UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Master degree in Industrial Engineering

Master final thesis

Study of the PMSM modelisation for Hardware-in-the-Loop simulation applications

Author:

Marsà Fargas, Jordi

Tutor:

García Espinosa, Antoni

Advisor:

Colls Castro, Carles

Motion Control and Industrial Applications (MCIA)

School of Industrial, Aerospace and Audiovisual Engineering of Terrassa (ESEIAAT)

Autumn 2020

Volume content:

REPORT ATTACHMENT

Table of contents

List of figures

List of tables

1 Simulink models

In this chapter, all the Simulink models are depicted for a detailed view. The first section shows the whole model with all the subsystems. The second one contains the template models for generating real-time applications.

1.1 Whole system model

This is the model in the file named 'SPMSM2018b.slx'.

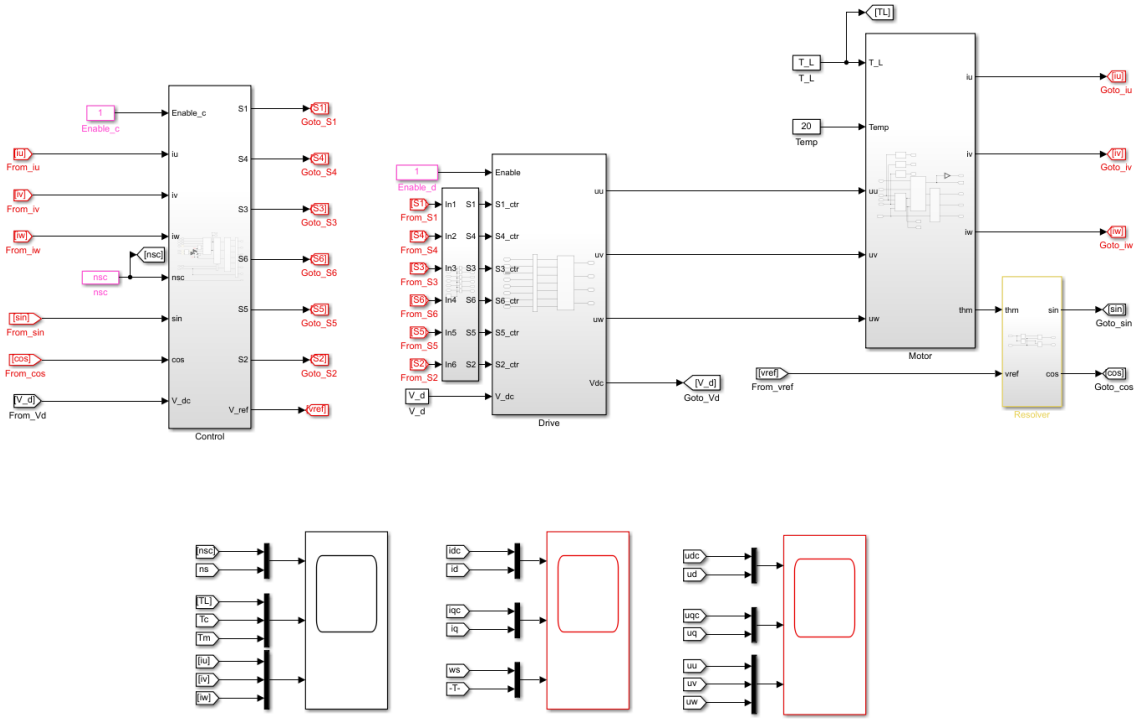

Figure 1.1 Whole system model.

The scopes have been configured as follows. The 'Logging' tab can be utilised for storing the data as 'Structure With Time' in MATLAB workspace after a Simulation. This is used for comparing the results of Simulink and real-time simulations.

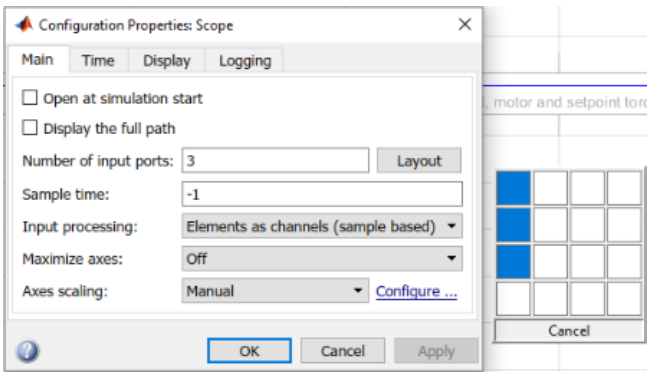

Figure 1.2 Scope configuration properties dialogue box.

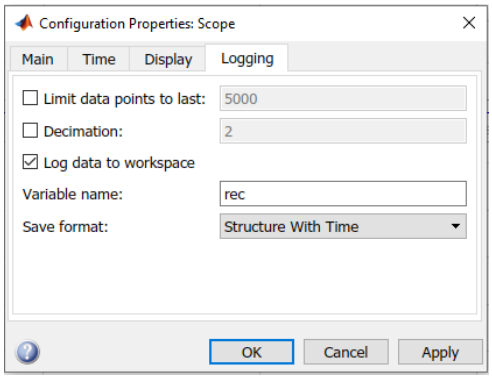

Figure 1.3 Scope logging tab configuration.

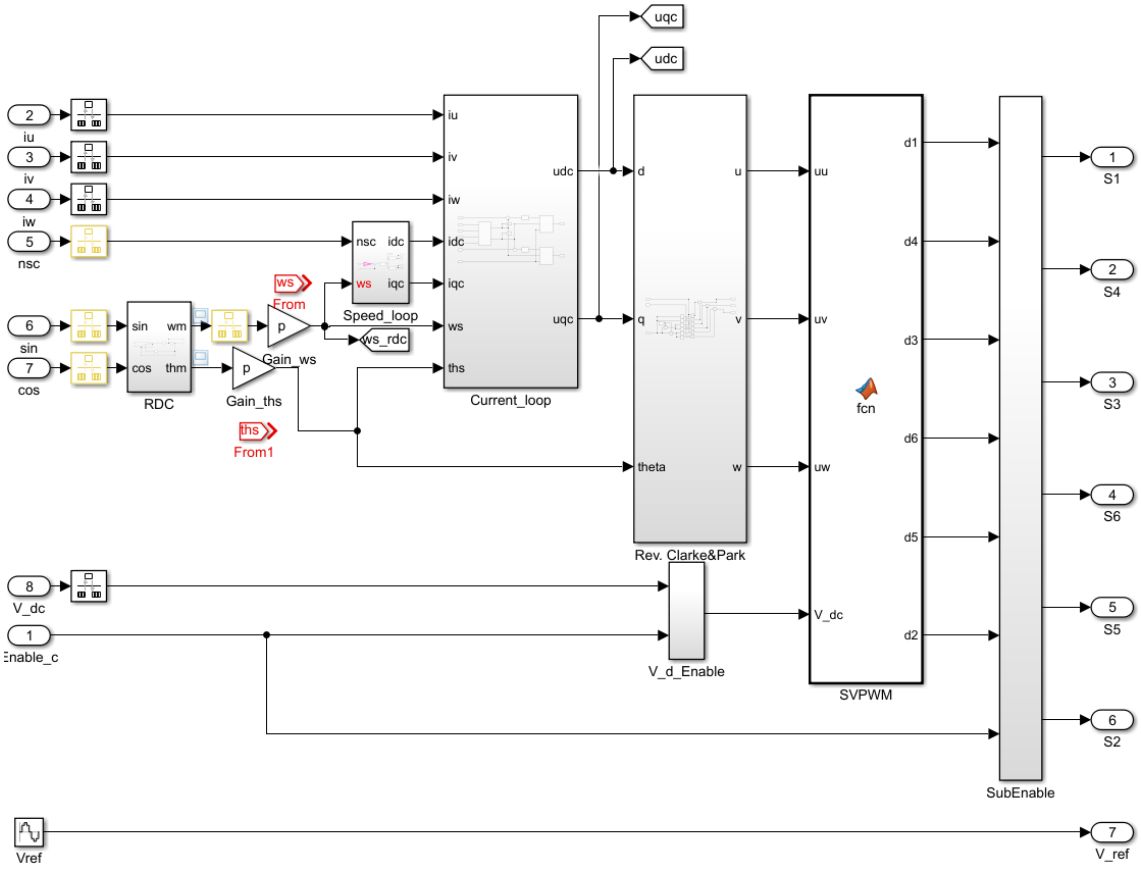

Figure 1.4 Control subsystem.

The 'ws' and 'ths' 'From' blocks are utilised for testing the model without using the Resolver. In the 'Motor' subsystem there are the 'Goto' blocks related to them. All the 'Goto' blocks inside a subsystem must be configured as 'Global' visibility with a unique name as done in the next figure.

The following figure shows how to configure the sinewave 'Vref' for the resolver (on the left) and all the rate-transition blocks. 'Rate Transition1' applies for 'iu', 'iv' and 'iw'; 'Rate Transition4' for 'sin' and 'cos' signals; and 'Rate Transition6' for 'nsc', 'wm' and 'V_dc'. These blocks utilise the sampling frequency variables of MATLAB for changing the sampling time of each signal.

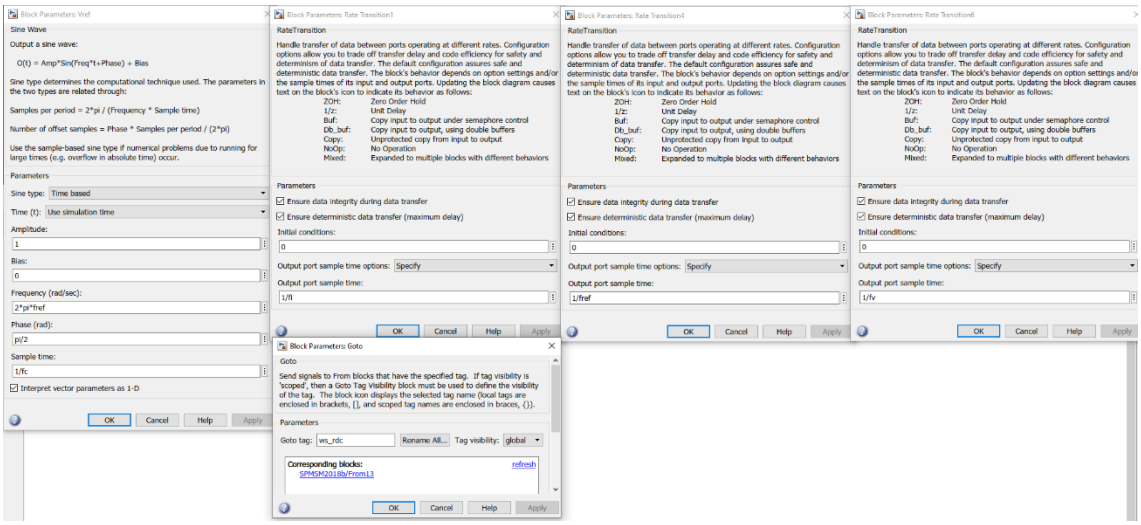

Figure 1.5 Sinewave generator, Rate-transition and Goto blocks configuration.

The RDC subsystem is shown in the following picture with the PI configuration. It is only required to change the Controller type to 'PI', the time domain to 'Discrete-time' and the constants to the variables of MATLAB workspace. The integrator block must be changed to 'Discrete-time' as well while the rest of the parameters can be left as the default values.

