vessel, requiring a heat exchanger to provide the cold source.

For small systems, this form of cooling is very easy and relatively inexpensive, but it has the disadvantages of a fixed temperature, the limited size of the superconducting device due to cryostat size, and a large amount of coolant required.

2.3.3 Internal Cooling

When using internal cooling, the flow of the coolant is within the superconducting device and can be driven by either natural convection (buoyancy) or by forced convection (pumps or compressors) (Seidel, 2015). The main heat transport mechanism is convection, caused by the flow itself. The flow can be either proximal, where the coolant is in direct contact with the superconductor, or indirect, cooling the superconductor by conduction.

Internally cooled devices are capable of large predictable heat transfers due to the ability to control mass flow and have a smaller coolant quantity needed compared to bath cooling. The downsides are the need of leak tight circuits for the flow, and the need to sustain the mass flow, which has consequences on the thermodynamic efficiency of the cryogenic system.

2.3.4 Cryocoolers

For cases where obtaining the coolant for fluid cooling is difficult, a cryocooler can be used. The main advantage is the consumption of electricity instead of, for example, liquid nitrogen. On the other hand, a typical cryocooler has vibrations, electromagnetic interferences and temperature variations (for regenerative cryocoolers) (Seidel, 2015).

Two main types of classical cryocoolers are used at this time: the higher frequency Stirling, and the lower frequency Gifford-McMahon. Both the design and the common principle of these cryocoolers are shown in Figure 2.10.

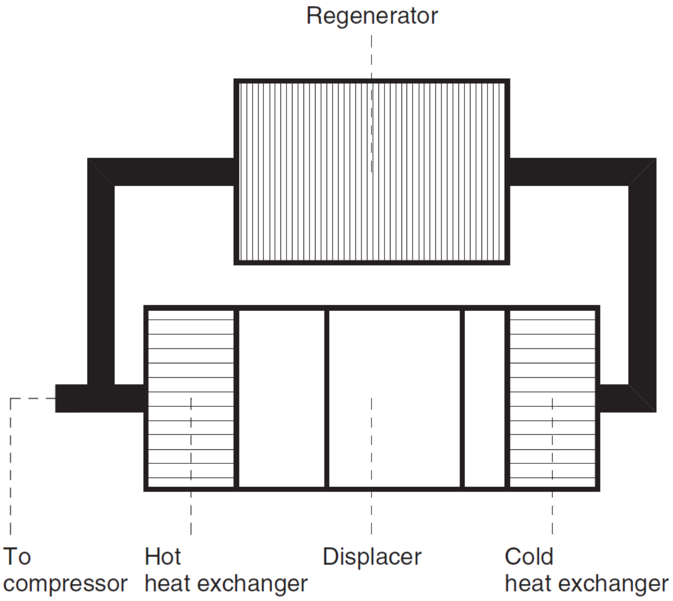


Figure 2.10: Design principle of the Stirling and Giffor-McMahon cryocoolers (from (Seidel, 2015))

A Stirling cryocooler consists in a pressure wave generator and a cold head. If they are connected directly, it's referred to as an integral cryocooler, else it's referred to as a split cryocooler. Normally, a Stirling cryocooler operates in frequencies between 30 and 100Hz, but in most cases operates in an equal frequency to the electric line, such as 50 or 60 Hz.

On the other hand, Gifford-McMahon cryocoolers are always split cryocoolers, consisting of a compressor package and a cold head package. They operate at much lower frequencies, in the range of 1 to 5 Hz.

Other types of cryocoolers exist, such as pulse tube cryocoolers and mixture Joule-Thomson cryocoolers.

2.4 Data Driven Models and Machine Learning

A data-driven model is based on the analysis of data about a specific system with the goal of discovering correlations between its input and output without understanding of the system's actual behavior. Machine learning is one of the most popular data-driven model.

Machine Learning (ML), in its most basic form, is a technique to get a model from data, where the model is the final product (P. Kim, 2017). By training with the data, the ML process learns until a solution is reached.

ML have been successfully applied to areas where employing mathematical models would prove to be extremely difficult or even impossible, such as the identification of images or speech recognition.

2.4.1 Artificial Neural Networks

One of the most widely implemented algorithms in ML is the Artificial Neural Network (ANN). Intended to copy a real brains architecture, its fundamental component is the neuron, analogous to its biological counterpart, which interacts with other neurons to form layers and solve the problem at hand. Each connection will have a corresponding weight that will be optimized until the best solution is reached.

2.4.2 Advantages of Artificial Neural Networks

Currently, FEM is the most widely used form of simulation of superconducting equipment. When using this method, one can expect a long computational time, increasing with the size of the simulation desired (Brambilla, Grilli, Martini, & Sirois, 2008). On the contrary, by using an ML approach, one can describe the system in a macro way and predict results with an arbitrarily precision in a much quicker manner.

Some of the most important properties and capabilities of an ANN are (Haykin, 2009):

- **Non linearity**. Each neuron can be linear or, most importantly, non linear, enabling the network to solve extremely complex problems.
- **Adaptability**. If the environment of a neural network is changed it can be rapidly retrained to adapt to the new circumstances. In the case of a dynamic environment, it is possible to change the weights in real time.
- **Fault tolerance**. In a case which a neuron or a connection is damaged, the neural network suffers little degradation. For a large degradation effect to be felt on the response, the network must be broadly affected.
- **Input-Output Mapping**. Applying supervised learning (see Section 2.4.6) a neural network is capable of obtaining a general function for a problem and predict outputs for previously unknown inputs.

Currently, ANN are currently used in a multitude of fields, from electronics to speech recognition[2].

[2] For a broad list of applications see pages 1-5 to 1-7 of (Hagan, Demuth, Beale, & Jesús, 2014)

2.4.3 Neuron

The neuron is the basis of a neural network. A simple single input neuron is shown in Figure 2.11. The input x multiplies by the weight scalar w, forming the weight function wx. The value of wx is then summed[3] with a bias b, originating a net input $v = wx + b$. Finally, the net input goes into an activation function φ, producing the output y (Haykin, 1999). This output is therefore calculated as

$$y = \varphi(v) = \varphi(wx + b) \tag{2.13}$$

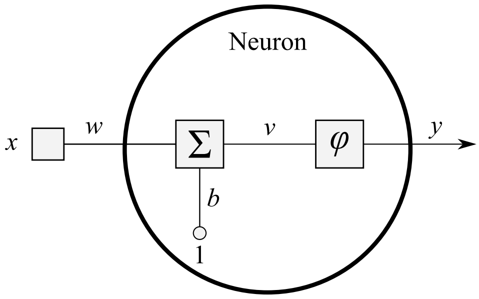


Figure 2.11: Single input neuron

The case where a neuron only has one input is not the norm: in a neural network, a neuron will most likely have multiple inputs, such as is shown in Figure 2.12(a). Each input $x_1, x_2, ..., x_R$ of the input matrix $\mathbf{x}$ has a corresponding weight $w_{11}, w_{12}, ..., w_{1R}$, stored in the weight matrix $\mathbf{W}$. The output is rewritten as

$$y = \varphi(v) = \varphi(\mathbf{W}\mathbf{x} + b) \tag{2.14}$$

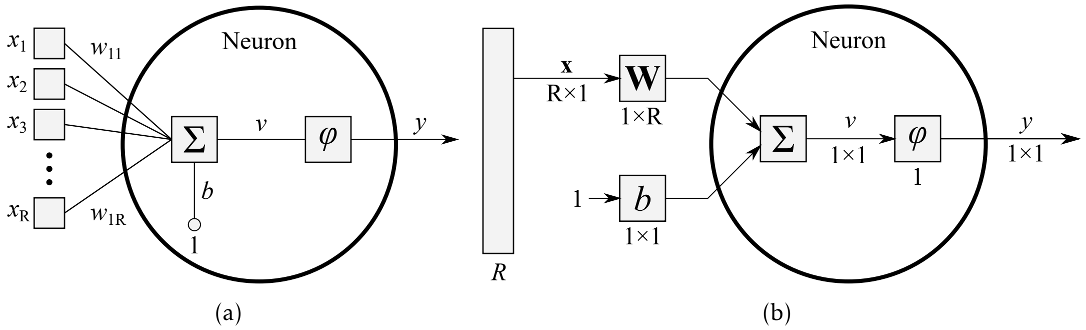


Figure 2.12: Multiple input neuron in (a) extensive notation (b) abbreviated notation

When using many neurons, the current notation can be problematic. For this reason an abbreviated notation is used. In Figure 2.12(b) a neuron with R inputs is presented, where the dimensions of $\mathbf{x}$ are displayed as $R \times 1$ below the symbol. The $\mathbf{W}$ matrix, on the other hand, has $1 \times R$ as its dimension, which is also shown below the variable box.

[3] Normally a sum is used, but a multiplication can also be applied

In Figure 2.12(b) one can also see the activation function φ represented. The most used functions are the linear function and the log-sigmoid function, both of which are presented in Figure 2.13.

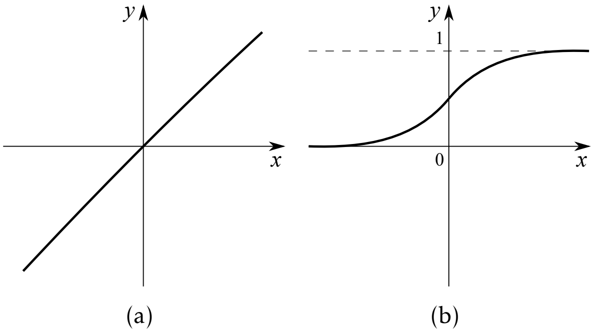


Figure 2.13: Representation of a (a) linear function (b) log-sigmoid function

Neurons with linear activation function have its output equal to the net input, as shown in Equation 2.15, and are often used in the output layer (Aggarwal, 2018). On the other hand, neurons with log-sigmoid activation functions (Equation 2.16) are used in hidden layers for back-propagation (see Section 2.4.8) due to their differentiability. This function flattens the outputs in the range of 1 to 0.

$$y = v \tag{2.15}$$

$$y = \frac{1}{1 + e^{-v}} \tag{2.16}$$

2.4.4 Layers

By placing several neurons working in parallel, a layer is obtained. For R inputs and S neurons, the weight matrix will now have an $S \times R$ size. Each layer has a bias vector **b**, containing all of the bias values for each of the S neurons, and will output an output vector **y**, also with size S (Aggarwal, 2018). The abbreviated notation for one layer is shown in Figure 2.14.

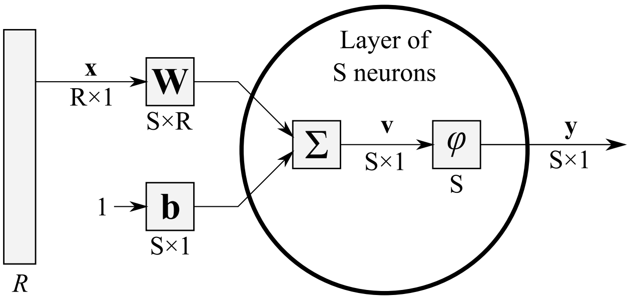


Figure 2.14: Abbreviated notation of a layer with S neurons and R inputs

When using multiple layers, the same notation is used. An example of a three layer neural network is shown in Figure 2.15. The inputs are not considered as a layer in the sense here discussed: they are a separate entity to the other layers, and therefore are not counted as one.

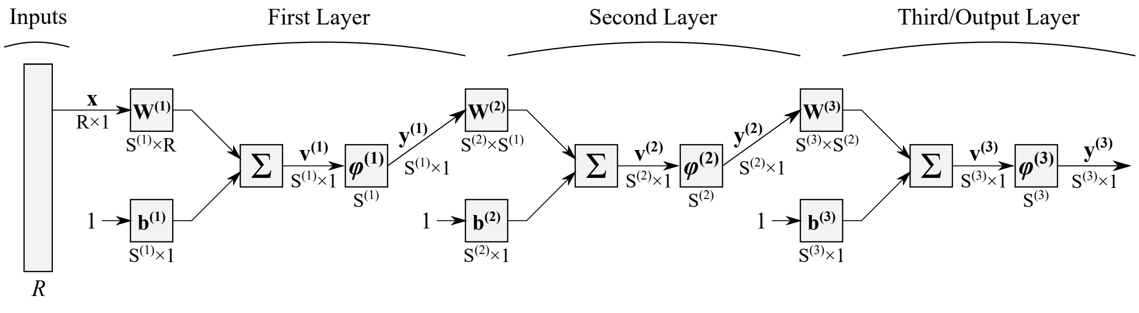


Figure 2.15: Three layer neural network example

To differentiate the matrices, a superscript is used identifying the layer number. For example, the weight matrix of layer 1 is represented as $\mathbf{W}^{(1)}$, and the number of neurons in layer 2 as $\mathbf{S}^{(2)}$. In this case, the third layer is also the last layer, so it is referred to as *output layer*. The layers in-between the input and the output layer, in this case the first and second, are called *hidden layers*.

Using a multi-layer neural network, one can approximate most functions with an arbitrarily precision (Hagan et al., 2014). If a multi-layer network has only two layers, one hidden and one output layer, it is referred to as a *shallow neural network*. For more than two layers, it is reffered to as a *deep neural network* (P. Kim, 2017).

2.4.5 Data Set Split

When training a neural network, the division of the data set is of great importance, since an incorrect division of data can yield overfitted networks, or poor performance (Aggarwal, 2018; Haykin, 2009). When the data is split it must be done so in a way that is at the same time random but has a representation of all types of data, as to avoid the previously referred problems. Typically, the original data is separated into three sets:

- **Training set**. Most of the data goes to this subset, and is used to train the neural network. Using this set, the network will train itself applying the algorithms discussed in the following sections.

- **Validation set**. This set is used to monitor the performance of the network after the training process has finished. If the network doesn't yeild a good performance, one returns to the train process with altered values of the network.

- **Test set**. When the netowrk has finished its training, and a suficiently good performance is achived, the test set is used to verifiy the performance with new data. With this last set one can easily see problems such as overfitting.

One of the most common ratios of set splitting is 0.7:0.15:0.15, for training, validation and testing, respectively (Haykin, 2009).

2.4.6 Training a Neural Network

To train a network a learning rule is used. There are three types of learning (Hagan et al., 2014) (P. Kim, 2017):

- **Supervised learning**. Providing a combination of inputs and target outputs, the network is trained by comparing its response to an input to the corresponding target, changing weights, until an adequate level of performance is fulfilled.
- **Reinforcement learning**. Similar to supervised learning, but instead of a target output being provided, the network is given a grade over its performance over certain inputs. This type of learning is more suited to control system applications.
- **Unsupervised learning**. Only inputs are given, with no targets available. Most algorithms of this type are used in clustering, such as pattern recognition.

