DSP Based
Electric Drives Laboratory

User Manual

Department of Electrical and Computer Engineering

University of Minnesota

Drives Board Familiarization

1 Introduction

The drives board which we use in the Electric Drives Laboratory has been designed to enable us to perform a variety of experiments on AC/DC machines. The main features of the board are:

- Two completely inependent 3-phase PWM inverters for complete simultaneous control of two machines
- 42 V dc-bus voltage to reduce electrical hazards
- Digital PWM input channles for real-time digital control
- Complete digital/analog iterface with dSPACE board

2 Board Familiarization

The basic block diagram of drives board is shown in Fig. 1 and the actual drives board is shown in Fig. 2. Please note that various components on the board are indicated in Table. 1.

2.1 Inverters

Each 3-phase inverter uses MOSFETs as switching devices. The 3-phase outputs of the first inverter are marked A1 (D-6 in Fig. 2), B1 (E-6 in Fig. 2), C1 (F-6 in Fig. 2) and those of the second inverter are marked A2 (I-6 in Fig. 2), B2 (K-6 in Fig. 2), C2 (L-6 in Fig. 2).

2.2 Signal Supply

± 12 volts signal supply is required for the isolated analog signals output form the dirves board. This is obtained from a wall-mounted isolated power supply, which plugs into the DIN connector J90 (B-2 in Fig. 2). Switch S90 (C-2 in Fig. 2) controls the signal power to the board. The green LED D70 (C-2 in Fig. 2) indicates if the signal supply is available to the board. Fuses F90 (C-2 in Fig. 2) and F95 (B-2 in Fig. 2) provide protection for the $+12$ V and -12 V supplies respectively. Please note that the green LED indicates the presence of only the $+12$ V supply. Please note that turning off S90 will not stop the PWM signals from being gated to the inverters.

The power supply for the 3-phase bridge drivers for the inverters is derived from the DC Bus through a flyback converter (A-2 in Fig. 2).

Table 1: Locations of components on drives board

No.	Component	Ref. Des.	Location in Fig. 2
1	Terminal +42	J1	A-4
2	Terminal GND	J2	A-3
3	Terminal PHASE A1	J3	D-6
4	Terminal PHASE B1	J4	E-6
5	Terminal PHASE C1	J5	G-6
6	Terminal PHASE A2	J6	J-6
7	Terminal PHASE B2	J7	K-6
8	Terminal PHASE C2	J8	L-6
9	DIN connector for $\pm$12 V signal supply	J90	B-2
10	Signal supply switch	S90	C-2
11	Signal supply +12 V fuse	F90	C-2
12	Signal supply $-$12 V fuse	F95	B-2
13	Signal supply LED	D70	C-2
14	MOTOR1 FAULT LED	D66	D-2
15	MOTOR2 FAULT LED	D67	L-2
16	DIGITAL POWER LED	D68	I-2
17	MAIN POWER LED	D69	B-3
18	Inverter 1		D-3 to G-4
19	Inverter 1		I-3 to L-4
20	DC Link capacitor of Inverter 1	C1	B-5
21	DC Link capacitor of Inverter 2	C2	G-5
22	Driver IC IR2133 for Inverter 1	U1	E-2
23	Driver IC IR2133 for Inverter 2	U3	J-2
24	Digital Supply Fuse	F2	G-1
25	dSPACE Input Connector	P1	H-1 and I-1
26	RESET switch	S1	L-1
27	Phase A1 current sensor (LEM)	CS2	C-5
28	Phase B1 current sensor (LEM)	CS3	D-5
29	Phase A2 current sensor (LEM)	CS5	H-5
30	Phase B2 current sensor (LEM)	CS6	J-5
31	DC link current sensor (LEM)	CS1	L-5
32	VOLT DC	BNC5	B-4
33	CURR A1	BNC1	B-3
34	CURR B1	BNC2	C-3
35	CURR A2	BNC3	H-3
36	CURR B2	BNC4	I-3

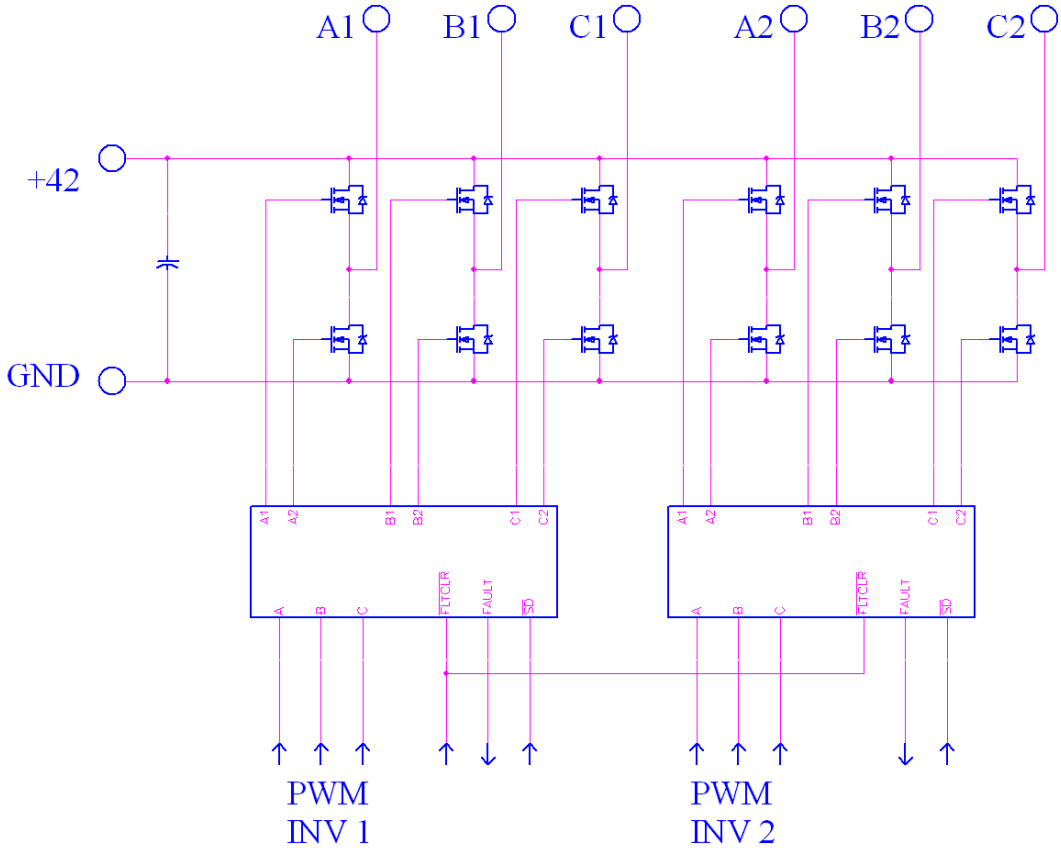


Figure 1: Block Diagram of Electric Drives Board

2.3 Voltage Measurement

Test points are provided to observe the inverter output voltages. BNC connector VOLT DC (B-4 in Fig. 2) has been provided to sense the DC bus voltage. To measure the DC bus voltage,

- Connect a BNC cable to VOLT DC BNC connector.
- The scaling factor of input voltage is 1/10.

2.4 Current Measurement

LEM sensors are used to measure the output current of the inverters. Only A and B phase currents are sensed. The C phase current can then be calculated using the current relationship $I_a+I_b+I_c = 0$, assuming that there is no neutral connection for the machines. The calibration of the current sensor is such that for 1 A current flowing through the current sensor, output is 0.5 V.

To measure the output current of phase A of inverter 1,

- Connect BNC connector to CURR A1 (B-3 in Fig. 2).

To measure the output current of phase B of inverter 1,

- Connect BNC connector to CURR B1 (C-3 in Fig. 2).

To measure the output current of phase A of inverter 2,

- Connect BNC connector to CURR A2 (H-3 in Fig. 2).

To measure the output current of phase B of inverter 2,

- Connect BNC connector to CURR B2 (I-3 in Fig. 2).

2.5 Inverter Drive Circuit

The inverters are driven by 3-phase bridge drivers (IR2133). The PWM inputs are isolated before being fed to the drivers.

2.6 PWM/Digital Signals

PWM and other digital signals for the board are to be given to the 37-pin DSUB connector (H-1 in Fig. 2). For pinout of the connector, see Table 2.

2.7 Fault Protection

The Drives Board consist of overcurrent protection for each inverter. An overcurrent fault occuring on inverter 1 is indicated by red LED "MOTOR FAULT 1" (D-2 in Fig. 2), while that of inverter 2 is indicated by red LED "MOTOR FAULT 2" (L-2 in Fig. 2). Each time a fault occurs, reset the fault using the "RESET" switch (L-1 in Fig. 2) on the board. All the faults are reset by this switch.

Table 2: 37-pin DSUB Connector

No.	Pin Number	Description
1	GND_Digital	Digital ground
2	FAULT 1	Inverter 1 Fault output. Fault Signal high
3	NC	Not Connected
4	GND_Digital	Digital ground
5	NC	Not Connected
6	GND_Digital	Digital ground
7	PWM A1	A phase PWM signal of Inverter 1
8	PWM B1	B phase PWM signal of Inverter 1
9	PWM C1	C phase PWM signal of Inverter 1
10	PWM A2	A phase PWM signal of Inverter 2
11	PWM C2	C phase PWM signal of Inverter 2
12	GND_Digital	Digital ground
13	GND_Digital	Digital ground
14	GND_Digital	Digital ground
15	GND_Digital	Digital ground
16	$\overline{SD1}$	Shutdown signal for Inverter 1
17	$\overline{FLTCLR-IN}$	Clear Fault signal
18	VCC	
19	VCC	
20	GND_Digital	Digital ground
21	FAULT 2	Inverter 2 Fault output. Fault Signal high
22	NC	Not Connected
23	NC	Not Connected
24	NC	Not Connected
25	GND_Digital	Digital ground
26	NC	Not Connected
27	NC	Not Connected
28	NC	Not Connected
29	PWM B2	C phase PWM signal for Inverter 2
30	GND_Digital	Digital ground
31	GND_Digital	Digital ground
32	GND_Digital	Digital ground
33	GND_Digital	Digital ground
34	NC	Not Connected
35	$\overline{SD2}$	Shutdown signal for Inverter 2
36	GND_Digital	Digital ground
37	GND_Digital	Digital ground

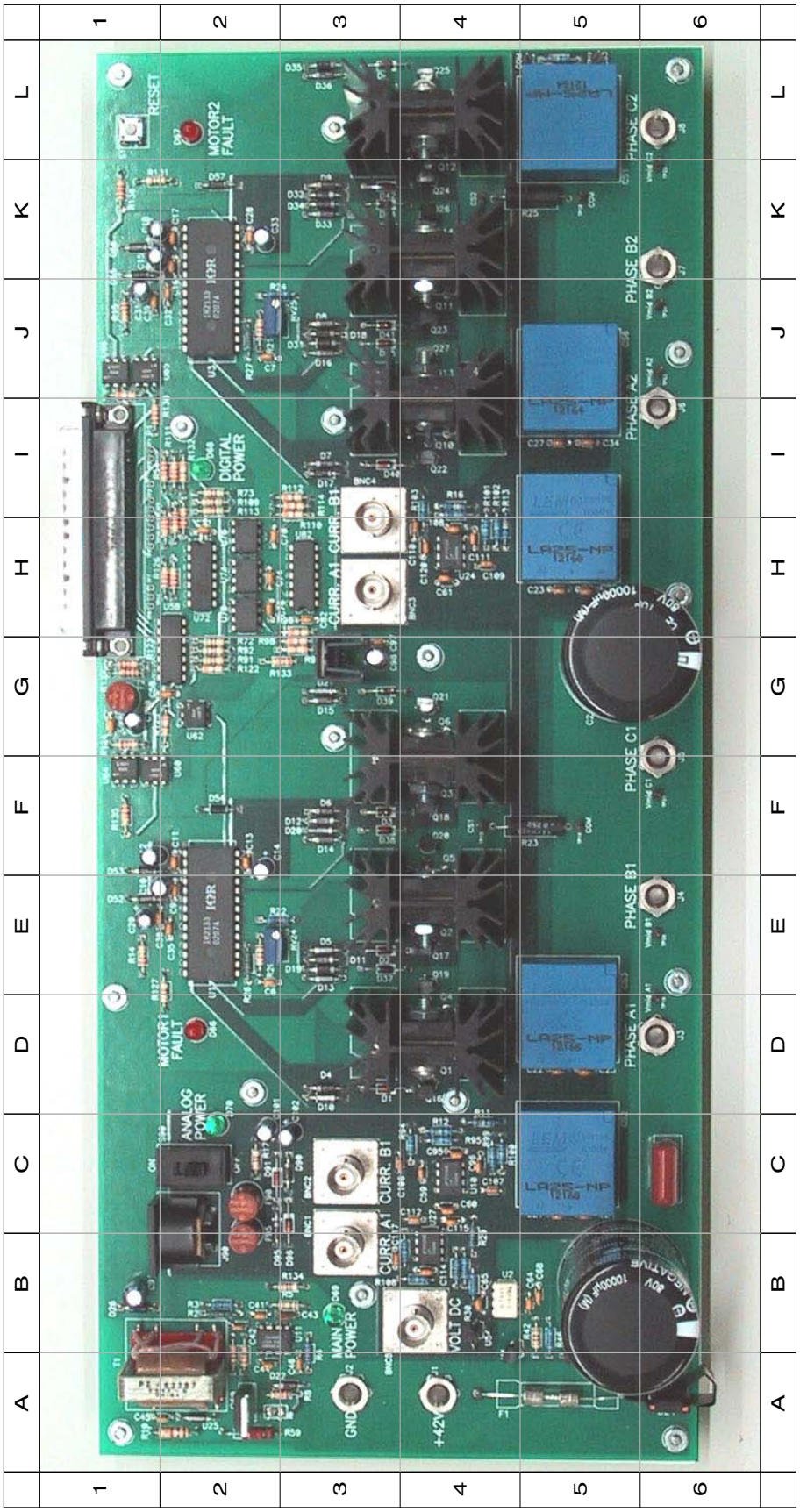


Figure 2: Electric Drives Board

Experiment 1

Building A simple model in SIMULINK

1.1 Introduction

Mathematical modeling of electric machines and drives involves solving a set of differential equations. Mathematical tools like Matlab helps in solving these differential equations in a fast and easy way. Matlab also contains a modeling tool SIMULINK, which helps to pose the problem in a graphical way using interconnected blocks, and capability to visualize the solution to the set of differential equations using graphs and plots. In fact, many real-time systems like dsp now comes with an interface to SIMULINK by which they can convert the SIMULINK block-set to a machine code that can be run on a DSP-based system. This greatly reduces the development and prototyping time for a variety of drive systems.

In this experiment, we will try to learn to use SIMULINK so that we can

- Build a simple model using SIMULINK blocks.
- Simulate the model to see the performance of system in different scenarios.

1.2 Creating a Model in Simulink

Using SIMULINK, we will develop a simple continuous time system to understand the mechanical interactions rotating system consisting of motor and load. Follow the steps below,

- Create a folder expt1.

- Start MATLAB 6.5 from the Start menu.
- At the top of the screen you will see a box that lists the Current Directory. This is your Path Browser and needs to be changed to the folder that you have just created (i.e expt0).
- Type **Simulink** at the prompt line. A new window will pop up that contain the various Libraries provided by Simulink and dSPACE.

We are trying to develop a simple rotating mechanical system model for which following equations apply.

$$T_J = T_M - T_L \tag{1.1}$$

$$J_{eq} = J_M + J_L \tag{1.2}$$

$$\alpha = \frac{T_J}{J_eq} \tag{1.3}$$

$$\omega_m = \int \alpha.dt \tag{1.4}$$

$$\theta = \int \omega_m.dt \tag{1.5}$$

For our example, we will use $J_{eq} = 0.058kg.m$

- To create the above model in Simulink, first create a New Model from the File menu.
- To add blocks to your Simulink model you simply click on the block you want in the Library window and drag it to your model and drop it. For our model we will need the following blocks.
  - One sum block from the Math library.
  - One gain block from the Math library.
  - Two slider gains from the Math library
  - Two integrators from the Continuous library.
  - Two constants from the Source library.
  - One mux from the Signals and Systems library.
  - One scope from the Sinks library
- In many cases the Simulink blocks have properties that can be modified by double clicking on them. Make the following modifications.
  - Change the sum block so that is has $|+-$ instead of $|++$.

- Change the bounds on the slider gains to -10 and 10. Set the current values for one block to 5 and the other one to 0.
- Change the mux to 3 inputs.
- Change the value of the gain to $\frac{1}{J_{eq}} = \frac{1}{0.058}$.

- To connect the blocks, click on an input or output arrow and drag it to the other input or output you would like it to go to. Make the connections as stated in the equations for the rotating mechanical system
- To rename blocks click on their current names and change them. To name connections double click on them and a text block will come up.
- After the connections are made the model should look like Fig. 1.1

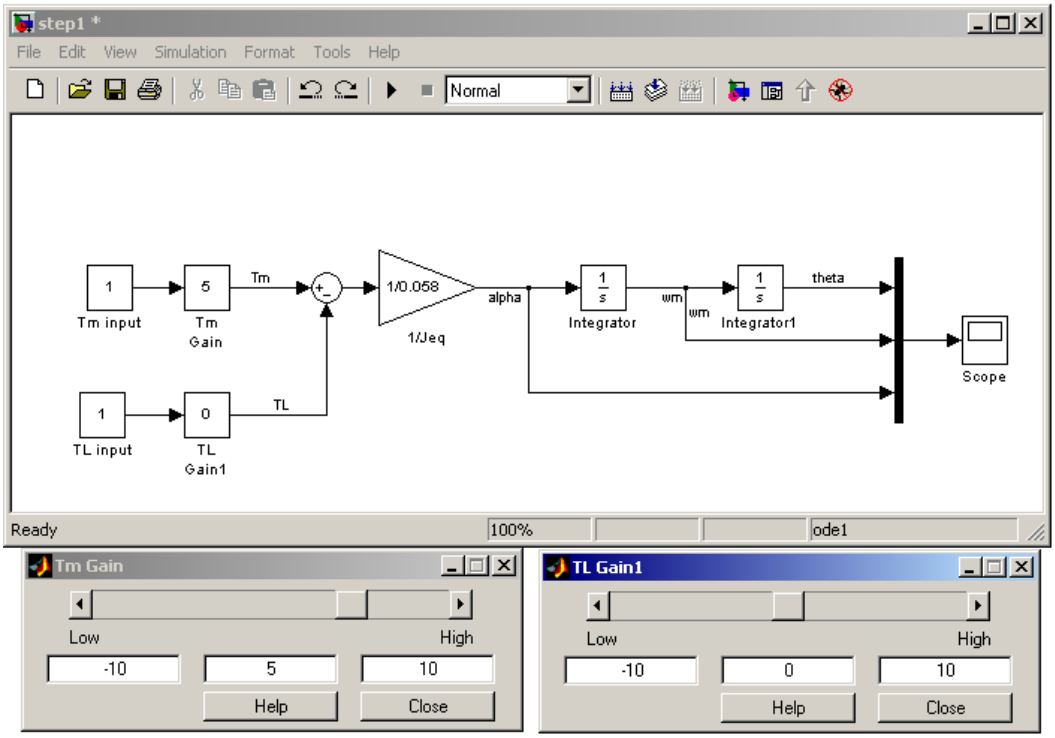


Figure 1.1: Simple Mechanical System Model

- The next step is to run the simulation. Before running the simulation, do the simulation parameters settings. To do this go to the Simulation menu and select simulation parameters. Set the parameters as shown in Fig. 1.2
- Once simulation parameter setting is done, model is ready for simulation.
- Click the triangular button to start the simulation.

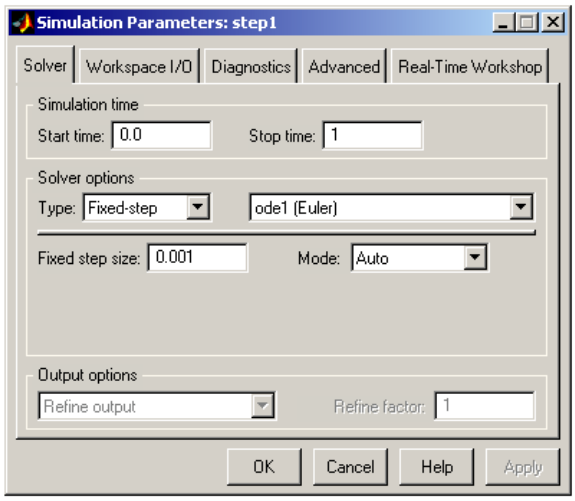


Figure 1.2: Simulation Parameters Settings

- Once the simulation is completed, double click on the scope box. To auto-scaling the axes, click the binoculars on the scope screen. Your scope results should look like Fig. 1.3.

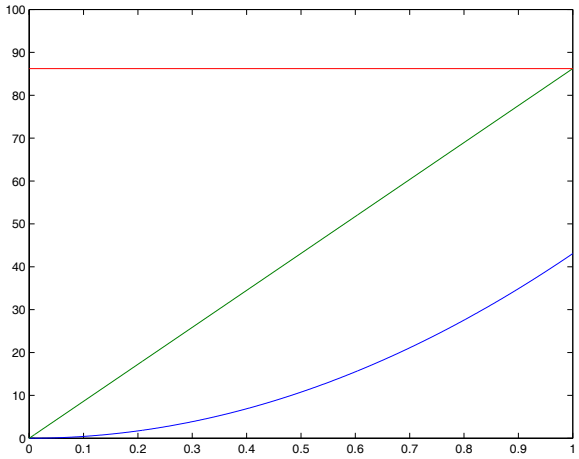


Figure 1.3: Step 1 Simulation Results

- Now by changing either of the slider gains and running the simulation using triangular button, you will obtain different results in your scope.

1.3 Changing the Simulation Model

Now that we have created a simple mechanical rotating model, let's add friction to our rotating system

$$T_{fric} = B.\omega \tag{1.6}$$

For our example we will use $B = 0.025$.

- First make sure you have saved your previous model as Step1. Then make another copy of it and name it Step2.
- In addition to making the change for the above equation it would also be nice if theta would become a repeating type sequence instead of continuously rising or falling. To do this add either a sine or cosine block from the math library.
- Also remove the mux block and add another input port to the scope by double clicking on the scope. Find the parameters icon in the upper right and change number of axes to two and time range to one second as in Fig. 1.4

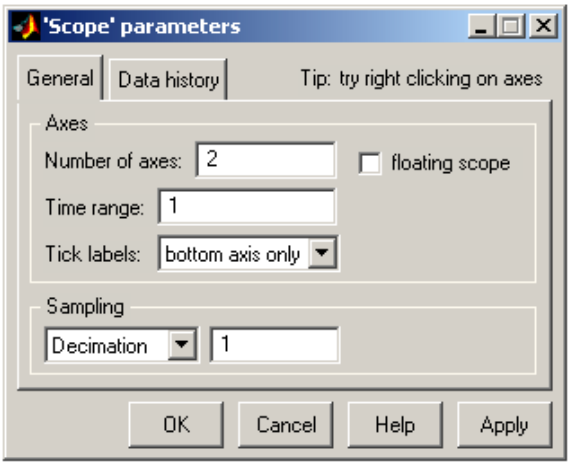


Figure 1.4: Scope Parameter Setting

- We would also like the system to run continuously. To do this go to the simulation parameters as was done in Step 1 and make sure all the parameters are the same as in Step 1, but change the Stop Time to "inf".
- When you are finished changing the model it should look like Fig. 1.5
- Run the simulation again and change T_M and T_L as the simulation is running to see how the system responds.