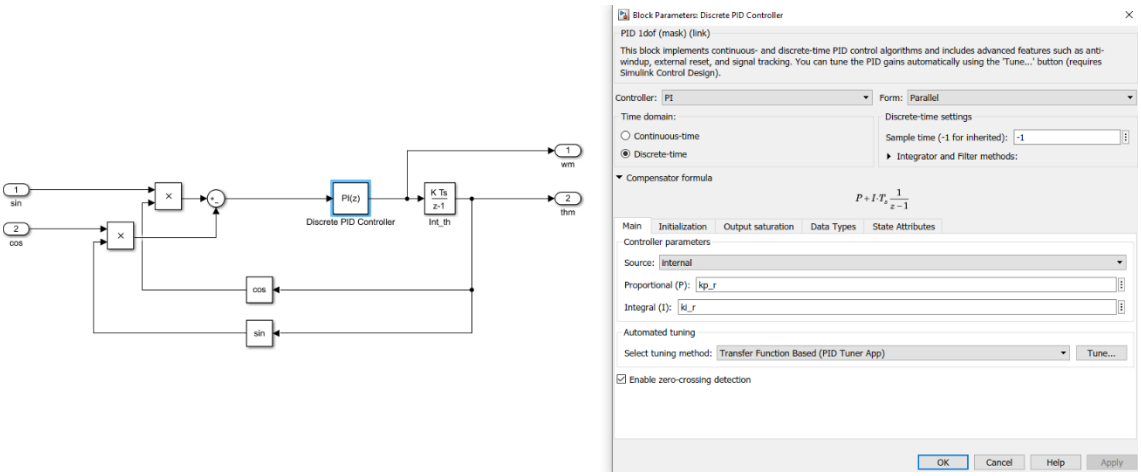

Figure 1.6 RDC subsystem and PI configuration.

The following figure shows the 'Speed_loop' with the configuration of the PI controller and both Lookup Tables.

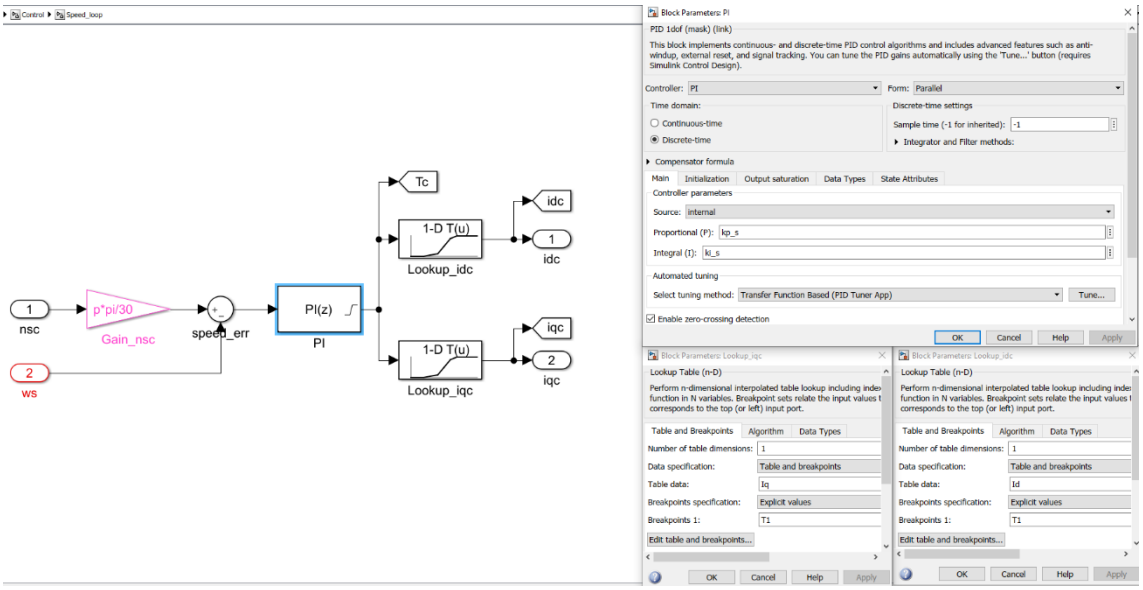

Figure 1.7 Speed_loop subsystem with PI controller and Lookup table blocks configuration.

It is really important to add an output saturation on the tab of this PI controller and the current controllers as it will be seen. In this case, the values are set to the maximum torque variable called 'Tn'. And the Anti-windup method must be set to 'Clamping' for preventing issues.

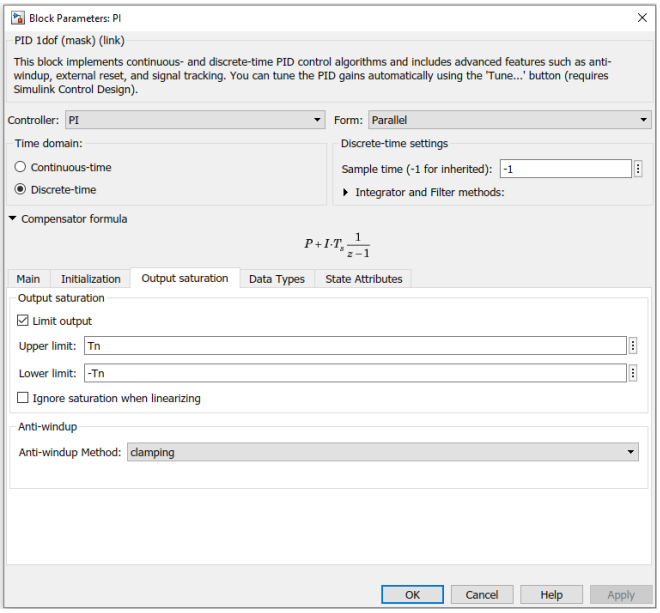

Figure 1.8 PI controller output saturation tab of Speed_loop PI controller.

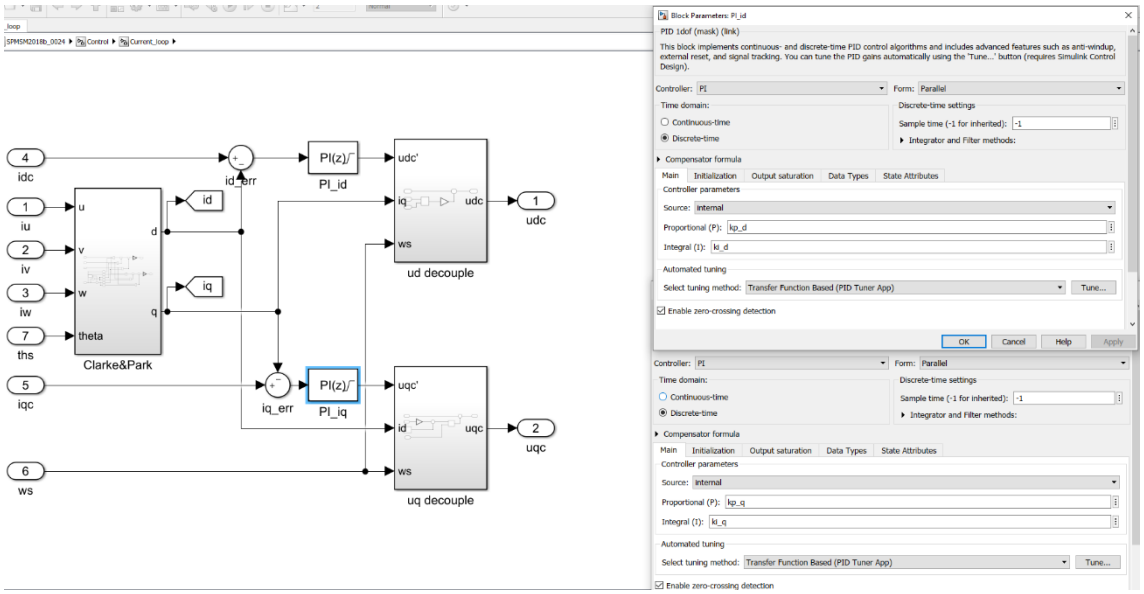

Figure 1.9 Current_loop subsystem with both PI controller blocks configuration.

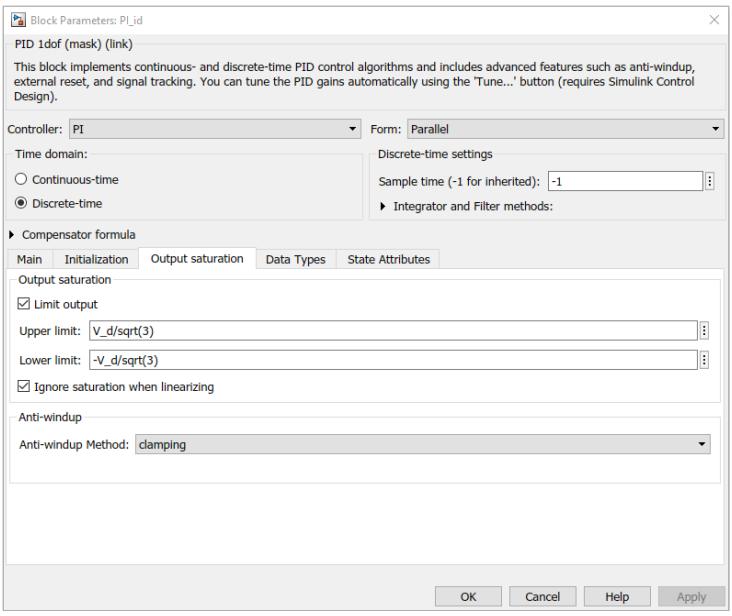

Figure 1.10 PI controller output saturation tab configuration of id current equivalent for iq.

Both current PI controllers have the same configuration in the 'Output saturation' tab.

The 'Clarke&Park' and 'Rev. Clarke&Park' subsystems are also utilised in the motor, so the blocks shown in the following picture can be directly copied and pasted once created.

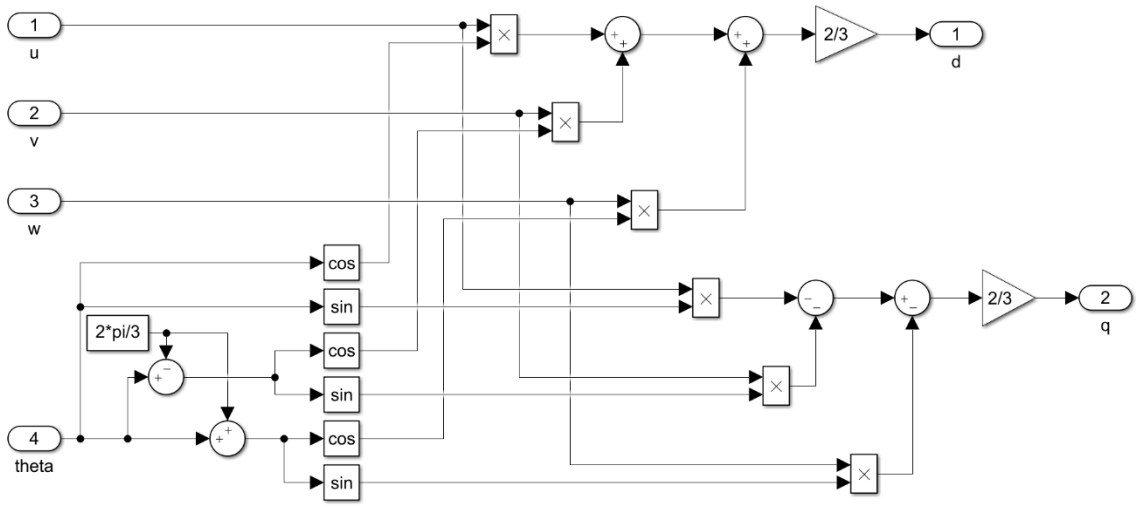

Figure 1.11 Clarke&Park subsystem.

