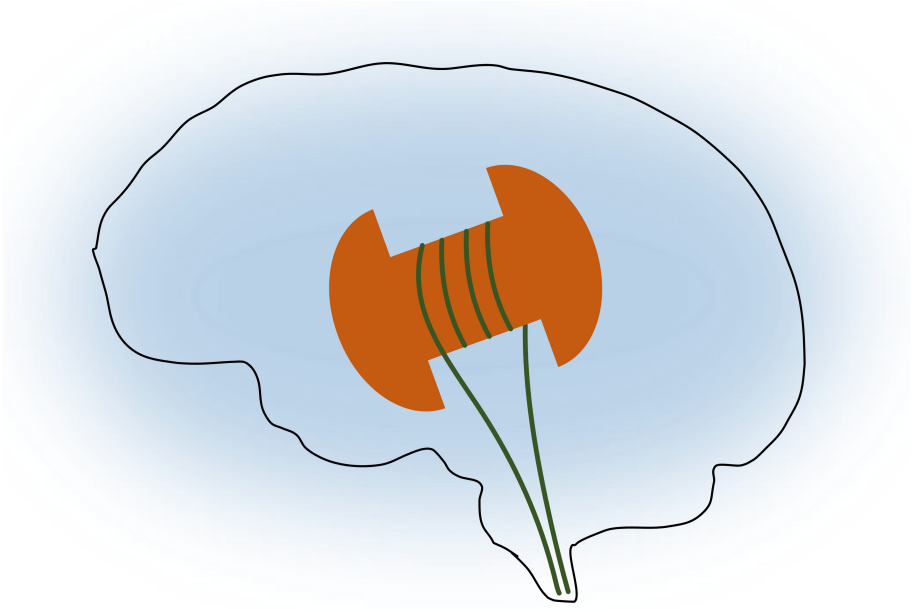
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Thomas Grong

Adaptive Neural Network-based PSS Designs for Modern Power Systems

Master's thesis in Electric Power Engineering
Supervisor: Jonas Kristiansen Nøland
June 2020



NTNU
Norwegian University of Science and Technology

Thomas Grong

Adaptive Neural Network-based PSS Designs for Modern Power Systems

Master's thesis in Electric Power Engineering
Supervisor: Jonas Kristiansen Nøland
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

NTNU
Norwegian University of
Science and Technology

Summary

As global power systems transition to a larger share of renewable energy, changes in their operating characteristics become increasingly prominent. Sources such as photovoltaic panels and wind turbines trigger more frequent instability events and do not contribute in stabilising the system. Thus, the performance of classical bulk power generation facilities become increasingly important for robust operation of power systems with high renewable penetration. Historically, little effort is put into the tuning of these generating units, as they commonly are set with typical values from literature and briefly tested for positive damping. This was an adequate methodology in the classical power system, but as modern power systems are run much closer to their stability limits, optimising the equipment becomes essential.

The excitation system is the main actuator of the synchronous generator and has a large impact on its stability performance. It consists of the exciter, automatic voltage regulator (AVR) and power system stabiliser (PSS). A well-tuned excitation system provides benefits such as improved oscillation damping, relay coordination and first-swing transient stability. However, traditional tuning occurs with the generator out of operation, which results in huge financial losses to the owner and makes subsequent re-tuning unlikely.

The objective of this thesis is to explore approaches to make the PSS design more adaptive and versatile by applying neural networks (NN). Two approaches are presented. Firstly, an NN-based auto-tuning system for the conventional PSS (CPSS) design is proposed, where the NN is trained using optimised data from the particle swarm optimisation (PSO) technique. The PSO obtains optimal CPSS parameters from a simplified linear model of the synchronous machine. Secondly, a NN controller to act as the PSS is proposed, where its phase response is a control variable. The controller is named the sine shifting neural network (SSNN). The SSNN controller is unique in that it does not rely on any electrical machine theory in its creation. Finally, the two approaches are compared to a static CPSS and a no PSS approach by time-domain simulations using a more accurate flux-linkage model of the synchronous machine. The disturbances performed in the simulations are steps in the external network reactance. All simulations are performed in the MATLAB/Simulink environment.

The time-domain simulations of the rotor speed deviation show that the SSNN provides superior oscillation damping compared to the other approaches. It was able to reduce the settling time to well under 1 second for all tests, where the other approaches were in the 2-4 s range. The creation and implementation of the CPSS auto-tuning system were successful, yet it did not give a consistent improvement in damping compared to the static CPSS.

The work in this thesis shows that applying neural networks to the PSS design has great potential in improving its performance and to make it adaptive and versatile. The neural network is a powerful tool that can aid in the global energy transition to maintain the robustness that is expected of the power system.

Sammendrag

Når andelen av fornybar energi i et kraftsystem øker, endres systemets egenskaper på en merkbar måte. Fornybare energikilder som solcelleanlegg og vindturbiner skaper mer hyppige forstyrrelser, men bidrar ikke til å stabilisere nettet. Det betyr at ytelsen til klassiske kraftverk med roterende masser er svært viktig i kraftsystemer med en høy andel fornybare energikilder. Til tross for dette har det historisk sett blitt lagt lite innsats i innstillingene av disse generatorene. De er normalt innstilt med typiske verdier fra faglitteratur og testes enkelt for positiv demping av svingninger. I klassiske kraftsystemer var dette en tilfredsstillende fremgangsmåte, men siden moderne systemer kjøres nærmere operasjonsgrensene og opplever mer regelmessige forstyrrelser, blir optimering av det eksisterende utstyret essensielt.

Magnetiseringssystemet er synkrongeneratorens viktigste styringssystem, og har stor innvirkning på maskinens stabilitet. Styringssystemet består hovedsakelig av en spenningsregulator (AVR) og dempetilsats (PSS). Et godt innstilt styringssystem kan gi fordeler slik som forbedret demping av svingninger, enklere koordinering av vern og høyere transient stabilitet i første svingning. Innstillingen av systemet har tradisjonelt blitt gjort med generatoren frakoblet nettet, noe som kan forårsake store kostnader for eieren av kraftverket og gjør korrigering av innstillingene på et senere tidspunkt uaktuelt.

Målet med denne avhandlingen er å utforske måter nevrale nettverk (NN) kan benyttes for å gjøre PSS-designet mer tilpasningsdyktig og fleksibelt. To fremgangsmåter er presentert. Den første er et system for automatisk innstilling av det konvensjonelle PSS-designet (CPSS), basert på en NN-innstiller. Nettverket er trent på optimerte parametere fra "particle swarm optimisation"-teknikken (PSO). PSO finner de optimale parameterne fra en forenklet lineær modell av synkronmaskinen. Den andre fremgangsmåten er å benytte en NN-basert regulator som en PSS der faseresponsen gis som en kontrollvariabel. Regulatoren er navngitt "sine shifting neural network" (SSNN). Denne regulatoren er unik i den forstand at det kreves ingen elektrisk maskinteori for å designe den. Til slutt ble begge disse fremgangsmåtene sammenlignet med en statisk PSS og ved bruk av ingen PSS. Sammenligningene er gjort ved hjelp av en mer nøyaktig maskinmodell i tids-domenet. Forstyrrelsen som er simulert er et steg i den eksterne nettreaktansen. Alle simuleringer er utført i MATLAB/Simulink.

Simuleringene viste at SSNN-regulatoren ga overlegen dempeevne sammenlignet med de andre fremgangsmåtene. Den var i stand til å redusere stabiliseringstiden til under 1 sekund, mens de andre metodene endte på 2-4 sekunder. Opprettelsen og implementeringen av den automatiske NN-innstilleren for CPSS-designet var vellykket, men simuleringene viste at den ikke ga konsekvent bedre demping enn den statiske CPSS-en.

Resultatet i denne avhandlingen viser at å benytte nevrale nettverk i designet av PSS-en har et stort potensial for å øke dempeevnen, samt å gjøre den mer tilpasningsdyktig og fleksibel. Et nevralt nettverk et er kraftig verktøy som kan bidra positivt i overgangen til fornybar kraftproduksjon for å sikre påliteligheten som er forventet av kraftsystemet.

Preface

This thesis is the final submission at the Norwegian University of Science and Technology for the fulfilment of the degree of Master of Science in electric power engineering. It marks the finalisation of the 2-year Electric Power Engineering master program.

I wish to thank my supervisor, Associate Professor Jonas Kristiansen Nøland, for guiding me through the project and for constantly reminding me not to overcomplicate every problem. A more dedicated supervisor, I could not hope for.

This thesis is the follow-up to a specialisation project the preceding semester. Some content has been adapted from that report, and when this occurs the report is directly cited. The reader is expected to have some familiarity with electrical machine analysis and control theory.

Thomas Grong

June 13, 2020

Table of Contents

List of Figures

List of Tables

Abbreviations

AVR	=	Automatic voltage regulator
CDI	=	Comprehensive damping index
CPSS	=	Conventional power system stabiliser
emf	=	Electromotive force
GPU	=	Graphics processing unit
IEEE	=	Institute of Electrical and Electronics Engineers
LM	=	Levenberg-Marquardt (training algorithm)
LSTM	=	Long short-term memory
MLP	=	Multi-layer perceptron (neural network)
MSE	=	Mean square error
NN	=	Neural network
ODE	=	Ordinary differential equation
OEL	=	Overexcitation limiter
PF	=	Power factor
PI	=	Proportional-integral (controller)
PSO	=	Particle swarm optimisation
PSS	=	Power system stabiliser
pu	=	per-unit
ReLU	=	Rectified linear unit (activation function)
RMS	=	Root-Mean-Square
RNN	=	Recurrent neural network
SCL	=	Stator current limiter
SCR	=	Silicon-controlled rectifier
SL	=	Supervised learning
SLNN	=	Supervised learning neural network
SSNN	=	Sine shifting neural network
TSO	=	Transmission system operator
UEL	=	Underexcitation limiter

Chapter 1

Introduction

1.1 Background

The world of electric power production is changing rapidly. The current trend shows a steadily increasing share of renewable energy sources, and this trend is not likely to change. Power sources like wind turbines and photovoltaic panels are being implemented for both large-scale power plants and small-scale distributed generation. Due to the intermittent nature of weather conditions, the power production from these sources is unreliable. Moreover, they are interfaced with the power system through power electronics, giving little to no contribution to the system inertia. For this reason, a high renewable penetration is associated with more frequent instability events while contributing little to the stabilising ability of the system. It is then up to the generating units with rotating masses to manage these additional instability events in addition to keeping power reserves and handling increasingly frequent changes in load conditions. Consequently, these generators must be able to provide a more continuous regulation in their production than ever before.

The strain on the power system is only increasing as time progresses, due to increased load and generation. Yet, the expansion of the grid is restricted by environmental and economical factors. Thus, the power system is consistently being operated closer to its limits, making optimisation of the existing equipment an essential part of the modern power system.

A disturbance in the interconnected power system might originate from a fault, load change, component disconnection, motor starting or any such occurrence that impacts the load flow. A power system may be vast and a single machine's response to a disturbance depends on its electrical distance to the disturbance. A sufficiently close disturbance will pull the machine from steady-state and quantities such as voltage, frequency and load angle will deviate and oscillate. The ability to recover and return to steady-state relies heavily on the control systems of the machine. For a synchronous machine, the most important control system is the excitation system, which controls the field voltage to regulate the machine excitation. The machine excitation has a large impact on quantities such as terminal voltage, reactive power flow and electromechanical torque.

The power system stabiliser (PSS) has become very widespread since its inception in the 1960s, as it has proven to be proficient at damping oscillatory behaviour in the machine rotor. However, even though it has gained in popularity, its general design has barely improved. Historically, little effort is put into tuning the PSS before commissioning. Commonly, it is tuned using

typical values from the literature to just satisfy the performance requirements of the transmission system operator (TSO). This means there is much room for optimisation in the current system.

1.2 Motivation

The Norwegian power production is dominated by hydropower [2], in which salient-pole wound-field synchronous machines are the main working force. The use of properly tuned excitation systems is therefore essential for optimal operation of the power system. Traditionally, excitation systems are tuned before they are commissioned and rarely retuned afterwards since retuning would require to put the machine out of operation for the duration. This is associated with a very high cost as the machine is not producing power. Moreover, quickly tuning an excitation system manually is challenging and requires high expertise. Therefore, providing some automated procedures for determining excitation system parameters would be very beneficial. Well tuned systems provide advantages such as improved dynamic oscillation damping, easier relay coordination and improved first-swing transient stability [3].

1.3 Objective

This thesis aims to answer the following questions:

1. How can a simplified synchronous machine model be derived for the use in machine learning applications, and how can a more accurate model be derived for time-domain performance assessment?
2. How can a neural network be applied to improve the conventional power system stabiliser design?
3. How can a neural network-based PSS be created to improve stability without relying on complex machine theory?
4. How can the proposed system be implemented into the MATLAB/Simulink environment for testing purposes?

1.4 Related research

The power system stabiliser (PSS) is a topic that has been around for decades, with the most common design being a transfer function with a lead/lag structure introduced in 1969 [4]. The inherent weakness of this design is that it made to work adequately for a great range of scenarios, making it sub-optimal in any one scenario. Consequently, much research has gone into improving its design. Several intelligent algorithms have been applied to tuning the conventional PSS (CPSS) for improving its stability performance. Some examples of algorithms that have been applied for PSS tuning are: Particle swarm optimisation [5, 6]; Genetic algorithm [7, 8]; Fuzzy logic systems [9, 10]; Artificial bee colony [11, 12]; Bat search algorithm [13, 14]; Bacterial foraging optimisation [15, 16]; Differential evolution [17, 18]; Tabu search algorithm [19, 20]; Simulated annealing [21] and several more. The disadvantage with most of these is that they are focused on the optimal tuning of the CPSS before commissioning, and not adaptively adjusting

it to changing conditions. Therefore, neural network-based PSS designs were explored, which are well fitted for adaptive solutions.

The idea of neural networks is approaching a century old. The first model of neuron activity was presented in 1943 [22], where a simple neural network was modelled with electrical circuits. In 1958, the first artificial neural network, the perceptron, was introduced [23] and is a design still actively used today. It was in the 1990s the exploration of NN-based PSSs properly began.

Neural networks (NN) have been applied to the PSS problem to varying degrees of complexity. In the early stages, the NN was used to automatically tune the parameters of the conventional lead/lag type PSS [24]. Also, some proposals were made replacing the conventional PSS with NNs altogether [25, 26, 27, 28]. These networks were trained before commissioning and were not updated dynamically. Some more complex NN structures applied in some papers [29, 30, 31, 32] use variations on a two-network system, where one NN acts as a neuro-identifier, emulating the machine behaviour for next-step prediction, and the other as the neuro-controller, which is dynamically trained by the neuro-identifier to adapt to changing conditions. This two-network structure is based on the work in [33]. Later, some adaptations and variations have been proposed in the 2000s. A review of several of these, along with other intelligent algorithms in PSS applications, was performed in [34].

A common factor among these is that their approaches apply a mathematical model of the machine to train their neural networks. However, to apply such models correctly, high competence is needed along with detailed information of the machine and connected system. Thus, it could be beneficial to have an approach where such models are not necessary, where simpler principles are applied to design a PSS able to improve the damping capability of the system.

1.5 Approach

1.5.1 Scope and assumptions

In such a theoretically heavy field of study, there are countless levels of detail one could go into. However, to keep the topic of the thesis focused and to satisfy time constraints some restrictions and assumptions are made:

- This thesis will focus on the PSS while letting the AVR parameters remain constant.
- Stability studies in this thesis will keep to small-signal stability. The purpose of the PSS is to damp oscillatory behaviour over several swings. In a large-signal stability study, the main concern would be the first-swing transient stability, where the PSS has little impact compared to the AVR gain.
- The effect of the governor system will not be considered. Consequently, the mechanical torque to the machine is considered constant.
- The standard parameters of the machine are assumed known and constant. Generators commonly have data sheets where these are listed. Additionally, high-quality sources describing the determination of these parameters already exist [35, 36].
- Saturation effects of the machine will not be considered. During small-signal disturbances the deviations are usually small enough that saturation does not occur to any major degree,

making this a fair simplification.

1.5.2 Methodology

The following approach is used to work towards the objectives of this thesis:

- Review the mathematical modelling of the synchronous machine, for a detailed understanding of the machine dynamic behaviour.
- Derive a simplified linear machine model for small-signal stability studies
- Introduce the concept of intelligent algorithms and give insight into neural networks
- Propose a way to apply neural networks to automatically tune the conventional PSS
- Propose a new, model-free approach to the PSS design using neural networks
- Evaluate the proposed designs and compare to traditional methods

The information used in this thesis is gathered from textbooks, the University's online library of articles and literature, talking to professionals at the University and through the author's personal experience.

1.6 Structure of the thesis

This thesis is constructed with six chapters:

Chapter 1 gives the introduction of the thesis problem. It highlights the background and objectives, provides insight into some previous research done with the problem and presents the layout of the thesis.

Chapter 2 goes into detail of the derivation of the synchronous machine mathematical model and introduces the excitation system. It subsequently derives a simplified linear model of the machine connected to an infinite bus system and its state-space representation. Lastly, it gives some insight into how the responses of small-signal disturbances might be quantified.

Chapter 3 introduces the concept of intelligent learning before going into detail of the two intelligent systems used in this thesis: the particle swarm optimisation (PSO) algorithm and neural networks (NN).

Chapter 4 proposes an NN auto-tuning system for the conventional power system stabiliser (CPSS) for tuning during online operation and shows how such an auto-tuner can be implemented into the MATLAB/Simulink environment.

Chapter 5 proposes a novel NN-type controller where the phase response is a control variable. Its structure and training data creation is given and its performance is tested and discussed. Next, the NN is applied as a PSS and its damping performance compared to three other PSS philosophies is shown.

Chapter 6 discusses the procedures and results in this thesis. It gives the concluding remarks and provides suggestions for further work.

Appendix A provides the numeric values of model parameters used in the simulations during the work of this thesis that are not provided during the text.

Appendix B gives a brief guide on how a simple neural network can be created in MATLAB.

Chapter 2

Synchronous machine representation

> *"This vast enterprise of supplying electrical energy presents many engineering problems that provide the engineer with a variety of challenges. The planning, construction, and operation of such systems become exceedingly complex. Some of the problems stimulate the engineer's managerial talents; others tax his knowledge and experience in system design. The entire design must be predicated on automatic control and not on the slow response of human operators. To be able to predict the performance of such complex systems, the engineer is forced to seek ever more powerful tools of analysis and synthesis."*
>
> – Anderson and Fouad [37, p. 3]

In the studies of PSS tuning presented in Section 1.4 the performance is commonly tested through time-domain simulations. However, it is not always clear what machine model is used for the simulations, and some even use the simplified linear model for the time-domain simulations. To ensure the performance assessments are kept as realistic as possible, the tests should employ a more accurate model of the machine, even though the tuning methods can be built on simpler models. Simulink and similar software have pre-built models readily available, and for this reason, they easily become "black boxes", where there is little understanding of the inner dynamics. This chapter aims to alleviate this and to show the models that will be used for performance assessment in the later chapters.

There are several ways of modelling a synchronous machine to various degrees of complexity. Which model is appropriate depends on each study. In a large-scale study with several machines being simulated, highly detailed models might prove to be too computationally intensive. Additionally, machines far away from the point of interest will have less impact on the behaviour at that point, such that less complex models will be sufficient. [36]

In this thesis, a single machine is being modelled. Only the behaviour of this machine and the impact of its excitation system is of any real interest. However, to create a more realistic model, it needs to be connected to a network. A simple way to represent an external network is to consider it an infinite bus, which is a constant voltage source behind an impedance. This is not a highly accurate description of a power system, as the voltage and frequency realistically vary. Still, it is a sufficient representation for this type of study.

2.1 Derivation of the mathematical model

The development of the mathematical model in this chapter largely follows the procedures in Chapter 11 of [36] and Chapter 4 of [37].

To limit the degree of complexity in the modelling process, some underlying assumptions are made: [36]

- All winding capacitances are neglected, as they are small compared to their inductances
- The three-phase stator windings are identical, symmetrically distributed and wye-connected
- Each distributed winding may be represented as a concentrated winding
- The time variations in the stator inductances are sinusoidal and does not contain higher harmonics due to stator slots and other effects. Hence, the inductances can be represented by a constant term added to a single periodic term.
- All hysteresis losses are negligible
- Saturation in the machine is negligible.

2.1.1 Notation

When researching the topic of electrical machine modelling, one quickly realises that differences in notation between authors can become confusing. Therefore, an overview of the important notation used in this chapter is given here:

- A quantity given a superbar represents a vector or matrix quantity, e.g. $\overline{\Psi}$.
- Dotted variables are time derivatives, where the derivative order is equal to the number of dots, e.g. $\dot{\delta}$ or $\ddot{\delta}$.
- Subscript *d* and *q* denote quantities associated with the d- and q-axes respectively, e.g. V_d or X_q.
- Subscripts *D* and *Q* denote quantities associated with the d- and q-axis damper/amortisseur windings respectively, e.g. L_D or i_Q. This thesis will henceforth use the term *damper winding*.
- Subscript lower-case f denotes a field quantity, e.g. V_f.
- Subscript upper-case *R* denotes rated values, e.g. ω_R.
- Subscript lower-case *l* denotes a leakage quantity, e.g. X_l.
- Where there might be ambiguities, per-unit quantities are given the subscript *,pu*, e.g. $T_{m,pu}$

2.1.2 Flux linkage equations

Electrical machines are predominantly built with three phases to fit the power system. The phase windings are shifted $120°$ from each other and are referred to as phases A, B and C. Since these

windings are mounted on the stator, they are called the stator windings. In a salient-pole wound-field synchronous machine, which is the most common type in hydropower plants and the type considered here, there are some windings mounted on the rotor as well. Most importantly, the field winding is wound around the poles of the machine. Additionally, there are damper windings (sometimes also called amortisseur windings) mounted at the edge of the poles, near the air gap. These provide damping torque when the rotor angle deviates from its steady-state position. All these windings have self- and mutual inductances, whose flux linkages describe the machine behaviour. Figure 2.1 shows a simple illustration of how the windings and axes are positioned in relation to each other.

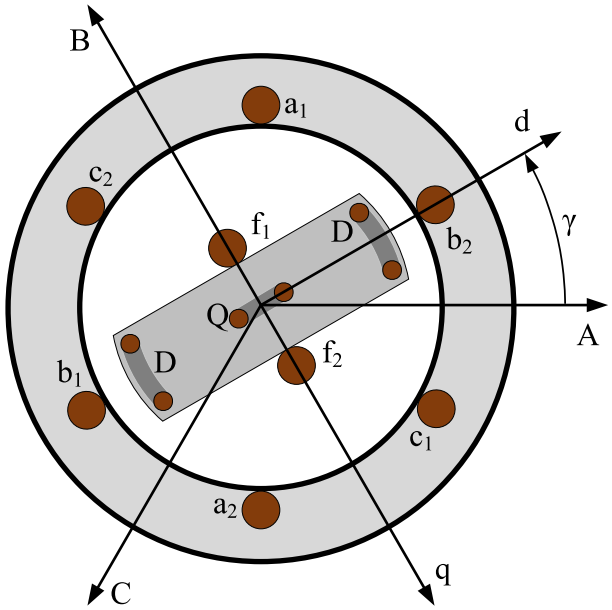


Figure 2.1: Cross-sectional schematic of the synchronous machine windings displaying the ABC and dq reference frames

There are six windings to consider: Phases A, B and C, the field winding and the damper windings in the d- and q-axes. The full expression for the machine flux linkages is:

$$\begin{bmatrix} \Psi_A \\ \Psi_B \\ \Psi_C \\ \hdashline \Psi_f \\ \Psi_D \\ \Psi_Q \end{bmatrix} = \left[\begin{array}{ccc:ccc} L_{AA} & L_{AB} & L_{AC} & L_{Af} & L_{AD} & L_{AQ} \\ L_{BA} & L_{BB} & L_{BC} & L_{Bf} & L_{BD} & L_{BQ} \\ L_{CA} & L_{CB} & L_{CC} & L_{Cf} & L_{CD} & L_{CQ} \\ \hdashline L_{fA} & L_{fB} & L_{fC} & L_{ff} & L_{fD} & L_{fQ} \\ L_{DA} & L_{DB} & L_{DC} & L_{Df} & L_{DD} & L_{DQ} \\ L_{QA} & L_{QB} & L_{QC} & L_{Qf} & L_{QD} & L_{QQ} \end{array}\right] \begin{bmatrix} i_A \\ i_B \\ i_C \\ \hdashline i_f \\ i_D \\ i_Q \end{bmatrix} \tag{2.1}$$

or, in compact form:

$$\begin{bmatrix} \overline{\Psi}_{ABC} \\ \overline{\Psi}_{fDQ} \end{bmatrix} = \begin{bmatrix} \overline{L}_{SS} & \overline{L}_{SR} \\ \overline{L}_{RS} & \overline{L}_{RR} \end{bmatrix} \begin{bmatrix} \overline{i}_{ABC} \\ \overline{i}_{fDQ} \end{bmatrix} \tag{2.2}$$

where:

- $\overline{L}_{SS}$ = Stator to stator inductances
- $\overline{L}_{SR}, \overline{L}_{RS}$ = Stator-rotor inductances
- $\overline{L}_{RR}$ = Rotor to rotor inductances

The diagonal elements in the full matrix contain self-inductances and the off-diagonals contain mutual inductances. Additionally, $L_{ij} = L_{ji}$. As mentioned earlier, the time varying inductances can be represented as a constant term added to a periodic term. However, the phase of the periodic term is not arbitrary. By inspecting Figure 2.1 it is possible to determine at which rotor angle the reluctance path for a certain inductance is minimum. At this point the inductance value will be maximum.

2.1.2.1 Stator inductances

For the stator self-inductances the reluctance path is minimum when the d-axis aligns with the stator phase's magnetic axis. Thus. they can be denoted as:

$$\begin{aligned} L_{AA} &= L_s + L_m\, cos\big(2\gamma\big) \\ L_{BB} &= L_s + L_m\, cos\big(2\gamma - \frac{2\,\pi}{3}\big) \\ L_{CC} &= L_s + L_m\, cos\big(2\gamma + \frac{2\,\pi}{3}\big) \end{aligned} \tag{2.3}$$

where L_s and L_m have constant values. Also, $L_s > L_m$.

The reluctance paths for the stator mutual inductances are minimum when the d-axis is midway between two of the stator winding axes. The mutual inductances will also have negative signs, since the stator windings are distributed $120°$ in space. Also, take note of the sign in the phase shifts, as they are based upon the phase order of Figure 2.1. The stator mutual inductances are:

$$\begin{aligned} L_{AB} = L_{BA} &= -M_s - L_m\, cos\big(2\gamma + \frac{\pi}{3}\big) \\ L_{BC} = L_{CB} &= -M_s - L_m\, cos\big(2\gamma - \pi\big) \\ L_{CA} = L_{AC} &= -M_s - L_m\, cos\big(2\gamma + \frac{5\,\pi}{3}\big) \end{aligned} \tag{2.4}$$

2.1.2.2 Rotor inductances

The rotor self-inductances are already in the rotor frame of reference and are therefore not time varying,

$$L_{ff} = L_f \qquad L_{DD} = L_D \qquad L_{QQ} = L_Q \tag{2.5}$$

As the damper windings in the d- and q-axes are orthogonal, there is no magnetic coupling between them. The same applies to the field winding and the q-axis damper winding. Thus, their mutual inductances are zero. The mutual inductance between the field winding and the d-axis damper winding is non-zero and does not vary with time. Thus:

$$L_{fD} = L_{Df} = M_d \qquad L_{fQ} = L_{Qf} = 0 \qquad L_{DQ} = L_{QD} = 0 \tag{2.6}$$

2.1.2.3 Stator-rotor inductances

The reluctance path between a stator and rotor winding is minimum when their axes align. The mutual inductance will be maximum when both windings have the same positive flux direction. The stator-field mutual inductances are:

$$\begin{aligned} L_{Af} &= L_{fA} = M_f \, cos(\gamma) \\ L_{Bf} &= L_{fB} = M_f \, cos\big(\gamma - \frac{2\,\pi}{3}\big) \\ L_{Cf} &= L_{fC} = M_f \, cos\big(\gamma + \frac{2\,\pi}{3}\big) \end{aligned} \tag{2.7}$$

Mutual inductances from stator to d-axis damper winding are:

$$\begin{aligned} L_{AD} &= L_{DA} = M_D \, cos(\gamma) \\ L_{BD} &= L_{DB} = M_D \, cos\big(\gamma - \frac{2\,\pi}{3}\big) \\ L_{CD} &= L_{DC} = M_D \, cos\big(\gamma + \frac{2\,\pi}{3}\big) \end{aligned} \tag{2.8}$$

Finally, the mutual inductances from the stator to q-axis damper winding are:

$$\begin{aligned} L_{AQ} &= L_{QA} = M_Q \, cos(\gamma) \\ L_{BQ} &= L_{QB} = M_Q \, cos\big(\gamma - \frac{2\,\pi}{3}\big) \\ L_{CQ} &= L_{QC} = M_Q \, cos\big(\gamma + \frac{2\,\pi}{3}\big) \end{aligned} \tag{2.9}$$

2.1.3 The dq0 reference frame

When analysing the machine and power system behaviours, doing it in the ABC reference frame can become very complex, since most inductances in Equation 2.1 contain time-varying components. It is possible to alleviate this issue by doing a transformation on the stator quantities. The transformation involves projecting the three phases onto two new axes that rotate along with the rotor: The direct axis (or d-axis), which is aligned with the protruding pole of the machine, and the quadrature axis (or q-axis), which is perpendicular to the d-axis. To uniquely define the d-axis a fixed reference must be chosen. The common choice of reference is the magnetic axis of phase A. The d-axis (and the rotor) then leads the reference by the angle γ. Since the dq-axes are rotating with the rotor, the angle γ is continuously increasing at the rate of the rotational speed ω_r [36]. Figure 2.1 has the d- and q-axes drawn in along with the ABC-axes.

The described transformation projects three axes onto two. Consequently, some information is lost. A third axis is required, which is commonly the zero-sequence axis, as defined by symmetrical component theory [38]. It lies orthogonal to both the d- and q-axes, conveniently making them magnetically uncoupled. The ABC to dq0 transformation is commonly called a Park transformation, in honour of Robert H. Park [39]. The Park transformation is defined by

Equation 2.10.

$$\begin{bmatrix} i_d \\ \hdashline i_q \\ \hdashline i_0 \end{bmatrix} = \left[\begin{array}{c:c:c} k_d\, cos(\gamma) & k_d\, cos(\gamma - \frac{2}{3}\pi) & k_d\, cos(\gamma - \frac{4}{3}\pi) \\ \hdashline k_q\, sin(\gamma) & k_q\, sin(\gamma - \frac{2}{3}\pi) & k_q\, sin(\gamma - \frac{4}{3}\pi) \\ \hdashline k_0 & k_0 & k_0 \end{array}\right] \begin{bmatrix} i_A \\ \hdashline i_B \\ \hdashline i_C \end{bmatrix} \tag{2.10}$$

or, in compact form:

$$\bar{i}_{dq0} = \overline{P} \cdot \bar{i}_{ABC} \tag{2.11}$$

Similar expressions can be made for voltages and flux linkages:

$$\overline{v}_{dq0} = \overline{P} \cdot \overline{v}_{ABC} \qquad \overline{\Psi}_{dq0} = \overline{P} \cdot \overline{\Psi}_{ABC} \tag{2.12}$$

The coefficients k_d, k_q and k_0 are arbitrary scaling factors from the transformation. Even though they are arbitrary, there are some beneficial choices for them. A common choice is $k_d = k_q = 2/3$. This ensures the peak stator currents in the dq-axes are the same as the amplitude of the ABC-frame stator currents [35]. However, when defining a per-unit base it is beneficial to have a power invariant dq0 transformation, such that the base power for all windings are equal. For a power invariant transformation, the coefficients are chosen as $k_d = k_q = k = \sqrt{2/3}$. The coefficient k_0 is chosen based upon the definition of the zero sequence current:

$$i_0 = k_0\big(i_A + i_B + i_C\big) \tag{2.13}$$

Thus, to keep the power invariance, $k_0 = 1/\sqrt{3}$. It should be mentioned that this choice of transformation has some drawback in that there will not be a comparable relationship between the dq- and ABC-quantities, as it would for the first choice [35]. Moreover, the original derivation by Park did not use a power invariant transformation [39].

The final transformation matrix becomes:

$$\overline{P} = \sqrt{\frac{2}{3}} \left[\begin{array}{c:c:c} cos(\gamma) & cos(\gamma - \frac{2}{3}\pi) & cos(\gamma - \frac{4}{3}\pi) \\ \hdashline sin(\gamma) & sin(\gamma - \frac{2}{3}\pi) & sin(\gamma - \frac{4}{3}\pi) \\ \hdashline \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{array}\right] \tag{2.14}$$

A special property of this particular matrix is that it is orthogonal, i.e. $\overline{P}^{-1} = \overline{P}^T$. This gives it its power invariant nature, resulting in:

$$\begin{aligned} p &= v_A i_A + v_B i_B + v_C i_C \\ &= v_d i_d + v_q i_q + v_0 i_0 \end{aligned} \tag{2.15}$$

Since the fDQ-variables already are in the rotor frame, only the ABC-variables need to be transformed. The transformation is made by premultiplying Equation 2.2 by

$$\begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix}$$

where $\overline{P}$ is the Park transformation matrix as defined in Equation 2.14 and $\bar{I}_3$ is the 3x3 identity matrix [37]. The transformation becomes:

$$\begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \overline{\Psi}_{ABC} \\ \overline{\Psi}_{fDQ} \end{bmatrix} = \begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \overline{L}_{SS} & \overline{L}_{SR} \\ \overline{L}_{RS} & \overline{L}_{RR} \end{bmatrix} \begin{bmatrix} \overline{P}^{-1} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \bar{i}_{ABC} \\ \bar{i}_{fDQ} \end{bmatrix} \tag{2.16}$$

which equals:

$$\begin{bmatrix} \overline{\Psi}_{dq0} \\ \overline{\Psi}_{fDQ} \end{bmatrix} = \begin{bmatrix} \overline{P}\overline{L}_{SS}\overline{P}^{-1} & \overline{P}\overline{L}_{SR} \\ \overline{L}_{RS}\overline{P}^{-1} & \overline{L}_{RR} \end{bmatrix} \begin{bmatrix} \bar{i}_{dq0} \\ \bar{i}_{fDQ} \end{bmatrix} \tag{2.17}$$

Performing the matrix multiplication and writing the result in expanded form, the flux linkage equations in the rotor reference frame become:

$$\left[\begin{array}{c} \Psi_d \\ \Psi_q \\ \Psi_0 \\ \hdashline \Psi_f \\ \Psi_D \\ \Psi_Q \end{array}\right] = \left[\begin{array}{ccc:ccc} L_d & 0 & 0 & kM_f & kM_D & 0 \\ 0 & L_q & 0 & 0 & 0 & kM_Q \\ 0 & 0 & L_0 & 0 & 0 & 0 \\ \hdashline kM_f & 0 & 0 & L_f & M_d & 0 \\ kM_D & 0 & 0 & M_d & L_D & 0 \\ 0 & kM_Q & 0 & 0 & 0 & L_Q \end{array}\right] \left[\begin{array}{c} i_d \\ i_q \\ i_0 \\ \hdashline i_f \\ i_D \\ i_Q \end{array}\right] \tag{2.18}$$

where $k = \sqrt{3/2}$ and the newly defined inductances are constants:

- $L_d = L_s + M_s + (3/2)L_m$
- $L_q = L_s + M_s - (3/2)L_m$
- $L_0 = L_s - 2M_s$

The inductance matrix now only contain constants, i.e. no time dependency, which is the great benefit of the Park transformation.

