Modelling and Simulation of a DC Motor Drive

1 Introduction

A simulation model of the DC motor drive is built using the Matlab/Simulink environment. This assignment aims to familiarize you with basic features of Simulink and to demonstrate modelling approaches applicable to electric drives. After this assignment, you are able to:

1. Model and simulate a DC motor drive in the Matlab/Simulink environment.
2. Tune the current and speed controllers using a model-based approach.
3. Explain effects of the pulse-width modulation (PWM) on control performance.

The model of the DC motor is first built and tested in Section 2. Detailed step-by-step instructions are given for creating the model. In Section 3, the motor model is augmented with models for the DC-DC converter and PWM. In Section 4, the current and speed controllers are added to the model.

A report is to be written on this assignment in groups of two (or alone). Submit your report as a PDF file to the MyCourses portal (`mycourses.aalto.fi`) no later than on *Wednesday, 22.11.2023, at 16:00*.

In your report, answer *briefly* the questions given inside this kind of framed boxes. The report should be clearly and consistently written. The requested figures describing the models and simulation results should be included in the report. Submit also the requested Simulink models to MyCourses. These models will be used to check that you have built the models yourself.

Guidance is available in Auditorium T2 on

- Wednesday, 1.11. at 10:15–12:00
- Wednesday, 15.11. at 10:15–12:00

The assignment will be graded on a scale of 0...20 (two points per problem). You are encouraged to discuss with other students but copying solutions from other groups is not allowed! The reports and models will be checked for plagiarism.

2 DC Motor Model

2.1 Dynamic Equations

First, a simulation model of the permanent-magnet DC motor is built. The following state equations are taken as a starting point:

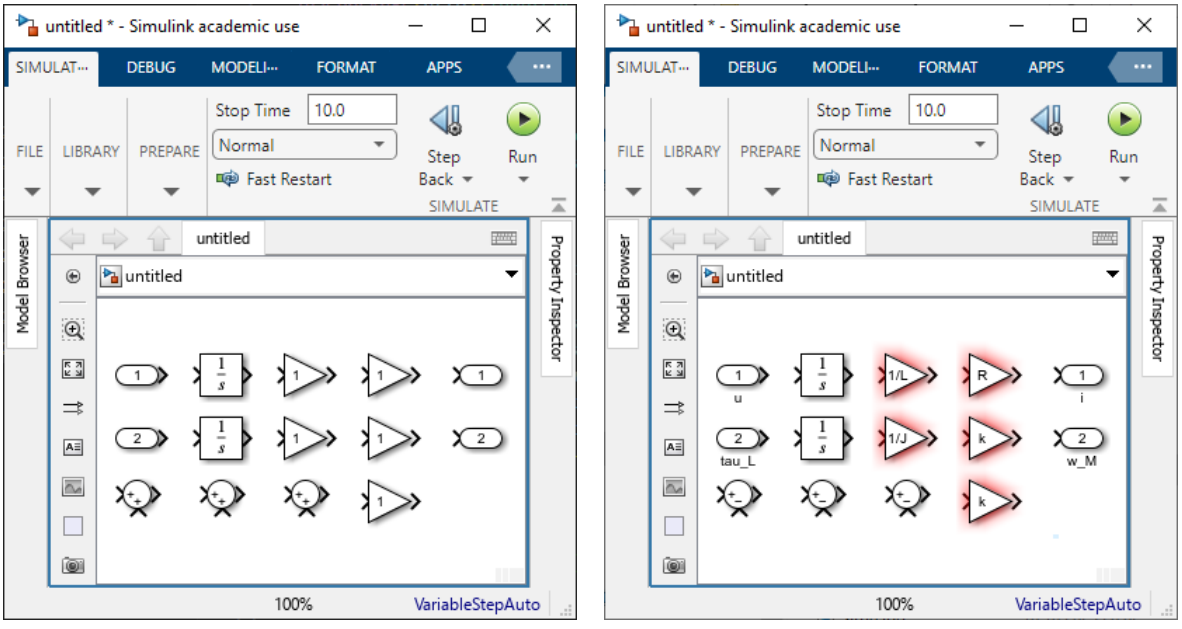
$$L\frac{\mathrm{d}i}{\mathrm{d}t} = u - Ri - k\omega_\mathrm{M} \tag{1a}$$

$$J\frac{\mathrm{d}\omega_\mathrm{M}}{\mathrm{d}t} = ki - \tau_\mathrm{L} \tag{1b}$$

The voltage u and the load torque τ_L are the inputs of the model. The current i and the angular rotor speed ω_M are the outputs of the model. Table 1 gives the rating and parameters of the motor.

Table 1: Rating and parameters of the DC motor.