Due to the nature of this work, only supervised learning is discussed, since a data set with both inputs and outputs will be used. The iterative weight modification according to some information is called the *learning rule* (P. Kim, 2017). In the case of supervised learning, the changes occur based on the error in relation with the target output. Multiple learning rules exist, and will be discussed in the following sections.

2.4.7 Delta Learning Rule

In 1960, Bernard Widrow and Marcian Hoff introduced the ADAptive LInear NEuron (ADALINE) network as well as the delta rule (Widrow & Hoff, 1960). Even though the delta rule is not capable of dealing with multi-layer networks, since it uses a linear activation function[4], it is still the best introduction to the subject of learning rules. Consider the network presented in Figure 2.16.

In expression form, the delta rule is given by the sum of the weight with the weight adjustment Δw_{ij}, such as

$$w_{ij} \leftarrow w_{ij} + \alpha e_i x_j = w_{ij} + \Delta w_{ij} \tag{2.17}$$

where w_{ij} is the weight between the output neuron i and input neuron j, α the learning rate, x_j the output from input neuron j, and e_i the error of the output neuron i,

[4] A multi layer network using only linear activation functions can be reduced to a single layer network

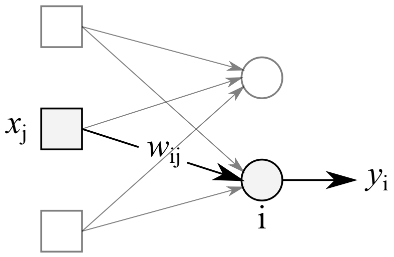


Figure 2.16: Single layer network

defined as the difference between the correct output (also called target) t_i and the real output y_i, such as

$$e_i \triangleq t_i - y_i \tag{2.18}$$

The learning rate α determines the weight change per iteration. If α is too high the network fails to converge, and if it is too low, the conversion will be slow.

The process of training a network with the delta rule is presented in Figure 2.17, where i is the subscript representing the number of the output node, and j represents the number of the input node. The weights are randomly initialized, so training multiple networks with the same training data yields different resulting networks. Every time all the weights are adjusted, an epoch passes, representing the number of iterations passed.

The previously presented delta rule is in fact only a particular case of the generalized delta rule, presented in Equation 2.19, which allows any activation function to be used.

$$w_{ij} \leftarrow w_{ij} + \alpha \delta_i x_j \tag{2.19}$$

In this equation, the error is replaced by δ_i, given by

$$\delta_i = \varphi'(v_i) e_i \tag{2.20}$$

where φ' is the derivative of the activation function, v_i the weighted sum, and e_i the error. By using a linear activation function $\varphi(v) = v$, the same result as before is achieved, since $\varphi'(v) = 1$.

The generalized delta rule still cannot train a multi layered network, since the error is the difference between the correct output and the networks output, which is not defined in the hidden layers of a multi layer network. It will however be used in the back propagation algorithm, which permits multi layered networks, and is presented in Section 2.4.8.

2.4.8 Back Propagation Algorithm

One of the most widely used learning rules, the back propagation algorithm is based on the generalized delta rule discussed in Section 2.4.7. With this approach, the error is propagated backwards throughout all the hidden layers, which then apply the delta rule to adjust the weights (Aggarwal, 2018; Haykin, 2009; P. Kim, 2017).

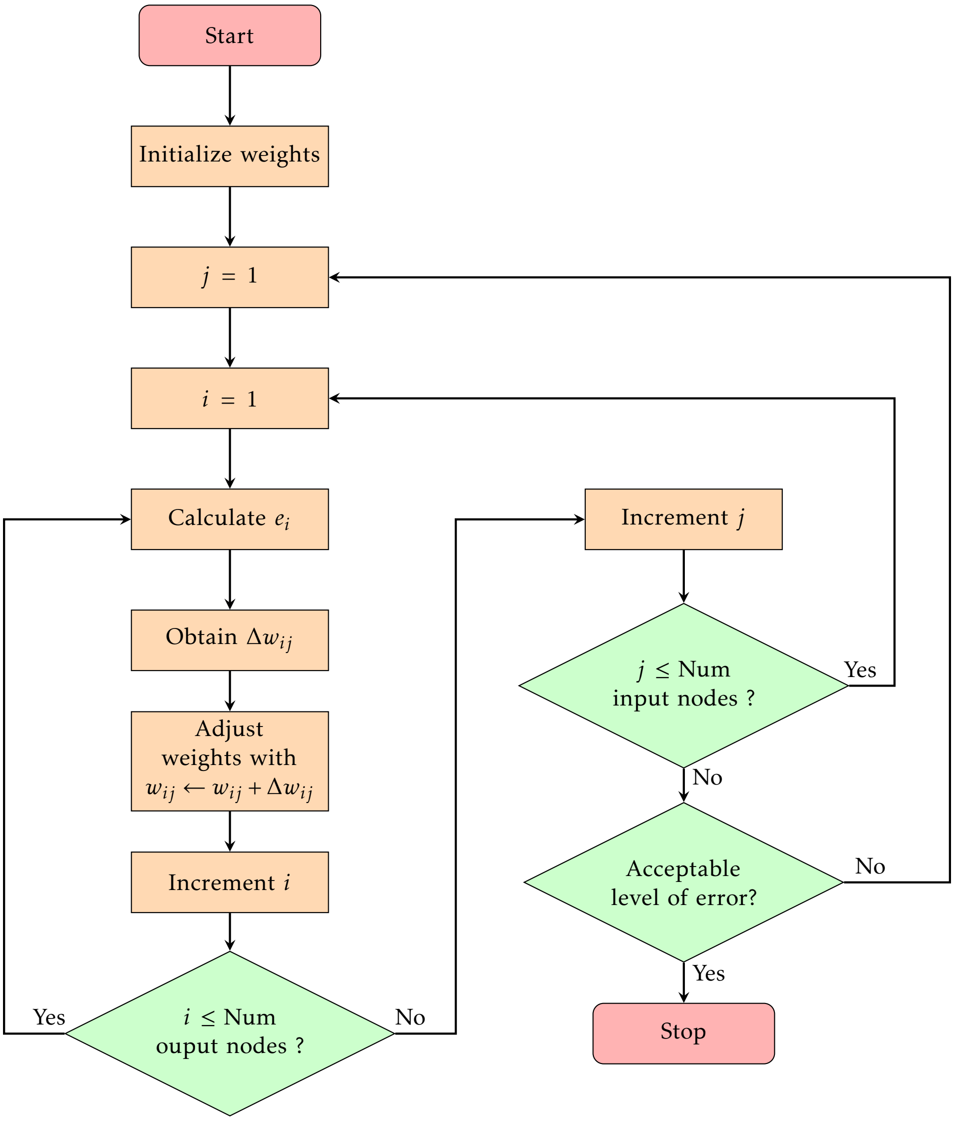

Start
Initialize weights
$j = 1$
$i = 1$
Calculate e_i
Obtain Δw_{ij}
Adjust weights with $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$
Increment i
$i \leq$ Num ouput nodes ?
Yes
No
Increment j
$j \leq$ Num input nodes ?
Yes
No
Acceptable level of error?
No
Yes
Stop

Figure 2.17: Single layer neural network training using delta rule, where i is the number of the output node and j is the number of the input node

Consider the two layer network presented in Figure 2.18. Firstly, the weighted sum of the hidden layer is calculated by using the weight matrix $\mathbf{W}^{(1)}$, such as

$$\begin{bmatrix} v_1^{(1)} \\ v_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq \mathbf{W}^{(1)} \boldsymbol{x} \tag{2.21}$$

Applying the activation function to the previous weight sum yields the outputs of the hidden neurons, in this case $y_1^{(1)}$ and $y_2^{(1)}$.

$$\begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \end{bmatrix} = \begin{bmatrix} \varphi(v_1^{(1)}) \\ \varphi(v_2^{(1)}) \end{bmatrix} \tag{2.22}$$

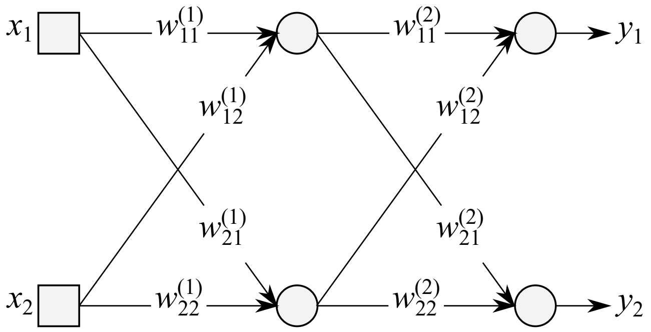


Figure 2.18: Shallow neural network, with one hidden layer and one output layer (adapted from P. Kim, 2017)

Similarly, the weighted sum of the output neurons is calculated, and by applying the activation function, the neural network output is given in Equation 2.23.

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \triangleq \mathbf{W}^{(2)}\boldsymbol{y}^{(1)} \qquad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \varphi(v_1) \\ \varphi(v_2) \end{bmatrix} \tag{2.23}$$

After the previous calculations have been completed, the back-propagation algorithm can be used. To begin, the generalized delta rule is used to calculate the delta of each output neuron using Equation 2.18. For the current example, δ_1 and δ_2 must be attained.

Secondly, one moves to the left, finding the hidden nodes layer, and computes their delta. This backwards movement is what gives this algorithm its name. The error of a hidden neuron is defined in the back-propagation algorithm as the weighted sum of the back-propagated deltas from the immediate right layer. In the example at hand, the errors are given by

$$\begin{aligned} e_1^{(1)} &= w_{11}^{(2)}\delta_1 + w_{21}^{(2)}\delta_2 \\ e_2^{(1)} &= w_{12}^{(2)}\delta_1 + w_{22}^{(2)}\delta_2 \end{aligned} \tag{2.24}$$

With the error obtained, the generalized delta rule is once again applied and the delta values are calculated for the hidden nodes.

$$\begin{aligned} \delta_1^{(1)} &= \varphi'(v_1^{(1)})e_1^{(1)} \\ \delta_2^{(1)} &= \varphi'(v_2^{(1)})e_2^{(1)} \end{aligned} \tag{2.25}$$

The process of obtaining the hidden layer errors can be presented in a matrix form, shown in Equation 2.26. It is therefore possible to obtain the error as the product of the transposed weight matrix and delta vector.

$$\begin{bmatrix} e_1^{(2)} \\ e_2^{(1)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} \\ w_{12}^{(2)} & w_{22}^{(2)} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \mathbf{W}^{(2)T} \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \tag{2.26}$$

After all the deltas are obtained, the network is trained using the generalized delta rule (Equation 2.19). The only difference is the calculation of the error for the hidden neurons.

The process of training a neural network using the back-propagation algorithm is presented in Figure 2.19.

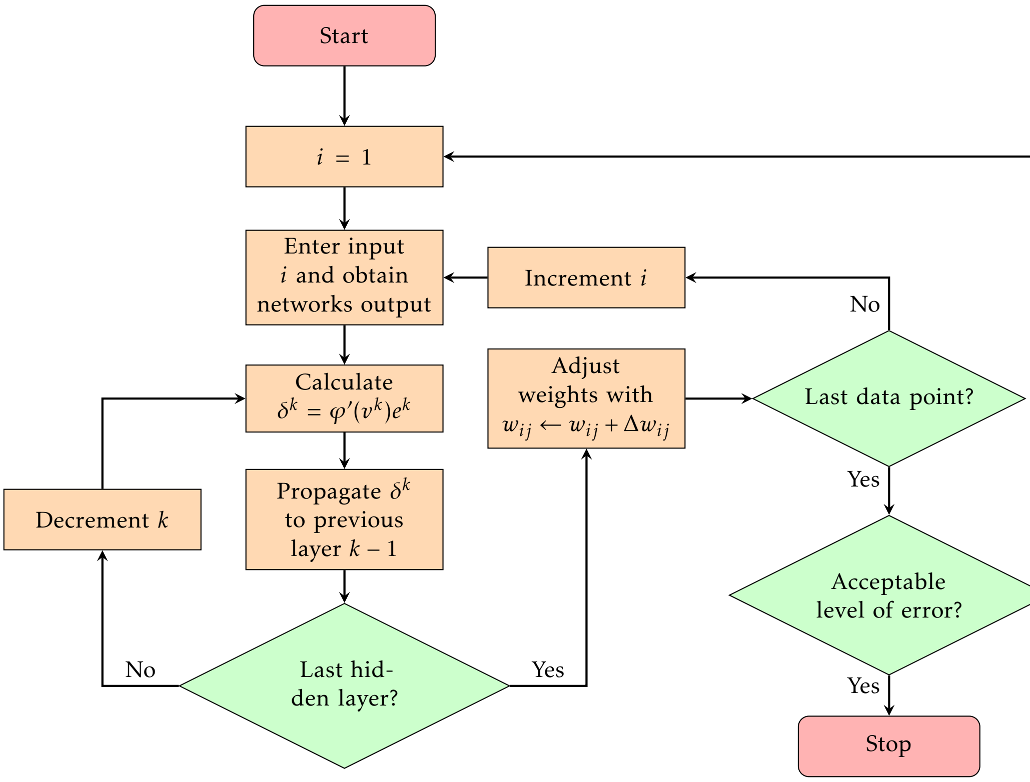

Start
$i = 1$
Enter input i and obtain networks output
Increment i
No
Calculate $\delta^k = \varphi'(v^k)e^k$
Adjust weights with $w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$
Last data point?
Yes
Decrement k
Propagate δ^k to previous layer $k-1$
Acceptable level of error?
No
No
Last hidden layer?
Yes
Yes
Stop

Figure 2.19: Neural network training using the back-propagation algorithm

2.4.9 Weight Adjustment Calculation

When training a neural network, one tries to minimize the cost functions (see Section 2.4.10). To do this, the weights need to be adjusted. There are several methods for calculating the weight adjustment Δw_{ij}, typically three (Aggarwal, 2018; P. Kim, 2017):

- **Stochastic Gradient Descent (SGD)**. The error is calculated for each training data and the weight is instantly updated. In other words, the number of times the weights are updated is the number of training data points. The delta rule presented in Section 2.4.7 is, therefore, a type of SGD.

- **Batch**. For each training data set, all weight updates are gathered, and the associated average is determined. All of the weights are only modified once, using the average. Due to the average calculation, this method consumes needs significantly more training time.

- **Mini-batch**. The mini-batch method, a hybrid of SGD and batch methods, selects a subset of the training set and applies the batch method to it. This method combines the speed of SGD with the stability of the batch method when using an appropriate number of data points.

