CHAPTER SIX

Wind turbine generator modelling

This chapter describes the procedure to construct and run a wind farm simulation program considering doubly fed induction generator (DFIG)— and permanent magnet synchronous generator (PMSG)—based wind turbine technologies. By the end of the chapter, the readers will acquire the competence to simulate a wind turbine generator (WTG) connected to an infinite bus in Matlab/Simulink software. The chapter also includes an example to build a model of a wind farm containing mix of DFIG- and PMSG-type WTGs. It is advised to build the Simulink model while reading the chapter and use the example results provided to validate their model.

The chapter commences with modelling of DFIG-based WTG, which is divided into seven component models such as models of network, turbine, generator, filter, converter capacitor, machine-side converter (MSC) and grid-side converter (GSC). Equations describing each component model, figures for their Simulink model and associated Matlab programs are presented. Modelling of the DFIG is followed with simulation results on a single machine infinite bus (SMIB) test system. Dynamic simulation results and modal analysis results are presented for comparison. Programming of the PMSG-based WTG model is discussed later, which utilizes many of the subsystems of the DFIG system. At the end, the models developed are used to build a simulation program for a wind farm containing both types of WTGs.

6.1 Introduction

Wind is the fastest growing renewable energy source in the world. As the cost of electricity from wind, especially offshore wind farms, is decreasing, wind farms will contribute to a major portion of electricity generated in many electricity grids. Their characteristics will significantly influence the stability and operation of the grid soon. So, the modelling and analysis of wind turbines will play an important role in power system studies.

The early generation WTGs is fixed speed machines using induction generators. The rotor speed is fixed relative to grid frequency and independent of the wind speed. The output control is achieved using pitch and stall control mechanisms. The use of DFIG and power electronics converters

Simulation of Power System with Renewables
ISBN: 978-0-12-811187-1
https://doi.org/10.1016/B978-0-12-811187-1.00006-8

made variable speed operation possible, allowing greater power extraction under wide range of wind speeds. More recently, full converter—based WTGs have been developed, whose generator is isolated from grid through a power electronics converter making the generator rotor speed independent of the grid frequency. PMSGs are widely used in the full converter WTG configuration. The DFIG- and full converter—based WTGs offer reactive power and voltage control capabilities. This chapter will explain the modelling of DFIG- and PMSG-type WTGs.

Note: In the model developed in Chapter 3, variables of synchronous machine have 's_' prefix. Similarly, the DFIG and PMSG variables discussed in this chapter use 'd_' and 'p_' prefixes, respectively.

6.2 Building blocks of DFIG-SMIB simulation model

Single Machine Infinite Bus

Fig. 6.1 shows building blocks of a DFIG-SMIB system, which consists of a wind turbine, DFIG and network. The stator windings of the DFIG are fed directly from the grid, whereas the rotor windings are fed through a back-to-back converter, which is connected to the grid through a filter. The frequency and magnitude of the rotor voltage is controlled through the back-to-back converter, which can carry up to 30% of the active power output depending on the rotor speed. The filter is used to remove switching frequency harmonic components.

For modelling purposes, the system is divided into seven blocks, namely turbine, network, generator, MSC, back-to-back capacitor (B2BC), GSC and filter. The exchange of data between these main blocks is indicated using arrows in the figure. For example, inputs to the generator model are rotor speed from the turbine block and generator bus voltage from the network block. The generator model sends electrical torque to the turbine model and output current to the network model. The division into

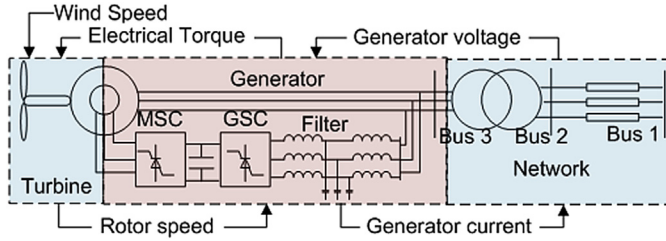

Figure 6.1 Schematic diagram of a single machine infinite bus (SMIB) system using doubly fed induction generator (DFIG)—type wind turbine generator.

submodels helps to develop and test individual blocks separately before integrating them to a DFIG-SMIB system simulation model. Readers are encouraged to develop the model and test as they go through this chapter to achieve the best outcome. The Script 6.1 contains parameters for the system at one operating condition, which can be used to test the individual blocks and the complete DFIG-SMIB system. Before proceeding to next

```
%Values associated with network
Znet = 0.0472 + 1i*0.4700;  d_Vinf = 1;  Dmachs = 3;
% Values associated with generator
d_Lm = 4;        d_Xm = d_Lm;     d_Rs = 0.005; d_Rr = 0.0055;
d_Lss = 4.04;    d_Lrr = 4.0602; d_kopt = 1;    d_ktg = 0.3;
d_ctg = 0.01;    d_Ht = 4;        d_Hg = 0.4;

d_Ls_d = d_Lss - (d_Lm^2/d_Lrr);
d_Kmrr = d_Lm/d_Lrr;
d_R2 = d_Kmrr^2*d_Rr;
d_R1 = d_Rs + d_R2;
d_Tr = d_Lrr/d_Rr;

Vdfig = 0.9794 + 1i*0.3983;
d_vsq = real(Vdfig);       d_vsd = imag(Vdfig);
d_Theta = angle(Vdfig);

d_isq = 0.8544;      d_isd = 0.2454;
d_irq =  -0.9629;    d_ird = -0.0020;
d_vrq =  0.0357;     d_vrd = 0.0154;

% Values associated with Filter
d_Ri= 0.0;    d_Rg= 0.0;     d_Rc= 0.7333;     d_Li= 0.1667;
d_Lg= 0.0033;     d_Cf = 0.0150;
d_iiq  = -0.0361;     d_iid  = 0.0024;
d_igq  = -0.0303;     d_igd  = -0.0123;
d_viq  = 0.9790;    d_vid  = 0.3922;
d_vcq  = 0.9837;    d_vcd  = 0.3874;


% Values associated with converters
d_Cdc = 2; d_VDC = 1.5;  d_Qfilter = 0;    d_Qs = 0.1;

d_MSC_IL1_kp = -0.23;  d_MSC_IL1_ki = -3;  d_MSC_IL1_iv = 0.0389;
d_MSC_IL2_kp = -0.23;  d_MSC_IL2_ki = -3;  d_MSC_IL2_iv = 0.00078156;
d_MSC_OL1_kp = 0;      d_MSC_OL1_ki = -60; d_MSC_OL1_iv = -0.8927;
d_MSC_OL2_kp = 0;      d_MSC_OL2_ki = 90;  d_MSC_OL2_iv = 0.3610;

d_GSC_IL1_kp = 0.3; d_GSC_IL1_ki = 200;  d_GSC_IL1_iv = 1.0546;
d_GSC_IL2_kp = 0.3; d_GSC_IL2_ki = 200;  d_GSC_IL2_iv = -0.0055;
d_GSC_OL1_kp = -22; d_GSC_OL1_ki = -870; d_GSC_OL1_iv = -0.0327;
d_GSC_OL2_kp = 0;   d_GSC_OL2_ki = -60;  d_GSC_OL2_iv = 0;


% Values associated with turbine
rho = 1.225;
d_wtrated = 3.0337; d_wt = 0.9688; d_wg = 0.9688;
d_bl = 40.05; d_Lambda = 8.1;
d_Beta = 0; d_vw = 14.5316; d_Ts = 0.9385;
```

Script 6.1 Test values for the doubly fed induction generator (DFIG)—single machine infinite bus (SMIB) model. Save as test_dfig.m.

section, copy this script and save it as *test_dfig.m*. Section 6.4 discusses the method to find these values for different operating conditions.

6.2.1 Network

In Chapter 3, we have developed a simulation program for a synchronous machine connected to an infinite bus. The Simulink block for the network developed in Section 3.5 has been reused for the DFIG-SMIB model. However, we will use a different bus and line matrix given in Script 6.2 to match with the new network in Fig. 6.1. The script is used to calculate the network impedance. These data refer to a 5 MW WTG on a 5 MVA base.

```
% This script contains the bus and line matrix for an example three
% bus
% network shown in Figure 6.1. The WTG is connected at Bus 3 and Bus
% 1 is the infinite bus. The WTG output is fixed and hence bus 3 is
% defined as a load bus. Ensure that functions,
% form_Ymatrix.m and power_flow.m, given in Chapter 2 are saved in
% the current working folder. The functions are used to obtain the
% power flow and admittance matrix. The network impedance to use in
% the network block of Simulink program is calculated from the
% admittance matrix.

clear all
%**** Part 1: Power Flow, Calculation of Network Impedance  *******
% bus data format
% bus: number, voltage(pu), angle(degree), p_gen(pu), q_gen(pu),
%      p_load(pu), q_load(pu),G-shunt (pu), B shunt (pu); bus_type
%      bus_type - 1, swing bus
%               - 2, generator bus (PV bus)
%               - 3, load bus (PQ bus)

bus = [...
    01 1.05  0.00  1.00  0.00  2.00   0.30  0.00  0.00  1;
    02 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    03 1.00  0.00  0.80  0.10  0.00   0.00  0.00  0.00  3];

% line data format
% line: from bus, to bus, resistance(pu), reactance(pu),
%       line charging(pu), tap ratio

line = [
    01  02 0.037  0.37  0.001 0.0 0.0;
    02  03 0.010  0.10  0.000 0.0 0.0];

Y = form_Ymatrix(bus,line);
[bus_sln, flow] = power_flow(Y, bus, line);
% Generator bus
Gen_Bus = 3;
% Infinite bus voltage
vinf = bus_sln(1,2).*exp(1i*bus_sln(1,3)*pi/180);
% Generator bus voltage
Vg = bus_sln(Gen_Bus,2).*exp(1i*bus_sln(Gen_Bus,3)*pi/180);
% Current injection from generator
d_ig = conj((bus_sln(Gen_Bus, 4)+1i*bus_sln(Gen_Bus, 5))/Vg);

Znet = (Vg - vinf)/d_ig;  % Network Impedance
```

Script 6.2 Network impedance calculation.

The bus matrix has three rows: one of them represents a slack bus and other two PQ buses. Bus 1 is the infinite bus and Bus 3 is where the WTG is connected. The WTG's active and reactive powers are specified in row 3, which can be changed to simulate different operating conditions. Line matrix has two rows indicating a transformer and infinite bus impedance. Now one can develop a network model as shown in Fig. 3.9 and Fig. 3.10 in Section 3.5.

6.2.2 Wind turbine model

The wind turbine model relates the wind velocity with generator rotor speed. A typical turbine configuration has three blades that capture wind energy and rotate a generator rotor. Depending on the generator speed requirements, a gearbox may be placed between the turbine and generator. As shown in Fig. 6.1, the turbine model block receives wind speed and electrical torque inputs and provides generator speed as output.

The turbine model is divided into two submodels. (1) An aerodynamic model converting wind energy to mechanical torque and (2) a drive train model relating the mechanical torque, electrical torque input from generator block and generator speed output.

6.2.2.1 Wind turbine aerodynamic modelling

The wind turbine aerodynamic model relates wind speed with turbine mechanical output. Let the example turbine have three blades of length R (40.05 m). The equation for mechanical power output is given by (6.1)

$$P_t = 0.5\rho\pi R^2 C_p(\beta, \lambda) v_w^3 [W] \tag{6.1}$$

where ρ is air density $= 1.225$ kg/m^3, $\pi = 3.1416$, v_w is the wind velocity in m/s and $C_p(\beta, \lambda)$ is the power coefficient of the blades. The theoretical maximum value of C_p is limited to 0.59, which is called Betz limit.

Eq. (6.1) shows that for a wind speed v_w, the turbine output depends on the C_p coefficient, which is a function of two parameters, β and λ. C_p represents the portion of wind power that can be extracted by the turbine. Usually manufacturers, using field-testing data, specify the performance curve for their turbine. However, for academic studies, numerical approximate equations can be used. This chapter will use one such equation (Mei, 2008) given in (6.2).

$$C_p(\beta, \lambda) = 0.5176\left(\frac{116}{\lambda + 0.08\beta} - \frac{4.06}{1+\beta^3} - 0.4\beta - 5\right) e^{\left(\frac{-21}{\lambda+0.08\beta} + \frac{0.735}{1+\beta^3}\right)} + 0.0068\lambda \tag{6.2}$$

where β is the pitch angle of the blade and λ is the tip speed ratio. The blades of the turbine can be turned in and out of the wind to control the turbine speed. The **pitch angle** refers to the angle of rotation of the blade on its longitudinal axis. For the wind turbine data used in this chapter, the pitch angle varies from 0 to 23 degrees; the pitch angle is zero degrees when the blades are facing the wind extracting maximum energy. The **tip speed ratio** is the ratio between the speed of the tip of a turbine blade and the wind speed, i.e., $\lambda = \omega_t R/v_w$, where ω_t is the turbine rotational speed in rad/sec.

Plots of C_p coefficient for different values of pitch angles (0, 4 and 8 degrees) are shown in Fig. 6.2. It is clear that C_p decreases as the blades are turned away from the wind by increasing the pitch angle. For a given wind speed v_w and pitch angle β, C_p (and hence turbine output power P_t) varies with the tip speed ratio λ. The tip speed ratio can be controlled by controlling the turbine speed ω_t.

Yaw control: This is another important mechanism to orient the turbine into the wind by turning the nacelle of the wind turbine. In older WTGs, the control was used to regulate the output. However, modern WTGs do not use Yaw control for power regulation. This control is not discussed

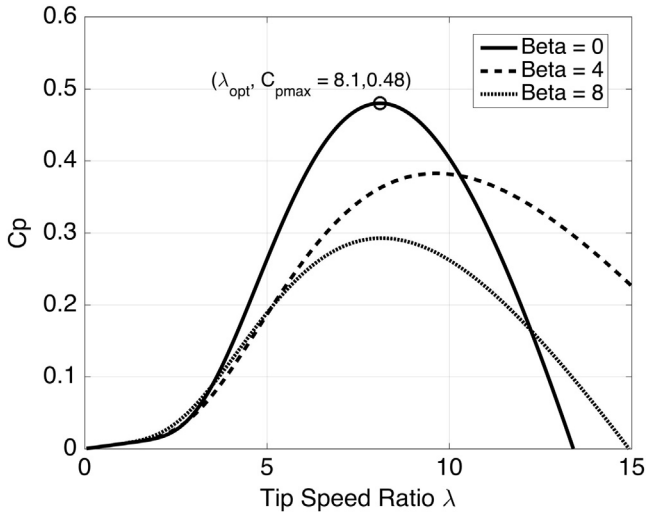

Figure 6.2 Plot of turbine performance coefficient versus tip speed ratio for different pitch angles.

further in this chapter. A simplified assumption is made such that the yaw control keeps the turbine aligned with wind direction always.

