Efficient Modelling of a Wind Turbine System for Parameter Estimation Applications

by
Johannes Cornelius Bekker

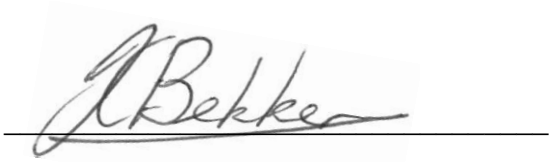
Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Engineering
at Stellenbosch University

Supervisor: Prof. H.J. Vermeulen
Faculty of Engineering
Department of Electrical and Electronic Engineering

March 2012

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.



Date: 02 March 2012

ABSTRACT

Wind energy is a very current topic, both locally and internationally. It is one of the most rapidly growing renewable energy sources with installed capacity doubling every three years. South Africa's installed wind energy currently accounts for only 10 MW of the 197 GW worldwide installed capacity. With a 10 TWh renewable energy production target set for 2013 by the South African government, renewable energy projects have gained momentum in recent years. This target, together with data from case studies and reports on resource planning and technical requirements, shows that South Africa is well positioned for the implementation of wind energy sources.

All this development in the local wind generation market creates a need for local knowledge in the field of wind energy as well as a need to efficiently model and analyse wind turbine systems and grid interactions for local operating conditions. Although the relevant model topologies are well established, obtaining or deriving appropriate parameter values from first principles remains problematic. Some parameters are also dependent on operating conditions and are best determined from site measurements using parameter estimation methodologies. One of the objectives of this project is to investigate whether the system parameter values can be obtained by performing parameter estimation on the model of a wind turbine system. The models used for parameter estimation processes require fast simulation times. Therefore, basic C-code S-function models of the wind turbine system components, i.e., the wind turbine blade, gearbox and generator, were developed and compiled as a Simulink library. These library components were then used for the parameter estimation process.

The developed models, as well as the complete wind turbine system model, were validated and their performance evaluated, by comparing them to existing Simulink block models. These models all proved to be accurate and all showed reductions in simulation times.

The principle of performing parameter estimation on C-code S-function models is proven by case studies performed on the individual models and the complete wind turbine system. The power coefficient matrix parameter values of the individual turbine blade model estimated with 100% accuracy for the excited elements. The individual gearbox parameter values all estimated accurately with errors below 2.5%. The parameter values of the individual generator models were estimated accurately for the ABC model, with errors below 4%, and less accurately for the DQ model with errors below 13%. The estimation results obtained for

the complete wind turbine system model showed that the parameter values of the gearbox model and generator model were estimated accurately when the system model was excited through a step in angular velocity and steps in amplitude of the stator voltages respectively. A final estimation showed that a combination of gearbox and generator parameter values were accurately estimated when the model was excited through both a step in angular velocity and steps in the amplitude of the stator voltages.

OPSOMMING

Windenergie is 'n baie aktuele onderwerp beide plaaslik en internasionaal. Windenergie is een van die vinnigste groeiende hernubare energie bronne met die geïnstalleerde kapasiteit wat driejaarliks verdubbel. Suid-Afrika se geïnstalleerde windenergie maak tans slegs 10 MW uit van die wêreldwye geïnstalleerde kapasiteit van 197 GW. Die Suid-Afrikaanse regering het ’n 10 TWh hernubare-energie produksie teiken gestel vir 2013. As gevolg hiervan het hernubare-energie projekte die laaste paar jaar momentum gekry. Hierdie teiken, tesame met die data van gevallestudies en verslae oor hulpbronbeplanning en tegniese vereistes, toon dat Suid-Afrika goed geposisioneer is vir die implementering van windenergiebronne.

Hierdie ontwikkelinge in die plaaslike windenergie mark skep ’n behoefte aan plaaslike kennis op die gebied van windenergie, asook die behoefte vir ’n doeltreffende wyse vir die modellering en analisering van windturbine stelsels en netwerk integrasie vir plaaslike werkskondisies. Alhoewel die betrokke model topologieë reeds goed gevestig is, is die verkryging van toepaslike parameter waardes vanuit eerste beginsels steeds problematies. Sommige parameters is ook afhanklik van die werkskondisies en kan die beste bepaal word deur gebruik te maak van parameter estimasie metodologieë vanaf terrein metings. Een van die doelwitte van die projek is om ondersoek in te stel na die moontlikheid om die stelsel parameter waardes te verkry deur parameter estimasie toe te pas op ’n windturbine stelsel. Die modelle wat gebruik word vir die parameter estimasie prosesse benodig vinnige simulasie tye. Daarom is basiese C-kode S-funksie modelle vir die komponente van windturbine stelsels, d.w.s., die wind turbine lemme, ratkas en generator, ontwikkel en saamgestel as ’n Simulink biblioteek. Die komponente in hierdie biblioteek was toe gebruik vir die parameter estimasie proses.

Die ontwikkelde modelle sowel as die hele windturbine stelsel model was gevalideer en hul werksverrigting geëvalueer, deur dit te vergelyk met bestaande Simulink blok modelle. Hierdie modelle het almal getoon dat hulle akkuraat is en het almal ’n vermindering in simulasie tyd getoon.

Die beginsel van parameter estimasie wat uitgevoer word op C-kode S-funksie modelle, is bewys deur gevallestudies wat op die individuele modelle en die hele windturbine stelsel model uitgevoer was. Die geperturbeerde elemente van die kragkoëffisiënt-matriks

parameter van die individuele turbine lemme model se waardes het 100% akkuraatheid geëstimeer. Die individuele ratkas model se parameter waardes was almal akkuraat geëstimeer, met foute kleiner as 2.5%. Die individuele generator modelle se parameter waardes was akkuraat geëstimeer vir die ABC model, met foute kleiner as 4%, en minder akkuraat vir die DQ model, met foute kleiner as 13%. Die resultate wat verkry is van die estimasie wat uitgevoer is op die volledige windturbine stelsel model, het getoon dat die parameter waardes van die ratkas model en die generator model akkuraat geëstimeer word, wanneer die stelsel model onderskeidelik deur ’n trap in die hoeksnelheid en trappe in die amplitude van die stator spannings geperturbeer word. ’n Finale estimasie het getoon dat ’n kombinasie van ratkas en generator parameter waardes akkuraat geëstimeer kan word as die model deur beide die trap in hoeksnelheid en die trappe in die amplitude van die stator spannings geperturbeer word.

ACKNOWLEDGEMENTS

I would like to thank Prof HJ Vermeulen, Department of Electrical and Electronics Engineering, University of Stellenbosch, for his invaluable contribution to this project. Thank you to the members and Ex-Officio members of Batteljon E222 for their input, assistance and support during this project. I would further like to thank my family and friends for their constant support and encouragement. I would also like to thank the Lord for providing me with good health and the ability to do this project. Finally, I thank my wife, Anél, for her patience, constant support and encouragement.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

A	Ampere
ABC	Refers to the a, b and c reference axis
C_P	Power Coefficient
C_q	Torque Coefficient
CCGT	Closed Cycle Gas Turbine
CSP	Concentrated Solar Power
CVT	Continuously Variable Transmission
DEA&DP	Department of Environmental Affairs and Development Planning
DFIG	Double-Fed Induction Generator
DoE	Department of Energy
DQ	Refers to the d and q reference axis
DSP	Digital Signal Processing
Eskom	Eskom Holdings SOC Limited
EWEA	European Wind Energy Association
FBC	Fluidised Bed Combustion
GTZ	Deutsch Gesellschaft für Technische Zusammenarbeit GmbH
GUI	Graphical User Interface
I	Current
IET	Institute of Energy Technology
IG	Induction Generator
IRP	Integrated Resource Plan
MEX	MATLAB Executable
MIMO	Multi-Input-Multi-Output
mmf	Magnetomotive Force
NERSA	National Energy Regulator of South Africa
Nm	Newton meter
OCGT	Open Cycle Gas Turbine
OOP	Object-Oriented Programming
PF	Pulverised Fuel
PM	Permanent Magnet
PV	Photo-Voltaic

Q	Reactive Power
REFIT	Renewable Energy Feed-In Tariff
RMS	Root-Mean-Square
rpm	Revolutions per minute
SANERI	South African National Energy Research Institute
TSR	Tip Speed Ration
V	Voltage
WWEA	World Wind Energy Association

1 PROJECT OVERVIEW

1.1 Introduction

This chapter presents the project overview. The project motivation and description section provides background information on the current standing of wind energy and the driving forces behind this study. This is followed by the project objectives section that describes the research objectives. The chapter concludes with a document structure overview.

1.2 Project motivation and description

This section provides an overview of the current wind industry worldwide by looking at the currently installed capacity, the development of wind turbine technology and the financial turnover. This is followed with a discussion of wind energy in the context of Africa, focusing on South Africa that looks at the currently installed capacity, legislation having an impact on the future of wind energy as well as a case study that was conducted.

Wind energy is a very current topic internationally, in Africa and specifically in the South African context. It is one of the most rapidly growing renewable energy sources with the worldwide installed capacity doubling every three years [1]. Figure 1-1 shows the growth of installed capacity over the last ten years. Even with the decline of newly installed capacity in 2010, the three year doubling trend is still visible [1].

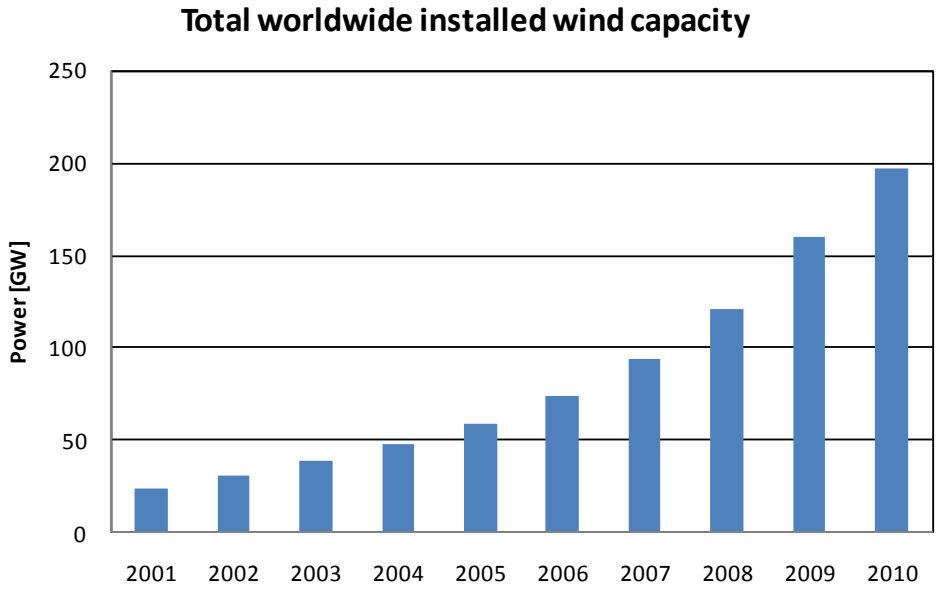


Figure 1-1: Total worldwide installed wind capacity 2001–2010 [1]**.**

This continuous growth together with the large financial turnover, which amounted to 40 billion Euro (388 billion ZAR) worldwide in 2010 [1], leads to a lot of research being conducted in the field of wind energy. This research can be seen by the advancement in wind turbine technology, illustrated in Figure 1-2, showing the increase in mechanical size and capacity per turbine. Enercon is currently producing a turbine system with a capacity of 7.5 MW [2]. Wind energy also has a great economic impact with the European wind energy sector that is currently employing 192 000 people. This number is expected to more than double by the year 2020 [3].

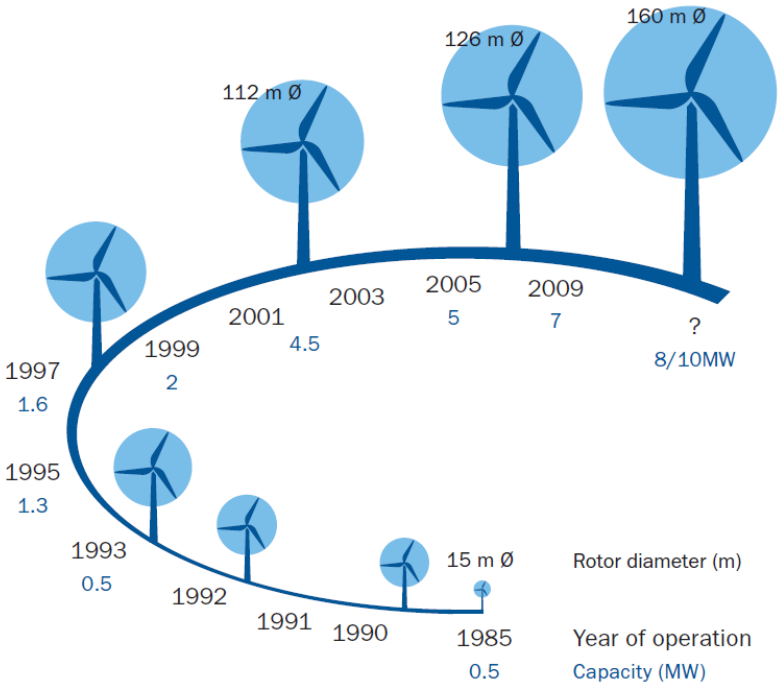


Figure 1-2: Size and power increases of commercially produced wind turbines over time [3].

With this background on the international market, the focus is shifted to the African market and, more specifically, the South African market. According to the World Wind Energy Report published in 2010 by the World Wind Energy Association (WWEA), Africa accounts for only 906 MW of worldwide installed capacity, totalling about 197 GW [1]. Egypt has the highest installed capacity with a value of 550 MW. Comparatively, South Africa accounts for only 10 MW [1, 4]. In this report the WWEA mentions that with the new Renewable Energy Feed-In Tariff (REFIT) published in 2009, South Africa has the potential to become the wind energy leader in southern Africa. The aim of the REFIT is to be "*...a mechanism to promote the deployment of renewable energy that places an obligation on specific entities to purchase the output from qualifying renewable energy generators at pre-determined prices.*" [5], with wind being one of the qualifying renewable energies. This, together with the White

Paper on renewable energy published by the South African government in 2003, which set a 10 TWh renewable energy production target for 2013 [6], shows that South Africa is well positioned for the implementation of renewable energy.

At the beginning of 2009 with the White Paper published and the REFIT process on the way, the Department of Environmental Affairs and Development Planning (DEA&DP) of the Western Cape, in conjunction with Deutsch Gesellschaft für Technische Zusammenarbeit GmbH (GTZ) and the South Africa power utility, Eskom, consulted DIgSILENT GmbH to perform a feasibility study for wind generation in the Western Cape. The study showed that as much as 2800 MW of wind energy can be integrated into the existing transmission grid [7].

One of the concerns that came forth during the case study performed by DIgSILENT was the lack of grid connection requirements for wind energy in Eskom's grid code. The consultants advised Eskom on requirements to be appended to the grid code to make provision for wind energy. During 2010, Eskom compiled a document with the grid requirements entitled "*Grid Code Requirements for Wind Turbines Connected to Distribution or Transmission Systems in South Africa*". NERSA approved this document in February 2011 [8].

In March 2011, the South African Department of Energy (DoE) published the "*Final report of the Integrated Resource Plan (IRP) for electricity 2010-2030*" [9]. The report discusses the future of the South African energy resources by considering factors such as emissions, risk, energy cost, forecasted peak demand, power plant construction and commissioning times and power plant decommissioning. Table 1-1 shows the new build option part of Policy-Adjusted IRP. The projects that are currently committed and those that urgently need firm commitment are highlighted. Looking at the wind column, a total of 9100 MW of wind energy capacity is required before 2030. This accounts for 16.3% of the planned new capacity. A total of 700 MW of this capacity is seen as committed with a further 800 MW needing firm commitment [9].

Table 1-1: IRP – New build part of adjusted IRP policy (adapted from [9]).

	New build options							
	Coal (PF, FBC imports, own build)	Nuclear	Import hydro	Gas-CCGT	Peak-OCGT	Wind	CSP	Solar PV
	MW	MW	MW	MW	MW	MW	MW	MW
2010	0	0	0	0	0	0	0	0
2011	0	0	0	0	0	0	0	0
2012	0	0	0	0	0	300	0	300
2013	0	0	0	0	1020	400	0	300
2014	500	0	0	0	0	400	100	300
2015	500	0	0	0	0	400	100	300
2016	0	0	0	0	0	400	100	300
2017	0	0	0	0	0	400	100	300
2018	0	0	0	0	0	400	100	300
2019	250	0	0	237	0	400	100	300
2020	250	0	0	237	0	400	100	300
2021	250	0	0	237	0	400	100	300
2022	250	0	1143	0	805	400	100	300
2023	250	1600	1183	0	805	400	100	300
2024	250	1600	283	0	0	800	100	300
2025	250	1600	0	0	805	1600	100	1000
2026	1000	1600	0	0	0	400	0	500
2027	250	0	0	0	0	1600	0	500
2028	1000	1600	0	474	690	0	0	500
2029	250	1600	0	237	805	0	0	1000
2030	1000	0	0	948	0	0	0	1000
Total	6250	9600	2609	2370	4930	9100	1200	8400

Firm commitment necessary now
Committed

The IRP clearly shows that there is a need for wind energy. The case study indicates that, with minor changes to the current grid, wind energy to the capacity of 2800 MW can be accommodated. The Grid Code provides the technical detail on the grid connection requirements and required turbine technology. The REFIT, that also includes the Power Purchase Agreement, provides the potential Independent Power Producers (IPPs), investors and wind turbine manufacturers with a basis for developing wind energy in South Africa.

All this development in the local wind generation market creates a need for local knowledge in the field of wind energy, as well as the need to model and analyse wind turbine systems and grid interactions for local operating conditions in an efficient manner [10, 11]. The South African National Energy Research Institute (SANERI) has realised this need for research in the field of sustainability and has initiated the Hub and Spokes programme. The purpose of this programme is to create a centre of excellence in a specific area of energy research and development, i.e., the hub, by working with other tertiary education institutes on a multi-lateral basis, i.e., the spokes [12]. In 2006, the contract for the Renewable and Sustainable Energy Hub was awarded to the University of Stellenbosch, with wind energy

technologies being one of the spokes assigned to Stellenbosch University in conjunction with the University of Cape Town [12]. This created the opportunity to conduct a study on the modelling, implementation and experimentation on the models of a wind turbine system.

Models of the different components of wind turbine systems already exist and are implemented by simulation software packages, e.g., DIgSILENT. These models require parameter values of real wind turbine systems. Although these parameter values are supplied by the turbine manufacturers, the data for these parameters are not always reliable. Some of the parameter values change as the components age or if a component is in the process of breaking down. Some of the parameter values also depend on the operating condition of the system. Therefore, the parameter values supplied by the manufacturer might not be applicable for the local operating conditions, e.g., turbulence flow of the wind or grid behaviour. The values of these parameters are therefore best determined from site measurements using parameter estimation. Furthermore one of the emerging applications of parameter estimation is condition monitoring of systems.

This project was initiated to investigate whether the values of the system parameters could be obtained by performing parameter estimation on the model of a wind turbine system. Figure 1-3 shows the block diagram of a wind turbine system.

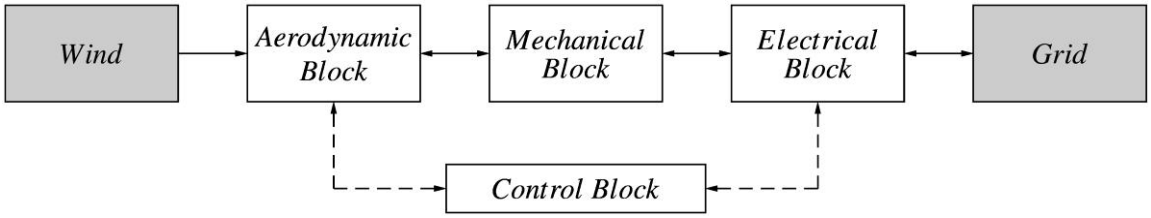


Figure 1-3: Complete wind turbine system [13]**.**

The models used for parameter estimation processes require fast simulation times, since the models are simulated numerous times. As mentioned, models for wind turbine systems already exist, but these models are mostly developed for forward simulation. As a result, not to great emphasis are placed on the simulation time. This gives rise to the need for implementation of models with fast simulation times for the dynamic modelling of a wind turbine system to use with parameter estimation.

The main aim of this study is not to develop and implement models for a wind turbine system, but to improve simulation times for parameter estimation applications. Therefore, basic models for the aerodynamic, mechanical and electrical blocks are developed and their

simulation times are compared to existing models. For this study, the grid is taken as an infinite bus, the wind input consists only of wind speed and the control block is not modelled. The implemented models are compiled as a library to be used for the parameter estimation investigation.

Parameter estimation is performed on the individual models, as well as combined model topologies, to determine what parameters can readily be estimated.

1.3 Project objectives

As previously stated, this project was initiated to investigate whether the values of the system parameters can be obtained by performing parameter estimation on the chosen model of the wind turbine system. This section provides the objectives of the project and the processes followed to achieve these objectives.

This project is comprised of two main objectives. The first is to develop a toolbox for a wind turbine system to be used for the parameter estimation process. The second objective is to perform an introductory study to determine which parameters of the wind turbine system can be readily estimated.

The objective of developing a wind turbine system toolbox to be used for the parameter estimation process is further broken down into the following sub-objectives:

- *Literature review*: A literature review is required to determine the topologies used for the current wind turbine systems. This information is used to decide how to model the components of the wind turbine system.
- *Modelling*: The components of the wind turbine system are modelled by deriving mathematical models for each of the components.
- *Models implementation*: The derived mathematical models of the system components are implemented as C-code S-functions.
- *Models validation*: The accuracy of the implemented models is verified by comparing their simulated results to the results obtained with existing models.
- *Comparing efficiency*: The simulation times of the derived models are compared to that of existing models.
- *Creating a toolbox*: The implemented models are compiled as a Simulink toolbox.

The objective of performing an introductory study to determine which parameters of the wind turbine system can be readily estimated is further broken down into the following sub-objectives:

- *Literature review*: A literature review is required to review the parameter estimation process.
- *Estimation of individual component*: Parameter estimation is performed on the models of the individual components of the wind turbine system to determine which of their parameters can readily be estimated.
- *Estimation of combined model topologies*: Parameter estimation is performed on combined model topologies including the complete wind turbine system constructed by connecting the individual models together to determine which parameters can readily be estimated.

1.4 Thesis structure

This thesis is structured into six chapters and a number of appendices. The following details apply:

- *Chapter 1:* Chapter 1 presents the project overview. The project motivation and description section summarises the background information and discusses the driving forces behind this study. The research objectives of the study are also presented in this chapter.
- *Chapter 2:* Chapter 2 presents a literature review on the main components of this study. Different wind turbine system technologies are discussed, an overview of system identification and parameter estimation are presented and numerical analysis software with the focus on MATLAB is discussed.
- *Chapter 3:* Chapter 3 summarises the modelling of the different components of the wind turbine system and presents the implementations of these models as C-code S-functions in Simulink, as well as comprising a Simulink toolbox of the implemented models.
- *Chapter 4:* Chapter 4 presents the results of the validation and performance evaluation of the implemented models. The validation and performance evaluation is performed by comparing results obtained for the implemented models to those obtained from existing models.

- *Chapter 5:* Chapter 5 presents the results of the parameter estimation case studies. These results are obtained by comparing the values of the system parameters obtained from the parameter estimation process to the parameter values used for generating the input-output data used for the estimation process.
- *Chapter 6:* Chapter 6 summarises the results of the study, presents conclusions and gives recommendations for further work.

2 LITERATURE STUDY

2.1 Overview

The literature study consists of three sections. The first section reviews wind turbine technology. The second section starts off with an overview of the System Identification and Parameter Estimation that leads into a review of optimisation algorithms used for parameter estimation processes. The last section reviews available software for optimisation and implementation of mathematical models.

2.2 Wind turbine systems technology

2.2.1 Introduction

This section discusses the classifications of wind turbine systems, followed by an overview of different wind turbine system topologies. The advantages, disadvantages and prominence under the current wind turbine manufacturers of the different topologies are discussed.

2.2.2 Classification of wind turbine systems

Wind turbine systems can be divided into two main groups based on the shaft orientation. These groups can be further subdivided by looking at rotor blade configuration or the number of blades and the way the turbine system is connected to the grid.

The two main groups of turbines consist of those with horizontal shaft configuration and those with vertical shaft configuration. As the names state, in the case of the horizontal shaft configuration the shaft is in a horizontal position with the blade\blades connected to the one end of the shaft, as shown in Figure 2-1. The vertical shaft wind turbine has a much longer shaft in a vertical position with the blades connected to the shaft at more than one point, as shown in Figure 2-2.

The vertical shaft configuration turbines are further subcategorised into groups by looking at the positioning and form of the blades. The most well known vertical shaft wind turbine is the Darrieus phi configuration, also known as the eggbeater configuration. Other well-known groups are the Musgrove, Diamond, Savonius, Giromill and Phi types [14, 15].

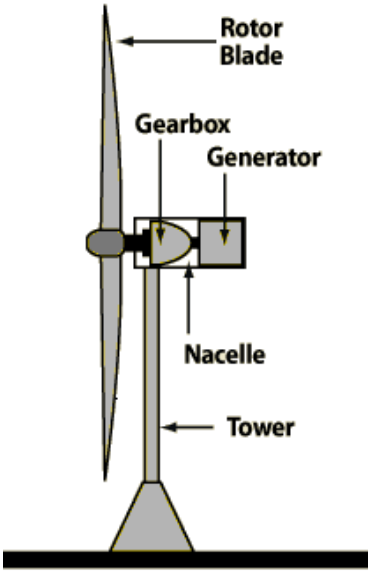


Figure 2-1: Horizontal shaft configuration (adopted from [16]**).**

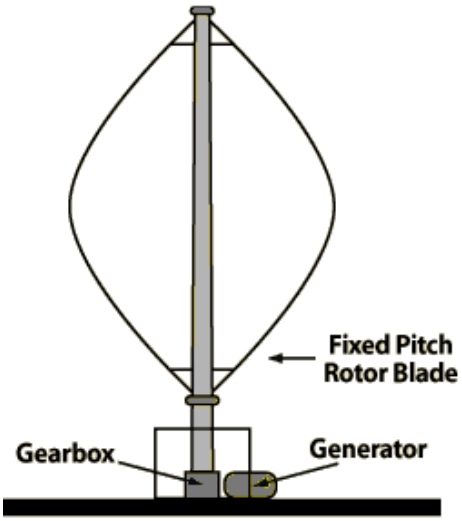


Figure 2-2: Vertical shaft configuration (adopted from [16]**).**

The biggest advantage of the vertical shaft wind turbine is the fact that it is omnidirectional, thus energy could be generated by wind blowing from any side without any adjustments needing to be made to the wind turbine. Such a configuration eliminates the need for a yaw motor or yaw gears. Another advantage is that the generator and gearbox can be housed at ground level, leading to a simple and cheaper design, and most of the maintenance of the wind turbine can be done at ground level.

The major disadvantage of the vertical shaft wind turbine is the fact that these turbines are mostly not self-starting; additional mechanisms are required to start the wind turbine. Vertical shaft wind turbines are also known for lower efficiency as a result of the aerodynamically dead zones the blades need to pass through to complete their rotation. Guy wires are also required to provide stability to the structure [14, 15, 17]

Work on the Darrieus technology has practically ceased. Although some research is still being done on the H-type configuration with self-starting capability, there are currently no megawatt capacity commercial vertical shaft wind turbines available [14]. Further discussion of the wind turbine systems are, therefore, focused on the horizontal shaft configuration. For further reading on vertical shaft wind turbine designs and their history, [14], [15], [17] or [18] can be viewed.

Most modern wind turbine systems make use of the horizontal shaft configuration. The horizontal shaft turbine configuration can be further subcategorised by looking at the number

of turbine blades, which include single-blade, two-blade, three-blade and multi-blade configurations.

The advantages of having only a single blade are minimum drag losses and high ideal operation speed. High operation speed leads to a low-ratio gearbox which is cheaper. The disadvantages that stem from high operating speed are more wear and tear on the system and high noise emissions. Another disadvantage is that the blade needs to be balanced with a counter-weight. This weight does not contribute to the energy extracted but adds to drag losses. The single-blade configuration is unpopular because of problems with the balance, visual acceptability and high noise emission due to high aerodynamic loading and high speed [14, 15, 19]. Some manufacturers claim they can produce three simple blades for the same cost as one high-performance blade required for a single-blade rotor. There is currently no major manufacturer that manufactures single-blade turbines of megawatt capacities [14].

As with the single-blade turbine, the two-blade turbine's ideal operating speed is high and, therefore, still quite noisy [14]. The biggest disadvantage that turbines with an even number of blades have is that the uppermost blade gets maximum power from the wind at the exact time that the lowermost blade gets minimum power from the wind due to the influence of the tower. This results in stability problems in a machine or gearbox with a stiff structure. Consequently, turbines with an even number of blades are not used that often [17]. The only major turbine manufacturer that currently has two-blade turbine models available is Australian based WindPacific, with models rated at 2.5 MW, 2.75 MW and 3 MW [20].

Since traditional manufacturers of two-blade turbines have switched to three-blade configurations, the three-blade upwind configuration is currently the most internationally used commercial wind turbine [15, 17]. The three-blade turbine is also known as the classical Danish concept. This turbine provides greater dynamic stability than single- or two-blade turbines as its aerodynamic loading is relatively uniform. Three-blade turbines have an optically smoother operation, hence, visually integrating better into the landscape. All major manufacturers have three-blade turbine models available ranging from hundreds of kilowatt to as big as 6.15 MW [19] as presented in APPENDIX A.

The multi-blade turbine category groups all turbines with four or more blades together. This group is also known as the high solidity rotors. The higher the solidity of the turbine, the easier the turbine self-starts because the initial rotor area is greater. The disadvantage of the multi-blade turbines comes from the fact that the higher number of blades results in higher

aerodynamic losses, and the lower ideal operational speed leads to larger and more expensive gearboxes [15].

Looking at horizontal shaft wind turbines in general, their advantages compared to vertical shaft wind turbines are lower cut-in speeds, easy furling and relatively high power coefficients. Their biggest disadvantage comes from the fact that the generator and gearbox are situated on top of the tower leading to a complex and more expensive design. Another disadvantage of horizontal shaft wind turbines is that they are directional.

Horizontal shaft wind turbines can be subdivided into two more groups, namely downwind turbines and upwind turbines [15]. Downwind turbines require no yaw mechanisms but have the big disadvantage of large tower shadow effects. Upwind turbines require yaw mechanisms to direct the blades into the wind, thereby positioning the blades in front of the tower relative to the wind direction. Although the rotor blades are upwind from the tower, the tower still has an aerodynamic effect on the blades, but to a lot lesser effect than in the case of downwind turbines [15].

After years of research by all the major turbine manufacturers in the megawatt capacity range, it seems that most have opted for the horizontal shaft configuration with three blades and situated upwind.

2.2.3 Fixed-speed generator topology

The fixed-speed generator topology makes use of an induction generator connected directly to the grid. The generator shaft is coupled to the rotor blades via a mechanical gearbox, as shown in Figure 2-3. The gearbox has a gear ratio that reduces the generator speed and increases the turbine torque to improve the rotor's power coefficient [19, 21].

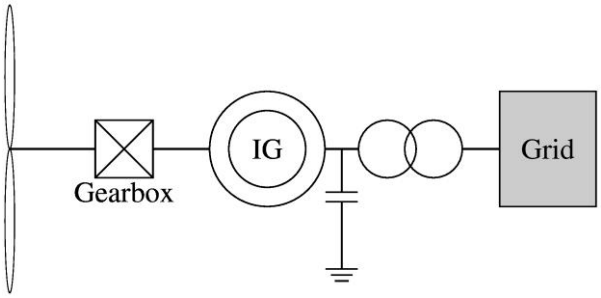


Figure 2-3: Fixed-speed generator topology.

The biggest advantage of this topology is that it makes use of an induction machine that is an asynchronous machine and, therefore, does not need to be synchronised with the grid. The system is also inexpensive, robust and requires minimum maintenance.

A disadvantage of the asynchronous machine is that it only allows for a small variation in the speed provided by the slip. The slip is limited to 1-2%, providing about 10% change in speed [19, 21, 22]. This small variation causes the system to only draw optimal power from the wind at a very narrow band of wind speeds, as shown in Figure 2-4. In this case the system can only draw optimal power for wind speed in the region of 7-8 m/s [19]. As the slip increases, the losses increase and efficiency drops [19]. The induction generator also needs excitation power from the grid, which is undesirable especially in weaker grids [15, 22]. To reduce this problem, capacitors are added to the circuit, as shown in Figure 2-3. Soft start equipment could also be added to reduce cut-in current [15, 22].

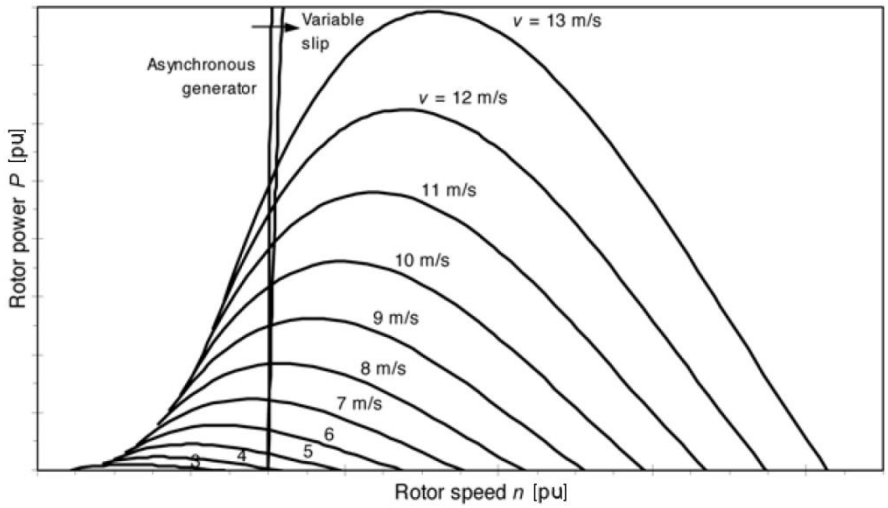


Figure 2-4: Operating points for a wind turbine with induction generator connected directly to the grid [19]**.**

The fixed-speed generator topology was used by the first commercial wind turbines for grid connection. These were commissioned in the late 1970's to early 1980's and made use of the stall control [22]. There are still some turbine systems in operation making use of this topology, but very few of the major manufacturers still manufacture it. The few that still do, use it for the smaller wind turbines. For example, the German manufacturer, Fuhrländen, has a 1.25 MW unit, the American manufacturer, Mitsubishi Power Systems, has a 1 MW unit and the Danish manufacturer, Vestas, has a 1.65 MW unit, as presented in APPENDIX A.

2.2.4 Two-speed induction generator topology

The two-speed generator topology has the same arrangement as the fixed-speed generator topology shown in Figure 2-3. The only difference is that the system is designed to operate optimally for two wind speeds, as shown in Figure 2-5. This can be accomplished by using two induction generators with different synchronous speeds, using a belt to shift between them or by having two sets of stator windings on the generator [19, 21, 22]. This topology is well suited for a wind site with two dominant average wind speeds. For instance, the systems with power to rotor speeds, as shown in Figure 2-5, would be ideal for a wind site with a lower average wind speed of 4 m/s and higher average wind speed of 10 m/s.

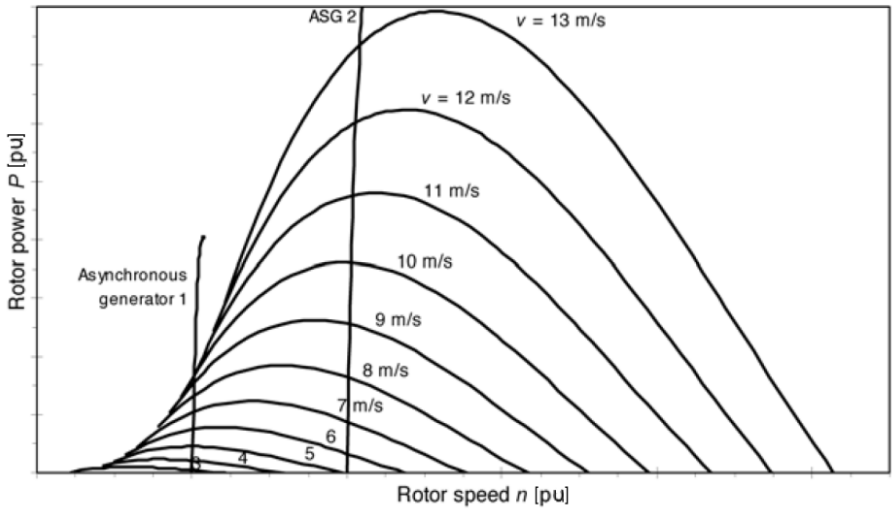


Figure 2-5: Operating points for a wind turbine with two-speed induction generator topology [19]**.**

The efficiency of this topology is better than that of the fixed speed induction generator topology, and it narrowly matches the efficiency of the full variable speed topologies that will be discussed in a following section [22]. Also, the losses in the rotor and the noise level of the system are reduced [21]. The problem is that wind is always turbulent, therefore, requiring regular system changes that cause great strain on all components [22]. Additionally, as with the fixed-speed generator topology, capacitance needs to be added to provide the excitation power for the inductance generator.

The only major manufacturer that still manufactures this topology is the Indian manufacturer Suzlon, with two 1.25 MW models [23].

2.2.5 Variable rotor resistance generator topology

The variable rotor resistance generator topology also makes use of the same arrangement as the fixed-speed generator topology, but instead of using a rotor with short-circuited winding endings (squirrel-cage), the winding endings are connected to variable resistors, as shown in Figure 2-6. These resistors can either be connected via slip rings to external resistors or to internal resistors rotating with the rotor [19]. This allows the generator to vary its speed, which is known as variable slip method.

An induction generator can only cushion low power fluctuations with slip, whereas a variable slip system can also cushion higher power fluctuations. Therefore, the variable slip system has the ability to absorb the power produced by sudden gusts without affecting the output frequency or power; this excess power is turned into heat [19, 22]. From this, it is clear that the advantage of variable slip topology is that it can vary the speed to a certain extent, a slip of about 10%. The disadvantage is that the high slip leads to high losses in the rotor circuit, thereby reducing the efficiency of the topology. As is the case with the fixed-speed and two-speed topology, the variable slip topology has no reactive power control [19].

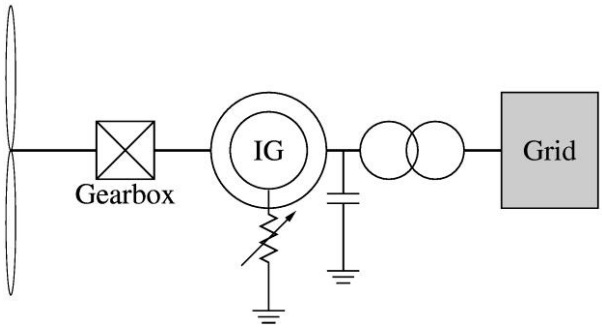


Figure 2-6: Variable rotor resistance generator system.

The only major manufacturer that still manufactures this topology is Suzlon, which produces a 1.25 MW and a 2.1 MW model [23].

2.2.6 Generator with fully-rated converter topology

The generator with fully-rated converter topology makes use of an induction generator, a synchronous generator or a permanent magnet generator connected via a gearbox to the turbine blades and through a fully-rated power converter to the grid, as the diagram in Figure 2-7 shows.

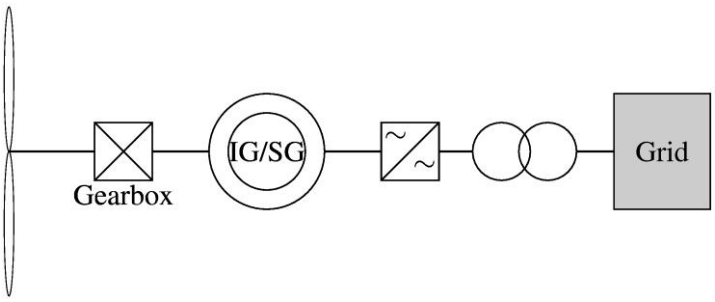


Figure 2-7: Generator with fully-rated converter topology.

The major advantage of the fully-rated converter topology is its variable speed capability. Therefore, the system can obtain maximum power from different wind speeds by varying the speed so that the system sustains an optimal power coefficient value [19]. Another advantage is that the power-electronics can be operated remotely, making it ideal for offshore wind farm applications. Furthermore, this topology has the ability to provide reactive power control that makes it suitable for weaker grids [21].

The major disadvantage of the fully-rated converter topology is the reliability of the electronic converter; this is mainly a result of the high power levels that need to be transmitted by the converter, since all power generated needs to be transmitted through the converter to the grid [24]. Additionally, although recent years have shown a steady decline in power electronic prices, the fully-rated converter is still expensive, the power electronic converter produces high frequency harmonics that need to be filtered to meet grid quality specifications and there are power losses in the converter that reduce the efficiency of the system [21].

This topology is becoming more popular as the cost of power electronics decreases. Major manufacturers like Siemens and Vestas have various models available rating from low megawatt to about 4 MW in capacity [25, 26].

Some manufacturers like WinWind and WindPacific have started to produce systems based on this topology. Instead of using standard 4 or 6 pole generators, these manufacturers use higher pole counts, thereby, reducing the need for multi-stage gearboxes. A one- or two-stage gearbox is smaller, and consequently weighs less, and has lower noise emissions [20, 27].

2.2.7 Generator with direct drive and fully-rated converter topology

This topology eliminates the need for a gearbox by making use of a generator that has a low synchronous speed, thus a high pole count generator. The high pole count generator is used in combination with a fully-rated converter to connect to the grid, as shown in Figure 2-8.

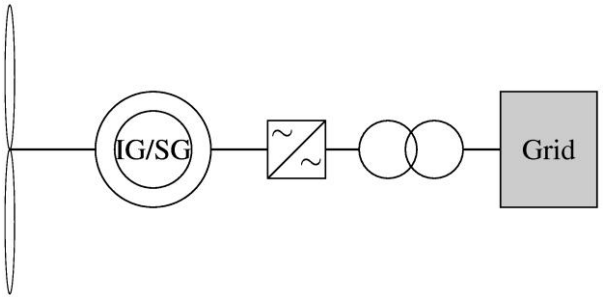


Figure 2-8: Synchronous or induction generator with direct drive and fully-rated converter topology.

The current generator of choice is the permanent magnet (PM) for its very high efficiency. A major disadvantage of PM generators is the need for significantly stringent tolerance requirements due to the inability to control the field strength leading to high cost [28].

This topology has the same advantages as the Generator with fully-rated converter topology with the added advantage of the elimination of the gearbox. The typical lifetime of a gearbox is three to four times shorter than that of the typical design lifetime of a wind turbine. By eliminating the gearbox, the maintenance cost and replacement cost are reduced [28]. Other advantages include reduction in noise and vibration levels [21] and lower power losses [19].

As is the case with the generator with fully-rated converter topology, a fully-rated converter is used, and the power electronics are expensive and prone to failure. Another disadvantage is the fact that multi-pole generators are big and heavy.

The direct drive topology has an approximate 13-15% market share. The German company, Enercon, is currently dominating the direct wind turbine market. Other companies that also have direct drive models in the market are Siemens and GE Energy. These machines rage in capacity from as small as 750 kW to 7.5 MW [2, 25, 28, 29].

2.2.8 Double-fed induction generator topology

The double-fed induction generator topology consists of a double-fed induction generator (DFIG)--also known as a wound-rotor induction generator--that is connected to the turbine blades via a gearbox, as shown in Figure 2-9. The stator of the wound-rotor generator is

connected directly to the grid, whereas the rotor is connected via a power electric converter to the grid. This arrangement allows for power in the rotor to be at a different frequency than that of the grid frequency, thereby allowing for speed control by adjusting this frequency.

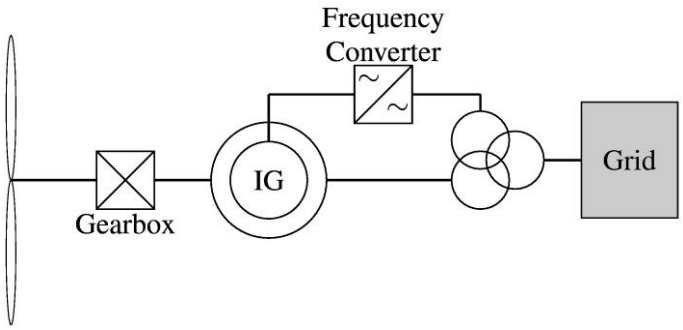


Figure 2-9: Double-fed induction generator topology.

There are several advantages to the DFIG topology. The variable speed allows for optimal power to be extracted for a wide range of wind speeds. The power converter is only 20%-30% of the rated power of the generator making it less expensive than a full-rated converter. This system also allows for reactive power to be controlled [19]. Unlike the variable rotor resistance generator topology where the energy generated in the rotor is dissipated in the rotor resistance, this energy is transmitted to the grid via the converter, which increases the efficiency of the topology.

The disadvantages to the DFIG include a higher cost resulting from the cost of power electronics used in the converter [21], harmonics generated by the power converter need to be filtered to comply with grid connection specifications [19] and a decrease in reliability due to the slip rings used to access the wound rotor.

Most of the major wind turbine manufacturers are incorporating the DFIG topology. Those manufacturers include Nordex, RePower and Envision. Eviag's entire range of wind turbine systems makes use of this topology [30-33]. Manufacturers like Vestas, GE Energy and Fuhrländer also have models available using this topology [26, 29, 34].

2.2.9 Directly coupled synchronous generator with variable gearbox topology

The directly coupled synchronous generator with variable gearbox topology consists of a synchronous generator directly connected to the grid. However, instead of a fixed speed gearbox that connects the rotor blade to the generator shaft, a variable gearbox is used, as shown in Figure 2-10.

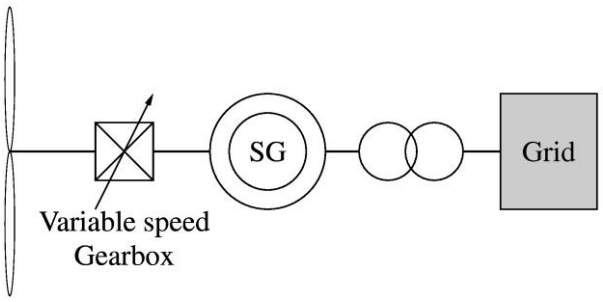


Figure 2-10: Directly coupled synchronous generator with variable gearbox topology.

In the past, this topology was considered but proved to add more problems than benefits. The problems that lead to the abandoning of this topology included high mechanical losses, high cost and short maintenance cycles [21]. In recent years the gearbox technologies have improved, and with the development of the Continuously Variable Transmission (CVT) constructed from a combination of planetary differential transmitters with variators, new life has been blown into this topology [24, 25].

The biggest advantage of this topology is that there is no need for power electronics, which is one of the main sources of failure of modern wind turbine systems [24]. A study by M.J. Verdonschot [24] has shown that, compared to conventional wind turbines that use the torque of the generator to control the rotor speeds by making use of power electronic converters, this topology could capture more energy from the wind during periods of low wind speeds and an equivalent amount of energy at higher wind speeds. Another advantage that results from not using power electronics is that there are no harmonics that need to be filtered before power is provided to the grid.

Although a lot of research is currently being done on the directly coupled synchronous generator with variable gearbox topology, none of the major wind turbine manufacturers have any models available that make use of this topology.

2.2.10 Rotor blade control

Another point of consideration is the control of the blade of the wind turbine. In Chapter 3 it will be shown that the amount of power that can be extracted from the wind is a function of the power coefficient C_p of the rotor blade defined as [35]

$$C_p = \frac{\text{Extracted Power}}{\text{Total Power in wind}}. \tag{2.1}$$

Theoretically, the maximum power extraction occurs when the downstream wind speed is a third of the upstream wind speed corresponding to a C_p value of 0.59, referred to as the Betz limit, but practical power coefficient values range from 0.2 to 0.5 [18]. The power coefficient is a parameter of the rotor blade which is a function of the blade pitch β, angular velocity ω and wind speed v. Figure 2-11 shows the typical relationship between C_p and Tip Speed Ratio (TSR) for various blade pitch angles [36]. The TSR denoted as λ is defined as [37, 38],

$$\lambda = \frac{R\omega}{v}. \tag{2.2}$$

where R denotes the length of the blade.

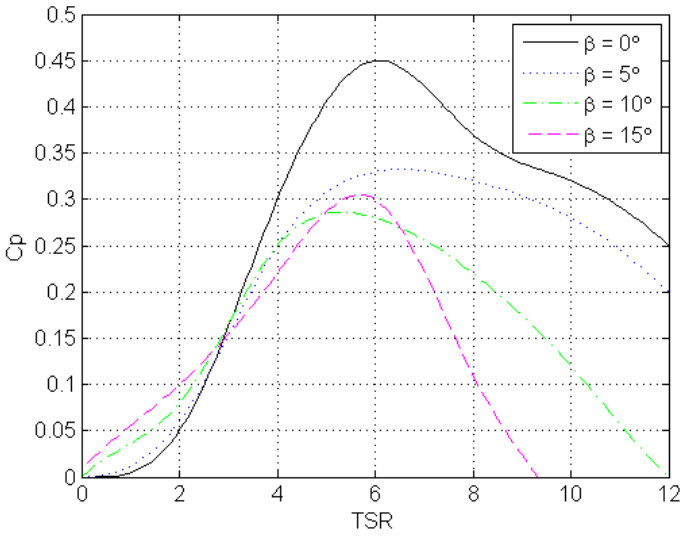


Figure 2-11: Power coefficient versus Tip Speed Ratio (TSR) for typical three-blade wind turbine [36]**.**

The power extracted from the wind can be controlled by changing the blade pitch angle β [21, 37, 39]. As discussed in [35] and [37], there are three major types of control, namely passive stall, active stall and pitch regulation.

For passive stall power control, the blade is aerodynamically designed to operate near the optimal TSR for low wind speeds while at higher wind speeds the design causes the blade to stall, limiting power output. The advantage of this method is that there are no moving parts. Two thirds of the installed wind turbines make use of stall control [40].

Active stall power controlled blades are designed with a mechanism for changing the pitch of the blades. For speeds below rated power, the pitch is mostly kept constant to reduce wear on

the pitch mechanism. At rated power, the pitch is controlled to produce constant power output. The advantage is that rated power is delivered at high speeds, unlike in the case of passive stall were power drops as the blades are pushed further into the stall region.

Pitch power control adjusts the blade pitch to optimise the power at all wind speeds. When maximum power is reached, the blades are adjusted to reduce the lift of the blades thereby reducing the power to keep it constant at maximum power.

2.2.11 Conclusion

After discussing the different classifications of wind turbines and the different topologies, commercial turbine systems available on the market were investigated. The conclusion is drawn that the current trend in the megawatt commercial turbine market is to make use of the horizontal shaft configuration with three blades with the blades positioned upwind from the tower. By looking at the models currently available from the different manufacturers, it seems that the DFIG topology is the most common. The market, however, appears to be moving in the direction of the generator with fully-rated converter topology and especially the direct drive generator with fully-rated converter.

2.3 **System identification and parameter estimation**

2.3.1 Overview

This section provides an overview of the system identification process and carries on to discuss the parameter estimation process. This is followed by an overview of cost functions. The section is concluded by a discussion of optimisation algorithms.

2.3.2 Overview of system identification and parameter estimation processes

Before discussing the process of system identification or parameter estimation, the concept model and the way models are classified first need to be considered. In his book *System Identification – Theory for the User* [41], Ljung provides a broad definition of the word model as: "*The assumed relationship among observed signals of a system*". Models can be divided into two main groups: Mental models, also called Intuitive models, and Mathematical models, which include graphical models [42]. Mental models are the type of models people use every day when driving a bicycle, e.g., pedalling faster increases the speed or pushing the

shopping trolley, e.g., pushing harder with the right hand while pulling back with the left makes the trolley turn left.

The models that are more of interest to engineers are the mathematical models [41, 42]. Mathematical models can further be subdivided into models obtained by the modelling of a system or models obtained by System Identification. Models obtained by modelling a system, also referred to in some texts as grey-box models [41], are constructed from basic laws of physics, like Newton's laws, circuit analysis or balance equations [42]. Since these models are constructed from laws of physics, the parameters of the model have physical meaning. Models obtained by the System Identification process are also referred to in some texts as black-box models, in which case the parameters of the model do not necessarily represent any physical meaning. Further classifications of mathematical models of dynamic systems include the following [42]:

- single input, single output- vs. multivariable models
- linear- vs. nonlinear models
- parametric- vs. nonparametric models
- time invariant- vs. time varying models
- time domain- vs. frequency domain models
- discrete time- vs. continuous time models
- lumped- vs. distributed parameter models
- deterministic- vs. stochastic models

The above mentioned linear or nonlinear refers to the relationship of input to past data and not to the dimensionality in the parameter of the model [42].

From this it is clear that there are numerous different model types, therefore, a general model is used when applying System Identification. A brief discussion of this model follows.

As mentioned earlier in this section, the aim of System Identification is to obtain a mathematical model of a dynamical system from measured input-output data. The model is regarded as a black-box with little or no knowledge about the inner workings of the black-box. The System Identification process can be visualised with the block diagram shown in Figure 2-12. The *Get Data*, *Choose Model Set* and *Choose Criterion for Fit* boxes all require prior knowledge; this knowledge can be knowledge obtained from previous experiments or a general understanding of the system to be modelled. The first step of System Identification is

illustrated by the *Get Data* block in Figure 2-12. This block represents the process of obtaining a data set consisting of input and output data of the system being identified. This data can be obtained from experimental measurements on the system, measurements on an operational system or a combination of both.

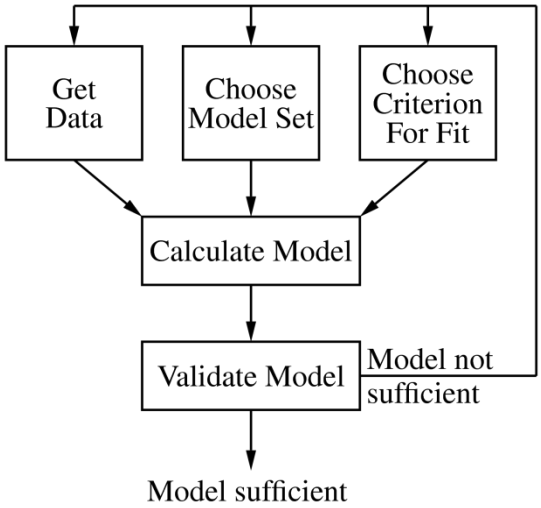


Figure 2-12: Block diagram of system identification process [41, 42]**.**

This is followed by the important step of finding a candidate model structure, illustrated by *Choose Model Set* block. As previously mentioned, numerous model types exist, therefore, a general model structure is used for the System Identification process. The general model structure for linear time-invariant single-input-single-output system can be described as [41]:

$$A(q)y(t)=\frac{B(q)}{F(q)}u(t)+\frac{C(q)}{D(q)}e(t), \tag{2.3}$$

where

$y(t)$ denotes the measured output of system to be modelled,

$u(t)$ denotes the measured input of system to be modelled,

$e(t)$ denotes the error

and

$A(q), B(q), C(q), D(q), F(q)$ denote polynomials.

The error can be caused by inaccuracies of the measurement equipment or other external disturbances. Depending on which of the polynomials are used, (2.3) can give rise to 32

different model sets. For example, using A, B and C gives the well known ARMAX model structure [41] with block diagram shown in Figure 2-13.

The general model structure defined by (2.3) can also be expressed as:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{2.4}$$

where $G(q)$ denotes the transfer function of the deterministic part of the system, and $H(q)$ denotes the transfer function of the stochastic part of the system [42].

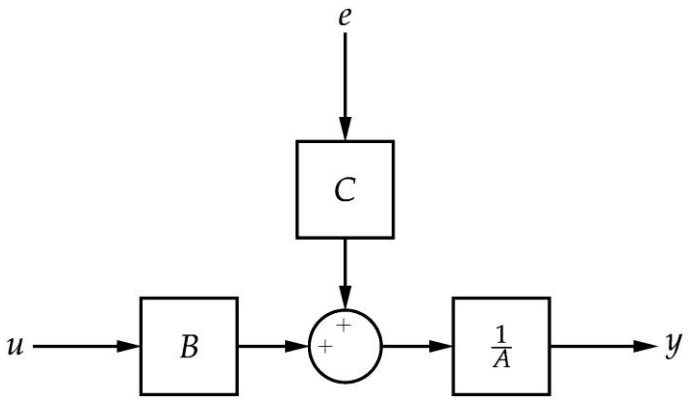


Figure 2-13: ARMAX model structure [42]**.**

This general model structure given in (2.4) can be expanded for Multi-Input-Multi-Output (MIMO) systems. If the MIMO system consists of *m* number of inputs and *n* number of outputs, $u(t)$ becomes an *m*-element column vector and $y(t)$ and $e(t)$ become an *n*-element column vector. $G(q)$ and $H(q)$ become an $n \times m$ element and an $n \times n$-element matrix respectively. These models work well in most cases. However in the case of complex higher order systems with several inputs, several outputs and a large number of measurements, these models can suffer from problems like converging to local minima instead of global minima, numerical instability and excessive computation time. These problems can be overcome by using the general state-space model [43] given as

$$\begin{aligned} x(n+1) &= A_{ss}x(n) + B_{ss}u(n) + K_{ss}e(n) \\ y(n) &= C_{ss}x(n) + D_{ss}u(n) + e(n) \end{aligned}, \tag{2.5}$$

where

$y(n)$ denotes the system output vector,

$u(n)$ denotes the system input vector,

$x(n)$ denotes the state vector,

$e(n)$ denotes the error

and

$A_{ss}(n)$, $B_{ss}(n)$, $C_{ss}(n)$, $D_{ss}(n)$, $K_{ss}(n)$ denote system matrices.

From (2.3) and (2.5), it is clear that the unknown variables are the coefficients of the polynomials or the elements of the system matrices, and these need to be estimated. As mentioned earlier, these coefficients or elements do not necessarily represent any real system parameter; it is a mere numerical value of the mathematical model that leads to the model behaving like the real system.

Having the input-output data and model set, the next steps are to choosing a cost function for the fitting of the data and determining the coefficients of the model through an optimisation algorithm. These processes are shown by *Choose Criterion For Fit* and *Calculate Model* blocks. The cost function and optimisation algorithm are discussed in detail further on in this section.

The only remaining block in Figure 2-12 is the *Validate Model.* After estimation the model needs to be validated. In the case where the model proves to be sufficient, the System Identification process is done. If it is not sufficient, the process loops back to the start. An insufficient result can be due to the following:

- An insufficient model set that was chosen
- The input-output data could be insufficient in the sense that it does not excite the system sufficiently for estimation purposes
- The identification criteria for fitting is insufficient for the model type

All of these points need to be checked and necessary adjustments need to be made, thereby making this an iterative process that loops runs until the model is sufficient.

It follows that in the case of System Identification, the model is unknown. A general model structure needs to be chosen, and the parameters of this chosen structure then need to be estimated by an estimation algorithm. In the case of a parameter estimation process, the model for the system has a known model structure.

Since all the components of the wind turbine system can be modelled by models constructed from the laws of physics, System Identification is not necessary [44]. The remainder of this section focuses on the parameter estimation process.

From this information it is clear that the aim of System Identification is to obtain a mathematical model of a real system. In the case of Parameter Estimation, the aim is to obtain the values of the parameters of a known model. Although the aim of System Identification and Parameter Estimation is quite different, both techniques share a common problem, namely that of finding the coefficients or parameters of the model; as such, the behaviour of the model simulates that of the real system as closely as possible.

The parameter estimation process can be visualised with the block diagram shown in Figure 2-14. The *System* block represents the real system and the *Model* block represents the model of the system. *Input* denotes the measured input for the system, y denotes the output of the real system and $\bar{y}$ denotes the simulated output of the model. The *Cost Function* and *Estimation Algorithm* block and their symbols will be discussed in detail as the section continues.

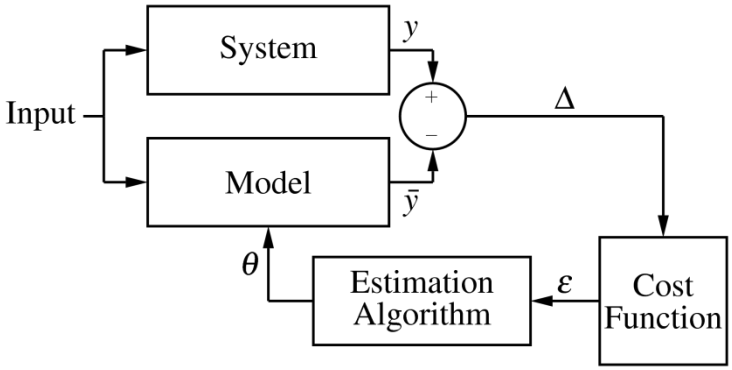


Figure 2-14: Block diagram of the parameter estimation process [44]**.**

The Cost Function and Estimation Algorithm blocks work in tandem. The Cost function provides the criteria whereby the model performance can be measured. This is used by the estimation algorithm to change the system parameters until the optimised parameters are found.

2.3.3 Overview of cost functions

To evaluate how well the model simulates the real system, a way of gauging the accuracy of the model is required. This can be done by looking at the difference between the output of the real system and the output of the model at each data point, known as the residual. The

smaller the residual at all of the data points, the more accurate is the model. By constructing a cost function, also known as a residual function, of the residual at the different data points the model can be optimised by minimising the cost function. The first method that comes to mind is the summation of all the residuals [45], giving

$$\varepsilon = \sum_{i=1}^{n} \left(y_i - \overline{y_i} \right) = \sum_{i=1}^{n} \Delta_i \tag{2.6}$$

where

y_i denotes the output of real system at the ith data point,

$\overline{y_i}$ denotes the output of model at the ith data point,

Δ_i denotes the residual at the ith data point,

n denotes the number of data points

and

ε denotes the residual error which is the answer to the residual function.

It should be clear that this criterion is inadequate since the residual can be positive or negative. For example, the residual error could amount to a zero when the sum of the residuals is calculated, but in actual fact it is the negative residuals cancelling the positive residuals. The obvious solution is to remove the effect of the sign by taking the absolute value of the residual before adding it together. This yields

$$\varepsilon = \sum_{i=1}^{n} \left| y_i - \overline{y_i} \right| = \sum_{i=1}^{n} \left| \Delta_i \right| . \tag{2.7}$$

This criterion works well to get a more accurate result for gauging the accuracy of the model. However, as will be seen in the section on estimation algorithms, most estimation algorithms make use of the derivative of the cost function to determine the parameter that minimises the residual error. Therefore, this criterion is not well suited since an absolute value cannot be differentiated at all points [46].