2.4.10 Cost Function

In both the generalized delta rule and the back-propagation algorithm, the last and arguably most important decision has been the satisfaction of a certain level of error. The measurement of the error is made by the cost function. One of the most common cost functions is the sum of the MSE, given by

$$J = \sum_{i=1}^{S} \frac{1}{2}(t_i - y_i)^2 \tag{2.27}$$

where y_i is the output from the output neuron, t_i the correct output, and S the number of neurons on the output layer. From the general representation of a MSE in Figure 2.20 it is easy to see that for a greater difference between the target and the networks output, a greater cost value is obtained.

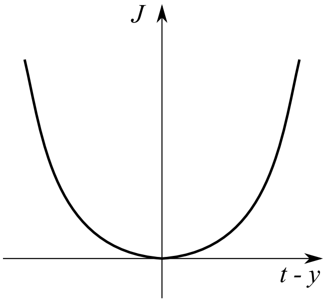


Figure 2.20: General representation of a MSE function

2.4.11 Levenberg-Marquardt Optimization Algorithm

A common training algorithm to train an ANN is the Levenberg-Marquardt (LM) optimization algorithm. A mixture of Gradient Descent (GD) and Gauss-Newton algorithms, the LM algorithm solves non linear squares minimization, so the cost function must have the form presented in Equation 2.28, where S is the number of neurons in the output layer and r a residual function. Representing r in vector form Equation 2.28 simplifies to Equation 2.29.

$$f(x) = \frac{1}{2} \sum_{j=1}^{S} r_j^2 \tag{2.28}$$

$$f(x) = \frac{1}{2} \|r(x)\|^2 \tag{2.29}$$

For linear r_j the Jacobian matrix is constant, so the gradient and the Hessian matrix are given by

$$\nabla f(x) = J^{\mathrm{T}}(Jx + r) \tag{2.30}$$

$$\nabla^2 f(x) = J^{\mathrm{T}} J \tag{2.31}$$

For the general case, the Jacobian matrix gives all the first order partial derivatives, so the gradient can be simplified to Equation 2.32 and the Hessian matrix to Equation 2.33.

$$\nabla f(x) = J(x)^{\mathrm{T}} r(x) \tag{2.32}$$

$$\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x) + \sum_{j=1}^{S} r_j(x) \nabla^2 r_j(x) \tag{2.33}$$

With small values of $\nabla^2 r_j(x)$ or small residuals $r_j(x)$, a sufficiently good approximation for the Hessian matrix is given by $\nabla^2 f(x) = J(x)^{\mathrm{T}} J(x)$ which is the same as in the linear case.

The GD algorithm updates it's parameters using the scaled gradient, such as

$$x_{i+1} = x_i - \lambda \nabla f \tag{2.34}$$

A simple GD suffers from convergence problems, since it takes big steps for large gradient and small steps for a small gradient, and it does not have in consideration the curvature. Using the 2$^{\mathrm{nd}}$ derivatives, the Gauss-Newton algorithm uses information given by the curvature and better gradient information. Expanding the gradient in a Taylor series around x_0 and assuming f quadratic around x_0, the Gauss-Newton update rule is

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{2.35}$$

This expression implicitly assumes f quadratic, so there is no need to evaluate the Hessian matrix. Using both these algorithms, the LM algorithm's rule of update is given by

$$x_{i+1} = x_i - (H + \lambda \mathrm{diag}[H])^{-1} \nabla f(x_i) \tag{2.36}$$

If the error decreases it means that the quadratic assumption of f is correct, so the factor λ is decreased by a certain factor as to reduce the effect of GD. If the error increases, the gradient must be followed, so λ is increased by the same factor. Since the Hessian matrix is proportional to the curvature of f, using it's diagonal gives a bigger movement for a smaller gradient and vice-versa.

2.4.12 Examples of Prediction of AC Loss Using ANN

In this section some works in which artificial intelligence models are used to estimate AC losses are discussed. It is worth noting that there have not been many research published that can be applicable to the topic of this work. As a result, only two works with a significant connection to the theme are presented.

2.4.12.1 Prediction of Non-sinusoidal AC Loss of Superconducting Tapes

In (Yazdani-Asrami, Taghipour-Gorjikolaie, Song, Zhang, & Yuan, 2020), a comparison between the performance of different Artificial Intelligence (AI) models is conducted using four measurements: Theil's U Coefficients (U_Accuracy and U_Quality), Root Mean Squared Error (RMSE) and Regression value (R-value). These models predict AC loss in HTS tape under non-sinusoidal currents.

Using H formulation FEM modelling method in COMSOL Multi-physics, the AC losses under different Total Harmonic Distortion (THD) are calculated, and later used as inputs to train the AI based models. Three order of harmonics were used: 3rd, 5th and 7th.

The models used in this paper are Support Vector Machine Regression Model (SVM), Generalized Linear Regression Model (GL), Decision Tree Model (DT), Feed Forward Neural Network (FFNN), Adaptive Neuro Fuzzy Interference System (ANFIS) and RBFNN. It is worth to note that while GL and DT are mathematical based regression model, SVM, FFNN, ANFIS and RBFNN are Computer Intelligence (CI) based models.

For each order of harmonics, 1110 experimental data were extracted, leading to a total of 3330 experimental data available. From these results the authors conclude that ANFIS and RBFNN present the better performance, with RBFNN showing better performance.

Results for a RBFNN with spread value of 0.5 are shown in Figure 2.3, showing values close to zero for U_Accuracy, U_Quality and RMSE, and one for R-value. In this paper the results for DT, FFNN and ANFIS are also considered promising.

		Training Data		
Model	**U_Accuracy**	**U_Quality**	**RMSE**	**R_Value**
RBF	0.0006±0.0004	0.0003±0.0002	0.0000±0.0000	1.0000±0.0000
		Testing Data		
Model	**U_Accuracy**	**U_Quality**	**RMSE**	**R_Value**
RBF	0.0034±0.0003	0.0017±0.0001	0.0003±0.0000	1.0000±0.0000

Table 2.3: Results of RBFNN models for all experimental testing data (Yazdani-Asrami, Taghipour-Gorjikolaie, Song, Zhang, & Yuan, 2020)

2.4.12.2 Prediction of AC Loss in Superconducting Round Filaments

In (Leclerc, Makong, Lorin, & Masson, 2016), a hybrid model combining an analytical scaling law and two ANNs is used to predict AC loss in superconducting round filaments. This model estimates hysteresis losses in a round filament of radius r carrying an AC transport current and on which an external elliptical magnetic field is applied. The electromagnetic behaviour of the superconductor is represented by the E–J power law.

Nine parameters are used to describe the system and calculate the losses Q: B_m, α, θ, f, k, I_m, J_c, n and r, where k is the applied field ellipticity. The critical electrical field is considered to be 1μV/cm.

The analytical scaling law model is a semi-analytical model that estimates numerical data generated by finite element analysis using COMSOL, and is used to predict losses when the superconducting filament is subjected to an elliptical magnetic field transverse to its cross-section with no transport current. A more detailed presentation of this model is made in (Lorin & Masson, 2013; Lorin, Netter, & Masson, 2015).

The first ANN estimated AC losses generated by any elliptical magnetic field and a transport current equal to the full penetration current of the filament. On the other hand, the second ANN uses as inputs the outputs of the first ANN and of the analytical scaling law model, and outputs the final prediction of the AC loss.

Each neural network is trained using the back propagation algorithm. The first ANN used 2400 data points for training and 200 for validation, and has three hidden layers with 35, 28 and 18 neurons, while the second ANN had 255 data points for the training process ans 200 validation points, with only two hidden layers of 4 neurons each.

The final output of this hybrid model presented an average absolute error of 3.12% in the validation set, allowing for a much faster estimation of AC losses in a superconducting filament when compared to FEM, with a low error.

Chapter 3

Experimental Apparatus and Results

The entire assembly of the experimental apparatus is given in this chapter, from the creation of the coils and cryostats to the loss measurement methods used. Both theoretical values derived from Norris' model and experimental values obtained via experience are presented at the end of the chapter.

3.1 Coil Assembly

The design of the coil molds was the initial stage in obtaining experimental results. Shown in Figure 3.2 are the previously utilized mould, used as a building foundation, and the improved upon one. The main difference is an accommodation space at the top for two pieces of copper for power supply purposes, as seen in (2). Unnecessary parts were also cut off to reduce material usage, as well as reduced height, shown in (1) and (3), respectively. The 3D printer used was the Creality Ender-3.

Copper pieces were also required for the coil's assembly in order to provide power to the superconductor. Each coil required four pieces, as shown in Figure 3.2, with the accompanying dimensions and a thickness of 5mm. The bigger component features a hole through which a wire can be inserted to supply current to the coil. Since all of the pieces were cut by hand, the real dimensions were only estimates. As seen in Figure 3.2(b), the copper pieces used in this work were repurposed from another experiment, thus many of them contain extra holes that are not necessary.

In addition, four screws and four nuts were employed to secure the superconducting tape and copper pieces, as shown in Figure 3.3. The screws on the underside hold a 3D printed plate in place, as well as two copper pieces that press the wire against them. The screws on the upper side press two copper components together, holding the superconductor in the middle in a mechanical fashion.

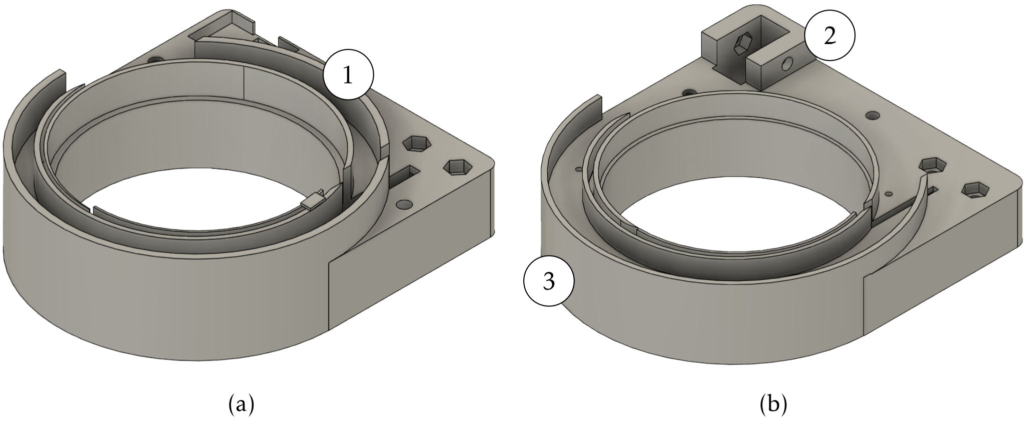


Figure 3.1: Coil mould (a) before improvements (b) after improvements. (1) extra material discarded, thus saving costs (2) accommodation for two copper pieces for power supply purposes (3) smaller height in comparison with the older model

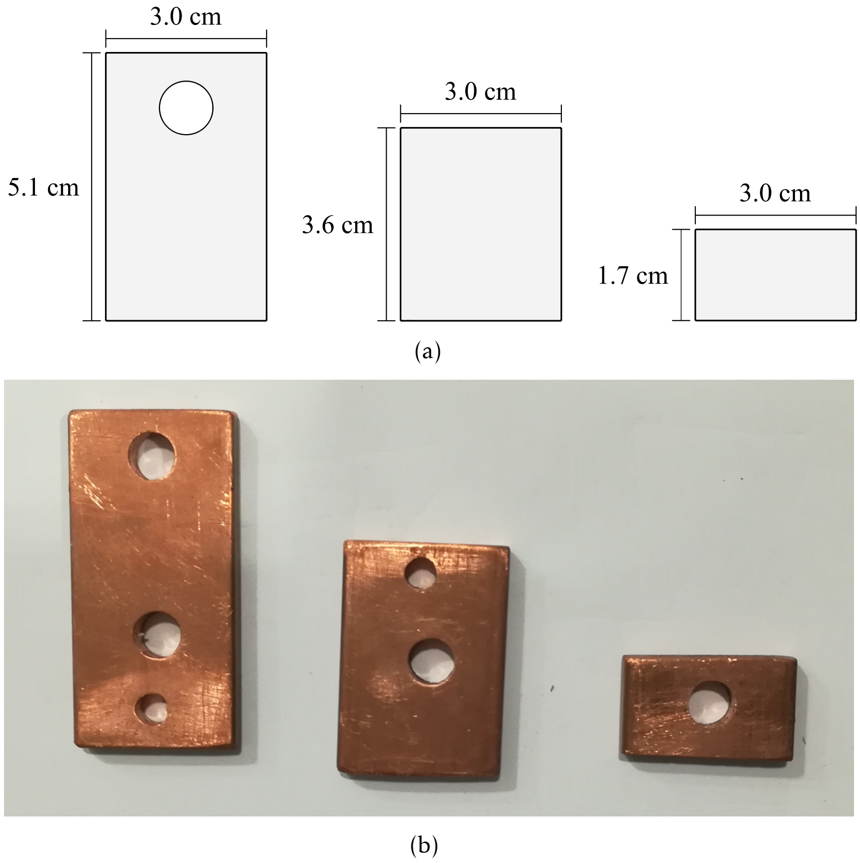
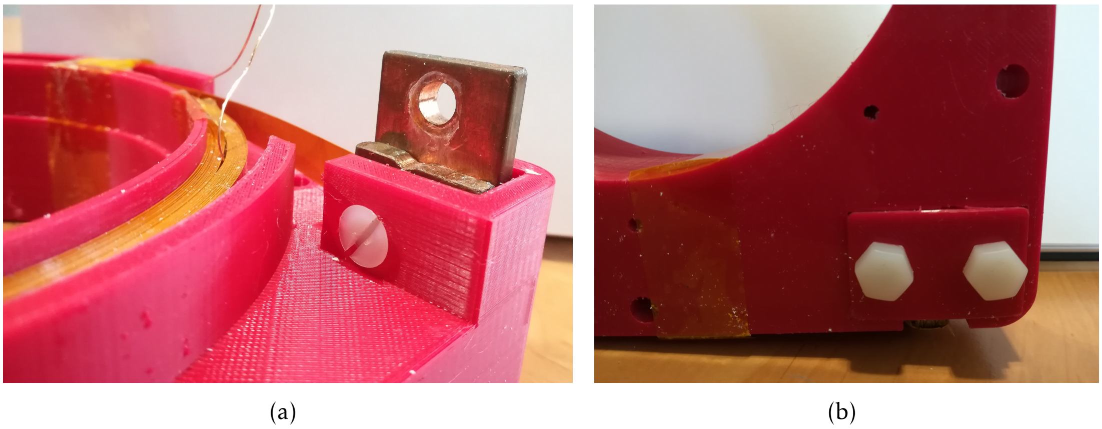


Figure 3.2: Copper parts (a) in a schematic (b) used in a coil



(a) (b)

Figure 3.3: Screws (a) at the top of the mould (b) at the bottom of the mould

For the assembly, the tape is first inserted into the underside and secured with screws and copper pieces. Following that, half of the desired turns are completed. At this point, a 4cm copper wire is tapped to the superconducting tape (for details of the tape see Section 3.4), which will later be used to measure loss. Following the completion of the second half of the turns, the tape is locked in the upper side between two pieces of copper, finishing the coil assembly, as illustrated in Figure 3.4.