The maximum value for C_p is achieved for $\beta = 0$ when the blades are turned into the wind. A plot of C_p against λ for $\beta = 0$ is shown by a solid line in Fig. 6.2. For the given turbine parameters, the maximum value of C_p called $C_{pmax} = 0.48$ occurs at $\lambda_{opt} = 8.1$, which is the optimum value of tip speed ratio at $\beta = 0$. Substituting $C_p = C_{pmax}$ and $P_t = 5$ MW in (6.1), the wind speed required to produce the rated output at $\beta = 0$ is obtained as 15 m/s. This is the **rated wind speed** of the turbine. Above the rated wind speed, the pitch angle of the turbine is increased to reduce C_p to limit output to 5 MW. The operating region at and above rated wind speed is called **rated wind speed region**. Below the rated wind speed, the wind turbine output will be less than rated output, and the region is called **subrated operating region**.

In the subrated region, the maximum power from wind is extracted by setting pitch angle $\beta = 0$ and tip speed ratio $\lambda = \lambda_{opt}$. The later condition is met by adjusting the rotational speed of the turbine $\omega_t = \lambda_{opt} v_w / R$. However, turbine and generator designs restrict deviation in turbine rotational speed. For example, in case of a DFIG, rotor slip must be within ±30%. The performance coefficient will not be constant for the entire range of subrated operating region, especially close to **cut in wind speed**, which is the minimum wind speed required for a turbine to start producing useful output. However, for simplicity, assume $C_p = C_{pmax}$ at the subrated operating region (*This assumption results in constant tip speed ratio (8.1 at $\beta = 0$) under the subrated operating region is irrespective of wind speed. This means a large deviation in the rotor speed and slip, which is not acceptable. Hence, for the given parameters, the assumption $C_p = C_{pmax}$ makes the simulation model suitable for a limited range of wind speed. Readers must take note of this point. This issue can be fixed by using more accurate turbine parameters*). Above the rated wind speed, β and λ are varied to control C_p and hence limit the turbine output P_t to the rated value. The dashed and dotted plots in Fig. 6.2 show C_p curves for $\beta = 4$ degrees and $\beta = 8$ degrees, respectively. The pitch angle control is used to decrease C_p at above rated wind speed and hence controls turbine output. However, beyond a certain wind speed, called **cut out wind speed**, the turbine shuts down to protect itself from mechanical damage due to the strong wind. A plot between wind speed and turbine output for a typical WTG is shown in Fig. 6.3 The cut in wind speed, rated wind speed and cut out wind speed are clearly marked.

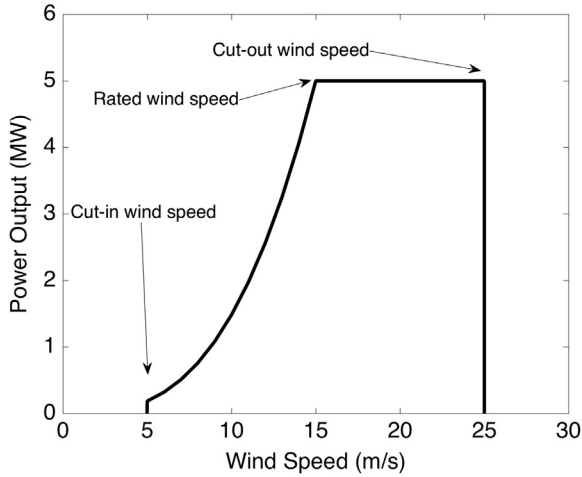

Figure 6.3 Turbine output versus wind speed plot.

6.2.2.1.1 Simulink representation of turbine model

Open a new Simulink model file and add a subsystem block. The subsystem block can be found in the 'commonly used blocks' section. Rename the subsystem block to Cp_Calc. Open the Cp_Calc block and implement the turbine performance coefficient equation in (6.2) as shown in Fig. 6.4. Save the model as Turbine_model.slx. To test the model, add two constant blocks and a display block as shown in Fig. 6.5. Now run test_dfig.m and Turbine_model.slx and verify the value in the display block.

Having implemented the Cp_Calc block, it is time to relate the wind speed input to turbine torque output. Append the model with additional elements to create the Simulink representation of the turbine aerodynamic model as shown in Fig. 6.6. The tip speed ratio is calculated using the equation $\lambda = \frac{\omega_t R}{v_w} = \frac{\omega_{t\ pu}\ \omega_{t\,\text{rated}} R}{v_w}$, where the turbine rated speed is $\omega_{t\,\text{rated}} = 3.0337$ rad/sec. The variable d_wt in the *constant* block represents pu speed. The *Gain3* block is used to convert the actual turbine power to the per unit value by dividing the actual value by the machine base (d_base). The output of the *Gain3* block is turbine mechanical power in per unit, and dividing it by the p.u. turbine speed produces turbine mechanical torque. Now run test_dfig.m and the Simulink model. Once model is validated, select all the components except the constant blocks and display block, click right mouse button and then select the *Create Subsystem From Selection* option to make a subsystem. Name the new subsystem *Turbine_Aero*. Refer to Fig. 6.9 for guidance.

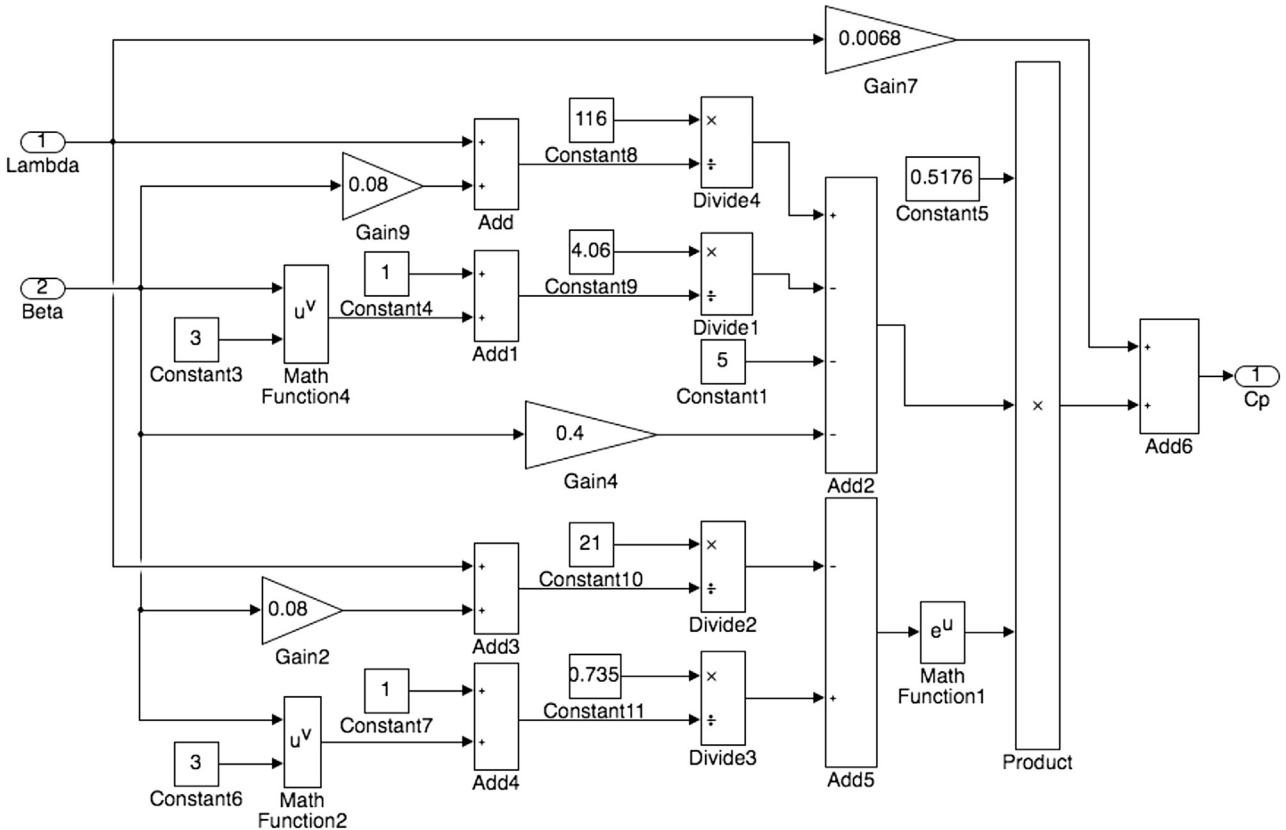

Figure 6.4 Simulink model of turbine performance coefficient equation.

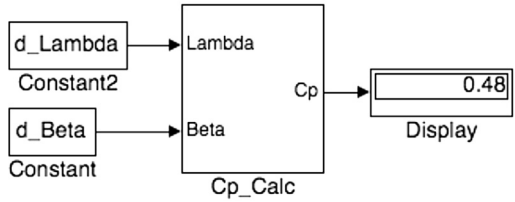

Figure 6.5 Simulink subsystem for turbine performance coefficient equation.

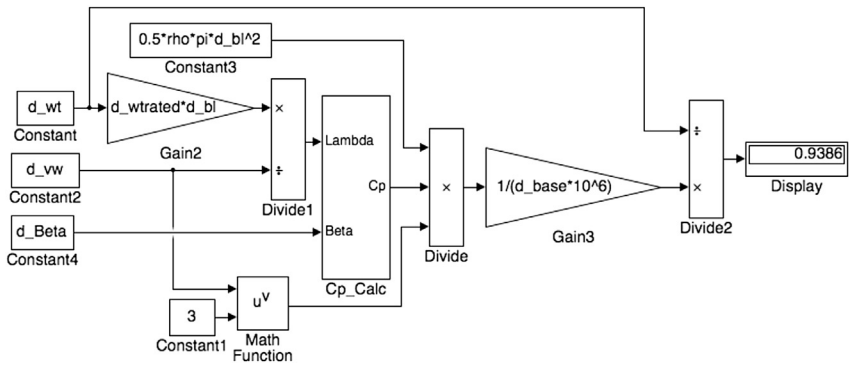

Figure 6.6 Simulation representation of turbine aerodynamic model.

6.2.2.2 *Turbine generator mechanical drive train model*

The second part of the wind turbine model is the drive train model relating the mechanical torque, electrical torque from the generator block and generator speed output. It represents the turbine, gearbox and generator masses. Models of varying complexity are reported in the literature, as already discussed in Chapter 3. For the simulations presented in this chapter, the drive train is modelled using two masses coupled through a shaft as shown in Fig. 6.7. Eqs. (6.3–6.6) are used to represent the model. The parameters and their values are listed in Table 6.1. The variable name used in the program is shown in brackets under the column symbol.

$$\frac{d}{dt}\omega_g = \frac{1}{2H_g}(T_s - T_g) \tag{6.3}$$

$$T_s = k_{tg}\theta_{tg} + c_{tg}\frac{d}{dt}\theta_{tg} \tag{6.4}$$

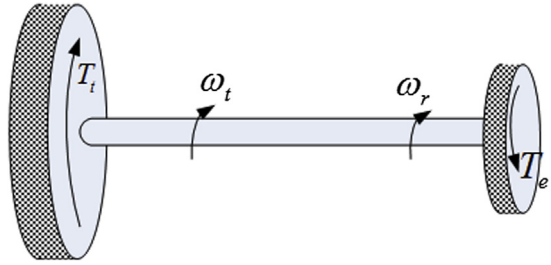

Figure 6.7 A two-mass drive train model of wind turbine.

$$\frac{d}{dt}\theta_{tg} = \omega_{elB}(\omega_t - \omega_g) \tag{6.5}$$

$$\frac{d}{dt}\omega_t = \frac{1}{2H_t}(T_t - T_s) \tag{6.6}$$

where T, ω and θ_{tg} represent torque, rotational speed and shaft twist angle. The subscripts g, s and t refer to generator, shaft and turbine, respectively.

The system of equations requires two inputs: turbine mechanical torque T_t and the electrical torque produced by the generator coupled to the turbine T_g. The speed of turbine ω_t and generator ω_g are outputs obtained from the system of equations. A Simulink representation of the equations is shown in Fig. 6.8. You are advised to compare the figure and equations and build the model yourself. As explained in Chapter 1, the initial values of state variables are specified in the block parameters dialogue box of the *Integration* block under *initial condition*. The initial values are calculated using a separate Matlab script file that will be discussed in Section 6.4. In this book, the variable for the initial value for each state variable is used as the name of the integration block. For example, in Fig. 6.8, d_Ts is the name of an integration block. Double click the block to open its block parameter dialogue

Table 6.1 Drive train parameters of the wind turbine.

Parameter	Symbol (variable name)	Value
Turbine inertia	H_t (d_Ht)	4 s
Generator inertia	H_g (d_Hg)	0.4 s
Drive train shaft stiffness	k_{tg} (d_ktg)	0.3 pu/el.rad
Drive train damping coefficient	c_{tg} (d_ctg)	0.01 pu.s/el.rad
Electrical base speed	ω_{elB} (d_welb)	$2\pi 50$ rad/s

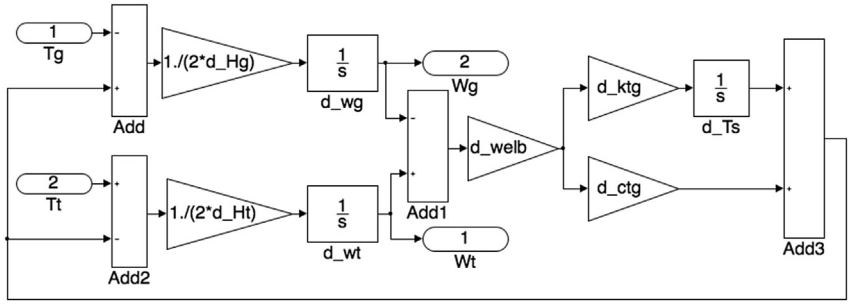

Figure 6.8 Drive train model of the wind turbine.

box and input d_Ts in the initial condition section of the dialogue box. Now convert the model to a subsystem named *Drive_Train* and connect it to the *Turbine_Aero* subsystem as shown in Fig. 6.9.

The integrated model takes wind speed and generator electrical torque as input and generator speed as output. Test the model after loading the variables from dfig_test.m. The turbine model of the DFIG-SMIB system is ready now.

6.1.3 Doubly fed induction generator

So far, the wind turbine and network representation have been discussed. The missing link is a generator model that takes generator rotor speed (ω_g) from the turbine model and generator bus voltage (v_g) from the network model as inputs and produces generator output current (i_g) and electrical torque (T_g) as outputs.