2.1.4 Voltage equations

With all flux linkages defined in the rotor reference frame in the previous section, it is now possible to define expressions for the voltages in the machine.

Consider the schematic diagram in Figure 2.2. This schematic represents the wye-connected three-phase machine. Additionally, the three rotor windings are illustrated as equivalent circuits. Only the field winding has a voltage source, as the two damper windings are short-circuited. Their voltages are both zero, $v_D = v_Q = 0$. The six windings are magnetically coupled, so their respective voltage expressions must contain flux linkage components. The general voltage expression would become:

$$v = -R \cdot i - \dot{\Psi} \tag{2.19}$$

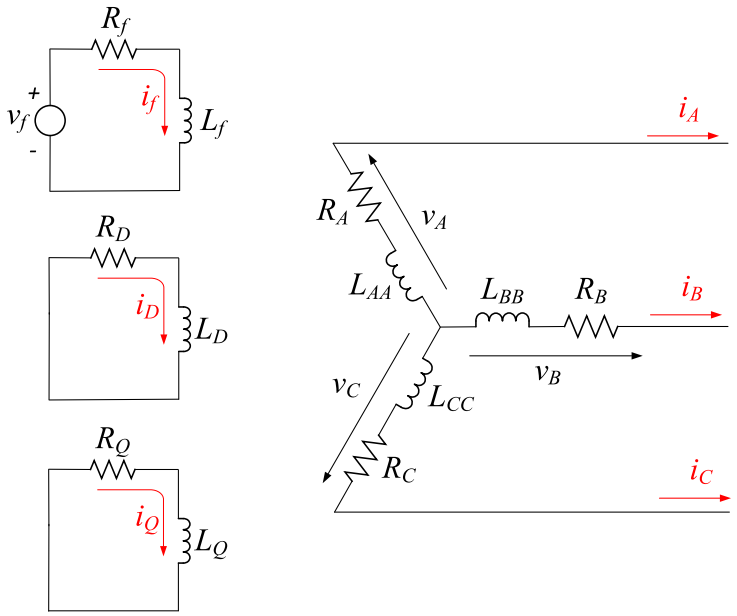


Figure 2.2: Schematic of a wye-connected machine, including the three-phase stator windings and the equivalent rotor windings: the field circuit and the d- and q-axis damper windings.

Applying Kirchhoff's voltage law on Figure 2.2 gives the expression:

$$\left[\begin{array}{c} v_A \\ v_B \\ v_C \\ \hdashline -v_f \\ 0 \\ 0 \end{array}\right] = -\left[\begin{array}{ccc:ccc} R_A & 0 & 0 & 0 & 0 & 0 \\ 0 & R_B & 0 & 0 & 0 & 0 \\ 0 & 0 & R_C & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & R_f & 0 & 0 \\ 0 & 0 & 0 & 0 & R_D & 0 \\ 0 & 0 & 0 & 0 & 0 & R_Q \end{array}\right] \left[\begin{array}{c} i_A \\ i_B \\ i_C \\ \hdashline i_f \\ i_D \\ i_Q \end{array}\right] - \left[\begin{array}{c} \dot{\Psi}_A \\ \dot{\Psi}_B \\ \dot{\Psi}_C \\ \hdashline \dot{\Psi}_f \\ \dot{\Psi}_D \\ \dot{\Psi}_Q \end{array}\right] \tag{2.20}$$

or, in compact form:

$$\left[\begin{array}{c} \bar{v}_{ABC} \\ \hdashline \bar{v}_{fDQ} \end{array}\right] = -\left[\begin{array}{c:c} \overline{R}_{ABC} & \bar{0} \\ \hdashline \bar{0} & \overline{R}_{fDQ} \end{array}\right] \left[\begin{array}{c} \bar{i}_{ABC} \\ \hdashline \bar{i}_{fDQ} \end{array}\right] - \left[\begin{array}{c} \dot{\bar{\Psi}}_{ABC} \\ \hdashline \dot{\bar{\Psi}}_{fDQ} \end{array}\right] \tag{2.21}$$

It is again beneficial to transform the voltage expressions to the dq0 reference frame, because of the time-varying flux linkages. Applying the Park transform to Equation 2.21, first on the left-hand side:

$$\left[\begin{array}{cc} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{array}\right] \left[\begin{array}{c} \bar{v}_{ABC} \\ \bar{v}_{fDQ} \end{array}\right] = \left[\begin{array}{c} \bar{v}_{dq0} \\ \bar{v}_{fDQ} \end{array}\right] \tag{2.22}$$

and for resistive voltage drop on the right-hand side:

$$\begin{aligned}
&\begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \left[\begin{array}{c|c} \overline{R}_{ABC} & \bar{0} \\ \hline \bar{0} & \overline{R}_{fDQ} \end{array}\right] \begin{bmatrix} \bar{i}_{ABC} \\ \bar{i}_{fDQ} \end{bmatrix} \\
&= \begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \left[\begin{array}{c|c} \overline{R}_{ABC} & \bar{0} \\ \hline \bar{0} & \overline{R}_{fDQ} \end{array}\right] \begin{bmatrix} \overline{P}^{-1} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \bar{i}_{ABC} \\ \bar{i}_{fDQ} \end{bmatrix} \\
&= \begin{bmatrix} \overline{P} \cdot \overline{R}_{ABC} \cdot \overline{P}^{-1} & \bar{0} \\ \bar{0} & \overline{R}_{fDQ} \end{bmatrix} \begin{bmatrix} \bar{i}_{dq0} \\ \bar{i}_{fDQ} \end{bmatrix} \\
&= \begin{bmatrix} \overline{R}_{ABC} & \bar{0} \\ \bar{0} & \overline{R}_{fDQ} \end{bmatrix} \begin{bmatrix} \bar{i}_{dq0} \\ \bar{i}_{fDQ} \end{bmatrix}
\end{aligned} \tag{2.23}$$

The last equality in Equation 2.23 is valid since $\overline{R}_{ABC}$ is a diagonal matrix consisting of the three phase resistances, which have been assumed equal, $R_A = R_B = R_C = R_s$. Lastly, the Park transform is applied to the last term in Equation 2.21, the flux linkage matrix:

$$\begin{bmatrix} \overline{P} & 0 \\ 0 & \bar{I}_3 \end{bmatrix} \begin{bmatrix} \dot{\overline{\Psi}}_{ABC} \\ \dot{\overline{\Psi}}_{fDQ} \end{bmatrix} = \begin{bmatrix} \overline{P} \cdot \dot{\overline{\Psi}}_{ABC} \\ \dot{\overline{\Psi}}_{fDQ} \end{bmatrix} \tag{2.24}$$

From Equation 2.12, $\overline{\Psi}_{dq0} = \overline{P} \cdot \overline{\Psi}_{ABC}$. Taking the derivative on both sides yields $\dot{\overline{\Psi}}_{dq0} = \frac{d}{dt}(\overline{P} \cdot \overline{\Psi}_{ABC})$. From the product rule in standard calculus, the expression for the time derivative of the dq0 flux linkages becomes:

$$\dot{\overline{\Psi}}_{dq0} = \overline{P}\, \dot{\overline{\Psi}}_{ABC} + \dot{\overline{P}}\, \overline{\Psi}_{ABC} \tag{2.25}$$

Rearranging gives:

$$\begin{aligned}
\overline{P}\, \dot{\overline{\Psi}}_{ABC} &= \dot{\overline{\Psi}}_{dq0} - \dot{\overline{P}}\, \overline{\Psi}_{ABC} \\
&= \dot{\overline{\Psi}}_{dq0} - \dot{\overline{P}}\, \overline{P}^{-1}\, \overline{\Psi}_{dq0}
\end{aligned} \tag{2.26}$$

From the definition of $\overline{P}$ in Equation 2.14, it can be shown that:

$$\dot{\overline{P}}\, \overline{P}^{-1}\, \overline{\Psi}_{dq0} = \begin{bmatrix} 0 & -\omega & 0 \\ \omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Psi_d \\ \Psi_q \\ \Psi_0 \end{bmatrix} = \begin{bmatrix} -\omega\Psi_q \\ \omega\Psi_d \\ 0 \end{bmatrix} = \overline{S} \tag{2.27}$$

This term represents the voltages resulting from the variation in speed, also known as the speed voltages. Now all the terms of Equation 2.21 have been transformed to the dq0 reference frame. Recombining from Equations 2.22, 2.23 and 2.27, the voltage expression becomes:

$$\begin{bmatrix} \bar{v}_{dq0} \\ \bar{v}_{fDQ} \end{bmatrix} = - \begin{bmatrix} \overline{R}_{ABC} & \bar{0} \\ \bar{0} & \overline{R}_{fDQ} \end{bmatrix} \begin{bmatrix} \bar{i}_{dq0} \\ \bar{i}_{fDQ} \end{bmatrix} + \begin{bmatrix} \overline{S} \\ \bar{0} \end{bmatrix} - \begin{bmatrix} \dot{\overline{\Psi}}_{dq0} \\ \dot{\overline{\Psi}}_{fDQ} \end{bmatrix} \tag{2.28}$$

In a real power plant, the machine flux linkages are not easily available. Thus, it can be helpful to express them as functions of inductances and currents, as current measurements are normally readily available. In the chosen transformation, all inductance values have become constants. Therefore, the inductances can be extracted from the derivative flux linkage term, leaving derivative currents, $\dot{\Psi} = L\,\dot{i}$. Substituting Equation 2.18 into Equation 2.28 in expanded form gives:

$$\left[\begin{array}{c} v_d \\ v_q \\ v_0 \\ \hdashline -v_f \\ 0 \\ 0 \end{array}\right] = -\left[\begin{array}{ccc:ccc} R_s & \omega L_q & 0 & 0 & 0 & \omega k M_q \\ -\omega L_d & R_s & 0 & -\omega k M_f & -\omega k M_D & 0 \\ 0 & 0 & R_s & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & R_f & 0 & 0 \\ 0 & 0 & 0 & 0 & R_D & 0 \\ 0 & 0 & 0 & 0 & 0 & R_Q \end{array}\right]\left[\begin{array}{c} i_d \\ i_q \\ i_0 \\ \hdashline i_f \\ i_D \\ i_Q \end{array}\right] - \left[\begin{array}{ccc:ccc} L_d & 0 & 0 & k M_f & k M_D & 0 \\ 0 & L_q & 0 & 0 & 0 & k M_Q \\ 0 & 0 & L_0 & 0 & 0 & 0 \\ \hdashline k M_f & 0 & 0 & L_f & M_d & 0 \\ k M_D & 0 & 0 & M_d & L_D & 0 \\ 0 & k M_Q & 0 & 0 & 0 & L_Q \end{array}\right]\left[\begin{array}{c} \dot{i}_d \\ \dot{i}_q \\ \dot{i}_0 \\ \hdashline \dot{i}_f \\ \dot{i}_D \\ \dot{i}_Q \end{array}\right] \tag{2.29}$$

In the inductance matrices in Equation 2.29 only ω is time-varying, which is a large improvement over Equation 2.20, where nearly all inductances are time-varying.

2.1.5 Per-unit conversion

Until now, all parameters have been considered with real units, such as amperes and volts. However, when analysing the complete machine, this can be impractical as the values between the stator and rotor may differ in orders of magnitude. Normalising the quantities to appropriate base values may alleviate those challenges. From this point onward, a balanced system is assumed. Consequently, $i_A + i_B + i_C = 0$, and $i_0 = 0$. Hence, the zero-sequence equations are omitted in the following.

The stator base quantities are the simplest to define. They are commonly given by the machine's rated values.

- Base power, S_B = The machine's MVA rating per phase (Volt-amperes)
- Stator base voltage, V_B = The machine's rated line-to-neutral RMS terminal voltage (Volts)
- Base speed, ω_B = The machine's rated speed (electrical rad/s)

From these definitions, the rest of the stator base quantities can be determined. The three definitions above, along with Table 2.1, completely defines the stator base quantities.

Symbol	Description	Relationship	Unit
t_B	Base time	$t_B = \frac{1}{\omega_B}$	s
I_B	Base current	$I_B = \frac{S_B}{V_B}$	A
Z_B	Base impedance	$Z_B = \frac{V_B}{I_B}$	Ω
L_B	Base inductance	$L_B = \frac{V_B\, t_B}{I_B} = \frac{V_B}{I_B\, \omega_B}$	H
Ψ_B	Base flux linkage	$\Psi_B = V_B\, t_B = L_B\, I_B$	Vs

Table 2.1: Stator per-unit base quantities with their relation to each other

Any defined per-unit system is not unique, as there are several ways the base quantities can be chosen. A common choice is a per-unit system based upon equal mutual flux linkages [37]. The principle behind this system is that the base field (or d-axis) current is defined such that they will produce the same fundamental air gap flux as the base current acting in the d-axis armature winding. This will result in all mutual flux linkages in an axis being equal.

Each self-inductance quantity in Equation 2.18 can be split into a magnetising and leakage inductance:

$$\begin{aligned} L_d &= L_{md} + l_d & L_D &= L_{mD} + l_D & L_f &= L_{mf} + l_f \\ L_q &= L_{mq} + l_q & L_Q &= L_{mQ} + l_Q \end{aligned} \tag{2.30}$$

where lower-case l represents a leakage inductance and the subscript m denotes magnetising quantities. Only the magnetising inductances contribute to the linking with other windings. The constraint for this per-unit base is that the resulting mutual flux linkages must be equal, which for Equation 2.18 means that:

$$\begin{aligned} \Psi_{md} &= L_{md}\, I_B = kM_f\, I_{fB} = kM_D\, I_{DB} \\ \Psi_{mf} &= kM_f\, I_B = L_{mf}\, I_{fB} = M_d\, I_{DB} \\ \Psi_{mD} &= kM_D\, I_B = M_d\, I_{fB} = L_{mD}\, I_{DB} \\ \Psi_{mq} &= L_{mq}\, I_B = kM_Q\, I_{QB} \\ \Psi_{mQ} &= kM_Q\, I_B = L_{mQ}\, I_{QB} \end{aligned} \tag{2.31}$$

multiplying with the stator current base gives the fundemental constraint for the base currents,

$$\begin{aligned} L_{md}\, I_B^2 &= L_{mf}\, I_{fB}^2 = L_{md}\, I_{DB}^2 \\ &= kM_f\, I_B\, I_{fB} = kM_D\, I_B\, I_{DB} = M_d\, I_{fB}\, I_{DB} \\ L_{mq}\, I_B^2 &= kM_Q\, I_B\, I_{QB} = L_{mQ}\, I_{QB}^2 \end{aligned} \tag{2.32}$$

Recall that the dq0-transformation performed in the previous section was power invariant, meaning the base power of all windings are equal. Along with the constraints above, this creates

the outline for the rotor scaling factors:

$$\begin{aligned}
\frac{V_{fB}}{V_B} &= \frac{I_B}{I_{fB}} = \sqrt{\frac{L_{mf}}{L_{md}}} = \frac{kM_f}{L_{md}} = \frac{L_{mf}}{kM_f} = \frac{M_d}{kM_D} \equiv k_f \\
\frac{V_{DB}}{V_B} &= \frac{I_B}{I_{DB}} = \sqrt{\frac{L_{mD}}{L_{md}}} = \frac{kM_D}{L_{md}} = \frac{L_{mD}}{kM_D} = \frac{M_d}{kM_f} \equiv k_D \\
\frac{V_{QB}}{V_B} &= \frac{I_B}{I_{QB}} = \sqrt{\frac{L_{mQ}}{L_{mq}}} = \frac{kM_Q}{L_{mq}} = \frac{L_{mQ}}{kM_Q} \equiv k_Q
\end{aligned} \tag{2.33}$$

With these scaling factors, it is now possible to define the rotor base quantities. This is done in Table 2.2.

Symbol	Description	Relationship	Unit
Z_{fB}	Base field impedance	$Z_{fB} = k_f^2\, Z_B$	Ω
Z_{DB}	Base d-axis damper impedance	$Z_{DB} = k_D^2\, Z_B$	Ω
Z_{QB}	Base q-axis damper impedance	$Z_{QB} = k_Q^2\, Z_B$	Ω
L_{fB}	Base field inductance	$L_{fB} = k_f^2\, L_B$	H
L_{DB}	Base d-axis damper inductance	$L_{DB} = k_D^2\, L_B$	H
L_{QB}	Base q-axis damper inductance	$L_{QB} = k_Q^2\, L_B$	H
M_{fB}	Base field mutual inductance	$M_{fB} = k_f\, L_B$	H
M_{DB}	Base d-axis damper mutual inductance	$M_{DB} = k_D\, L_B$	H
M_{QB}	Base q-axis damper mutual inductance	$M_{QB} = k_Q\, L_B$	H
M_{dB}	Base d-axis damper to field mutual inductance	$M_{dB} = k_f\, k_D\, L_B$	H

Table 2.2: Rotor per-unit base quantities with their relation to the stator bases

To prove that the choice of base quantities satisfy the desire for equal mutual inductances, the relationships in Equation 2.33 and Table 2.2 can be combined to show that, in per-unit,

$$\begin{aligned}
L_{md} &= L_{mf} = kM_f = kM_D = L_{ad} \\
L_{mq} &= L_{mQ} = kM_Q = L_{aq}
\end{aligned} \tag{2.34}$$

It is common practice that the per-unit magnetising inductances are given the subscript a. Lastly, substituting Equations 2.30 and 2.34 into Equation 2.29 gives:

$$
\left[\begin{array}{c} v_d \\ v_q \\ v_0 \\ \hdashline -v_f \\ 0 \\ 0 \end{array}\right] = -
\left[\begin{array}{ccc:ccc} R_s & \omega(L_{aq}+l_q) & 0 & 0 & 0 & \omega L_{aq} \\ -\omega(L_{ad}+l_d) & R_s & 0 & -\omega L_{ad} & -\omega L_{ad} & 0 \\ 0 & 0 & R_s & 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 & R_f & 0 & 0 \\ 0 & 0 & 0 & 0 & R_D & 0 \\ 0 & 0 & 0 & 0 & 0 & R_Q \end{array}\right]
\left[\begin{array}{c} i_d \\ i_q \\ i_0 \\ \hdashline i_f \\ i_D \\ i_Q \end{array}\right]
$$

$$
- \left[\begin{array}{ccc:ccc} L_{ad}+l_d & 0 & 0 & L_{ad} & L_{ad} & 0 \\ 0 & L_{aq}+l_q & 0 & 0 & 0 & L_{aq} \\ 0 & 0 & L_0 & 0 & 0 & 0 \\ \hdashline L_{ad} & 0 & 0 & L_{ad}+l_f & L_{ad} & 0 \\ L_{ad} & 0 & 0 & L_{ad} & L_{ad}+l_D & 0 \\ 0 & L_{aq} & 0 & 0 & 0 & L_{aq}+l_Q \end{array}\right]
\left[\begin{array}{c} \dot{i}_d \\ \dot{i}_q \\ \dot{i}_0 \\ \hdashline \dot{i}_f \\ \dot{i}_D \\ \dot{i}_Q \end{array}\right] \tag{2.35}
$$

Equation 2.35 describes the behaviour of the synchronous machine voltages. Note that the d- and q-axis stator leakage inductances are usually nearly equal, so it is fair to assume that $l_d = l_q$.

2.1.6 Equivalent circuits for the dq-axes

The large number of equations and matrices in the previous sections may be somewhat confusing, as they are on a highly theoretical basis. Therefore, a visual representation of the flux relationships can be helpful.

Equation 2.30 shows how the winding self-inductances can be split into a magnetising and leakage inductance, and Equation 2.34 states the magnetising inductances in each axis are equal. Multiplying each with their respective winding current gives the magnetising flux linkages. This relation can be written as:

$$
\begin{aligned} \Psi_d - l_d\, i_d = \Psi_f - l_f\, i_f = \Psi_D - l_D\, i_D \equiv \Psi_{ad} \\ \Psi_q - l_q\, i_q = \Psi_Q - l_Q\, i_Q \equiv \Psi_{aq} \end{aligned} \tag{2.36}
$$

where lower-case l denotes leakage inductances. Rearranging yields:

$$
\begin{aligned} \Psi_{ad} = (L_d - l_d) i_d + L_{mf}\, i_f + L_{mD}\, i_D = L_{ad}\big(i_d + i_f + i_D\big) \\ \Psi_{aq} = (L_q - l_q) i_q + L_{mQ}\, i_Q = L_{aq}\big(i_q + i_Q\big) \end{aligned} \tag{2.37}
$$

Notice how all currents in an axis flow through a common inductance, L_{ad} or L_{aq}. Additionally, an inductance must be added for each winding to represent the leakage. These must be in series with the magnetising inductance ($L = L_a + l$), but not in series with each other. Thus, three branches must be added to the d-axis (d, f and D) and two to the q-axis (q and Q). The voltage

equations from Equation 2.28 can be written as:

$$\begin{aligned} v_d &= -R_s\, i_d - \dot{\Psi}_d - \omega\Psi_q \\ &= -R_s\, i_d - l_d\, \dot{i}_d - L_{ad}(\dot{i}_d + \dot{i}_f + \dot{i}_D) - \omega\Psi_q \end{aligned} \tag{2.38}$$

Similarly, for the remaining voltage equations, rewriting gives:

$$\begin{aligned} -v_f &= -R_f\, i_f - l_f\, \dot{i}_f - L_{ad}(\dot{i}_d + \dot{i}_f + \dot{i}_D) \\ v_D = 0 &= -R_D\, i_D - l_D\, \dot{i}_D - L_{ad}(\dot{i}_d + \dot{i}_f + \dot{i}_D) \\ v_q &= -R_s\, i_q - l_q\, \dot{i}_q - L_{aq}(\dot{i}_q + \dot{i}_Q) + \omega\Psi_q \\ v_Q = 0 &= -R_Q\, i_Q - l_Q\, \dot{i}_Q - L_{aq}(\dot{i}_q + \dot{i}_Q) \end{aligned} \tag{2.39}$$

Equations 2.37 to 2.39 lay the foundation for the equivalent diagrams. Making the stator voltages v_d and v_q open outputs and representing v_f and the speed voltage terms as controlled voltage sources, the d- and q-axis equivalent circuits can be drawn as in Figures 2.3 and 2.4 respectively.

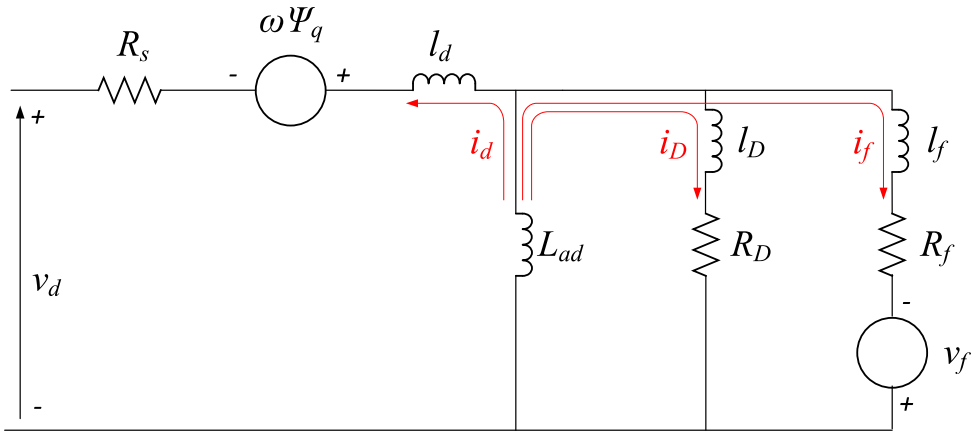


Figure 2.3: d-axis equivalent circuit of a synchronous machine with one d-axis damper circuit

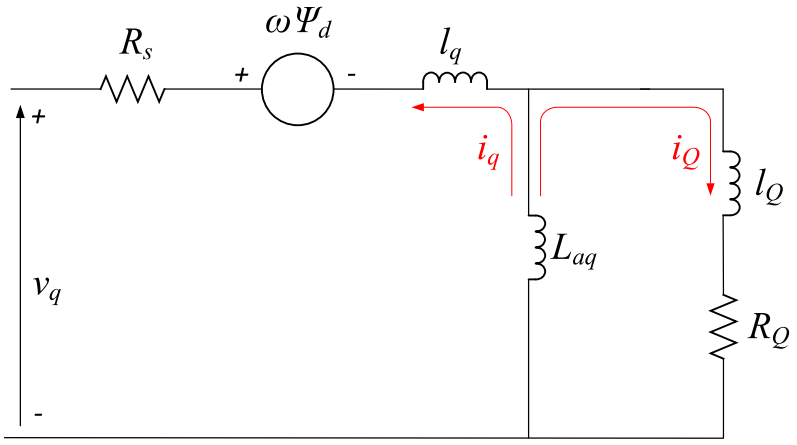


Figure 2.4: q-axis equivalent circuit of a synchronous machine with one q-axis damper circuit

Notice the current directions in the circuits. This section adopts the current convention in [36] and [37]. However, some textbooks, such as [35], applies a current convention such that Equation 2.37 rather becomes:

$$\begin{aligned} \Psi_{ad} &= L_{ad}(i_d - i_f - I_D) \\ \Psi_{dq} &= L_{aq}(i_q - i_Q) \end{aligned} \tag{2.40}$$

This difference comes from the chosen unit conventions. In the convention applied in [35] the positive field current direction is chosen to go out of the machine. However, this is not the case for the convention used here. Instead, in Equation 2.35, a negative sign has been given to v_f. Thus, the result becomes the same. This fact is pointed out to highlight the importance of sign conventions. Both describe identical machines and behaviour while using different positive current directions. Consequently, it is important to be aware when doing machine analysis which convention is used.

2.1.7 Mechanical equations

In the previous sections the electrical model of the machine has been described. To get a complete picture of the machine behaviour it is also necessary to describe the mechanical dynamics. Whether the rotor angle is constant, increasing or decreasing depends on a balance of torques. In generator operation the generator shaft is accelerated by a turbine. The driving torque from the turbine is called the mechanical torque and is denoted T_m. An opposing torque is created from the loading of the generator and is called the electromagnetic or electrical torque, denoted by T_e. The balance between these torques determines the rotor angle acceleration. Thus, their difference is commonly referred to as the accelerating torque,

$$T_a = T_m - T_e \tag{2.41}$$

This convention implies that a positive mechanical torque accelerates the rotor, while a positive electrical torque decelerates it. This is a realistic description of the torque relationship. Since T_a describes the rotor acceleration it is possible to express it as the derivative of machine speed,

$$J\,\ddot{\delta}_m = J\,\dot{\omega}_m = T_a = T_m - T_e \tag{2.42}$$

where J is the moment of inertia of the generator shaft and turbine, δ_m is the mechanical torque angle and ω_m is the rotor's mechanical angular velocity. It is common to normalise this equation by introducing the per-unit inertia constant H, which is defined by:

$$\begin{aligned} H &= \frac{1}{2}\frac{J\,\omega_{mR}^2}{S_B} \\ \Rightarrow J &= \frac{2H}{\omega_{mR}^2}\,S_B \end{aligned} \tag{2.43}$$

where ω_{mR} is the rated mechanical angular velocity and S_B is the base power. Substituting Equation 2.42 into Equation 2.43 and rearranging yields:

$$\begin{aligned} \frac{2H}{\omega_{mR}^2}\,S_B\,\dot{\omega}_m &= T_m - T_e \\ \Rightarrow 2H\,\frac{\dot{\omega}_m}{\omega_{mR}} &= \frac{T_m - T_e}{\frac{S_B}{\omega_{mR}}} \\ \Rightarrow 2H\,\dot{\omega}_r &= T_{m,pu} - T_{e,pu} \end{aligned} \tag{2.44}$$

where ω_r is the per-unit angular velocity. Note that the per-unit mechanical speed is equal to the per-unit electrical speed, as they are divided by their respective bases. Sometimes, an extra term accounting for the damping torque component is added. This is implemented as a damping coefficient K_D multiplied with the speed deviation,

$$2H\,\dot{\omega}_r = T_{m,pu} - T_{e,pu} - K_D\,\Delta\omega_r \tag{2.45}$$

This is a well-known equation referred to as the swing equation. It can also be expressed in terms of the rotor angle:

$$\frac{2H}{\omega_B}\,\ddot{\delta} = T_{m,pu} - T_{e,pu} - \frac{K_D}{\omega_B}\,\dot{\delta} \tag{2.46}$$

It was stated that the electrical torque comes from the loading of the generator. Still, it is useful to have a mathematical description of it. The Park transformation in Section 2.1.3 was said to be power invariant, expressed by Equation 2.15. Thus, under balanced conditions,

$$P_{out} = v_d\, i_d + v_q\, i_q \tag{2.47}$$

Substituting the voltages from Equation 2.28,

$$P_{out} = \left(i_d\,\dot{\Psi}_d + i_q\,\dot{\Psi}_q\right) + \left(i_q\,\Psi_d - i_d\Psi_q\right)\omega - R_s\left(i_d^2 + i_q^2\right) \tag{2.48}$$

From rotational physics, the work done by a small rotation is given by $dW = T\,d\theta$, where T is a torque and θ is the rotation angle. Taking the time derivative on both sides yields $dP = T\,d\omega$, where P is a power and ω is the rotational speed. Rearranging and substituting from Equation 2.48 gives:

$$\begin{aligned} T_e =& \frac{dP}{d\omega} = \frac{d}{d\omega}\left[\left(i_d\,\dot{\Psi}_d + i_q\,\dot{\Psi}_q\right) + \left(i_q\,\Psi_d - i_d\Psi_q\right)\omega - R_s\left(i_d^2 + i_q^2\right)\right] \\ =& i_q\,\Psi_d - i_d\Psi_q \end{aligned} \tag{2.49}$$

2.2 The excitation system

The main actuator of the synchronous machine is the excitation system controller. It controls the field voltage to regulate the terminal voltage and reactive power delivery. Its performance is of great importance to the machine's stability. There are three main components to the excitation system: the exciter, AVR and PSS, see Figure 2.5 [35]. This subsection will give a brief introduction to each of those components.

In Norway, there are specific guidelines for how excitation systems should be designed based on the generator rating and application area. These are published by Statnett, which is the transmission system operator (TSO) in Norway. The current guideline for Norwegian power systems is called *FIKS 2012* [40]. However, in 2020 it is to be revised. The new guideline will be called *NVF 2020* and is under hearing at the time of writing [41].

2.2.1 The exciter

The exciter has the purpose of providing the field current to the rotor windings, such to create the magnetising field for the synchronous machine. There are mainly three types of exciters: the AC, DC and static exciters. In the new guidelines *NVF 2020* it is recommended that all generators rated 30 MW or higher have static excitation, whereas in *FIKS 2012* it was 25 MVA or higher. Thus, static excitation is most common. Several reliable sources describe AC and DC type exciters in detail [35, 42], so these will not be discussed further here.

One of the most popular types of static exciters is the potential source-fed system, see Figure 2.6 [1, 42], where the generator terminals are the power source for the exciter. The AC terminal voltage is transformed down to the required level and rectified through a thyristor rectifier (SCR). This works well for machines connected to large systems, as excitation power can be provided from external sources during a short-circuit. [1].

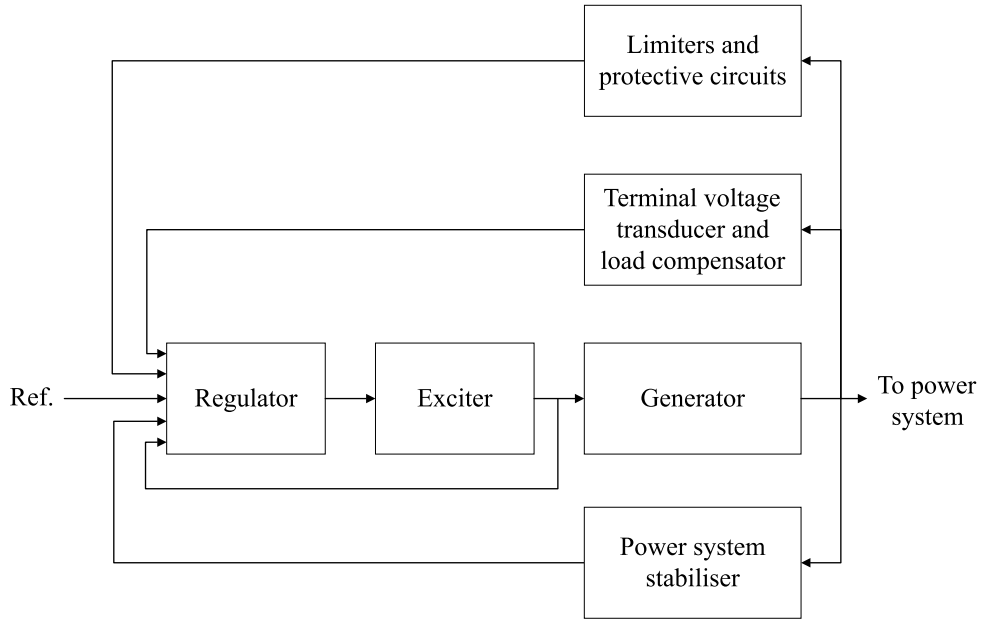


Figure 2.5: Principle block diagram of an excitation system, showing its feedback loops and information transfer

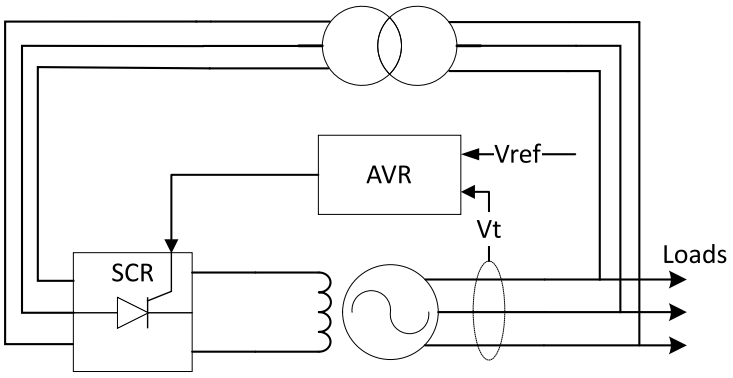


Figure 2.6: Block diagram of a potential-source static excitation system

2.2.2 The automatic voltage regulator

The automatic voltage regulator (AVR) is the main controller of the excitation system. Its responsibility is to observe machine terminal voltage changes and respond accordingly by adjusting the field voltage. There have been developed guidelines for modelling of AVRs and their associated components for different applications. The Institute of Electrical and Electronics Engineers (IEEE) has published recommended practices for excitation system modelling [43]. For example, a potential-source fed static excitation system with a PI regulator type AVR can be categorised as an ST7C model.

2.2.2.1 Inputs and outputs

The AVR typically has four inputs:

- The control or compensated voltage V_c. This is the measurement of the terminal voltage.

It is not uncommon that the control voltage is compensated to a certain electrical distance away from the machine terminals. This compensation is the job of the load compensator in Figure 2.5. An electrical distance of around 0.8 times the step-up transformer's reactance is not uncommon [36].