Rated voltage u_N	120 V
Rated current i_N	20 A
Rated speed n_N	3 000 r/min
Rated torque τ_N	7 Nm
Resistance R	0.5 Ω
Inductance L	2.5 mH
Flux factor k	0.35 Vs
Total moment of inertia J	0.001 kgm^2



(a) (b)

Figure 1: Blocks needed to build the DC motor model: (a) Drag and drop these blocks from the Library Browser to your model; (b) Rename the input and output signals by selecting their names and typing new names. Double click the Sum and Gain blocks and specify them according to the figure.

2.2 Building the Model

Start Simulink by writing the command `simulink` in the Matlab Command Window. Create the blank model and open the Simulink Library Browser, in which you can see the blocks available. Figure 1(a) shows the blocks needed for the motor model. Drag and drop

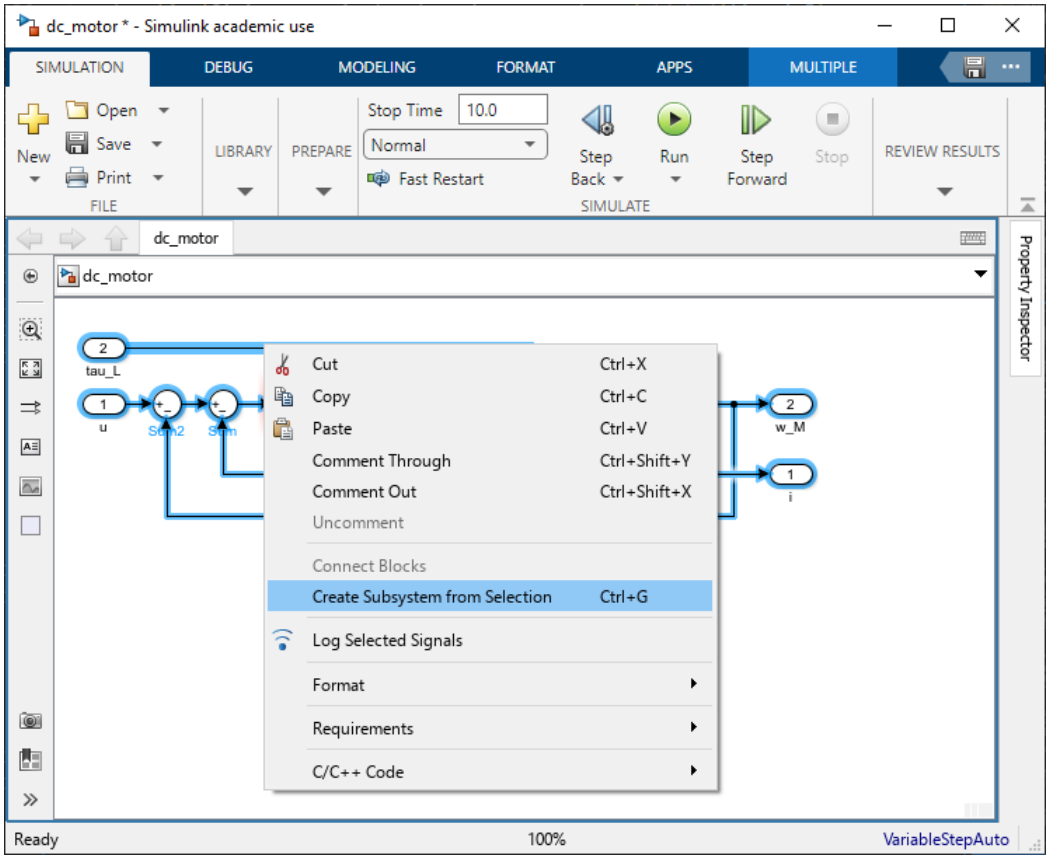


Figure 2: Creating a subsystem.