As the name suggests, a DFIG is an induction generator having two feeds: one through to the stator and another one through to the rotor windings. The stator is connected to a three-phase voltage at grid frequency (f_s = 50 or 60 Hz). The stator windings are designed such that the resulting stator currents produce a rotating magnet field in the air gap with angular

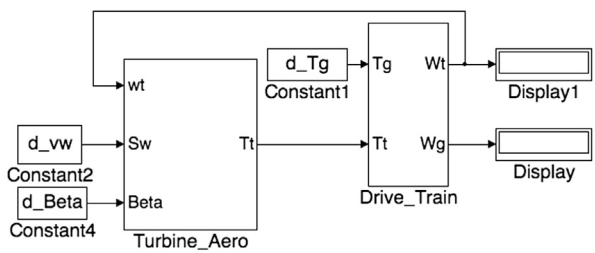

Figure 6.9 Turbine model of the single machine infinite bus (SMIB) system.

speed, $\omega_s = 2*pi*f_s$. To transfer energy from the rotor windings to the stator windings, there must be a synchronously rotating magnetic field produced by the rotor windings in the air gap. That means if the turbine drives the rotor at a speed $\omega_g = \omega_s$, a direct current through the rotor windings can produce a magnetic field rotating synchronously with the rotating field produced by the stator windings. But for a wind turbine, generator speed varies as the wind speed fluctuates. Suppose the turbine drives the rotor at a speed slightly higher or lower than the synchronous speed ($\omega_g \neq \omega_s$), an alternating rotor current at slip frequency $f_r = (\omega_s - \omega_g)f_s$ is required to generate a rotor field at the synchronous speed. This variable frequency feeding to rotor windings helps the DFIG-based WTG to continuously generate output even when the rotor speed varies due to changes in wind speed. This is the main feature that made the DFIG-WTG attractive to wind farm developers. As shown in Fig. 6.1, a back-to-back converter is used to produce the rotor voltage at variable frequency and magnitude. The full converter—based WTG discussed later has more flexibility as its converter isolates the generator from grid.

The model of the back-to-back converter is discussed in Sections 6.5—6.7. For the time being, let us develop a model of the DFIG assuming that the rotor voltage is supplied by an ideal voltage source as shown in Fig. 6.10. $v_{abc,s}$ and $v_{abc,r}$ are the three-phase voltage applied to stator and rotor windings, respectively. The differential equations representing the machine are presented in a d-q reference frame in which the q-axis is aligned with the infinite bus a-phase voltage. We have discussed the dq-reference frame concept in Chapter 3.

Eqs. (6.7—6.23) represent the differential and algebraic equations governing the dynamics of the DFIG (Mei, 2008; Abad et al., 2011; Wu et al., 2011) shown in Fig. 6.10. The definitions of the variables used in the model and their values are listed in Table 6.2.

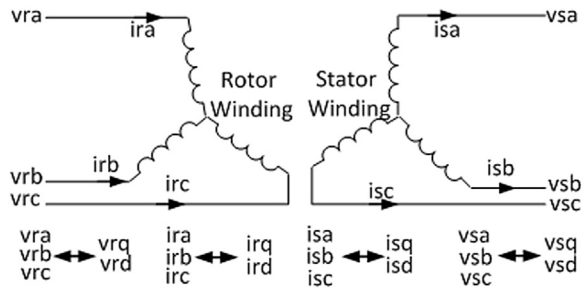

Figure 6.10 Simplified representation of a doubly fed induction generator (DFIG).

Table 6.2 Parameters of doubly fed induction generator.

Parameter	Symbol (variable name)	Value
Mutual inductance	L_m (d_Lm)	4 pu
Stator inductance	L_s (d_Ls)	4.04 pu
Rotor inductance	L_r (d_Lr)	4.0602 pu
Stator resistance	R_s (d_Rs)	0.005 pu
Rotor resistance	R_r (d_Rr)	0.0055
	$K_{mrr} = L_m/L_r$	
	$R_2 = K_{mrr}^2 R_r$	
	$R_1 = R_2 + R_s$	
	$L_s^{'} = L_s - L_m K_{mrr}$	
	$T_r = L_r/R_r$	
	$X_m = L_m$	

Stator currents (i_{sq}, i_{sd}) are

$$\frac{L_s'}{\omega_{elB}}\frac{d}{dt}i_{sd} = -\omega_s L_s' i_{sq} - R_1 i_{sd} + \frac{e_{sq}'}{\omega_s T_r} + \frac{\omega_g e_{sd}'}{\omega_s} - v_{sd} + K_{mrr} v_{rd} \tag{6.7}$$

$$\frac{L_s'}{\omega_{elB}}\frac{d}{dt}i_{sq} = -R_1 i_{sq} + \omega_s L_s' i_{sd} + \frac{\omega_g e_{sq}'}{\omega_s} - \frac{e_{sd}'}{\omega_s T_r} - v_{sq} + K_{mrr} v_{rq} \tag{6.8}$$

Voltages behind transient impedance (e_{sq}', e_{sd}') are

$$\frac{1}{\omega_s \omega_{elB}}\frac{d}{dt}e_{sd}' = -R_2 i_{sq} - \left(1 - \frac{\omega_g}{\omega_s}\right)e_{sq}' - \frac{e_{sd}'}{\omega_s T_r} + K_{mrr} v_{rq} \tag{6.9}$$

$$\frac{1}{\omega_s \omega_{elB}}\frac{d}{dt}e_{sq}' = R_2 i_{sd} - \frac{e_{sq}'}{\omega_s T_r} + \left(1 - \frac{\omega_g}{\omega_s}\right)e_{sd}' - K_{mrr} v_{rd} \tag{6.10}$$

where, e_{sq}' and e_{sd}' are related to rotor flux ψ_{rq} and ψ_{rd}, respectively, by

$$e_{sq}' = K_{mrr}\omega_s \psi_{rd} \tag{6.11}$$

$$e_{sd}' = -K_{mrr}\omega_s \psi_{rq} \tag{6.12}$$

The stator and rotor fluxes are given by

$$\psi_{sq} = L_s i_{sq} + L_m i_{rq} \tag{6.13}$$

$$\psi_{sd} = L_s i_{sd} + L_m i_{rd} \tag{6.14}$$

$$\psi_{rq} = L_r i_{rq} + L_m i_{sq} \tag{6.15}$$

$$\psi_{rd} = L_r i_{rd} + L_m i_{sd} \tag{6.16}$$

Rotor currents i_{rq} and i_{rd} are

$$i_{rq} = -\left(\frac{e'_{sd}}{X_m}\right) - K_{mrr} i_{sq} \tag{6.17}$$

$$i_{rd} = \left(\frac{e'_{sq}}{X_m}\right) - K_{mrr} i_{sd} \tag{6.18}$$

Stator active power P_s, rotor active power P_r, stator reactive power Q_s, rotor reactive power Q_r and electric torque T_g are given by

$$P_s = v_{sq} i_{sq} + v_{sd} i_{sd} \tag{6.19}$$

$$P_r = v_{rq} i_{rq} + v_{rd} i_{rd} \tag{6.20}$$

$$Q_s = -v_{sq} i_{sd} + v_{sd} i_{sq} \tag{6.21}$$

$$Q_r = -v_{rq} i_{rd} + v_{rd} i_{rq} \tag{6.22}$$

$$T_g = L_m (i_{sq} i_{rd} - i_{sd} i_{rq}) \tag{6.23}$$

Assume that stator voltage (v_{sq}, v_{sd}), rotor voltage (v_{rq}, v_{rd}) and rotor speed ω_g are available. The Simulink block representing (6.7) can be developed as shown in Fig. 6.11. Do not forget to assign the initial condition in the integration block. For representation of (6.7), assign the initial condition as

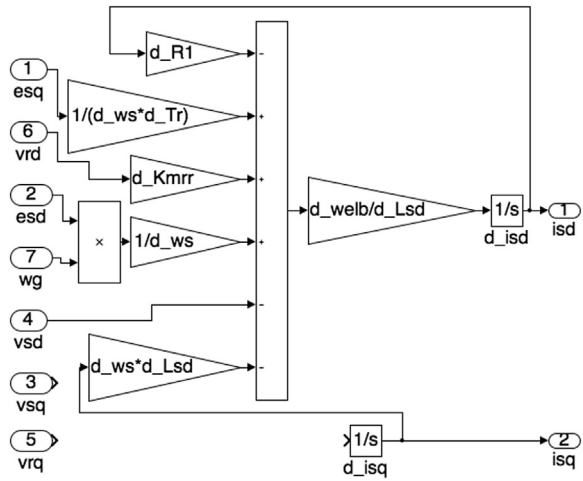

Figure 6.11 Simulink block for stator current equations.

d_isd. The representation of (6.8) is like that of Eq. (6.7) and so is omitted from Fig. 6.11 for the readers to complete. Develop a Simulink model for both equations, convert it to a subsystem and name it *isd_isq*. The *isd_isq* block should take seven inputs ($e_{sq}, e_{sd}, v_{sq}, v_{sd}, v_{rq}, v_{rd}, \omega_g$) and produce two outputs (i_{sq}, i_{sd}).

Similarly, develop subsystems *eds_eqs_idr_iqr* and *Output* subsystems as shown in Fig. 6.12. *eds_eqs_idr_iqr* represents Eqs. (6.9, 6.10, 6.17, 6.18), and the *Output* represents Eqs. (6.20), (6.21) and (6.23). Specify the initial values of state variables as d_esd, d_esq, d_isd and d_isq, in the respective integration blocks. The parameters for the model are listed in Table 6.2. Name the subsystem *Generator* as shown in Fig. 6.13.

Now assemble the turbine, generator and network blocks to form a DFIG_SMIB Simulink model as shown in Fig. 6.13. Note that the output current of DFIG consists of stator current and current through the filter. However, we have not developed the filter model yet. Hence, use constant

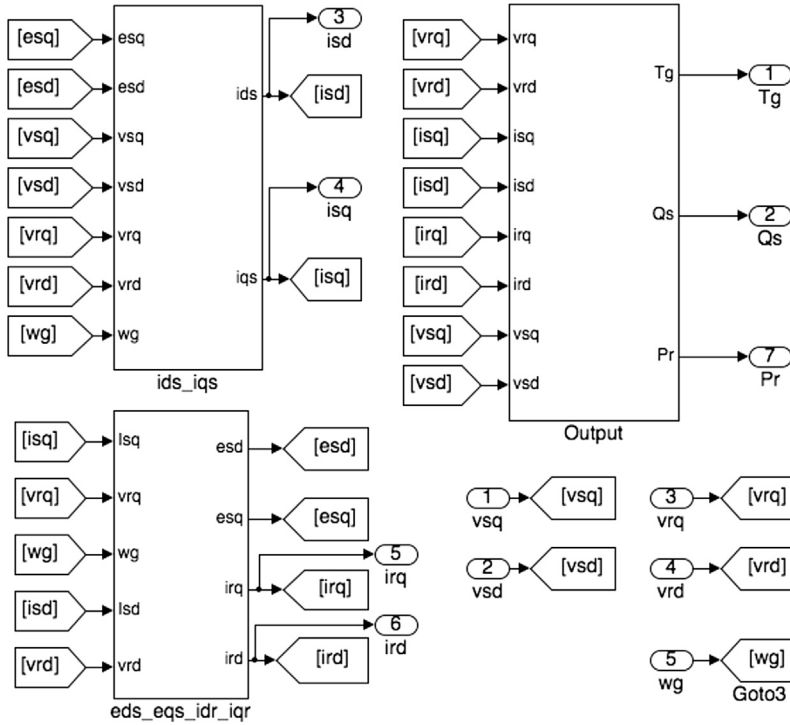

Figure 6.12 Inside view of the generator block.

blocks, d_iqg and d_idg, to represent the current output of filter. Test the model developed so far using the values in test_dfig.m. First run test_dfig.m and then run the Simulink model developed. Compare different variables in the simulation against the initial values in test_dfig. As the input remains constant, the value of state variables should not change. Two example values are shown in Fig. 6.13.

6.1.4 LCL filter

Now it is time to complete the DFIG-SMIB model by linking the rotor winding to grid, which consists of the back-to-back converter and filter. In this book, an LCL filter as shown in Fig. 6.14 is used. The filter consists of two inductors (L_i, L_g), a capacitor (C_f) and a damping resistor (R_c). The resistances in series with inductors (R_i, R_g) represent stray resistance of the inductor. Remember to use use 'd_' prefix for all the parameters and variables in the program. The design of LCL filters is discussed widely in

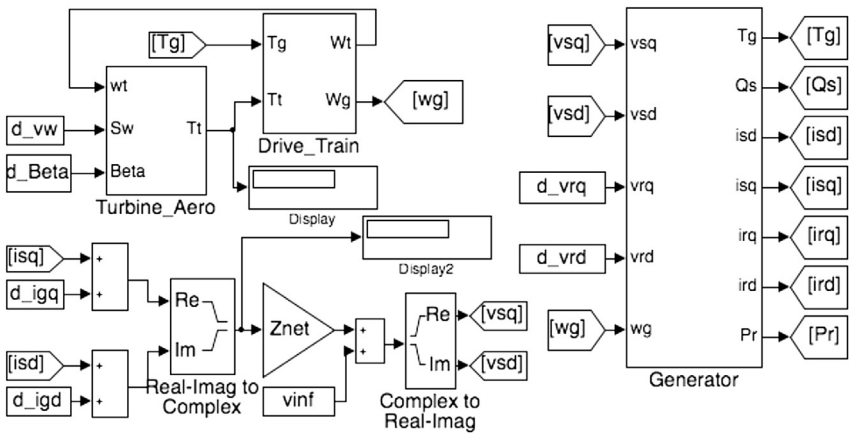

Figure 6.13 Doubly fed induction generator (DFIG)—single machine infinite bus (SMIB) Simulink model without converter and filter.

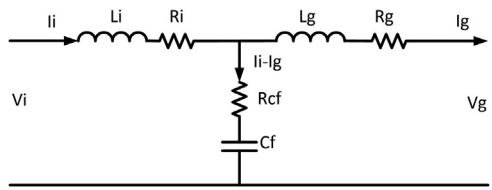

Figure 6.14 LCL filter.

