

Technical University of Denmark



Wind turbine blockset in Saber. General overview and description of the models

Iov, Florin; Timbus, Adrian Vasile; Hansen, Anca Daniela; Sørensen, Poul Ejnar; Blaabjerg, F.

Publication date:
2004

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Iov, F., Timbus, A. V., Hansen, A. D., Sørensen, P. E., & Blaabjerg, F. (2004). Wind turbine blockset in Saber. General overview and description of the models. Aalborg: Aalborg University.

DTU Library

Technical Information Center of Denmark

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Abstract. This report presents a new developed Saber Toolbox for wind turbine applications. This toolbox has been developed during the research project “Simulation Platform to model, optimize and design wind turbines”. The report provides a quick overview of the Saber and then explains the structure of this simulation package, which is different than other tools e.g. Matlab/Simulink. Then the structure of the toolbox is shown as well as the description of the developed models. The main focus here is to underline the special structure of the models, which are a mixture of Saber built-in blocks and new developed blocks. Since the developed models are based on Saber built-in blocks, a description of the libraries from Saber is given. Then some simulation results using the developed models are shown. Finally some general conclusions regarding this new developed Toolbox as well as some directions for future work are made.

Table of Content

Table of Content	3
Preface	4
Chapter 1 Introduction	5
Chapter 2 SABER® - An overview	6
2.1 SaberSketch	6
2.2 Model Architect Tool	7
2.3 MAST	9
2.4 Saber Scope	10
Chapter 3 Saber Toolbox for Wind Turbine Applications	11
3.1 Wind Turbine Blockset in Saber	11
3.2 Wind model	12
3.3 Wind turbine rotor model	14
3.4 ABC/abc model of induction machine	16
3.5 Power converters	20
3.5.1 AC Controllers (Soft-Starter)	20
3.5.2 Back to back power converter	23
3.6 Modulation strategies for power converters	24
3.6.1 PWM sinusoidal modulator using sine-triangle with third harmonic (STTH)	24
3.6.2 Space Vector Modulation	24
3.7 Summary	26
Chapter 4 Description of Saber's built-in libraries	27
4.1 Mechanisms library	27
4.2 Generators library	27
4.3 Motors library	29
4.4 Transformers library	30
4.5 Control System library	31
4.6 Cables & Wires library	32
4.7 Summary	33
Chapter 5 Simulation results	34
Start-up sequence of a soft-starter-fed induction machine	34
Conclusions and Future work	36
References	37
Appendix A MAST Template for developed ABC/abc Model of Induction Machine	38

Preface

This report describes the *Wind Turbine Blockset* developed in Saber during the project “A Simulation Platform to Model, Optimize and Design Wind Turbines”. The project has been funded by the Danish Energy Agency contract #ENS-1363/01-0013, and it was carried out in cooperation between Aalborg University and RISØ National Laboratory.

Chapter 1

Introduction

Saber is a simulation tool used in power circuit and systems design including electrical, thermal, magnetic and mechanical components. The wind turbine systems are an example of such hybrid systems, involving mechanical, electromagnetical and power electronics subsystems.

Currently, this simulation package is used in the automotive, aerospace, power and IC industry to simulate and analyze systems, sub-systems and components to reduce the need for prototypes. However, this tool is currently not focused on the wind turbine applications.

This report presents a new developed Saber Toolbox for wind turbine applications. This toolbox has been developed during the research project “Simulation Platform to model, optimize and design wind turbines”. The new developed models from this Toolbox have been first developed and verified in Matlab/Simulink and then implemented in Saber.

The report provides first a quick overview over Saber issues and how can be developed new models. Then explains the structure and mathematical models of the new developed components. Since these models involves both new developed blocks as well as the built-in blocks from Saber libraries a short description of these built-in libraries is given.

Some simulation results using the developed models are shown.

Chapter 2

SABER® - An overview

Saber is a simulation tool used in power circuit and systems design including electrical, thermal, magnetic and mechanical components [1].

Saber is e.g. used in the automotive, aerospace, power and IC industry to simulate and analyze systems, sub-systems and components to reduce the need for prototypes.

Some of the features of this simulation package can be summarized as follows:

- Saber can simulate physical effects in a wind turbine system (hydraulic, electric, electronic, mechanical, etc.);
- It has one of the largest model library in the industry. New models can easily be built using the existing blocks (graphical modelling), Model Architect Tool and MAST language;
- Saber supports: continuous analog models, event-driven systems (including digital), data flow (control systems);
- It can handle electric, magnetic, thermal and mechanical variables;
- Different analysis can be performed in Saber: DC Operating Point/Transfer, Time-domain Transients, Frequency: Small signal AC/Distortion/Noise etc, Linear System Analysis: Pole-Zero/ Frequency response, etc. Parametric: Vary/Sensitivity, Statistical: Monte Carlo/Histogram etc., Stress, Fourier, etc.

Saber includes basically two tools namely SaberSketch and SaberScope. SaberSketch is used to build a simulation diagram while SaberScope is used to analyse the simulation results.

2.1 SaberSketch

SaberSketch contains of a schematic editor and a symbol editor [3]. The user can create a schematic in the schematic editor and then, if the user are using the schematic block as part of a larger system, SaberSketch is used to generate a symbol for the schematic.

Fig. 2.1 shows the major user interface items of the SaberSketch tool. Different tools are available in this interface such as:

- Analysis toolbar - include all different analysis which can be performed in Saber;
- Schematic window – a window where the user creates the simulation diagram;
- Command line interface – allows the user to enter AIM commands, write scripts etc. AIM is a scripting language, which is used to write user defined algorithms and interfaces for Saber;
- Parts Library – gives access to the extensive MAST parts libraries, allowing the user to select and place parts in schematic window;

- Macro recorder – this tool records a series of actions performed in Guide, Scope, Sketch, and the Saber simulator, allows the user to edit these actions, and plays them back as a script.
- Model Architect Tool- consists of characterization tools that are utilities for the creation and characterization of models. The output of a characterization tool consists of a MAST template and an application file so that the model can be loaded back into the tool.
- Drawing Tool – used to draw masks for the new developed blocks.

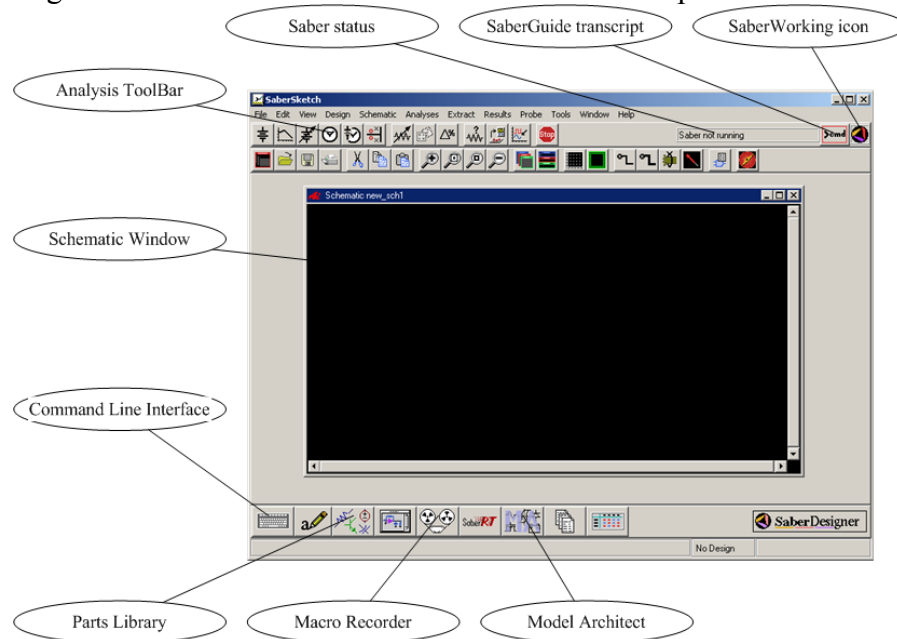


Fig. 2.1. SaberSketch User Interface.

2.2 Model Architect Tool

Model Architect consists of characterization tools that are utilities for the creation and characterization of models as shown in Fig. 2.2 [4].

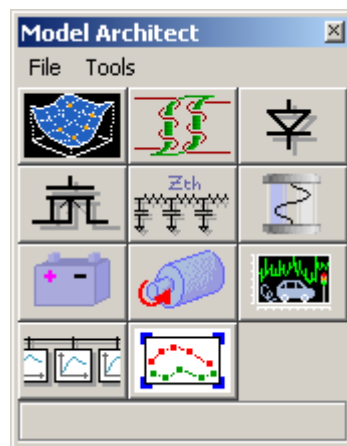


Fig. 2.2. Model Architect Interface in Saber.

The output of a characterization tool consists of a MAST template and an application file so that the model can be loaded back into the tool. These characterization tools allow the user to edit a model or inspect its behaviour before running a simulation. Model Architect includes the following Characterization Tools:

Table 1.1. Characterization Tools from Model Architect

Tool	Function
Table Look-Up Modeling Tool	An interface used to create models from measurement, datasheets, or other simulation softwares for finite element analysis.
Scanned Data Utility	An interactive method of importing data from a scanned graph, thus avoiding point-by-point coordinate entry into a spreadsheet. The utility handles non-orthogonal scanned graphs, linear or logarithmic axis scales, and parametric curves.
Magnetic Component Characterization Tool	Provides interactive characterization for non-linear magnetic core models and windings to produce magnetic components with variable geometry, eddy current losses, flux leakage, and capacitances.
Diode Characterization Tool	Provides support to create diode models for use in power electronic circuits. These models are well suited for examining switching transients and losses in power supplies.
Power MOSFET Characterization Tool	Provides support to generate level-1 MOSFET models for power electronic circuits which are well suited for examining switching transients and losses in power supplies. Models are easily characterizable from datasheet information. The tool allows interactive tuning of parameters.
Thermal Characterization Tool	The Thermal Tool provides support to characterize the Saber thermal model.
Fuse Characterization Tool	Provides an easy interactive way to characterize the fuse model available in the simulator. The graphical interface is also useful in understanding the functionality of the model.
Battery Characterization Tool	The Battery Tool provides support to characterize the Saber behavioral lead-acid battery model, <code>batt_pb_1</code> .
DCPM Motor Characterization Tool	The Motor Tool provides support to characterize the Saber behavioral DC Permanent Magnet Motor model, <code>dc_pm2</code> .
Drive Cycle Editor	Characterizes vehicle speed as a function of time.
Load Profile Editor	Characterizes resistance, power, and current load models as a function of either speed or time.

Using these tools the desired model can be easily build.

2.3 MAST

The MAST language is a unique concept in simulator input—the same language describes models of elements, of subsystems, and of full systems [5], [6]. A system model can be as simple as a netlist that describes the interconnections of existing system components (using pre-defined models from a MAST library) or as complex as the full system description (using no pre-defined models).

This language lets the user to create a model of any analog, discrete or control system that can be defined in terms of nonlinear “lumped” algebraic or differential equations. Some extensions are also provided by the use of ideal delay and scheduling. A design can be a combination of more than one functional block to comprise a complete system.

With the MAST language and the Saber simulator, the user can model and simulate most physical systems: electronic, mechanical, optical, hydraulic, etc. (or any combination of them).

Designers can create their own MAST models or rely on a comprehensive library of pre-existing MAST models. Once a device or design has been described with MAST, the behavior of the model can be simulated with the Saber simulator.

MAST provides two different styles for creating models: structured and unstructured. The structured approach divides the model into sections. Each section contains specific model functionality. This modeling style is recommended for complicated models. Most models fall into this category. The unstructured modeling style is beneficial for use with simple models and can be created with less lines of code. Although the term “unstructured” implies a lack of model structure, this is not the case. The model’s structure is made simpler by eliminating the need to partition different model functionality into specific model sections.

At minimum, the Saber simulator requires (for each system to be simulated) an input file containing a netlist that completely describes the system in the MAST language.

The process of replacing a system with a simplified model and then investigating its properties by controlled experiments is called *simulation*. In order to obtain meaningful simulation results, the model must accurately represent the original system within the constraints of those experiments. Beyond that, the model and the original system may exhibit completely different behavior. This means that a system can be represented by a variety of different models, each having a specific purpose and a set of related limitations.

In computer simulation, the model typically consists of a mathematical description of the system properties, and each controlled experiment is an analysis of a particular type. For example, a transient analysis is typically used to investigate the response of the system to a time-dependent excitation. To yield meaningful simulation results for all types of excitation, the system model must include accurate descriptions (such as nonlinearities and time constants) of the system's dynamic properties. On the other hand, a linear approximation of the system (if it exists) can be sufficiently accurate for feasibility studies. Most simulators also use a linearized model for small signal AC analysis to determine the frequency response of the system. They generate such a model by linearizing a more accurate nonlinear model about the operating point of the system.

Unlike other simulators that flatten a hierarchical system description, the Saber simulator exploits system hierarchy when it solves the system equations. Thus, the computational effort to simulate a hierarchical system grows sublinearly with increasing system size—with most

other simulators the growth rate is close to quadratic. Therefore, hierarchical system descriptions generally improve simulation efficiency. This rule, however, is not absolute. If a subsystem described hierarchically is very small, the overhead required to handle the hierarchy may exceed the benefit of having hierarchy. It is best to model such systems without introducing a new level of hierarchy.

2.4 Saber Scope

SaberScope is a graphical waveform analyzer tool that allows the user to view and analyze simulation results in the form of waveforms displayed on graphs, or as values displayed in lists as shown in Fig. 2.3 [7]. SaberScope includes the following tools:

- Signal Manager, through which the plotfiles are opened, filtered, and placed into a graph window or calculator;
- Measurement tool, which provides over 50 measurements that can be applied to a waveform;
- Waveform Calculator, which emulates a hand-held calculator that interacts graphically with the application;
- Command Line tool, which allows you to enter AIM commands, write scripts, and save them into files.
- SaberLink Analysis - provides an interface to MATLAB software applications, and enables data transfers between SaberDesigner applications and MATLAB applications. Text, waveforms, plot files, and Vector/Matrix/Arrays can be selected from other sources and pasted directly into the SaberLink window. The user can also write AIM language scripts to operate MATLAB through SaberLink. The Macro Recorder tool can be used to facilitate developing these scripts.