- The reference voltage V_{ref}. This is the voltage the AVR strives to keep the control voltage at.
- Terminal voltage V_t. Even if the terminal voltage is not used as the control voltage, it may be used for limiting purposes as a base value for per-unit conversion.
- PSS contribution V_{pss}. This additional signal from the power system stabiliser aims to increase the damping of oscillatory behaviour in the rotor angle. The PSS will be described in Section 2.2.3.

The AVR generally only has a single output. Technically, the output is the pulse train to the thyristor rectifier. However, for modelling purposes, the rectifier can be considered a simple transfer function, and the AVR is said to have the generator field voltage as output. [1]

2.2.2.2 Limiters

In real systems, the machine windings cannot be exposed to overly high currents for too long, as they will get damaged. Therefore, some limiters are implemented into the AVR. There are mainly four limiters:

- The overexcitation limiter (OEL) limits the maximum field current to the machine rotor windings according to their thermal capabilities. Still, the OEL allows for some overcurrent to flow for an amount of time such that a high field forcing during disturbances is still possible. If the overexcitation lasts too long, the OEL starts to limit the field current to safe levels. [1, 43].
- The underexcitation limiter (UEL) ensures the machine does not become too underexcited. If this occurs, the machine may lose synchronism or associated loss-of-excitation relays may trip.
- The stator current limiter (SCL) protect the stator winding from overcurrents by limiting the field voltage. Since the SCL uses the excitation level to adjust the stator current, it only affects the reactive power flow. Thus, the correct control action to do when the machine is overexcited (capacitive operation) is to reduce the excitation and increase excitation when the machine is underexcited (inductive operation) [1, 43].
- The field voltage limiter limits the output of the AVR to avoid damaging the field windings. This limiter is what determines the field ceiling voltage.

2.2.2.3 The AVR and stability

Consider the generator-infinite bus system in Figure 2.7 [1]. Assuming the internal emf E_q is constant and $X_d = X_q$, the air-gap power delivered by the machine is given by:

$$P_e = \frac{E_q\, V_s}{x_d}\, sin(\delta) \tag{2.50}$$

where,

- P_e = The delivered air-gap power of the generator
- E_q = The internal induced voltage of the machine
- V_s = The voltage level at the infinite bus
- $x_d = X_d + X_t + X_s$ The sum of the synchronous, transformer and network equivalent reactances
- δ = Power angle of the generator

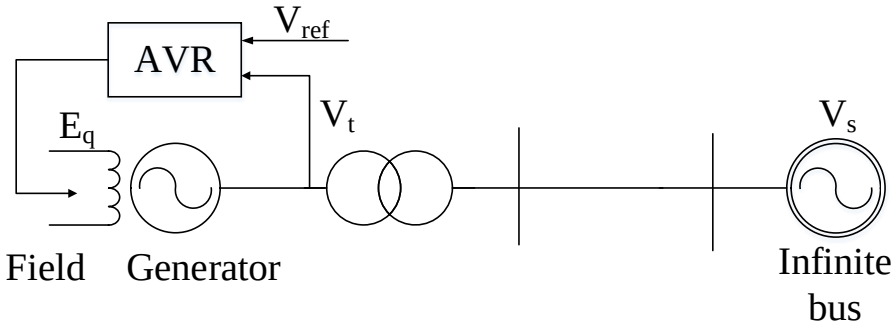


Figure 2.7: Line diagram of a generator-infinite bus system. The field voltage is determined by the AVR, which uses the terminal voltage to calculate the error from the reference. [1]

E_q being constant implies that there is no AVR action and the system is unregulated. The peak air-gap power occurs at $\delta = \frac{\pi}{2}$. If the load angle increases any further it will result in unstable operation, as further increases in load will result in a decrease in power delivery, see the blue curve in Figure 2.8.

When introducing an AVR the regulator will strive to keep the control voltage constant by adjusting the field current. In a steady-state condition, the power-angle characteristic may look the same as for the unregulated one. However, if a load increase occurs, the stator current will increase, resulting in a larger voltage drop across the internal reactance of the machine. Thus, the terminal voltage will drop. This will be detected by the AVR which will work to increase the generator's excitation level. When that happens a new, higher power-angle characteristic will be valid. This sequence of events will repeat at further load increases, and reversely so for load decreases. Consequently, the total power-angle curve for a regulated machine will have a shape like the red curve in Figure 2.8 [1, 36]. The regulated power-angle curve is derived from:

$$P_e = \frac{V_s}{x_d} sin(\delta) \sqrt{\left(\frac{x_d V_t}{X}\right)^2 - \left(\frac{X_d}{X} V_s sin(\delta)\right)^2} - \frac{1}{2} \frac{X_d}{X} \frac{V_s^2}{x_d} sin(2\delta) \tag{2.51}$$

where,

- P_e = The delivered air-gap power of the generator
- E_q = The internal induced voltage
- V_t = Generator terminal voltage
- V_s = The voltage level at the infinite bus

- X = $X_t + X_s$ = The sum of transformer and network equivalent reactances
- x_d = $X_d + X$ = The sum of the synchronous, transformer and network equivalent reactances
- δ = Power angle of the generator

An AVR will notably increase the amount of power the machine can deliver. The new maximum load angle δ_m is higher than for the unregulated curve. This means that when a large disturbance happens, like a fault or a line disconnection, the generator has a much higher stability limit and de-acceleration area. Thus, the inclusion of an AVR significantly increases the synchronising torque, and consequently the transient stability of the generator.

Even though the effects of an AVR have positive implications for transient stability, it can be shown that the AVR may introduce negative damping and decrease oscillatory stability. After a disturbance, the generator may keep synchronism after the first rotor swing yet become unstable in an oscillatory fashion after a few swings. The prominence of this negative damping is dependent on factors as generator load, AVR gain and network reactance. A large value in any of these enhances the negative damping [36]. Compensating for the negative damping commonly involves introducing an additional control loop to the AVR, called a power system stabiliser. [1]

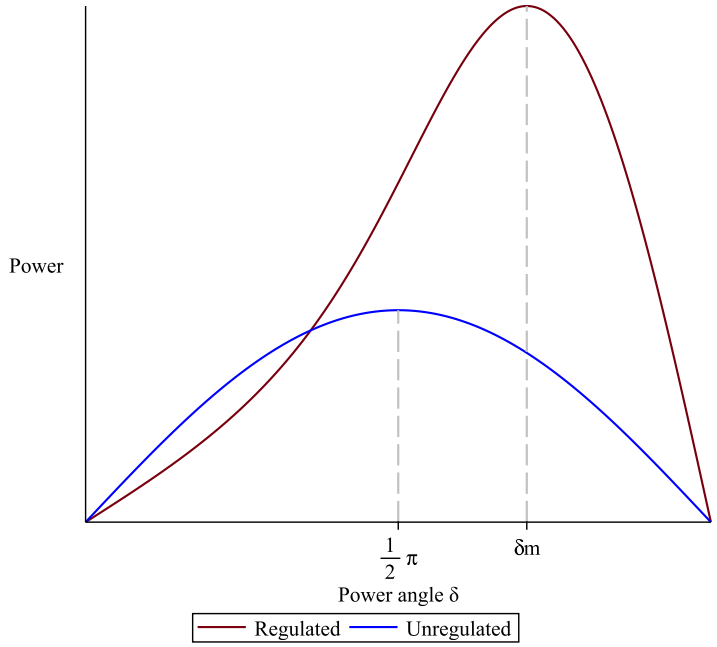


Figure 2.8: Air-gap power for both an unregulated and regulated system. This shows that a regulated system (with AVR) has a higher stability limit than an unregulated system (without AVR).

2.2.3 The power system stabiliser

The power system stabiliser (PSS) has the main purpose of counteracting oscillatory behaviour in the rotor angle. It is designed to damp low-frequency oscillations in the range of 0.2 to 3 Hz by adding a signal to the AVR with opposing phase [43, 44]. From 2020, it is recommended that all Norwegian generators rated 30 MW or higher should have a PSS installed [41].

The PSS has dominantly been designed as a transfer function. Various models have been suggested by IEEE's recommended practices [43], the most common model being a simplified version of the *PSS1A* type PSS. This type can be seen in Figure 2.9 and is commonly known as a conventional PSS (CPSS).

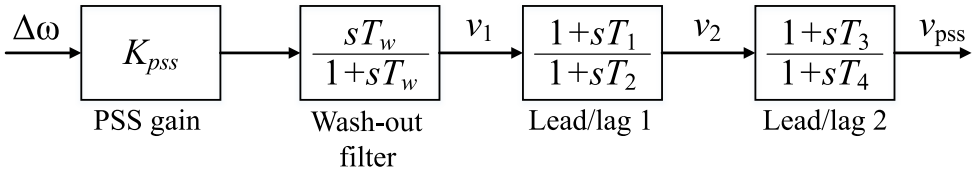


Figure 2.9: Block diagram of a conventional PSS with two lead/lag stages

The three components of the CPSS model is the PSS gain, a wash-out (high-pass) filter to eliminate any offset of the voltage regulation in steady-state and one or more lead/lag stages to shape the phase response of the stabiliser [1, 45]. At the output of the PSS, there is commonly a limiter, such not to overtake the AVR action and to prevent overly large excitation. An upper limit in the range 0.1 to 0.2 pu and lower limit -0.05 to -0.1 pu are appropriate [35].

It is beyond the scope of this thesis to go into detail about the calculation behind tuning each type of PSS for different inputs. However, some good explanations of the different types and simple examples can be found in chapters 12.5 and 17.2.1 in [35]. Also, in [45], chapter 8 goes into detail of PSS performances with different input quantities. Chapters 5, 6 and 9 in the same reference goes in great detail of the system modelling and how the PSS parameters can be tuned accordingly. Chapter 5.5 in [46] describes three common ways of designing a PSS using either a damping torque, frequency response or eigenvalue approach.

This type of PSS, which uses a gain and lead-lag structure, is typically tuned before it is set into operation and subsequently left alone. There is rarely any great effort put into optimising their parameters. Rather, they are often tuned with the philosophy of "an adequately tuned PSS is better than no PSS." While this statement is true, there are clear advantages to optimising the tuning. [1]

2.3 Simplified linear model for stability assessment

In the previous sections, the complete electrical behaviour of the machine was derived. However, for many stability studies, this level of complexity may be unnecessary. Particularly in small-signal stability analysis, in which the PSS tuning problem lies, simplified models are commonplace. This section will derive a model of the synchronous machine that is appropriate for small-signal stability and show how this model can be linearised for stability analysis. The procedure described in this section is mainly inspired by chapters 6.1 and 6.5 in [37].

2.3.1 Linearisation

Many dynamic systems, including synchronous machines, are non-linear. This means that their behaviour cannot be represented by a linear system of equations. Typical occurrences of non-linearities in models are product non-linearities, trigonometric functions and saturation effects.

Linearisation assumes disturbances that are so small that they perturb the state variables only slightly. After the disturbance, the system will experience oscillatory behaviour before it settles on a new operating point. The new operating point should be near the pre-disturbance operating point. If these requirements are met the system is said to be linearised about the operating point. In other words, the system equations are approximated to a first-order system, valid near the operating point.

To linearise an equation, determine which parameters of the equation are non-changing and which are perturbed. The perturbed parameters are state variables or can be expressed by a linear combination of state variables.

As was mentioned, the perturbation of the state variables is said to be small. A common notation is to use the letter Δ to signify a change. A perturbation in a state variable can be written as $x_{i0}+x_{i\Delta}$, where the variable x_i initiates at x_{i0} and moves by $x_{i\Delta}$, which represents a small change. The initial values, denoted by a subscript 0, are known values and treated as constants.

Product non-linearities are solved by first-order approximations. Let the product between two state variables is $x_i x_j$. A perturbation occurs, and both states deviate from their operating point by a small amount. The new operating point becomes:

$$(x_{i0} + x_{i\Delta})(x_{j0} + x_{j\Delta}) = x_{i0}x_{j0} + x_{i0}x_{j\Delta} + x_{j0}x_{i\Delta} + x_{i\Delta}x_{j\Delta} \tag{2.52}$$

Notice how the last term is a second-order term. Since both perturbations are small, their product is assumed to be negligible, hence the name first-order approximation,

$$(x_{i0} + x_{i\Delta})(x_{j0} + x_{j\Delta}) \approx x_{i0}x_{j0} + x_{i0}x_{j\Delta} + x_{j0}x_{i\Delta} \tag{2.53}$$

Trigonometric non-linearities, where state variables are angles, are treated with simple approximations. Let a perturbation in the state $cos(\delta)$ be given by $cos(\delta_0 + \delta_\Delta)$. A well known trigonometric identity gives:

$$cos(\delta_0 + \delta_\Delta) = cos(\delta_0)\, cos(\delta_\Delta) - sin(\delta_0)\, sin(\delta_\Delta) \tag{2.54}$$

Since the angle perturbation is small, $sin(\delta_\Delta) \approx \delta_\Delta$ and $cos(\delta_\Delta) \approx 1$. A change in $cos(\delta)$ is therefore given by:

$$cos(\delta_0 + \delta_\Delta) - cos(\delta_0) \approx -sin(\delta_0)\, \delta_\Delta \tag{2.55}$$

Similarly, for a perturbation in $sin(\delta)$:

$$sin(\delta_0 + \delta_\Delta) - sin(\delta_0) \approx cos(\delta_0)\, \delta_\Delta \tag{2.56}$$

Thus, a change in $cos(\delta)$ is given by $-sin(\delta_0)\ \delta_\Delta$, and a change in $sin(\delta)$ is given by $cos(\delta_0)\ \delta_\Delta$.

Saturation effects represent large non-linearities, no less so for synchronous machines. However, for small disturbances, the saturation in the machine is usually neglected. This is because the implications of the disturbance are said to be so small that saturation does not occur to any great effect. This may lead to great inaccuracies in some studies, but is an acceptable approximation in such small-disturbance analyses.

2.3.2 Linearised synchronous machine model

When the machine behaviour during a disturbance is to be analysed, it is necessary to implement a model of the connected power system. However, the power system is vast and may have hundreds of connected components. Thus, deriving a detailed model is highly impractical. For the analysis of a single machine, a much simpler model is sufficient. In this section, a machine connected to an infinite bus through a transmission line is considered, as illustrated in Figure 2.10. The figure only shows phase A of the machine, assuming no coupling between stator phases.

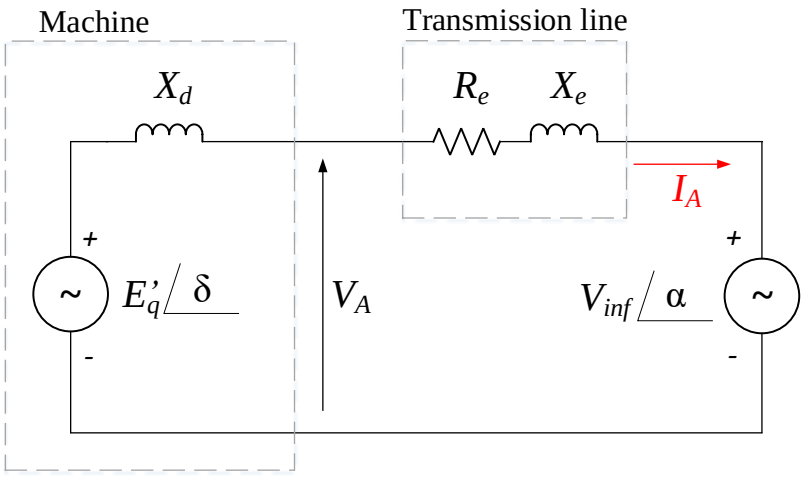


Figure 2.10: Equivalent circuit of a machine connected to an infinite bus through a transmission line

The main assumptions the linear model is based upon are:

- The effects of damper windings are negligible
- The stator winding resistance R_s is negligible
- The $\dot{\Psi}$ voltage terms are negligible compared to the speed voltage terms $\omega\Psi$
- The speed voltage terms $\omega\Psi$ are approximately equal to $\omega_R\Psi$
- The system is balanced, i.e. $i_0 = 0$
- All saturation effects are negligible

2.3.2.1 emf equation

Firstly, it is helpful to determine expressions for the machine internal voltages. In the literature [35, 37], the following emfs are commonly defined[1]:

[1]The exact symbols, subscripts and definitions of these emfs vary somewhat in the literature. Here, the emf notation follows that of [35], but their definition follows that of [37], where the $\sqrt{3}$ is included. This difference originates in the authors' different choices of k in the dq0-transformation.

$$\begin{aligned} E_I = \tfrac{\omega_R\, L_{ad}}{\sqrt{3}}\, i_f &= \text{emf proportional to the field current } i_f \\ E_q^{'} = \tfrac{\omega_R\, Lad}{\sqrt{3}\, Lf}\, \Psi_f &= \text{emf proportional to field flux linkage } \Psi_f \\ E_{fd} = \tfrac{\omega_R\, L_{ad}}{\sqrt{3}\, R_f}\, v_f &= \text{emf proportional to the field voltage } v_f \end{aligned}$$

The field winding flux linkage expression from Equation 2.18 is:

$$\Psi_f = L_{ad}\, i_d + L_f\, i_f \tag{2.57}$$

Multiplying by $(\omega_R\, L_{ad})/(\sqrt{3}\, L_f)$ on both sides,

$$\begin{aligned} \Psi_f\, \frac{\omega_R\, L_{ad}}{\sqrt{3}\, L_f} &= L_{ad}\, i_d\, \frac{\omega_R\, L_{ad}}{\sqrt{3}\, L_f} + L_f\, i_f\, \frac{\omega_R\, L_{ad}}{\sqrt{3}\, L_f} \\ \Rightarrow E_q^{'} &= \frac{\omega_R\, L_{ad}^2}{L_f}\, I_d + E_I \end{aligned} \tag{2.58}$$

where $I_d = i_d/\sqrt{3}$. From the definition of the transient d-axis inductance [35], it can be shown that,

$$\begin{aligned} L_d^{'} = l_d + \frac{L_{ad}\, l_f}{L_{ad} + l_f} = (L_d - L_{ad}) + \frac{L_{ad}\, l_f}{L_{ad} + l_f} &= L_d - \frac{L_{ad}^2}{L_f} \\ \Rightarrow \frac{L_{ad}^2}{L_f} &= L_d - L_d^{'} \end{aligned} \tag{2.59}$$

Combining Equations 2.58 and 2.59 results in:

$$E_q^{'} = E_I + \omega_R \left(L_d - L_d^{'}\right) I_d = \frac{\omega_R\, L_{ad}}{\sqrt{3}}\, i_f + \omega_R \left(L_d - L_d^{'}\right) I_d \tag{2.60}$$

This equation is notably linear, where $E_q^{'}$, i_f and I_d are state variables. A perturbed version would be:

$$E_{q\Delta}^{'} = \frac{\omega_R\, L_{ad}}{\sqrt{3}}\, i_{f\Delta} + \omega_R \left(L_d - L_d^{'}\right) I_{d\Delta} \tag{2.61}$$

Next is to define the emf E_{fd}. From Equation 2.28 and the assumptions made, the field voltage expression is:

$$\dot{\Psi}_f = v_f - R_f\, i_f \tag{2.62}$$

Multiplying by $(\omega_R\, L_{ad})/(\sqrt{3}\, L_f)$ on both sides,

$$\begin{aligned} \frac{d}{dt}\Big(\frac{\omega_R\, L_{ad}}{\sqrt{3}\, L_f}\, \Psi_f\Big) = \frac{\omega_R\, L_{ad}}{\sqrt{3}\, L_f} \frac{R_f}{R_f}\, v_f - \frac{\omega_R\, R_f}{\sqrt{3}\, L_f}\, L_{ad}\, i_f \\ \Rightarrow \dot{E}_q^{'} = \frac{1}{T_{d0}^{'}}\big(E_{fd} - E_I\big) \end{aligned} \tag{2.63}$$

where $T_{d0}^{'} = L_f/R_f$ is the machine open-circuit transient time constant [35]. Rearranging, denoting the derivative in the s-domain and inserting from Equation 2.60 yields:

$$\begin{aligned} E_{fd} = T_{d0}^{'}\, E_q^{'}\, s + E_I = T_{d0}^{'}\, E_q^{'}\, s + E_q^{'} - \left(X_d - X_d^{'}\right) I_d \\ \Rightarrow E_{fd} = \left(1 + T_{d0}^{'}\, s\right) E_q^{'} - \left(X_d - X_d^{'}\right) I_d \end{aligned} \tag{2.64}$$

Note that $X = \omega_R\, L$ and s is the Laplace transform variable. The above equation is linear, with E_{fd}, $E_q^{'}$ and I_d as state variables. A perturbed version would be:

$$E_{fd\Delta} = \left(1 + T_{d0}^{'}\, s\right) E_{q\Delta}^{'} - \left(X_d - X_d^{'}\right) I_{d\Delta} \tag{2.65}$$

Next, the linearised voltages at infinite bus loading are needed. Inspection of Figure 2.10 gives the voltage relation:

$$v_A = v_{inf,A} + R_e\, i_A + L_e\, \dot{i_A} \tag{2.66}$$

Applying the same relation to all three phases and using matrix notation:

$$\bar{v}_{ABC} = \bar{v}_{inf,ABC} + R_e\, \bar{i}_{ABC} + L_e\, \dot{\bar{i}}_{ABC} \tag{2.67}$$

where resistances and inductances are assumed equal across the transmission line phases. $\bar{v}_{inf,ABC}$ is assumed to be a set of balanced peak voltages. Applying the Park transformation from Equation 2.14 to Equation 2.67, with zero-sequence omitted, gives:

$$\bar{v}_{dq} = \bar{v}_{inf,dq} + R_e\, \bar{i}_{dq} + L_e\, \overline{P}\, \dot{\bar{i}}_{ABC} \tag{2.68}$$

It is possible to show [37], that through manipulation of trigonometric identities, the dq-voltages at the infinite bus can be written as:

$$\bar{v}_{inf,dq} = \overline{P}\, \bar{v}_{inf,ABC} = \sqrt{3}\, V_{inf} \begin{bmatrix} -sin(\delta - \alpha) \\ cos(\delta - \alpha) \end{bmatrix} \tag{2.69}$$

where V_{inf} is the magnitude of the RMS phase-neutral voltage. Similarly to Equation 2.27,

$$\overline{P}\, \dot{\bar{i}}_{ABC} = \dot{\bar{i}}_{dq} - \dot{\overline{P}}\, \overline{P}^{-1}\, \bar{i}_{dq} = \dot{\bar{i}}_{dq} - \omega \begin{bmatrix} -i_q \\ i_d \end{bmatrix} \tag{2.70}$$

However, the assumptions made for the linear model states that this speed voltage term can be approximated to rated speed, so $\omega = \omega_R$. Moreover, the derivative current term in Equation 2.70 is neglected. Including the voltage expression from Equation 2.35, the voltage equations become:

$$\begin{aligned} \begin{bmatrix} v_d \\ v_q \end{bmatrix} &= \begin{bmatrix} -\omega_R\, L_q\, i_q \\ \omega_R\, L_d\, i_d + \omega_R\, kM_f\, i_f \end{bmatrix} \\ &= \sqrt{3}\, V_{inf} \begin{bmatrix} -sin(\delta - \alpha) \\ cos(\delta - \alpha) \end{bmatrix} + R_e \begin{bmatrix} i_d \\ i_q \end{bmatrix} - \omega_R\, L_e \begin{bmatrix} -i_q \\ i_d \end{bmatrix} \end{aligned} \tag{2.71}$$

Now it possible to linearise. Following the linearisation procedure described in Section 2.3.1, Equation 2.71 can be linearised about its operating point. For now, i_d, i_q, i_f and δ will be treated as perturbed state variables. Also, recall that $X = \omega_R\, L$ and $I = i/\sqrt{3}$. The linearised voltage equations become:

$$\begin{aligned} 0 &= -R_e\, I_{d\Delta} - \left(X_q + X_e\right) I_{q\Delta} + \left[V_{inf}\, cos(\delta_0 - \alpha)\right] \delta_\Delta \\ 0 &= -R_e\, I_{q\Delta} + \left(X_d + X_e\right) I_{d\Delta} + \frac{\omega_R\, kM_f}{\sqrt{3}}\, i_{f\Delta} + \left[V_{inf}\, sin(\delta_0 - \alpha)\right] \delta_\Delta \end{aligned} \tag{2.72}$$

Inserting Equation 2.61 into Equation 2.72 and rearranging gives:

$$\begin{aligned} R_e\, I_{d\Delta} + \big(X_q + X_e\big)\, I_{q\Delta} &= \big[V_{inf}\, cos(\delta_0 - \alpha)\big]\delta_\Delta \\ R_e\, I_{q\Delta} - \big(X_d^{'} + X_e\big)\, I_{d\Delta} &= E_q^{'} + \big[V_{inf}\, sin(\delta_0 - \alpha)\big]\, \delta_\Delta \end{aligned} \tag{2.73}$$

Equation 2.73 can be treated as a system of equations, where $I_{d\Delta}$ and $I_{q\Delta}$ are the variables to be solved for. Solving the set of equations and expressing the solution in matrix form,

$$\begin{bmatrix} I_{d\Delta} \\ I_{q\Delta} \end{bmatrix} = K_l \begin{bmatrix} -(X_e + X_q) & R_e\, cos(\delta_0 - \alpha) - \big(X_q + X_e\big)\, sin(\delta_0 - \alpha) \\ R_e & \big(X_d^{'} + X_e\big)\, cos(\delta_0 - \alpha) + R_e\, sin(\delta_0 - \alpha) \end{bmatrix} \begin{bmatrix} E_{q\Delta}^{'} \\ V_{inf}\, \delta_\Delta \end{bmatrix} \tag{2.74}$$

where the scaling factor K_l is defined by:

$$K_l = \frac{1}{R_e^2 + (X_q + X_e)(X_d^{'} + X_e)} \tag{2.75}$$

The expression for $I_{d\Delta}$ can be substituted into Equation 2.65, giving:

$$E_{fd\Delta} = \Big(\frac{1}{K_3} + T_{d0}^{'}\, s\Big)\, E_{q\Delta}^{'} + K_4\, \delta_\Delta \tag{2.76}$$

where the K-factors are defined as:

$$\begin{aligned} K_3 &= \frac{1}{1 + K_l\, \big(X_d - X_d^{'}\big)\big(X_q + X_e\big)} \\ K_4 &= V_{inf}\, K_l\, \big(X_d - X_d^{'}\big)\Big[\big(X_q + X_e\big)\, sin(\delta_0 - \alpha) - R_e\, cos(\delta_0 - \alpha)\Big] \end{aligned} \tag{2.77}$$

Subsequently, combining Equations 2.76 and 2.77 and keeping in the s-domain, the impact on $E_{q\Delta}^{'}$ by changes in $E_{fd\Delta}$ and δ_Δ is given by:

$$E_{q\Delta}^{'} = \frac{K_3}{1 + K_3\, T_{d0}^{'}\, s}\, E_{fd\Delta} - \frac{K_3\, K_4}{1 + K_3\, T_{d0}^{'}\, s}\, \delta_\Delta \tag{2.78}$$

2.3.2.2 Torque equation

With the chosen per-unit system the (per-unit) electrical torque is numerically equal to the three-phase power,

$$T_e = \frac{1}{3}\big(v_d\, i_d + v_q\, i_q\big) = V_d\, I_d + V_q\, I_q \tag{2.79}$$

From Equation 2.35, the assumptions made initially and by inserting from Equation 2.60,

$$\begin{aligned} V_d &= -X_q\, I_q \\ V_q &= X_d\, I_d + \frac{\omega_R\, L_{ad}}{\sqrt{3}}\, i_f = X_d^{'}\, I_d + E_q^{'} \end{aligned} \tag{2.80}$$

The simplified electrical torque expression can therefore be rearranged to:

$$T_e = \big[E_q^{'} - (X_q - X_d^{'})\, I_d\big]\, I_q \tag{2.81}$$

This equation can be linearised using the procedure described in Section 2.3.1. The linearised equation becomes:

$$\begin{aligned} T_{e\Delta} &= I_{q0}\, E^{'}_{q\Delta} + \left[E^{'}_{q0} - (X_q - X^{'}_d)\, I_{d0}\right] I_{q\Delta} - \left(X_q - X^{'}_d\right) I_{q0}\, I_{d\Delta} \\ &= I_{q0}\, E^{'}_{q\Delta} + E_{qa0}\, I_{q\Delta} - \left(X_q - X^{'}_d\right) I_{q0}\, I_{d\Delta} \end{aligned} \tag{2.82}$$

where E_{qa0} and $E^{'}_{q0}$ have been defined as:

$$\begin{aligned} E_{qa0} &= E^{'}_{q0} - \left(X_q - X^{'}_d\right) I_{d0} \text{ and,} \\ E^{'}_{q0} &= V_{q0} - X_d\, I_{d0} \end{aligned} \tag{2.83}$$

Now, substituting the currents calculated in Equation 2.74 into Equation 2.82 it is possible to write:

$$T_{e\Delta} = K_1\, \delta_\Delta + K_2\, E^{'}_{q\Delta} \tag{2.84}$$

where the new K-constants are defined as:

$$\begin{aligned} K_1 =& K_l\, V_{inf} \Big[E_{qa0}\big(R_e\, sin(\delta_0 - \alpha) + (X^{'}_d + X_e)\, cos(\delta_0 - \alpha)\big) \\ &+ I_{q0}(X_q - X^{'}_d)\big((X_q + X_e)\, sin(\delta_0 - \alpha) - R_e\, cos(\delta_0 - \alpha)\big)\Big] \\ K_2 =& K_l \left[I_{q0}\big(R_e^2 + (X_q + X_e)^2\big) + E_{qa0}\, R_e \right] \end{aligned} \tag{2.85}$$

The swing equation is given in Equation 2.46. It can be linearised by two first-order equations:

$$\begin{aligned} 2H\, \dot{\omega}_\Delta &= T_{m\Delta} - T_{e\Delta} - K_D\, \dot{\delta}_\Delta = T_{m\Delta} - K_1\, \delta_\Delta + K_2\, E^{'}_{q\Delta} - K_D\, \omega_\Delta \\ \dot{\delta}_\Delta &= \omega_R\, \omega_\Delta \end{aligned} \tag{2.86}$$

where all terms are in per-unit except δ_Δ, which is in electrical radians.

2.3.2.3 Terminal voltage equation

When later introducing the voltage regulator to the linear model, it is necessary to have the terminal voltage calculated such that the AVR gets a proper input. The terminal voltage is given by:

$$V_{tk}^2 = V_d^2 + V_q^2 \tag{2.87}$$

Using the linearisation steps described in Section 2.3.1, it is possible to show that the terminal voltage can be linearised to:

$$V_{tk\Delta} = \frac{V_{d0}}{V_{t0}}\, V_{d\Delta} + \frac{V_{q0}}{V_{t0}}\, V_{q\Delta} \tag{2.88}$$

Substituting from Equation 2.80 gives:

$$V_{tk\Delta} = -\frac{V_{d0}}{V_{t0}}\, X_q\, I_{q\Delta} + \frac{V_{q0}}{V_{t0}}\left(X^{'}_d\, I_{d\Delta} + E^{'}_{q\Delta}\right) \tag{2.89}$$

Lastly, substituting for the currents calculated in Equation 2.74 gives the linearised terminal voltage:

$$V_{tk\Delta} = K_5\, \delta_\Delta + K_6\, E^{'}_{q\Delta} \tag{2.90}$$

Where the two new K-constants are defined as:

$$\begin{aligned} K_5 =& \frac{K_l\, V_{inf}\, X_d^{'}\, V_{q0}}{V_{t0}} \Big[R_e\, cos(\delta_0 - \alpha) - \left(X_q + X_e\right) sin(\delta_0 - \alpha) \Big] \\ & - \frac{K_l\, V_{inf}\, X_q\, V_{d0}}{V_{t0}} \Big[R_e\, sin(\delta_0 - \alpha) + \left(X_d^{'} + X_e\right) cos(\delta_0 - \alpha) \Big] \\ K_6 =& \frac{V_{q0}}{V_{t0}} \Big[1 - K_l\, X_d^{'} \left(X_q + X_e\right) \Big] - \frac{V_{d0}}{V_{t0}}\, K_l\, X_q\, R_e \end{aligned} \tag{2.91}$$

The linear terminal voltage has here been given a subscript k. This is because this voltage is not a state variable, as it can be defined explicitly as a function of other state variables. The state variable V_t is created when inserting the voltage transducer at the machine terminals. The transducer is usually represented as a simple lag function, sometimes with a small gain,

$$V_{t\Delta} = \frac{K_r}{1 + T_r\, s}\, V_{tk\Delta} = \frac{K_r}{1 + T_r\, s} \left(K_5\, \delta_\Delta + K_6\, E_{q\Delta}^{'}\right) \tag{2.92}$$

with K_r being the transducer gain and T_r its time constant.

2.3.2.4 AVR and PSS equations

The last part of the system that has not been linearised is the excitation system control. In this simplified linear model, the AVR is considered a constant gain, such that:

$$E_{fd} = V_{error}\, K_a \tag{2.93}$$

where K_a is the AVR gain and $V_{error} = V_{ref} - V_t + v_{pss}$. Linearising and expanding yields:

$$E_{fd\Delta} = K_a\, V_{ref\Delta} - K_a\, V_{t\Delta} + K_a\, v_{pss\Delta} \tag{2.94}$$

The PSS will be represented as a conventional PSS, as in Figure 2.9, with just a single lead/lag stage. The wash-out filter and the lead/lag stage both increase the order of the system by one, such that two new state variables must be defined. They are:

- v_1 - At the output of the wash-out filter
- v_{pss} - At the output of the lead/lag block, which is the PSS output

The linearised equations for the PSS then become:

$$\begin{aligned} v_{1\Delta} =& \frac{s\, T_w}{1 + s\, T_w}\, K_{pss}\, \omega_\Delta \\ v_{pss\Delta} =& \frac{1 + s\, T_1}{1 + s\, T_2}\, v_{1\Delta} \end{aligned} \tag{2.95}$$

2.3.3 State-space representation

The idea behind the state-space representation is that a dynamic system may be represented by a system with a finite number of first order ordinary differential equations (ODE) [35]. This is

very useful for stability analysis of dynamic systems. For linearised models, the general form of a state-space representation is,

$$\begin{aligned} \dot{\bar{x}}_\Delta &= \overline{A}\,\bar{x}_\Delta + \overline{B}\,\bar{u}_\Delta \\ \bar{y}_\Delta &= \overline{C}\,\bar{x}_\Delta + \overline{D}\,\bar{u}_\Delta \end{aligned} \tag{2.96}$$

where,

- $\bar{x}_\Delta$ is the vector of perturbed state variables
- $\bar{u}_\Delta$ is the vector of perturbed system inputs
- $\bar{y}_\Delta$ is the vector of system outputs
- $\overline{A}$ is the state matrix, giving the state variables' impact on themselves and each other when perturbed
- $\overline{B}$ is the input matrix, giving the inputs' impact on the state variables when perturbed
- $\overline{C}$ is the output matrix, giving the state variables' impact on the output y when perturbed. Often a state variable is an output. Its corresponding element in $\overline{C}$ will then be 1.
- $\overline{D}$ is the input feedforward matrix, giving the inputs' direct impact on the output when perturbed.

For analysing the stability of a linear system, the state matrix $\overline{A}$ contains all needed information, as the eigenvalues of $\overline{A}$ represent the modes of the system.