Table 6.3

Parameter	Symbol (variable name)	Value (p.u)
Inverter-side inductance	L_i (d_Li)	0.1667
Inverter-side resistance	R_i (d_Ri)	0.0000
Grid-side inductance	L_g (d_Lg)	0.0033
Grid-side resistance	R_g (d_Rg)	0.0000
Filter capacitor	C_f (d_Cf)	0.0150
Damping resistance	R_c (d_Rc)	0.7333

literature (Rosyadi et al., 2012; Araujo et al., 2007), and example parameters used in this chapter are listed in Table 6.3. In the d-q synchronous reference frame, Eqs. (6.24)–(6.29) are used to represent the LCL model. In Fig. 6.1, the LCL filter connects the inverter output to Bus 3. This means the LCL filter model takes inverter voltages (v_{iq}, v_{id}) and Bus 3 voltages or stator voltage (v_{sq}, v_{sd}) as inputs and provides the current injected at Bus 3 through the filter (i_{gq}, i_{gd}) as output.

$$\frac{L_i}{\omega_b}\frac{d}{dt}i_{iq} = v_{iq} - v_{cq} - (R_i + R_c)i_{iq} + \omega L_i i_{id} + R_c i_{gq} \tag{6.24}$$

bit confusing....

$$\frac{L_i}{\omega_b}\frac{d}{dt}i_{id} = v_{id} - v_{cd} - (R_i + R_c)I_{id} - \omega L_i I_{iq} + R_c I_{gd} \tag{6.25}$$

$$\frac{L_g}{\omega_b}\frac{d}{dt}I_{gq} = v_{cq} - v_{gq} - (R_g + R_c)I_{gq} + \omega L_g I_{gd} + R_c I_{iq} \tag{6.26}$$

$$\frac{L_g}{\omega_b}\frac{d}{dt}I_{gd} = v_{cd} - v_{gd} - (R_g + R_c)I_{gd} - \omega L_g I_{gq} + R_c I_{id} \tag{6.27}$$

$$\frac{C_f}{\omega_b}\frac{d}{dt}v_{cq} = I_{iq} - I_{gq} - \omega C_f V_{cd} \tag{6.28}$$

$$\frac{C_f}{\omega_b}\frac{d}{dt}v_{cd} = I_{id} - I_{gd} + \omega C_f V_{cq} \tag{6.29}$$

A Simulink representation of the LCL filter model is shown in Fig. 6.15, which represents (6.24)–(6.29). The Iside (inverter side), Gside (grid side) and capacitor blocks in the figure represent Eqs. 6.24–6.29, respectively. Ensure that the 'd_' prefix is used in the parameters and state variables. For example, v_{cd} must be entered as d_vcd. The figure contains two

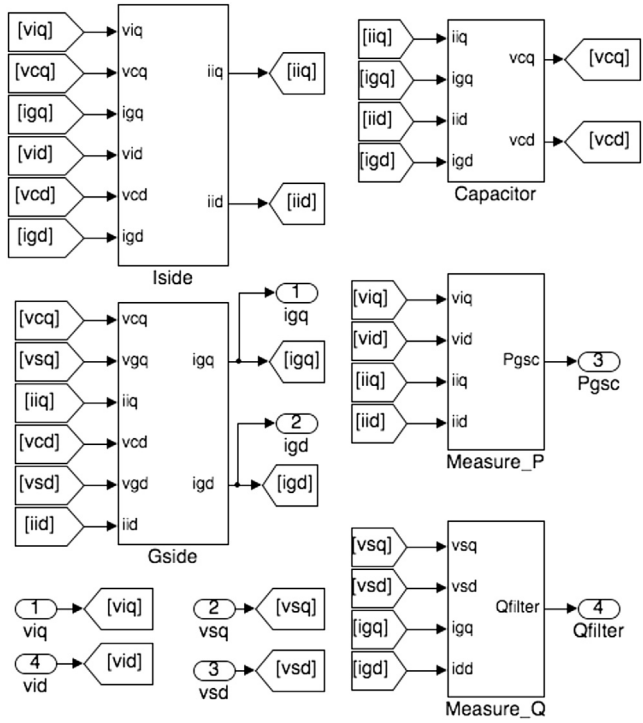

Figure 6.15 LCL filter Simulink model.

additional blocks Measure_P and Measure_Q representing (6.30) and (6.31), respectively. Eq. (6.30) represents active power output at the converter terminal, and Eq. (6.31) represents reactive power output at the filter terminal. Both outputs will be used for the GSC blocks discussed in Section 6.6.7.

$$P_{gsc} = v_{iq} i_{iq} + v_{id} i_{id} \tag{6.30}$$

$$Q_{gsc} = -v_{sq} i_{gd} + v_{sd} i_{gq} \tag{6.31}$$

6.1.5 Back-to-back capacitor

The back-to-back converter connects the rotor windings with the grid through the LCL filter. It consists of a MSC, dc link capacitor and a GSC. The switching effects of the converters are neglected. It is assumed that the converter dynamics are fast and that the converters follow the reference generated by the converter controller in real time; hence, it is sufficient to model the control alone.

Neglecting switching and conduction losses in the converter, the dynamics of the capacitor voltage can be represented using (6.32). Here, P_{msc} is the active power flowing through the MSC, P_{gsc} is the active power flowing through GSC, v_{dc} is the capacitor voltage and C_{dc} is the capacitance. In case of the DFIG, P_{msc} is equal to rotor power P_r. At steady-state, rotor power (P_r) is equal to GSC output power (P_{gsc}), so that the capacitor voltage (v_{dc}) is constant. Build a subsystem named B2BC for the capacitor with MSC power (rotor power) and GSC power as inputs to the subsystem and capacitor voltage as output of the subsystem. The block is shown in Fig. 6.21.

$$\frac{1}{C_{dc}} p v_{dc} = \frac{1}{v_{dc}} (P_{msc} - P_{gsc}) \tag{6.32}$$

6.1.6 Machine-side converter controller

There is an important distinction between the blocks so far developed and the controllers. The controller models represent a set of software code run in a microcontroller, whereas the models so far developed represent mechanical or electrical systems. The controller models receive electrical signals corresponding to voltage, current and generator rotor speed and produce switching signals for the converter. The converters use a vector control approach where independent control of active power or torque and reactive power or voltage is achieved. However, the independent control is only possible when one of the d-q axes is aligned with the stator voltage. The generator model so far developed assumes that the q-axis is aligned with infinite bus voltage. The current and voltage measurements must be aligned to a new reference frame inside the controllers. A simplified version of controllers consisting of two cascaded PI controllers will be used in this chapter. There are numerous variations of controllers proposed in the literature. Once a working simulation is developed, the readers can explore these methods and incorporate them into their simulation program.

A block diagram representation of the MSC controller is shown in Fig.6.16. It has two cascaded PI control loops in the d- and q-axis. The inner loops (MSC_IL1, MSC_IL2) are fast current controllers and outer loops (MSC_OL1, MSC_OL2) are slower torque or reactive power controllers.

Reactive power and torque reference: The reactive power reference (Qs) can be selected based on the grid requirement. In this simulation, the reactive power output reference is obtained from the power flow solution. When the DFIG is operating at the rated operating condition, the torque reference can be set to 1 p.u. However, at subrated operating condition,

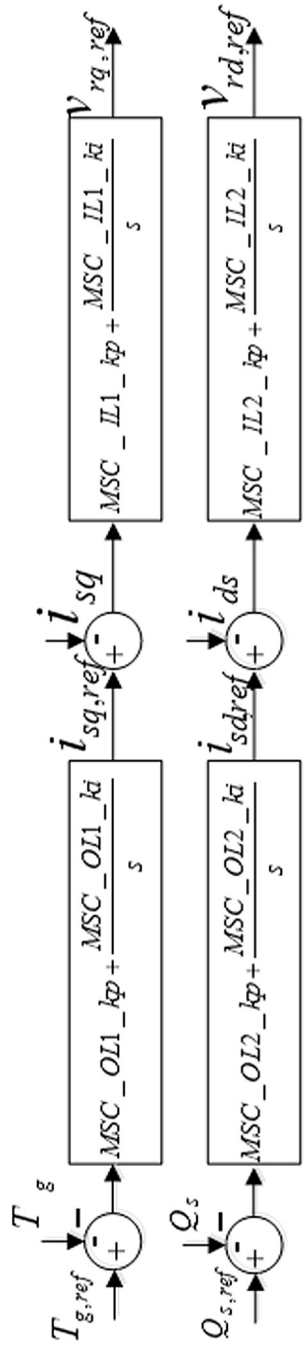

Figure 6.16 Machine-side converter (MSC) block diagram representation.

the reference torque can be set to extract maximum wind power, called maximum power point tracking (MPPT). The MPPT is achieved by setting the pitch angle to zero and controlling the turbine speed such that the tip speed ratio $\lambda = \lambda_{opt}$. Let $P_t = P_{t\,\text{rated}} T_t \omega_t$ (W), $v_w = \omega_{t\,\text{rated}} \omega_t R / \lambda_{opt}$ (m/s) and $C_p(\beta, \lambda) = C_{pmax}$ in (6.1)

$$P_{t\,\text{rated}} T_t \omega_t = 0.5 \rho \pi R^2 C_{pmax} \left(\frac{\omega_{t\,\text{rated}} \omega_t R}{\lambda_{opt}} \right)^3 \tag{6.33}$$

From (6.33), the reference value of torque is $T_t = K_{opt} \omega_t^2$ at subrated operating condition, where $K_{opt} = 0.5 \rho \pi R^5 C_{pmax} \omega_{t\,\text{rated}}^3 \Big/ \left(\lambda_{opt}^3 P_{t\,\text{rated}} \right)$. For the set of turbine parameters given in Section 6.6.2, $K_{opt} = 1$.

The Simulink representation of the MSC block is shown in Fig. 6.17. As explained before, inside the controller, the currents (i'_{rq}, i'_{rd}) and voltages (v'_{rq}, v'_{rd}) are in a new d-q reference frame where the q-axis is aligned with the stator voltage. Remember, in the model developed so far, the currents (i_{rq}, i_{rd}) and voltages (v_{rq}, v_{rd}) are in a d-q reference frame where the q-axis is aligned with the infinite bus voltage. The conversion between the two-reference frames is carried out by rotating the phasor to an angle θ, which is the angle of the stator voltage (DFIG bus voltage). The dq2dq′ and dq′2dq blocks at both the ends of Fig. 6.17 represent the conversion between the two-reference frames. For example, in the dq2dq′ block,

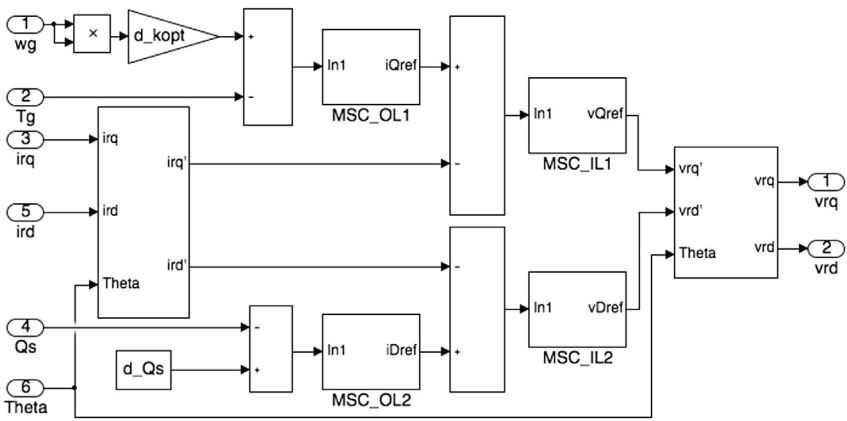

Figure 6.17 Simulation model of machine-side converter (MSC).

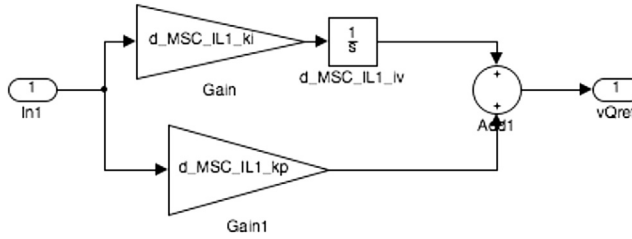

Figure 6.18 Example PI controller structure.

current is transformed using $i'_{rq} + j*i'_{rd} = (i_{rq} + j*i_{rd})e^{-j\theta}$, and in the dq'2dq block, voltage is transformed using $v_{rq} + j*v_{rd} = \left(v'_{rq} + j*v'_{rd}\right)e^{j\theta}$.

There are four PI controllers (MSC_OL1, MSC_OL2, MSC_IL1, MSC_IL2) having a common structure. Fig. 6.19 shows the structure of MSC_OL1. The gains and initial condition of the block are d_MSC_OL1_kp, d_MSC_OL1_ki and d_MSC_OL1_iv. Ensure that the correct variable names of the initial condition are specified in the integration blocks. Their values are assigned in the program given in Script 6.1. The other three PI controller blocks have same structure, but with slight change in the variable names. For example, for MSC_IL2 block, the gains and initial condition of the block are d_MSC_IL2_kp, d_MSC_ IL2_ki and d_MSC_ IL2_iv.

6.1.7 Grid-side converter controller

Fig. 6.19 shows the block diagram representation of the GSC controller. Like the RSC controller, it also has two cascaded PI controllers regulating the capacitor voltage and reactive power flow from the GSC at the WTG bus. By regulating the capacitor voltage to its reference value, the rotor power will be transferred to the grid ($P_r = P_{gsc}$). The simulation presented in this chapter will assume that no reactive power is transferred to the grid through the GSC; hence, the reactive power reference is set to zero. Fig. 6.20 shows the Simulink representation of the GSC controller, which can be built like the RSC controller. The structure of the PI controllers is like the one in Fig. 6.18. The gains and initial condition names must be changed. For example, in GSC_OL1 blocks, the values are d_GSC_OL1_kp, d_GSC_OL1_ki and d_GSC_OL1_iv. As explained earlier, the voltage and current phasors are aligned to the new d-q reference

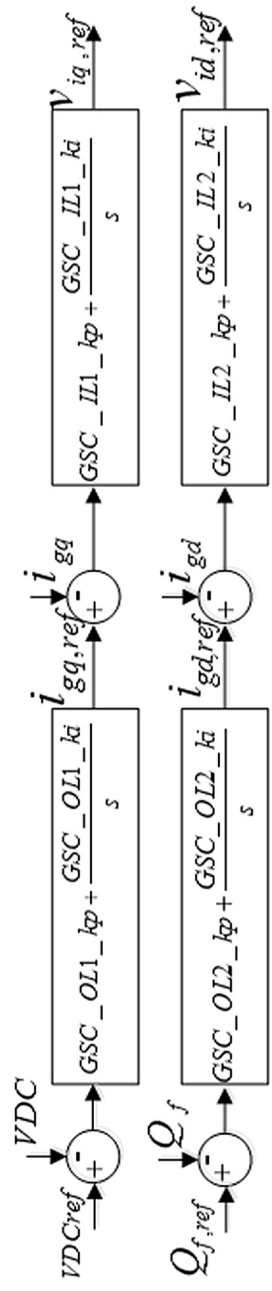

Figure 6.19 Block diagram representation of grid-side converter (GSC) controller.