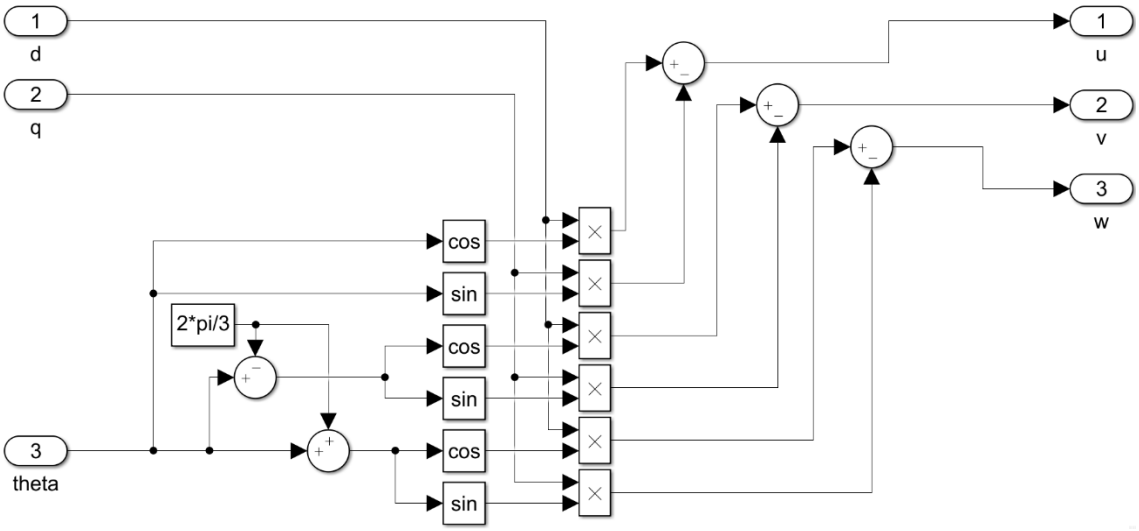

Figure 1.12 Rev. Clarke&Park subsystem.

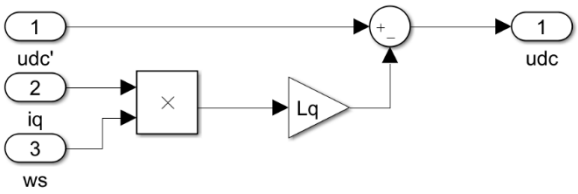

Figure 1.13 ud decouple subsystem.

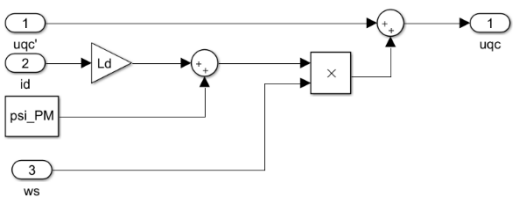

Figure 1.14 uq decouple subsystem.

The ‘V_d_Enable’ and ‘SubEnable’ blocks are switches utilising the Enable signal as shown in the following figure.

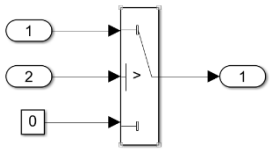

Figure 1.15 V_d_Enable subsystem.

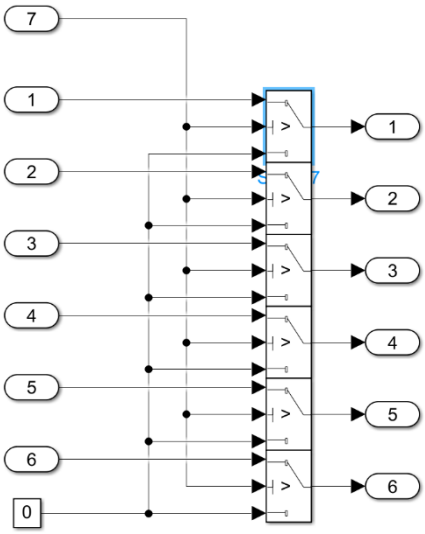

Figure 1.16 SubEnable subsystem.

The 'SVPWM' block is a MATLAB function and the code is described as follows.

```
function [d1,d4,d3,d6,d5,d2]= fcn(uu,uv,uw,V_dc)
alp=pi/3;
ua=(2*uu/3-uv/3-uw/3);
ub=(uv-uw)/sqrt(3);
th=atan2(ub,ua);
if th>=0
   th=th;
else
   th=2*pi+th;
end
if th>=0 && th<alp
   n=1;
elseif th>=alp && th<2*alp
   n=2;
elseif th>=2*alp && th<3*alp
   n=3;
elseif th>=3*alp && th<4*alp
   n=4;
elseif th>=4*alp && th<5*alp
   n=5;
else n=6;
end
if n==1
   v=[1 1 1 ; 0 0 1 ; 0 1 1 ; 1 0 1 ; 0 0 1 ; 1 1 1];
elseif n==2
   v=[1 0 1 ; 0 1 1 ; 1 1 1 ; 0 0 1 ; 0 0 1 ; 1 1 1];
elseif n==3
   v=[0 0 1 ; 1 1 1 ; 1 1 1 ; 0 0 1 ; 0 1 1 ; 1 0 1];
elseif n==4
   v=[0 0 1 ; 1 1 1 ; 1 0 1 ; 0 1 1 ; 1 1 1 ; 0 0 1];
elseif n==5
   v=[0 1 1 ; 1 0 1 ; 0 0 1 ; 1 1 1 ; 1 1 1 ; 0 0 1];
else
   v=[1 1 1 ; 0 0 1 ; 0 0 1 ; 1 1 1 ; 1 0 1 ; 0 1 1];
end
An_inv=(sqrt(2)/(V_dc))*[sin(n*pi/3) -cos(n*pi/3) ; -sin((n-1)*pi/3) cos((n-1)*pi/3)];
Vref=[ua; ub];
```

```
tn=An_inv*Vref;
t0by2=(1-tn(1)-tn(2))/2;
t120=[tn(1);tn(2);t0by2];
tgx = v*(t120);
d1=tgx(1);
d4=tgx(2);
d3=tgx(3);
d6=tgx(4);
d5=tgx(5);
d2=tgx(6);
```

This code should not be modified since is not just a script but part of the model.

The following figure shows the PWM generator block placed before the drive. This block is actually part of the control. However, since in RTI libraries the PWM signal is directly generated by an RTI block, this subsystem must be utilised in this model and also in the real-time model of the motor that uses the PWM measurement instead of normal digital input blocks.

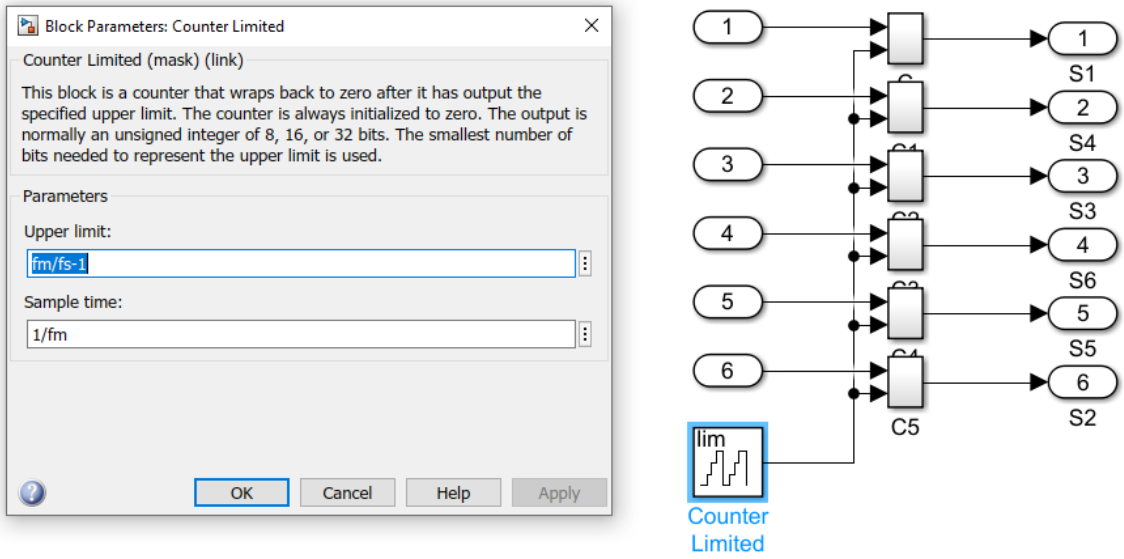

Figure 1.17 PWM generator subsystem with Counter Limited configuration dialogue box.

A counter limited with the shown parameters is required, but the most important thing is the block that compares each signal with it.

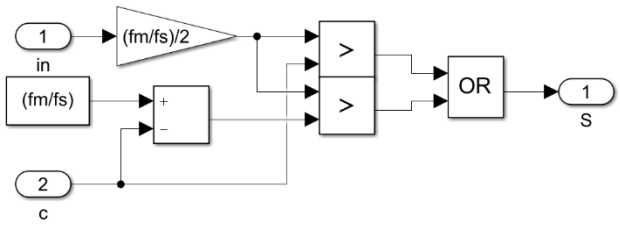

Figure 1.18 Comparator block in PWM generator subsystem.

The drive subsystem contains a 'V_d_Enable' block as it does the 'Control' subsystem. Inside it, there is another system called Drive.

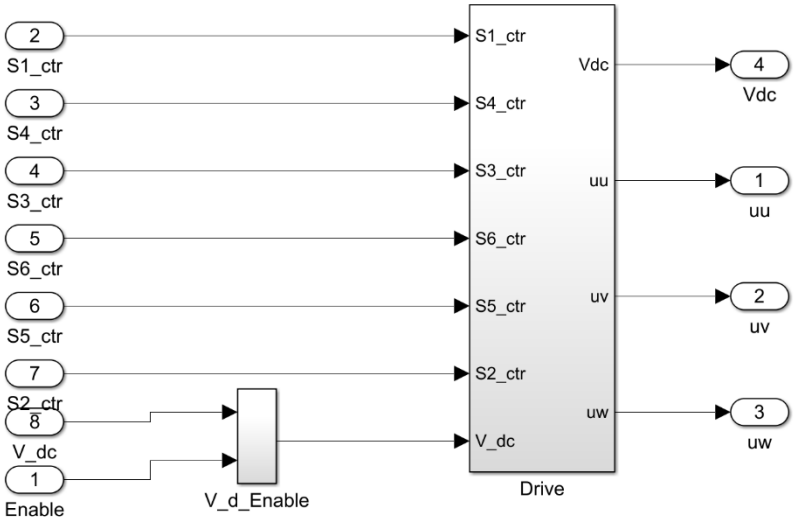

Figure 1.19 Drive subsystem.

