MATLAB MODULE 3

Simulink

Objectives:

- Building simple Simulink simulations.

- Running Simulink simulation to predict a system's behaviour.

The MATLAB Control System Toolbox offers functions for finding the transfer functions from a given block diagram. However, as we shall shortly see, the simulation environment provided by MATLAB's Simulink Toolkit obviates the need for block diagram reduction. The Simulink model mimics the block diagram of a feedback control system and is used to evaluate the response of controlled variable to any test input. It also provides the response of any internal variable of the control system (output variable of a subsystem block) without the need for block diagram reduction.

Let us reiterate the fact we have emphasized earlier: a good plant/process model is the backbone of any realistic control design. A Simulink model based on the structure and parameters of the system model is constructed. The responses of the actual system and its Simulink model are obtained using a set of test inputs. If the actual responses to the test inputs were significantly different from the Simulink responses, certain model parameters would have to be revised, and/or the model structure would have to be refined to better reflect the actual system behaviour. Once satisfactory model performance has been achieved, various control schemes can be designed and implemented.

In practice, it is best to test a control scheme off-line by evaluating the system performance in the "safety" of the Simulink environment. The key components of a control system include actuators, sensors, and the plant/process itself. A decision to include all aspects such as amplifier saturation, friction in the motor, backlash in gears, dynamics of all the devices, etc., may improve the model, but the complexity of the model may result in a more complicated controller design, which will ultimately increase the cost and sophistication of the system. The design is usually carried out using an approximated model; the evaluation of the design is done on the "true" model, which includes nonlinearities, and other aspects neglected in the approximate model. Simulink is an excellent tool for this evaluation.

SIMULINK (SIMUlation LINK) is an extension of MATLAB for modeling, simulating, and analyzing dynamic, linear/nonlinear, complex control systems. Graphical User Interface (GUI) and visual representation of simulation process by simulation block diagrams are two key features which make SIMULINK one of the most successful software packages, particularly

suitable for control system design and analysis.

Simulation block diagrams are nothing but the same block diagrams we are using to describe control system structures and signal flow graphs. SIMULINK offers a large variety of ready-to-use building blocks to build the mathematical models and system structures in terms of block diagrams. Block parameters should be supplied by the user. Once the system structure is defined, some additional simulation parameters must also be set to govern how the numerical computation will be carried out and how the output data will be displayed.

Because SIMULINK is graphical and interactive, we encourage you to jump right in and try it. To help you start using SIMULINK quickly, we describe here the simulation process through a demonstration example with MATLAB version 7.0, SIMULINK version 6.0.

To start SIMULINK, enter **simulink** command at the MATLAB prompt. Alternatively one can also click on SIMULINK icon [icon] shown in Fig. M3.1.

Fig. M3.1 *MATLAB Desktop main menu and SIMULINK icon*

A SIMULINK Library Browser (Fig. M3.2) appears which displays tree-structured view of the SIMULINK block libraries. It contains several nodes; each of these nodes represents a library of subsystem blocks that is used to construct simulation block diagrams. You can expand/collapse the tree by clicking on the [+]/[−] boxes beside each node and block in the block set pan.
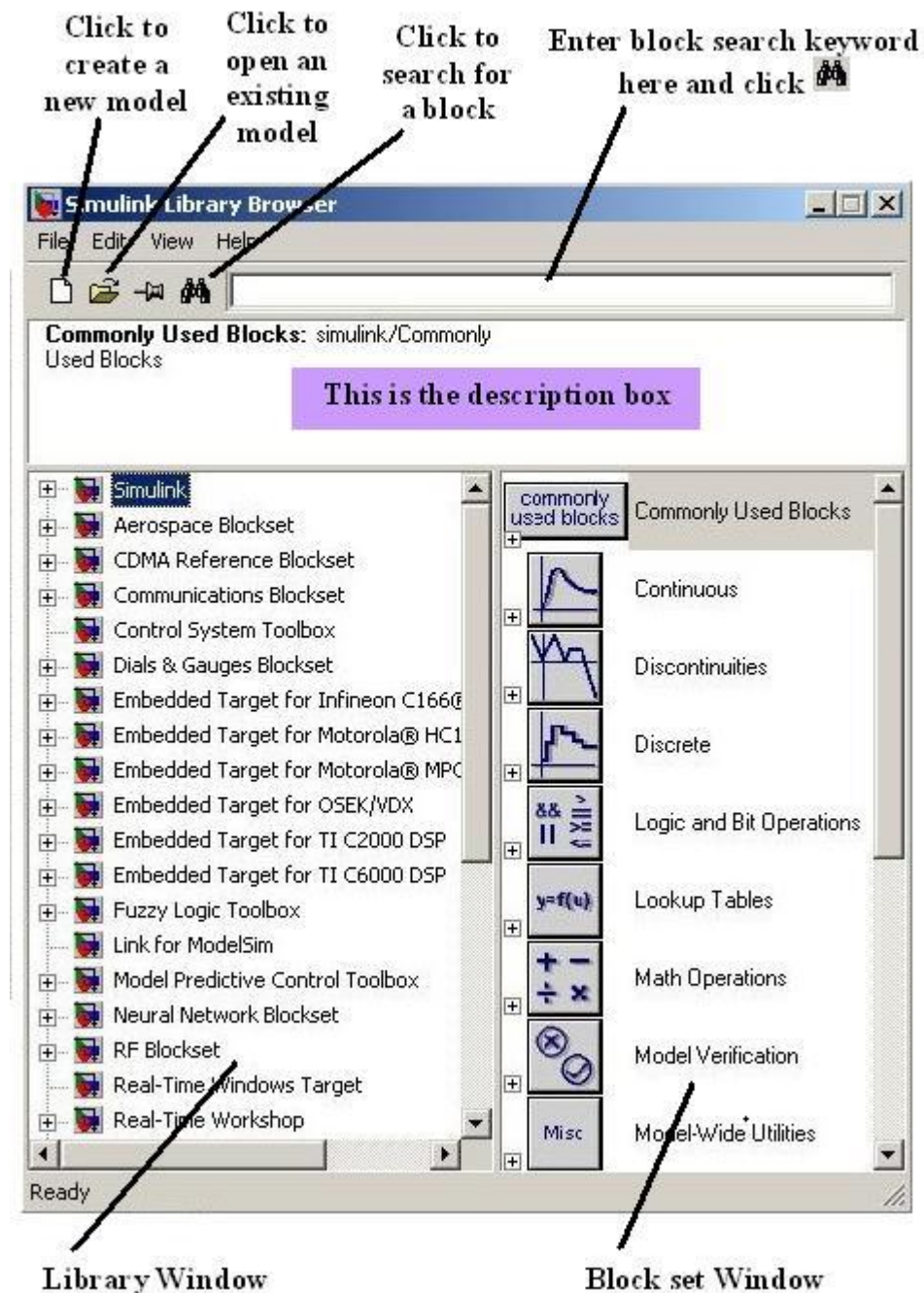
Fig. M3.2 *SIMULINK Library Browser*

