Laboratory Four - Advanced Control Systems

Objectives

The objective of this simulation study is to learn how to use MATLAB/Simulink SimPowerSystem Toolboxes for building physical model of a permanent magnet synchronous machine (PMSM) electrical system in order to design and implement the cascaded PI speed control in the following lab. The simulation skills acquired in this lab can be extended to simulate power converter control, induction motor control and other electrical systems using similar approaches.

Lab Requirements

1. A maximum of three students are allowed per group. The groups must be fixed between Labs 4−5 and all group members must attend the same lab session. No exceptions will be made to these rules.
2. Due to the work required to complete the lab, students who do not attend a lab with their group will not receive a mark for the report. Remember to sign the attendance sheet.
3. Labs 4−5 will be combined into a single report worth 15% of your ACS grade. Please submit the report on Canvas. Only one submission is required per group.
4. Please ensure that the content included in your report is your own intellectual property. RMIT's policies on Academic Integrity will be observed while marking.
5. Please make sure to save your files from this lab as they will be required for completing Lab 5.

0.1 Building Embedded Functions for Park-Clarke Transformation

The main reason to write these transformations as embedded functions is to avoid mistakes and confusions. These embedded functions are written based on the mathematical equations given in the book entitled 'PID and Predictive Control of Electrical Drives using MATLAB/Simulink'(soft copy free to download from RMIT library). To ensure consistency and clarity, two embedded functions are produced for Park-Clarke transformation and inverse Park-Clarke transformation.

0.1.1 Park-Clarke Transformation for Current Measurements

We assume that the three phase currents i_a, i_b and i_c are balanced so that

$$i_a(t) + i_b(t) + i_c(t) = 0$$

With this assumption, only two current measurements are required. Supposing that the phase A current and phase C current are measured using current sensors to obtain $i_a(t)$ and $i_c(t)$, then the phase B current is calculated using

$$i_b(t) = -i_a(t) - i_c(t)$$

Then from the Clarke transformation, the currents in the $\alpha - \beta$ reference frame are calculated using the relationship below:

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}$$

which results in

$$\begin{aligned} i_\alpha(t) &= i_a \\ i_\beta(t) &= -\frac{\sqrt{3}}{3} i_a - \frac{2\sqrt{3}}{3} i_c \end{aligned}$$

With the following Park transformation, the currents in the $d - q$ reference frame are calculated:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}$$

Tutorial 1 *The objective of this tutorial is to produce a MATLAB embedded function that will be used to convert phase A and phase C current measurements to i_d and i_q currents for the control systems designed in the $d-q$ synchronous reference frame.*

Step by Step

1. *Create a new Simulink file and save it with the name PCCurrent.slx.*
2. *In Simulink's Library Browser, find the block of 'MATLAB function' (with MATLAB icon) and copy it to the PCCurrent.slx model.*
3. *Double-click on the icon of the embedded function, and remove any default instructions. Type the following instructions to define the input and output variables to the model:*

```
function [id,iq]= PCCurrent(ia,ic,theta)
```

4. *Right-click on the embedded function block, and select 'Explore' among the options. This will open a new window. Select 'Inherited' for the 'update method' and tick the boxes for 'Support variable-size arrays', 'Saturate on integer overflow' and 'Allow direct feedthrough'. Select 'Fixed point' for 'Treat these inherited Simulink signal types as fi objects'. Click 'Apply' to save the changes and close this window.*
5. *Again double-click on MATLAB function block and define the initial condition for $\boldsymbol{\theta}$. Enter the following program into the file:*

```
if isempty(theta)
    theta=0;
end
```

6. *Perform Park-Clarke transform to convert the $\boldsymbol{i_a}$ and $\boldsymbol{i_c}$ current measurements to $\boldsymbol{i_\alpha}$ and $\boldsymbol{i_\beta}$ and then to $\boldsymbol{i_d}$ and $\boldsymbol{i_q}$ currents. Enter the following instructions into the MATLAB function:*

```
id=cos(theta)*ia+sin(theta)*(-sqrt(3)/3*ia-2*sqrt(3)/3*ic);
iq=-sin(theta)*ia+cos(theta)*(-sqrt(3)/3*ia-2*sqrt(3)/3*ic);
end
```

7. *Save the block diagram by clicking the 'Save' icon.*

0.1.2 Inverse Park-Clarke Transformation for Voltage Actuation

To convert the control signals v_d and v_q in the synchronous reference frame to three phase voltage v_a, v_b and v_c going to the inverter, an embedded MATLAB function is needed.