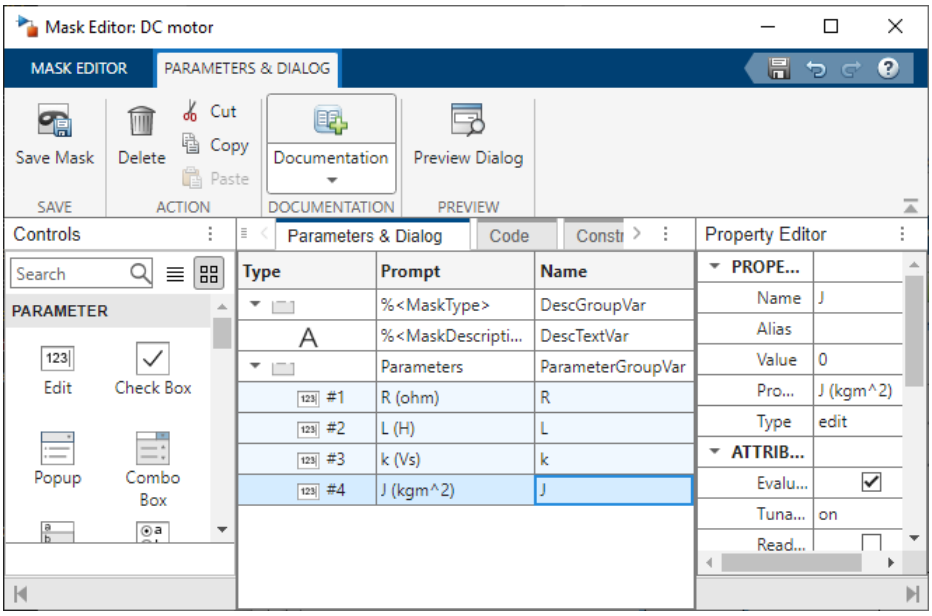


Figure 3: Masking the subsystem.

these blocks from the Library Browser to your new model. Then, rename the input and output signals by selecting their names and typing new names according to Figure 1(b). Double click the Sum and Gain blocks and specify them according to Figure 1(b).

Next, connect the blocks according to the equations given in (1). Use the current i and the speed ω_M as state variables in your model. This means that the output of one integrator block should be the current i and the output of the other integrator block should be the speed ω_M. The inputs of these integrator blocks should be $\mathrm{d}i/\mathrm{d}t$ and $\mathrm{d}\omega_\mathrm{M}/\mathrm{d}t$, which you can solve from (1).

Create the subsystem from your model. You can create the subsystem by selecting all the blocks and then right-clicking one of the selected blocks. This opens a menu similar to one in Figure 2, where you should choose Create Subsystem from Selection. You can rename the new subsystem as `DC motor` (click its name and type the new name).

Next, the subsystem will be masked. Open the Mask Editor by right-clicking the subsystem and choosing Mask→Create Mask. In the Mask Editor, choose the pane Parameters & Dialog. Mask the subsystem according to Figure 3. After you have masked it, you can set numerical values for the model parameters by double-clicking the subsystem, see Figure 4.

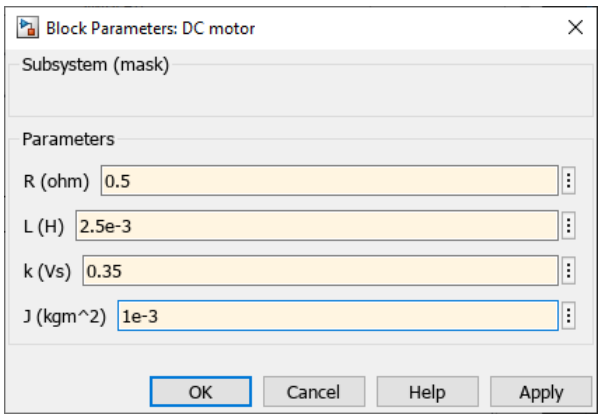


Figure 4: Giving numerical values for parameters.

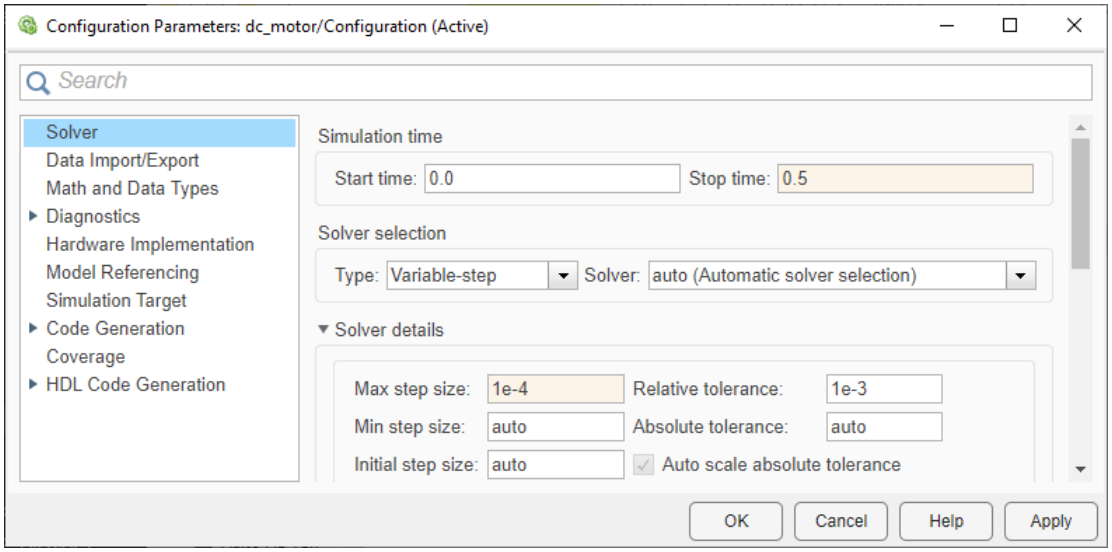


Figure 5: Configuration parameters. Set the new values for the parameters Stop time and Max step size. Furthermore, disable the option Single simulation output on the Data Import/Export pane.