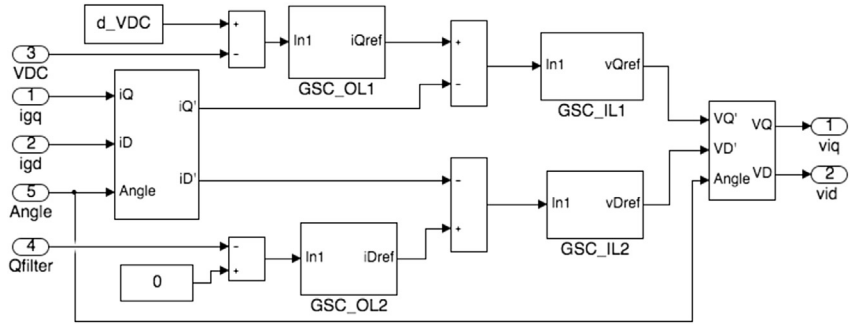

Figure 6.20 Simulation representation of grid-side converter (GSC) controller.

frame inside the controller to satisfy the independent active and reactive power control requirement.

6.3 Single machine infinite bus model integration and testing

Now all the Simulink blocks required for the SMIB simulation are ready, and it is time to integrate the model and test it. Connect the blocks so far developed as shown in Fig. 6.21 and save the model as smib_dfig_sim. Run test_dfig.m and run the simulation. Validate the output against Fig. 6.21. Now linearize the model using the *linmod* command and calculate the eigenvalues using the *eig* command as described in Chapter 4. Table 6.4 shows the eigenvalues of the SMIB system for comparison. If you are getting the same eigenvalues, the model developed is correct. If it is not, try to debug it using the procedure explained in Section 3.6.

6.3.1 Dynamic simulation

Let us run a dynamic simulation. Make two changes in the model. (1) Replace d_vw constant block with a step block. Open the block parameter dialogue and make changes: step time = 1, initial value = d_vw and final value = d_vw-2. The changes mean at time $t = 1$ s, the wind speed will reduce by 2 m/s from its initial value. (2) Replace vinf block by a step input block and make changes in its block parameter dialogue. Step time = 10 s, initial value = vinf and final value = vinf+0.02. Now change the simulation stop time to 20 s and run the model. Some of the example outputs are shown in Fig. 6.22. Are you getting same outputs? Congratulations. Your model is ready now.

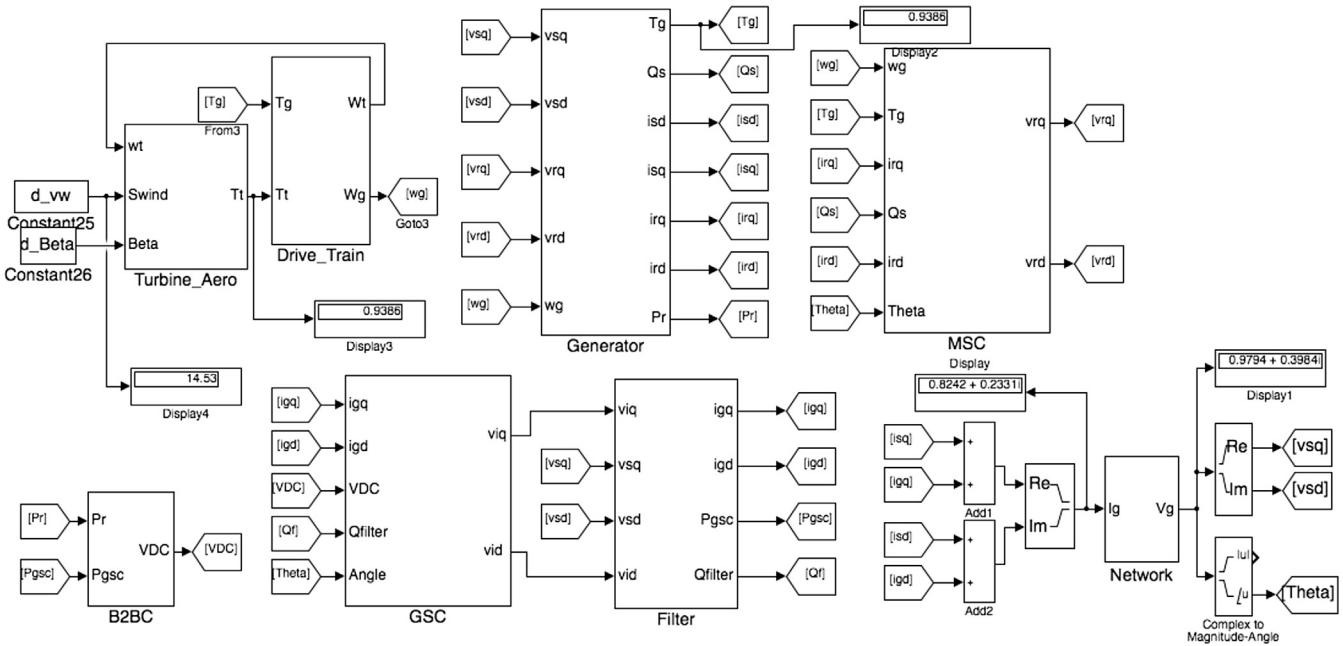

Figure 6.21 Simulation model of single machine infinite bus (SMIB) system using doubly fed induction generator (DFIG).

Table 6.4 Eigenvalues of the single machine infinite bus (SMIB)—doubly fed induction generator (DFIG) system.

Eigenvalues	Frequency (Hz)	Damping ratio (%)
$-58581 \pm 62892i$	10010	68.15
$-16564 \pm 17901i$	2849.1	67.92
$-674.3 \pm 1979.9i$	315.11	32.23
$-322.3 \pm 645.5i$	102.73	44.67
$-211.4 \pm 335.4i$	53.37	53.32
$-79.40 \pm 97.15i$	15.46	63.28
-62.12	0	100
$-37.76 \pm 74.71i$	11.89	45.10
$-4.15 \pm 16.91i$	2.69	23.87
-12.91	0	100
-11.54	0	100
$-2.94 \pm 11.01i$	1.75	25.85
-0.33	0	100

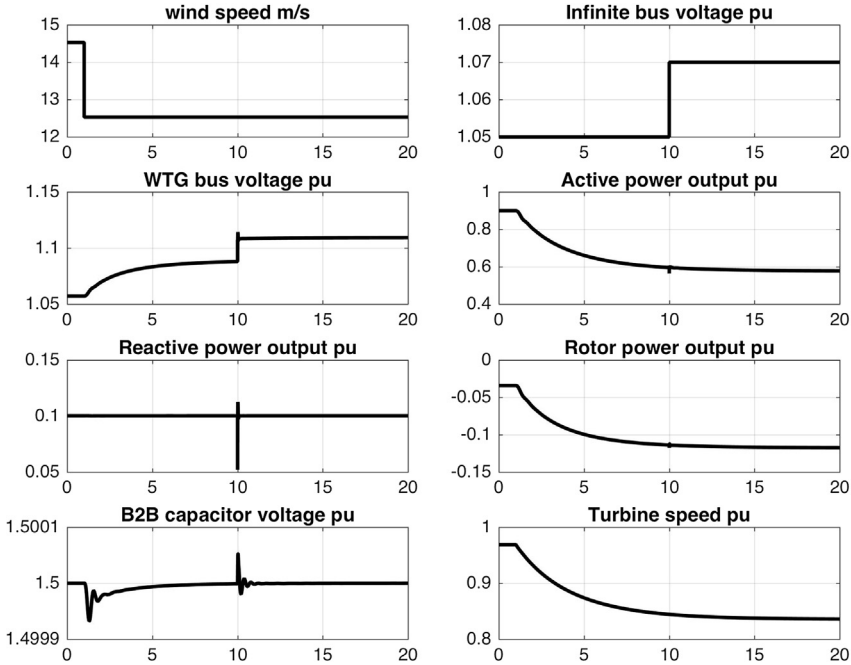

Figure 6.22 Dynamic simulation results of doubly fed induction generator (DFIG)—single machine infinite bus (SMIB) system.

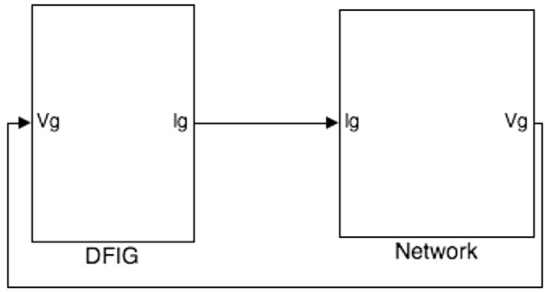

Figure 6.23 Complete single machine infinite bus (SMIB)—doubly fed induction generator (DFIG) simulation model.

So far we have used the initial values in Script 6.1 for the Simulink model. It corresponds to a power output of $0.8 + 0.1j$ pu. This is not very helpful, as we may like to run the model at different operating conditions, which requires different sets of initial values. Section 6.4 discusses the steps required to find initial values of state variables at any operating condition.

Before we move to initialization, let us create a subsystem named DFIG as shown in Fig. 6.23. The network block takes complex value of DFIG output current and outputs DFIG bus voltage. The DFIG block can be readily used in the wind farm simulation model developed in Section 6.9.

6.4 Initialization of SMIB-DFIG system

The initialization program finds the initial value of all state variables in the system, which is used in the integration blocks. The program for the DFIG initialization is given in Scripts 6.2–6.4. The program can be divided into the following simple steps.

Step 1: Run the power flow program to determine the voltage and power injection at the WTG bus.

```
Dmachs = [3];      % Bus where PMSG is connected
Omega = 2*pi*50;
if size(Pmachs,1)
      find_dfig_state_initial_conditions
end
```

Script 6.3 Program to call doubly fed induction generator (DFIG) initialization.

```
%STEP 2 BEGINS: Gather machine parameter for DFIG
d_welb = 2*pi*50; % electrical base speed rad/sec
d_ws = 1.0;      % Synchronous speed in pu

% DFIG parameters
%data_DF= [mach# bus# d_Lm  d_Rs   d_Rr    d_Lss d_Lrr  d_Kopt d_ktg d_ctg  d_Ht d_Hg BaseMVA]
data_DF = [1     5    4     0.005 0.0055 4.04  4.0602 1      0.3   0.01   4    0.4  5];

% DFIG-Filter parameters
%FLTR = [d_Ri, d_Rg, d_Rc,  d_Li,  d_Lg,  d_Cf];
d_FLTR =  [0.000, 0.000, 0.7333, 0.1667, 0.0033, 0.0150];
d_Cdc = 2; % Converter capacitor

%DFIG-MSC Controller Parameters
d_MSC_IL1_kp = -0.23;  d_MSC_IL1_ki = -3;
d_MSC_IL2_kp = -0.23;  d_MSC_IL2_ki = -3;
d_MSC_OL1_kp = 0;      d_MSC_OL1_ki = -60;
d_MSC_OL2_kp = 0;      d_MSC_OL2_ki = 90;

% DFIG-GSC Controller Parameters
d_GSC_IL1_kp = 0.3; d_GSC_IL1_ki = 200;
d_GSC_IL2_kp = 0.3; d_GSC_IL2_ki = 200;
d_GSC_OL1_kp = -22; d_GSC_OL1_ki = -870;
d_GSC_OL2_kp = 0;   d_GSC_OL2_ki = -60;

% Read filter parameters from FLTR vector
d_Ri= 0.0;
d_Rg= 0.0;
d_Rc= d_FLTR(3);
d_Li= d_FLTR(4);
d_Lg= d_FLTR(5);
d_Cf = d_FLTR(6);

% Read machine data from data_DF
d_Lm = data_DF(3);
d_Xm = d_Lm;
d_Rs = data_DF(4);         % stator resistance
d_Rr = data_DF(5);         % rotor resistance
d_Lss = data_DF(6);        % stator self inductance
d_Lrr = data_DF(7);        % rotor inductance
d_kopt = data_DF(8);
d_ktg = data_DF(9);
d_ctg = data_DF(10);
d_Ht = data_DF(11);
d_Hg = data_DF(12);
d_base = data_DF(13);

d_bl = 40.05;   % blade length
d_wtrated = 3.0337;  % rated turbine speed
rho = 1.225;  % air density

% Calcuating derived variable using equation () - ()
d_Ls_d = d_Lss - (d_Lm^2/d_Lrr);
d_Kmrr = d_Lm/d_Lrr;
d_R2 = d_Kmrr^2*d_Rr;
d_R1 = d_Rs + d_R2;
d_Tr = d_Lrr/d_Rr;
```

Script 6.4 Program for find_dfig_state_initial_conditions.m.

```
    % STEP 2 ENDS

    %STEP 3 BEGINS: Initialization of State variables for generator,
%Converter and Filter
    Vdfig = bus_sln(Dmachs,2).*exp(1i*bus_sln(Dmachs,3)*pi/180);
    Pdfig = bus_sln(Dmachs, 4);
    Qdfig = bus_sln(Dmachs, 5);
    d_vsq = real(Vdfig);      d_vsd = imag(Vdfig);
    d_Theta = angle(Vdfig);

xdfig  = zeros(size(Dmachs,1),15);
    for d_index = 1:size(Dmachs,1)
        xdfig0 = ones(1,15);
        if Pdfig(d_index)<1    % If below rated speed

% The function fsolve is used to solve set of SSCs described in the
%function init_dfig_mpt. The output is stored in xdfig variable
            xdfig(d_index,:) = fsolve(@(x)...
init_dfig_mpt(x,Vdfig(d_index),Pdfig(d_index),...
Qdfig(d_index),data_DF,d_FLTR),...
            xdfig0,optimset('TolFun',1e-16,'TolX',1e-16));
        else
% The function fsolve is used to solve set of SSCs described in the
%function init_dfig_cpt. The output is stored in xdfig variable

            xdfig(d_index,:) = fsolve(@(x)...
init_dfig_cpt(x,Vdfig(d_index),Pdfig(d_index),...
Qdfig(d_index),data_DF,d_FLTR),...
            xdfig0,optimset('TolFun',1e-16,'TolX',1e-16));
        end
    end

    d_isq =  xdfig(:,1);     d_isd = xdfig(:,2);
    d_irq =  xdfig(:,3);     d_ird = xdfig(:,4);
    d_vrq =  xdfig(:,5);     d_vrd = xdfig(:,6);
    d_iiq  = xdfig(:,7);    d_iid  = xdfig(:,8);
    d_igq  = xdfig(:,9);    d_igd  = xdfig(:,10);
    d_viq  = xdfig(:,11);   d_vid  = xdfig(:,12);
    d_vcq  = xdfig(:,13);   d_vcd  = xdfig(:,14);
    d_wg   = xdfig(:,15);

% d_esq and d_esd are calculated using (6.12) and (6.13)
    d_esq =  d_Kmrr.*d_ws.*(d_Lrr.*d_ird + d_Lm.*d_isd);
    d_esd = -d_Kmrr.*d_ws.*(d_Lrr.*d_irq + d_Lm.*d_isq);

    %******** STEP 4 BEGINS: Initialization of converter controllers
%and turbine  *******
    % Parameters for RSC controller model
    d_vr_dash =(d_vrq+1i*d_vrd).*exp(-1i*d_Theta);
    d_ir_dash =(d_irq+1i*d_ird).*exp(-1i*d_Theta);
    d_MSC_IL1_iv = real(d_vr_dash);
    d_MSC_IL2_iv = imag(d_vr_dash);
    d_MSC_OL1_iv = real(d_ir_dash);
    d_MSC_OL2_iv = imag(d_ir_dash);

    d_Qs = Qdfig;   % Reactive power reference RSC controller


    % B2B capacitor
    d_VDC = 1.5;
```

Script 6.4 (*continued*).

```
            % Parameters for GSC controller model
            d_vi_dash = (d_viq+1i*d_vid).*exp(-1i*d_Theta);
            d_ii_dash =  (d_igq+1i*d_igd).*exp(-1i*d_Theta);

            d_GSC_IL1_iv = real(d_vi_dash);
            d_GSC_IL2_iv = imag(d_vi_dash);
            d_GSC_OL1_iv = real(d_ii_dash);
            d_GSC_OL2_iv = imag(d_ii_dash);

            d_Qfilter = 0;  % Reactive power reference GSC controller

            % Turbine initialisation
            d_Tg = d_esq.*d_isq+d_esd.*d_isd;
            d_Ts = d_Tg;
            d_wt = d_wg;
            d_Pt = d_Ts.*d_wt;

            for d_index = 1:size(Dmachs,1)

                if d_wg(d_index)>=1   % Above rated wind speed operation
                    d_Sw(d_index) = 15;
                    d_Lambda = 3.0337*d_wg(d_index).*d_bl/d_Sw(d_index);
    Cp_req = (d_base*d_Pt(d_index)*1e6)/(0.5*1.225*pi*d_bl^2*d_Sw(d_index)^3);
                    d_Beta(d_index) = find_beta(Cp_req, d_Lambda);
                else                          % Below rated wind speed
                    d_Lambda = 8.1;
                    d_Beta = 0;
d_Sw(d_index,1) = (d_Pt(d_index)*d_base*1e6/(0.5*1.225*pi*d_bl^2*0.48))^(1/3);

                end
            end
```

Script 6.4 (*continued*).