In the transformation, the following relationships are used.

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_d \\ v_q \end{bmatrix}$$

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \\ v_0 \end{bmatrix}$$

where $v_0 = 0$.

Tutorial 2 *The objective of this tutorial is to create a MATLAB embedded function that will convert control signals v_d and v_q to three phase voltage signals for the IGBT inverter.*

Step by Step

1. *Create a new Simulink file called IPCVoltage.slx*
2. *In Simulink's Library Browser, find the block of 'MATLAB function' (with MATLAB icon) and copy it to the IPCVoltage.slx model.*
3. *Double-click on the icon of the embedded function, and remove any default instructions. Type the following instructions to define the input and output variables to the model:*

```
function Vabc= IPCVoltage(vd,vq,theta)
```

4. *Right-click on the embedded function block, and select 'Explore' among the options. This will open a new window. Select 'Inherited' for the 'update method' and tick the boxes for 'Support variable-size arrays', 'Saturate on integer overflow' and 'Allow direct feedthrough'. Select 'Fixed point' for 'Treat these inherited Simulink signal types as fi objects'. Click 'Apply' to save the changes and close this window.*
5. *Again double-click on MATLAB function block and define the initial condition for θ. Enter the following program into the file:*

```
if isempty(theta)
    theta=0;
end
```

6. *Calculate the three phase voltage signals. Enter the following program into the file:*

```
T=[1 0;-1/2 sqrt(3)/2;-1/2 -sqrt(3)/2]...
*[cos(theta) -sin(theta);sin(theta) cos(theta)];
Vabc=T*[vd;vq];
end
```

7. *Save the block diagram by clicking the 'Save' icon.*

0.2 Building Simulation Model for PMSM

Tutorial 3 *The objective of this tutorial is to produce a Simulink program to simulate the dynamics of a PMSM.*
Step by Step

1. *Create a new Simulink model and save it with the name PMSMModel.slx.*
2. *To begin building the physical model based simulator, we need to define the simulation environment for the PMSM. Perform the following tasks:*
   a. *In the Simulink 'Library Browser', find 'powergui' and copy it to PMSMModel.slx.*
   b. *Double-click on 'powergui'; select 'Simulation type' as 'Discrete' and for 'Sample time', enter* `Tcs` *which is the sampling interval for power electronic model.*
3. *Once the simulation environment is defined, a physical model is to be built for the PMSM machine. In this step, we will choose the machine type and define the machine parameters according to name-plate data which is given at the end of this tutorial. Perform the following tasks:*
   a. *Open 'Library Browser' and from the left menu navigate to following path: Simscape→Electrical→Specialized Power Systems→Fundamental Blocks→Machines. Here, on the right hand side, select 'Permanent Magnet Synchronous Machine' and copy it to PMSMModel.slx.*
   b. *Double-click on 'Permanent Magnet Synchronous Machine'. In 'Configuration' tab, choose the following: Number of phases: 3, Back EMF waveform: 'Sinusoidal', Rotor type: Round, Mechanical input: Torque Tm, Preset model: No.*
   c. *Under the 'Parameters' tab, type the following: Stator phase resistance:* `Rs`*, Armature inductance:* `Ld`*. For 'Specify', choose 'Flux linkage established by magnets', Flux linkage:* `Phi_mg`*, and Inertia, viscous damping, pole pairs, static friction:* `[Jm 0 2 Bv]`*. Figure 0.1 shows the settings of all physical parameters needed in the item 'Parameters'.*
   d. *Get a 'Constant' block from Library Browser and connect it to 'Tm' terminal of PMSM. Enter the value of the 'Constant' block as* `TL`*.*
4. *Build the power electronics devices for operating the PMSM. This requires a three-phase IGBT inverter. Perform the following tasks:*
   a. *In the 'Library Browser', type 'Universal Bridge' and copy it to PMSMModel.slx Simulink model.*
   b. *Double-click on 'Universal Bridge'. Select Number of bridge arms as 3 and Power electronic device as IGBT/Diodes, and leave others as default. Close this window.*
   c. *Right-click on 'Universal Bridge' block and select 'Rotate & Flip' and flip the block 'left-right'.*