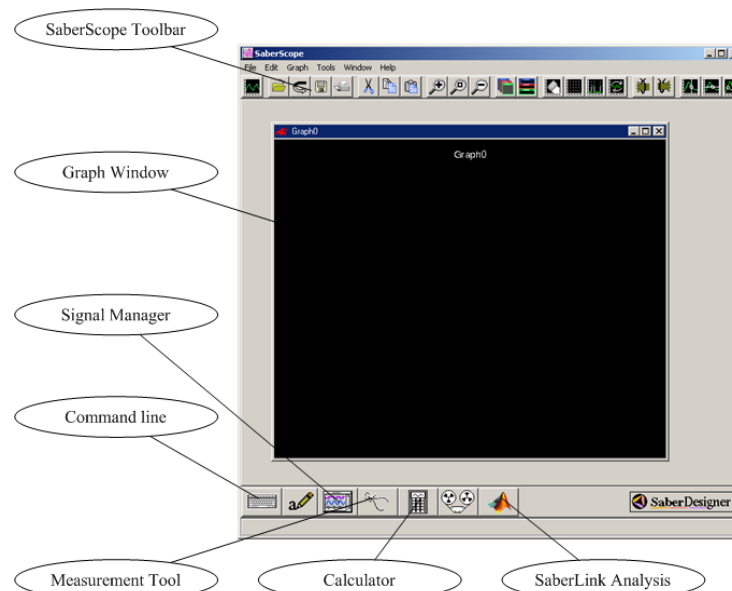


Fig. 2.3. SaberScope's main window.

Chapter 3

Saber Toolbox for Wind Turbine Applications

In this chapter the structure of the Saber Toolbox for wind turbine applications is presented. A detailed description of the new developed models is also given.

3.1 Wind Turbine Blockset in Saber

A Saber Toolbox for wind turbine applications is under development. Besides the built-in blocks from Saber: induction machine models, synchronous machine models, cables, etc. some specific blocks have been added. Since the modelling approach is different in Saber than in Matlab/Simulink the structure of this Toolbox is changed as shown in Fig. 3.1

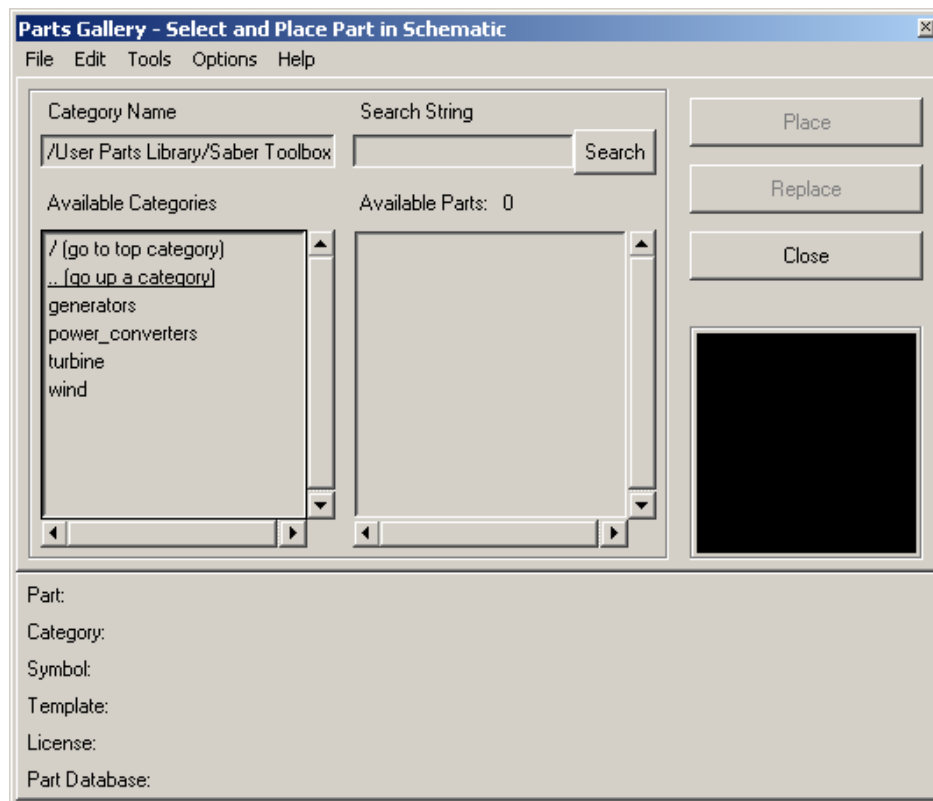


Fig. 3.1. Structure of the Saber Toolbox for wind turbine applications.

Due to the modelling procedure in Saber it has been decided to divide the developed models into four libraries. The models for generators are written in MAST language while; the wind model and the wind turbine rotor model are based on graphical modelling. Moreover, these models involves some blocks written also in MAST language or built-in blocks from Saber's libraries.

3.2 Wind model

The wind model has been developed at RISØ National Laboratory based on the Kaimal spectra [10]. The wind speed is calculated as an average value of the fixed-point wind speed over the whole rotor, and it takes the tower shadow and the rotational turbulences into account.

Since one of the main components in this model is the normally distributed white noise generator some investigations have been done in order to obtain the same wind time series in all considered simulation tools used in the Simulation Platform. It has been found that the built-in white noise generator from different simulation tools uses a different algorithm and thus a different wind time series is obtained. A new normally distributed white noise generator has been implemented using a ‘C’ S-Function based on the Ziggurat Algorithm developed by G. Marsaglia [11].

This new white noise generator block has been first implemented and verified in Matlab/Simulink [21] and then in Saber using a MAST template.

Currently, the wind model is available in 2 basic versions. The first one uses the normally distributed white noise generator from Saber, while the second one is based on the new developed normally distributed white noise generator.

The general structure of these models is shown in Fig. 3.2

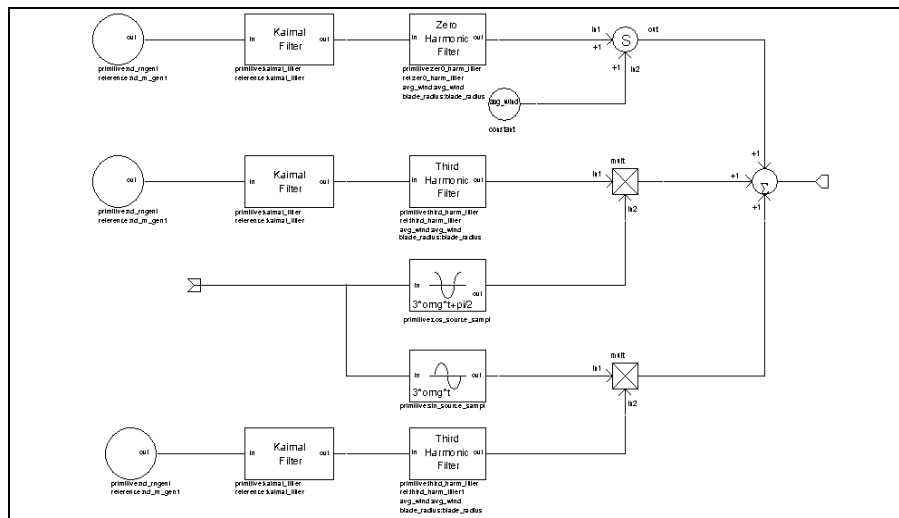


Fig. 3.2. Structure of the wind model in Saber.

The parameters defined in the block’s mask are: rotor diameter of the wind turbine, average wind speed, length scale, turbulence intensity and sample time as shown in Fig. 3.3.

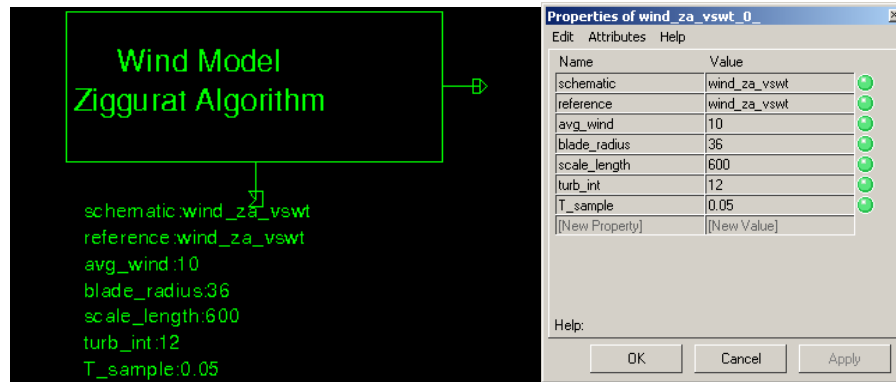


Fig. 3.3. Symbol and the mask interface for the wind model in Saber.

In order to validate the new white noise generator model some comparisons have been done. A wind time series for 3600 sec with 0.05 sec sample time, an average wind speed of 10 m/sec and 12% turbulence intensity has been generated for both models as shown in Fig. 3.4.

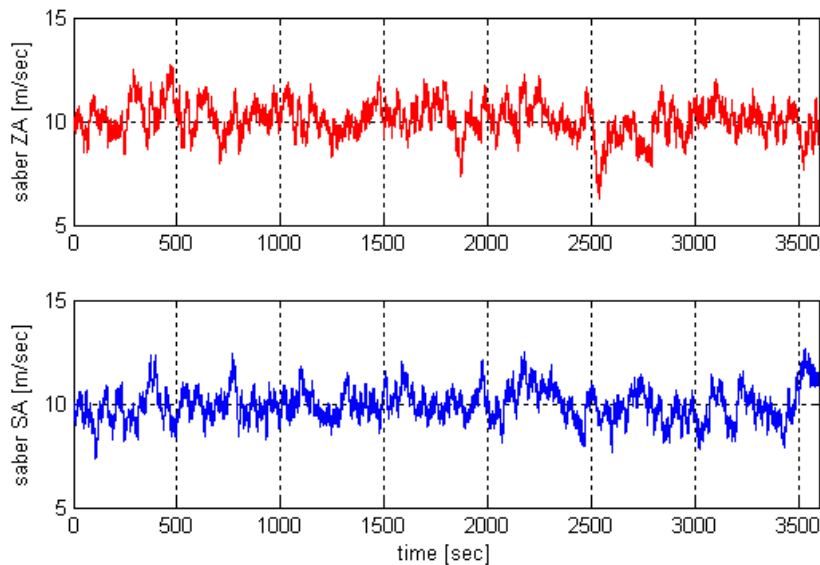


Fig. 3.4. Wind time series for the considered models (ZA - ziggurat algorithm, SA – built-in Saber block).

The 20 bin-width histograms for these wind time series are presented in Fig. 3.5

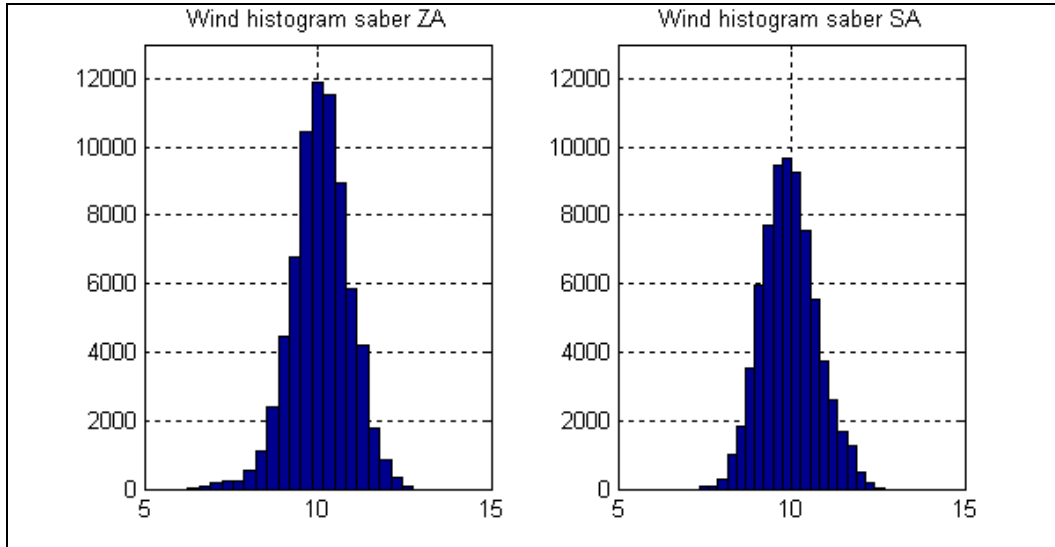


Fig. 3.5. Wind histograms for the considered models (ZA - ziggurat algorithm, SA – built-in Saber block).

Finally, the power spectra density for both wind time series has been calculated as shown in Fig. 3.6.

