Electric Drives Experiment 4

Designing and Testing a PMDC Motor Controller

4.1 Objective

The objective of this activity is to design and implement a speed controller for a permanent-magnet DC (PMDC) motor. This activity was derived from those developed and distributed by the University of Minnesota (UMN).

4.2 Designing the PMDC Motor's Current Controller

Proportional-integral (PI) controllers will be used to control both the speed and the current (i.e., torque) of the lab's PMDC motor, with the current loop being nested inside the speed loop. The integral constant for the current control loop, k_{iI}, is given by:

$$k_{iI} = \frac{\omega_{cI} R_a}{k_{PWM}}$$

and the proportional constant for the current control loop, k_{pI}, is given by:

$$k_{pI} = \tau_e k_{iI}$$

where:

- $\omega_{cI} = 2\pi f_{cI}$ represents the crossover frequency of the current controller in radians/second
- k_{PWM} for this lab's power electronics board is V_{in}, which is set to 40 V
- R_a is the resistance of the PMDC motor's armature coil
- τ_e represents the electrical time constant, L_a/R_a
- L_a is the inductance of the PMDC motor's armature coil

For this lab, set f_{cI} = 300 Hz. Remember that a higher f_{cI} will result in a control system that updates faster, but f_{cI} also needs to be one to two orders-of-magnitude less than f_s to ensure that the switching frequency noise is well damped in the DC voltage applied to the motor. Given that the switching frequency of the lab's PWM signals f_s = 10 kHz, the value f_{cI} = 300 Hz is a reasonable compromise between the two design requirements. Use the values of R_a and L_a that your group calculated in the previous labs and the information presented above to complete Table 4.1. *Include the completed table in your lab report.*

Table 4.1: Parameters used in the PMDC motor's current control loop.

Parameter	Value
ω_{cI}	
k_{PWM}	
R_a	
L_a	
τ_e	
k_{iI}	
k_{pI}	

4.3 Implementing the Current Controller in MATLAB Simulink

Start the 32-bit version of MATLAB. After the startup sequence is complete, as indicated by a >> prompt in the Command Window, select the "Home" tab at the top of the main window and then select **New->Simulink Model** from the appropriate pulldown menu. On the resulting Simulink window, select **View->Library Browser** to see the available pre-made modules.

Double-click on the "Commonly Used Blocks" library, and then drag the "In1" module into the Simulink window to start creating your current controller. As shown in Figure 4.1, rename this input to "Ia_Reference" – this will represent the desired armature current. Copy and paste this input module (Ctrl-C, Ctrl-V), renaming the copy to "Ia_Measured".

You now need to subtract the measured armature current from the desired value, so the error can be fed into your controller. Drag and drop the "Sum" module from the Commonly Used Blocks library into your model. Double-click on the module, and change the second plus in the "List of signs:" entry to a minus, so the entry changes from "|++" to "|+-". Click "OK", and then click and drag the mouse pointer to connect the output arrows on the "Ia_Reference" and "Ia_Measured" modules to the input ports on the Sum module, as shown in Figure 4.1.

Complete the controller module as shown in Figure 4.1. You will use the Gain, Sum, Saturation, and Out1 modules from the Commonly Used Blocks library, along with the "Integrator Limited" module from the "Continuous" library. Name and connect the blocks as shown in the figure. Start the line at the k_{iI} block when connecting it to the error signal. Enter your calculated values for k_{iI} and k_{pI} in the appropriate blocks (double-click on a block to change its value). Expand those blocks so that your numeric entries are visible - **your values will be quite different from those shown in the figure**, up to an order-of-magnitude. If your values are a few or more orders-of-magnitude different, check your calculations, and ask the instructor or TA for help if you can't find any error.

Set the upper limit in both the Integrator and the Saturation blocks to +1, and set the lower limit in both blocks to -1. This will result in a maximum of 40 V and a minimum of -40 V being applied to the PMDC motor's armature coil. Note that *Vc_i* represents the control voltage output from the current control loop. *Save an image of your Simulink model for your lab report.* Save a copy of your Simulink current controller model in the ".mdl" format, in a desktop folder called "Exp4".