5. *Build the DC power supply model. Perform the following tasks:*
   a. *Open 'Library Browser' and from the left menu navigate to following path: Simscape→Electrical→Specialized Power Systems→Fundamental Blocks→Electrical Sources. Here, on the right hand side, find 'DC voltage source', move it to PMSMModel.slx and position it to the left of Universal Bridge. Make another copy of the 'DC voltage source'.*
   b. *Double-click and set the amplitude to* `Vdc/2` *for each voltage source. Connect them in series and then connect '+' terminal from the first DC voltage source with '+' terminal in Universal Bridge and connect the '-' terminal of the second DC voltage source to the '-' terminal of the Universal Bridge. Refer to Fig. 0.2 for correct connections.*
6. *Build current measurements. Perform the following tasks:*
   a. *In the Library Browser, find 'Current Measurement' block and copy this to PMSMModel.slx and position it between the 'Universal Bridge' and the PMSM block. Make another copy of 'Current Measurement' block, and label them 'Ia' and 'Ic', respectively.*
   b. *Connect terminal 'A' from Universal Bridge to the terminal '+' in the 'Ia' 'Current Measurement' block, and connect the terminal '-' of 'Ia' to the terminal 'A' of the PMSM block. Repeat this process with terminal 'C' for the 'Ic' 'Current Measurement' block.*
   c. *Connect the terminal 'B' from Universal Bridge to the terminal 'B' in PMSM without current measurement.*
7. *To transform the phase A and phase C current measurements into d&q axis currents, the 'PCCurrent.slx' file produced in the Tutorial 1 is used. Perform the following tasks:*
   a. *Open PCCurrent.slx and copy the block of 'PCCurrent' into PMSMModel.slx and position it close to the current measurements.*
   b. *In the Library Browser, find 'Zero-order hold' and place it on PMSMModel.slx model. Type* `Tcs` *into its sample time and make a copy of this block.*
   c. *Connect the output of current measurement 'Ia' to the input of one 'Zero-order hold' block and connect the output of this 'Zero-order hold' block to 'ia' of the PCCurrent block. Repeat this process for 'Ic' using the other 'Zero-order hold' block.*
8. *Build the Pulse-Width-Modulators. Perform the following tasks:*
   a. *Type '2-level' in the Library Browser and select the block 'PWM Generator (2-level)'. Copy it to PMSMModel.slx.*
   b. *Double-click and set the Generator type to 'three-phase bridge (6 pulses)', Frequency (Hz) to* `Fc`*, and Sample time to* `Tcs` *which is the same as sampling interval for the power electronics.*

c. *Get a 'Unit delay' block from Library Browser and connect it before 'PWM Generator (2-level)'. This is done in order to avoid the singularity problem in simulation.*
d. *Connect a 'Gain' block before the 'Unit delay' to normalize the stator voltage to give modulation index between -1 to 1. For this, select the gain of the 'gain' block as* `1/Vdc*sqrt(3)`.
e. *Connect the output of the 'PWM Generator (2-level)' to the input of the 'Universal Bridge' at terminal 'g' (for gate signal).*

9. *The 'PWM Generator (2-level)' accepts a normalized three phase voltage vector,* $[v_a, v_b, v_c]^T$ *as input. As the control and modeling are done in d&q axis, the* v_d *and* v_q *voltage need to be converted into* v_a, v_b *and* v_c. *For this, we will use the 'IPCVoltage.slx' file, generated in Tutorial 2. Perform the following tasks:*

   a. *Open IPCVoltage.slx and copy 'IPCVoltage' block and paste into PMSM-Model.slx.*
   b. *Connect the output of 'IPCVoltage' to the input of the 'Gain' block.*
   c. *In the Library Browser, find a 'Constant' block and connect it to* v_d *input of 'IPCVoltage'. Make a copy of the 'Constant' block and connect it to the* v_q *input.*