The criterion known as the sum of squares is the most commonly used since it overcomes the problem of the sign of the residuals by taking the square of the residuals and is still differentiable. The residual function for the sum of squares is

$$\varepsilon = \sum_{i=1}^{n} \left(y_i - \overline{y_i} \right)^2 = \sum_{i=1}^{n} \left(\Delta_i \right)^2 . \tag{2.8}$$

A disadvantage of the sum of squares criterion is the influence of outlier data points caused by noise or disturbances. These have a big impact on the residual error since the residual at these outliers is big and squaring it makes it even bigger. A way to overcome this problem is by adding a weight factor, thereby the contribution of residual at the outliers can be forced to weigh less. Equation (2.9) shows the weighted sum of the squares residual function where w_i denotes the weight factor.

$$\varepsilon = \sum_{i=1}^{n} w_i \left(y_i - \overline{y_i} \right)^2 = \sum_{i=1}^{n} w_i \left(\Delta_i \right)^2 \tag{2.9}$$

The above examples of residual functions are not the only possibilities for calculating the residual error. Any function, simple or complex, can be used. For example, the sum of sixth-power terms as shown in (2.10) could also be used.

$$\varepsilon = \sum_{i=1}^{n} \left(y_i - \overline{y_i} \right)^6 = \sum_{i=1}^{n} \left(\Delta_i \right)^6 \tag{2.10}$$

Although these other functions are used, in some cases the sum of squares criterion is the most commonly used as a result of it being solidly grounded in statistics [45]. Assuming the function of which the parameter needs to be estimated is $\phi(\mathbf{x}, t)$, the residual denoted by Δ_i at each observation can be defined as

$$\Delta_i = y_i - \phi(\mathbf{x}, t_i), \tag{2.11}$$

where y_i denotes a set of observations. Making the assumption that the residuals are independent and are identically distributed with a normal distribution function with a variance of σ^2, the likelihood of a set of observations $i = 1, 2, \ldots, m$ can be given as

$$p(y; \mathbf{x}, \sigma) = \prod_{i=1}^{m} g_\sigma (\Delta_i) = \prod_{i=1}^{m} g_\sigma \left(y_i - \phi(\mathbf{x}; t_i) \right) \tag{2.12}$$

and the distribution function as

$$g_\sigma (\Delta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\Delta^2}{2\sigma^2} \right). \tag{2.13}$$

Substituting (2.11) and (2.13) into (2.12) yields:

$$p(y;\mathbf{x},\sigma)=\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^m \exp\left(-\tfrac{1}{2}\sum_{i=1}^{m}\frac{\left[y_i-\phi(\mathbf{x};t_i)\right]^2}{\sigma^2}\right). \tag{2.14}$$

From this it is clear that for a fixed σ^2 the likelihood function is maximised when the sum of squares is minimised. The assumptions made are often good assumptions, for instance, when the model accurately reflects the actual system and the errors made in obtaining the observation data do not contain a systematic component [45]. Although the assumptions made to obtain (2.14) are common and sufficient to show that the sum of squares as cost function are solidly grounded in statistics, this is not the only situation for which the minimisation of the sum of squares makes statistical sense. Further reading on this topic can be found in [45].

With a criterion in place to calculate the residual error, the *Cost Function* block in the block diagram is covered leaving only the *Estimation Algorithm* block to be discussed.

2.3.4 Overview of optimisation algorithms

With the cost function defined, all that remains is to find the parameter values that minimise this cost function, thereby, obtaining the parameters that provide a model that sufficiently represents the behaviour of the real system. The minimisation process points towards the field of optimisation. Optimisation, also known as Programming, is the process of minimising or maximising a function by manipulation of its control variable (parameters) within or without certain constraints [47]. It should be noted that in optimisation literature, for example [48, 49], the function to be optimised is referred to in many different ways, such as objective function, cost function, energy function or criterion function. In system identification and parameter estimation literature it is mostly referred to as the objective or cost function. These will, therefore, be used in the remainder of this text.

For the sake of clarity, an overview of optimisation algorithms is provided followed by an overview of the basic concept of iterative optimisation before looking at algorithms suitable for the solving of sum of squares objective functions, also referred to as least-square objective functions.

The main properties of the optimisation algorithm are robustness, efficiency and accuracy. Robustness refers to the algorithm being able to solve a wide range of problems in the class it

was developed for, given reasonable initial variables. Efficient use of the processor and memory of the computer, whilst obtaining the solution in the quickest possible time, is a property that has become all the more important over the last few years. The last property is the accuracy property. The algorithm should be able to obtain an accurate solution without being overly sensitive to errors that might be caused by data errors or errors as a result of computer implementation. Ideally, it would be preferred that all algorithms perform well with regards to all of these properties, but this is not always possible. In most cases, one or more properties need to be weighed off against another. For example a situation can occur where the algorithm is very robust, but as a result the efficiency is low. By decreasing the robustness, the efficiency can be increased [45].

Figure 2-15 shows a tree diagram of the different categories of optimisation problems. These categories are generated by analysing the objective function focusing on the behaviour of the objective function with reference to its parameters. Points to consider when choosing an optimisation algorithm include: whether or not there are constraints on the optimisation problem, whether local or global minima or maxima is required, whether the problem is continuous or discrete and whether the problem is stochastic or deterministic. The wrong choice of algorithm could lead to the wrong solution and/or the optimisation process being unnecessarily time and/or computationally intensive.

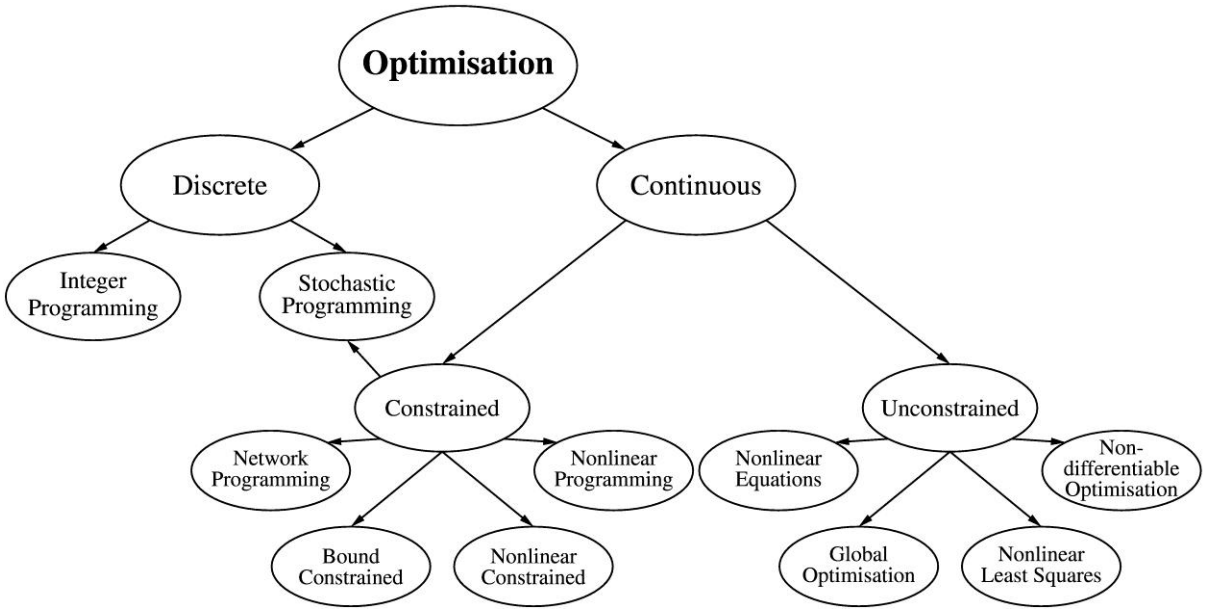


Figure 2-15: Optimisation tree diagram.

From the algorithm properties and the vast majority of categories of the optimisation algorithm, it should be clear that there exists no universal algorithm for optimising all

objective functions. Discussing all of these different algorithms falls outside the scope of this study, but further reading on this topic can be found in [45], [47], [48], [49], [50], [51], [52] and [53].

Keeping all of this in mind, as well as the discussion of the cost function, the rest of this section gives an overview of some of the basic concepts of optimisation on which most of the algorithms are based.

Although no universal algorithm exists for solving all optimisation problems, most algorithms rely on an iterative process that starts with an initial guess for the parameters that need to be estimated, and, hereafter, the accuracy of the parameters are improved with each step of the iterative process. The iterative process can be represented mathematically by [47, 53]

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{\Delta x}_i \tag{2.15}$$

where

$\mathbf{x}_i$ denotes the current n - element parameter vector ,

$\mathbf{x}_{i+1}$ denotes the new n - element parameter vector

and

$\mathbf{\Delta x}_i$ denotes the n - element step-vector that applies the chance to the current parameter values.

The step-vector can be seen as a scalar α, known as the step size, times a direction vector $\mathbf{d}$ yielding

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha \mathbf{d}_i \,. \tag{2.16}$$

For an objective function $f(\mathbf{x})$, the result of $f(\mathbf{x}_i + \alpha \mathbf{d}_i)$ can be compared to the result of $f(\mathbf{x}_i)$. If $f(\mathbf{x}_i + \alpha \mathbf{d}_i)$ is smaller, the step is successful and $\mathbf{x}_i + \alpha \mathbf{d}_i$ gets assigned as the new parameter vector. If $f(\mathbf{x}_i + \alpha \mathbf{d}_i)$ is bigger, either the direction, step size or both need to be changed.

The procedure of selecting $\mathbf{d}$ varies, resulting in all the different available algorithms. For example, a well known group of algorithms is the Descent Methods. The following figures show the search path for two methods belonging to this group. Figure 2-16 shows that of the

Gradient Decent method, also known as the Steepest Descent method, in which case the direction vector is taken as the negative of the gradient at the current point,

$$d_i = -\nabla f\left(\mathbf{x}_i\right). \tag{2.17}$$

Figure 2-17 shows that of the Coordinate Descent method, in which case the direction vector is set to the direction of a different parameter for each step. Although this method is not used much on its own, it is often used as a starting procedure for some of the more complex methods [47]. In both Figure 2-16 and Figure 2-17 the contour lines represents the value of the cost function, with highest value in the upper left and lower right corners and lowest value in the middle of the figure. To illustrate the large amount of possible optimisation algorithms, it should be mentioned here that there exists at least three methods for deciding the sequence of the parameters for the Coordinate Descent method; Cyclic coordinate descent where the sequence is $x_1,...,x_n$, $x_1,...,x_n$, etc, Aitken double sweep where the sequence is $x_1,...,x_n$, $x_n,...,x_1$ or Gauss–Southwell where the parameter that corresponds to the largest component of the gradient vector is used [53].

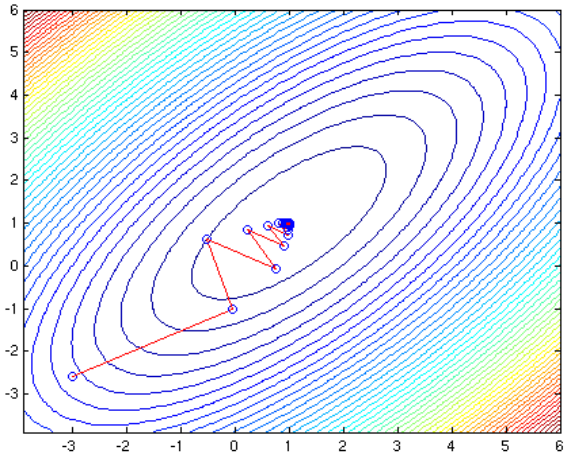


Figure 2-16: Typical search path for gradient descent for two variables [54].

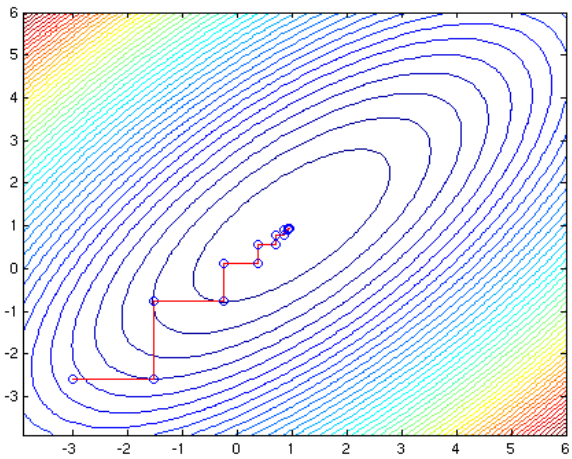


Figure 2-17: Typical search path for coordinate descent for two variables [54].

With the direction vector fixed, the focus shifts to the step size α for which an optimal solution can be found by solving

$$\frac{d}{d\alpha} f\left(\mathbf{x}_i + \alpha \mathbf{d}_i\right) = 0. \tag{2.18}$$

In practice, this equation would not always have an exact solution. Another iterative search or approximation is needed, but in fact all that is required is to verify that $f(\mathbf{x}_i + \alpha \mathbf{d}_i)$ is

smaller than $f(\mathbf{x}_i)$ [53]. Therefore, assuming the direction vector is descent, $f(\mathbf{x}_i + \alpha \mathbf{d}_i) < f(\mathbf{x}_i)$ would be true for a small enough α.

With this background on the general iterative optimisation process, an overview is provided of the well known Newton algorithm leading to a discussion of three algorithms mostly used for solving least-square objective functions $f(\mathbf{x})$ that have the special form,

$$f(\mathbf{x}) = \tfrac{1}{2}\sum_{j=1}^{m} r_j^2(\mathbf{x}). \tag{2.19}$$

where the residual r_j is a smooth function from $\mathbb{R}^n$ to $\mathbb{R}^m$, $\mathbf{x} = [x_1, x_2, ..., x_n]$ and assuming $m \geq n$ with m being the number of data points and n the number of parameters. Before continuing to the algorithms, some mathematical expression first needs to be discussed. By expressing the residual as a vector [45]

$$r(\mathbf{x}) = \left(r_1(\mathbf{x}), r_2(\mathbf{x}), ..., r_m(\mathbf{x})\right)^T, \tag{2.20}$$

equation (2.19) can be rewritten as $f(\mathbf{x}) = \frac{1}{2}\left\|r(\mathbf{x})\right\|^2$, and the Gradient and Hessian can be expressed as

$$\nabla f(\mathbf{x}) = J(\mathbf{x})^T r(\mathbf{x}) \tag{2.21}$$

and

$$\nabla^2 f(\mathbf{x}) = J(\mathbf{x})^T J(\mathbf{x}) + \sum_{j=1}^{m} r_j(\mathbf{x}) \nabla^2 r_j(\mathbf{x}) \tag{2.22}$$

respectively, where the $J(\mathbf{x})$ is the $m \times n$ Jacobian matrix given by

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1(\mathbf{x})}{\partial x_1} & \frac{\partial r_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial r_2(\mathbf{x})}{\partial x_1} & \frac{\partial r_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_m(\mathbf{x})}{\partial x_1} & \frac{\partial r_m(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial r_m(\mathbf{x})}{\partial x_n} \end{bmatrix}. \tag{2.23}$$

The remainder of this section discusses four optimisation methods. The Newton method is reviewed first. The section then moves on to discuss the methods commonly used for solving non-linear least-square objective functions, namely the Gauss-Newton method, the Trust-region method and the Levenberg-Marquardt method.

2.3.4.1 Newton method

The Newton optimisation method is a linear search method which, as mentioned, is one of the most well known optimisation algorithms. Given an objective function f of a one dimensional optimisation problem, the aim is to find the parameter value x_{opt} where the objective function is a minimum, thus $f'(x_{opt}) = 0$. This point is referred to as the stationary point. By taking the first three terms of the Taylor expansion of the objective function, a quadratic model of the function is obtained. This yields

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + 0.5 f''(x)\Delta x^2, \tag{2.24}$$

where $\Delta x = x_{i+1} - x_i$ with x_{i+1} denoting the new point to be estimated from the current point, x_i. The extremum of $f(x+\Delta x)$ is obtained by solving the linear equation

$$f'(x) + f''(x)\Delta x = 0. \tag{2.25}$$

From (2.25) the equation for the new point can be derived as

$$x_{i+1} = x_n - \frac{f'(x_i)}{f''(x_i)}, \tag{2.26}$$

which converges toward a root of f'.

The multi-dimensional iteration scheme can be obtained from (2.25) by replacing the derivative $f'(x_i)$ with the gradient matrix $\nabla f(\mathbf{x})$ and the second derivative $f''(x_i)$ with the Hessian matrix $\nabla^2 f(\mathbf{x})$. This yields

$$\nabla f(\mathbf{x}_i) + \nabla^2 f(\mathbf{x}_i)\Delta x = 0, \tag{2.27}$$

Solving (2.27) for $\Delta\mathbf{x}$, substituting $\Delta\mathbf{x} = \mathbf{x}_{i+1} - \mathbf{x}_i$, and rearranging the result yields

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \left[\nabla^2 f(\mathbf{x}_i)\right]^{-1} \nabla f(\mathbf{x}_i). \tag{2.28}$$

The Newton algorithm requires the second derivative and, therefore, falls into the group of algorithms known as Second-Derivate Methods. Although the Newton method generally solves problems faster than First-Derivate methods, such as the Gradient Method discussed earlier, the Newton method requires the function to be twice-differentiable. In the multi-dimensional case, the Hessian matrix also needs to be invertible and this operation is numerically intense. For the Newton method to work well the function needs to be well approximated by its second order Taylor expansion and the initial guess x_o needs to be close enough to x_{opt}.

2.3.4.2 Gauss-Newton method

The Gauss-Newton method is based on the Newton method and modified by making the assumption that the residual and Hessian of the residuals are small in the vicinity of the solution. This is the case for many least-square problems [45]. Using this assumption, the Hessian matrix can be approximated by

$$\nabla^2 f\left(\mathbf{x}\right) \approx J\left(r(\mathbf{x})\right)^T J\left(r(\mathbf{x})\right). \tag{2.29}$$

Substituting (2.21) and (2.29) into (2.28) yields

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \left[J\left(\mathbf{x}_i\right)^T J\left(\mathbf{x}_i\right) \right]^{-1} J\left(\mathbf{x}_i\right)^T r\left(\mathbf{x}_i\right). \tag{2.30}$$

By neglecting the second term of the Hessian, a significant saving in computational time is achieved. This is due to the fact that no additional derivative evaluations are required for obtaining the Hessian because the Jacobian is already evaluated for the calculation of the gradient matrix [45]. This method converges fast to a local minimum for systems that are mildly non-linear and requires only one iteration for linear systems. For systems that are badly non-linear or have big residuals, it might not converge to a local minimum. Enhanced Gauss-Newton methods like the Damped Gauss-Newton method also exist. More detail on these methods can be obtained from [45] and [55].

Both the Newton and Gauss-Newton methods are classified as Line search methods. The process these methods follow is finding the search direction, i.e., $\mathbf{d}_i$ referring to (2.16) followed by finding a suitable step length, i.e., α referring to (2.16). The following two methods also make use of the quadratic model of the objective function. These methods, however, start by defining a region around the current iteration for which the model is trusted

to be a good representative of the objective function. Thereby first fixing the step size and then chooses a direction for the step. These methods are, therefore, classified as trust-region methods [45].

2.3.4.3 Trust-region method

Similar to the Newton and Gauss-Newton methods, the Trust-region method is iterative and provides local minimisers. As mentioned, the Trust-region method also makes use of the quadratic model $q_i(\Delta\mathbf{x})$ given by (2.31). This is obtained from the first three terms of the Taylor expansion [56] as is the case for the Newton method.

$$q_i(\Delta\mathbf{x}) = f(\mathbf{x}_i) + \nabla f(\mathbf{x}_i)\Delta\mathbf{x} + 0.5\Delta\mathbf{x}^T\nabla^2 f(\mathbf{x}_i)\Delta\mathbf{x} \tag{2.31}$$

Knowing that the quadratic model is only accurate close to $\mathbf{x}_i$, a region around the point is defined where the accuracy of the quadratic model is trusted. This region is defined as the trust-region around the point $\mathbf{x}_i$ with a radius known as the trust-region radius. The method starts with an initial value. The next step is approximating the objective function using the quadratic model about the initial point. A step size and direction is then obtained by solving the optimisation of the sub-problem $q_i(\Delta\mathbf{x})$ within the boundary of the trust-region. Before the point obtained from this sub-optimisation is assigned as the new point, it is evaluated by analysing the ratio of actual versus predicted reduction given as the relationship [57]

$$\rho_i = \frac{\text{Actual reduction of } f}{\text{Predicted model reduction of } f} = \frac{f(x_i) - f(x_i + \Delta x_i)}{q_i(0) - q_i(\Delta x_i)}. \tag{2.32}$$

To prevent the algorithm from approximating an objective function unnecessarily at a point that is quite expensive, the value of ρ_i is also compared to another trust-region parameter known as minimum step ratio η. If ρ_i is smaller than this value, the step is rejected and recomputed. The value of the minimum step ratio is normally quite small to prevent the rejection of steps that are progressing towards the minimum. Although this will reduce the number of expensive approximations of the objective function, it might lead to an increase in function evaluations. Furthermore, if the value of the ratio is close to 1, the approximated function closely resembles the actual objective function. Therefore, the trust-region radius is increased for the next iteration; whereas, a negative or small ratio value would suggest that the trust-region is a bad approximation and the trust-region radius is decreased for the next

iteration. This process carries on till the minimum point is obtained or the maximum amount of iteration denoted by m is reached. The process can be illustrated by the flow diagram in Figure 2-18 where $\hat{\Delta}$ denotes the overall bound on the step length and $\Delta_i = \|\Delta \mathbf{x}_i\|$. Since this method converges to a local minimum, a different minimum might be obtained by choosing a different initial value.

It is clear that some of these steps involve a lot more than what is explained here; for instance, the algorithm for solving the optimisation of the sub-problem as well as choosing the ranges of ρ_i for which Δ_i are changed and to what values it is changed. This falls outside the scope of this study, but further reading on these techniques can be found in [45], [50], [51], [56] and [57].

2.3.4.4 Levenberg-Marquardt method

As mentioned, both the Newton and Gauss-Newton methods make use of the line search algorithm, but the Gauss-Newton method uses an approximation of the Hessian matrix, $\nabla^2 f(\mathbf{x}) \approx J(r(\mathbf{x}))^T J(r(\mathbf{x}))$. Although some texts consider the Levenberg-Marquardt method as the predecessor of the Trust-Region method [45], the Levenberg-Marquardt method can be best described as the Trust-Region method making use of the same assumption for the Hessian as the Gauss-Newton [45]. Therefore, the flow diagram for the Levenberg-Marquardt method is the same as that off the Trust-Region method shown in Figure 2-18 with the model of the trust region replaced by

$$q_i(\Delta \mathbf{x}) = f(\mathbf{x}_i) + \Delta \mathbf{x} J(r(\mathbf{x}))^T r(\mathbf{x}) + 0.5 \Delta \mathbf{x}^T J(r(\mathbf{x}))^T J(r(\mathbf{x})) \Delta \mathbf{x}, \tag{2.33}$$

where

$$f(\mathbf{x}) = \frac{1}{2} \|r(\mathbf{x})\|^2.$$

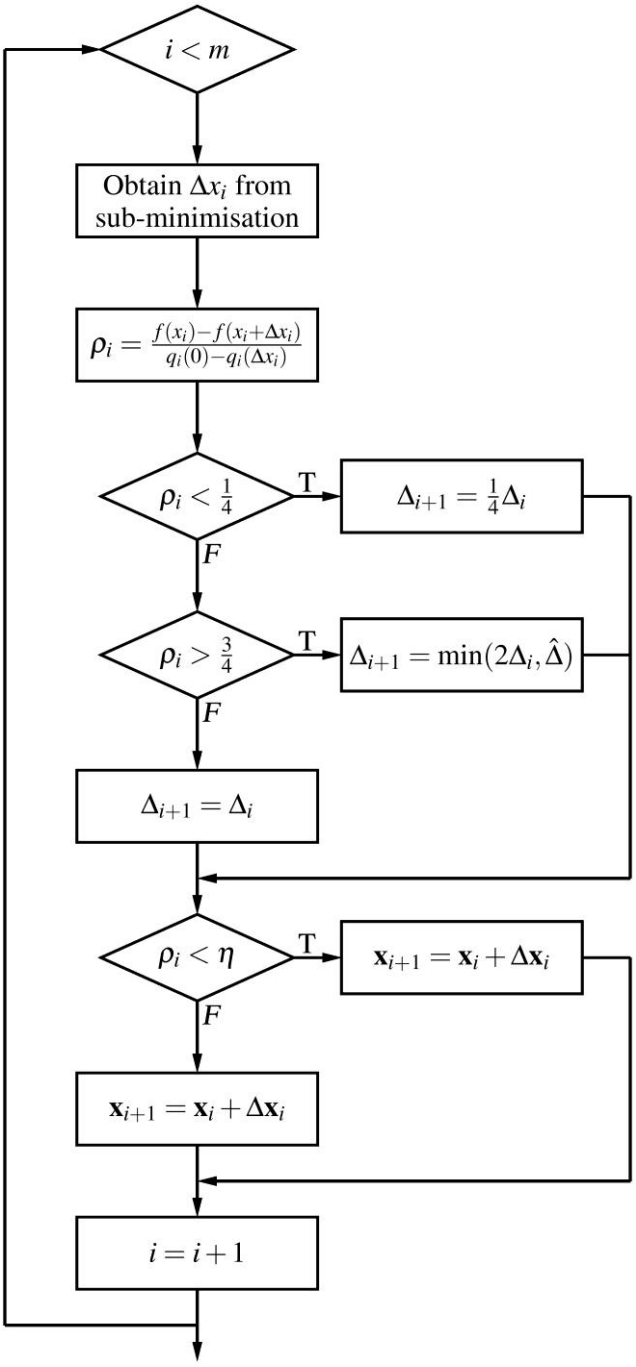


Figure 2-18: Flow diagram of trust-region method.

Figure 2-19 nicely shows the difference between the steps of the methods classified as Line search methods and Trust-region methods. The Newton and the Gauss-Newton methods belong to the group of algorithms referred to as line search algorithms and the Trust-Region and Levenberg-Marquardt methods belong to the group of trust region algorithms. All four methods use the quadratic model to model the objective function, but the Gauss-Newton and Levenberg-Marquardt methods use an approximation of the Hessian matrix.

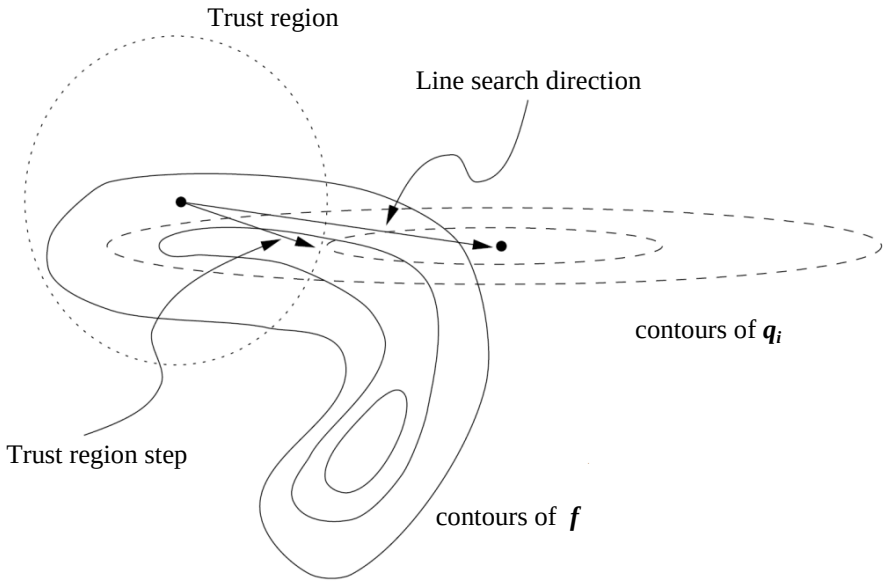


Figure 2-19: Trust-region and line search steps [45]**.**

2.4 **MATLAB**

This section reviews software products with the ability to implement mathematical models, have optimisation algorithms for performing parameter estimation and the ability to interface with other software packages.

MATLAB has the ability to perform all the activities mentioned above. MATLAB is a contraction of Matrix and Laboratory, as the matrix part spurs from the fact that MATLAB is very powerful at performing matrix operations. MATLAB was developed by Cleave Moler, a professor at the University of New Mexico. His aim was to give students access to FORTRAN subroutine libraries without FORTRAN programming knowledge. In 1983, a Graphical User Interface (GUI) was developed in C, and in 1984, MATLAB was commercialised by MathWorks, Inc [58]. Since then, MATLAB has become one of the most used numerical analysis software packages by commercial users, as well as researchers, with more than 1 million users worldwide [59].

In 2002, MathWorks released an extension called Simulink (from Simulation and Link) for MATLAB creating "*an environment for multidomain simulation and Model-Based Design for dynamic and embedded systems. It provides an interactive graphical environment and a customisable set of block libraries that let you design, simulate, implement, and test a variety of time-varying systems*" [59]. Simulink also provides the user with the flexibility to develop models in different programming languages. MathWorks has produced about 100 additional add-ons and toolboxes ranging from products for the financial sector to digital image

processing toolboxes [59]. There are also hundreds of third-party solutions available that build on MATLAB and Simulink [59].

Simulink also provides the capability for implementing Simulink models in low-level coding (C, FORTAN, etc.) languages, as well as the ability to develop user defined Simulink toolboxes [60]. Furthermore, as a result of the large market penetration of MATLAB, commercial specialised analysis software (e.g., the power system software DIgSILENT and PSS [61]) makes provision for interfacing with MATLAB/Simulink.

There exist a few well-known MATLAB-like numerical analysis software packages:

- GNU Octave
- SciLab
- Rlab Plus
- SciPy
- IDL

All of these are freeware except for IDL, by ITT Visual Information Solutions. The first three are MATLAB clones of which the compatibility with MATLAB is the highest for Octave and the lowest for Rlab. In the case of SciPy and IDL, there is basically no compatibility, but they have the same kind of capabilities as MATLAB. SciPy achieve this capability by bundling a set of Python packages together. IDL interpreted language is based on FORTRAN and is mostly used in the field of Digital Signal Processing (DSP).

All of these have optimisation capability making use of the Levenberg–Marquardt algorithm either through a build-in algorithm or by wrapping MINPACK coded in FORTRAN , which is the oldest implementation still in use [62-64]. Octave and SciLab are the only two that have Simulink-like extensions, but neither of these is currently able to compete with the well established Simulink and its wide range of toolboxes.

From this it is concluded that MATLAB would be best suited for this study. It, together with the Simulink, provides functionality that is ideal as a research environment for the modelling of a wind turbine system. It also has the required optimisation algorithms required for the parameter estimation process, and its ability to interact with other software packages leaves room for future research.

3 MODELLING OF SYSTEM COMPONENTS, IMPLEMENTATION AS SIMULINK MODELS USING S-FUNCTIONS AND DEVELOPING OF TOOLBOX COMPRISING OF IMPLEMENTED MODELS

3.1 Overview

In this chapter, the modelling as well as choices behind the modelling of all the components of the wind turbine system will be discussed. This is followed by the discussion of the implementation of these models as Simulink block models making use of S-function blocks in conjunction with C-code. This chapter is concluded with a discussion on how the S-function models can be integrated as a Simulink toolbox.

3.2 Modelling of components of wind turbine system

3.2.1 Introduction

As discussed in Chapter 2, there are numerous types of wind turbine system topologies. As shown in Figure 3-1, these can be broken up into four major blocks, namely the aerodynamic block, mechanical block, electrical block and control block. This section looks at the different blocks and the modelling of these components.

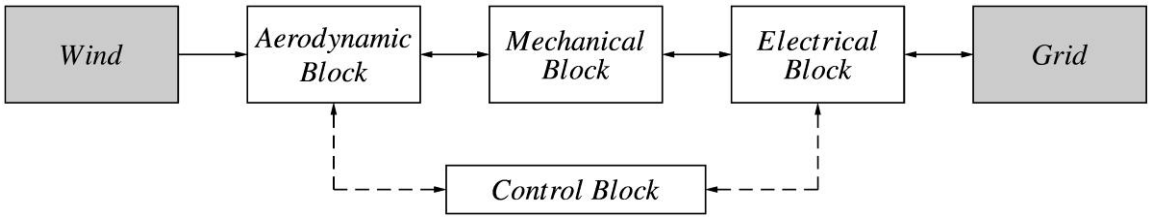


Figure 3-1: Total wind turbine system [13].

3.2.2 Aerodynamic block

The aerodynamic block models the extraction of mechanical power from energy in the wind. The block represents the physical rotor blade of the real wind turbine system. The mechanical power is then either fed directly to the electrical block or fed through the mechanical block to the electrical block.

To obtain the mechanical energy extracted from the wind, the kinetic energy E_k [J] of the wind mass needs to be considered. The kinetic energy contained in a mass m [kg] of moving air can be calculated using

$$E_k = \frac{1}{2} m v^2 \tag{3.1}$$

where v denotes the wind speed [m/s] [65]. From (3.1), the equation for the total mechanical power P_T [W] in the mass of moving air can be obtained as

$$P_T = \frac{1}{2} \dot{m} v^2 \tag{3.2}$$

where $\dot{m}$ denotes mass flow per second [kg/s]. The flow can be calculated using

$$\dot{m} = \rho A v \tag{3.3}$$

where ρ denotes air density [kg/m^3] and A denotes area swept by the blades [m^2]. Substituting (3.3) into (3.2), the total mechanical power as a function of swept area and wind speed are obtained as

$$P_T = \frac{1}{2} \rho A v^3 . \tag{3.4}$$

It is not possible to extract all the power from the wind. The amount of energy that can be extracted depends on the aerodynamics of the rotor blade. It is sufficient to know that the ratio of total mechanical power to extracted mechanical power is known as the power coefficient, denoted as C_p. This yields the equation of extracted mechanical power P as

$$P = \frac{1}{2} \rho A v^3 C_p . \tag{3.5}$$

The power coefficient is a function of the rotational speed of the turbine, wind speed and pitch angle of the blade parameters. The power coefficient can be obtained from the manufacturers or by experimental measurements. The rotational speed and the wind speed are generally combined into a single variable, namely the Tip Speed Ratio (TSR), denoted as λ. This yields

$$\lambda = \frac{R\omega}{v} \tag{3.6}$$

where

R denotes the length of the rotor blade [m],

ω denotes the mechanical angular velocity [rad/s]

and

v denotes wind speed [m/s].

Figure 2-11 shows a typical power coefficient graph with the different lines for the different blade pitch angles and the TSR as the x-axis.

The angular velocity can be calculated from

$$\omega = \frac{2\pi n}{60} \tag{3.7}$$

where n denotes rotational speed in [$r/\min$].

The equation for torque is obtained as

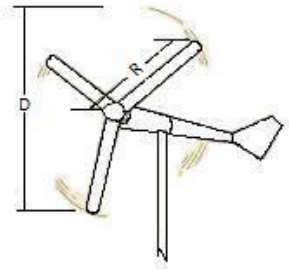
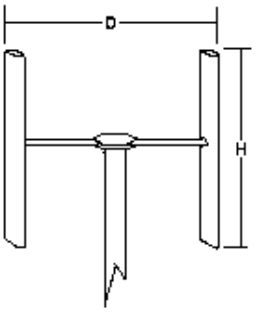
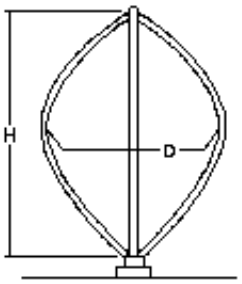
$$T = \frac{0.5\rho A v^3 C_p}{\omega} \tag{3.8}$$

by dividing (3.5) by angular velocity ω.

The swept area A depends on the rotor blade configuration as shown in Table 3-1. As discussed in Chapter 2, the focus of this study is on the conventional rotor blade configuration. Therefore, substituting the swept area of the conventional rotor into (3.8) yields

$$T = \frac{0.5\rho\pi R^2 v^3 C_p}{\omega}. \tag{3.9}$$

Table 3-1: Different rotor blade configuration with swept area calculation [14]**.**

Rotor blade configuration	 Conventional rotor	 H-rotor	 Darrieus rotor
Swept Area	$A = \pi R^2$	$A = DH$	$A = 0.65DH$

Another commonly used equation for the torque is

$$T = 0.5\pi\rho R^3 v^2 C_q \tag{3.10}$$

where C_q denotes the torque coefficient defined as C_p/λ.

Equation (3.9) provides a mathematical model of the turbine rotor with torque as output, wind speed, hub speed and air density as input, and blade length and power coefficient as parameters. The different ways to control power or torque through changing the blade characteristics, as discussed in Chapter 2, are accounted for by the power coefficient in the mathematical model. Most wind turbine systems have a mechanical brake that stops the rotor blades for wind speeds lower than the cut-in speed and wind speeds higher than the cut-out speed. Therefore, two extra parameters need to be added to account for the cut-in and cut-out speeds of the rotor blades.

Figure 3-2 shows the aerodynamic block with input and output variables as defined in Table 3-2 and parameters as defined in Table 3-3.

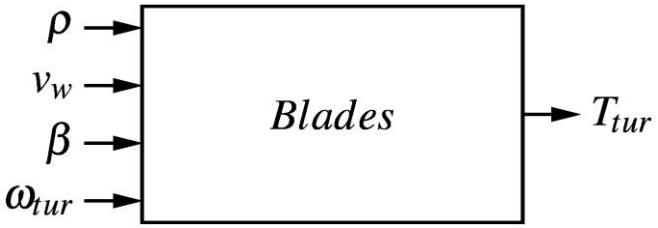


Figure 3-2: Block diagram of aerodynamic model.

Table 3-2: Input and output variable definitions of the aerodynamic model.

Variable	Description	Unit	Variable	Description	Unit
ρ	Air Density	kg/m^3	ω_{tur}	Turbine Angular Velocity	rad/s
v_w	Wind Speed	m/s	T_{tur}	Turbine Torque	Nm
β	Blade Pitch Angle	degrees			

Table 3-3: Parameter definitions of the aerodynamic model.

Parameter	Description	Unit	Parameter	Description	Unit
R	Blade Length	m	v_{out}	Cut-out Wind Speed	m/s
v_{in}	Cut-in Wind Speed	m/s	C_p	Power Coefficient	

3.2.3 Mechanical block

The mechanical block forms the connection between the aerodynamic block and the electrical block. The main component of the mechanical block is the gearbox. In Chapter 2 it was stated that the market is moving in the direction of the generator with full converter topology, and especially the direct drive generator, thus removing the need for a gearbox. The mechanical block will be modelled using a gearbox. In the case of the direct drive system, the mechanical block can still be used to model the inertias in the system and the damping and stiffness coefficients of the shaft.

There are different ways of modelling the gearbox, e.g., the three-mass, two-mass or one-mass models. Because the complex three-mass model can be reduced to the simpler two-mass model without losing the dynamic behaviour of the model, the two-mass model was chosen to model the gearbox.

Figure 3-3 shows the three-mass gearbox model with J_{tur} and J_{gen} denoting the rotor blade's actual inertia and generator's actual inertia in [Nm] respectively. In reducing the three-mass model to the two-mass model, the stiffness K_{tur} and damping D_{tur} of the rotor blade shaft are combined with the stiffness K_{gen} and damping D_{gen} of the generator shaft to form a single equivalent shaft with damping D and equivalent stiffness K in [Nms/rad] and [Nm/rad] respectively. The inertias of the shafts and gearbox are neglected as they are small compared to the inertia of the generator and rotor blades. Therefore, only the gear ratio k is used in the two-mass model [13].

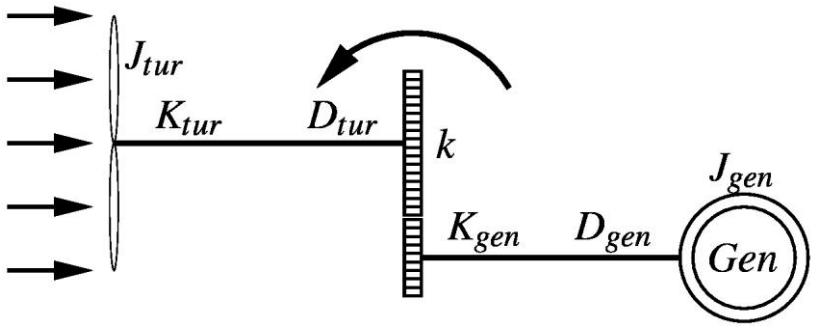


Figure 3-3: Three-mass gearbox model [13]**.**

By referring the generator side through the gearbox, the three-mass model is reduced to the two-mass model, as shown in Figure 3-4, where the equivalent damping D and equivalent stiffness K relationships are given as [13]

$$D = D_{tur} + D_{gen}k^2 \tag{3.11}$$

and

$$K = K_{tur} + K_{gen}k^2 . \tag{3.12}$$

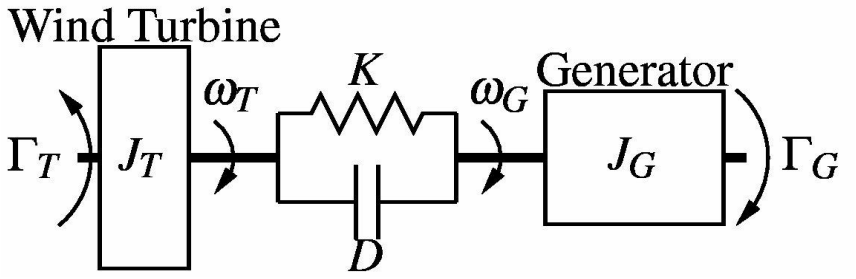


Figure 3-4: Two-mass gearbox model [66]**.**

The other relationships between the two- and three-mass models are given as [13]

$$\begin{aligned} J_T &= J_{rot} & J_G &= J_{gen}k_{gear}^2 \\ \Gamma_T &= \Gamma_{rot} & \Gamma_G &= k_{gear}\Gamma_{gen} \\ \omega_T &= \omega_{rot} & \omega_G &= \omega_{gen} / k_{gear} \\ \theta_T &= \theta_{rot} & \theta_G &= \theta_{gen} / k_{gear} \end{aligned} , \tag{3.13}$$

where the subscripts are defined as

T referring to the rotor blades of the two-mass model,

G referring to the generator referred to the rotor blades side of the two-mass model,

tur referring to the rotor blades of the three-mass model,

and

gen referring to the generator of the three-mass model.

The variables are defined as

J = inertia,

Γ = torque,

ω = angular velocity

and

θ = angular position.

The dynamic torque equations for Figure 3-4 are given by

$$
\begin{aligned}
&J_T \frac{d^2\theta_T}{dt^2} + D\left(\omega_T - \omega_G\right) + K\left(\theta_T - \theta_G\right) = -\Gamma_T \\
&J_G \frac{d^2\theta_G}{dt^2} + D\left(\omega_G - \omega_T\right) + K\left(\theta_G - \theta_T\right) = \Gamma_G
\end{aligned}
\quad . \tag{3.14}
$$

Substituting the relationships of (3.13) into (3.14) yields

$$
\begin{aligned}
&J_T \frac{d^2\theta_T}{dt^2} + D\left(\omega_T - \omega_G\right) + K\left(\theta_T - \theta_G\right) = -\Gamma_T \\
&\Rightarrow \dot{\omega}_{tur} = \frac{-D\left(\omega_{tur} - \dfrac{\omega_{gen}}{k_{gear}}\right) - K\left(\theta_{tur} - \dfrac{\theta_{gen}}{k_{gear}}\right) - \Gamma_{tur}}{J_{tur}}
\end{aligned}
\tag{3.15}
$$

and

$$
\begin{aligned}
&J_G \frac{d^2\theta_G}{dt^2} + D\left(\omega_G - \omega_T\right) + K\left(\theta_G - \theta_T\right) = \Gamma_G \\
&\Rightarrow \dot{\omega}_{gen} = \frac{-D\left(\omega_{gen} - k_{gear}\omega_{tur}\right) - K\left(\theta_{gen} - k_{gear}\theta_{tur}\right) + k_{gear}^2\Gamma_{gen}}{J_{gen}k_{gear}^2}
\end{aligned}
\tag{3.16}
$$

which represent the mathematical model of the mechanical block. Figure 3-5 shows the block diagram of the mechanical block with input and output variables as defined in Table 3-4 and Table 3-5.

Figure 3-5: Block diagram of mechanical model.

Table 3-4: Input and output variable definitions of the mechanical model.

Variable	Description	Unit	Variable	Description	Unit
T_{tur}	Turbine Torque	Nm	ω_{tur}	Turbine Angular Velocity	rad/s
T_{gen}	Generator Torque	Nm	ω_{gen}	Generator Angular Velocity	rad/s

Table 3-5: Parameter definitions of the mechanical model.

Parameter	Description	Unit	Parameter	Description	Unit
J_{gen}	Generator Moment of Inertia	kg.m^2	D	Shaft Damping coefficient	Nm.s/rad
J_{tur}	Turbine Moment of Inertia	kg.m^2	GR	Gear Ratio	
K	Shaft Stiffness coefficient	Nm/rad			

3.2.4 Electrical block

3.2.4.1 Introduction

The electrical block represents the components responsible for converting the mechanical power to electrical power that can be fed into the power grid. Depending on the topology of the wind turbine system, the electrical block can consist of only an induction generator for the fixed speed generator topology, a generator with full-converter for the generator with fully-rated converter topology or a Double-Fed Induction Generator (DFIG) with partial-rated inverter for the double-fed induction generator topology.

The DFIG can also be used as an induction generator by applying a zero voltage to the rotor terminals. Therefore, it was decided to model the generator component as a DFIG that will be used as an induction generator for the fixed speed topology and that can later be extended to the DFIG topology by modelling the power converter. This section discusses the modelling of the DFIG, starting with the derivation and discussion of the ABC model and followed by the derivation and discussion of the DQ model.

3.2.4.2 Double-fed induction generator ABC model

Figure 3-6 shows the simplified diagram of a DFIG with subscripts a, b and c referring to phases A, B and C, subscripts r referring to the rotor winding and subscripts s referring to the stator winding. The basic equations for the terminal voltages, assuming sinusoidal magnetomotive force and neglecting saturation and losses in the core, are as follows [67, 68]:

$$\begin{aligned} v_{as} &= r_{as} i_{as} + \frac{d\lambda_{as}}{dt} & v_{ar} &= r_{ar} i_{ar} + \frac{d\lambda_{ar}}{dt} \\ v_{bs} &= r_{bs} i_{bs} + \frac{d\lambda_{bs}}{dt} & v_{br} &= r_{br} i_{br} + \frac{d\lambda_{br}}{dt} \\ v_{cs} &= r_{cs} i_{cs} + \frac{d\lambda_{cs}}{dt} & v_{cr} &= r_{cr} i_{cr} + \frac{d\lambda_{cr}}{dt} \end{aligned} \tag{3.17}$$

where

r_{as}, r_{bs}, r_{cs} = stator winding resistances of phases A, B and C [Ω] respectively,

r_{ar}, r_{br}, r_{cr} = rotor winding resistances of phases A, B and C [Ω] respectively,

i_{as}, i_{bs}, i_{cs} = stator winding currents of phases A, B and C [A] respectively,

i_{ar}, i_{br}, i_{cr} = rotor winding currents of phases A, B and C [A] respectively,

λ_{as}, λ_{bs}, λ_{cs} = stator winding phases A, B and C flux-linkages [Wb] respectively,

λ_{ar}, λ_{br}, λ_{cr} = rotor winding phases A, B and C flux-linkages [Wb] respectively,

v_{as}, v_{bs}, v_{cs} = stator winding voltages of phases A, B and C [V] respectively

and

v_{ar}, v_{br}, v_{cr} = rotor winding voltages of phase A, B and C [V] respectively.

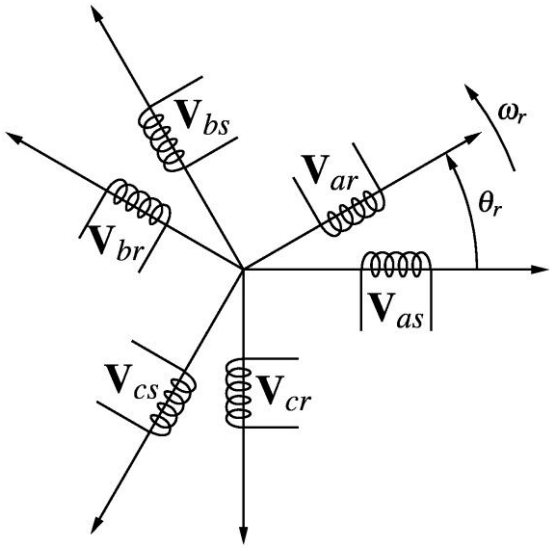


Figure 3-6: Three-phase machine diagram [67].

The flux-linkage of a single phase, for example phase A λ_{as}, consists of a self inductance and leakage inductance due to the current flowing in the winding, as well as all the mutual inductances due to the currents flowing in the other windings. This gives rise to the relationship

$$\lambda_{as} = L_{asas} i_{as} + L_{asbs} i_{bs} + L_{ascs} i_{cs} + L_{asar} i_{ar} + L_{asbr} i_{br} + L_{ascr} i_{cr} . \tag{3.18}$$

Deriving the equations for the flux-linkage of the other windings and substituting it into (3.17) yields the terminal voltages in matrix for form as

$$\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \\ v_{ar} \\ v_{br} \\ v_{cr} \end{bmatrix} = \begin{bmatrix} r_{as} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{bs} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{cs} & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{ar} & 0 & 0 \\ 0 & 0 & 0 & 0 & r_{br} & 0 \\ 0 & 0 & 0 & 0 & 0 & r_{cr} \end{bmatrix} + p \left\{ \begin{bmatrix} L_{asas} & L_{asbs} & L_{ascs} & L_{asar} & L_{asbr} & L_{ascr} \\ L_{bsas} & L_{bsbs} & L_{bscs} & L_{bsar} & L_{bsbr} & L_{bscr} \\ L_{csas} & L_{csbs} & L_{cscs} & L_{csar} & L_{csbr} & L_{cscr} \\ L_{aras} & L_{arbs} & L_{arcs} & L_{arar} & L_{arbr} & L_{arcr} \\ L_{bras} & L_{brbs} & L_{brcs} & L_{brar} & L_{brbr} & L_{brcr} \\ L_{cras} & L_{crbs} & L_{crcs} & L_{crar} & L_{crbr} & L_{crcr} \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{ar} \\ i_{br} \\ i_{cr} \end{bmatrix} \right\} \tag{3.19}$$

where p denotes the time derivative, L_{asas}, L_{bsbs}, L_{cscs}, L_{arar}, L_{brbr}, L_{crcr} denote the self-inductances plus the leakage inductances [H] and all other inductances denote the mutual inductances between phases [H].

Expanding the derivative in (3.19) yields the matrix equation

$$\mathbf{V} = \mathrm{R}\mathbf{I} + \frac{d\,\mathrm{L}}{dt}\mathbf{I} + \mathrm{L}\frac{d\,\mathbf{I}}{dt}. \tag{3.20}$$

From Figure 3-6 it is clear that the self and mutual inductances in matrix L are a function of electrical rotor position θ_r. Applying the chain rule to the time derivative of the induction matrix in (3.20) gives

$$\frac{d\,\mathrm{L}}{dt} = \frac{d\,\mathrm{L}}{d\theta_r}\frac{d\theta_r}{dt} = \frac{d\,\mathrm{L}}{d\theta_r}\omega_r \tag{3.21}$$

where ω_r denotes the electrical rotational speed [rad/s]. Substituting (3.21) into (3.20) yields

$$\mathbf{V} = \mathrm{R}\mathbf{I} + \frac{d\,\mathrm{L}}{d\theta_r}\omega_r\mathbf{I} + \mathrm{L}\frac{d\mathbf{I}}{dt}. \tag{3.22}$$

Assuming the rotor and stator to be electrically and magnetically symmetrical, the resistance matrix and inductance matrix can be simplified to yield [68]

$$\mathrm{R} = \begin{bmatrix} r_s & 0 & 0 & 0 & 0 & 0 \\ 0 & r_s & 0 & 0 & 0 & 0 \\ 0 & 0 & r_s & 0 & 0 & 0 \\ 0 & 0 & 0 & r_r & 0 & 0 \\ 0 & 0 & 0 & 0 & r_r & 0 \\ 0 & 0 & 0 & 0 & 0 & r_r \end{bmatrix} \tag{3.23}$$

and

$$\mathrm{L} = \begin{bmatrix} L_s & -0.5M_{sr} & -0.5M_{sr} & M_{sr}a_1 & M_{sr}a_2 & M_{sr}a_3 \\ -0.5M_{sr} & L_s & -0.5M_{sr} & M_{sr}a_3 & M_{sr}a_1 & M_{sr}a_2 \\ -0.5M_{sr} & -0.5M_{sr} & L_s & M_{sr}a_2 & M_{sr}a_3 & M_{sr}a_1 \\ M_{sr}a_1 & M_{sr}a_3 & M_{sr}a_2 & L_r & -0.5M_{sr} & -0.5M_{sr} \\ M_{sr}a_2 & M_{sr}a_1 & M_{sr}a_3 & -0.5M_{sr} & L_r & -0.5M_{sr} \\ M_{sr}a_3 & M_{sr}a_2 & M_{sr}a_1 & -0.5M_{sr} & -0.5M_{sr} & L_r \end{bmatrix} \tag{3.24}$$

respectively, where

M_{sr} = the mutual inductance between a stator and rotor winding,

$L_s = L_{ls} + M_{sr}$ where L_{ls} is the leakage inductance of a stator winding,

$L_r = L_{lr} + M_{sr}$ where L_{lr} is the leakage inductance of a rotor winding,

$a_1 = \cos(\theta_r)$,

$a_2 = \cos\left(\theta_r + \frac{2\pi}{3}\right)$

and

$a_3 = \cos\left(\theta_r - \frac{2\pi}{3}\right)$.

Rearranging (3.22) into state-variable form with current as the state variable gives

$$\frac{d\,\mathrm{I}}{dt} = \mathrm{L}^{-1}\left\{-\mathrm{R} - \omega_r \frac{d\,\mathrm{I}}{d\theta_r}\right\}\mathrm{I} + \mathrm{L}^{-1}\mathrm{V} \tag{3.25}$$

from which the currents of the DFIG can be calculated.

By multiplying (3.22) by the transpose of the current vector, the equation for the instantaneous power P_{ins} is obtained. This yields [66]

$$P_{ins} = \mathrm{I}^T\mathrm{V} = \mathrm{I}^T\mathrm{RI} + \mathrm{I}^T \frac{d\,\mathrm{L}}{d\theta_r}\omega_r\mathrm{I} + \mathrm{I}^T\mathrm{L}\frac{d\,\mathrm{I}}{dt} \tag{3.26}$$

where

$\mathrm{I}^T\mathrm{R}\,\mathrm{I}$ = copper losses in the generator windings P_{copper},

$\mathrm{I}^T\mathrm{L}\frac{d\,\mathrm{I}}{dt}$ = magnetic power stored in the generator $P_{magnetic}$

and

$\omega_r\mathrm{I}^T \frac{d\,\mathrm{L}}{d\theta_r}\mathrm{I}$ = mechanical power P_{mech}.

The mechanical torque T in [Nm] is obtained by dividing the mechanical power by the mechanical rotational speed of the rotor Ω_r to yield [68]

$$T = \frac{P_{mech}}{\Omega_r} \tag{3.27}$$

where $\Omega_r = \frac{2\omega_r}{P}$ and P denotes the total number of poles in the generator. Substituting the mechanical torque and mechanical rotational speed into (3.27) yields

$$T = \frac{P}{2}\mathrm{I}^T \frac{d\,\mathrm{L}}{d\theta_r}\mathrm{I}. \tag{3.28}$$

Further substitution of the inductance matrix given in (3.24) and performing the matrix operations yields the following equation for the torque of the generator [68]:

$$T = -\frac{P}{2}M_{sr}\left\{\left(i_{as}i_{ar}+i_{bs}i_{br}+i_{cs}i_{cr}\right)\sin\left(\theta_r\right)+\left(i_{as}i_{br}+i_{bs}i_{cr}+i_{cs}i_{ar}\right)\sin\left(\theta_r+\tfrac{2\pi}{3}\right)+\left(i_{as}i_{cr}+i_{bs}i_{ar}+i_{cs}i_{br}\right)\sin\left(\theta_r-\tfrac{2\pi}{3}\right)\right\}. \tag{3.29}$$

The state space equation, (3.25), together with the torque equation, (3.29), models the DFIG. This model is known as ABC model and is represented by the system block shown in Figure 3-7 with input and output variables as defined in Table 3-6 and Table 3-7.

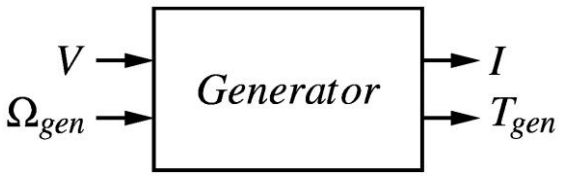


Figure 3-7: Block diagram of electrical model.

Table 3-6: Input and output variable definitions of the electrical model.

Variable	Description	Unit	Variable	Description	Unit
$\mathbf{V}_{abc}$	Stator and Rotor Voltage	V	$\mathbf{I}_{abc}$	Stator and Rotor Current	A
Ω_{gen}	Generator Mechanical Angular Velocity	rad/s	T_{gen}	Generator Torque	Nm

Table 3-7: Parameter definitions of the electrical model.

Parameter	Description	Unit	Parameter	Description	Unit
R_s	Stator resistance	Ω	L_r	Rotor inductance	H
R_r	Rotor resistance	Ω	L_m	Magnetising inductance	H
L_s	Stator inductance	H	P	Number of poles in machine	

From (3.25) it is clear that the inverse of the inductance matrix is required to obtain the currents. Equation (3.24) shows that some of the elements of the inductance matrix are a

function of the electrical rotor position θ_r and thus a function of time. This implies that the inverse of the inductance matrix needs to be calculated at each step in the simulation, leading to long simulation times [68].

There are two ways of speeding up the simulation time. One is to analytically inverse the inductance matrix. This process is generally difficult, but if the phase impedances are symmetrical and the currents of the rotor and stator are balanced, i.e.,

$$i_{as} + i_{bs} + i_{cs} = 0 \tag{3.30}$$

and

$$i_{ar} + i_{br} + i_{cr} = 0, \tag{3.31}$$

an explicit expression can easily be obtained. Using the explicit expression of the inverse inductance matrix, an explicit expression for the derivatives of the currents can be obtained. This yields

$$\frac{d\,\mathbf{I}_{abc}}{dt} = \mathrm{A}\mathbf{I}_{abc} + \mathrm{B}\mathbf{V}_{abc} \tag{3.32}$$

where

$$\mathbf{I}_{abc} = \begin{bmatrix} i_{as} & i_{bs} & i_{cs} & i_{ar} & i_{br} & i_{cr} \end{bmatrix}^T,$$

$$\mathbf{V}_{abc} = \begin{bmatrix} v_{as} & v_{bs} & v_{cs} & v_{ar} & v_{br} & v_{cr} \end{bmatrix}^T$$

and

A and B are 6 x 6 matrices with values as given in APPENDIX B.

As discussed by Pillay in [68], this explicit expression produces the same results as the original ABC model but with reduced simulation time because (3.32) integrates faster than (3.25).

The second way of speeding up the simulation time is by modelling the DFIG by making use of the Direct–Quadrature (DQ) reference frame. This process is discussed in the following section.

3.2.4.3 Double-fed induction generator Direct–Quadrature (DQ) model

The aim of this section is to model a DFIG making use of the DQ reference frame. The section starts with the modelling of a two-phase machine and then looks at how to mathematically convert a three-phase machine into a two-phase machine.

Figure 3-8 shows the diagram of a two-phase machine. The terminal voltages can be expressed as (3.33) [69-71].