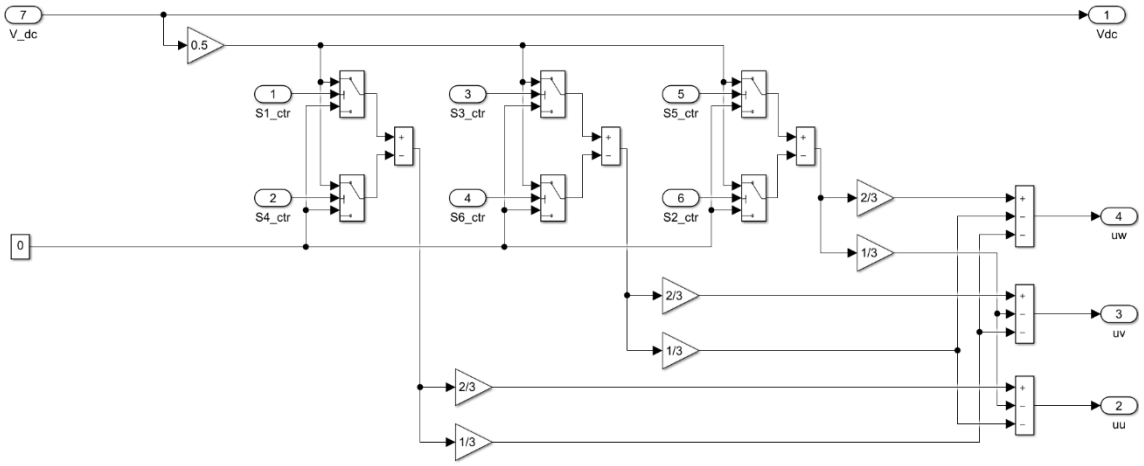

Figure 1.20 Drive subsystem inside the drive block.

The filters are utilised for achieving three sinewaves from the voltages instead of a square wave. The constants are calculated in the script 'Filters.m' of 'Codes' folder. These blocks can be deleted when it is implemented on the real-time model for better performance and permitting to increase the step frequency of the motor. The Clarke and Park transform blocks can be reused from control blocks since they are equivalent.

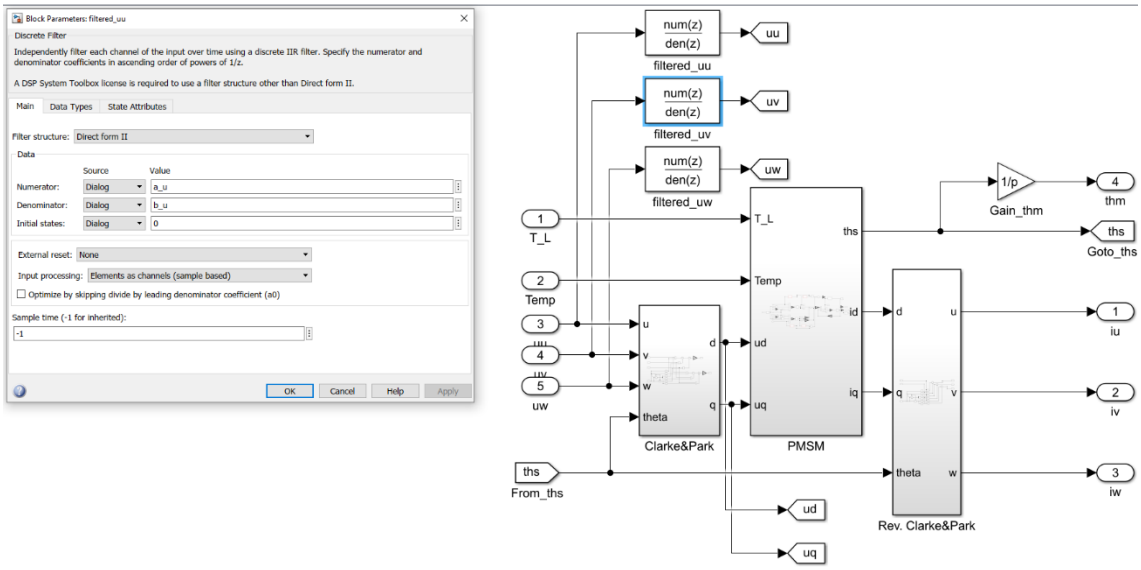

Figure 1.21 Motor subsystem with voltage filter configuration equivalent for all of three filters.

The same applies to the filters in the motor. In this case, they are required for obtaining the mechanical and electrical power and the efficiency of the motor. If these values are not required or the values will be saved for postprocessing, these blocks can be removed as well. In the following figure, all the filters have the same configuration and the Integrator blocks are discrete.

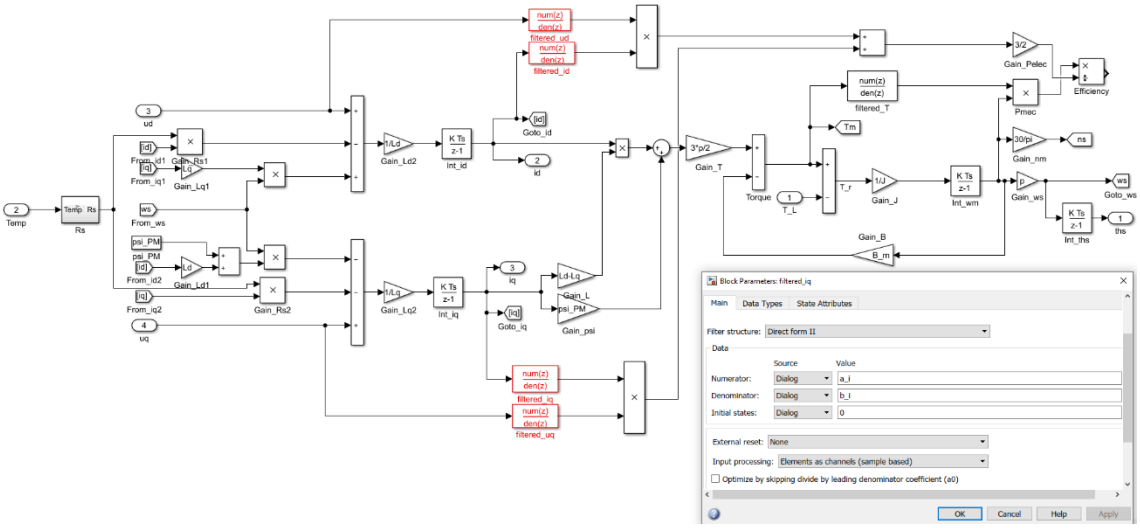

Figure 1.22 PMSM subsystem.

The ‘Rs’ block is the one that determines the real resistance of the motor by its temperature that can be changed by the user. In real-time simulations, this constant can be connected to one of the potentiometers and change it manually.

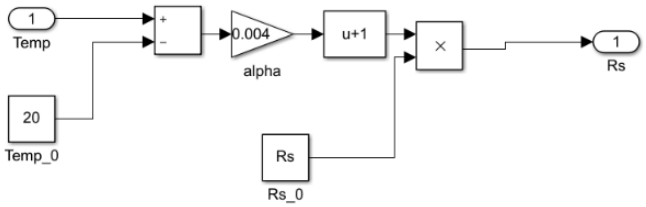

Figure 1.23 Rs subsystem inside PMSM block.

And the last block is 'Resolver' show in the following figure.

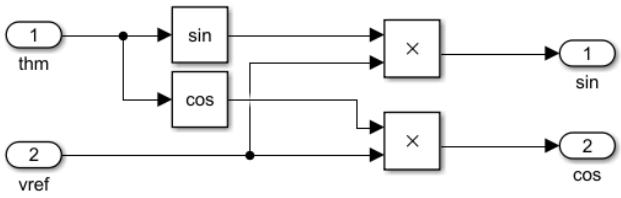

Figure 1.24 Resolver subsystem.

1.2 RTI models with the digital input block

These models must be developed with the 'DIO_CLASS1_BIT_IN_BLx' blocks from RTI libraries for MicroLabBox. The following figure shows the model of the motor.

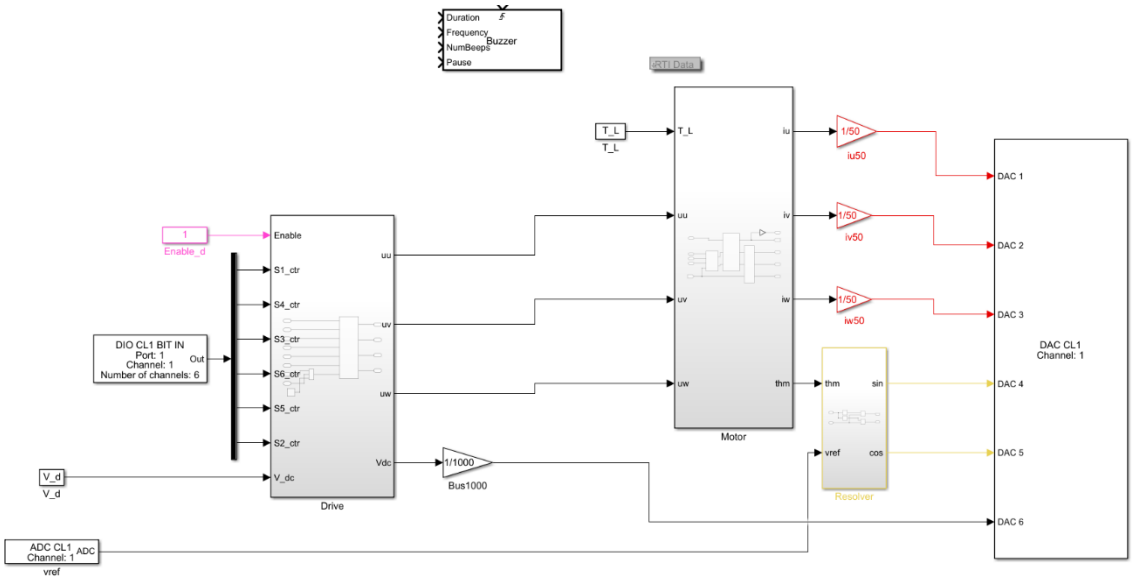

Figure 1.25 Motor model with digital input blocks.

Since the motor subsystem has been already developed in the previous section, the block itself can be directly copied to the 'motor.slx' file as shown in the previous figure. In the next figure, it is appreciable the most important configurations that are to be set in the buzzer block, the digital and analogue input and the analogue output blocks. In addition, for this model, it is recommended to remove the filter blocks of the motor for increasing the step frequency of motor real-time simulation.

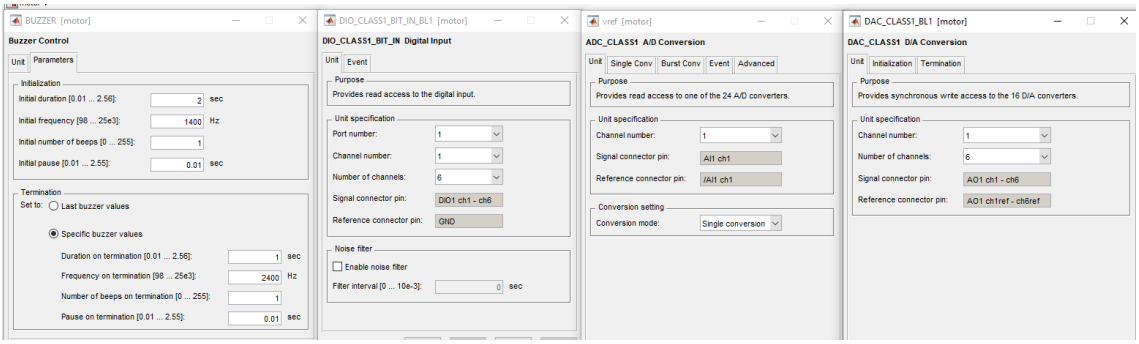

Figure 1.26 Configuration parameters of the buzzer, digital and vref inputs, and analogue output blocks.