Script 6.2 contains the program for this step. The system data (bus_sln, line_sln) is obtained using power flow code in Chapter 2. The *form_Ymatrix* and *power_flow* functions explained in Chapter 2 must be saved in the current working directory. Create a new file dfig_smib_main.m by combining Scripts 6.1 and 6.3.

Create a new .m file named find_dfig_state_initial_conditions.m and copy the program from Script 6.4. The script is divided into the following steps as discussed below.

Step 2: Gather machine parameters. The generator and turbine parameters are specified as data_DF vector. Each column of data_DF represents machine id, bus number, L_m, R_s, R_r, L_s, L_r, k_{opt}, k_{tg}, c_{tg}, H_t, H_g and base MVA at which the data are specified. Filter data are bundled to vector FLTR where the elements represent R_i, R_g, R_c, L_i, L_g and C_f.

Step 3: Initialization of state variables for generator, converter and filter.

The differential equations representing the generator and filter are presented in the previous section. Steady-state equations of the model can be obtained by setting the differential term $\left(\frac{d}{dt}\right)$ of the equation to zero. For

easy reference in the Matlab script, let us call these equations steady-state conditions (SSCs). We will define 15 of them for a subrated operating condition.

SSC 1–4: The first four conditions represent the steady-state voltage equations of the DFIG given by (6.34)–(6.37).

$$v_{qs} = -R_s i_{qs} + \omega_s (L_{ss} i_{ds} + L_m i_{dr}) \tag{6.34}$$

$$v_{ds} = -R_s i_{ds} - \omega_s (L_{ss} i_{qs} + L_m i_{qr}) \tag{6.35}$$

$$v_{qr} = -R_r i_{qr} + s\omega_s (L_{rr} i_{dr} + L_m i_{ds}) \tag{6.36}$$

$$v_{dr} = -R_r i_{dr} - s\omega_s (L_{rr} i_{qr} + L_m i_{qs}) \tag{6.37}$$

where $\omega_r = (1 - s)\omega_s$.

SSC 5: At steady state, the electrical torque (6.24) produced by the machine under subrated operating condition is equal to $K_{opt}\omega_t^2$. At rated operating condition, this is equal to 1.

SSC6: Sum of active power output through stator and rotor windings $(P_s + P_r)$ in (6.20) and (6.21) is equal to active power injection at the WTG bus, which is obtained from the power flow solution.

SSC 7: The reactive power generation from stator in (6.22) is equal to reactive power injection at the WTG bus, which is obtained from the power flow solution. This assumes that no reactive power is supplied to grid through the filter path.

SSC 8: The reactive power injection at the WTG bus through the filter is zero, i.e., $-v_{sq} i_{gd} + v_{sd} i_{gq} = 0$.

SSC 9: Assuming no converter losses, rotor output power $(v_{rq} i_{rq} + v_{rd} i_{rd})$ is equal to GSC output power $(v_{iq} i_{iq} + v_{id} i_{id})$.

SSC 10–15: The last six equations represent the steady-state equations of the LCL filter, which are obtained by setting the left-hand side of Eqs. (6.24)–(6.29) to zero.

The SSCs 1–15 can be represented as $F = Ax$ and solved for x using the fsolve function in Matlab. The Script 6.5 represents a function representing the $F = Ax$ equation, which returns vector F. Copy the function and save it as init_dfig_mpt.m. Once init_dfig_mpt function is ready, the fsolve function in Matlab can be used to solve the 15 equations representing the 15 SSCs. The solution of the SSCs is stored in the variable xdfig. e'_{sq} and e'_{sd} are calculated using (6.12) and (6.13)

```
function F = init_dfig_mpt(x,Vdfig,Pg,Qg, data_DF, FLTR)
% This function is used to initialize DFIG for maximum power point
%tracking
% region. The prefix 'd_' is not used in the variables inside the
%function
%for ease of reading.The variables defined inside function are not
%visible
%outside it.

% Get voltage, and power from Vdfig
vqs = real(Vdfig);
vds = imag(Vdfig);
ws = 1.0;      % synchronous speed

% Get machine parameters from data_DFIG matrix
Lm = data_DF(:,3);
Rs = data_DF(:,4); % stator resistance
Rr = data_DF(:,5); % rotor resistance
Lss = data_DF(:,6); % stator self inductance
Lrr = data_DF(:,7); % rotor inductance
kopt = data_DF(8);

% Get filter parameters from FLTR matrix
R1f = FLTR(1);       R2f = FLTR(2);       Rcf = FLTR(3);
L1f = FLTR(4);       L2f = FLTR(5);       Cf = FLTR(6);

% x is the solution vector. The commented section below shows index
%of
% various parameters
% x(1) = iqs, x(2) = ids, x(3) = iqr, x(4) = idr
% x(5) = vqr, x(6) = vdr
% x(7) = i1q, x(8) = i1d, x(9) = i2q, x(10) = i2d
% x(11) = v1q, x(12) = v1d, x(13) = vcq, x(14) = vcd  x(15) = wr

% set of Ax-b = 0 equations. Refer Section xx for details.
F = [-Rs*x(1)+ws*(Lss*x(2)+Lm*x(4))-vqs;   % SSC 1
    -Rs*x(2)-ws*(Lss*x(1)+Lm*x(3))-vds;    % SSC 2
    -Rr*x(3)+(ws-x(15))*(Lrr*x(4)+Lm*x(2))-x(5);  % SSC 3
    -Rr*x(4)-(ws-x(15))*(Lrr*x(3)+Lm*x(1))-x(6);  % SSC 4
    Lm*(x(1)*x(4)-x(2)*x(3))-kopt*x(15)^2;  % SSC 5
    vqs*x(1)+vds*x(2)+vqs*x(9)+vds*x(10)-Pg;  % SSC 6
    -vqs*x(2)+vds*x(1)-Qg;          % SSC 7
    -vqs*x(10)+vds*x(9);          % SSC 8
    x(5)*x(3)+x(6)*x(4)-x(11)*x(7)-x(12)*x(8);  % SSC 9
    x(7)*(R1f+Rcf) - x(8)*L1f - x(9)*Rcf + x(13) - x(11); % SSC 10
    x(8)*(R1f+Rcf) + x(7)*L1f - x(10)*Rcf + x(14) - x(12); % SSC 11
    -x(7)*Rcf + x(9)*(Rcf+R2f) - x(10)*L2f + vqs - x(13); % SSC 12
    -x(8)*Rcf + x(10)*(Rcf+R2f) + x(9)*L2f + vds - x(14); % SSC 13
    x(7) - x(9) + Cf*x(14); % SSC 14
    x(8)-x(10)-Cf*x(13)];   % SSC 15
```

Script 6.5 Function used to initialize doubly fed induction generator (DFIG) under subrated operating condition.

Step 4: Initialization of converter controllers and turbine. In this step, the initial values of converter controllers and turbine are calculated. At subrated condition, the wind speed is calculated using Eq. (6.1) by setting $\beta = 0$ and $\lambda = \lambda_{opt}$. We will discuss the method to obtain turbine initial condition for rated operating condition in the next section.

The programs required for the initial condition calculation can be summarized as follow:

(1). Make form_Ymatrix.m and power_flow.m as discussed in Chapter 2.
(2). Make dfig_smib_main.m by combining Scripts 6.1 and 6.3.
(3). Make find_dfig_state_initial_conditions.m from Script 6.4.
(4). Make init_dfig_mpt.m using the Script 6.5.

Ensure that all the codes are saved in one folder. To simulate different network or DFIG operating conditions, make changes in the bus or line matrices. For example, to change the output of DFIG, change bus (3,4) element. But do not make it unity, as we have not discussed the rated operating condition scenario.

6.5 Further modifications in DFIG-WTG model

So far, we have cut many corners to quickly build and run a DFIG simulation. Let us discuss couple of modifications and additions that can improve the model.

Simulating rated operating condition: The program init_dfig_mpt.m can initialize the DFIG to subrated operating condition. The program should be modified to work for rated operating condition. Modify SSC5 such that the torque is unity at rated operating condition and save it as init_dfig_cpt.m.

```
Lm*(x(1)*x(4)-x(2)*x(3))-1;   % SSC 5 for rated operating condition
```

Secondly, we need to find the wind speed and pitch angle at rated operating condition. At rated operating conditions, wind speed can take values between 15 and 25 m/s for the given turbine parameters. Once wind speed is selected, find λ using the turbine (generator) speed obtained from the generator initialization. Now write a function find_beta(Cp_req, d_Lambda) that will return β using Eq. (6.2).

MSC and GSC controllers: Adding feed forward path in the MSC and GSC controllers can enhance their performance. The structure of the controllers is available in many literatures (Rosyadi et al., 2014; Wu et al., 2007). What would be the initial condition of the MSC and GSC controller state variable when feed forward path is introduced?

Pitch angle controller: Add a pitch angle controller to simulate the rated operating condition. A block diagram for the pitch angle controller is shown in Fig. 6.24.

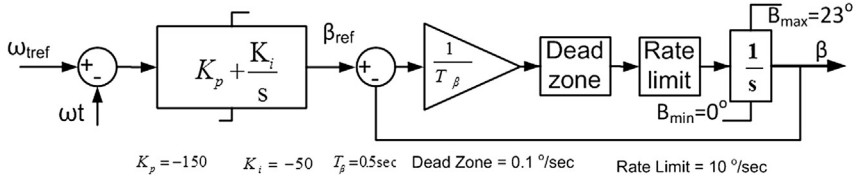

Figure 6.24 Block diagram representation of pitch angle controller.

6.6 Permanent magnet synchronous generator modelling

This section presents the steps to build a Simulink program for a PMSG-based WTG connected to an infinite bus as shown in Fig. 6.25. It is assumed that the readers have already built a DFIG-SMIB model as discussed in the previous section. We will reuse many of the blocks from the DFIG model.

The PMSG uses permanent magnets to generate rotor field. They may come with or without the gearbox. A gearless direct drive configuration is widely used for offshore applications. Increasing the number of poles of generator, the operating speed can be matched with turbine speed to avoid gearbox in the design. Note that the generator frequency and speed are independent of the grid frequency. The generator is fed from a back-to-back converter, which is designed to carry rated output power. It also contains a filter to remove switching frequency components in the current.

Like DFIG-SMIB model, the PMSG-SMIB model can be divided into seven components or sections: turbine, generator, MSC, B2BC, GSC, filter and network. Let us make the Simulink model of the system first.

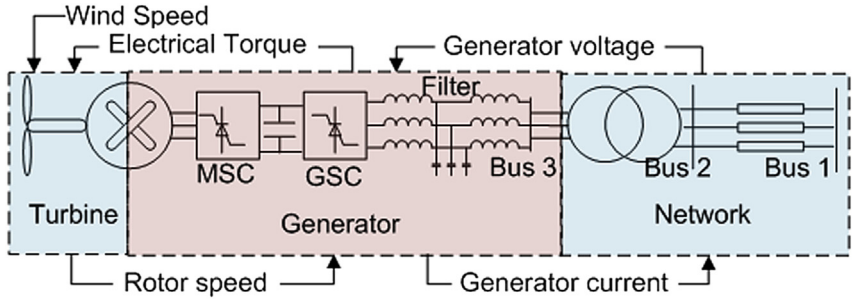

Figure 6.25 Schematic diagram of a single machine infinite bus (SMIB) system using permanent magnet synchronous generator (PMSG)—type wind turbine generator.

6.6.1 Turbine model

For continuity from DFIG model, we will assume that the turbine parameters are same for both DFIG-SMIB and PMSG-SMIB systems. This will allow us to reuse the wind turbine aerodynamic model discussed in Section 6.2.2.2. However, because of the gearless configuration, a one mass model will be used for the drive train model.

Let us start building the model from the turbine side of the system. For the aerodynamic model of turbine, use the blocks developed in Section 6.2a by changing the prefix of all parameters from 'd_' to 'p_'. A single-mass model of the drive train can be developed using the following equation:

$$\frac{d\omega_t}{dt} = \frac{1}{2H}(T_m - T_e)$$

where ω_t, H, T_m and T_e are turbine speed, inertia constant, mechanical torque and electrical torque. Note that the state variable in the one-mass model is p_wt. Now, build a Simulink block—representing turbine that takes wind speed and electrical torque as input and turbine speed as output.

6.6.2 Permanent magnet synchronous generator model

Eqs. (6.38–6.43) represent widely cited electrical model of PMSG in synchronous reference frame, where v, ψ, ω_{re}, R_a, L, i, ϕ_{pm} represent voltage, flux, speed in electrical radians/second, stator resistance, inductance, current and permanent magnet flux, respectively (Kim et al., 2010; Li et al., 2010).

$$v_d = p\psi_d - \psi_q\omega_{re} - R_a i_d \tag{6.38}$$

$$v_q = p\psi_q + \psi_d\omega_{re} - R_a i_q \tag{6.39}$$

where $\psi_d = -L_d i_d + \phi_{pm}$ and $\psi_q = -L_q i_q$.