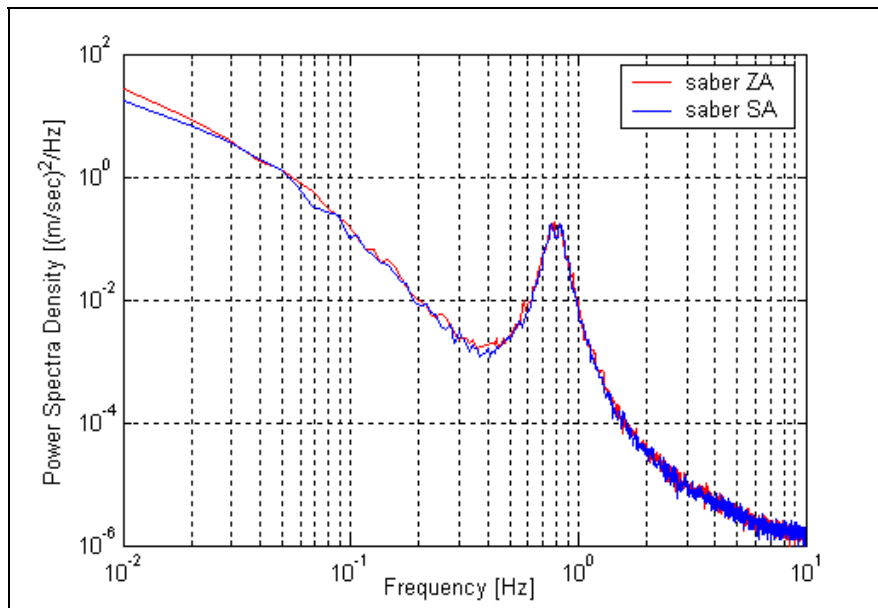


Fig. 3.6. Power spectra density for the considered models. (ZA - ziggurat algorithm, SA – built-in Saber block).

Even if the wind time series for the considered models are not identically as shown in Fig. 3.4 the power spectra density is approximately the same.

3.3 Wind turbine rotor model

The aerodynamic model of the wind turbine rotor is based on the torque coefficient C_Q look-up table. The torque coefficient C_Q is used to determine the aerodynamic torque directly by using:

$$T_{wt} = 0.5\pi\rho R^3 v_{\infty}^2 C_Q \tag{3.3.1}$$

where: ρ is the air density, R is the blade radius, v_{∞} is the wind speed and C_Q is the torque coefficient.

The Saber model is presented in Fig. 3.7.

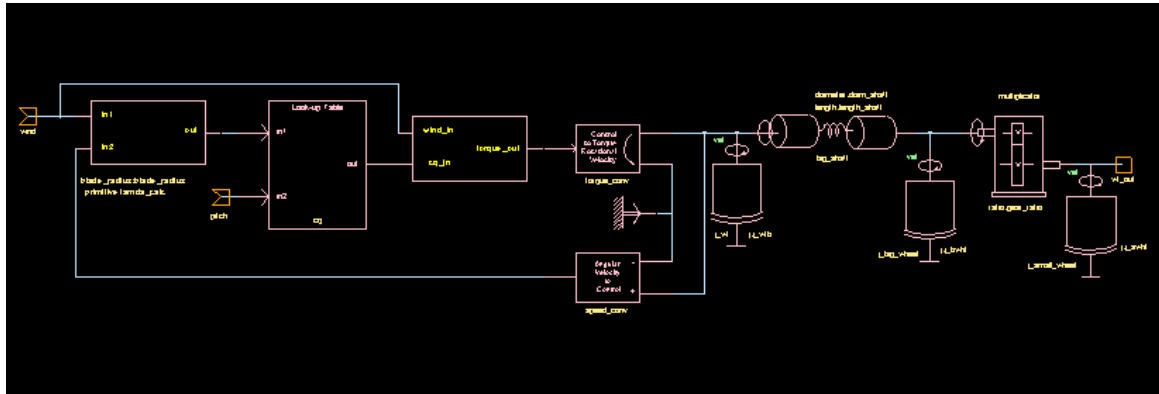


Fig. 3.7. Saber structure of the wind turbine rotor model.

This model involves some blocks written in MAST language (lambda calculator, torque calculator), a look-up table built with **Table Look-Up Modelling Tool** and some built-in blocks from Saber libraries such as flexible shaft model, gear-box model, inertia model, etc.. The symbol and the mask interface for this block are shown in Fig. 3.8.

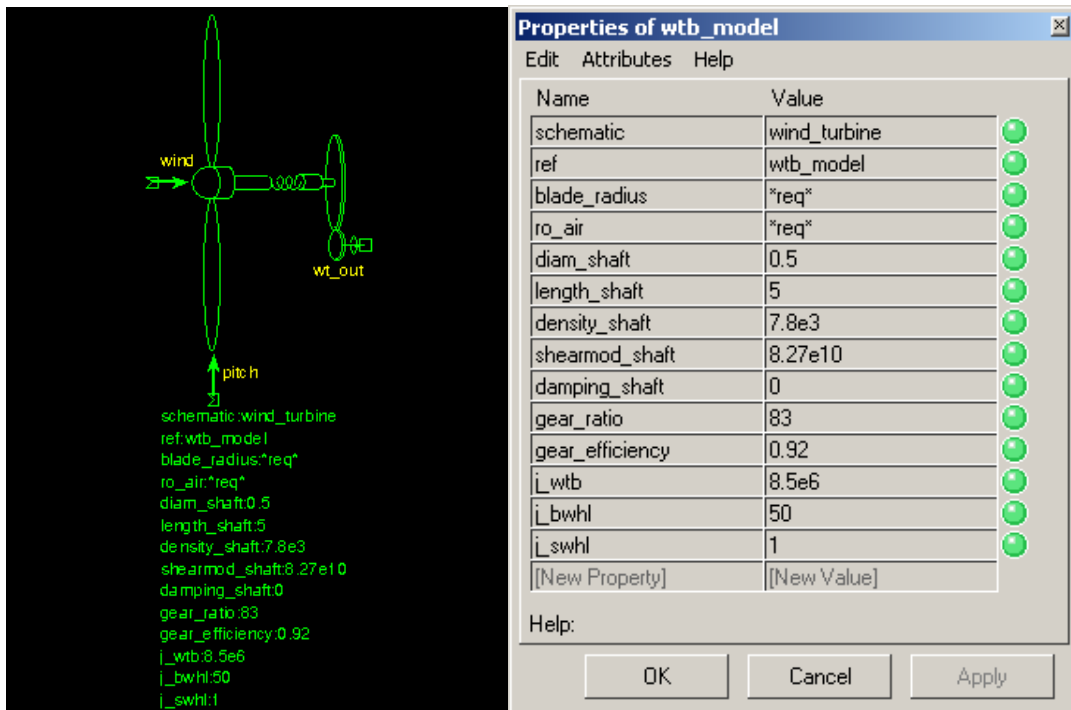


Fig. 3.8. The symbol and the mask interface for the wind turbine rotor model.

The parameters from the mask interface are: blade length, air density, length and diameter of the main shaft, density of material, shearing modulus of elasticity, valve internal damping, gear-box ratio and efficiency and moment of inertia for wind turbine rotor and gear-box wheels.

Using these models an accurate model for the wind turbine drive train can be obtained.

3.4 ABC/abc model of induction machine

Besides the built-in models for induction machines from Saber libraries a new *abc/abc* model for induction machine has been added. This model is similar with the Matlab/Simulink model and can be used both for a squirrel-cage or wound rotor machine with different connection types for the stator windings.

In the following the dynamic equations for this model are presented. Some other features include in the model e.g. deep-bar effect, saturation and iron losses will be included later.

The basic equations of the machine (assuming sinusoidal MMF and neglecting saturation and losses in the core) by using the diagram shown in Fig. 3.9 are [12]-[15]:

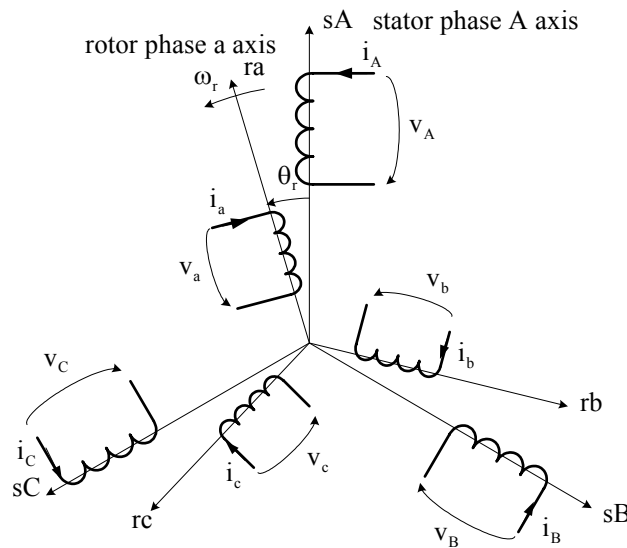


Fig. 3.9. Simplified diagram of induction machine with rotor and stator windings.

$$[V] = [R] \cdot [I] + \frac{d[\Psi]}{dt} \quad (3.4.1)$$

$$[\Psi] = [L] \cdot [I] \quad (3.4.2)$$

where:

- $[V] = [v_A \ v_B \ v_C \ v_a \ v_b \ v_c]^t$ - are the voltages applied to each stator and rotor phases;
- $[I] = [i_A \ i_B \ i_C \ i_a \ i_b \ i_c]^t$ - are the currents in each stator and rotor phases;
- $[\Psi] = [\varphi_A \ \varphi_B \ \varphi_C \ \varphi_a \ \varphi_b \ \varphi_c]^t$ - are the fluxes linked with each stator and rotor phases;
- $[R]$ is the resistance matrix and $[L]$ is the inductance matrix.

Substituting (3.4.2) in (3.4.1) and arranged the equation in state-variable form results:

$$\frac{d[\mathbf{I}]}{dt} = [\mathbf{L}]^{-1} \cdot \left\{ -[\mathbf{R}] - \frac{d[\mathbf{L}]}{dt} \right\} \cdot [\mathbf{I}] + [\mathbf{L}]^{-1} \cdot [\mathbf{V}] \quad (3.4.3)$$

Most terms of the inductance matrix $[\mathbf{L}]$ are functions of the rotor position θ_r . Taking this into account:

$$\frac{d[\mathbf{L}]}{dt} = \frac{d[\mathbf{L}]}{d\theta_r} \cdot \frac{d\theta_r}{dt} = \frac{d[\mathbf{L}]}{d\theta_r} \cdot \omega_r \quad (3.4.4)$$

where ω_r is the rotor speed. Grouping (3.4.3) and (3.4.4) results:

$$\frac{d[\mathbf{I}]}{dt} = [\mathbf{L}]^{-1} \cdot \left\{ -[\mathbf{R}] - \omega_r \cdot \frac{d[\mathbf{L}]}{d\theta_r} \right\} \cdot [\mathbf{I}] + [\mathbf{L}]^{-1} \cdot [\mathbf{V}] \quad (3.4.5)$$

which is the state space form of the dynamic equations for the induction machine.

The inductance matrix is defined as:

$$[\mathbf{L}] = \begin{bmatrix} L_s & -0.5M_s & -0.5M_s & M_{sr}f_1 & M_{sr}f_2 & M_{sr}f_3 \\ -0.5M_s & L_s & -0.5M_s & M_{sr}f_3 & M_{sr}f_1 & M_{sr}f_2 \\ -0.5M_s & -0.5M_s & L_s & M_{sr}f_2 & M_{sr}f_3 & M_{sr}f_1 \\ M_{sr}f_1 & M_{sr}f_3 & M_{sr}f_2 & L_r & -0.5M_r & -0.5M_r \\ M_{sr}f_2 & M_{sr}f_1 & M_{sr}f_3 & -0.5M_r & L_r & -0.5M_r \\ M_{sr}f_3 & M_{sr}f_2 & M_{sr}f_1 & -0.5M_r & -0.5M_r & L_r \end{bmatrix} \quad (3.4.6)$$

where:

- $L_s = L_{\sigma s} + M_s$ - stator self-inductance;
- $L_r = L_{\sigma r} + 1.5M_r$ - rotor self-inductance;
- $L_{\sigma s}, L_{\sigma r}$ - stator and rotor leakage inductance;
- M_s, M_r - stator and rotor mutual inductance;
- M_{sr} - stator-rotor mutual inductance.

The coefficients f_1, f_2, f_3 are:

$$f_1 = \cos \theta_r, \quad f_2 = \cos \left(\theta_r + \frac{2\pi}{3} \right), \quad f_3 = \cos \left(\theta_r - \frac{2\pi}{3} \right) \quad (3.4.7)$$

All these inductances can easily be measured in a wound rotor induction machine. For squirrel-cage induction machines, these inductances can normally be estimated from no-load and locked rotor tests. When the rotor parameters are referred to stator:

$$M_s = M_r = M_{sr} \quad (3.4.8)$$

If it is taken into account that:

$$i_A + i_B + i_C = 0 \quad \text{and} \quad i_a + i_b + i_c = 0 \quad (3.4.9)$$

the inductance matrix $[\mathbf{L}]$ can be rewritten as:

$$[L] = \begin{bmatrix} L'_s & 0 & 0 & M_{sr}f_1 & M_{sr}f_2 & M_{sr}f_3 \\ 0 & L'_s & 0 & M_{sr}f_3 & M_{sr}f_1 & M_{sr}f_2 \\ 0_s & 0 & L'_s & M_{sr}f_2 & M_{sr}f_3 & M_{sr}f_1 \\ M_{sr}f_1 & M_{sr}f_3 & M_{sr}f_2 & L'_r & 0 & 0 \\ M_{sr}f_2 & M_{sr}f_1 & M_{sr}f_3 & 0 & L'_r & 0 \\ M_{sr}f_3 & M_{sr}f_2 & M_{sr}f_1 & 0 & 0 & L'_r \end{bmatrix} \quad (3.4.10)$$

where: $L'_s = L_{cs} + 1.5M_s$ and $L'_r = L_{or} + 1.5M_s$.