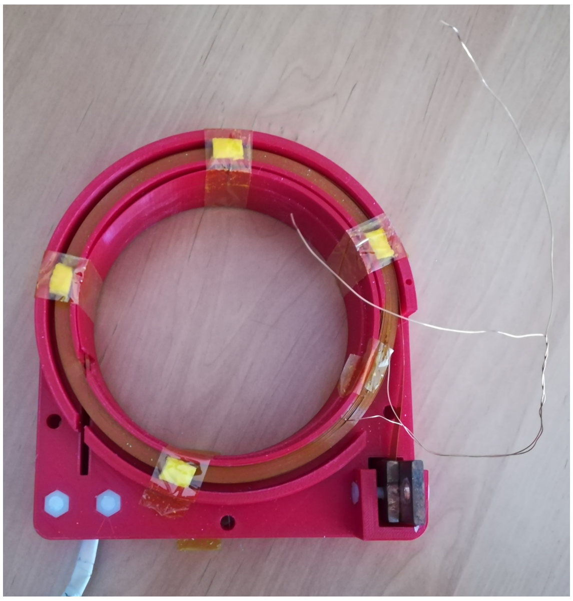


Figure 3.4: Finished coil

To ensure the length of the wire, the usage of a flex PCB board, as shown in Figure 3.5, was proposed instead of a copper wire. It was also intended to try to insert the PCB board after the coil was wound, since it would facilitate the winding of the coils and reduce man-hours. This solution could not be attempted due to the high cost of flex PCB, but this idea could be tried in a future work.

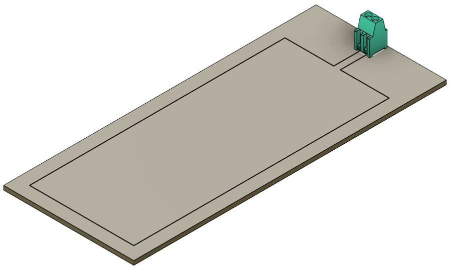


Figure 3.5: Flex PCB Fusion 360 model

3.2 Cryostat Building

A bath cooling cryostat, with styrofoam as the building material, was chosen, due to the low cost associated. From a block of 30x30x30cm it is possible to build two cryostats that fit the ferromagnetic core.

To build the cryostat, the tool presented in Figure 3.6 was used, from (Vilhena, 2012). By placing the block in the center of the tool and defining the radius, one can cut in a circular fashion with the desired depth, allowing to create the coveted shape for the cryostat.

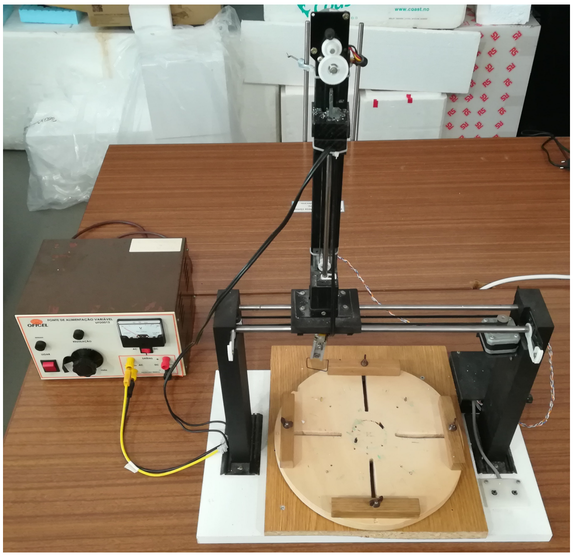


Figure 3.6: Tool utilized to cut the styrofoam (Vilhena, 2012)

Due to the low precision of the styrofoam cutting tool, some adjustments had to be made while cutting, resulting in an imperfect cryostat. Many times, instead of a clean cut in the walls, bubbles of styrofoam would pop creating irregularities in the walls. This irregularities are pointed in Figure 3.7, where a finished cryostat is presented.

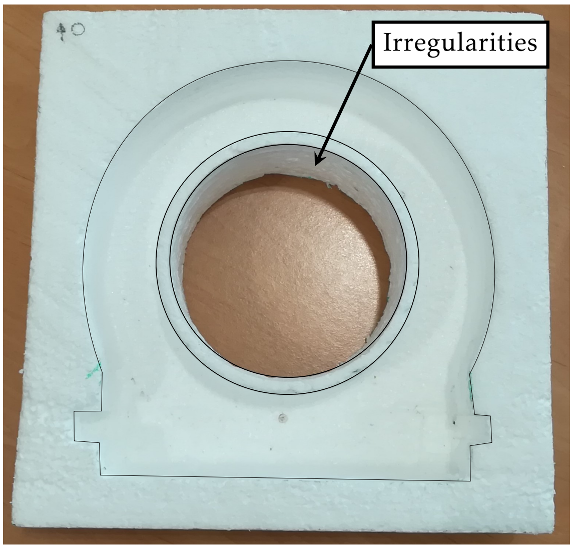


Figure 3.7: Finished cryostat with drawn contour for easy viewing.

3.3 AC Loss Measurement Methods

The most common methods of measuring losses in superconducting systems are presented in a brief manner in Section 2.2.8. For this work, two methods were used: the LIA method, and a direct method using an oscilloscope. A presentation of both methods is given, followed by the real test bench.

3.3.1 Lock-In Amplifier (LIA)

In a simplistic way, a LIA supplies a fixed frequency to the experiment and detects the experiments response at that frequency. A general block diagram is presented in Figure 3.8.

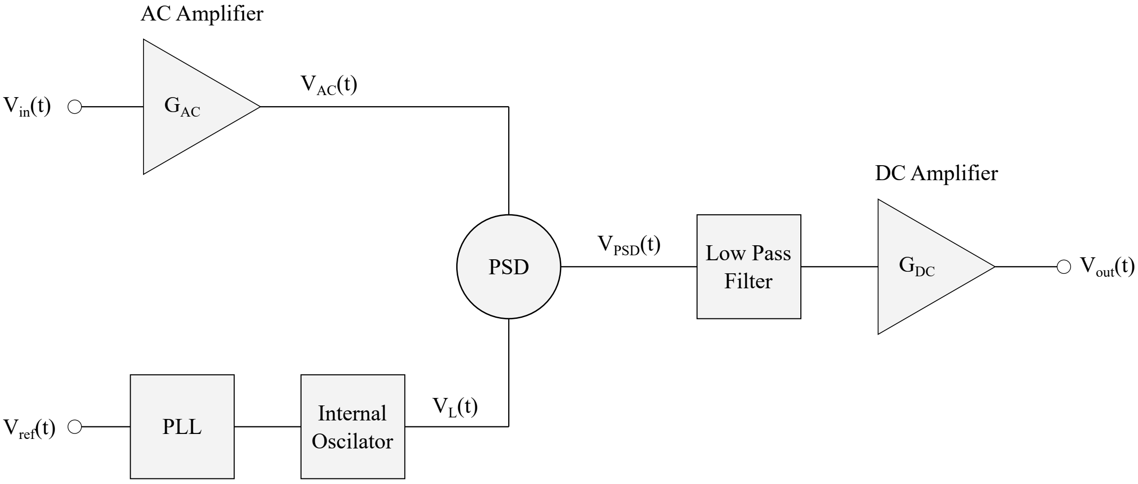


Figure 3.8: LIA block diagram (adapted from (Scofield, 1994))

Consider a sinusoidal input signal $V_{in}(t) = V_0 \cos(\omega_0 t + \theta_{sig})$. The input signal is amplified in the AC amplifier with a gain of G_{AC}, resulting in Equation 3.1, where V_0 is the input signal amplitude, $\omega_0 = 2\pi f_0$ the angular frequency, and θ_{sig} the phase of the signal.

$$V_{AC}(t) = G_{AC} V_0 \cos(\omega_0 t + \theta_{sig}) \tag{3.1}$$

The reference signal is produced either by an external source or using the LIA's internal oscillator. This reference needs to be both phase and frequency locked with the input signal, so a Phase-Locked Loop (PLL) is utilized. The LIA reference signal is

$$V_L(t) = E_0 \cos(\omega_0 t + \theta_L) \tag{3.2}$$

where E_0 is the amplitude of the reference signal and θ_L its phase. Both the input and the reference signals pass through a Phase-Sensitive Detector (PSD), taking both signals and calculating the product between them, leading to Equation 3.3.

$$V_{PSD}(t) = G_{AC} V_0 E_0 \cos(\omega_0 t + \theta_{sig}) \cos(\omega_0 t + \theta_L) \tag{3.3}$$

The relevance of the phase and frequency lock becomes clear at this point: with $\theta_{sig} - \theta_L$ constant, and taking in to consideration that both signals also have the same frequency ω_0, Equation 3.3 can be rewritten to

$$V_{PSD}(t) = \frac{1}{2} G_{AC} V_0 E_0 \left[\cos(2\omega_0 t + \theta_{sig} + \theta_L) + \cos(\theta_{sig} - \theta_L) \right] \tag{3.4}$$

which only has components at the second harmonic and at DC, both proportional to V_0. By passing the output of the PSD through a low-pass filter, the second harmonic can be significantly attenuated and the filtered DC signal obtained. Finally, the signal is amplified by passing it through a DC amplifier with G_{DC} gain. The final voltage is a DC signal proportional to V_0, as shown in Equation 3.5.

$$V_{out} = \frac{1}{2} G_{DC} G_{AC} V_0 E_0 \cos(\theta_{sig} - \theta_L) \tag{3.5}$$

To maximize V_{out}, θ_L should be adjusted as to get $\theta_{sig} - \theta_L \approx 0$. This is arranged by the PLL which adapts to changes in phase.

A single phase lock-in has only one PSD and gives the previous output. This type of LIA creates a scenario where if $\theta = \theta_{sig} - \theta_L = 90°$ the output measured will be 0.

To solve the phase dependency problem, dual-phase LIA add a second PSD. The second PSD multiples the signal with the reference in quadrature, in other words, shifted by 90°. Consider for simplicity the in-phase component $X = V_{sig} \cos(\theta)$ and quadrature component $Y = V_{sig} \sin(\theta)$. When computing the magnitude R the phase dependency is removed, measuring the signal amplitude directly (Equation 3.6).

$$R = \sqrt{X^2 + Y^2} = V_{sig} \tag{3.6}$$

3.3.2 LIA Measurement Method

Using the oscillator of the LIA, a signal can be generated with multiple frequencies and amplitudes. With the use of LabVIEW this signal is easily manipulated. The signal from the oscillator goes through both a power amplifier and a transformer as a means of amplification, and the amplified signal powers the superconducting coil. Between the power amplifier and the transformer a resistance bank is placed to get a closer impedance match. A Rogowski coil is placed between the transformer and the coil, serving as the first reference signal for the LIA. This signal will be purely inductive, so on the other channel of the LIA the resistive and inductive components of the AC loss are measured. This is accomplished by connecting the copper wire from the coil to the second channel of the LIA. This method is shown in Figure 3.9.

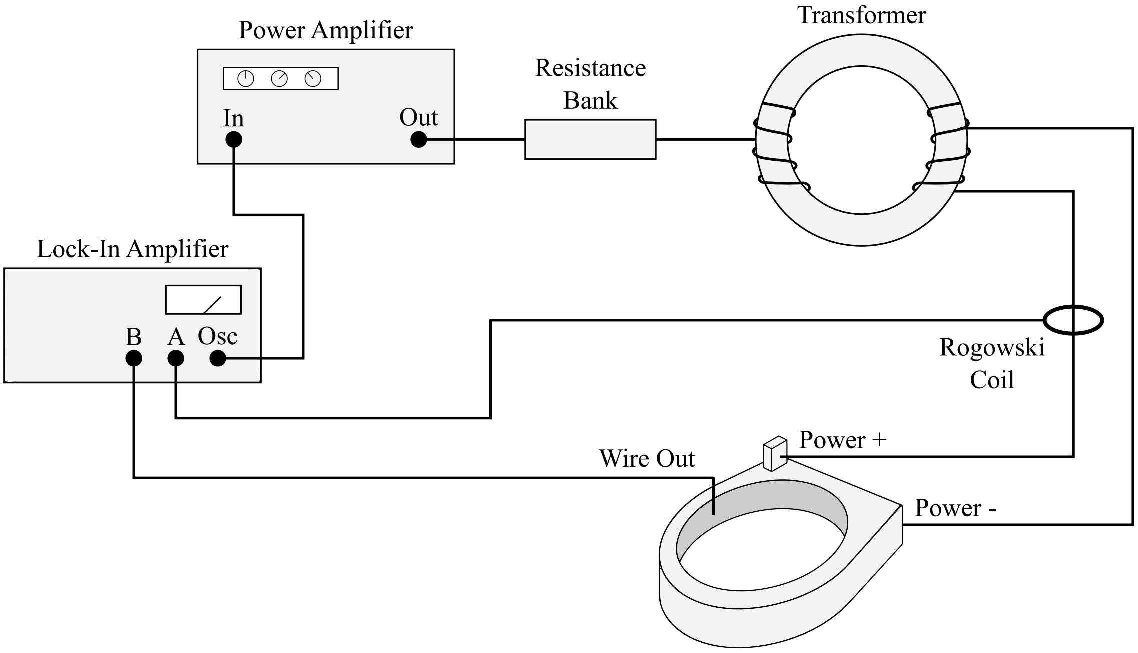


Figure 3.9: Schematic of the LIA measurement method

The transformer is only used during measurements of a single coil because it was not necessary to increase the signal during tests with coils inserted in a ferromagnetic coil.

The LIA used was the Signal Recovery 7265 DSP Lock-in Amplifier, connected to a PC and controlled via LabVIEW. An improved version of the LabVIEW software provided by the manufacturer was used, with the main difference being the automatic return of the losses in J/cycle/m. A QSC RMX 5050 audio amplifier was utilized as the power amplifier, with a maximum amplification at $\approx 4\Omega$, and the transformer was a toroidal transformer with a 1:6 ratio. The resistance bank had 3Ω, since the transformer had a resistive load of around 1.3Ω. The experimental setup used is shown in Figure 3.10.