Equation (6.38) and (6.39) can be rewritten as

$$L_d p i_d = -v_d + L_q i_q \omega_{re} - R_a i_d \tag{6.40}$$

$$L_q p i_q = -v_q - L_d i_d \omega_{re} + \phi_{pm}\omega_{re} - R_a i_q \tag{6.41}$$

The electrical torque generated is given by

$$T_e = -L_d i_d i_q + \phi_{pm} i_q + L_q i_q i_d \tag{6.42}$$

The active power output at the PMSG terminal is given by

$$P_{pmsg} = v_d i_d + v_q i_q \tag{6.43}$$

Now, build a generator model subsystem using Eqs. (6.40–6.43). The inputs to this model will be the voltages (v_q, v_d) obtained from MSC blocks and turbine speed (ω_t) obtained from turbine block. The outputs of generator block are i_d, i_q, T_e and P_{pmsg}.

6.6.3 Machine-side converter controller

Like the controllers in the DFIG model, the MSC controller in PMSG uses a decoupled control strategy, where q-axis regulates torque and d-axis regulates reactive power. The d-q axis is selected such that $v_q = |V|$ and $v_d = 0$. Let us assume the reactive power at the MSC terminal ($-v_q i_d + v_d i_q$) is zero. This is true when the d-axis current (i_d) is zero. Hence, the d-axis current reference to zero. The torque reference is obtained using the MPPT discussed in the DFIG-SMIB modelling. For $i_d = 0$, from (6.42), the electrical torque $T_e = k_{opt}\omega_t^2 = \phi_{pm} i_q$. Hence, reference value of $i_q = K_{opt}\omega_t^2 \big/ \phi_{pm}$. The Simulink model representation of the MSC controller is shown in Fig. 6.26. It has two PI controllers, MSC_IL1 and MSC_IL2, like the DFIG-SMIB model. The gains and initial state variable must be specified like the DFIG-SMIB model with 'p_' suffix. For example, MSC_IL1 has variables p_MSC_IL1_kp, p_MSC_IL1_ki and p_MSC_IL1_iv.

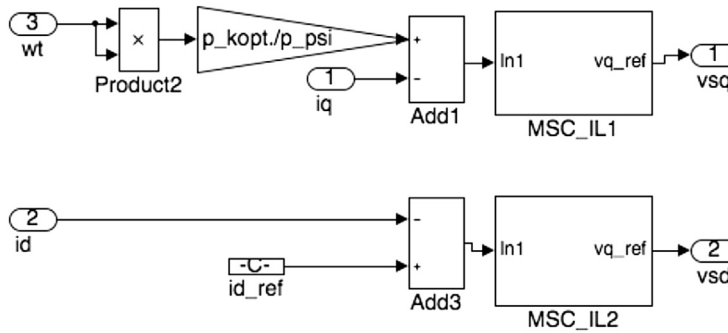

Figure 6.26 Simulink representation of permanent magnet synchronous generator (PMSG) machine-side converter (MSC).

Table 6.5 Parameters of LCL filter in permanent magnet synchronous generator (PMSG)—type wind turbine generator (WTG).

Parameter	Symbol (variable name)	Value (p.u.)
Inverter-side inductance	L_i	0.1667
Inverter-side resistance	R_i	0.0
Grid-side inductance	L_g	0.0033
Grid-side resistance	R_g	0.0
Filter capacitor	C_f	0.0150
Damping resistance	R_c	0.7333

6.6.4 Back-to-back capacitor, GSC controller, LCL filter and network

The model of B2BC, GSC and filter developed for DFIG can be directly copied to use in a PMSG simulation as the working principles of these components are same. Only (in fact the most important) change required in the Simulink blocks is in the prefix of name of parameters and state variables, which must be changed from 'd_' to 'p_'. In the DFIG, the filter connects GSC at the rotor circuit with the generator bus, which carry maximum of 30% of the rated power. However, in PMSG, it is designed to carry full rated output. For this reason, new set of parameters listed in Table 6.5 are used for the filter in PMSG.

Network model: In the DFIG model, the sum of stator current and filter current is fed to the network model. However, in PMSG_SMIB, the filter current alone is fed to the network model. The network block of DFIG-SMIB system can be copied to be adapted and used in the PMSG-SMIB system.

So far, we have discussed all the blocks required to build a PMSG-SIMB system simulation program. Integrate various blocks as shown in Fig. 6.27. Let us write a program to find initial values of the state variables. We will test the Simulink program later.

6.7 Initialization of PMSG-SMIB system

The program for initializing the system is given in Scripts 6.6 and 6.7. Create a new file pmsg_smib_main.m by combining Scripts 6.1 and 6.6.

Create a new .m file named find_pmsg_state_initial_conditions.m and copy the program in Script 6.7. The comments in the program explain the calculation steps.

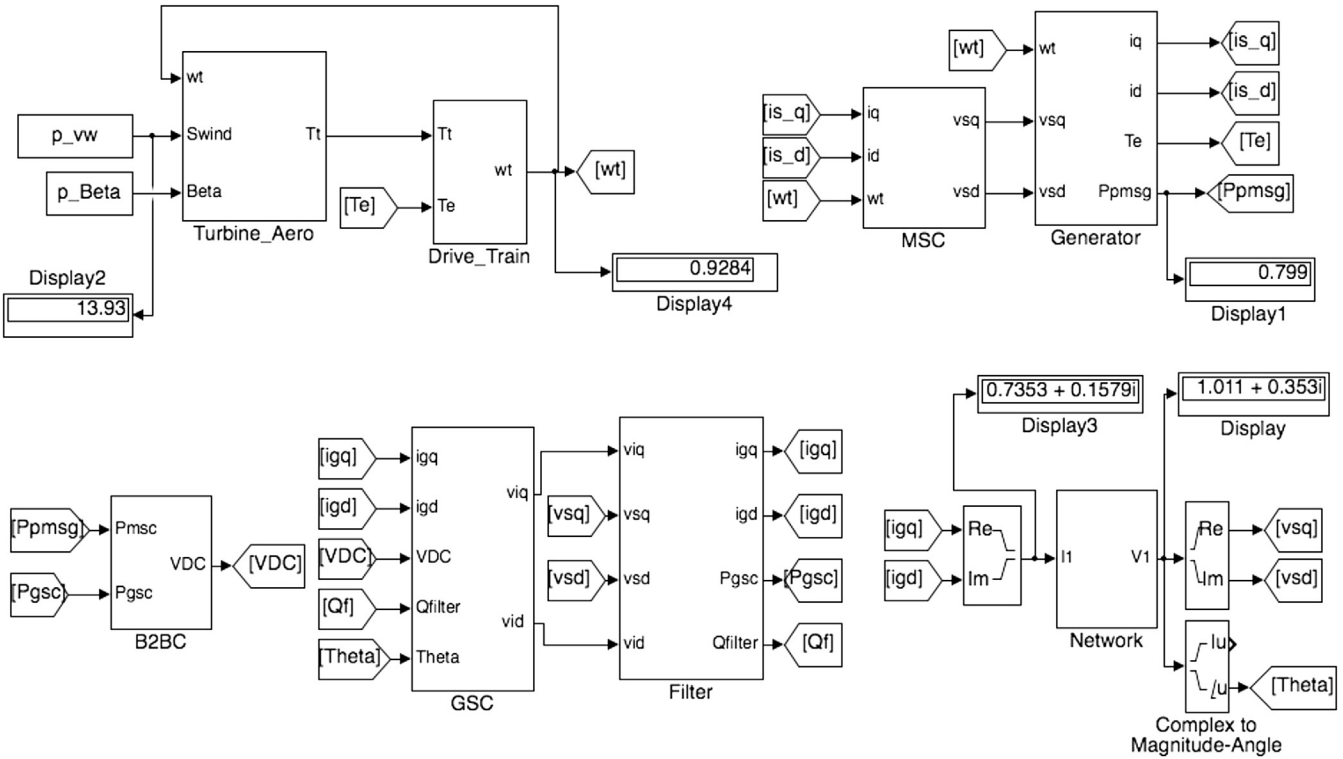

Figure 6.27 Simulink model of permanent magnet synchronous generator (PMSG)—single machine infinite bus (SMIB) system.

```
Pmachs = [3];      % Bus where PMSG is connected
Omega = 2*pi*50;
if size(Pmachs,1)
    find_pmsg_state_initial_conditions
end
```

Script 6.6 Program to call permanent magnet synchronous generator (PMSG) initialization.

6.8 Modal analysis and dynamic simulation results

The eigenvalues of the test system for the data given in Scripts 6.1 and 6.7 are listed in Table 6.6. Find eigenvalues of the Simulink model and compare with the values in the table. In case you are not getting the correct eigenvlaues, debug the Simulink model using the method discussed in Section 3.6. Let us run a dynamic simulation. Apply changes mentioned in Section 6.3. Fig. 6.28 shows dynamic simulation results by applying a step change in wind speed at time = 1 s and step change in infinite bus voltage at time = 10 s.

Finally, make a PMSG block like DFIG block in Fig. 6.23. To do this, select all blocks in Fig. 6.27 except the network block, click the right mouse button and select the 'create subsystem from selection' option. Change the name of new subsystem to PMSG. We will use this block for building a wind farm simulation program in the next section.

6.9 Simulation of wind farm having DFIG- and PMSG-type WTGs

So far, we have discussed modelling and simulation of a DFIG- and a PMSG-type WTG connected to an infinite bus. However, an actual wind farm contains tens or hundreds of WTGs connected to a medium voltage network called collector system. Depending on study requirement, we may like to model all the WTGs together or part of the system. This section will discuss how to use the model so far developed to build a simulation model for a big practical wind farm.

Let us take an example wind farm as shown in Fig. 6.29. It contains six WTGs, three of them (at buses 1, 2 and 3) are DFIG type and other (at buses 4, 5 and 6) three are PMSG type. There are 13 buses in the system. Consider Bus 13 as infinite bus. The program to initialize the model is given in Script 6.8. In the initial part of the program, bus and line matrices are specified, and load flow solution and admittance matrix are obtained. In the simulation programs discussed so far, one WTG is connected to

```
    %BEGINS: Parameter for PMSG
    p_ra = 0.0025; p_Lq = 0.7; p_Ld = 0.7;
    p_H = 2;
    p_psi = 1.222;
    p_kopt = 1;
    p_bl = 40.05;
    p_lambda_opt = 8.1;
    p_wt_rated = 3.0337;
    p_base = 5;

    % Filter parameters
    p_FLTR =  [0.000, 0.000, 0.7333, 0.1667, 0.0033, 0.0150];
    p_Cdc = 0.3; % Converter capacitor

    p_Ri= 0.0;
    p_Rg= 0.0;
    p_Rc= p_FLTR(3);
    p_Li= p_FLTR(4);
    p_Lg= p_FLTR(5);
    p_Cf = p_FLTR(6);
    %

    % BEGINS: LCL Filter initialization

    % Output of the WTG at the generator bus
    Spmsg = bus_sln(Pmachs,4)+1i*bus_sln(Pmachs,5);
    % Voltage at the PMSG bus
    vpmsg = bus_sln(Pmachs,2).*exp(1i*bus_sln(Pmachs,3)*pi/180);
    % output current of WTG or current through LCL filter grid side
%inductance
    ipmsg = conj(Spmsg./vpmsg);
    % Angle of voltage at the generator bus
    p_Theta = angle(vpmsg);

    % Voltage across R and C in the filter
    vnode = vpmsg+ipmsg.*(p_Rg+1i*p_Lg);
    % Current through the capacitor
    icf = vnode./(p_Rc-1i/(p_Cf));
    % Voltage across capacitor
    vcf = icf.*(-1i/(p_Cf));
    % Current through inverter side inductor
    iLi = ipmsg+icf;
    % Voltage at the inverter side of filter/ voltage output of GSC
    vLi = vnode+iLi*(p_Ri+1i*p_Li);

    % State variables in the program
    p_iiq = real(iLi);   p_iid = imag(iLi);
    p_igq = real(ipmsg); p_igd = imag(ipmsg);
    p_vcq = real(vcf);   p_vcd = imag(vcf);
    p_viq = real(vLi);   p_vid = imag(vLi);


    % BEGINS: Initialisation of grid side converter

    % Parameters for GSC controller model

    % Current and voltage in new reference inside controller
    p_vi_dash = (p_viq+1i*p_vid).*exp(-1i*p_Theta);
    p_ii_dash =  (p_igq+1i*p_igd).*exp(-1i*p_Theta);

p_GSC_IL1_iv = real(p_vi_dash); p_GSC_IL1_kp = 0.3; p_GSC_IL1_ki= 200;
```

Script 6.7 Program for find_pmsg_state_initial_conditions.m.

```
p_GSC_IL2_iv = imag(p_vi_dash); p_GSC_IL2_kp = 0.3; p_GSC_IL2_ki = 200;
p_GSC_OL1_iv = real(p_ii_dash); p_GSC_OL1_kp = -22; p_GSC_OL1_ki = -870;
p_GSC_OL2_iv = imag(p_ii_dash); p_GSC_OL2_kp = 0;   p_GSC_OL2_ki = -60;


    % Reactive power reference GSC controller
    p_Qfilter = imag(Spmsg);

    % B2B capacitor
    p_VDC = 1.5;

    % Initialization of generator states and turbine
    % Assuming no loss in the converter, generator output is equal to
%GSC
    % output
    Pg = real(vLi.*conj(iLi));

    % wind speed
    p_vw = (Pg*p_base*1e6./(0.5*1.225*pi*p_bl^2*0.48)).^(1/3);
    % turbine speed
    p_wt = p_Sw*p_lambda_opt/(p_bl*p_wt_rated);
    p_beta = 0;
    % assuming d-axis current is equal to zero, q-axis current and
%voltage
    % are
    p_isq = Pg./(p_wt.*p_psi);
    p_vsq = Pg./p_isq;

    % d-axis current and voltage are
    p_isd = 0;
    p_vsd = p_Lq.*p_isq.*p_wt - p_ra*p_isd;


    % Initialisation of machine side converter
    p_MSC_IL1_kp = -90;
    p_MSC_IL1_ki = -1000;
    p_MSC_IL1_iv = p_vsq;

    p_MSC_IL2_kp = -90;
    p_MSC_IL2_ki = -1000;
    p_MSC_IL2_iv = p_vsd;
```

Script 6.7 (*continued*).