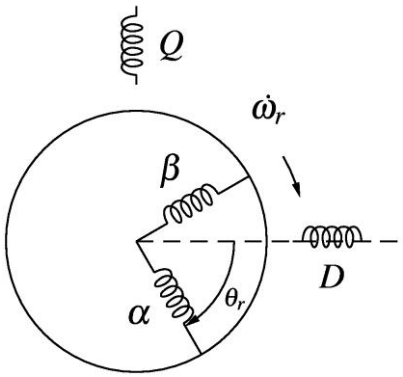


Figure 3-8: Two-phase machine diagram [71]**.**

$$\begin{aligned}
v_D &= R_{DD}i_D + p\left(L_{DD}i_D\right) + p\left(L_{DQ}i_Q\right) + p\left(L_{D\alpha}i_\alpha\right) + p\left(L_{D\beta}i_\beta\right) \\
v_Q &= p\left(L_{QD}i_D\right) + R_{QQ}i_Q + p\left(L_{QQ}i_Q\right) + p\left(L_{Q\alpha}i_\alpha\right) + p\left(L_{Q\beta}i_\beta\right) \\
v_\alpha &= p\left(L_{\alpha D}i_D\right) + p\left(L_{\alpha Q}i_Q\right) + R_{\alpha\alpha}i_\alpha + p\left(L_{\alpha\alpha}i_\alpha\right) + p\left(L_{\alpha\beta}i_\beta\right) \\
v_\beta &= p\left(L_{\beta D}i_D\right) + p\left(L_{\beta Q}i_Q\right) + p\left(L_{\beta\alpha}i_\alpha\right) + R_{\beta\beta}i_\beta + p\left(L_{\beta\beta}i_\beta\right)
\end{aligned} \tag{3.33}$$

Since the air gap is mostly uniform, the assumption is made that all the self-inductances will be independent of the angular position of the rotor and may be regarded as constant if saturation is ignored. Using this assumption and the assumption that the windings are balanced gives that $L_{DD} = L_{QQ}$, $L_{\alpha\alpha} = L_{\beta\beta}$, $R_{DD} = R_{QQ}$ and $R_{\alpha\alpha} = R_{\beta\beta}$. These equalities are substituted by L_S, L_r, R_S and R_r respectively. The symmetry of the arrangement shows that there will be no linkage with any winding by flux set by a current in the winding at 90° to it. Therefore, $L_{\alpha\beta} = L_{\beta\alpha} = 0$ and $L_{DQ} = L_{QD} = 0$. Substituting these into (3.33) yields [69]

$$\begin{aligned}
v_D &= R_S i_D + p\left(L_S i_D\right) + p\left(L_{D\alpha}i_\alpha\right) + p\left(L_{D\beta}i_\beta\right) \\
v_Q &= R_S i_Q + p\left(L_S i_Q\right) + p\left(L_{Q\alpha}i_\alpha\right) + p\left(L_{Q\beta}i_\beta\right) \\
v_\alpha &= p\left(L_{\alpha D}i_D\right) + p\left(L_{\alpha Q}i_Q\right) + R_r i_\alpha + p\left(L_r i_\alpha\right) \\
v_\beta &= p\left(L_{\beta D}i_D\right) + p\left(L_{\beta Q}i_Q\right) + R_r i_\beta + p\left(L_r i_\beta\right)
\end{aligned} \tag{3.34}$$

The remaining mutual inductances $L_{\alpha D}$, $L_{D\alpha}$ $L_{\alpha Q}$, $L_{Q\alpha}$, $L_{\beta D}$, $L_{D\beta}$ $L_{\beta Q}$ and $L_{Q\beta}$ are all functions of the electrical rotor position θ_r and are, therefore, functions of time. It is apparent that their variation with θ_r is cyclic with a period corresponding to one revolution of the rotor. For simplicity, it is desirable to assume that they vary sinusoidally, although this may only approximately be true in practice. Since all pairs of windings involved are similar and the air-gap is assumed uniform, the maximum value will be the same in all cases. This maximum value will come about when the axes of two windings are aligned. Thus, $L_{\alpha D} = M\cos\theta_r$, $L_{\beta D} = M\sin\theta_r$, $L_{\alpha Q} = -M\sin\theta_r$, $L_{\beta Q} = M\cos\theta_r$ and given $L_{\alpha D} = L_{D\alpha}$, $L_{\alpha Q} = L_{Q\alpha}$, $L_{\beta D} = L_{D\beta}$ and $L_{\beta Q} = L_{Q\beta}$. Substituting these into (3.34) yield [69]

$$\begin{aligned} v_D &= R_S i_D + L_S p i_D + Mp\left(\cos\theta_r i_\alpha\right) + Mp\left(\sin\theta_r i_\beta\right) \\ v_Q &= R_S i_Q + L_S p i_Q - Mp\left(\sin\theta_r i_\alpha\right) + Mp\left(\cos\theta_r i_\beta\right) \\ v_\alpha &= Mp\left(\cos\theta_r i_D\right) - Mp\left(\sin\theta_r i_Q\right) + R_r i_\alpha + p\left(L_r i_\alpha\right) \\ v_\beta &= Mp\left(\sin\theta_r i_D\right) + Mp\left(\cos\theta_r i_Q\right) + R_r i_\beta + p\left(L_r i_\beta\right) \end{aligned} . \tag{3.35}$$

The d-axis and q-axis terminal voltages, i.e., v_D and v_Q respectively, can be rearranged as

$$\begin{aligned} v_D &= R_S i_D + L_S p i_D + Mp\left(\cos\theta_r i_\alpha + \sin\theta_r i_\beta\right) \\ v_Q &= R_S i_Q + L_S p i_Q + Mp\left(-\sin\theta_r i_\alpha + \cos\theta_r i_\beta\right) \end{aligned} . \tag{3.36}$$

From (3.36) two new variables, namely i_d and i_q, can be introduced as [69, 71]

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}, \tag{3.37}$$

or given in symbolic format as

$$\left[i_{dq}\right] = \left[S\right]\left[i_{\alpha\beta}\right]. \tag{3.38}$$

Substituting i_d and i_q variables into (3.36) gives [69]

$$\begin{aligned} v_D &= R_S i_D + L_S p i_D + Mp i_d \\ v_Q &= R_S i_Q + L_S p i_Q + Mp i_q \end{aligned} . \tag{3.39}$$

Obtaining i_α and i_β from the inverse of (3.37) as

$$\begin{aligned} i_\alpha &= \cos\theta_r i_d - \sin\theta_r i_q \\ i_\beta &= \sin\theta_r i_d + \cos\theta_r i_q \end{aligned} \tag{3.40}$$

and substituting it into the equations for v_α and v_β in (3.35) yields

$$\begin{aligned} v_\alpha &= Mp\left(\cos\theta_r i_D\right) - Mp\left(\sin\theta_r i_Q\right) + R_r\left(\cos\theta_r i_d - \sin\theta_r i_q\right) + L_r p\left(\cos\theta_r i_d - \sin\theta_r i_q\right) \\ v_\beta &= Mp\left(\sin\theta_r i_D\right) + Mp\left(\cos\theta_r i_Q\right) + R_r\left(\sin\theta_r i_d + \cos\theta_r i_q\right) + L_r p\left(\sin\theta_r i_d + \cos\theta_r i_q\right) \end{aligned}. \tag{3.41}$$

Expanding the derivatives using the product rule gives

$$\begin{aligned} v_\alpha &= M\left[\cos\theta_r p i_D - \sin\theta_r \dot{\theta}_r i_D - \sin\theta p i_Q - \cos\theta_r \dot{\theta}_r i_Q\right] + \\ &\qquad R_r\left[\cos\theta_r i_d - \sin\theta_r i_q\right] + L_r\left[\cos\theta_r p i_d - \sin\theta_r \dot{\theta}_r i_d - \sin\theta_r p i_q - \cos\theta_r \dot{\theta}_r i_q\right] \\ v_\beta &= M\left[\sin\theta_r p i_D + \cos\theta_r \dot{\theta}_r i_D + \cos\theta_r p i_Q - \sin\theta_r \dot{\theta}_r i_Q\right] + R_r\left[\sin\theta_r i_d + \cos\theta_r i_q\right] + \\ &\qquad L_r\left[\sin\theta_r p i_d + \cos\theta_r \dot{\theta}_r i_d + \cos\theta_r p i_q - \sin\theta_r \dot{\theta}_r i_q\right] \end{aligned} \tag{3.42}$$

Two new voltages, v_d and v_q, are constructed using v_α, v_β and the *S* transformation; these are given as

$$\begin{aligned} v_d &= \cos\theta_r v_\alpha + \sin\theta_r v_\beta \\ v_q &= -\sin\theta_r v_\alpha \cos\theta_r v_\beta \end{aligned}. \tag{3.43}$$

Expressions for v_d and v_q in terms of i_d and i_q are obtained by substituting the equations for v_α and v_β given by (3.42) into (3.43), yielding

$$v_d = Mpi_D - \dot{\theta}_r Mi_Q + R_r i_d + L_r pi_d - \dot{\theta}_r i_q L_r \tag{3.44}$$

and

$$v_q = M\dot{\theta}_r i_D + pi_Q M + L_r \dot{\theta}_r i_d + R_r i_q + L_r pi_q . \tag{3.45}$$

Combining the equations for v_D, v_Q, v_d and v_q into matrix form yields [68, 69]

$$\begin{bmatrix} v_D \\ v_Q \\ v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + L_S p & 0 & Mp & 0 \\ 0 & R_S + L_S p & 0 & Mp \\ Mp & -\dot{\theta}_r M & R_r + L_r p & -\dot{\theta}_r L_r \\ \dot{\theta}_r M & Mp & \dot{\theta}_r L_r & R_r + L_r p \end{bmatrix} \begin{bmatrix} i_D \\ i_Q \\ i_d \\ i_q \end{bmatrix} \tag{3.46}$$

which can be expressed in symbolic format as

$$\mathbf{V}_{DQdq} = \mathrm{R}\,\mathbf{I}_{DQdq} + \mathrm{H}\,\omega_r\,\mathbf{I}_{DQdq} + \mathrm{L}\,\dot{\mathbf{I}}_{DQdq} \tag{3.47}$$

where

$$\mathbf{V}_{DQdq} = [v_D \quad v_Q \quad v_d \quad v_q]^T, \tag{3.48}$$

$$\mathbf{I}_{DQdq} = [i_D \quad i_Q \quad i_d \quad i_q]^T, \tag{3.49}$$

$$\mathrm{H} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -M & 0 & -L_r \\ M & 0 & L_r & 0 \end{bmatrix}, \tag{3.50}$$

$$\mathrm{R} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_r & 0 \\ 0 & 0 & 0 & R_r \end{bmatrix} \tag{3.51}$$

and

$$\mathrm{L} = \begin{bmatrix} L_S & 0 & M & 0 \\ 0 & L_S & 0 & M \\ M & 0 & L_r & 0 \\ 0 & M & 0 & L_r \end{bmatrix}. \tag{3.52}$$

Equation (3.47) can be rearranged to give the state vector form in terms of the current as

$$\dot{\mathbf{I}}_{DQdq} = -\mathrm{L}^{-1}\mathrm{G}\,\mathbf{I}_{DQdq} + \mathrm{L}^{-1}\,\mathbf{V}_{DQdq} \tag{3.53}$$

where

$$\mathrm{G} = \mathrm{R} + \mathrm{H}\omega_r .$$

From (3.52), it is clear that the inductance matrix is no longer a function of time. It is not necessary to invert a matrix at each time step of the simulation, which reduces the simulation time [68].

The instantaneous power P_{ins} can be calculated using [66]

$$P_{ins} = \mathbf{I}_{DQdq}{}^{T}\mathbf{V}_{DQdq}. \tag{3.54}$$

Substituting (3.47) into (3.54) yields [66]

$$P_{ins} = \mathbf{I}_{DQdq}{}^{T}\,\mathrm{R}\,\mathbf{I}_{DQdq} + \omega_r \mathbf{I}_{DQdq}{}^{T}\,\mathrm{H}\,\mathbf{I}_{DQdq} + \mathbf{I}_{DQdq}{}^{T}\mathrm{L}\,\dot{\mathbf{I}}_{DQdq} \tag{3.55}$$

where

$P_{copper} = \mathbf{I}_{DQdq}{}^{T}\mathrm{R}\,\mathbf{I}_{DQdq}$ is the copper losses in the generator windings,

$P_{mech} = \omega_r \mathbf{I}_{DQdq}{}^{T}\,\mathrm{H}\,\mathbf{I}_{DQdq}$ is the mechanical power

and

$P_{magnetic} = \mathbf{I}_{DQdq}{}^{T}\,\mathrm{L}\,\dot{\mathbf{I}}_{DQdq}$ is the magnetic power in the generator as a result of the variation in time of the magnetic energy.

The electromagnetic torque is obtained by dividing the mechanical power by the mechanical speeds as defined in (3.27) to give [66]

$$T = \frac{P}{2}\mathbf{I}_{DQdq}{}^{T}\mathrm{H}\,\mathbf{I}_{DQdq} = \frac{P}{2}M\left(i_D i_q - i_Q i_d\right). \tag{3.56}$$

The DQ state space model given by (3.53) requires DQ reference frame input voltages and produces DQ reference frame output currents. Therefore, the ABC reference frame input voltages need to be transformed to the DQ reference frame before being applied to the state space equation, and the DQ reference frame output currents need to be transformed back to the ABC reference frame. This transformation from the ABC reference frame to the DQ reference frame, and vice versa, is known as the DQO transform. Although this transform differs slightly from a similar transform proposed by Park in 1929, it is sometimes referred to as the park-transformation [72].

O'Kelly and Simmons [71] derive the DQO transformation current and voltage relationship between three-phase and stationary, or pseudo-stationary two-phase systems, as

$$\begin{bmatrix} x_d \\ x_q \\ x_o \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & \cos\left(\theta - 2\pi/3\right) & \cos\left(\theta + 2\pi/3\right) \\ -\sin\theta & -\sin\left(\theta - 2\pi/3\right) & -\sin\left(\theta + 2\pi/3\right) \\ \sqrt{1/2} & \sqrt{1/2} & \sqrt{1/2} \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \tag{3.57}$$

and the inverse relationship as

$$\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \cos\theta & -\sin\theta & \sqrt{1/2} \\ \cos\left(\theta - 2\pi/3\right) & -\sin\left(\theta - 2\pi/3\right) & \sqrt{1/2} \\ \cos\left(\theta + 2\pi/3\right) & -\sin\left(\theta + 2\pi/3\right) & \sqrt{1/2} \end{bmatrix} \begin{bmatrix} x_d \\ x_q \\ x_o \end{bmatrix}, \tag{3.58}$$

where x denotes current or voltage.

Using (3.57) with $x = v$, the three-phase voltages of the stator can be transformed to a stationary two-phase systems with $\theta = 0^\circ$. It can also be used to transform the three-phase rotor voltages to a pseudo-stationary two-phase voltages with $\theta = \theta_r$. By applying these two transforms, the three-phase generator shown in Figure 3-6 is transformed into a two-phase generator shown in Figure 3-8. This allows for calculation of the generator currents using (3.53) in the DQ reference frame, and these currents are then transformed back to three-phase currents using (3.58) with $x = i$ and $\theta = \theta_r$ for the pseudo-stationary two-phase current and with $\theta = 0^\circ$ for the stationary two-phase current. Consequently, although the voltages and currents are required in the DQ reference frame to solve the differential equations, (3.53), the inputs and outputs are still in the ABC frame work. The block diagram can still be represented by Figure 3-7 with input and output variables and parameters defined as in Table 3-6 and Table 3-7.

3.2.5 Control block

Modern wind turbine systems make use of sophisticated control systems to extract maximum power from the wind at all times. As discussed in Chapter 2, one way of controlling the extract power is by adjusting the pitch angle of the rotor blade. In the case of turbine topologies that make use of an electrical converter, the real power extracted as well as the reactive power can be controlled by controlling the converters. As mentioned earlier in Chapter 3, the electrical generator is simulated as a fix speed induction generator. Therefore, neither the electrical converter nor the control systems need to be modelled for the purpose of

this investigation. For future expansion of the electrical model by including the converter, references [73], [74] and [75] can be consulted for modelling of the control system.

3.3 Overview of S-function functionality and implementation process

This section provides an overview of the S-function functionality offered by MATLAB. This is followed by a discussion of the code topology by referring to the functions required for implementing a model as a C-code S-function.

Dynamic simulations combined with numerical optimisation routines induce high processor loads and result in long simulation times. Simulink makes provisions for implementing C-code models with its S-function block shown in Figure 3-9. This functionality can be used to reduce the overall model computation times, thereby reducing simulation times dramatically, especially for parameter estimation applications.

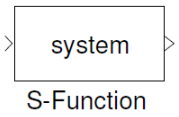


Figure 3-9: Simulink S-function block.

The S-function can be coded in C, Fortran, Ada or M [76]. This code needs to be coded in a MATLAB-specific format and compiled as a MEX-file using the *mex* utility in the MATLAB command window.

Figure 3-10 shows the window where the block parameters of the S-function can be set. The S-function name parameter requires the name of the C-code file that was compiled as a MEX-file without the extension. All the parameters required by the S-function model are entered as a vector in the S-function parameters field. The S-function modules field is only used for C-code S-function intended for use with Simulink Coder software that generates code.

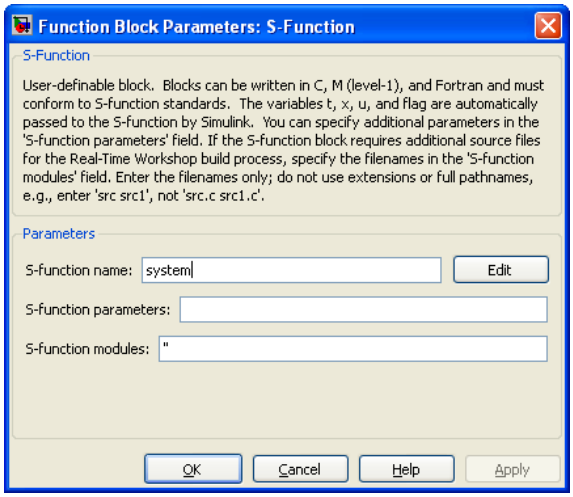


Figure 3-10: S-function – Function block parameter configuration window.

The flow diagram of the main function blocks associated with the S-function C-code is shown in Figure 3-11. This flow diagram consists of three sections, namely initialisation, simulation and termination.

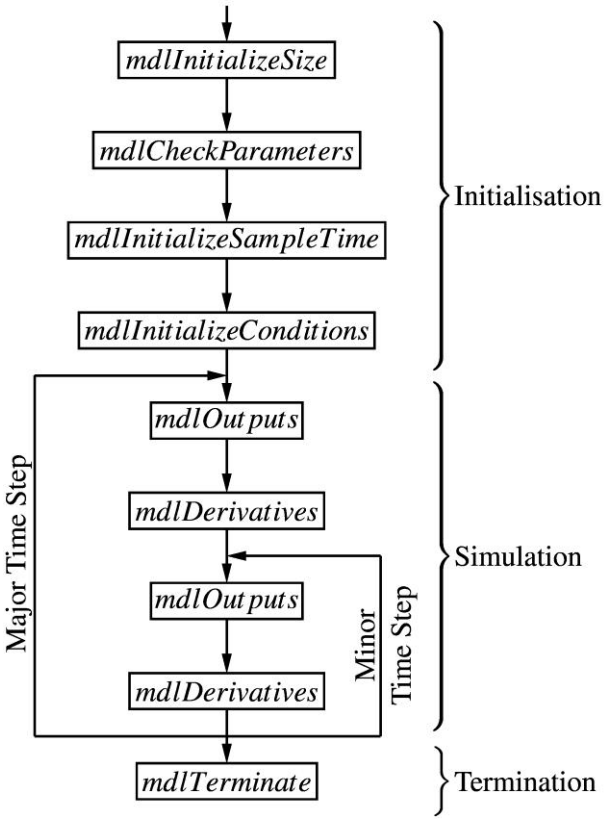


Figure 3-11: Flow diagram of the main functional components of C-code S-function [76]**.**

The initialisation section is responsible for configuring, verifying and initialising input ports, output ports, parameters, states, work vectors and sample times. This section consists of the following main functions: *mdlInitializeSize*, *mdlCheckParameters*, *mdlInitializeSample-Time*

and *mdlInitializeConditions*. The *mdlInitializeSize* is the first function the Simulink engine calls during a simulation. This function is responsible for the following:

- *Configuring of the number of parameters*: This is achieved using the *ssNumSFcnParams* command and for configuring whether parameters can be changed during the simulation, the *ssSetSFcnParamTunable* command is used. For this application, all parameters are set as tuneable, enabling the parameter estimation algorithm to change the parameter during simulation. After this command, a check is performed to verify that the correct number of parameters was entered into the S-functions using the *ssGetNumSFcnParams* and *ssGetSFcnParamsCountThis* commands.
- *Configuring of the number of continuous and discreet states*: Configuring the states is achieved by using the *ssSetNumberContStates* and *ssSetNumDiscStates* command respectively. For this application, all states are continuous.
- *Configuring the number and dimension of the input ports of the S-function*: The input ports are configured by using the *ssSetNumInputPorts* and *ssSetInputPort-DimensionInfo* commands respectively. If inputs are used in the *mdlOutputs* function the *ssSetInputPortDirectFeedThrough* command is used to set direct feed through for the required input ports.
- *Configuring the number and dimension of the output ports of the S-function*: This is achieved using *ssSetNumOutputPorts* and *ssSetOutputPortDimensionInfo* respectively.
- *Configure the number of sample times of the S-function*: This is achieved using the *ssSetNumSampleTimes* command.
- *Configuring the number of work vectors*: The number of work vectors are set using the ssSetNumDWork command. Their width and data type are set using the *ssSetDWorkWidth* and *ssSetDWorkDataType* respectively.
- *Configuring simulation options*: This is achieved using the *ssSetOptions*, e.g., speeding up the simulation by setting exception free code if the code does not contain any routines that have the potential of performing long-jumping; *mexErrMsgTxt*.

The *mdlCheckParameters* function is called from the *mdlInitializeSize* function after the number of parameters has been configured and whenever a parameter is changed in the dialog box. If the parameter is changed during a simulation step, it is called immediately. The

simulation however continues with the original parameter values and the function is recalled at the end of the simulation step. The parameter values are then changed for the next simulation step. This redundant call is needed to maintain simulation consistency. This function can only access the parameters and performs the task of verifying that the parameters meet predefined criteria such as dimension, data type and boundary values. The function displays an error message if criteria are not met.

The *mdlInitializeSampleTime* function is called to configure the sample time of the S-function using the *ssSetSampleTime* command to set whether it is a continuous, discreet or variable step. The *ssSetOffsetTime* command is used to set the offset while *ssSetModelReference-SampleTimeInheritance* command sets the S-function to inherit its sample time from the driving block.

The *mdlInitializeConditions* function is called to set the initial conditions of the states, outputs and work vectors as required.

The simulation section has two main functions; *mdlDerivatives* and *mdlOutputs*. The *mdlOutputs* function is called to update the output signals, and *mdlDerivatives* is called to solve the derivatives if required. For example, if a model is represented by the state space model given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.59}$$

where x denote the states, y denotes the output variables and u denotes the input variables, the code for the *mdlDerivatives* and *mdlOutputs* functions are given by (3.60) and (3.61) respectively.

$$\begin{aligned} dx[0] &= a_{11}x[0] + a_{12}x[1] \\ dx[1] &= a_{21}x[0] + a_{22}x[1] \end{aligned} \tag{3.60}$$

$$\begin{aligned} y[0] &= b_{11}x[0] + b_{12}x[1] + c_{11}u(0) + c_{12}u(1) \\ y[1] &= b_{21}x[0] + b_{22}x[1] + c_{21}u(0) + c_{22}u(1) \end{aligned} \tag{3.61}$$

The termination section is responsible for the housekeeping at the end of the simulation. This is done with the function *mdlTerminate*. It is a mandatory task which provides the S-function

with an opportunity to perform tasks such as clearing memory assigned to work vectors. If no tasks are required, the *UNUSED_ARG()* macro is used to indicate that the input argument is required but not used in the function.

Some practical considerations of importance in coding S-functions include ensuring that the correct working directory is used when compiling the MEX file, that the function has a unique name and that the states are initialised. Uninitialised states lead to random behaviour of the S-function. Standard C-code precautions apply, e.g., ensuring that the dimensioning of variables take cognisance of the arithmetic rules, etc.

3.4 Implementation of models as S-functions

3.4.1 Introduction

This section describes the process followed to implement the different models derived in Section 3.2 as S-function models by discussing the configuration that needs to be done in each of the functions described in Section 3.3. For the sake of clarity, some of the mathematical and/or state space equations of the individual models will be reproduce from which the different S-function functions will be discussed.

All three of the models, i.e., the aerodynamic, mechanical and electrical models, are continuous models which require only one sample time. The sample time can be configured similarly for each of these S-function models. The number of sample times can be set to one in the *mdlInitializeSize* function and is configured as continuous and to be inherited from the driving block in the *mdlInitializeSampleTime* function. Another configuration that applies to all models is the tune ability of the parameters, since these parameters will be changed by the parameter estimation algorithm during the simulation. All parameters are configured to be tuneable.

3.4.2 Aerodynamic block

The mathematical model for the aerodynamic block derived in Chapter 3, is represented by the block model shown in Figure 3-2, with input and output variables as defined in Table 3-2 and Table 3-3, from which the S-function can be constructed. The input ports, output ports and parameters of the S-function are configured in the *mdlInitializeSize* function. Four input ports, namely wind speed v, air density ρ, blade pitch β and angular velocity ω, are configured each with width set to one and the feed through flags set since all inputs are used

in the *mdlOutputs* function. One output port is configured for the torque T with width set to one. The *mdlInitializeSizes* function are further configured for four parameters, namely blade radius R, cut-in wind speed v_{in}, cut-out wind speed v_{out} and the power coefficient C_p matrix. The *mdlCheckParameters* function is configured to validate R, v_{in} and v_{out} as scalar variables with magnitudes greater than zero and the matrix dimensions of C_p matrix to be a 101 by 31 element matrix. This matrix accounts for a TSR range of 0 to 20 in steps of 0.2 and a pitch angle range of 0° to 30° in steps of 1°. This model does not require any work vectors, nor does it have any states. No configuration of work vectors or states is, therefore, required in the *mdlInitializeSizes* function. The *mdlInitializeConditions* function is configured to set the initial value of the output port to zero.

The *mdlOutputs* function can be represented by the flow diagram shown in Figure 3-12.

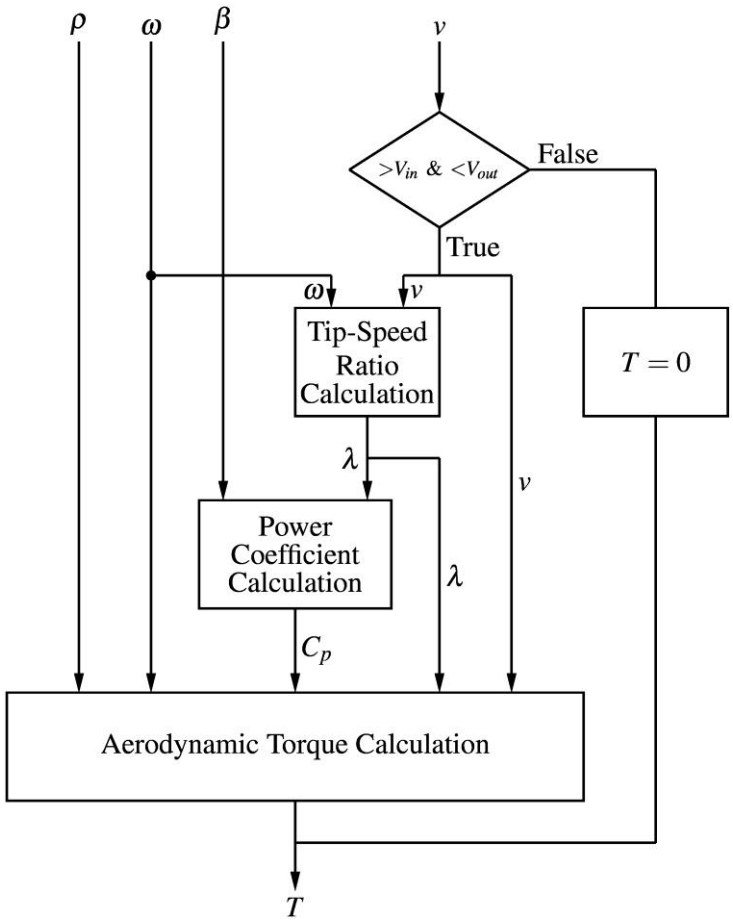


Figure 3-12: Flow diagram for *mdlOutputs* function of aerodynamic block.

As shown in Figure 3-12 the wind speed input is tested to determine whether it is larger than the cut-in wind speed and smaller than the cut-out wind speeds. If this is not the case, the torque output is set to zero; otherwise, the TSR is calculated using (3.6). The power

coefficient calculation block represents a two-dimensional look-up table for obtaining the power coefficient value from the C_p matrix parameter. The algorithm developed for the look-up table takes the TSR and blade pitch angle as input and uses linear interpolation to extract the required power coefficient from the C_p parameter. The S-function C-code implementation of the two-dimensional linear interpolating look-up table is given in Figure 3-13.

```
/* Setting the row value by rounding the Tip Speed
 * Ratio(TSR) to the closes 0.2m/s */
ry = floor(5*TSR);
/* Setting the column value by Rounding Blade pitch
 * (W(0)) down to closes degree */
kolom = floor(W(0));
/* Calculates vector index from the column and row values */
i = ry + 101*kolom;

/* Gets value of Power Coefficient at index i */
Cp1 = mxGetPr(PARAM4(S))[i];
/* Gets value of Power Coefficient at 1 index value higher
 * in TSR direction */
Cp2 = mxGetPr(PARAM4(S))[i+1];
/* Gets value of Power Coefficient at 1 index value higher
 * in Blade Pitch direction */
Cp11 = mxGetPr(PARAM4(S))[i+101];
/* Gets value of Power Coefficient at 1 index value higher
 * in Blade Pitch and TSR direction */
Cp22 = mxGetPr(PARAM4(S))[i+1+101];

/* Obtain interpolated CpTemp1 value between Cp1
 * and Cp2 with TSR */
CpTemp1 = (Cp2-Cp1)/0.2*(TSR-0.2*ry)+Cp1;
/* Obtain interpolated CpTemp1 value between Cp11
 * and Cp22 with TSR */
CpTemp2 = (Cp22-Cp11)/0.2*(TSR-0.2*ry)+Cp11;
/* Obtain final Power Coefficient value by interpolating
 * between CpTemp1 and CpTemp2 with Blade Pitch */
Cp = (CpTemp2-CpTemp1)*(W(0)-kolom)+CpTemp1;
```

Figure 3-13: S-function C-code implementation of power coefficient look-up table.

Finally, the torque output is then updated using (3.10). Since no work vectors are used in the S-function, no action needs to be performed by the *mdlTerminate* function.

3.4.3 Mechanical block

The S-function model of the mechanical block can be implemented from the block diagram shown in Figure 3-5 with input and output variables as defined in Table 3-4 and Table 3-5. The input ports, output ports and parameters of the mechanical model S-function are configured in the *mdlInitializeSize* function using Table 3-4 and Table 3-5. The S-function

model is configured for two output ports, namely turbine torque and generator torque, both with width one. Two input ports are configured, namely turbine angle velocity and generator angular velocity, both with width one and direct feed flag set. The six parameters required for the mechanical model S-function, i.e., generator moment of inertia J_{gen}, turbine moment of inertia J_{tur}, shaft stiffness coefficient K, shaft damping coefficient D, gear ratio GR and a vector with four elements containing the initial conditions of the states, are also configured here.

The *mdlCheckParameters* function is configured to verify that J_{gen}, J_{tur}, K, D and GR are all positive scalars and that the initiation conditions parameter is a four element vector, displaying an error message if any parameter is invalid.

From (3.15) and (3.16) the four states $x[0]$ to $x[3]$ are defined as

$$\begin{aligned} x[0] &= \theta_{gen} \\ x[1] &= \theta_{tur} \\ x[2] &= \omega_{gen} \\ x[3] &= \omega_{tur} \end{aligned}, \tag{3.62}$$

from which the differential equations of the mechanical model used in the *mdlDerivatives* can be obtained as

$$\begin{aligned} \frac{dx[0]}{dt} &= x[2] \\ \frac{dx[1]}{dt} &= x[3] \\ \frac{dx[2]}{dt} &= \frac{-D\left(x[2]-k_{gear}x[3]\right)-K\left(x[0]-k_{gear}x[1]\right)+k_{gear}^2\Gamma_{gen}}{J_{gen}k_{gear}^2} \\ \frac{dx[3]}{dt} &= \frac{-D\left(x[3]-\frac{x[2]}{k_{gear}}\right)-K\left(x[1]-\frac{x[0]}{k_{gear}}\right)-\Gamma_{tur}}{J_{tur}} \end{aligned}. \tag{3.63}$$

From this it is clear that four states are required to be configured in the *mdlInitializeSize* function. The *mdlInitializeConditions* function is used to assign the initial values of the states using the values of initial conditions parameter.

The *mdlDerivatives* function is configured to calculate (3.63) and the *mdlOutputs* is configured to assign state three, i.e., ω_{gen}, and state four, i.e., ω_{tur}, to the output ports. As with the aerodynamic model, no work vectors are used. As a result, no memory needs to be free in the *mdlTerminate*.

The C-code of the implemented gearbox model is provided in APPENDIX C.

3.4.4 Electrical block

3.4.4.1 Introduction

Both the ABC model and the DQ model of the DFIG can be represented by the block diagram shown in Figure 3-7 with input and output variables and parameters defined as in Table 3-6 and Table 3-7. From this it is clear that most of the configuration settings required in the *mdlInitializeSize* and *mdlCheckParameters* functions are consistent for both models. These configurations are discussed first and thereafter the remainder configurations are discussed separately for each model. Both the *mdlInitializeSize* functions require the configuration of two inputs, the applied voltage $\mathbf{V}_{abc}$ and mechanical angular velocity Ω_r of the generator. The width of the angular velocity is one and the width of the applied voltage input is six, to account for the three phase voltages of both the stator and the rotor. The direct feed configuration will be discussed separately. Two outputs are configured in both cases, one with width one for the generator torque and one with width six for the three phase currents of the stator and rotor. Both models require the configuration of seven parameters, namely stator resistance R_s, rotor resistance R_r, stator inductance L_s, rotor inductance L_r, magnetising inductance L_m, number of poles P and initial conditions. The *mdlCheckParameters* function is configured in both cases to validate that all parameters are scalars except the initial conditions parameter that is a vector with its size equal to the number of states of the respective models. The *mdlCheckParameters* function is further configured to validate the resistance and inductance parameters as positive values of type doubles, P as a positive value of type integer and the initial conditions as type double. If any of these validations fail, an error window is displayed with a specified error message. The *mdlInitializeConditions* functions of both models are configured to assign the values of the initial condition parameter to the initial values of the states.

Parts of the *mdlInitializeSize*, the *mdlDerivatives*, the *mdlOutputs* and the *mdlTerminate* functions require different configurations for each model. These configurations are discussed in the remainder of the section starting with the configurations required for the ABC DFIG model and followed by that of the DQ DFIG model.

3.4.4.2 ABC model

In section 3.2 it is shown that a DFIG can be modelled by (3.25) and (3.29), in the ABC reference frame, with the rotor and stator voltages $\mathbf{V}_{abc}$ and mechanical angle velocity Ω_r as inputs and rotor and stator currents $\mathbf{I}_{abc}$ and torque T as outputs. The explicit form of (3.25) is

$$\frac{d\,\mathbf{I}_{abc}}{dt} = \mathrm{A}\mathbf{I}_{abc} + \mathrm{B}\mathbf{V}_{abc} \tag{3.64}$$

with the currents as states and matrices A and B defined in APPENDIX B. Equation (3.64) has six states, namely the phase currents a, b and c of the rotor and stator. The electrical rotor position θ_r is required in (3.25) and (3.29). Therefore, the following additional differential equation needs to be solved to obtain θ_r from the mechanical angle velocity of the rotor Ω_r:

$$\frac{d\theta_r}{dt} = \frac{P}{2}\Omega_r\,. \tag{3.65}$$

Figure 3-14 shows a simple block diagram of the *mdlOutputs* and *mdlDerivatives* functions illustrating the inputs, outputs and state of the S-function. The *mdlDerivatives* function is configured to solve these seven differential equations with states *x*[0] to *x*[6], defined as $x[0] = I_{as}$, $x[1] = I_{bs}$, $x[2] = I_{cs}$, $x[3] = I_{ar}$, $x[4] = I_{br}$, $x[5] = I_{cr}$ and $x[6] = \theta_r$.

The *mdlOutputs* function is configured to calculate the torque from the states calculated in the *mdlDerivatives* function using

$$\begin{aligned} T[0] = -\frac{P}{2}M_{sr}\Big\{&\big(x[0]x[3]+x[1]x[4]+x[2]x[5]\big)\sin\big(x[6]\big)+\\ &\big(x[0]x[4]+x[1]x[5]+x[2]x[3]\big)\sin\big(x[6]+\tfrac{2\pi}{3}\big)+\\ &\big(x[0]x[5]+x[1]x[3]+x[2]x[4]\big)\sin\big(x[6]-\tfrac{2\pi}{3}\big)\Big\} \end{aligned}$$

and the output states one to six, i.e., *x*[0] to x[5], are assigned to the current output port (I[0] to I[5]).

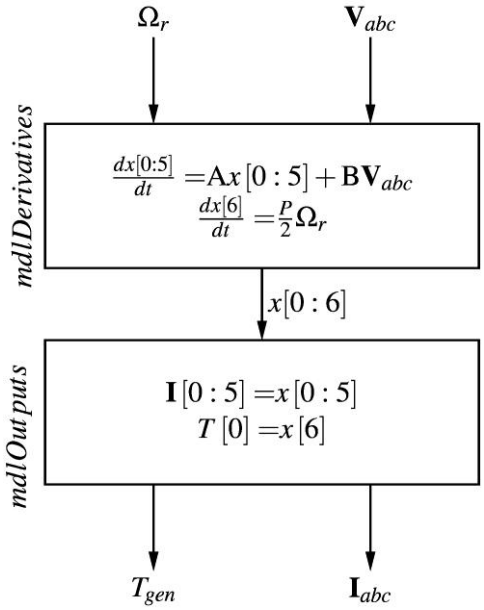


Figure 3-14: Block diagram of the *mdlDerivatives* and *mdlOutputs* functions of ABC DFIG S-function model.

From Figure 3-14 it is clear that the inputs are not used in the *mdlOutputs* function. Therefore, the feed through flags of the input ports can be set to zero in the *mdlInitializeSizes* function. No work vectors are required for this model, therefore no memory needs to be cleared in the *mdlTerminate* function.

3.4.4.3 DQ model

Figure 3-15 shows a block diagram of the *mdlOutputs* and *mdlDerivatives* functions of the S-function model of the DFIG using the DQ reference frame. Section 3.2 shows that the DFIG can be modelled in the DQ reference frame using (3.53) and (3.56) together with (3.57) and (3.58) to convert from the ABC-reference frame to the DQ-reference frame and vice versa. As for the ABC model, an extra differential equation given by (3.65) is required to calculate the electrical rotor angle position from the mechanical rotor angler velocity. The input voltage $\mathbf{V}_{abc}$ is converted to the DQ-reference frame voltages $\mathbf{V}_{dq}$ that is used with (3.53) and (3.65) to obtain the DQ-reference frame currents $\mathbf{I}_{dq}$. The values of $\mathbf{I}_{dq}$ are converted back to the ABC-reference frame to obtain the output currents $\mathbf{I}_{abc}$ using (3.56) and (3.58). $\mathbf{I}_{dq}$ is also used to calculate the output torque using (3.56). A work vector is used for the electrical angle θ_r to ensure that the angle used from transforming the ABC-reference frame to the DQ-reference frame is the same angle used for the transformation back to the ABC-reference frame.

From this, the *mdlInitializeSizes* function is configured for one work vector with dimension one as well as for five states *x*[0] to x[5], four for the currents as used by (3.53) and one for the rotor angle position as given by (3.65). The *mdlDerivatives* function is configured to solve the differential equations and the *mdlOutputs* function is configured to calculate the torque output and update the current output with the values of the states. Lastly, the *mdlTerminate* function is configured to free the memory used for the work vector.

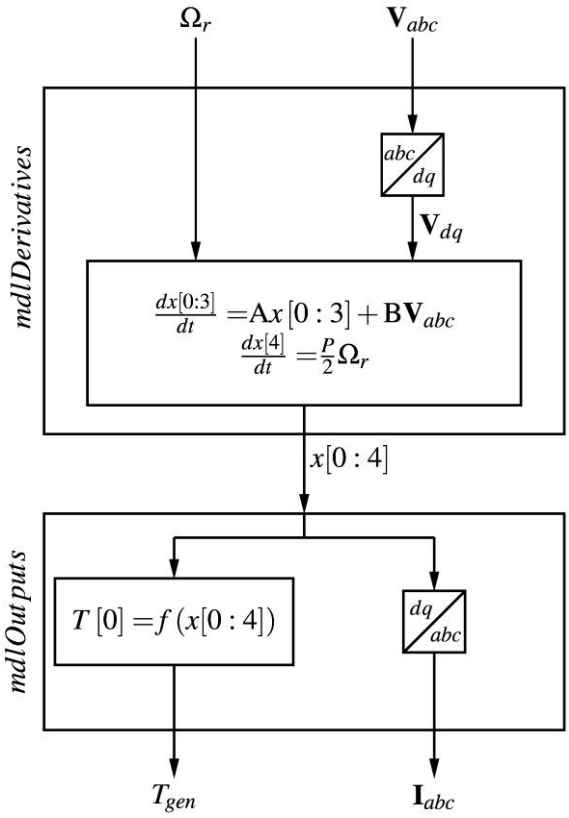


Figure 3-15: Block diagram of the *mdlDerivatives* and *mdlOutputs* functions of DQ DFIG S-function model.

3.4.5 Simulink library

This section provides an overview of the procedure for creating a Simulink toolbox for the S-function models developed in the previous sections. Simulink toolboxes are also referred to as libraries or block sets. The process of creating a Simulink library consists of two parts. Firstly, the S-function blocks need to be altered to be more user-friendly and secondly these blocks need to be put together to form a Simulink Library.

The models created in the previous section are not very user-friendly. For example, Figure 3-16 shows the block of the DFIG ABC model created when the MEX-file is assigned to the user-defined S-function block. The block displays two input ports and two output ports,

namely an input for the voltages, an input for the angular velocity, an output port for the currents and an output port for the torque. Unless the user of the block knows the input ports and output ports are arranged from top to button in order of first to last assigned in the C-code, the user will have trouble figuring out which input or output signal belongs to which port.

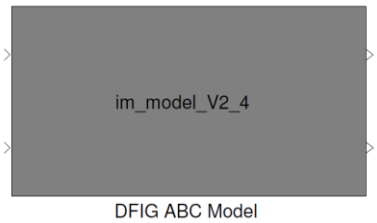


Figure 3-16: Unmasked DFIG ABC model S-function block.

Furthermore, opening the parameter window of the function block, shown in Figure 3-17, does not provide any assistance. The user needs to input the required parameter into the *S-function parameters* field in the parameter window. The parameter window does not provide any information on the required parameters. Therefore, the user needs to know the number of parameters required, type and dimension of each parameter, as well as the order of the parameters. The risk of breaking the link between the MEX-file and the S-function block also exists since the user can easily change the *S-function name* field. Users without knowledge of S-function coding might also try to edit the C-code file without knowing that the changes will have no effect on the operation of the model unless the C-code is recompiled to create a new MEX-file.

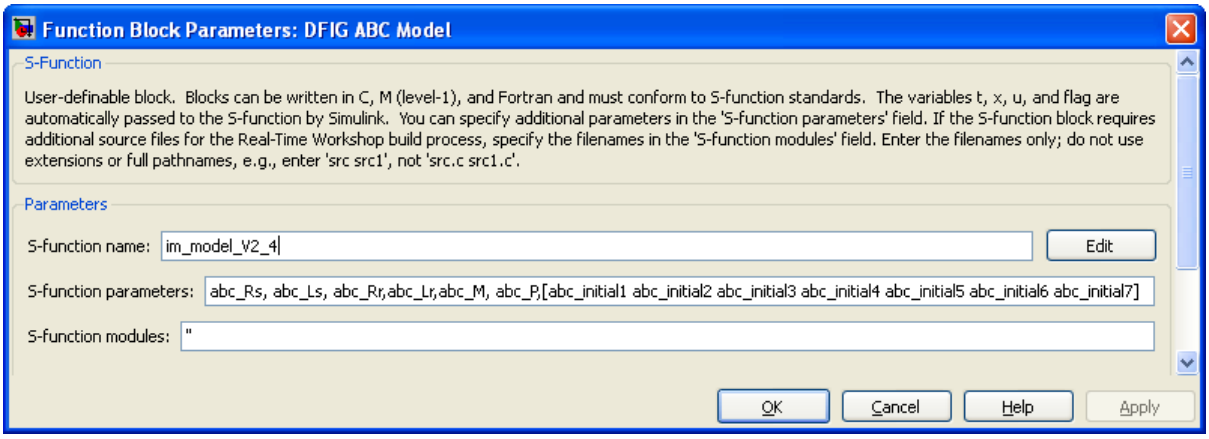


Figure 3-17: Parameter window of unmasked DFIG ABC model S-function block.

It follows that the S-function, as is, is not user-friendly, and the risk of setting up the model incorrectly is high. To overcome this problem, MATLAB has the functionality to create a

block that covers the S-function block with its own function block parameter window, known as a mask.

The mask allows its creator to manage the setting of the block displayed in the Simulink window as well as the parameter window. The DFIG ABC model will be used to illustrate the configuration of a mask and some of its functionality. Creating a mask for the model provides one with the Mask Editor shown in Figure 3-18 with four tabs, namely *Icon&Ports*, *Parameters*, *Initialization* and *Documentation*. The first tab, *Icon&Ports*, allows for labelling of the input and output ports as well as displaying text and/or graphics on the block. The code displayed in Figure 3-18 is that of the mask for Figure 3-16 resulting in the block shown in Figure 3-19. This clearly shows which input or output belongs to which port.

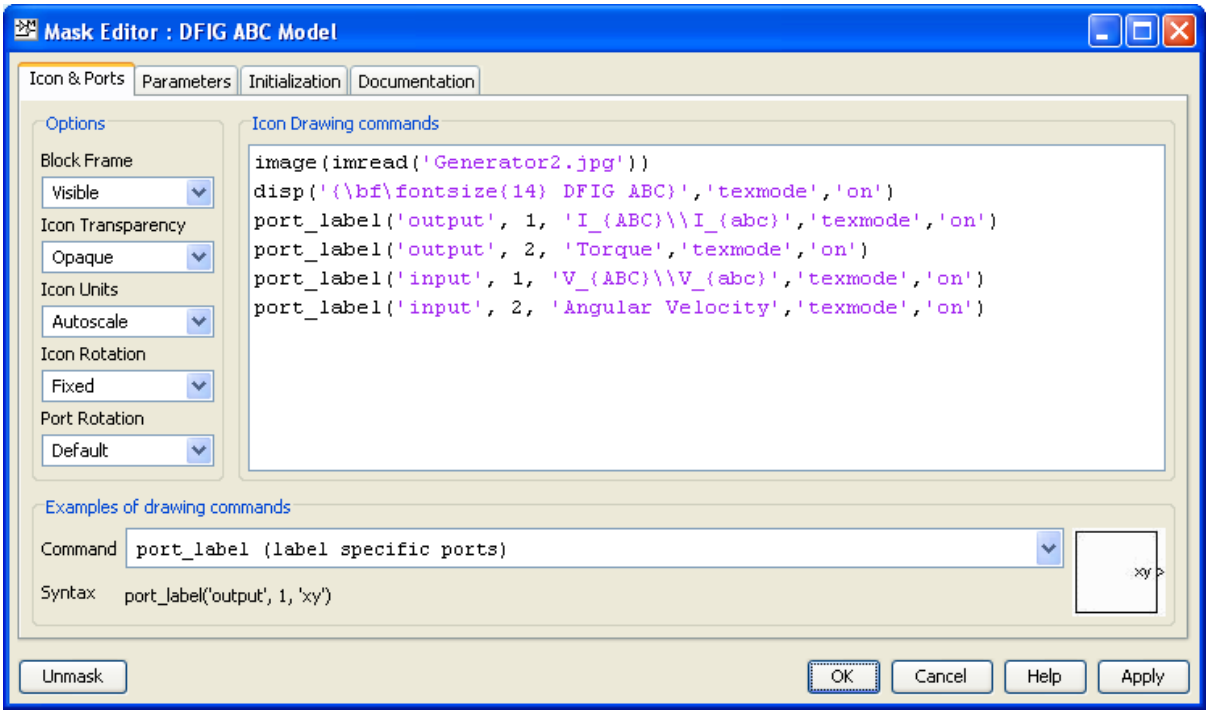


Figure 3-18: Mask Editor for DFIG ABC model – Input&Ports tab.

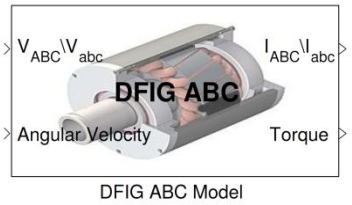


Figure 3-19: Masked DFIG ABC model S-function block.

The *Parameters* tab, shown in Figure 3-20, allows the creator of the mask to manipulate the *Function Block Parameter* window of the block. The variables in the *Variable* column refer to the variables in the S-function parameters field of the unmasked block, shown in Figure

3-17. The *Prompt* column allows for the full name of the parameter to be entered. This will be displayed where prompted for the parameter value. Information such as the unit or dimension of the parameter can also be added here. Using the *Tab name* column, different tabs can be created. In this example, two tabs are created, namely one for the generator parameters and one for the initial conditions as shown in Figure 3-21 and Figure 3-22. The *Initialization* tab makes provision for code that needs to be executed when the model is initialised. The *Documentation* tab allows for the description of the model to be added. This description is displayed at the top of the Function Block Parameters window. The help-text for the model is also entered in the *Documentation* tab.

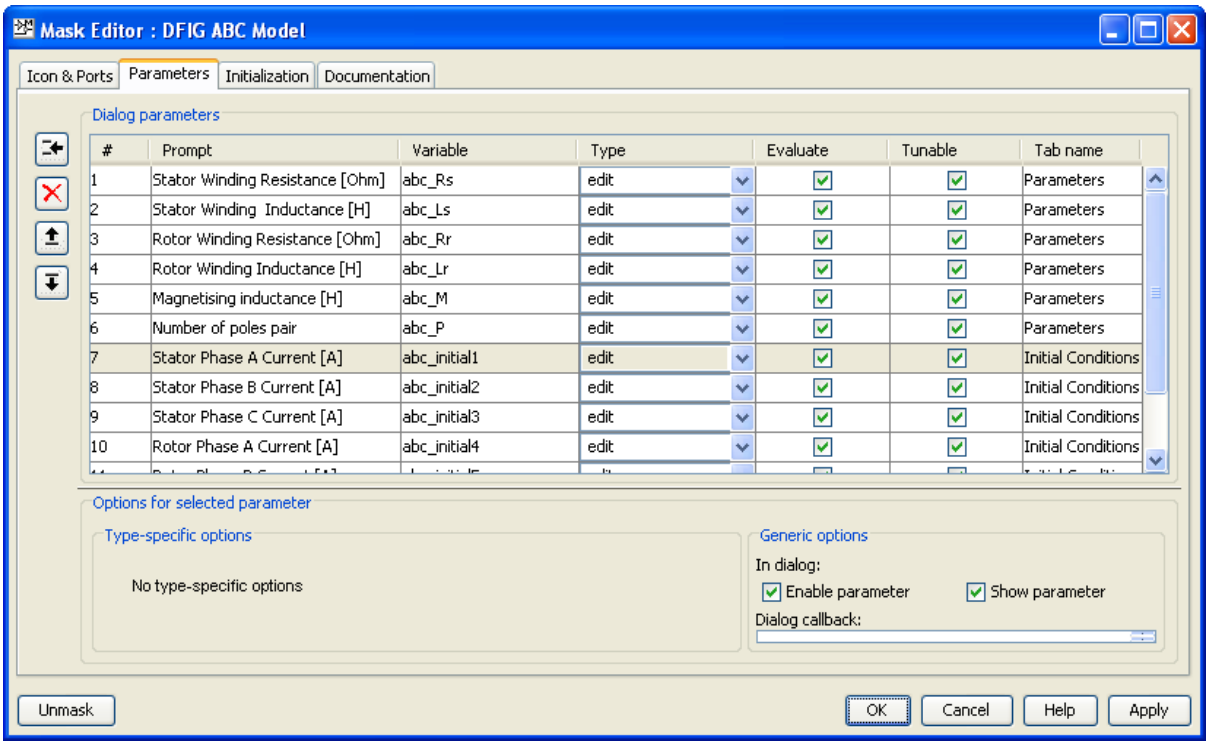


Figure 3-20: Mask Editor for DFIG ABC model – Parameters tab.

With the mask in place, the model is much more user friendly. However, if the user now wants to reuse the model, the user needs to find the model-file (.mdl) with the required model and copy it to a new project. By creating a Simulink Library file, the required model or models can be added to the Simulink Library Browser. This is done by creating a new library file from the Simulink window. In this study, there are three sets of models, namely Aerodynamics, Electrics and Mechanics. These subsets can be created by adding a Simulink Subsystem blocks to the library and placing the derived models within these subsystem blocks. The final step in creating the library is saving the library-file in a folder together

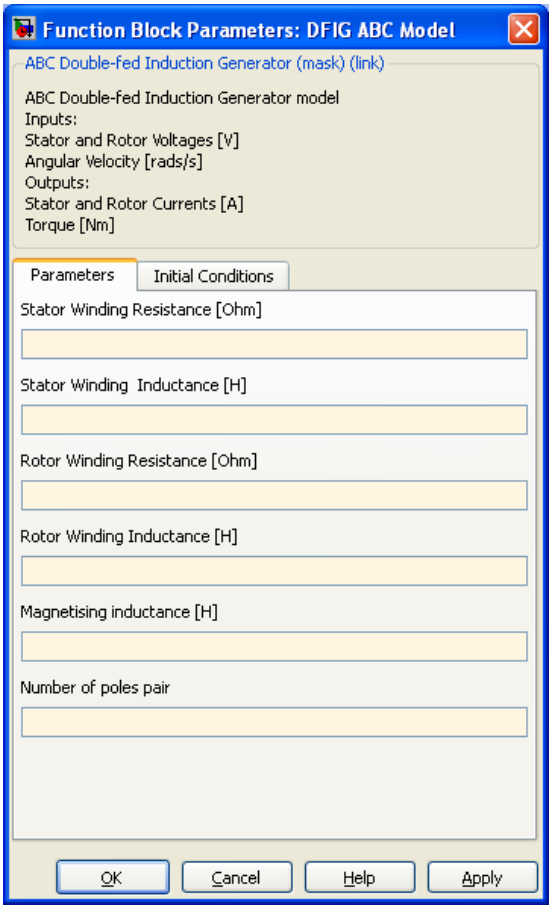


Figure 3-21: DFIG ABC model parameter dialog window – Parameters tab.

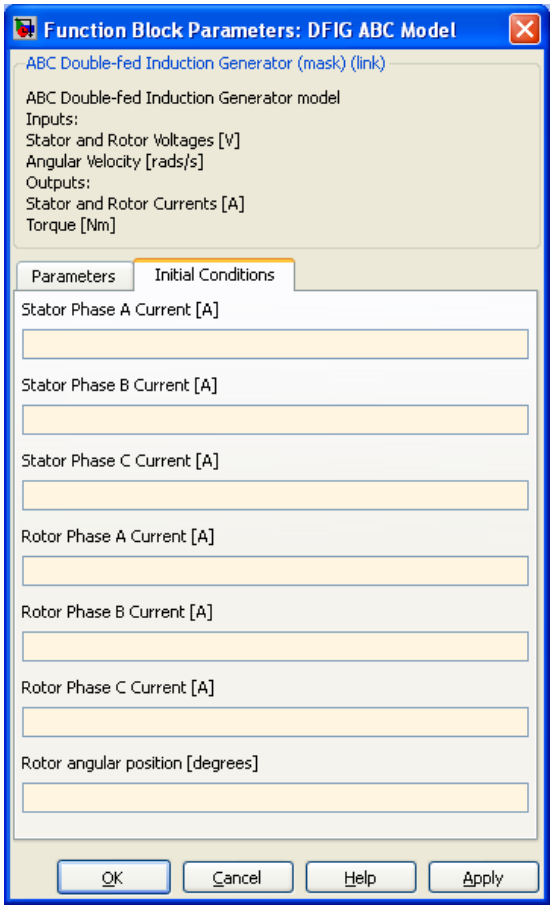


Figure 3-22: DFIG ABC model parameter dialog window – Initial Conditions tab.

with an edited copy of the slblocks.m file that can be found in the …\MATLAB\R200xx\toolbox\simulink\blocks folder. In the slblocks.m file the *Browser(1).Library* property needs to be changed to the filename of the newly created library-file and the *Browser(1).Name* property to the name that will be displayed in the Library Browser. All that remains to complete the process is to add the folder to the MATLAB path and reload the Simulink Library Browser. Figure 3-23 shows the added SUWindSystem Blockset in the Simulink Library Browser. The configurations of the masks for the remaining models are given in APPENDIX C.

.9

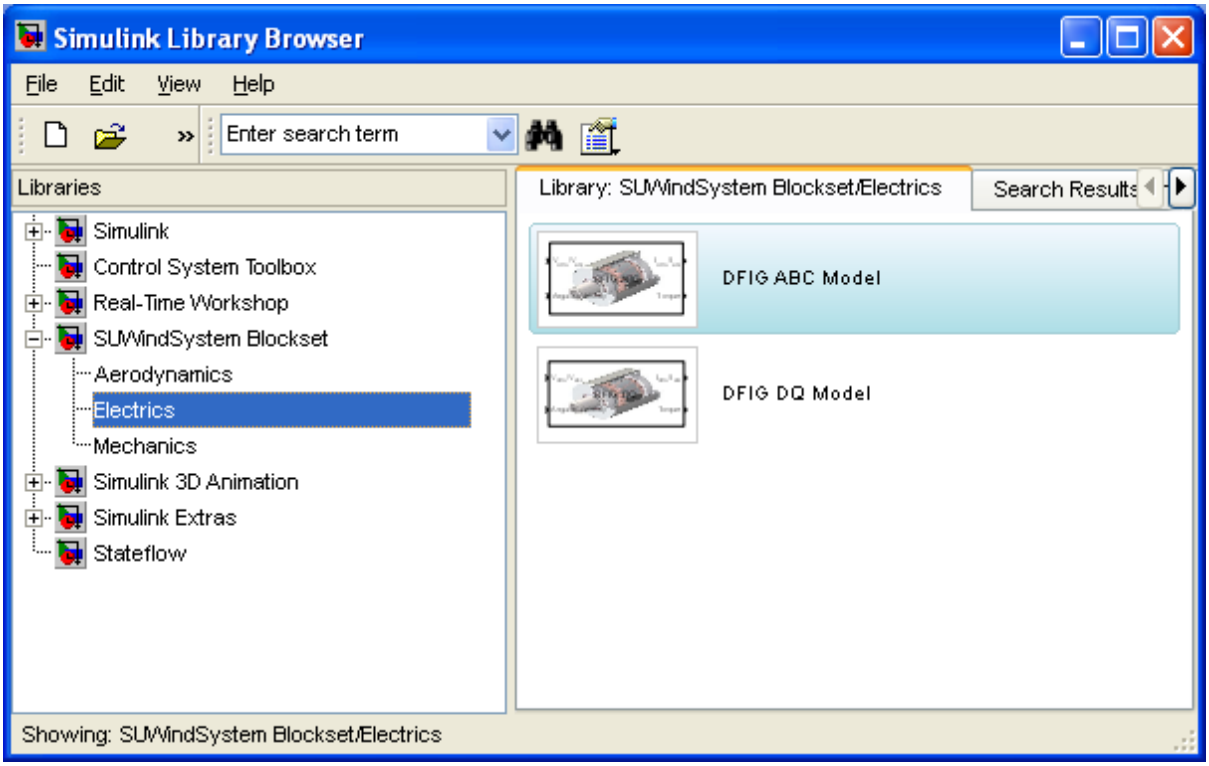


Figure 3-23: Simulink library browser with added SUWindSystem blockset.

.

4 VALIDATION AND PERFORMANCE EVALUATION OF SYSTEM COMPONENTS

4.1 Introduction

This chapter discusses the process of model validation and performance evaluation. The different developed models are validated by comparing their outputs to that of existing Simulink block models, developed by the Institute of Energy Technology at the University of Aalborg. These comparisons are evaluated by considering accuracy and simulation times. For the remainder of the discussion, the models developed by the Institute of Energy Technology are referred to as the IET models, whereas the models derived and implemented in the previous chapter are referred to as the Derived models. The different system block models are compared separately in sections 4.2 to 4.4 and the separate component models are connected to form a wind turbine system and compared in section 4.5.

Simulink uses the numerical solvers of MATLAB. There are several solvers available, including solvers for discreet systems, continuous systems, fix step solvers, variable step solvers as well as solvers for stiff systems. The definition of a stiff system is given in APPENDIX F together with an overview of the available continuous system solvers. The differential equations obtained from modelling electrical machines have a tendency to be stiff and their states are known to be continuous. MATLAB recommended the use of one of the following variable step size solvers for these types of problems: ode15s, ode23s, ode23t or ode23tb. Through experiments it was found that the ode23s solver produces the best results for the models. This solver is used to obtain the results reported on in this chapter as well as those in the following chapters.

4.2 Aerodynamic block

This section compares the accuracy and efficiency of the Derived S-function turbine blade model to the corresponding IET Simulink block model.

Figure 4-1 shows the altered IET Simulink block model used to validate the derived turbine blade model. The original IET model makes use of the torque coefficient C_q, but since the Derived model makes use of the power coefficient, the IET model is altered to use the power coefficient as well. Since the torque coefficient is equal to the power coefficient divided by

the Tip Speed Ratio (TSR), the only change required is the addition of a divider block (Divide) to the existing model.

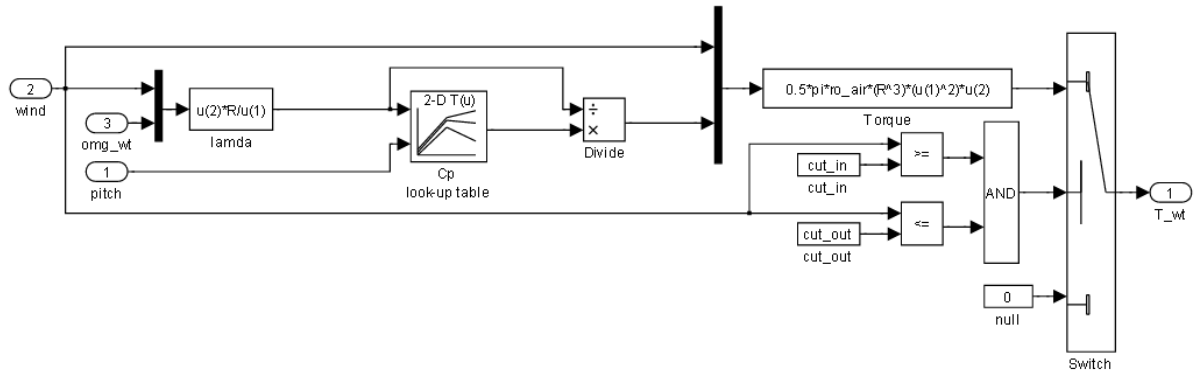


Figure 4-1: IET turbine blade block model.

The first step in validating the Derived model is done by comparing the torque output to that of the IET model for wind speed inputs starting at 0 m/s and increasing linearly to 30 m/s. The blade pitch, hub speed and air density inputs are kept constant at 0°, 1.5 m/s and 1 kg/m^3 respectively. Figure 4-2 shows a graphical representation of the power coefficient parameter for selected blade pitch angles. These values are obtained from an equation approximating the power coefficient by the nonlinear function given in APPENDIX D [77]. The remaining parameter values are tabulated in Table 4-1.

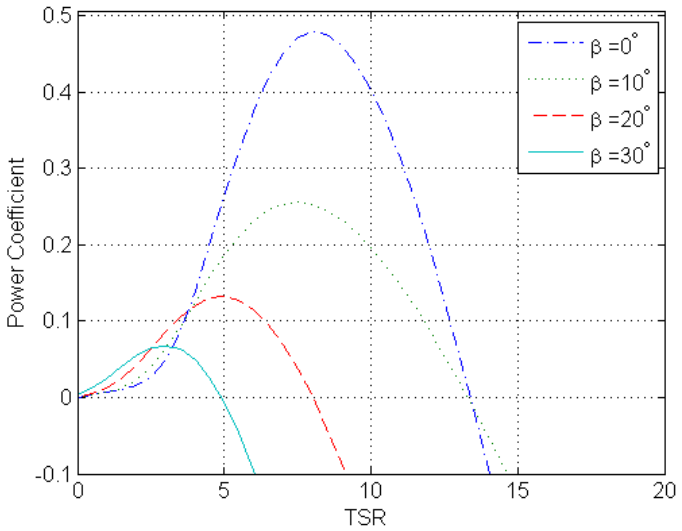


Figure 4-2: Graph of power coefficient values versus Tip Speed Ratios (TSR) for different pitch angles (β) [77]**.**

Table 4-1: Turbine blade parameter values.

Parameter	Description	Value	Parameter	Description	Value
R	Blade length	50 m	v_{out}	Cut-out Wind Speed	20 m/s
v_{in}	Cut-in Wind Speed	6 m/s			

Simulating both models and plotting the output torque results versus wind speed produces the plot given in Figure 4-3. Due to the IET model not performing a check to verify that the inputs to the loop-up table block are within its boundaries, an error is produced for wind speed values smaller than 3.75 m/s. The data points of the IET model, therefore, only start at 3.75 m/s. Figure 4-3 shows that both the cut-in and cut-out wind speed parameters of the derived model are implemented correctly. The model produces 0 Nm torque for wind speed values not contained within the cut-in and cut-out wind speed range.