Expand the node labeled **Simulink**. Subnodes of this node (**Commonly Used Blocks, Continuous, Discontinuities, Discrete, Logic and Bit Operations**, etc…) are

displayed. Now for example, expanding the **Sources** subnode displays a long list of Sources library blocks. Simply click on any block to learn about its functionality in the description box (see Fig. M3.3).
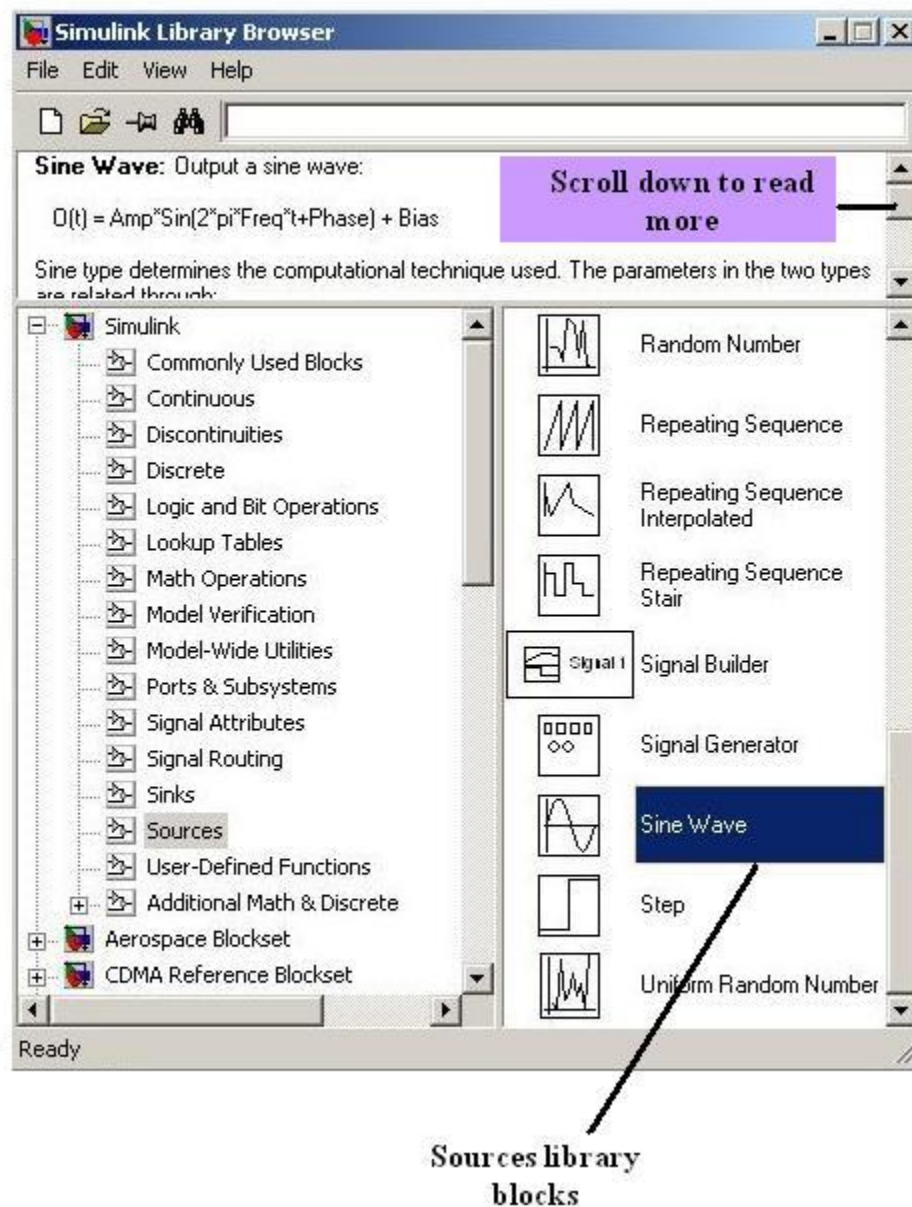