Table 6.6

Eigenvalues	Frequency (Hz)	Damping ratio (%)
$-58837 \pm 62318i$	9918	68.65
$-16236 \pm 17285i$	2751	68.46
$-517.78 \pm 805.68i$	128	54.06
$-50.23 \pm 386.62i$	61.5	12.88
-64.27	0	100.0
$-23.65 \pm 35.28i$	5.61	55.68
$-378.37 \pm 2.90i$	0.46	99.99
-0.69	0	100.0
$-11.45 \pm 0.09i$	0.014	99.99

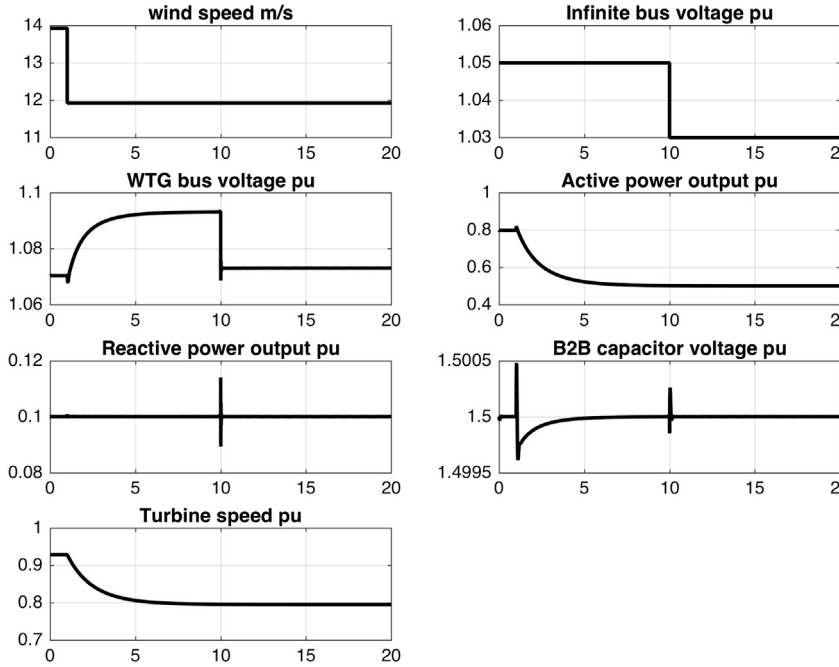

Figure 6.28 Dynamic simulation results of permanent magnet synchronous generator (PMSG)—single machine infinite bus (SMIB) system.

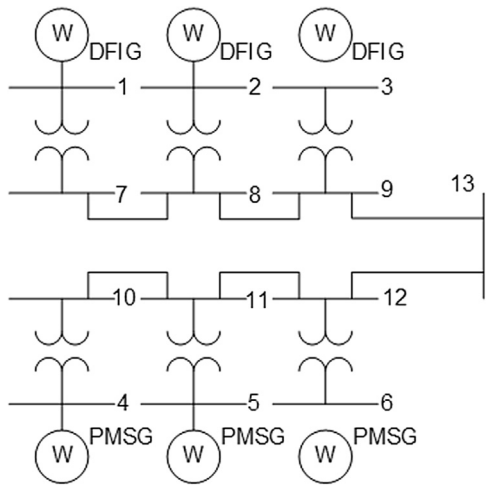

Figure 6.29 Schematic diagram of an example wind farm.

```
clear all

bus = [...
    01 1.00  0.00  0.80  0.26  0.00   0.00  0.00  0.00  3;
    02 1.00  0.00  0.95  0.31  0.00   0.00  0.00  0.00  3;
    03 1.00  0.00  0.90  0.29  0.00   0.00  0.00  0.00  3;
    04 1.00  0.00  0.85  0.28  0.00   0.00  0.00  0.00  3;
    05 1.00  0.00  0.90  0.29  0.00   0.00  0.00  0.00  3;
    06 1.00  0.00  0.95  0.21  0.00   0.00  0.00  0.00  3;
    07 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    08 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    09 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    10 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    11 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    12 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  3;
    13 1.00  0.00  0.00  0.00  0.00   0.00  0.00  0.00  1];

line = [
    01  07 0.010  0.10  0.000 0.0 0.0;
    02  08 0.010  0.10  0.000 0.0 0.0;
    03  09 0.010  0.10  0.000 0.0 0.0;
    04  10 0.010  0.10  0.000 0.0 0.0;
    05  11 0.010  0.10  0.000 0.0 0.0;
    06  12 0.010  0.10  0.000 0.0 0.0;
    07  08 0.0048 0.080 0.0010 0.0 0.0;
    08  09 0.0048 0.080 0.0010 0.0 0.0;
    09  13 0.0048 0.080 0.0010 0.0 0.0;
    10  11 0.0048 0.080 0.0010 0.0 0.0;
    11  12 0.0048 0.080 0.0010 0.0 0.0;
    12  13 0.0048 0.080 0.0010 0.0 0.0];

Y = form_Ymatrix(bus,line);
[bus_sln, flow] = power_flow(Y, bus, line);
Znet = inv(Y);
vinf = bus_sln(13,2)*exp(1i*bus_sln(13,3)*pi/180);
ZA = Znet(1:end-1,1:end-1); ZB = Znet(1:end-1,end);
ZC = Znet(end,1:end-1); ZD = Znet(end,end);  % Znet = [Za Zb; Zc Zd]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Pmachs = [1,2,3]';      % Buses where PMSG is connected
Dmachs = [4,5,6]';       % Buses where DFIG is connected
Omega = 2*pi*50;
if size(Pmachs,1)

    find_pmsg_state_initial_conditions
    pmsg_mult = zeros(size(bus_sln,1)-1,size(Pmachs,1));
    pmsg_mult(Pmachs,:) = eye(size(Pmachs,1));
end


if size(Dmachs,1)

    find_dfig_state_initial_conditions
    dfig_mult = zeros(size(bus_sln,1)-1,size(Dmachs,1));
    dfig_mult(Dmachs,:) = eye(size(Dmachs,1));
end
```

Script 6.8 Program to initialize wind farm model states.

an infinite bus through a network. However, in this system, six WTGs are connected to an infinite bus through a network. Let us discuss how to represent the network in this case.

6.9.1 Network representation

Let V_b, Z_{net} and I_b be vector (13 × 1) of bus voltages, impedance matrix (13 × 13) and vector (13 × 1) of current injection at the buses, respectively. Then,

$$V_b = Z_{net} I_b \tag{6.44}$$

$$\begin{bmatrix} V_1 \\ . \\ . \\ V_{12} \\ V_{13} \end{bmatrix} = \begin{bmatrix} & & & & \\ Z_{1,1} & \cdot\cdot & Z_{1,12} & Z_{1,13} \\ \cdot & \cdot\cdot & \cdot & \cdot \\ \cdot & \cdot\cdot & \cdot & \cdot \\ Z_{12,1} & \cdot\cdot & Z_{12,12} & Z_{12,13} \\ Z_{13,1} & \cdot\cdot & Z_{13,12} & Z_{13,13} \\ & & & & \end{bmatrix} \begin{bmatrix} I_1 \\ . \\ . \\ I_{12} \\ I_{13} \end{bmatrix} \tag{6.45}$$

The current injection at buses 1−6 can be obtained from WTG models. There is no current injection at buses 7−12. As Bus 13 is an infinite bus, we do not know the current injection (I_{13}) at Bus 13, but we know the voltage (V_{13}). Let us rewrite the equation as

$$\begin{bmatrix} V_b \\ V_{13} \end{bmatrix} = \begin{bmatrix} Z_A & Z_B \\ Z_C & Z_D \end{bmatrix} \begin{bmatrix} I_b \\ I_{13} \end{bmatrix} \tag{6.46}$$

From (6.46),

$$I_{13} = Z_D^{-1}(V_{13} - Z_C I_b) \tag{6.47}$$

Using (6.47), we can write

$$V_b = \left(Z_A - Z_B Z_D^{-1} Z_C\right) I_b + Z_B Z_D^{-1} V_{13} \tag{6.48}$$

In the program, the matrix Znet is divided into four sub matrices: ZA, ZB, ZC and ZD. Simulink representation of the network using Eq. (6.48) is shown in Fig. 6.30.

6.9.2 Wind farm simulink model

The Simulink model for the wind farm simulation is shown in Fig. 6.31. Now develop the wind_farm_init as given in Script 6.8 and wind_farm_model

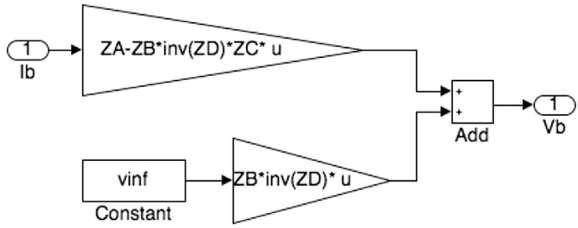

Figure 6.30 Simulink representation of the wind farm network.

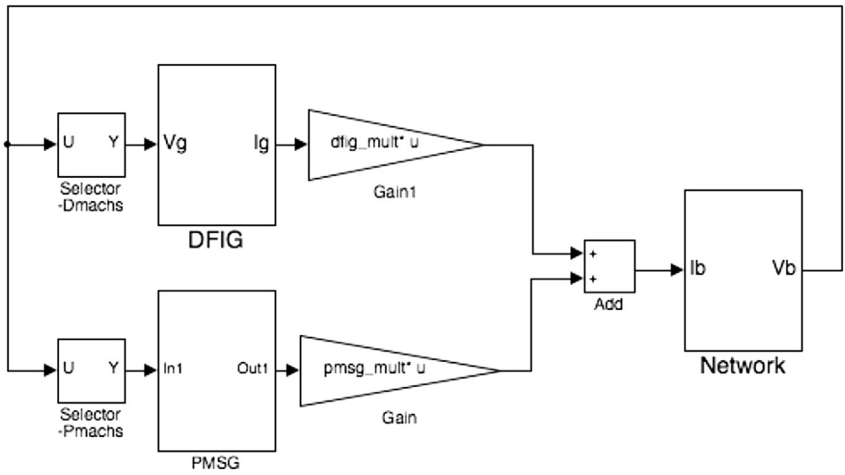

Figure 6.31 Simulink representation of the wind farm.

as shown in Fig. 6.31. The output of the network block is vector of size 12 representing bus voltages at buses 1−12. The selector elements from the 'Signal Routing' library of the Simulink are used to select appropriate voltages for the DFIG and PMSG blocks. The variables Dmachs and Pmachs are used to specify the bus numbers. Make changes in the block parameter dialogue of the selector elements: index option: Index vector (dialogue), Index: Dmachs or Pmachs and Input port size: 12.

Now the output of DFIG and PMSG blocks are current vector of size equal to the number of DFIG and PMSG in the network. We need to convert them into a vector of 12 elements representing current injections at buses 1−12. The variables dfig_mult and pmsg_mult are used to modify the current vector. In the 'block parameter' dialogue of the dfig_mult and pmsg_mult, select Matrix(K*u) for the multiplication option. A

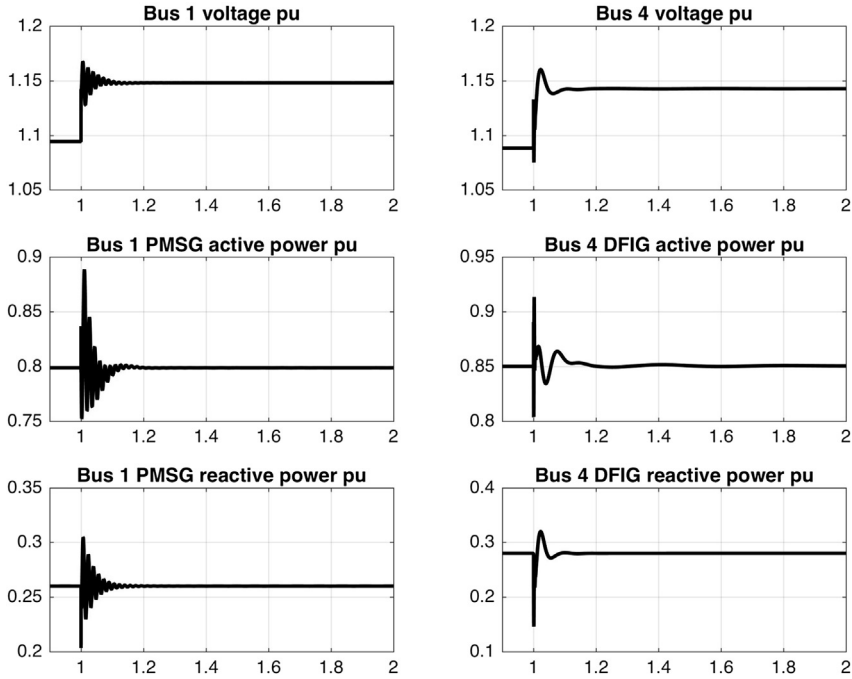

Figure 6.32 Wind farm time domain simulation results.

response to step change in the infinite bus voltage from 1.0 pu to 1.05 pu at time = 1sec is shown in Fig. 6.32 for comparison.

References

Abad, G., Lopez, J., Rodriguez, M., Marroyo, L., Iwanski, G., December 2011. Doubly Fed Induction Machine: Modeling and Control for Wind Energy Generation. Wiley-IEEE Press.

Araujo, S.V., Engler, A., Sahan, B., Antunes, F.L.M., 2007. LCL filter design for grid-connected NPC inverters in offshore wind turbines. In: Proc. 7th Int. Conf. Power Electron., Daegu, pp. 1133–1138.

Kim, H.-W., Kim, S.-S., Ko, H.-S., 2010. Modeling and control of PMSG-based variable-speed wind turbine. Electr. Power Syst. Res. 80 (1), 46–52 (PMSG model reference).

Li, S., Haskew, T.A., Xu, L., 2010. Conventional and novel control designs for direct driven PMSG wind turbines. Electr. Power Syst. Res. 80 (3), 328–338.

Mei, F., 2008. Small-Signal Modelling and Analysis of Doubly-Fed Induction Generators in Wind Power Applications. PhD thesis. Imperial College, London, United Kingdom.

Rosyadi, M., Muyeen, S.M., Takahashi, R., Tamura, J., 2012. New controller design for PMSG based wind generator with LCL-filter considered. In: Proc. XXth Int. Conf. Elect. Mach. Marseille, pp. 2112–2118.

Rosyadi, M., Umemura, A., Takahashi, R., Tamura, J., Uchiyama, N., Ide, K., 2014. A new simple model of wind turbine driven Doubly-Fed Induction Generator for dynamic analysis of grid connected large scale wind farm. In: 3rd Renewable Power Generation Conference (RPG 2014), Naples, pp. 1–6 [Open Access].

Wu, B., Lang, Y., Zargari, N., Kouro, S., August 2011. Power Conversion and Control of Wind Energy Systems. Wiley-IEEE Press.

Wu, F., Zhang, X.P., Godfrey, K., Ju, P., September 2007. Small signal stability analysis and optimal control of a wind turbine with doubly fed induction generator. IET Gen. Trans. Distrib. 1 (5), 751–760.