Some settings of the simulation model will be changed. Open the Configuration Parameters window, e.g., using the menu: Modeling→Setup→Model Settings. Choose the pane Solver and configure the following: Stop time = `0.5` and Max step size = `1e-4`, according to Figure 5. Furthermore, choose the pane Data Import/Export and disable the option Single simulation output. Remember to save your model regularly.

2.3 Testing the Model

Before starting to build the control system, the motor model is tested. Connect the blocks Step and To Workspace to the motor model according to Figure 6. Specify the names of the workspace variables to which the To Workspace block writes the data according to the figure. Specify the voltage step block so that the voltage is stepped from zero to its rated value 120 V at $t = 0.1$ s. The load torque should be stepped from zero to the rated torque of 7 Nm at $t = 0.3$ s.

Simulate the model, for example, using the menu: Simulation→Run. After the simulation, you can plot the results in the Matlab workspace using the following commands:

```
subplot(2,1,1);                 % Divides the figure to two subplots
plot(i.time,i.data); grid on;
%i_N = 20;                      % Rated current
%plot(i.time,i.data/i_N);       % This would plot the p.u. current
xlabel('Time (s)'); ylabel('Current (A)');
subplot(2,1,2);
plot(w_M.time,w_M.data); grid on;
xlabel('Time (s)'); ylabel('Speed (rad/s)');
```

It is practical to write and save these commands as a script (e.g., using the name `fig.m`) and then run the script by typing its name (`fig`) in the workspace. This allows you to edit and reuse your scripts. Your simulation results should look similar to those shown in Figure 7. If they look different, you should debug your model.

1. Simulate the sequence corresponding to Figure 7. Modify the plotting script so that the per-unit current and the per-unit speed are plotted (use their rated values as base values and do not normalize time). Show this result in your report. Remember to change the axis labels. Explain why there is a very large peak in the current after the voltage step is applied.
2. Using the analytical motor model, calculate the values for the current i and the rotor speed ω_M in the steady state, when the voltage $u = u_\mathrm{N}$ and the load torque $\tau_\mathrm{L} = \tau_\mathrm{N}$. Compare these values to your simulation results.
3. Limit the rising rate of the voltage to 120 V/0.1 s using the Rate Limiter block, see Figure 8. Place this block between the voltage step and the motor model. Simulate the model and show the results in your report. Briefly comment on the current and speed responses.

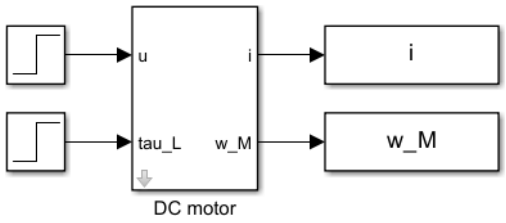


Figure 6: Step inputs for the voltage and load torque.

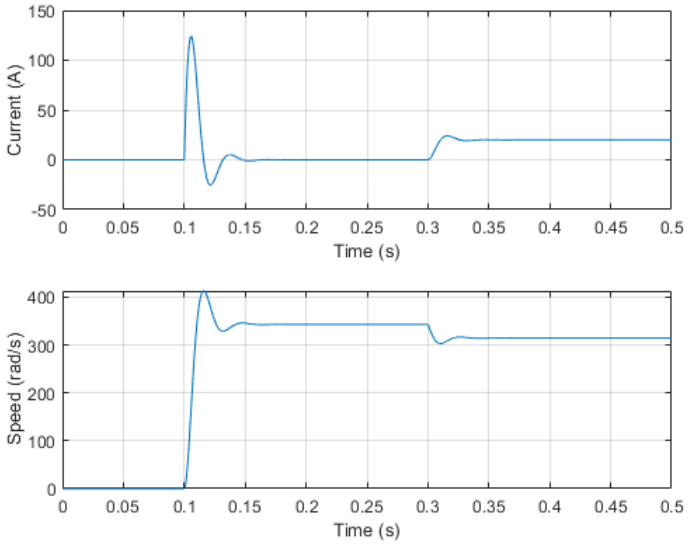


Figure 7: Rated voltage step at $t = 0.1$ s and rated load torque step $t = 0.3$ s.