From (3.4.10) the results are:

$$\frac{d[L]}{d\theta_r} = -M_{sr} \begin{bmatrix} 0 & 0 & 0 & g_1 & g_2 & g_3 \\ 0 & 0 & 0 & g_3 & g_1 & g_2 \\ 0_s & 0 & 0 & g_2 & g_3 & g_1 \\ g_1 & g_3 & g_2 & 0 & 0 & 0 \\ g_2 & g_1 & g_3 & 0 & 0 & 0 \\ g_3 & g_2 & g_1 & 0 & 0 & 0 \end{bmatrix} \quad (3.4.11)$$

$$\text{and: } g_1 = \sin \theta_r, \quad g_2 = \sin\left(\theta_r + \frac{2\pi}{3}\right), \quad g_3 = \sin\left(\theta_r - \frac{2\pi}{3}\right) \quad (3.4.12)$$

The main problem in implementing (3.4.5) is to calculate the inverse of the inductance matrix at each simulation step [13]. A solution is to use modern software tools, which have these capabilities (e.g. Matlab). However, it is useful to use an analytical expression for this matrix since the coefficients depend on the electrical angle θ_r . One approach is to apply a Cholesky decomposition since the inductance matrix is ‘‘symmetric positive definite’’[13].

Starting from the general form (3.4.10) the inverse of the inductance matrix can be written as:

$$[L]^{-1} = [B] = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} \\ b_{41} & b_{42} & b_{43} & b_{44} & b_{45} & b_{46} \\ b_{51} & b_{52} & b_{53} & b_{54} & b_{55} & b_{56} \\ b_{61} & b_{62} & b_{63} & b_{64} & b_{65} & b_{66} \end{bmatrix} \quad (3.4.13)$$

Using a symbolic mathematical tool e.g. Matlab or Mathcad the coefficients from (3.4.13) can be determined easily. First, some coefficients are calculated as:

$$K_1 = \frac{L_s L_r}{M_{sr}^2} \quad K_2 = \frac{K_1 - \frac{3}{4}}{K_1 - \frac{9}{4}} \quad K_3 = \frac{-\frac{3}{4}}{K_1 - \frac{9}{4}} \quad K_4 = \frac{-\frac{1}{M_{sr}^2}}{K_1 - \frac{9}{4}} \quad (3.4.14)$$

and then all other elements of the matrix B:

$$\begin{aligned}
 b_{11} &= b_{22} = b_{33} = \frac{K_2}{L_s} \\
 b_{44} &= b_{55} = b_{66} = \frac{K_2}{L_r} \\
 b_{12} &= b_{13} = b_{21} = b_{23} = b_{31} = b_{32} = \frac{K_3}{L_s} \\
 b_{45} &= b_{46} = b_{54} = b_{56} = b_{64} = b_{65} = \frac{K_3}{L_r} \\
 b_{14} &= b_{25} = b_{36} = b_{41} = b_{52} = b_{63} = K_4 f_1 \\
 b_{15} &= b_{26} = b_{34} = b_{43} = b_{51} = b_{62} = K_4 f_2 \\
 b_{16} &= b_{24} = b_{35} = b_{42} = b_{53} = b_{61} = K_4 f_3
 \end{aligned} \tag{3.4.15}$$

The electromagnetic torque can be obtained starting from the voltage equations using currents as state variables

$$[V] = [R][I] + \frac{d}{dt}([L][I]) = [R][I] + \frac{d[L]}{dt}[I] + [L]\frac{d[I]}{dt} \tag{3.4.16}$$

and multiplying it with the transpose of the currents matrix:

$$[I]^t [V] = [I]^t [R][I] + [I]^t \left(\frac{d}{dt}[\Psi] \right) = [I]^t [R][I] + [I]^t \frac{d[L]}{dt}[I] + [I]^t [L] \frac{d[I]}{dt} \tag{3.4.17}$$

Alternatively using $\frac{d[L]}{dt} = \frac{d[L]}{d\theta_r} \cdot \frac{d\theta_r}{dt} = \frac{d[L]}{d\theta_r} \cdot \omega_r$, where ω_r is the electrical speed of the rotor results in:

$$[I]^t [V] = [I]^t [R][I] + \omega_r [I]^t \frac{d[L]}{d\theta_r}[I] + [I]^t [L] \frac{d[I]}{dt} \tag{3.4.18}$$

The terms from (3.4.18) have the following meaning:

- $P_i = [I]^t [V]$ - instantaneous power;
- $P_{\text{copper}} = [I]^t [R][I]$ - copper losses in the machine windings;
- $P_{\text{mag}} = [I]^t [L] \frac{d[I]}{dt}$ - magnetic power stored in the machine (due to the variation in time of the magnetic energy);
- $P_m = \omega_r [I]^t \frac{d[L]}{d\theta_r}[I]$ - mechanical power.

The electromagnetic torque is then:

$$T_e = \frac{P_m}{\Omega_r} = p \frac{P_m}{\omega_r} = p [I]^t \frac{d[L]}{d\theta_r}[I] \tag{3.4.19}$$

where: p is the number of pole pairs and Ω_r is the mechanical speed of the rotor.

Substituting (3.4.11) in (3.4.19) the electromagnetic torque as a function of currents is:

$$T_e = -pM_{sr} \{ (i_A i_a + i_B i_b + i_C i_c) \sin \theta + (i_A i_b + i_B i_c + i_C i_a) \sin(\theta + 2\pi/3) + (i_A i_c + i_B i_a + i_C i_b) \sin(\theta - 2\pi/3) \} \quad (3.4.20)$$

Based on (3.4.5) and (3.4.20) the dynamic model of the induction machine has been implemented in Saber using the MAST language (see Appendix A). The symbol and the mask interface for this model are shown in Fig. 3.10.

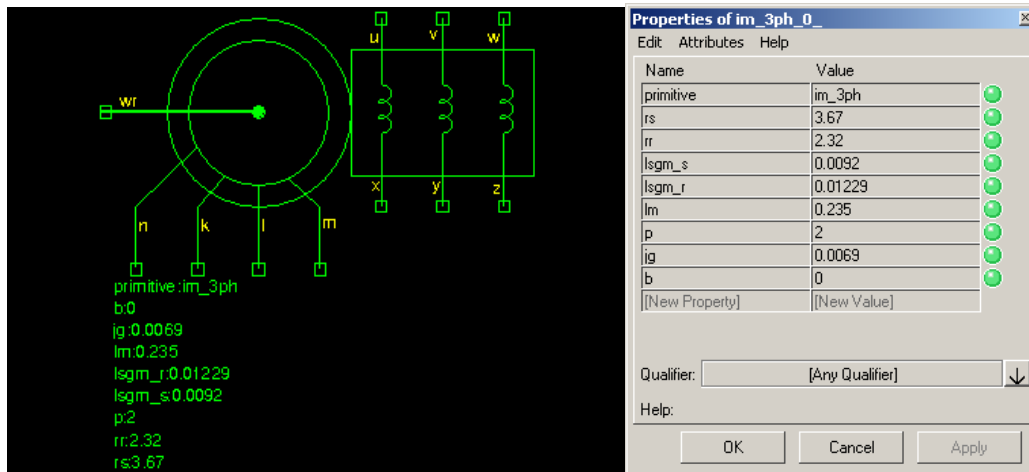


Fig. 3.10. Symbol and mask interface for the abc/abc model of the induction machine.

The input parameters in this interface are based on the standard data sheet: resistance and leakage inductance for the stator and rotor windings, magnetizing inductance, number of pole pair and initial conditions for the state-variables (currents and electrical angle of the rotor).

Using this model all the soft-starter fed induction machine types can be analyzed and simulated. Moreover, this model can be used for a doubly fed generator.

3.5 Power converters

3.5.1 AC Controllers (Soft-Starter)

At present many wind turbines, up to 2.3 MW, are based on the “Danish concept” in which a squirrel-cage induction generator is directly connected to the grid as shown in Fig. 3.11.

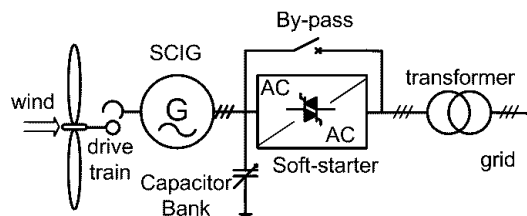


Fig. 3.11. Block diagram of directly grid-connected wind turbine.

The scheme comprises the wind turbine rotor, linked via a gearbox to a generator, which through an electrical interface is connected to the grid. A control system is necessary to assure a proper operation of the wind turbine under all conditions. The electrical interface consists of a soft-starter, a capacitor bank and a transformer, which makes the connection with the

medium voltage grid. The capacitor bank is used to control the power factor of the generator output.

Soft-starters are used only during the start-up sequence of the generator in order to limit the inrush currents and the starting torque transients in the drive train.

There are many configurations of soft-starters, which fed an induction machine, although of interest in wind turbine applications are those presented in Fig. 3.12.

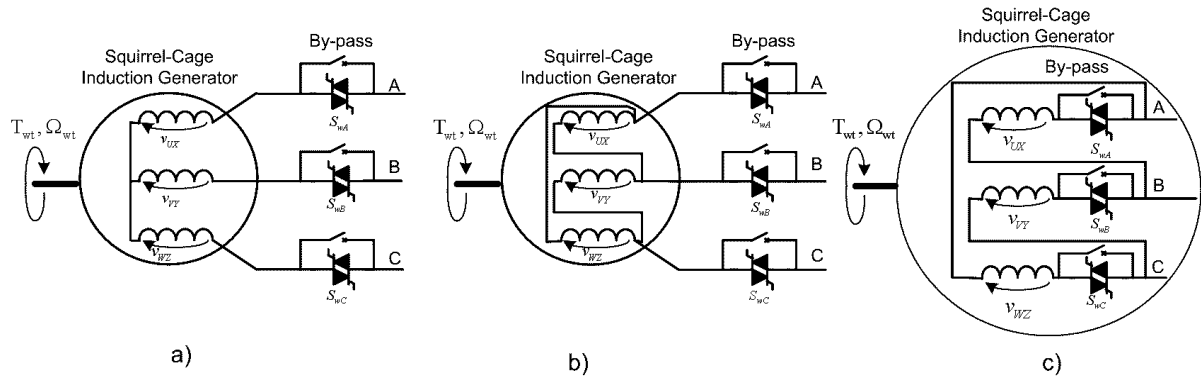


Fig. 3.12. Possible configurations of soft-starter-fed induction machine in wind turbine applications:
a) star connection, b) delta connection and c) branch-delta connection.

The star and delta configurations have basically the same layout for the semiconductors (SCRs), the difference consists in the machine winding connection. There are two anti-parallel thyristors on each phase, each one conducting on the positive cycle of the applied voltage. For a star connection the applied per phase voltages depend on the on-state of the SCRs on each phase. So, the soft-starter can operate only when two or three SCRs are conducting in the same moment. The similar considerations can be done for a delta connection.

However a soft-starter with a branch delta connection, will operate only with one SCR conducting at a given moment.

In wind turbine applications mainly the delta connection for the induction machine are used because the current rating of the stator windings can be reduced, and the third harmonic in the line currents is eliminated in this case.

Depending on the firing angle α , three different modes of operation of the soft-starter can be distinguished when a star or delta connected resistive load is used [16], [17]:

- Mode 1 - $0 \leq \alpha < 60^\circ$ - two or three SCRs are conducting (in either direction);
- Mode 2 - $60^\circ \leq \alpha < 90^\circ$ - two SCRs are conducting;
- Mode 3 - $90^\circ \leq \alpha < 150^\circ$ - none or two SCRs are conducting.

where α is the firing angle for the soft-starter.

When a resistive-inductive load is used the analysis of the controller is difficult, since the operation modes depend on the extinction angle ξ and the limit angle α_{lim} , both dependent on the phase angle φ .

Mode 2 of operation, characterized by rapid changes of the output current, is not possible due to the load inductance. The ranges of the two remaining operation modes are:

- Mode 1: $\varphi \leq \alpha < \alpha_{lim}$ - two or three SCRs are conducting;
- Mode 3: $\alpha_{lim} \leq \alpha < 150^\circ$ - none or two SCRs are conducting.

The Saber model of the soft-starter consists in three separate branches with two anti-parallel SCR's. The SCR's command is included in this model. An RC snubber for each SCR pair is provided. The model uses the ideal models for SCR from Saber libraries. However, a real component can be inserted in the model. The general structure for this model is shown in Fig. 3.13.