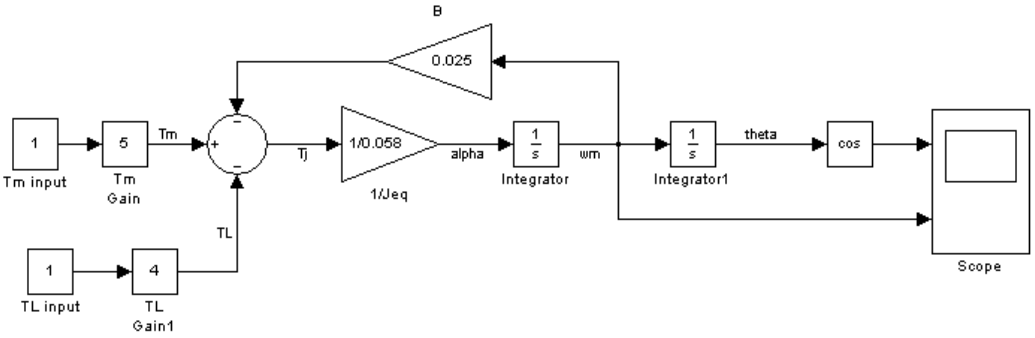


Figure 1.5: Simulation Model with Friction Included

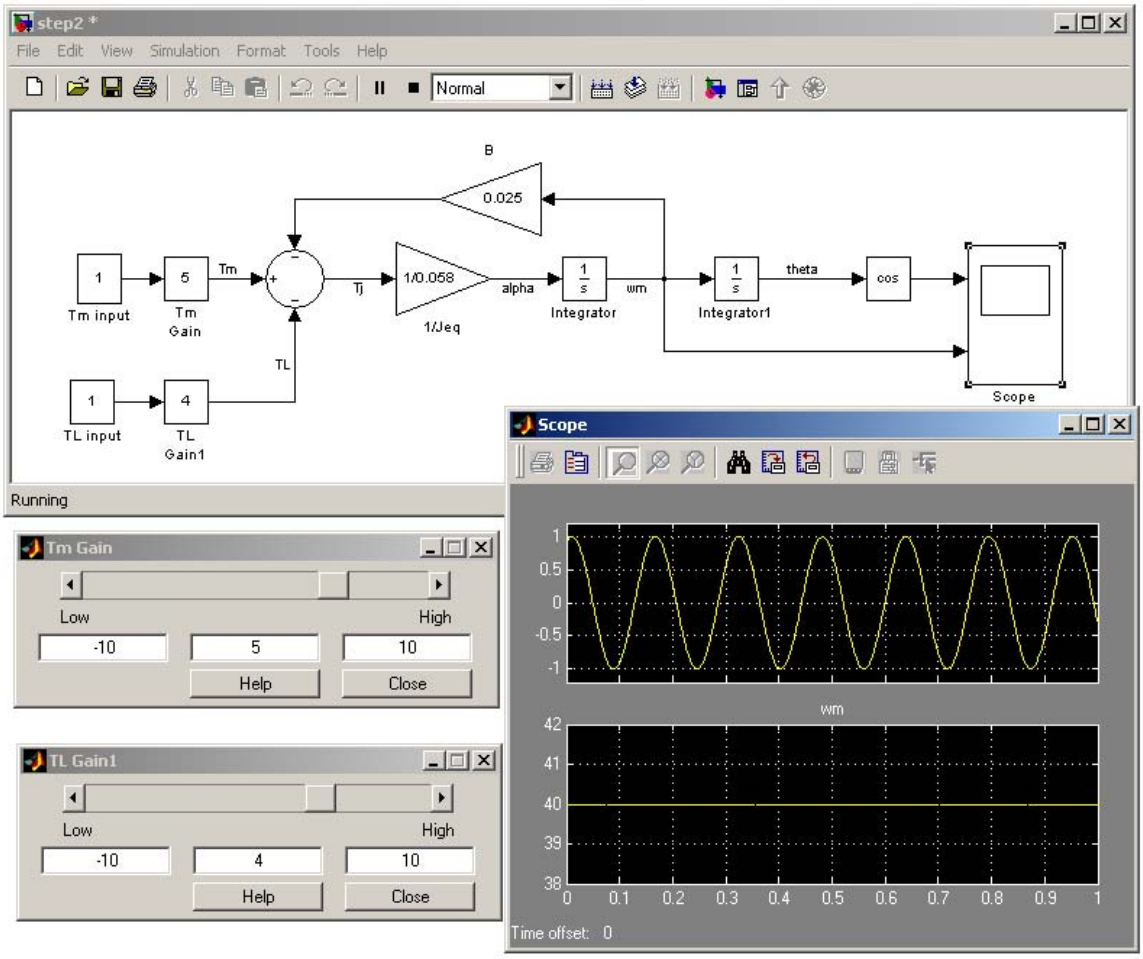


Figure 1.6: Simulation Results for Model Including Friction

1.4 Lab Report

The lab report should contain a brief, along with the details what have been done in the lab and details asked below.

- Simulation results of mechanical system obtained in section 1.2. Comment on the results obtained.
- Simulation results of the mechanical system along with friction included in the model (i.e sec. 1.3). Comment on the results obtained. What's the effect of friction on the system performance?
- Change the value of T_M and T_L. What is the effect of each and explain why ?
- Change the load torque T_L. What is the effect of sudden change. Observe the dynamics obtained after the load change and comment about its effect on speed, acceleration and theta.

Experiment 2

Building A simple REAL-TIME model in SIMULINK

2.1 Introduction

One of the best features of the dSPACE package is the ease of building real-time applications. The time between converting the design into digital instructions for the DSP and effectively running the application depends only on how fast your computer can compile the initial code.

Basically a real-time application can be created by means of two methods:

- Using MATLAB/Simulink for building the model and automatically generate the C code and download it into the DSP memory.

- Hand-coding in C and compile the model into DSP code.

The fastest way of developing a real-time code is developing the model in Simulink and preparing a real-time model from that. Basically once you have completed the Simulink model which you want to run in real-time, the only command required is **RTW Build** under **Tools** menu in **SIMULINK**. Once the command is executed, dSPACE software creates the object (*.obj) file, downloads it on DS1104 board and automatically starts the hardware execution.

However, there are some important settings you have to make before "transporting" your model into the real-time world. Let's start with a simple example

2.2 Creating a model in Simulink

Our first example introduces the analog channels input-output communication with external devices. For this example a Signal Generator is used to generate different waveforms as inputs to our signal processing algorithm. The result of this process will be directed to an analog output channel, in order to be monitored with the lab oscilloscope.

Suppose that we need to analyze the response of a second-order system at different types of input signals, with variable amplitudes. The second-order system is defined by the following parameters:

- Damping (ξ) = 0.7, Natural frequency (ω_n) = 20Hz
- We need its response at different types of input signals such as Sine-wave, Square-wave, Saw-tooth wave.
- The input has a variable gain, in the range of [0 ... 5].

Firstly simulation model will be developed using a model for signal generator and model for Oscilloscope.

- Create a folder expt03.
- Start MATLAB and set the Path Browser to your working folder (expt03).
- Type Simulink at the prompt line and create a new model from **File** menu.
- Choose from the Simulink *Continuous library* the **Transfer Fcn** block and drag it into a new simulation model.

The second order system, with the parameters specified, can be described in transfer function with the following relation:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega s + \omega_n^2} \tag{2.1}$$

where $\omega_n = 2\pi f = 2\pi.20 = 62.63$ and $\xi = 0.7$ The numerical model becomes

$$G(s) = \frac{3947.84}{s^2 + 87.962s + 3947.84} \tag{2.2}$$

- Set the parameters of the **Transfer Fcn** block as shown in Fig. 2.1
- Next, drag a **Signal Generator** block from *Sources* library, a **Slider Gain** from *Math* library and a **Scope** from *Sinks* library. Connect all blocks as in Fig. 2.2.
- Now, set the simulation parameters by pressing Ctrl-E as:

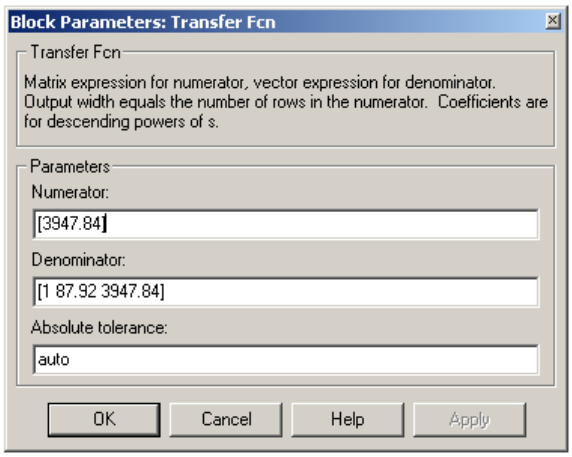


Figure 2.1: Parameters for Second Order System

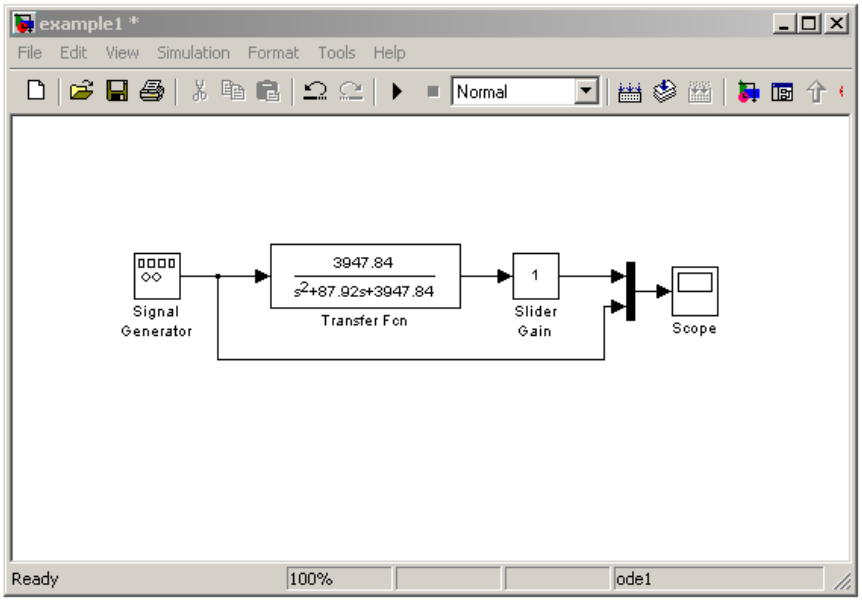


Figure 2.2: Simulink Model of 2^{nd} Order System

- Set the **solver options** to *fixed step* and *ode1*.
- You need to specify a fixed step size. Let it be "0.001" i.e 1ms.
- Stop-time as "2" seconds.
- Set the Signal Generator output to *Square-wave* and frequency as "2Hz".
- Keeping other parameters unchanged.

- Run the simulation by pushing the triangular play button and adjust the display of the Scope with the autoscale option. You will obtain a plot similar to the one shown in Fig 2.3.

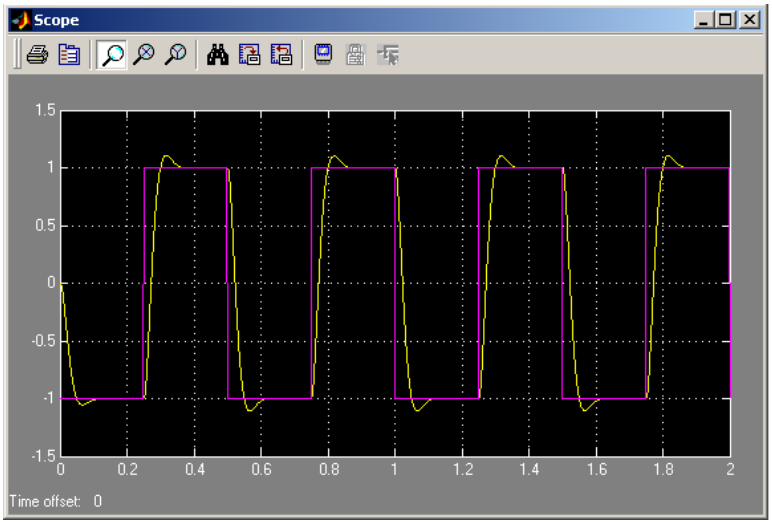


Figure 2.3: Simulation Results of 2^{nd} Order System for Square Wave Input

This simulation was really fast. The generator waveform can be changed by double-clicking the signal generator block. Now, since we have the idea how the system work, we will implement the system in "real-time" and observe the results.

2.3 Build the real-time simulation model

The model developed for simulation is now to be connected to the external devices (Signal generator and Oscilloscope). Since these devices are physically generating/accepting signals going to or coming from the DSP board, we need to stream these signals via the analog Input/Output channels, located on the controller box.

Firstly, make sure that the Signal Generator and the Oscilloscope are connected via shielded BNC cables to ADC#5 and DAC#1 respectively. The Signal Generator output is set such that its signal amplitude is approximately 1V.

Communication with the input/output channels is performed via two dSPACE blocks found in the *dSPACE RTI1104* library under the sub-library *DS1104 MASTER PPC*, named **DS1104ADC_C5** and **DS1104DAC_C1**. They will replace our **Signal Generator** block and **Scope** block respectively. The analog input channel is down-scaled by the hardware with a ratio of 1:10. This means that 10 V at the input will be read as 1V in our model. The analog output channel is also down scaled in the hardware with the same ratio. Thus, a 1 V signal generated within the model will have an amplitude of 10V at the connector. Thus, two **Gain** blocks, from the **Math** Library, will be required to correctly read and write the values from and to the analog channels.

- Drag the **DS1104ADC_C5** and **DS1104DAC_C1** blocks into the Simulink model and replace the Signal generator and Scope blocks. Place the two gains of 10 and 0.1 on the input and output signals respectively. The model should become similar to the one depicted in Fig. 2.4. You might need to save the model with another name, to preserve the simulation model.

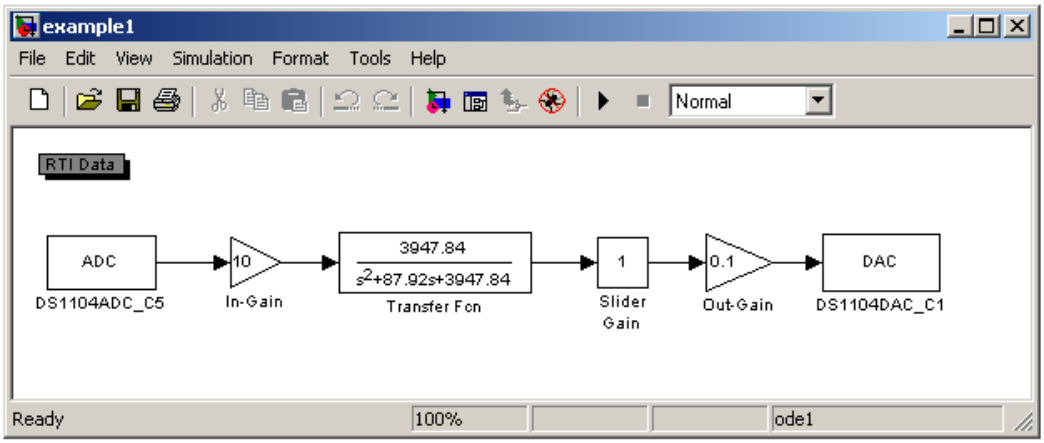


Figure 2.4: Real-Time model for dSPACE in Simulink

Remember that we performed the simulation for only 2 seconds. In real-time, however, the system needs to run continuously. Therefore, press CTRL-E for simulation parameters and

- Set the **Stop-Time** as '**inf**'. The simulation parameters should look like shown in Fig. 2.5
- In Simulation Parameters, go to "Advanced" and make **Block Reduction** "**OFF**".
- Next, choose the **Real-Time Workshop**, **Build Model** from the **Tools** menu.

Once the above command is given, you will observe a list of messages displayed in Matlab Command window. These messages correspond to the different steps that the RTI Software perform in order to transform the Simulink code of the *example1.mdl* file into DSP code.

First there is a compilation stage, in which the Simulink file is transformed into a C file, then comes the link stage where all the variables and subroutines are correlated with the DSP environment, and finally the code is transformed into an object file and downloaded into the DSP memory. The MATLAB window screen will look like shown in Fig. 2.6. You can see that the file was successfully built. The result is "example1.obj" which was already loaded in the DSP memory and its execution started.

Please note that the directory in which the model was built is the same you choose for creating the Simulink model. If you look now in this directory you will find several files, generated during the

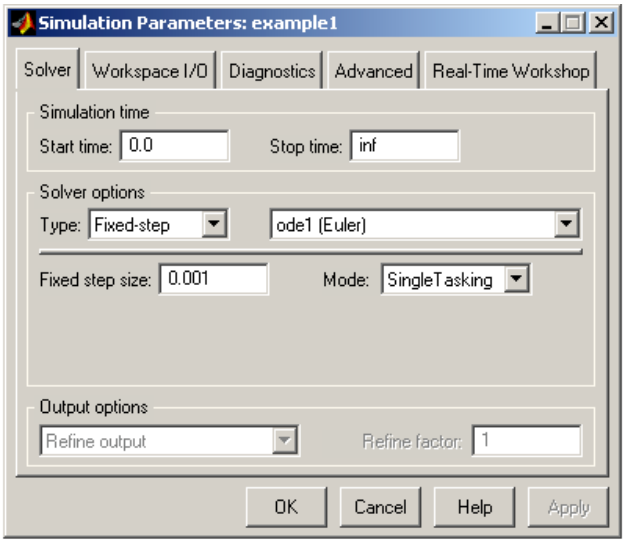


Figure 2.5: Simulation Parameter setting for real-time model

build command. Due to the large number of files generated, it is advisable that each project to be located in a separate sub-directory.

Now, our simulation is running in the DSP board, in a digital, real-time format. We can predict that this is much faster than what we saw on the Scope screen, in Simulink. More important, now, seems to be the interaction with the system. We need to visualize, modify and analyze the variables. For this, dSPACE comes with its own Graphical User Interface, called **CONTROL DESK**.

So, let's learn how to use the Control Desk to interact with the real-time simulation.

2.4 Creating a new experiment file with Control Desk

ControlDesk is a software that allows the user to look at the variables, display their behavior and modify the simulation parameters by interacting directly with the DSP board.

- Start ControlDesk and select only the toolbars checked as shown in Fig. 2.7.

  The Tool Window is displayed at the bottom of Control Desk screen as shown in Fig. 2.7. The Tabs display the tool currently used. In the figure only three of the working tools are available: *Log Viewer*, *Interpreter* and *File Selector*.

  As we shall further describe, there is one tool, very important, to which we shall give a special attention. This tool is called Variable Browser and the Parameter Editor. It provides access to the variables of an application. These variables are stored in a file called **example1.sdf**.

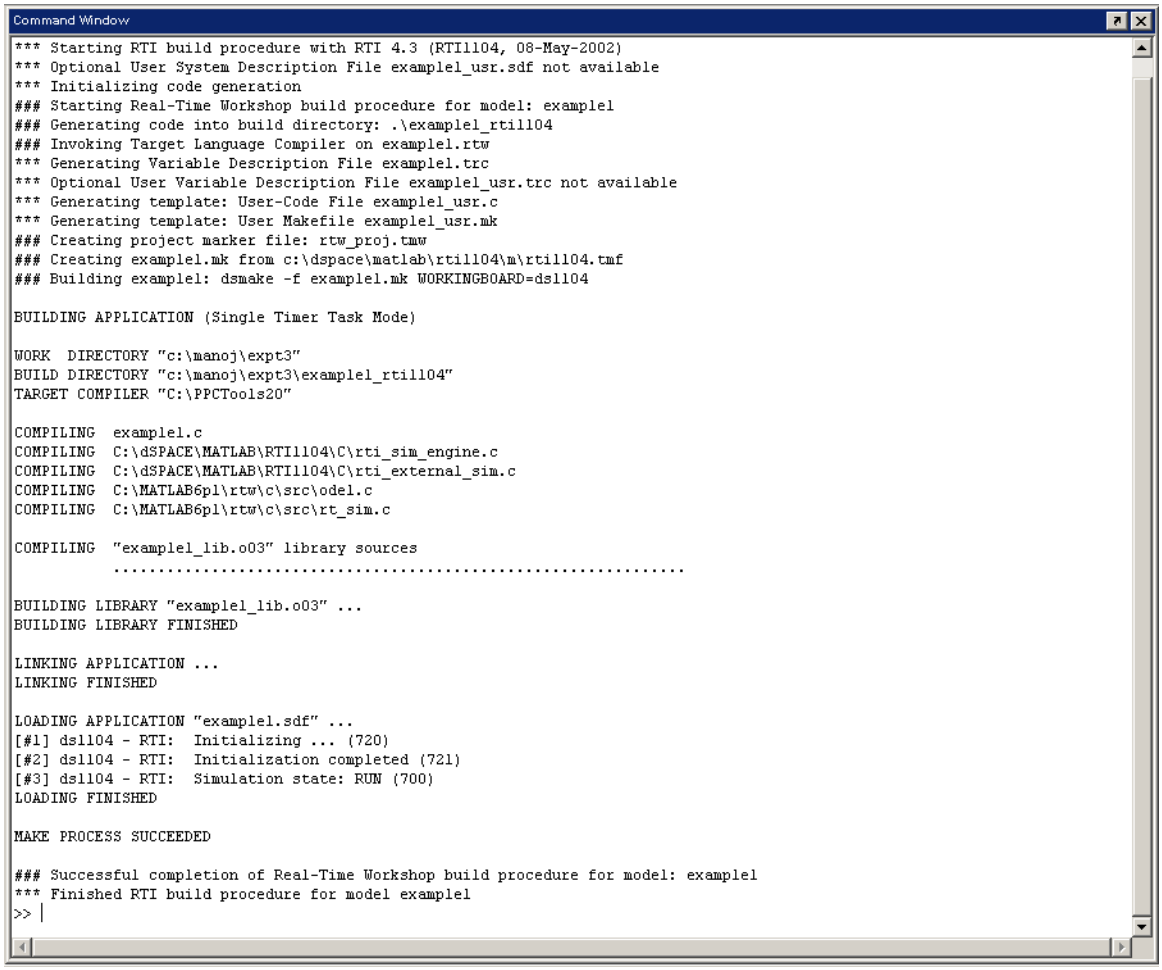


Figure 2.6: Compilations details developed by dSPACE and SIMULINK in Matlab Screen

Thus, for handling the variables of a simulation we must load the *.sdf file before starting the graphical design.