The buzzer settings have been set to easily realise when the application has been loaded or unloaded. The parameters of this block can be modified or suppressed if not required while the other must remain as seen. Notice that the channel numbers must be adapted to the connection designed. On the other hand, the 'control.slx' model must contain the 'EMC_MC_PWM_BLx' block for generating the pulses for the drive as the next figure depicts.

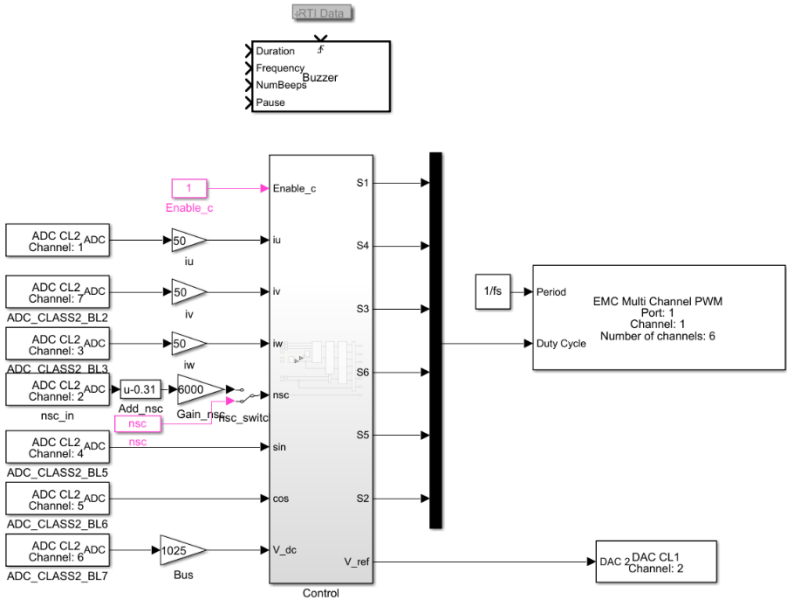

Figure 1.27 Control model for real-time simulations.

This model is the same for the one of the next section since the only change is in the way the motor model receives the pulses signals of the PWM generator that can be seen on the right side of the figure. The configuration for all the analogue inputs is equivalent varying only the channel number as it can be seen on the figure. All the configurations can be seen in the next figure.

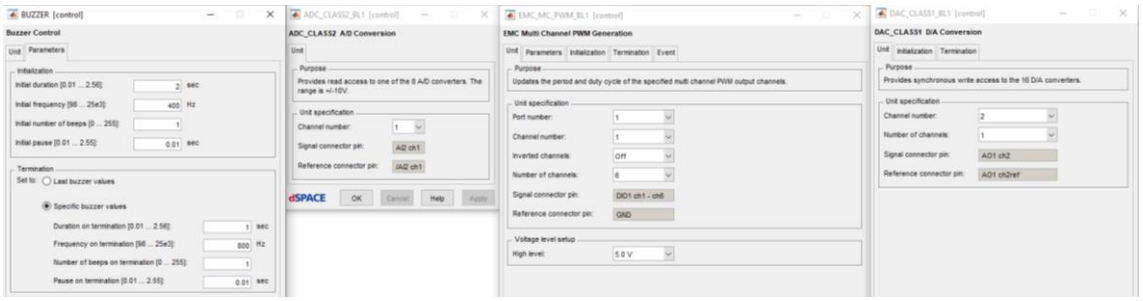

Figure 1.28 Configuration parameters of the buzzer, analogue input and output blocks and the PWM generator block.

In the ‘Unit’ tab of the PWM generator parameters box, the configuration is similar to the digital input block. However, the ‘Parameters’ tab shows something important as the next figure depicts.

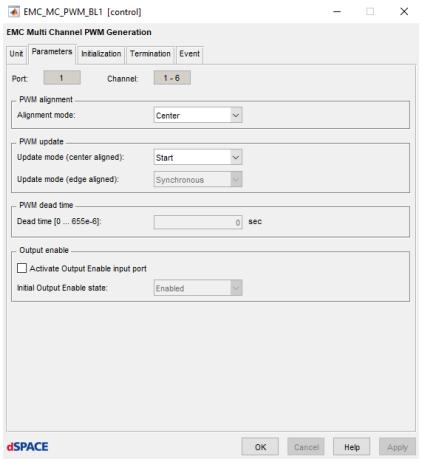

Figure 1.29 Configuration of PWM generator block 'Parameters' tab.

The alignment mode must be set to ‘Center’ to have a space-vector PWM.

1.3 RTI models with the PWM measurement block

The ‘control.slx’ model can be taken from the previous section since there is not a single modification regarding it. Nevertheless, in this case, the model of the motor has changed significantly by replacing the digital input blocks by ‘DIO_CLASS1_PWM2D_BLx’ block. Besides, the PWM generator must be introduced in this model to generate the pulse signals from the duties that the PWM measurement block provides.

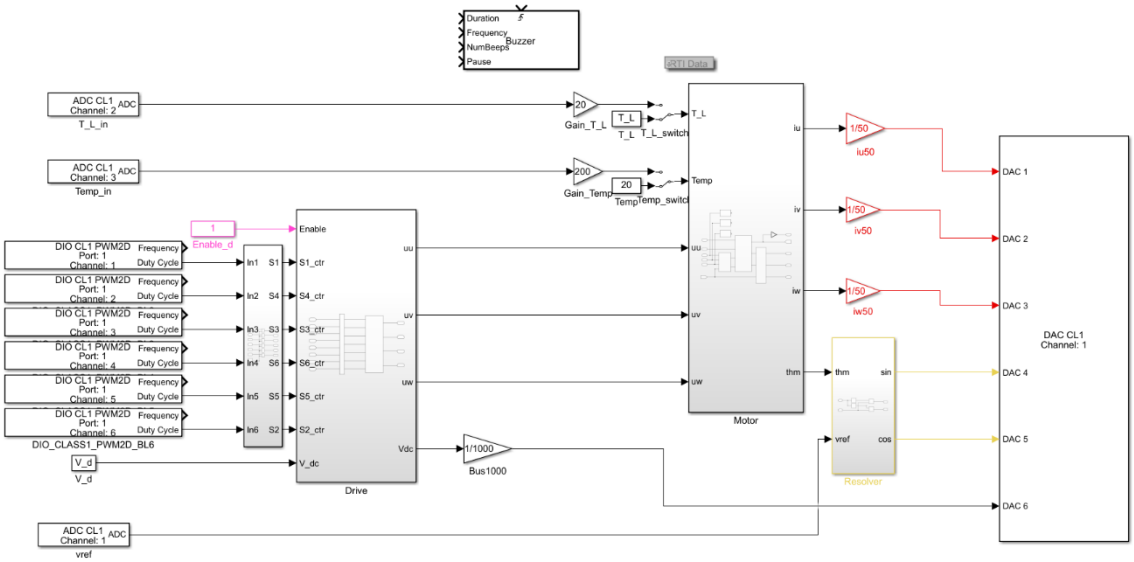

Figure 1.30 Motor model with PWM measurement blocks.

The modification provides the same duties that the control sends to the PWM generator block. For this reason, the PWM generator block utilised in 'SPMSM2018b.slx' model must be included for providing the required pulses to the drive. Each of these blocks must be configured with its channel as the figure shows and the frequency outport can be left without connection since it is not required. The configuration parameters dialogue box is shown in the following figure.

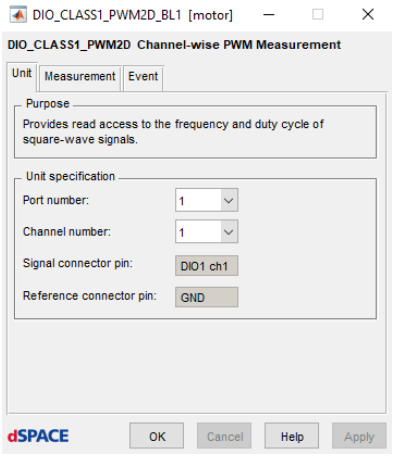

Figure 1.31 Configuration dialogue box of PWM measurement block of channel 1.

2 MATLAB scripts

In this chapter, some scripts have been utilised for calculating parameters and automating the procedure. Additionally, some plot functions are shown to provide additional information about useful functions for data postprocessing.

2.1 Operational scripts and functions

In 'PMSM' folder there must be two main scripts named 'Simulink_loading.m' and 'RTI_loading.m'. The first one is shown below.

```
if contains(pwd,'\PMSM')
   if not(endsWith(pwd, '\PMSM\Matlab_files'))
      cd([extractBefore(pwd,'\PMSM') '\PMSM\Matlab_files']);
   end
   run([pwd '\..\Codes\Lookup_table.m']);
   run([pwd '\..\Codes\PI_currents.m']);
   run([pwd '\..\Codes\PI_speed.m']);
   run([pwd '\..\Codes\PI_resolver.m']);
   run([pwd '\..\Codes\Filters.m']);
   open([pwd '\..\SPMSM2018b.slx']);
else
   disp('Set PMSM as the current folder!')
end
```

This script checks the path of the current folder and, if it contains the 'PMSM' folder, sets the 'Matlab_files' as the current folder. Afterwards, it executes all the functions in the 'Codes' folder and opens the 'SPMSM2018b.slx' model for Simulink simulations of the full system.

The code of the second main script is the following.

```
version=4;

if contains(pwd,'\PMSM')
   if not(endsWith(pwd, '\PMSM\Matlab_files'))
      cd([extractBefore(pwd,'\PMSM') '\PMSM\Matlab_files']);
   end
   run([pwd '\..\Codes\Lookup_table.m']);
   run([pwd '\..\Codes\PI_currents.m']);
   run([pwd '\..\Codes\PI_speed.m']);
   run([pwd '\..\Codes\PI_resolver.m']);
   run([pwd '\..\Codes\Filters.m']);
   open([pwd '\..\RTI_models\' num2str(version) '\control.slx']);
   open([pwd '\..\RTI_models\' num2str(version) '\motor.slx']);
   rtwbuild('control');
   rtwbuild('motor');
else
   disp('Set PMSM as the current folder!')
end
clear version;
```

It is similar but it includes the number of the version that must be changed for the desired RTI model version in the 'RTI_models' folder. The models opened are the ones in the folder of the version number and it also builds them into each MicroLabBox. This procedure takes a long time since both models must be opened and uploaded.

The functions inside the 'Codes' folder are described next. The first one is the 'Lookup_table.m' file.

```
clc; clear Id Iq T1; close all;
```

```
Isn=In*sqrt(2);
k=1;
for is=0:0.001:Isn
  if Ld==Lq
    id=0;
    iq=is;
    T1(k)=1.5*p*psi_PM*iq;
  else
    a=[2 psi_PM/(Ld-Lq) -is*is];
    R=roots(a);
    id=min(R(1),R(2));
    iq=sqrt(is*is-id*id);
    T1(k)=1.5*p*(psi_PM+(Ld-Lq)*id)*iq;
  end
  Iq(k)=iq;
  Id(k)=id;
  k=k+1;
end

clear k Isn is iq id i a R;
```

This function makes iterations to calculate the current distribution between d and q axes for achieving the maximum torque for a given current module. If the motor is an SMPMSM and the inductances are set to the same value, the current in d axis is set to 0. Otherwise, the maximum torque is calculated by the roots of the derivative function dependant on the currents. This iteration is done for each value of current from 0 to the rated current with a precision of 0.001A.