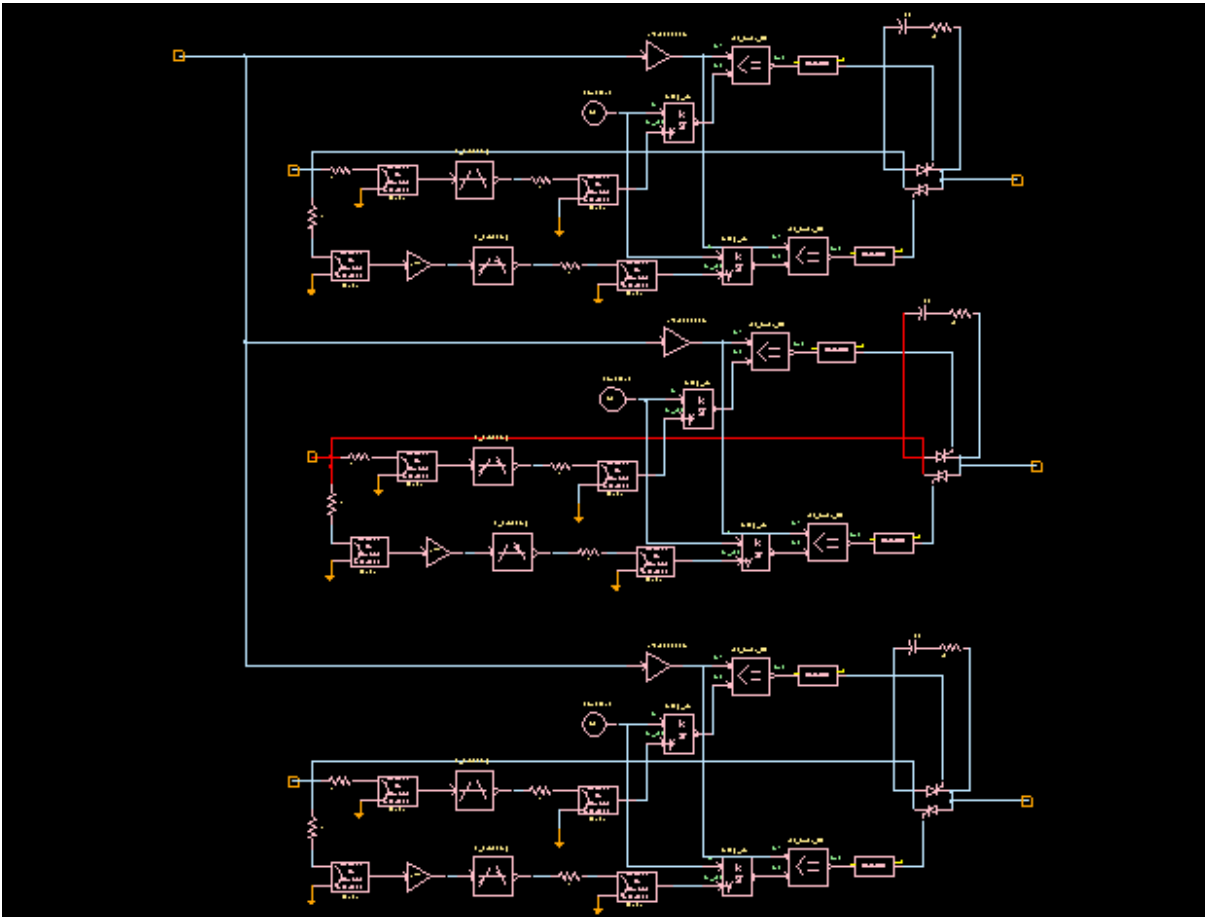


Fig. 3.13. Saber model for soft-starter including the SCR's command.

The symbol for the 3-phase model of the soft-starter is shown in Fig. 3.14.

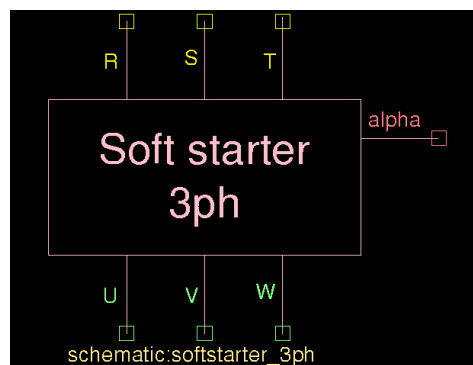


Fig. 3.14. Saber symbol for the three-phase soft-starter.

The control signal for this block is the firing angle α for the SCR's, which is given from a separate block.

Using this model in conjunction with the *abc/abc* model of the induction machine the basic topologies presented in Fig. 3.12 can easily be obtained.

3.5.2 Back to back power converter

Currently, in variable speed wind turbine applications mainly the back-to-back voltage source converter (VSC) is used as shown in Fig. 3.15.

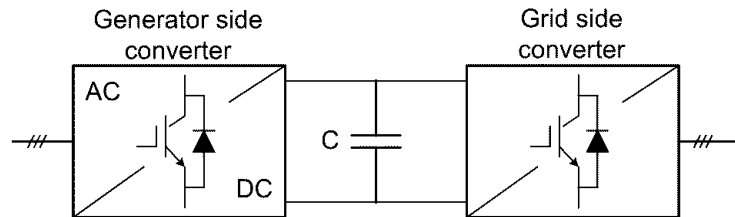


Fig. 3.15. Structure of the back-to-back voltage source converter

This topology comprises a double conversion from AC to DC and then from DC to AC. Both converters can operate in rectifier or inverter mode and therefore a bi-directional power flow can be achieved.

A voltage source converter can be implemented in several ways: six-step, pulse amplitude modulated (PAM) or pulse width modulated (PWM). Moreover, the implementation of a PWM VSC may be realized by three methods: harmonic elimination, "sinusoidal" PWM or space vector strategy (SV-PWM).

The Saber model for this power converter consists of two voltage source converters connected via a DC-link circuit as shown in Fig. 3.16.

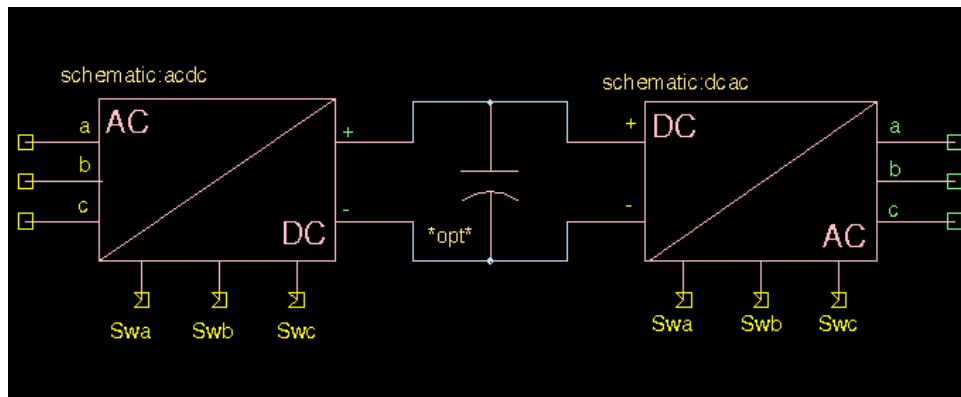


Fig. 3.16. Saber model for a back-to-back voltage source converter.

The model uses the ideal model for IGBT from Saber libraries. However, real IGBT's models can be used in it.

3.6 Modulation strategies for power converters

3.6.1 PWM sinusoidal modulator using sine-triangle with third harmonic (STTH)

The normalized duty cycle functions for a Sine-Triangle with Third Harmonic (STTH) modulation algorithm are:

$$\begin{aligned}
 \text{duty_a} &= 0.5 \cdot m \cdot \frac{2}{\sqrt{3}} \cdot \left[\sin(\theta) + \frac{\sin(3\theta)}{6} \right] + 0.5 \\
 \text{duty_b} &= 0.5 \cdot m \cdot \frac{2}{\sqrt{3}} \cdot \left[\sin\left(\theta - \frac{2\pi}{3}\right) + \frac{\sin(3\theta)}{6} \right] + 0.5 \\
 \text{duty_c} &= 0.5 \cdot m \cdot \frac{2}{\sqrt{3}} \cdot \left[\sin\left(\theta - \frac{4\pi}{3}\right) + \frac{\sin(3\theta)}{6} \right] + 0.5
 \end{aligned} \tag{3.6.1}$$

where

- θ - the reference angle is calculated based on the desired output frequency;
- m - is the modulation index $m = \sqrt{3} \frac{V_{\text{ref}}}{V_{\text{dc}}}$

Equations (3.6.1) are implemented in Saber using the MAST language. The conversion from duty cycles to switching signals for IGBT's as well as the pulse dropping option and minimum pulse width one are included in the model.

The Saber symbol for this modulation strategy is shown in Fig. 3.17.

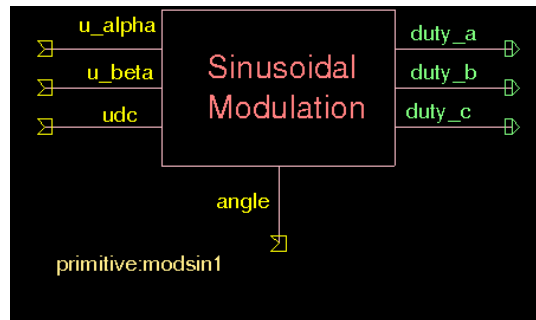


Fig. 3.17. Saber model for STTH modulator.

3.6.2 Space Vector Modulation

The Space Vector Modulation (SVM) strategy simultaneously performs the waveform generation of all three phases within a two-dimensional vectorial reference frame, eliminating the computational redundancy of considering each phase separately [18]. SVM is based on the concept of approximating the reference voltage vector defined as:

$$\mathbf{V}_{\text{ref}} = \frac{2}{3} \cdot \left(v_a + v_b \cdot e^{-j\frac{2\pi}{3}} + v_c \cdot e^{-j\frac{4\pi}{3}} \right) \tag{3.6.2}$$

by one switching-cycle time-averaged combination of two adjacent physically realisable voltage vectors in a three-phase voltage source converter.

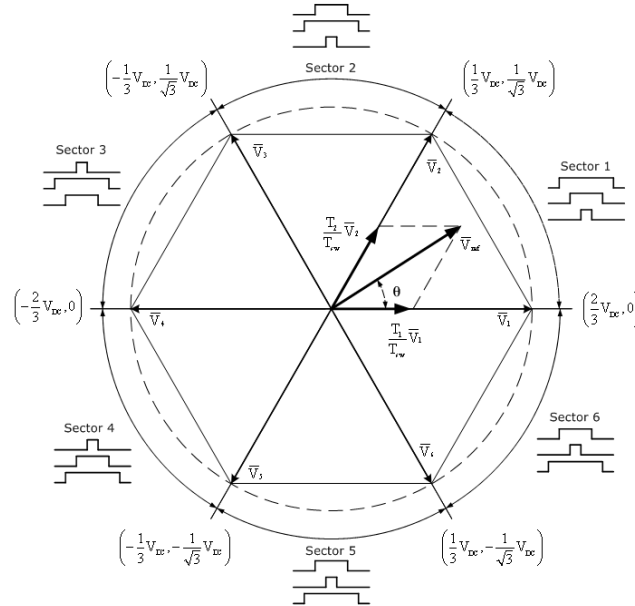


Fig. 3.18. Voltage vectors and space vectors.

The two adjacent voltage vectors are chosen from $V_1 - V_6$ according to the six active switching states ([011], [001], [101], [100], [110], [010]), plus two null voltage vectors, i.e. V_0 and V_7 , corresponding to the passive switching states ([000] and [111]). Switching state [111] means all three upper transistors are switched on.

For high switching frequencies, the reference voltage vector can be considered constant during each switching period. The on-duration of active vectors V_1 and V_2 that are adjacent to V_{ref} are found by solving the vectorial equation (Fig. 3.18):

$$V_{ref} \cdot \frac{T_{sw}}{2} = V_1 \cdot T_1 + V_2 \cdot T_2 \tag{3.6.3}$$

$$T_0 = \frac{T_{sw}}{2} - T_1 - T_2$$

where T_0 denoting the time period left from a half switching period is used by the null voltage vectors using a pattern that ensure that only one leg is switched at every state transition.

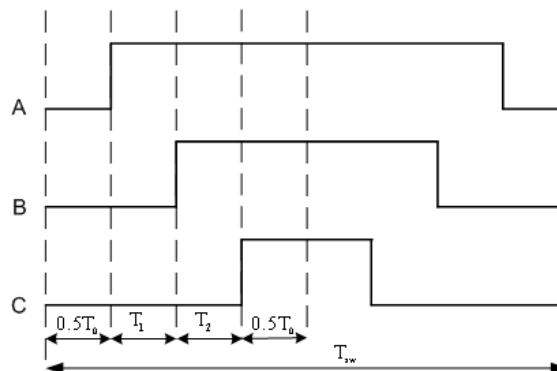


Fig. 3.19. Switching pattern for the first sector.

Assuming V_{ref} is in the first sector, (3.6.3) can be solved in orthogonal d-q axis yielding:

$$\begin{cases} T_1 = m \cdot \frac{\sqrt{3}}{2} \cdot \frac{T_{sw}}{2} \cdot \left(\cos(\gamma) - \frac{1}{\sqrt{3}} \sin(\gamma) \right) \\ T_2 = m \cdot \frac{T_{sw}}{2} \cdot \sin(\gamma) \end{cases} \quad (3.6.4)$$

$$m = \frac{V_{ref} \cdot \sqrt{3}}{V_{dc}}$$

where an equivalent modulation index m was introduced and V_{ref} denotes the amplitude of the reference voltage.

Therefore, SVM provides accurate control of voltage amplitude, frequency and phase within every switching period being very suitable for field-oriented control.

Equations (3.6.3) are implemented in Saber using the MAST language. Based on (3.6.4) the duty cycles for each phase are calculated for each sector (Fig. 3.18). Then, the pulse dropping option is included in the model as well as the transformation to switching states and the dead time option.

The Saber symbol for SVM Strategy is shown in Fig. 3.20.

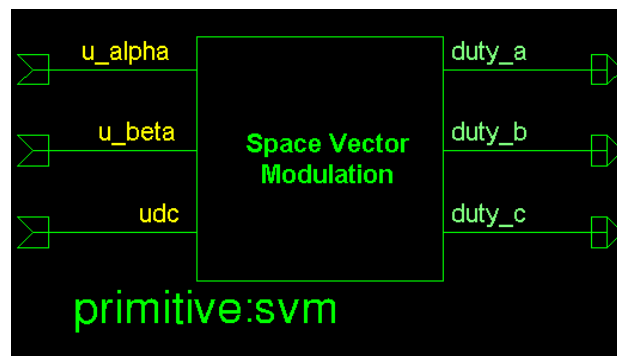


Fig. 3.20. Saber model for SVM strategy.

3.7 Summary

In this chapter the mathematical models as well as the Saber implementation for the new developed models are given. The focus is on the components, which describes the new models and not in the implementation of the constitutive blocks from these models.

Chapter 4

Description of Saber's built-in libraries

In this chapter a description of the built-in libraries from Saber is presented. Since almost all blocks from these libraries do not have a help file or documentation available and only the MAST source is accessible, few remarks for each block will be presented. Moreover, only the header is visible in the MAST files, the rest of the file being encrypted [5].

4.1 Mechanisms library

This library contains models for different mechanic units: gearboxes, pulley, spring, windlass and clutch, etc. as shown in Fig. 4.1.