- First we start a new experiment. Click **File/New Experiment**. In the pop-up window write the name of the experiment, and very important, set the path where the simulation files are stored. (see Fig. 2.8). **Note** When creating a new experiment, don't forget to set correct working directory.

- Next, load the file containing the variables of the simulation. Click **File/Open Variable File** and select example1.sdf.

The **Variable Manager Tab** appears at the bottom of the screen (see Fig. 2.9). The window contains the structure of the simulation model.

At the highest level we see the simulation control variables. Their function is described in Table1.

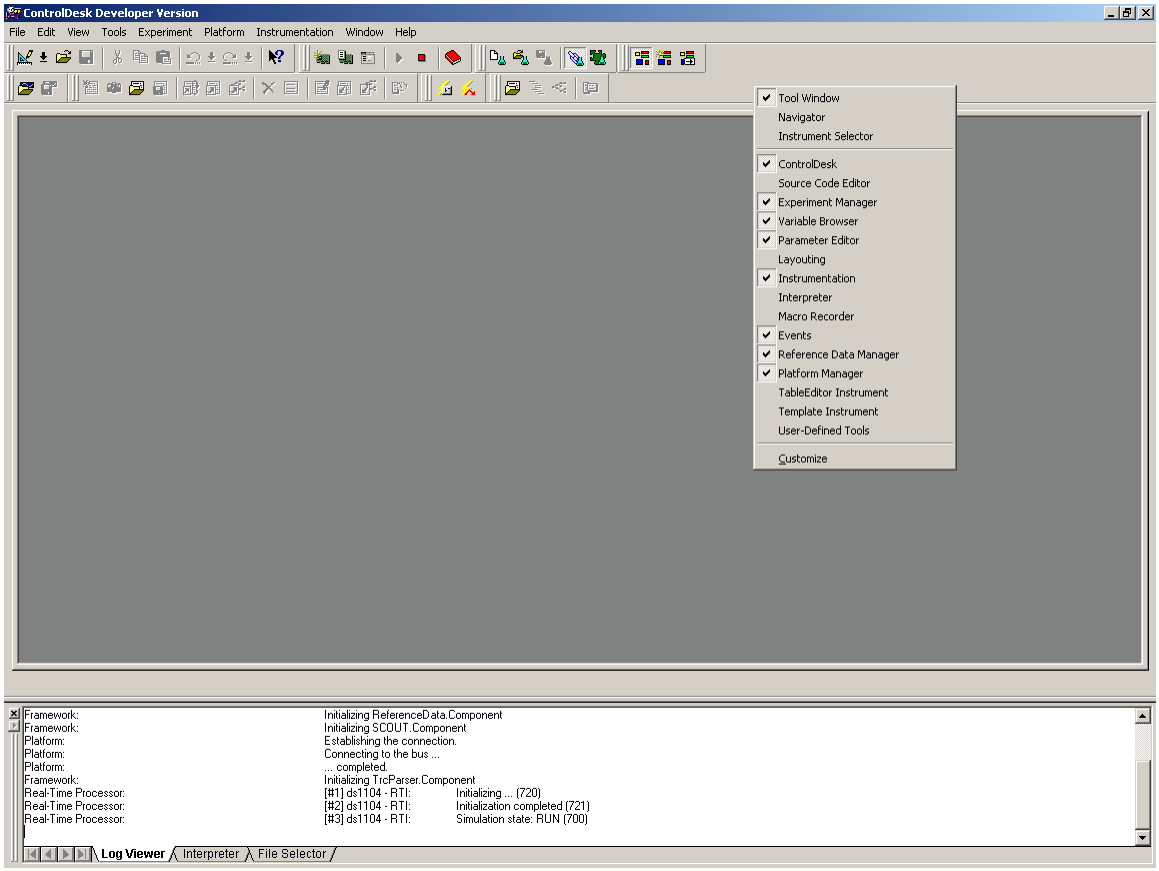


Figure 2.7: Control Desk Screen

For the purpose of our simulation, the variables of interest are contained under the **Model Root** group. This group contains the variables belonging to the top-level of the Simulink model. Variables from subsystems go into further groups in deeper hierarchical levels. The variables available have a prefix to distinguish the different variable types and are generated in the following order as detailed in Table. 2.1.

Prefix : Example	**Variable Type**
L:Output	labeled signals
B:Integrator	block outputs
S:Scope	inputs of signal sinks
P:Signal Generator Amplitude	block parameters

Table 2.1: Variable Types and Signal Correlation

The name following the prefix is the block name, except for labeled signals where the label itself is used. Here are all the variables corresponding to the **Model Root** for our example are detailed in Table. 2.2.

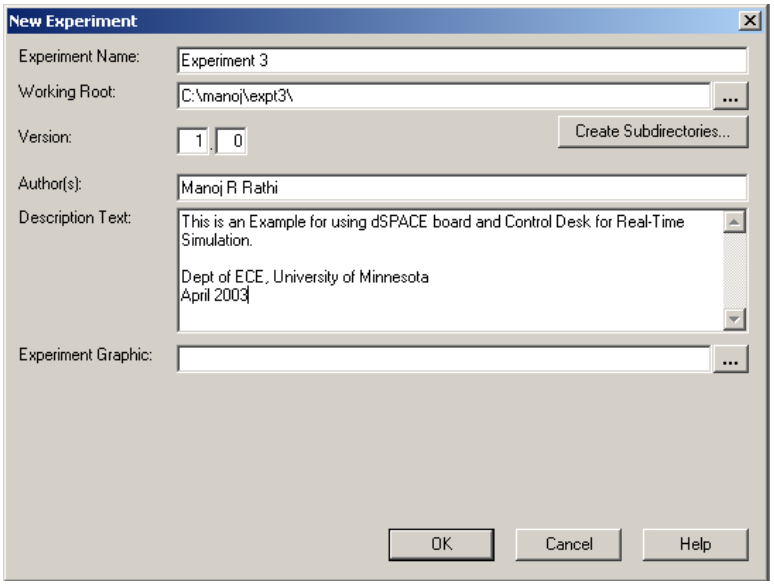


Figure 2.8: Experiment Settings

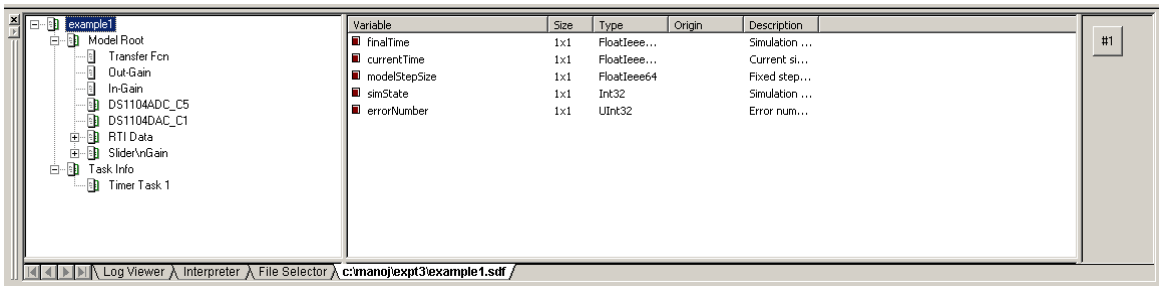


Figure 2.9: Variable Tab Manager

2.5 Display Controls and Scopes with Instrumentation Management Tools

In order to see the behavior of each variable and modify the parameters in real-time, while the system is running, we need a series of buttons, knobs, slider-gains, plotter etc, that can handle these variables.

Therefore, we need to start a new **LAYOUT** screen in which all those instruments can be added.

- Click File/New/Layout, from the menu.

Two new windows appear in the ControlDesk workspace as shown in Fig. 2.10. The one called Layout1, will contain the instruments used for managing the experiment. The second window is

'*Model Root/B:DS1104ADC_C5->ADC#5*'	Display the variable received on A/D channel #5.
'*Model Root/B:Slider Gain>Out1*'	Display the output of Slider Gain.
'*Model Root/B:In-Gain*'	Display the output of the Input Gain block.
'*Model Root/B:Out-Gain*'	Display the output of the Output Gain block.
'*Model Root/B:Transfer Fcn*'	Display the output of the Second order Transfer Function
'*Model Root/Slider Gain/P:Slider Gain.Gain*'	Contains the value of Slider Gain.
'*Model Root/P:In-Gain.Gain*'	Contains the value of Input Gain.
'*Model Root/P:Out-Gain.Gain*'	Contains the value of Output Gain.
'*Model Root/P:Transfer Fcn.A(1)*'	Handle the transfer function parameters (numerator and denominator)
'*Model Root/P:Transfer Fcn.A(2)*'	
'*Model Root/P:Transfer Fcn.C(1)*'	
'*Model Root/P:Transfer Fcn.C(2)*'	

Table 2.2: Variables handled by dSPACE in the *example1* experiment

actually a toolbar which let us drag and drop the necessary controls for the experiment.

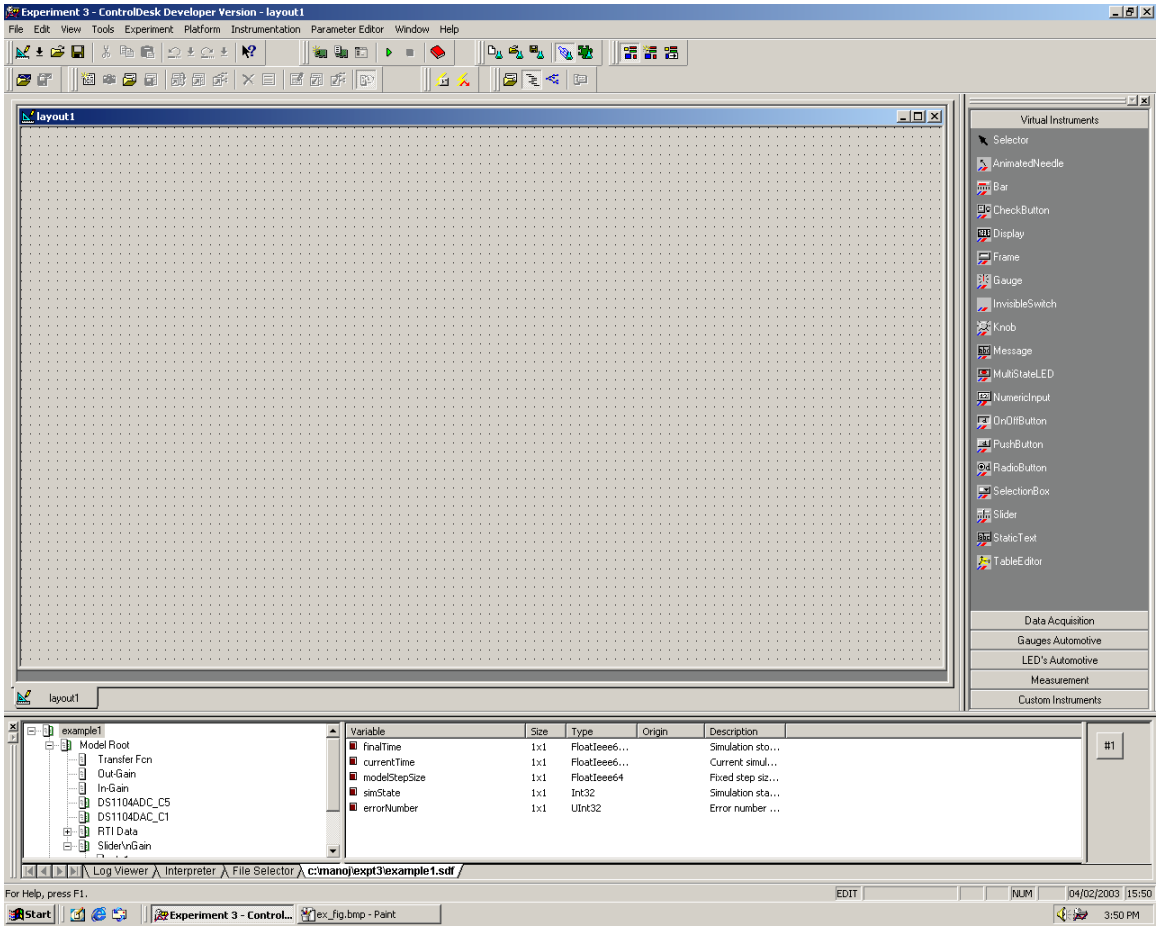


Figure 2.10: New Layout Window for Instrumentation and Control

The controls displayed in the Virtual Instruments toolbar let us handle only the variables that can be modified on-line, i.e. the **P**:-type variables. Our example has 5 such variables. Four of them were listed in Table. 2.2 and there is one more under the **Slider Gain** hierarchical level, which controls the amplitude of the Gain block.

- Select the **SLIDER button** from the right toolbar.
- The cursor changes into a square target. Click and hold the mouse while dragging a rectangular shape in the Layout1 window (see Fig. 2.11).

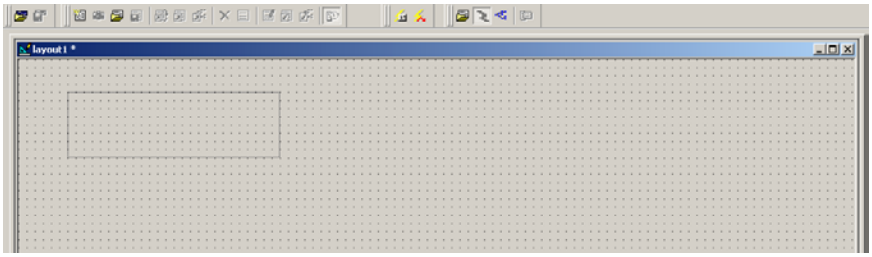


Figure 2.11: Selecting Slider Bar from Visual Instruments and drawing

- When the mouse is released, you will obtain a slider control, which let you continuously change any **P**:variable between the limits selected.
- As long as we decided to use amplifications between 0 and 5, double-click the Slider control, select the Slider Tab, and set the Range Min and Range Max as shown in Fig. 2.12.
- Now the limits where the cursor can be adjusted are 0 to 5. But the slider is still bordered with a red line. This means that it hasn't been assigned a variable to control yet.
- In the Variable Manager window, at the bottom of the screen, select the Slider Gain.
- Click the **P:Slider Gain.Gain** variable and drag it to the rectangle drawn in the Layout window.
- The new Slider control will display the handled variable and will no longer be bordered with a red line as shown in Fig. 2.13.

Now, we can add some visualization equipment.

In our example, only two signals are worth monitoring: the system input, coming from the real signal generator connected to ADC#1 and the output of the second-order system. Both should be displayed on the same scope. The output signal will also be monitored with the Lab Oscilloscope connected to the DAC#1 channel, on the dSPACE Interface Box. Remember that the real values

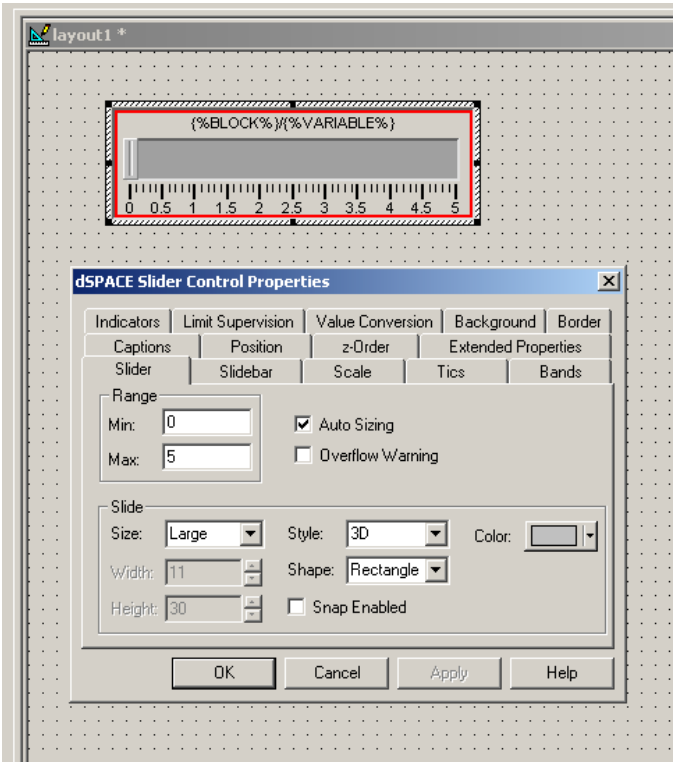


Figure 2.12: Slider Control Setting Window

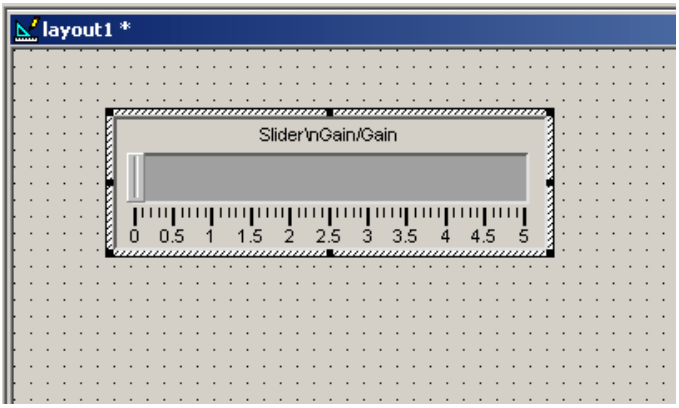


Figure 2.13: Slider Control with Variable handled and its control limits

of the input signals will be obtained after the **In-Gain** block, while the real values of the output signals will be obtained before the **Out-Gain** block.

- Click on the Data Acquisition tab, in the Instrument toolbar at the right.
- Select the Plotter icon and draw a larger rectangle in the Layout window.

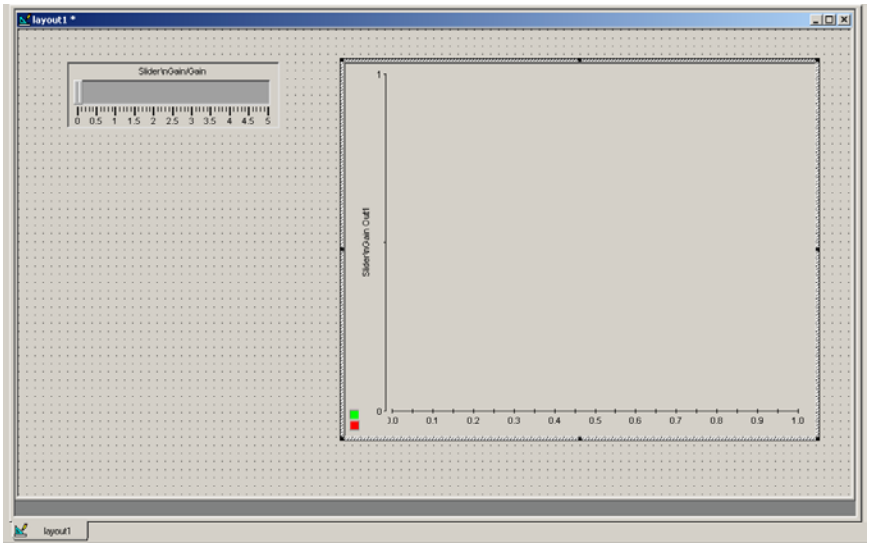


Figure 2.14: Drawing a Plotter for Signal Monitoring

- Drag the signals which you need to monitor, in the new plotter:

  'Model Root/In-Gain->Out1' and 'Model Root/Slider Gain->Out1'.

- When dragging the second signal, make sure that you release the mouse button above the first one, on the vertical axis. Otherwise, a new vertical axis will be drawn, and you will have less available space for visualize the waveforms. Now both signals are assigned to the plotter and will be displayed with different colors. The label on the vertical axis will show only the last signal dragged to the scope. You should have your layout as shown in Fig. 2.14

- Any time you wish to make a correction, or see what signal was added to the plot, right-click in the scope area and select **Edit Data Connections** command. You can delete any signal by selecting it and press Delete key. You can even delete or modify your signals or changing the colour of the signals, by double clicking Y-axis and going in to the "*Signals*".

- To format the scope, double-click in the rectangle and select **Y-axis** tab. When the signals are displayed on the same y-axis, the graph settings will be identical for both. Otherwise, you will be able to select different y-axis limits for each of the signals, independently.

- Set Y-limits, Width, Scaling Mode as shown in Fig. 2.15

- At the X-label you can write "Time" and at Y-label write "Input and output signals".

  One more step, before you actually can start the simulation: setting the visualization parameters. The process parameters can be set in the **Capture Settings Window** under View/Controlbars menu

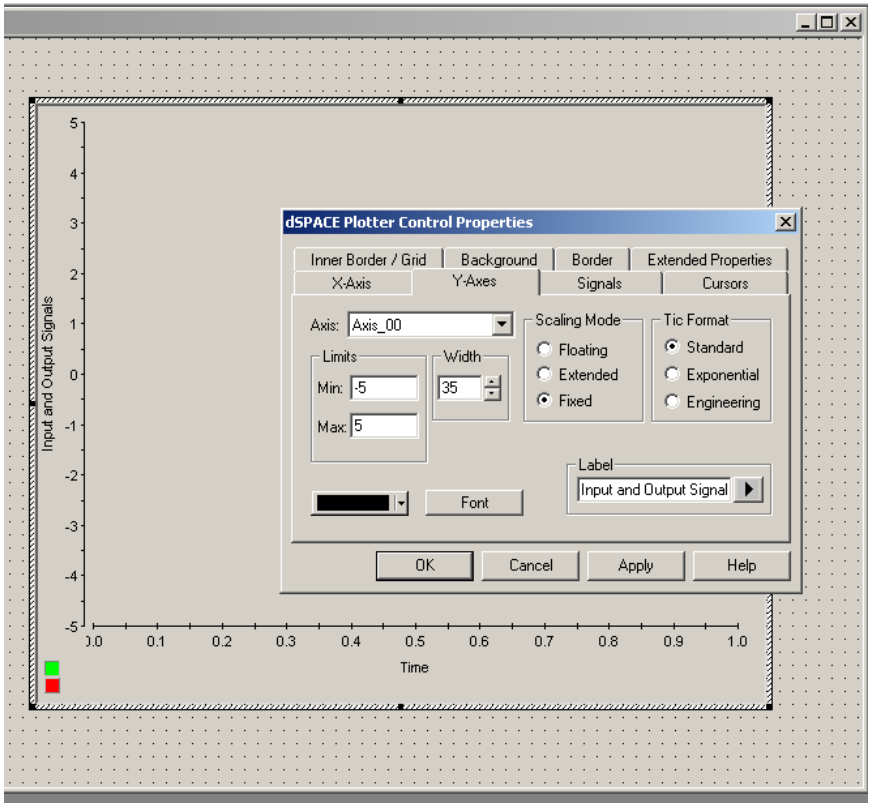


Figure 2.15: Plotter Setting Window

- Open the Capture Settings Window and set the Length of the simulation as 2, while leaving unchanged the Downsampling number. Your capture settings should look like as shown in Fig. 2.16. For more complex systems this number has to be increased when the Length is larger than 20 times the sampling time.