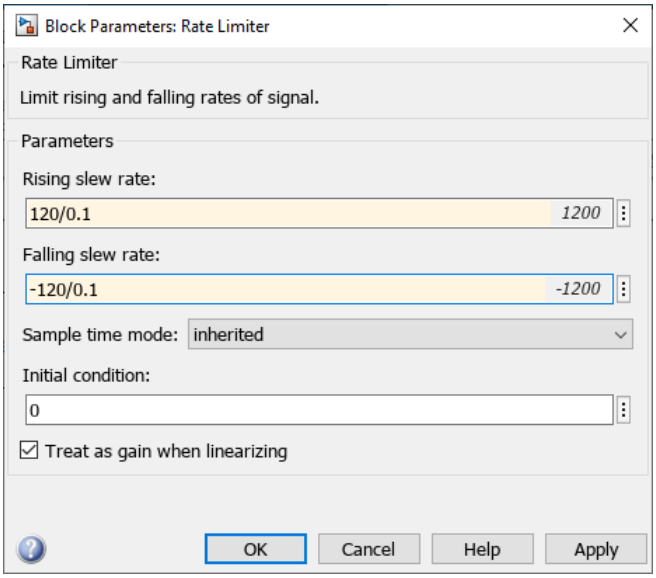


Figure 8: Rate limiter.

3 DC-DC Converter and Unipolar PWM

The motor is fed from a four-quadrant DC-DC converter, whose DC-bus voltage is $U_{\mathrm{dc}} = 140$ V. Ideal power switches are assumed. Hence, the converter can be modelled using the equivalent circuit shown in Figure 9(a). The switching states of the two bi-positional switches are denoted by q_{a} and q_{b}. The value of the switching state is 1 if the switch is connected to the positive potential of the DC bus and otherwise 0. The instantaneous output voltage of the converter is $u = u_{\mathrm{aN}} - u_{\mathrm{bN}}$, where u_{aN} is the voltage between potentials a and N and u_{bN} is the voltage between potentials b and N. Hence, the instantaneous output voltage can be expressed as

$$u = (q_{\mathrm{a}} - q_{\mathrm{b}})U_{\mathrm{dc}} \tag{2}$$

The average voltage over the switching cycle is

$$\overline{u} = (d_{\mathrm{a}} - d_{\mathrm{b}})U_{\mathrm{dc}} \tag{3}$$

where $0 \leq d_{\mathrm{a}} \leq 1$ and $0 \leq d_{\mathrm{b}} \leq 1$ are the duty ratios.

The switching states of the converter are to be generated using unipolar PWM, as illustrated in Figure 9(b). The references for the duty ratios are given by

$$d_{\mathrm{a}} = \frac{1}{2}\left(1 + \frac{u_{\mathrm{ref}}}{U_{\mathrm{dc}}}\right), \qquad d_{\mathrm{b}} = \frac{1}{2}\left(1 - \frac{u_{\mathrm{ref}}}{U_{\mathrm{dc}}}\right) \tag{4}$$

where u_{ref} is the reference voltage. The duty ratios are compared to the carrier signal v_{tri}, which is a triangular wave alternating between 0 and 1 and having a period of $T_{\mathrm{sw}} = 200$ μs. When the duty ratio is higher than the carrier, the corresponding switching state is 1 and otherwise 0. Figure 10(a) shows an implementation of unipolar PWM.

For simulating PWM waveforms with good accuracy, the solver time step should be a few decades shorter than the switching period (or, alternatively, the solver should be informed about the switching instants). Open the Configuration Parameters window and set the parameter Max step size = `1e-6` (see Figure 5). Naturally, the simulation becomes slower due to the shorter time step.