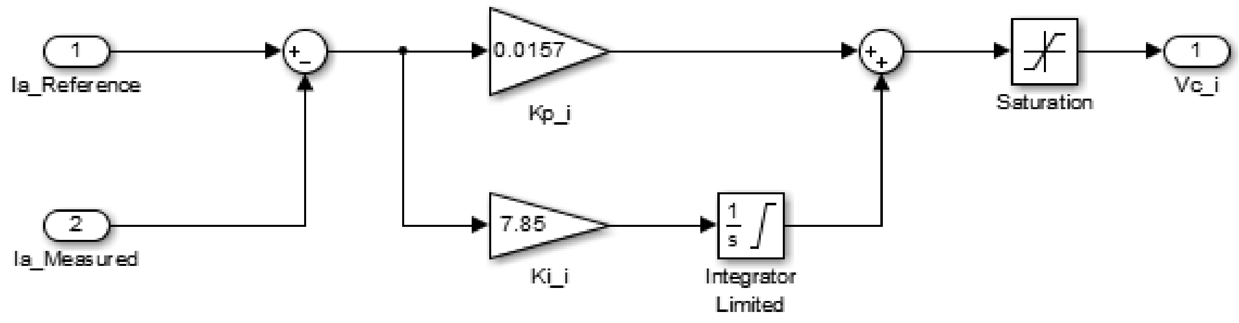

Figure 4.1: Example MATLAB Simulink model of a PMDC motor's current controller.

4.4 Designing the PMDC Motor's Speed Controller

A PI design will also be used for the PMDC motor's speed controller. In this case, the integral constant is determined by the following equation:

$$k_{i\Omega} = \frac{J_{eq}\,\omega_{c\Omega}^2}{k_T\sqrt{1+\left(\tan(-120°)\right)^2}} = \frac{J_{eq}\,\omega_{c\Omega}^2}{2k_T}$$

and the proportional constant, $k_{p\Omega}$, is given by:

$$k_{p\Omega} = \frac{k_{i\Omega}\tan(-120°)}{\omega_{c\Omega}} = \frac{k_{i\Omega}\sqrt{3}}{\omega_{c\Omega}}$$

where the subscript 'Ω' refers to the speed controller, and:

- J_{eq} represents the equivalent moment-of-inertia for the Motorsolver PMDC system
- $\omega_{c\Omega} = 2\pi f_{c\Omega}$ represents the crossover frequency of the speed loop in radians/second
- k_T is the PMDC motor's torque constant

In addition, these equations were designed so that the control system would have an ideal phase of -120° at the crossover frequency (i.e., a phase margin of 60° above the unstable -180°). Since the current controller is nested within the speed controller, the speed controller should update at a slower rate (one to two orders-of-magnitude slower); otherwise, the two controllers may update in ways that cause harmful oscillations or other negative side effects. To meet this design requirement, set $f_{c\Omega} = f_{ci} / 10 = 30$ Hz.

Use the values of J_{eq} and k_T (= k_E for the PMDC motor) that your group measured in previous labs along with the information presented above to complete Table 4.2. Don't worry if you're using a different motor in this lab. *Include the completed table in your lab report.*

Table 4.2: Parameters used in the PMDC motor's speed control loop.

Parameter	Value
J_{eq}	
$\omega_{c\Omega}$	
k_T	
$k_{i\Omega}$	
$k_{p\Omega}$	

4.5 Implementing the Speed Controller in MATLAB Simulink

Open the MATLAB Simulink model of your current controller. Click and drag the mouse across the complete model to select all of its components. Right-click on any of the blocks within your model, and select "Create Subsystem from Selection". This action will create a new MATLAB Simulink module representing your PI controller. Rename your new module from the default "Subsystem" to "PI_Current_Controller", and expand the module so that the input and output variable names are readable. Delete the input and output blocks connected to the PI current controller module (use Ctrl-X), along with the lines which connected the modules, so that you just see the module called PI_Current_Controller.