Now, we are ready to start the simulation.

2.6 Running the Experiment

There are two operations that have to be done to run/stop the experiment. First, the DSP execution has to be started. Second, the animation and data acquisition/printing needs to be initiated. When stopping an experiment, the operations have to be done in the reverse order.

- To start and stop the DSP you can use the icons in the Platform Management toolbar (see Fig. 2.17.
- Edit mode toolbar contains edit, test or animation icons. Refer Fig. 2.17 for edit, test and

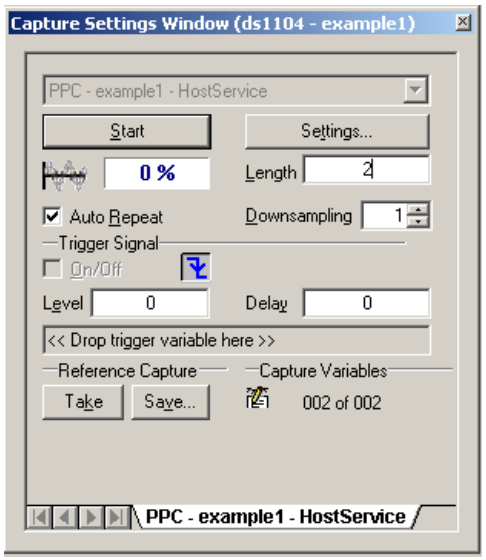


Figure 2.16: Capture Setting Window

animation tool icons.

Edit mode is used to edit or modify your layout, start and stop your real-time simulation. Once you start the real-time simulation, click the animation icon, this will let you modify parameters in real-time and observe the waveform. Unless real-time simulation starts, you cannot go into this mode. Clicking on Test icon lets you go into the Test mode. In this mode, you can observe the snap-shot of your real-time simulation.

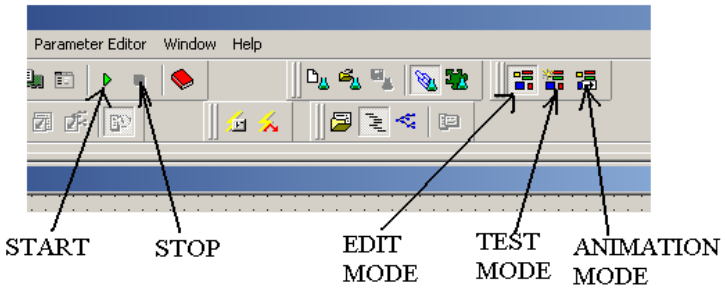


Figure 2.17: Execution and Animation Control Toolbar

- Start the DSP by pressing the green triangle button. If the example1.obj is loaded in the DSP memory then it will start running and the Stop, red-rectangle shaped button will become active.

- Start the Animation by pressing the animation icon in the Edit Mode toolbar.

- You will see both the signal generator and the system output signals on the plotter. Every two seconds the scope will clear and a new set of data are displayed.

- Change any of the parameters and watch the modifications in the signals displayed on the scope. Also observe the waveforms in the plotter window and they should like shown in Fig. 2.18 for the slider gain value displayed in Fig. 2.18. Now the experiment is running and our design job is done.

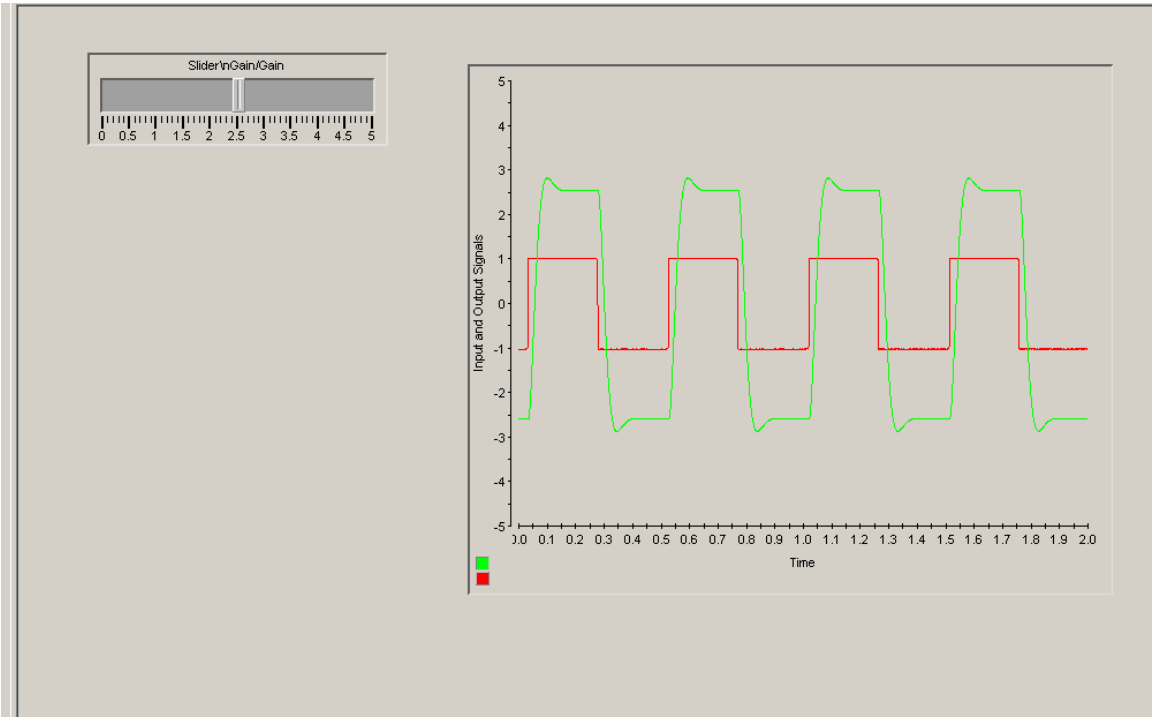


Figure 2.18: Real-Time Simulation Waveforms

- For saving an experiment, you must follow these steps, otherwise you might loose some precious design in the layout, or some simulation settings.

  1. Click File/Add All Opened Files. This implies that the experiment will remember the *.sdf file, containing the variables to handle, will know the path for all other files and will open the connections between the layout and the variables.
  2. Save the Layout in a file with the *.lay extension.
  3. Save experiment in a *.cdx type file.

- For loading an experiment, simply click File/Open Experiment and, if saved as previously instructed, the layout and the variables will appear on the screen.

Important note: Remember that the object file compiled has to be in the DSP memory. It is not implicitly loaded with the experiment!

2.7 Lab Report

The lab report should contain a brief, along with the details what have been done in the lab and details asked below.

- Simulation results of the 2^{nd} order system obtained in sec. 3.2.4.
- Results obtained on oscilloscope and in control desk in sec. 2.6.
- Compare the simulation and real-time results and comment on the that.
- Change the value of slide gain in real-time simulation and see its effect.

Experiment 3

Switch-mode DC Converter and No Load DC Motor Test

3.1 Introduction

In this experiment the operation of a DC switch-mode power converter will be studied. First, the building block approach will be used for simulation of converter operation. Then, a PWM algorithm will be implemented for a DC power converter. The purpose of the real-time implementation is obtaining variable voltage at the output of the power converter, while controlling its amplitude with a dSPACE-based user interface.

3.2 Simulation of DC Switchmode Converter

3.2.1 Triangular Waveform

To modulate the width of the voltage pulses in a power converter, a control voltage has to be compared with a triangular waveform signal. Before we build the model for the converter, a triangular waveform generator will be built in Simulink, using the **Sources/Repeating** Sequence block, provided by Simulink.

- Create a new directory for the experiment *Expt03*.
- Start Matlab and set the path to this directory.
- Open a new Simulink model.
- Drag and drop the **Repeating Sequence** block from the ***Sources*** library.

- Double click the block and enter the following set of data:
  - Time values: [0 0.5/fsw 1/fsw]
  - Output values: [-1 1 -1]

  Where the *fsw* variable will be our switching frequency set as a global variable in Matlab.
- Add a **Scope** to the **Repeating Sequence** block output.
- Type the value of ***fsw*** at the Matlab prompt (say ***fsw***=10000 - which means 10kHz)
- Set the *Simulation parameters* to the following values:
  - Stop time : 0.002
  - Fixed step size : 0.000001
  - Solver Options: fixed step, ode1 (Euler)
- Run the experiment.

The result should look similar to the one shown in Fig. 3.1

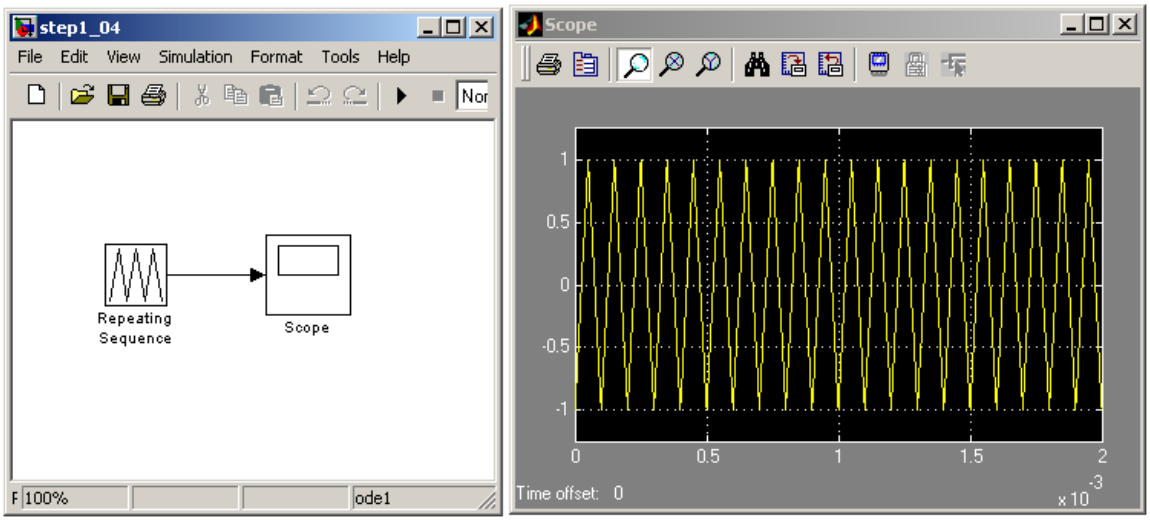


Figure 3.1: Triangular Waveform generator with 10kHz frequency

3.2.2 Duty Ratio and Switching Function

To obtain the duty-ratio for one pole PWM generation, we can use the equation (4-7) in Chapter 4, Electric Drives by Ned Mohan, i.e.:

$$d_A = \frac{1}{2} + \frac{1}{2}\frac{v_{control,A}}{\hat{V}_{tri}} \tag{3.1}$$

Using the relation (4-10) the duty-factor can be obtained with

$$d_A = \frac{\bar{v}_{AN}}{V_d} \tag{3.2}$$

Solving the Eqn. 3.1 and Eqn. 3.2 and considering the amplitude of the triangular voltage equal to unity i.e $\hat{V}_{tri} = 1$, we obtain the relationship for control voltage of pole A as :

$$v_{control,A} = \frac{2\bar{v}_{AN}}{V_d} - 1 \tag{3.3}$$

The above relationship in Eqn. 3.3 is implemented in Simulink. The control voltage is compared with the triangular signal. The **Relay block** output is set to '**1**', when the difference is positive, and '**0**' when the difference is negative.

The desired voltage $\bar{v}_{AN}$, with respect to the negative dc-bus ground is set by a **Constant** block with the value of one, and can be varied with a **Slider gain** from '0' to the maximum dc-bus voltage 'V_d' ($V_d = 42V$ in the model). The simulink model is shown in Fig. 3.2.

3.2.3 Two Pole Converter Model

To obtain the model for a DC converter we need to add in our model two building blocks for a converter pole. The building block is the **Switch** block from ***Nonlinear*** library, which allow the upper signal to pass when the middle input is greater than the specified threshold and the lower signal in the opposite case. The converter output voltage will be the difference between the two pole-output voltages, measured with respect to the dc-bus ground.

Now, the reference value for our model will be v_{AB}. We know that:

$$\bar{v}_{AB} = \bar{v}_{AN} - \bar{v}_{BN} \tag{3.4}$$

At any given instant of time, the control voltages for the two poles are complementary, i.e.

$$v_{control,A} = -v_{control,B} \tag{3.5}$$

Thus by solving the Eqn. 3.1 through Eqn. 3.5, we can write

$$v_{control,A} = -v_{control,B} = \frac{\bar{v}_{AB}}{V_d} \tag{3.6}$$

and

$$\begin{aligned} d_A &= \tfrac{1}{2}\left(v_{control,A} + 1\right) \\ d_B &= \tfrac{1}{2}\left(-v_{control,A} + 1\right) \end{aligned} \tag{3.7}$$

The above relations can be implemented in a Simulink model, together with the two-pole switches as in Fig. 3.3. The input is the desired average output-voltage v_{AB}. The instantaneous output

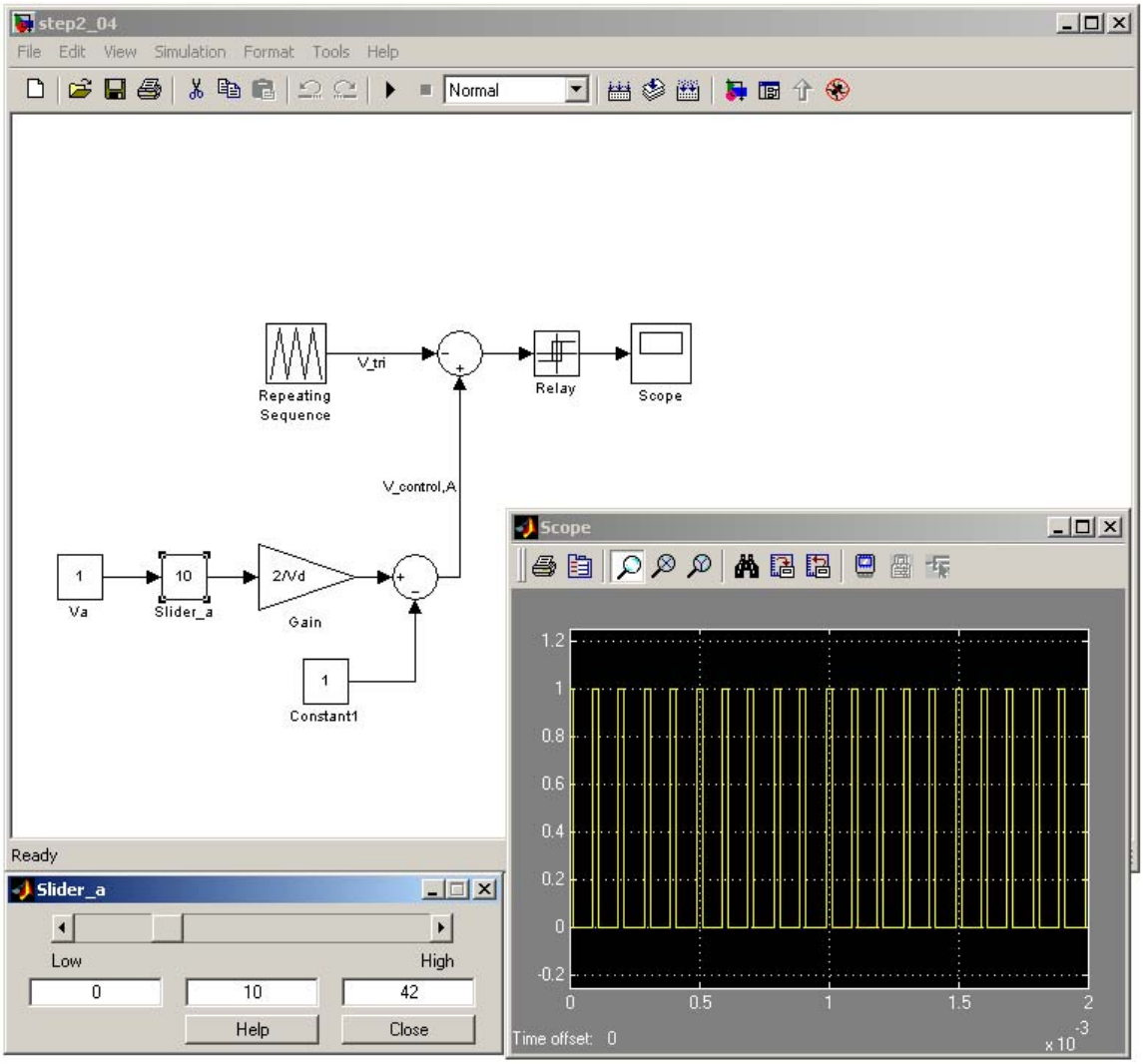


Figure 3.2: Switching Function generation for single pole converter

voltage will be a square wave signal and the average value will be equal to the value set by slider gain.

By varying the value of slider gain (i.e vary the desired average output voltage value), the output voltage value changes. Now set the simulation time to '***inf***' and start the simulation. Vary the slider gain value and observe the output voltage waveform. Set the axes values of the scope as shown in Fig. 3.3.

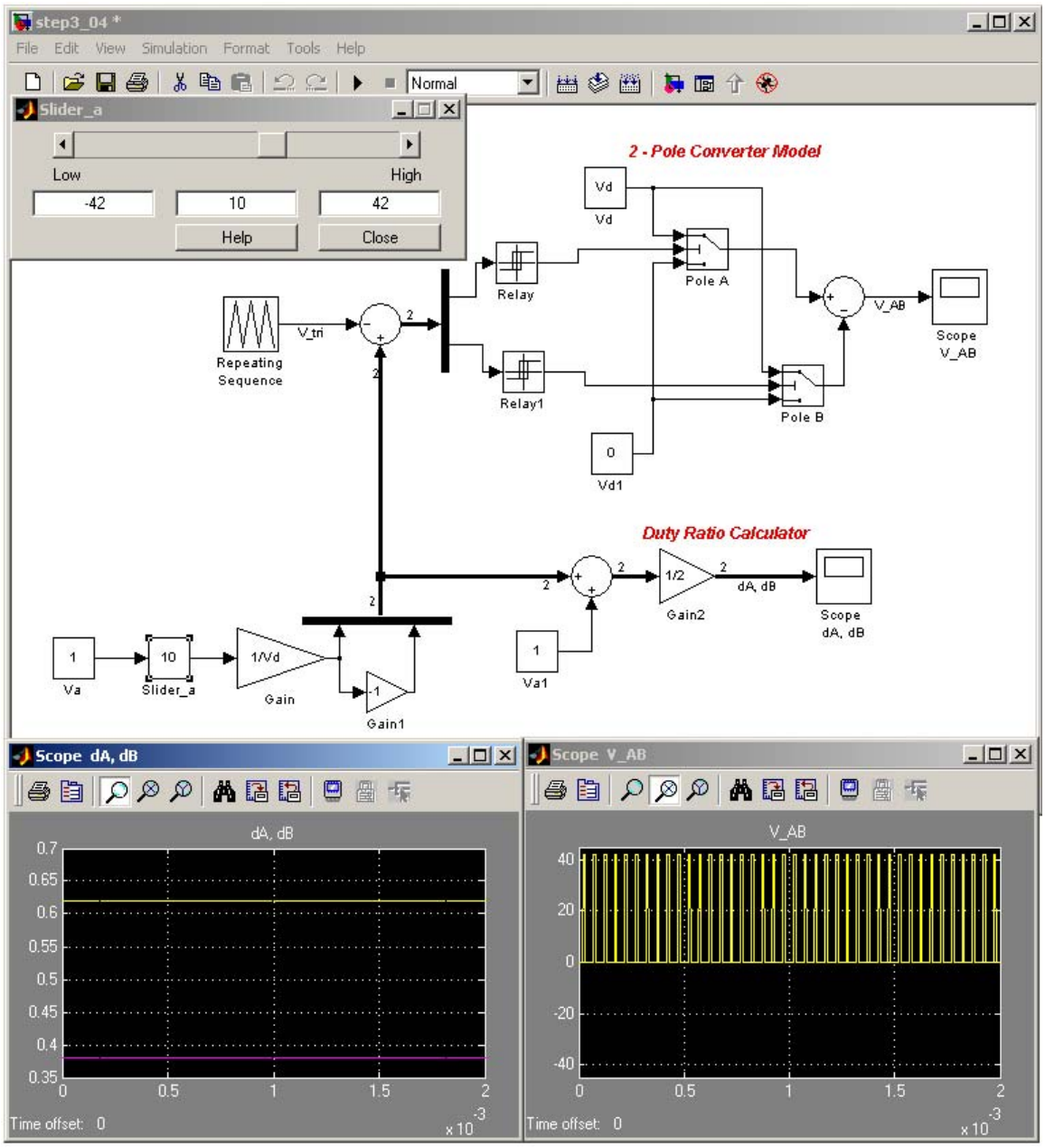


Figure 3.3: Two Pole Switch-Mode Comverter Model in Simulink

3.2.4 Simulation Results

Once the above system is ready, collect the following results for the two pole switch-mode dc converter.

- Switching function $q(t)$ for single pole converter.
- Simulation results of two pole converter model for two different values of V_{AB}, one positive

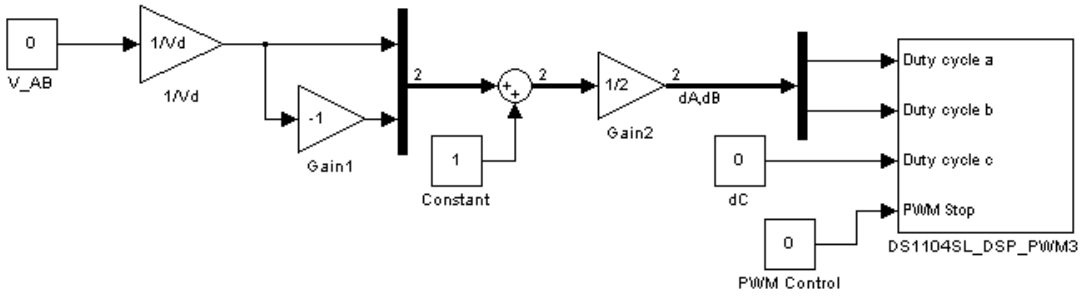


Figure 3.4: Two Pole Switch-Mode Comverter Model in Simulink

and another one negative.

3.3 Real-time DC Switchmode Converter Implementation

The lower side of the model is grouped in a block so-called Duty-Factor Calculator. Since in the real-time implementation the upper part is physically present outside the PC, the only computation required is the duty-ratios for each converter pole.

DSPACE provides a block called **DS1104SL_DSP_PWM3**, which embeds the triangular waveform generator and the comparator for all converter poles. Thus, when building a real-time system, all the upper portion of the model is replaced with this dSPACE block to generate the PWM signal, as shown in Fig. 3.4. The inputs in the **DS1104SL_DSP_PWM3** are the duty-ratios needed for converter control.