In a wind turbine system are of great interest the gearbox models, which take into account the efficiency of the gearbox, and power rating parameter, which is used in the stress analysis. One of these gearbox models has been used in the drive-train model of the wind turbine.

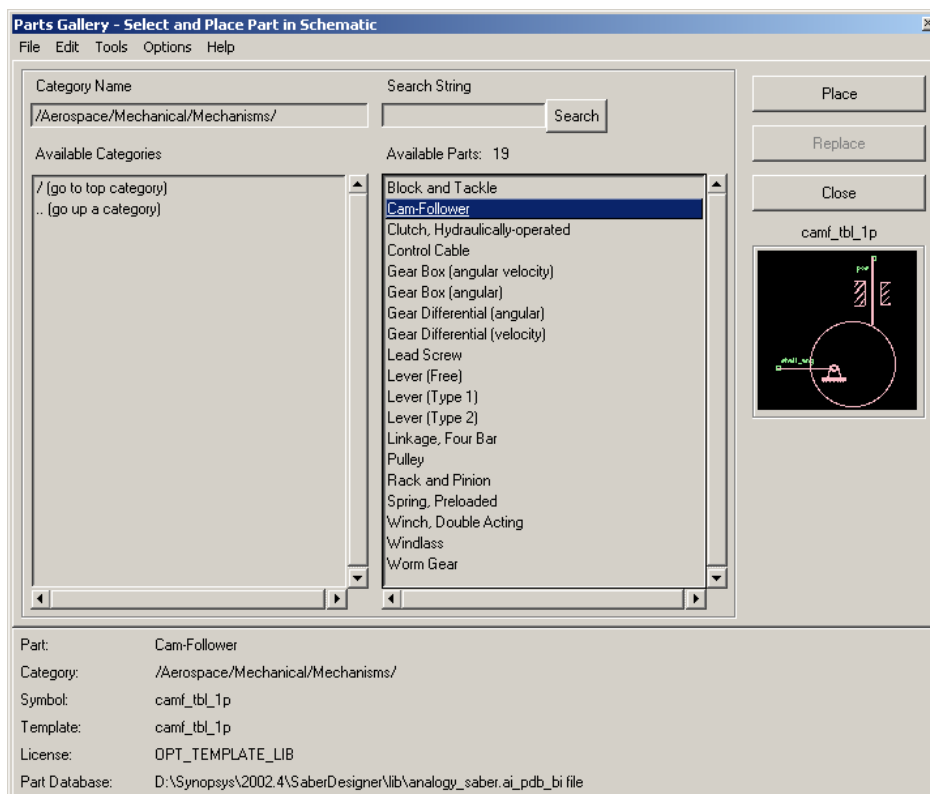


Fig. 4.1. Content of Mechanisms library.

4.2 Generators library

This library contains models for all types of DC generators and a dq model for synchronous generator, which can be used both in motor or generator mode as shown in Fig. 4.2.

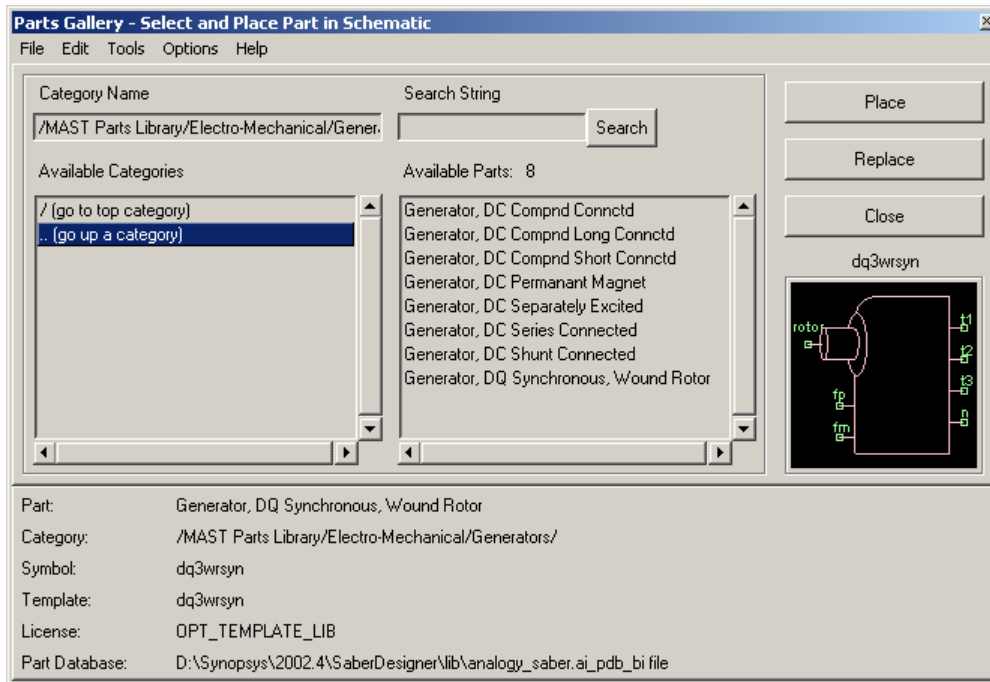


Fig. 4.2. Content of Generators library.

The model for the synchronous generators has constant parameters and no damping cage. The only parameters for this model are (Fig. 4.3): stator resistance, moment of inertia, viscous damping coefficient, number of pole pairs and some initial conditions for the currents.

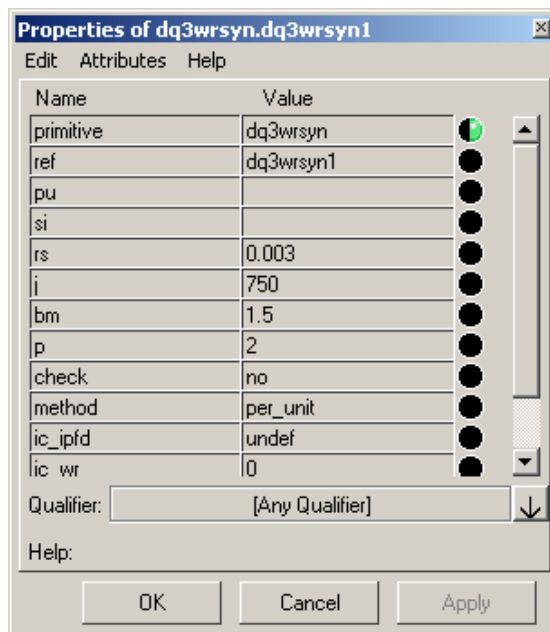


Fig. 4.3. Mask interface for synchronous machine model.

All reactances and time constants for this model are fixed: stator leakage reactance, zero-sequence reactance, d-axis synchronous reactance, q-axis synchronous reactance, transient d-axis synchronous reactance, transient q-axis synchronous reactance, subtransient d-axis synchronous reactance, subtransient q-axis synchronous reactance, d-axis transient short

circuit time constant, q-axis transient short circuit time constant, d-axis subtransient short circuit time constant, q-axis subtransient short circuit time constant.

The references for this model are [19], [20].

4.3 Motors library

This library contains models for single-phase motors, 3-phase squirrel-cage and wound rotor induction machines, permanent magnet synchronous machine, all types of DC motors as shown in Fig. 4.4. All these models are written in MAST language.

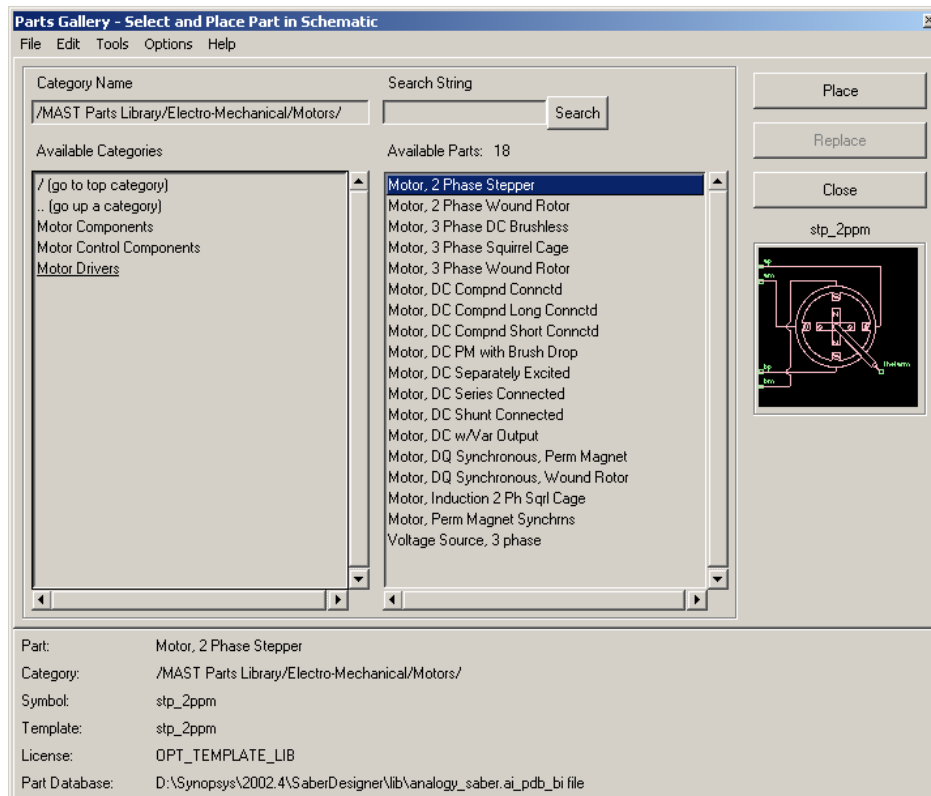


Fig. 4.4. Content of Motors library.

The model for squirrel cage induction motor is written in MAST language and implements a wye-connected, symmetrical machine in general dqo reference frame with constant parameters. The saturation, deep-bar effect and iron losses are not taken into account. The mask interface for this model is shown in Fig. 4.5a.

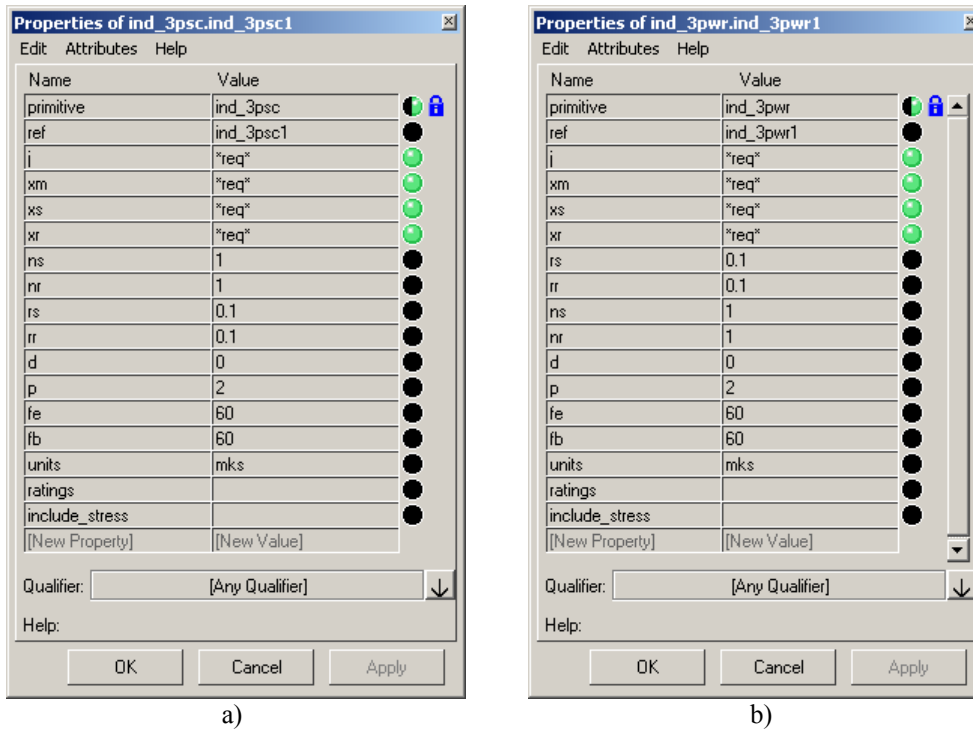


Fig. 4.5. Mask interfaces for induction machine models:
a) squirrel-cage machine and b) wound rotor machine.

The model for wound rotor induction machine is written in MAST language and implements a symmetrical machine in stationary dq reference frame with constant parameters. The saturation, deep-bar effect and iron losses are not taken into account. The mask interface for this model is shown in Fig. 4.5b.

The reference [20] is given in the documentation for both models.

4.4 Transformers library

This library contains various models for transformers e.g. single-phase, single-phase 2 windings, 3-phase 2 windings (wye-wye, delta-wye, etc), nonlinear transformers, etc. as shown in Fig. 4.6.

The models for 3-phase transformers have constant parameters. In order to include saturation and iron losses for a 3-phase transformer 3 separate models for each limb should be used. This model is called “Non-Linear two-winding transformer - xfrnl2”. Moreover, some geometrical dimensions e.g. length of magnetic path and cross-section areas are needed. Some consideration should be done for a 3-phase 3-windings transformer. In this case the model called “Non-linear three winding transformer - xfrnl3” is used.

For each model a material library is provided, which contains characterization data for some magnetic materials (Ferrocube materials).