10. *Build ground. Open 'Library Browser' and from the left menu navigate to following path: Simscape→Electrical→Specialized Power Systems→Fundamental Blocks→Elements. Here, find 'Ground' and move it to PMSMModel.slx. Connect 'Ground' to the middle point of two DC voltage sources.*
11. *Send outputs to workspace from Simulink. Perform the following tasks:*

   a. *In the Library Browser, find 'Bus selector' and move it to PMSM-Model.slx.*
   b. *Connect the input of the 'Bus selector' to the output of 'Permanent Magnet Synchronous Machine' block at terminal 'm'.*
   c. *Double-click on 'Bus selector'. In the right table, remove the pre-selected options by selecting all of them and clicking 'Remove'. Then in the left table, click on 'Rotor speed wm' and then click 'Select>>'. Repeat the process for 'Rotor angle thetam'. These are the mechanical speed and mechanical angle of the motor, respectively.*
   d. *Find and place a 'Zero-order hold' block between the 'Bus selector' and 'PMSM' to specify the sampling rate. The sampling time should be equal to* `Tcs`.

12. *Convert mechanical speed and angle into electrical speed* $\omega_e = p\omega_m$ *and electrical angle* $\theta_e = p\theta_m$, *repsectively. Perform the following steps:*

   a. *Connect a 'Gain' block to the output 1 of the 'Bus selector' with its gain selected as* `p` *in order to produce electrical speed. Do the same for output 2 of the 'Bus selector' to produce electrical angle.*

b. *Connect the output of the electrical angle 'Gain' block to the input 'theta' of the PCCurent and IPCVoltage blocks.*
c. *Find 'Mux' from Library Browser and choose its number of inputs as 3 in order to put together three outputs i_d, i_q and w_e in a vector.*
d. *Find 'To workspace' from Library Browser and place it on the output of the 'Mux'. Choose the 'Save format' as 'Array'.*
e. *Find 'Scope' block from Library Browser and make three copies. Connect these scopes to i_d, i_q and w_e for display.*

In the MATLAB console window, type 'edit' and enter following information (name-plate data) into a new m-file. Save this file with a suitable name such as 'Lab4Parameters.m'.

*Rs=2.98; Ld=7e-3; Lq=7e-3; Phi_mg=0.125; p=2; Kt=1.5*p*Phi_mg; Jm=0.47e-4; Bv=11e-5; TL=0.2; Vdc=100;.*

The following parameters are suggested for the numerical computations: Tcs=1e-6; Fc=2e3;

Your physical model based simulation program is tested by choosing reference signals $v_d = 0$ and $v_q = 45$ as inputs to IPCVoltage block, and selecting a simulation time of 0.2 seconds. These inputs should produce an angular speed of $w_e \approx 300 rad/s$.