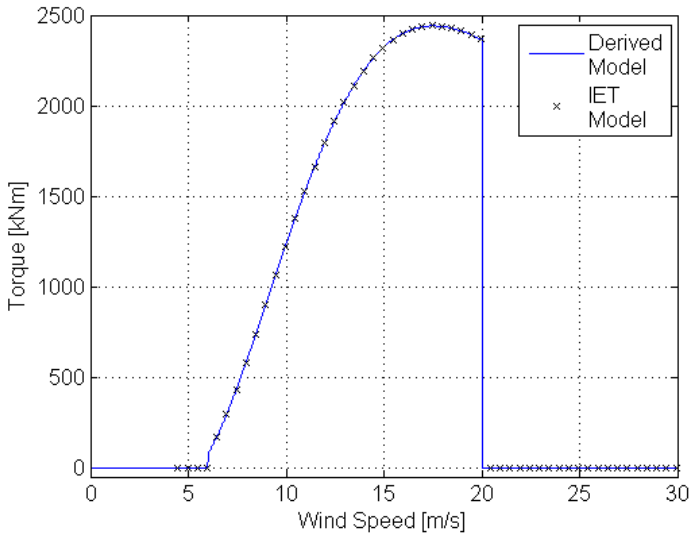


Figure 4-3: Plot comparing the torque versus wind speed of derived and IET model.

The verification process is continued by applying a constant wind speed of 15 m/s and varying the blade pitch linearly from 0° to 30°. The remaining inputs and parameters are kept the same. The plot of the simulated torque outputs are displayed in Figure 4-4. This result confirms that the derived model produces the same output as the IET model, thereby verifying that the developed linear interpolating look-up algorithm produces the same results as the Simulink look-up table block.

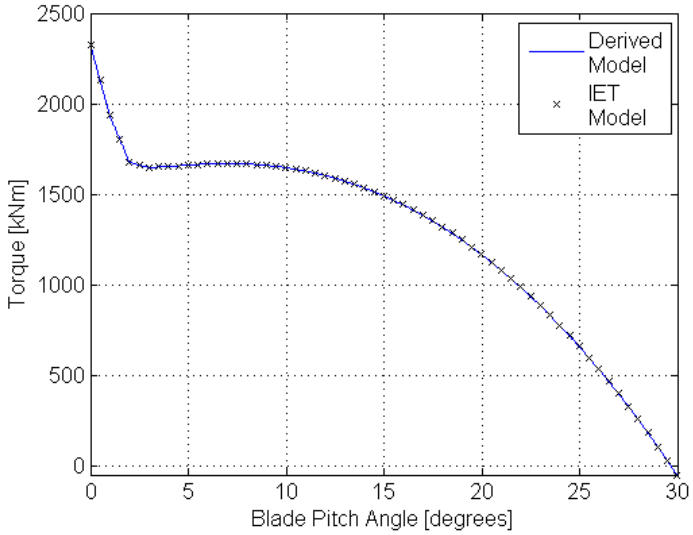


Figure 4-4: Plot comparing the torque versus pitch angle of derived and IET model.

For the last data set used to validate the Derived turbine blade model, the blade pitch angle is fixed to 0° and real wind data is used as wind speed input. The remaining inputs and parameters are kept the same as for the previous experiments. The real wind data is obtained from the Gorgonio wind measurement site in the USA. Figure 4-5 shows the recorded data with a 5 Hz sampling frequency for a portion of a day [78]. The simulated output torque of the derived model is an exact match to that of the IET model, verifying the model accuracy. Figure 4-6 shows the 9:50 to 9:52 segment of the input wind data (top) with the simulated outputs of both models (bottom).

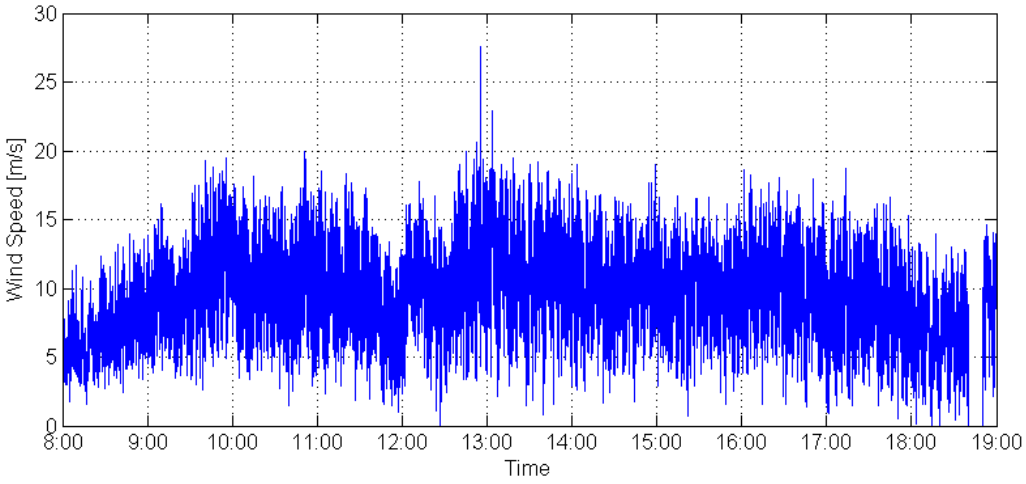


Figure 4-5: Real wind data from the Gorgonio wind measurement site in the USA [78]**.**

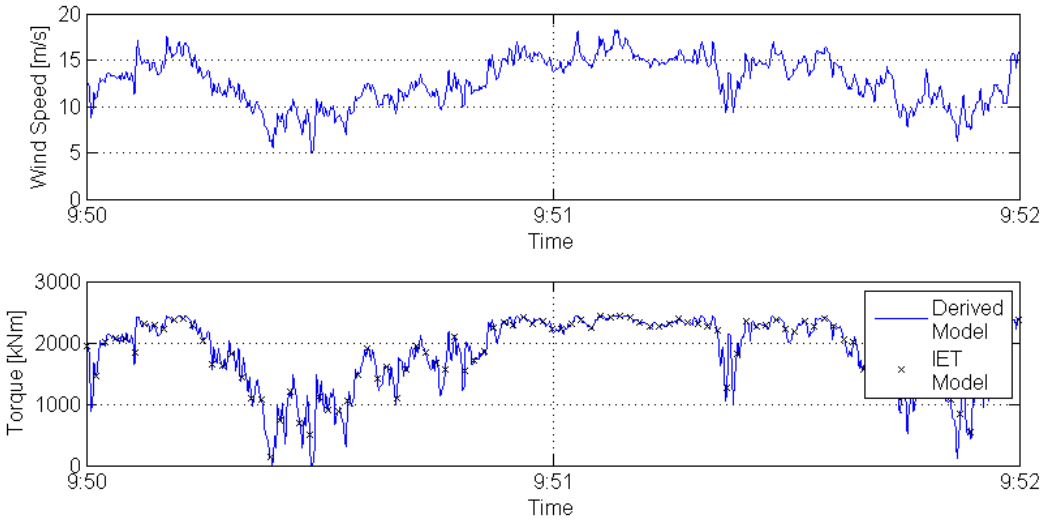


Figure 4-6: Two minute window of wind data shown in Figure 4-5.

The simulation times are recorded to compare the performance of the models with regards to simulation times. The models are simulated with the wind input set to a constant value (12 m/s) as well as wind data from Gorgonio wind site. The blade pitch angle is simulated for a constant value of 0° and for a sinusoidal input with an average of 15°, amplitude of 15 and frequency of 0.25 Hz. All of these simulations are run with the parameters shown in Table 4-1 and Figure 4-2. The Air Density and Hub speed inputs are kept constant at 1 kg/m^3 and 1.5 m/s respectively. Table 4-2 presents the resulting mean and standard deviation of the simulation times recorded for 100 simulations of both models for each of the different combinations of the wind speed and blade pitch angle input. The Ratio column provides the ratio of the mean simulation time of the IET to the mean simulation time of the Derived model. The results presented in this section give rise to the conclusion that the Derived turbine blade model is accurate and that it shows a reduction in simulation time by a factor of approximately three.

Table 4-2: Simulation time results of IET and derived turbine blade model.

			IET model Simulation Time		Derived model Simulation Time		
Wind Speed	Blade Pitch Angle	Number of Simulations	Mean [s]	Std Dev [s]	Mean [S]	Std Dev [s]	Ratio
Real Data	Constant	100	9.625	0.069	3.348	0.040	2.875
Real Data	Sinusoidal	100	9.613	0.063	3.337	0.036	2.880
Constant	Constant	100	9.606	0.071	3.341	0.031	2.875
Constant	Sinusoidal	100	9.615	0.070	3.344	0.030	2.875

4.3 **Mechanical block**

This section compares the accuracy and efficiency of the Derived gearbox model to the corresponding IET Simulink block model.

Figure 4-7 shows the Simulink block model of the IET gearbox model used for comparison with the Derived gearbox model. Input one is the torque generated by the turbine blade and input two the torque generated by the generator. Output 1 is the angular velocity of the generator and output 2 is the angular velocity of the turbine blades.

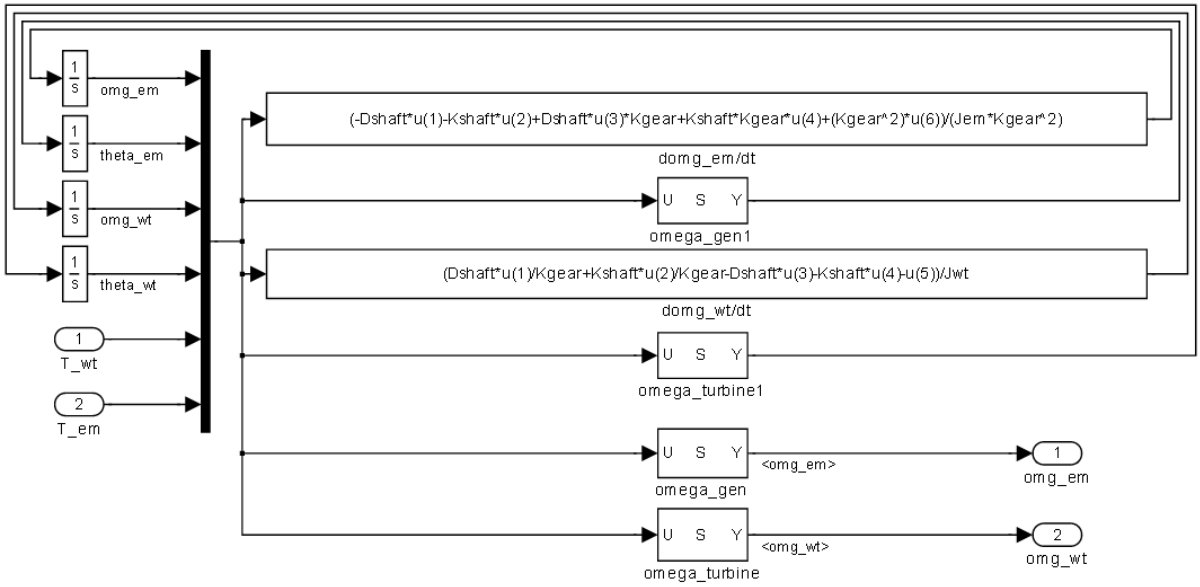


Figure 4-7: IET gearbox Simulink block model.

To compare the dynamic behaviour of the implemented S-function gearbox model, a multi-input-multi-output (MIMO) bode plot is generated and compared to the bode plot of the IET Simulink block model. These bode plots are generated using MATLAB's bode function with the parameter values provided in Table 4-3. The results are displayed in Figure 4-8 showing that the frequency response of the Derived model, i.e., the solid line, is the same as that of the IET model, i.e., the $\times$ markers.

Table 4-3: Parameter definitions of mechanical model.

Parameter	Description	Value	Parameter	Description	Value
J_{gen}	Generator Moment of Inertia	90 kg.m^2	D	Shaft Damping coefficient	750 kNm.s/rad
J_{tur}	Turbine Moment of Inertia	4.95 Mkg.m^2	GR	Gear Ratio	83
K	Shaft Stiffness coefficient	114 MNm/rad			

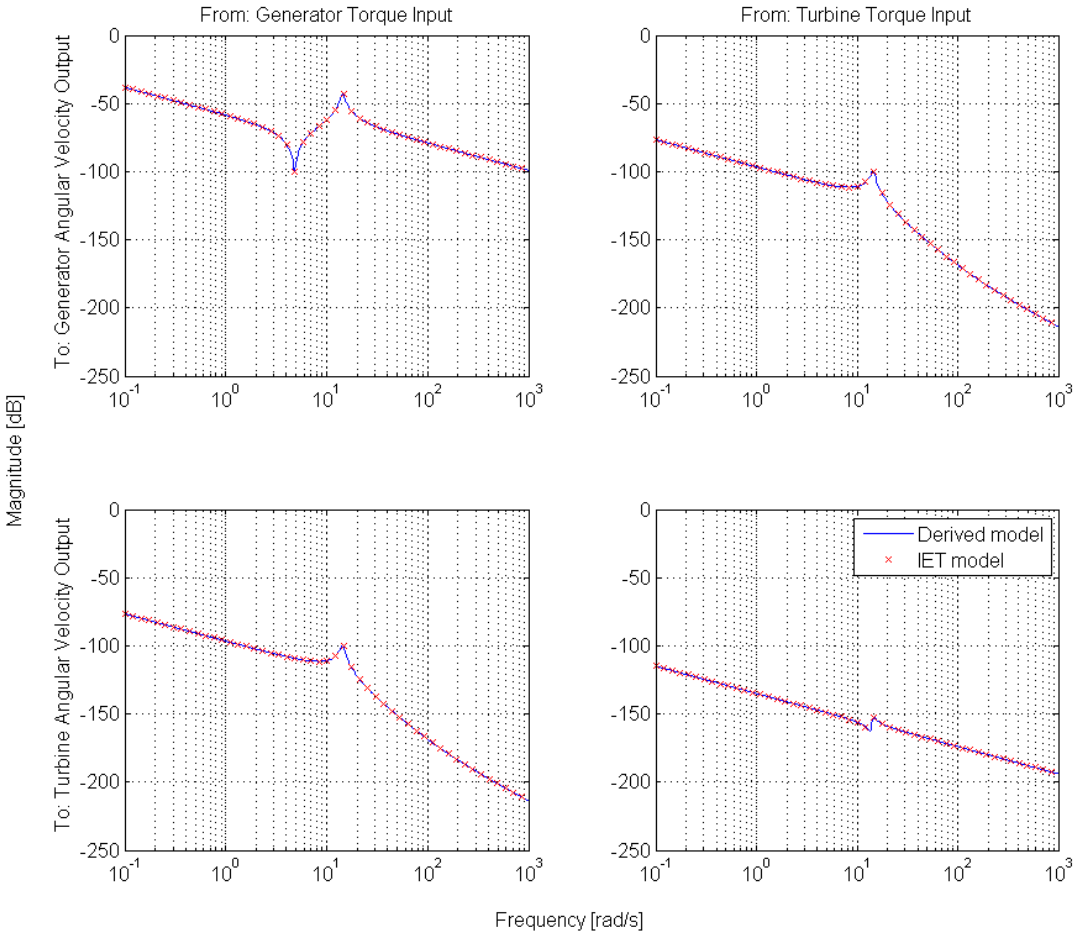


Figure 4-8: Bode plot comparison of MIMO Gearbox systems.

The model is further validated by applying the turbine blade torque and generator torque, shown in Figure 4-9, to both models and comparing the simulated outputs. Figure 4-10 displays the outputs of both models. It is clear that the outputs correlate well, verifying that the model is accurate.

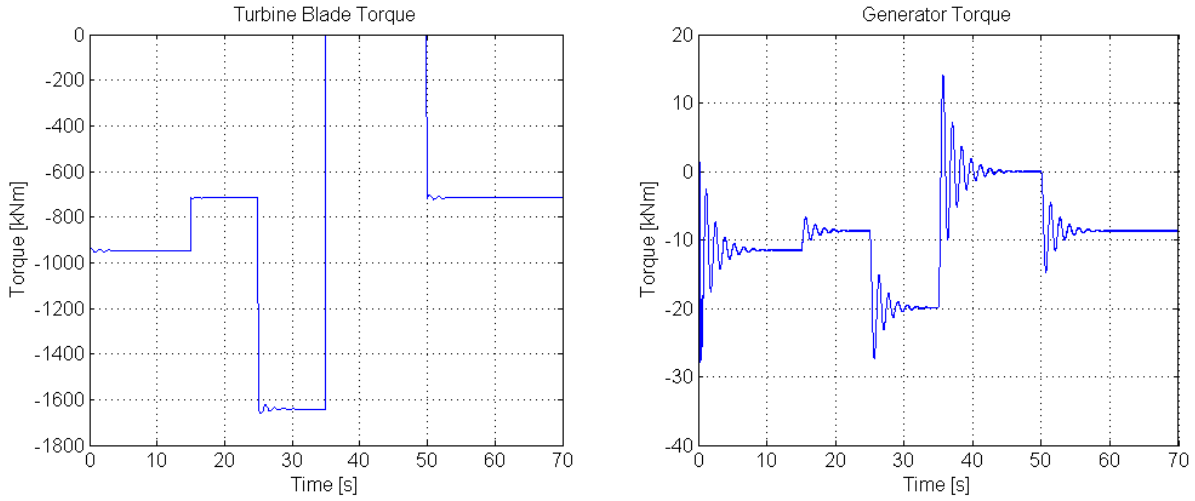


Figure 4-9: Torque inputs for comparison of the gearbox models.

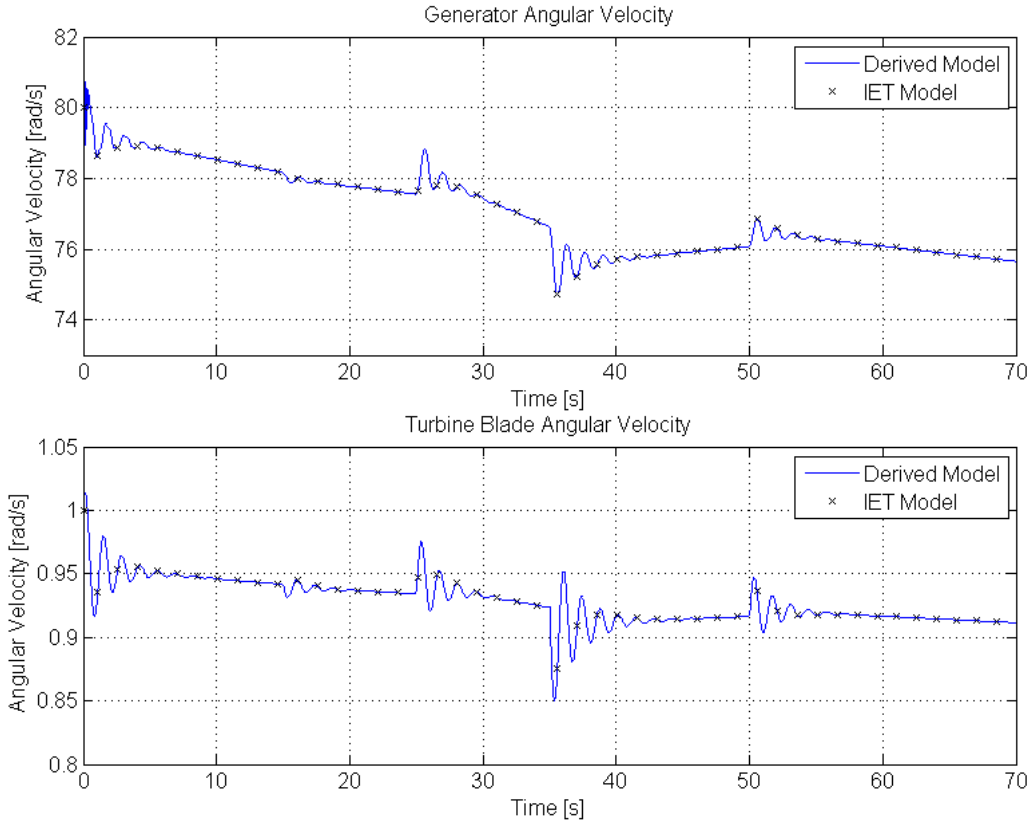


Figure 4-10: Angular velocity output results for Derived and IET gearbox model comparison.

The recorded simulation times are once again used to compare the Derived model to that of the IET model. These results show a reduction in simulation time varying between 30% and 40%.

From these results it is concluded that the Derived model is accurate. Although the model works correctly, there is no radical improvement in the simulation time. This may be

explained by the fact that Simulink's block models are very well optimised for the solving of differential equations and that the model of the two-mass gearbox simply consists of two differential equations.

4.4 Electrical block

This section compares the accuracy and efficiency of the Derived ABC and DQ reference frame DFIG models to the corresponding IET Simulink block models.

In order to verify that the generator model was implemented correctly, the implemented S-function DFIG models with 4 pole configuration are simulated for rotor speeds ranging from 0 to 3000 rpm and compared to the values generated by the corresponding IET Simulink model. The models are operated as induction generators with applied voltages as shown in Table 4-4. Table 4-5 summarises the parameter values used for the simulations. The torque-speed curve simulation results for the S-function model and the IET model are shown in Figure 4-11 for the ABC reference framework and in Figure 4-12 for the DQ reference framework. In both cases excellent correlations are achieved. Both figures also show that the torque goes to zero at 1500 rpm as expected for a 4 pole induction machine.

Table 4-4: Input values electrical model.

Variable	Description	Value	Variable	Description	Value
$\mathbf{V}_{ABC}$	Stator Voltage	3-phase 600 V 50 HZ	$\mathbf{V}_{abc}$	Rotor Voltage	0 V

Table 4-5: Parameter definitions of electrical model.

Parameter	Description	Value	Parameter	Description	Value
R_s	Stator resistance	115 mΩ	L_r	Rotor inductance	1.7 mH
R_r	Rotor resistance	184 mΩ	L_m	Magnetising inductance	46.6 mH
L_s	Stator inductance	1.7 mH			

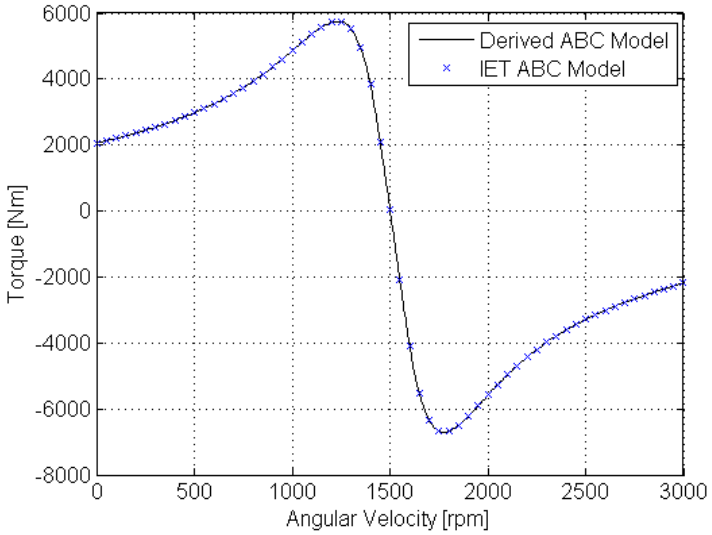


Figure 4-11: Torque versus angular velocity comparison of the ABC models.

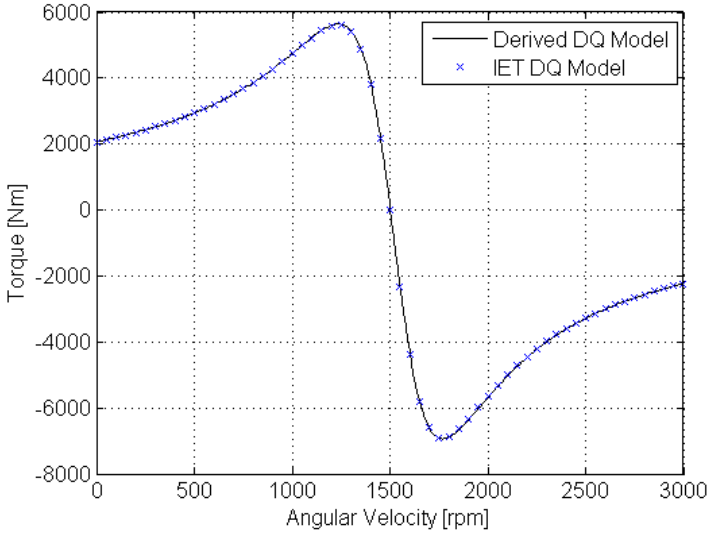


Figure 4-12: Torque versus angular velocity comparison of the DQ models.

To evaluate the efficiency of the S-function models, the simulation times of both models are recorded for each simulation run to obtain the torque curves. The simulation times of each model are plotted as a function of the angular velocity, as shown Figure 4-13.

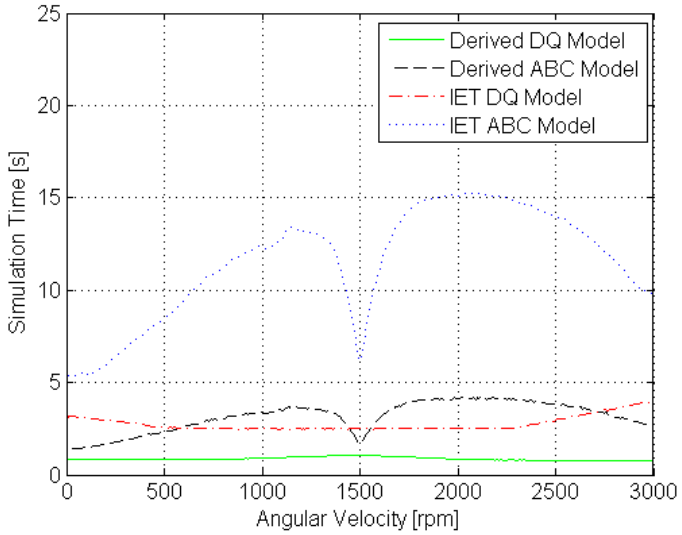


Figure 4-13: Simulation time comparison of the ABC and DQ models.

It is clear from the figure that the Derived models show a considerable reduction in simulation time compared to the IET model. To compare the simulation times, they are averaged, giving 0.870 s, 3.150 s, 2.750 s and 11.518 s for the Derived DQ model, Derived ABC mode, IET DQ model and IET ABC model respectively. In the case of the ABC models there is an average reduction of about 72% between the Derived model and IET model, calculated using

$$\text{Percentage reduction} = \frac{\bar{t}_{IET} - \bar{t}_{Derived}}{\bar{t}_{IET}} \tag{4.1}$$

where

$\bar{t}_{Derived}$ denotes the average simulation time of the Derived model and

$\bar{t}_{IET}$ denotes the average simulation time of the IET model.

In the same way the reduction in simulation time between the Derived DQ model and the IET DQ model can be obtained as about 68% and reduction between the Derived DQ model and the IET ABC model as about 92%. All of these are significant reductions, especially considering that the model is simulated numerous times during a parameter estimation routine.

The results obtained from these simulations show that both the ABC and DQ Derived models produce matching output signals for identical input signals when compared to the existing IET models and that there is a significant reduction in simulation time for both models.

4.5 Wind turbine system

This section compares the accuracy and efficiency of the Derived wind turbine system model, constructed of the different Derived component models, to the corresponding IET Simulink block models.

The models of the different components are connected together, as shown in Figure 4-14, to form the wind turbine system model where the generator component can either be the ABC DFIG model or the DQ DFIG model.

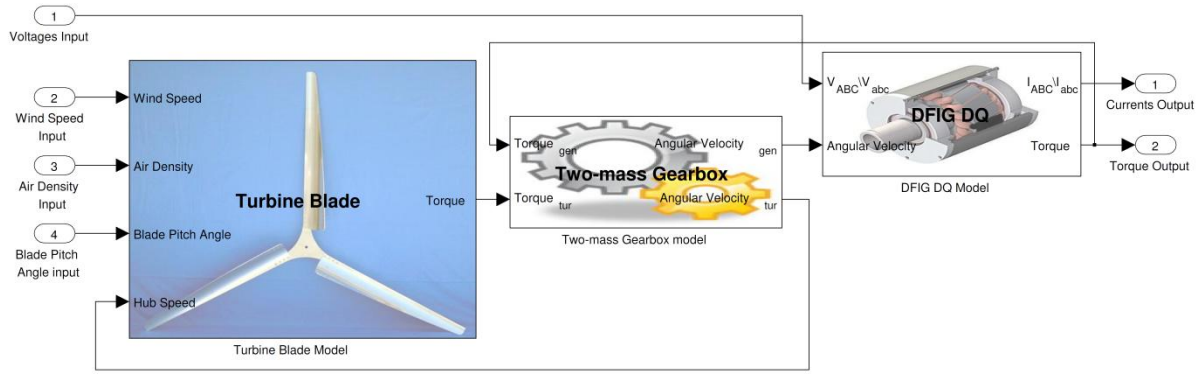


Figure 4-14: Wind turbine system.

The DFIG is again simulated as an IG with zero voltage supplied to the rotor and a 600 V 50 H supply to the stator. The first experiment conducted uses the signal shown in Figure 4-15 as input wind speed and constant air density and blade pitch angle of 1 kg/m^3 and 5° respectively. These inputs, together with the parameters listed in APPENDIX E, are used to compare the Derived wind turbine system to the IET wind turbine. The systems are compared both for the topology with the ABC DFIG as generation element as well as for the topology with the DQ DFIG as generation element.

The MATLAB solver's settings were initially all set to automatic, but this led to two problems. First, in the case of the IET DQ DFIG, the signals became distorted. Figure 4-16 shows a zoomed in part of the stator currents that is supposed to be three sinusoidal signals with equal peak values and each phase shifted 120°. From this figure it is clear that the sampling tempo is too low to capture the system dynamics. According to Nyquist criteria, the sampling tempo should be at least double the highest frequency of the system dynamics. Since the sampling tempo is inversely proportional to step size, this problem can be overcome by setting the maximum step size, thereby forcing the sampling tempo to be higher than the specified value.

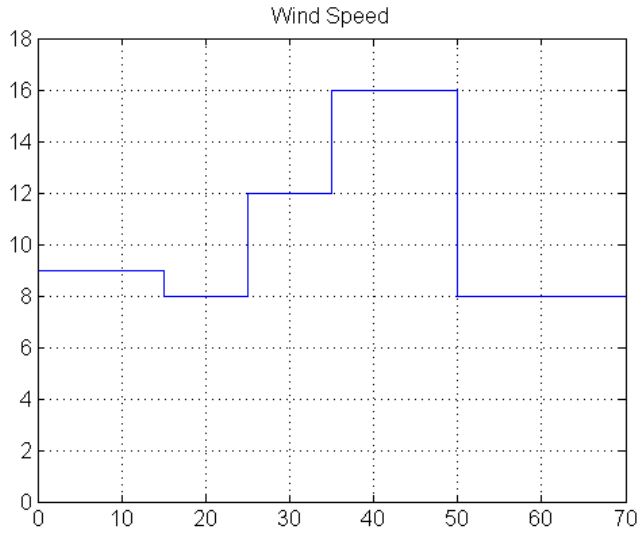


Figure 4-15: Step wind speed input for wind turbine system.

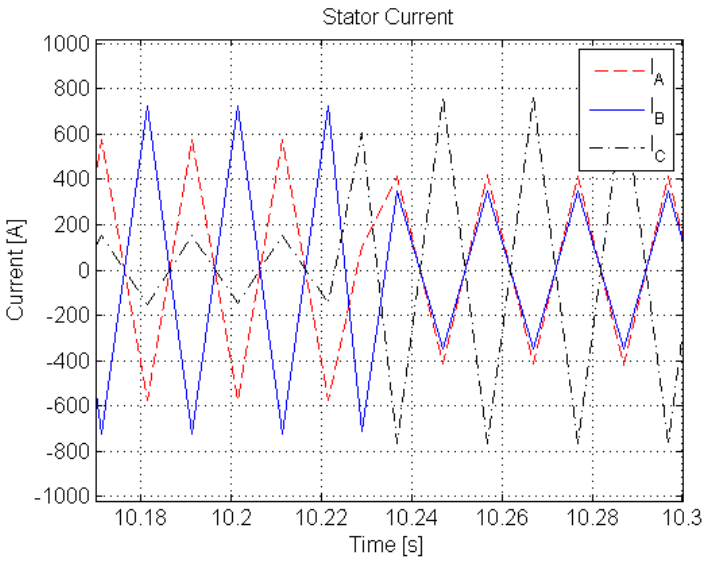


Figure 4-16: Zoomed in portion of IET DQ DFIG simulated stator current with auto maximum step size.

The second problem was encountered with the system using the Derived DQ DFIG model as the generating element. Looking at the generator torque curve Figure 4-17, the curve shows the expected dynamic behaviour, but on closer inspection it is found that there is an unexpected high frequency oscillation on the torque signal, as shown in Figure 4-18.

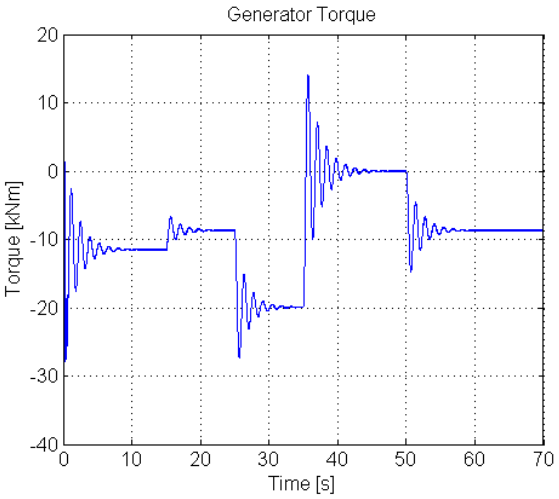


Figure 4-17: Derived DQ DFIG torque output for step sizes set to automatic.

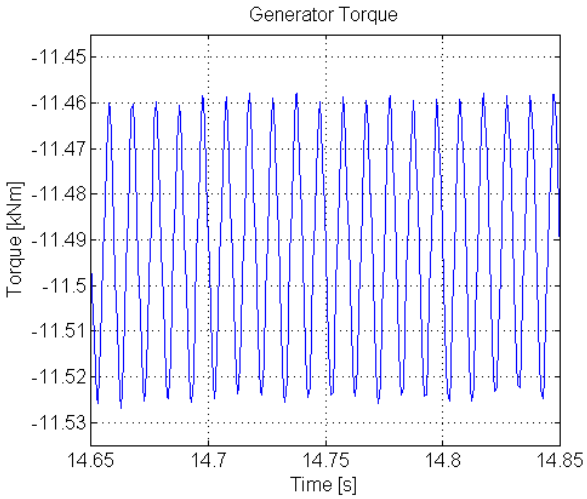


Figure 4-18: Zoomed in portion of generator torque in Figure 4-17.

After checking the model and performing experiments with different simulation configurations, it was found that by reducing the step size sufficiently, the oscillation no longer occurs. It is, therefore, concluded that the oscillation is caused by a numerical instability, caused either by the solver, model or a combination of the two. This requires further investigation. For this study the step size was set small enough to overcome the oscillation problem. It was found that a maximum step size of 1 ms is sufficient to overcome the sampling problem, but an even smaller maximum step size of 0.5 ms is required to overcome the oscillation problem.

To evaluate the accuracy and efficiency of the Derived wind turbine system models, simulations are run to compare the output while the simulation times are recorded. The systems are first simulated with their maximum step size set to 1 ms for both generated wind data and real wind data as wind speed input. The generated wind data signal used is shown in Figure 4-15 and the real wind data signals used is a section of the signal shown in Figure 4-5. These simulations were performed again with the step size configured to 0.5 ms.

The resulting outputs of both Derived system models correlate well with that of the corresponding IET system models. Except for the simulation performed on the system with DQ DFIG as generating element and the maximum step size set to 1 ms, the slow and steady state response still correlates well but, the torque output of the Derived model has the added high frequency oscillation. APPENDIX G presents the resulting output signals obtained at different points in the wind turbine system for simulations performed with both the generated

wind input signal and the real wind input signal on the Derived wind turbine system model with DQ DFIG as generating element.

The recorded simulation times for the different configurations are shown in Table 4-6. The differences in simulation times between the 1 ms and 0.5 ms maximum step sizes in the Derived DQ DFIG row are found to be relatively small compared to the IET DQ DFIG row. The smaller difference in simulation time may be a result of the oscillation on the output of the 1 ms maximum step size simulations. This is believed to be caused by a numerical instability that results in these simulations having longer simulation times. The possibility, therefore, exists that the simulation time of the Derived DQ model for the 1 ms step size can be reduced by solving the numerical instability problem.

Table 4-6: Simulation times of wind turbine system models.

	Simulation Time [s]			
Wind Speed Input Data:	**Generated Data**		**Real Data**	
Maximum Step Size [ms]:	**1**	**0.5**	**1**	**0.5**
Electrical Block:				
Derived DQ DFIG	25.3	33.3	36.8	48.5
IET DQ DFIG	35.5	68.2	51.7	101.7
Derived ABC DFIG	42.1	45.1	82.3	83.8
IET ABC DFIG	100.1	112.7	210.8	207.7

Using (4.1) with the data in Table 4-6, the percentage reduction in simulation time between the different models can be calculated. Table 4-7 shows the percentage reduction for the simulations using the generated wind signal as wind speed input. Table 4-8 shows the percentage reduction for the simulations performed using the real wind signal as wind speed input.

Table 4-7: Percentage reduction in simulation time for generated wind data.

	Derived ABC DFIG		**Derived DQ DFIG**	
Maximum Step Size [ms]:	**1**	**0.5**	**1**	**0.5**
IET ABC DFIG	58%	60%	75%	70%
IET DQ DFIG	-	-	29%	51%

Table 4-8: Percentage reduction in simulation time for real wind data.

	Derived ABC DFIG		**Derived DQ DFIG**	
Maximum Step Size [ms]:	**1**	**0.5**	**1**	**0.5**
IET ABC DFIG	61%	60%	83%	77%
IET DQ DFIG	-	-	29%	52%

These tables show that the percentage reduction in simulation time between the Derived and IET wind turbine system models with the ABC DFIG model as generating element is about 60%, for both the generated and real data cases as well as for both the maximum step size configurations. The percentage reduction in simulation time between the Derived wind turbine system model with the DQ DFIG model as generating element and the IET wind turbine system model with the ABC DFIG model as generating element is in the range of 70% to 83% for all cases. The percentage reduction in simulation time between the Derived and IET wind turbine system models with the DQ DFIG model as electrical block is about 50% for both wind input signals with the maximum step size set to 0.5 ms and only 29% for both wind input signals with the maximum step size set to 1 ms. As mentioned earlier, this reduction could possibly be improved by solving the numerical instability problem.

The results presented in this section give rise to the conclusion that the wind turbine systems comprised of the Derived models are accurate and that these systems show an overall reduction in simulation time.

5 ESTIMATION OF PARAMETERS OF SYSTEM COMPONENTS

5.1 Overview

This chapter starts with a discussion of the general configuration required for using MATLAB to perform parameter estimation on Simulink models. This is followed by a section presenting the results obtained from the parameter estimation case studies performed on the four individual models, i.e., the turbine blade model, the gearbox model, the ABC DFIG model and the DQ DFIG model. The chapter concludes with a section presenting the results obtained from the parameter estimation case studies performed on combined model topologies including the complete wind turbine system.

5.2 Parameter estimation configuration

Simulink allows the user to perform parameter estimations using the Control and Estimation Graphical User Interface (GUI) or by making use of the MATLAB command line commands. A decision was made to use the command line commands rather than the GUI since this provides a greater understanding of the process and structure of parameter estimation in MATLAB. The MATLAB parameter estimation process makes use of Object-Oriented Programming (OOP) with eight classes that are of importance, namely *Transient Data*, *State Data*, *Transient Experiment*, *Parameter*, *State*, *Estimation*, *Simulation Options* and *Optimisation Options*. Figure 5-1 shows a diagram of the most important configuration settings for performing parameter estimation. The remainder of this section uses this topology to provide an overview of the configuration required to perform parameter estimation on a Simulink model. The code used for performing parameter estimation on the model shown in Figure 4-14 is provided in APPENDIX H.

The *Model* block refers to the Simulink model that needs to be configured for the parameter estimation process. This entails replacing all input and output signals with input and output ports. In the case of the gearbox model for example, the configured Simulink model would resemble the one shown in Figure 5-2. Appropriate names also need to be given to the variables assigned to the parameter field of the model for easy identification. This model is saved as a MATLAB model file, in this case *Gearbox_Estimation.mdl*, and the filename is then assigned to the *model* property of the *Estimation* object.

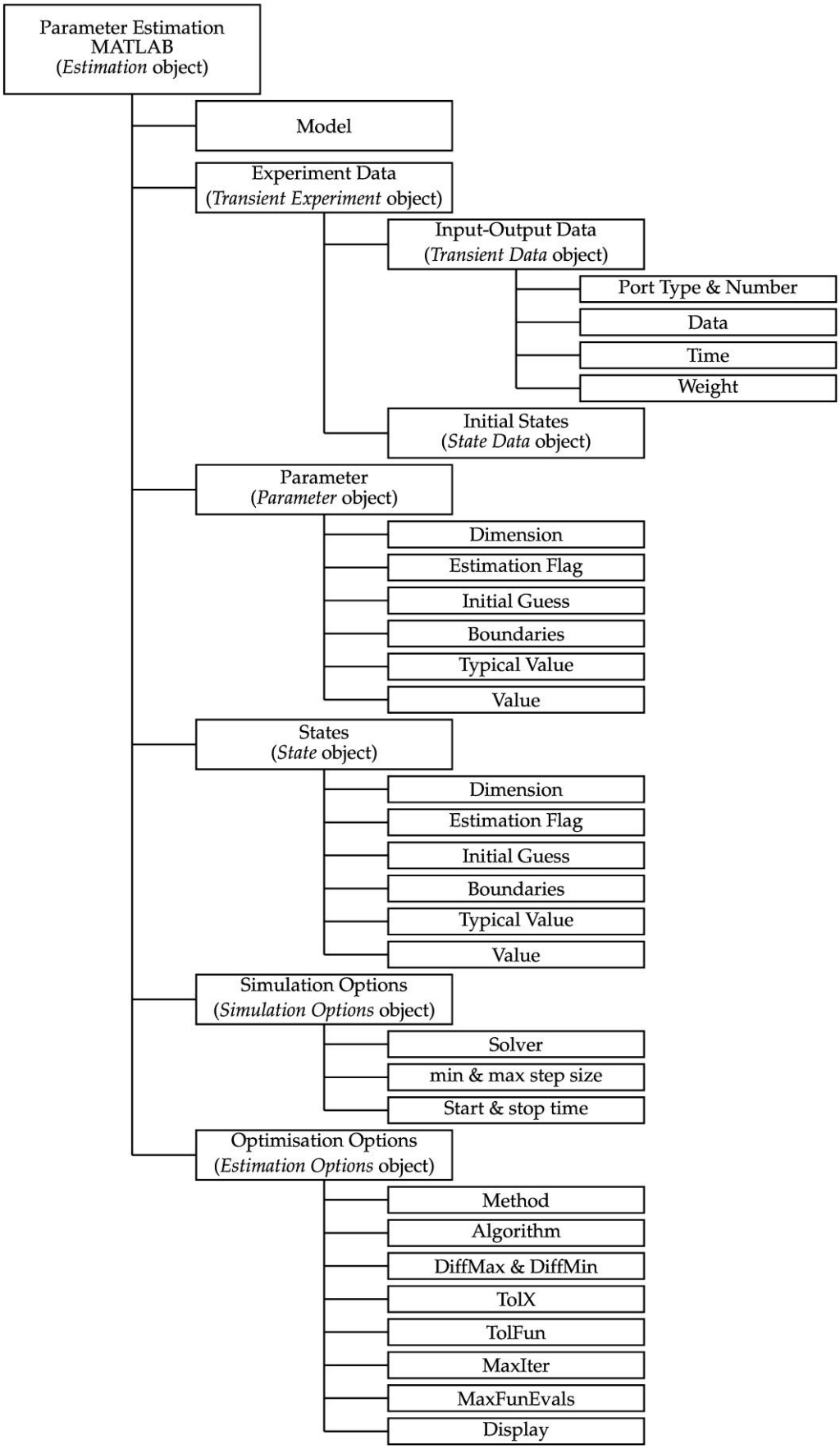


Figure 5-1: Topology of configuration parameters for the parameter estimation process.

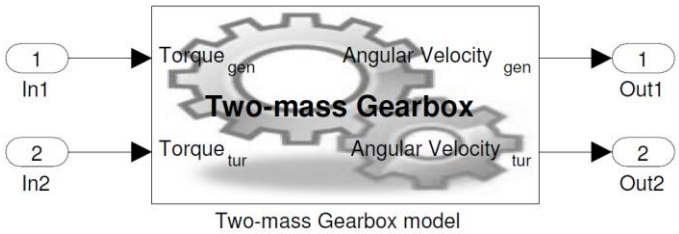


Figure 5-2: Configuration of gearbox model for parameter estimation.

With the model configured, the data structure used for the estimation needs to be configured. A *Transient Data* object is created for each input and output port. This object has properties for the port type, port number, data and time values of the signal as well as a weight factor specifying the relative importance of the signal. Four *Transient Data* objects are created for the gearbox example, namely two for the input signals and two for the output signals. Furthermore, a *State Data* object is created for each Simulink block with states. The *State Data* objects are configured with the initial values of the states for the experiment. The *Transient Data* objects together with the *State Data* objects are assigned to the *input-output data* and *initial states* properties of the *Transient Experiment* object respectively. The *Transient Experiment* object is then assigned to the *experiment data* property of the *Estimation* object.

Further creation and configuration of the *Parameter* objects are required. A *Parameter* object is needed for each parameter of the model. The important properties of this object are *dimension*, *value*, *estimation flag*, *initial guess*, *boundaries* and *typical values*. The *dimension* and *initial guess* properties are self-explanatory. Considering a one dimensional parameter, the *estimation flag* property is a boolean variable configured with a true value for a parameter to be estimated and a false value for a parameter not to be estimated. The *value* property initially contains the initial guess value of the parameter but changes as the value is estimated. If the parameter value is known to be in a specific range, the *boundary* property is configured with a minimum and maximum value. The final property of the *Parameter* object to discuss is the *typical value* property. This value is used for scaling purposes in the estimation process. The typical value of the parameter can be assigned if it is known; otherwise, the initial value of the parameter is automatically assigned to this property. In the case where a parameter is a vector/matrix parameter (e.g., the power coefficient matrix of the turbine blade model) the *value*, *estimation flag*, *initial guess*, *boundaries* and *typical values* properties are vectors/matrices. These vectors/matrices have the same dimensions as the parameter with each element of the property corresponding to an element of the parameter

vector/matrix. All the *Parameter* objects of a model are grouped together and assigned to the *parameter* property of the *Estimation* object.

The *State* object has the same properties as the *Parameter* object and is configured in the same manner as the *Parameter* object. All the *State* objects of a model are grouped together and assigned to the *states* property of the *Estimation* object.

The numerical solver used by Simulink and the optimisation algorithm used for the estimation routine also require configuration. The configuration of the numerical solver is done by creating and configuring a *Simulation Options* object. The most important property of the *Simulation Options* object is the *solver* property. As discussed in Chapter 4, the ode23s solver was found to produce sufficient results and is, therefore, assigned to the *solver* property for all estimations. The *Simulation* object also has properties for minimum and maximum step sizes as well as for stop and start times. If no values are assigned to these properties, it defaults. In such a case, the solver has full control over the step sizes and the stop and start times are retrieved from the *Transient Experiment* object. This *Simulation Opti*ons object is assigned to the *simulation options* property of the *Estimation* object.

An *Optimisation Options* object needs to be created and configured to be assigned to the *optimisation options* property of the *Estimation* object. The *Optimisation Options* object has the following important properties:

- *Method*: This refers to the method used for the optimisation process. Table F-2 provided in APPENDIX E helps the user to decide which method to use. For this study, the sum of least-square objective function was chosen for reasons discussed in Chapter 2. It is also known that the objective function is non-linear in its parameters and that the parameters have either no constraints or only boundary constraints. Therefore, the non-linear least-square method (*lsqnonlin*) is assigned to the *method* property.
- *Algorithm*: This property pertains to the algorithm used for the optimisation process. The options available depend on the configuration of the Method property. With the *method* property set to *lsqnonlin*, the options available are the Trust-region-reflective or Levenberg-Marquardt algorithm. The user's guide [63] recommends using the Trust-region-reflective algorithm for problems that are not underdetermined, that is, problems with fewer equations than dimensions. Since the problems in this study are not underdetermined, the algorithm property is set to *trust-region-reflective*.

- *DiffMin* and *DiffMax*: *DiffMin* and *DiffMax* are the minimum and maximum differences properties respectively. These properties define the minimum and maximum step size for finite differences in gradient estimation.
- *TolX*: The Parameter tolerance property, or *TolX*, is a lower bound for the norm of the step size. The optimisation is terminated if the algorithm tries to take a step smaller than this bound.
- *TolFun*: *TolFun* refers to the function tolerance property. This property defines the lower bound on the change in the value of the objective function from one step to the next. The optimisation is terminated if the change is smaller than this bound.
- *MaxIter*: *MaxIter* is the maximum iterations property. This property defines an upper bound on the number of optimisation iterations. The optimisation is terminated if this boundary is reached.
- *MaxFunEvals*: *MaxFunEvals* is the maximum number of function evaluations property. This property defines an upper bound on the number of times the function can be evaluated. The optimisation is terminated if the boundary is reached.
- *Display*: The display property can be set to display different information about the estimation process.

To conclude the discussion on the *Estimation* object, one more property and method of the object should be mentioned. The parameter estimation process is initiated by invoking the estimate method, e.g., ParamEstimObject.estimate would initiate the parameter estimation process on the ParamEstimObject *Estimation* object. After completion of the estimation process, the *EstimInfo* property of the *Estimation* object holds all the information about the estimation, i.e., the number of iterations, number of function evaluations, values of parameters at each iteration step, the value of the cost function at each iteration and step size of each iteration.

5.3 **Parameter estimation cases for individual models**

5.3.1 Introduction

This section presents the results obtained from the parameter estimation case studies performed on the individual models, i.e., the turbine blade model, gearbox model, ABC DFIG model and DQ DFIG model. The data sets used as experimental data for the estimation processes are sterile simulated data. These data sets are obtained by supplying the

model under investigation with known input data and recording the output data of the model. Considering the block diagram of the parameter estimation process, shown in Figure 2-14, the known input data is denoted as *Input* and the recorded output data is denoted as y.

5.3.2 Turbine blade model

5.3.2.1 Introduction

The turbine blade Simulink model used for the parameter estimation case studies is shown in Figure 5-3.

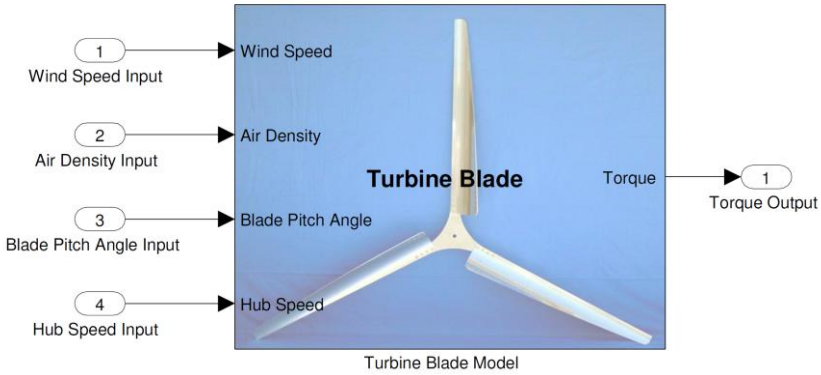


Figure 5-3: Turbine blade Simulink model used for parameter estimation case studies.

The turbine blade model has four parameters, i.e., the blade length R, cut-in wind speed v_{cut-in}, cut-out wind speed $v_{cut-out}$ and the power coefficient C_p. The cut-in and cut-out wind speeds are fixed by the control system of the mechanical brake and, therefore, do not need to be estimated. The length of the blade is easily obtainable either from the manufacturer or by measurement and, therefore, also does not need to be estimated. The parameter of interest for parameter estimation of the turbine blade model is the power coefficient parameter.

As stated in Chapter 3, the power coefficient parameter of the turbine blade model is a matrix with 101 rows for Tip Speed Ratios (TSRs) ranging from 0 to 20 in steps of 0.2 and 31 columns for blade pitch angles ranging from 0° to 30° in steps of 1°. Since the control system of the wind turbine system is not modelled for this study, the turbine blades are used as fixed pitch angle blades with the pitch angle fixed at 5°. Two parameter estimation cases were performed. The first was performed using a generated wind speed input signal. The second was performed with real wind data, obtained from the Gorgonio wind measurement site [78]. In both case studies the air density input was supplied with a constant value of 1 kg/m^3.

5.3.2.2 Case Study 1 – Generated wind speed signal

Figure 5-4 and Figure 5-5 show the wind speed and hub speed signals respectively, used for the first case study. The wind speed signal is randomly generated with different step amplitudes and step lengths in an attempt to excite as many elements of the power coefficient matrix as possible. The hub speed signal was recorded from a forward simulation of the complete wind turbine system using the generated input wind speed. Figure 5-6 shows the resulting output torque obtained when applying these input signals to the turbine blade model. These input and output signals were assigned to the *Transient Experiment* object used for this case study. Since the elements of the power coefficient matrix is known to be positive, the lower boundary property of the power coefficient *Parameter* object was set to 0. The Betz limit, mentioned in Chapter 2, provides a theoretical maximum value of 0.57 for the power coefficient. Therefore, the upper boundary property of the power coefficient *Parameter* object was set to 0.6. The *Parameter* object of the power coefficient parameter was further configured to be estimated by setting the *estimation flag* property, and the *initial guess* property was assigned arbitrary values of 0.2. The *Parameter* objects of the remaining parameters were set not to be estimated and assigned the values as given in Table 5-1.

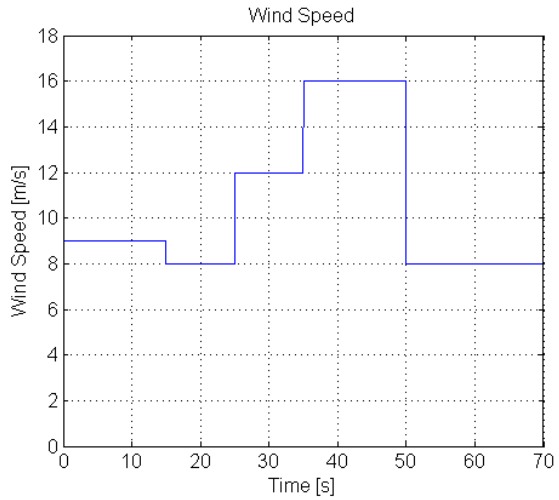


Figure 5-4: Wind speed input to blade turbine model for first blade estimation case study.

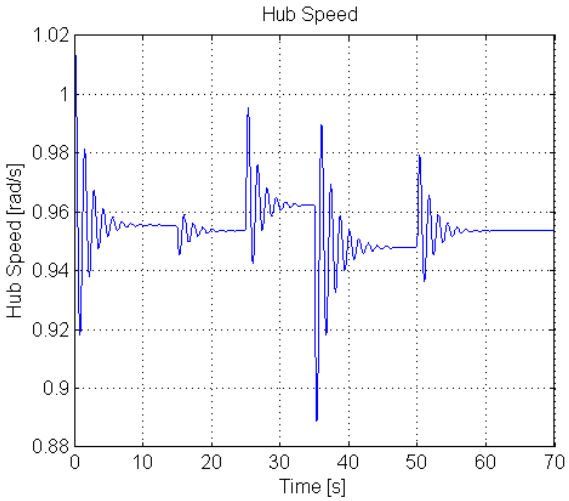


Figure 5-5: Hub speed input to blade turbine model for first blade estimation case study.

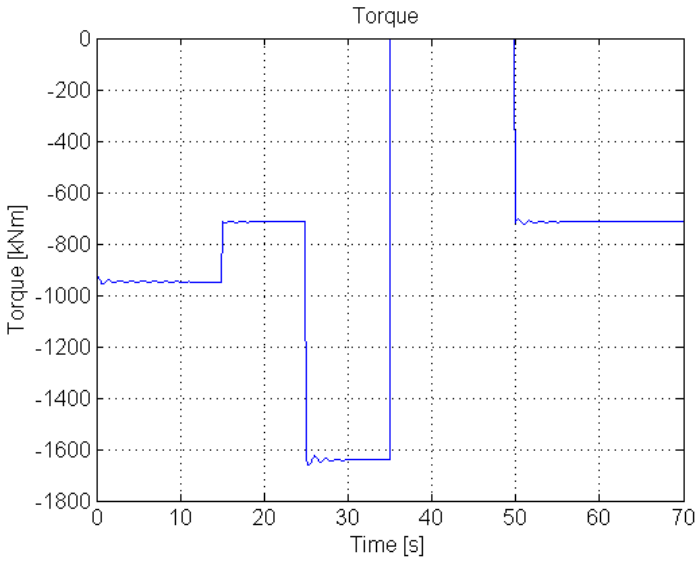


Figure 5-6: Torque output of blade turbine model for first blade estimation case study.

Table 5-1: Turbine blade parameter values.

Parameter	Description	Value	Parameter	Description	Value
R	Blade length	50 m	$v_{cut-out}$	Cut-out Wind Speed	20 m/s
v_{cut-in}	Cut-in Wind Speed	6 m/s			

On the first attempt of estimating the elements of the power coefficient matrix MATLAB produced an "*Out of memory*" error. The estimation was performed on a computer with 32-bit Windows XP operating system. Table F-1 shows that this operating system has a process limit of 2 GB [79]. With the power coefficient matrix being a 3131 element matrix, this limit proves to be insufficient. Since the blade pitch angle is fixed, the decision was made to limit the estimation process to one column of the power coefficient matrix, namely the 5° pitch angle column, thus reducing the number of elements to be estimated from 3131 to 101.

Using this approach, the parameter estimation process performed successfully. The process required 6 iterations and an estimation time of 651 seconds. As expected, the input signals only excited certain elements in the power coefficient matrix, therefore, only these excited elements were successfully estimated. Table 5-2 shows the results for a portion of the 5° blade pitch angle column of the power coefficient matrix including the elements that were excited. The elements that were not excited still have the values of the initial guesses. The percentage error is calculated using

$$Percentage\,Error = \frac{Actual\ Value - Estimated\ Value}{Actual\ Value} \times 100\%. \tag{5.1}$$

Table 5-2: Estimated values of the 5° blade pitch angle column of the C_p matrix for the first parameter estimation case study performed on the turbine blade model.

TSR	Actual C_p Value	Estimated C_p Value	Percentage Error	TSR	Actual C_p Value	Estimated C_p Value	Percentage Error
3.0	0.1329	0.2000	-	5.2	0.3115	0.3115	0%
3.2	0.1539	0.2000	-	5.4	0.3202	0.3202	0%
3.4	0.1754	0.2000	-	5.6	0.3282	0.3282	0%
3.6	0.1951	0.2000	-	5.8	0.3352	0.3352	0%
3.8	0.2143	0.2143	0%	6.0	0.340	0.3402	0%
4.0	0.2320	0.2320	0%	6.2	0.3453	0.3453	0%
4.2	0.2478	0.2478	0%	6.4	0.3494	0.2000	-
4.4	0.2636	0.2000	-	6.6	0.3518	0.2000	-
4.6	0.2765	0.2000	-	6.8	0.3543	0.2000	-
4.8	0.2899	0.2000	-	7.0	0.3563	0.2000	-
5.0	0.3005	0.3005	0%	7.2	0.3578	0.2000	-

- denotes elements that are not excited

These results show that the excited elements of the power coefficient matrix were estimated with 100% accuracy.

5.3.2.3 Case study 2 – Real wind speed data

The second case study was conducted with the wind speed and hub speed signals shown in Figure 5-7 and Figure 5-8 respectively.

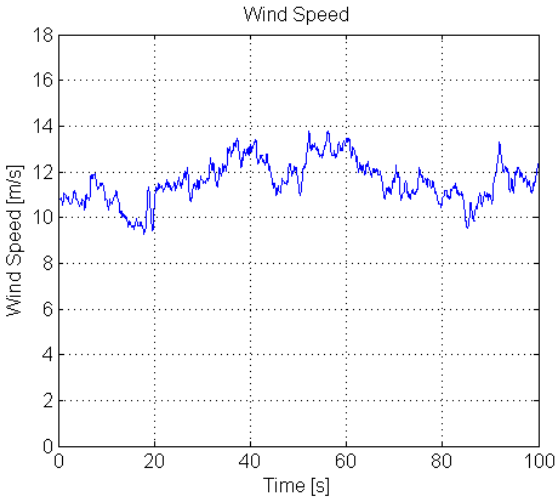


Figure 5-7: Wind speed input to the blade turbine model for second blade estimation case study.

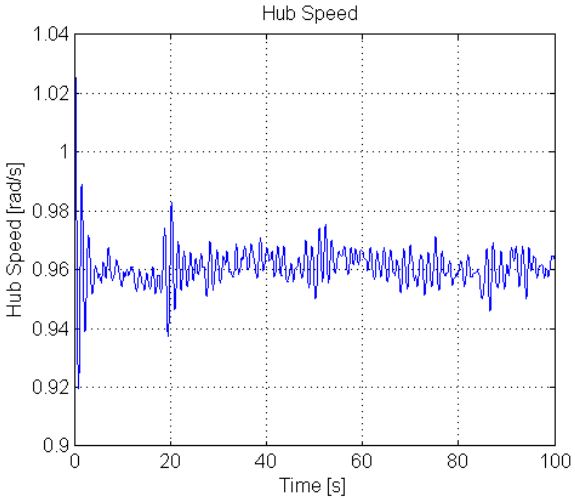


Figure 5-8: Hub speed input to the blade turbine model for second blade estimation case study.

The wind speed input is real wind data obtained from the Gorgonio wind measurement site [78], and the hub speed is again recorded data of a forward simulation of the complete wind turbine system. Figure 5-9 shows the resulting output torque signal used for this case study. The remaining inputs and parameters were kept the same as for the first case study.

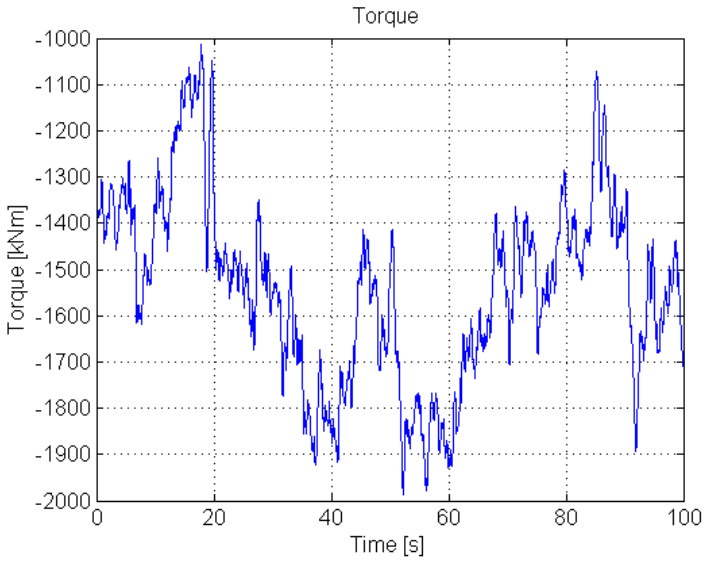


Figure 5-9: Torque output of the blade turbine model for second blade estimation case study.