The previous subsection defined all the linearised state variables and their expressions. The model ended up as a sixth-order model, where the state variables are:

- ω_Δ - Speed deviation of the rotor
- δ_Δ - Angle deviation of the rotor
- $E'_{q\Delta}$ - Internal emf of the machine
- $V_{t\Delta}$ - Terminal voltage
- $v_{1\Delta}$ - PSS wash-out filter output
- $v_{pss\Delta}$ - PSS output

with the inputs,

- V_{ref} - Reference voltage for the AVR
- P_m - Mechanical torque from the turbine system

To achieve a state-space representation the state variables must be represented as first-order differential equations that are functions of other state variables and inputs.

2.3.3.1 Speed and angle deviation

The equation for the linear speed deviation is given in Equation 2.86 and rearranges to:

$$\dot{\omega}_\Delta = \frac{1}{2H}\left(T_{m\Delta} - K_1\,\delta_\Delta + K_2\,E'_{q\Delta} - K_D\,\omega_\Delta\right) \tag{2.97}$$

This equation is already in a first-order ODE form, so no more work needs to be done for it. Replacing with state matrix notation (the a-coefficients will be stated explicitly later),

$$\dot{\omega}_\Delta = a_{11}\,\omega_\Delta + a_{12}\,\delta_\Delta + a_{13}\,E'_{q\Delta} + b_{11}\,T_{m\Delta} \tag{2.98}$$

The same goes for the angle deviation, also given in Equation 2.86,

$$\begin{aligned} \dot{\delta}_\Delta =& \omega_R\,\omega_\Delta \\ \Rightarrow \dot{\delta}_\Delta =& a_{21}\,\omega_\Delta \end{aligned} \tag{2.99}$$

2.3.3.2 Internal emf

The linear internal emf is given by Equation 2.78. In block diagram form it can be represented as in Figure 2.11, where $V_{e\Delta} = E_{fd\Delta} - K_4\,\delta_\Delta$ is the input of the field dynamics block.

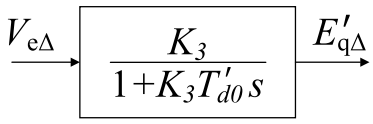


Figure 2.11: Block diagram for the internal emf transfer function. The input is the AVR output subtracted by the load-angle feedback loop, and its output is the internal emf of the machine.

The input can be expressed as:

$$V_{e\Delta} = E'_{q\Delta}\,\frac{1}{K_3}\,(1 + K_3\,T'_{d0}\,s) \tag{2.100}$$

It is desired to express the internal emf as a first-order equation. Recalling that s is the Laplace transform variable and represents a time derivative, the following can be derived:

$$\begin{aligned} V_{e\Delta} =& \frac{1}{K_3}\,E'_{q\Delta} + T'_{d0}\,\dot{E}'_{q\Delta} \\ \Rightarrow \dot{E}'_{q\Delta} =& \frac{1}{T'_{d0}}\,E_{fd\Delta} - \frac{K_4}{T'_{d0}}\,\delta_\Delta - \frac{1}{K_3\,T'_{d0}}\,E'_{q\Delta} \end{aligned} \tag{2.101}$$

Substituting $E_{fd\Delta}$ from Equation 2.94,

$$\dot{E}'_{q\Delta} = \frac{K_a}{T'_{d0}}\,V_{ref\Delta} - \frac{K_a}{T'_{d0}}\,V_{t\Delta} + \frac{K_a}{T'_{d0}}\,v_{pss\Delta} - \frac{K_4}{T'_{d0}}\,\delta_\Delta - \frac{1}{K_3\,T'_{d0}}\,E'_{q\Delta} \tag{2.102}$$

Now the expression of the internal emf is in the desired form, as it is a first-order ODE with respect to only state variables and inputs. Replacing with state matrix notation,

$$\dot{E}'_{q\Delta} = a_{32}\,\delta_\Delta + a_{33}\,E'_{q\Delta} + a_{34}\,V_{t\Delta} + a_{36}\,v_{pss} + b_{32}\,V_{ref\Delta} \tag{2.103}$$

2.3.3.3 Terminal voltage

The linear terminal voltage is given by Equation 2.92, and can be represented as in Figure 2.12.

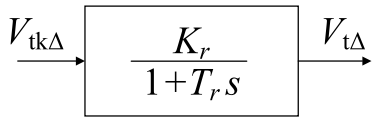


Figure 2.12: Block diagram for the terminal voltage transducer transfer function. The input is the calculated terminal voltage, and the output is the measured voltage with a time delay.

Similarly to Equation 2.100, the first-order equation can be found by deriving:

$$\begin{aligned} V_{tk\Delta} =& V_{t\Delta} \frac{1}{K_r} \left(1 + T_r\, s\right) = \frac{1}{K_r}\, V_{t\Delta} + \frac{T_r}{K_r}\, \dot{V}_{t\Delta} \\ \Rightarrow \dot{V}_{t\Delta} =& \frac{K_r}{T_r} \left[V_{tk\Delta} - \frac{1}{K_r}\, V_{t\Delta}\right] = \frac{K_r}{T_r} \left[K_5\, \delta_\Delta + K_6\, E'_{q\Delta} - \frac{1}{K_r}\, V_{t\Delta}\right] \\ \Rightarrow \dot{V}_{t\Delta} =& \frac{K_r\, K_5}{T_r}\, \delta_\Delta + \frac{K_r\, K_6}{T_r}\, E'_{q\Delta} - \frac{1}{T_r}\, V_{t\Delta} \end{aligned} \tag{2.104}$$

Replacing with state matrix notation,

$$\dot{V}_{t\Delta} = a_{42}\, \delta_\Delta + a_{43}\, E'_{q\Delta} + a_{44}\, V_{t\Delta} \tag{2.105}$$

2.3.3.4 PSS internal voltage v_1

The linear expression for the voltage v_1 is given in Equation 2.95 and represents the effect of the wash-out filter. The filter has a block diagram as shown in Figure 2.9. The input ω_Δ can be expressed as:

$$\omega_\Delta = \frac{1}{K_{pss}} \frac{1 + s\, T_w}{s\, T_w}\, v_{1\Delta} = \frac{1}{K_{pss}} \Big(\frac{1}{s\, T_w}\, v_{1\Delta} + v_{1\Delta} \Big) \tag{2.106}$$

Multiplying throughout with $s\, T_w$,

$$\begin{aligned} T_w\, \dot{\omega}_\Delta =& \frac{1}{K_{pss}}\, v_{1\Delta} + \frac{T_w}{K_{pss}}\, \dot{v}_{1\Delta} \\ \Rightarrow \dot{v}_{1\Delta} =& -\frac{1}{T_w}\, v_{1\Delta} + K_{pss}\, \dot{\omega}_\Delta \end{aligned} \tag{2.107}$$

Substituting from Equation 2.98,

$$\dot{v}_{1\Delta} = -\frac{1}{T_w}\, v_{1\Delta} + K_{pss} \left[a_{11}\, \omega_\Delta + a_{12}\, \delta_\Delta + a_{13}\, E'_{q\Delta} + b_{11}\, T_{m\Delta}\right] \tag{2.108}$$

Expanding and replacing with state matrix notation,

$$\dot{v}_{1\Delta} = a_{51}\, \omega_\Delta + a_{52}\, \delta_\Delta + a_{53}\, E'_{q\Delta} + a_{55}\, v_{1\Delta} + b_{51}\, T_{m\Delta} \tag{2.109}$$

2.3.3.5 PSS output voltage v_{pss}

The linear expression for the voltage $v_{pss\Delta}$ is given in Equation 2.95 and represents the PSS' lead/lag stage. Its block diagram can be seen in Figure 2.9. The input $v_{1\Delta}$ can be expressed

as:

$$\begin{aligned} v_{1\Delta} = \frac{1+T_2\,s}{1+T_1\,s}\,v_{pss\Delta} =& \frac{1}{1+T_1\,s}\,v_{pss\Delta} + \frac{T_2}{1+T_1\,s}\,\dot{v}_{pss\Delta} \\ \Rightarrow v_{1\Delta}\,(1+T_1\,s) =& v_{pss\Delta} + T_2\,\dot{v}_{pss\Delta} \\ \Rightarrow \dot{v}_{pss\Delta} = \frac{1}{T_2}\,v_{1\Delta} &- \frac{1}{T_2}\,v_{pss\Delta} + \frac{T_1}{T_2}\,\dot{v}_{1\Delta} \end{aligned} \tag{2.110}$$

Substituting from Equation 2.109,

$$\dot{v}_{pss\Delta} = \big(\frac{1}{T_2} + \frac{T_1}{T_2}\,a_{55}\big)\,v_{1\Delta} - \frac{1}{T_2}\,v_{pss\Delta} + \frac{T_1}{T_2}\left[a_{51}\,\omega_\Delta + a_{52}\,\delta_\Delta + a_{53}\,E'_{q\Delta} + b_{51}\,T_{m\Delta}\right] \tag{2.111}$$

Expanding and replacing with state matrix notation,

$$\dot{v}_{pss\Delta} = a_{61}\,\omega_\Delta + a_{62}\,\delta_\Delta + a_{63}\,E'_{q\Delta} + a_{65}\,v_{1\Delta} + a_{66}\,v_{pss\Delta} + b_{61}\,T_{m\Delta} \tag{2.112}$$

2.3.3.6 State matrix

Now, from the previous subsections, all the first-order equations have been defined. Thus, the state-space equations of the linear system can be set up. In Equation 2.113 the full system is given in the state-space form $\dot{x} = Ax + Bu$.

$$\begin{bmatrix} \dot{\omega}_\Delta \\ \dot{\delta}_\Delta \\ \dot{E}'_{q\Delta} \\ \dot{V}_{t\Delta} \\ \dot{v}_{1\Delta} \\ \dot{v}_{pss\Delta} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 & a_{36} \\ 0 & a_{42} & a_{43} & a_{44} & 0 & 0 \\ a_{51} & a_{52} & a_{53} & 0 & a_{55} & 0 \\ a_{61} & a_{62} & a_{63} & 0 & a_{65} & a_{66} \end{bmatrix} \begin{bmatrix} \omega_\Delta \\ \delta_\Delta \\ E'_{q\Delta} \\ V_{t\Delta} \\ v_{1\Delta} \\ v_{pss\Delta} \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ 0 & 0 \\ 0 & b_{32} \\ 0 & 0 \\ b_{51} & 0 \\ b_{61} & 0 \end{bmatrix} \begin{bmatrix} T_{m\Delta} \\ V_{ref\Delta} \end{bmatrix} \tag{2.113}$$

where,

$a_{11} = -\frac{K_D}{2H},\ \ a_{12} = -\frac{K_1}{2H},\ \ a_{13} = -\frac{K_2}{2H},$

$a_{21} = \omega_R,$

$a_{32} = -\frac{K_4}{T'_{d0}},\ \ a_{33} = -\frac{1}{K_3\,T'_{d0}},\ \ a_{34} = -\frac{K_a}{T'_{d0}},\ \ a_{36} = \frac{K_a}{T'_{d0}},$

$a_{42} = \frac{K_r\,K_5}{T_r},\ \ a_{43} = \frac{K_r\,K_6}{T_r},\ \ a_{44} = -\frac{1}{T_r},$

$a_{51} = K_{pss}\,a_{11},\ \ a_{52} = K_{pss}\,a_{12},\ \ a_{53} = K_{pss}\,a_{13},\ \ a_{55} = -\frac{1}{T_w},$

$a_{61} = \frac{T_1}{T_2}\,a_{51},\ \ a_{62} = \frac{T_1}{T_2}\,a_{52},\ \ a_{63} = \frac{T_1}{T_2}\,a_{53},\ \ a_{65} = (\frac{1}{T_2} + \frac{T_1}{T_2}\,a_{55}),\ \ a_{66} = -\frac{1}{T_2},$

and,

$b_{11} = \frac{1}{2H},\ \ b_{51} = K_{pss}\,b_{11},\ \ b_{61} = \frac{T_1}{T_2}\,b_{51},\ \ b_{32} = \frac{K_a}{T'_{d0}}$

It is also possible to draw a block diagram of the linearised model. This is done in Figure 2.13 and it gives a helpful visual interpretation of the system. The state variables have been highlighted in the diagram.

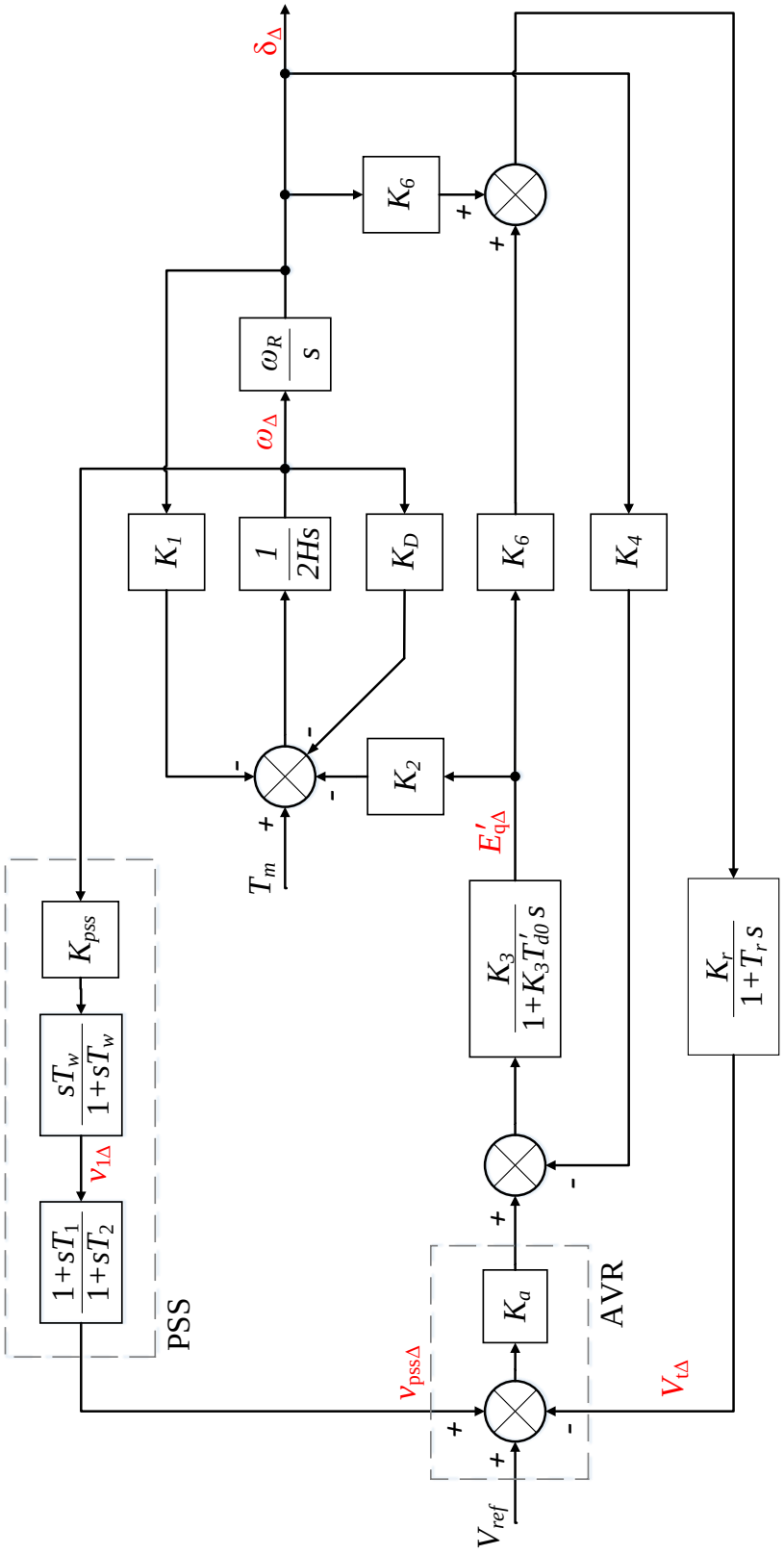


Figure 2.13: Complete block diagram of the simplified sixth-order linear synchronous machine model with the state variables highlighted in red

2.3.4 Initial conditions

The definitions of the constants $K_1 - K_6$ from Equations 2.77, 2.85 and 2.91 are notably dependent on certain initial conditions. This dependency comes from the fact that the linearisation assumes a small perturbation from an initial position. Thus, the initial position must be defined. It is assumed that the machine is in steady-state when a disturbance happens. This is a fair assumption, as a stable system will most likely reach steady-state before a new disturbance occurs. Therefore, the initial conditions for the linearisation will be the steady-state values for the machine. The necessary quantities are:

- initial d- and q-axis currents, I_{d0} and I_{q0},
- initial d- and q-axis terminal voltages, V_{d0} and V_{q0},
- initial load angle, δ_0
- initial internal emf, E_{q0}

From this point the subscript 0 will be omitted, as all quantities in this subsection are steady-state quantities. Using Figure 2.10 as a basis, some values must be known to calculate the others. Thus, a few assumptions are made:

- The infinite bus voltage is set as the reference, such that $\alpha = 0°$
- The generator is running at rated load, $P = 1\ pu$, and is over-excited at a known power factor $PF = cos(\phi)$
- The voltage at the infinite bus is $V_{inf} = 1\ pu$
- Power loss in the transmission line is approximated at 1 pu current, such that $P_e = R_e 1^2 = R_e$

A symbolic phasor diagram can be made for the machine-bus system, highlighting the important voltages, angles and currents. This is done in Figure 2.14. From the third assumption above, the power delivered to the infinite bus is:

$$P_{inf} = P - P_e = 1\ pu - R_e \tag{2.114}$$

Now, a decomposition of the complex current $\overline{I}_a$ is defined such that $\overline{I}_a = I_r + j\ I_x$, where I_r is in phase with the infinite bus voltage. Thus,

$$I_r = \frac{P_{inf}}{V_{inf}} = \frac{P_{inf}}{1}\ pu = P_{inf} \tag{2.115}$$

From the phasor diagram in Figure 2.14 the following angle relationships can be extracted [37]:

$$\begin{aligned} tan(\theta) =& \frac{I_x}{I_r} \\ tan(\beta) =& \frac{X_e\ I_r + R_e\ I_x}{V_{inf} - X_e\ I_x + R_e\ I_r} \\ tan(\phi) =& tan\big[cos^{-1}(PF)\big] \end{aligned} \tag{2.116}$$

Knowing that $\phi = \beta + \theta$ and consequently $tan(\phi) = tan(\beta + \theta)$, the trigonometric identity for phase shifted tangent terms can be used to set up an equation to find I_x. However, in an

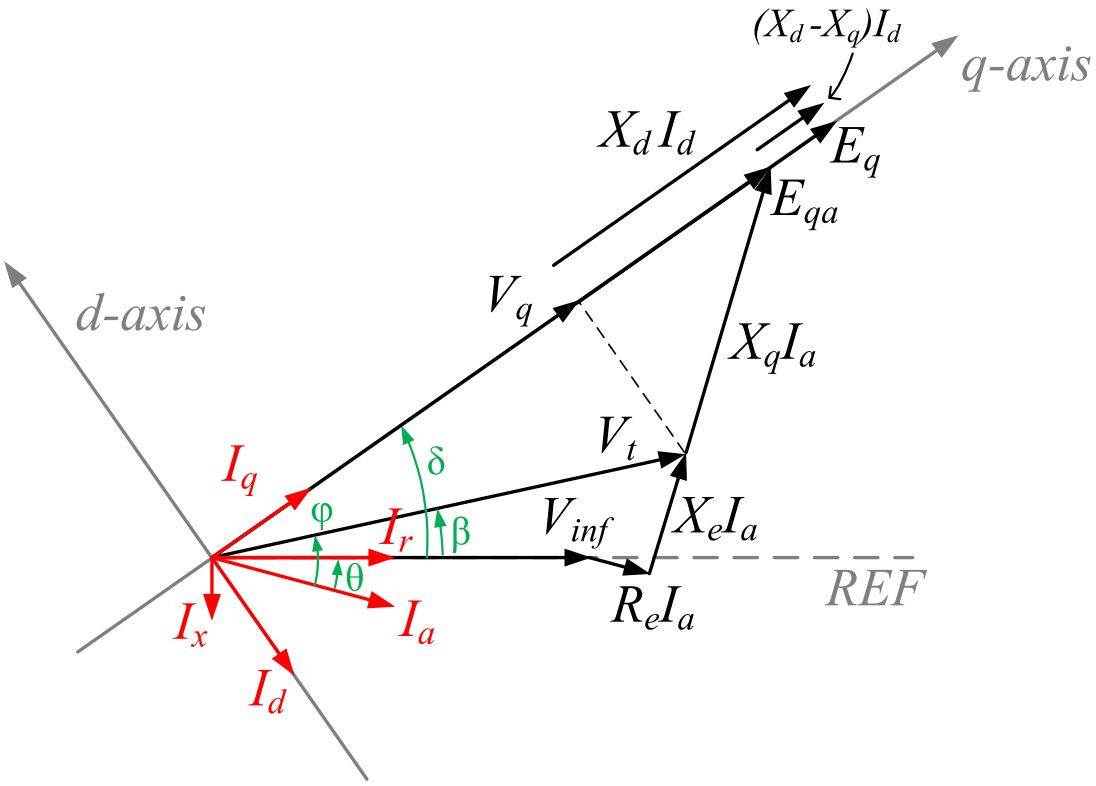


Figure 2.14: Phasor diagram of the machine-infinite bus system for the calculation of initial conditions

over-excited generator the current angle θ is negative, such that $tan(\phi) = tan(\beta - \theta)$. The identity then gives:

$$tan(\phi) = tan(\beta - \theta) = \frac{tan(\beta) - tan(\theta)}{1 + tan(\beta)\, tan(\theta)} \tag{2.117}$$

Inserting from Equation 2.116 gives:

$$tan(\phi) = \frac{\frac{X_e\, I_r + R_e\, I_x}{V_{inf} - X_e\, I_x + R_e\, I_r} - \frac{I_x}{I_r}}{1 + \frac{X_e\, I_r + R_e\, I_x}{V_{inf} - X_e\, I_x + R_e\, I_r}\, \frac{I_x}{I_r}} \tag{2.118}$$

Equation 2.118 can be used to solve for I_x, since all other quantities are known. The symbolic solution is not shown explicitly here, as it is too long to be helpful. From the now known quantities it is possible to calculate the load angle δ, which is the angle from the reference (infinite bus voltage) to the q-axis. Inspecting Figure 2.14 gives the following relation:

$$\begin{aligned} E_{qa} =& V_{inf} + \left[R_e + j(X_q + X_e)\right] \overline{I}_a = V_{inf} + \left[R_e + j(X_q + X_e)\right](I_r + j\, I_x) \\ =& V_{inf} - (X_q + X_e)\, I_x + R_e\, I_r + j\left[(X_q + X_e)\, I_r + R_e\, I_x\right] \end{aligned} \tag{2.119}$$

Here, the real terms are aligned with the reference and the imaginary terms are 90 degrees ahead. Since E_{qa} is on the q-axis, the angle δ can be found by:

$$\delta = tan^{-1}\left(\frac{(X_q + X_e)\, I_r + R_e\, I_x}{V_{inf} - (X_q + X_e)\, I_x + R_e\, I_r}\right) \tag{2.120}$$

This value of δ is what is used as the initial load angle δ_0. Now, the d- and q-axis currents and voltages can be found,

$$\begin{aligned} I_q &= |I_a| \ cos(\delta - \beta + \phi) \\ I_d &= -|I_a| \ sin(\delta - \beta + \phi) \\ V_q &= |I_a| \ cos(\delta - \beta) \\ V_d &= -|I_a| \ sin(\delta - \beta) \end{aligned} \tag{2.121}$$

where the angles β and ϕ are found from Equation 2.116. Lastly, the internal emf E_q is found by using Equation 2.83 at steady-state,

$$E_q = V_q - X_d \, I_d \tag{2.122}$$

2.4 Performance assessment

It is helpful to gain an understanding of what "better stability" really means. In stability analyses, there are two main approaches: time domain and frequency domain analyses. In time-domain studies, one observes how the signal changes over time and how quickly it does or does not return to steady-state. Three major concepts describe a dynamic response in the time domain: rise time, overshoot and settling time, see Figure 2.15. Rise time gives the time the signal takes to rise from 10 % to 90 % of the post-disturbance steady-state value. Overshoot describes how much the signal surpasses the steady-state value on the first swing. Settling time denotes the time before the signal settles within predetermined boundaries, commonly ±2-5 % of the steady-state value.

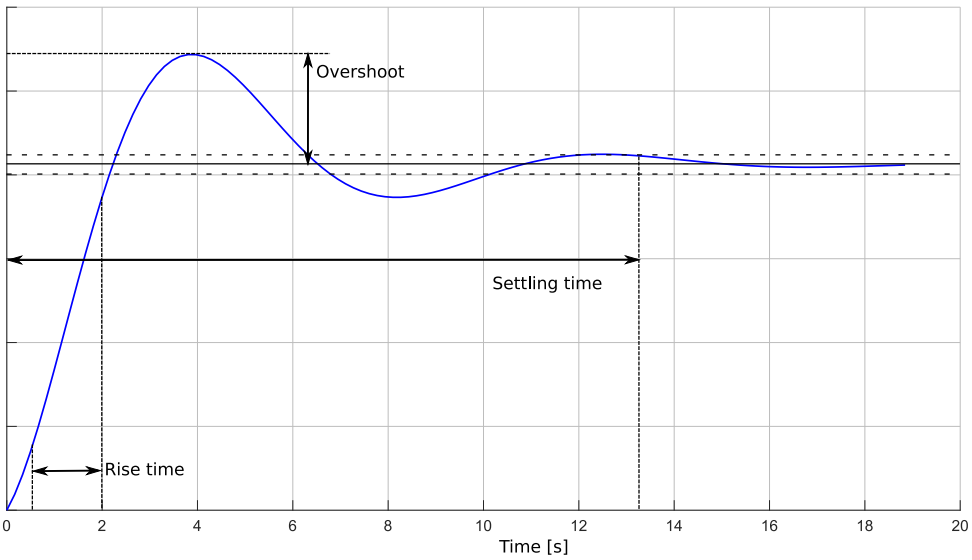


Figure 2.15: Stability analysis terms in the time domain, showing the definitions of rise time, overshoot and settling time

Frequency domain analysis can give a very intuitive understanding of complex stability problems. Central terms are frequency response (bode plot) and eigenvalue (modal) analysis. In power system stability studies modal analysis is often applied to find the various oscillations in the power system. Put simply, a mode corresponds to an eigenvalue of the state matrix which impacts the system stability with the time characteristic $e^{\lambda t}$, where λ is the eigenvalue [35].

Thus, a complex eigenvalue corresponds to an oscillatory mode. Say $\lambda = \sigma + j\omega$. The real part σ represents the damping power of the mode, and the imaginary part ω represents the frequency of oscillation. Thus, if $\sigma > 0$ the mode is unstable and will grow to infinity over time, and $\sigma < 0$ gives a stable mode. Also, if $\omega = 0$, the mode is non-oscillatory. A common measure of a mode's stability is its damping factor, which is defined as:

$$\xi = \frac{-\sigma}{\sqrt{\sigma^2 + \omega^2}} \tag{2.123}$$

A large value of ξ represents a highly damped mode, while a negative ξ implies the mode is unstable.

Chapter 3

Algorithms for optimisation

> *"This algorithm belongs ideologically to that philosophical school that allows wisdom to emerge rather than trying to impose it, that emulates nature rather than trying to control it, and that seeks to make things simpler rather than more complex. Once again nature has provided us with a technique for processing information that is at once elegant and versatile."*
>
> – Kennedy and Eberhart [47]

Optimisation is a well-established field of mathematics. There are countless ways of optimising a problem. However, many of the conventional methods fall short with highly non-linear problems. With the great increase in computational power over the last few decades, new methods have been developed that are very proficient at such problems. These methods are algorithms that are able to learn and adapt to specific problems and are called intelligent algorithms. Learning, in this context, is the algorithm's ability to adapt its behaviour to new input data and "remember" how historical data already has affected it. It observes the past and assumes the future behaves in the same way and is, therefore, able to predict the best-fitting output.

A common challenge when applying intelligent algorithms is defining the measure that determines its performance. This measure is called the objective function (or cost function, fitness function). Usually, in optimisation problems, there is a behaviour that is desired to be enhanced or diminished. However, quantifying this behaviour to a simple measure can be difficult. Still, defining the objective function is necessary for the algorithm to know whether its performance is improving or not.

Often in the optimisation of non-linear problems, there is more than one solution that is considered optimal. If a cost function is to be minimised the program may find a satisfying minimum within it. However, this may not be the globally optimal point. High-order problems often have more than one local minimum (or maximum) that may trap the algorithm. Thus, the algorithms must take measures to reduce the risk of getting stuck in local minima, such that the global optimum can be found. [1]

An interesting observation is that many intelligent algorithms are inspired by features of nature and animal behaviour. Evolution and natural selection are good examples of how nature has tried and failed and improved over time to end up with the complex structures and behaviours of living beings that exist today. [1]

The following sections will give insight into two well-established types of intelligent algorithms, namely the particle swarm optimisation technique and neural networks.

3.1 Particle swarm optimisation

The particle swarm optimisation algorithm (PSO) was originally introduced in 1995 [47], where the aim was partly to find ways to simulate human behaviour. The algorithm draws inspiration from the movement of flocking birds and schooling fish, as they seem to have some mysterious way of moving in synchrony to find food and avoid predators. PSO uses a set of particles (also called agents, individuals or candidates in the literature) that move in the search space according to simple rules to find the optimal position.

The dimension of the search space is equal to the number of parameters that are being optimised. This means that each position in the search space represents a valid set of parameters for the fitness function. Moving in a certain direction along a dimensional axis is equivalent to adjusting the respective parameter. For example, a two-variable fitness function $f(x, y)$ will have two dimensions in the search space, one representing x, and one y. Any particle's position $[x_i, y_i]$ in the search space can then be sent to the fitness function $f(x, y)$ to find its fitness.

The fitness function in a PSO problem can be any linear or non-linear function. The technique does not use the function gradient, so it can also be non-differentiable. The function needs to be called numerous times during the optimisation, so it is a great advantage if it is quick to calculate. The function's complexity will therefore directly affect the computation time and should be chosen with care.

There have been numerous variations on the PSO algorithms since its inception. This section will only cover one select variation, inspired by the Global Optimisation Toolbox [48]. For a more comprehensive review of various PSO modifications, one may refer to [49].

3.1.1 The elements of PSO

The great advantage of PSO is that it is very simple to understand and implement while remaining fast and effective. The main elements of the PSO algorithm are:

3.1.1.1 Particle position

The jth particle's position in the search space at time (iteration) t is denoted x_j^t. It is a multi-dimensional vector, where each dimension corresponds to an optimised parameter. The position is updated every iteration according to Equation 3.1, where v_j^t is the respective particle's velocity. The positions in each dimension are limited by pre-defined search boundaries.

$$x_j^t = x_j^{t-1} + v_j^t \tag{3.1}$$

3.1.1.2 Particle velocity

The jth particle's velocity at time t is denoted v_j^t. The velocity is a vector describing the trajectory of each particle. This is a key element, as applying correct velocities will increase the convergence time. There are three components to calculating the velocity: the inertial, individual and social components. It is determined by:

$$v_j^t = w^t\, v_j^{(t-1)} + c_1\, r_1 \left[p_{j,best}^{(t-1)} - x_j^{(t-1)}\right] + c_2\, r_2 \left[g_{best}^{(t-1)} - x_j^{(t-1)}\right] \tag{3.2}$$

where,

- w is the inertia weight. This is a control variable that adds a velocity component in the same direction as the previous time step. A large value of this component aids in the exploration of the search space, while a low value improves the local search ability of the particle. Thus, it is not uncommon in PSO applications to have a varying inertia weight, which starts large and gets decreased as the search approaches the optimum.
- c_1 and c_2 are control variables that affect the impact of the personal and global bests, respectively.
- r_1 and r_2 are random numbers between 0 and 1 drawn from a uniform distribution. These are updated each iteration.
- $p_{j,best}$ is the position with the best fitness the jth particle has found. Every iteration as the particle flies through the space it compares the new fitness with its previous best and updates it if it has improved.
- g_{best} is the position with the best fitness among all the personal best fitnesses. At the end of the PSO procedure, g_{best} will hold the optimised parameters.

The three velocity components (inertial, personal and social) are added vectorially, as visualised in Figure 3.1. Keep in mind that the velocities in each dimension are independent of each other, and the figure shows the combined velocities for the displayed dimensions.

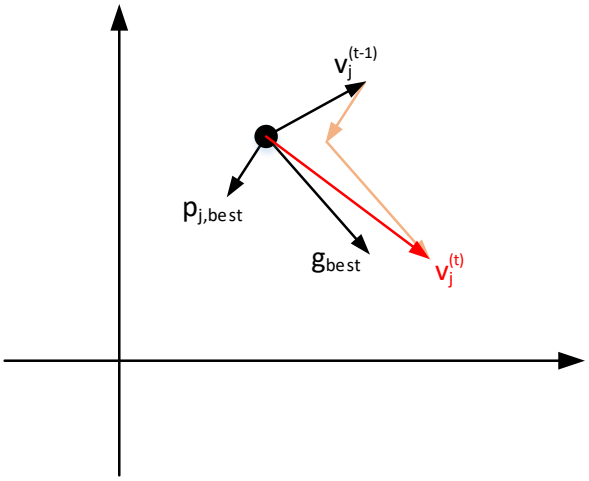


Figure 3.1: Illustration of a particle's velocity calculation in the PSO algorithm. The new velocity is a weighted sum of the previous velocity, the particle's personal best position and global best positions

3.1.1.3 Neighbours

In the PSO technique, an individual particle's neighbourhood is the subset of other particles it can communicate with [50]. There are different ways of defining the neighbourhoods, each giving different swarm behaviours. The simplest is the *gbest* type neighbourhood, where each

particle can communicate with every other particle. Their social components are then attracted to the global best solution across every particle.

It is possible to reduce the neighbourhood sizes using the *lbest* type, where the individuals' social components are attracted to the best solution among their local neighbourhood. The original PSO model calculated the distance between each particle to decide which were the closest neighbours. However, this was found to be inefficient. Another way is the ring model, where each particle is only communicating with two others. The particles will then have a slower convergence rate, but a lesser chance of getting stuck in local optima. This is very convenient for large multi-dimensional problems.

A large neighbourhood size yields a fast convergence towards the suggested optimum, which is the advantage of the *gbest* model. However, it has a tendency to prematurely converge, such that the solution may end in a suboptimal extremum. A population with small neighbourhood sizes does not have this problem. It is then logical to implement a dynamic neighbourhood size that increases over time. A low size at the beginning aids in the initial exploration of the search space. When the swarm has found appropriate optimal areas the neighbourhood sizes can be increased to speed the local convergence.

3.1.1.4 Stopping criteria

Performing a large number of iterations will give a more accurate result, though each additional iteration adds computational load. To avoid excessive consumption of time and resources appropriate stopping conditions are necessary. The simplest form of stopping condition is to set a fixed amount of iterations. The procedure will then stop no matter the result. Another common criterion is to set a stalling threshold. If the global best has not improved for a certain amount of iterations it can be assumed the optimum has been found, and the procedure can be stopped. Other stopping criteria might be a tolerance level for known targets or maximum computation time.

3.1.2 PSO procedure

The PSO procedure can be broken down into separate steps, as it is an iterative procedure. Figure 3.2 shows a flowchart for better visual understanding. As mentioned previously, there are several variations to the PSO procedure. The procedure described below follows that of the Global Optimisation Toolbox for MATLAB version 2020b [51].