Select the PI_Current_Controller module, and press Ctrl-C to copy and then Ctrl-V to paste it. Rename the copy to "PI_Speed_Controller" and then double-click on the module to access its components. As shown in Figure 4.2, rename the inputs to "Wm_Reference" and "Wm_Measured" to represent the desired and actual motor speeds in radians/second. Rename the PI constants to "Kp_w" and "Ki_w" to emphasize that this is the speed controller, and update these constants with the $k_{p\Omega}$ and $k_{i\Omega}$ values that you calculated for Table 4.2. Your values will likely be almost an order-of-magnitude larger than those shown in the figure; check for errors if your calculations if your values are a couple orders-of-magnitude or more different than those shown in the figure. Rename the output control variable to "Ia_Reference", since the output of this controller will be the reference input to the current controller. Change the limits of **both** the Integrator Limited and Saturation blocks to +/- 4, to ensure a safe distance from the 5 A current limit of the lab bench's power supply. Check that your MATLAB Simulink model of the PMDC motor's speed controller is similar to the one shown in Figure 4.2, although of course your numbers will be different. *Save an image of your Simulink model for the speed controller.*

Select the backwards arrow in the Simulink toolbar to return to your main Simulink model.

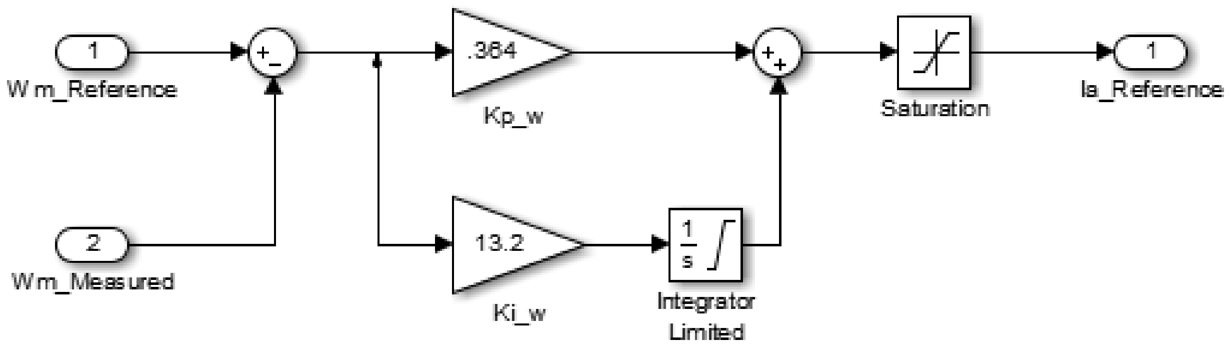

Figure 4.2: Example MATLAB Simulink model of a PMDC motor's speed controller.

4.6 Implementing the PMDC Motor's Equations in Simulink

To create a block diagram representing the PMDC motor's behavior, the main equations associated with the PMDC motor (used in previous labs) are first converted to the Laplace domain. For example, the equation for the armature voltage:

$$v_a(t) = e_a(t) + R_a i_a(t) + L_a \frac{di_a(t)}{dt}$$

transforms into the following equation in Laplace domain:

$$V_a(s) = E_a(s) + R_a I_a(s) + sL_a I_a(s)$$

where the principle that a derivative in the time domain equates to multiplication by $s = j\omega$ in the Laplace domain was used. Solving this equation for the current, which is the variable we want to control, results in the following:

$$I_a(s) = \frac{V_a(s) - E_a(s)}{R_a + sL_a} \tag{1}$$

Similarly, start with the equation for the PMDC motor's electromagnetic torque:

$$T_{em}(t) = T_L(t) + B\omega_m(t) + T_f + J_{eq}\frac{d\omega_m(t)}{dt}$$

Then ignore the frictional losses, convert the equation to the Laplace domain, and solve for the speed (which we want to control) to obtain the following:

$$\omega_m(s) = \frac{T_{em}(s) - T_L(s)}{sJ_{eq}} \tag{2}$$

The behavior represented by equations (1) and (2) above can be put into block diagram form as shown in Figure 4.3, where the substitutions $E_a(s) = k_E\omega_m(s)$ and $T_{em}(s) = k_T I_a(s)$ were used. **Ensure that you understand the conversion from time to Laplace domain as well as the conversion between equations (1) & (2) and the block diagram.** (Potential exam questions…)