Table 5-3 shows a portion of the 5° blade pitch angle column containing the excited elements of the power coefficient matrix. This table shows that the elements of the power coefficient matrix for TSRs ranging from 3.4 to 5.2 were estimated with 100% accuracy for the applied input and output signals. These results were obtained after 6 iterations and an estimation time of 915 seconds.

Table 5-3: Estimated values of the 5° blade pitch angle column of the C_p matrix for the second parameter estimation case study performed on the turbine blade model.

TSR	Actual C_p Value	Estimated C_p Value	Percentage Error	TSR	Actual C_p Value	Estimated C_p Value	Percentage Error
3.0	0.1329	0.2000	-	4.6	0.2765	0.2765	0%
3.2	0.1539	0.2000	-	4.8	0.2899	0.2899	0%
3.4	0.1754	0.1754	0%	5.0	0.3005	0.3005	0%
3.6	0.1951	0.1951	0%	5.2	0.3115	0.3115	0%
3.8	0.2143	0.2143	0%	5.4	0.3202	0.2000	-
4.0	0.2320	0.2320	0%	5.6	0.3282	0.2000	-
4.2	0.2478	0.2478	0%	5.8	0.3352	0.2000	-
4.4	0.2636	0.2636	0%	6.0	0.340	0.2000	-

- denotes elements that are not excited

5.3.2.4 Conclusion

The results obtained from the two parameter estimation case studies performed on the turbine blade model show that the excited elements of the power coefficient matrix can be estimated with 100% accuracy. These estimations were limited to one column of the power coefficient due to the memory usage constraint of the operating system. This limitation can be overcome by changing to a 64-bit operating system.

5.3.3 Gearbox model

5.3.3.1 Introduction

The Simulink model of the gearbox used for the parameter estimation case studies is shown in Figure 5-10.

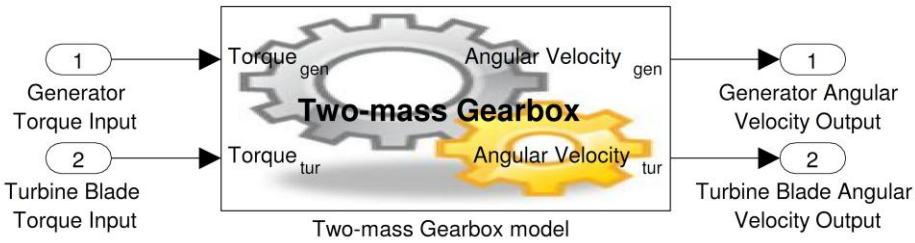


Figure 5-10: Gearbox Simulink model used for parameter estimation case studies.

The gearbox model has five parameters, i.e., moment of inertia of the generator J_{gen}, moment of inertia of the turbine blades J_{tur}, damping coefficient D, stiffness coefficient K and the gear ratio GR. The values of these parameters are mostly supplied by the manufacturers of the generator, but as mentioned in Chapter 1, the values may change due to ageing and operating conditions. Considering a gearbox with a fixed gear ratio, the gear ratio parameter value is fixed and easily obtainable. Therefore, the gear ratio parameter is not included in the investigation of determining which parameters can be estimated. The model has four states, i.e., angular velocity of the generator ω_{gen}, angular velocity of the turbine blades ω_{tur}, angular position of the generator θ_{gen} and angular position of the turbine blades θ_{tur}. The initial values of the states are denoted by adding $init$ to the subscripts of the states. Three case studies were performed on the model to determine which parameters can be estimated and which states require estimation. Table 5-4 summarises the parameter estimation case studies performed on the gearbox model. The reasoning behind the choice of case studies is discussed as the section continues.

The output signals used for the parameter estimation case studies were obtained by performing a forward simulation on the gearbox model. The parameter values and initial state values used for the simulation are given in the Actual Value column of Table 5-5. The generator torque input was supplied with a constant value of 8433.73 Nm, shown in Figure 5-11, and the turbine blade torque input was supplied with the changing signal, shown in Figure 5-12.

Table 5-4: Summary of parameter estimation case studies performed on the gearbox model.

	Case Study 1	Case Study 2	Case Study 3
Data			
Windowed	×	×	✓
Parameters/Initial States			
J_{gen} [kg.m^2]	✓	✓	✓
J_{tur} [kg.m^2]	✓	✓	✓
K [Nm/rad]	✓	✓	✓
D [Nm.sec/rad]	✓	✓	✓
GR	×	×	×
θ_{gen_init} [rad]	✓	✓	✓
ω_{gen_init} [rad/s]	✓	×	×
θ_{tur_init} [rad]	✓	✓	✓
ω_{tur_init} [rad/s]	✓	×	×

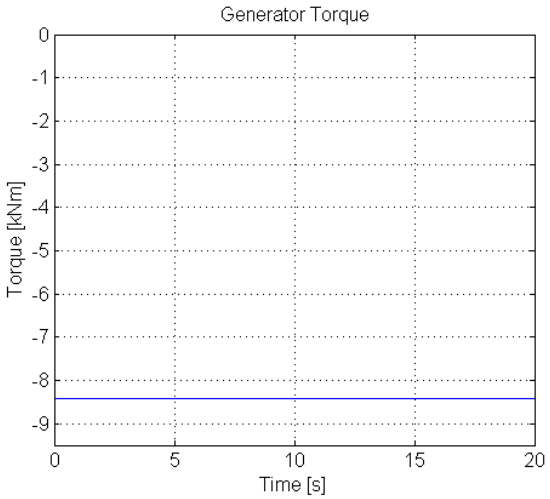


Figure 5-11: Generator torque input to gearbox model for case studies performed on the gearbox model.

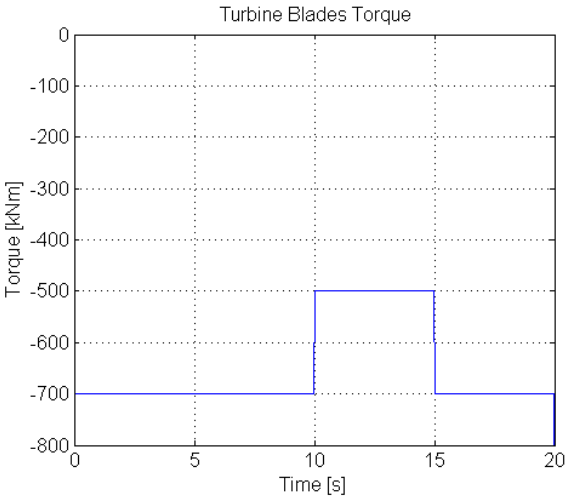


Figure 5-12: Turbine blades torque input to gearbox model for case studies performed on the gearbox model.

Using these input signals, parameter values and initial state values, the simulation was performed on the gearbox model, and the generator and turbine blade angular velocity outputs were recorded. These output signals are shown in Figure 5-13 and Figure 5-14 respectively.

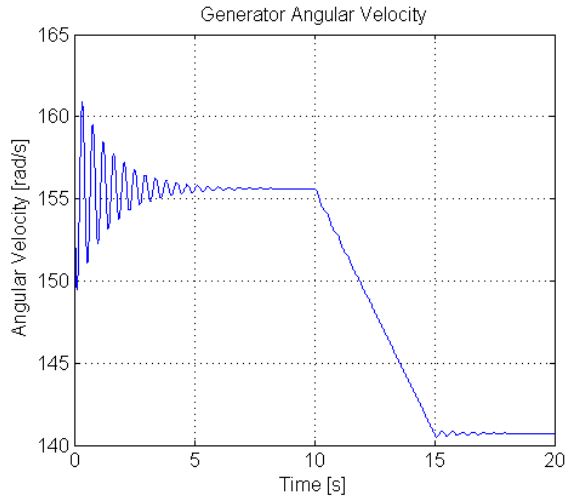


Figure 5-13: Generator angular velocity output of gearbox model for case studies performed on the gearbox model.

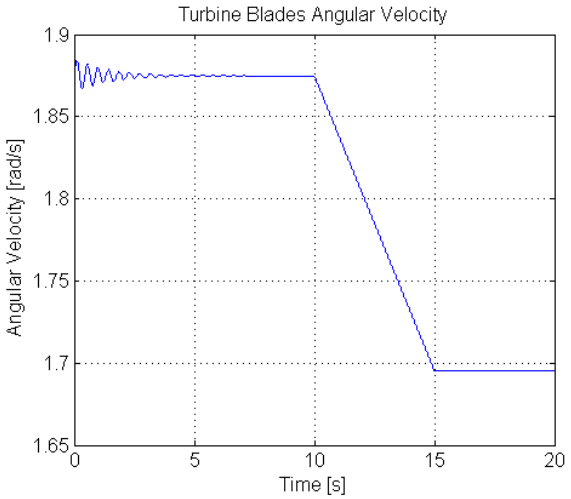


Figure 5-14: Turbine blades angular velocity output of gearbox model for case studies performed on the gearbox model.

5.3.3.2 Case Study 1 – Estimating gearbox parameter values and initial state values

The first case study performed investigated the possibility of estimating all the generator parameters, except the gear ratio, and all the initial state values. These initial state values were set to the values given in the Initial Guess column of Table 5-5. The lower boundary property of all the *Parameter* objects was configured to be 0, since the parameter values are all positive. The *State* object of the turbine blade angular position was configured with the minimum and maximum values for the initial value property set to 0 radians and 6.283 radians respectively. The *State* object of the generator angular position was configured with the minimum value for the initial value property set to 0 radians and the maximum value for the initial value property set to 6.283 times the gear ratio. The range of the parameter values are normally known, therefore, the upper boundary of the *Parameter* objects of the turbine blades inertia, gearbox inertia, stiffness coefficient and damping coefficient were configured to be $1x10^8$ kg.m^2, 200 kg.m^2, $1x10^9$ Nm/rad and $1x10^8$ Nm.sec/rad respectively.

The estimated values together with the percentage error, obtained after 23 iterations of the parameter estimation with an estimation time of 418.9 seconds, are shown in Table 5-5.

Table 5-5: Parameter estimation results for the first case study performed on the gearbox model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	20	19.88515412	77.91%

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
J_{tur} [kg.m^2]	✓	4.95 x10^6	1000000	5433020.895	-9.76%
K [Nm/rad]	✓	114x10^6	10000000	27645684.94	75.75%
D [Nm.sec/rad]	✓	755658	10000	183252.2288	75.75%
GR	×	83	83	83	-
θ_{gen_init} [rad]	✓	1	0.5	22.16080836	-2116.08%
ω_{gen_init} [rad/s]	✓	156.5	41.5	156.4999938	0.00%
θ_{tur_init} [rad]	✓	0.012	0.5	0.286723646	-2279.81%
ω_{tur_init} [rad/s]	✓	1.8735	0.5	1.874597948	-0.06%

✓ denotes parameters estimated and × denotes parameters not estimated

From these results it is clear that the only gearbox parameter that was estimated reasonably accurately is the inertia of the turbine blades. The initial values of the angular velocity of the generator and turbine blades were estimated with nearly 100% accuracy. By performing forward simulations with different initial values assigned to the angular velocity states, it was determined that these initial conditions shift the entire torque output up or down on the graph. By changing these initial values, the steady state response was changed. From this the assumption was made that the parameter estimation process converges to a local minimum when the steady state response of the signal is matched without all the remaining dynamic behaviour matching. The local minimum assumption was investigated by rerunning the estimation with different initial conditions. The results obtained from these estimations showed that for different initial state values, different parameters or state values were estimated more accurately whereas others were less accurately estimated. It was concluded that the parameter estimation process converges to different local minimums depending on the starting (initial) point of the estimation. The following case study considers an approach to overcome this problem.

5.3.3.3 Case study 2 – Estimating gearbox parameter values and initial values of angular position states

In an attempt to try to overcome the problem of the parameter estimation process converging to a local minimum, all the gearbox parameters and states were considered to determine if any can be easily obtained through measurements. Since angular velocity is relatively easy to measure, this case study investigated the possibility of using the actual initial angular velocity

values and estimating the remaining initial values of the states together with the gearbox parameters.

The *State* objects were configured to estimate the initial values of the angular position states, and the initial values of the angular velocity states were set to their actual values. Performing the parameter estimation process, using this configuration, produced the results shown in Table 5-6 after 29 iterations that took 391 seconds to complete.

Table 5-6: Parameter estimation results for the second case study performed on the gearbox model.

Parameters/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	20	91.850517	-2.06%
J_{tur} [kg.m^2]	✓	4.95 x10^6	1000000	4937220.51	0.26%
K [Nm/rad]	✓	114x10^6	10000000	116044079.1	-1.79%
D [Nm.sec/rad]	✓	755658	10000	769211.4862	-1.79%
GR	×	83	-	83	-
θ_{gen_init} [rad]	✓	1	0.5	17.276606	-1627.66%
ω_{gen_init} [rad/s]	×	156.5	-	156.5	-
θ_{tur_init} [rad]	✓	0.012	0.5	0.2080278	-1626.63%
ω_{tur_init} [rad/s]	×	1.8735	-	1.8735	-

✓ denotes parameters estimated and × denotes parameters not estimated

Table 5-6 shows that by providing the parameter estimation process with the actual initial values of the angular velocities, the process no longer converges to a local minimum. The gearbox parameters were estimated accurately with errors below 2.5% for this configuration.

5.3.3.4 Case Study 3 – Estimating gearbox parameter values and initial values of angular position states using windowed data

In both of the previous case studies performed on the gearbox model the entire response of the output signals were used, including the initial part of the output signals with the start-up transients of the simulation. Since the entire output signals were used, these start-up transients were included into the cost function. Actual data obtained from measurements will not have this transient behaviour. Therefore, this case study investigates how removing the start-up transients from the cost function effects the accuracy of the estimated parameters. This case study uses the same configuration as the second case study, but the start-up transients are removed from the cost function.

To remove the start-up transient parts of the output signals from the cost function, a window needs to be applied to the signals, in this case the angular velocity signals, before passing it on to the cost function. MATLAB's parameter estimation process does not make provision for windowing data. Therefore, a window was implemented by using standard Simulink blocks, as shown in Figure 5-15. This configuration makes use of the Simulink simulation time, i.e., Clock, and compares it to the parameters of the two switching elements, i.e., Switch and Switch1. The output is set to a one for time values bigger than the start time of the window and smaller than the end time of the window. For the remainder of the time the configuration produces a zero output. The output of this configuration is then multiplied by the outputs of the Simulink model that require windowing.

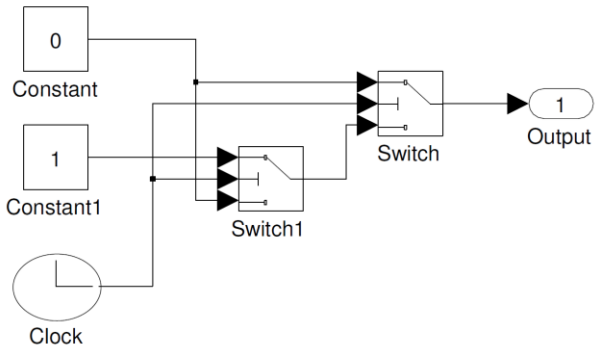


Figure 5-15: Simulink block implementation of a window for parameter estimation application.

By examining the output angular velocity signals, Figure 5-13 and Figure 5-14, it was determined that the start-up transients die out after about 8 s. A window spanning from 8 s to 30 s was, therefore, applied to the output signals for this case study. The results obtained using the windowed data are displayed in Table 5-7. These results were obtained after 31 iterations with an estimation time of 428 seconds.

Table 5-7: Parameter estimation results for the third case study performed on the gearbox model.

Parameters/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	20	91.843151	-2.05%
J_{tur} [kg.m^2]	✓	4950000	1000000	4937274.708	0.26%
K [Nm/rad]	✓	114000000	10000000	116035406.5	-1.79%
D [Nm.sec/rad]	✓	755658	10000	768977.5946	-1.76%
GR	×	83	83	83	-
θ_{gen_init} [rad]	✓	1	0.5	49.314762	-4831.48%
ω_{gen_init} [rad/s]	×	156.5	156.5	156.5	-

Parameters/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
θ_{tur_init} [rad]	✓	0.012	0.5	0.594252	-4832.30%
ω_{tur_init} [rad/s]	×	1.87353	1.8735	1.8735	-

✓ denotes parameters estimated and × denotes parameters not estimated

Table 5-7 shows that the errors of the gearbox parameters are still below 2.5% when the start-up transients were windowed out and the actual initial values were supplied to the angular velocity states.

5.3.3.5 Conclusion

Considering these three case studies the following was concluded. All the gearbox parameters and initial state conditions cannot be estimated simultaneously, but by supplying the estimation process with the actual initial values of the angular velocities the gearbox parameters can be estimated accurately. Accurate estimation of the gearbox parameters can also be obtained when the start-up transients caused by the simulation are windowed out.

5.3.4 ABC DFIG model

5.3.4.1 Introduction

The Simulink ABC DFIG model used for the parameter estimation case studies is shown in Figure 5-16.

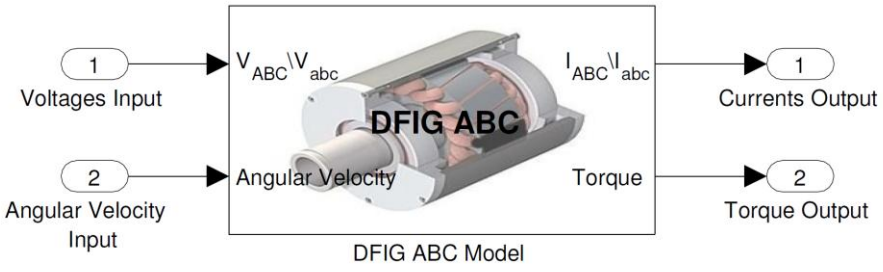


Figure 5-16: DFIG ABC Simulink model used for parameter estimation case studies.

The DFIG ABC model has six parameters, i.e., the stator winding resistance R_s, rotor winding resistance R_r, stator winding inductance L_s, rotor winding inductance L_r, mutual inductance M and the number of poles P. The values of these parameters are mostly supplied by the manufacturers of the generator, but as in the case of the gearbox parameters, these values may change due to ageing and operating conditions. The only generator parameter that is fixed is the number of poles. The model has seven states, namely three for

the stator currents, i.e., I_{as}, I_{bs} and I_{cs}, three for the rotor currents, i.e., I_{ar}, I_{br} and I_{cr} and one for the rotor angular positions θ_r. As for the gearbox model, the initial values of the states are denoted by adding *init* to their subscript. Different case studies were performed on the model to determine which parameters can be estimated and which states need to be estimated. As mentioned earlier, the DFIG model was operated as an induction generator for this study, therefore, a 0 V voltage was supplied to the rotor windings. The stator windings were supplied with a balanced 50 Hz three phase voltage with a Root-Mean-Square (RMS) voltage of 220 V. These supply voltages were used for all parameter estimation case studies performed on the ABC DFIG as well as the case studies performed on the DQ DFIG model in the following section unless otherwise stated. The number of poles of the generator is fixed, therefore, the *Parameter* object of the number of poles was configured not to be estimated in any of the following case studies.

Table 5-8 summarises the parameter estimation case studies performed on the ABC DFIG model. The reasoning behind the choice of case studies is discussed as the section continues.

Table 5-8: Summary of parameter estimation case studies performed on the ABC DFIG model.

	Case Study 1	**Case Study 2**	**Case Study 3**	**Case Study 4**	**Case Study 5**	**Case Study 6**
Data						
Parameter Set	Set 1	Set 1	Set 1	Set 2	Set 2	Set 2
Windowed	×	×(✓)	×	✓	✓	✓
Actual Initial State Values	✓	×	×	×	×	✓
Excitation Signal	Angular Velocity	Angular Velocity	Angular Velocity	Angular Velocity	Stator Voltage	Stator Voltage
Parameters/Initial States						
L_r [H]	✓	✓	✓	✓	✓	✓
L_s [H]	✓	✓	✓	✓	✓	✓
M [H]	✓	✓	✓	✓	✓	✓
P	×	×	×	×	×	×
R_r [Ω]	✓	✓	✓	✓	✓	✓
R_s [Ω]	✓	✓	✓	✓	✓	✓
I_{as_init} [A]	×	×	×	×	×(✓)	×
I_{bs_init} [A]	×	×	×	×	×(✓)	×
I_{cs_init} [A]	×	×	×	×	×(✓)	×
I_{ar_init} [A]	×	×	×	×	×(✓)	×

I_{br_init} [A]	×	×	×	×	×(✓)	×
I_{cr_init} [A]	×	×	×	×	×(✓)	×
θ_{r_init} [rad]	×	×	✓	✓	✓	✓

(✓) denotes results discussed in this section but results only presented in APPENDIX I.

5.3.4.2 Case study 1 – Estimating generator parameter values using actual initial state values

The first parameter estimation case study on the ABC DFIG model was performed by supplying the angular velocity input of the model with the angular velocity signal shown in Figure 5-17. This signal was used to obtain the output currents and output torque signals used for this case study by performing a forward simulation on the model. The parameter and initial state values used to obtain the torque and current output data are given in the Actual Value column of Table 5-9. The output torque signal is shown in Figure 5-18. Although the output current signals are not shown, they were used as part of the cost function for the estimation process.

For this case study, all *Parameter* objects of the generator parameters, except the number of poles, were configured to be estimated and assigned initial guess values of 0.0001. The initial state values are configured not to be estimated and are assigned their actual values. Knowing that the generator parameter values are positive and in the milli and micro order range, their *Parameter* objects were configured with the lower boundaries set to 0 and the upper boundaries set to 1. Table 5-9 displays the values used for the estimation process as well as the results obtained from the successful estimation process. The process required 19 iterations that took 9 minutes to complete.

Table 5-9 presents the percentage error, calculated using (5.1), for each of the estimated parameters. This shows that the parameters were accurately estimated. Those with the highest percentage errors are the stator and rotor winding inductances. Although these are good results, the initial conditions of the states were required to obtain these results. The initial values of the current states are available since the currents are measured as output of the model, but the angular position of the rotor is not that easily obtainable. The next case study investigates the results obtained for a case where the initial values of the states are not known.

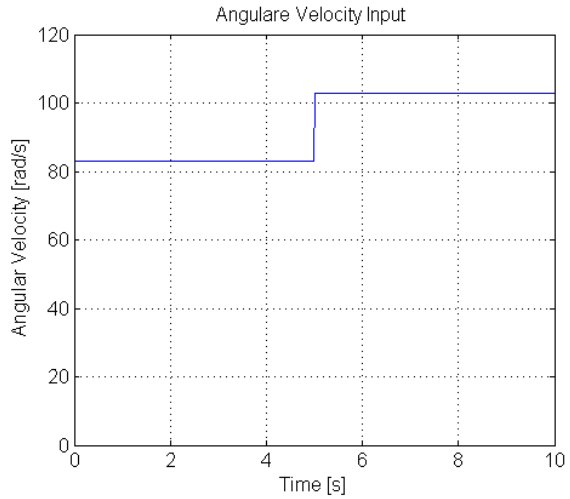


Figure 5-17: Angular velocity input for parameter estimation case studies 1 to 3 performed on ABC DFIG model.

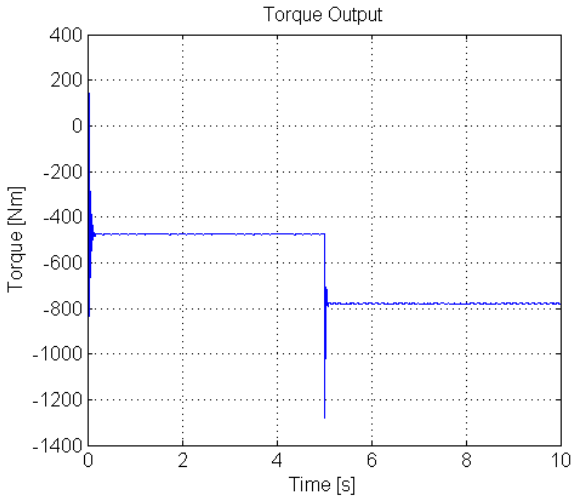


Figure 5-18: Torque output for parameter estimation case studies 1 to 3 performed on ABC DFIG model.

Table 5-9: Parameter estimation results for the first case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.001731	-3.06%
L_s [H]	✓	0.00165	0.0001	0.001597	3.22%
M [H]	✓	0.0466	0.0001	0.046771	-0.37%
P	×	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.183897	0.06%
R_s [Ω]	✓	0.115	0.0001	0.114903	0.08%
I_{as_init} [A]	×	0	0	0	-
I_{bs_init} [A]	×	0	0	0	-
I_{cs_init} [A]	×	0	0	0	-
I_{ar_init} [A]	×	0	0	0	-
I_{br_init} [A]	×	0	0	0	-
I_{cr_init} [A]	×	0	0	0	-
θ_{r_init} [rad]	×	0	0	0	-

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.4.3 Case study 2 – Estimating generator parameter values using random initial state values

This case study investigated the effect on the results obtained when the initial values of the states were not known. This case study makes use of the same input signals as the first case study. New output signals were generated using the parameter values and initial state values

given in the Actual Value column of Table 5-10. The initial guesses and the boundaries of the *Parameter* objects were also set to the same values as for the first case study. The initial values of the states were configured not to be estimated and assigned the values shown in the Initial Guess column of Table 5-10. Table 5-10 presents the results obtained from the parameter estimation process. The process required 22 iterations that took 10 minutes to complete.

Table 5-10: Parameter estimation results for the second case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.003492	-107.86%
L_s [H]	✓	0.00165	0.0001	2.16 x10^{-5}	98.69%
M [H]	✓	0.0466	0.0001	0.027429	41.14%
P	×	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.224112	-21.80%
R_s [Ω]	✓	0.115	0.0001	0.057244	50.22%
I_{as_init} [A]	×	10	0	0	-
I_{bs_init} [A]	×	10	0	0	-
I_{cs_init} [A]	×	10	0	0	-
I_{ar_init} [A]	×	10	0	0	-
I_{br_init} [A]	×	10	0	0	-
I_{cr_init} [A]	×	10	0	0	-
θ_{r_init} [rad]	×	1	0.5	0.5	-

✓ denotes parameters estimated and × denotes parameters not estimated

It is clear from Table 5-10 that the parameter estimation process did not succeed in accurately estimating any of the generator parameter without estimating the states or knowing the correct initial values of the states. The two reasons considered for causing this result were that the transient behaviour at the start of the simulation caused by the initial condition might have dominated the cost function or that one or more of the states have a large enough impact on the output data that this lead to the estimation not succeeding.

In examining the output signals of the model, it was seen that the start-up transients settled in under a second. Using the window configuration discussed in section 5.3.3 a window was applied over the time period 2 to 10 s, thereby giving the start-up transients more than sufficient time to settle. The parameter estimation process was then performed again. The results obtained from this estimation were similar to that of the initial estimation without the

window. These results are presented in Table I-1 of APPENDIX I. From this it was concluded that the start-up transient of the simulation did not cause the estimation process to estimate the generator parameters inaccurately.

To investigate the second theory, the differential equation (3.25) and torque equation (3.29) of the generator was considered. It is clear that the angular position plays a major part in both of these equations. Therefore, the next case study investigated the case where the initial value of the angular position state is estimated together with the generator parameters.

5.3.4.4 Case study 3 – Estimating generator parameter values and initial value of angular position state

The third case study investigated the effect on the results of adding the initial values of the angular position to be estimated. This case study used the same input signals, output signals, parameter values and initial state values as the second case study. All the generator parameters, except the number of poles, were again configured to be estimated. These parameters were configured with the same initial guesses and the boundary values as in the second case study. The initial values of the current states were again configured not to be estimated and were assigned the value shown in the Initial Guess column of Table 5-11. The initial value of the angular position state was configured to be estimated and its initial value was set to 0.5 radians. The estimation process completed successfully with the results given in Table 5-11. The estimation required 19 iterations and took 11 minutes to complete.

Table 5-11: Parameter estimation results for the third case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.001732	-3.08%
L_s [H]	✓	0.00165	0.0001	0.001597	3.22%
M [H]	✓	0.0466	0.0001	0.046825	-0.48%
P	×	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.183885	0.06%
R_s [Ω]	✓	0.115	0.0001	0.115037	-0.03%
I_{as_init} [A]	×	10	0	0	-
I_{bs_init} [A]	×	10	0	0	-
I_{cs_init} [A]	×	10	0	0	-
I_{ar_init} [A]	×	10	0	0	-

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
I_{br_init} [A]	×	10	0	0	-
I_{cr_init} [A]	×	10	0	0	-
θ_{r_init} [rad]	✓	1	0.5	1.000378	-0.04%

✓ denotes parameters estimated and × denotes parameters not estimated

Table 5-11 shows that the generator parameters can be accurately estimated, even with the initial values of the current states not equal to their actual values, as long as the initial value of the angular position state is estimated. This confirmed that one of the states, i.e., the angular position, has a large impact on the output data and is, therefore, required to be estimated for successful estimation of the parameter values of the ABC DFIG model.

5.3.4.5 Case study 4 – Estimating generator parameter values and initial value of angular position state using windowed data

For the fourth parameter estimation case study, a different set of generator parameter and initial state values were used. These are given in the Actual Value column of Table 5-12. The values together with the angular velocity signal, shown in Figure 5-19, were used to obtain the simulated output currents and torque. Figure 5-20 shows the simulated output torque. Although the output currents are not shown, they were again used as part of the cost function for the estimation process.

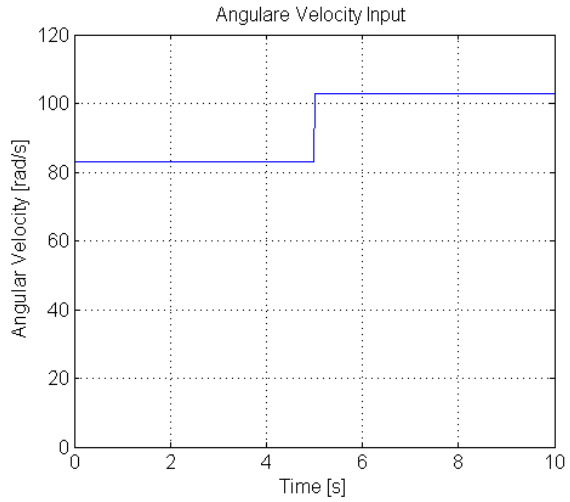


Figure 5-19: Angular velocity input for fourth parameter estimation case study performed on ABC DFIG model.

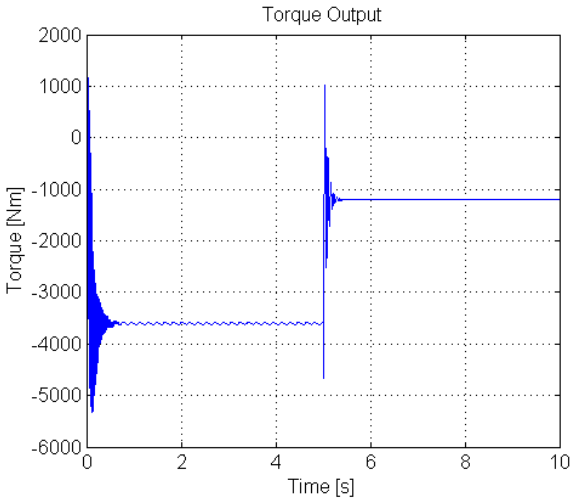


Figure 5-20: Torque output for fourth parameter estimation case study performed on ABC DFIG model.

As for the previous case studies, all the generator parameters, except the number of poles, were set to be estimated and assigned initial values of 0.0001. The initial values of the

current states were configured not to be estimated and assigned initial values of 0 A. The initial value of the angular position state was set to be estimated and assigned an initial value of 0.5 radians. The lower and upper boundaries of the generator parameters were again assigned the values 0 and 1 respectively. The initial value of the angular position state was configured with the values -3.141 radians and 3.141 radians for the lower and upper boundaries respectively.

The parameter estimation process was performed with a window ranging from 2 to 10 s. The process required 33 iterations and took 22 minutes to complete. The results are presented in Table 5-12. These results further confirm that the generator parameters can be estimated accurately if the initial value of the angular position state is estimated.

Table 5-12: Parameter estimation results for the fourth case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.000299	0.0001	0.000296	1.22%
L_s [H]	✓	0.000407	0.0001	0.000411	-0.78%
M [H]	✓	0.016	0.0001	0.015918	0.51%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.008904	-0.05%
R_s [Ω]	✓	0.005	0.0001	0.004961	0.79%
I_{as_init} [A]	×	1	0	0	-
I_{bs_init} [A]	×	1	0	0	-
I_{cs_init} [A]	×	1	0	0	-
I_{ar_init} [A]	×	1	0	0	-
I_{br_init} [A]	×	1	0	0	-
I_{cr_init} [A]	×	1	0	0	-
θ_{r_init} [rad]	✓	1	0.5	0.999642	0.04%

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.4.6 Case study 5 – Voltage signal excitation: Estimating generator parameter values and initial value of angular position state with random initial values assigned to current states

For the fifth case study the angular velocity input was supplied with a constant 83 rad/s instead of the step angular velocity signal used for the previous case studies. To excite the generator parameters the generator's stator windings were supplied with a changing voltage

amplitude. Figure 5-21 shows the A phase voltage supplied to the stator. Figure 5-22 shows the 0.9-1.1 s portion of the supply voltage for the stator of all three phases. These signals were used to obtain the output currents and output torque signals used for this case study by performing a forward simulation on the model. The parameter and initial state values used to obtain the torque and current output data are given in the Actual Value column of Table 5-13. These parameter and initial state values together with the voltage input signals were used to obtain the simulated output currents and torque signals. Figure 5-23 and Figure 5-24 shows the simulated output current signals of the stator and rotor respectively. The output toque signal is shown in Figure 5-25.

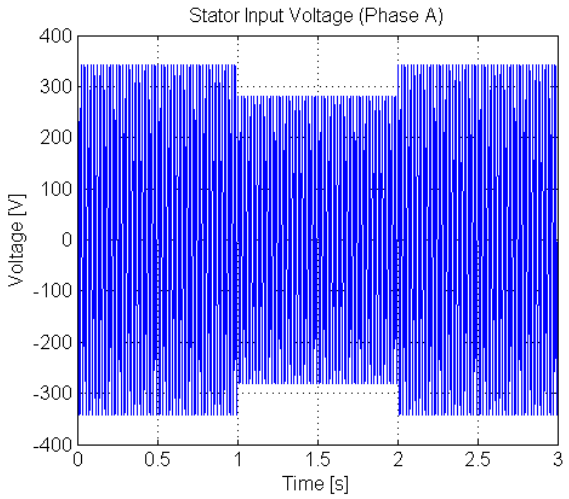


Figure 5-21: Phase A stator voltage input for 5th and 6th parameter estimation case studies performed on ABC DFIG model.

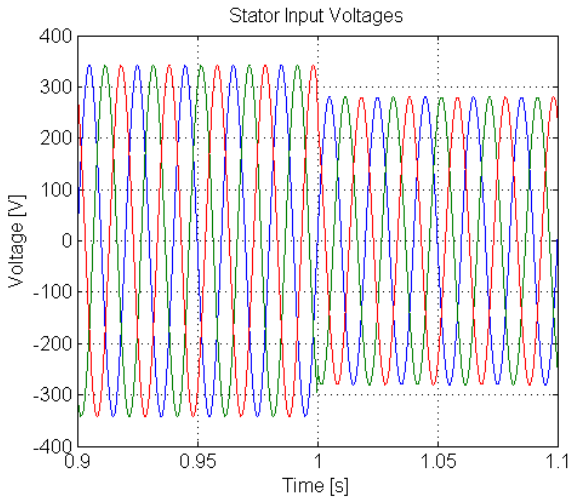


Figure 5-22: Zoomed in portion of stator voltages input for 5th and 6th parameter estimation case studies performed on ABC DFIG model.

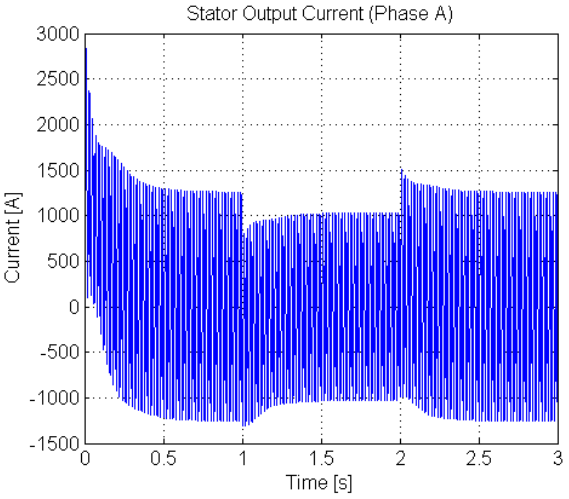


Figure 5-23: Phase A stator current output for 5th and 6th parameter estimation case studies performed on ABC DFIG model.

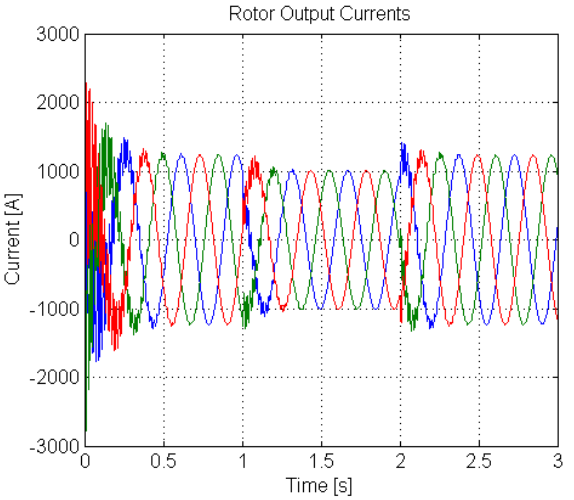


Figure 5-24: Three-phase rotor output currents for 5th and 6th parameter estimation case studies performed on ABC DFIG model.

As for the previous case studies, all the generator parameters, except the number of poles, were set to be estimated and assigned initial values of 0.0001. The initial values of the current states were configured not to be estimated and assigned initial values of 10 A. The initial value of the angular position state was set to be estimated and assigned an initial value of 0.5 radians. The lower and upper boundaries of the generator parameters were assigned the values 0 and 0.1 respectively. The initial value of the angular position state was configured with the values -3.141 radians and 3.141 radians for the lower and upper boundaries respectively. The parameter estimation process was performed with a window ranging from 0.8 to 2.5 s to remove the initial transient caused by the simulation.

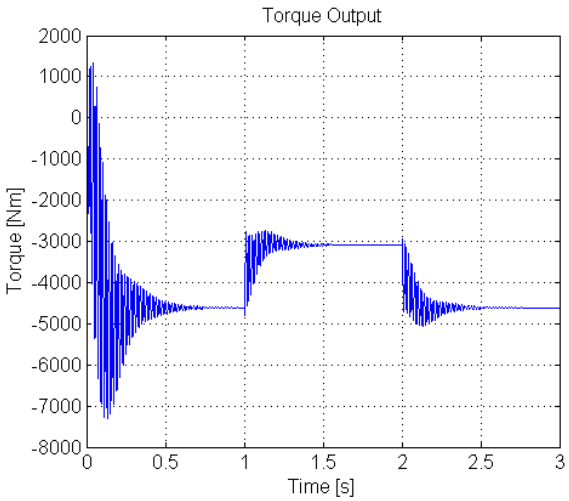


Figure 5-25: Torque output for fifth and sixth parameter estimation case studies performed on ABC DFIG model.

Table 5-13 shows the values used for the estimation process as well as the results obtained from the successful estimation process. The process required 23 iterations that took 650 seconds to complete. These results show that the resistances of the windings are estimated accurately with errors less than 2% but the inductances are inaccurate with errors as high as 40%. To investigate whether the model excited by the stator voltage is more sensitive to the initial values of the state currents compared to it being excited by a step in angular velocity the estimation was performed again with the initial values of the current states also set to be estimated. The results obtained from this estimation are presented in Table I-2 of APPENDIX I. These results show that the resistances of the windings are again estimated accurately with errors less than 1%. The accuracy of the inductances improved, with the highest error being below 12%. As previously mentioned the current is an output of the system that is used in the cost function, therefore, the initial values of the current states are

available. Knowing this, the sensitivity to the initial values of the current states were further investigated by performing a final case study with the initial values of the currents assigned their actual values.

Table 5-13: Parameter estimation results for the fifth case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.0002992	0.0001	0.0001796	39.98%
L_s [H]	✓	0.0004074	0.0001	0.0005201	-27.65%
M [H]	✓	0.016	0.0001	0.0100056	37.46%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.0089727	-0.82%
R_s [Ω]	✓	0.005	0.0001	0.0050799	-1.60%
I_{as_init} [A]	×	1	10	10	-
I_{bs_init} [A]	×	1	10	10	-
I_{cs_init} [A]	×	1	10	10	-
I_{ar_init} [A]	×	1	10	10	-
I_{br_init} [A]	×	1	10	10	-
I_{cr_init} [A]	×	1	10	10	-
θ_{r_init} [rad]	✓	1	0.5	0.9820703	1.79%

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.4.7 Case study 6 – Voltage signal excitation: Estimating generator parameter values and initial value of angular position state with actual initial values assigned to current states

The final parameter estimation case study performed on the ABC DFIG model uses the same input and output signals as the fifth case study. The same configurations were also used for the generator parameters and the angular position state. The initial values of the current states were assigned there their actual values and were set not to be estimated. Table 5-14 shows the values used for the estimation process as well as the results obtained from the successful estimation process. The process required 22 iterations, that took 626 seconds to complete. These results show that all the generator parameter values are estimated accurately with errors below 4%. This confirms that the ABC DFIG model is sensitive to the initial values of the current states when excited by the stator voltages.

Table 5-14: Parameter estimation results for the first case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.0002992	0.0001	0.000288	3.75%
L_s [H]	✓	0.0004074	0.0001	0.000418	-2.64%
M [H]	✓	0.016	0.0001	0.015423	3.61%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.008908	-0.09%
R_s [Ω]	✓	0.005	0.0001	0.005009	-0.18%
I_{as_init} [A]	×	1	1	1	-
I_{bs_init} [A]	×	1	1	1	-
I_{cs_init} [A]	×	1	1	1	-
I_{ar_init} [A]	×	1	1	1	-
I_{br_init} [A]	×	1	1	1	-
I_{cr_init} [A]	×	1	1	1	-
θ_{r_init} [rad]	✓	1	0.5	0.998823	0.12%

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.4.8 Conclusion

The results obtained from the first four parameter estimation case studies performed on the ABC DFIG show that the estimation process is not sensitive to the initial state values of the current states when the model is excited by a step in angular velocity. However, the process shows to be sensitive to the initial value of the angular position state. By estimating the generator parameter values together with the initial value of the angular position state, the parameter values of the generator are accurately estimated with errors below 3.5%. The final two case studies show that the model is also sensitive to the initial values of the current states when excided by steps in the amplitude of the stator voltages. By estimating the generator parameter values together with the initial values of all states, the generator parameter values were accurately estimated with errors below 12%. By assigning the initial current states with their actual values and only estimating the initial value of the angular position state together with the generator parameter values the accuracy is increased to errors below 4%.

From these results it is concluded that the generator parameter values of ABC DFIG model can be estimated accurately, within certain constraints, for both the cases where the model is excited by a step in angular velocity and by steps in amplitude of the stator voltages.

5.3.5 DQ DFIG model

5.3.5.1 Introduction

The Simulink DQ DFIG model used for the parameter estimation case studies is shown in Figure 5-26.

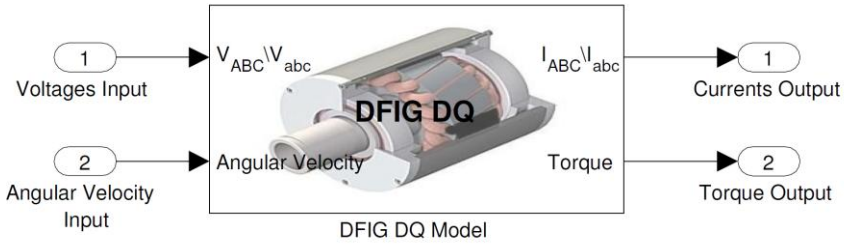


Figure 5-26: DFIG DQ Simulink model used for parameter estimation process.

The DFIG DQ model has the same six parameters as the DFIG ABC model, but instead of seven states it has only five, namely two for the DQ stator currents, i.e., I_{ds} and I_{qs}, two for the rotor currents, i.e., I_{dr} and I_{qr}, and one for the rotor angular position θ_r. The initial values of the states are denoted again by adding *init* to the subscripts of the states. Three case studies were performed on the model to determine which parameters can be estimated. As with the prior parameter estimation case studies performed on the ABC DFIG model, all the generator parameters, except the number of poles, were configured with lower boundaries assigned zeros and upper boundaries assigned ones. The initial value of the angular position state was assigned the values -3.141 radians and 3.141 radians for the lower and upper boundary respectively.

Table 5-15 summarises the parameter estimation case studies performed on the DQ DFIG model. The reasoning behind the choice of case studies is discussed as the section continues.

Table 5-15: Summary of parameter estimation case studies performed on the DQ DFIG model.

	Case Study 1	**Case Study 2**	**Case Study 3**
Data			
Data Set	Data Set 1	Data Set 2	Data Set 2
Windowed	✓(×)	✓	✓
Actual Initial State Values	×(✓)	×	✓(×)
Excitation Signal	Angular Velocity	Angular Velocity	Stator Voltage
Parameters/Initial States			

	Case Study 1	Case Study 2	Case Study 3
R_r [Ω]	✓	✓	✓
R_s [Ω]	✓	✓	✓
L_r [H]	✓	✓	✓
L_s [H]	✓	✓	✓
M [H]	✓	✓	✓
P	×	×	×(✓)
I_{as_init} [A]	×	×	×(✓)
I_{bs_init} [A]	×	×	×(✓)
I_{cs_init} [A]	×	×	×(✓)
I_{ar_init} [A]	×	×	×(✓)
I_{br_init} [A]	×	×	×(✓)
I_{cr_init} [A]	×	×	×(✓)
θ_{r_init} [rad]	✓	✓	✓

(✓) and (×) denote results discussed in this section but results only presented in APPENDIX I.

5.3.5.2 Case Study 1 – Estimating generator parameter values and initial value of angular velocity state for Data Set 1 with window applied

Figure 5-27 shows the angular velocity signal used for the first estimation case study of the DQ DFIG model. This signal was used together with the parameter values and initial state values, given in the Actual Value column of Table 5-16, to obtain the output currents and torque by performing a forward simulation on the DQ DFIG model. The output torque is shown in Figure 5-28. Although the current signals are not shown, they were used as part of the cost function for the estimation process.

For this case study all the generator parameters, except the number of poles, were set to be estimated and assigned initial values of 0.0001. The parameter estimation case studies performed on the ABC DFIG model showed that the angular position plays an integral part in the estimation of the generator parameters. Therefore, the initial value of the angular position was set to be estimated for this case study. The initial state values of the currents were set not to be estimated and assigned random values not equal to their actual values. The output signals for this case study were windowed for the time spanning from 2 to 8 s to remove the start-up transient caused by the simulation.

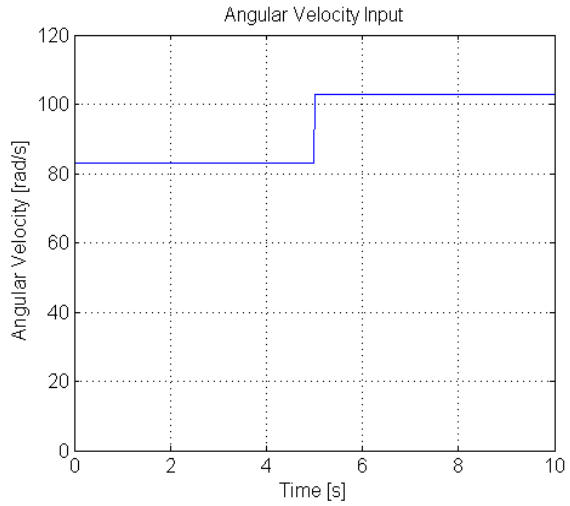


Figure 5-27: Angular velocity input for first parameter estimation case study performed on the DQ DFIG model.

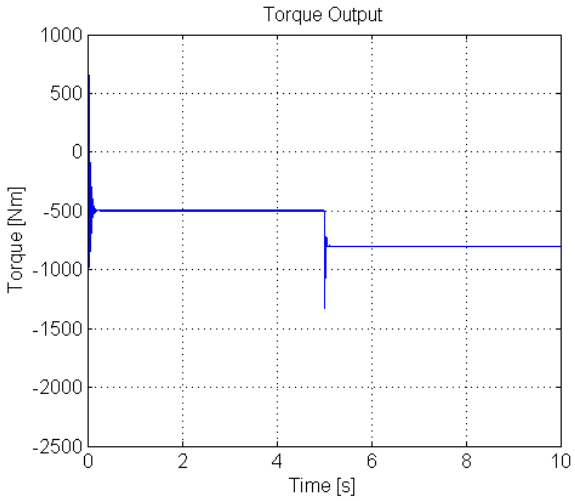


Figure 5-28: Torque output for first parameter estimation case study performed on the DQ DFIG model.

Table 5-16 displays the results obtained from the parameter estimation process as well as the values used for the process. The process required 20 iterations and 174 seconds to complete.

Table 5-16: Parameter estimation results of first case study performed on the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.001856	-10.46%
L_s [H]	✓	0.00165	0.0001	0.001448	12.23%
M [H]	✓	0.0466	0.0001	0.0486	-4.29%
P	✓	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.184015	-0.01%
R_s [Ω]	×	0.115	0.0001	0.118597	-3.13%
I_{ds_init} [A]	×	10	1	1	-
I_{qr_init} [A]	×	10	1	1	-
I_{dr_init} [A]	×	10	1	1	-
I_{qr_init} [A]	×	10	1	1	-
θ_{r_init} [rad]	✓	1	2	0.997852	0.21%

✓ denotes parameters estimated and × denotes parameters not estimated

The results in Table 5-16 show that by estimating the generator parameter values and the initial value of angular position state, the values of the generator parameters can be estimated with reasonable accuracy for the resistances and mutual inductance, but the winding inductances have error percentages above 10%. Two additional estimations processes were performed on the model to try to improve the accuracy. The first estimation was performed with all the data remaining the same except the initial values of the current states, which were

assigned their actual values. For the second estimation, the configuration was reset back to the original configuration and performed with the window removed. The results of these two additional case studies are presented in APPENDIX I. The results of the first, given in Table I-3, are similar to that of the original case. The results of the second case, given in Table I-4, are much less accurate than the results of the original case study.

The estimation process was reset to the original configuration and further estimations were performed with different initial guesses for the generator states. These estimations all produced results similar to that given in Table 5-16. From this it was concluded that this case study has a local minimum close to the global minimum to which the algorithm continues to converge.

5.3.5.3 Case study 2 – Estimating generator parameter values and initial value of angular velocity state for Data Set 2 with window applied

The second case study performed on the DQ DFIG model made use of a different set of generator parameter and initial state values to simulate the torque output, shown in Figure 5-30, and current outputs. The parameter values and initial state values are given in the Actual Value column of Table 5-17. This case study made use of the same configuration as the first case study. All the generator parameters and initial states, except the number of poles, and the initial value of the angular position state were set to be estimated.

Table 5-17 shows the results of the parameter estimation process after 18 iterations that took 150 seconds to complete. From these results it is clear that the generator parameters were estimated more accurately with the error percentages of all generator parameters below 9%.

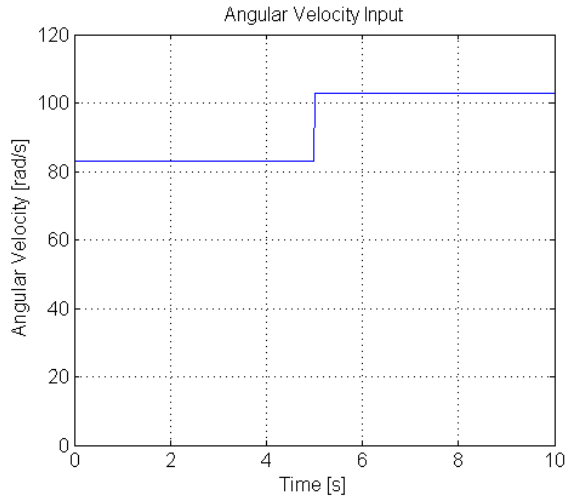


Figure 5-29: Angular velocity input for second parameter estimation case study performed on DQ DFIG model.

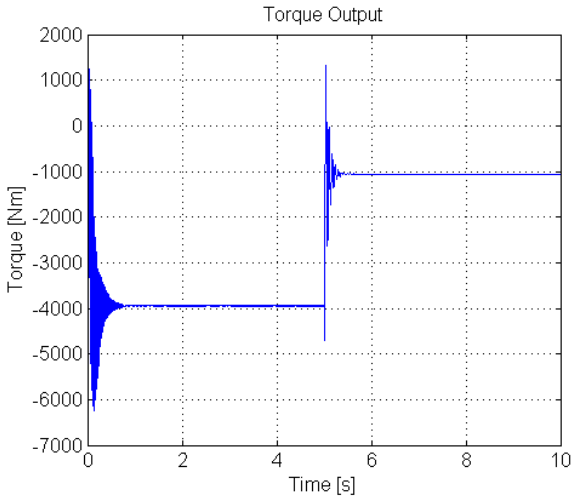


Figure 5-30: Torque output for second parameter estimation case study performed on DQ DFIG model.

Table 5-17: Parameter estimation results of second case study performed on DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.000299	0.0001	0.000296	1.22%
L_s [H]	✓	0.000407	0.0001	0.000404	0.81%
M [H]	✓	0.016	0.0001	0.015422	3.61%
P	✓	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.008896	0.05%
R_s [Ω]	×	0.005	0.0001	0.005433	-8.66%
I_{ds_init} [A]	×	1	0	0	-
I_{qr_init} [A]	×	1	0	0	-
I_{dr_init} [A]	×	1	0	0	-
I_{qr_init} [A]	×	1	0	0	-
θ_{r_init} [rad]	✓	1	2	0.999642	-0.04%

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.5.4 Case study 3 – Voltage signal excitation: Estimating generator parameters and initial value of angular position state with actual initial values assigned to current states

As for the ABC DFIG model the case studies of the DQ DFIG model are concluded by investigating the results obtained by exciting the model through the stator voltages rather than the angular velocity. The input angular velocity is once again kept constant at 83 rad/s. The stator voltage input signals used are shown in Figure 5-31 and Figure 5-32. These signals together with the generator parameter and initial state values shown in the Actual Value

column of Table 5-18 were used to obtain the output signals used for the estimation process. The resulting stator current, rotor current and torque output signals are shown in Figure 5-33, Figure 5-34 and Figure 5-35 respectively.

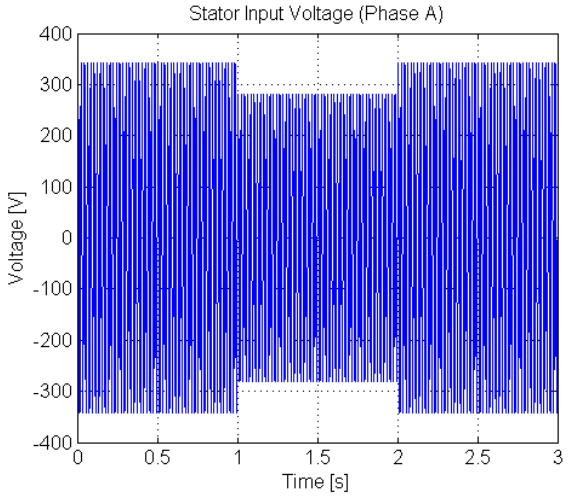


Figure 5-31: Phase A stator voltage input for third parameter estimation case study performed on DQ DFIG model.

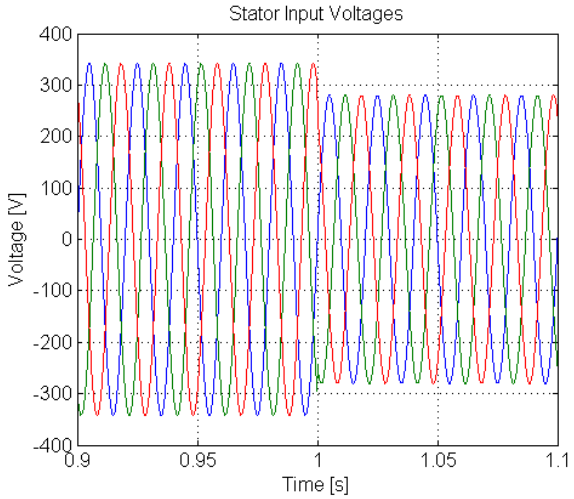


Figure 5-32: Zoomed in portion of stator voltages input to DQ DFIG model for third parameter estimation case study.

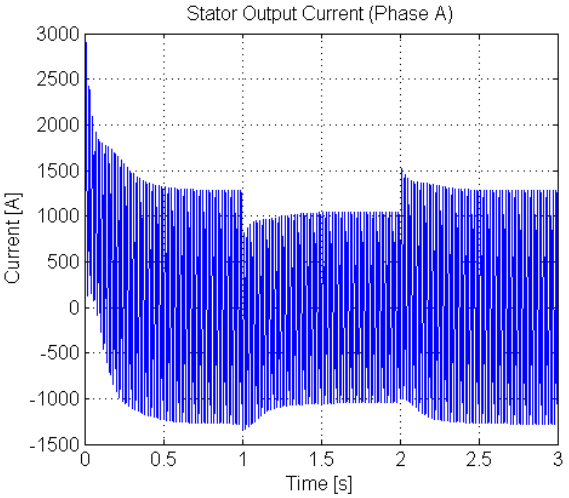


Figure 5-33: Phase A stator current output for third parameter estimation case study performed on DQ DFIG model.

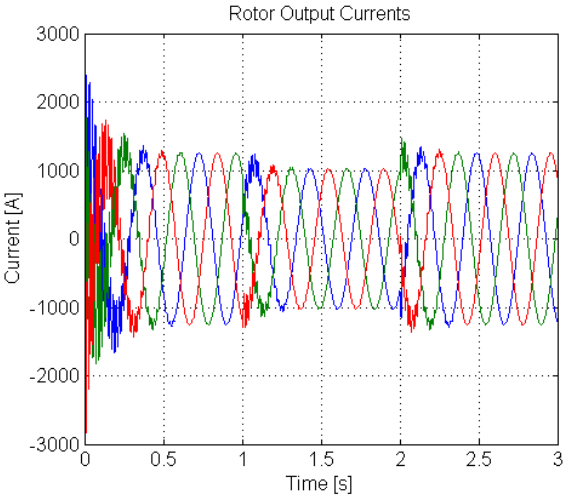


Figure 5-34: Three-phase rotor output currents for third parameter estimation case study performed on DQ DFIG model.

As for the previous case studies performed on the DQ DFIG model, all the generator parameter values, except the number of poles, were set to be estimated and assigned initial values of 0.0001. The initial value of the angular position state was set to be estimated and assigned an initial value of 0.5 radians. The lower and upper boundaries of the generator parameters were again assigned the values 0 and 0.1 respectively. The initial value of the angular position state was configured with the values -3.141 radians and 3.141 radians for the

lower and upper boundaries respectively. The parameter estimation process was performed with a window ranging from 0.8 to 2.5 s.

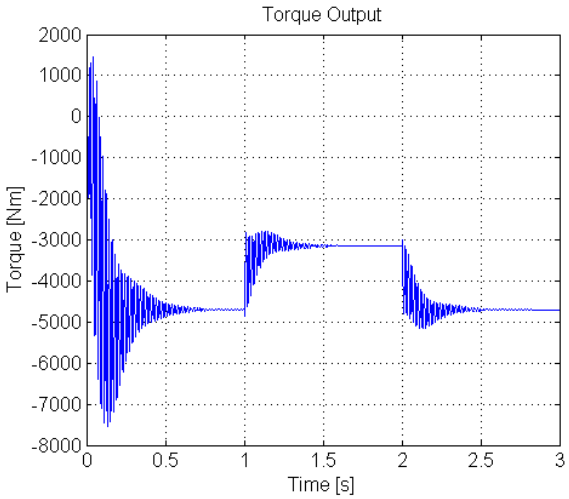


Figure 5-35: Torque output for third parameter estimation case study performed on DQ DFIG model.

Three parameter estimation case studies were performed on the DQ DFIG model with stator voltage excitation, the results of the first two case studies are presented in APPENDIX I. The first case study was performed using the above mentioned configuration together with the initial values of the current states that were assigned initial values of 10 A and set not to be estimated. The results obtained using this configuration is presented in Table I-5. As for the case study performed on the ABC DFIG model with the same configuration the resistances are accurately estimated with errors below 1.5% while the inductances are inaccurately estimated with errors above 19%. The estimation configuration was then changed to also estimate the initial values of the current states. The results of this estimation process are presented in Table I-6. All the generator parameters were estimated accurately with errors below 7%. This again pointed to the generator model being sensitive to the initial values of the current states when excited by the stator voltages.

As mentioned in the ABC DFIG case studies sections, the currents are an output of the model that is used in the cost function, therefore, the initial values of these states are available. For the final case study of the DQ DFIG model the current states were assigned their actual values and set not to be estimated. Table 5-18 shows the results of the parameter estimation process after 31 iterations that took 10 minutes to complete. From these results it is clear that the generator parameter values were estimated more accurately with the error of all

parameters below 6%. Thereby confirming that the DQ DFIG model is also sensitive to the initial values of the current states when excited by the stator voltages.

Table 5-18: Parameter estimation results for the third case study of the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.0002992	0.0001	0.0002816	5.88%
L_s [H]	✓	0.0004074	0.0001	0.0004237	-3.98%
M [H]	✓	0.016	0.0001	0.0150644	5.85%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.0089002	0.00%
R_s [Ω]	✓	0.005	0.0001	0.004998	0.04%
I_{as_init} [A]	×	1	1	1	-
I_{bs_init} [A]	×	1	1	1	-
I_{cs_init} [A]	×	1	1	1	-
I_{ar_init} [A]	×	1	1	1	-
I_{br_init} [A]	×	1	0.5	1.0019703	-0.20%
I_{cr_init} [A]	×	0.0002992	0.0001	0.0002816	5.88%
θ_{r_init} [rad]	✓	0.0004074	0.0001	0.0004237	-3.98%

✓ denotes parameters estimated and × denotes parameters not estimated

5.3.5.5 Conclusion

The results obtained from these three case studies together with the additional case studies show that the DQ DFIG model, like the ABC DFIG model, is sensitive to the initial value of the angular position state when excited by the angular velocity. When excited by the stator voltages it is also sensitive to the initial values of the current states. When these initial state values were either estimated or assigned their actual values the generator parameter values were estimated accurately. Although the results of the case studies show that the generator parameter values were estimated accurately with the highest error being below 13% these results are less accurate than those of the ABC DFIG model. From this it is concluded that the cost function is less sensitive to the generator parameters in the case of the DQ model than in the case for the ABC model. Further investigation is required to determine the cause of the cost function of the DQ DFIG model being less sensitive to the parameters than that of the ABC DFIG model.