3.3.1 Simulink model for real-time implementation

- Create a new simulink model and save it as *Step4_03.mdl*.
- Drag and Drop the **DS1104SL_DSP_PWM3** block from the dSPACE library.
- Set the switching frequency to 10000 Hz and dead band to 0 by double-clicking the block.
- Cut and Paste the Duty-ratio calculator portion from *Step3_03.mdl*.
- Connect a constant block to the remaining inputs as shown in Fig. 3.4.
- Set the value of constant blocks to '0'.

3.3.2 Creating the user-interface in Control Desk

- Connect the Lab Oscilloscope to the **PHASE A1** and **PHASE B1** terminals of the motor drives board.
- Open the simulation parameters and change the fixed step size to 0.0001.
- Build (CTRL+B) the Simulink model.
- Open ControlDesk and create a new experiment in the same directory.
- In a new Layout, drag and drop a slider gain and assign the V_AB variable to it.
- Add some Plotters to visualise the duty-factors. Your layout should look similar to the one shown in Fig. 3.5
- Run the experiment and after turning on the power supply check the output waveforms on the Lab Oscilloscope.

3.3.3 Real-time Results of Switch-mode DC Converter

Once the setup is ready, collect the following data :

- Record the output voltage waveform of oscilloscope for the values of V_{AB} set in Sec. 3.2.4.
- Record the corresponding duty ratio waveforms for the above values.
- Measure the output voltage frequency and comment on the result obtained.

3.4 Controlling of DC Motor under No Load Condition in Open Loop

Before we start the implementation of the model for control of DC Motor in open-loop, make sure that you have connect the armature of the dc-motor under test to the output of two converter poles A and B. The **IA1** current measurement port on the drives board has to be connected to **Channel 5 A/D converter** of the ***dSPACE*** controller box. Also, the encoder output is connected to the **INC1 9-pins DSUB** connector on the ***dSPACE*** controller.

The speed of a dc-motor can be modified by varying its supply voltage. The model of output voltage control of the switch-mode dc converter was discussed in the previous section and we shall use the same.

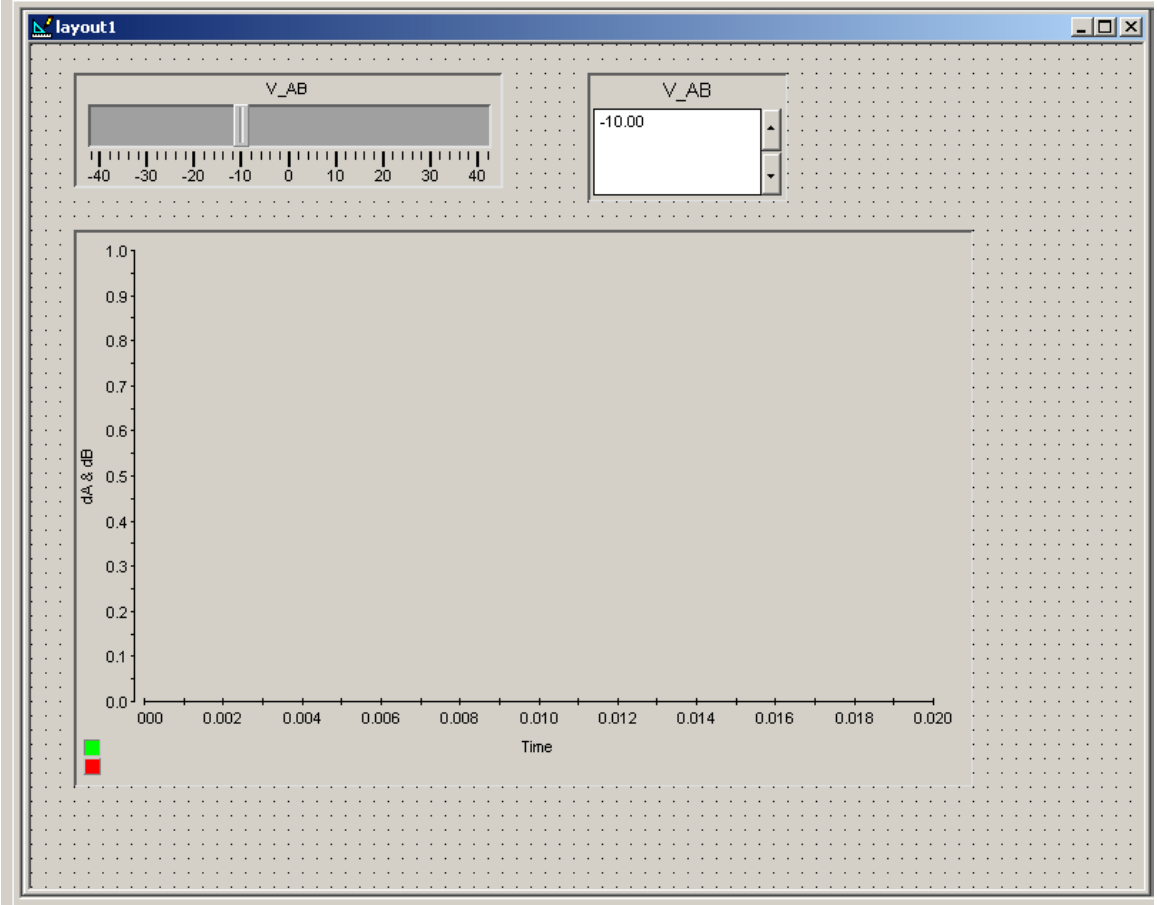


Figure 3.5: Control Desk layout for Switchmode DC COnverter

- Go back to SIMULINK.
- Open a new Simulink model.
- Set the simulation parameters as described in the section 3.2.1.
- Copy the ***Step4_03.mdl*** model and save it with a new name: ***Step5_04.mdl***.
- Change the name of the Constant block from **V_AB** to **V_motor**, since this will be the input in our dc-motor drive system.

Now, we have an open-loop control of the dc motor under test. However, we are not able to monitor the current and the speed while controlling the motor. Hence we will have to add the current and speed measurement blocks to our existing model.

3.4.1 Current measurement

For measuring the current we will be using **Channel 5** of the **A/D** converter. Remember from the first experiment that the data have to be scaled with 10. In addition on the motor drives current sensor 1V equals 2 amps so it actually needs to be scaled by 20.

- Drag and drop the **DS1104ADC_C5** block from the ***dSPACE*** library.
- Connect a **Gain** block at Channel 1 output and set its value at 20.
- Connect a **Terminator** at the output of the **Gain** block and label the signal as *Ia*.

3.4.2 Speed measurement

To measure speed we shall use the **DS1104ENC_POS_C1** block from the ***dSPACE*** library. This block provides read access to the delta position and position of the first encoder interface input channel. The delta position represents the scaled difference of two successive position values of a channel. To receive the radian angle from the encoder the result has to be multiplied with:

$$\frac{2\pi}{encoder\_lines} = \frac{2\pi}{1000} \tag{3.8}$$

where **encoder_lines** is 1000 for the encoders used in the laboratory setup.

Since we need to determine the speed, the delta position scaled to a radian angle has to be divided by the sampling time, as in:

$$\omega = \frac{d\theta}{dt}\bigg|_{digital} = \frac{\Delta\theta}{t_{k+1} - t_k} = \frac{\Delta\theta}{T_s} \tag{3.9}$$

Drag and drop the **DS1104ENC_POS_C1** block from the ***dSPACE*** library. In addition the encoder block, the **DS1104ENC_SETUP** block is to be added to the model. Connect a **Terminator** block to the Enc position.

Connect a **Gain** block at Channel 1 output (i.e Enc delta position) and set its value as $\frac{2\pi}{1000*T_s}$ where Ts is the sampling time set in the simulation parameters under the fixed-step box.

Now if you try to measure the low speeds, you will see oscillations between the two speed values. Hence we will add an averaging to get more accurate readings. For preparing the averaging block,

- Select the **Unit Delay** block from ***Discrete*** library and drag the required number into your model (say 10 for 11 point averaging).
- Select the input port **In1** from ***Source*** library and drag it into your model.
- Drag a **sum** block and double click on the sum block and add the required number of '+' signs in list of signs.

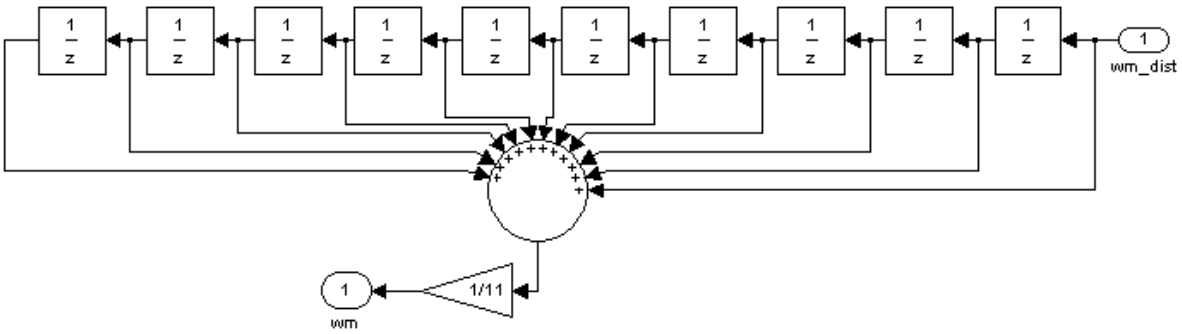


Figure 3.6: Averaging model in Simulink

- Drag a **Gain** block and connect it at the output of **Sum** block. Set the value to 1/11 for 11 point averaging.
- Select the output port **Out1** from ***Sink*** library and drag it into your model.
- Connect all the blocks as shown in Fig. 3.6.
- Select the unit delay's, input port, output port, gain and sum block together and create a subsystem. For creating a subsystem, go to **Edit** menu of your model and then ***Create Subsystem***.
- The system obtained is nothing but 11 point avaraging system. Change the subsystem name to Averaging.
- Connect a **Terminator** at the output of this 11 point averaging of speed measurement and label the signal: wm.

The model obtained should look like the one shown in Fig. 3.7. Before building the real-time model, don't forget to define **Ts** and V_d as global variables at Matlab prompt.

3.4.3 Creating Control Desk Layout

- Build the DSP code for the simulink model saved as **Step5_03**.
- Start the control desk and create the new experiment in the same working directory as that of simulink file.
- Create a layout and drag a slider gain control and two plotters.
- Drag and drop V_motor into slider gain control.

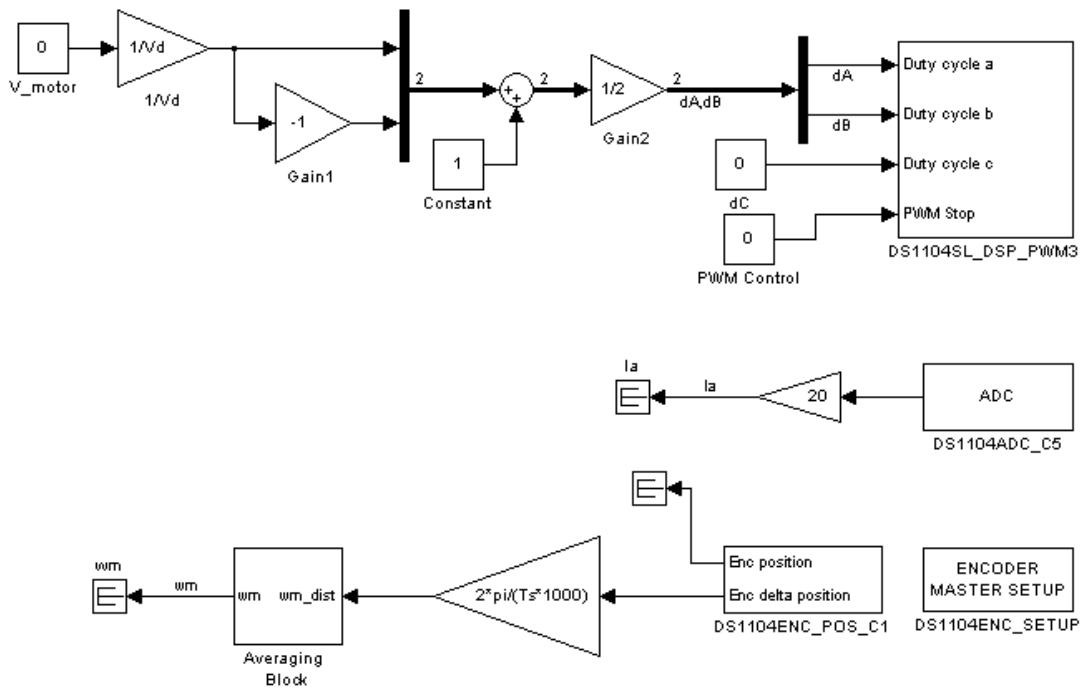


Figure 3.7: Real-time model for no-load motor test

- Assign one of the plotter for Ia and another one for wm.
- In order to record the numerical values of current and speed, add two **DISPLAY** into the layout. Drag and drop **Ia** in one of the display and **wm** in another one.

Now your experiment should look similar to the one shown in Fig. 3.8.

3.4.4 Results of No Load condition of DC Motor

- Record the values of current and spend for different set of readings for the corresponding voltage values specified in Table. 3.1
- Plot the voltage vs speed curve.
- Find the slope of the above graph

Note : Please keep the readings obtained in Sec. 3.4.4. It will be required in future experiment.

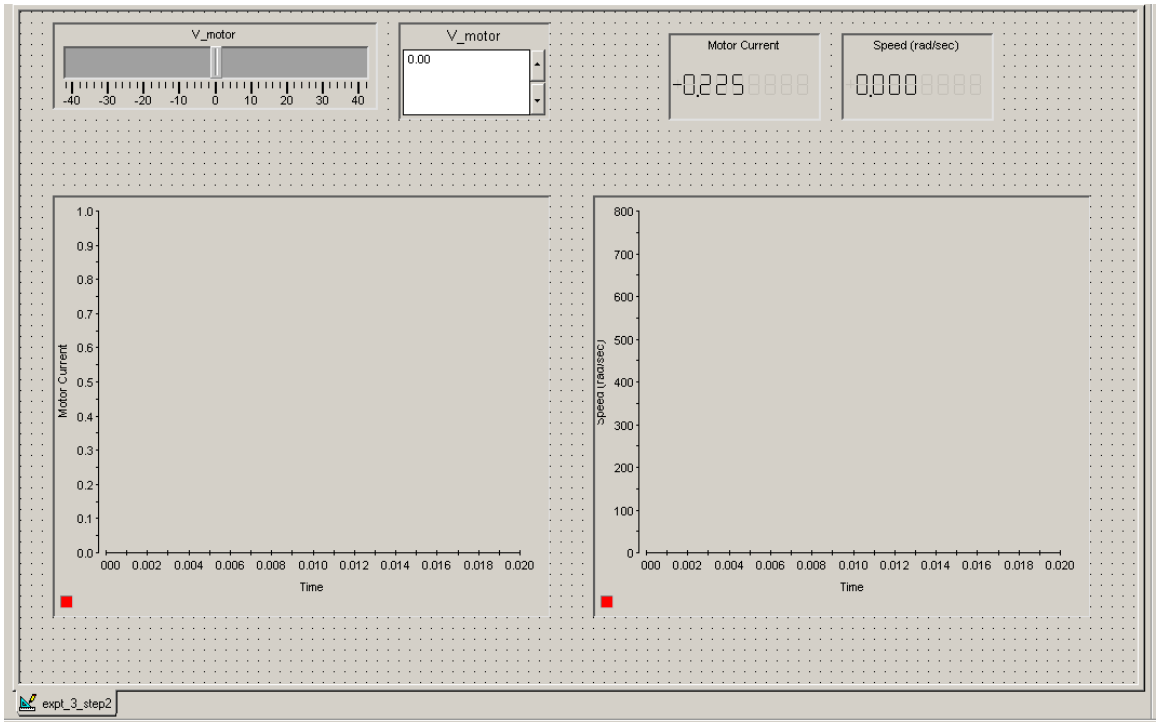


Figure 3.8: Control Desk Layout

Table 3.1: No Load Measurement

V_{motor}	(V)	0.5	1.0	3.0	5.0	10.0	15.0	21.0	30.0	42.0
Speed	(rad/sec)									
Current	(A)									

3.5 Lab Report

The lab report should be short with all the details asked below.

- Switching function result along with the simulation results of the 2 pole converter.
- Results of 2 pole converter obtained in real-time for some positive value of V_{AB} and for some negative value of V_{AB}.
- Measure the switching frequency of the output voltage of the converter. Explain why you get that value.
- The output voltage result of real-time implementation of two pole switch-mode converter along with its corresponding duty ratio waveforms.
- Plot of voltage vs speed curve for the no load condition of DC machine.

- Specify the slope of the curve obtained.

Experiment 4

Characterization of DC Machine

4.1 Introduction

In the earlier experiment, you designed and implemented the switchmode control and no load measurement of DC machine along with the current and speed measurement. In this experiment you will try to derived dc machine characteristic, which will be helpful in designing the closed loop control of DC motor.

We will be using the model prepared in the earlier experiment and modify the same one to derive the dc machine characteristic.

4.2 Open loop control of DC Drive with load

We will use the same model used in earlier experiment to do the no load testing of DC machine.

- Create a new folder **Expt 4**.
- Start ***Matlab*** and change the directory path to **Expt 4**
- Open the **Simulink** model ***Step5_03.mdl***.

Once we have the model for controlling the motor, we will have to add appropiate blocks for controlling the load.

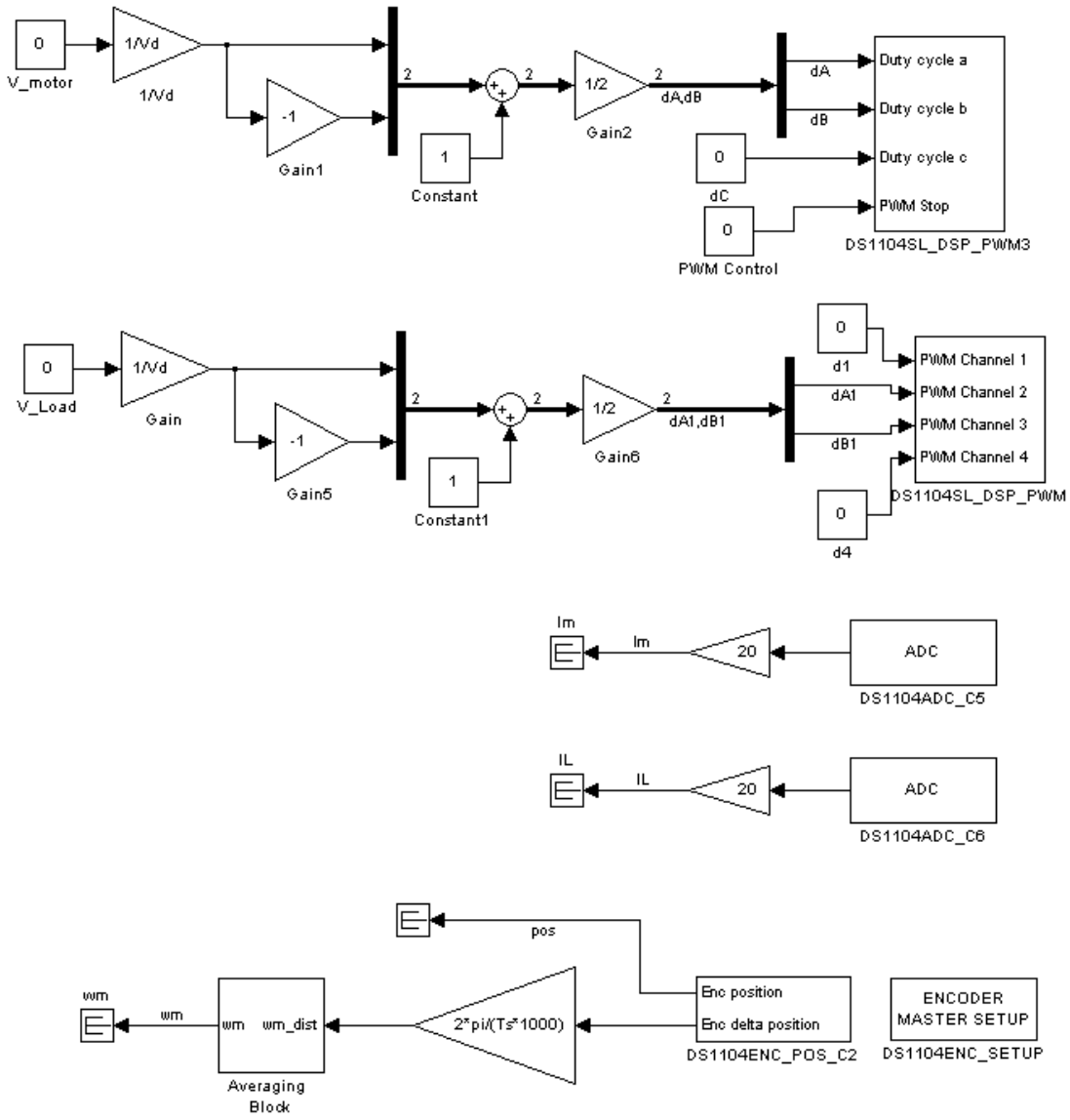


Figure 4.1: Real-time Simulink Model for Motor and Load Control

4.2.1 Adding a dc-load to the dc-drive

To determine the dc-motor steady state characteristics, a second dc-motor will be axially coupled to the motor under test (MUT). The second motor will be voltage controlled in an open-loop, similarly to the MUT.

- The armature voltage of the load should be connected to **PHASE A2** and **PHASE B2** terminals on the power board. Set the Bus Voltage, V_d to $42V$.

- Ensure a firm mechanical coupling between the motors.
- Add a second voltage control set of blocks in the Simulink model and connect the duty-cycles to the 2 and 3 inputs of ***DS1104SL_DSP_PWM***.
- Set the switching frequency in Unit as $10000Hz$.
- Double click the ***DS1104SL_DSP_PWM*** block, go to **PWM Stop and Termination**, uncheck "*Set all Ch*" and then click on "*Set all*".
- Connect the ***IA2*** channel to *ADCH6* on the controller box
- Make the load current available in the Simulink model by copying the first current measurement, and then double clicking on ***DS1104ADC_C5*** and changing it to channel 6.
- Save the model as ***Step1_04.mdl***. Your model should like the one shown in Fig. 4.1.

4.2.2 Creating the Control Desk Interface

- First build the DSP code out of the original Simulink model, saved in Step1_04.mdl.
- Open Control Desk and create a new experiment in the same working root as the simulink file.
- Create a new layout and drag two **Slider Gain** controls and two **Plotters**.
- Drag and drop the *V_motor* and *V_load* variables to the Slider gains.
- Assign one plotter to display *Ia* and *IL* currents and one plotter for the speed *wm*.
- In order to record the numerical values of currents and speed, add to your layout three **Display** type controls and assign them the speend and current label variables.