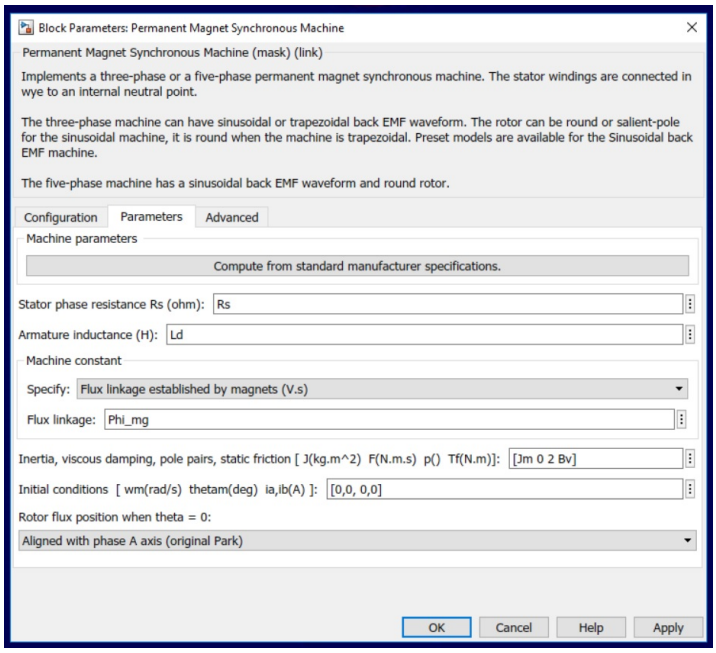

Fig. 0.1 Parameters definition in Simulink Model for PMSM

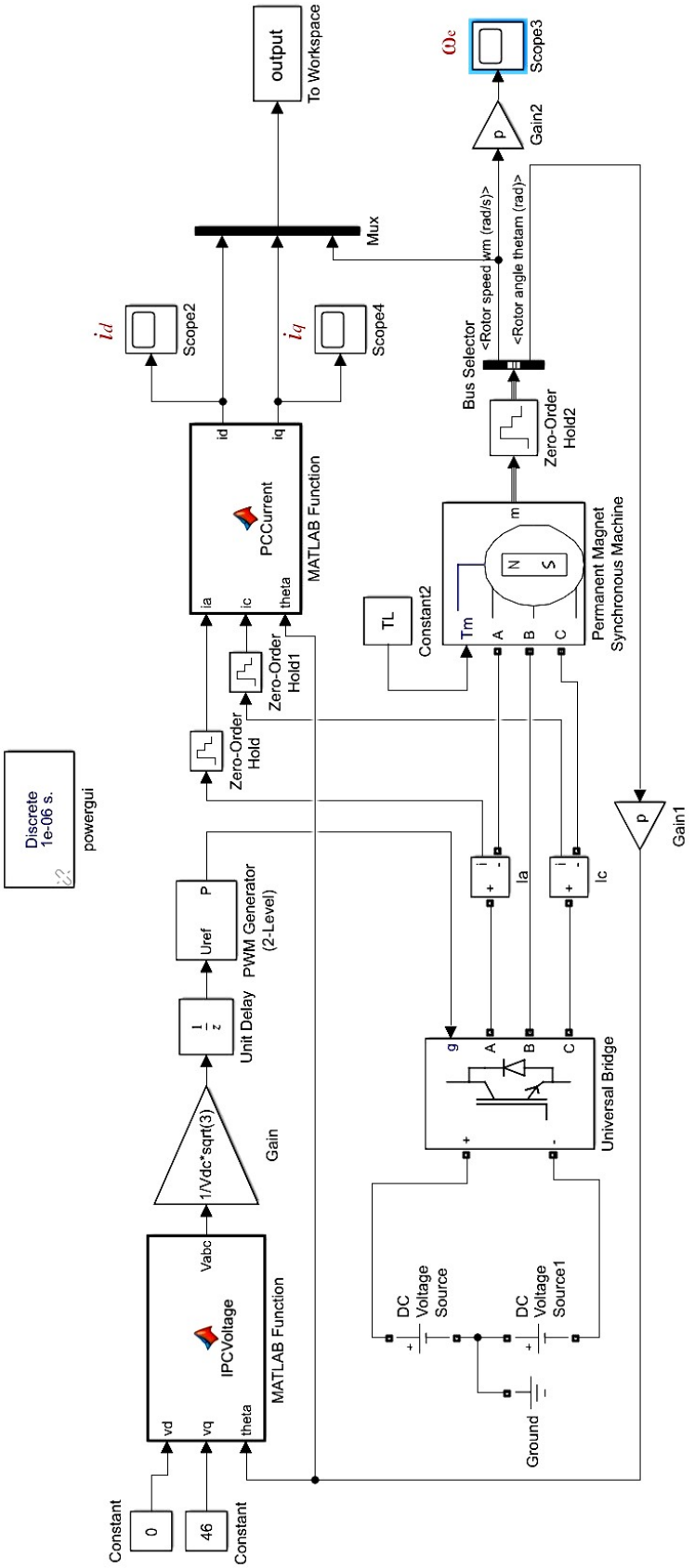

Fig. 0.2 Simulink diagram for simulation of PMSM