Step 1 - initialisation
Before the iterative process can be started, all control variables and parameters must be declared and initialised. The particles are randomly placed in the search space within the bounds. The velocities in each dimension are randomly initiated between $-v_{k,max}$ and $v_{k,max}$. For the kth dimension, $v_{k,max}$ is determined by:

$$v_{k,max} = \frac{x_{k,max} - x_{k,min}}{N_v} \tag{3.3}$$

where $x_{k,max}$ and $x_{k,min}$ are the upper and lower boundaries in the kth dimension, respectively. N_v is the velocity interval number, where a large value will give greater diversity of velocities

in the initial step. This parameter can be chosen as any convenient value, though some care should be taken, as it may impact the PSO performance [5].

When the particles have been initialised, the initial fitness for each particle is calculated. That values are stored to their respective p_{best} parameters. Among all the personal bests, the one with the best fitness is stored to the g_{best} parameter.

The initial neighbourhood size is set to a predetermined minimum value, and the initial inertia weight is set to a predetermined maximum.

Step 2 - Create neighbours
For each particle, a random subset of all other particles is set as its neighbourhood. The size of the subset is equal to the current neighbourhood size, dependent on the search progress. The fitness and position of the best neighbour are stored to use in the velocity calculation.

Step 3 - Update velocities
The velocities of each particle in each dimension are calculated according to Equation 3.2. In this case, g_{best} is the position of the best neighbour.

Step 4 - Update positions and enforce bounds
The position of each particle is calculated according to Equation 3.1. If any particle position has breached the search boundary in a dimension, it is set equal to that boundary and its corresponding velocity component is forced to zero.

Step 5 - Calculate new fitnesses and update the bests
For each particle, the fitnesses at their new positions are calculated. If a particle's new fitness is better than the previous best, it (along with the position) is stored to the p_{best} variable. If any of the new personal bests are better than the current global best, the g_{best} is updated accordingly.

Step 6 - Update dynamic variables
In this specific procedure, the neighbourhood size is updated only when the optimisation is stalling, i.e. the global best is not getting improved. Each time a new global best is registered the size is reset to its initial value. For every consecutive stalled iteration the neighbourhood size is increased by some fixed number, ensuring it does not exceed the total population number. The growth number may be equal to the initial neighbourhood size.

Every iteration the inertia weight can either be increased, decreased or not changed at all. Which operation occurs depends on how many iterations the optimisation has been stalling. If the procedure is not stalling, the weight is increased each iteration, ensuring it does not exceed its maximum value. If the procedure has been stalling for a few iterations, say 3, the weight may "plateau," keeping it unchanged. After another few stalled iterations the weight is decreased to improve local search. It is decreased every subsequent stalled iteration until it reaches its minimum value, or an improved global best is registered.

Step 7 - Check stopping criteria and iterate
The stopping criteria may be a maximum amount of total or stalled iterations, as described earlier. If a stopping criterion is met the iteration stops and the procedure outputs the global best and its position. The best position coordinates contain the optimised parameters of the problem. It no stopping criteria is met the iteration count is increased by one and the procedure repeats, beginning at step 2.

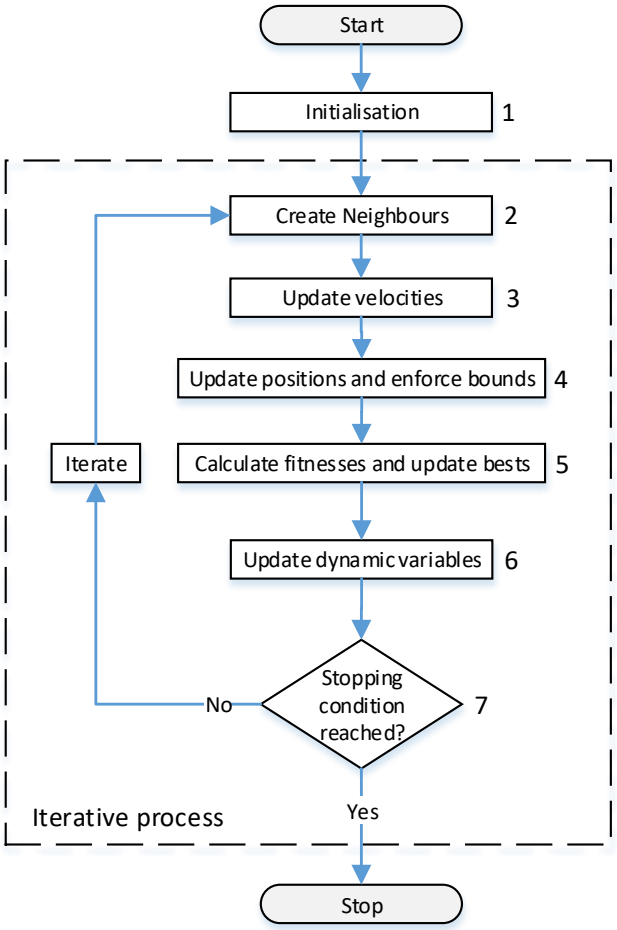


Figure 3.2: Flowchart of the PSO procedure

3.2 Neural networks

In the field of machine learning, one of the most well-known techniques is the neural network (NN). Its prominence comes from its ability to be relatively easy to understand while being very versatile. In human or animal brains a magnitude of single neurons communicate using electrical impulses. A neuron by itself provides no useful function, yet together, complex behaviour emerges. This is the phenomenon neural networks seek to emulate. By configuring a system of "neurons" in a particular way, each performing a simple operation, the desired behaviour can be created.

This section will introduce the concept of neural networks, to provide enough detail to be able to implement a simple NN on a pre-built platform and understand its inner workings. This is often enough for many engineering applications such as this thesis.

3.2.1 Structure

There are several different types of neural networks, and most types have their own subcategories [52]. However, this section will focus on the feedforward multilayer perceptron (MLP) model for its popularity and simplicity. The neurons in an MLP are arranged in three conceptual layers: the input, hidden and output layers. Its goal is to receive some input to the input layer and give the desired output at the output layer, be it a classification, pattern matching, optimisation or any other prediction. A visual representation of a three-layer MLP network is shown in Figure 3.3 [1].

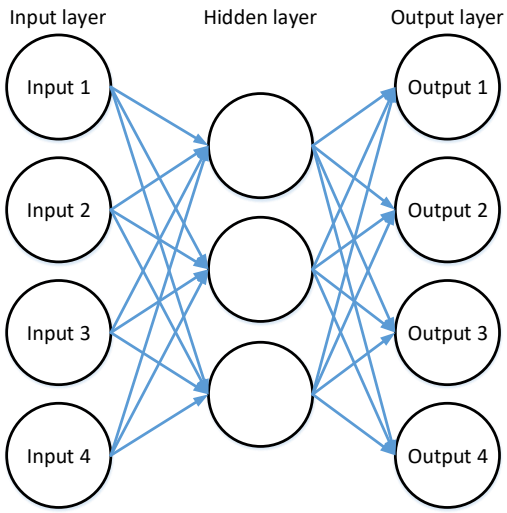


Figure 3.3: Illustration of the three-layer feedforward perceptron with four neurons in the input and output layers, and three neurons in the hidden layer.

This model only works when using supervised learning (SL). SL means that the NN is trained using labelled data. A labelled data point is an input paired with the desired output value. For example, for image recognition, the input might be the pixels of an image and the desired output might be a category, e.g. "dog" or "cat".

There are several internal parameters in an NN. Their values are fundamental to the performance of the network. Each connection between nodes has an associated *weight* and each node has an associated *bias*. The output of each node is a weighted sum of the outputs from the previous layer. The output of a neuron is also called its *activation*. Take Figure 3.4, where the jth node in the nth layer is depicted with three nodes in the previous layer. The output y_j^n can be expressed as:

$$y_j^n = w_{j1}\, a_1^{(n-1)} + w_{j2}\, a_2^{(n-1)} + w_{j3}\, a_3^{(n-1)} + b_j^n \tag{3.4}$$

However, to ensure the outputs of the different nodes all end in the same number ranges they are sent through an *activation function*, which remaps the output onto a consistent number range. Two common activation functions are the hyperbolic tangent (tanh) function and the rectified linear unit (ReLU) function, see Figure 3.5. The tanh function is given by Equation 3.5 and outputs -1 for highly negative numbers, 1 for highly positive numbers and a smooth transition when close to zero. The ReLU function is given by Equation 3.6. It outputs 0 for all negative

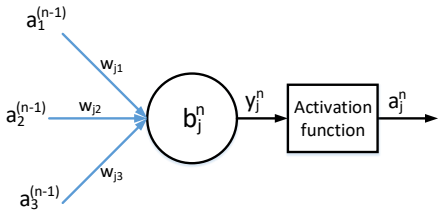


Figure 3.4: A single neuron with its parameters in a feedforward MLP neural network. The output a_j^n is a weighted sum of the activations of the previous layer with an added bias, sent through an activation function.

numbers and returns the input linearly for positive numbers.

$$tanh(y) = \frac{e^{2y} - 1}{e^{2y} + 1} \tag{3.5}$$

$$ReLU(y) = max(0, y) \tag{3.6}$$

Assuming the tanh function, the node activation would become:

$$a_j^n = tanh\big(y_j^n\big) = tanh\big(w_{j1}\, a_1^{(n-1)} + w_{j2}\, a_2^{(n-1)} + w_{j3}\, a_3^{(n-1)} + b_j^n\big) \tag{3.7}$$

The activation in the nth layer with j nodes can be expressed generally in matrix form,

$$A^n = tanh\big(W\, A^{(n-1)} + B^n\big) \tag{3.8}$$

where,

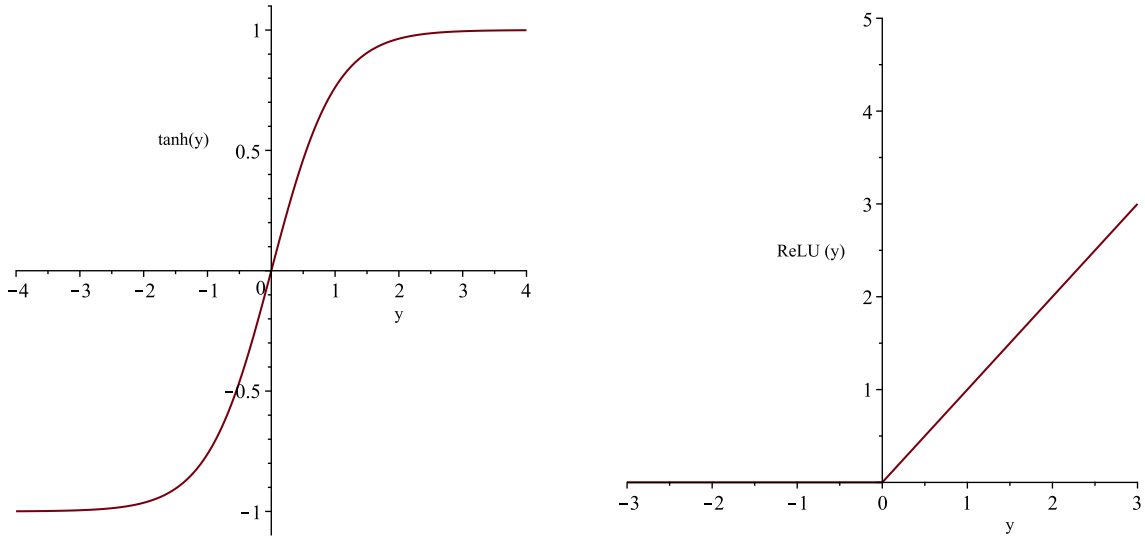
- A^n is a $jx1$ column vector with the activation of the nth layer
- W is a jxn matrix containing the connection weights
- B^n is a $jx1$ column vector with the biases at the nth layer.

Every neuron in a layer is connected to every neuron in the previous layer through a weight. Therefore, there can be an enormous amount of parameters in a NN to be tuned. The more neurons or layers, the larger the number of parameters. Thus, various assignments of these parameters can provide practically endless applications. The tuning of the inner NN parameters to achieve a specific behaviour is called training of the neural network.

3.2.2 Training

The statement that an MLP neural network can create emergent desired behaviour is true only so far as its training has been sufficient. When a NN is newly generated, its inner parameters (weights and biases) are pseudo-randomly initiated, thus providing no function of value. In supervised learning, training is done by creating input-target pairs that are fed to the network repeatedly until the network response to an input matches the corresponding target sufficiently. This indicates the first important step in neural network training: collecting and preprocessing training data.

A large amount of training data is recommended, often required, to achieve a properly trained network. The source of the data set can vary. It may be collected from pre-existing data sets



(a) The hyperbolic tangent (tanh) activation function **(b)** The rectified linear unit (ReLU) activation function

Figure 3.5: Two common activation functions for neural networks - the tanh and ReLU functions

published online, iteratively calculated from known algorithms, retrieved from simulation results or any other convenient source. Nonetheless, The training data set should be sufficiently large and encompass all scenarios the network is expected to handle. When the training data set is generated, it is randomised and split into three subsets: the training, validation and testing subsets. The training subset is what the network will be trained with. However, to confirm that the trained network can manage inputs not in the training set (yet still within the expected range of inputs) a subset of the data is reserved to the validation of the network. This is to avoid what is called overfitting. One might theoretically achieve 100 % accuracy with the training data, but that does not mean its accuracy to other inputs within the expected range will be 100 %. When the NN's accuracy to the training set consistently surpasses its accuracy to the validation set it is a sign of overfitting, and training should be terminated if continued. The testing subset is reserved purely for the testing of the trained NN and is not involved in the training itself.

Figure 3.6 shows the general training scheme of supervised learning. The NN output is compared to the target output. The error is represented by an error function, which by the means of gradient descent is minimised by a training algorithm. This is an iterative process, where an iteration has passed when the entire training data subset had been applied once. In NN theory, such an iteration is commonly called an epoch. The training of this type of neural network can be considered an optimisation problem, where all the network inner parameters are to be optimised. The error function can be considered the cost function of the optimisation. A very common error function is the mean square error (MSE) function, which is defined as:

$$E = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{3.9}$$

where E denotes the error function, n is the number of data points, y_i is the target output and $\hat{y}_i$ is the calculated output. The error function is dependent on all inner parameters, following Equation 3.8.

If the activation function is differentiable (which both tanh and ReLU are) the error function

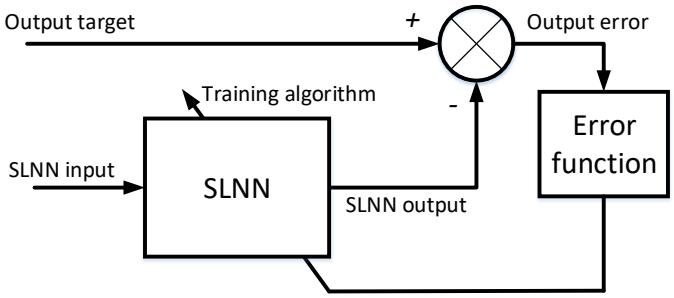


Figure 3.6: The supervised learning neural network training scheme. When giving the NN a controlled input, the difference of the target output and the NN output is sent through an error function and used to train the NN with the help of a training algorithm.

is also differentiable. The method of minimising E is, therefore, most commonly by using the gradient of steepest descend. Let θ^t denote all weights and biases at iteration t, making the error function $E(\theta^t)$. The parameters are updated according to:

$$\theta^{(t+1)} = \theta^t - \alpha \frac{\partial E(\theta^t)}{\partial \theta} \tag{3.10}$$

where α is called the learning rate, which is a control parameter determining how fast the network should "learn" each iteration.

The above procedure is a description of the back-propagation algorithm [53], which is one of the most used training algorithms. Some other common algorithms found in the literature are the quasi-Newton [54], conjugate gradient [55] and Levenberg-Marquardt [56, 57] algorithms. The main difference between these is how the gradient is used to update the weights and biases [58]. They are all trajectory-driven, meaning they use the gradient-descent method to minimise the output error. The details of the training algorithms will not be derived here, but a comprehensive review of the mentioned algorithms, along with other methods, can be found in [59].

Chapter 4

Auto-tuning the CPSS

> *"It is a challenge [...] to find a universal function which would be adequate for the whole spectrum of possibilities. By examining this spectrum of possibilities over a credible band of machine and system parameter values, as well as loading conditions, it appears that a fair compromise function can be recommended that will be almost universally applicable."*
>
> – Demello and Concordia [4]

The conventional power system stabiliser (CPSS) is the most popular stabiliser design. It is simple to understand and implement, and many power plants have this type of PSS already installed. Therefore, some way of automatically tuning the CPSS in online operation should be discussed before proposing redesigns that some operators may not want to switch to, for practical or economic reasons.

This chapter will propose a method for observing the changes in a connected power system to automatically tune the CPSS for small-signal stability. The proposed method will function without taking the PSS or machine out of operation. It involves implementing an auto-tuning neural network that is trained on optimised parameters obtained from the PSO technique. The approach is similar to that of [24], though the procedure of obtaining the training data is different. Figure 4.1 shows an overview of the proposed auto-tuning system for visual understanding. Further, it will be explained how this auto-tuning system can be implemented into the Simulink environment while accommodating for its limitations and minimising simulation time.

4.1 Optimising from linear model

A linear model of the synchronous machine is derived in Section 2.3, and is appropriate for small-signal stability studies. The dynamics of that model can be used to analyse its ability to damp oscillatory behaviour. See Appendix A for the model parameters used in this thesis.

4.1.1 Simplifications

As this section has the purpose of being introductory to the design, the linear model used in this chapter follows the same simplifications made in Section 2.3. Important simplifications are the AVR being considered a constant gain and the PSS only having a single lead/lag stage.

In the lead/lag stage, only the numerator time constant T_1 is tuned. Since the ratio between

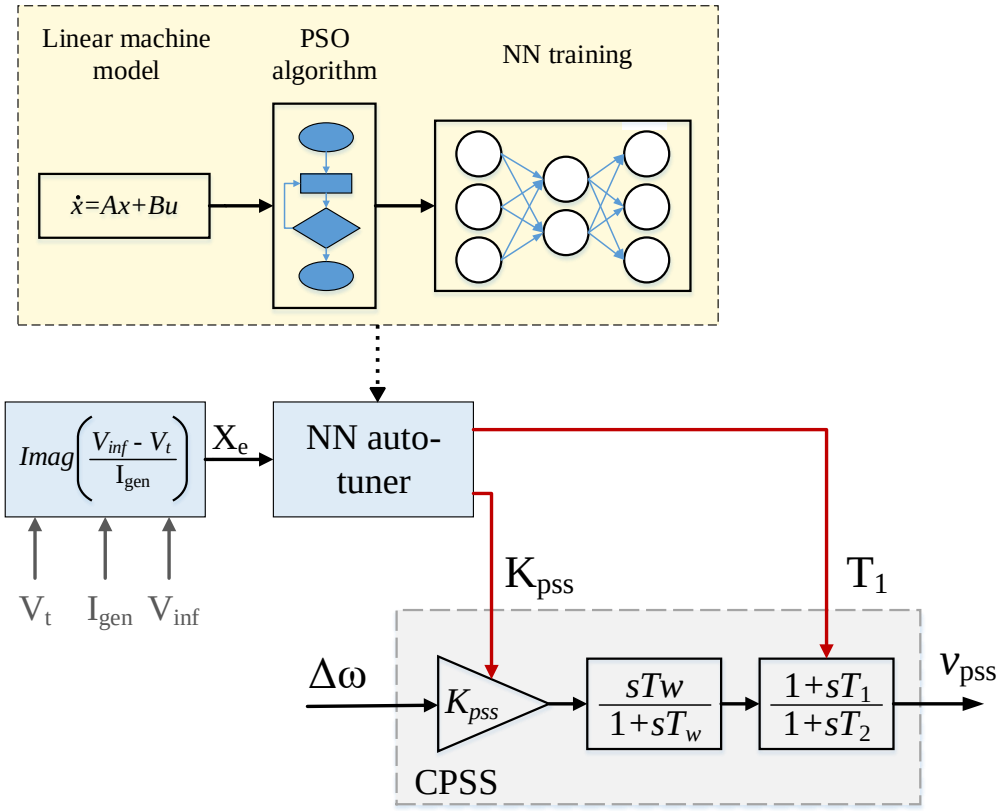


Figure 4.1: Overview illustration of the CPSS auto-tuning system. The PSO algorithm optimises the linear machine model, where the result is used to train the NN auto-tuner.

the two time constants decides the phase response of the block, keeping one constant and only tuning the other is sufficient. Additionally, the wash-out filter time constant T_w is kept constant, as it has little impact on the damping performance. Finally, the infinite bus voltage is considered constant, along with the external network resistance. This means that changes to the network short-circuit power only comes from changes in the external reactance X_e.

These simplifications leave only two parameters to be tuned: the PSS gain K_{pss} and the lead/lad numerator time constant T_1.

4.1.2 Building training data with PSO

Now that the linear model is ready, there needs to be a way of determining the optimal parameters for K_{pss} and T_1 at various levels of X_e. This may be a problem with many local minima, so the PSO algorithm is well fitted for this purpose. As a fitness function for the PSO, the comprehensive damping index (CDI) is used [6]. The CDI is calculated as:

$$CDI = \sum_{i=1}^{n} (1 - \zeta_i) \tag{4.1}$$

where ζ_i is the damping ratio and n is the number of oscillatory eigenvalues. Minimising this index will give the parameters where the oscillation damping of the linear model is greatest. The damping ratios of the system modes are found by eigenvalue analysis of the state matrix of

the linear model in Equation 2.113. The advantage of using eigenvalue analysis as the fitness function for the PSO is its speed. PSO uses a great number of function evaluations – one for each particle every iteration – so having a quick fitness function is important for the optimising speed. One could conceivably use high-accuracy simulation results as the fitness function, though this would take a considerable amount of time.

A range of values for X_e is chosen and the PSO algorithm is run for each of them, optimising K_{pss} and T_1. Per-unit reactances from 0.001 to 0.1 has been used for this chapter, as these are typical values. See Appendix A for the PSO settings used in these optimisations.

4.1.3 Auto-tuning neural network

When the PSO procedures are finished, the training data set is ready. The NN input is the per-unit external reactance and the target (desired output) is the corresponding optimised PSS parameters. This can be classified as a function fitting problem, which neural networks are very proficient at. In fact, the *Universal Approximation Theorem* states that a multilayer feedforward network with a single hidden layer is able to represent any function, given some assumptions to the activation function and sufficient hidden neurons [60]. Also, no advanced function fitting theory is required, as the NN training algorithm efficiently finds the appropriate parameters for the fit.

The Deep Learning Toolbox [61] contains easy-to-use functions for shallow neural network creation and training. The `fitnet` function allows for training of function fitting NNs with little effort. When the training tool has finished training the network, a Simulink model is created by the `gensim` function. The external reactance can then be fed directly into the network, and the optimal PSS parameters will be output continuously. See Appendix B for how to use the `fitnet` function in MATLAB.

It could be argued that the PSO algorithm can be put directly to the auto-tuning model, forgoing the NN completely. While this is possible, optimising with PSO on a continuous basis is very slow and computationally heavy. Since a trained MLP neural network is practically just a weighted sum of the neuron activations, computing the output is a simple operation, taking virtually no time. Hence, it is beneficial to implement a NN rather than just using the PSO technique.

4.1.4 Calculating the external reactance

The input to the NN is the network external reactance, so this must be calculated. The external network is considered as an infinite bus in series with an impedance, similar to Figure 2.10. Here it is assumed that the infinite bus voltage is known and can be measured. The complex external impedance is then calculated by:

$$Z_e = \frac{\overline{V}_{inf} - \overline{V}_t}{\overline{I}_{gen}} \tag{4.2}$$

where,

- $\overline{V}_{inf}$ is the complex voltage at the infinite bus,
- $\overline{V}_t$ is the complex voltage at the generator terminals,

- $\overline{I}_{gen}$ is the complex generator stator current, which is equal to the infinite bus current

The external reactance is found by the imaginary component of the impedance,

$$X_e = imag(Z_e) \tag{4.3}$$

The per-unit value is found by either dividing the input voltages and currents by their respective bases V_B and I_B, or by dividing the calculated reactance by the impedance base Z_B.

4.2 Implementation into Simulink

When the theory is understood, it needs to be set into practice. However, this may not be such an easy task. Firstly, one must choose in which environment the implementation should happen. It could be done with a real apparatus or through computational simulation. When a simulation is to be performed an appropriate software must be chosen. Then, the models need to be implemented properly while accounting for software limitations and restrictions.

In this thesis, simulations are done through the MATLAB/Simulink environment, with the Simscape Power Systems add-on. This choice is mainly due to their already existing library of functions and models which can easily be applied. Still, Simulink has its own set of limitations. Its pre-existing model blocks have so-called "masks," which are user interfaces where the necessary parameters for the model are entered. The mask also has the signal inputs and outputs easily accessible. Still, the model beneath the mask is locked, meaning it cannot be edited without rebuilding it manually, and all the operations the block performs are not always accessible to the user. Additionally, the masked block parameters cannot be edited dynamically during a simulation; they are locked to their initial values.

The main block of the simulation is the synchronous machine. The *Synchronous Machine pu Standard* block is used for this purpose. It follows the same theory derived in Section 2.1, and gives the dynamic behaviour of the machine using per-unit quantities. The block has two inputs: the mechanical torque and field voltage, both in per-unit. Its outputs are the three-phase lines and various measurements for its internal quantities.

The following subsections will introduce how the transformer, excitation system and external system can be implemented into Simulink while accommodating for its limitations.

4.2.1 Step-up transformer

Experience shows that when introducing an inductance into Simulink, the simulation time increases significantly. Also when using a transformer block does this problem occur. To alleviate this, the transformer per-unit reactance X_t can be put into the synchronous machine model reactances. The transformer resistance R_t is neglected. The new machine parameters then

become: [35]

$$\begin{aligned}
\hat{X}_d &= X_d + X_t \\
\hat{X}_d^{'} &= X_d^{'} + X_t \\
\hat{X}_d^{''} &= X_d^{''} + X_t \\
\hat{X}_q &= X_q + X_t \\
\hat{X}_q^{''} &= X_q^{''} + X_t \\
\hat{X}_l &= X_l + X_t
\end{aligned} \tag{4.4}$$

The parameters with the hat notation represent the values that are given to the Simulink model, while those without the hat notation are the actual machine parameters. The consequence of this adjustment is that the generator model now outputs the voltage at the high-voltage side of the transformer. The generator can in this state be comparable to a powerformer, as it is a generator directly connected to the high-voltage system [62]. However, as mentioned in Section 2.2.2, the control voltage for the AVR is commonly around 80 % into the transformer reactance. Thus, to have the desired input to the AVR. it might be necessary to calculate back to that voltage. If the AVR control voltage is set at 100 % into the transformer reactance, this is not necessary.

The synchronous machine d- and q-axis equivalent circuits are given in Figures 2.3 and 2.4 respectively. Recall that $L_d = L_{ad} + L_l$ and that $L = X$ in per-unit. Since X_t is added to both X_d and X_l the relation becomes:

$$\begin{aligned}
\hat{L}_d &= \hat{L}_{ad} + \hat{L}_l \\
\Rightarrow L_d + L_t &= \hat{L}_{ad} + L_l + L_t \\
\Rightarrow L_d &= \hat{L}_{ad} + L_l
\end{aligned} \tag{4.5}$$

Thus, $\hat{L}_{ad} = L_{ad}$ and does not need to be adjusted in the equivalent. The same argument can be made for L_{aq}. However, $\hat{L}_l$ has the transformer reactance included and needs to be accounted for. Additionally, the speed voltage term $\omega\Psi_q$ can be expanded into:

$$\omega\Psi_q = \omega L_q i_q \tag{4.6}$$

Hence, this term is dependent on L_q and consequently $\hat{L}_q$. Equation 4.7 shows how the adjusted speed voltage can be split into two terms. The same argument can be made for $\omega\hat{\Psi}_d$.

$$\omega\hat{\Psi}_q = \omega\hat{L}_q i_q = \omega(L_q + L_t)i_q = \omega L_q i_q + \omega L_t i_q = \omega\Psi_q + \omega\Psi_{qt} \tag{4.7}$$

The remaining inductances in the equivalent circuits represent either the field winding or the amortisseur windings. Neither of those have been affected by the implementation of the transformer, so all those can be ignored in the back-calculation. $\hat{L}_l$, $\omega\hat{\Psi}_q$ and $\omega\hat{\Psi}_d$ are now represented as a sum of two voltage terms. The equivalent circuits can then be expanded as shown in Figures 4.2 and 4.3. The voltages v_{dt} and v_{qt} are the voltages output by the generator model, while v_d and v_q are the voltages whose Pythagorean sum is the desired control voltage to the AVR. Applying Kirchhoff's voltage law to the circuits, the desired voltages can be found.

$$\begin{aligned}
v_d &= v_{dt} - \omega\Psi_{qt} + v_{L_t} \\
v_q &= v_{qt} + \omega\Psi_{dt} + v_{L_t}
\end{aligned} \tag{4.8}$$

Stating those equations in terms of known parameters,

$$\begin{aligned} v_d &= v_{dt} - \omega L_t i_q + L_t \dot{i}_d \\ v_q &= v_{qt} - \omega L_t i_d + L_t \dot{i}_q \end{aligned} \tag{4.9}$$

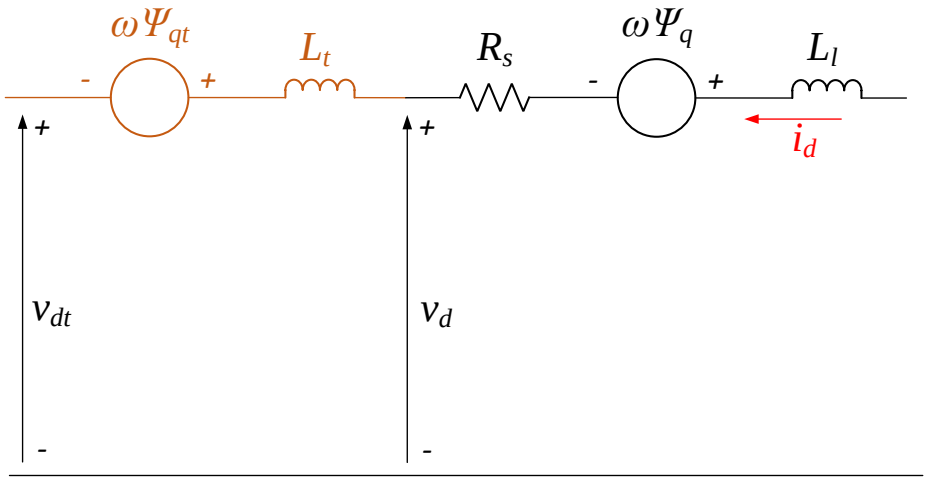


Figure 4.2: Expanded d-axis equivalent circuit when the step-up transformer in included in the machine model. It shows that the d-axis terminal voltage v_d can be calculated from the model output voltage v_{dt}

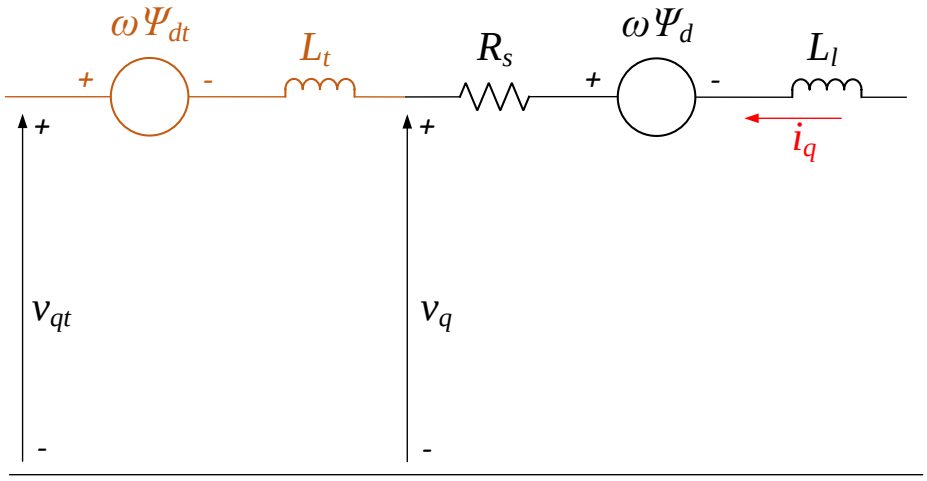


Figure 4.3: Expanded q-axis equivalent circuit when the step-up transformer in included in the machine model. It shows that the q-axis terminal voltage v_q can be calculated from the model output voltage v_{qt}

When the voltages have been accounted for, it must be ensured that all other parameters affected by the transformer implementation are compensated. In fact, the short-circuit time constants $T_d^{'}$ and $T_d^{''}$ are both dependant on the leakage inductance L_l [35], according to the following equations:

$$T_d^{'} = \frac{1}{R_f}\left(l_f + \frac{L_{ad}\, L_l}{L_{ad} + L_l}\right) \tag{4.10}$$

and,

$$T_d^{''} = \frac{1}{R_D}\Big(l_D + \frac{L_l\, l_f\, L_{ad}}{l_f\, L_{ad} + l_f\, L_l + L_{ad}\, L_l}\Big) \tag{4.11}$$

After the transformer implementation, there are four unknowns in the two above equations: the field winding leakage inductance $\hat{l}_f$ and resistance $\hat{R}_f$ and the d-axis damper winding leakage inductance $\hat{l}_D$ and resistance $\hat{R}_D$. The damper windings are not affected by the transformer implementation, and since the real (pre-implementation) machine parameters are known they can be used to calculate the unknown (post-implementation) inductances and resistances. Then the new values of the time constants can be calculated. The equation for l_f is:

$$L_d^{'} = L_l + \frac{L_{ad}\, l_f}{L_{ad} + l_f} \Rightarrow l_f = \frac{L_{ad}\,(L_d^{'} - L_l)}{L_{ad} - L_d^{'} + L_l} \tag{4.12}$$

and for l_D:

$$\begin{aligned} L_d^{''} &= L_l + \frac{L_{ad}\, l_f\, l_D}{L_{ad}\, l_f + L_{ad}\, l_D + l_f\, l_D} \\ \Rightarrow l_D &= \frac{L_{ad}\, l_f\, (L_d^{''} - L_l)}{L_{ad}\, L_d^{''} - L_{ad}\, l_f - L_{ad}\, L_l + L_d^{''}\, l_f - l_f\, L_l} \end{aligned} \tag{4.13}$$

The unknown resistances can be found by rearranging the equations for the d-axis open-circuit time constants,

$$T_{d0}^{'} = \frac{L_{ad} + l_f}{R_f} \Rightarrow R_f = \frac{L_{ad} + l_f}{T_{d0}^{'}} \tag{4.14}$$

and,

$$T_{d0}^{''} = \frac{1}{R_D}\Big(l_D + \frac{l_f\, L_{ad}}{l_f + L_{ad}}\Big) \Rightarrow R_D = \frac{1}{T_{d0}^{''}}\Big(l_D + \frac{l_f\, L_{ad}}{l_f + L_{ad}}\Big) \tag{4.15}$$

With all the inductances and resistances acquired, the new time constants, $\hat{T}_d^{'}$ and $\hat{T}_d^{''}$, can be found by replacing L_l by $\hat{L}_l$ in Equations 4.10 and 4.11. Since all other machine parameters are unaffected by the transformer implementation, getting these new values for v_d, v_q, $T_d^{'}$ and $T_d^{''}$ as explained in this section will give an accurate representation of the generator.