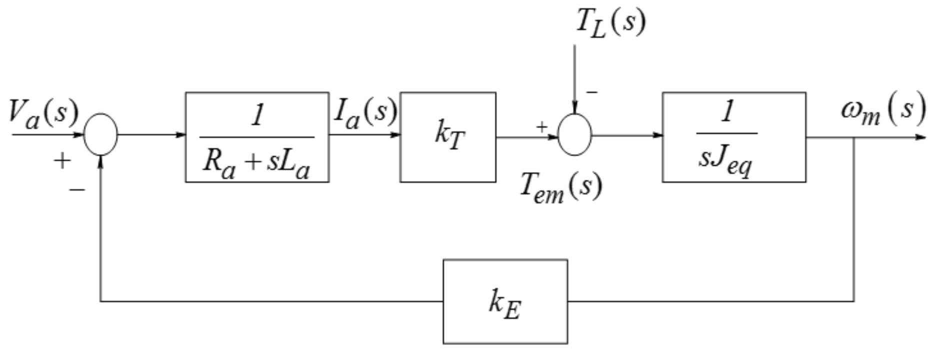

Figure 4.3: Block diagram of the PMDC motor's transfer function (UMN, Mohan)

Download the "PMDC_motor.mdl" file from BbLearn to your desktop Exp4 folder, and rename the file if necessary to remove any added parentheses. Double-click on the PMDC_motor.mdl file to load it into Simulink. It should look similar to Figure 4.4. This model is an implementation of Figure 4.3, with the current equation modified slightly as follows:

$$V_a(s) = E_a(s) + R_a I_a(s) + sL_a I_a(s) \quad \text{becomes} \quad I_{a,new}(s) = \frac{V_a(s) - E_a(s) - I_{a,old}(s)R_a}{sL_a}$$

Double-click on the appropriate blocks of the PMDC motor's Simulink model to update the L_a, R_a, J_{eq}, k_T, and k_E values to those that you measured in previous experiments. Do not worry if you are using a different motor than you did when measuring those values. Resize the blocks so that your numerical entries are shown. *Save an image of the PMDC motor's Simulink model which shows the five values that you entered.*

Create a module which represents the PMDC motor: Click and drag the mouse to highlight all components of the PMDC motor's Simulink model; right-click on a block within the model and select "Create Subsystem from Selection"; rename the module from "Subsystem" to "PMDC_Motor"; and delete the input and output blocks and lines connecting to the module. Copy and paste your PMDC motor module into the Simulink window containing your PI controller modules. Keep Simulink open.

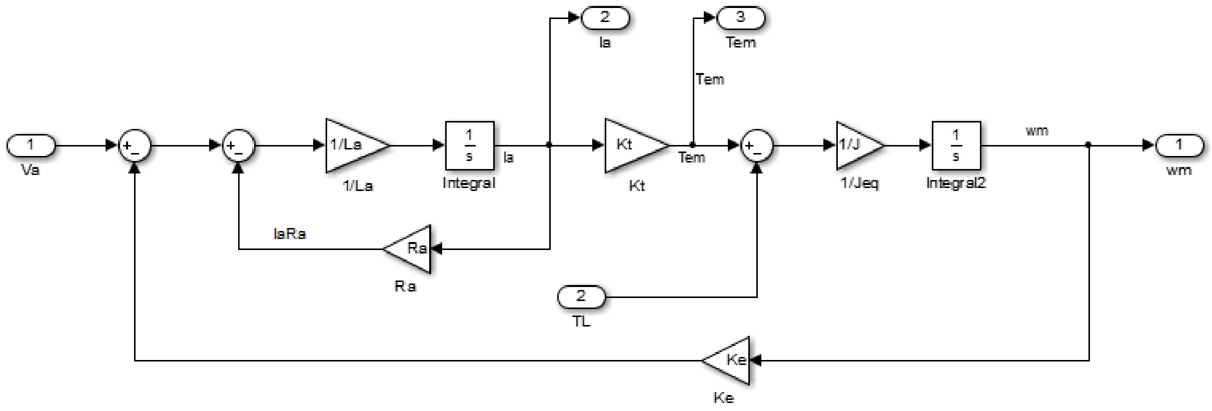

Figure 4.4: MATLAB Simulink model of the PMDC motor equations.