Fig. M3.3 *Blocks in Sources subnode*

You may now collapse the **Sources** subnode, and expand the **Sinks** subnode. A list

of **Sinks** library block appears (Fig.M3.4). Learn the purpose of various blocks in **Sinks** subnode by clicking on the blocks.
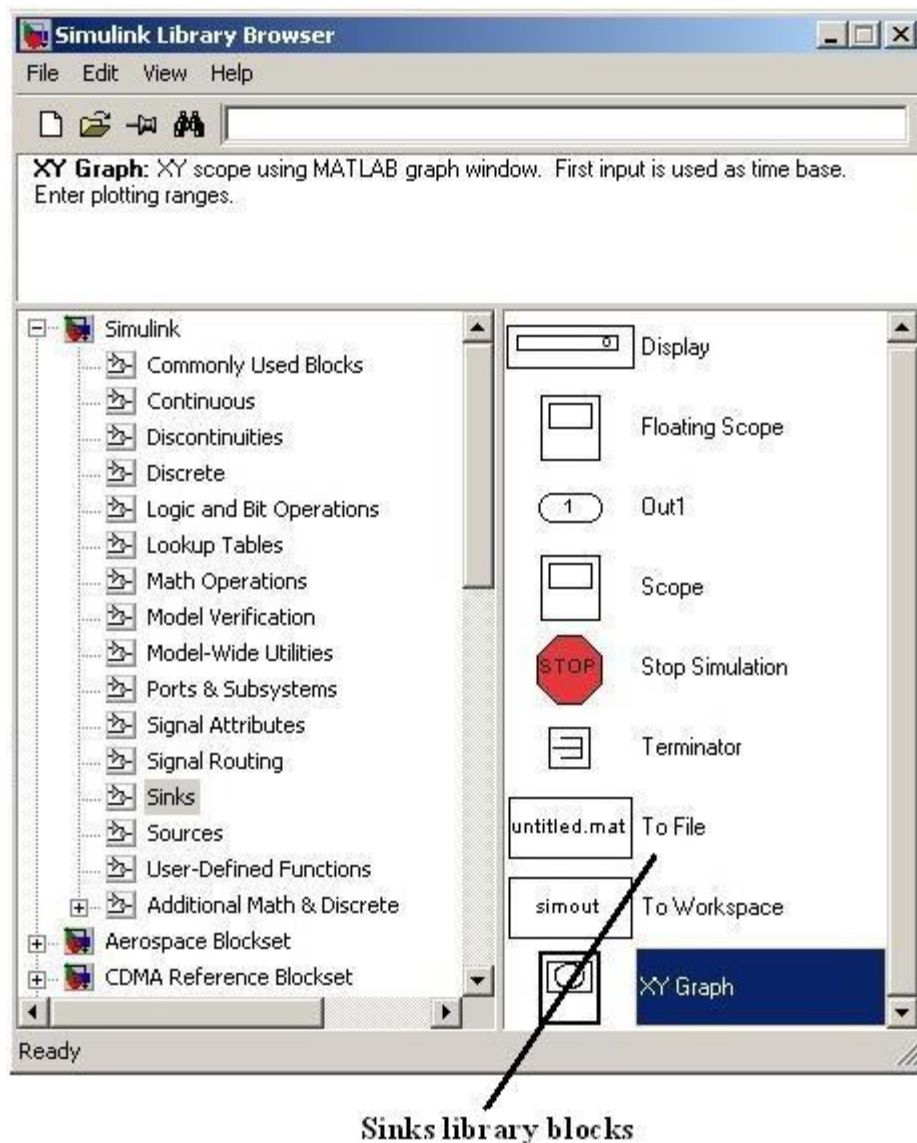

Fig. M3.4 *Blocks in Sinks subnode*

Exercise M3.1

Expand the **Continuous**, **Discontinuities**, **Discrete**, and **Math Operations** subnodes. Study the

purpose of various blocks in these subnodes in description box.

We have described some of the subsystem libraries available that contain the basic building blocks of simulation diagrams. The reader is encouraged to explore the other libraries as well. You can also customize and create your own blocks. For information on creating your own blocks, see the MATLAB documentation on "Writing S- Functions".