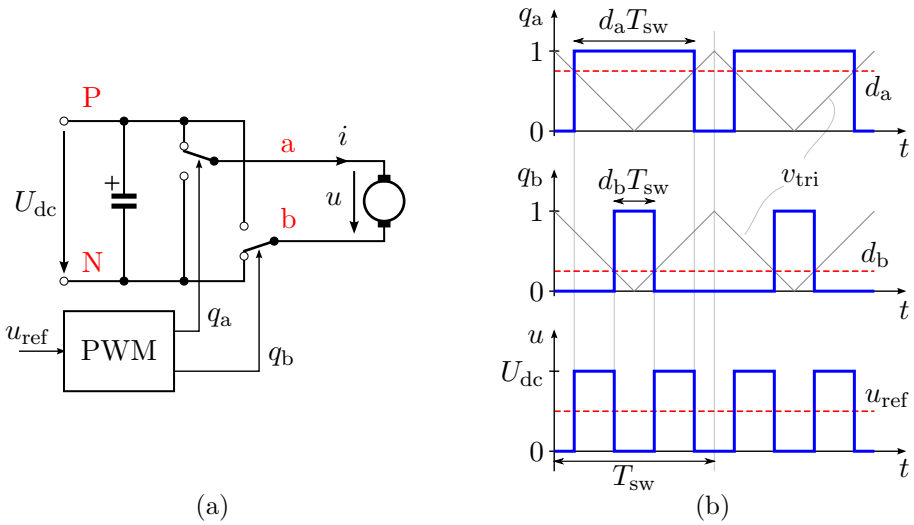
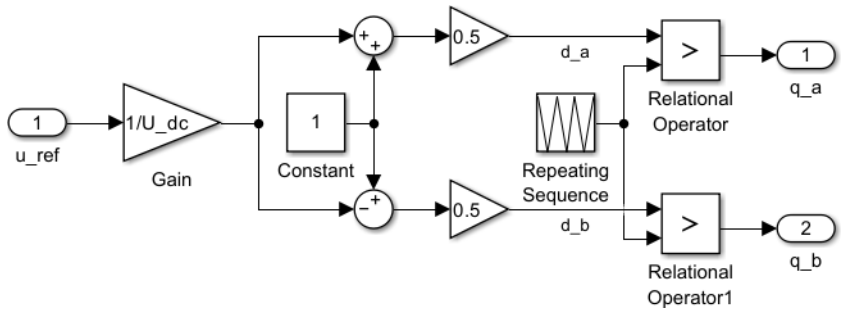


Figure 9: (a) Equivalent circuit of the four-quadrant DC-DC converter. The potentials a, b, and N are marked in the circuit. The positions of the bi-positional switches in the figure correspond to the switching states $q_{\mathrm{a}} = 1$ and $q_{\mathrm{b}} = 0$, and the output voltage is $u = U_{\mathrm{dc}}$. (b) Example waveforms in unipolar PWM.

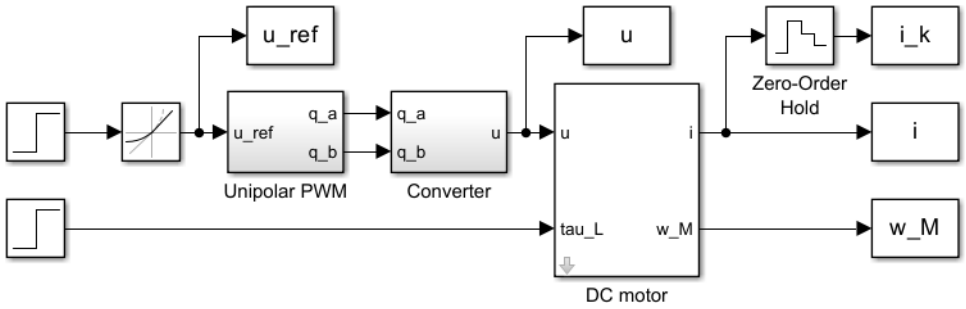
4. Augment your simulation model with unipolar PWM and converter models. Your model should function similarly to the model in Figure 10(b). Simulate the model and show the results in your report. Briefly comment on differences compared to the previous simulation, where an ideal voltage source was assumed. Submit this version of your simulation model to MyCourses.
5. Plot the waveforms of the actual current `i` and the synchronously sampled current `i_k` in the same subplot. Show also the waveform of the voltage `u`. You can plot the results using the following script:

```
subplot(2,1,1)
plot(i.time,i.data); grid on; hold on;
stairs(i_k.time,i_k.data,'r');  % Discrete signal
axis([0.15 0.1504 9 11]);       % Controls axis scaling
xlabel('Time (s)'); ylabel('Current (A)');
subplot(2,1,2)
plot(u.time,u.data); grid on;
axis([0.15 0.1504 -10 150]);
xlabel('Time (s)'); ylabel('Voltage (V)');
```

Show the results in your report and briefly comment on them.



(a)



(b)

Figure 10: (a) Simulink implementation of unipolar PWM. (b) Motor model with unipolar PWM and converter model. Specify the Repeating Sequence block by setting Time values = `T_sw*[0 0.5 1]` and Output values = `[1 0 1]`. The Zero-Order Hold block represents sampling synchronized to the PWM (set Sampling period = `T_sw/2`). You can assign the values for the variables `U_dc` and `T_sw` either via masking the subsystem (as was done in the case of the DC motor) or simply via the Matlab workspace (i.e., type `U_dc = 140` and `T_sw = 200e-6` in the workspace).

4 Cascade Control

Figure 11 shows the cascade control system. The outer loop is the speed-control loop and the inner loop is the current-control loop. PWM will be omitted in the following simulations (but the voltage saturation will be taken into account). Save the simulation model you have made, and copy it using a new name for the following changes. Remove the PWM and converter models. To speed up your simulations, open the Configuration Parameters window and set Max step size = `1e-4`.