4.7 Completing and Analyzing the MATLAB Simulink Model of the Entire System

Use the PI speed and current controller Simulink modules that you created, along with the PMDC motor module that you modified, to build the cascaded control system shown in Figure 4.5. Use the "Constant" module from the "Commonly Used Blocks" library to enter the reference speed (100 rad/sec) and load torque (0.12, to approximate the losses we observed in lab under no-load conditions). Enter "Step" into the Simulink Library Browser's search utility, and use this module to increase the desired speed by 100 rad/second at t = 2.0 sec (Step time = 2, Initial value = 0, Final value = 100). Don't forget the k_{PWM} gain block (set to 40), which models the effect of the lab's power electronics. The scope, found under the "Commonly Used Blocks" library, will allow you to plot the system's variables to analyze the behavior of the control system. Use a "Mux" block to display both the desired and actual values of the controlled variable on the same scope. Double-check your model with Figure 4.5.

Save your model of the PMDC motor's cascaded control system. Double-click on each scope to bring up its graphing window. In **each** scope window (unfortunately), select the "Parameters" toolbar icon, and select the "History" tab on the resulting pop-up window. Uncheck the "Limit data points to last" option, as this would truncate your graphs too much. Select "OK", and remember that this step must be done on each scope window.

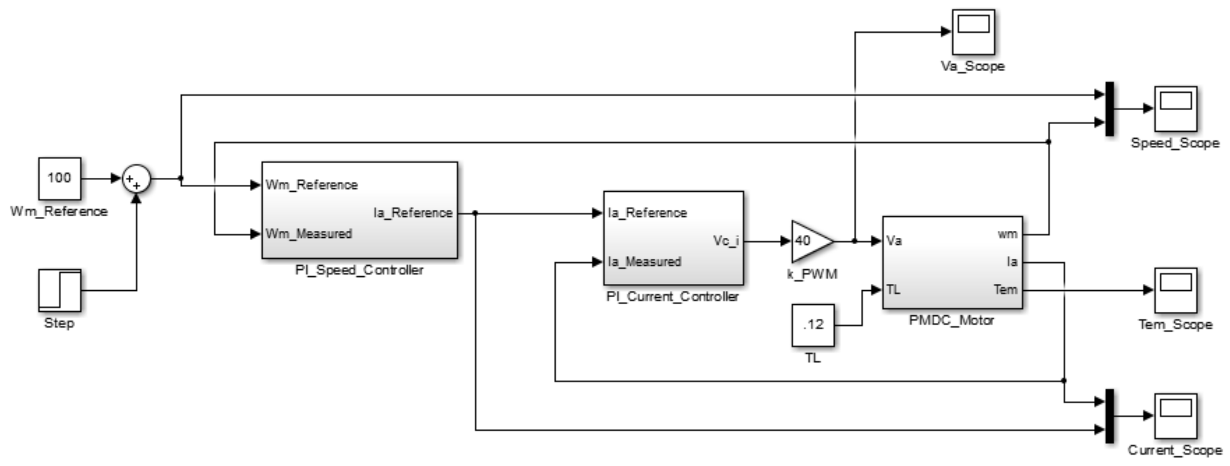

Figure 4.5: MATLAB Simulink model of the PMDC motor's cascaded control system.

The scope is designed to react quickly while displaying signals real-time. As a compromise for this speed, the capability to add descriptive information such as axis labels, legends, and so forth is limited. Note for future reference that you could output the scope's data to the MATLAB command window and create more descriptive figures from there, but for now just carefully describe your plots when you save them in MS Word.

Set the "Simulation stop time" in the main Simulink toolbar (the open entry box) to 10. Press the "Run" icon (the green arrow) in the Simulink toolbar to simulate your model. Select the "Autoscale" icon in each scope to scale the data. If the plots of the motor's desired and actual speeds do not match well, check your model for errors. Common problems include entering L or J instead of $1/L$ or $1/J$, or transposing the proportional and integral constants. If you find no "simple" errors and your plots are oscillating wildly between large positive and negative values, try reducing the current controller's proportional constant by a factor of two or more. If problems continue, compare your PI controllers' values with those of other groups. If problems continue, ask the instructor or TA for assistance. After fixing any errors, *save an image of your Simulink model for the cascaded PMDC motor control system*.