We are now ready to proceed to the next step, which is the construction of a simulation diagram. In the SIMULINK library browser, follow **File** →**New** →**Model** or hit **Ctrl+N** to open an 'untitled' workspace (Fig.M3.5) to build up an interconnection of SIMULINK blocks from the subsystem libraries.
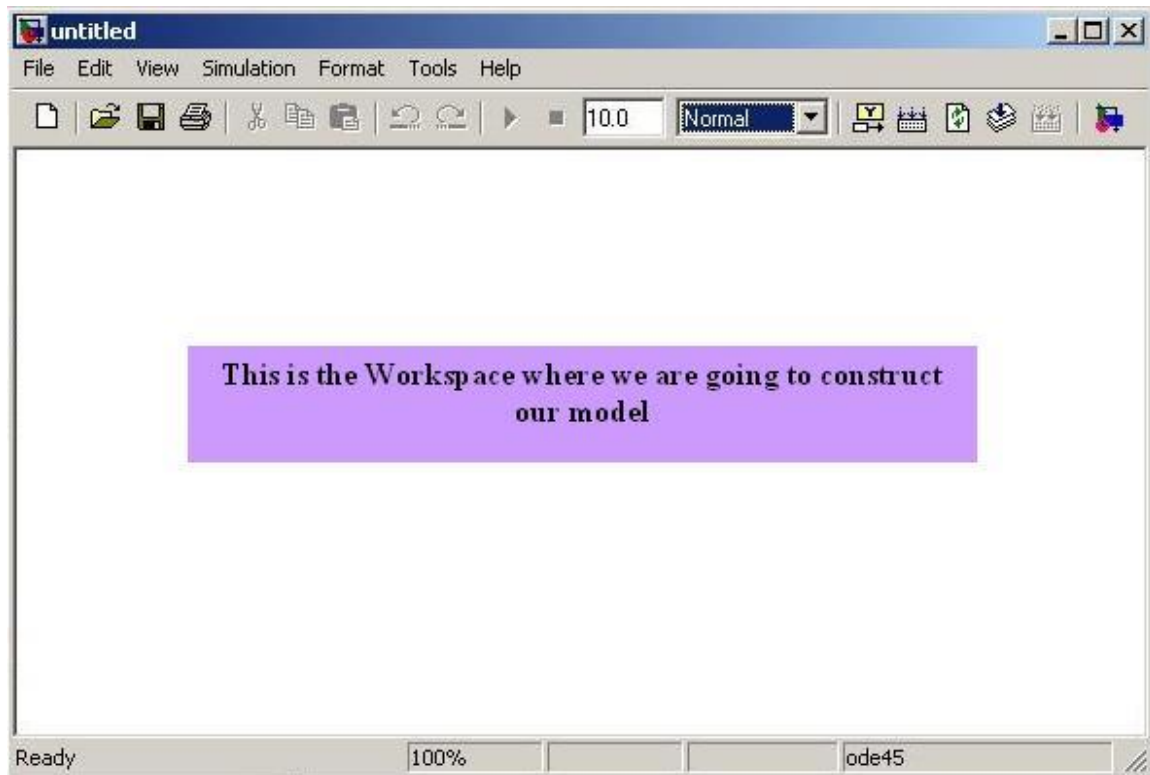

Fig. M3.5 *Untitled workspace*

Let us take a simple example. The block diagram of a dc motor (armature-controlled) system is shown in Fig. M3.6
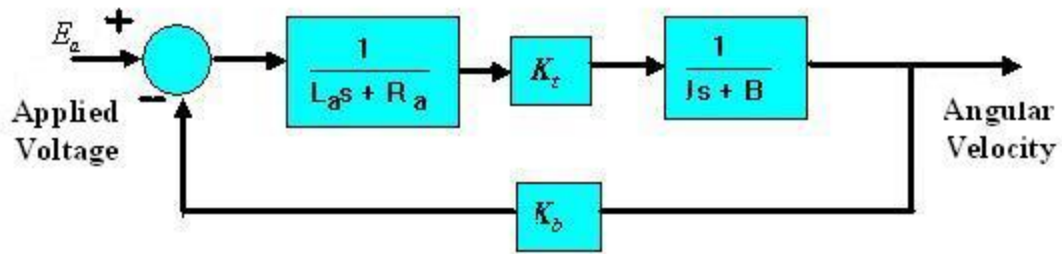
Fig. M3.6 *Block diagram of a dc motor (armature-controlled) system*

where

R_a is the resistance of the motor armature (ohms) = 1.75

L_a is the inductance of the motor armature (H) = 2.83 $\times 10^{-4}$

K_t is the torque constant (Nm/A) = 0.0924

K_b is the back emf constant (V sec/rad) = 0.0924

J is the inertia seen by the motor (includes inertia of the load) (kg-m^2) = 30×10^{-4}

B is the mechanical damping coefficient associated with rotation (Nm/(rad/sec))= 5.0 $\times 10^{-3}$

E_a is the applied voltage (volts) = 5 volts

We will implement the model shown in Fig. M3.6 in the **untitled** work space (Fig. M3.5).