5.4 Parameter estimation case studies for combined model topologies

5.4.1 Overview

This section presents the results obtained from the parameter estimation case studies performed on combined model topologies.

The data sets used as experimental data for the estimation processes are sterile simulated data as is the case for the experimental data used for the case studies of the individual models. These data sets are obtained by supplying the combined model topologies under investigation with known input data and recording the output data of the model.

This first combined model topology that was investigated is that of the turbine blade and gearbox model together. This is followed by the combined model topology with the turbine blade model, gearbox model and DFIG model all connected. This topology represents a complete wind turbine system.

5.4.2 Turbine blade and gearbox combined model topology case studies

This section presents the results obtained from parameter estimation case studies performed on the combined topology consisting of the turbine blade model and gearbox model shown in Figure 5-36. These case studies investigated which of the gearbox parameter values can be estimated.

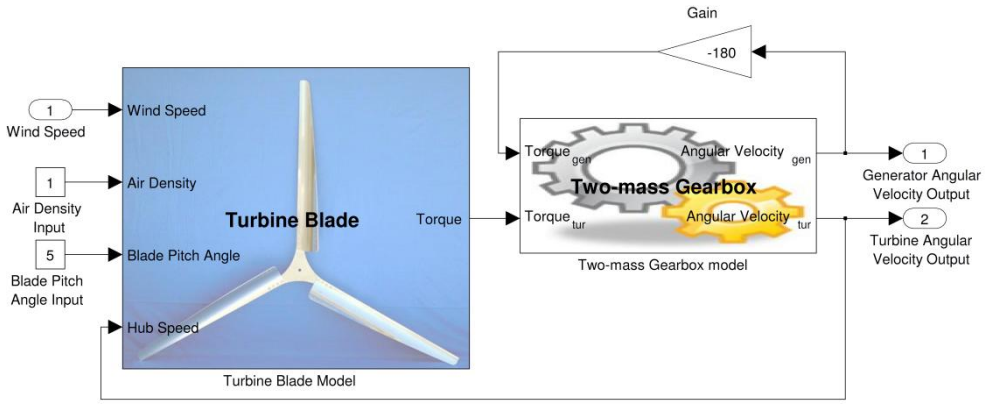


Figure 5-36: Combined topology of turbine blade model and gearbox model used for parameter estimation case studies.

The complete wind turbine system requires the generator torque to be fed back to the gearbox as well as the turbine blade angular velocity to be fed back to the turbine blade model for the system to be stable. Since the generator model is not present in this topology the generator torque feedback loop needed to be added. This was achieved by adding the feedback loop

with the gain block (Gain) shown in Figure 5-36. This configuration represents a linear torque curve.

The output data set used for the parameter estimation process was generated by performing a forward simulation on the topology, using the wind speed signal shown in Figure 5-37 as input. The values of the power coefficient parameter used for the simulation are shown in Figure 4-2. The remaining parameter values of the turbine blade model are provided in Table 5-19 and the parameter values of the gearbox model are provided in the Actual Value column of Table 5-20.

Table 5-19: Turbine blade parameter values used for parameter estimation case study performed on turbine blade and gearbox combined topology.

Parameter	Description	Value	Parameter	Description	Value
R	Blade length	50 m	$v_{cut-out}$	Cut-out Wind Speed	20 m/s
v_{cut-in}	Cut-in Wind Speed	6 m/s			

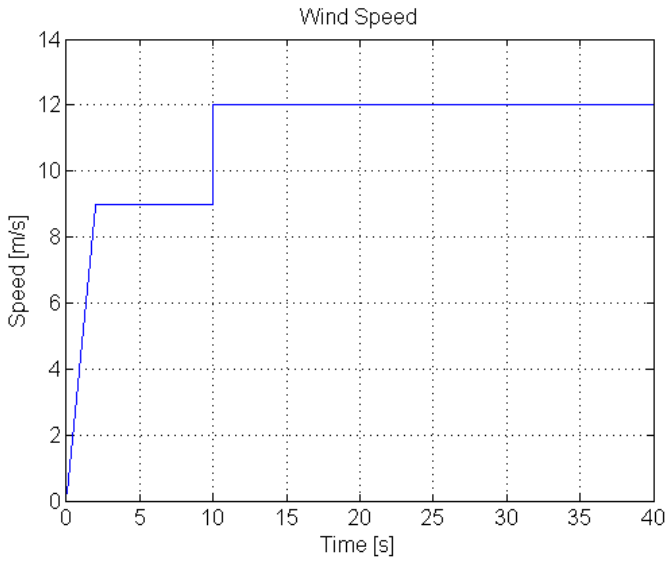


Figure 5-37: Wind speed input to turbine blade and gearbox combined topology for parameter estimation case study.

The first parameter estimation case study was performed with all the gearbox parameter values, except the gear ratio, together with the initial values of the angular position states configured to be estimated. The initial values of the angular velocity states were assigned their actual values. As for the first parameter estimation case study performed on the individual gearbox model, the lower boundaries of the parameter were all set to 0. The upper boundary of the turbine blades inertia, gearbox inertia, stiffness coefficient and damping coefficient parameters were set to $1x10^8$ kg.m^2, 200 kg.m^2, $1x10^9$ Nm/rad and

$1x10^8$ Nm.sec/rad respectively. A parameter estimation process was performed using this configuration with the turbine blade angular velocity and the generator angular velocity used for the cost function. The results obtained from the estimation showed it to converge to a local minimum with none of the estimated parameter or initial state values even close to their actual values.

The feedback loops in this topology cause an entirely different transfer function compared to that of the transfer functions of the two individual models. This may lead to the system becoming less or more sensitive for certain parameters. Certain parameter values could also lead to the system becoming unstable. In an attempt to overcome these possible problems, the boundaries of the parameters were tightened. The parameter estimation process was performed again with the boundary values set to the values given in the Lower (Upper) bound column of Table 5-20. The results obtained from performing the parameter estimation process using these configurations are presented in Table 5-20. These results show that all the gearbox parameter values were accurately estimated with errors below 9%. These results were obtained after 22 iterations that took 423 minutes to complete.

Table 5-20: Parameter estimation results for case study performed on turbine blade and gearbox combined topology.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Lower (Upper) Bound	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	120	50 (150)	86.8739	3.47%
J_{tur} [kg.m^2]	✓	4.95 x10^6	8 x 10^6	1 x10^5 (1 x10^7)	49.72 x10^6	-0.44%
K [Nm/rad]	✓	114 x10^6	10000000	1 x10^6 (1 x10^9)	110.68 x10^6	2.91%
D [Nm.sec/rad]	✓	755658	400000	1x10^5 (1x10^6)	692726	8.33%
GR	×	83	83	-	-	-
θ_{gen_init} [rad]	✓	0	0.5	0 (6.283)	3.2790	-
ω_{gen_init} [rad/s]	×	80	80	-	-	-
θ_{tur_init} [rad]	✓	0	0.5	0 (6.283)	0.0401	-
ω_{tur_init} [rad/s]	×	1	1	-	-	-

✓ denotes parameters estimated and × denotes parameters not estimated

From these results it is concluded that the gearbox parameters can be accurately estimated for this topology when the estimation process is limited to consider parameter values in the range of the parameters' typical values.

5.4.3 Complete wind turbine system case studies

5.4.3.1 Introduction

This section presents the results obtained from parameter estimation case studies performed on the wind turbine system model consisting of the turbine blade model, gearbox model and DFIG model. These are connected as shown in Figure 5-38. Four case studies were performed on this model. The first two case studies investigated whether the gearbox parameter values could be estimated, whereas the third case study investigated whether the generator parameter values could be estimated. The fourth case study investigated whether a combination of gearbox and generator parameter values could be estimated.

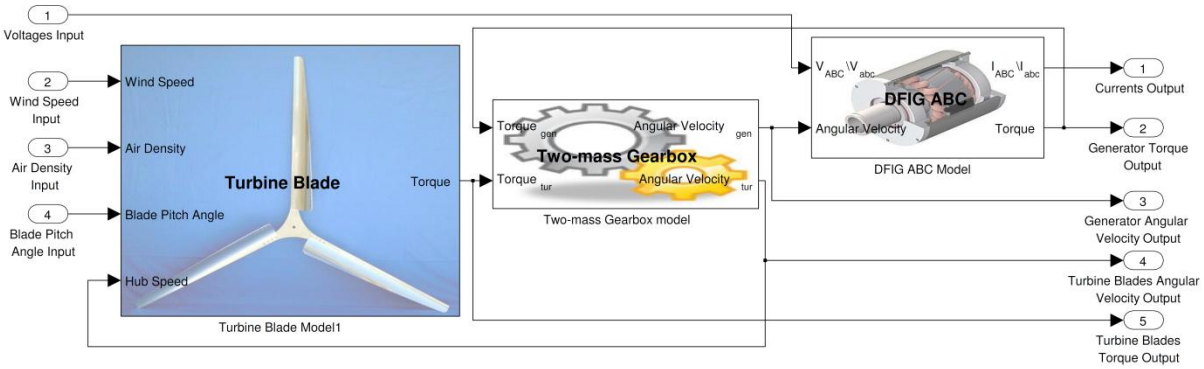


Figure 5-38: Wind turbine system Simulink model used for parameter estimation case studies.

5.4.3.2 Case study 1 – Gearbox parameter values

To investigate whether the gearbox parameter values could be estimation within the complete system model a forward simulation was performed on the complete system shown in Figure 5-38 to obtain the output signals required for the parameter estimation process. The same wind speed input signal was used as for the turbine blade and gearbox combined model topology case studies, shown in Figure 5-37. The power coefficient parameter values used of the turbine blade model are shown in Figure 4-2. The remaining parameter values of the turbine blade model are provided in Table 5-19. The parameter values of the generator model are provided in Table 5-21 and the parameter values of the gearbox are given in the Actual Value column of Table 5-22.

Table 5-21: DFIG model parameter values.

Parameter	Description	Value	Parameters	Description	Value
R_s	Stator resistance	5 mΩ	L_r	Rotor inductance	299.2 μH
R_r	Rotor resistance	8.9 mΩ	L_m	Magnetising inductance	16 mH

Parameter	Description	Value	Parameters	Description	Value
L_s	Stator inductance	407.5 μH	P	Number of poles in machine	4

For this first case study only the gearbox parameters were set to be estimated, excluding the gear ratio, the boundaries of the parameters were configured to the same values as for the final case study performed on the turbine blade and gearbox combined model. These values are shown in the Lower (Upper) Bound column of Table 5-22. The initial values of the states were assigned their actual values. Since angular velocity is easier to measure than torque the generator and turbine blade angular velocities were used for the cost function. The initial transient caused by the simulation was windowed out. The results obtained from the parameter estimation process using this configuration produced very inaccurate results with the lowest error being 32% as shown in Table 5-22. These results were obtained after 10 iterations that took 129 minutes to complete.

Table 5-22: Parameter estimation results of first case study performed on the complete wind turbine system model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Lower (Upper) Bound	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	120	50 (150)	50.000001	44.44 %
J_{tur} [kg.m^2]	✓	4.95 x10^6	8 x 10^6	1 x10^5 (1 x10^7)	49.72 x10^6	-53.77%
K [Nm/rad]	✓	114 x10^6	10000000	1 x10^6 (1 x10^9)	110.68 x10^6	91.06%
D [Nm.sec/rad]	✓	755658	400000	1x10^5 (1x10^6)	692726	-32.33%
GR	×	83	83	-	-	-
θ_{gen_init} [rad]	×	0	0	0 (6.283)	-	-
ω_{gen_init} [rad/s]	×	80	80	-	-	-
θ_{tur_init} [rad]	×	0	0	0 (6.283)	-	-
ω_{tur_init} [rad/s]	×	1	1	-	-	-

✓ denotes parameters estimated and × denotes parameters not estimated

With the addition of the DFIG model, another transfer function was added to the system. This together with the feedback loops in the system causes the response to changes in parameter values of the system to become more unpredictable without proper analysis of the total transfer function. The probability of the system becoming unstable for certain parameter

values also increased. To investigate whether this caused the estimation to produce inaccurate results the boundaries were tightened even more.

5.4.3.3 Case study 2 – Gearbox parameter values adjusted boundary values

The second case study was performed with tightened boundaries. The upper boundaries were set to be about 50% higher than the actual values and the lower boundaries to be about 50% lower than the actual values. The initial guess values were set equal to the values of the lower boundaries. The boundary values and initial guess values used for the estimation process are given in the Lower (Upper) Bound and Initial Guess column of Table 5-23 respectively. The results obtained from performing the parameter estimation process using this configuration are shown in Table 5-23. These results were obtained after 32 iterations that took 341 minutes to complete.

Table 5-23: Parameter estimation results of second case study performed on the complete wind turbine system model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Lower (Upper) Bound	Estimated Value	Percentage Error
J_{gen} [kg.m^2]	✓	90	45	45 (135)	98.2	-9.17%
J_{tur} [kg.m^2]	✓	4.95 x10^6	2.475x10^6	2.475 x10^5 (7.43 x10^7)	48.88 x10^6	1.24%
K [Nm/rad]	✓	114 x10^6	57 x10^6	57 x10^6 (1.72 x10^8)	112.52 x10^6	1.29%
D [Nm.sec/rad]	✓	755658	3.77 x10^5	3.77 x10^5 (1.13x10^6)	842982	-11.56%
GR	×	83	83	-	-	-
θ_{gen_init} [rad]	×	0	0	-	-	-
ω_{gen_init} [rad/s]	×	80	80	-	-	-
θ_{tur_init} [rad]	×	0	0	-	-	-
ω_{tur_init} [rad/s]	×	1	1	-	-	-

✓ denotes parameters estimated and × denotes parameters not estimated

These results show that the generator parameters can be estimated accurately with the errors of the stiffness coefficient and the inertia of the turbine blades below 1.5% and the damping coefficient and generator inertia below 12%.

5.4.3.4 Case study 3 – Generator parameter values

To investigate whether generator parameter values could be estimated within the complete wind turbine system model the model was excited by applying a stator voltage signals with

steps in its amplitudes while supplying a constant wind speed input signal. The A-phase signal of the stator voltage used for this case study is shown in Figure 5-39 and Figure 5-40 shows a zoomed-in section of all three phases. The output current and torque signals used for the cost function were generated by performing a forward simulation on the complete system model. The power coefficient parameter values of the turbine blade model for the simulation are shown in Figure 4-2. The remaining parameter values of the turbine blade model are provided in Table 5-19 and the parameter values of the gearbox model are given in Table 5-24. The parameter values of the generator model are given in the Actual Values column of Table 5-25. The output torque signal of the generator model is shown in Figure 5-41, with Figure 5-42 showing a zoomed-in section of the signal to illustrate the transient response caused by the step in the voltage amplitude.

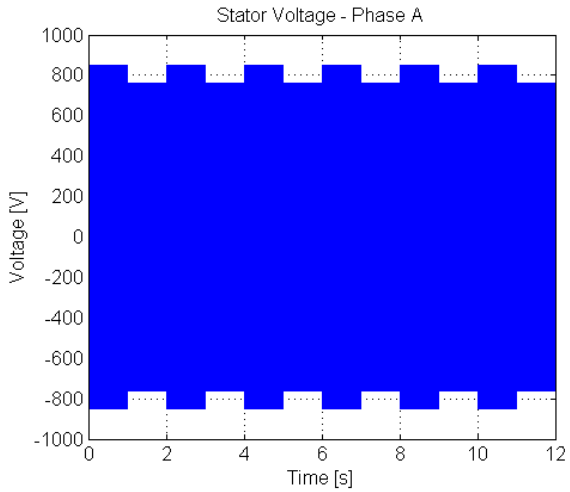


Figure 5-39: Phase A stator voltage input to ABC DFIG model for third case study performed on complete wind turbine system model.

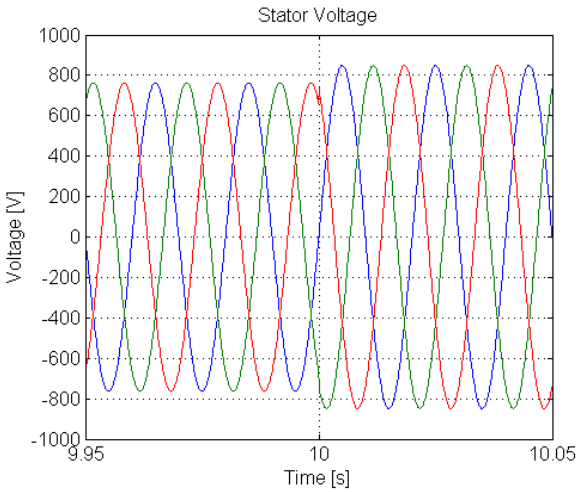


Figure 5-40: Zoomed in portion of stator voltages shown in Figure 5-39.

Table 5-24: Gearbox model parameter values for study 3 of complete wind turbine system model.

Parameter	Description	Values
J_{gen}	Generator Moment of Inertia	90 kg.m^2
J_{tur}	Turbine Moment of Inertia	4.95 Mkg.m^2
K	Shaft Stiffness coefficient	114 MNm/rad
D	Shaft Damping coefficient	756 kNm.s/rad
GR	Gear Ratio	83

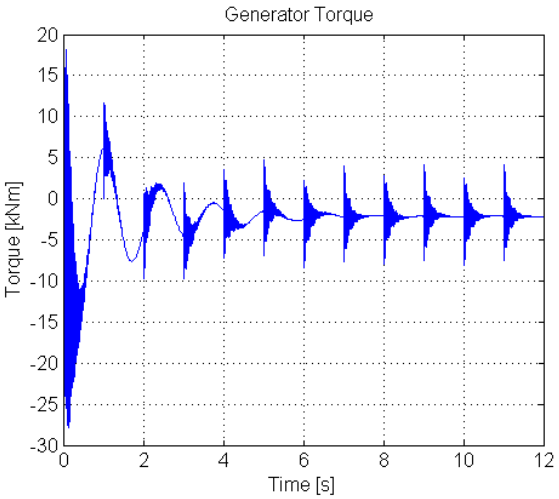


Figure 5-41: Generator torque output of ABC DFIG model for case study 3 of complete wind turbine system model.

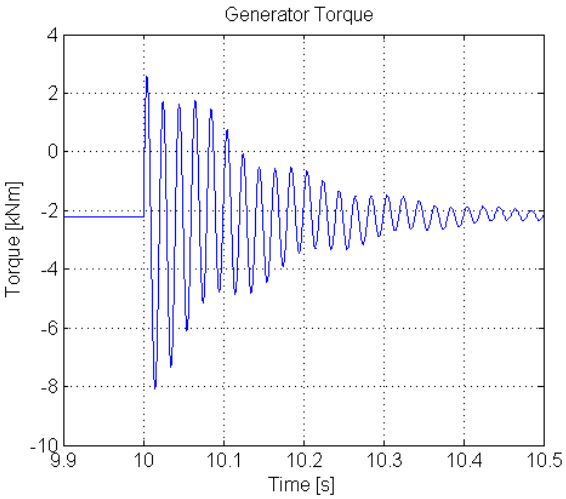


Figure 5-42: Zoomed in portion of generator torque output of ABC DFIG model shown in Figure 5-41.

The parameter estimation process was performed using these input signals and parameter values. The torque and current output signals were used for the cost function. These output signals were windowed with a window spanning from 10 s to 12 s. The generator parameters were all set to be estimated as well as the initial value of the angular position state. The initial values of the current states were assigned their actual values. Table 5-25 presents the results obtained after 40 iterations that took 286 minutes to complete.

Table 5-25: Parameter estimation results of third case study performed on the complete wind turbine system model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00029921	0.0001	0.000331436	-10.77%
L_s [H]	✓	0.00040744	0.0001	0.000376073	7.70%
M [H]	✓	0.016	0.001	0.015963753	0.23%
P	×	4	4	-	-
R_r [Ω]	✓	0.0089	0.001	0.008901834	-0.02%
R_s [Ω]	✓	0.005	0.001	0.004999697	0.01%
I_{as_init} [A]	×	1	1	-	-
I_{bs_init} [A]	×	1	1	-	-
I_{cs_init} [A]	×	1	1	-	-
I_{ar_init} [A]	×	1	1	-	-
I_{br_init} [A]	×	1	1	-	-
I_{cr_init} [A]	×	1	1	-	-

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
θ_{r_init} [rad]	✓	1	0.5	1.017449949	-1.74%

✓ denotes parameters estimated and × denotes parameters not estimated

These results show that the mutual inductance and resistances were estimated with close to 100% accuracy, the winding inductances were less accurate but the errors were still below 11%.

5.4.3.5 Case study 4 – Gearbox and generator parameter values

A final case study was performed on the complete wind turbine system model to investigate whether a combination of generator and gearbox parameter values could be estimated together. For this estimation the model was excited by both a step in wind speed and steps in the stator voltages. The stator voltage signals used for the estimation are the same signals used for the third case study performed on the complete system model, shown in Figure 5-39, while the input wind speed signal used is shown in Figure 5-43. These input signals were used to obtain the output signals used for the estimation by again performing a forward simulation on the complete system. The power coefficient parameter values used for this simulation are shown in Figure 4-2. The remaining parameter values of the turbine blade model are provided in Table 5-19. The parameter values of the gearbox and generator models are given in the Actual Value column of Table 5-27. The initial values of the states used for the simulation are provided in Table 5-26.

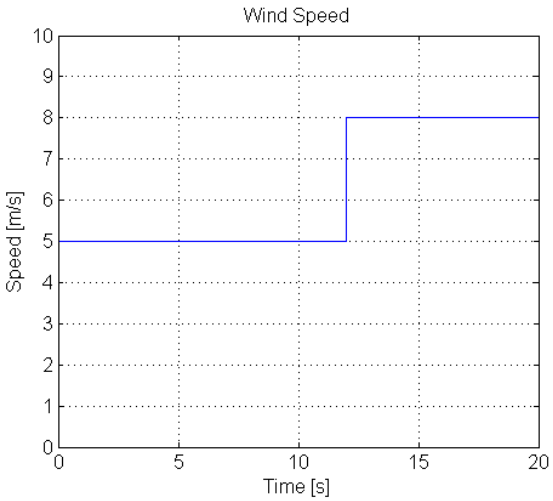


Figure 5-43: Input wind speed signal for parameter estimation case study 4 of complete wind turbine system.

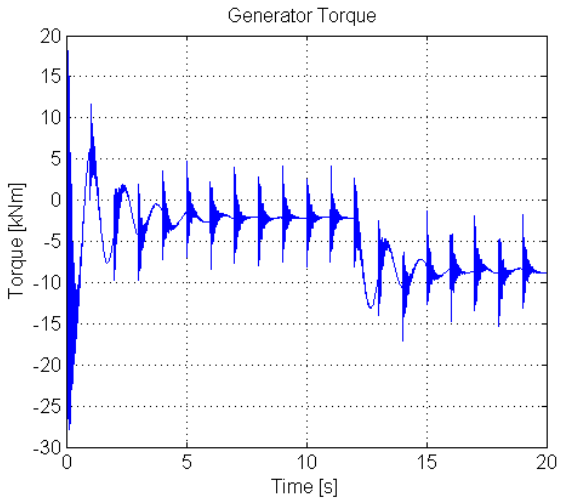


Figure 5-44: Generator torque output of ABC DFIG model for parameter estimation case study 4 of complete wind turbine system.

Table 5-26: Initial state values for case study 4 of complete wind turbine system model.

State	Initial Value	State	Initial Value	State	Initial Value
I_{as_init} [A]	1	I_{br_init} [A]	1	θ_{gen_init} [rad]	0
I_{bs_init} [A]	1	I_{cr_init} [A]	1	θ_{tur_init} [rad]	1
I_{cs_init} [A]	1	θ_{r_init} [rad]	1	ω_{tur_init} [rad/s]	0
I_{ar_init} [A]	1	ω_{gen_init} [rad/s]	80		

The output signals were obtained using these input signals, parameter values and initial state values. Figure 5-44 shows the generator torque output signal, Figure 5-45 shows the three phase rotor current output signals and Figure 5-46 shows the A phase of the stator current output signal. These signals were windowed with a window spanning form 11 to 18 s and then used for the cost function.

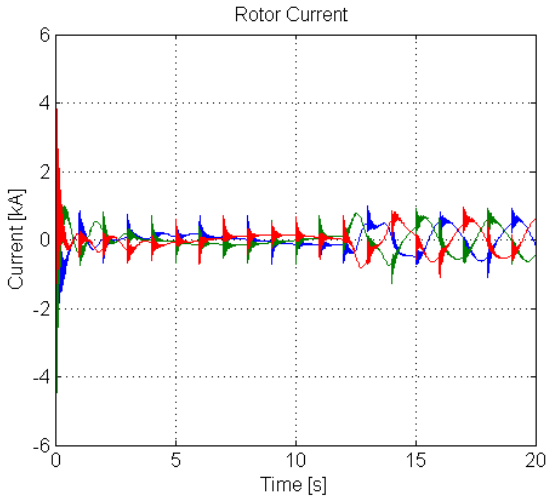


Figure 5-45: Generator rotor current output of ABC DFIG model for parameter estimation case study 4 of complete wind turbine system.

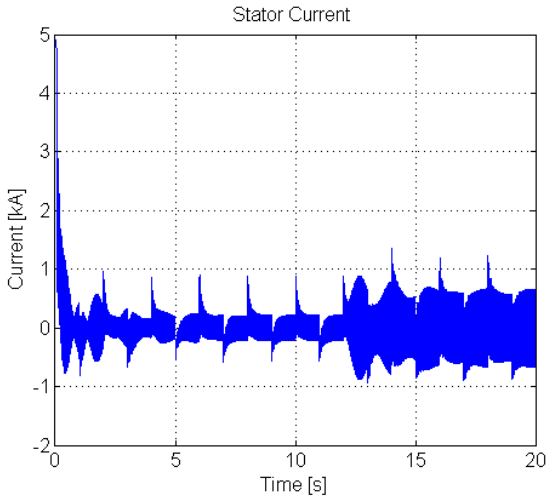


Figure 5-46: Generator stator current output of ABC DFIG model for parameter estimation case study 4 of complete wind turbine system.

Initial attempts to estimate all the gearbox parameter, generator parameter and initial state values again produced the "*Out of memory*" error message. To overcome this problem the number of parameters and initial values of states to be estimated were reduced. All the generator parameters, except the number of poles, were set to be estimated together with the moment of inertia parameter of the gearbox. The remaining parameter and initial state values were assigned their actual values and set not to be estimated. Table 5-27 presents the results obtained after 47 iterations that took 499 minutes to complete. These results shows that a combination of gearbox and generator parameter values can be estimated simultaneously with accurate results

Table 5-27: Parameter estimation results of fourth case study performed on the complete wind turbine system model.

	Parameter/ Initial State	Estimation Flag	Actual value	Initial Guess	Estimated Value	Percentage Error
Gearbox Model	D [Nm.sec/rad]	×	755658	755658	-	-
	GR	×	83	83	-	-
	J_{gen} [kg.m^2]	×	90	90	-	-
	J_{tur} [kg.m^2]	✓	4.95 x10^6	2.48 x10^6	4.9503 x10^6	-0.01%
	K [Nm/rad]	×	114 x10^6	114 x10^6	-	-
ABC DFIG Model	L_r [H]	✓	0.00029921	0.0001	0.000303	-1.36%
	L_s [H]	✓	0.00040744	0.0001	0.000402	1.17%
	M [H]	✓	0.016	0.001	0.159235	0.48%
	P	×	4	4	-	-
	R_r [Ω]	✓	0.0089	0.001	0.008903	-0.04%
	R_s [Ω]	✓	0.005	0.001	0.005000	-0.00%

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Overview

This chapter summarises the results of the study and presents the conclusions. This is followed by recommendations for further studies.

6.2 Conclusions

6.2.1 Introduction

As mentioned in Chapter 2 this project was initiated to investigate whether the values of system parameters of a wind turbine system can be obtained through parameter estimation. This section provides an overview of the project objectives and continues to summarise the results from which conclusions are drawn.

This project is comprised of two main objectives. The first is to develop a toolbox for a wind turbine system to be used for the parameter estimation process. The second objective is to perform an introductory study to determine which parameters of the wind turbine system can be readily estimated.

6.2.2 Development of a wind turbine system toolbox for parameter estimation application

A literature review was performed to determine which wind turbine system topology to use as the model for this study. The following topologies were considered.

- Fixed-speed generator topology
- Two-speed induction generator topology
- Variable rotor resistance generator topology
- Generator with fully-rated converter topology
- Generator with direct drive and fully-rated converter topology
- Double-fed induction generator topology
- Directly coupled synchronous generator with variable gearbox topology

From the review it was concluded that the market is appearing to be moving in the direction of the generator with fully-rated converter topology and especially the direct drive generator

with fully-rated converter. The Double-Fed Induction Generator (DFIG) topology is, however, currently the most common topology when considering the wind turbine systems available from the leading manufacturers. The decision was made to model the components of a fix-speed generator topology, namely the turbine blades, gearbox and generator, for this study. With future studies in mind, the generator component was modelled as a DFIG, but was used as an induction generator for this study by applying 0 V voltages to the rotor windings.

Since the models were developed for use with parameter estimation processes, they were required to be efficient, as they are simulated numerous times during these processes. The mathematical models were derived for each of the components. In an attempt to minimise the simulation time, these models were then implemented as C-code S-function models. The DFIG was initially modelled in the ABC reference frame, but to improve the simulation time it was also modelled in the DQ reference frame.

All four models, i.e., the turbine blade, gearbox, ABC DFIG and DQ DFIG, were validated and their performance evaluated by comparing them to existing Simulink block models developed by the Institute of Energy Technology at the University of Aalborg. By individually supplying the models with identical test input signals and parameter values and comparing the outputs obtained, all four models were proven to be accurate. The components were then connected to form a wind turbine system and compared to the corresponding existing block model system, again using identical test input signals and parameters for both models. The results obtained from this comparison further verified the accuracy of the models.

The performances of the models were evaluated by performing test simulations on the models and comparing the simulation times to that of the corresponding existing block models. This was done for the individual models, as well as the complete system. The test simulations performed on the turbine blade model were done using both generated wind speed data and real wind speed data. The derived turbine blade model showed a reduction in simulation time of about a factor 3.

The gearbox test simulation was performed with generated turbine blade torque and generator torque input signals. This test simulation showed a reduction in simulation time varying between 30% and 40%.

To evaluate the performance of the generator models, the models were simulated with different input angular velocities ranging from 0 rpm to 3000 rpm and the simulation times recorded. The simulation times of each model were averaged and compared to the average simulation times of the existing models. A reduction of 76% was obtained when comparing the ABC models and a reduction of about 68% was obtained when comparing the DQ models. When comparing the DQ model to the ABC model, a reduction of about 92% was obtained.

These results show that the simulation times of all the individual derived models are significantly shorter than that of the existing Simulink block models, with the gearbox model showing the smallest reduction. This small reduction in simulation time is attributable to the fact that Simulink is very well optimised for solving differential equations and the model of the two-mass gearbox simply consists of two differential equations.

The individual models were then connected to form wind turbine system models and test simulations were performed to compare their simulation times. These systems were simulated with two sets of wind input data, i.e., generated wind speed and real wind speed as well as for two maximum simulation step sizes, i.e., 1 ms and 0.5 ms. Comparing the simulation times of the system with the ABC DFIG as generating element showed a reduction of about 60% for both maximum step sizes. The system with the DQ DFIG as generating element and configured for a maximum step size of 0.5 ms showed a reduction in simulation time of about 50%. With the maximum step configured as 1 ms, the reduction in simulation time was only 29%. This relatively low reduction in simulation time was attributable to an oscillation in the DQ DFIG model caused by a numerical instability problem. The results obtained from the performance evaluation of the wind turbine system model showed a significant overall reduction, especially when considering that these models are simulated numerous times during the parameter estimation process.

With the models shown to be accurate and performing efficiently, masks were created for the models to make them more user-friendly. These masked models were all compiled into a Simulink library.

From this it is concluded that the objective of developing a wind turbine system toolbox for parameter estimation application was achieved.

6.2.3 Introductory study to determine which parameters of the wind turbine system can be readily estimated

A literature review was performed to gain insight into the parameter estimation process. The literature review initially focussed on the general process of system identification, which is used when the model of a system is not known. It carries on to the parameter estimation process which is used when the model is known and the parameters of the model mostly have physical meaning. All the components of the wind turbine system can be modelled mathematically; therefore, parameter estimation is required for this study. Different cost functions were reviewed from which it was concluded that the least-square cost function is best suited for this study. Four optimisation algorithms, namely Newton method, Gauss-Newton method, Trust-region method and Levenberg-Marquardt method were reviewed to gain insight into the process of optimisation algorithms used for parameter estimation. The literature review concluded with a quick overview of available software products with the ability to implement mathematical models and have optimisation algorithms for parameter estimation. It is concluded that MATLAB is best suited for this study. MATLAB, together with its extension Simulink, provides all the functionality required for this study in a manner that is ideal for the research environment.

Parameter estimation case studies were performed on the individual models of the system followed by case studies performed on combined model topologies to determine which parameters of the models can be readily estimated and what constraints apply. These case studies were conducted with output data obtained from performing forward simulation on the models with known input signals, parameter values and initial state values.

Two parameter estimation case studies were performed on the turbine blade model. Since the control system of the pitch control was not modelled, the blade pitch angle was fixed for these case studies. The first case study was conducted using a generated signal as the wind speed input. The second case study was performed using real wind data from the Gorgonio wind measurement site [78]. Of the four parameters of the turbine blade model, namely the blade length, cut-in wind speed, cut-out wind speed and power coefficient, only the power coefficient requires to be estimated. Due to a process limitation of the operating system, only the 5° pitch angle column of the power coefficient matrix was estimated. The excited elements of the power coefficient matrix were estimated with 100% accuracy for both case studies. From these results it is concluded that the excited elements of the power coefficient

parameter can be accurately estimated, within the constraints of the process limitation of the operating system.

Three parameter estimation case studies were performed on the gearbox model. These case studies were all conducted using a constant input as generator torque input and a generated signal as turbine blade torque input. The first case study was performed with all the generator parameter values, except the gearbox ratio, i.e., turbine blade inertia, generator inertia, damping coefficient and stiffness coefficient, as well as all the initial values of the states, i.e., angular velocities and angular positions, set to be estimated. The results of the first case study together with additional estimations performed using different initial values for the states showed that this configuration converged to different local minimums depending on the initial value of the states. Before the second case study was performed the parameters and states were considered. The decision was made to provide the estimation with the actual initial values of the angular velocity states, since angular velocity is relatively easy to measure. The results obtained from the second case study showed that all the gearbox parameter values can be estimated accurately with an error below 2%. For the firsts two case studies performed the entire response of the output signals were used, including the start-up transient. This start-up transient will not be present in actual data. Therefore, the third case study was performed without this start-up transient to investigate whether the generator parameters could still be estimated accurately. The same configuration was used for this case study as for the second case study, except for the output signals being windowed to remove the start-up transients, before it was passed to the cost function. The results obtained showed that the gearbox parameter values can still be estimated accurately with an error less than 2.5%. Considering the results of these case studies, it is concluded that the gearbox parameter values can be estimated accurately if the actual initial values of the angular velocity states are provided.

Six parameter estimation case studies were performed on the ABC DFIG model. All six were conducted with the generator acting as an induction generator by supplying the rotor winding with 0 V voltages. For the first four case studies the stator windings were supplied with 220 V 50 Hz balanced three phase voltages and a generated step signal was supplied as angular velocity input. The first case study was performed with all the generator parameter values except the number of poles, i.e., the stator winding resistance, rotor winding resistance, stator winding inductance, rotor winding inductance and mutual inductance set to

be estimated. All the initial values of the states, i.e., stator currents, rotor currents and angular position were provided with their actual values and were not estimated. These estimation results showed that the generator parameters can be estimated accurately with errors below 3.5% for this configuration. The second case study was performed with the same configuration as the first, but random values were assigned to the initial values of all the states. This resulted in the parameter values not being estimating accurately with errors ranging between 21% and 107%. A window was then applied to the output signals to investigate whether the start-up transient might have be causing the bad estimation results, but this produced similar results. Considering the states of the generator model, it was found that the angular position plays a major part in the model's equations. The third case study was, therefore, performed with the same configuration and initial values as the second but with the initial values of angular position state also set to be estimated. This configuration produced accurately estimated generator parameter values with errors below 3.5%. The fourth case study was performed with a different set of generator parameter values and a window was applied to the outputs to remove the start-up transient caused by the simulation. This case study produced results with the parameter values accurately estimated with errors below 1.5%. From these four case studies it was concluded that the parameter estimation process for the ABC DFIG model excited by the angular velocity step is not sensitive to the initial values of the current states, but it is sensitive to the initial value of the angular position state. For the remaining two case studies, performed on the ABC DFIG model, the angular velocity input was supplied with a constant value while the stator windings were supplied by voltages with steps in their amplitudes. These two case studies were performed with the same set of generator parameter values as the fourth case study. The results of these case studies showed that the parameter estimation process using this configuration was sensitive to the initial values of the current states. By assigning the initial values of the current states their actual values, the generator parameter values were estimated accurately with errors below 4%. From all these case studies performed on the ABC DFIG model it is concluded that the generator parameter values can be estimated accurately, within certain constraints, for both the cases where the parameters are excited by a step in angular velocity and by steps in the stator voltages.

Three case studies were performed on the DQ DFIG model. All three were performed with the start-up transient windowed out. The first two case studies were performed with the same

voltages and angular velocity input signals as for the first four case studies of the ABC DFIG model. The first case study was performed with the generator parameter values and initial value of angular position state set to be estimated. This configuration produced accurately estimated generator parameter values with errors below 13%. Two additional case studies were performed based on the first case study. One with the actual initial state values provided and the other with the data window removed, these produced results with similar errors and higher errors respectively. The second case study was performed with the same configuration as the first case study but used a different set of parameter values. This case study produced more accurate results with errors below 9%. The third case study and two additional case studies were performed with the same changing stator voltages and constant angular velocity input signals as used for the last two case studies of the ABC DFIG model. The results obtained from these case studies showed that the generator parameter values were estimated accurately, when the actual initial values of the current states were supplied, with errors below 6%. Thereby, confirming that the DQ DFIG model excited by the stator voltages is also sensitive to the initial values of the current states. The results obtained from these three case studies together with the additional case studies show that the generator parameters can be accurately estimated, within certain constraints. When the model is excited through the angular velocity the initial value of the angular position state are required to be estimated. When the model is excited through the stator voltages the initial values of the current states are also required to be estimated. Although all the results of the case studies show that the generator parameter can be estimated accurately, with errors below 13%, these results are less accurate than those of the ABC DFIG model. From this it is concluded that the cost function is less sensitive to the generator parameters in the case of the DQ model than in the case of the ABC model. Further investigation is required to determine the cause of the cost function of the DQ DFIG being less sensitive to the parameters than that of the ABC DFIG model.

The first of the combined model topology parameter estimation case studies were performed on the topology consisting of the turbine blade model and gearbox model. The gearbox parameter values together with initial values of the angular position states were set to be estimated. The remaining parameter and initial state values were set not to be estimated and assigned their actual values. The first estimation produced inaccurate gearbox parameter values, whereas the second with boundary values in the range of the parameters' typical

values produced accurate results, with errors below 9%. From these results it is concluded that by connecting these models together with the feedback loops the sensitivity of the system to its parameter values changes. Certain parameter values could also lead to the system becoming unstable.

The second set of combined model topology parameter estimation case studies were performed on the complete wind turbine system model consisting of the turbine blade model, gearbox model and DFIG model. Four case studies were performed on this model. The first two case studies investigated whether the gearbox parameter values could be estimated, whereas the third case study investigated whether the generator parameter values could be estimated. The fourth case study investigated whether a combination of gearbox and generator parameter values could be estimated. From the results obtained for the first two case studies performed on the complete wind turbine system model, excited by a step in angular velocity, the conclusion was drawn that the adding of the DFIG model further changes the sensitivity of the system model to parameter values and also increases the probability of the model becoming unstable. The second case study showed that these problems could be overcome by applying sufficient boundaries for the parameter values. Applying these boundaries produce accurately estimated gearbox parameter values with errors below 12%. This result showed that the gearbox parameter values can be accurately estimated within the complete wind turbine system model when excited by a step in angular velocity. The third case study showed that the generator parameter values could be estimated accurately within the complete wind turbine system model when excited through the stator voltages, with errors below 11%. The final case study was performed with the parameters of the model excited by both a step in angular velocity and steps in the amplitudes of the stator voltages. The results showed that a combination of gearbox and generator parameter values can be estimated simultaneously with accurate results.

From all the case studies performed, on the individual models and combine model topologies, it is concluded that the objective of performing an introductory study to determine which parameters of the wind turbine system can be readily estimated was successfully completed. The case studies performed for the introductory study proves the principle of performing parameter estimation of C-code S-function models in Simulink. Further studies can now be applied to perform project specific parameter estimation case studies. The following section

provides recommendations for future research that could be conducted for the expanding of the models and improving of the parameter estimation process.

6.3 Recommendations

Considering the modelling of the wind turbine system that forms part of the first objective, further research can be conducted by expanding the model:

- The turbine blade model can be expanded to model the tower shadow.
- The infinite bus model of the electrical grid can be replaced by a more complex grid model.
- The generator model can be expanded by modelling the deep-bar effect and saturation.
- The gearbox model can be expanded by modelling backlash of the gears.
- The power electronic converter can be modelled.
- The control system of the system can be modelled.

Further research should be conducted on the implemented DQ DFIG model to determine the cause of the oscillation discussed in Chapter 3.

Considering the case studies performed for the second objective of the project, which is an introductory study to determine which parameters of the wind turbine system can be readily estimated, it is clear that there exists an endless amount of possible case studies that can be performed. The parameter estimation processes performed for this study made use of only one set of parameter data for the gearbox model as well as for the turbine blades model and two parameter data sets for the DFIG model. Further studies can be done using project specific parameter data sets. The study was also performed with limited input signals. These could also be replaced with different signals for further case studies.

Further research should be conducted on the models by analysing the individual models and the complete wind turbine system model using the technical computing software Mathematica. Using this software, the open and closed loop transfer function can be analysed to determine the model's sensitivity to parameter changes for different frequencies in both the open and closed loop cases. By so doing, insight would be gained into the model's response to changes made by the parameter estimation process and required input signals for existing parameters. From these analyses it should be found that by changing some parameter values the complete frequency response of the system changes, whereas

changes to other parameter values influence only the high or low frequency response. Hereby, it could be determined which parameters are excited by which frequencies. Using this information, the two step approach to parameter estimation can be applied. For the two step approach the input signals are passed through a low pass filter and only the parameters excited by low frequencies are set to be estimated. The signals are then passed through a high pass filter with the parameter values estimated from the first estimation fixed and only the remaining parameter values are estimated.

Once sufficient insight is gained into the model of the wind turbine system as well as the parameter estimation process, further research should be conducted using data obtained for a real wind turbine system.

7 REFERENCES

[1] World Wind Energy Association, "World wind energy report 2010," World Wind Energy Association, Bonn, Germany, 2010.

[2] ENERCON Gmbh, available at: http://www.enercon.de/en-en/Windenergieanlagen. htm, accessed March 2011.

[3] European Wind Energy Association, "Wind energy factsheets," European Wind Energy Association, Belgium, Germany, 2010.

[4] Department of Energy: Republic of South Africa, available at: http://www.energy.gov.za/files/esources/renewables/r_wind.html, accessed April 2011.

[5] National Energy Regulator of South Africa, "NERSA Consultation Paper Renewable Energy Feed-in Tariff Phase II," 2009.

[6] Department of Minerals and Energy, "White Paper on Renewable Energy," November 2003.

[7] M. Pöller, *Grid Integration of Wind Energy in Western Cape - Results of Feasibility Studies.* Gomaringen, Germany: DIgSILENT GmbH, 2009.

[8] Eskom, *Grid Code Requirements for Wind Turbines Connected to Distribution Or Transmission Systems in South Africa.* South Africa: NERSA, 2011.

[9] Department of Energy (DoE), "Integrated Resource Plan for Electricity 2010-2030 - Revision 2 - Final Report," 2011.

[10] A. D. Hansen, P. Sørensen, F. Blaabjerg and J. Becho, "Dynamic modelling of wind farm grid interaction," *Wind Eng,* vol. 26, pp. 191-210, 2002.

[11] P. Sørensen, A. Hansen, L. Janosi, J. Bech and B. Bak-Jensen, "Simulation of interaction between wind farm and power system," *RISØ Report R-1281 (EN)(Online: www.Risoe.dk/rispubl/VEA/veapdf/ris-r-1281.Pdf), Roskilde, Denmark,* 2001.

[12] South African National Energy Research Institute, available at: http://www.saneri.org.za, accessed March 2011.

[13] S. Santoso and H. T. Le, "Fundamental time–domain wind turbine models for wind power studies," *Renewable Energy,* vol. 32, pp. 2436-2452, 11, 2007.

[14] P. Gipe, *Wind Power.* London: James & James, 2004.

[15] S. Mathew, *Wind Energy: Fundamentals, Resource Analysis and Economics.* Berlin: Springer, 2006.

[16] American Wind Energy Association, available at: http://www.awea.org/, accessed June 2010.

[17] F. A. Farret and M. G. Simões, *Integration of Alternative Sources of Energy.* Piscataway: IEEE Press, 2006.

[18] R. Gasch and J. Twele, *Wind Power Plants: Fundamentals, Design, Construction and Operation.* Berlin: Solarpraxis AG, 2002.

[19] V. Quaschning, *Understanding Renewable Energy Systems.* London: Earthscan, 2006; 2005.

[20] Wind Pacific (Aust) Pty Ltd, available at: http://www.windpacific.com/turbines.html, accessed March 2011.

[21] M. R. Patel, *Wind and Solar Power Systems.* Boca Raton, Fla.: CRC Press, 1999.

[22] T. Wizelius, *Developing Wind Power Projects: Theory and Practice.* London: Earthscan, 2007.

[23] Suzlon Energy, at: http://www.suzlon.com/, accessed March 2011.

[24] M. Verdonschot, "Modeling and Control of wind turbines using a Continuously Variable Transmission," 2009.

[25] Siemens AG, available at: http://www.energy.siemens.com/hq/en/power-generation/renewables/wind-power/, accessed March 2011.

[26] Vestas Wind Systems A/S, available at: http://www.vestas.com/en/wind-power-plants/procurement/turbine-overview.aspx#/vestas-univers, accessed March 2011.

[27] WinWinD Ltd, available at: http://www.winwind.com/, accessed March 2011.

[28] A. Ragheb and M. Ragheb, "Wind turbine gearbox technologies," in *Nuclear & Renewable Energy Conference (INREC), 2010 1st International*, 2010, pp. 1-8.

[29] GE Energy AG, available at: http://www.gepower.com/prod_serv/products/wind_turbines/en/index.htm, accessed March 2011.

[30] Nordex SE, available at: http://www.nordex-online.com/en/produkte-service/wind-turbines.html ,accessed March 2011.

[31] Repower Systems AG, available at: http://www.repower.de/produkte/windenergieanlagen/?L=1, accessed March 2011.

[32] ENVISION Energy Ltd, available at: http://www.envisioncn.com/index.aspx, accessed March 2011.

[33] Eviag AG, available at: http://www.envisioncn.com/index.aspx, accessed March 2011.

[34] Fuhrländer AG, available at: http://www.fuhrlaender.de/index_en.php?menu_gewaehlt=1, accessed March 2011.

[35] J. F. Manwell, J. G. McGowan and A. L. Rogers, *Wind Energy Explained.* UK: Wiley & Sons LTD, 2002.

[36] DigSILINT, *Dynamic Modelling of Doubly-Fed Induction Machine Wind-Generators.* Gomaringen, Germany: DIgSILENT GmbH, 2003.

[37] T. Burton, *Wind Energy: Handbook.* Chichester: J. Wiley, 2008; 2001.

[38] D. A. Spera, *Wind Turbine Technology: Fundamental Concepts of Wind Turbine Engineering.* New York, N.Y.: Asme, 1994.

[39] E. Muljadi and C. P. Butterfield, "Pitch-controlled variable-speed wind turbine generation," *Industry Applications, IEEE Transactions on,* vol. 37, pp. 240-246, 2001.

[40] M. Ragheb, *Control of Wind Turbines.* Champaign, USA: University of Illinois, 2009.

[41] L. Ljung, *System Identification: Theory for the User.* Upper Saddle River, NJ: Prentice Hall PTR, 1999.

[42] T. Soderstrom and P. Stoica, *System Identification.* New York, N.Y.: Prentice-Hall, 1989.

[43] R. Mehra, S. Mahmood and R. Waissman, "Identification of aircraft and rotorcraft aeroelastic modes using state space system identification," in *Proceedings of the 4th IEEE Conference onControl Applications* 1995, pp. 432-437.

[44] J. C. Bekker and H. J. Vermeulen, "Implementation of a wind turbine system as a native C-code MATLAB model for parameter estimation application," in *46th International Universities' Power Engineering Conference,* Soest, Germany, 2011, pp. 88.

[45] J. Nocedal and S. J. Wright, *Numerical Optimization.* New York: Springer, 1999.

[46] S. C. Chapra, *Applied Numerical Methods with MATLAB for Engineers and Scientists.* McGraw-Hill Higher Education, 2008.

[47] A. O. Converse, *Optimization.* United States of America: Holt, Rinehart and Winston, Inc., 1970.

[48] G. R. Walsh, *Methods of Optimization.* England: John Wiley & Sons, 1975.

[49] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms.* LibreDigital, 2006.

[50] A. R. Conn, N. I. M. Gould and P. L. Toint, *Trust-Region Methods.* Society for Industrial Mathematics, 2000.

[51] R. Fletcher, *Practical Methods of Optimization.* Chichester: John Wiley & Sons, 1987.

[52] A. B. Levy, *The Basics of Practical Optimization.* Society for Industrial Mathematics, 2009.

[53] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming.* Springer Verlag, 2008.

[54] J. Zhang, *Iterative Methods for Optimization.* USA: Purdue University, 2008.

[55] Å. Björck, *Numerical Methods for Least Squares Problems.* Society for Industrial Mathematics, 1996.

[56] D. J. Higham, "Trust Region Algorithms and Timestep Selection," *SIAM Journal on Numerical Analysis,* vol. 37, pp. 194-210, 2000.

[57] Y. Yuan, "A review of trust region algorithms for optimization," in *Proceedings of the Fourth International Congress on Industrial & Applied Mathematics,* Edinburgh, 1999.

[58] Y. K. Singh and B. Chaudhuri, *Matlab Programming.* PHI Learning Pvt. Ltd., 2007.

[59] MathWorks, available at: http://www.mathworks.com/, accessed June 2011.

[60] MathWorks, "Simulink - User's Guide R2010a," 2010.

[61] P. Bojorklund, J. Pan, C. Yue and K. Srivastava, "A new approach for modeling complex power systems components in different simulation tools," in *16th Power Systems Computation Conference (PSCC2008),* Glasgow, Scotland, 2008.

[62] V. Couvert, S. Steer and M. Baudin, "Optimization in Scilab," 2009.

[63] MathWorks, "Optimazation Toolbox - User's Guide R2011b," 2011.

[64] J. W. Eaton, D. Bateman and S. Hauberg, *GNU Octave - A High-Level Language for Numerical Computations.* UK: Network Theory Limited, 2008.

[65] M. R. Patel, *Wind and Solar Power Systems - Design, Analysis, and Operation.* Boca Raton, Florida: CRC, 2006.

[66] F. Iov, A. D. Hansen, P. Sørensen and F. Blaabjerg, "Wind turbine blockset in matlab/simulink," *Institute of Energy Technology, Aalborg University,* 2004.

[67] P. Krause, O. Wasynczuk and S. Sudhoff, *Analysis of Electric Machinery and Drive Systems,* 2002.

[68] P. Pillay and V. Levin, "Mathematical models for induction machines," in *Industry Applications Conference, 1995. Thirtieth IAS Annual Meeting, IAS '95., Conference Record of the 1995 IEEE,* 1995, pp. 606-616 vol.1.

[69] N. N. Hancock, *Matrix Analysis of Electrical Machinery.* Oxford: Pergamon, 1964.

[70] M. G. Say, *Alternating Current Machines.* London: Pitman, 1976.

[71] D. O'Kelly and S. Simmons, *Introduction to Generalized Electrical Machine Theory.* London: McGraw-Hill, 1968.

[72] R. H. Park, "Two-reaction theory of synchronous machines generalized method of analysis-part I," *American Institute of Electrical Engineers, Transactions of the,* vol. 48, pp. 716-727, 1929.

[73] M. Y. Uctug, I. Eskandarzadeh and H. Ince, "Modelling and output power optimisation of a wind turbine driven double output induction generator," *Electric Power Applications, IEEE Proceedings,* vol. 141, pp. 33-38, 1994.

[74] J. G. Slootweg, H. Polinder and W. L. Kling, "Dynamic modelling of a wind turbine with doubly fed induction generator," in *Power Engineering Society Summer Meeting, 2001. IEEE,* 2001, pp. 644-649 vol.1.

[75] F. Mei and B. C. Pal, "Modelling of doubly-fed induction generator for power system stability study," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE,* 2008, pp. 1-8.

[76] MathWorks, "Simulink 7 - Writing S-functions," 2008.

[77] S. Heier, *Grid Integration of Wind Energy Conversion Systems.* Chichester, England: Wiley, 2006.

[78] WindData.com, available at: http://www.winddata.com, accessed March 2010.

[79] MathWorks, "MATLAB - Programming Fundamentals," 2011.

[80] ACCIONA Energy S.A., available at: http://www.acciona-energia.com/activity_areas.aspx, accessed March 2011.

[81] AREVA Wind GmbH, available at: http://www.areva-wind.com/index.php?id=8&L=1, accessed March 2011.

[82] Chipper Windpower Inc., available at: http://www.clipperwind.com/aboutus.html, accessed March 2011.

[83] E.N.O. Energy GmbH, available at: http://www.eno-energy.com/index.php?id=932&L=1, accessed March 2011.

[84] Gamesa Corporation Technologies, available at: http://www.gamesa.es/en/products-and-services/wind-turbines/, accessed March 2011.

[85] HZwindpower, available at: http://en.hzwindpower.com/Enproducts.asp, accessed March 2011.

[86] Mitsubishi Power Systems, available at: http://www.mpshq.com/products/wind_turbines/index.html, accessed March 2011.

[87] Northern Power Systems, available at: http://www.northernpower.com/utility-wind/the-product.php, accessed March 2011.

[88] STX Windpower B.V., at: http://www.stxwind.com/#, accessed March 2011.

APPENDIX A: WIND TURBINE SYSTEMS BY MANUFACTURER

Table A-1: Wind Turbine Systems by manufacturer [2, 20, 23, 25-27, 29-34, 80-88]

Manufacturer	Model Name	Power Rating [MW]	Number of Blades	Speed Type	Generator Type	Gearbox	Converter type
1. ENERCON	E33	0.33	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E48	0.8	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E53	0.8	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E44	0.9	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E70	2.3	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E82	2	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E82	2.3	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E82	3	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E101	3	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
	E126	7.5	3	VS	Multi-pole permanent magnet	n/a	Full-scale power converter
2. WinWind	WWD1	1	3	VS	Synchronous, permanent magnet	Planetary (1-stage planetary)	Full-scale IGBT power conversion
	WWD3	3	3	VS	Synchronous, permanent magnet	Planetary	Full-scale IGBT power conversion
3. Vestas	V82-1.65MW	1.65	3	FS	Asynchronous water-cooled	One planetary stage, two helical stages	n/a
	V80-2.0MW	2	3	VS	Double-fed asynchronous with slipring	Three-stage planetary/helical	Partial-scale power converter
	V80-2.0MW GridStreamer	2	3	VS	Permanent magnet generator	One planetary stage and two helical stages	Full-scale power converter
	V90-1.8/2.0MW	1.8/2	3	VS	Double-fed asynchronous with slipring	Three-stage planetary/helical	Partial-scale power converter
	V90-1.8/2.0MW GridStreamer	1.8/2	3	VS	Permanent magnet generator	One planetary stage and two helical stages	Full-scale power converter
	V100-1.8MW	1.8	3	VS	Double-fed asynchronous with slipring	Unknown	Partial-scale power converter
	V100-1.8MW GridStreamer	1.8	3	VS	Permanent magnet generator	One planetary stage and two helical stages	Full-scale power converter
	V90-3.0MW	3	3	VS	Double-fed asynchronous	Two planetary stages and one helical stage	Partial-scale power converter
	V90-3.0MW Offshore	3	3	VS	Double-fed asynchronous	two planetary stages and one helical stage	Partial-scale power converter
	V112-3.0MW	3	3	VS	Permanent magnet	4-stage planetary/helical	Full-scale power converter
	V112-3.0MW Offshore	3	3	VS	Permanent magnet	4-stage planetary/helical	Full-scale power converter
4. Nordex	N117	2.4	3	VS	Double-fed asynchronous	Combined spur/planetary gear or differential gearbox	Partial-scale IGBT power converter
	N100	2.5	3	VS	Double-fed asynchronous	Combined spur/planetary gear or differential gearbox	Partial-scale IGBT power converter
	N90	2.5	3	VS	Double-fed asynchronous	Combined spur/planetary gear or differential gearbox	Partial-scale IGBT power converter
	N80	2.5	3	VS	Double-fed asynchronous	Combined spur/planetary gear or differential gearbox	Partial-scale IGBT power converter
	N82	1.5	3	VS	Double-fed asynchronous	Three-stage design with one planetary and two spur gear stages	Partial-scale IGBT power converter
	N77	1.5	3	VS	Double-fed asynchronous	Three-stage design with one planetary and two spur gear stages	Partial-scale IGBT power converter
	N70	1.5	3	VS	Double-fed asynchronous	Three-stage design with one planetary and two spur gear stages	Partial-scale IGBT power converter
5. Siemens	SWT-2-3-82-VS	2.3	3	VS	Asynchronous	Three-stage planetary-helical design	Full-scale power converter
	SWT-2-3-93	2.3	3	VS	Asynchronous	Three-stage planetary-helical design	Full-scale power converter
	SWT-2-3-101	2.3	3	VS	Asynchronous	Three-stage planetary-helical design	Full-scale power converter
	SWT-3-6-107	3.6	3	VS	Asynchronous	Three-stage planetary-helical design	Full-scale power converter
	SWT-3-0-101	3	3	VS	Synchronous, PMG	n/a	Full-scale power converter
	SWT-3-6-120	3.6	3	VS	Asynchronous	Three-stage planetary-helical design	Full-scale power converter
6. GE Energy	GE1.5XLE	1.5	3	VS	Double-fed asynchronous	Three-stage planetary-helical design	Partial-scale IGBT power converter
	GE2.5XL	2.5	3	VS	Permanent magnet	n/a	Full-scale power converter
	GE4.0-110	4	3	VS	Permanent magnet	n/a	Full-scale power converter

Manufacturer	Model Name	Power Rating [MW]	Number of Blades	Speed Type	Generator Type	Gearbox	Converter type
7. Suzlon	S88-2.1MW	2.1	3	VerSlip	Asynchronous slip ring type induction generator	3 Stages (One planetary & Two helical)	n/a
	S82-1.5MW	1.5	3	VerSlip	Single speed induction generator with slip rings	3 Stages (One planetary & Two helical)	n/a
	S66-1.25MW	1.25	3	2S	Dual Speed Induction generator (Asynchronous)	3 Stages (One planetary & Two helical)	n/a
	S64-1.25MW	1.25	3	2S	Dual Speed Induction generator (Asynchronous)	3 Stages (One planetary & Two helical)	n/a
	S52-600kW	0.6	3	F'S	Single speed induction generator (Asynchronous)	3 Stages (One planetary & Two helical)	n/a
	S88 MarkII DFIG 2.25 MW	2.25	3	VS	Double-fed asynchronous with slipring	3 Stages (One planetary & Two helical)	Partial-scale power converter
8. Repower	6M	6.15	3	VS	Double-fed asynchronous	Three stage planetary/spur-gear system	Partial-scale PWM IGBT power converter
	5M	5.075	3	VS	Double-fed asynchronous	Two helical planetary stage and one spur stage	Partial-scale PWM IGBT power converter
	3.2M114	3.2	3	VS	Double-fed asynchronous	Three stage planetary/spur-gear system	Partial-scale PWM IGBT power converter
	3.4M104	3.4	3	VS	Double-fed asynchronous	Three stage planetary/spur-gear system	Partial-scale PWM IGBT power converter
	MM100	1.8	3	VS	Double-fed asynchronous	Combined planetary/spur wheel gearbox	Partial-scale PWM IGBT power converter
	MM92	2.05	3	VS	Double-fed asynchronous	Combined planetary/spur wheel gearbox	Partial-scale PWM IGBT power converter
	MM82	2.05	3	VS	Double-fed asynchronous	Combined planetary/spur wheel gearbox	Partial-scale PWM IGBT power converter
9. Hzwindpower	H56-850	0.85	3	VS	Synchronous	1 stage planet, 2 stage parallel axis	Full-scale power converter
	H82-2000	2	3	VS	Double-fed asynchronous	Unknown	Partial-scale power converter
	H87-2000	2	3	VS	Double-fed asynchronous	Unknown	Partial-scale power converter
	H93-2000	2	3	VS	Double-fed asynchronous	Unknown	Partial-scale power converter
10. Northern Power Systems	Northern Power 2.3	2.3	3	VS	Permanent magnet generator	n/a	Full-scale IGBT power converter
11. STXwind	STX72	2	3	VS	Multi-pole synchronous PM machine	n/a	Full-scale power converter
	STX82 1.5M	1.5	3	VS	Multi-pole synchronous PM machine	n/a	Full-scale power converter
	STX82 2.0M	2	3	VS	Multi-pole synchronous PM machine	n/a	Full-scale power converter
	STX93 2.0M	2	3	VS	Multi-pole synchronous PM machine	n/a	Full-scale power converter
12. Acciona Energy	AW-3000	3	3	VS	Double-fed asynchronous	3 Stages: 2 planetary/helical	Partial-scale PWM IGBT power converter
	AW-1500	1.5	3	VS	Double-fed asynchronous	3 Stages: 2 planetary/helical	Partial-scale PWM IGBT power converter
13. WindPacific	MY1.5Se	1.5	3	VS	Double-fed asynchronous	Three stage with 2 planetary gears	Partial-scale IGBT power converter
	MY1.5S	1.5	3	VS	Double-fed asynchronous	Three stage with 2 planetary gears	Partial-scale IGBT power converter
	SCD 2.5	2.5	2	VS	Permanent magnet generator	Two-stage planetary gear	Full-scale IGBT power converter
	SCD 2.75	2.75	2	VS	Permanent magnet generator	Two-stage planetary gear	Full-scale IGBT power converter
	SCD 3.0	3	2	VS	Permanent magnet generator	Two-stage planetary gear	Full-scale IGBT power converter
14. Aveva	M5000	5	3	VS	Permanent magnet generator	Step-planetary gear, helical	Full-scale IGBT power converter
15. Gamesa	G10X-4.5MW	4.5	3	VS	Permanent magnet generator	2 Planetary stages	Full-scale power converter
	G97-2.0MW	2	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel stages	Partial-scale PWM IGBT power converter
	G94-2.0MW	2	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel stages	Partial-scale PWM IGBT power converter
	G90-2.0MW	2	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel stages	Partial-scale PWM IGBT power converter
	G87-2.0MW	2	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel stages	Partial-scale PWM IGBT power converter
	G80-2.0MW	2	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel stages	Partial-scale PWM IGBT power converter
	G52-850kW	0.85	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel axis stages	Partial-scale PWM IGBT power converter
	G58-850kW	0.85	3	VS	Double-fed asynchronous	1 Planetary, 2 parallel axis stages	Partial-scale PWM IGBT power converter
16. Fuhrlander	FL 2500	2.5	3	VS	Double-fed asynchronous with slipring	3 Stage combined spur wheel/planet	Partial-scale power converter
	FL 1500	1.5	3	VS	Double-fed asynchronous	3 Stage combined spur wheel/planet	Partial-scale PWM IGBT power converter
	FL MD 70/77	1.5	3	VS	Double-fed asynchronous	3 Stage combined spur wheel/planet	Partial-scale PWM IGBT power converter
	FL 1250	1.25	3	FS	Asynchronous machine	3 Stage combined spur wheel/planet	n/a
17. Mitsubishi Power Systems	MWT 62/1.0	1	3	FS	Asynchronous machine	Unknown	n/a
	MWT 92/2.4	2.4	3	VS	Double-fed asynchronous machine	Unknown	Partial-scale IGBT power converter
	MWT 95/2.4	2.4	3	VS	Double-fed asynchronous machine	Unknown	Partial-scale IGBT power converter
	MWT 92/2.3	2.3	3	VS	Double-fed asynchronous machine	Unknown	Partial-scale IGBT power converter
	MWT 100/2.4	2.4	3	VS	Double-fed asynchronous machine	Unknown	Partial-scale IGBT power converter
	MWT 100/2.4	2.4	3	VS	Double-fed asynchronous machine	Unknown	Partial-scale IGBT power converter

Manufacturer	Model Name	Power Rating [MW]	Number of Blades	Speed Type	Generator Type	Gearbox	Converter type
19. E.O.N. Energy	e.o.n. 92 - 2.2	2.2	3	VS	Synchronous generator	Combined spur/planetary gear	Full-scale power converter
	e.o.n. 82 - 2.0	2	3	VS	Double-fed asynchronous	Combined spur/planetary gear	Partial-scale power converter
20. Envision	E77	1.5	3	VS	Double-fed asynchronous	3 Stages - planetary / spur gear	Partial-scale PWM IGBT power converter
	E70	1.5	3	VS	Double-fed asynchronous	3 Stages - planetary / spur gear	Partial-scale PWM IGBT power converter
	E82	1.5	3	VS	Double-fed asynchronous	3 Stages - planetary / spur gear	Partial-scale PWM IGBT power converter
	E87	1.5	3	VS	Double-fed asynchronous	3 Stages - planetary / spur gear	Partial-scale PWM IGBT power converter
21. Eviag	EV100	2.5	3	VS	Double-fed asynchronous with slipring	Two-stage planetary gear, one spur gear stage	Partial-scale power converter
	EV90	2.5	3	VS	Double-fed asynchronous with slipring	Two-stage planetary gear, one spur gear stage	Partial-scale power converter
	EV2.93	2.05	3	VS	Double-fed asynchronous with slipring	Two-stage planetary gear, one spur gear stage	Partial-scale power converter

VS	Variable Speed
FX	Fixed Speed
VerSlip	Variable Slip

APPENDIX B: ADDITIONAL DETAIL FOR DFIG MODELS

This appendix provides the expanded elements of the state-variable matrices of the ABC and DQ DFIG models.