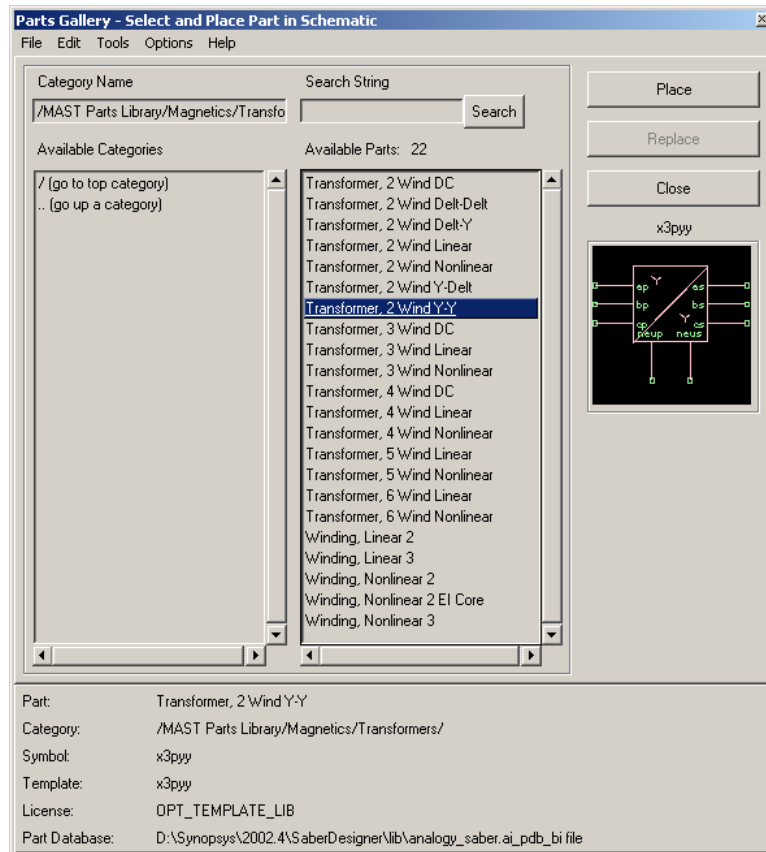


Fig. 4.6. Content of Transformers library.

4.5 Control System library

This library contains blocks, which can be used for the control system design. The library is structured in six modules namely: Control System Sources, Functions, Integrators, Non-linear, Relational operators and Signal combines as shown in Fig. 4.7.

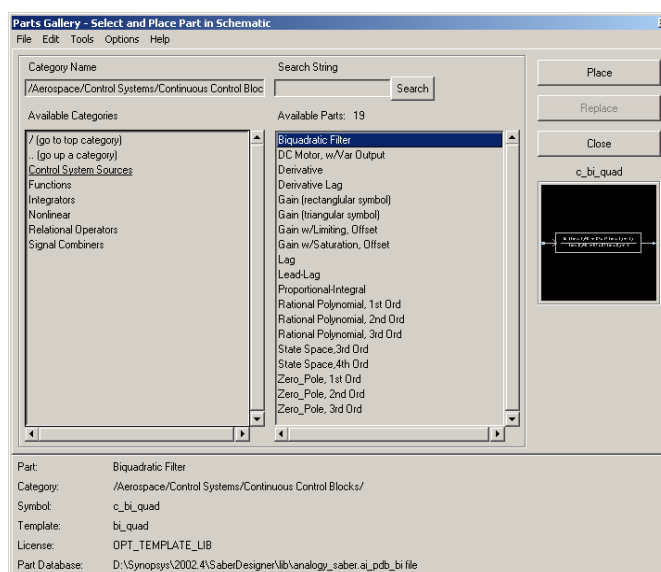


Fig. 4.7. Content of Control System library.

Using the blocks from this library a control system can easily be built like in Matlab/Simulink. However, when these blocks are used in a hierarchical subsystem it is impossible to provide parameters from the main mask to these blocks. All blocks accept only numerical values into their mask interface. This was the case with the wind model, where the transfer function block has been used for the Kaimal filter. Since the mask of the wind model block provides the parameters for the Kaimal filter block, it was impossible to use the transfer function block from this library. A new transfer function block has been built based on the MAST language.

4.6 Cables & Wires library

Saber has a wide range of models for cables and wires as shown in Fig. 4.8. There are models for different types of cables 2, 3 or 4 conductors, transmission line with one conductor, etc.

The shielded 4-conductor cable model is based on the gamma - equivalent circuit as shown in Fig. 4.9.

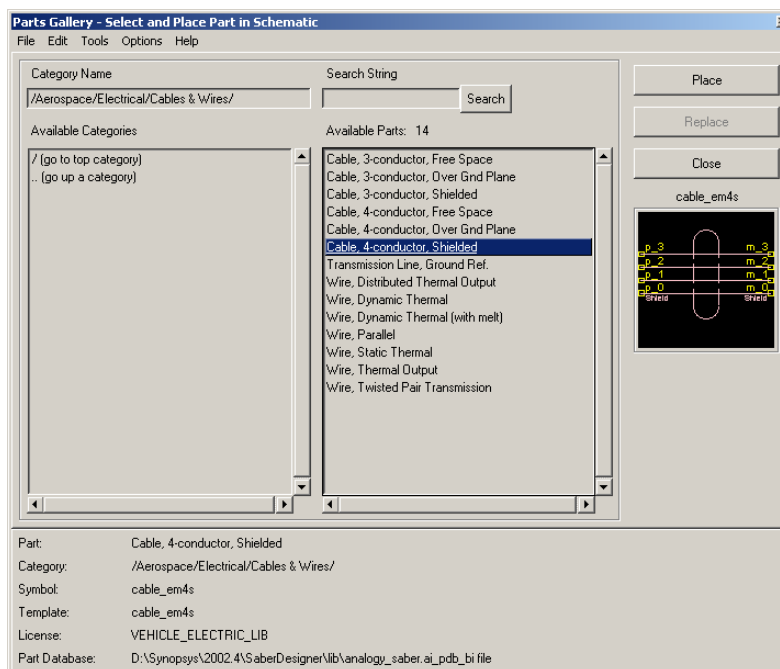


Fig. 4.8. Content of Cables & Wires library in Saber.

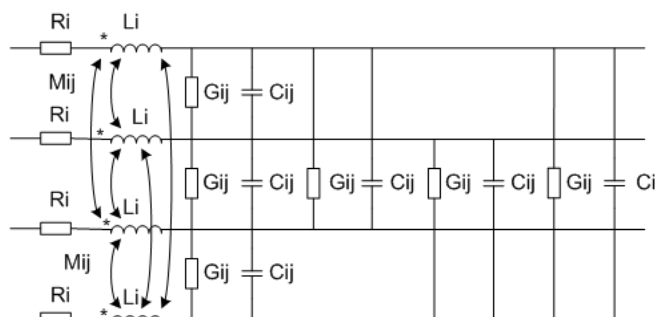


Fig. 4.9. Equivalent circuit for shielded 4-conductor cable model used in Saber.

The cable parameters are derived from conductor geometry or entered values by the user using the mask interface (Fig. 4.10). Mutual inductances are either calculated or specified by

the user (Fig. 4.10). No modelling of skin effects is offered for this model. No model for a 3-phase transmission line is provided in this library.

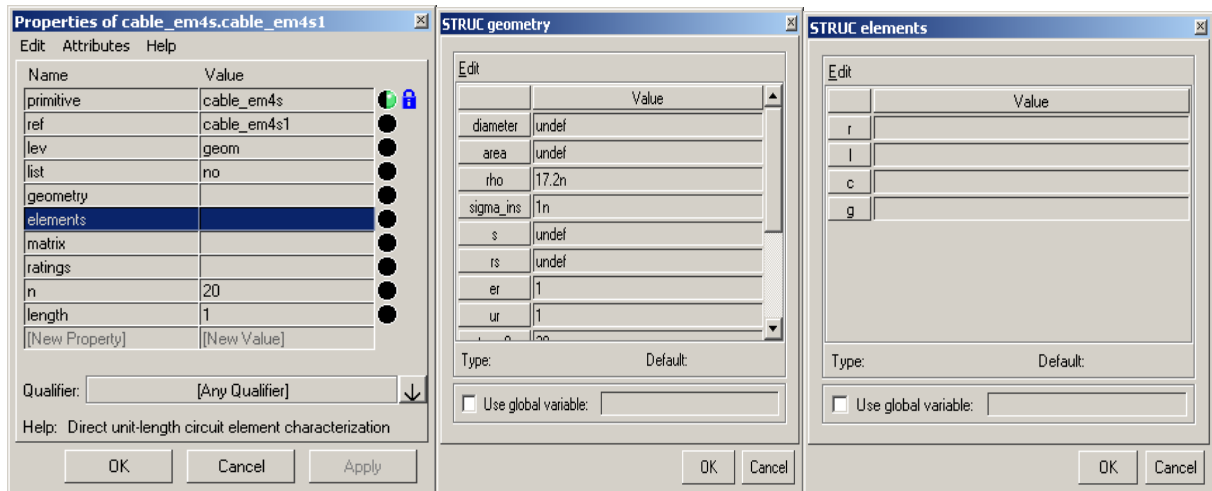


Fig. 4.10. Interfaces for cable model in Saber.

4.7 Summary

In this chapter an overview of the existing models from Saber libraries is presented. Since Saber has a large library with built-in models the focus is in presenting the relevant components, which can be used in modelling of a wind turbine system. Where is possible the relevant references for these models are indicated.

Chapter 5

Simulation results

Start-up sequence of a soft-starter-fed induction machine

The start-up sequence of a 2.2 kW soft-starter-fed squirrel-cage induction machine with a delta connection for the stator windings has been investigated using the developed models from Saber. The behaviour of the considered system in this case is similar with a 2 MW machine.

The simulation diagram consists in a voltage source, a soft-starter, a by-pass switch (used in normal operation) and an *abc/abc* model for induction machine as shown in Fig. 5.1.

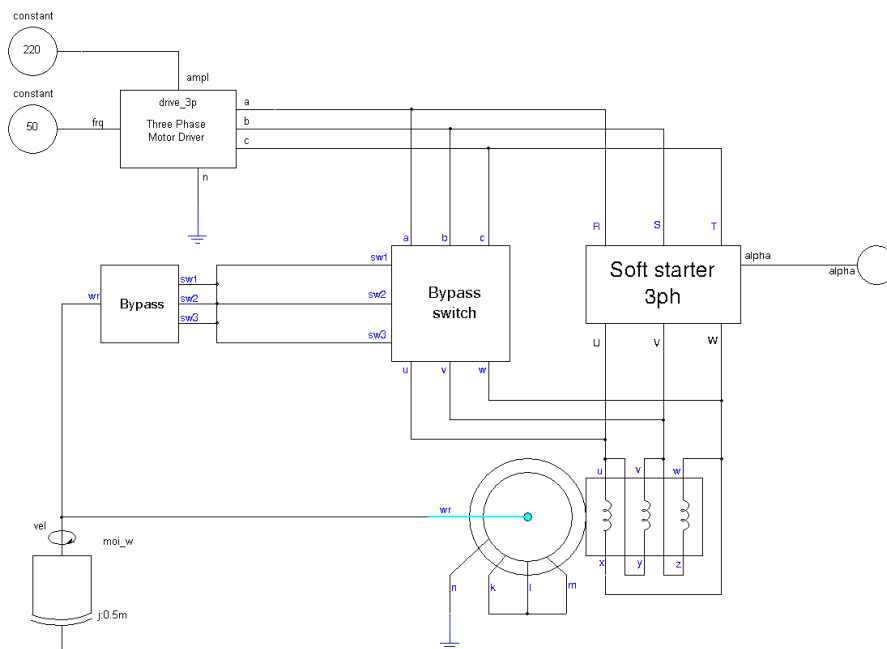


Fig. 5.1. Saber simulation diagram for a soft-starter-fed induction machine with delta connection for the stator windings.

The simulation results in terms of envelope of the stator current, electromagnetic torque and shaft speed are shown in Fig. 5.2.

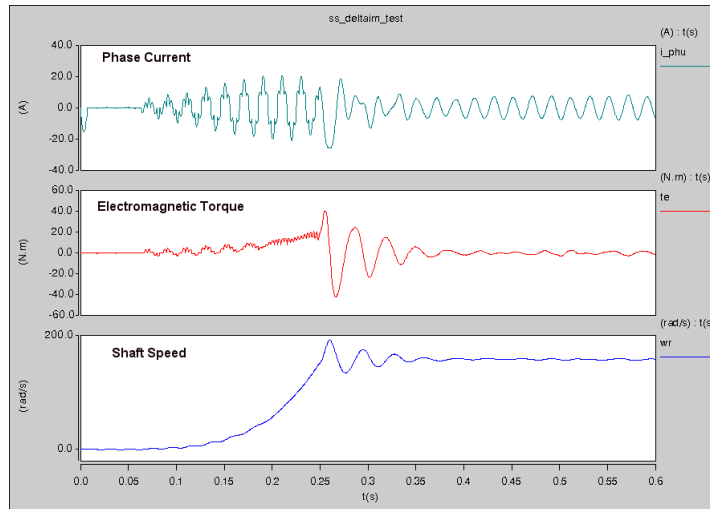


Fig. 5.2. Start-up sequence for a soft-starter fed induction machine in Saber.

A soft-starter fed induction machine can operate only in two modes of operation. Each of these modes is characterized by some special patterns for the voltages and currents [16], [17]. The voltage and line current waveforms for these operation modes are shown in Fig. 5.3.