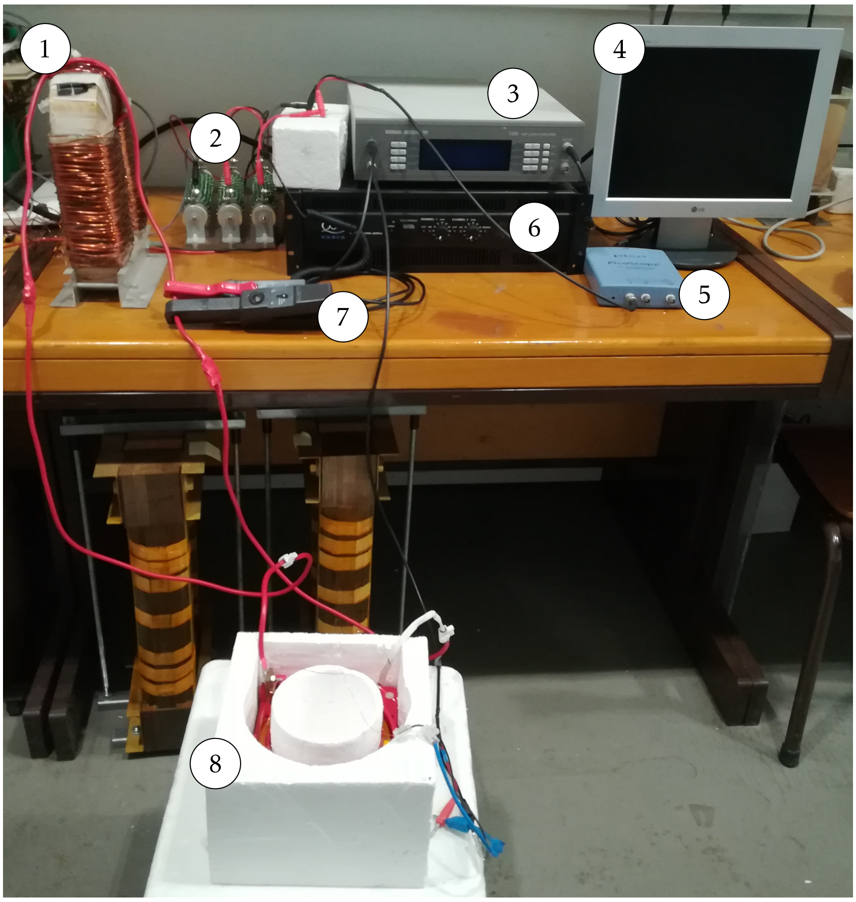


Figure 3.10: Experimental setup for AC loss measurement using a LIA. (1) Toroidal transformer (2) Resistance bank (3) Lock-In amplifier (4) Computer (5) Oscilloscope (6) Power amplifier (7) Rogowski coil (8) Superconducting coil

3.3.3 V–I Measurement Method

To acquire the AC losses in the transformer's secondary, a direct loss measuring method due to induced currents was used, in which both current and voltage are measured using an oscilloscope. The corresponding waveforms are passed through a CPU, where a point-to-point multiplication is made, yielding the instantaneous power in watts. Integrating for time and using the frequency of operation, the AC loss is obtained in the desired unit, J/cycle. A schematic for this method is shown in Figure 3.11.

3.3.4 Differences between measuring methods

Both the V–I and the LIA measuring methods have advantages and disadvantages. If using the LIA method the measurements are made much more easily, since that voltage variation is dealt with by the LIA, and the LIA allows for frequency variation. This method also has more possibilities of wrong measurements and invalid values due to the amplification needed and the impedance matching.

On the other hand, the V–I method provides higher reliability, since both the current

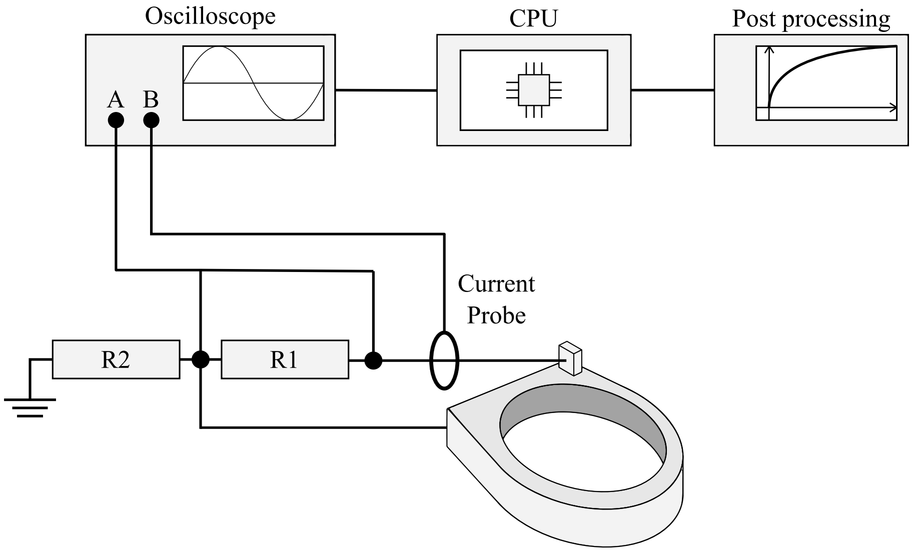


Figure 3.11: Schematic of the V–I measurement method

and voltage are measured directly using an oscilloscope. This method also enables the measurement of the secondary in a transformer. It has the drawback that when used alone, the voltage needs to be manually changed, and the AC loss needs to be calculated after the measurements are done, from the waveforms of the current and the voltage.

An ideal case uses both methods in tandem, especially for measures of AC loss in a superconducting transformer. The LIA method supplies the reference signal and measures the loss of the primary, while the V–I method measures the loss of the secondary due to induced currents. With this approach, the AC loss of both coils can be obtained simultaneously.

3.4 Tape and Coils' Properties

Only one type of superconducting tape was used, from Shanghai Superconductors, with its properties presented in Table 3.1.

Property	Shanghai Superconductor
Width (mm)	10 ± 0.1
Total Thickness (µm)	155 ± 5
Avg. I_C at 77K s.f. (A)	498
Substrate Thickness (µm)	50
Substrate Property	Hastelloy

Table 3.1: Shanghai Superconductor HTS tape properties

Various coils were built and numbered, using the following methodology: the first letter will always be C, as to distinguish from other superconducting coils in the lab; the second letter indicates the tape type, with B being Shanghai Superconductor; the last element is a number simply to distinguish the coils with the same tape. All coils with the respective dimensions are shown in Annex I.

3.5 Tape Coefficients

A HTS tape can be described approximately by the following expression:

$$Q = a \cdot \left(\frac{I}{I_c} \right)^b \tag{3.7}$$

where Q is the loss in $\mathrm{H\,m^{-1}}$, I/I_c the quotient between the current I and the critical current I_c, and a and b two coefficients to be found via least squares fitting.

Using Norris model (Norris, 1970), given in Equation 3.8, it is possible to analytically obtain an accurate representation of the AC loss in the tape using only its critical current (Gu & Han, 2005; Wang et al., 2018).

$$Q_{Norris} = \frac{I_c^2 \mu_0}{\pi} \left[\left(1 - \frac{I}{I_c}\right) \ln\left(1 - \frac{I}{I_c}\right) + \left(1 + \frac{I}{I_c}\right) \ln\left(1 + \frac{I}{I_c}\right) - \left(\frac{I}{I_c}\right)^2 \right] \tag{3.8}$$

Firstly multiple points are defined for I/I_c, for which the corresponding AC losses are calculated using Norris equation and shown in Figure 3.12. From these values, the coefficients a and b can be obtained via least square fitting. The a and b coefficients for the Shanghai Superconductor tape are 0.023834476341953 and 4.225648812595794, respectively. With this coefficients it is possible to get the AC loss analytically from Equation 3.7, which are also plotted in Figure 3.12. As it is possible to see, using the coefficients is a good description of a HTS tapes AC loss in self field.

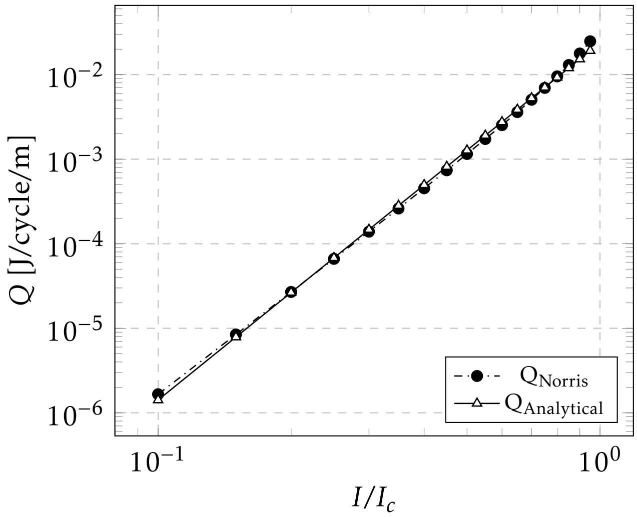


Figure 3.12: AC loss in the Shanghai Superconductor HTS tape using the Norris model ($\mathrm{Q_{Norris}}$) and the analytical model ($\mathrm{Q_{Analytical}}$)

3.6 Experimental Data

Due to problems with the experimental material it was not possible to use the LIA method. It was suspected that the power amplifier was working incorrectly, but since no other amplifier existed it cannot be confirmed which part of the apparatus was malfunctioning. As such, only values at 50Hz were obtained and the V–I measuring method the only one used. Figure 3.13 shows the coil test results at 50Hz. Since the experimental data for the coils was obtained after they were used in the transformer assembly, and the CB2 coil showed structural problem in the tape, it was not possible to test it alone.

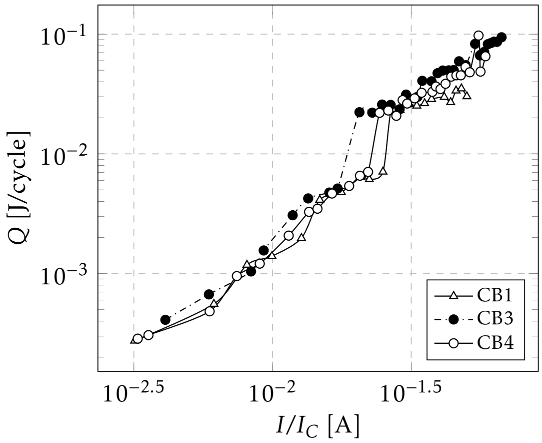


Figure 3.13: AC loss in coils at 50Hz with YBCO tape, where CB1 has 40 turns, CB3 60 turns, and CB4 50 turns.

The AC loss in the coils inserted in a ferromagnetic core is illustrated in Figure 3.14 for various coil configurations. The testing results were taken using the V–I method presented in Section 3.3.3.

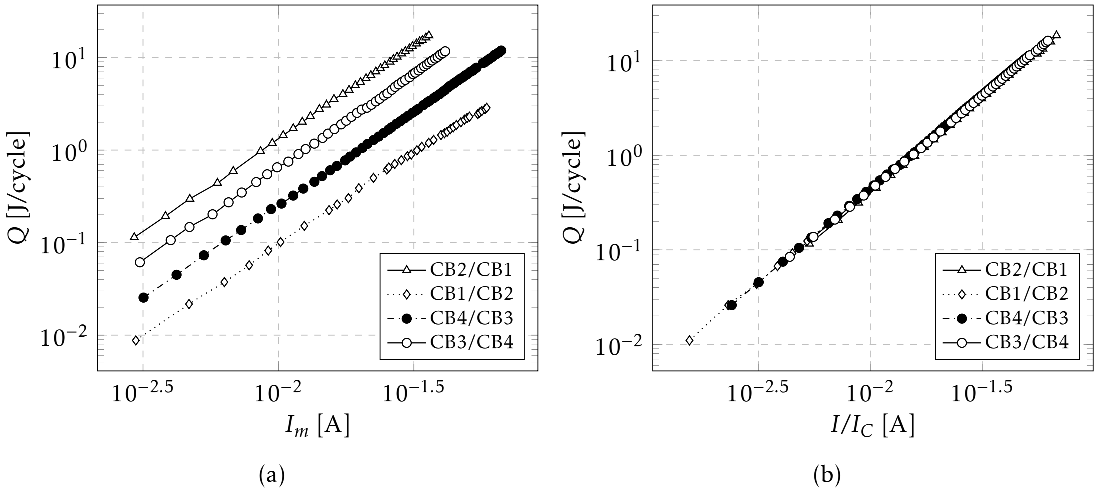


Figure 3.14: AC loss measured in (a) the primary; (b) the secondary; both at 50Hz, where the first coil in the label is the primary and the second the secondary (primary/secondary).

Chapter 4

ANN Modelling

In this chapter, multiple networks are trained and validated to predict AC loss. All of the networks are implemented in MATLAB using the Deep Learning Toolbox library, due to the implementation simplicity. The building process of the networks is presented, namely the training, validation and testing. All the results of the networks are presented, both from a global network and a series network.

4.1 Data Import

Firstly, the data is imported to the script and the unused information is dropped, while the current is changed to I/I_c based on the critical current of the tapes.

After the data has been cleared, a division is made between inputs and targets. An example of an input for the coils' network is given in Table 4.1, while the target will be the corresponding loss Q.

Characteristic	Value
$\mathbf{I/I_c}$	0.1
Frequency [Hz]	50
Turns	60
Internal diam. [mm]	118
a	0.023834476341953
b	4.225648812595794

Table 4.1: Example of an input received by the coil neural network

4.2 Global Network

Figure 4.1 depicts a schematic of a global network for estimating AC loss in an HTS transformer. The inputs used for this network are frequency (f in Hz), current of the primary (I_p/I_c), current of the secondary (I_s/I_c), number of turns of the primary (N_p), number of turns in the secondary (N_s), a and b coefficients, internal diameter of the coil (Int. Diam.) and the core characteristics (Core Ch.). The core characteristics are: core height, core diameter, window size, core grain and lamination width, all in mm. On the other hand, the outputs of the ANN are the AC loss in the primary (Q_p in J/cycle) and and in the secondary (Q_s in J/cycle).

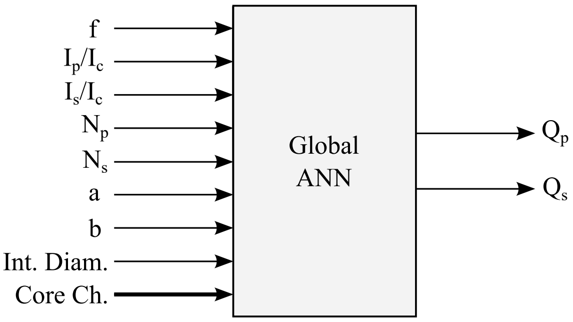


Figure 4.1: Global ANN schematic for prediction of AC loss.

Figure 4.2 illustrates the best performing network, a three hidden layer neural network. The first layer is made up of 15 neurons using the tansig function, the second of 12 neurons that use the Elliot sigmoid function, and the third of 13 neurons that utilize the log-sigmoid activation function.