Let us first identify the SIMULINK blocks required to implement the block diagram of Fig. M3.6. This is given in Fig. M3.7.
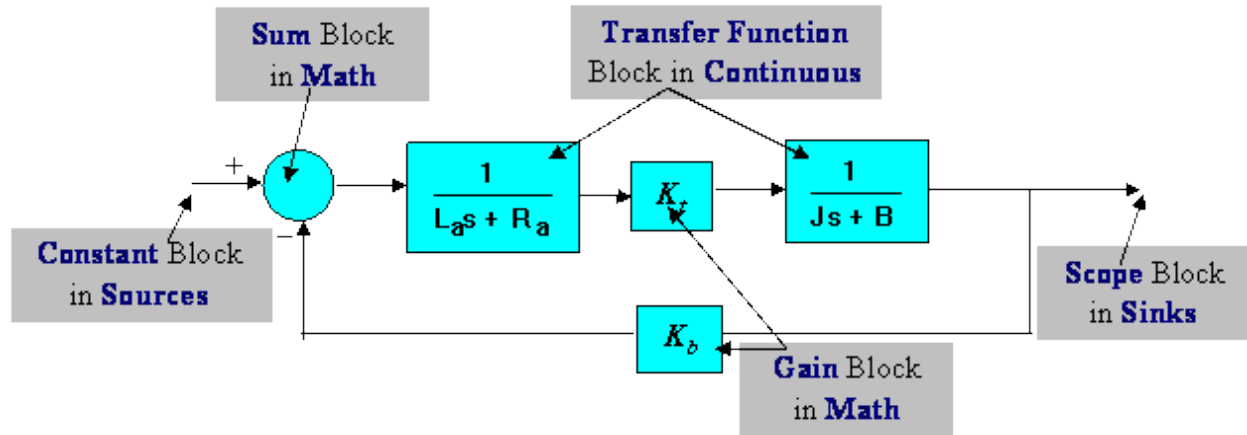
Fig. M3.7 *SIMULINK blocks required for implementation*

Identifying the block(s) required for simulation purpose is in fact the first step of the construction of simulation diagram in SIMULINK. The next step is to "***drag and drop*** " the required blocks from SIMULINK block libraries to **untitled** workspace. Let us put the very first block for applied voltage (E_a) to workspace.

Expand the **Sources** subnode, move the pointer and click the block labeled **Constant**, and while keeping the mouse button pressed down, drag the block and drop it inside the Simulation Window; then release the mouse button (Fig. M3.8).

Right clicking on the block will provide various options to users from which one can cut, copy, delete, format (submenu provides facilities for rotation of the block, flipping, changing the font of block name,...), etc...

Exercise M3.2

Drag and drop all the blocks we have identified (Fig. M3.7) from the Library Browser to the **untitled** Workspace and place them as shown in Fig. M3.9.
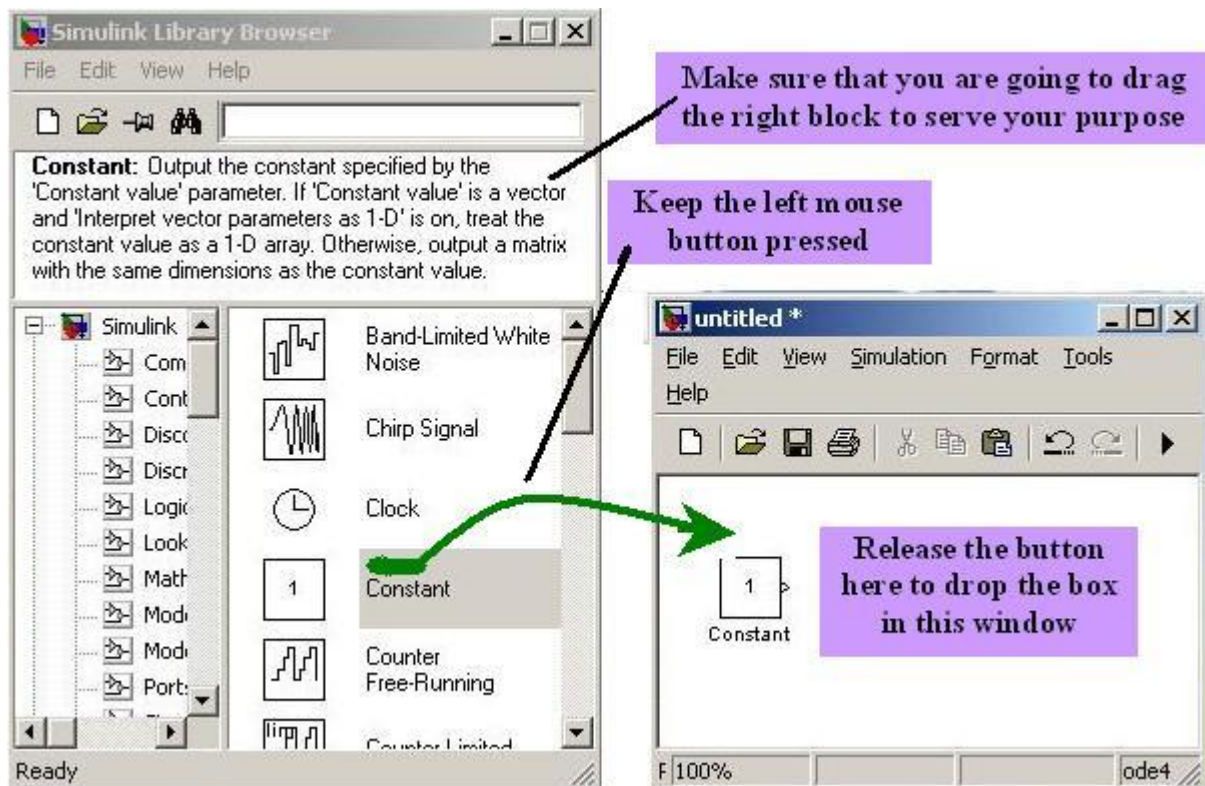