B.1 ABC generator state-variable form

In Chapter 3 the state-variable equation for solving the currents of a DFIG in the ABC reference frame is obtained as

$$\frac{d\,\mathrm{I}}{dt} = \mathrm{L}^{-1}\left\{-\mathrm{R} - \omega_r \frac{d\,\mathrm{I}}{d\theta_r}\right\}\mathrm{I} + \mathrm{L}^{-1}\mathrm{V}\,. \tag{8.1}$$

It is clear that the inverse of the inductance matrix is required to obtain the currents. Some of the elements of the inductance matrix are a function of the electrical rotor position θ_r and thus a function of time. This implies that the inverse of the inductance matrix needs to be calculated at each step in the simulation leading to long simulation times [68]. One way of speeding up the simulation time is to analytically inverse the inductance matrix. This process is generally difficult but if the phase impedances are symmetrical and the currents of the rotor and stator are balanced, thus

$$i_{as} + i_{bs} + i_{cs} = 0 \tag{8.2}$$

and

$$i_{ar} + i_{br} + i_{cr} = 0\,, \tag{8.3}$$

the explicit expression can easily be obtained. An explicit expression for the derivatives of the currents can be obtained in the form of

$$\frac{d\,\mathbf{I}_{abc}}{dt} = \mathrm{A}\mathbf{I}_{abc} + \mathrm{B}\mathbf{V}_{abc}\,, \tag{8.4}$$

from the explicit expression of the inverse of the inductance matrix L, where

A is the explicit 6 x 6 matrix that is equal to $\mathrm{L}^{-1}\left\{-\mathrm{R} - \omega_r \frac{d\,\mathrm{I}}{d\theta_r}\right\}$,

B is the explicit 6 x 6 matrix that is equal to L^{-1},

$\mathbf{I}_{abc} = \begin{bmatrix} i_{as} & i_{bs} & i_{cs} & i_{ar} & i_{br} & i_{cr} \end{bmatrix}^T$ and

$$\mathbf{V}_{abc} = \begin{bmatrix} v_{as} & v_{bs} & v_{cs} & v_{ar} & v_{br} & v_{cr} \end{bmatrix}^T .$$

Equation (8.5) denotes the elements of the A and B matrices followed by the equations obtained for each element.

$$\frac{d}{dt}\begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{ar} \\ i_{br} \\ i_{cr} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{bmatrix}\begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \\ i_{ar} \\ i_{br} \\ i_{cr} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} \end{bmatrix}\begin{bmatrix} v_{as} \\ v_{bs} \\ v_{cs} \\ v_{ar} \\ v_{br} \\ v_{cr} \end{bmatrix} \tag{8.5}$$

where

$$b_{11} = b_{22} = b_{33} = \frac{K_2}{L_s},$$

$$b_{44} = b_{55} = b_{66} = \frac{K_2}{L_r},$$

$$b_{12} = b_{13} = b_{23} = b_{21} = b_{32} = b_{31} = \frac{K_3}{L_s},$$

$$b_{45} = b_{56} = b_{46} = b_{54} = b_{65} = b_{64} = \frac{K_3}{L_r},$$

$$b_{14} = b_{25} = b_{36} = b_{41} = b_{52} = b_{63} = K_4 f_1,$$

$$b_{15} = b_{26} = b_{34} = b_{43} = b_{51} = b_{62} = K_4 f_2,$$

$$b_{16} = b_{24} = b_{35} = b_{53} = b_{61} = b_{42} = K_4 f_3,$$

$$a_{11} = a_{22} = a_{33} = -b_{11}R_s + b_{14}\omega Mg_1 + b_{15}\omega Mg_2 + b_{16}\omega Mg_3,$$

$$a_{13} = a_{21} = a_{32} = -b_{12}R_s + b_{14}\omega Mg_2 + b_{15}\omega Mg_3 + b_{16}\omega Mg_1,$$

$$a_{31} = a_{12} = a_{23} = -b_{12}R_s + b_{14}\omega Mg_3 + b_{15}\omega Mg_1 + b_{16}\omega Mg_2,$$

$$a_{41} = a_{52} = a_{63} = -b_{14}R_s + b_{66}\omega Mg_1 + b_{45}\omega Mg_2 + b_{45}\omega Mg_3,$$

$$a_{51} = a_{62} = a_{43} = -b_{15}R_s + b_{66}\omega Mg_2 + b_{45}\omega Mg_3 + b_{45}\omega Mg_1,$$

$$a_{61} = a_{42} = a_{53} = -b_{16}R_s + b_{66}\omega Mg_3 + b_{45}\omega Mg_1 + b_{45}\omega Mg_2,$$

$$a_{66} = a_{55} = a_{44} = b_{14}\omega Mg_1 + b_{16}\omega Mg_3 + b_{15}\omega Mg_2 - b_{66}R_r,$$

$$a_{64} = a_{45} = a_{56} = b_{14}\omega Mg_2 + b_{16}\omega Mg_1 + b_{15}\omega Mg_3 - b_{45}R_r,$$

$$a_{54} = a_{65} = a_{46} = b_{14}\omega Mg_3 + b_{16}\omega Mg_2 + b_{15}\omega Mg_1 - b_{45}R_r,$$

$$a_{14} = a_{36} = a_{25} = b_{11}\omega Mg_1 + b_{12}\omega Mg_3 + b_{12}\omega Mg_2 - b_{14}R_r,$$

$$a_{26} = a_{15} = a_{34} = b_{11}\omega Mg_2 + b_{12}\omega Mg_1 + b_{12}\omega Mg_3 - b_{15}R_r,$$

and

$$a_{16} = a_{24} = a_{35} = b_{11}\omega Mg_3 + b_{12}\omega Mg_2 + b_{12}\omega Mg_1 - b_{16}R_r,$$

with

$$K_2 = \frac{K_1 - \frac{3}{4}}{K_1 - \frac{9}{4}},$$

$$K_3 = \frac{-\frac{3}{4}}{K_1 - \frac{9}{4}},$$

$$K_4 = \frac{-\frac{1}{M_{sr}}}{K_1 - \frac{9}{4}},$$

$$f_1(\theta_r) = \cos(\theta_r),$$

$$f_2(\theta_r) = \cos\left(\theta_r + \frac{2\pi}{3}\right),$$

$$f_3(\theta_r) = \cos\left(\theta_r - \frac{2\pi}{3}\right),$$

$$g_1(\theta_r) = \sin(\theta_r),$$

$$g_2(\theta_r) = \sin\left(\theta_r + \frac{2\pi}{3}\right) \text{ and}$$

$$g_3(\theta_r) = \sin\left(\theta_r - \frac{2\pi}{3}\right).$$

B.2 DQ generator state-variable form

In Chapter 3 the voltage equation of the DQ reference frame with the d-axis fixed to a-phase of the stator voltage is given as

$$\begin{bmatrix} v_D \\ v_Q \\ v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + L_S p & 0 & Mp & 0 \\ 0 & R_S + L_S p & 0 & Mp \\ Mp & -\dot{\theta}_r M & R_r + L_r p & -\dot{\theta}_r L_r \\ \dot{\theta}_r M & Mp & \dot{\theta}_r L_r & R_r + L_r p \end{bmatrix} \begin{bmatrix} i_D \\ i_Q \\ i_d \\ i_q \end{bmatrix}. \tag{8.6}$$

This can be rearranged to form the state-variable equation, given as

$$\dot{\mathbf{I}}_{DQdq} = \mathrm{A}\mathbf{I}_{DQdq} + \mathrm{B}\mathbf{V}_{DQdq} \tag{8.7}$$

where

$$\mathbf{V}_{DQdq} = [v_D \quad v_Q \quad v_d \quad v_q]^T,$$

$$\mathbf{I}_{DQdq} = [i_D \quad i_Q \quad i_d \quad i_q]^T,$$

$$\mathrm{A} = -\mathrm{L}^{-1}\mathrm{G}$$

and

$$\mathrm{B} = \mathrm{L}^{-1},$$

with

$$\mathrm{G} = \mathrm{R} + \mathrm{H}\omega,$$

$$\mathrm{L} = \begin{bmatrix} L_S & 0 & M & 0 \\ 0 & L_S & 0 & M \\ M & 0 & L_r & 0 \\ 0 & M & 0 & L_r \end{bmatrix},$$

$$\mathrm{R} = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_r & 0 \\ 0 & 0 & 0 & R_r \end{bmatrix}$$

and

$$\mathrm{H} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -M & 0 & -L_r \\ M & 0 & L_r & 0 \end{bmatrix}.$$

From these the elements of the B matrix are obtained as

$$\mathrm{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} = d \begin{bmatrix} L_r & 0 & -M & 0 \\ 0 & L_r & 0 & -M \\ -M & 0 & L_S & 0 \\ 0 & -M & 0 & L_S \end{bmatrix}$$

where

$$d = \frac{1}{L_S L_r - M^2}.$$

The elements of the A matrix are obtained as

$$\mathrm{A} = -\mathrm{BG} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = d \begin{bmatrix} -L_r R_S & -M^2\omega & MR_r & -ML_r\omega \\ M^2\omega & -L_r R_S & ML_r\omega & MR_r \\ MR_S & L_S M\omega & -L_S R_r & L_S L_r\omega \\ -L_S M\omega & MR_S & -L_S L_r\omega & -L_S R_r \end{bmatrix}.$$

This matrix can be simplified as

$$A = \begin{bmatrix} a_{11} & -a_{21} & a_{24} & a_{14} \\ a_{21} & a_{11} & -a_{14} & a_{24} \\ a_{31} & -a_{41} & a_{44} & a_{34} \\ a_{41} & a_{31} & -a_{34} & a_{44} \end{bmatrix}$$

where

$$a_{11} = a_{22} = \frac{-L_r R_S}{L_S L_r - M^2},$$

$$a_{21} = -a_{12} = \frac{M^2\omega}{L_S L_r - M^2},$$

$$a_{31} = a_{42} = \frac{MR_S}{L_S L_r - M^2},$$

$$a_{41} = -a_{32} = \frac{-L_s M\omega}{L_S L_r - M^2},$$

$$a_{14} = -a_{23} = \frac{-ML_r\omega}{L_S L_r - M^2},$$

$$a_{24} = a_{13} = \frac{MR_r}{L_S L_r - M^2},$$

$$a_{34} = -a_{43} = \frac{L_S L_r \omega}{L_S L_r - M^2}$$

and

$$a_{44} = a_{33} = \frac{-L_S R_r}{L_S L_r - M^2} .$$

Substituting A and B into (8.7) and performing the matrix multiplication produces the following four differential equations.

$$
\begin{aligned}
\dot{i}_D &= \frac{-Lr_{dq}Rs}{Ls_{dq}Lr_{dq}-M^2} i_D + \frac{-M^2\omega}{Ls_{dq}Lr_{dq}-M^2} i_Q + \frac{MR_r}{Ls_{dq}Lr_{dq}-M^2} i_d + \frac{-Lr_{dq}M\omega}{Ls_{dq}Lr_{dq}-M^2} i_q + \frac{Lr_{dq}}{Ls_{dq}Lr_{dq}-M^2} v_D + \frac{-M}{Ls_{dq}Lr_{dq}-M^2} v_d \\
\dot{i}_Q &= \frac{M^2\omega}{Ls_{dq}Lr_{dq}-M^2} i_D + \frac{-Lr_{dq}R_S}{Ls_{dq}Lr_{dq}-M^2} i_Q + \frac{Lr_{dq}M\omega}{Ls_{dq}Lr_{dq}-M^2} i_d + \frac{R_rM}{Ls_{dq}Lr_{dq}-M^2} i_q + \frac{Lr_{dq}}{Ls_{dq}Lr_{dq}-M^2} v_Q + \frac{-M}{Ls_{dq}Lr_{dq}-M^2} v_q \\
\dot{i}_d &= \frac{R_SM}{Ls_{dq}Lr_{dq}-M^2} i_D + \frac{Ls_{dq}M\omega}{Ls_{dq}Lr_{dq}-M^2} i_Q + \frac{-Ls_{dq}R_r}{Ls_{dq}Lr_{dq}-M^2} i_d + \frac{Ls_{dq}Lr_{dq}\omega}{Ls_{dq}Lr_{dq}-M^2} i_q + \frac{-M}{Ls_{dq}Lr_{dq}-M^2} v_D + \frac{Ls_{dq}}{Ls_{dq}Lr_{dq}-M^2} v_d \\
\dot{i}_q &= \frac{-Ls_{dq}M\omega}{Ls_{dq}Lr_{dq}-M^2} i_D + \frac{R_SM}{Ls_{dq}Lr_{dq}-M^2} i_Q + \frac{-Ls_{dq}Lr_{dq}\omega}{Ls_{dq}Lr_{dq}-M^2} i_d + \frac{-Ls_{dq}R_r}{Ls_{dq}Lr_{dq}-M^2} i_q + \frac{-M}{Ls_{dq}Lr_{dq}-M^2} v_Q + \frac{Ls_{dq}}{Ls_{dq}Lr_{dq}-M^2} v_q
\end{aligned}
\tag{8.8}
$$

APPENDIX C: S-FUNCTION MODELS

This appendix shows the masks created for each of the implemented component models of the wind turbine system, i.e., the turbine blades, gearbox, ABC DFIG and DQ DFIG. The mask configurations are provided as well as the C-code of the S-function models.

C.1 Turbine blades model

C.1.1 Mask

Figure C-1 shows the masked Simulink block of the implemented turbine blade model. The masked parameter dialog window of the block is shown in Figure C-2.

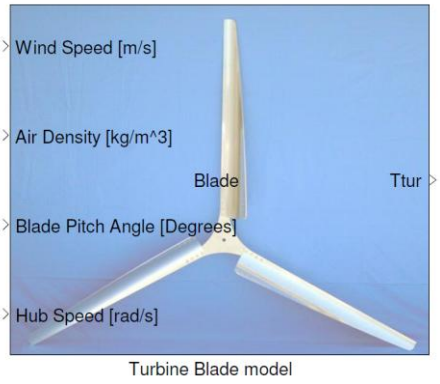


Figure C-1: Masked Simulink block of turbine blades model.

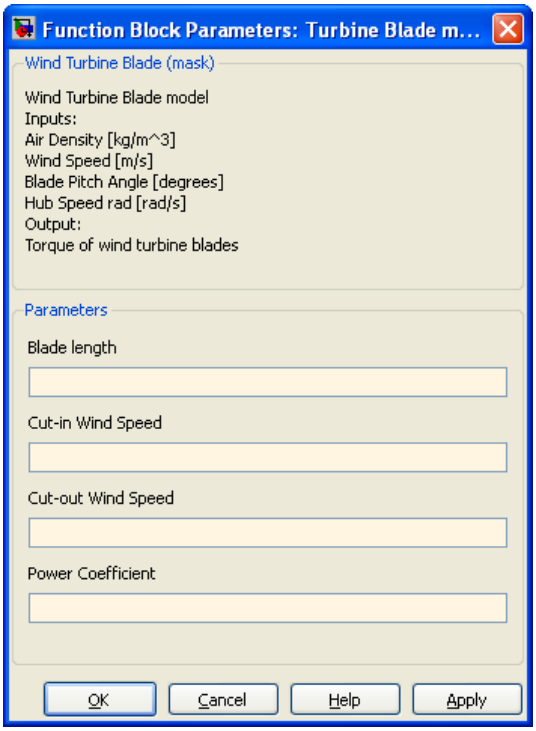


Figure C-2: Masked parameter dialog window of wind turbine blade model.

C.1.2 Under the mask

Figure C-3 shows the unmasked parameter window of the turbine blade model, showing the variable names of the four parameters of the turbine blade model. The name of the MEX-file used for the model is displayed in the S-function name field.

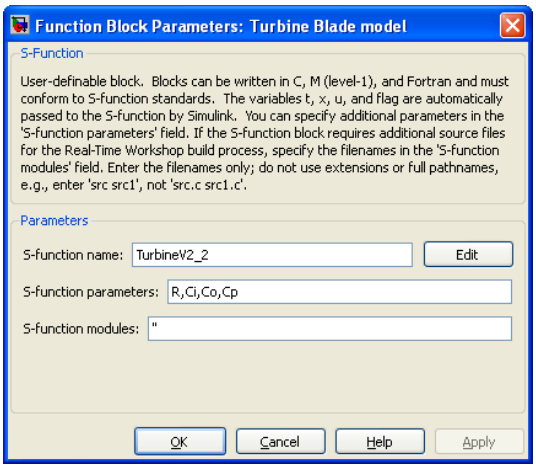


Figure C-3: Wind turbine blade model S-function configuration window.

C.1.3 Mask configuration

Figure C-4 and Figure C-5 show the mask configuration windows, with the configuration required to obtain the masked block shown in Figure C-1 and masked parameter window shown in Figure C-2.

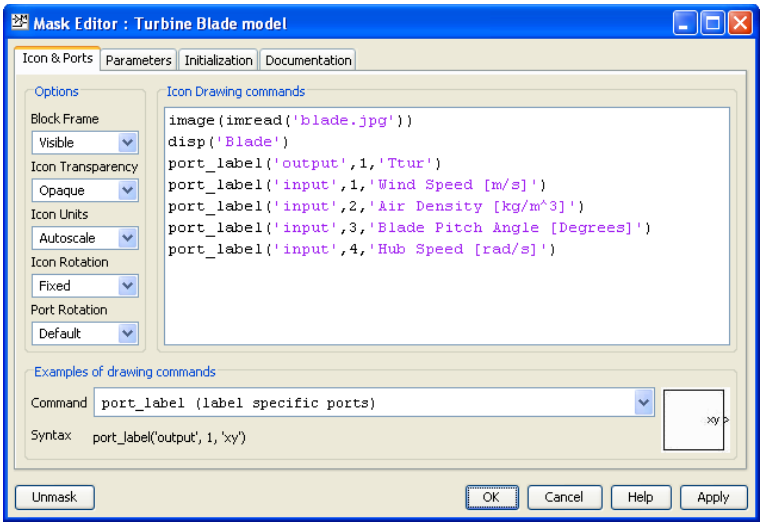


Figure C-4: Wind turbine blade model mask configuration window – Icon & Port tab.

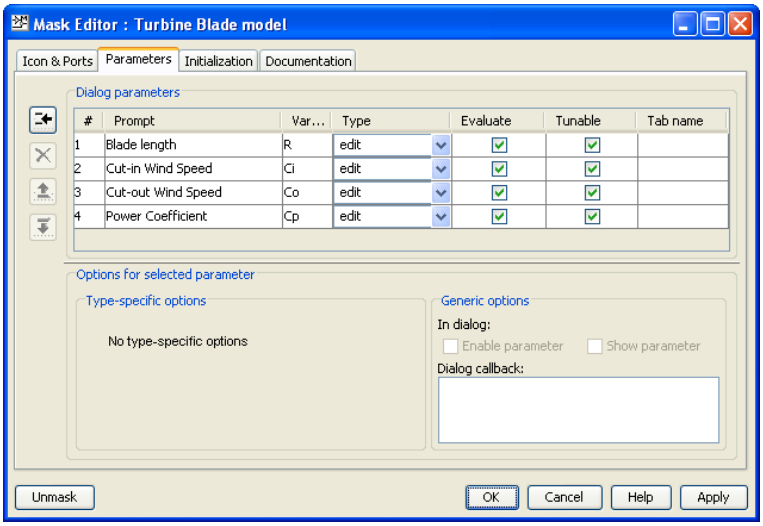


Figure C-5: Wind turbine blade model mask configuration window – Parameters tab.

C.1.4 Code

The code of the C-code S-function turbine blades model is available on the accompanying DVD.

C.2 Two-mass gearbox model

C.2.1 Mask

Figure C-6 shows the masked Simulink block of the implemented two-mass gearbox model. The masked parameter windows of the block are shown in Figure C-7 and Figure C-8.

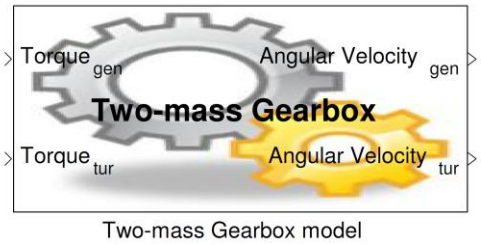


Figure C-6: Masked Simulink block of two-mass gearbox model.

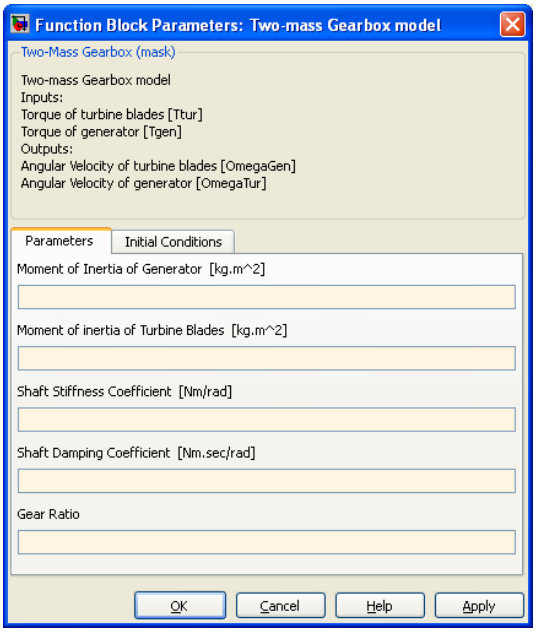


Figure C-7: Masked parameter dialog window of two-mass gearbox model – Parameters tab.

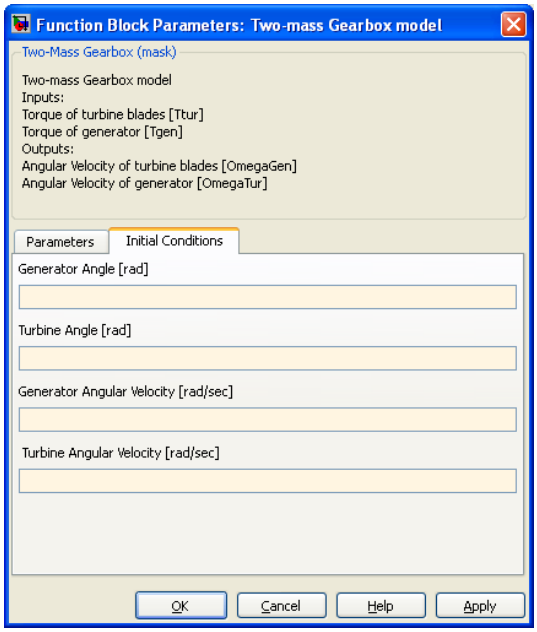


Figure C-8: Masked parameter dialog window of two-mass gearbox model – Initial Conditions tab.

C.2.2 Under the mask

Figure C-9 shows the unmasked parameter window of the gearbox model, showing the variable names of the four parameters and four initial state values. The name of the MEX-file used for the model is displayed in the S-function name field.

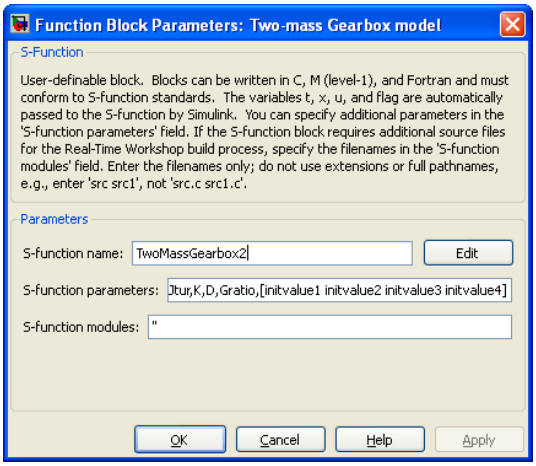


Figure C-9: Two-mass gearbox model S-function configuration window.

C.2.3 Mask configuration

Figure C-10 and Figure C-11 show the mask configuration windows with the configuration required to obtain the masked block shown in Figure C-6 and the masked parameter windows shown in Figure C-7 and Figure C-8.

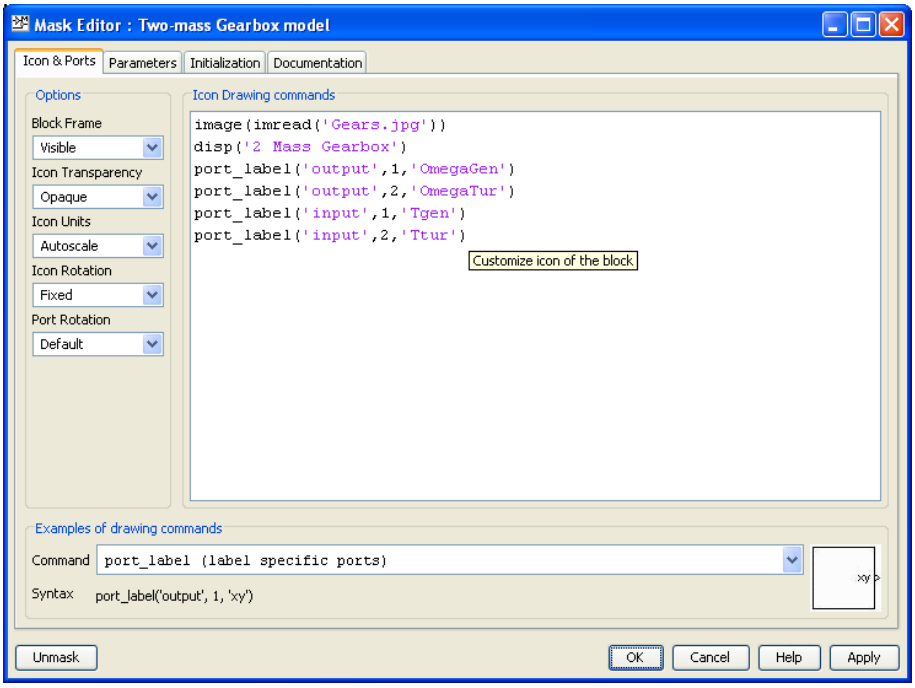


Figure C-10: Two-mass gearbox model mask configuration window – Icon & Ports tab.

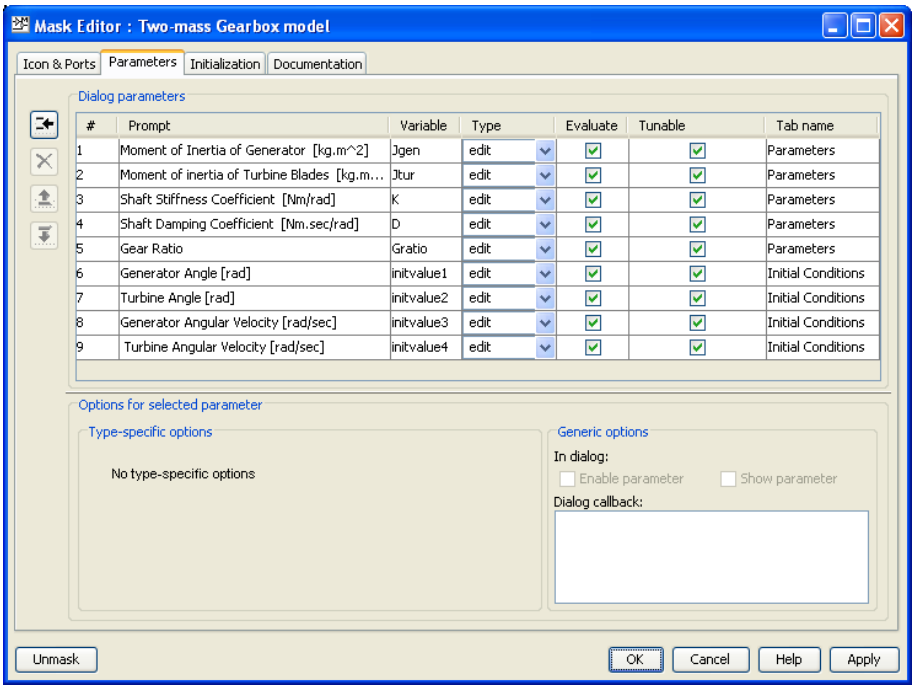


Figure C-11: Two-mass gearbox model mask configuration window – Parameters tab.