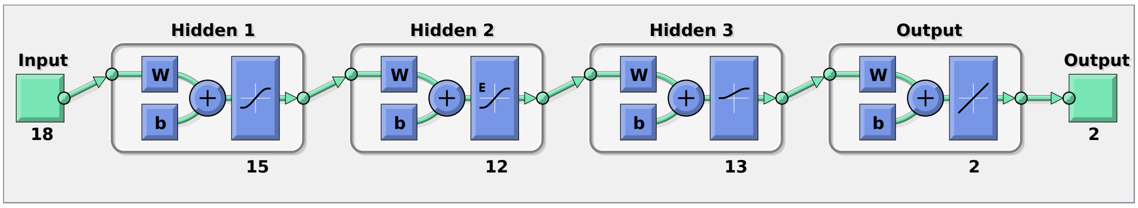
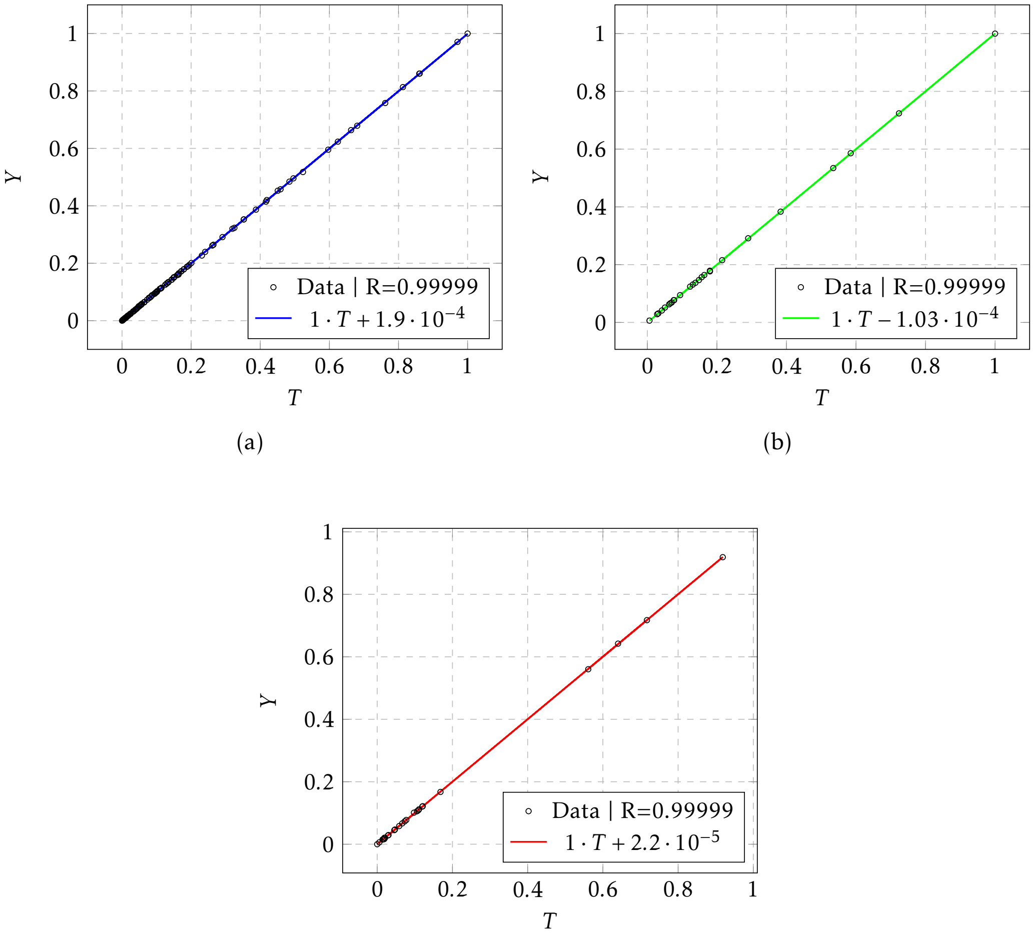


Figure 4.2: Global neural network with three hidden layers. The first layer has 15 neurons using the tansig function, the second layer has 12 neurons using the Elliot sigmoid function, and the third layer has 13 neurons using the log-sigmoid function.

This network presented a validation performance of 2.6879×10^{-6} at the 155th epoch, and had an R-value of 0.99999 for training, validation and testing. All three of the regression plots are shown in Figure 4.3. From this values it is possible to see that the network provides an extremely good performance.



(a)

(b)

(c)

Figure 4.3: FFNN regression plots for the transformer device. (a) Training (b) Validation (c) Testing

4.3 Coil ANN in Series with Transformer ANN

When the need to retrain the global network presented in 4.2 arises, due to additional data being collected, every data point must be used, regardless of the type of new input. For example, if new measurements are taken on a different core but with previously used coils, the network cannot be trained solely with transformer data points; the coil data points must also be inputted. This necessity to retrain with every type of input leads to lesser flexibility of the network, and a higher training time when new data is obtained. For this reason, a series-like montage of two different ANNs if proposed, differentiating between measurements made on coils alone, and on the HTS transformer.

In Figure 4.4 the schematic for this assembly is presented. Certain inputs must be entered in both networks, namely the frequency, currents, number of turns and the *a* and *b* coefficients. The first network is the coil network, and has as its inputs the frequency, current, number of turns, and the *a* and *b* coefficients. This network provides as its output the AC loss in an independent coil. Two coil networks must be used, one for the primary and another for the secondary. In the transformer network, the outputs of both the coil networks serve as inputs, as well as the core characteristics.

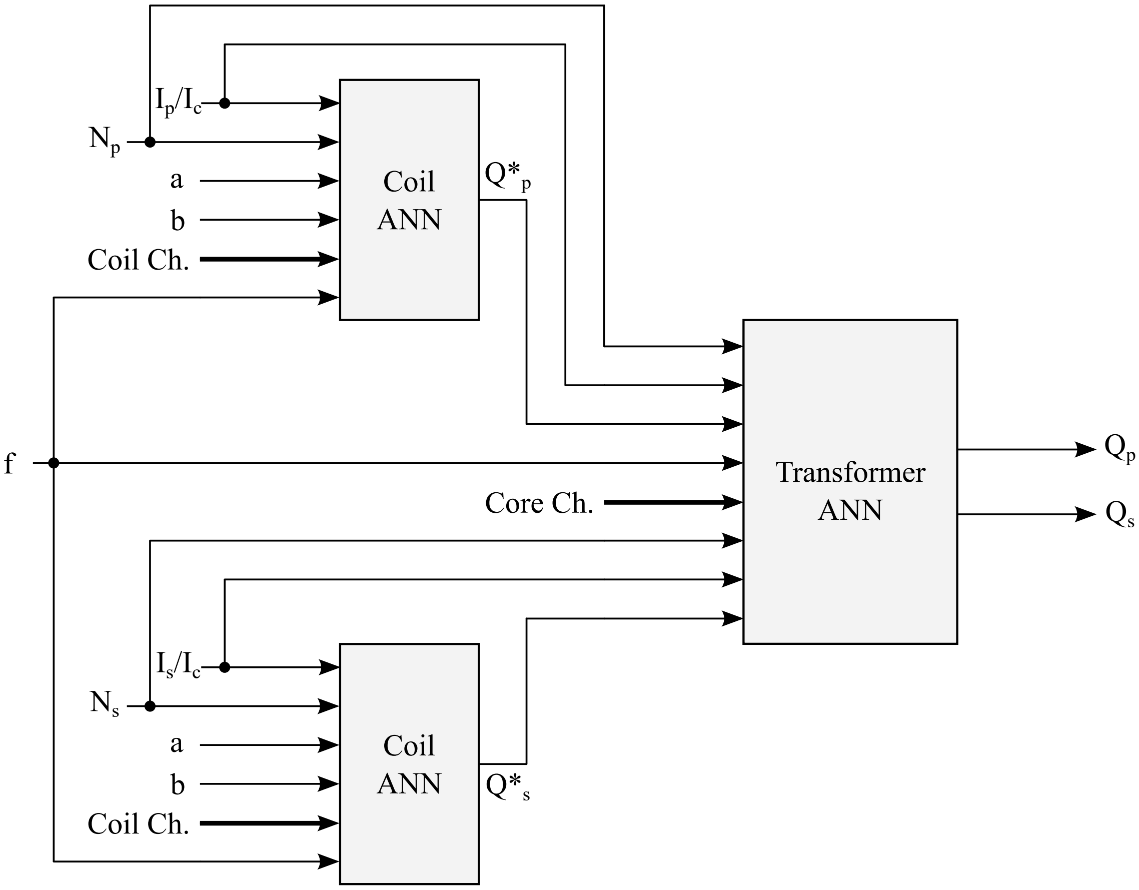


Figure 4.4: Series-like schematic of the two different networks.

Represented in Figure 4.5 is the best performing coil network, which has 12 neurons in the first layer, 15 in the second, and 13 in the third. An Elliot sigmoid activation function was used in the first two hidden layers, a radbas function in the third and a linear function in the output layer.

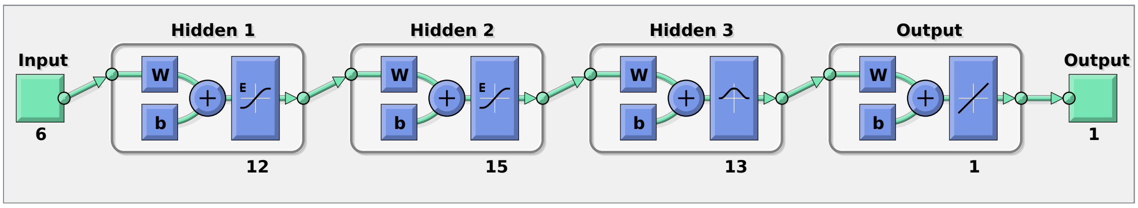
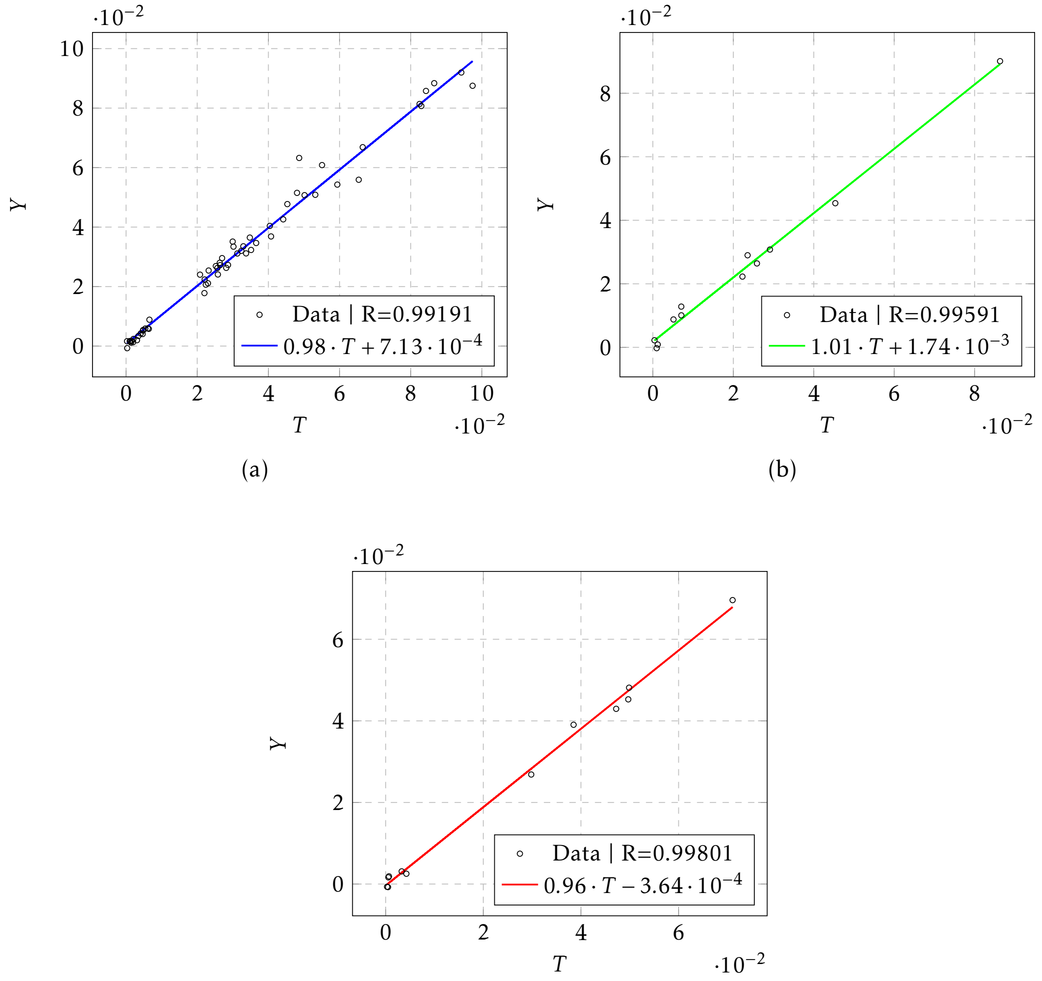


Figure 4.5: Feed forward network for the coils, with 12 neurons in the first layer, 15 in the second, and 13 in the third. The activation functions used are the Elliot sigmoid function in the first two layers, and the radbas function in the third layer.

The networks results are plotted in Figure 4.6 with a regression line. Both the training and the validation steps resulted in R-values higher than 0.99, and the testing step resulted in a R-value of 0.99801. This results are especially promising considering the low amount of inputs available to train the network.



(a)

(b)

(c)

Figure 4.6: FFNN regression plots for the coils. (a) Training (b) Validation (c) Testing

The completed coil network can now be used to train the transformer network. To begin, several inputs are provided to the coil network, yielding AC loss values for the primary and secondary, which are now used as inputs for the transformer network, together with the core and coil parameters. In this approach, a series network is built that matches the outcomes of the global network of Section 4.2 while providing greater training flexibility. The number of turns of the primary and secondary coils, internal diameter of both coils, *a* and *b* coefficients, frequency, current over critical current, core characteristics and AC loss of both coils generated by the coil network are the inputs to this network.

The best performing transformer network is presented in Figure 4.7, having two layers. Training with three layers caused overfitting, so it was discarded. Both the first and second layer use a radbas activation function, having 21 and 20 neurons respectively.