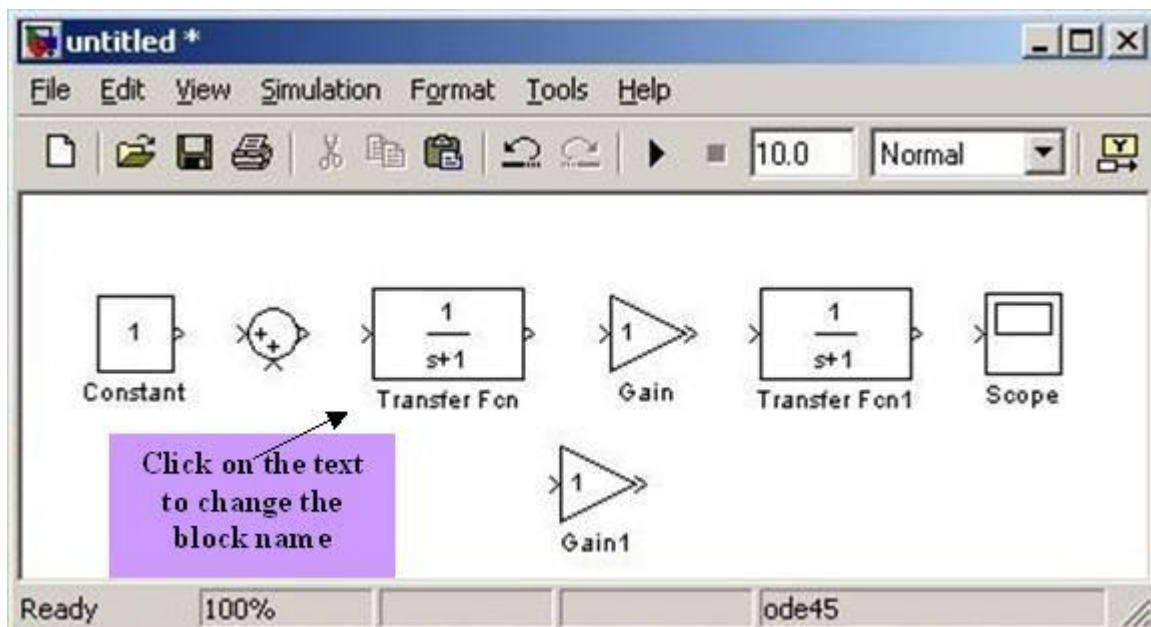**Fig. M3.8** *Drag and drop blocks to Workspace from Library Browser*

Fig. M3.9 *Unconnected blocks in Workspace*

It is visible that all the block parameters are in their default settings. For example, the default

transfer function of **Transfer Fcn** block is $\dfrac{1}{s+1}$ and default signs of**Sum** block are + +. We need to configure these block parameters to meet our modeling requirements. It is straightforward. Double click the block to set up its parameters. For example, double clicking the **Transfer Fcn** block opens the window titled **Block Parameters: Transfer Fcn,** shown in Fig. M3.10.
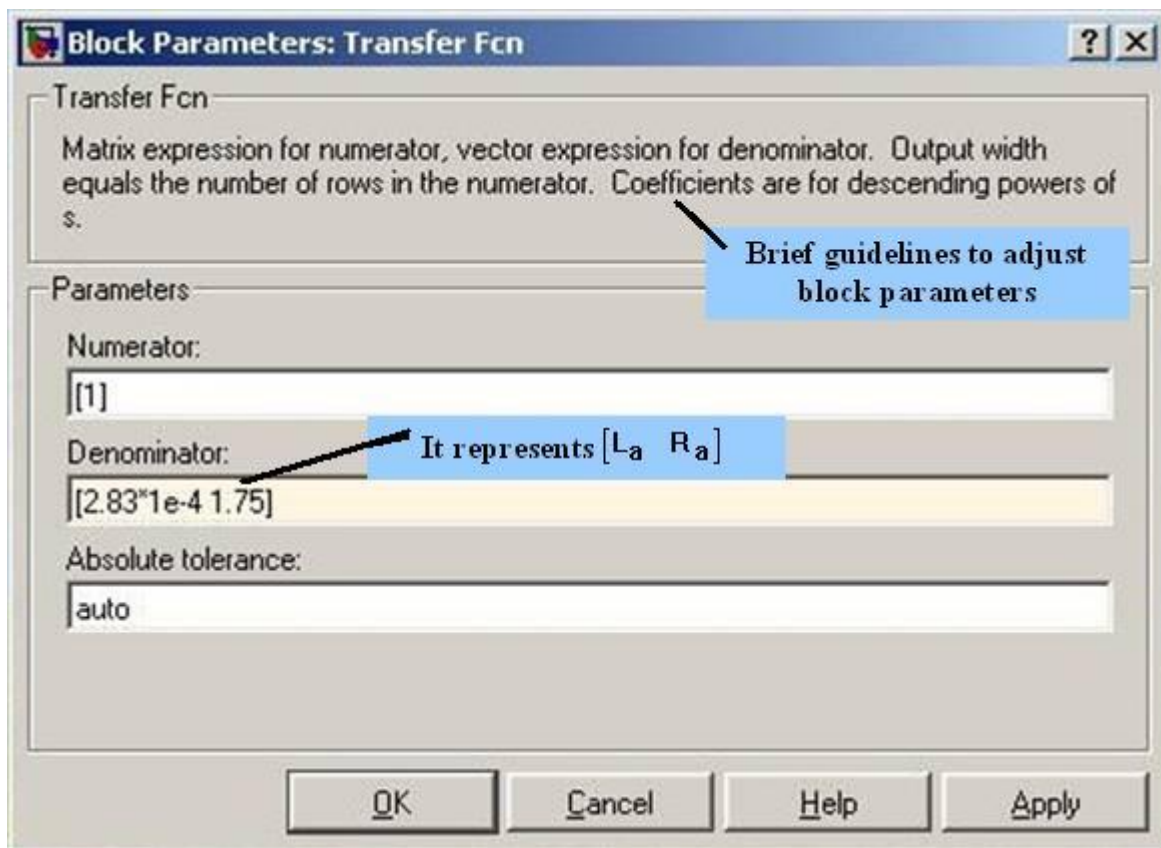