It should also be mentioned that if the open-circuit time constants $T_{d0}^{'}$ and $T_{q0}^{'}$ are entered into the model instead of the short-circuit, the compensation in time constants is not necessary. During an open-circuit (no-load) no current flows through L_l, such that adding X_t to it makes no difference.

4.2.2 Excitation system

The excitation system has the purpose of providing an appropriate field voltage to restore the machine voltage level and reduce rotor oscillations. Its two main components are the AVR and PSS, as described in Section 2.2. There exist pre-built model blocks of various excitation systems in the Simulink libraries. However, as mentioned earlier, these masked blocks cannot be edited or tuned dynamically. Thus, they need to be built manually in such a way that the tuned parameters can come from an external source.

The AVR is implemented as a simple gain, satisfying the equation:

$$v_f = K_a\,\big(V_{ref} - V_c + V_{pss}\big) \tag{4.16}$$

where,

- v_f is the field voltage to the generator
- K_a is the AVR gain
- V_{ref}, V_c and V_{pss} are the voltage reference, control voltage and PSS voltage respectively

Keep in mind that all values are in per-unit in the implementation. If the AVR gain is to be tuned dynamically, the *gain* block in Simulink can be replaced by a *multiply* block, where one of the factors comes from an external tuning system.

The PSS is slightly more complicated. Its block diagram is shown in Figure 2.9. In Simulink, it is possible to add *transfer function* blocks to represent the PSS stages. However, they pose the same challenge of being non-tunable. To alleviate this, it is possible to represent the transfer functions as integrator-feedback systems. Take the closed-loop feedback system in Figure 4.4. From control theory, the transfer function of a closed loop system is:

$$H_{cl}(s) = \frac{A(s)}{1 + A(s)\,B(s)} \tag{4.17}$$

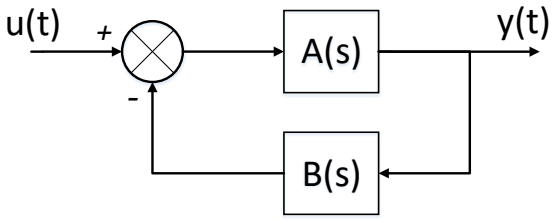


Figure 4.4: A simple feedback control loop often seen in basic control theory

The wash-out filter has the transfer function,

$$H_{wo} = \frac{T_w\, s}{1 + T_w\, s} \tag{4.18}$$

It is then necessary to find valid expressions for $A(s)$ and $B(s)$ in Equation 4.17 such that $H_{cl} = H_{wo}$. Since the the transfer function is to be made of integrators and gains, it must be ensured that $A(s)$ and $B(s)$ only contain integrator ($1/s$) terms, and no derivative (s) terms. The equation becomes,

$$\frac{A(s)}{1 + A(s)\,B(s)} = \frac{s\,T_w}{1 + s\,T_w} \tag{4.19}$$

Since there are more unknowns than equations, it cannot be solved analytically. Still, this equation is simple enough to solve experimentally. Dividing by $T_w s/(T_w s)$ on the right-hand side,

$$\frac{A(s)}{1 + A(s)\,B(s)} = \frac{1}{\frac{1}{T_w\,s} + 1} = \frac{1}{1 + \frac{1}{T_w}\frac{1}{s}} \tag{4.20}$$

From visual inspection, a solution to this equation is $A(s) = 1$ and $B(s) = 1/(T_w\, s)$. Entering that solution into the block diagram of Figure 4.4, an equivalent block diagram of the wash-out filter is made. Thus, both block diagrams in Figure 4.5 are equivalent and give the same

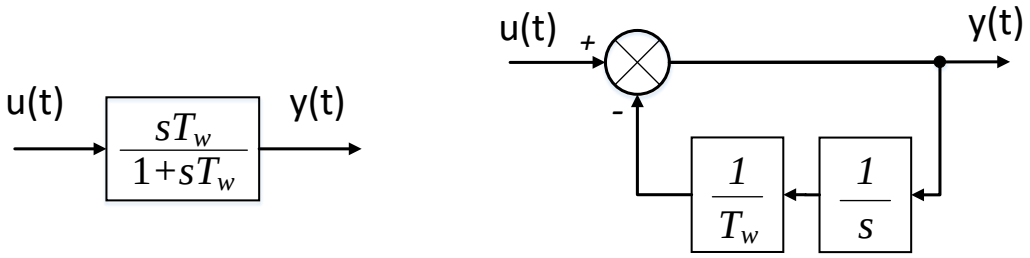


Figure 4.5: Equivalent block diagrams of the wash-out filter, displayed as a transfer function and an integrator-feedback system

response. The gain with T_w can be replaced by a *divide* block in Simulink, where T_w can come from an external tuner if desired.

The transfer function for a lead/lag stage is:

$$H_{ll} = \frac{1 + T_1\, s}{1 + T_2\, s} \tag{4.21}$$

This expression can be split to two terms, which will be evaluated separately. In a block diagram, two added terms result in a parallel connection, which can easily be implemented later. The equations for the two separate terms become:

$$\begin{aligned} \frac{A_1(s)}{1 + A_1(s)\, B_1(s)} &= \frac{T_1\, s}{1 + T_2\, s} \\ \frac{A_2(s)}{1 + A_2(s)\, B_2(s)} &= \frac{1}{1 + T_2\, s} \end{aligned} \tag{4.22}$$

Inspecting the first equation, dividing by $T_2 s/(T_2 s)$ on the right-hand side and introducing a T_1/T_1 term in the denominator gives:

$$\frac{A_1(s)}{1 + A_1(s)\, B_1(s)} = \frac{\frac{T_1}{T_2}}{1 + \frac{T_1}{T_2}\frac{1}{T_1\, s}} \tag{4.23}$$

Through inspection, $A_1(s) = T_1/T_2$ and $B_1(s) = 1/(T_1 s)$. Next, inspecting the second equation in Equation 4.22, dividing by $T_2 s/(T_2 s)$ on the right-hand side gives:

$$\frac{A_2(s)}{1 + A_2(s)\, B_2(s)} = \frac{\frac{1}{T_s\, s}}{1 + \frac{1}{T_2\, s}} \tag{4.24}$$

Through inspection, $A_2(s) = 1/(T_2 s)$ and $B_2(s) = 1$. Then, the equivalent block diagram with integrators and gains can be made. Both block diagrams in Figure 4.6 give the same response. Similarly as for the wash-out filter, the time constants T_1 and/or T_2 can be received from an external tuner by replacing the *gain* blocks with *multiply* blocks.

There is one important note to make. Since the per-unit version of the synchronous machine model was chosen, it must be ensured that all quantities are in per-unit. The integrator blocks, however, keep time in seconds. Thus, after every integrator block, the signal must be multiplied by the time base t_B (or divided by the speed base ω_B). If the per-unit machine model is not used, this is not necessary.

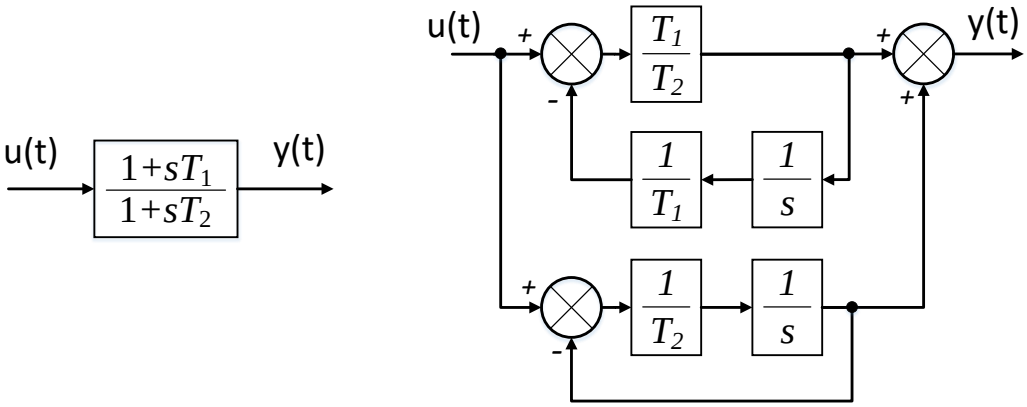


Figure 4.6: Equivalent block diagrams of a lead/lag stage, displayed as a transfer function and an integrator-feedback system

4.2.3 External network

To model the connection to a power system, a simple Thévenin equivalent system is implemented. It takes the shape of an infinite bus in series with an impedance. The infinite bus is represented by a *Three-phase voltage source* block. The impedance split into its resistive and reactive parts and is represented by two separate *Three-phase RLC parallel branch* blocks.

As mentioned initially, the parameters of masked blocks cannot be edited dynamically. Thus, a load change in the external network must be implemented in another way than adjusting the RLC branch parameters. For this purpose, the resistance and infinite bus voltage are considered constant, such that a change in the external network short-circuit power is represented by a change in the Thévenin reactance. To apply a change, another reactance is added in parallel to the first with a breaker in series. The external network is then as in Figure 4.7. A step up in reactance is equivalent to breaking the second reactance, and a step down is equivalent to connecting it. This model is sufficient to analyse the step response of the machine.

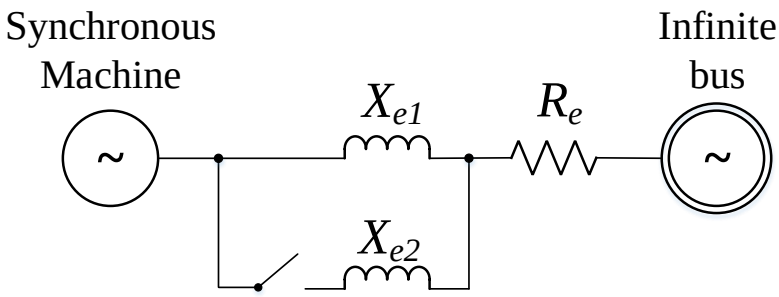


Figure 4.7: Line diagram of the external network model, displaying how the reactance step is made

When performing a step up, the initial parallel reactance is,

$$X_{par} = \frac{X_{e1}\, X_{e2}}{X_{e1} + X_{e2}} \tag{4.25}$$

After the step, the reactance is equal to X_{e1}. During a step down, the opposite is true. When defining the pre- and post-step reactances, it is therefore important that $X_{e1} > X_{par}$ to give a valid parallel connection. The step direction is determined by the initial state of the breaker. When X_{par} and X_{e1} have been chosen, the last reactance is calculated by:

$$X_{e2} = \frac{X_{par}\, X_{e1}}{X_{e1} - X_{par}} \tag{4.26}$$

4.2.4 Improving simulation time

Continuous simulations may be very slow. A few actions are made to increase the speed of the simulations. Firstly, the simulation mode is set to *Accelerator*. This alters the model code generation for a decreased running time [63]. Experiments show that the *Accelerator* mode does not decrease the simulation accuracy to any significant degree. The *Rapid Accelerator* mode, however, can not be used in this application, as the accuracy drops beyond acceptable levels.

In addition, the machine model creates some initial dynamics. These initial oscillations need to settle to steady-state before any further tests can be made, and this adds simulation time. Since the initial dynamics are of no interest to the test, the limiters of the PSS and AVR can be disabled the first few seconds in the simulation. This will yield unrealistic components, as the field voltage amplitude can become very large, but will noticeably increase the initial damping. The limiters must be re-enabled before any proper testing is done.

Moreover, if the same initial conditions are to be used for several tests, Simulink has the option to save and restore simulation states (or operation points) [64]. It is then possible to save the state just before the test is performed and resume it multiple times for different tests. If this option is used, the initial oscillations will only need to be simulated once.

4.3 Testing the CPSS auto-tuning system

A Simulink model is created by applying the methods described in the previous sections. First, the PSO results and the function fitting neural network are evaluated. Figure 4.8 shows the optimised parameters at several different values of X_e within the defined range of $[0.001, 0.1]$. The regression line from the trained auto-tuning NN is also shown. The NN is able to predict the PSS parameters from the external reactance well. Table 4.1 gives the NN output for a few values of X_e, which can be seen matches the regression in Figure 4.8.

External reactance $\mathbf{X_e}$ [pu]	**$\mathbf{K_{pss}}$**	**$\mathbf{T_1}$ [s]**
0.005	3.133	0.258
0.03	8.371	0.127
0.05	12.675	0.103
0.1	23.429	0.082

Table 4.1: Outputs of the CPSS auto-tuner for the four values of the external reactance X_e that have been tested in this thesis.

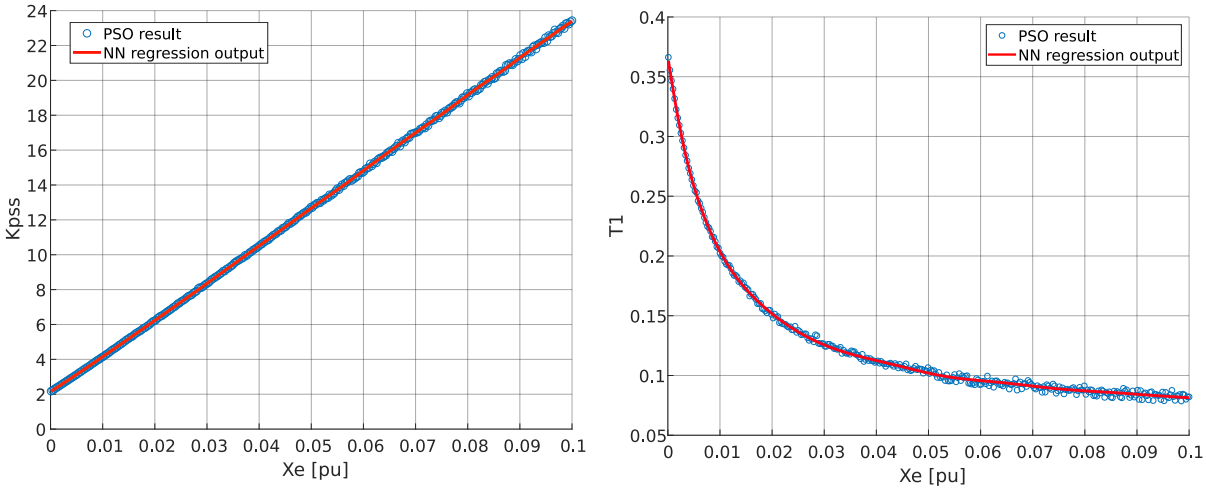


Figure 4.8: K_{pss} and T_1 for different external reactances from the PSO procedure, with the corresponding regression lines from the auto-tuning neural network

To test the system a small step in X_e is performed at 10 seconds (after letting the initial dynamics settle), where it is increased from 0.03 to 0.05 pu. The response in speed deviation is shown in Figure 4.9. The system is stable. Note that the control voltage to the AVR has been set to 100 % into the transformer in this test, to circumvent the need for the back-calculation in Section 4.2.1. In Figure 4.10 the live calculations of X_e, K_{pss} and T_1 are shown. The reactance calculation performs very well, even though there are some small oscillations when the step occurs. Consequently, the same happens to the NN output. It can be seen that the values of K_{pss} and T_1 before and after the step correspond with the regression lines in Figure 4.8. Hence, the auto-tuning system works as intended by successfully tracking the external network changes and applying previously optimised tuning parameters. A study of its performance compared to other PSS methods will be given in Section 5.4.

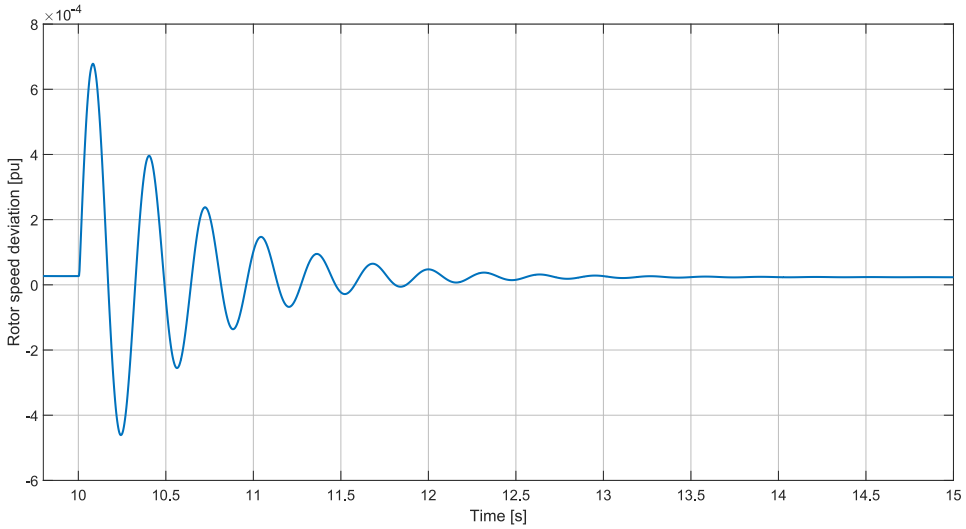
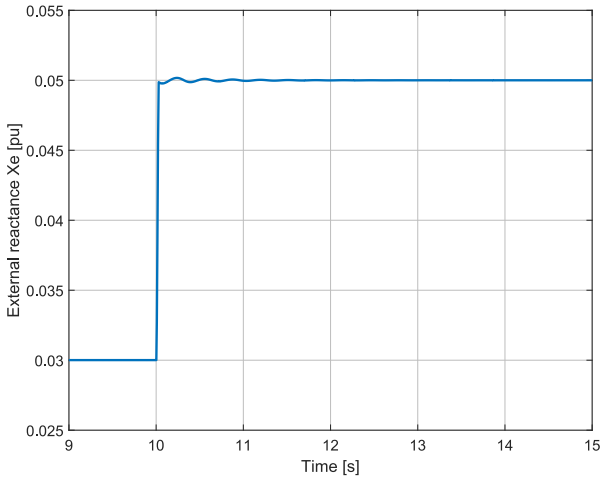
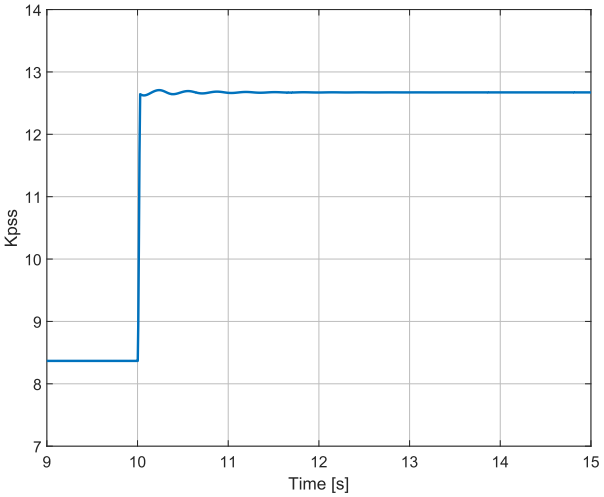


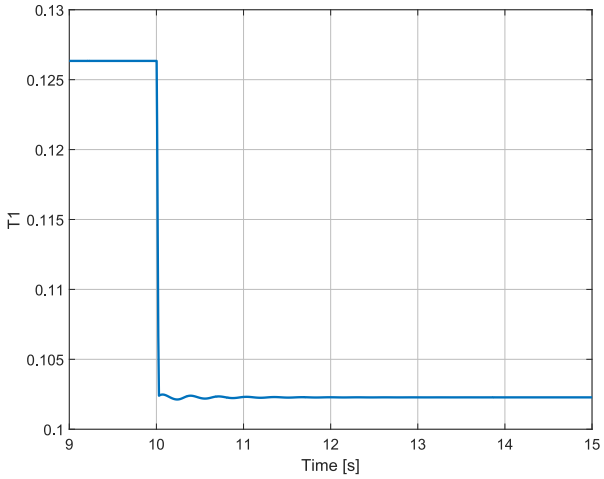
Figure 4.9: Rotor speed deviation response to a step in X_e from 0.03 to 0.05 pu, with the NN auto-tuner active



(a) Calculation of external reactance during a reactance step



(b) Auto-tuned value of PSS gain K_{pss} from the neural network during the reactance step



(c) Auto-tuned value of PSS time constant T_1 from the neural network during the reactance step

Figure 4.10: Online calculation of external reactance X_e and the corresponding response from the auto-tuning NN, during a step in X_e from 0.03 to 0.05 pu

Chapter 5

The sine shifting neural network controller

> *"Thus our knowledge of the world, including ourselves, is incomplete as to space and indefinite as to time. This ignorance, implicit in all our brains, is the counterpart of the abstraction which renders our knowledge useful. The role of brains in determining the epistemic relations of our theories to our observations and of these to the facts is all too clear, for it is apparent that every idea and every sensation is realized by activity within that net, and by no such activity are the actual afferents fully determined."*
>
> – McCulloch and Pitts [22]

Most power generators that have a PSS installed applies a version of the conventional PSS. Although this model does improve the dynamic stability of the generator, it can only do so to a limited degree. If the dominant modes of oscillation are known for a generator system, it could be beneficial to create a specialised PSS designed to damp those frequencies.

The rotor speed deviation $\Delta\omega$ is a common input signal for the PSS, as it is a good indicator of the rotor behaviour. Due to its link to the machine's physical mass, it is a continuous signal without breaks. Moreover, it is an oscillatory signal that oscillates around the zero-line. The speed deviation can, therefore, be considered a damped sinusoidal signal. There are mainly two ways to alter a sinusoid of fixed frequency: changing its amplitude and phase. The former is done simply by adding a gain. The latter, however, is more challenging. The CPSS alters the phase with lead/lag blocks, adding poles and zeros for phase shifts at opportune frequencies. Unfortunately, this provides little adaptivity.

The frequency response of a simple CPSS with a single lead/lag stage may look like Figure 5.1. The figure shows how the PSS attempts to provide a stable phase shift in the range of 0.2-5 Hz, which is the frequency range of electromechanical modes. However, since it is made to perform decently on all frequencies, it does not perform optimally on any frequency. Furthermore, it is not clear which phase shift is optimal. There may very well be another phase shift which will provide better damping.

The idea behind the sine shifting neural network (SSNN) is to create a controller where the phase shift is a control variable. Consequently, for a specific frequency, the gain of the controller will be 0 dB and the phase response will be exactly as entered into the controller. A transfer function will not be able to provide this functionality, so a shallow neural network has been chosen to work as the controller. Moreover, the SSNN is unique to most other PSS adaptations in that no electrical machine theory is involved in its creation.

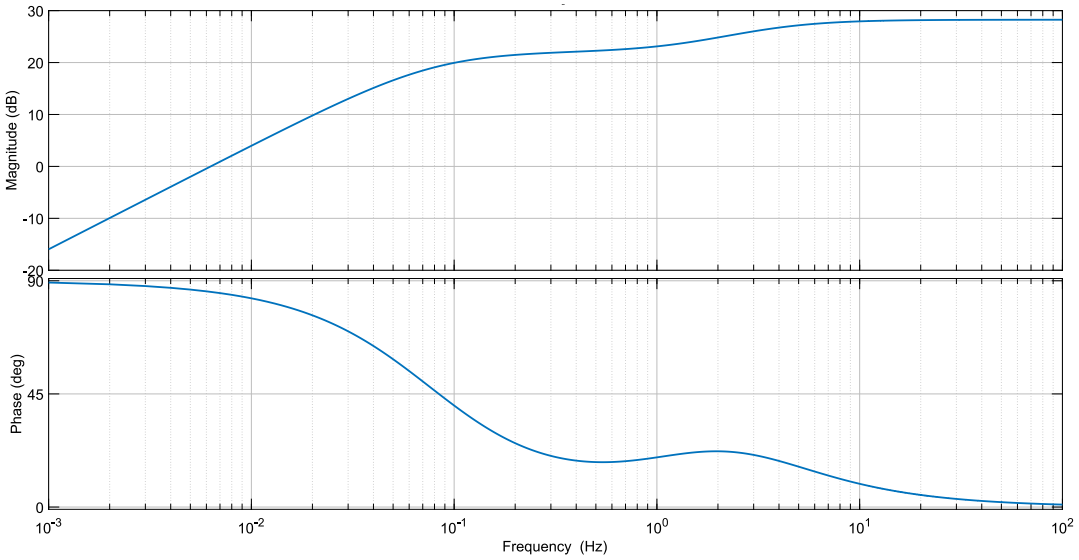


Figure 5.1: Frequency response of a simple CPSS with a single lead/lag stage

The following sections will describe the creation of this novel SSNN controller and test its performance. Lastly, it will be compared to three other PSS approaches to validate its damping capabilities to a step in X_e. Figure 5.2 shows an overview of the intended SSNN scheme that will be explained in this chapter.

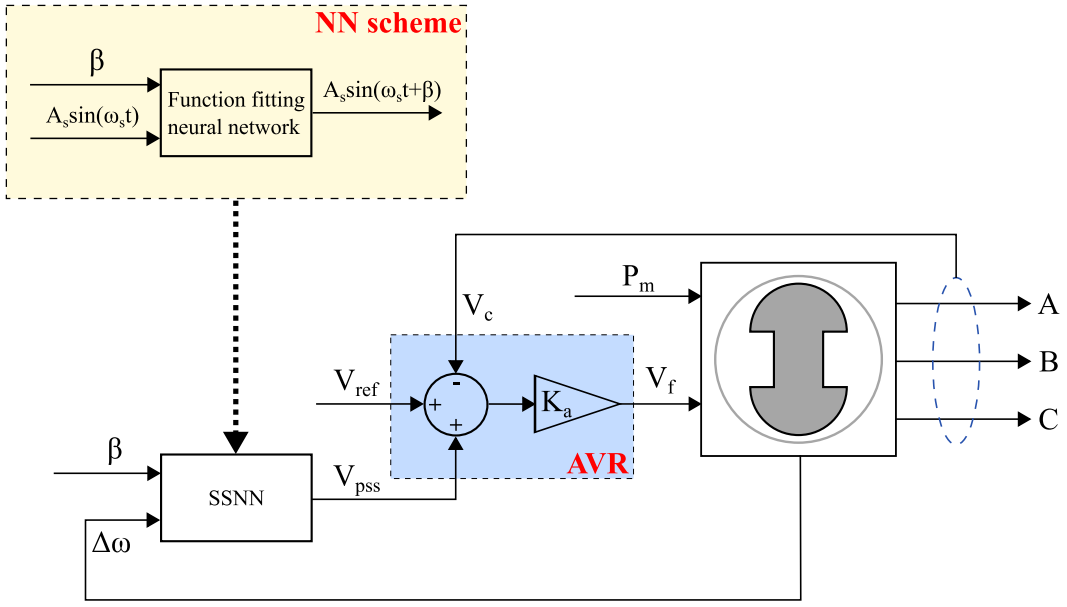


Figure 5.2: Illustration of the SSNN model discussed in this chapter within the machine topology

5.1 Training

The fact stated initially that the speed deviation can be considered a damped sinusoid is vital for the choice of training data. It means that the SSNN itself does not need to "know" anything about electrical machine theory. Rather, it can be trained on purely theoretical sine waves.

The SSNN should ideally have two inputs: the desired phase shift in radians and the input sine wave (speed deviation). The only output should be an identical sine wave as the input, phase-shifted by the specified angle. In other words, say the input signal is,

$$A_s \, sin(\omega_s \, t) \tag{5.1}$$

where A_s is the sine wave amplitude and ω_s is its frequency in rad/s. If the SSNN is told to shift the input by β rads, its output should be,

$$A_s \, sin(\omega_s \, t + \beta) \tag{5.2}$$

When the frequency is fixed three parameters can vary: A_s, t and β. The amplitude will vary continuously, as the input will be a damped signal. β may not necessarily vary, but it is important to train for several shift angles to increase the versatility of the controller. Training for several values of t is also necessary, as the NN must be able to recognize all parts of a sine wave.

In essence, the SSNN will be a sine wave predictor, where it can predict the next time step in a sine wave. To be able to do so, a single time step is an insufficient input. See Figure 5.3. Knowing a single point on a curve is only enough to tell its y-axis position. With two points the slope can be determined. However, since a sine wave is symmetrical, it is uncertain whether the two points are on the upper or lower half of the wave. Thus, three consecutive time steps are needed to uniquely determine the current position on a sine wave with known frequency, and subsequently to predict the next point.

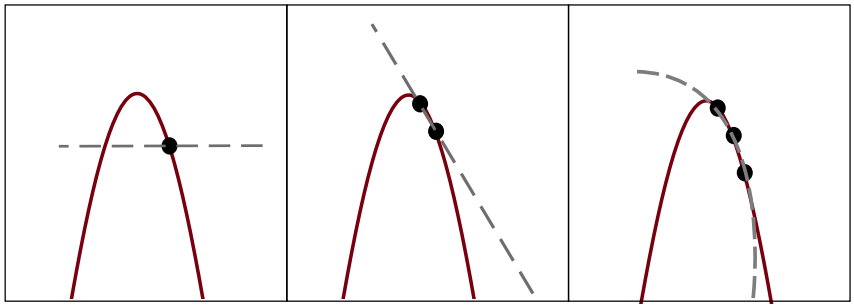


Figure 5.3: Illustration showing that three consecutive points are necessary to determine the current position on a sine wave and predict the next step

The SSNN must, therefore, have the following four inputs:

1. Desired phase shift angle β
2. Speed deviation at time t
3. Speed deviation at time $t - 1$
4. Speed deviation at time $t - 2$

Having the inputs and outputs established the training data set can be created. As mentioned, three parameters can vary in the sine wave. Each must be given upper and lower bounds and a resolution, determining how many iterations of each parameter is performed. A larger resolution will give a more accurate controller, but will quickly increase the size of the data set and subsequently the training time. Therefore, a balance must be made between accuracy and training time.

Table 5.1 shows the selected bounds and resolution for the varying parameters. Theoretically, the phase shift β could vary between $-180°$ and $180°$. However, experiments have shown that phase shift values less than $0°$ or greater than $130°$ never cause improved damping, so the training set is limited to those values. From Figure 4.9 it can be observed that the dominant mode of oscillation of the machine under study is at 3.2 Hz, giving a period $T = 0.3125\ s$. This will be the frequency the SSNN is trained at. Since consecutive time steps are needed, the simulations must be discrete. Experimentation in Simulink has proven that a sampling time greater than $T_s = 3 \cdot 10^{-5}\ s$ gives highly inaccurate results. The training data sampling time must be the same as the simulation sampling time. Thus, T_s is set at this maximum to keep the training data set as small as possible. The time step resolution then becomes T/T_s.

	Lower bound	**Upper bound**	**Resolution**
Phase shift β	$0°$	$130°$	90
Amplitude	0.001	0.1	30
Time step	0 s	$0.3125\ s$	10 417

Table 5.1: The bounds and resolution for the SSNN training data set

The training data set is then built iteratively. A pseudo-code of how the set is created is given in Figure 5.4. The set can easily become very large, into the tens of millions, making a high-end computer very beneficial for training. In this case, the training data set size becomes:

$$30 \cdot 90 \cdot 10\,417 = 28\,125\,900 \tag{5.3}$$

The SSNN is chosen to be a function fitting shallow NN, with one hidden layer having 10 neurons. See Appendix A for the training parameters and settings used to create the NN and Appendix B for a guide on how such an NN can be trained in MATLAB.

```
For every amplitude Aᵢ
  For every phase shift βⱼ
    For each time step tₖ in one period
      SSNN input 1 = βⱼ
      SSNN input 2 = Aᵢ sin(ωtₖ)
      SSNN input 3 = Aᵢ sin(ωtₖ₋₁)
      SSNN input 4 = Aᵢ sin(ωtₖ₋₂)
      SSNN output = Aᵢ sin(ωtₖ + βⱼ)
    end
  end
end
```

Figure 5.4: Pseudo-code for building the SSNN training data set

5.2 Performance assessment

When the SSNN is fully trained, there is a need to verify its performance. This section will assess the network's performance as a stand-alone sine wave predictor. Its performance for

rotor angle damping compared to other PSS models will be covered in Section 5.4. The test in this section involves inputting a damped sine wave while requesting various phase shifts. The testing system in Simulink is shown in Figure 5.5. A sine wave with a damped element is given to the input of the SSNN such that:

$$S(t) = \begin{cases} A_s \, sin(\omega_s \, t) & , \quad t < T_0 \\ A_s \, sin(\omega_s \, t) \, e^{-k(t-T_0)} & , \quad t \geq T_0 \end{cases} \tag{5.4}$$

where $S(t)$ is the sine wave used as reference and input to the SSNN; T_0 is the time where the damping component is enabled; k is the damping coefficient; A_s is the sine wave amplitude and ω_s is its frequency.

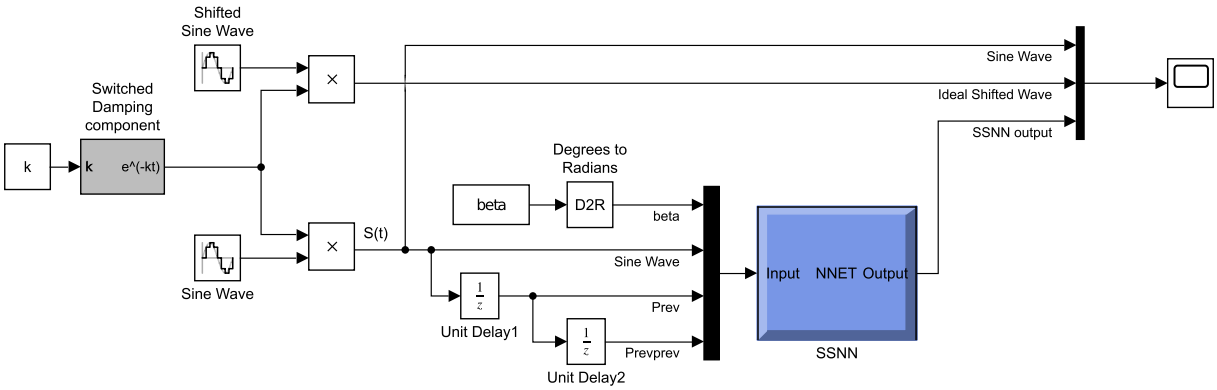
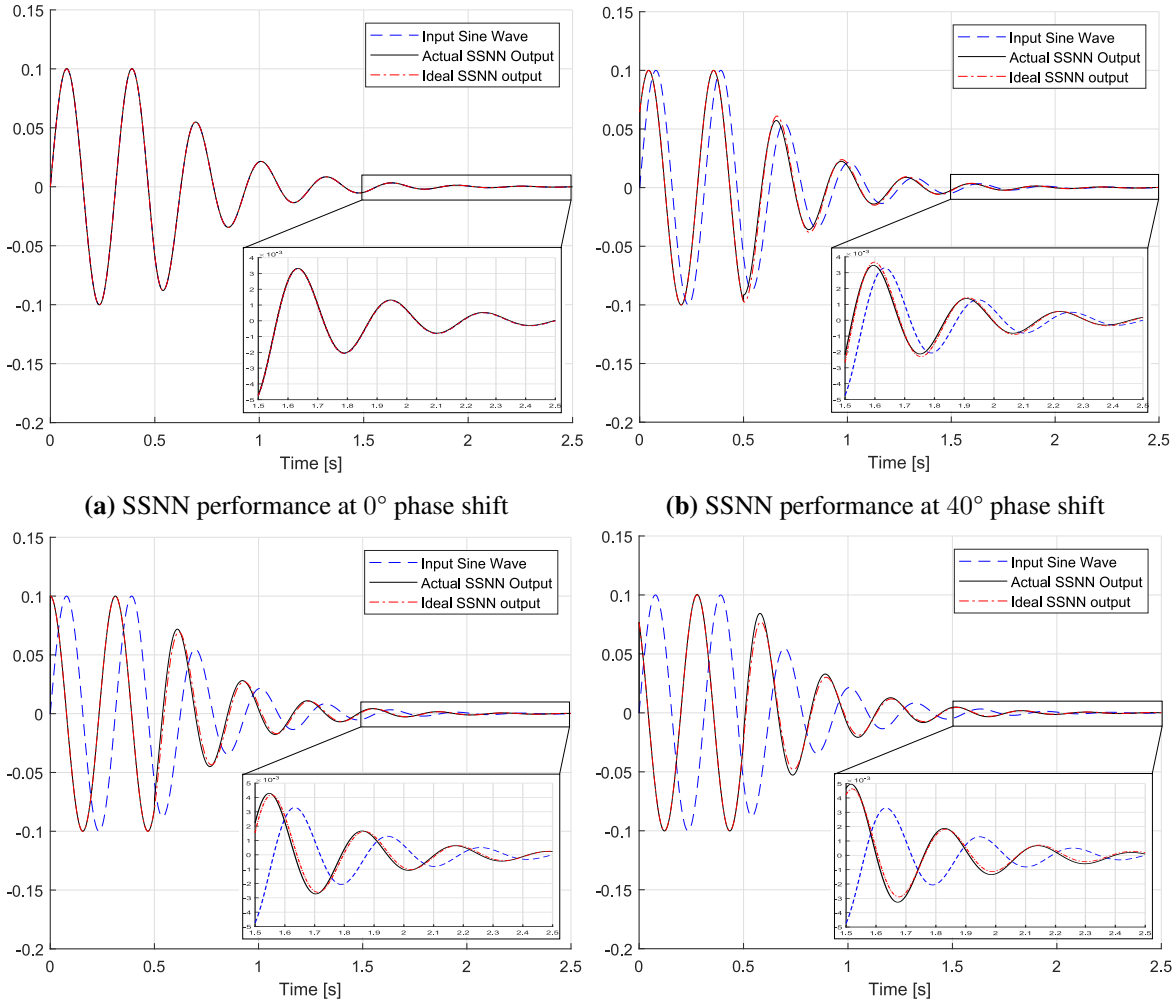


Figure 5.5: Simulink model for testing the SSNN response to a damped sine wave

In Figure 5.6, four tests are done with amplitudes and phase shifts spanning the ranges defined in Table 5.1. The amplitude is initiated at 0.1 and is decreased when the damping component is enabled. The tested phase shifts are $\beta = \{0, 40, 90, 130\}$ degrees, and the frequency is 3.2 Hz. Along with the SSNN output, the ideal phase-shifted wave $A_s \, sin(\omega_s \, t + \beta)$ is shown to validate its performance.