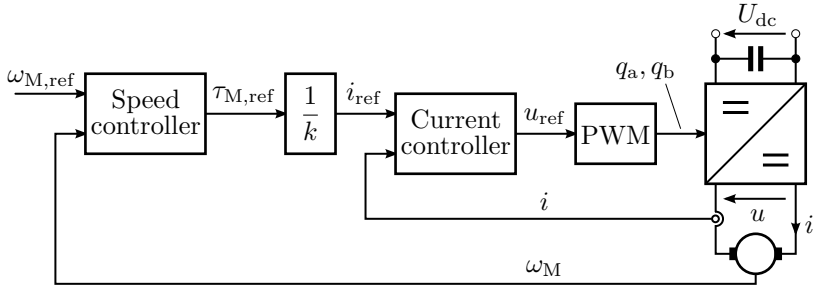


Figure 11: Cascade control system of the DC motor.

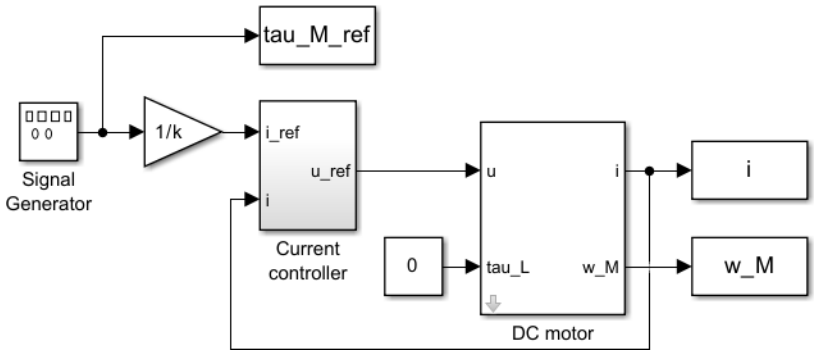
4.1 Current Control

Augment your simulation model according to Figure 12(a). Figure 12(b) shows an implementation of the 2DOF PI current controller equipped with the anti-windup scheme. Tune the controller using the following script, which you should run before starting the simulation:

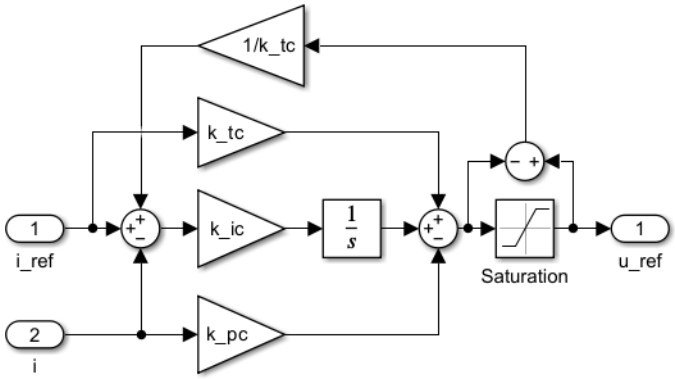
```
clear;                          % Removes variables from the workspace
%% Parameter estimates
R = 0.5;                        % Resistance
L = 2.5e-3;                     % Inductance
k = 0.35;                       % Flux factor
J = 1e-3;                       % Moment of inertia
%% Gains of the 2DOF PI current controller
alpha_c = 2*pi*400;             % Closed-loop bandwidth
k_pc = 2*alpha_c*L - R;         % Proportional gain
k_ic = alpha_c^2*L;             % Integral gain
k_tc = alpha_c*L;               % Reference feedforward gain
u_min = -140;                   % Saturation: lower limit
u_max = 140;                    % Saturation: upper limit
%% Gains of the 2DOF PI speed controller (for Section 4.2)
%alpha_s = 0.1*alpha_c;         % Closed-loop bandwidth
%k_ps = ...;                    % Proportional gain
%k_is = ...;                    % Integral gain
%k_ts = ...;                    % Reference feedforward gain
%tau_N = 7;                     % Rated torque
%tau_min = -2*tau_N;            % Saturation: lower limit
%tau_max = 2*tau_N;             % Saturation: upper limit
```

Test your model using a 1-Nm 100-Hz square-wave torque reference. The results should look similar to those in Figure 13. You can add To Workspace blocks in your model to save all necessary signals for plotting.

6. Calculate the theoretical rise time of the torque and compare it to the simulated rise time.



(a)



(b)

Figure 12: (a) Torque control. For testing the model, specify the Signal Generator block to generate the square wave with an amplitude of 1 Nm and a frequency of 100 Hz. (b) 2DOF PI current controller with anti-windup. Set `u_min` for the lower limit and `u_max` for the upper limit in the Saturation block.