Your experiment should look like the one shown in Fig. 4.2.

4.3 Steady-state characteristics of dc-motor drives

In this experiment, you will derive the characteristics of a dc-motor, using the second motor as load. For a constant V_motor voltage, the load is modified using V_load. The MOTOR current and the speed are recorded in a table. A set of measurements are obtained for different supply voltages. The characteristics will be drawn using Matlab.

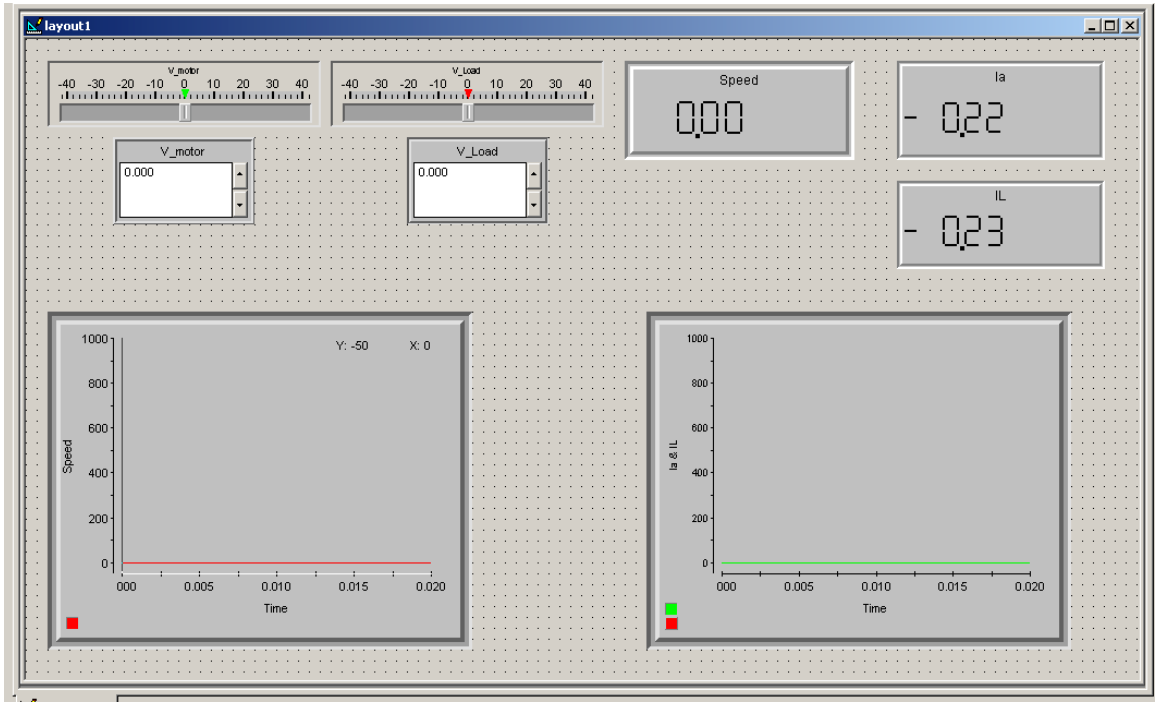


Figure 4.2: Control Desk Layout

4.3.1 Theoretical background

The steady-state mechanical characteristics of each dc-machine are the dependency between the electromagnetic torque (Nm) and the electrical radian speed (rad/s). Since the dependency is linear, the characteristics will be straight lines, with a constant slope for the whole voltage range 0 - V_{rated} and independent on the load. The motor equations reflect this linearity:

$$V_{motor} = R_a I_a + K_e \omega \tag{4.1}$$

$$T_e = K_t I_a \tag{4.2}$$

From Eqn. 4.1, one can obtain the steady-state motor characteristic:

$$\omega(I_a) = -\frac{R_a}{k_e} I_a + \frac{V_{motor}}{k_e} = mI_a + n \tag{4.3}$$

where

$$m = -\frac{R_a}{k_e} \quad and \quad n = \frac{V_{motor}}{k_e}$$

The steady-state model for the load can be approximated with a friction-type model, where the torque is proportional to the speed, and a constant friction torque is always present:

$$T_e = T_L + B\omega + T_{friction} \tag{4.4}$$

where all terms in the right member are load related. For our setup, where the load is a voltage controlled dc-machine, the load torque T_L is, in fact, the electromagnetic torque developed by the second dc-motor.

The parameter determination process for the steady-state model uses the current, voltage and speed measurements to obtain a linear approximation for both equation 4.3 and 4.4. Once the slope and the interception point are found, the parameters can be easily derived.

4.3.2 Steady-state parameters determination

According to the theoretical description from section 4.3.1, the motors will be driven in several steady-state operating points, as follows:

Estimation of k_e $(= k_t)$:

- At the rated armature voltage supplying the MOTOR, control the LOAD in the active region, such that the MOTOR current would become zero (I_a= 0), and measure the speed. Substituting value of I_a, Eqn. 4.1 becomes

$$k_e = \frac{V_{a\_rated}}{\omega} \quad (V_{a\_rated} = 42V) \tag{4.5}$$

- Increase V_MOTOR and decrease V_LOAD slider gains until V_MOTOR reaches 42 V. Notice the speed increasing to about 565 rad/s. Once the maximum voltage is obtained, decrease the V_LOAD voltage reference such that the active torque will decrease the MOTOR current until it becomes zero.
- Record the values for speed, at $I_{MOTOR} = 0$ and calculate k_e as per Eqn. 4.5.

Drawing the torque-speed characteristics:

- Maintain the MOTOR voltage at constant levels {42; 21; 10; 3 V}. Adjust the LOAD voltage reference such that the MOTOR current takes the following values at each voltage: {0; 1.0; 2.0; 3.0; 4.0; 5.0 A}.
- Record the speed (w_m), the LOAD current (I_{LOAD}) and LOAD voltage (V_{LOAD}) required to obtain the specified MOTOR currents. All current measurements will be multiplied with k_e to obtain the corresponding torque values - refer Eqn. 4.2.
- Fill in the TABLE 4.1 with the measured data.
- Draw the MOTOR and LOAD characteristics using the data acquired during the measurement process.

Table 4.1: Steady-state Experimental Data

V_Motor[V]	**V_Load[V]**	**I_Motor[A]**	**I_Load[A]**	**Speed[rad/s]**
42		0.0		
42		1.0		
42		2.0		
42		3.0		
42		4.0		
42		5.0		
21		0.0		
21		1.0		
21		2.0		
21		3.0		
21		4.0		
21		5.0		
10		0.0		
10		1.0		
10		2.0		
10		3.0		
10		4.0		
10		5.0		
3		0.0		
3		1.0		
3		2.0		
3		3.0		
3		4.0		
3		5.0		

- **Find the slope (m) and intercept (n) using MATLAB**:
  - The slope and the intercept of the linearized characteristic could be determined with the following instructions:
    p = polyfit(ia_data, w_data,1)
    m = p(1)
    n = p(2)
  - w_data is the array of speed data, while ia_data is the corresponding current data for the speed.
  - Two set of parameters (m,n) are determined for each dc-machine (MOTOR and LOAD).

- **Determine R_a, k_e**: Using the corresponding m and n, the machine parameters can be determined using the following equations

$$k_{e\_i} = \frac{V_{a\_i}}{n}$$

where i = 1...4, the number of voltage levels considered previously.

$$k_e = avg\{k_{e\_i}\}$$

since k_e is a machine constant, the best estimation is achieved by averaging the previous determinations.

$$R_a = -k_e m$$

A table for R_a, k_e final values is obtained (see Table 4.2).

Table 4.2: Calculation of the electrical steady-state parameters

Voltage[V]	m (slope)	n (intercept)	k_e[V/(rad/s)]	$R_a[\Omega]$
42				
21				
10				
3				

- **Determine the friction model parameters**:
  For determining the friction parameters, you will have to run the motor under no-load condition which you had done in the earlier experiment.
  Since the MOTOR has to overcome only the friction ($T_L = 0$), the electromagnetic torque ($T_e = k_e I_a$) will follow the linear friction model (see Eqn. 4.4) in steady-state. Fill in the Table 4.3 provided using the values obtained in the earlier experiment.
  Linearize the dependency of $T_e(w)$ by determining the (m, n) coefficients depicted in Eqn. 4.4 by using the same procedure as for drawing the torque speed characteristics.

- **Determine B and $T_{friction}$**: Using Eqn. 4.4, the friction parameters will be

$$B = m; \quad T_{friction} = n$$

  Fill in the Table 4.4 provided.

4.4 Dynamics of DC Machine

In this section we will try to derive the dynamic characteristic of DC drives. The dynamics can be divided into electrical and mechanical dynamics. Independently studying both transients, two motor parameters can be determined: armature inductance (L_a) and moment of inertia (J).

Table 4.3: Steady-state friction model characteristic

V_motor[V]	Speed[rad/s]	Current[A]	Torque[Nm]
1			
3			
5			
10			
15			
21			
30			
42			

Table 4.4: Calculation of the mechanical steady-state parameters

m (slope)	n (intercept)	B[Nm/(rad/s)]	$T_{friction}$[Nm]

For this section we shall use the same Simulink model and dSPACE layout as in earlier section. To analyze the dynamics of a dc-motor drive some theoretical background will be presented, then using the already determined steady-state data, the experiment steps will be described.

The setup consists of two dc-motors, axially coupled and supplied from two converters, in four-quadrant configuration. One motor is controlled in current, such that it will act as either a passive or an active load. This motor will be named here LOAD. The second motor is controlled in open-loop, with variable voltage. This motor will be named here MOTOR.

In this experiment two parameters need to be determined: L_a - armature inductance $[H]$ and J - moment of inertia $[kg.m^2]$

4.4.1 Dynamic Model Characteristic

There are several ways to determine the inductance and the inertia. All methods involve the dynamic analysis of the machines in transient operation. The dynamic equations of a dc-motor are:

$$V_a = R_a i_a + k_e \omega + L_a \frac{di_a}{dt} \tag{4.6}$$

and

$$T_e = T_L + T_{friction} + B\omega + J\frac{d\omega}{dt} \tag{4.7}$$

The system of two first order differential equations shows that the dc-motor is a second order system. The two state variables, armature current (i_a) and angular speed (ω), are not independent.

Therefore, the inductance (L_a) and the moment of inertia (J) would both contribute to the variation of each of the two state variables.

It is convenient to "isolate" the variations described in 4.6 and 4.7, thus only a first-order differential equation has to be solved for each variable. Two sets of experiments are then required to determine L_a and J, while keeping the speed and, respectively, the current to a zero value.

Inductance Determination

To estimate the armature inductance, the motor must be held standstill ($\omega = 0$). Block the rotor with a mechanical brakes and then apply a step voltage at the armature terminals. The current increases exponentially in time, and equation 4.6 becomes:

$$V_a = R_a i_a + L_a \frac{di_a}{dt} \tag{4.8}$$

Solve the above Eqn. 4.8 for the current, we get

$$i_a(t) = \frac{V_a}{R_a}\left(1 - e^{-\frac{t}{\tau_a}}\right) \tag{4.9}$$

where

$$\tau_a = \frac{L_a}{R_a}$$

The current increases exponentially to the final value equal to $\frac{V_a}{R_a}$. The slope of this exponential curve, measured at $t = 0$, is depenedent on the value of inductance (L_a) as given below :

$$\left.\frac{di_a}{dt}\right|_{t=0} = \frac{V_a}{R_a}\frac{R_a}{L_a}e^{-\frac{R_a}{L_a}.0} = \frac{V_a}{L_a} \tag{4.10}$$

A graphical determination of the slope, at a given voltage, would lead to the determination of the motor inductance (L_a).

Determination of inertia

To make electrical torque (T_e) effect equal to zero in the mechanical dynamics eqn. 4.7, it requires a complete shutdown of the motor supply. The motor is brought to a no-load speed ω_0, such that the armature current is zero and the friction losses are compensated by controlling the active load. At $t = 0$ the whole system is shutdown. This implies that the electromagnetic torques in both the MOTOR (T_e) and the LOAD (T_L) becomes zero. The dynamic equation will become

$$0 = T_{friction} + B\omega + J\frac{d\omega}{dt} \tag{4.11}$$

The speed decreases exponentially in time:

$$\omega(t) = \left(\omega_0 + \frac{T_{friction}}{B}\right)e^{-\frac{B}{J}t} - \frac{T_{friction}}{B} \tag{4.12}$$

Our friction model is approximate. The "constant" friction torque is, in fact, dependent on speed. It should cancel once the speed reaches zero, otherwise it would become an active torque and drive the motor into the fourth quadrant. Rearranging equation 4.12:

$$\left(\omega(t) + \frac{T_{friction}}{B}\right) = \left(\omega_0 + \frac{T_{friction}}{B}\right) e^{-\frac{B}{J}t} \Longrightarrow \Omega(t) = \Omega_0 e^{-\frac{B}{J}t} \tag{4.13}$$

from the above eqn. 4.13, it is clear that by up-shift the exponential curve with a constant, interia is just dependent on the slope of the curve at $t = 0$

$$\left.\frac{d\Omega}{dt}\right|_{t=0} = \left.\frac{d\omega}{dt}\right|_{t=0} = -\Omega_0 \frac{B}{J} \tag{4.14}$$

Knowing ω_0 and B, and graphically determining the slope of the curve, one can calculate the motor equivalent inertia J.

***Note**: Inertia is a scalar. The value obtained is the algebraic sum of each motor's inertia.*

4.4.2 Simulink model for dynamic parameter determination

The model presented in section 4.3 will be used for this section. However, one additional block has to be inserted, such that a step command in voltage is possible.

Both, electrical and mechanical dynamic parameters require either a positive (from 0 to V) or a negative (from V to 0) step change in supply voltage.

The $V = 0$ condition implies also that the motors have no armature current, i.e. they do not enter the regenerative braking mode with negative torque. To achieve open-circuit of the motors armatures, the converters need to be shutdown, and all the switches opened when the $V = 0$ command occurs. This operation is possible by using the **SHUTDOWN** signal on the drives board. The **SHUTDOWN** signals are controlled by the ***digital I/O*** channels 11 and 12. When ***IO11/12*** is **0** (OFF state) the switching signals are inhibited and the switches are opened. Setting ***IO11/12*** to **1** (ON state) and resetting(***IO10***) resumes the regular operation of the converters. The ***IO10/11/12*** digital channels will be added as slave bit out blocks for our model from the slave library. In addition two constant blocks and two Boolean conversion blocks should be added with **SD1** and **SD2** using the same signal. The model should like the one shown in Fig. 4.3.

4.4.3 dSPACE Experiment Layout for dynamic model determination

- A new experiment will be created in the same directory containing the **Step1_04.mdl** file.
- Copy the Layout from the previous experiment and rename it.
- Add 2 *CheckButtons* controls to the Layout.

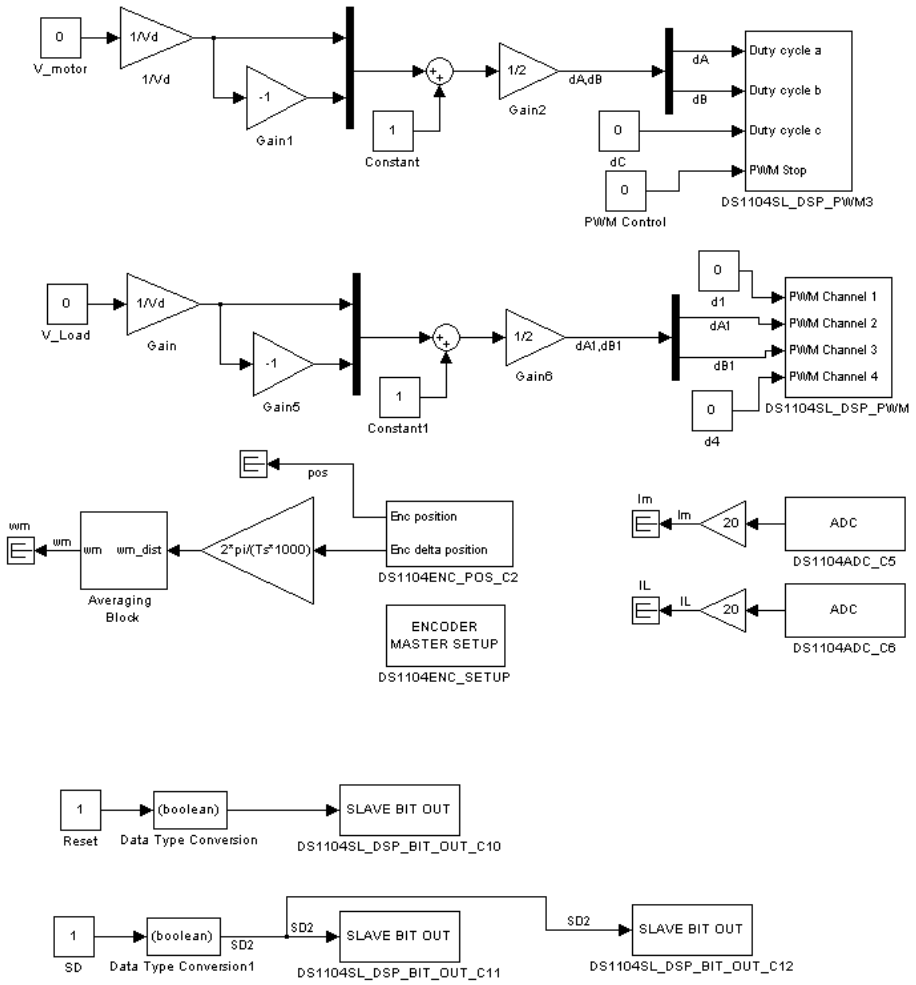


Figure 4.3: Simulink Model for Dynamic Parameter Characterisation

- Link all variables from the model with the controls in the Layout, as in earlier section.
- Link the **Reset** and **SD** to the *CheckButton* control.

The layout of the experiment is shown in Fig. 4.4.

4.4.4 Dynamical parameters determination

Inductance determination

- Using the blocking device, block the rotors firmly.
- Uncheck and then recheck the SD control. This button works as a switch to connect and

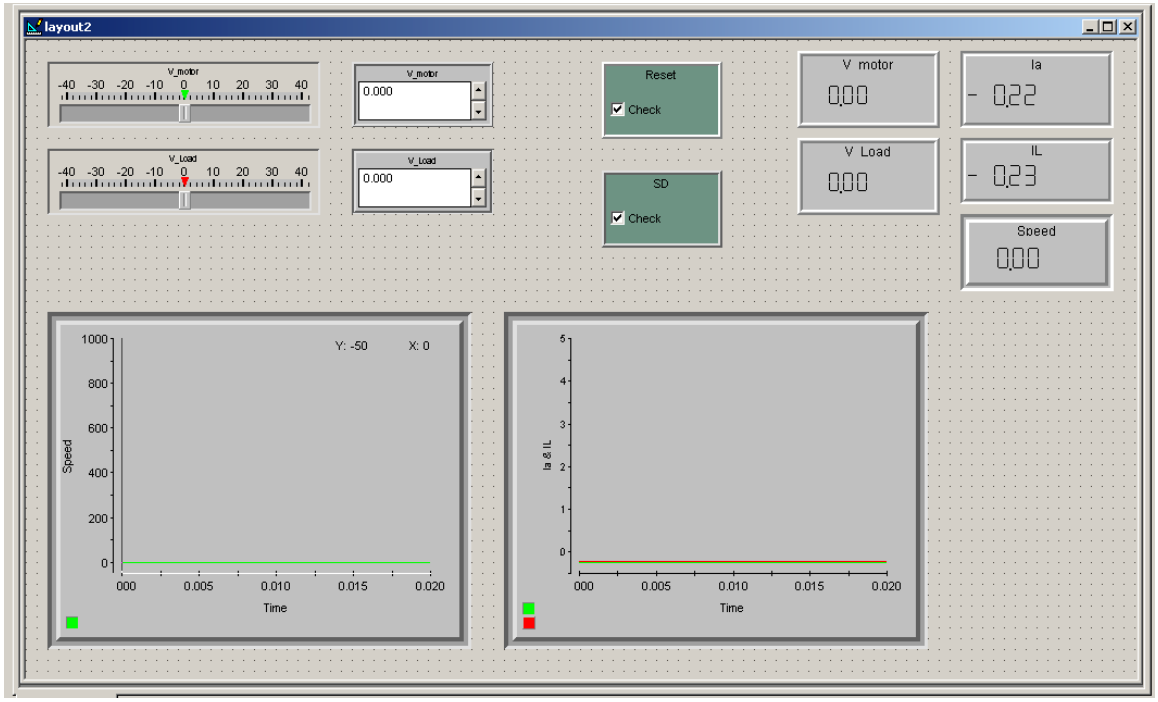


Figure 4.4: Control Desk Layout for Dynamic Parameter Characterisation

disconnect the machines from the power supply.

- Set the V_MOTOR to a low value (around 3 V) and uncheck Reset to give a step in voltage and then recheck Reset. The current should increase exponentially (as shown in Fig. 4.6) and reach a constant steady state value.

- To save the current response, open View/Controlbars/Capture Settings Window. Drag the P:Reset signal from the Tool Window into the gray box situated below the Level - Delay set boxes. Check the box called On/Off, check the edge direction, and set the Level value to 0.5.

- Now, you will observe that every time you uncheck the Reset control in your layout the plot area will display the current and it will stop when it reaches the maximum measurement time. Set the **Length** to 0.2 (see Fig.4.5). This will set the data capture time as 0.2s which is large enough to observe the whole transient process in current.

- Check and uncheck SD and Reset to make some measurements. You current waveform will look some what similar to the one shown in Fig. 4.6. After you are satisfied with the data displayed go to the Capture Settings Window and press the SAVE button. The dialog box will ask you to name the .mat file that will contain the graphic data in all plot areas.

- To extract the inductance information as in 4.9, this equation will be used in a Matlab program to determine the slope of the current transient. Run "Inddet.m" at the Matlab

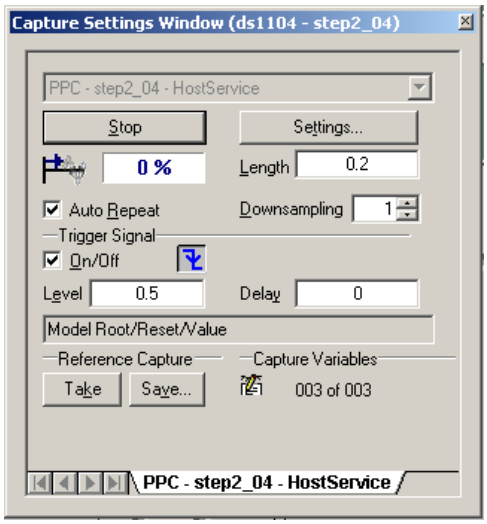


Figure 4.5: Current Waveform

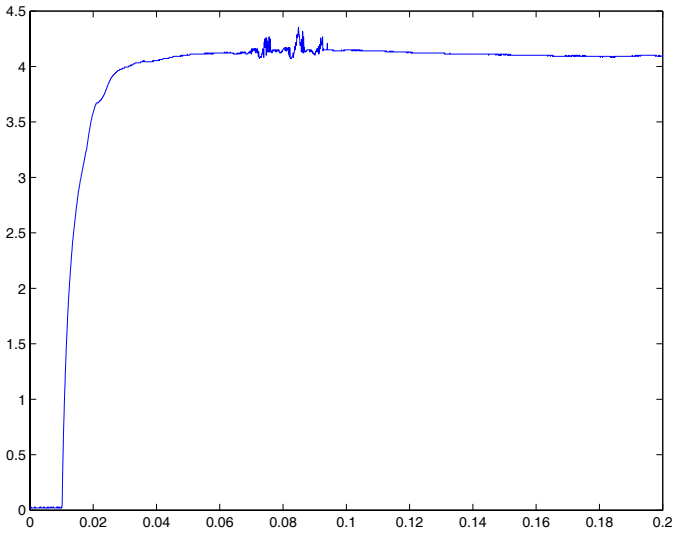


Figure 4.6: Current Waveform

prompt. See the program listing in Appendix 1.