Fig. M3.10 *Transfer function block parameters window*

For armature circuit transfer function, no need to change the numerator parameter. For

denominator parameters, enter $\begin{bmatrix} 2.83*1e\text{-}4 & 1.75 \end{bmatrix}$ for $\begin{bmatrix} L_a & R_a \end{bmatrix}$, which will be interpreted   by

SIMULINK as $L_a s + R_a$.

To enhance the interpretability of simulation diagram, we can also change the block identification name. Simply click on the text **Transfer Fcn** to activate the small window around

the text to change the block name. For our simulation block diagram, the suitable text for **Transfer Fcn** block may be **Armature circuit.**

Before we move to the last step of interconnecting the blocks as per the desired structure, just finish Exercise M3.3. Note that the **Decimation** parameter value by default is 1. Increasing this value reduces the number of data samples taken over the simulation time. We have used the default value of 1.

Exercise M3.3

Modify all the block parameters as per system parameters given for Fig. M3.7, and give appropriate names to the blocks.

Lines are drawn to interconnect these blocks as per the desired structure. A line can connect output port of one block to the input port of another block. A line can also connect the output port of one block with input ports of many blocks by using branch lines. We suggest readers to perform the following line/block operations on blocks dragged in workspace to get hands on practice.

To connect the output port of one block to the input port of another block:

1. Position the pointer on the first block's output port; note that the cursor shape changes to cross hair.

2. Press and hold down the left mouse button.

3. Drag the pointer to second block's input port.

4. Position the pointer on or near the port; the pointer changes to a double cross hair.

5. Release the mouse button. SIMULINK replaces the port symbol by a connecting line with an arrow showing the direction of signal flow.

Another simple methodology doesn't require dragging the line. **Block1** output port is required to be connected to **Block2** input port.

1. select Block1 by clicking anywhere inside the block.

2. Hold down the Ctrl key.

3. Click on block2; both the blocks will be connected.

To connect the output port of one block with the input ports of several blocks, we can use *branch lines*. Both the existing line and the branch line carry the same signal. To add a branch line, do the following:

1. Position the pointer on the line where you want the branch line to start.

2. While holding down the Ctrl key, left click on the line segment; note that the cursor shape changes to cross hair.

3. Release the control key, while pressing down the left mouse button; drag the pointer to the input port of the target block.

4. Release the mouse button; target block will be connected to the line segment.

Some of the important line-segment and block operations are as follows:

1. To move a line segment, position the pointer on the segment you want to move. Press and hold down the left mouse button. Drag the pointer to the desired location and release. Note that this operation is valid with line segments only, not with the dedicated connecting lines between two blocks.

2. To disconnect a block from its connecting lines, hold down the shift key, then drag the block to a new location. Incoming and outgoing lines will be converted to red colored dotted lines. One can insert a different block between these lines.

Exercise M3.4

Connect all the blocks appropriately as per the block diagram given in Fig. M3.7. Make use of the block interconnection points discussed above.

Now let us give a name to the **untitled** workspace. Hit **Ctrl + S** to save the developed simulation diagram to the disk with an appropriate name. The file will be saved with the extension **.mdl** , an abbreviation for the 'model'.

We save the file using the name **armature_dcmotor.mdl;** the complete simulation diagram is shown in   Fig. M3.11.