The following script, named 'PI_currents.m', calculates the optimal values for Kp and Ki constants of the PI controllers of the current loop.

```
clc;
kpd_min=0.0001;
kpd_max=10000;
kpq_min=kpd_min;
kpq_max=kpd_max;
for i=0:1:25
  kpd=(kpd_min+kpd_max)/2;
  kpq=(kpq_min+kpq_max)/2;
  hd=mag2db(bode(tf([kpd kpd*Rs/Ld],[Ld (kpd+Rs) kpd*Rs/Ld]),pi_i*2*pi));
  hq=mag2db(bode(tf([kpq kpq*Rs/Lq],[Lq (kpq+Rs) kpq*Rs/Lq]),pi_i*2*pi));
  if hd<-3
    kpd_min=kpd;
  else
    kpd_max=kpd;
  end
  if hq<-3
    kpq_min=kpq;
  else
    kpq_max=kpq;
  end
end

kp_d=kpd;
ki_d=kpd*Rs/Ld;
kp_q=kpq;
ki_q=kpq*Rs/Lq;
clear i hd hq kpd kpq kpd_max kpd_min kpq_max kpq_min;
```

An iterative method with a binary search of Kp values is performed looking for an approximation of the constant to the value that whose cutoff frequency is 'pi_i' value. This parameter must be previously set and included in the 'Set.mat' file and it represents the cutoff frequency of PI currents. For this purpose, the bode value at the given frequency is compared with -3dB for

checking which limit must be shrunken for the next iteration. Note that if Kp value stacks at the maximum or the minimum value, the limits must be modified and the code executed again.

The 'PI_speed.m' script has the same effect with the speed loop.

```
clc;
kp_min=0.00001;
kp_max=100000;

for i=0:1:50
   kp=(kp_min+kp_max)/2;
   h=mag2db(bode(tf([kp kp*B_m/J],[J (kp+B_m) kp*B_m/J]),pi_s*2*pi));
   if h<-3
      kp_min=kp;
   else
      kp_max=kp;
   end
end

kp_s=kp;
ki_s=kp*B_m/J;
clear i h kp kp_max kp_min;
```

The same applies to the resolver PI controller.

```
clc;
kp_min=0.00001;
kp_max=100000;

for i=0:1:50
   kp=(kp_min+kp_max)/2;
   h=mag2db(bode(tf([kp kp^2/2],[1 kp kp^2/2]),pi_r*2*pi));
   if h<-3
      kp_min=kp;
   else
      kp_max=kp;
   end
end

kp_r=kp;
ki_r=kp^2/2;
clear i h kp kp_max kp_min;
```

The iterations can be modified but it is not recommended. There is no need to have the exact value for the constants that provides the cutoff frequency. However, it is useful to understand the exact effect that a given frequency for each PI controller can have over the system. As it happens with the currents, the speed cutoff frequency is determined by the variable 'pi_s' and 'pi_r' for the RDC.

Finally, the 'Filters.m' script file is simply as follows.

```
clc;
[a_u b_u]=butter(2,nsc*p/(30*fm));
[a_i b_i]=butter(2,5/fm);
```

The array variables that 'butter()' function sets are the values introduced in the discrete filters of the model.

2.2 Plot code script

Although many script codes have been created for this project, only one of them is shown since they depend on the data file format and the order utilised in ControlDesk.

```
i=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;84 81 20 23 49 252 249 220 225 223 201 192 189 156 200];

ti1=[5.48 8.3 6.26 2.9 2.45 2.35 5.3 4.5 5.14 5.4 7.4 4.6 8 5.3 6];
ti2=[1665 1468.1 5149 5356 1711 3169 3040 1031 1243 1161 98 167.5 0.6 392.6 48.665];
len=[.01 .01 .01 .01 .01 .01 .01 .01 .01 .01 .3 .3 .3 .3 .3];

irange=[4 4 4 4 4 7 7 7 7 7 5 5 5 5 5];
urange=[350 350 350 350 350 350 350 350 350 350 60 60 30 30 60];

fign=5;
if contains(pwd,'\PMSM')
   dir=[extractBefore(pwd,'\PMSM') '\PMSM\ABB\'];
else
   disp('Set PMSM as the current folder!')
   return
end
if not(exist('carlos'))
   load([dir 'ABB_data.mat']);
end
for j=1:15
   load([dir 'ControlDesk_data\Proves_io_modules\' num2str(i(1,j)) '.mat']);
   mod1=(floor((j-1)/fign)+1)*3;
   mod2=(mod(j-1,fign)+1)*2;

   figure(mod1-2);

   subplot(fign,2,mod2-1);
   plot(rec.X(2).Data,rec.Y(24).Data,rec.X(2).Data,rec.Y(25).Data,rec.X(2).Data,rec.Y(26).Data);
   xlim([ti1(j) ti1(j)+len(j)]);
   ylim([-irange(j) irange(j)]);
   if mod2-1==1
      a=i(1,j);
      str=[num2str(a)];
      for k=1:fign-1
         str=[str ', ' num2str(a+k)];
      end
      title(['Currents (iu, iv, iw) tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('I (A)','FontSize',18);
%    ylabel('Current (A)','FontSize',18);
%    legend({'iu','iv','iw'},'FontSize',16);

   subplot(fign,2,mod2);
   plot(carlos(i(2,j)).t, carlos(i(2,j)).currents);
   xlim([ti2(j) ti2(j)+len(j)]);
   ylim([-irange(j) irange(j)]);
   if mod2-1==1
      title(['Real currents (iu, iv, iw) tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('I (A)','FontSize',18);
%    ylabel('Current (A)','FontSize',18);
%    legend({'iu','iv','iw'},'FontSize',16);

   figure (mod1-1);

   subplot(fign,2,mod2-1);
   [m,l]=butter(2,500/25000);
   plot(rec.X(2).Data,filter(m,l,rec.Y(21).Data),rec.X(2).Data,filter(m,l,rec.Y(22).Data),rec.X(2).Data,filter(m,l,rec.Y(23).Data));
%    plot(rec.X(2).Data,rec.Y(21).Data,rec.X(2).Data,rec.Y(22).Data,rec.X(2).Data,rec.Y(23).Data);
   xlim([ti1(j) ti1(j)+len(j)]);
   ylim([-urange(j) urange(j)]);
   if mod2-1==1
      title(['Filtered phase voltages (uu, uv, uw) tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('V (V)','FontSize',18);
```

```
%    ylabel('Voltage (V)','FontSize',18);
%    legend({'uu','uv','uw'},'FontSize',16);

   subplot(fign,2,mod2);
   plot(carlos(i(2,j)).t, carlos(i(2,j)).vphase);
   xlim([ti2(j) ti2(j)+len(j)]);
   ylim([-urange(j) urange(j)]);
   if mod2-1==1
      title(['Real phase voltages (uu, uv, uw) tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('V (V)','FontSize',18);
%    ylabel('Voltage (V)','FontSize',18);
%    legend({'uu','uv','uw'},'FontSize',16);

   figure (mod1);

   subplot(fign,2,mod2-1);
   [m,l]=butter(2,5/25000);
   plot(rec.X(2).Data,rec.Y(35).Data,rec.X(2).Data,filter(m,l,rec.Y(35).Data));
   ylim([-2 4]);
   xlim([ti1(j) ti1(j)+len(j)]);
   if mod2-1==1
      title(['Motor and filtered torques tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('T (Nm)','FontSize',18);
%    ylabel('Torque (Nm)','FontSize',18);
%    legend({'Motor torque','Filtered torque'},'FontSize',16);

   subplot(fign,2,mod2);
   plot(carlos(i(2,j)).t, carlos(i(2,j)).torque,carlos(i(2,j)).t,ones(1,100000)*carlos(i(2,j)).T_avg);
   ylim([-2 4]);
   xlim([ti2(j) ti2(j)+len(j)]);
   if mod2-1==1
      title(['Real motor and avegrage torques tests ' str],'FontSize',18,'fontweight','light');
   end
   ylabel('T (Nm)','FontSize',18);
%    ylabel('Torque (Nm)','FontSize',18);
%    legend({'Motor torque','Filtered torque'},'FontSize',16);
end

clear i j ti1 ti2 len dir irange urange l m fign str a k mod1 mod2;
```

The ‘i’ array relates the ControlDesk data files generated with their corresponding code of all 290 real tests data. The arrays ‘ti1’, ‘ti2’ and ‘len’ describe the start time for each test of simulation and real data and the time length of each test respectively. The ‘irange’ and ‘urange’ are utilised for describing the maximum and minimum values of the y axis plot of each test. And the ‘fign’ variable describes the number of subplots for each figure. Note that all the matrix must have the same dimensions, in this case, 15 values, and the ‘fign’ value must be an integer multiple of that value.

If ‘carlos’, the name of the structure variable, where all the real data of the motor has been stored, does not exist, it loads the file that is stored in the folder of the motor to upload it. The file could be uploaded every time the code is executed. However, since its weight is about 3GB, it is faster to check if it already exists.

The ‘for’ loop is utilised for applying the same code for all 15 tests so the first line loads the data struct whose name is ‘rec’ as described in the data acquisition section of the report. The variables ‘mod1’ and ‘mod2’ are utilised for determining the subplot and figure where each graphic must be plotted. The function plot utilising the referencing methodology of each struct is utilised. The ‘xlim()’ and ‘ylim()’ functions are used for setting the minimum and maximum values of the plot and the ‘ylabel()’ for the label of y-axis.

The title is only displayed on the first graphic adding the sequential numbers of all the tests in the same figure. For this reason, the loop that concatenates all the string values is inside the conditional clause with the 'title()' function for setting the title label. The 'legend()' function is utilised for setting a legend for each graphic. However, the character '%' has been placed in front of it for commenting the line and therefore avoiding its execution (this can be done with the shortcut 'Ctrl+R', for commenting, and 'Ctrl+T' for uncommenting). The 'FontSize' parameter is set in label functions for setting a bigger font size for readable characters on the figures of the document.

This procedure has been replicated for torque and voltage plots. However, since the filters were not utilised for these test, the filter must be made in this code as postprocessing. The function 'butter()' already used in the 'Filters.m' script, is used to set the variables 'l' and 'm' respectively. The function 'filter()' take these two variables for filtering the voltage square waves and the distorted torque.

For more information about 'butter()' and 'filter()' functions, and structs refer to the following webpages.

https://es.mathworks.com/help/signal/ref/butter.html

https://es.mathworks.com/help/matlab/ref/filter.html

https://es.mathworks.com/help/matlab/ref/struct.html

3 Datasheets

This chapter includes the datasheets of three motors utilised in the project. The first one is the Siemens SMPMSM whose information is in the following figure.