It is clear from the test that the SSNN tracks the ideal wave very well at all phase shifts. Slight deviations appear when the amplitude is varying, as the network is unable to predict the future amplitude values. This confirms that the SSNN performs well at the frequency it was trained for.

The natural follow-up is to analyse the response to other frequencies. It is expected that the performance will decrease at frequencies differing from the training frequency. See Figure 5.7. It illustrates that if three consecutive points y_1, y_2, y_3 on a sine wave are known and the frequency is not fixed, the wave can not be uniquely determined. These consecutive points can be found on both a high-amplitude, low-frequency wave and on a low-amplitude, high-frequency wave. In the "world" of the SSNN, only sine waves at 3.2 Hz exist. So if it observes a wave with higher frequency, it will interpret it as a 3.2 Hz wave of high amplitude. The opposite applies for lower frequencies, as it will interpret it as a low-amplitude wave. The SSNN will be able to follow the input wave in terms of frequency, but the amplitude and phase responses will deteriorate as a result. For this reason, training the network for more than one frequency will not help either, as it would be unable to distinguish them. The outcome would be a network behaving as if it was trained on the average of all the training frequencies. For example, if the SSNN is trained on 1 and 3 Hz, its best operating point would be at 2 Hz. Furthermore, since the SSNN is



(a) SSNN performance at $0°$ phase shift

(b) SSNN performance at $40°$ phase shift

(c) SSNN performance at $90°$ phase shift

(d) SSNN performance at $130°$ phase shift

Figure 5.6: Performance assessment of the SSNN at requested phase shift β of 0, 40, 90 and 130 degrees. The frequency is fixed at 3.2 Hz. The input sine wave is shown along with the ideal phase shifted output wave for validation. The SSNN seems to track the ideal output well.

trained on periodic signals centred at the zero-line, it is unfamiliar with non-oscillatory errors. Its response to steady-state errors or non-oscillatory modes can therefore not be expected to be of high quality.

Electromechanical modes oscillate at 0.1-5 Hz, which is the frequency range of interest. The frequency response is created by exposing the network to the various frequencies and observing the output amplitude and phase. At each frequency, several values of β are entered. In Figure 5.8 the gain and phase responses of the SSNN is shown. The ideal gain response would be a constant 0 dB gain at all scenarios and the ideal phase response would be a constant phase shift equal to β at all frequencies. The figure shows that when $\beta = 0°$ this is the case. Additionally, at the training frequency of 3.2 Hz, the response is nearly perfect with zero gain and phase equal to β.

The SSNN performs worst at frequencies much lower than the training frequency. For β close to $0°$ there is little attenuation and phase drift. At $\beta = 90°$ the attenuation peaks and the phase is least stable (except at exactly $90°$, where has little phase drift). With β increasing from $90°$

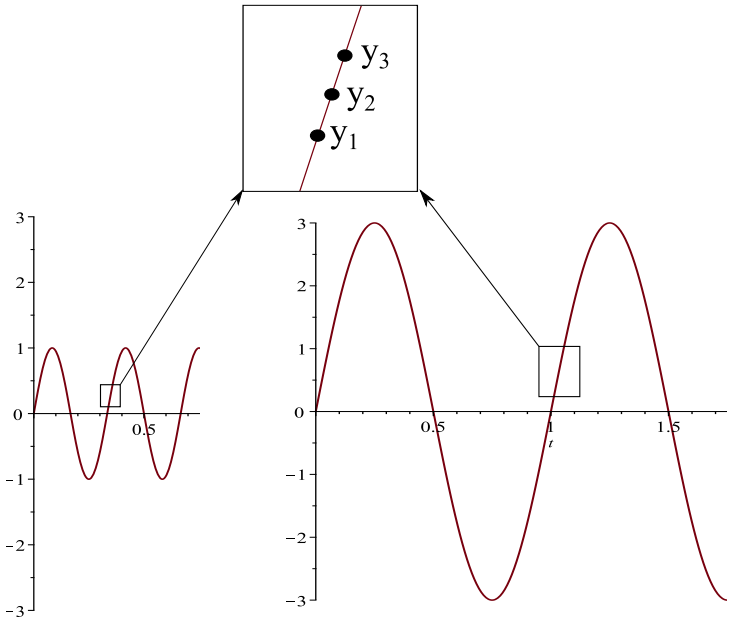


Figure 5.7: Graph illustrating that three consecutive points on a low-frequency high-amplitude sine wave may also be found on a low-amplitude high-frequency wave.

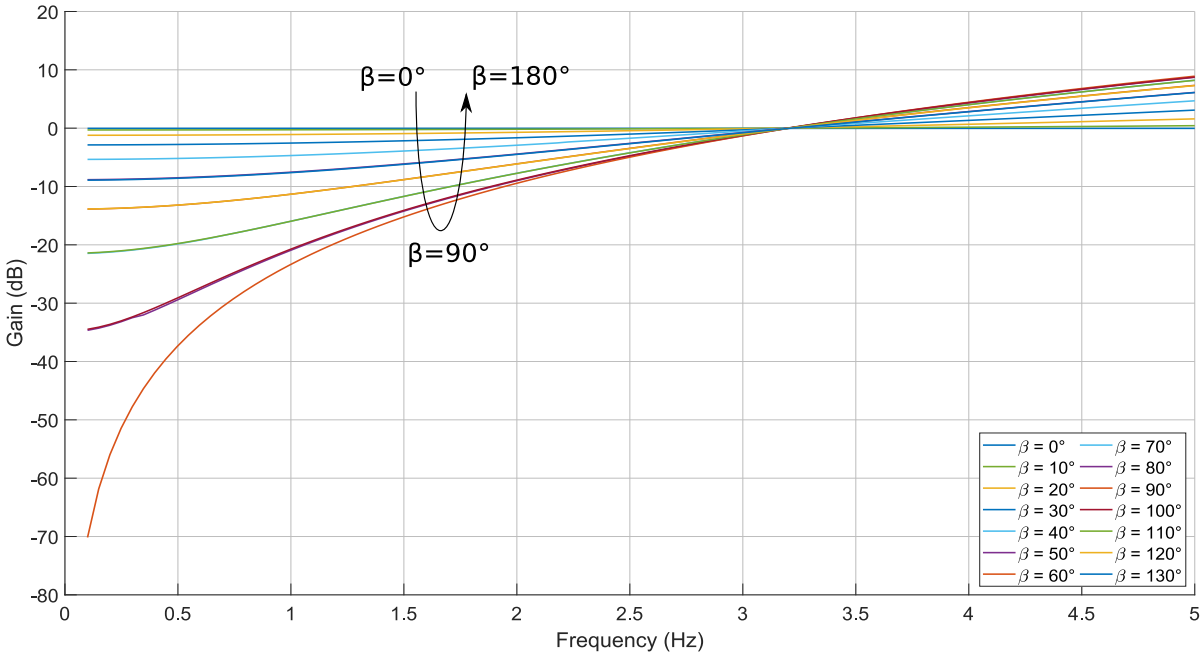
the performance improves again. At frequencies higher than the training frequency the response behaves oppositely to the lower frequencies, yet to a much lesser degree.

To make a controller more fitted to work at various frequencies the SSNN could be trained at a lower frequency, say 1.5 Hz. The frequency response to such an SSNN is shown in Figure 5.9. The phase response is improved, as it has a larger area with constant phase shifts. The gain at higher frequencies, however, has increased. Here, it can be more clearly seen that for frequencies lower than the training frequency the phase converges to either $0°$ or $180°$, depending on whether β is greater than $90°$, and the phase slowly converges to $90°$ at higher frequencies (with the notable exception when $\beta = 0°$).

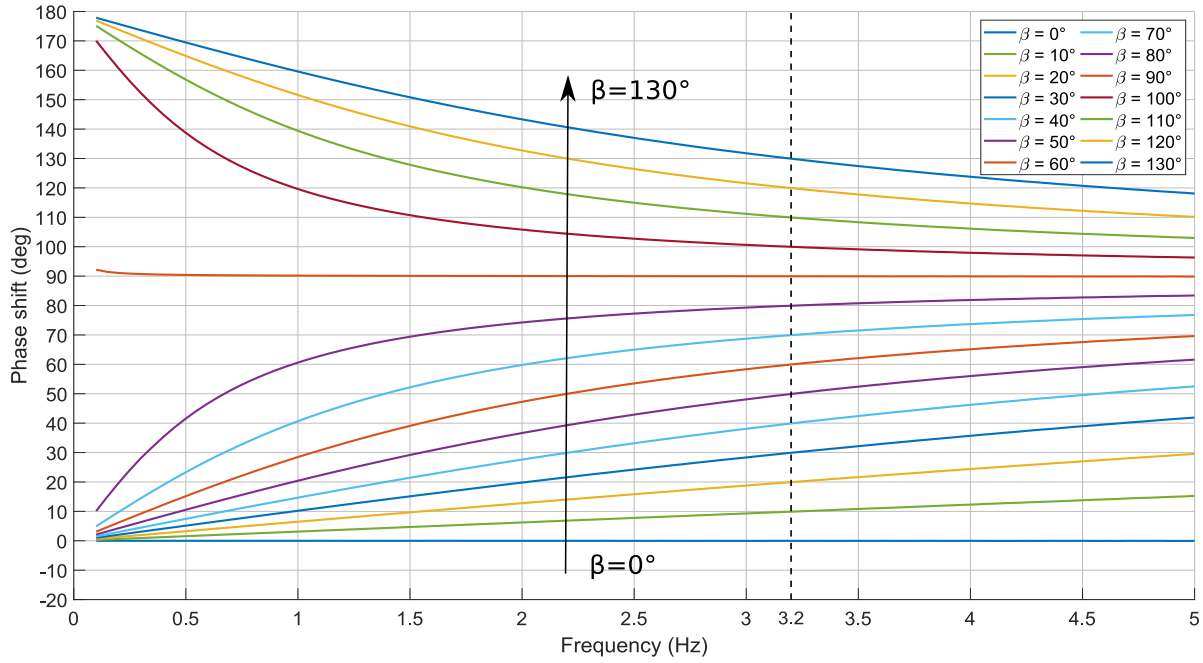
5.2.1 Correcting the amplitude and phase drifts

The neural network structure the proposed SSNN is built on is relatively simple. It is a 3-layer feedforward network trained with supervised learning. Its performance is therefore highly limited by the training data. Any operation outside the training data set will yield suboptimal results. This subsection will briefly discuss how these deviations may be corrected. Implementing and testing the corrections will, however, be left to further studies.

If the frequency of oscillation is known, but at a frequency different from the training frequency, it could be possible to develop an algorithm to compensate the deviations. Take Figure 5.8. Unique positions on the gain response plot can be found with a specified frequency and phase shift angle β. Thus, corresponding gains can be provided to the output signal to compensate the attenuation (or gain) of the SSNN. Still, this would not fix the phase drift. In this case, an SSNN trained at a lower frequency, such as in Figure 5.9, would be beneficial, as there is less phase drift. Another alternative would be to implement an array of SSNNs trained at different frequencies and enable them according to the current oscillation frequency. In that case, commissioning the array according to a Fourier transform of the input could even accommodate for

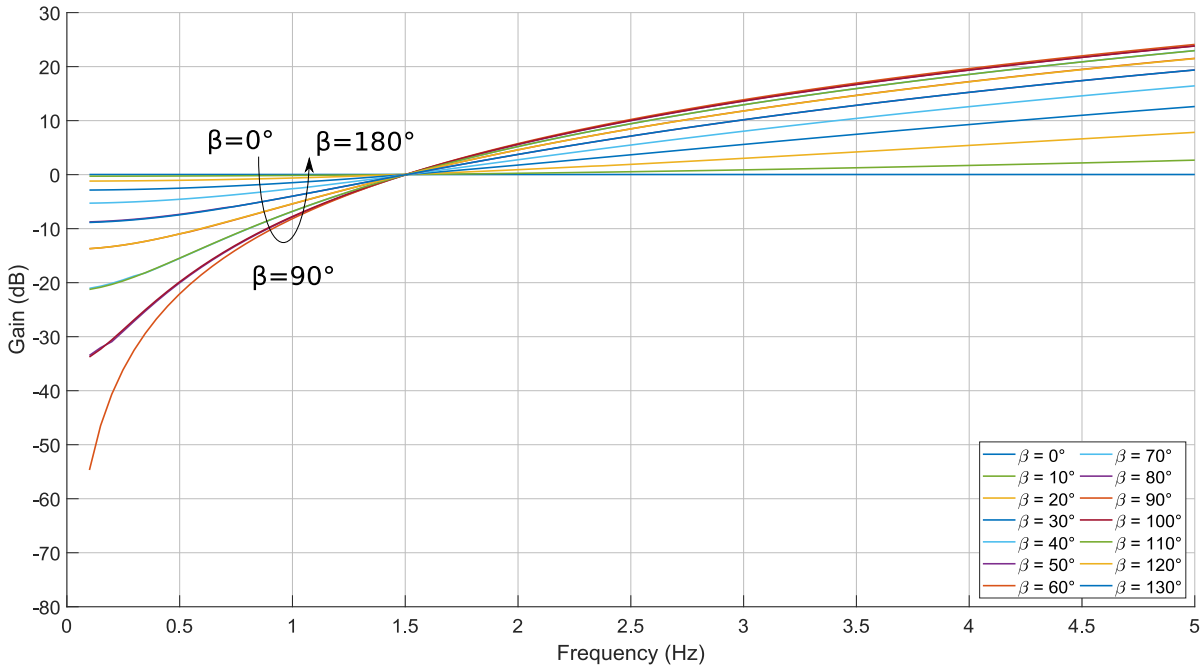


(a) Gain response of SSNN at low frequencies

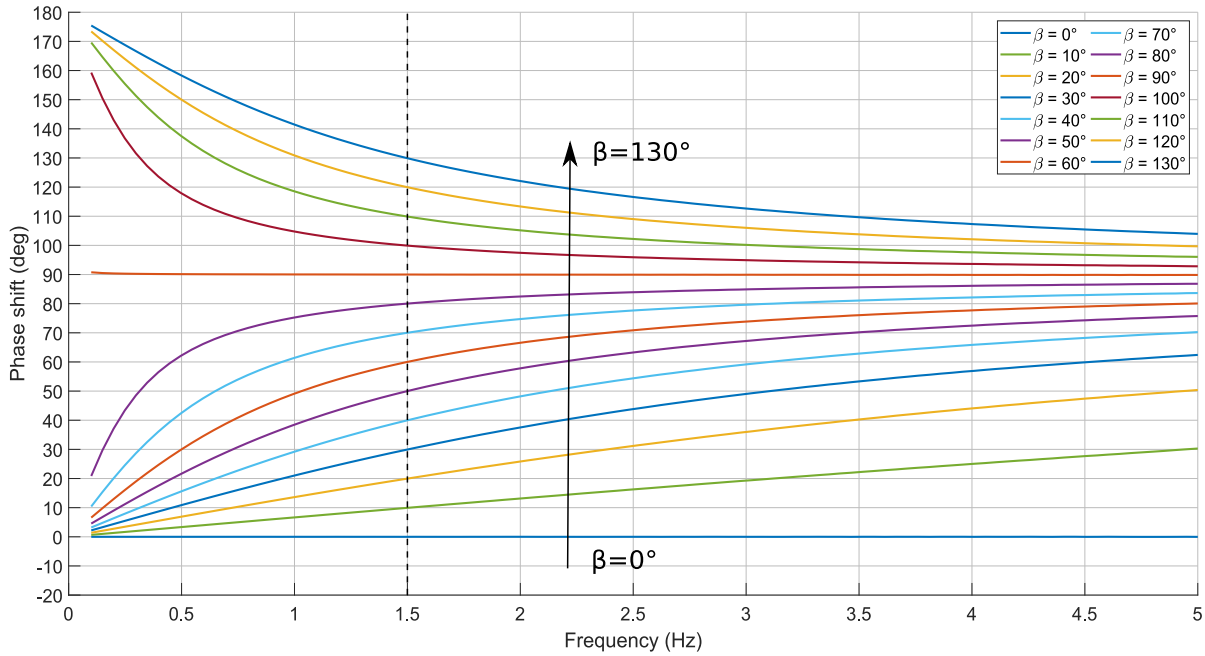


(b) Phase response of SSNN at low frequencies

Figure 5.8: Frequency response of the SSNN at the low electromechanical frequencies. At the training frequency of 3.2 Hz it performs almost perfectly, though at other frequencies there is some deviation.



(a) Gain response



(b) Phase response

Figure 5.9: Frequency response of the SSNN at the low electromechanical frequencies when trained at 1.5 Hz. This controller shows less extreme deviations at low frequencies, with a trade-off of larger deviations at high frequencies.

harmonics in the oscillation.

If the frequency of oscillation is not known, the above would not work. The simple design of the SSNN is no longer sufficient. Therefore, more complex types of neural networks can be considered. For example, recurrent neural networks (RNN) based on long short-term memory (LSTM) have been successfully used to forecast periodic signals [65]. RNNs not only use the current input signal to determine its output, but it also uses the output from the previous time step. The LSTM design gives an option to how much the network should remember or forget from its previous time steps. The network does no longer only have inputs and outputs, it also has a state which is updated through time. Applying such a network to the sine-shifting problem could solve the issue of gain and phase drift without the need for external compensation. The theoretical details of LSTM are beyond the scope of this chapter, but a description of the LSTM network, along with an example where it is applied to forecast a discrete sine function, can be found in Part II of [65].

5.3 Applying as a PSS

The SSNN ideally gives a gain of 0 dB. Its input is the rotor speed deviation $\Delta\omega$ and its output is the PSS contribution to the AVR v_{pss}, which is added to the voltage error $V_{ref} - V_c$. However, since $\Delta\omega$ usually is much smaller than the voltage error during disturbances, a PSS gain must be added for the SSNN to have any real impact on the system. The size of this gain should not be so big as to override the AVR or saturate it to easily. In the machine system under study, a PSS gain of 50 has been found to give a good contribution. This gain will be kept constant for the remainder of this study. Furthermore, the frequency of oscillation of the system is 3.2 Hz, so the controller shown in Figure 5.8 will be applied.

The SSNN (with the aforementioned gain) replaces the CPSS in its entirety. Moreover, the AVR is considered a simple gain with the control voltage being 100 % into the step-up transformer, such that the back-calculation in Section 4.2.1 can be circumvented. The test system then looks like Figure 5.10, following the other implementation procedures in Section 4.2. Three major quantities affect the test results: The pre- and post-disturbance external reactances and the phase shift angle β. As a baseline, a small step in reactance from 0.03 to 0.05 pu is performed. The step is made four times at different phase shifts, $\beta = \{0, 40, 90, 110\}\ deg$. The step responses can be seen in Figure 5.11.

There are a few important observations that can be made. Firstly, the value of β alters the frequency of oscillation. As discussed in the previous section, this will impact the performance of the SSNN, where low frequencies generally have the worst performance. Secondly, the SSNN is well able to improve the rotor oscillation damping by increasing the angle β. Yet, if it is increased too much, it has a detrimental effect on the damping. This means an optimum exists. With the goal of finding the best value for β the simulation was run iteratively for several different pre- and post-disturbance values of X_e. The angle β was incremented by $10°$ each iteration. The MATLAB command `lsiminfo` was used to determine the damping performance and consequently the angle of largest damping for each X_e step. However, the best angle changes whether settling time or overshoot is prioritised. Tables 5.2 and 5.3 show the obtained best values for β for each X_e step, when prioritising settling time and overshoot respectively. Consider Figure 5.11. The first-swing overshoot (initial peak) is nearly equal for

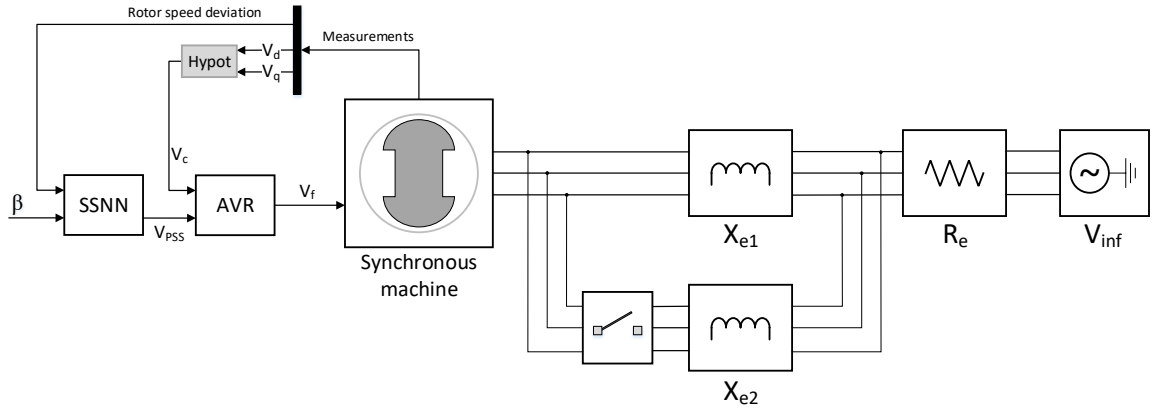


Figure 5.10: Model schematic for testing the SSNN performance as a PSS. It is redrawn from the Simulink diagram for a cleaner look

all angles. The value of the second peak, however, varies greatly. Thus, the overshoot in these tables is considered the second-swing overshoot. It can be seen that for the majority of the reactance steps, a phase shift β between $40° - 70°$ gives the best performance.

		Post-disturbance X_e [pu]										
		0.005	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**	**0.1**
Pre-disturbance X_e [pu]	**0.005**	N/A	100	100	90	80	80	70	70	70	70	40
	0.01	100	N/A	40	90	80	40	70	70	60	60	60
	0.02	10	40	N/A	30	40	70	70	70	60	60	60
	0.03	40	30	40	N/A	30	70	70	70	60	60	60
	0.04	40	40	40	50	N/A	70	70	70	60	60	60
	0.05	40	40	40	60	50	N/A	70	70	60	60	60
	0.06	40	40	70	60	50	50	N/A	70	60	60	60
	0.07	50	50	70	60	50	50	40	N/A	40	60	60
	0.08	50	50	70	60	50	50	50	40	N/A	60	60
	0.09	50	50	70	60	50	50	50	40	40	N/A	60
	0.1	50	50	50	60	50	50	50	50	40	40	N/A

Table 5.2: Phase shift angles β for several pre- and post-disturbance external reactances X_e giving the shortest settling time. All angles are in degrees.

5.4 Comparison to other PSS approaches

In the previous sections and chapters, three different approaches to the PSS problem has been introduced: the static CPSS, auto-tuned CPSS and the SSNN applied as a PSS. In this section, a comparison of the damping performance of these approaches will be given. Also, the performance when applying no PSS models will be shown. The disturbance will be a step in the external reactance X_e, where four tests will be made: small step-up, large step-up, small step-down and large step-down. Table 5.4 shows the four steps along with corresponding plots of the response in rotor speed deviation $\Delta\omega$.

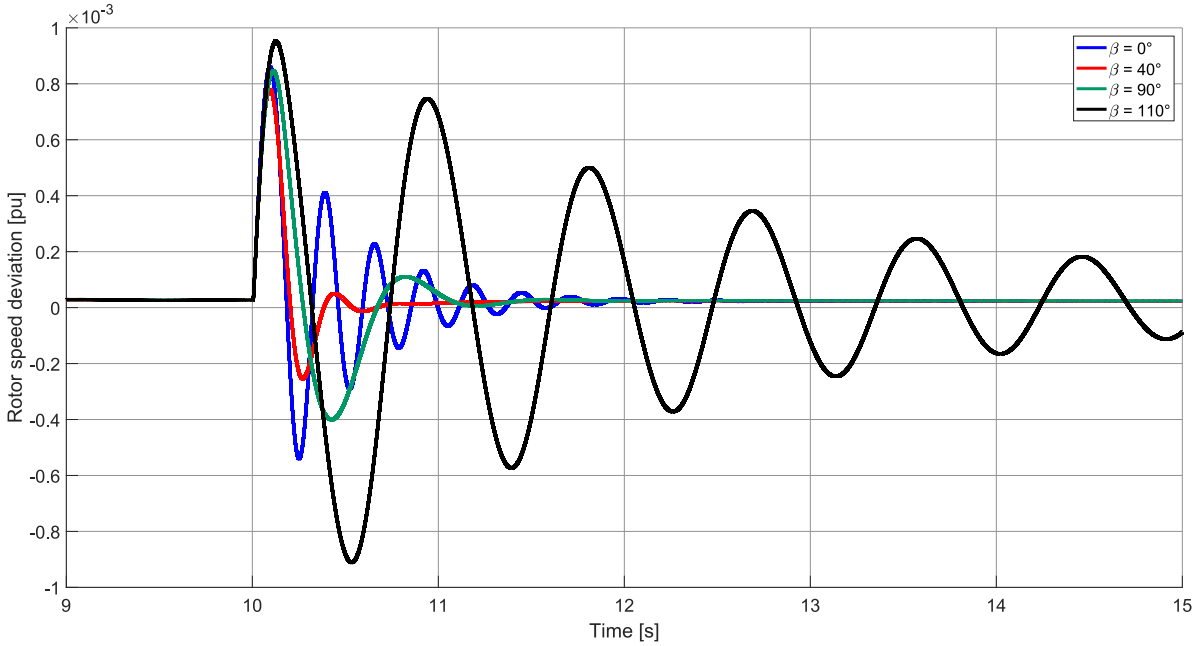


Figure 5.11: Response of the SSNN for a step in external reactance from 0.03 to 0.05 pu, at four different phase shifts β.

The CPSS model in this study has a single lead/lag stage. In the auto-tuned CPSS approach only the gain K_{pss} and lead/lag numerator time constant T_1 are auto-tuned (see Table 4.1 for their values), while the other parameters are kept constant. In the static CPSS simulations, K_{pss} and T_1 are tuned according to the pre-disturbance reactance (see Tables 4.1 and 5.4) and kept constant during the step, while the other parameters are the same as for the auto-tuned CPSS. See Appendix A for the numerical values of the constant parameters applied in the simulations. For the SSNN, the phase shift β is chosen to give the smallest settling time for each step, following Table 5.2.

Table 5.5 presents the respective settling times for the speed deviations in Figures 5.12 to 5.15 to show quantitatively how the different PSS approaches perform. The settling time is counted from when the step occurs and until the signal settles within ± 2 % of its steady-state value. As expected, applying no PSS gives very poor damping and is the worst option of the four. The auto-tuned CPSS seem to only provide improvement in damping from the static CPSS during a step-up. During a step-down, it performs worse. However, It is clear from all four step response tests that the SSNN provides superior oscillation damping, despite its imperfections described in the previous sections. It is able to reduce the settling time to well under 1 s for all the performed tests, while the other approaches are in the 2-4 s range.

Pre-disturbance X_e [pu] \ Post-disturbance X_e [pu]	**0.005**	**0.01**	**0.02**	**0.03**	**0.04**	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**	**0.1**
0.005	N/A	40	40	60	60	60	60	60	60	60	60
0.01	60	N/A	60	60	60	60	60	60	60	60	60
0.02	70	70	N/A	60	60	50	60	60	60	50	60
0.03	70	70	70	N/A	60	60	60	60	60	60	70
0.04	60	60	70	70	N/A	60	50	50	60	50	60
0.05	60	60	60	70	70	N/A	50	50	50	50	50
0.06	60	50	60	70	70	70	N/A	50	50	50	50
0.07	60	60	60	50	60	70	60	N/A	40	40	50
0.08	60	60	60	60	60	60	60	60	N/A	50	50
0.09	60	60	60	60	60	60	60	60	60	N/A	70
0.1	60	60	60	60	60	60	60	60	60	50	N/A

Table 5.3: Phase shift angles β for several pre- and post-disturbance external reactances X_e giving the smallest second-swing overshoot. All angles are in degrees.

Pre-disturbance X_e [pu]	**Post-disturbance X_e [pu]**	**Response in $\Delta\omega$**
0.03	0.05	Figure 5.12
0.005	0.1	Figure 5.13
0.05	0.03	Figure 5.14
0.1	0.005	Figure 5.15

Table 5.4: Performed reactance steps and responses in speed deviation for comparison of the four PSS approaches: No PSS, static CPSS, auto-tuned CPSS and the SSNN applied as a PSS.

Reactance step [pu]	Settling time [s]			
	No PSS	Static CPSS	Auto-tuned CPSS	SSNN
$0.03 \rightarrow 0.05$	>5	2.482	2.339	0.621
$0.005 \rightarrow 0.1$	>5	2.909	2.085	0.878
$0.05 \rightarrow 0.03$	>5	2.732	3.161	0.537
$0.1 \rightarrow 0.005$	>5	3.474	4.331	0.710

Table 5.5: Settling time (to ± 2 % of the steady-state value) for each reactance step across all PSS approaches. The SSNN is able to get the settling time well below 1 s, which is much better than the other approaches.