Save an image of all four scope plots. Compare the simulated motor armature current, armature voltage, and electromagnetic torque to the values you measured in previous labs (look at your torque-speed data for no-load conditions, in cases where the speeds were near but not exactly equal to 100 and 200 radians/second). If the simulated and measured values don't match well (say, within 20%), double-check your simulated motor values and compare your controller values to those of other groups (a factor of two difference is ok). Ask the instructor or TA for assistance if your model still doesn't seem to match the actual system well.

Why do the motor's torque and current values change significantly when the speed suddenly changes? Do you see any effect in the plots of the limits that you set in the controllers? Explain. Analyze the effectiveness of the control systems. For example, do the systems produce zero steady-state error (put another way, does the actual current equal the desired current after the transients have settled down, and are the desired and actual speeds equal in steady-state conditions)? When the step change is applied to the reference speed at t = 2.0 seconds, how do the current and speed controllers respond in terms of rise time, overshoot, oscillation, and settling time? In what ways might the controllers' performance be improved? **Save your model.**

4.8 Testing the Control System with the Electric Drives Lab Equipment

Open the “Exp4.mdl” file downloaded from BbLearn, and insert your proportional and integral constants into both the speed and the current controllers. Note that the measured values being input to both controllers are now coming from actual sensors via the DS1104 board. Also note that the simple PWM constant used in the mathematical analysis and earlier simulations has now been replaced by the two-pole switch-mode converter, which uses the PWM generating capabilities of the DS1104 board and drives the transistors on the power electronics board. Also notice that a repeating stepped waveform with the same speeds used in your simulation is being used as the speed reference, and reset signals have been added to both controllers.

Save your modified version of Exp4.mdl, and type Ctrl-B to build the C-code and an executable of the Simulink model (unless, of course, you’d rather write the controller’s C-code yourself ☺). When the messages in MATLAB’s control window indicate that the executable has successfully uploaded to the DS1104 board, open the ControlDesk software. Create a “New Project + Experiment”, using the variable definitions from the “exp4.sdf” file that you just created in MATLAB.

Create a plotter, and drag-and-drop the reference and actual current data (easiest to find under PI_Current_Controller, Ia_Reference->Out1 and Ia_Measured->Out1) so that they share the same y-axis. Change the plot’s time duration to at 10 seconds (accessed via the “Measurement Configuration” tab, Duration Triggers, Duration Trigger 1, then the “Duration” entry box). Create a second plotter for the reference and actual speed values.

Create a checkbox (Check Button) for the PI_Integrators_OnOff->Value variable, accessed at the top of the Model Root’s variable list. Right-click on the checkbox, and select “Instrument Properties”. In the Properties window, change the “On-Value” to 0 and the “Off-value” to 1 (click on the number to change it). Create another checkbox for the Transistors_OnOff variable, and change its On-Value to 0 and its Off-Value to 1. Save the project.

As in previous labs, obtain the specialized electric drives equipment from the lab cabinet.

- Set the DC power supply’s output to 40 V when no cables are connected to it, and then turn the power supply off.
- Use banana cables to connect the ‘+’ and ‘-‘ ports on the DC power supply to the +42V and GND ports on the inverter board, respectively.
- Use banana cables to connect the “Phase A1” port on the inverter board to the **black** connector on the MotorSolver’s DC Generator. Similarly, connect “Phase B1” to the DC Generator’s **red** connector.
- Use the 8-pin encoder cable to connect the DC Generator’s encoder port to the CP1104’s “Inc1” port.
- Connect the “Curr A1” port on the inverter board with the “ADC-5” port on the CP1104 board.

- Plug in the +/- 12 V power supply, and connect it to the inverter board's "analog power" connection, but leave the associated toggle switch in the off position.
- Connect the 37-pin cable between the inverter board and the CP1104's "Slave I/O PWM" port.
- **As the last step**, connect the CP1104's black cable to the DS1104 board (using the port at the back of the lab computer). Tighten the screws to ensure that the cable is well connected.

Select the "Start Measuring" icon in ControlDesk, and ensure that both the PI_Integrators_OnOff Transistors_OnOff boxes are checked, so that the integration part of the controllers and the transistors are both working. Turn on the toggle switch associated with the analog power supply for the power electronics board, and turn on the 40 V power supply. Your motor should start spinning. If not, check all of your hardware connections, press the "reset" button on the inverter board, and try toggling the transistor checkbox off and on again. Ask the TA or instructor if problems continue.