SIEMENS

Data sheet for SIMOTICS S-1FK7

MLFB-Ordering data 1FK7083-2AC71-1BA0

Figure similar

Client order no. :
Order no. :
Offer no. :
Remarks :

Item no. :
Consignment no. :
Project :

Engineering data	
Rated speed (100 K)	2000 rpm
Number of poles	8
Rated torque (100 K)	12.5 Nm
Rated current	6.3 A
Static torque (60 K)	13.30 Nm
Static torque (100 K)	16.0 Nm
Stall current (60 K)	6.10 A
Stall current (100 K)	7.50 A
Moment of inertia	0.0026000 kgm²
Efficiency	93.0 %

Physical constants	
Torque constant	2.13 Nm/A
Voltage constant at 20° C	138.5 V/1000*min⁻¹
Winding resistance at 20° C	0.66 Ω
Rotating field inductance	12.8 mH
Electrical time constant	19.40 ms
Mechanical time constant	1.13 ms
Thermal time constant	50 min
Shaft torsional stiffness	101000 Nm/rad
Net weight of the motor	15.6 kg

Mechanical data	
Motor type	Permanent-magnet synchronous motor
Motor type	Compact
Shaft height	80
Cooling	Natural cooling
Radial runout tolerance	0.050 mm
Concentricity tolerance	0.10 mm
Axial runout tolerance	0.10 mm
Vibration severity grade	Grade A
Connector size	1
Degree of protection	IP64
Design acc. to Code I	IM B5 (IM V1, IM V3)
Temperature monitoring	Pt1000 temperature sensor
Electrical connectors	Connectors for signals and power rotatable
Color of the housing	Standard (Anthracite RAL 7016)
Holding brake	without holding brake
Shaft extension	Feather key
Encoder system	Encoder AS24DQI: absolute encoder single-turn 24 bits

Technical data are subject to change! There may be discrepancies between calculated and rating plate values.

Page 1 of 2

Generated Sun Nov 29 16:28:03 CET 2020

Figure 3.1 Siemens SMPMSM datasheet first page.

The next datasheet figure belongs to the Marathon IPMSM. Combining the datasheet of the official webpage with the catalogue all the information can be found.

1800 RPM (8-Pole; 120 Hz input) - DATA AT 460VAC

Rated HP	FL Torque (Lb-Ft)	NEMA Frame	IEC Frame	FLA (460V)	F.L. Effic.	Min Const Torque Speed	Max Const HP Speed	BEMF (L-L @ 1800)	Resis/Ph (ohms)	Ld (mH)	Lq (mH)	Lq:Ld ratio	Rotor Inertia (Lb-Ft2)
15	43.8	254T	160M	18.5	94.5	90	2160	404	0.225	10.170	16.730	1.6	3.9
20	58.4	256T	160L	22.9	95.0	90	2160	400	0.207	6.570	11.040	1.7	4.8
25	72.9	284T	180M	28.1	95.0	90	2160	404	0.100	6.050	9.760	1.6	5.7
30	87.5	286T	180L	33.7	95.4	90	2160	387	0.062	4.810	7.670	1.6	6.5
40	116.7	286T	180L	45.5	95.4	90	2160	395	0.060	4.420	6.970	1.6	7.2

Product Information Packet: Model No: 254TPFSA10088, Catalog No:SY010 15 HP Permanent Magnet (PMAC) Motor, 3 phase, 1800 RPM, 460 V, 254TC Frame, TEFC

marathon Motors

Nameplate Specifications

Output HP	**15 Hp**	Output KW	**11.2 kW**
Frequency	**120 Hz**	Voltage	**460 V**
Current	**18.0 A**	Speed	**1800 rpm**
Service Factor	**1**	Phase	**3**
Efficiency	**94 %**	Power Factor	**80**
Duty	**Continuous**	Insulation Class	**H**
Design Code	**No Design Code**	KVA Code	**NO KVA CODE**
Frame	**254TC**	Enclosure	**Totally Enclosed Fan Cooled**
Thermal Protection	**No**	Ambient Temperature	**40 °C**
Drive End Bearing Size	**6309**	Opp Drive End Bearing Size	**6210**
UL	**Recognized**	CSA	**Y**
CE	**Y**	IP Code	**54**

Technical Specifications

Electrical Type	**AC Permanent Magnet**	Starting Method	**Inverter Only**
Poles	**8**	Rotation	**Reversible**
Resistance Main	**0 Ohms**	Mounting	**Rigid Base**
Motor Orientation	**Horizontal**	Drive End Bearing	**Ball**
Opp Drive End Bearing	**Ball**	Frame Material	**Cast Iron**
Shaft Type	**T**	Overall Length	**25.68 in**
Frame Length	**12.25 in**	Shaft Diameter	**1.625 in**
Shaft Extension	**3.75 in**	Assembly/Box Mounting	**F1 ONLY**
Outline Drawing	**SS203152-1225**	Connection Drawing	**EE7300**

This is an uncontrolled document once printed or downloaded and is subject to change without notice. Date Created:03/11/2020

2 of 6

Figure 3.2 Marathon IPMSM datasheet.

TABLE A.1. SPMSMs parameters

Manufacturer	ABB
Converter model	DGV 700
Rated voltage	380 Vac
Rated current	2.9 A
Rated torque	2.3 Nm
Rated speed	6000 r/min
Poles pairs	3
Stator slots	18
Stator slots per phase	6
N	144 turns/phase
R_S	1.50 Ω (1.1% unbalance)
L	4.9 mH
M	-0.89 mH
Magnets type	NdFeB
Magnets per pole	4
Magnets remanent magnetic flux density	1.18 T
Stator steel saturation magnetic flux density	1.8 T
Axial length	89 mm
Stator outer radius	40.49 mm
Air gap	0.6 mm
Rotor outer radius	22.8 mm
Magnets height	2.5 mm
Shaft radius	9 mm

Figure 3.3 Table of parameters of ABB SMPMSM by the PhD student.

Since some of the values are missing, and for simplicity, the flux linkage of the permanent magnet has been directly provided by the student that works with this motor and is 0.165Vs/rad. The friction factor has been estimated by making simulations and comparing the results with the real data recorded. And the inertia has been extracted from a similar motor of ABB that is shown in the following catalogue lists.

TYPE	Continuous torque at zero speed M_0 [Nm]	Current at continuous torque I_0 [A]	Rated torque M_N [Nm]	Rated current I_N [A]	Rated speed n_N [revi/min]	Mechanical rated power P_N [kW]	Peak torque M_{max} [Nm]	Current at peak torque I_{max} [A]	Motor current limit I_{limit} [A]
(4)	(3)	(1) (2) (3)	(3)	(1) (2) (3)		(3)		(1)	
8C1.1.30	1.3	1.4	1.2	1.3	3000	0.38	4.6	5.5	9.3
8C1.1.60	1.3	2.1	1.05	1.8	6000	0.66	4.6	8.1	13.8
8C1.2.30	2.5	2.5	2.2	2.3	3000	0.69	8.8	9.7	16.4
8C1.2.60	2.5	3.1	1.8	2.4	6000	1.13	8.8	12.2	20.7
8C1.3.30	3.6	2.4	3.1	2.2	3000	0.97	12.6	9.3	15.8
8C1.3.60	3.6	4.3	2.3	2.9	6000	1.45	12.6	16.7	28.3
8C1.4.30	4.5	2.8	3.8	2.5	3000	1.19	15.8	10.8	18.4
8C1.4.60	4.5	4.9	2.5	3	6000	1.57	15.8	19.2	32.5
8C4.0.15	3.9	1.5	3.8	1.5	1500	0.61	14	5.8	9.9

TYPE	Torque constant K_{t0} [Nm/A]	B.e.m.f. between phases at rated speed V [V]	Resistance at terminals R_{UV} [W]	Inductance at terminals L_{UV} [mH]	Moment of inertia of rotor J_m [kgcm2]	Weight m [kg]	Curves
	(1) (2) (3)	(1) (2) (3)	(1) (3)	(4)	(3) (6)		(5)
8C1.1.30	1.05	190	20.8	47	0.9	3.1	501000
8C1.1.60	0.71	257	9.07	21	0.9	3.1	501001
8C1.2.30	1.14	208	6.85	23	1.65	4.1	501002
8C1.2.60	0.9	328	4.26	14	1.65	4.1	501003
8C1.3.30	1.71	310	8.33	31	2.35	4.9	501004
8C1.3.60	0.95	346	2.6	9.6	2.35	4.9	501005
8C1.4.30	1.84	333	6.27	25	3	5.8	501006
8C1.4.60	1.04	376	2.02	8	3	5.8	501007
8C4.0.15	3.04	276	29.3	96	5	6.9	501008

Figure 3.4 ABB catalogue list with a motor similar to the one utilised.

4 Project useful tables

During the project, many versions of the models have been created to keep track of all the work done and quickly correct issues. This chapter provides the tables utilised even though there are not pictures of all the models. Notice that since they are versions, the modifications just change slightly the model. The following table shows the track of all the versions of the main model. The W column defines if it is working with a '1' or if it is not with a '0'.

Table 4.1 Versions of the whole model.

Id	Name	W	Description
0	Adapted version	1	The version from a previous project adapted from 2020a release to 2018b.
1	Load modification	0	The model has been tried to be split into 3 main parts of a real system, however, the load block prompts an error and does not allow to simulate. Next version the modifications are made step by step.
2	Division in 3 blocks	1	Model split into the 3 main blocks (control, inverter and motor) correctly without any additional changes.
3	Uvw currents motor outputs	1	The motor output dq currents have been converted with Clarke and Park transforms to obtain the uwv currents and converted again in the controller block.
4	Load split	1	The split has been extracted having an adjacent block with constant torque, friction torque and inertia.
5	6 pulse signals for the inverter	1	Addition of an 'AND' block to in the inverter and a 'unary minus' block for inverting each one of the 3 pulse signals for obtaining each leg signal couple.
6	Torque equation error and 'unary minus' substitution	1	The pulses can be converted to Booleans and substitute the 'unary minus' block by a 'NOT'. A gain in torque equation was wrong and has been changed from '2/3' to '3/2'.
7	Added DC bus and Enable button	1	The DC bus is a signal introduced into the Inverter and extracted by a voltage measurement. The enable is required in the control and the drive. The PI controllers must saturate by utilising the V_d variable.
8	The only output is the angular position signal	1	The motor only has the angle as output since it will be converted utilising the resolver. Besides, the speed setpoint can be switched between some profiles to make different tests.
9	Added Rate-Transition blocks	1	Added Rate-Transition blocks to the control and derivative blocks modified to 'discrete' type.
10	Transistor bridge	0	The inverter has been changed to utilise switches instead of multiplications and to have an improved representation of a real inverter. But the signal is not equivalent.
11	Transistor bridge 2	1	It has been changed the model of the inverter with a new design of the inverter ideated by logic deduction.
12	Rate-Transition sampling times	1	This has required to make a deep research on the functioning of sampling frequencies and it has been taken into account the frequency of the currents at the rated speed which is 133.33Hz for Siemens motor.