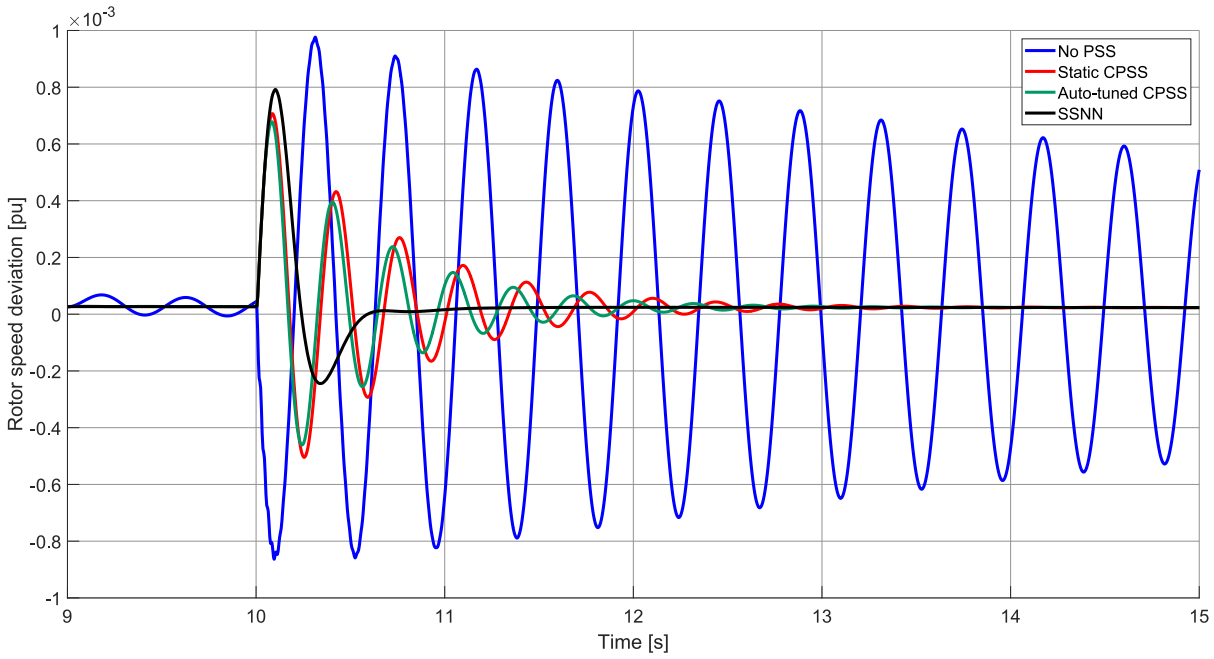


Figure 5.12: Comparison of PSS approaches for a step in X_e from 0.03 to 0.05 pu

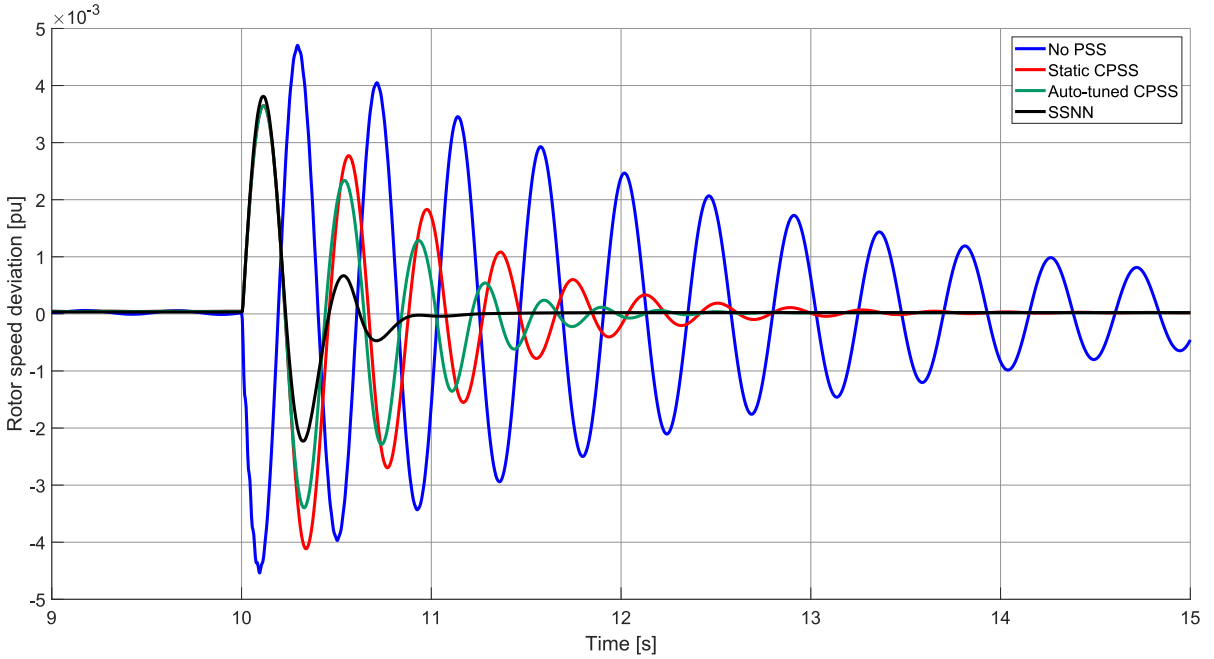


Figure 5.13: Comparison of PSS approaches for a step in X_e from 0.005 to 0.1 pu

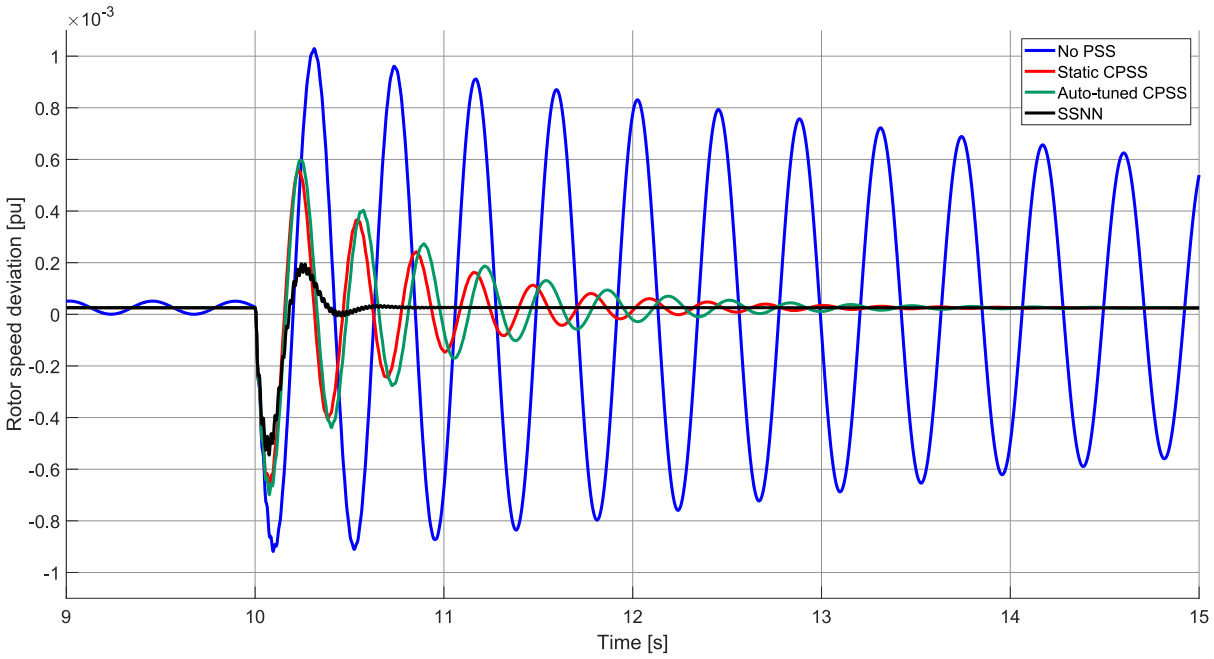


Figure 5.14: Comparison of PSS approaches for a step in X_e from 0.05 to 0.03 pu

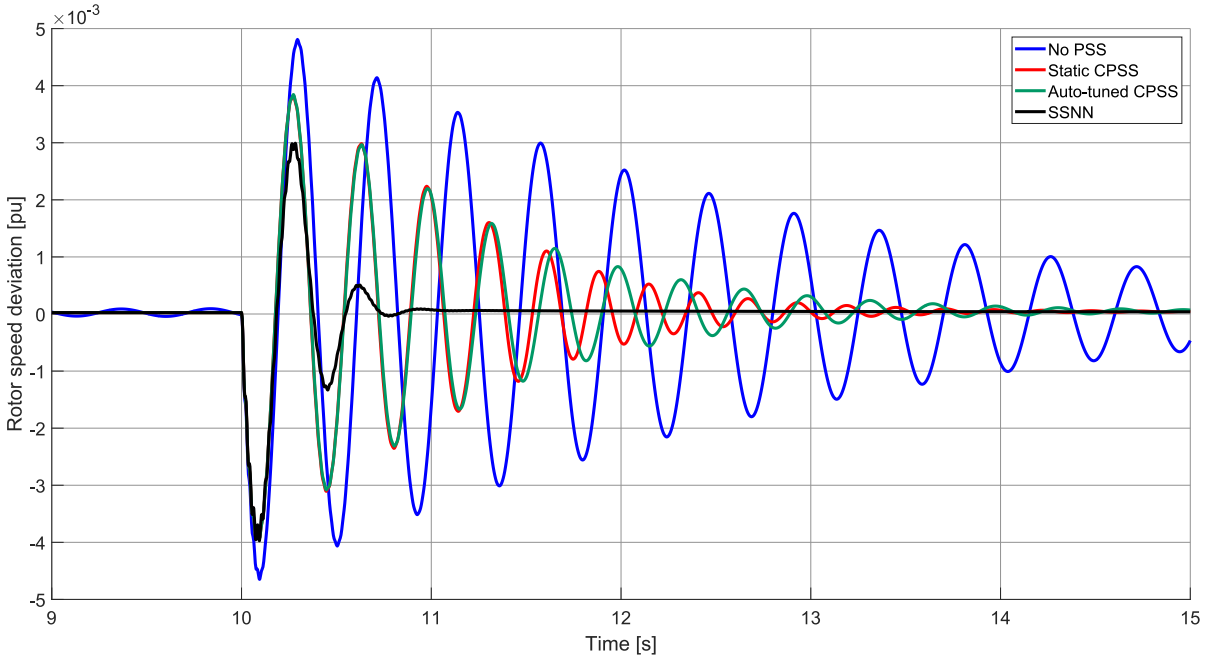


Figure 5.15: Comparison of PSS approaches for a step in X_e from 0.1 to 0.005 pu

Chapter 6

Discussion and conclusion

6.1 Discussion

This section gives a discussion of the results and methodology of the work and highlights possible pitfalls that may have been encountered, providing useful insight for potential reproductions.

It is important to realise that many components have been idealised in this thesis. In a real-life situation, no component is ideal. Considerations of instrument transformers, measurement noise, thyristor firing, saturation effects etc. have been omitted entirely. Their omission was necessary due to the limited time of a master thesis. Nevertheless, any lost accuracy due to this does not impact the conclusions and take-aways of the work in any meaningful way.

When doing computer simulations one must keep in mind that simulations are only simplified reflections of reality. However, sophisticated software solutions such as MATLAB/Simulink can provide simulations that are more than accurate enough for engineering applications. Still, user-defined settings impact accuracy. In the discrete simulations performed in this thesis, a sampling time of $3 \cdot 10^{-5}\ s$ was used, to minimise the required size of the neural network training data set. This sampling time was found to be close to its maximum value still giving expected behaviour of the machine model. Even though the simulation behaved as expected, there is some risk of lost accuracy in running it close to the maximum sampling time. This could also explain the ripples observed in the SSNN responses in Figures 5.14 and 5.15. Additionally, in real-life measurements the sampling time is usually much larger than this, meaning any real implementation needs to be adapted to the measurements' sampling rate.

The two algorithms introduced in Chapter 3 are both well-established techniques with multiple variations and adaptations. The PSO variation was chosen due to the easily accessible Global Optimisation Toolbox, and the NN structure was chosen based on its simplicity. This means that the algorithm variations implemented in this thesis may not be the best-performing for this application. In addition, reproducing results acquired from PSO or NN 100 % accurately is not possible, since the PSO procedure has stochastic elements, and retraining a NN from the same data will give a unique outcome every time. Using neural networks is a simple way to "brute force" adequate solutions to complex problems. This can be an advantage for engineers, as one may solve an advanced problem without needing to go into all the gritty details underlying the problem. Still, the same argument can be used to claim it a disadvantage, as a good understanding of the topic becomes less of a requirement to solve it.

As the time domain plots of the machine system model show, Figures 5.11 to 5.15, the rotor speed deviation does not settle to the expected steady-state value of zero. This is most likely a result of the Simulink solver. If the simulations were run for a long time, this error would slowly dissipate. Regardless, the error is very small, in the 10^{-5} pu range, and is only visible due to the relatively small perturbations performed in these tests. Thus, the error has little impact on the electrical models. However, the SSNN has been trained purely on sine waves centred on the zero-line. Therefore, it can get "confused" by the steady-state error and produce unexpected results. Again, this error is small enough for the impact to be small, but an impact nonetheless.

Chapter 4 introduced a procedure for auto-tuning the conventional power system stabiliser (CPSS). The creation and implementation of the auto-tuner were successful, yet its performance was not consistently better than the static CPSS. Since this approach only gave improved damping when the step made the PSS gain increase, it implies that the gains obtained from the PSO were lower than optimal. Most likely, there is a discrepancy between the linear model derived in Section 2.3 and the dynamic model in Simulink, making the optimal parameters in one different from the other. Moreover, using just the instantaneous value of the external reactance X_e as input to the auto-tuner may have been insufficient. Adding elements to the input such as its rate of change, or even machine measurements such as power delivery or speed deviation, could improve the prediction of optimal parameters. Additionally, giving the tuning parameters different priorities to the tuner might improve the performance further, as the PSS gain likely is of greater importance to the damping than the lead/lag time constants. Nonetheless, the auto-tuning system works and can be implemented further. Only the parameter optimisation needs to be revisited.

The sine shifting neural network (SSNN) was introduced in Chapter 5. Evaluation of the SSNN shows that a properly designed and implemented neural network can greatly improve the rotor oscillation damping. Its design is based upon the fact that the speed deviation can be considered a damped sinusoid, such that it does not rely on any heavy electrical machine modelling. This will consequently limit its range to oscillatory behaviour, making it unfit at responding properly to steady-state errors or non-oscillatory instabilities. Still, the design shows adaptability in that it can easily be retrained to work optimally on any frequency of oscillation. Moreover, having the phase shift of the PSS as a control variable gives great control over its response and can be adaptively adjusted. When also considering the gain external to the NN, the design gives full control over the gain and phase shift of the response, giving the operator a highly adaptable controller for responding to oscillatory behaviour. This also implies that the SSNN design is not limited to electrical engineering applications, but can also be applied to other fields where sinusoidal signals are prominent.

The work in this thesis has answered, or partially answered, all objectives stated in Section 1.3. The most effort went into the synchronous machine modelling and the neural network designs. A few dead ends in modelling approaches were encountered before the final designs were reached. This unfortunately limited the time available to explore the proposed solutions further.

6.2 Conclusion

This thesis has explored two ways of applying neural networks (NN) to create an adaptive power system stabiliser (PSS) design for improved damping of rotor oscillations in a salient-pole synchronous machine.

The first approach involved expanding the conventional PSS (CPSS) with an NN auto-tuner. The NN was trained on optimised PSS parameters obtained from the particle swarm optimisation (PSO) technique, optimising a simplified linear model of a synchronous machine connected to an infinite bus through an external (Thévenin) impedance. The auto-tuner observes the instantaneous value of the external reactance X_e and continuously tunes the PSS accordingly. It was shown how the auto-tuning system can be successfully implemented into Simulink. However, even though the auto-tuner functioned as intended, the proposed parameter optimisation did not result in a consistent improvement in oscillation damping from the static CPSS.

The second approach involved introducing the sine shifting neural network (SSNN) controller, where its phase response is a controlled input. It takes an oscillating signal and phase shifts it by the input angle β. The oscillating input is the rotor speed deviation, and along with a gain the SSNN replaces the CPSS in its entirety. The great advantage of the proposed design is its simplicity in that it does not rely on any electrical machine modelling. Implementing the SSNN as a PSS shows that it can significantly improve the rotor oscillation damping from the CPSS designs. Where the static and auto-tuned CPSS consistently had a settling time of 2-4 s in the performed step-response tests, the SSNN was able to reduce it to under 1 s. Giving the SSNN a phase shift β of $40 - 70°$ has proven to give the best damping performance throughout several step-response tests.

The work in this thesis shows that neural networks have great potential in improving the design of power system stabilisers. They can be implemented to dynamically tune existing conventional PSSs, or to replace the conventional design entirely. The implication of this work is a potential for better-performing synchronous machines as the modern power system progresses towards a larger share of renewable energy.

6.3 Suggestions for further work

This thesis only skims the surface of the topic at hand. There is much to be explored and expanded from this work.

6.3.1 Do a large-signal disturbance study

This thesis has been purely focused on small-signal stability. However, it would be interesting to analyse the large-signal response of the proposed systems. In that case, some of the assumptions made do no longer hold. For example, saturation effects in the machine and controller limiters are more prominent when the disturbance is large.

6.3.2 Expand the optimisation scope

This work has only studied the PSS for a single-machine system. There are a few ways the scope to be expanded for interesting studies:

- Include the AVR in the optimisation. The AVR design could be simple, as in this thesis, or a more advanced model
- Implement a turbine system model to include the governor action
- Replace the infinite bus with a small power system, where each machine is optimised with the proposed methods.

6.3.3 Improve the CPSS auto-tuner

As discussed, the auto-tuner proposed in Chapter 4 did not provide consistent improvement in damping to the machine. This was likely due to the optimisation not matching the testing system in Simulink. A different approach to the optimisation could be explored to alleviate this, be it a different linear model variation, initial conditions definition, optimising algorithm or a different approach altogether. Nonetheless, the auto-tuning system from this work can easily be adapted for a new optimisation by re-training the neural network.

6.3.4 Improve the SSNN design

As was discussed in Section 5.2, the SSNN controller does not perform optimally on all conditions, due to its relatively simple NN structure. As suggested in Section 5.2.1, a more advanced NN, such as a recurrent NN, might be more fitted as a versatile sine predictor. This could resolve the issues of unwanted attenuation or gain, phase drift and inaccurate response to non-oscillatory errors. Additionally, there is an opportunity of automatically tuning the external SSNN gain and phase shift β to changes in the operating condition. It would require a more comprehensive study to derive an optimiser for these values.

References

[1] T. Grong, "Intelligent algorithms for automatic excitation system tuning," Department of Electric Power Engineering, NTNU – Norwegian University of Science and Technology, Project report in TET5505, Dec. 2019.

[2] Energy Facts Norway, "Electricity production," Accessed: 2020-05-13. [Online]. Available: https://energifaktanorge.no/en/norsk-energiforsyning/kraftproduksjon/#hydropower

[3] R. C. Schaefer and K. Kim, "Auto tuning speeds commissioning of the generator excitation system," in *Conference Record of 2014 Annual Pulp and Paper Industry Technical Conference*, 2014, pp. 137–143.

[4] F. P. Demello and C. Concordia, "Concepts of Synchronous Machine Stability as Affected by Excitation Control," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-88, no. 4, pp. 316–329, 1969.

[5] M. A. Abido, "Optimal design of power-system stabilizers using particle swarm optimization," *IEEE Transactions on Energy Conversion*, vol. 17, no. 3, pp. 406–413, Sep. 2002.

[6] A. El-Zonkoly, "Optimal tuning of power systems stabilizers and AVR gains using particle swarm optimization," *Expert Systems With Applications*, vol. 31, no. 3, pp. 551–557, 2006.

[7] M. Shafiullah, M. Juel Rana, M. Shafiul Alam, and M. Abido, "Online tuning of power system stabilizer employing genetic programming for stability enhancement," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 3, pp. 287–299, 2018.

[8] A. L. B. Do Bomfim, G. N. Taranto, and D. M. Falcao, "Simultaneous tuning of power system damping controllers using genetic algorithms," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 163–169, Feb 2000.

[9] M. Ramirez-Gonzalez and O. P. Malik, "Simplified Fuzzy Logic Controller and Its Application as a Power System Stabilizer," in *2009 15th International Conference on Intelligent System Applications to Power Systems*, Nov 2009, pp. 1–6.

[10] R. Khezri and H. Bevrani, "Fuzzy-based coordinated control design for AVR and PSS in multi-machine power systems," in *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, Aug 2013, pp. 1–5.

[11] V. Ravi and K. Duraiswamy, "Effective optimization technique for power system stabilization using Artificial Bee Colony," in *2012 International Conference on Computer Communication and Informatics*, Jan 2012, pp. 1–6.

[12] M. Eslami and H. Shareef, "Artificial bee colony algorithm for optimal design of power system stabilizer," in *2012 10th International Power Energy Conference (IPEC)*, Dec 2012, pp. 1–6.

[13] N. N. Islam, M. A. Hannan, H. Shareef, and A. Mohamed, "Power system stabilizer design using BAT optimization algorithm in multimachine power system," in *2013 IEEE Student Conference on Research and Developement*, Dec 2013, pp. 540–544.

[14] D. K. Sambariya and R. Gupta, "Effective PID-PSS design using Bat algorithm for SMIB power system," in *2016 IEEE 6th International Conference on Power Systems (ICPS)*, March 2016, pp. 1–6.

[15] B. Sumanbabu, S. Mishra, B. K. Panigrahi, and G. K. Venayagamoorthy, "Robust tuning of modern power system stabilizers using Bacterial Foraging Algorithm," in *2007 IEEE Congress on Evolutionary Computation*, Sep. 2007, pp. 2317–2324.

[16] T. K. Das, G. K. Venayagamoorthy, and U. O. Aliyu, "Bio-Inspired Algorithms for the Design of Multiple Optimal Power System Stabilizers: SPPSO and BFA," *IEEE Transactions on Industry Applications*, vol. 44, no. 5, pp. 1445–1457, Sep. 2008.

[17] T. Mulumba and K. A. Folly, "Application of Self-Adaptive Differential Evolution to tuning PSS parameters," in *IEEE Power and Energy Society Conference and Exposition in Africa: Intelligent Grid Integration of Renewable Energy Resources (PowerAfrica)*, July 2012, pp. 1–5.

[18] B. Selvabala and D. Devaraj, "Co-ordinated tuning of AVR-PSS using differential evolution algorithm," in *2010 Conference Proceedings IPEC*, Oct 2010, pp. 439–444.

[19] M. A. Abido and Y. L. Abdel-Magid, "Robust design of electrical power-based stabilizers using tabu search," in *2001 Power Engineering Society Summer Meeting. Conference Proceedings (Cat. No.01CH37262)*, vol. 3, July 2001, pp. 1573–1578 vol.3.

[20] S. Dechanupaprittha and I. Ngamroo, "Design of robust power system stabilizers in a multimachine power system using tabu search algorithm," in *2002 IEEE International Conference on Industrial Technology, 2002. IEEE ICIT '02.*, vol. 1, Dec 2002, pp. 291–296 vol.1.

[21] M. A. Abido, "Robust design of multimachine power system stabilizers using simulated annealing," *IEEE Transactions on Energy Conversion*, vol. 15, no. 3, pp. 297–304, Sep. 2000.

[22] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of mathematical biology.*, vol. 5, no. 4, pp. 115–133, 1943.

[23] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review : published bimonthly by the American Psychological Association*, vol. 65, no. 6, pp. 386–408, 1958.

[24] Yuan-Yin Hsu and Chao-Rong Chen, "Tuning of power system stabilizers using an artificial neural network," *IEEE Transactions on Energy Conversion*, vol. 6, no. 4, pp. 612–619, 1991.

[25] A. M. Sharaf, T. T. Lie, and H. B. Gooi, "Neural network based power system stabilizers,"

in *Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, 1993, pp. 306–309.

[26] Y. Zhang, G. P. Chen, O. P. Malik, and G. S. Hope, "An artificial neural network based adaptive power system stabilizer," *IEEE Transactions on Energy Conversion*, vol. 8, no. 1, pp. 71–77, 1993.

[27] Y. Zhang, O. P. Malik, and G. P. Chen, "Artificial neural network power system stabilizers in multi-machine power system environment," *IEEE Transactions on Energy Conversion*, vol. 10, no. 1, pp. 147–155, 1995.

[28] L. Guan, S. Cheng, and R. Zhou, "Artificial neural network power system stabiliser trained with an improved BP algorithm," *IEE Proceedings - Generation, Transmission and Distribution*, vol. 143, no. 2, pp. 135–141, 1996.

[29] Young-Moon Park, Myeon-Song Choi, and K. Y. Lee, "A neural network-based power system stabilizer using power flow characteristics," *IEEE Transactions on Energy Conversion*, vol. 11, no. 2, pp. 435–441, 1996.

[30] P. Shamsollahi and O. P. Malik, "An adaptive power system stabilizer using on-line trained neural networks," *IEEE Transactions on Energy Conversion*, vol. 12, no. 4, pp. 382–387, 1997.

[31] B. Changaroon, S. C. Srivastava, and D. Thukaram, "A neural network based power system stabilizer suitable for on-line training-a practical case study for egat system," *IEEE Transactions on Energy Conversion*, vol. 15, no. 1, pp. 103–109, 2000.

[32] Wenxin Liu, G. K. Venayagamoorthy, and D. C. Wunsch, "Adaptive neural network based power system stabilizer design," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 4, 2003, pp. 2970–2975 vol.4.

[33] Nguyen and Widrow, "The truck backer-upper: an example of self-learning in neural networks," in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 357–363 vol.2.

[34] L. H. Hassan, M. Moghavvemi, and H. A. F. Mohamed, "Power system stabilization based on artificial intelligent techniques; a review," in *2009 International Conference for Technical Postgraduates (TECHPOS)*, 2009, pp. 1–6.

[35] P. Kundur, *Power system stability and control*. McGraw-Hill, 1994.

[36] J. Machowski, *Power system dynamics - Stability and control*, 2nd ed. Wiley, 2008.

[37] P. M. Anderson and A. A. Fouad, *Power System Control and Stability*, 2nd ed. John Wiley & Sons, 2002.

[38] E. Clarke, "Circuit analysis of A-C power systems : 1 : Symmetrical and related components," New York, 1943.

[39] R. H. Park, "Two-reaction theory of synchronous machines generalized method of analysis-part i," *Transactions of the American Institute of Electrical Engineers*, vol. 48, no. 3, pp. 716–727, 1929.

[40] Statnett, "Funksjonskrav i kraftsystemet," 2012.

[41] Statnett, "Nasjonal Veileder for Funksjonskrav i kraftsystemet," 2020.

[42] J. Nøland, S. Nuzzo, A. Tessarolo, and E. Alves, "Excitation System Technologies for Would-Field Sunchronous Machines: Survey of Solutions and Evolving Trends," *IEEE Access*, vol. 7, 2019.

[43] "IEEE Recommended Practice for Excitation System Models for Power System Stability Studies," *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005)*, pp. 1–207, Aug 2016.

[44] E. V. Larsen and D. A. Swann, "Applying Power System Stabilizers part I: General Concepts," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-100, no. 6, June 1980.

[45] M. J. Gibbart, *Small-signal stability, control and dynamic performance of power systems*. Adelaide: The University of Adelaide Press, 2015.

[46] B. Pal and B. Chaudhuri, *Robust Control in Power Systems*, ser. Power Electronics and Power Systems. Boston, MA: Springer US, 2005.

[47] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Nov 1995, pp. 1942–1948.

[48] The MathWorks, Inc., "Global Optimization Toolbox," 2020, Accessed: 2020-06-01. [Online]. Available: https://www.mathworks.com/products/global-optimization.html

[49] M. R. Bonyadi and Z. Michalewicz, "Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review," *Evolutionary Computation*, vol. 25, no. 1, pp. 1–54, 2017.

[50] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *2007 IEEE Swarm Intelligence Symposium*, 2007, pp. 120–127.

[51] The MathWorks, Inc., "Particle swarm," https://www.mathworks.com/help/gads/particle-swarm.html, 2020, Accessed: 2020-04-28,.

[52] A. Zaknich, *Principles of Adaptive Filters and Self-learning Systems*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer London, 2005.

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[54] D. F. Shanno, "Conditioning of quasi-newton methods for function minimization," *Mathematics of computation.*, vol. 24, no. 111, pp. 647–647, 1970.

[55] M. R. Hestenes, E. Stiefel *et al.*, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[56] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[57] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[58] A. Kattan, "Artificial neural network training and software implementation techniques," Hauppauge, N.Y., 2011.

[59] A. Abraham, "Meta learning evolutionary artificial neural networks," *Neurocomputing: an international journal*, vol. 56, pp. 1–38, 2004.

[60] B. C. Csáji *et al.*, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, no. 48, p. 7, 2001.

[61] The MathWorks, Inc., "Deep Learning Toolbox," 2020, Accessed: 2020-05-16. [Online]. Available: https://www.mathworks.com/products/deep-learning.html

[62] M. Leijon, F. Owman *et al.*, "Powerformer(r): a giant step in power plant engineering," in *IEEE International Electric Machines and Drives Conference. IEMDC'99. Proceedings (Cat. No.99EX272)*, May 1999, pp. 830–832.

[63] The MathWorks, Inc., "How Acceleration Modes Work," 2020, Accessed: 2020-05-18. [Online]. Available: https://www.mathworks.com/help/simulink/ug/how-the-acceleration-modes-work.html

[64] ——, "Save and Restore Simulation Operating Point," 2020, Accessed: 2020-05-18. [Online]. Available: https://www.mathworks.com/help/simulink/ug/saving-and-restoring-simulation-operating-point.html

[65] P. Melin, *Nature-Inspired Design of Hybrid Intelligent Systems.*, ser. Studies in Computational Intelligence Ser, 2016, vol. 667.

[66] The MathWorks, Inc., "Fit Data with a Shallow Neural Network," 2020, Accessed: 2020-06-01. [Online]. Available: https://www.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html

[67] ——, "Parallel Computing Toolbox," 2020, Accessed: 2020-06-01. [Online]. Available: https://www.mathworks.com/products/parallel-computing.html

Appendicies

There are two appendices for this thesis:

Appendix A contains tables with the numeric values of all constant model parameters used in the simulations during the work of this thesis. Additionally, the Simulink solver settings and software versions are provided.

Appendix B provides a guide on how a simple feedforward neural network, as applied in this thesis, can be created in MATLAB.

Appendix A

Model parameters used

Throughout this thesis, several simulations have been made. The constant model parameters have been consistent for all simulations. This appendix gives an overview of these parameters.

The machine has been considered a would-field salient-pole synchronous machine. Table A.1 gives the parameters entered into the machine model in Simulink. Table A.2 shows the constant parameters of the excitation system and the external network. Table A.3 gives the parameters of the simplified linear model that are not specified elsewhere.

Parameter	**Symbol**	**Value**	**Unit**
Rated power	S_n	32	MVA
Rated voltage	V_n	11.5	kV
Nominal frequency	f_n	50	Hz
d-axis synchronous reactance	X_d	1.18	pu
d-axis transient reactance	$X_d^{'}$	0.26	pu
d-axis subtransient reactance	$X_d^{''}$	0.16	pu
q-axis synchronous reactance	X_q	0.8	pu
q-axis subtransient reactance	$X_q^{''}$	0.16	pu
Stator winding leakage reactance	X_l	0.1	pu
d-axis transient short-circuit time constant	$T_d^{'}$	1.3	s
d-axis transient open-circuit time constant	$T_{d0}^{'}$	6	s
d-axis subtransient short-circuit time constant	$T_d^{''}$	0.06	s
q-axis subtransient short-circuit time constant	$T_q^{''}$	0.06	s
Stator winding resistance	R_s	0.003	pu
Pole pairs	pp	6	
Inertia constant	H	2.51	s
Friction and windage damping factor	F	0	pu

Table A.1: Parameters of the synchronous machine model, as entered into Simulink.

The names and versions of the software used to perform calculations, simulations and plotting in the work of this thesis are given in Table A.4. The solver settings used in Simulink is given in Table A.5. Here, the settings from both the continuous and discrete simulations are shown.

Parameter	Symbol	Value	Unit
AVR gain	K_a	200	
AVR output limits	$V_{f,max}$, $V_{f,min}$	± 5	pu
PSS output limits	$V_{pss,max}$, $V_{pss,min}$	± 0.15	pu
PSS lead/lad denominator time constant	T_2	0.05	s
PSS wash-out filter time constant	T_w	2	s
Reference voltage	V_{ref}	1	pu
Constant mechanical torque	P_m	1	pu
Step-up transformer reactance	X_t	0.8	pu
External network resistance	R_e	0.0005	pu
Infinite bus voltage (Line-Line RMS)	V_{inf}	1	pu

Table A.2: Parameters of the excitation system and external network

Parameter	Symbol	Value	Unit
Generator power factor (lagging)	PF	0.8	
Damping coefficient	K_D	3	pu
Terminal voltage trancducer gain	K_r	1	
Terminal voltage transducer time constant	T_r	0.2	s

Table A.3: Parameters of the linear generator-infinite bus system not specified elsewhere

Software name	Version
MATLAB	R2018a update 6
Simulink	9.1 (R2018a)
Simscape	4.4
Simscape Power Systems	6.9
Optimization toolbox	8.1
Global optimization toolbox	3.4.4
Neural network toolbox[1]	11.1
Parallel Computing Toolbox	6.12
Control system toolbox	10.4
Maple	2018.2

Table A.4: Names and versions of software used in this thesis for calculations, simulations and plotting.

When training the neural networks in this thesis, the network parameters were set as in Table A.6. The settings are shown for both the CPSS auto-tuner in Chapter 4 and the SSNN in Chapter 5. Lastly, the parameters used in the PSO algorithm is given in Table A.7. The parameters not specified here are kept at their default values as given in [51].

[1]The Neural Network toolbox has been integrated into the Deep Learning Toolbox in later MATLAB versions, which has been referred to regularly in the thesis.

Setting name	Continuous	Discrete
Solver	*ode23tb variable-step*	*ode23tb variable-step*
Max step size	$5 \cdot 10^{-4}$ s	$3 \cdot 10^{-5}$
Min step size	$1 \cdot 10^{-14}$ s	auto
Relative tolerance	$1 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
Zero-crossing control	*Enable all*	*Enable all*
Zero-crossing algorithm	*Adaptive*	*Adaptive*

Table A.5: Solver settings in Simulink for both the continuous and discrete simulations

Setting	CPSS tuner	SSNN
Hidden layer size	10	10
Training subset ratio	70 %	75 %
Testing subset ratio	15 %	10 %
Validation subset ratio	15 %	15 %
Training algorithm	LM	LM
Error function	MSE	MSE
Activation function	tanh	tanh

Table A.6: Settings used when creating the neural networks in this thesis

Setting	Value
K_{pss} bounds	[1,400]
T_1 bounds	[0.001,4]
Swarm size	200
c_1 and c_2	1.49

Table A.7: Settings used in the PSO algorithm

Appendix B

Creating a neural network in MATLAB

In this appendix, a brief guide will be given on how to create and train a shallow (3-layer) feedforward neural network in MATLAB. The *Deep Learning Toolbox* [61] is required for this procedure. As this thesis has used NNs for function fitting, this is the type being focused on in this appendix. A similar guide is also provided in [66].

Step 1 - Pre-process training data
The training data needs to be formatted in a particular way for the training function to interpret it correctly. The inputs need to be put in an *R-by-Q* matrix, where *R* is the number of inputs, and *Q* is the number of data points. The corresponding targets must be put in an *U-by-Q* matrix, where U is the number of outputs of the network. The input matrix will be denoted as X and the target matrix as T.

Step 2 - Determine network parameters
In any NN application – MATLAB or not – some network and training parameters must be chosen. Firstly, choose the number of neurons in the hidden layer, the default being 10. Next, determine how much of the data set should be reserved for training, testing and validation, the default being 70 %, 15 % and 15 % respectively. Then, decide which training algorithm should be applied. The default is the Levenberg-Marquardt algorithm.

Step 3 - Define network
First, delcare the network as a function fitting neural network with, say 12 hidden neurons. Add the argument `'trainlm'` to define the Levenberg-Marquardt training algorithm,

```
hls = 12; %Hidden layer size
tralg = 'trainlm'; %Training algorithm
net = fitnet(hls,tralg); %declare fitnet
```

Next, define the training data set division,

```
net.divideParam.trainRatio = 75/100; %Training data subset
net.divideParam.valRatio = 10/100; %Validation data subset
net.divideParam.testRatio = 15/100; %Testing data subset
```

Step 4 - Train network
The training of the network is initated by,

```
[net,tr] = train(net,X,T);
```

When this command is run, the NN training window opens, which contains information about the progress and allows for interruption at any time.

If the *Parallel Computing Toolbox* [67] is installed, adding the argument `'useParallel','yes'` will start a local parallel pool with the available processor cores. Alternatively, adding the argument `'UseGPU','yes'` with an appropriate GPU installed, the training will employ GPU processing. Using this toolbox is recommended if appropriate hardware is accessible, as it will greatly increase training speed.

Step 5 - Validate network

The resulting variable `net` will be the trained network and `tr` will contain the training record, which is information about the training data, epochs and performance. To check the NN performance, one can compare its output to the targets. Conveniently, `tr` has the performance of the training, testing and validation set stored. The performance measure is the same as the network was trained with, the default being the mean square error.

```
perf_tr = tr.best_perf; %Getting performance of training set
perf_test = tr.best_tperf; %Getting performance of testing set
perf_val = tr.best_vperf; %Getting performance of validation set
```

Alternatively, using the `perform` function, the network's performance to the validation data set can be checked. `tr.valInd` contains the indices of X that are reserved for the validation subset.

```
Xvalset = X(tr.valInd); %Get the validation set from X
Tvalset = T(tr.valInd); %Get the validation set from T
y = net(Xvalset); %Get NN output for the validation set
perf = perform(net,Tvalset,y); %Get performance measure of validation set
```

The performance of all three subsets should be of similar size. The network can be retrained with different hidden layer size, training algorithm etc. to check if the performance improves.

When applying the trained NN iteratively in a script, using `output = net(input)` has been found to be slow. Rather, a separate MATLAB function can be created with `genFunction(net)`. Calling this function rather than the *net* object is much faster

To get the trained network onto a Simulink model, the `genSim(net)` function is very convenient. It automatically generates a NN block in the Simulink environment that can easily be copied into the relevant model.

Alternative way of creating the NN

The *Deep Learning Toolbox* comes with a handy tool to train simple networks such as this. Typing `nnstart` in the MATLAB command window opens the NN tool start page. The fitting app is appropriate for this application. Alternatively, `nftool` could be entered directly to the command window. The tool provides a user-friendly interface to perform all the actions described in the above steps. The drawback is having slightly less control over all the parameters involved and less automation due to the required user engagement.

Thomas Grong

Adaptive Neural Network-based PSS Designs for Modern Power Systems

NTNU
Norwegian University of
Science and Technology