Inertia determination

- In the Capture Settings Window two modifications must be made: Increase the display length to 2 seconds and change the Trigger Signal to SD.
- Check the SD control and increase the voltage on the MOTOR to 42 V. Decrease the LOAD voltage reference to negative values, such that the MOTOR current becomes zero. The digital

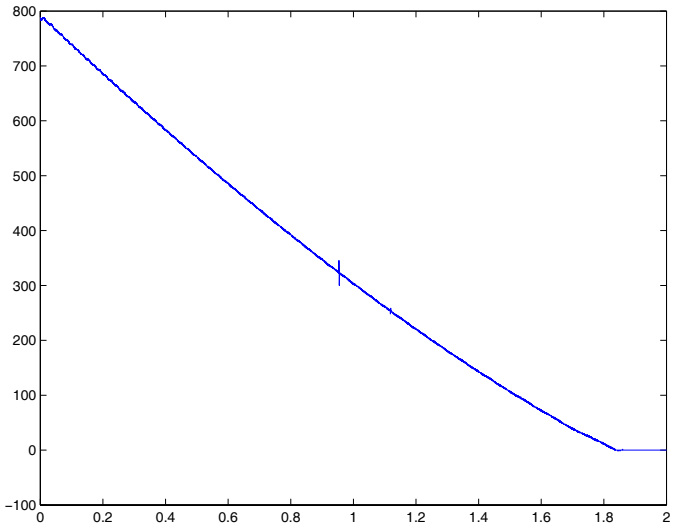


Figure 4.7: Current Waveform

displays let you monitor the operations.

- Record the speed value at this operating point.
- Uncheck the Shutdown button. This will initiate the display process and, after two seconds, the speed plot will stop and a decreasing exponential curve will be obtained (see Fig. 4.7).
- Press again the SAVE button in the Capture Settings Window and store the data in another .mat file. Run "Inerdet.m" at the Matlab prompt and record the result for the moment of inertia. The program listing is shown in Appendix 2.

4.5 Lab Report

The lab report should be brief and should contain the details asked below

- Complete all the tables.
- Provide all the machine parameters which you obtain using the above procedure.
- Provide all the graphs you obtain while finding the machine parameters.

```
function err = fitcur(lambda)
global Plothandle t y y0;
A = zeros(length(t),length(lambda));
for j = 1:length(lambda)
    A(:,j) = exp(-lambda(j)*t)';
end
c = A\(y-y0)';
z = A*c;
set(Plothandle,'ydata',z+y0);
drawnow;
err = norm(z-(y-y0)');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inddet.m       %
%       %
% Calculation of dc-motor inductance and resistance using       %
% a curve-fitting algorithm for the current response at a step   %
% in voltage and blocked rotor      %
%       %
%                                                                 %
% Author : Razvan Cristian Panaitescu                  %
% Date   : Feb 06, 2002            %
% University of Minnesota      %
% Department of Electrical and Computer Engineering        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


global Plothandle t y y0; % Variable declarations

% Load the current.mat file saved from the dSPACE Experiment
% The file contains the current response at a step in voltage
load current;
t=current.X.Data; % Define time variable
y=current.Y(:,2).Data; % Define the y-variable(current); 2 is sometimes a 1
                       % depending on ControlDesk setup

% Calculate the steady-state final value by averaging
% the current data during the last half of the time-interval
nmax=length(y);
y0=mean(y((nmax-1)/2:nmax));

% Plot the initial saved data
cla reset;
axis([0 .2 0 3.5]); % axis limits depend on the actual data
hold on;
plot(t,y,'r','EraseMode','none');
```

```
title('Armature current response');
hold on;
Plothandle = plot(t,y,'EraseMode','x');

% Run the curve-fitting algorithm using FITRAZ(lambda) and FMINSEARCH
% FITRAZ and FMINSEARCH returns the error between the data and the values
% computed by an exponential function of lambda.
% FITRAZ assumes a function of the form
% y = y0 + c(1)*exp(-lambda(1)*t) + ... + c(n)*exp(-lambda(n)*t)
% with n linear parameters and n nonlinear parameters.

lam0 = [100]; % Define one initial root (n=1)
lambda = fminsearch('fitraz',lam0); %Start iteration using initial root
hold off;



% Computing the motor parameters
% The algorithm returns the best lambda=Ra/La
% With the steady state current i0=y0 calculate the armature resistance
V=3;
i0=y0
Ra=V/i0
La=Ra/lambda
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% inerdet.m %
%     %
% Calculation of dc-motor inertia using a curve-fitting algorithm    %
% for the speed response on shutdown from steady-state operation    %
%                                                                   %
% Author : Razvan Cristian Panaitescu                       %
% Date   : Feb 06, 2002                %
% University of Minnesota    %
% Department of Electrical and Computer Engineering    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global Plothandle t y y0; % Variable declarations

% Load the speed.mat file saved from the dSPACE Experiment
% The file contains the speed response at shutdown
load speed;
t=speed.X.Data; % Define time variable
y=speed.Y(:,3).Data; % Define the y-variable (speed)

% Calculate the steady-state (final) value
B=0.000252; % These parameters were determined
Tf=0.0979; % in the previous experiment steps
y0=-Tf/B;

% Plot the initial saved data
cla reset;
axis([0 2 0 450]); % axis limits depend on the actual data
hold on;
plot(t,y,'r','EraseMode','none');
title('Speed response');
hold on;
Plothandle = plot(t,y,'EraseMode','x');

% Run the curve-fitting algorithm using FITRAZ(lambda) and FMINSEARCH
```

```
% FITRAZ and FMINSEARCH returns the error between the data and the values
% computed by an exponential function of lambda.
% FITRAZ assumes a function of the form
% y = y0 + c(1)*exp(-lambda(1)*t) + ... + c(n)*exp(-lambda(n)*t)
% with n linear parameters and n nonlinear parameters.

lam0 = [1]; % Define one initial root (n=1)
lambda = fminsearch('fitraz',lam0); %Start iteration using initial root
hold off;

% Computing the motor parameters
% The algorithm returns the best lambda=B/J
wfin=y0
J=B/lambda
```

Experiment 5

DC Motor Control

5.1 Introduction

The purpose of this experiment is to design and implement a cascade control of a dc-motor drive. We shall use the Lab-Kit dc-motor for which the parameters were calculated in the previous experiment. The controllers will be designed and will be tested on a simulation model of the dc-motor. Once the parameters are tuned, the model of the dc-motor will be replaced with the real motor and the control algorithm will be downloaded into the dSPACE board.

5.2 Simulink model of the dc-motor

The equations written in Laplace transform for a dc-motor were derived in Chapter 8. If the voltage and the current are assumed not to contain switching-frequency components, the motor model can be written:

$$I_a(s) = \frac{V_a(s) - E_a(s)}{R_a + sL_a}; \qquad E_a(s) = k_E.\omega_m(s) \tag{5.1}$$

$$\omega_m(s) = \frac{T_{em}(s) - T_L(s)}{sJ_{eq}}; \qquad T_{em}(s) = k_T I_a(s); \qquad k_T = k_E \tag{5.2}$$

Eqn. 5.1 and eqn. 5.2 can be easily be implemented in Simulink using standard blocks as presented in Fig. 5.1

- Create a new Simulink model and drag all blocks as shown in Fig. 5.1.
- Make the connections and define the parameters as shown in Fig. 5.1.

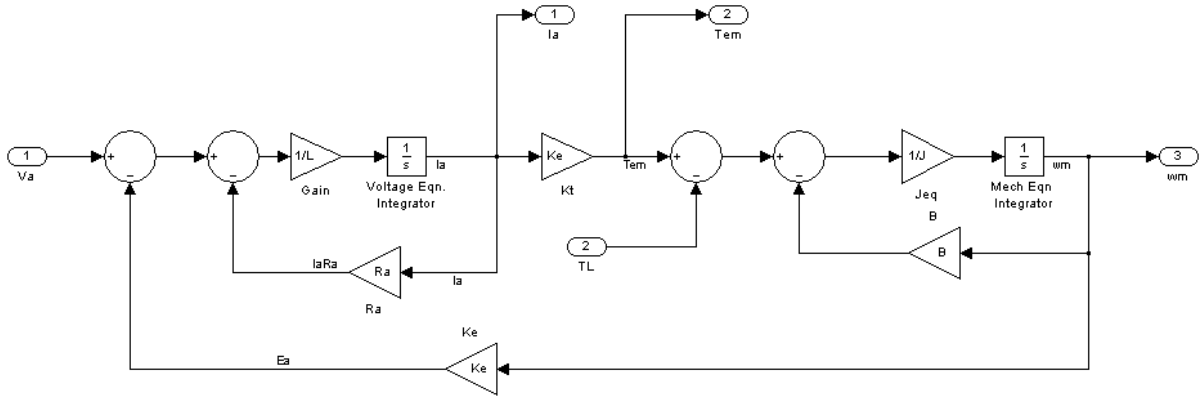
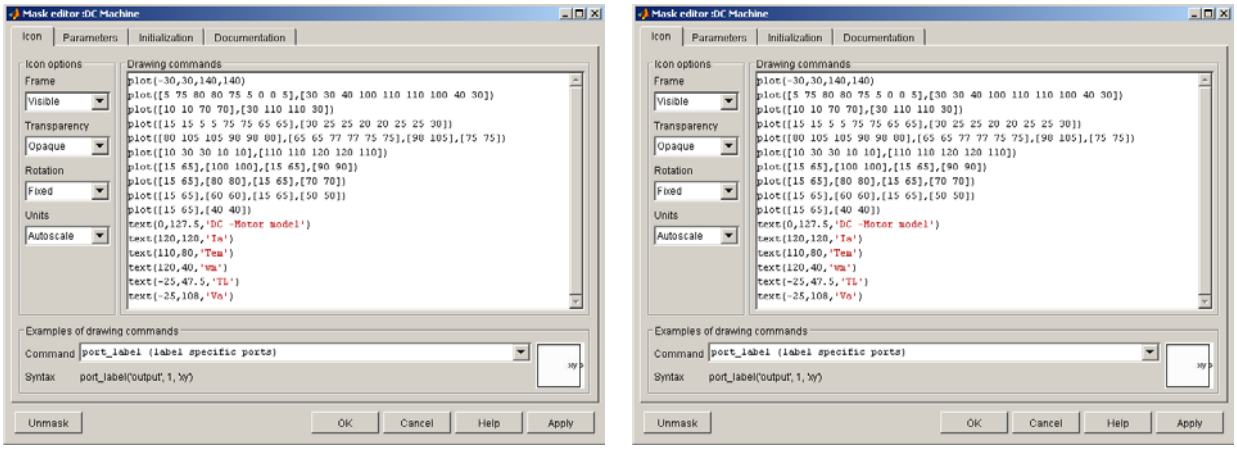


Figure 5.1: Simulink model of DC Motor

The representation in Fig.5.1 uses integrators instead of transfer functions. This allows to set initial conditions for the current and speed state variables. The model also includes the friction coefficient B. However, during simulations, B can be considered zero, and the model will be similar to the one described by eqn. 5.1 and eqn. 5.2. Note that the torque constant is already replaced with the voltage constant in the Gain block at the top of the figure.

- The model can further be grouped and masked in Simulink, such that all parameters can be inserted in a dialog box, by selecting the system and creating a sub-system. Modify the mask editor as seen in Fig. 5.2. After modifying the mask, double clicking on the subsystem should bring up a window to enter the DC motor parameters.



(a) DC Motor Icon

(b) DC Motor Parameters

Figure 5.2: Masked Model of DC Motor and Drawing commands for Graphical Representation

- Now enter the values of DC motor parameters which we have evalued in earlier experiment.

5.3 Controller Design

Once the dc-motor model is built, the controllers can be added and tuned. Start with the current loop for which a PI controller is required.

- The model for a PI controller is first created - see Fig.5.3. Double click the integrator block and enable limit output. Then set the Upper and Lower saturation limits to +lim/-lim. The lim value should be set to 1 as the control voltage maximum value will be ± 1, which is the input to K_{pwm} block. The resultant maximum value of voltage applied to the DC motor will be ± 42 which is the rating of the DC motor.

- The armature current is fed back to the controller input.

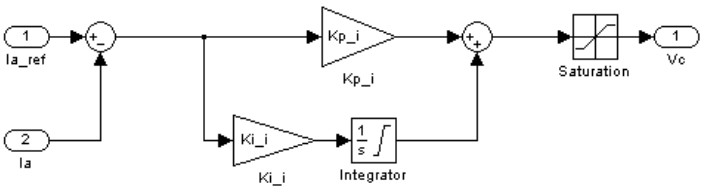


Figure 5.3: PI controller Model

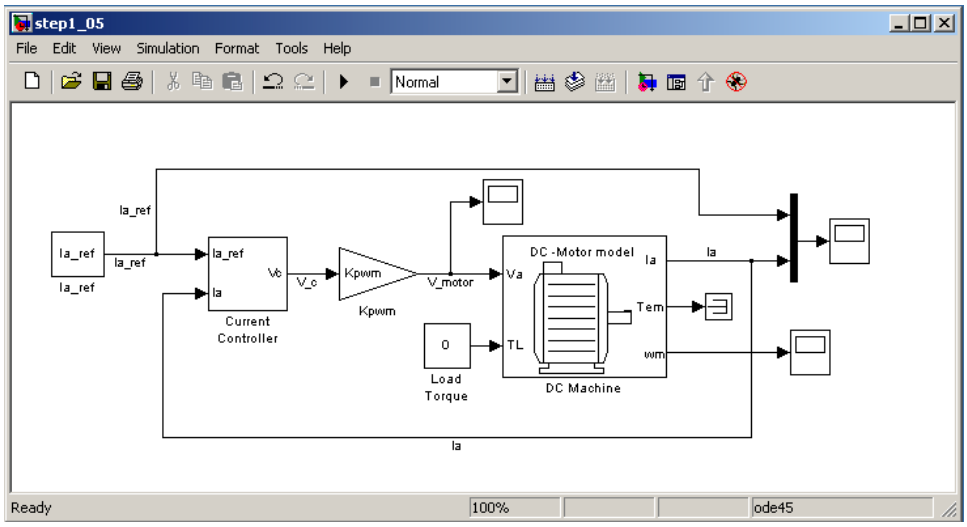
- The parameters of the PI controller (namely Kp_i and Ki_i) are computed using the motor parameters which were evalued in the earlier experiment and the algorithm described in section 8-7-1 of Electric Drives book for 500Hz bandwidth.

- The Saturation block sets the maximum and minimum limits for the control voltage (in our case ± 1).

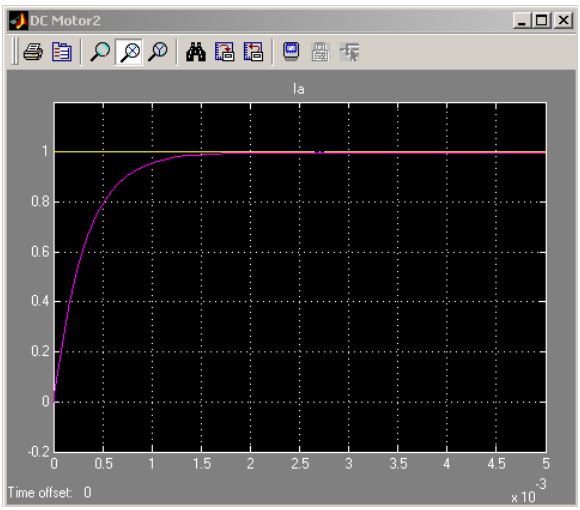
- Set the value of Kp_i and ki_i in Matlab prompt. Also set the values of lim = 1, Kpwm = 42, Ia_ref=1 in Matlab prompt.

Instead of setting the values of various variables in Matlab prompt, create a m-file which contains the values of all the variables. Run the m-file before running the simulation, which will load the values of all the variables.

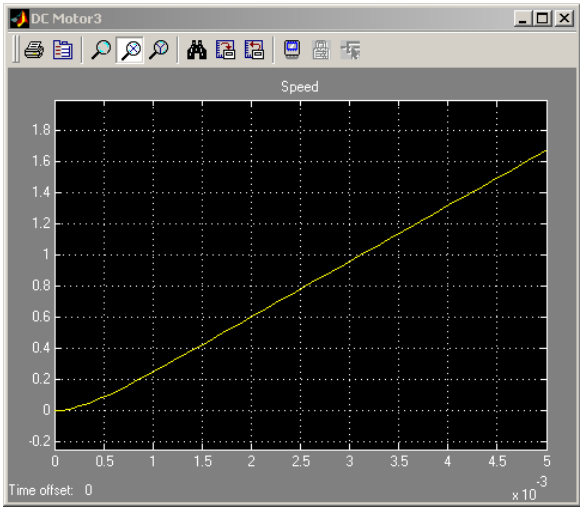
- Running the Simulink model for the current controller with reference current as 1A, results similar to the Fig.5.4(b) and Fig. 5.4(c) will be obtained.



(a) Simulink Model for Current Control Loop



(b) Current Waveform for 1A Reference Current



(c) Speed Waveform for 1A Reference Current

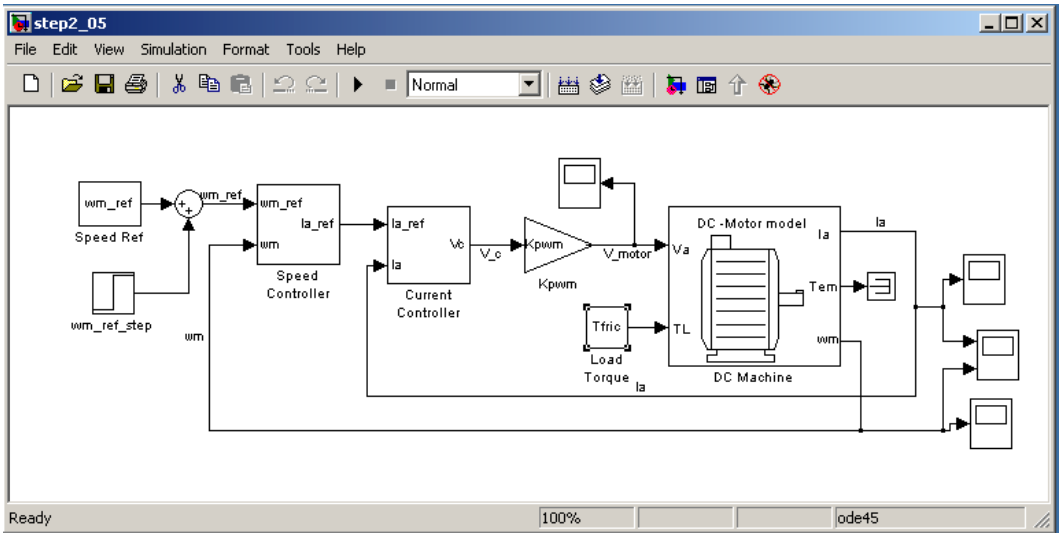
Figure 5.4: Simulink Model and Results for Current Control Loop

- Once the response in current is considered optimal (low overshoot, fast rise-time, zero steady state error), the speed controller can be designed.

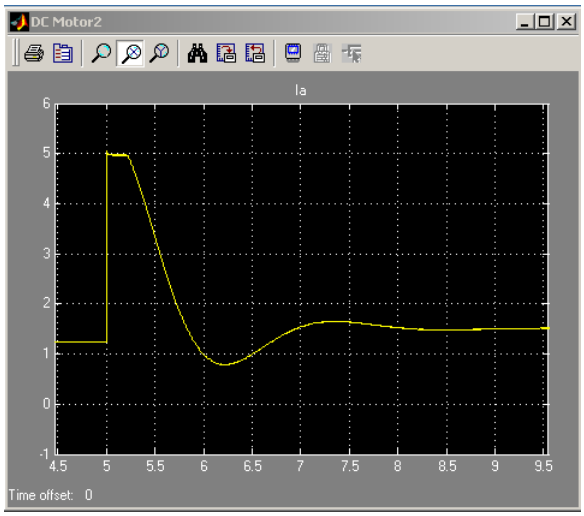
- A similar PI controller for the speed loop will be added to the Simulink model.

- Follow the algorithm described in paragraph 8-7-2 to design the speed control loop for 1Hz bandwidth, using the motor parameters determined in earlier experiment.

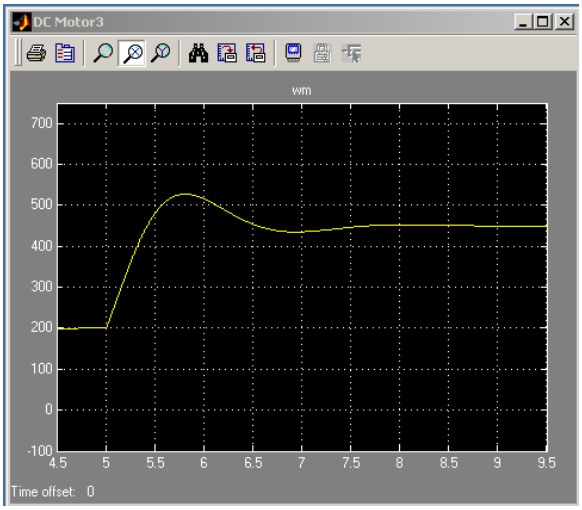
The Simulink model for the cascade control and the waveforms for speed and current are shown in Fig.5.5.



(a) Simulink Model for Cascade Control



(b) Current waveform for step change in speed



(c) Speed waveform for step change in Speed

Figure 5.5: Simulink Model and Results for Cascade Control

The Speed PI controller has a current limit output of ±5A, necessary to limit the current during transients (both in simulation and real-time systems). To check the controller design, we will give a step change in the speed reference. In the example of Fig.5.5, the speed is commanded to a step change to 200rad/s at $t = 0s$, then at $t = 5s$ its is changed to 450 rad/s. The above reference speed

command is implemented using a constant and step source blocks. The results of cascade control are shown in Fig. 5.5(b) and Fig. 5.5(b).