13	Simplification of the model for running in MicroLabBox	1	Some PWM outputs have been suppressed, products replaced by switches and inertia division modified to avoid division by 0. Also, the step time has been reduced to 0.00002s (50kHz) to have fluid functioning. The integrator blocks have been set to 'discrete' type.
14	Added repeating sequence	1	The speed setpoint has been changed to repeating sequence for simplifying the model and allow the board to calculate faster. The inverter has been replaced for one with 6 real inputs (without utilising an 'AND' function).
15	Replaced the new PWM_0002	0	The PWM generator has been replaced by one utilising the counter-limited. However, the model does not work correctly on real-time. The Kp of the speed controller has been reduced and it seems to have a positive effect but it is still unstable.
16	PI adjustment, parameter sets created and block separation	0	All the PI have been tuned to avoid the additional noise and the parameters sets have been created and substituted by the parameters sets in the model for easing the modifications. The drive and the control have been correctly separated to allow the application to run in MicroLabBox. The model does not work in real-time.
17	Resolver	0	Added resolver block of RTI lib that utilised the RS-232 connector to communicate with the resolver. However, it is still not working correctly.
18	Repeating sequence error detected	1	The repeating sequence has been replaced by a counter limited since the peak value was trimmed due to the step frequency of the model and the comparison with the highest values was not made.
19	Remodelled all the system from scratch	0	To find the sequential issues that have not been found a new model has been developed from scratch. However, the speed does not behave as expected and it does not work correctly.
20	Found anti-windup error	1	The PI controllers with saturation did not have the 'anti-windup' option selected and this prevents the system to keep adding the error of the loop when the output is saturated. Nevertheless, the sine and cosine waves of the resolver have a distortion every 9 seconds that produces an angular measurement distortion for the control.
21	Addition of tri-state mode in enable system	1	When activating the enable the upper transistors must be open while the lower must be closed instead of opening all of them.
22	Space-vector PWM and RDC in the control subsystem	1	The PWM has been converted to a MATLAB function block and provides the duties. The RDC has been included in the control subsystem.
23	Counter-limited block and scopes	1	The repeating sequence block has been replaced by the counter-limited with the comparator blocks. The scopes have been created with the most important signals for fast detection of issues.
24	Variable Rs and filters	1	Some filters have been included for calculating the powers and the voltages, and the resistance can be varied by the temperature of the motor.
25	Cosine wave for resolver and SubEnable modification	1	The sinewave has been replaced by a cosine wave to simplify the operations of the RDC. The enable block of the drive has been replaced by a simple switch of the voltage.

The following tables show the RTI model versions and some separated versions of Inverter, PWM generator and Resolver models made separately to ensure that it worked before advancing in the project and avoid issues.

Table 4.2 RTI model versions.

Id	Name	Works	Description
1	Basic model	1	A model with a basic PWM.
2	First full model	1	A model that utilises the PWM generator of RTI libraries and generates pulses with higher resolution. The space-vector PWM is created with MATLAB function.
3	Lookup tables model	1	The model has included lookup tables for the speed loop. However, the PI tuning is not good enough and many problems have shown up, despite it has been managed to make it work.
4	Digital inputs model	1	The model has been adapted for simply copying the blocks and avoid unnecessary modifications. Just the parameters of input and output blocks are to be modified.
5	PWM inputs model	1	The digital inputs model has been adapted to utilise the PWM measurement blocks.

Table 4.3 Inverter model versions.

Id	Name	Works	Description
1	Inverter_test_0001	1	Test with a Simscape inverter compared with the basic inverter. The comparison is made to check that the new model works correctly.
2	Inverter_test_0002	1	Test with a 3 switch inverter model extracted from: https://pdfs.semanticscholar.org/4c37/d150c975e90e87e4b7cfa098c6215cfc23b4.pdf
3	Inverter_test_0003	1	Test with an inverter with 6 switches modified from the previous inverter.
4	Inverter_test_0004	1	Inverter extracted from the Bimal K. Bose book with 3 switches.
5	Inverter_test_0005	1	Inverter modified from the previous version and adapted for 6 switches forming the definitive model.

Table 4.4 PWM model versions.

Id	Name	Works	Description
1	PWM_0001	0	Calculation of closing time of the transistor. However, it has not been found how to convert the calculated times into a pulse signal without utilising the clock block. This block is not recommended for real-time applications. In the end, the times must be compared with a carrier wave anyway.
2	PWM_0002	1	The sawtooth signal generator of resolution (uint8) that is divided by two when creating a triangular signal. It has been found that the sinusoidal signal must have a maximum value inside the incircle of the hexagon, so the voltage must be limited by its module instead of each component to work correctly on the under-modulation region.
3	PWM_0003	1	PWM generator that only extracts three duty signals extracted from the article [1].
4	PWM_0004	1	Adaptation of the previous version for achieving 6 duty signals from the PWM block.
5	PWM_0005	1	Creation of a new block PWM generator for the RTI model that utilises the PWM measurement block.

Table 4.5 Resolver and RDC models versions.

Id	Name	Works	Description
1	Resolver_0001	0	A first model by sending directly the angle from one side to the other has been tried. But since the range of the physical signals is limited, the 'mod()' function must be utilised and the measurement of this signal is complex since the jump from 2π to 0 produces an infinite slope for the derivative that calculates the speed.
2	Resolver_0002	1	To solve the previous error an 'unwrap' block has been utilised to receive the signal and it works correctly. However, the signal must be sent and received as if it was a real resolver.
3	Resolver_0003	1	A resolver block has been created and it has been tried to create an RDC by utilising the block 'Sample and Hold'.
4	Resolver_0004	1	The resolver block of RTI lib can be utilised for retrieving the angle by sending a sinewave signal. However, the returned value is the same signal of the second version.
5	Resolver_0005	1	The RDC has been created utilising a PLL model with the 'Sample and Hold' block.
6	Resolver_0006	1	The model has been optimised with a cosine wave as explained in the report and the 'Sample and Hold' block has been suppressed.

Additionally, a table with some webpages visited and the utility that each of them has added to the project is listed below.

Table 4.6 Webpages visited with the useful information extracted from each one.

Id	Title	Description	URL
1	MicroLabBox Implementation Documents	Global webpage of basic libraries and other MicroLabBox features.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxImpl&externalid=Container_73b8c0b3-0af1-4634-b626-9ea4850d9f7b_--_&Language=en-us&Release=RLS2020-A
2	Basics on MicroLabBox	RTI is used for Simulink blocks of inputs and outputs and RTLib is for C function development, timers, interruptions, task synchronisation…	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxFeatures&externalid=Topic_bd68baaf-f809-4479-b7b3-68c21a920670_--_&Language=en-us&Release=RLS2020-A
3	Feature overview	Explanation of all internal components and communications of MIcroLabBox. Timing O/I can generate PWM signals.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxFeatures&externalid=Topic_ea2ba2e9-bac1-4201-9db5-8620d9601394_--_
4	Basics on the Web Interface	Ethernet cables can be used as input/output or for Host PC communication for example to load the application.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxHardwareInstallationConfigurationGuide&externalid=Topic_b7b5a120-d2b5-4bf7-afa6-c1772a8b5027_--_
5	Running an Application from Flash Memory	Flash memory is utilised to load applications that work in a loop. The local memory is where the applications are run and it is completely deleted when the board is set off. If there is something loaded in the flash memory, this is loaded to the local memory when the device is set on.	https://www.dspace.com/en/ltd/home/support/documentation.cfm?helpsetid=dSPACESimulatorCompactFeatures&externalid=Topic_ov-00000004-0000-1c39-1104-000000000016_--_&Language=en-us&Release=RLS2019-B
6	Nonvolatile Data Handling (NVDATA)	The nonvolatile data is utilised for accessing the non-volatile memory via a real-time application.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=DS1007Features&externalid=Section_79521330-c5f3-410d-a711-3dc870da34d1_--_&Language=en-us&Release=RLS2019-B
7	Handle Rate Transitions	This Simulink block can be utilised for changing the sampling period of a signal. It can be done securely to ensure data integrity transfer. There is an option to “Automatically handle rate transition for data transfer" to add Rate-transition blocks automatically.	https://www.mathworks.com/help/rtw/ug/handle-rate-transitions.html#f1018329

8	Rate Transition	Meaning of each sampling time value like '-1' which is inherent. All the information about the rate transition block.	https://www.mathworks.com/help/simulink/slref/ratetransition.html
9	Types of Sample Time	Differences between different sample times available for simulations in Simulink.	https://www.mathworks.com/help/simulink/ug/types-of-sample-time.html
10	Model Switching Dynamics in Inverter	FOC model utilising Simscape Electrical libraries.	https://es.mathworks.com/help/mcb/gs/modeling-switching-dynamics-in-inverter-using-simscape-electrical.html
11	SV-PWM	Video d'un SV-PWM en simulink	https://www.youtube.com/watch?v=YwZvKgzlyMc
12	Encoder Wikipedia	Encoder functioning, the frequency of the pulses provides the speed whose resolution depends on the construction (pulses per turn). The controller must read many times until a change is detected for measuring a pulse signal.	https://en.wikipedia.org/wiki/Incremental_encoder
13	Resolver	How to convert output signals of a resolver into speed and angular position signals.	https://www.ti.com/lit/an/slyt661/slyt661.pdf?ts=1600791408641&ref_url=https%253A%252F%252Fwww.google.com%252F
14	How resolvers work	Resolver functioning.	https://www.roboteq.com/applications/all-blogs/14-how-resolvers-work
15	System definition and layout	How to create a Simulink model from scratch with some useful tips.	https://www.mathworks.com/help/simulink/gs/system-definition-and-layout.html
16	How to set up peer-to-peer connections	How to create a connection through the IP address of MicroLabBox. How to change the IP address of the board and check that everything is functioning correctly utilising dSPACE RCP and HIL command prompt.	https://www.dspace.com/en/ltd/home/support/documentation.cfm?topicID=189890&navigationID=191594
17	How to register a platform	How to register a platform board in ControlDesk. COntrolDesk must be utilised for registering a platform and give it a custom name.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=ControlDeskPlatformManagement&externalid=Topic_2c5e9ceb-e196-4ff4-822b-d68c5a9ccd14_--_
18	Connect Simulink-dSPACE	How to connect Simulink to a given dSPACE board. in the window "config>RTI load options" of Simulink type the dSPACE name provided in ControlDesk.	https://www.mathworks.com/matlabcentral/answers/286665-simulink-real-time-build-is-not-building

19	Connecting Resolvers	How to physically connect a resolver.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxHardwareInstallationConfigurationGuide&externalid=Topic_16d83ee5-8e28-4a54-925f-045fffdec738_--_
20	Resolver connectors	Resolver pinouts.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxHardwareInstallationConfigurationReference&externalid=Topic_bbbbe46f-2891-483d-bee8-5624532ba889_--_
21	Resolver interface	MicroLabBox resolver functioning	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxFeatures&externalid=Topic_802ce3ba-0159-44f5-a43c-e24eafccf62f_--_
22	EMC_RESOLVER_BLx	The MicroLabBox block of Resolver for Simulink.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=RTIElectricMotorControlBlocksetReference&externalid=Section_e93510b3-ca17-47f3-9e62-53148de29ac5_--_
23	Time interval measurement	RTILib functions that allow the utilisation of internal counter of MicroLabBox.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxRTLibReference&externalid=Section_ov-00000004-0000-1cfd-1001-0000000000d0_--_
24	Timer services	RTILib functions that allow using other timers of MicroLabBox.	https://www.dspace.com/en/pub/home/support/documentation.cfm?helpsetid=MicroLabBoxFeatures&externalid=Topic_b10854dd-0a24-4ccd-bc1c-8324055cfbbf_--_
25	HiL testing for power electronics systems	Explains many problems related to a HiL for motor controllers.	https://www.ni.com/es-es/innovations/white-papers/12/hil-testing-for-power-electronics-systems.html