C.2.4 Code

The following code is the C-code that is compiled as a MEX-file. The MEX-file is used with the S-function Simulink block to implement the two-mass gearbox model.

```
/*
 * File: TwoMassGearbox2.c
 * Author: JC Bekker
```

```
 * e-mail: jcbekker@gmail.com
 *
 * Abstract:
 *      S-Function model for induction machine
 *      Inputs:    Wind Turbine Torque [Nm]
 *                 Generator Torque [Nm]
 *      Output:    Shaft speed generator side [rad/sec]
 *                 Shaft speed wind turbine side  [rad/sec]
 *      Parameter: Moment of Inertia - Generator [kg*m^2]
 *                 Moment of Inertia - Wind Turbine [kg*m^2]
 *                 Stiffness - Shaft  [Nm/rad]
 *                 Damping coefficient - Shaft [Nm sec/rad]
 *                 Gearbox ratio - [no unit]
 *                 Initial conditions of shaft speeds of generator and wind
 *                      turbine side  [rad/sec]
 *
 * Copyright 20011 - University of Stellenbosch
 * $Revision: 2.0.0.0 $
 * Revision 1.0.0.0 makes use of equations from Wind Turbine Blockset notes
 * This revision is based on self derived equations with the help of the
paper:
 * Fundamental time-domain wind Turbine models for wond power studies by
Surya Santoso and Ha The Le
 */

#define S_FUNCTION_NAME  TwoMassGearbox2
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include <math.h>

#define Tgen(element) (*TgenPtrs[element])  //Pointer to Input Port0
#define Ttur(element) (*TturPtrs[element])  //Pointer to Input Port1

/* Moment of inertia of generator [kg.m^2] */
#define PARAM1(S) ssGetSFcnParam(S,0)

/* Moment of inertia of turbine blades [kg.m^2] */
#define PARAM2(S) ssGetSFcnParam(S,1)

/* Shaft stiffness coefficient */
#define PARAM3(S) ssGetSFcnParam(S,2)

/* Shaft damping coefficient */
#define PARAM4(S) ssGetSFcnParam(S,3)

/* Rear ratio */
#define PARAM5(S) ssGetSFcnParam(S,4)

/* initial conditions 4 element vector
 * element 1: Generator Angle [rad]
 * element 2: Turbine Angle [rad]
 * element 3: Generator Angular Velocity [rad/sec]
 * element 4: Turbine Angular Velocity [rad/sec]
 */
#define PARAM6(S) ssGetSFcnParam(S,5)

#define IS_PARAM_DOUBLE(pVal) (mxIsNumeric(pVal) && !mxIsLogical(pVal) &&\
!mxIsEmpty(pVal) && !mxIsSparse(pVal) && !mxIsComplex(pVal) &&
mxIsDouble(pVal))
```

```
#if defined(MATLAB_MEX_FILE)
  /* Function: mdlCheckParameters ===========================================
   * Abstract:
   *    This routine will be called after mdlInitializeSizes, whenever
   *    parameters change or get re-evaluated. The purpose of this routine
   *    is to verify that the new parameter setting are correct.
   *
   *    This routine is calles from mdlInitalizeSizes to check the
   *    parameters after setting the sizes elements
   */
# define MDL_CHECK_PARAMETERS
static void mdlCheckParameters(SimStruct *S)
{
    /* Checking parameter 1 for being a positive scalar of type double
     * and display relevant error message if check fails.
     */
    if (mxGetNumberOfElements(PARAM1(S))!=1 ||!IS_PARAM_DOUBLE(PARAM1(S)))
    {
        ssSetErrorStatus(S,"Parameter 1 to S-function must be a scalar");
        return;
    }
    else if (mxGetPr(PARAM1(S))[0] < 0)
    {
      ssSetErrorStatus(S,"Parameter 1 to S-function must be non-negative");
      return;
    }

    /* Checking parameter 2 for being a positive scalar of type double
     * and display relevant error message if check fails.
     */
    if (mxGetNumberOfElements(PARAM2(S))!= 1 ||!IS_PARAM_DOUBLE(PARAM1(S)))
    {
      ssSetErrorStatus(S,"Parameter 2 to S-function must be a scalar");
      return;
    }
    else if (mxGetPr(PARAM2(S))[0] < 0)
    {
      ssSetErrorStatus(S,"Parameter 2 to S-function must be non-negative");
      return;
    }

    /* Checking parameter 3 for being a positive scalar of type double
     * and display relevant error message if check fails.
     */
    if (mxGetNumberOfElements(PARAM3(S))!= 1 ||!IS_PARAM_DOUBLE(PARAM1(S)))
    {
      ssSetErrorStatus(S,"Parameter 3 to S-function must be a scalar");
      return;
    }
    else if (mxGetPr(PARAM3(S))[0] < 0)
    {
      ssSetErrorStatus(S,"Parameter 3 to S-function must be non-negative");
      return;
    }

    /* Checking parameter 4 for being a positive scalar of type double
     * and display relevant error message if check fails.
     */
```

```
    if (mxGetNumberOfElements(PARAM4(S))!= 1 ||!IS_PARAM_DOUBLE(PARAM1(S)))
    {
      ssSetErrorStatus(S,"Parameter 4 to S-function must be a scalar");
      return;
    }
    else if (mxGetPr(PARAM4(S))[0] < 0)
    {
      ssSetErrorStatus(S,"Parameter 4 to S-function must be non-negative");
      return;
    }

    /* Checking parameter 5 for being a positive scalar of type double
     * and display relevant error message if check fails.
     */
    if (mxGetNumberOfElements(PARAM5(S))!= 1 ||!IS_PARAM_DOUBLE(PARAM1(S)))
    {
        ssSetErrorStatus(S,"Parameter 5 to S-function must be a scalar");
        return;
    }
    else if (mxGetPr(PARAM5(S))[0] < 0)
    {
      ssSetErrorStatus(S,"Parameter 5 to S-function must be non-negative");
      return;
    }

    /* Checking parameter 6 for being a 4 element vector with elements of
     * type double and display relevant error message if check fails.
     */
     if(mxGetNumberOfElements(PARAM6(S))!= 4 ||!IS_PARAM_DOUBLE(PARAM1(S)))
     {
      ssSetErrorStatus(S,"Parameter 6 to S-function must be a 4 element
vector");
      return;
     }
  }
#endif

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *   The sizes information is used by Simulink to determine the S-function
 *   block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S)
{
    /* Sets number of expected parameters to 6*/
    ssSetNumSFcnParams(S, 6);
#if defined(MATLAB_MEX_FILE)
    if (ssGetNumSFcnParams(S) == ssGetSFcnParamsCount(S)) {
        mdlCheckParameters(S);
        if (ssGetErrorStatus(S) != NULL) {
            return;
        }
    } else {
        return; /* Parameter mismatch will be reported by Simulink */
    }
#endif
```

```
    {
        int iParam = 0;
        int nParam = ssGetNumSFcnParams(S);

        for ( iParam = 0; iParam < nParam; iParam++ )
        {
            ssSetSFcnParamTunable( S, iParam, SS_PRM_SIM_ONLY_TUNABLE );
        }
    }

    //States Initialize
    /*x[0] Generator Angle [rad]
     * x[1] Turbine Angle [rad]
     * x[2] Generator Angular Velocity [rad/sec]
     * x[3] Turbine Angular Velocity [rad/sec]
     */
    ssSetNumContStates(S, 4);

    ssSetNumDiscStates(S, 0);  //no discreet states

    //Inputs Initialize
    /* Set number of inputs to 2
     * Input 1: Torque of Turbine blades
     * Input 2: Torque of Generator
     *
     * Set input 1's port with to 1
     * Set input 2's port with to 1
     *
     * Configure input 1 and input 2 for direct fed through for use in
     * mdloutput function.
     */
    if (!ssSetNumInputPorts(S, 2)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortWidth(S, 1, 1);

    ssSetInputPortDirectFeedThrough(S, 0, 1);
    ssSetInputPortDirectFeedThrough(S, 1, 1);

    //Outputs Initialize
    /* Set number of outputs to 2
     * Output 1: Angular Velocity of generator
     * Output 2: Angular Velocity of turbine blades
     * Set output 1's port with to 1
     * Set output 2's port with to 1
     */
    if (!ssSetNumOutputPorts(S, 2)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetOutputPortWidth(S, 1, 1);

    //Sample time
    /* Set number of sample times block has to 1
     */
    ssSetNumSampleTimes(S, 1);

    //Work vectors (none setup)

    /* Take care when specifying exception free code - see sfuntmpl_doc.c*/
    //ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}
```

```
/* Function: mdlInitializeSampleTimes =================================
 * Abstract:
 *    Specify a continuous sample time.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
    ssSetModelReferenceSampleTimeDefaultInheritance(S);
}

#define MDL_INITIALIZE_CONDITIONS
/*Function: mdlInitializeConditions
 *Abstract:
 *  Initialize the continuous to initial conditions
 */
static void mdlInitializeConditions(SimStruct *S)
{
    real_T *x0 = ssGetContStates(S);
    int_T lp;
    real_T *initpar = mxGetPr(PARAM6(S));

    for(lp=0;lp<4;lp++)
    {
        *x0++=*initpar++;
    }
}

/* Function: mdlOutputs ===============================================
 * Abstract:
 *      wGen =  Angular Velocity of Generator
 *      wTur =  Angular Velocity of turbine blades
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T                  *wGen    = ssGetOutputPortRealSignal(S,0);
    real_T                  *wTur    = ssGetOutputPortRealSignal(S,1);
    real_T                  *x       = ssGetContStates(S);

    UNUSED_ARG(tid); /* not used in single tasking mode */

    wGen[0] = x[2];  // Assign Generator Angular Velocity state to output 1
    wTur[0] = x[3];  // Assign Turbine  Angular Velocity state to output 2
}

#define MDL_DERIVATIVES
/* Function: mdlDerivatives============================================
 *Abstract:
 *      Dynamic equations of two-mass model
 * x[0] - theta_gen
 * x[1] - theta_tur
 * x[2] - omega_gen
 * x[3] - omega_tur
 */
static void mdlDerivatives(SimStruct *S)
{
    real_T                  *dx        = ssGetdX(S);
    real_T                  *x         = ssGetContStates(S);
```

```
    InputRealPtrsType       TgenPtrs   = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType       TturPtrs   = ssGetInputPortRealSignalPtrs(S,1);

    real_T                  Jgen       = mxGetPr(PARAM1(S))[0];
    real_T                  Jtur       = mxGetPr(PARAM2(S))[0];
    real_T                  K           = mxGetPr(PARAM3(S))[0];
    real_T                  D           = mxGetPr(PARAM4(S))[0];
    real_T                  Gratio      = mxGetPr(PARAM5(S))[0];

//  Revision: 2.0.1.0    29 April 2010
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (-D*(x[2]-Gratio*x[3])-K*(x[0]-
Gratio*x[1])+Gratio*Gratio*Tgen(0))/(Jgen*Gratio*Gratio);
    dx[3] = (-D*(x[3]-x[2]/Gratio)-K*(x[1]-x[0]/Gratio)-Ttur(0))/Jtur;

}
/* Function: mdlTerminate =================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
    UNUSED_ARG(S); /* unused input argument */
}

#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file?*/
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

C.3 Generator models – Double-fed induction generator

C.3.1 ACB DFIG model

C.3.2 Mask

Figure C-12 shows the masked Simulink block of the implemented ABC DFIG model. The masked parameter dialog windows of the block are shown in Figure C-13 and Figure C-14.

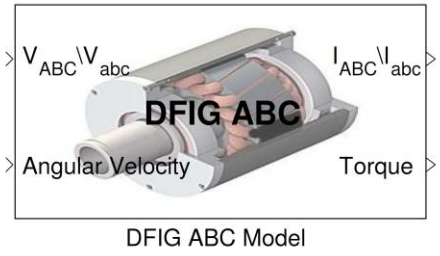


Figure C-12: Masked Simulink block of DFIG ABC model.

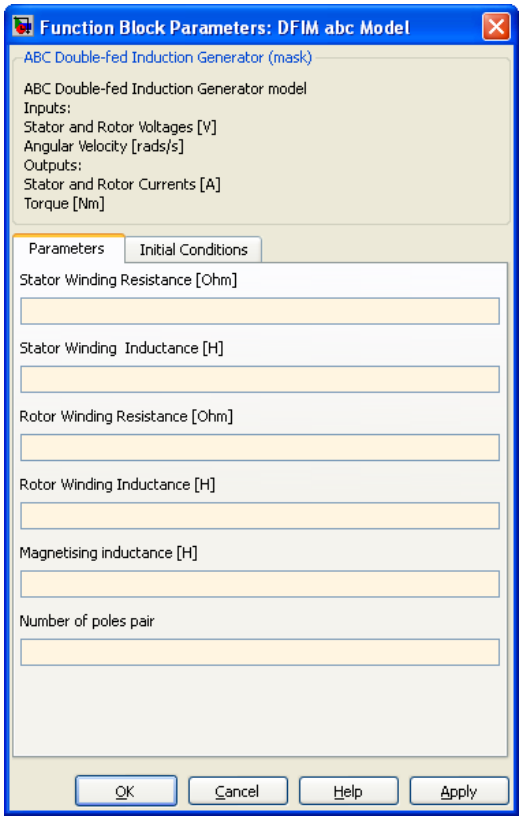


Figure C-13: Masked parameter dialog window of DFIG ABC model – Parameters tab.

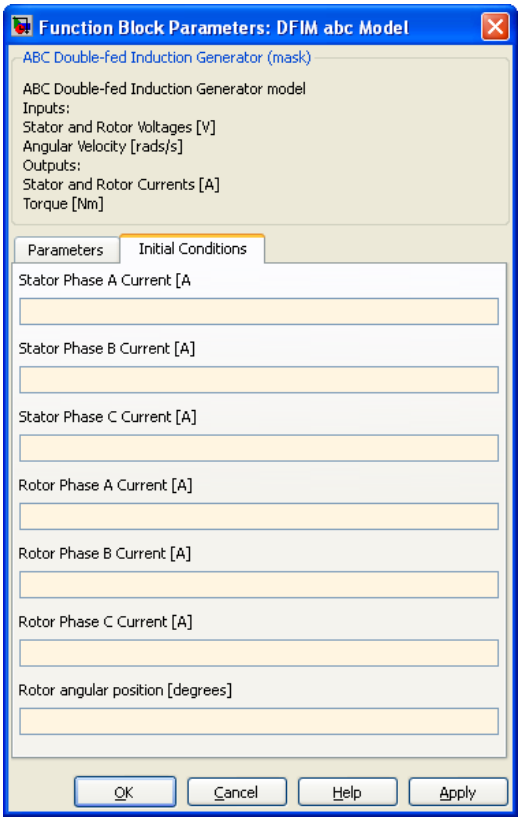


Figure C-14: Masked parameter dialog window of DFIG ABC model – Initial Conditions tab.

C.3.3 Under the mask

Figure C-15 shows the unmasked parameter window of the DFIG model, showing the variable names of the 6 parameter of the ABC DFIG as well as the 7 initial state value variable names. The name of the MEX-file used for the model is displayed in the S-function name field.

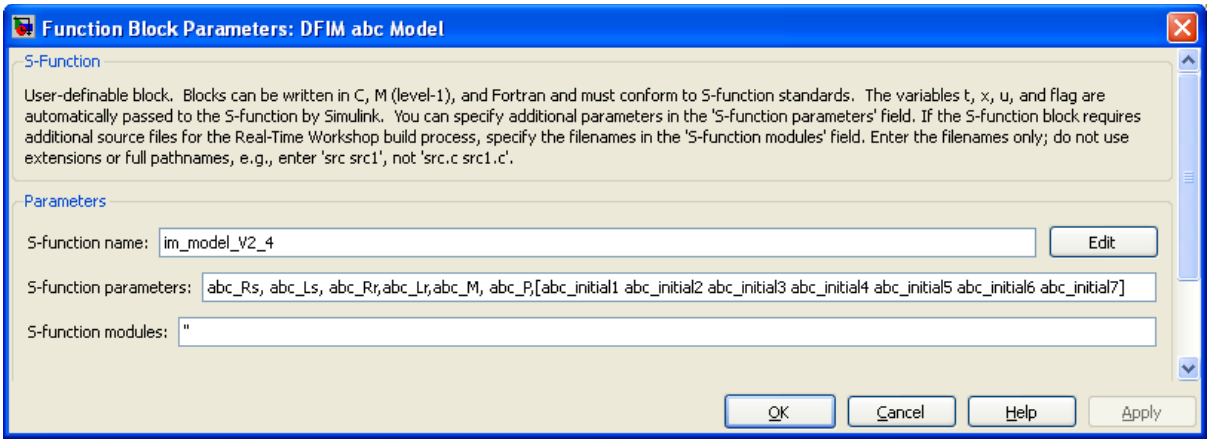


Figure C-15: DFIG ABC model S-function configuration window.

C.3.4 Mask configuration

Figure C-16 and Figure C-17 show the mask configuration windows, with the configuration required to obtain the masked block shown in Figure C-12 and the masked parameter windows shown in Figure C-13 and Figure C-14.

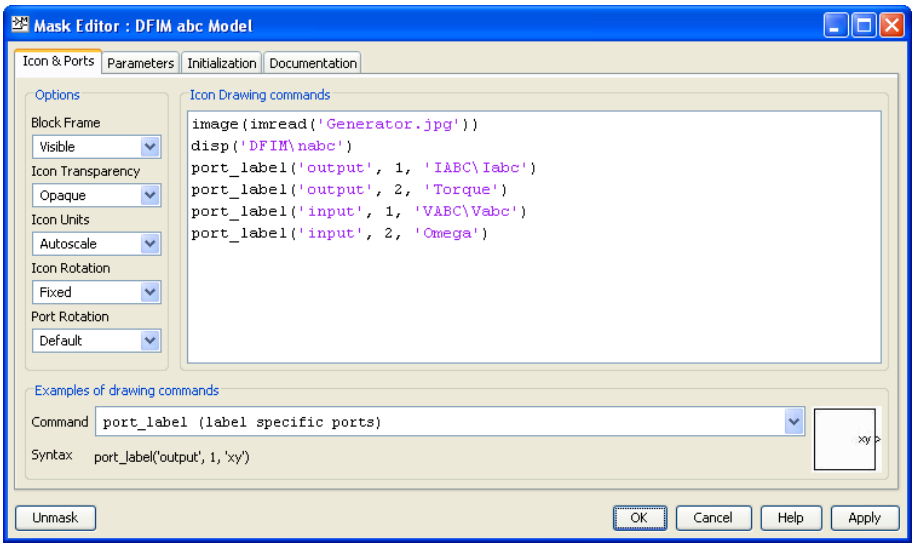


Figure C-16: DFIG ABC model mask configuration window – Icon & Ports tab.

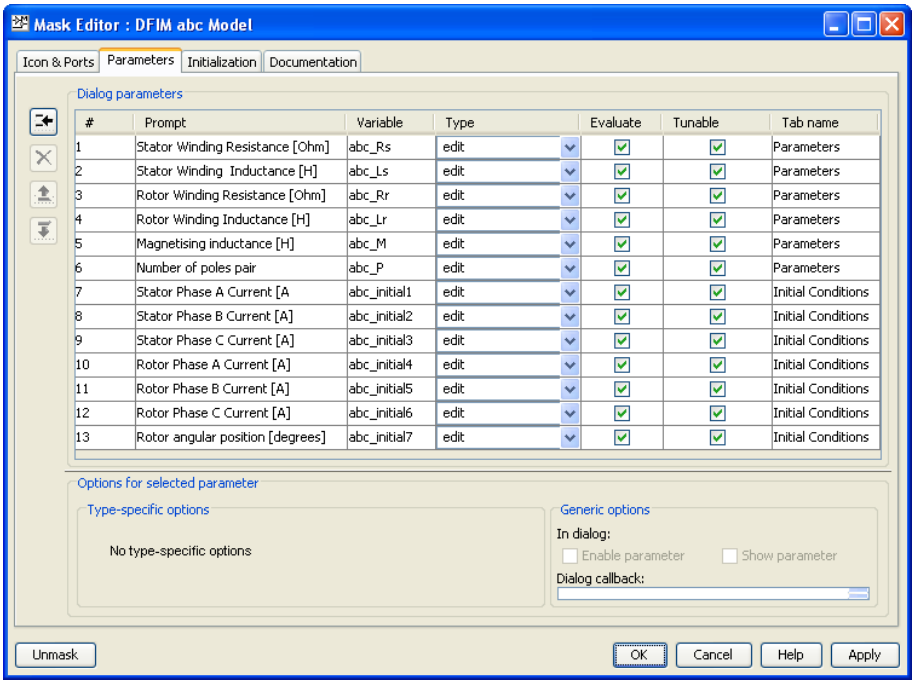


Figure C-17: DFIG ABC model mask configuration window – Parameters tab.

C.3.5 Code

The code of the C-code S-function ABC DFIG model is available on the accompanying DVD.

C.4 DQ DFIG model

C.4.1 Mask

Figure C-18 shows the masked Simulink block of the implemented DQ DFIG model. The masked parameter windows of the block are shown in Figure C-19 and Figure C-20.

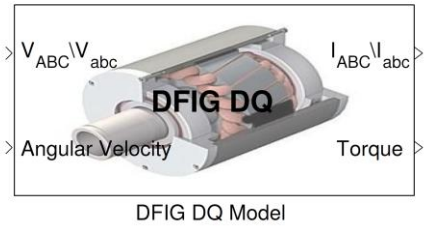


Figure C-18: Masked Simulink block of DFIG DQ model.

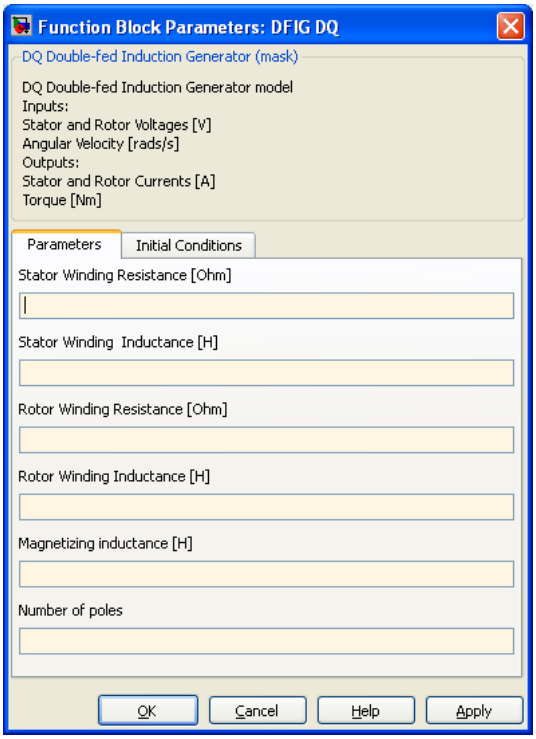


Figure C-19: Masked parameter dialog window of DFIG DQ model– Parameters tab.

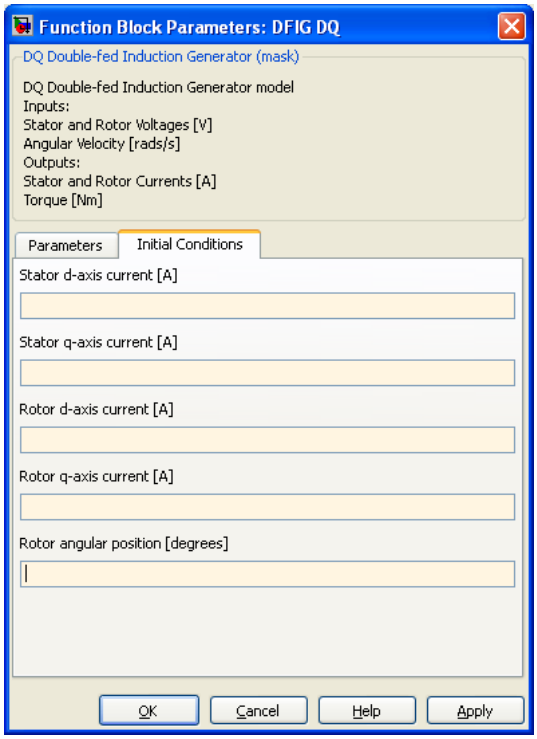


Figure C-20: Masked parameter dialog window DFIG DQ model – Initial Conditions tab.

C.4.2 Under the mask

Figure C-21 shows the unmasked parameter window of the DFIG model, showing the variable names of the 6 parameter of the DQ DFIG as well as the 5 initial state value variable names. The name of the MEX-file used for the model is displayed in the S-function name field.

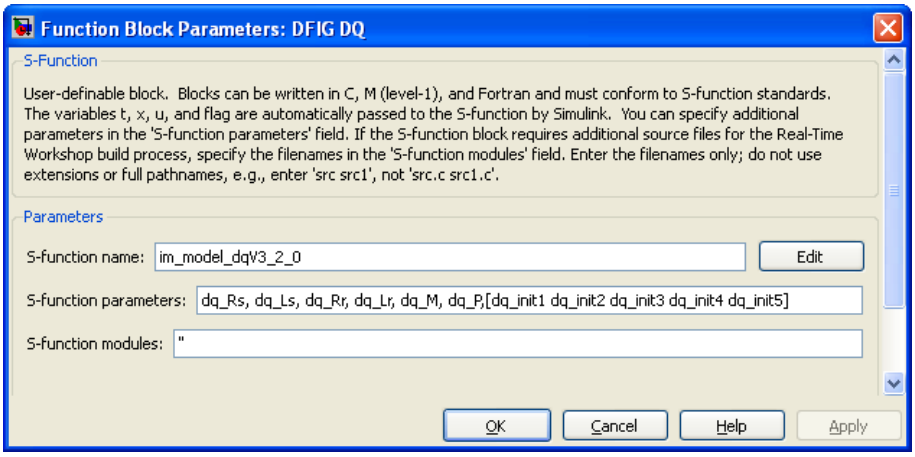


Figure C-21: DFIG DQ model S-function configuration window.

C.4.3 Mask configuration

Figure C-22 and Figure C-23 show the mask configuration windows with the configuration required to obtain the masked block shown in Figure C-18 and the masked parameter windows shown in Figure C-19 and Figure C-20.

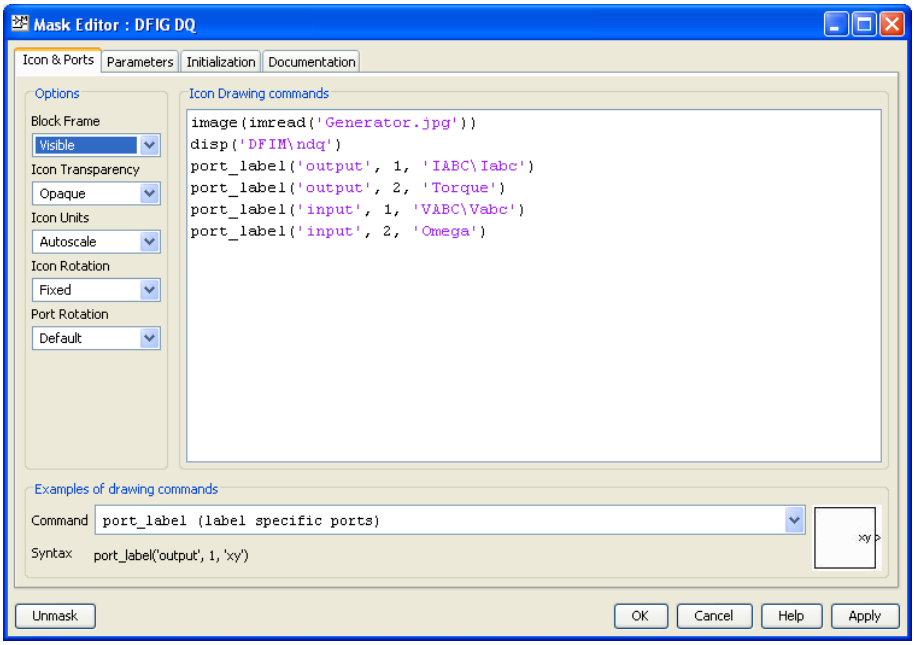


Figure C-22: DFIG DQ model mask configuration window – Icon & Ports tab.

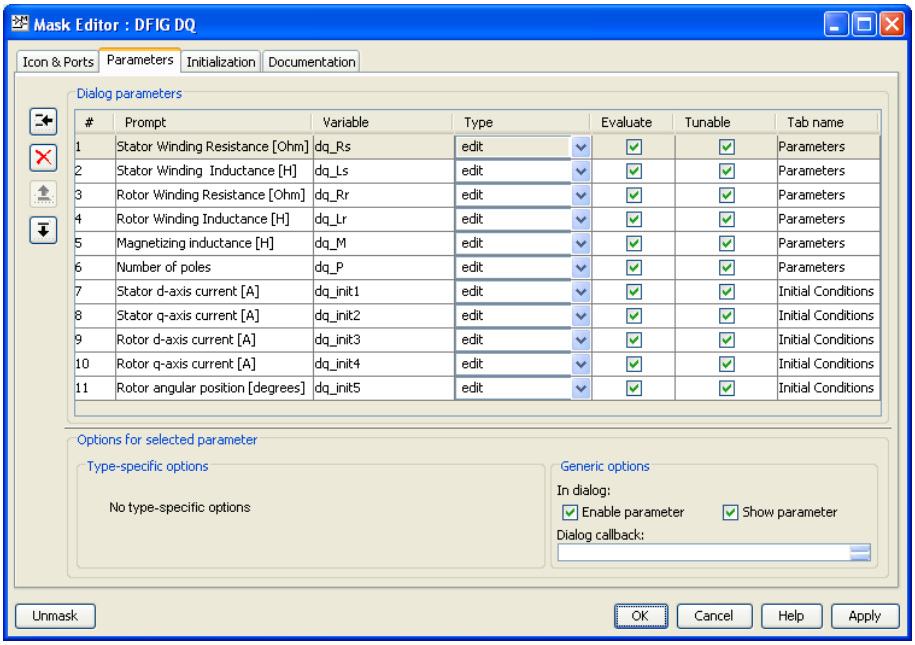


Figure C-23: DFIG DQ model mask configuration window – Parameters tab.

C.4.4 Code

The code of the C-code S-function DQ DFIG model is available on the accompanying DVD.

APPENDIX D: APPROXIMATION OF POWER COEFFICIENT BY ANALYTIC FUNCTION

This appendix reproduces the approximated nonlinear function for the power coefficient parameter of the turbine blade model provided in [77].

The power coefficient can be approximated by [77],

$$C_p(\lambda,\beta) = c_1\left(c_2 - c_3\beta - c_4\right)\mathrm{e}^{-c_5} \tag{8.9}$$

where

λ denotes tip speed ratio

β denotes blade pitch angle

$c_1 = 0.5$,

$c_2 = \dfrac{116}{\lambda_i}$,

$c_3 = 0.4$,

$c_4 = 5$

and

$$c_6 = \frac{21}{\lambda_i}$$

with

$$\lambda_i = \left(\frac{1}{\lambda + 0.08\beta} - \frac{0.035}{\beta^3 + 1}\right)^{-1}.$$

APPENDIX E: PARAMETER VALUES OF WIND TURBINE SYSTEM

E.1 Overview

This appendix provides the parameter values of the individual model of the wind turbine system model used for validating the model in Chapter 4.

E.2 Turbine blade model parameters

Table E-1: Turbine blade model parameter values.

Parameter	Symbol	Value	Parameter	Symbol	Value
R	Blade length	50 m	v_{out}	Cut-out Wind Speed	15 m/s
v_{in}	Cut-in Wind Speed	1 m/s			

E.3 Gearbox model parameters

Table E-2: Gearbox model parameter values.

Parameter	Description	Values	Parameter	Description	Value
J_{gen}	Generator Moment of Inertia	90 kg.m^2	D	Shaft Damping coefficient	756 kNm.s/rad
J_{tur}	Turbine Moment of Inertia	4.95 Mkg.m^2	GR	Gear Ratio	83
K	Shaft Stiffness coefficient	114 MNm/rad			

E.4 DFIG model parameters

Table E-3: DFIG model parameter values.

Parameter	Description	Value	Parameters	Description	Value
R_s	Stator resistance	5 mΩ	L_r	Rotor inductance	299.2 μH
R_r	Rotor resistance	8.9 mΩ	L_m	Magnetising inductance	16 mH
L_s	Stator inductance	407.5 μH	P	Number of poles in machine	4

APPENDIX F: ADDITIONAL MATLAB INFORMATION

F.1 Overview

This appendix provides information on the numerical solvers of MATLAB. It is followed by information on the process limits of MATLAB and concludes with information on optimisation in MATLAB.

F.2 Solver information

F.2.1 Overview

This section provides the definition of a stiff system and is followed by an overview of the variable step solvers available in MATLAB.

F.2.2 Stiff systems

When numerical integration is performed for solving differential equations, relatively small step sizes are expected in regions where the solution curve displays fast dynamics, whereas, a relatively large step size is expected in regions where the solution curve straightens out. For some problems the numerical solver forces down the step size to an unacceptable small level in regions where the solution curve is very smooth. Models with a mixture of fast and slow changing variables are numerically stiff; these require the use of solvers capable of solving stiff systems.

F.2.3 Variable step solvers

MATLAB has 7 variable step solvers to choose from depending on the type of problem. These are listed below with a short overview of the solvers [60].

- *ode45 (Dormand-Prince):* The ode 45 solver is based on an explicit Runge-Kutta (4,5) formula (Dormand-Prince pair). It is a one-step solver and in general is the best solver to apply as first try.
- *ode113 (Adams):* The ode113 solver is based on the Adams-Bashforth-Moulton PECE numerical integration technique. This solver can be more effective than yhe ode45 solver at stringent tolerances.
- *ode23 (Bogacki-Shampine):* The ode23 solver is based on an explicit Runge-Kutta (2,3) formula (Bogacki-Shampine pair) for numerical integration. Like the ode45

solver, it is a one step solver but is more efficient at crude tolerances and in the presence of mild stiffness.

- *ode15s (stiff/NDF):* The ode15s is a variable order solver based on the numerical differentiation formulas (NDFs). These NDFs are related to the backward differentiation formulas (BDFs), but are more efficient. If a problem is suspected to be stiff or the ode45 solver is very inefficient, this solver should be tried.
- *ode23s (stiff/Mod. Rosenbrock):* The ode23s solver is based on a modified Rosenbrock formula of order 2. Due to it being a one-step solver, it may be more efficient than ode15s at crude tolerances. It also has the ability to solve some kinds of stiff problems for which ode15s are not effective.
- *ode23t (Mod. stiff/Trapezoidal):* The ode23t solver is also a one-step solver. It makes use of the trapezoidal rule using a "free" interpolant. This solver can be used to solve problems that are only moderately stiff and require solutions without numerical damping.
- *ode23tb (stiff/TR-BDF2):* The ode23tb solver implements a multistep TR-BDF2. This consists of an implicit Runge-Kutta formula with a first stage that is a trapezoidal rule step and a second stage that is a backward differentiation formula of order 2. Like ode23s this solver may be more efficient than ode15s at crude tolerances, and can solve stiff problems for which ode15s is ineffective.

F.3 Process limits of MATLAB supported operating systems

The following table is provided in the *MATLAB – Programming Fundamentals* document [79] showing the process limitation on the different operating systems supported by MATLAB.

Table F-1: Process limit of MATLAB for supported operating systems [79]**.**

Operating System	**Process Limit**
32-bit Microsoft Windows XP, Windows Vista™, Windows 7	2 GB
32-bit Windows XP with 3 GB *boot.ini* switch or , 32-bit Windows Vista™ or Windows 7 with *increaseuserva* set	3 GB
32-bit Linux© (Linux is a registered trademark of Linus Torvalds)	~3 GB
64-bit Windows XP, Apple Macintoch® OS X, or Linux running 32-bit MATLAB	≤ 4 GB
64-bit Windows XP, Windows Vista, or Linux, running 64-bit MATLAB	8 TB

F.4 Optimisation information

The following table is provided in the user's guide for the optimisation toolbox [63]. Knowing the objective function and the constraint type, this table can be used to choose a suitable optimisation solver.

Table F-2: Table of MATLAB optimisation solvers by objective function and constraint type [63]**.**

Constaint Type	**Objective Type**				
	Linear	**Quadratic**	**Least Squares**	**Smooth nonlinear**	**Nonsmooth**
None	n/a (*f* = const, or min = -∞)	*quadprog*	\, *lsqcurvefit, lsqnonlin*	*fminsearch, fminunc*	*Fminsearch,**
Bound	*linprog*	*quadprog*	*lsqcurvefit, lsqlin, lsqnonlin, lsqnonneg*	*fminbnd, fmincon, fseminf*	*Fminbnd, **
Linear	*linprog*	*quadprog*	*lsqlin*	*fmincon, fseminf*	*
General Smooth	*fmincon*	*fmincon*	*fmincon*	*fmincon, fsemif*	*
Discrete	*bintprog, **	*	*	*	*

* - relevant solvers are found in Global Optimization Toolbox

APPENDIX G: SIMULATED DATA OF WIND TURBINE SYSTEM MODEL

G.1 Overview

This appendix presents the data obtained from a forward simulation performed on the Derived wind turbine system model with the DQ DFIG model as generating element, shown in Figure G-1. The parameter values used for the simulations are given in APPENDIX E. The air density and blade pitch angle inputs are kept constant with values 1 kg/m^3 and 5° respectively. The DFIG is simulated as an induction generator with supply voltages given in Table G-1. The signals presented in section G.2 are the generated input wind signal used as input for the simulation together with all the output signals of the different component models. Section G.3 presents the real wind signal used as input for the simulation together with all the output signals of the different component models.

Table G-1: Input values electrical model.

Variable	Description	Value	Variable	Description	Value
$\mathbf{V}_{ABC}$	Stator Voltage	3-phase 600 V 50 Hz	$\mathbf{V}_{abc}$	Rotor Voltage	0 V

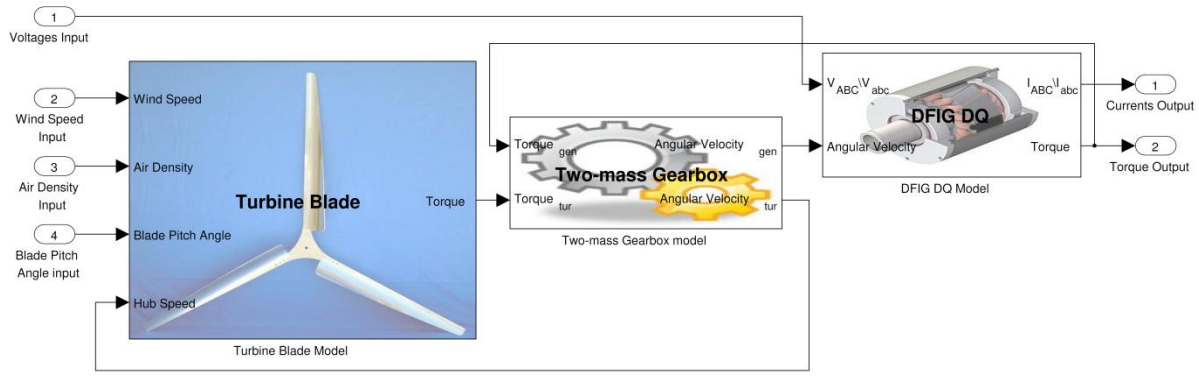


Figure G-1: Wind turbine system model with DQ DFIG model as generating element.

G.2 Simulation data for simulation performed with generated wind speed signals as input

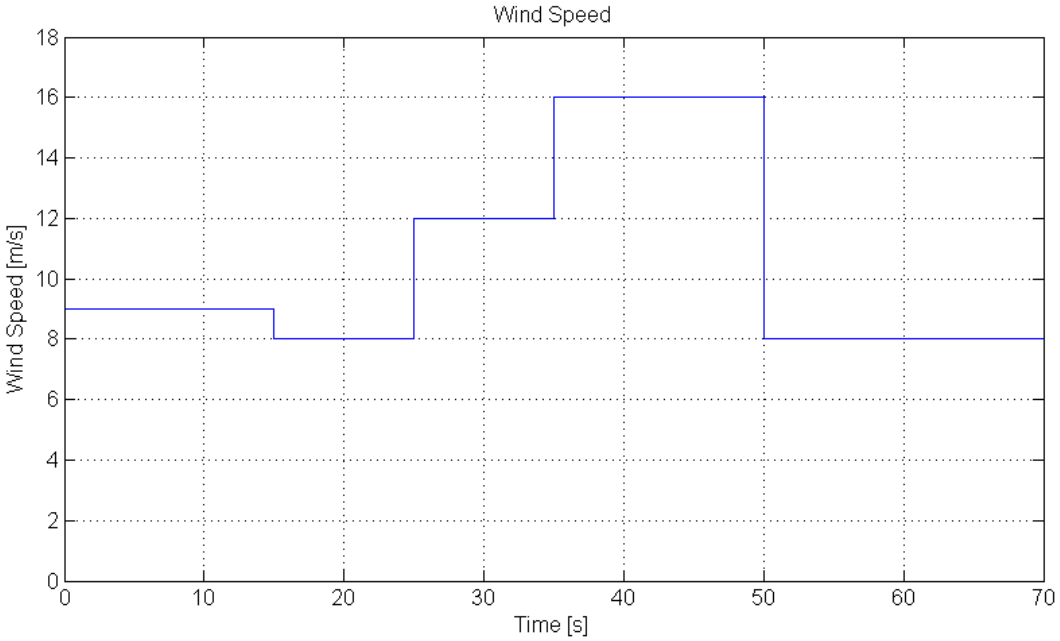


Figure G-2: Generated wind speed, input to turbine blade model.

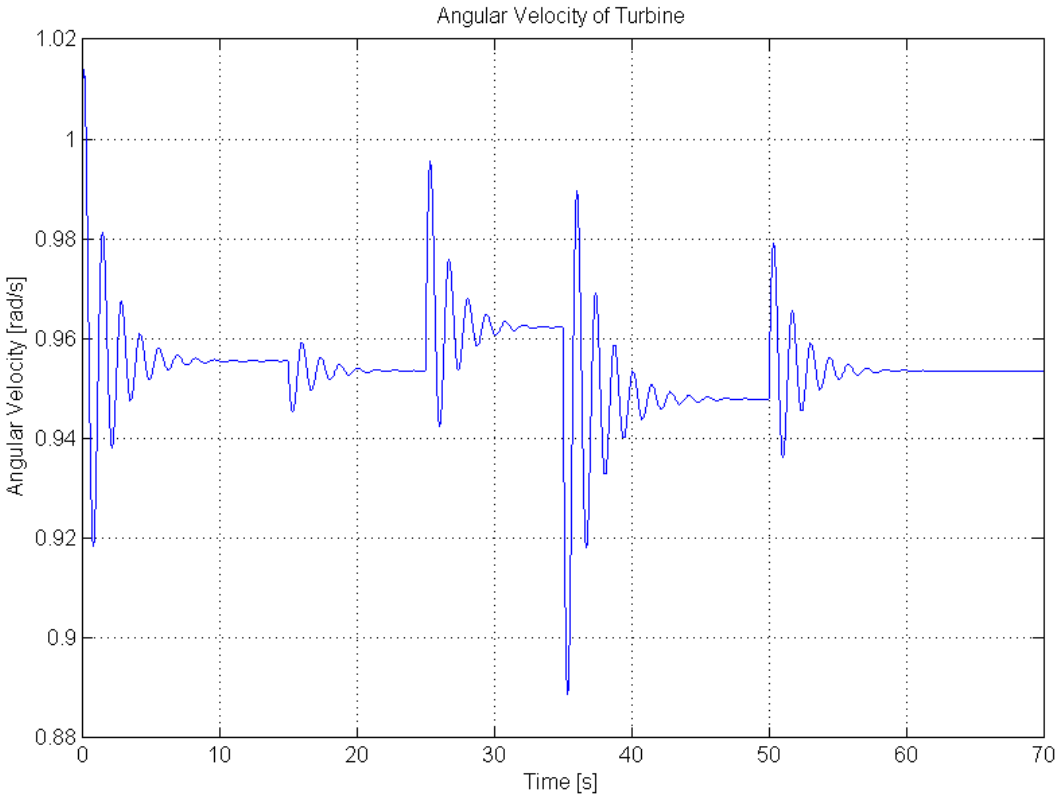


Figure G-3: Simulated turbine blade angular velocity for generated wind speed input signal, output of gearbox model.

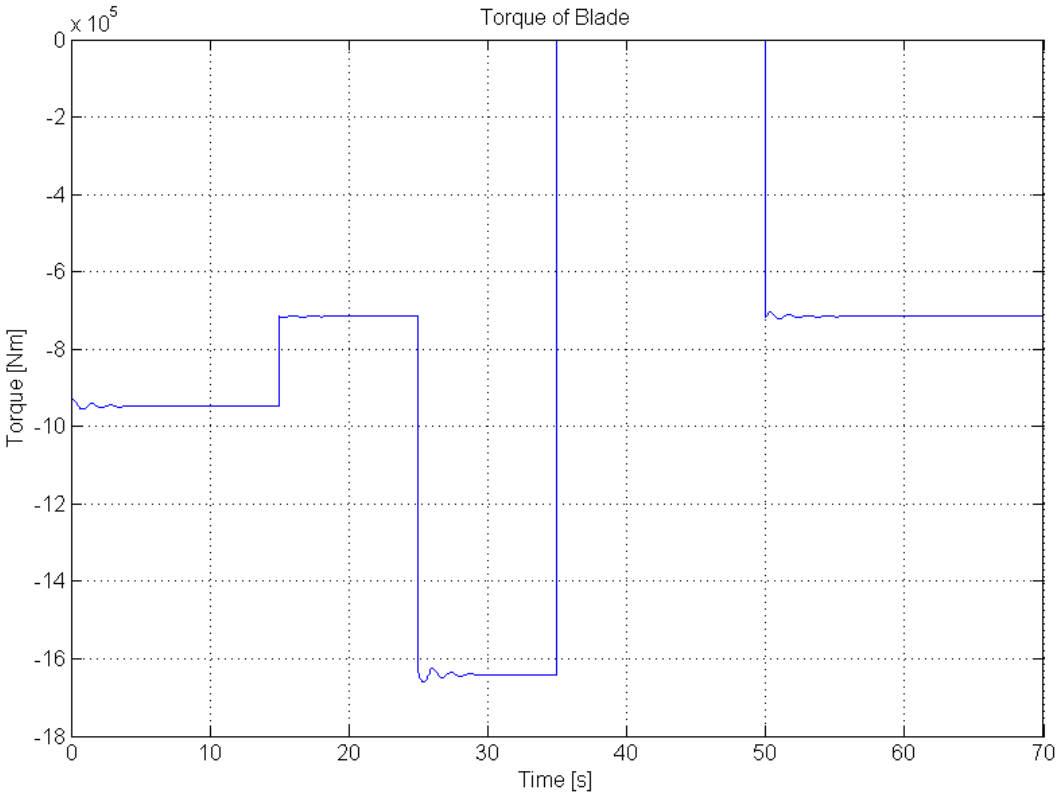


Figure G-4: Simulated turbine blade torque for generated wind speed input signal, output of turbine blade model.

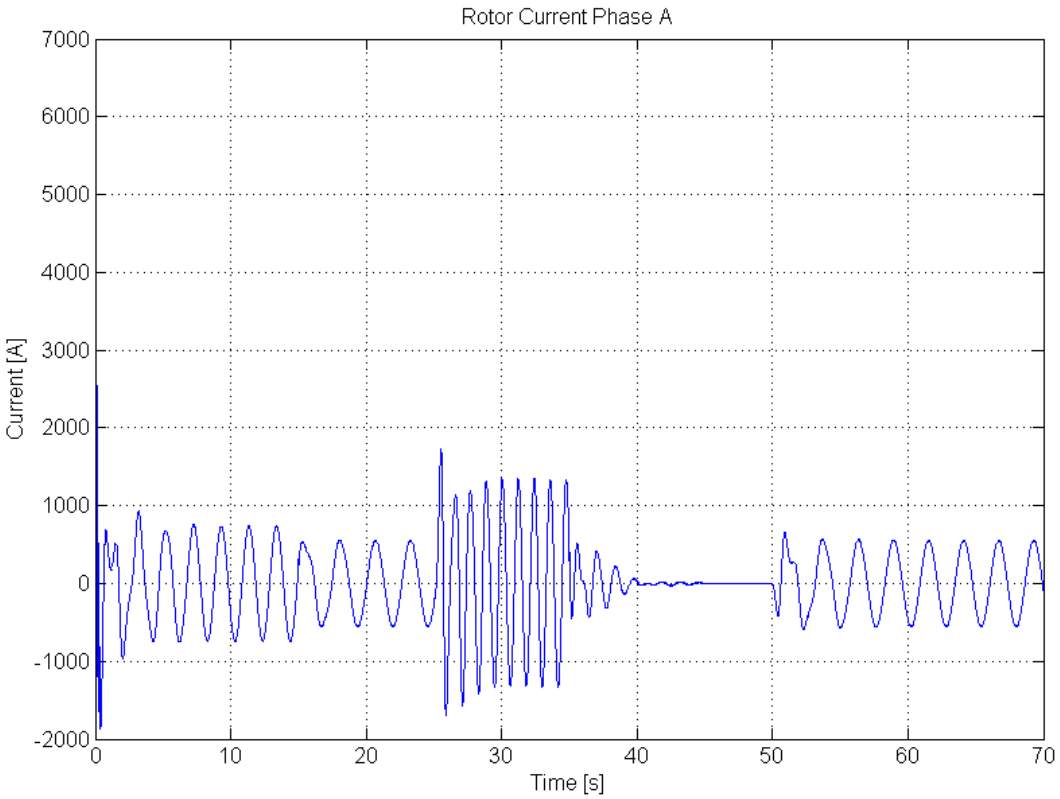


Figure G-5: Simulated A-phase rotor current for generated wind speed input signal, output of DFIG model.

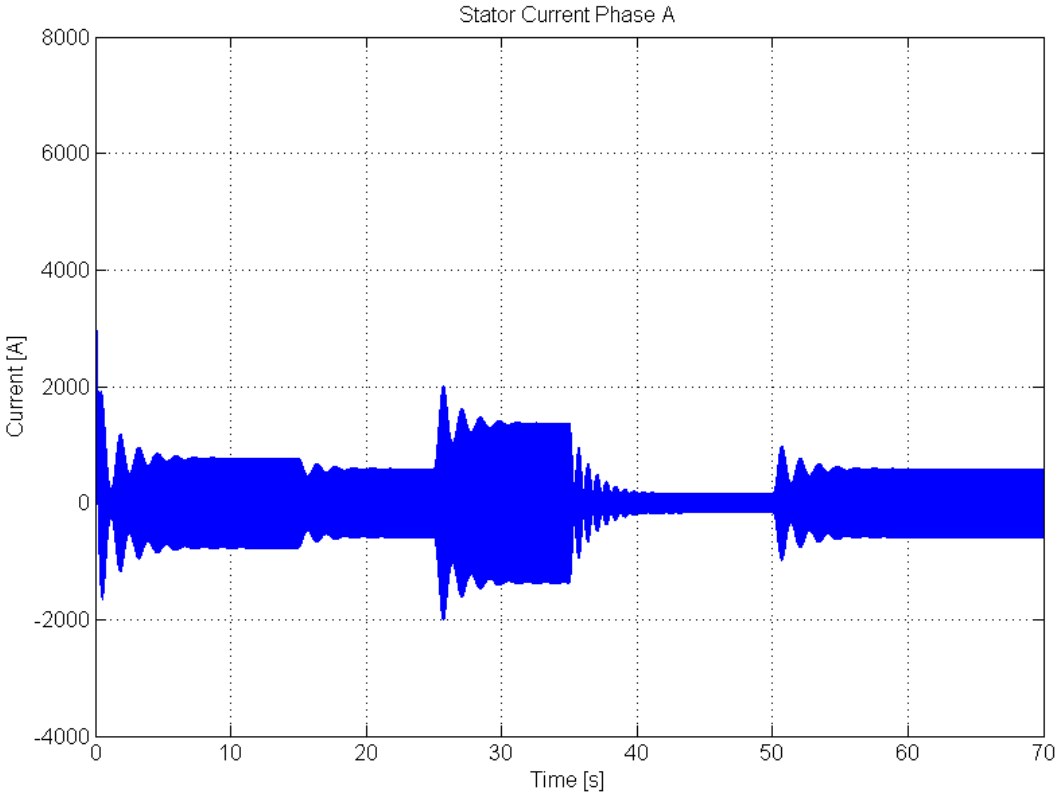


Figure G-6: Simulated A-phase stator current for generated wind speed input signal, output of DFIG model.

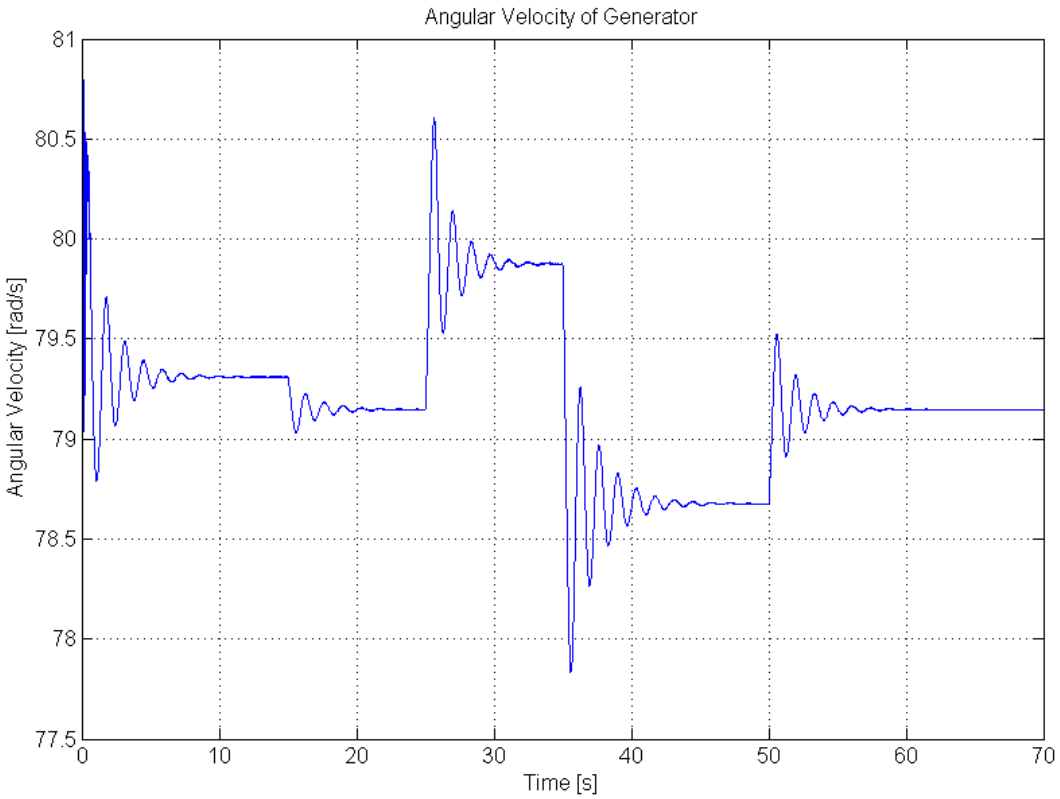


Figure G-7: Simulated generator angular velocity for generated wind speed input signal, output of gearbox model.

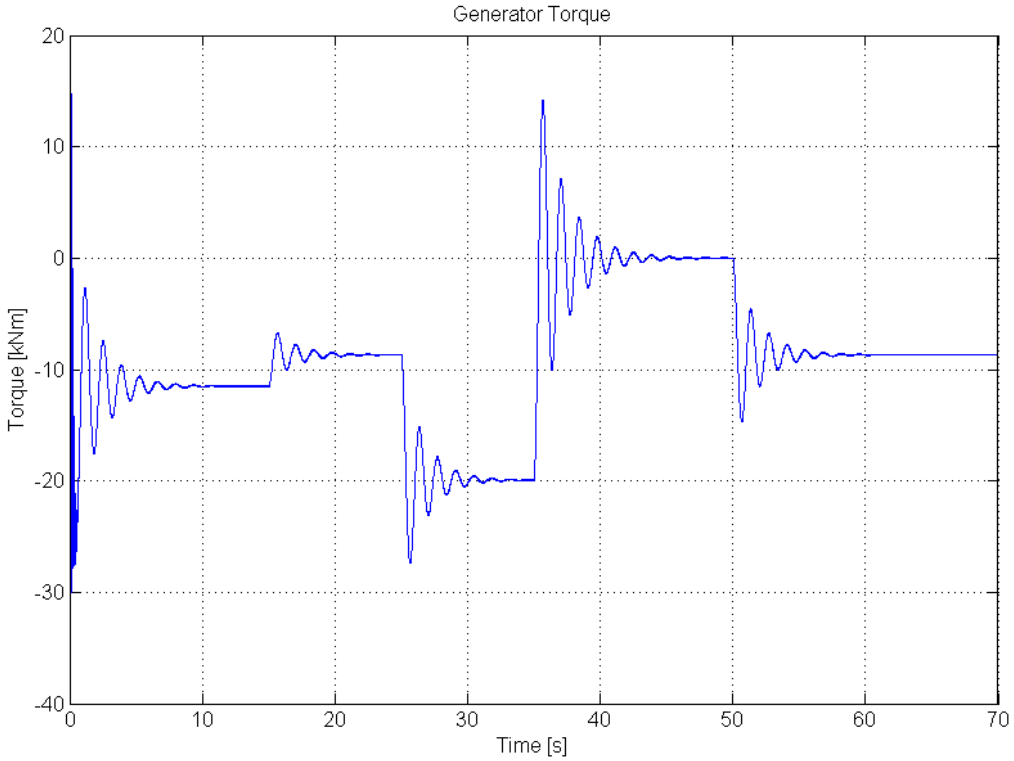


Figure G-8: Simulated generator torque for generated wind speed input signal, output of DFIG model.

G.3 Simulation data for simulation performed with real wind speed signal as input

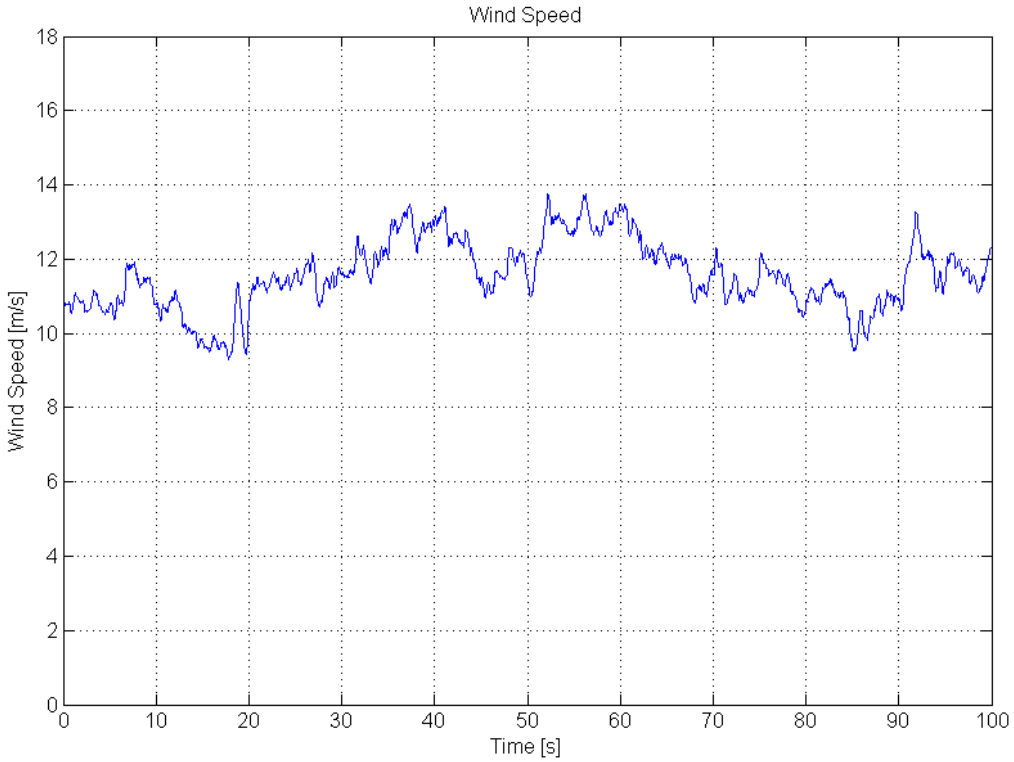


Figure G-9: Real wind speed signal, input to turbine blade model.

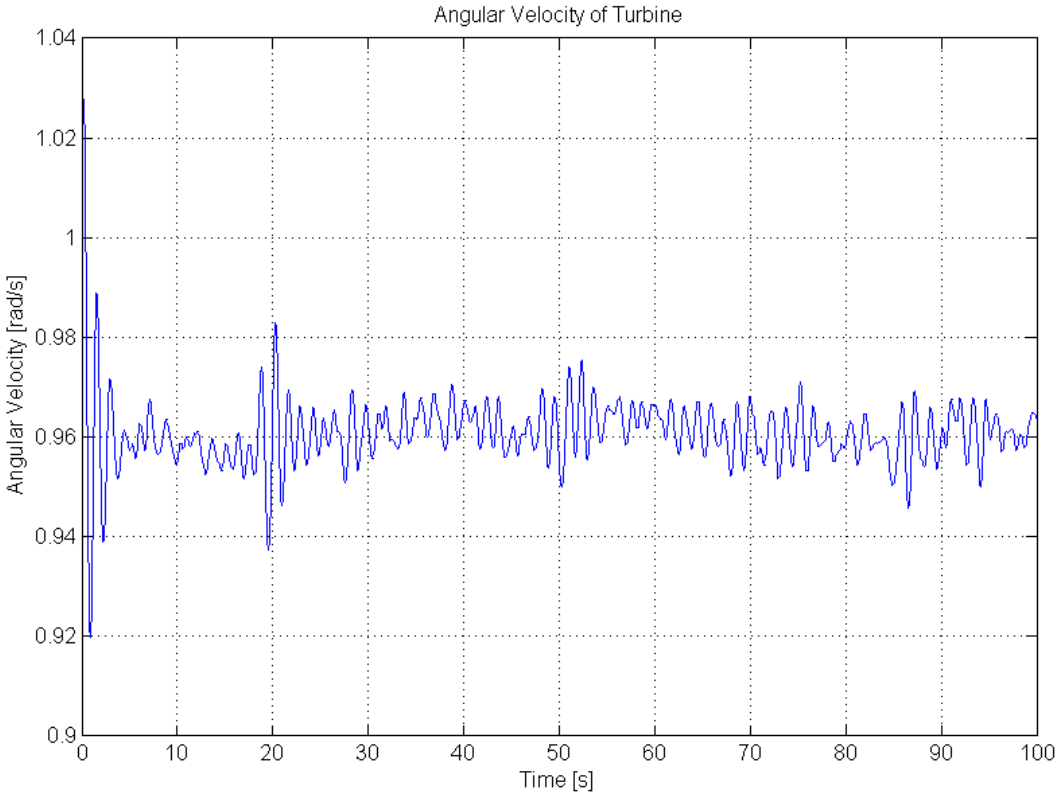


Figure G-10: Simulated turbine blade angular velocity for real wind speed input signal, output of gearbox model.

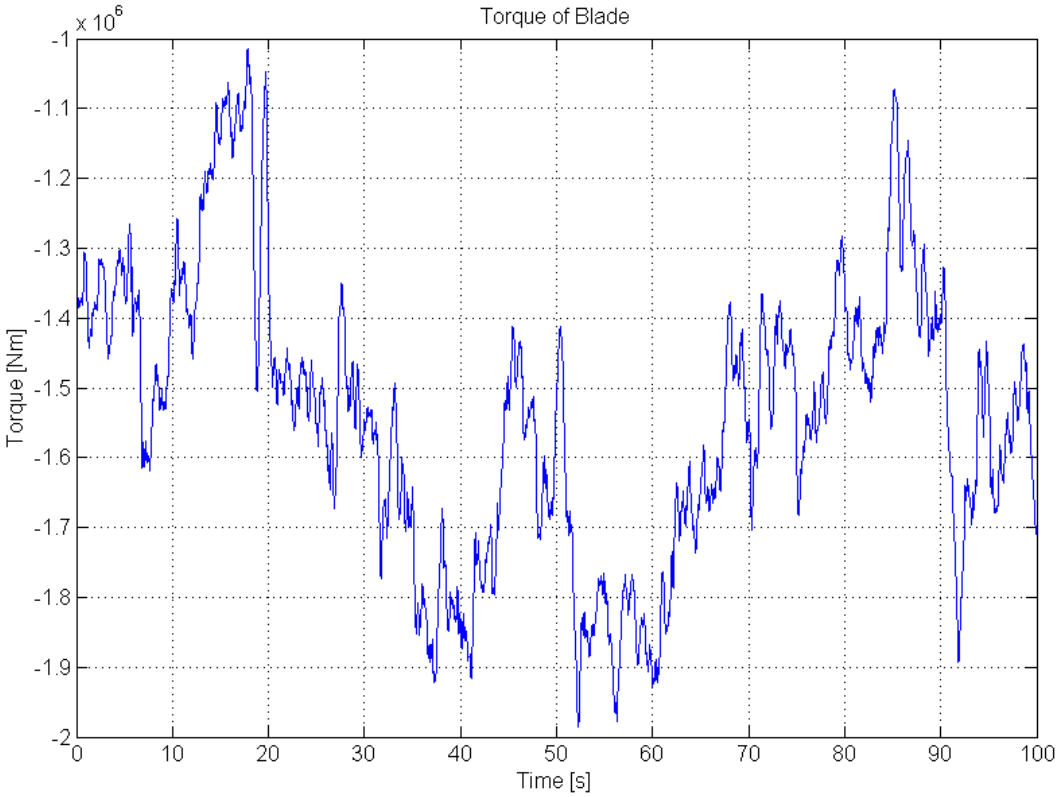


Figure G-11: Simulated turbine blade torque for real wind speed input signal, output of turbine blade model.

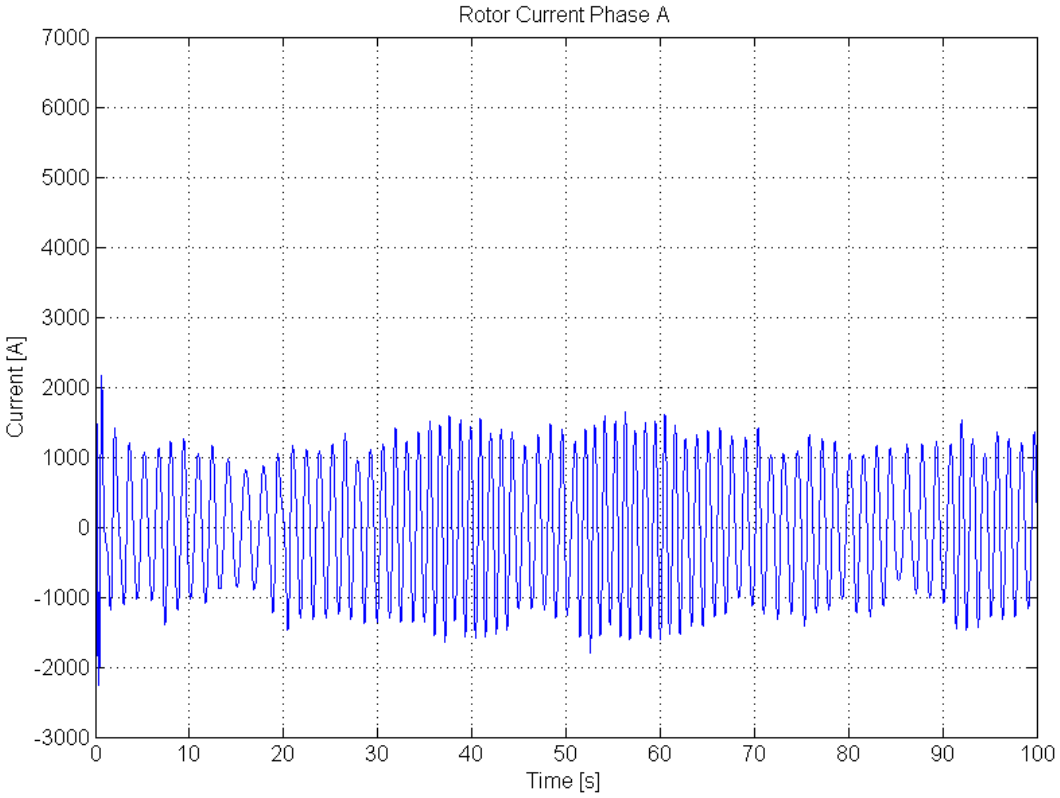


Figure G-12: Simulated A-phase rotor current for real wind speed input signal, output of generator model.

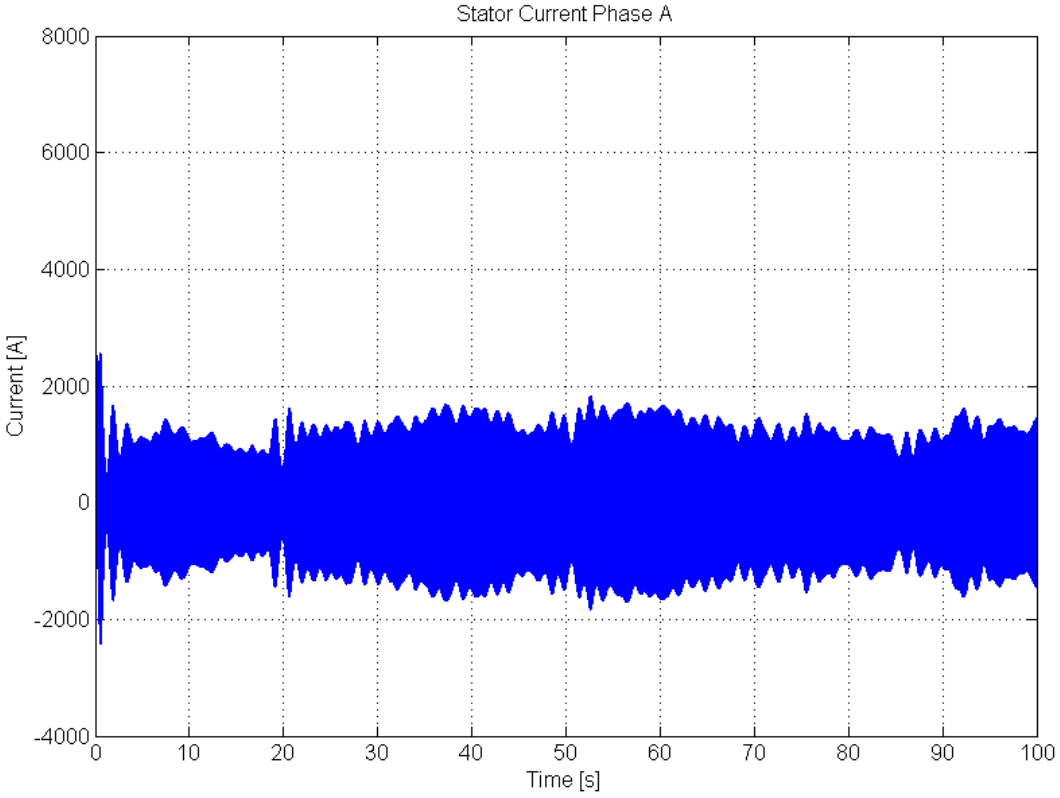


Figure G-13: Simulated A-phase stator current for real wind speed input signal, output of generator model.

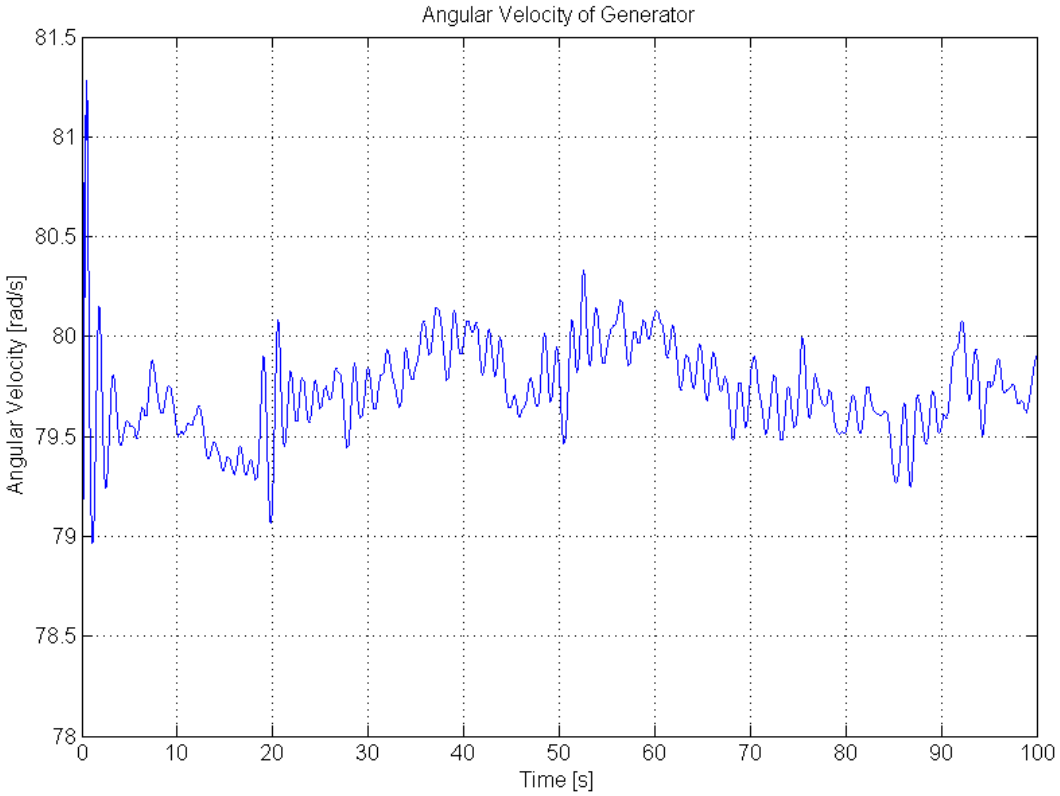


Figure G-14: Simulated generator angular velocity for real wind speed input signal, output gearbox model.

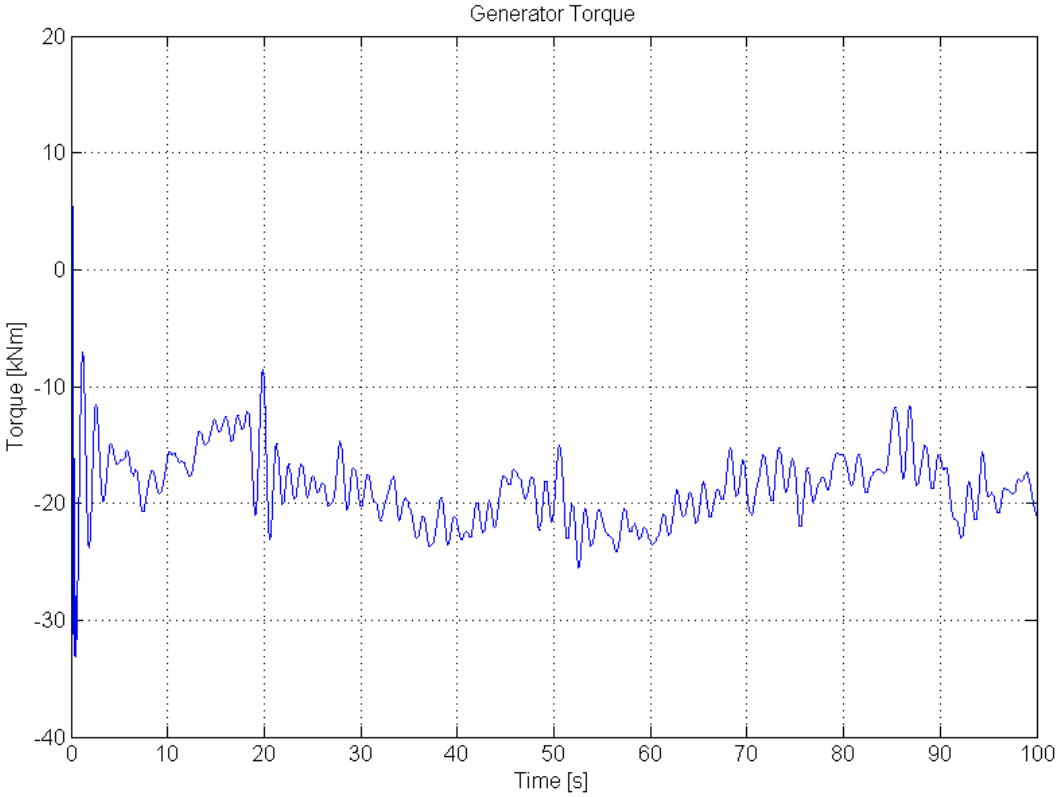


Figure G-15: Simulated generator torque for real wind speed input signal, output of generator model.

APPENDIX H: CONFIGURATION FOR PERFORMING PARAMETER ESTIMATION USING MATLAB'S COMMAND LINE

This appendix presents the code to performing parameter estimation on the model shown in Figure 4-14. The code configures the *Estimation* object to estimate all the generator parameters, except the number of poles with boundaries set. The code also configures the *Simulation Option* and *Optimisation Options*. The estimation is perform where after all the data of the estimation is saved as a MAT-file with the date and time as file name.

```
clc; clear all; close all
%% Open Model
open Tesis_TotalSystemABC_Est_system_DFIG_Outputs.mdl
modelName = 'Tesis_TotalSystemABC_Est_system_DFIG_Outputs'

%% Load experimental data.
load ('DataSetTesis_WTS_BackEx.mat','-mat')

%% Configure the Simulation Options object
JCB_sim_opt = simset();
JCB_sim_opt.MaxStep = 0.0001;
JCB_sim_opt.RelTol = 1e-3;
JCB_sim_opt.Solver = 'ode23s';

%% Window configuration
timeWindowStart = 10;
timeWindowStop =  12;
simtime = timeWindowStop;

%% Assigning the actual values of parameters from the experimental ...
%%  data to variables
%Generator Parameters
P_true = P;
Lr_true = Lr;
Ls_true = Ls;
Rr_true = Rr;
Rs_true = Rs;
M_true  = M;
a_true = a;
b_true = b;
c_true = c;
d_true = d;
e_true = e;
f_true = f;
g_true = g;
% Gearbox Parameters
Igen_true = Igen;
Itur_true = Itur;
K_true = K;
D_true = D;
GearRatio_true= GearRatio;
icga_true = icga;
icta_true = icta;
icgs_true = icgs;
```

```
icts_true = icts;
% Turbine Blade Parameters
BladeRadius_true = BladeRadius;
CutInSpeed_true = CutInSpeed;
CutOutSpeed_true = CutOutSpeed;
Cpp2_true = Cpp2;

%% Assigning the initial guess values of parameters
%Generator
P_init = 4;
Lr_init = 0.0001;
Ls_init = 0.0001;
Rr_init = 0.001;
Rs_init = 0.001;
M_init  = 0.001;
a_init = a_true;
b_init = b_true;
c_init = c_true;
d_init = d_true;
e_init = e_true;
f_init = f_true;
g_init = g_true;
% Gearbox
Igen_init = Igen_true;
Itur_init = Itur_true;
K_init = K_true;
D_init = D_true;
GearRatio_init = GearRatio_true;
icts_init = icts_true;
icgs_init = icgs_true;
% Turbine Blade
BladeRadius_init = BladeRadius_true;
CutInSpeed_init  = CutInSpeed_true ;
CutOutSpeed_init = CutOutSpeed_true;
Cpp2_init = Cpp2_true;

%% Assigning the initial guess values to the model variables
% Generator
P  = P_init;
Lr = Lr_init;
Ls = Ls_init;
Rr = Rr_init;
Rs = Rs_init;
M  = M_init;
a  = a_init;
b  = b_init;
c  = c_init;
d  = d_init;
e  = e_init;
f  = f_init;
g  = g_init;
% Gearbox
Igen = Igen_init;
Itur = Itur_init;
K = K_init;
D = D_init;
GearRatio = GearRatio_init;
icga = icga_init;
icta = icta_init;
```

```
icgs = icgs_init;
icts = icts_init;
% Turbine Blade
BladeRadius = BladeRadius_init;
CutInSpeed = CutInSpeed_init;
CutOutSpeed = CutOutSpeed_init;
Cpp2 = Cpp2_init;

%% Store True Parameter Values and State Values in matrix form
TrueParamterValues = [BladeRadius_true; -999; CutInSpeed_true;
                      CutOutSpeed_true; D_true;GearRatio_true;
Igen_true;...
                      Itur_true; K_true; Lr_true; Ls_true; M_true;...
                      P_true; Rr_true; Rs_true];

TrueState1Values = [a_true;b_true;c_true;d_true;e_true;f_true;g_true];
TrueState2Values = [icga_true; icta_true; icgs_true; icts_true];

clear *_init *_true

%% Create a Estimation object for modelName model
est = ParameterEstimator.Estimation(modelName)

%% Configuration of Parameter objects by setting estimation flags and
%% minimum and maximum values for all parameters
% Configuration of parameter: BladeRadius.
est.Parameters(1).Estimated = false;
set(est.Parameters(1), 'Minimum',  0, 'Maximum', 1);

% Configuration of parameter: Cpp2.
est.Parameters(2).Estimated = false;
set(est.Parameters(2), 'Minimum', 0, 'Maximum', 1);

% Configuration of parameter: CutInSpeed.
est.Parameters(3).Estimated = false;
set(est.Parameters(3), 'Minimum',  0, 'Maximum', 1);

% Configuration of parameter: CutOutSpeed.
est.Parameters(4).Estimated = false;
set(est.Parameters(4), 'Minimum',  1, 'Maximum', 16);

% Configuration of parameter: D.
est.Parameters(5).Estimated = false;
set(est.Parameters(5), 'Minimum', 377000, 'Maximum', 1134000);

% Configuration of parameter: GearRatio.
est.Parameters(6).Estimated = false;
set(est.Parameters(6), 'Minimum', 0, 'Maximum', 1);

% Configuration of parameter: Igen
est.Parameters(7).Estimated = false;
set(est.Parameters(7), 'Minimum', 45, 'Maximum', 135);

% Configuration of parameter: Itur
est.Parameters(8).Estimated = false;
set(est.Parameters(8), 'Minimum', 2475000, 'Maximum', 7425000);
```

```
% Configuration of parameter: K
est.Parameters(9).Estimated = false;
set(est.Parameters(9), 'Minimum', 57e6, 'Maximum', 1.72e8);

% Configuration of parameter: Lr
est.Parameters(10).Estimated = true;
set(est.Parameters(10), 'Minimum', 0.0001, 'Maximum',0.1);

% Configuration of parameter: Ls
est.Parameters(11).Estimated = true;
set(est.Parameters(11), 'Minimum', 0.0001, 'Maximum', 0.1);

% Configuration of parameter: M
est.Parameters(12).Estimated = true;
set(est.Parameters(12), 'Minimum', 0.001, 'Maximum', 0.1);

% Configuration of parameter: P
est.Parameters(13).Estimated = false;
set(est.Parameters(13), 'Minimum', 0, 'Maximum', 6.5);

% Configuration of parameter: Rr
est.Parameters(14).Estimated = true;
set(est.Parameters(14), 'Minimum', 0.001, 'Maximum', 0.1);

% Configuration of parameter: Rs
est.Parameters(15).Estimated = true;
set(est.Parameters(15), 'Minimum', 0.001, 'Maximum', 0.1);

%% Configuration of States objects by setting estimation flags and minimum
%% and maximum values for all parameters
%Generator States             a    b    c    d    e    f     g
est.States(1).Estimated = [   0    0    0    0    0    0     0  ];
est.States(1).Minimum   = [-inf -inf -inf -inf -inf -inf     0  ];
est.States(1).Maximum   = [ inf  inf  inf  inf  inf  inf   6.283];

%Gearbox States            icga  icta icgs icts
est.States(2).Estimated = [   0    0       0    0 ];
est.States(2).Minimum   = [   0    0    -inf -inf ];
est.States(2).Maximum   = [ 6.283 6.283  inf  inf ];

%% Window experimental data
window = (timeWindowStart<=Time_System & Time_System<=timeWindowStop);
TorqueGen = TorqueGen.*window;

CurrentOutput = [Current_stator Current_rotor];
window = [window window window window window window];
CurrentOutput=CurrentOutput.*window;

%% Create Transient Experiment Object
exp_data = ParameterEstimator.TransientExperiment(modelName);
% Assign input data of experiment
VoltageInput = [VoltageStator VoltageRotor];
set(exp_data.InputData(1), 'Data', VoltageInput, 'Time', Time_System);
set(exp_data.InputData(2), 'Data',  WindSpeed, 'Time', Time_System);
% Assign output data of experiment
set(exp_data.OutputData(1), 'Data', CurrentOutput, 'Time', Time_System);
set(exp_data.OutputData(2), 'Data', TorqueGen, 'Time', Time_System);
% Assign Transient Experiment to Experiments property of Estimation object
```

```
est.Experiments = exp_data

%% Configuration of Optimisation Options
% Configure to display estimation iterations.
est.OptimOptions.Display ='iter';

% Configure solver used by optimisation algorithm
est.SimOptions.Solver = 'ode23s';
est.SimOptions.maxStep = 0.0001;

 est.OptimOptions.TolX          =1e-5;
 est.OptimOptions.TolFun        =1e-5;

% Clear all variables not required for estimation process
save('temp.mat','VoltageInput','VoltageRotor','VoltageStator',...
    'AngVelGen', 'AngVelTur', 'CurrentOutput', 'Current_rotor', ...
    'Current_stator','TorqueGen','TorqueTur', 'Time_System','var*',...
    'window','WindSpeed')
clear VoltageInput VoltageRotor VoltageStator WindSpeed AngVelGen...
      AngVelTur CurrentOutput Current_rotor Current_stator TorqueGen...
      TorqueTur Time_System var55 var77 var88 window

%% Perform Estimation Process
tic
est.estimate
est_time = toc

% Assign Estiamted state values to original variables
% Generator
a = est.State(1).value(1);
b = est.State(1).value(2);
c = est.State(1).value(3);
d = est.State(1).value(4);
e = est.State(1).value(5);
f = est.State(1).value(6);
g = est.State(1).value(7);
% Gearbox
icga = est.State(2).value(1);
icta = est.State(2).value(2);
icgs = est.State(2).value(3);
icts = est.State(2).value(4);

% Load all variables
load('temp.mat','-mat','VoltageInput','VoltageRotor','VoltageStator',...
    'AngVelGen', 'AngVelTur', 'CurrentOutput', 'Current_rotor', ...
    'Current_stator','TorqueGen','TorqueTur','WindSpeed', 'Time_System',...
    'var*')

%% Save all data as mat-file with date and time as file name
filename = [datestr(now) '.mat'];
filename = regexprep(filename,':','_');
filename = regexprep(filename,'-','');
filename = regexprep(filename,' ','_');
save(filename)
close all
```

APPENDIX I: ADDITIONAL ESTIAMTION CASE STUDY RESULTS

I.1 Overview

This appendix presents results obtained for additional parameter estimation case studies performed on models presented in Chapter 5.

I.2 Results of additional parameter estimation case studies performed on ABC DFIG model

I.2.1 Overview

This section presents the results obtained from two additional case studies that was performed as an extension of two case studies performed in section 5.3.4 on the ABC DFIG. The first is an extension of the second case study performed on the ABC DFIG and the second is an extension of the fifth case study.

I.2.2 Results of extension of the second case study performed on the ABC DFIG model

This case study investigated whether the start-up transients of the simulation causes the second estimation to produce bad results when all generator parameters were estimated without estimating the initial states. This was achieved by applying a window to the data, to window out the start-up transients. The window spanned from 2 s to 10 s. The results obtained from this estimation are given in Table I-1. These results were obtained after 14 iterations that took 7 minutes to complete.

Table I-1: Parameter estimation results for the extension of second case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.0023	-36.90%
L_s [H]	✓	0.00165	0.0001	0.000905	45.13%
M [H]	✓	0.0466	0.0001	0.019951	57.19%
P	×	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.213345	-15.95%
R_s [Ω]	✓	0.115	0.0001	0.00343	97.02%
I_{as_init} [A]	×	10	0	0	-

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
I_{bs_init} [A]	×	10	0	0	-
I_{cs_init} [A]	×	10	0	0	-
I_{ar_init} [A]	×	10	0	0	-
I_{br_init} [A]	×	10	0	0	-
I_{cr_init} [A]	×	10	0	0	-
θ_{r_init} [rad]	×	1	0.5	0.5	-

✓ denotes parameters estimated and × denotes parameters not estimated

I.2.3 Results of extension of the fifth case study performed on the ABC DFIG model

This study investigated whether system that is excited by the stator voltage is more sensitive to the initial values of the state currents than then excited by the step in angular velocity. The same configuration was used as for the fifth case study but the initial values of the current states were also set to be estimated. The results obtained from this estimation are given in Table I-2. These results were obtained after 39 iterations that took 34 minutes to complete.

Table I-2: Parameter estimation results for the extension of fifth case study performed on the ABC DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.0002992	0.0001	0.0002656	11.25%
L_s [H]	✓	0.0004074	0.0001	0.0004399	-7.97%
M [H]	✓	0.016	0.0001	0.0144813	9.49%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.0089186	-0.21%
R_s [Ω]	✓	0.005	0.0001	0.0050227	-0.45%
I_{as_init} [A]	✓	1	10	108.75733	-10776%
I_{bs_init} [A]	✓	1	10	-263.1621	26416%
I_{cs_init} [A]	✓	1	10	157.53032	-15653%
I_{ar_init} [A]	✓	1	10	149.94371	-14894%
I_{br_init} [A]	✓	1	10	124.104299	-12310%
I_{cr_init} [A]	✓	1	10	-270.8396	27184%
θ_{r_init} [rad]	✓	1	0.5	0.9967851	0.32%

✓ denotes parameters estimated and × denotes parameters not estimated

I.3 Results of additional parameter estimation case study performed on DQ DFIG model

I.3.1 Overview

This section presents the results obtained from additional case studies that were performed as an extension of case studies performed in section 5.3.5 on the DQ DFIG. The first two are an extension of the second case study performed on the DQ DFIG and the final two is an extension of the third case study.

I.3.2 Results of extension of the second case study performed on the DQ DFIG model

Two additional estimation process were performed on the DQ DFIG to try to improve the accuracy of the results obtained from the second case study. Table I-3 shows the results obtained for the case study performed with all the data remaining the same as the original case study. The only change made to the configuration was the initial values of the current states that were assigned their actual values. The estimation required 20 iterations that took 174 seconds to complete.

Table I-3: Parameter estimation results for the first extension estimation of second case study performed on the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual value	Initial Guess	Estimated Value	Persentage Error
L_r [H]	✓	0.00168	0.0001	0.001857	-10.52%
L_s [H]	✓	0.00165	0.0001	0.001447	12.30%
M [H]	✓	0.0466	0.0001	0.048588	-4.27%
P	✓	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.184014	-0.01%
R_s [Ω]	×	0.115	0.0001	0.118526	-3.07%
I_{ds_init} [A]	×	10	10	10	-
I_{qr_init} [A]	×	10	10	10	-
I_{dr_init} [A]	×	10	10	10	-
I_{qr_init} [A]	×	10	10	10	-
θ_{r_init} [rad]	✓	1	2	0.997913	0.21%

✓ denotes parameters estimated and × denotes parameters not estimated

Table I-4 shows the results obtained for the case study that was performed with all the data and configurations the same as the original case study, but with the data window removed. The parameter estimation process required 21 iterations that took 185 seconds to complete.

Table I-4: Parameter estimation results for the second extension estimation of second case study performed on the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.00168	0.0001	0.002672	-59.07%
L_s [H]	✓	0.00165	0.0001	0.000756	54.17%
M [H]	✓	0.0466	0.0001	0.474372	-917.97%
P	✓	4	4	4	-
R_r [Ω]	✓	0.184	0.0001	0.190614	-3.59%
R_s [Ω]	×	0.115	0.0001	0.121892	-5.99%
I_{ds_init} [A]	×	10	1	1	-
I_{qr_init} [A]	×	10	1	1	-
I_{dr_init} [A]	×	10	1	1	-
I_{qr_init} [A]	×	10	1	1	-
θ_{r_init} [rad]	✓	1	2	0.995675	0.43%

✓ denotes parameters estimated and × denotes parameters not estimated

I.3.3 Results of extension of the third case study performed on the DQ DFIG model

The following studies investigated the results obtained then the DQ DFIG model is excited the stator voltages. The input signals, output signals and configuration that was used are provided in section 5.3.5.4 .

The results displayed in Table I-5 are those obtained when the initial values of the current states are assigned random values and set not to be estimated. These results were obtained after 27 iterations that took 494 seconds to complete.

Table I-5: Parameter estimation results for the first extension estimation of third case study performed on the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.000299	0.0001	0.0002132	28.75%
L_s [H]	✓	0.000407	0.0001	0.0004862	-19.33%
M [H]	✓	0.016	0.0001	0.01162069	27.37%
P	×	4	4	4	-

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
R_r [Ω]	✓	0.0089	0.0001	0.00889992	0.00%
R_s [Ω]	✓	0.005	0.0001	0.00505738	-1.15%
I_{ds_init} [A]	×	1	10	10	-
I_{qr_init} [A]	×	1	10	10	-
I_{dr_init} [A]	×	1	10	10	-
I_{qr_init} [A]	×	1	10	10	-
θ_{r_init} [rad]	✓	1	0.5	1.01128202	-1.13%

✓ denotes parameters estimated and × denotes parameters not estimated

The results displayed in Table I-6 are those obtained when the initial values of the current states are assigned random values and set to be estimated. These results were obtained after 45 iterations that took 22 minutes to complete.

Table I-6: Parameter estimation results for the second extension estimation of third case study performed on the DQ DFIG model.

Parameter/ Initial State	Estimation Flag	Actual Value	Initial Guess	Estimated Value	Percentage Error
L_r [H]	✓	0.0002992	0.0001	0.0002788	6.82%
L_s [H]	✓	0.0004074	0.0001	0.0004263	-4.62%
M [H]	✓	0.016	0.0001	0.0149369	6.64%
P	×	4	4	4	-
R_r [Ω]	✓	0.0089	0.0001	0.0089001	0.00%
R_s [Ω]	✓	0.005	0.0001	0.0050006	-0.01%
I_{ds_init} [A]	✓	1	10	13.896654	-1290%
I_{qr_init} [A]	✓	1	10	-15.40606	1641%
I_{dr_init} [A]	✓	1	10	-11.59325	1259%
I_{qr_init} [A]	✓	1	10	17.744857	-1674%
θ_{r_init} [rad]	✓	1	0.5	1.002238	-0.22%

✓ denotes parameters estimated and × denotes parameters not estimated