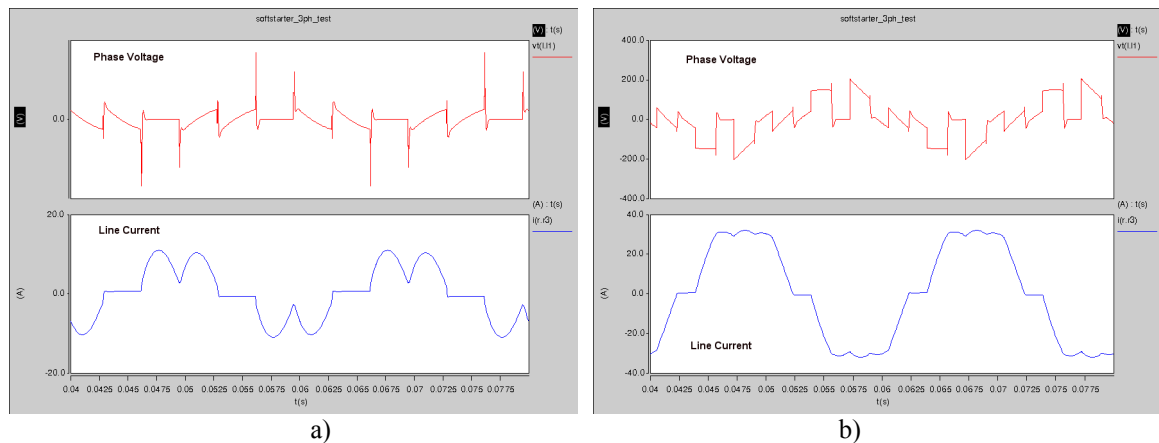


Fig. 5.3. Possible mode of operation of a soft-starter fed induction machine:
a) Mode 3 and b) Mode 1.

It can be noticed that these waveforms are similar with those obtained in Matlab/Simulink [21]. Future work should be done in order to simulate a 2 MW wind turbine in these conditions.

However, since Saber is a dedicated electrical circuit simulation tool, while Matlab/Simulink is based on the mathematical description of the model, a good “validation” for the mathematical model of the soft-starter is achieved.

Conclusions and Future work

Saber is a very powerful tool, which can be used to simulate and analyze physical effects in a wind turbine system. Saber supports a variety of analysis, which are not available in other simulation tools. Unfortunately, it has no focus until now in studying wind turbine systems and adequate models should be developed. Developing new models in Saber is not a simple task since there is a lack in documentation and there are no examples for systems which involves a wide range of variables e.g. mechanical, electrical, magnetical, etc. The Saber documentation shows only examples for electric and electronic components such resistors, capacitors, etc. Moreover is not so user friendly as Matlab/Simulink.

In order to analyze the main concepts used in wind turbine systems the following models should be improved or developed in the future:

- A two mass model for wind turbine drive train as in Matlab/Simulink (same equations and parameters);
- Including deep-bar effect and saturation in the *abc/abc* model for induction machine;
- A dq-model for synchronous machine based on the dynamic equations implemented in Matlab/Simulink. The existing built-in model requires some particular parameters, which are not available from standard data-sheets. The saturation should be included in this model. A long term goal is to develop an *abc/abc* model for this machine;
- A three-phase three-winding transformer model based on standard data-sheets. Usually the geometrical dimensions of the transformer core are not available, therefore the built-in Saber model is difficult to use it.
- Active and reactive power control for a wind turbine based doubly-fed induction generator.
- Pitch control for wind turbine – active stall and variable pitch;
- Capacitor bank control for a fixed-speed wind turbine;
- Supervisory control for wind turbine.

References

- [1]. Synopsis Inc. - Getting Started with SaberDesigner, Release 2001.4;
- [2]. Synopsis Inc. – SaberGuide. User's and Reference Manual, Release 2002.2;
- [3]. Synopsis Inc. – SaberSketch. User's and Reference Manual, Release 2002.2;
- [4]. Synopsis Inc. – Model Architect. User's and Reference Manual, Release 2002.2;
- [5]. Synopsis Inc. - MAST Language Reference Manual, Release 2002.2;
- [6]. Synopsis Inc. - Guide to Writing MAST Templates, Book 1 and Book 2, Release 2001.4;
- [7]. Synopsis Inc. - SaberScope/Cosmos-Scope. User's and Reference Manual, Release 2002.2;
- [8]. Synopsis Inc. - SaberLink User's and Reference Manual, Release 2002.2.
- [9]. Synopsis Inc. - Managing Symbols and Models, Release 2002.4;
- [10]. Langreder W. (1996) *Models for variable speed wind turbines* CREST, Dept. of Electrical Engineering, Loughborough University, UK and Risø National Laboratory, Denmark;
- [11]. Marsaglia, G., Tsang, W.W. – *The ziggurat method for generating random variables*, Journal of Statistical Software, vol. 5, no. 8, 2000, 7 pp.
- [12]. Boldea, I. - *Electrical Machines and Transformers*, Ed. Didactica si Pedagogica, Bucharest 1994, ISBN 973-30-2341-8 (in romanian);
- [13]. Goldemberg, C., De Arruda Penteado, A. – *Improvements on the inductance matrix inversion simplifying the use of the ABC/abc induction machine model*, Proceed. Of IEMD '99, pp. 422-424;
- [14]. Gorman, M.J., Grainger, J.J. – *Transformer modeling for Distribution System Studies. Part I: Linear Modeling Basics*, IEEE Trans. on Power Delivery, Vol.7, No. 2, April 1992, pp. 567-574;
- [15]. Pillay, P., Levin, V. – *Mathematical models for induction machines*, Industry Applications Conference, 1995. Thirtieth IAS Annual Meeting, IAS '95, Conference Record of the 1995 IEEE, Vol. 1 , 8-12 Oct. 1995 pp. 606 -616;
- [16]. Sheperd, W., Huley, L.N., Liang, D.T.W. – *Power Electronics and Motor Control*, Cambridge University Press 1995, ISBN 0-521-47241-05;
- [17]. Trzynadlowski, A.M. – *Introduction to Modern Power Electronics*, J. Wiley & Sons, 1998, ISBN 0-471-15303-6;
- [18]. Mohan, N., Undeland, T.M., Robbins, W.P. – *Power Electronics: Converters, Applications and Design*, J. Wiley & Sons, 1995, ISBN 0-471-58408-8;
- [19]. Ong, Chee-Mun – *Dynamic Simulation of Electric Machinery: Using Matlab/Simulink*, Prentice Hall PTR, 1997, ISBN 0137237855;
- [20]. Krause, P.C., Wasynczuk, O., Sudhoff, S.D. – *Analysis of Electric Machinery*, IEEE Press 1995, ISBN 0-7803-1101-9;
- [21]. Iov, F., Hansen, A.D., Sørensen, P., Blaabjerg, F. – *Wind Turbine Blockset in Matlab/Simulink. General overview and description of the Models*, Aalborg University, December 2003, ISBN 87-89179-46-3;

Appendix A

MAST Template for developed ABC/abc Model of Induction Machine

```
#####
ABC/abc model for Induction Machine #
#This model can be used both for a squirrel-cage machine and wound rotor machine #
#####

element template im_3ph u v w x y z k l m n wr = Rs, Rr, Lsgm_s, Lsgm_r, Lm, p, Jg, B

electrical u, v, w, x, y, z # Definition of connection pins # here for the stator
electrical k, l, m # rotor connection pins
electrical n # Rotor nul connection pin
rotational_vel wr # Rotor mechanical speed

#number Rs = 0.001164, # Stator resistance
# Rr = 0.001309, # Rotor resistance
# Lsgm_s = 70u, # Stator inductance
# Lsgm_r = 75.44u, # Rotor inductance
# Lm = 2.99m, # Mutual inductance
# p = 2, # Number of poles pairs
# Jg = 60, # Moment of inertia
# B = 0 # Friction factor

number Rs = 3.67,
Rr = 2.32,
Lsgm_s = 0.0092,
Lsgm_r = 0.01229,
Lm = 0.235,
p = 2,
Jg = 0.0069,
B = 0

{
# Start body template

val nu Msr
val nu Ls
val nu Lr
val nu K1, K2, K3, K4
val nu a11, a12, a13, a14, a15, a16, a17, a18
val nu a31, a32, a33, a34, a35, a36, a37, a38
val tq_Nm Tload

var i i_phu, i_phv, i_phw # Stator phase currents defined as variables
var i i_phk, i_phl, i_phm # Rotor phase currents defined as variables
var rad theta # Rotor position (electrical)
```

```

val w_radps omega_e           # Rotor speed (electrical)
val w_radps omega_m         # Rotor speed (mechanical)
val w_radps wm

val nu uu, uv, uw
val nu uk, ul, um

val nu f1, f2, f3
val nu g1, g2, g3

val tq_Nm Te                # Electromagnetic torque

number c1 = 0.6666666666666667 # Constants definition (2/3)
number c2 = 1.5                # (3/2)
number c3 = 0.75              # (3/4)
number c4 = 2.25              # (9/4)
number c5 = 1.2990           # (3*sqrt(3))/4
number c6 = 2.09439510239319549 # (2*pi/3)

values {

    omega_m = w_radps(wr)
    omega_e = p* omega_m

    msr = c1 * Lm                # mutual inductance
    Ls = Lsgm_s + c2 * Msr      # stator inductance
    Lr = Lsgm_r + c2 * Msr      # rotor inductance raported to the stator

    K1 = (Ls * Lr) / (Msr*Msr) # Definition of some coefficients
    K2 = (K1 - c3) / (K1 - c4)
    K3 = (-c3) / (K1 - c4)
    K4 = -1 / (Msr * (K1 - c4))

    a11 = (K2 * Rs) / Ls
    a12 = (K3 * Rs) / Ls
    a13 = c5 * K4 * Msr
    a14 = K4 * Rr
    a15 = (Msr *(K2 - K3))/Ls
    a16 = K2 / Ls
    a17 = K3 / Ls
    a18 = K4

    a31 = K4 * Rs
    a32 = (Msr *(K2 -K3))/Lr
    a33 = (K2 * Rr) / Lr
    a34 = (K3 * Rr) / Lr
    a35 = c5 * Msr * K4
    a36 = K4
    a37 = K2 / Lr
    a38 = K3 / Lr

    uu = v(u) - v(x)
    uv = v(v) - v(y)

```



```

uw = v(w) - v(z)

uk = v(k) - v(n)
ul = v(l) - v(n)
um = v(m) - v(n)

f1 = cos(theta)
f2 = cos(theta + c6)
f3 = cos(theta - c6)

g1 = sin(theta)
g2 = sin(theta + c6)
g3 = sin(theta - c6)

Te = - p*Msr*((i_phu*i_phk + i_phv*i_phl + i_phw*i_phm)*g1 + (i_phu*i_phl +
i_phv*i_phm + i_phw*i_phk)*g2 \
+ (i_phu*i_phm + i_phv*i_phk + i_phw*i_phl)*g3)

Tload = 0
} # end of values section

equations {
# Stator current for phase U (connection points U->X)

i(u->x) += i_phu
i_phu: d_by_dt(i_phu) = -a11*i_phu - (a12 + a13*omega_e)*i_phv - (a12 -
a13*omega_e)*i_phw \
- (a14*f1 - a15*omega_e*g1)*i_phk - (a14*f2 - a15*omega_e*g2)*i_phl - (a14*f3 -
a15*omega_e*g3)*i_phm + a16*(uu) \
+ a17*(uv) + a17*(uw) + a18*f1*(uk) + a18*f2*(ul) + a18*f3*(um)

# Stator current for phase V (connection points V->Y)

i(v->y) += i_phv
i_phv: d_by_dt(i_phv) = -a11*i_phv - (a12 - a13*omega_e)*i_phu - (a12 +
a13*omega_e)*i_phw \
- (a14*f3 - a15*omega_e*g3)*i_phk - (a14*f1 - a15*omega_e*g1)*i_phl - (a14*f2 -
a15*omega_e*g2)*i_phm + a17*(uu) \
+ a16*(uv) + a17*(uw) + a18*f3*(uk) + a18*f1*(ul) + a18*f2*(um)

# Stator current for phase W (connection points W->Z)

i(w->z) += i_phw
i_phw: d_by_dt(i_phw) = -a11*i_phw - (a12 + a13*omega_e)*i_phu - (a12 -
a13*omega_e)*i_phv \
- (a14*f2 - a15*omega_e*g2)*i_phk - (a14*f3 - a15*omega_e*g3)*i_phl - (a14*f1 -
a15*omega_e*g1)*i_phm + a17*(uu) \
+ a17*(uv) + a16*(uw) + a18*f2*(uk) + a18*f3*(ul) + a18*f1*(um)

# Rotor current for phase K (connection points K->n)

i(k->n) += i_phk

```

```

    i_phk: d_by_dt(i_phk) = -(a31*f1 - a32*omega_e*g1)*i_phu - (a31*f3 -
a32*omega_e*g3)*i_phv \
    - (a31*f2 - a32*omega_e*g2)*i_phw - (a34 - a35*omega_e)*i_phl - a33*i_phk - (a34
+ a35*omega_e)*i_phm \
    + a36*f1*(uu) + a36*f3*(uv) + a36*f2*(uw) + a37*(uk) + a38*(ul) + a38*(um)

    # Rotor current for phase L (connection points L->n)

    i(l->n) += i_phl
    i_phl: d_by_dt(i_phl) = - (a31*f2 - a32*omega_e*g2)*i_phu - (a31*f1 -
a32*omega_e*g1)*i_phv \
    - (a31*f3 - a32*omega_e*g3)*i_phw - (a34 + a35*omega_e)*i_phk - a33*i_phl - (a34 -
a35*omega_e)*i_phm \
    + a36*f2*(uu) + a36*f1*(uv) + a36*f3*(uw) + a38*(uk) + a37*(ul) + a38*(um)

    # Rotor current for phase M (connection points M->n)

    i(m->n) += i_phm
    i_phm: d_by_dt(i_phm) = - (a31*f3 - a32*omega_e*g3)*i_phu - (a31*f2 -
a32*omega_e*g2)*i_phv \
    - (a31*f1 - a32*omega_e*g1)*i_phw - (a34 - a35*omega_e)*i_phk - a33*i_phm - (a34
+ a35*omega_e)*i_phl \
    + a36*f3*(uu) + a36*f2*(uv) + a36*f1*(uw) + a38*(uk) + a38*(ul) + a37*(um)

    tq_Nm(wr) += Te - (d_by_dt(Jg*omega_m) + B*omega_m + Tload)

    theta: d_by_dt(theta)=(1-dc_domain)*omega_m*p + dc_domain*theta
}
}

```