Save images which show the speed and current plots for at least one cycle of the desired speed's square waveform. Uncheck the PI_Integrators_OnOff box, which will represent the system's behavior without the integral capacity of the controllers, and *save another set of plots.* (You may need to press the "Reset" button on the power electronics board if the motor experiences a current fault during this transition.) In most cases, the motor's speed will not be able to reach the desired value of 200 radians/second without the integral controller (that is, there will be a non-zero steady-state error for at least part of the desired waveform cycle). For some motors and controller values, the actual speed won't appear to follow the desired speed waveform at all. *Discuss the differences between the two sets of plots and how that relates to the effectiveness of the controller's integral components*.

Uncheck the "Transistors_OnOff" box to stop the power flow through the transistors. Then turn off the 40 V power supply, followed by the analog power's toggle switch. Then press the "Stop Measuring" button in ControlDesk and close the software.

4.9 Shutdown Procedures

Take care that you do not accidentally drape any cables or other items across the inverter board while power is still being applied to the board. Turn off and remove the three power sources first to reduce the probability of damaging the board.

- **Turn off the high-power DC power supply first, without reducing its output voltage.**
- Turn off the toggle switch associated with +/- 12V power supply, and unplug this power supply.
- Remove the black cable from the DS1104 PC board (at the back of the lab computer).
- Remove the ribbon cable from the inverter board and the CP1104 board.

- Remove the two BNC-BNC cables and the encoder cable.
- Remove all banana cables and store them at the lab benches as usual.
- Replace the inverter board inside the protective electrostatic cover.
- Carefully return all of the specialized electric drives components to a table near the lab's cabinet. The TA or the instructor will place the equipment back into the cabinet.
- Check that all of the test bench equipment is turned off, and leave the test bench area clean and organized. **Your lab grade will be lowered if you are not careful with the lab equipment and/or if you leave trash in the test bench area.**

4.10 Lab Report

The group of students at each lab station should submit a lab report. The report should focus on presenting and analyzing the data collected in this lab, although your instructor or TA may provide additional questions or requirements for your report. The requested data were often emphasized in the procedures with italicized text. The report should have section headings to help organize your information. Each figure should have a label and a brief description placed below the image, as demonstrated in these procedures, and each figure should be referenced in the text. Your analysis should be written in prose (i.e., complete sentences); the report should not just contain a collection of images and numbers. Think about writing the report such that you can use it later in the semester as a study guide. Show the equations or processes used to calculate values that were not directly measured. Your instructor or TA will announce the due date and method for submitting your report.

Checklist of items requested in these lab procedures:

- ☐ The completed table of parameters used in the PMDC motor's current control loop.
- ☐ An image of the Simulink model representing your PI current controller.
- ☐ The completed table of parameters used in the PMDC motor's speed control loop.
- ☐ An image of the Simulink model representing your PI speed controller.
- ☐ An image of the PMDC motor's Simulink model which shows the values of the five motor parameters that you entered.
- ☐ An image of the Simulink model representing the PMDC motor's cascaded control system.
- ☐ Save images of all four MATLAB scopes (voltage, speed, torque, and current) after simulating the cascaded control system.
- ☐ Compare the "measured" armature current, armature voltage, and electromagnetic torque in the MATLAB simulation to the values you obtained in previous labs for speeds near 100 and 200 radians/second under no-load conditions.

- ☐ Why do the torque and current values change significantly when the speed suddenly changes?
- ☐ Do you see any effect in the MATLAB scope plots of the limits that you set in the controllers? Explain.
- ☐ Analyze the effectiveness of the cascaded control system.
- ☐ Save images from ControlDesk which show the speed and current plots for at least one cycle of the desired speed’s square waveform when your controllers are completely enabled.
- ☐ Save images from ControlDesk which show the speed and current plots for at least one cycle of the desired speed’s square waveform when the integral controllers are turned off.
- ☐ Discuss the differences between the ControlDesk plots which show the motor’s behavior to sudden changes with and without the integral controllers. Discuss the effectiveness of the controllers in general.