If the controller parameters were correctly tuned, then it's time to go on for the next step, and implement the control algorithm in a real-time system.

5.4 Real-time implementation of feedback control

For dSPACE implementation, the dc-motor model will be replaced with the real motor and Kpwm block will be replaced by power converter with 42V dc supply. The control voltage to duty cycle conversion was already discussed and implemented in earlier experiments.

- Add the reset block used in the earlier experiment.
- Modify the Speed Control block as shown in Fig. 5.6. Change the integrator block parameters by double clicking on it and changing its external reset to either. Open the Current Controller and change its integrator's reset as was done in the Speed Control. Connect the reset inputs of speed controller and current controller as shown in Fig. 5.7. These changes allow the integrators to start up correctly in the real-time environment.

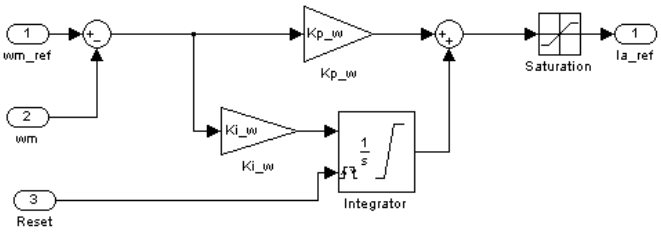


Figure 5.6: Simulink model for real-time speed control of DC Motor Control

- Remove the DC-motor mask model and gain Kpwm block.
- Copy and paste the duty-cycle calculator from the Simulink model used in earlier experiments (see figure below).
- The current and the speed are to be measured. For measurements use the blocks already designed in earlier experiments.
- Replace the speed ref, wm_ref_step and sum block with a constant block for setting the speed reference.

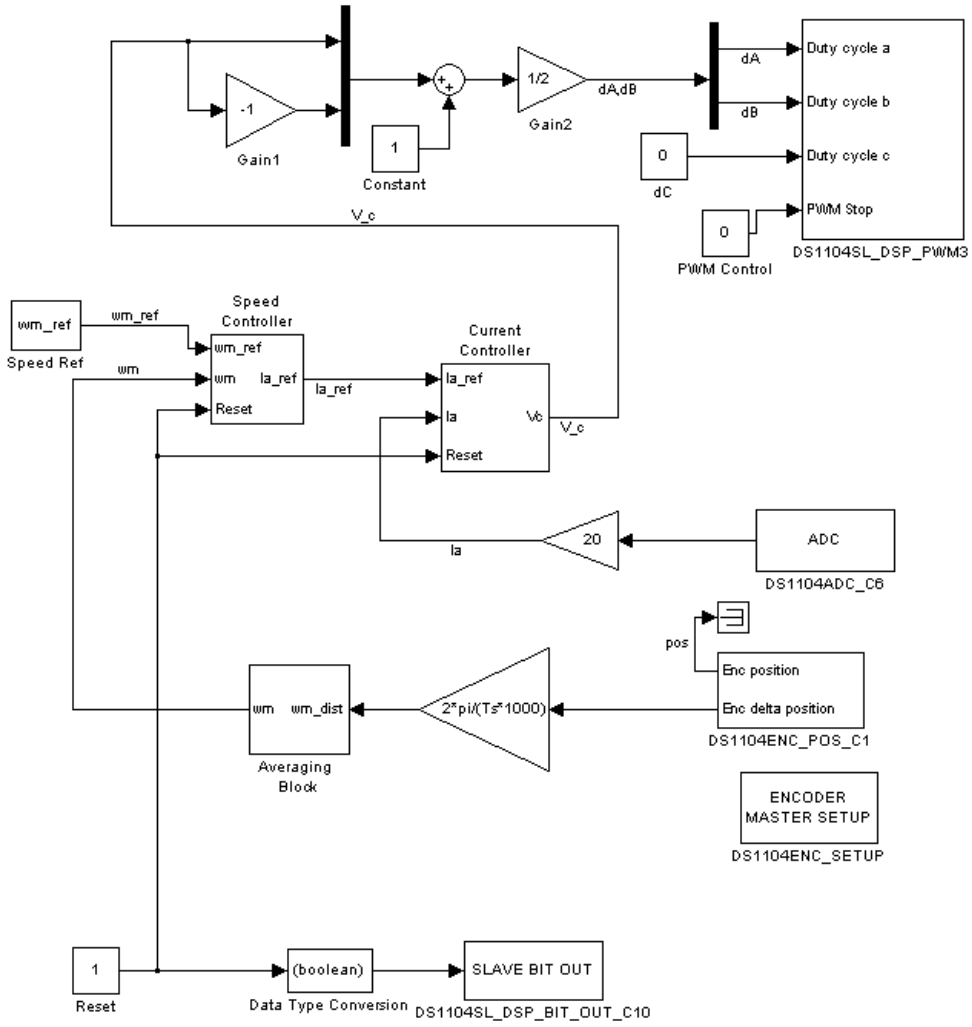


Figure 5.7: Simulink model for real-time implementation of DC Motor Control

- At the Matlab prompt, set the sampling time Ts=0.0001 and the dc-bus voltage at Vd=42V. Also set the values of various variables you have defined in the model.
- Change the switching frequency in the DS1104SL_DSP_PWM3 to be 50000Hz.
- Set the simulation parameters with Ts as sampling time and inf as total simulation time. Now, the model looks like in Fig.5.7.
- Build (CTRL+B) the model and start Control Desk.
- Create a new Experiment and set the working root the same as the path for the Simulink model.
- Create a new Layout and add some controls as shown in Fig.5.8.

- Run the experiment and compare the real-time results with the simulations.

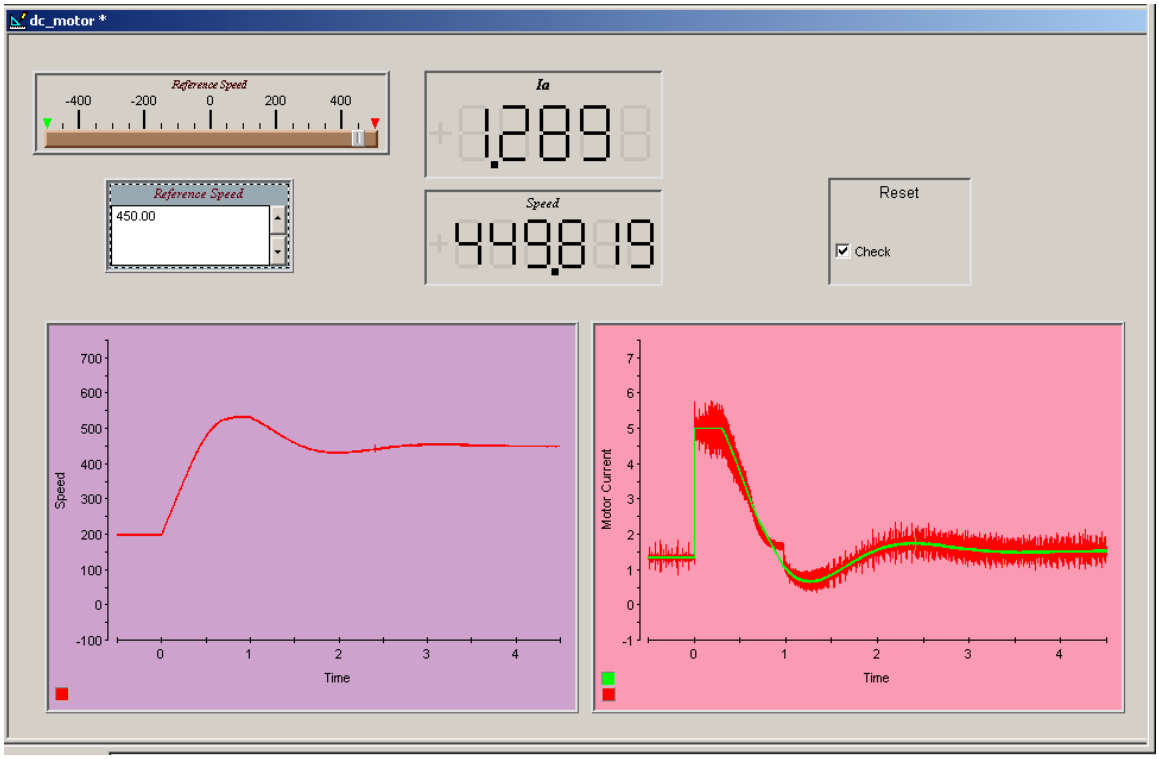


Figure 5.8: Control Desk interface for DC Motor control (Step change from 200 - 450 rad/s)

Experiment 6

Frequency Control Of AC-Motor Drives

1 Introduction

The experiment intends to design and implement a V/f control algorithm for an induction motor. First, a switch-mode sinusoidal PWM algorithm is implemented for a three-pole switch-mode converter. Then the V/f strategy is designed and the experiment is downloaded in the dSPACE control board to run a 42V induction motor.

2 Simulink model

To run an induction motor, we need to supply it with a sinusoidal voltage variable in amplitude and frequency. In electric drives and other power electronic applications, such as in switch-mode rectifiers, the intent is to supply three-phase sinusoidal currents. In the system of Fig.1a, the neutral "*n*" is the star-connection node of the induction motor stator windings. At switching frequencies relatively high compared to the fundamental frequency of synthesis, it is possible to represent the switching circuit of Fig.1a by means of ideal transformers in terms of switching average quantities, as shown in Fig.1b. Average variables with a bar ("—") on top do not contain switching frequency ripple.

The intent to supply balanced sinusoidal currents into a balanced ac motor implies that the line-to-line voltages across the motor terminals be sinusoidal in steady-state. In a balanced ac motor, it can be shown that the motor phase voltages (v_{an}, v_{bn} and v_{cn}) are also sinusoidal and, similarly to the motor currents, the three phase voltages sum to zero on an instantaneous basis.

$$v_{an}(t)+v_{bn}(t)+v_{cn}(t)=0 \tag{1}$$

Of course, the above equation is also valid on an average basis:

$$\overline{v}_{an}(t)+\overline{v}_{bn}(t)+\overline{v}_{cn}(t)=0 \tag{2}$$

when,

$$i_a(t)+i_b(t)+i_c(t)=0 \tag{3}$$

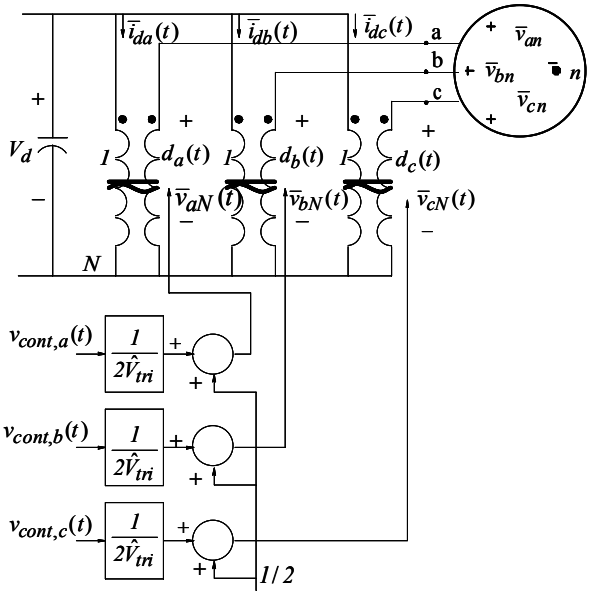


Fig. 1. (a) Switch-mode inverter for ac drives;
(b) Average representation of the three phase converter

2.3 Sinusoidal Pulse-Width Modulation

A traditional approach is sinusoidal pulse width modulation, where sinusoidal control voltages are compared with a triangular carrier signal to generate the switching functions $q_{a,b,c}$ in Fig.1a. The resulting duty factors (transformer turn-ratios in Fig.1b) are based on the following expression:

$$d_a(t)=\frac{1}{2}+\frac{1}{2}\frac{v_{cont,a}}{\hat{V}_{tri}}=\frac{1}{2}+\frac{\overline{v}_{an}}{V_d}\quad(\text{etc.}) \tag{4}$$

Since $\overline{v}_{aN}=d_a\cdot V_d$, $\overline{v}_{bN}=d_b\cdot V_d$, $\overline{v}_{cN}=d_c\cdot V_d$, expressions for $\overline{v}_{aN},\overline{v}_{bN},\overline{v}_{cN}$ can be similarly written, using (4):

$$\overline{v}_{aN}=\frac{V_d}{2}+\frac{1}{2}\cdot\left(\frac{V_d}{\hat{V}_{tri}}\right)\cdot v_{cont,a}\quad(\text{etc.}) \tag{5}$$

The inverter terminal voltages, with respect to the negative dc-bus *N*, contain a dc-offset $V_d/2$ which is of zero-sequence. This zero-sequence cancels out in the motor voltages $\overline{v}_{an}$, $\overline{v}_{bn}$, $\overline{v}_{cn}$.

2.2 Simulink implementation

To implement the algorithm in Simulink, we shall first assume that the three-phase voltages at the stator terminals must have the following expressions:

$$\begin{aligned}\overline{v}_{an}(t) &= V_m \cos(2\pi f \cdot t)\\ \overline{v}_{bn}(t) &= V_m \cos\left(2\pi f \cdot t - \frac{2\pi}{3}\right)\\ \overline{v}_{cn}(t) &= V_m \cos\left(2\pi f \cdot t - \frac{4\pi}{3}\right)\end{aligned} \tag{6}$$

The frequency *f* and the amplitude V_m are variables. However, the V/f control algorithm implies that there is a relationship between the amplitude of the voltage and the frequency, i.e. the ratio between the two quantities is constant:

$$K = \frac{V_m}{f} \tag{7}$$

Now, setting the frequency and knowing the constant *K* it is easy to derive the amplitude of the voltage, using (7).

The expressions for stator voltages are written in continuous domain, where the time is a continuous variable. For a digital system, time is a discrete variable, depending on the sampling time. Since a new value of the voltage is calculated every T_s seconds, the time will have to be updated by adding the sampling time to the precedent time value:

$$t_k = T_s + t_{k-1} \tag{8}$$

➲ This is easily done using a *Memory* block in Simulink, shown in Fig.2. In the model the time is already multiplied by $2\pi f$ to give the ωt variable.

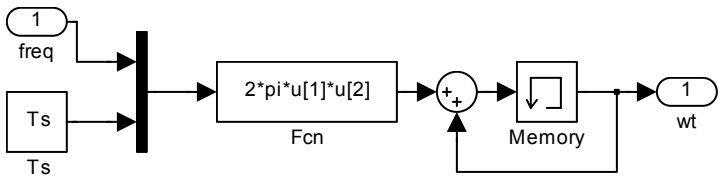


Fig.2. Simulink model for the voltage argument calculation

➲ Mask this model and don't forget to set the parameter in the *Memory* block to T_s. This ensures that the delay performed in time calculation will be exactly of one sampling time.

➲ Use three *Function* blocks to implement the relations in (6) and (4). The model becomes:

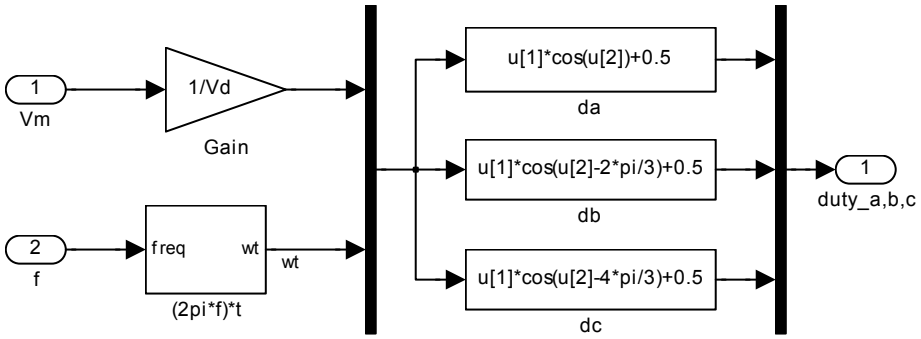


Fig.3. Voltage and duty-cycle calculation (using eq. (4) and (6))

➲ After masking the blocks in Fig.3, the Simulink model can be completed with the command value of the frequency, the K=V/f constant and the *DS1104SL_DSP_PWM3* block.

➲ A *Rate Limiter* block is inserted in the reference frequency path, to set the acceleration and deceleration times. Limitation of these two transients is necessary to limit the stator currents during start-up or breaking.

The *Sinusoidal PWM* block contains the masked model from Fig.3.

Of course, in our experiment we would like to monitor the speed and at least one stator current. Complete your model as follows:

➲ A duty cycle limiter must also be added the bounds should be set to 0.1 and 0.9 to ensure that the motor drive board works properly.

➲ Copy and paste the speed measurement blocks from a previous experiment

➲ Copy and paste the A/D current measurements from the dc-motor experiment.

➲ The model should be similar to the one in Fig.4.

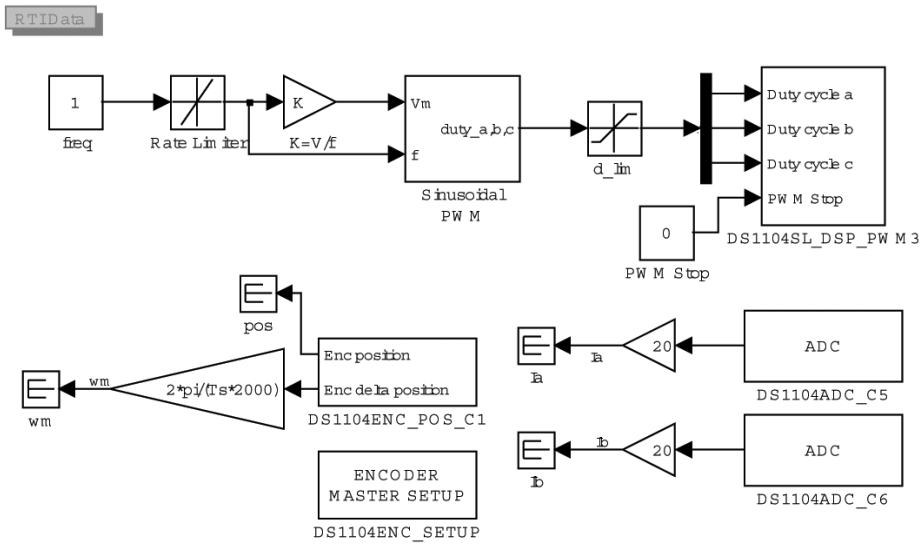


Fig.4. Complete Simulink model of experiment 6

➲ Make sure you define at Matlab prompt all the necessary parameters: V_d, T_s, K.

➲ Set the simulation parameters as in previous experiments.

➲ Set the switching frequency in the *DS1104SL_DSP_PWM3* block to 20000Hz.

➲ Build (CTRL+B) the model and obtain an **.obj* file.

3 dSPACE Implementation

➲ Start the ControlDesk and a new experiment in the same working root as where your model was saved.

➲ Create a new Layout and drag some controls and plotters:

- One slider gain for the frequency command
- Two Numeric Inputs for your acceleration and deceleration times
- One Numeric Input for the boost voltage.
- Three plotters: one for phase currents, one for speed and one for the duty-cycles.

➲ Your layout should look like in Fig.5.

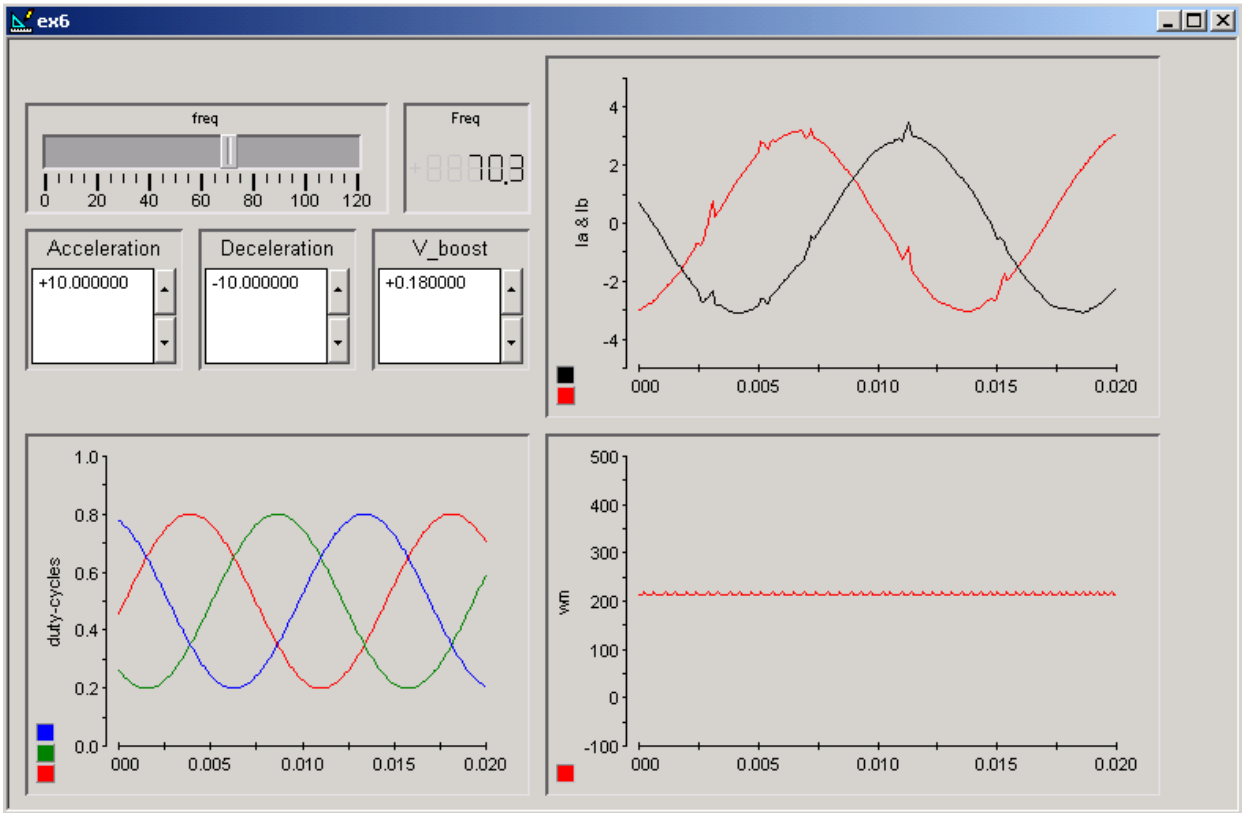


Fig.5. Layout of the Experiment 6

Note: The boost-voltage is a constant that can be added to your Simulink model to match the requirements specified in the V/f control method. The boost voltage compensates for resistance drops at low speeds and improves the startup process of the motor.

In Fig.5 the frequency has been increased to about 70Hz. The duty-cycles are sinusoidally generated and phase-shifted with 120^0. The currents in phase *a* and *b* are also sinusoidal in steady state.

The experiment can be used to determine the characteristics of the steady-state operation of an induction motor, and the transient characteristics for different acceleration and deceleration slopes.