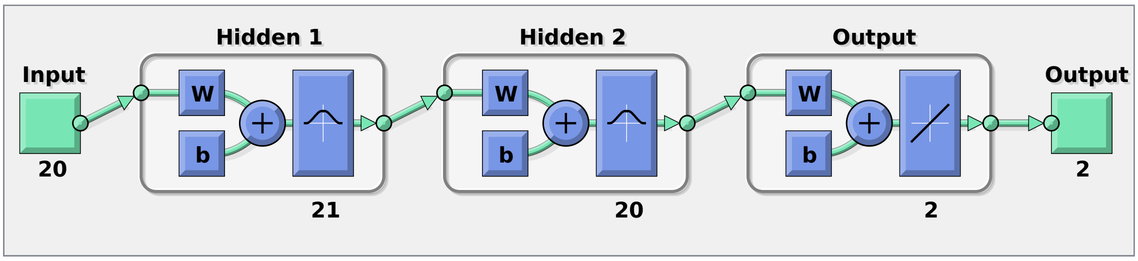
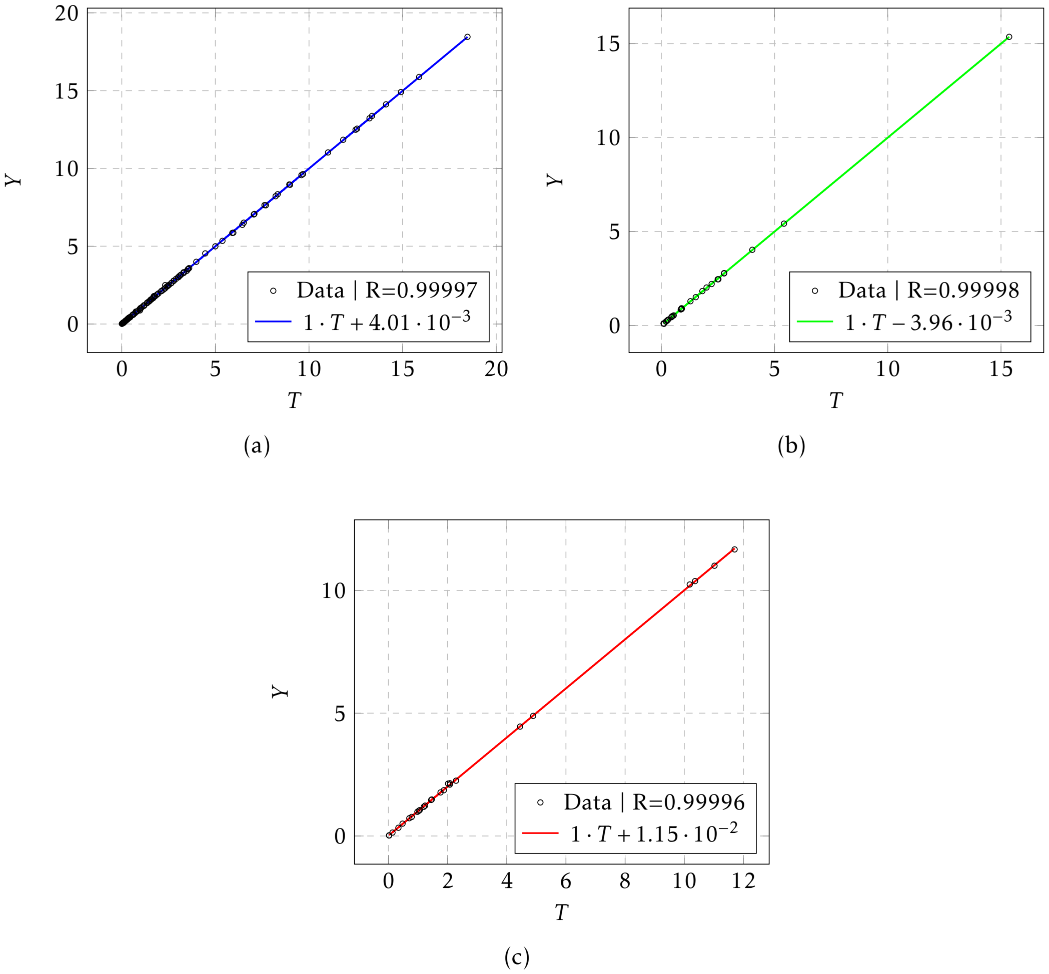


Figure 4.7: Two layered transformer network with radbas activation function in the hidden layers

With a validation performance of 0.0015729 at the 22nd epoch, this ANN has an R-value of 0.99997 for training, 0.99998 for validation and 0.99996 for testing. The regression plots are presented in Figure 4.8.



(a)

(b)

(c)

Figure 4.8: Regression plots of the transformer network trained with a series configuration. (a) Training (b) Validation (c) Testing

This values are very close approximation to the global network, but the series assembly of the networks provides a higher flexibility of training and usage. Of note is that even tough the CB2 coil was not used to train the coil network, its characteristics were still used to generate data for the transformer network.

4.4 Discussion

With this prediction technique, the AC loss for a transformer with HTS coils was obtained with very good results. Having low simulation time, an ANN shows high robustness against parameters changes, such as number of turns, and provides fast approximations of the loss. With a fast prediction it is possible to use the AC loss in the design process of a HTS device, namely in the cryogenic system.

On the other hand, a neural network needs multiple data points to be trained for higher precision and generalization. These data points must be taken either experimentally or by the use of FEM, which proves to be expensive and time-consuming. Another drawback is that when feeding inputs to the network, one must be sure that the measurements were taken correctly because if not, the network will be trained for wrong values and will not be able to provide trustworthy loss values.

It was not possible to achieve high current levels relative to the critical current in this work. As a result, the networks do not function adequately at higher current levels since no training for such values happened.

When compared to FEM, it shows tremendous promise because it requires significantly less time and computational resources, with the main time investment being the acquisition of data values, and in the future this time can be reduced with other method of generating data, including using ANN for creating data points.

Chapter 5

Conclusions and Future Work

A conclusion is presented in this chapter. Due to the research nature of this work, it is necessary to identify future work to continue the investigation of the theme.

5.1 Conclusions

The presented work shows experimental results of AC loss in superconducting devices, namely HTS transformers with ferromagnetic cores. With those results, neural networks were developed as to be able to predict the AC loss without the need of measurements or simulations using FEM.

The experimental results from the primary show the expected result that for a higher number of turns, a bigger AC loss exists. On the other hand, the secondary, which was connected to a resistive load, has approximately the same loss independent of the number of turns of each coil. As for the coils' losses, the results were more erratic, and one of the coils could not be tested. Of note is that the values of the *a* and *b* coefficients used to describe the tape should have been obtained experimentally, but such was not possible due to lack of equipment.

The global network, which has as inputs all information about the device's description, shows promising results for predicting AC loss in HTS devices, with R-values near to 1 for training, validation, and testing. The series network provided extremely close results, while adding flexibility and robustness, even with a low number of data points.

The computational time and power necessary for the neural networks presented is much lower than what would be needed for a FEM simulation, and a change of one parameter does not imply a whole new simulation. With accurate calculations of AC loss, the cryogenic system can be optimized, lowering the costs for both new systems in the design phase as well as existing systems, which can be improved.

5.2 Future Work

Due to the good performance of the neural networks with limitations in terms of the availability of data, future work with more data points and for different devices is of interest. Multiple suggestions and improvements that were not able to be performed in this work are now proposed:

- Application to different types of devices, such as HTS motors or HTS current limiters.
- Expansion to different frequencies and other types of superconducting tape.
- Conduct the experimental procedure in a three phase transformer.
- Design a cryogenic system based in the losses given by the networks.
- Winding of the coils in a more streamlined manner, to guarantee a more uniform result.
- Use of closed cryostats, for less evaporation of liquid nitrogen.
- Use of a flex PCB with a copper wire to measure using the LIA method, ensuring the size of the copper wire.
- Try a non-invasive measurement using a claw method, with two copper wires, one at the top and another at the bottom of the coil. In this way, it is possible to crate a plier like mechanism to measure the AC loss in coils with the system still working.

Bibliography

Aggarwal, C. (2018). *Neural networks and deep learning : A textbook*. Cham, Switzerland: Springer.

Bean, C. P. (1962). Magnetization of hard superconductors. *Phys. Rev. Lett.*, *8*, 250–253. doi:10.1103/PhysRevLett.8.250

Bean, C. P. (1964). Magnetization of high-field superconductors. *Rev. Mod. Phys.*, *36*, 31–39. doi:10.1103/RevModPhys.36.31

Bednorz, J. G., & Müller, K. A. (1986). Possible T_c superconductivity in the Ba-La-Cu-O system. *Zeitschrift für Physik B Condensed Matter*, *64*(2), 189–193. doi:10.1007/bf01303701

Brambilla, R., Grilli, F., & Martini, L. (2006). Development of an edge-element model for AC loss computation of high-temperature superconductors. *Superconductor Science and Technology*, *20*, 16. doi:10.1088/0953-2048/20/1/004

Brambilla, R., Grilli, F., Martini, L., & Sirois, F. (2008). Integral equations for the current density in thin conductors and their solution by the finite-element method. *Superconductor Science and Technology*, *21*. doi:10.1088/0953-2048/21/10/105008

Buzea, C., & Yamashita, T. (2001). Review of the superconducting properties of MgB2. *Superconductor Science and Technology*, *14*(11), R115–R146. doi:10.1088/0953-2048/14/11/201

Friedman, A., Wolfus, Y., Kopansky, F., & Yeshurun, Y. (2012). Critical currents and ac losses in ybco coils. *Physics Procedia*, *36*, 1169–1174. SUPERCONDUCTIVITY CENTENNIAL Conference 2011. doi:https://doi.org/10.1016/j.phpro.2012.06.196

Gu, C., & Han, Z. (2005). Calculation of ac losses in hts tape with fea program ansys. *IEEE Transactions on Applied Superconductivity*, *15*(2), 2859–2862. doi:10.1109/TASC.2005.848248

Guru, B., & Hiziroglu, H. (2001). *Electric machinery and transformers* (Third). Oxford University Press.

Hagan, M. T., Demuth, H. B., Beale, M. H., & Jesús, O. (2014). *Neural network design* (Second). Boulder, CO: University of Colorado Bookstore.

Haykin, S. (1999). *Neural networks : A comprehensive foundation*. Delhi: Pearson Education.

Haykin, S. (2009). *Neural networks and learning machines* (Third). New York: Prentice Hall/Pearson.

Ijaduola, A., Thompson, J., Goyal, A., Thieme, C., & Marken, K. (2004). Magnetism and ferromagnetic loss in Ni–W textured substrates for coated conductors. *Physica C: Superconductivity*, *403*(3), 163–171. doi:https://doi.org/10.1016/j.physc.2003.12.003

Kajikawa, K., Iwakuma, M., Funaki, K., Wada, M., & Takenaka, A. (1999). Influences of geometrical configuration on ac loss measurement with pickup-coil method. *IEEE Transactions on Applied Superconductivity*, *9*(2), 746–749. doi:10.1109/77.783402

Kim, J.-H., Kim, C. H., Iyyani, G., Kvitkovic, J., & Pamidi, S. (2011). Transport ac loss measurements in superconducting coils. *IEEE Transactions on Applied Superconductivity*, *21*(3), 3269–3272. doi:10.1109/TASC.2010.2089485

Kim, P. (2017). *MATLAB deep learning with machine learning, neural networks and artificial intelligence*. New York, NY: Apress.

Leclerc, J., Makong, L., Lorin, C., & Masson, P. (2016). Artificial neural networks for ac losses prediction in superconducting round filaments. *Superconductor Science and Technology*, *29*. doi:10.1088/0953-2048/29/6/065008

London, H., & London, F. (1935). The electromagnetic equations of the supraconductor. *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, *149*(866), 71–88. doi:10.1098/rspa.1935.0048

Lorin, C., & Masson, P. J. (2013). Numerical analysis of the impact of elliptical fields on magnetization losses. *IEEE Transactions on Applied Superconductivity*, *23*(3), 8201405–8201405. doi:10.1109/TASC.2013.2239347

Lorin, C., Netter, D., & Masson, P. J. (2015). Scaling law for hysteresis losses in round superconductors magnetized by alternating, rotating or elliptical magnetic fields. *IEEE Transactions on Applied Superconductivity*, *25*(1), 1–12. doi:10.1109/TASC.2014.2341255

Lourenço, J. M. (2021). *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. Retrieved from https://github.com/joaomlourenco/novathesis/raw/master/template.pdf

Martínez, J. C. (2010). *Analysis of AC losses in superconducting electrical components for application in the design of electrical systems* (Doctoral dissertation, Universidad de Extremadura).

Meissner, W., & Ochsenfeld, R. (1933). Ein neuer effekt bei eintritt der supraleitfähigkeit. *Die Naturwissenschaften*, *21*(44), 787–788. doi:10.1007/bf01504252

Melhem, Z. (2012). *High temperature superconductors (HTS) for energy applications*. Cambridge: Woodhead Publishing Limited.

Norris, W. (1970). Calculation of hysteresis losses in hard superconductors carrying ac: Isolated conductors and edges of thin sheets. *Journal of Physics D: Applied Physics*, *3*, 489.

Okamoto, H., Sumiyoshi, F., Miyoshi, K., & Suzuki, Y. (2006). The nitrogen boil-off method for measuring ac losses in hts coils. *IEEE Transactions on Applied Superconductivity*, *16*(2), 105–107. doi:10.1109/TASC.2005.869557

Onnes, H., Gavroglu, K., & Goudaroulis, Y. (1991). *Through measurement to knowledge: The selected papers of heike kamerlingh onnes 1853-1926*. Dordrecht: Springer Netherlands.

Oomen, M. P. (2000). *AC loss in superconducting tapes and cables* (Doctoral dissertation, Universiteit Twente).

Pina, J. M. (2010). *Desenho e modelização de sistemas de energia empregando materiais supercondutores de alta temperatura* (Doctoral dissertation, UNL/FCT). Retrieved from http://hdl.handle.net/10362/5059

Poole, C. (1999). *Handbook of superconductivity*. San Diego, CA: Academic Press.

Pronto, A. (2010). *Análise de perdas em sistemas de energia que empregam materiais supercondutores de alta temperatura* (Doctoral dissertation, UNL/FCT). Retrieved from http://hdl.handle.net/10362/5116

Rogalla, H., & Kes, P. H. (2012). *100 years of superconductivity*. Boca Raton: CRC Press/Taylor & Francis Group.

Scofield, J. H. (1994). Frequency-domain description of a lock-in amplifier. *American Journal of Physics*, *62*(2), 129–133. doi:10.1119/1.17629

Seidel, P. (2015). *Applied superconductivity : Handbook on devices and applications*. Weinheim an der Bergstrasse, Germany: Wiley-VCH.

Silsbee, F. B. (1916). A note on electrical conduction in metals at low temperatures. *Journal of the Washington Academy of Sciences*, *6*(17), 597–602.

Sjöström, M. (2001). *Hysteresis modelling of high temperature superconductors* (Doctoral dissertation, École Polytechnique Fédérale de Lausanne). doi:10.5075/epfl-thesis-2372

Steinmetz, C. P. (1892). On the law of hysteresis. *American Institute of Electrical Engineers Transactions*, *9*, 3–64.