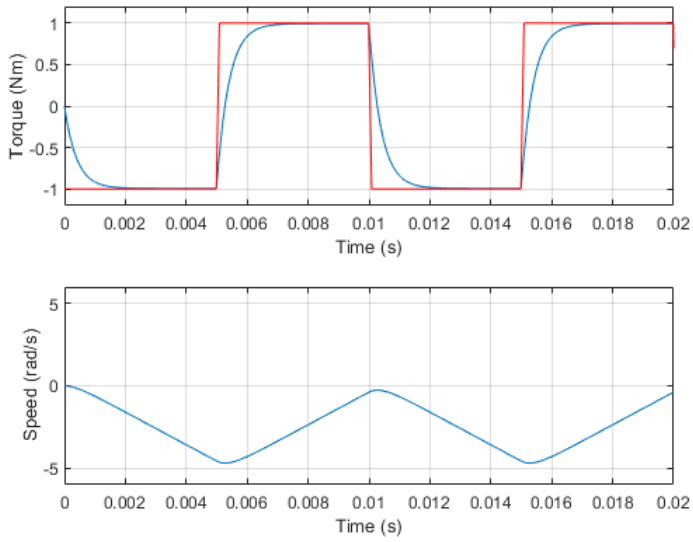


Figure 13: Testing of the torque controller with a 1-Nm 100-Hz square-wave torque reference.

4.2 Speed Control

Finally, a 2DOF PI speed controller will be implemented and tuned, in analogous manner to the current controller. Figure 14 shows an implementation of the control system.

7. Tune the speed controller of your simulation model for the closed-loop bandwidth $\alpha_s = \alpha_c/10$. Test your model using the square-wave speed reference, whose amplitude is 160 rad/s and frequency is 4 Hz. Generate the rated load torque step at $t = 0.3$ s. Show results of this simulation in your report. Show also the figures describing the main level of your simulation model and the implemented speed controller. Submit this version of your Simulink model to MyCourses (including your initialization script).
8. This problem aims to illustrate the robustness of the closed-loop control scheme against parameter errors. Generally, resistances depend on temperature (about 0.4%/K) and inductances may vary due to the magnetic saturation. Change the actual resistance R in the motor model to 150% of its original value and the actual inductance L to 70% of its original value, but do not change the values in the control system. Simulate the model. Show the results and comment on them in your report. After this problem, restore the parameter values back to their original values.
9. This problem aims to illustrate the importance of the anti-windup scheme. Remove the anti-windup in the speed controller (but do not remove the saturation of the controller output). Show results of your simulation and comment on them. After this problem, restore the anti-windup method back to the original form.
10. Parametrize the speed controller so that it becomes a regular (1DOF) PI controller, while keeping the closed-loop poles the same. Furthermore, parametrize the speed controller so that it becomes the proportional controller, while keeping the same reference-tracking performance as that of the original 2DOF PI controller. For both cases, show the simulation results and briefly comment on them.

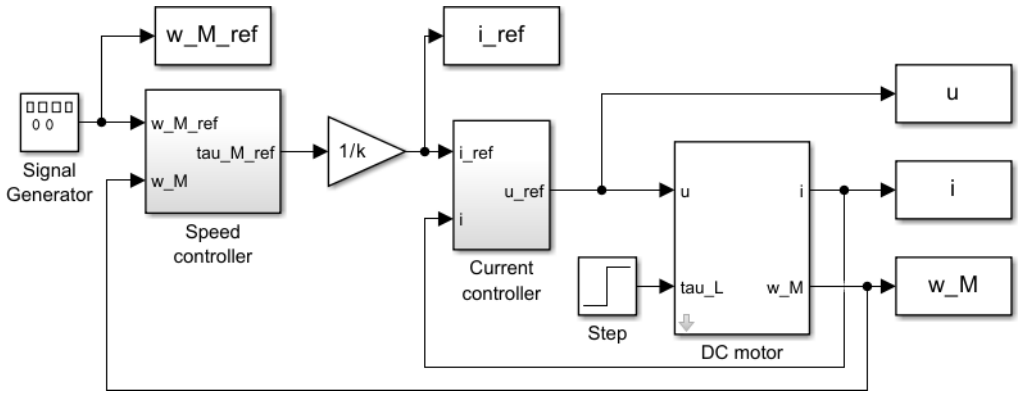


Figure 14: Cascade control. Specify the Signal Generator block to generate the square wave with an amplitude of 160 rad/s and a frequency of 4 Hz. Specify the Step block to generate the rated torque step at $t = 0.3$ s.

Give Us Feedback

In order to improve this assignment, please give us feedback. In order to estimate the student workload, we would also be happy to know how many hours did you use to do this assignment. All other comments are also welcome.