Tuyn, W., & Onnes, H. (1926). The disturbance of supra-conductivity by magnetic fields and currents. the hypothesis of Silsbee. *Journal of the Franklin Institute*, *201*(4), 379–410. doi:https://doi.org/10.1016/S0016-0032(26)90097-5

Vilhena, N. (2012). *Construção e ensaio de um limitador de corrente supercondutor de núcleos saturados* (Doctoral dissertation, UNL/FCT). Retrieved from http://hdl.handle.net/10362/8516

Wang, M., Li, Z., Hong, Z., Jin, Z., Wang, L., Zhong, L., … Li, L. (2018). Numerical study on ac loss of soldered-stacked-square(3s) hts wire with 1 mm width. In *2018 ieee international conference on applied superconductivity and electromagnetic devices (asemd)* (pp. 1–3). doi:10.1109/ASEMD.2018.8558814

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON convention record, part 4* (pp. 96–104). New York: IRE.

Xu, Y., Dai, S., Ma, T., & Wang, B. (2021). Thermal stability analysis of ybco tapes under dc overcurrent. *IEEE Transactions on Applied Superconductivity*, *31*(2), 1–6. doi:10.1109/TASC.2020.3045454

Yang, Y., Martinez, E., & Norris, W. (2004). Configuration and calibration of pickup coils for measurement of ac loss in long superconductors. *Journal of Applied Physics*, *96*, 2141–2149. doi:10.1063/1.1766100

Yazdani-Asrami, M., Taghipour-Gorjikolaie, M., Song, W., Zhang, M., & Yuan, W. (2020). Prediction of nonsinusoidal ac loss of superconducting tapes using artificial intelligence-based models. *IEEE Access*, *8*, 207287–207297. doi:10.1109/ACCESS.2020.3037685

Annex I

Coil Table

Coil name	N (turns)	Internal Diameter (mm)
CB1	40	122
CB2	20	118
CB3	60	118
CB4	50	122

Table I.1: Coils dimensions, where the second letter represents the tape type: A for S-Innovations and B for Shanghai Superconductor

Annex II

MATLAB Code for the Global ANN

```
% Import table
T = readtable('global.xlsx');

% Drop coil names
T = removevars(T,{'primary', 'secondary'});

% Find groups
T.core_material = findgroups(T.core_material);
T.core_grain = findgroups(T.core_grain);

% Normalize
T = normalize(T, 'range', [0 1]);

% Separate inputs and targets
Inputs = removevars(T, {'Qp', 'Qs'});
Targets = [T.Qp, T.Qs];
Inputs = table2array(Inputs);
Inputs = Inputs.';
Targets = Targets.';

% Transfer function
transferFunc = ["elliotsig", "logsig", "radbasn", "tansig"];

% Minimum and maximum number of hidden neurons
minH = 5;
```

```
maxH = 20;

% Number of NN to train
numNN = 20;

bestNetOne = cell(1, 3);
bestNetTwo = cell(1, 3);
bestNetThree = cell(1, 3);
best_perf = 10000;
best_perf2 = 10000;
best_perf3 = 10000;
j = 0;
global_start = tic;

for idxTF = 1:1:length(transferFunc)
    for k1 = minH:1:maxH
        net = feedforwardnet(k1);
        net.trainFcn = 'trainlm'; % Levenberg-Marquardt algorithm
        transfer1 = transferFunc(idxTF);
        net.layers{1}.transferFcn = transfer1;
        [best_perf, auxNet] = trainNetwork(net, best_perf, Inputs, ...
            Targets);
        if isa(auxNet, 'cell')
            bestNetOne = auxNet;
        end
        for idxTF2 = 1:1:length(transferFunc)
            for k2 = minH:1:maxH
                net = feedforwardnet([k1 k2]);
                net.trainFcn = 'trainlm';
                transfer2 = transferFunc(idxTF2);
                net.layers{1}.transferFcn = transfer1;
                net.layers{2}.transferFcn = transfer2;
                [best_perf2, auxNet] = trainNetwork(net, best_perf2, ...
                    Inputs, Targets);
                if isa(auxNet, 'cell')
                    bestNetTwo = auxNet;
                end
                for idxTF3 = 1:1:length(transferFunc)
                    for k3 = minH:1:maxH
                        net = feedforwardnet([k1 k2 k3]);
                        net.trainFcn = 'trainlm';
                        transfer3 = transferFunc(idxTF3);
                        net.layers{1}.transferFcn = transfer1;
                        net.layers{2}.transferFcn = transfer2;
                        net.layers{3}.transferFcn = transfer3;
                        [best_perf3, auxNet] = trainNetwork(net, ...
                            best_perf3, Inputs, Targets);
                        if isa(auxNet, 'cell')
                            bestNetThree = auxNet;
                        end
                    end
                end
            end
        end
    end
end

global_time = toc(global_start);
clear i k1;

% Get best performing NN with one layer
net = bestNetOne{1, 1};
tr = bestNetOne{1, 2};

% Get best performing NN with two layers
net2 = bestNetTwo{1, 1};
tr2 = bestNetTwo{1, 2};

% Get best performing NN with 3 layers
net3 = bestNetThree{1, 1};
tr3 = bestNetThree{1, 2};

% Save networks
save('global_net_1_layer.mat', 'net');
save('global_net_2_layers.mat', 'net2');
save('global_net_3_layers.mat', 'net3');

function [best_perf, bestNet] = trainNetwork(net, best_perf, ...
    Inputs, Targets)
    net.trainParam.showWindow = false;
    net = init(net);
    [net, tr] = train(net, Inputs, Targets);

    % Get outputs
    Outputs = net(Inputs);

    % Calculate performance
    perf = perform(net, Targets, Outputs);

    % Store best performing network
    if best_perf > perf
        best_perf = perf;
        bestNet{1, 1} = net;
        bestNet{1, 2} = tr;
        bestNet{1, 3} = perf;
    else
        bestNet = 0;
    end

end
```

Annex III

MATLAB Code for the Coil ANN

```
% Import table
T = readtable('coils.xlsx');

% Drop coil names
T = removevars(T,{'coil'});

% Separate inputs and targets
Inputs = removevars(T, {'Q'});
Targets = [T.Q];
Inputs = table2array(Inputs);
Inputs = Inputs.';
Targets = Targets.';

% Network variables
transferFunc = ["elliotsig”, "logsig”, "radbasn”, "tansig”];

% Minimum and maximum number of hidden neurons
minH = 5;
maxH = 20;

% Number of NN to train
numNN = 20;

bestNetOne = cell(1, 3);
bestNetTwo = cell(1, 3);
```

```
bestNetThree = cell(1, 3);

best_perf = 10000;
best_perf2 = 10000;
best_perf3 = 10000;
j = 0;
global_start = tic;

len_transfer_function = length(transferFunc);
for nnn = 1:1:numNN
    for idxTF = 1:1:len_transfer_function
        for k1 = minH:1:maxH
            loop_start = tic;
            net = feedforwardnet(k1);
            net.trainFcn = 'trainlm'; % Levenberg-Marquardt algorithm
            net.layers{1}.transferFcn = transferFunc(idxTF);
            [best_perf, auxNet] = trainNetwork(net, best_perf, ...
                Inputs, Targets);
            if isa(auxNet, 'cell')
                bestNetOne = auxNet;
            end
            for idxTF2 = 1:1:len_transfer_function
                for k2 = minH:1:maxH
                    net = feedforwardnet([k1 k2]);
                    net.trainFcn = 'trainlm';
                    net.layers{1}.transferFcn = transferFunc(idxTF);
                    net.layers{2}.transferFcn = transferFunc(idxTF2);
                    [best_perf2, auxNet] = trainNetwork(net, ...
                        best_perf2, Inputs, Targets);
                    if isa(auxNet, 'cell')
                        bestNetTwo = auxNet;
                    end
                    for idxTF3 = 1:1:len_transfer_function
                        for k3 = minH:1:maxH
                            net = feedforwardnet([k1 k2 k3]);
                            net.trainFcn = 'trainlm';
                            net.layers{1}.transferFcn = ...
                                transferFunc(idxTF);
                            net.layers{2}.transferFcn = ...
                                transferFunc(idxTF2);
                            net.layers{3}.transferFcn = ...
                                transferFunc(idxTF3);
                            [best_perf3, auxNet] = trainNetwork(net, ...
                                best_perf3, Inputs, Targets);
                            if isa(auxNet, 'cell')
                                bestNetThree = auxNet;
                            end
                        end
                    end
                end
            end
        end
    end
end

global_time = toc(global_start);
clear i k1;

% Get best performing NN
net = bestNetOne{1, 1};
tr = bestNetOne{1, 2};

% Get best performing NN with two layers
net2 = bestNetTwo{1, 1};
tr2 = bestNetTwo{1, 2};

% Get best performing NN with 3 layers
net3 = bestNetThree{1, 1};
tr3 = bestNetThree{1, 2};

% Save networks
save('coil_net_1_layer.mat', 'net');
save('coil_net_2_layers.mat', 'net2');
save('coil_net_3_layers.mat', 'net3');

function [best_perf, bestNet] = trainNetwork(net, best_perf, ...
    Inputs, Targets)

    net.trainParam.showWindow = false;
    net = init(net);
    [net, tr] = train(net, Inputs, Targets);

    % Get outputs
    Outputs = net(Inputs);

    % Calculate performance
    perf = perform(net, Targets, Outputs);

    % Store best performing network
    if best_perf > perf
        best_perf = perf;
        bestNet{1, 1} = net;
        bestNet{1, 2} = tr;
        bestNet{1, 3} = perf;
    else
        bestNet = 0;
    end

end
```

Annex IV

MATLAB Code for the Transformer ANN

```
% Import coil ANN
load coil_net_3_layer;
cc = net3;

transformer_info = [400;100;100;2;0;0;0.5];

transformer_table = readtable('transformer.xlsx');

input_coil_nn = removevars(transformer_table, {'primary', 'secondary', ...
    'core_height', 'core_diam', 'core_window', 'core_num_win', ...
    'core_material', 'core_grain', 'core_lamination'});

input_coil_prim = removevars(input_coil_nn, {'Ns', 'int_diam_p', 'as', ...
    'bs', 'I_Ic_s', 'Qp', 'Qs'});
input_coil_sec  = removevars(input_coil_nn, {'Np', 'int_diam_p', 'ap', ...
    'bp', 'I_Ic_p', 'Qp', 'Qs'});

input_coil_prim = table2array(input_coil_prim);
input_coil_sec = table2array(input_coil_sec);
input_coil_prim = input_coil_prim.';
input_coil_sec = input_coil_sec.';
output_coil_prim = cc(input_coil_prim);
output_coil_sec = cc(input_coil_sec);
total_coil_prim = vertcat(input_coil_prim, output_coil_prim);
total_coil_sec = vertcat(input_coil_sec, output_coil_sec);
```

```
% Order: Np Ns diam_p diam_s ap as bp bs f I_Ic_p I_Ic_s Qp Qs
total_coil = vertcat(total_coil_prim, total_coil_sec);
total_coil = total_coil([1 8 2 9 3 10 4 11 5 6 13 7 14],:);

Inputs = zeros(7, length(total_coil)) + transformer_info;
Inputs = vertcat(total_coil, Inputs);
Targets = [transformer_table.Qp transformer_table.Qs].';

% Network variables
transferFunc = ["elliotsig", "logsig", "radbasn", "tansig"];

% Minimum and maximum number of hidden neurons
minH = 20;
maxH = 22;

% Number of NN to train
numNN = 20;

bestNetOne = cell(1, 3);
bestNetTwo = cell(1, 3);
bestNetThree = cell(1, 3);

best_perf = 10000;
best_perf2 = 10000;
best_perf3 = 10000;
j = 0;
global_start = tic;

len_transfer_function = length(transferFunc);
for nnn = 1:1:numNN
    for idxTF = 1:1:len_transfer_function
        for k1 = minH:1:maxH
            net = feedforwardnet(k1);
            net.trainFcn = 'trainlm'; % Levenberg-Marquardt algorithm
            net.layers{1}.transferFcn = transferFunc(idxTF);
            [best_perf, auxNet] = trainNetwork(net, best_perf, ...
                Inputs, Targets);
            if isa(auxNet, 'cell')
                bestNetOne = auxNet;
            end
            for idxTF2 = 1:1:len_transfer_function
                for k2 = minH:1:maxH
                    net = feedforwardnet([k1 k2]);
                    net.trainFcn = 'trainlm';
                    net.layers{1}.transferFcn = transferFunc(idxTF);
                    net.layers{2}.transferFcn = transferFunc(idxTF2);
                    [best_perf2, auxNet] = trainNetwork(net, ...
                        best_perf2, Inputs, Targets);
                    if isa(auxNet, 'cell')
                        bestNetTwo = auxNet;
                    end
                    for idxTF3 = 1:1:len_transfer_function
                        net = feedforwardnet([k1 k2 k3]);
                        net.trainFcn = 'trainlm';
                        net.layers{1}.transferFcn = transferFunc(idxTF);
                        net.layers{2}.transferFcn = transferFunc(idxTF2);
                        net.layers{3}.transferFcn = transferFunc(idxTF3);
                        [best_perf, auxNet] = trainNetwork(net, ...
                            best_perf, Inputs, Targets);
                        if isa(auxNet, 'cell')
                            bestNet = auxNet;
                        end
                    end
                end
            end
        end
    end
end

global_time = toc(global_start);
clear i k1;

% Get best performing NN with 1 layer
net = bestNetOne{1, 1};
tr = bestNetOne{1, 2};

% Get best performing NN with 2 layers
net2 = bestNetTwo{1, 1};
tr2 = bestNetTwo{1, 2};
```

```
% Get best performing NN with 3 layers
net3 = bestNetThree{1, 1};
tr3 = bestNetThree{1, 2};

% Save networks
save('series_net_1_layer.mat', 'net');
save('series_net_2_layer.mat', 'net2');
save('series_net_3_layer.mat', 'net3');

function [best_perf, bestNet] = trainNetwork(net, best_perf, Inputs, Targets)
    net.trainParam.showWindow = false;

    net = init(net);
    [net, tr] = train(net, Inputs, Targets);

    % Get outputs
    Outputs = net(Inputs);

    % Calculate performance
    perf = perform(net, Targets, Outputs);

    % Store best performing network
    if best_perf > perf
        best_perf = perf;
        bestNet{1, 1} = net;
        bestNet{1, 2} = tr;
        bestNet{1, 3} = perf;
    else
        bestNet = 0;
    end
end
```

2022 A DATA-DRIVEN METHODOLOGY FOR MODELLING LOSSES IN HTS POWER SYSTEMS Henrique Simas

NOVA SCHOOL OF SCIENCE & TECHNOLOGY