Finally, we need to set the parameters for the simulation run. Press **Ctrl + E** to open the simulation parameter configuration window. Left panel of the window (Fig. M3.12) displays a tree structured view of parameter submenu.   In the **Solver** submenu, enter the start and stop time of the simulation (Fig. M3.13).
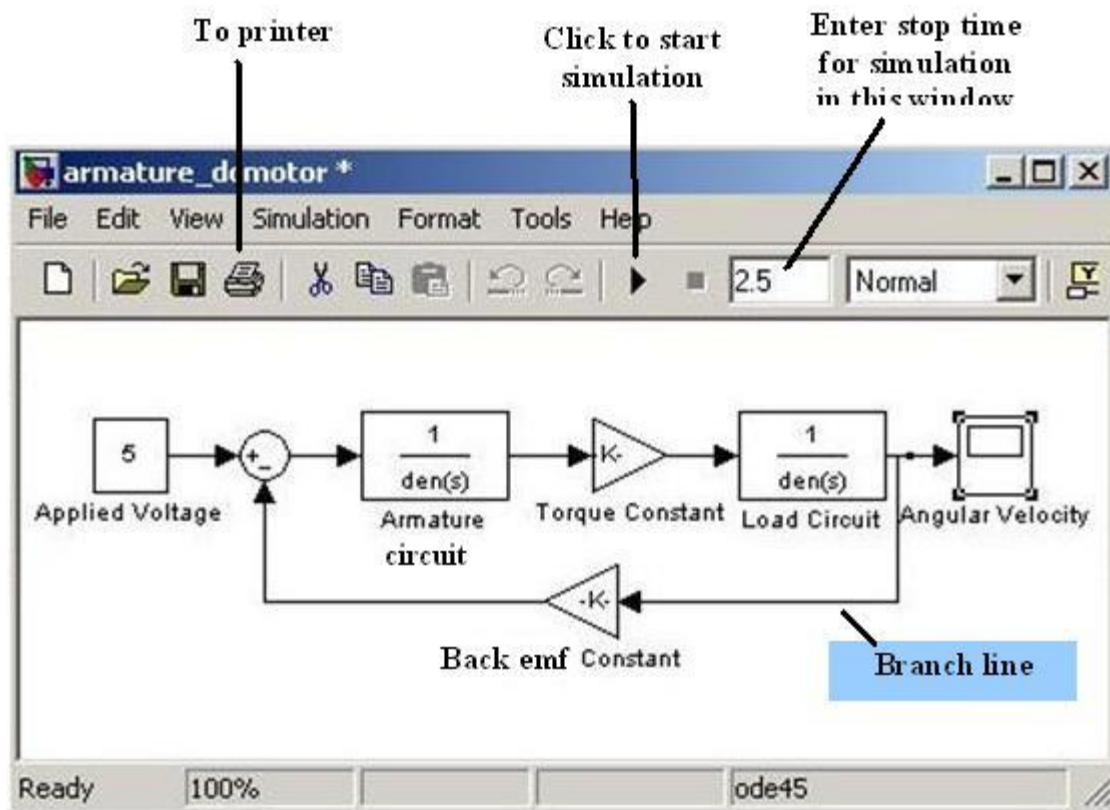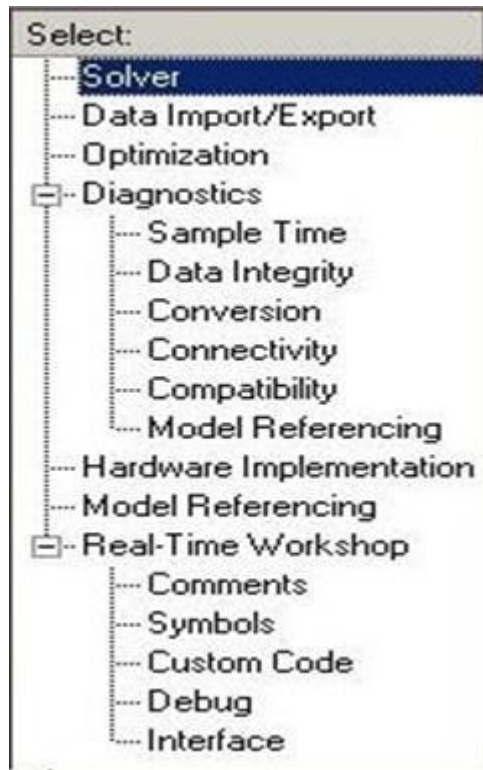
Fig. M3.11 *Final simulation diagram* (download)

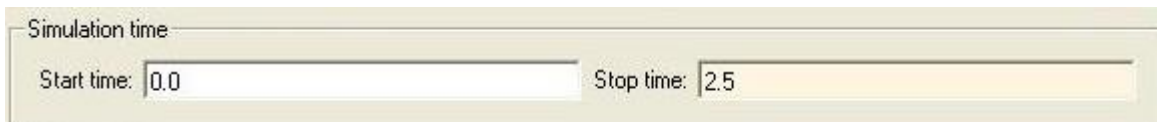**Fig. M3.12** *Parameter configuration submenu*

Fig. M3.13 *Enter simulation time*

Now we are ready to simulate our block diagram of armature-controlled dc motor. Press ▶ icon to start the simulation. Note that the icon changes to ❚❚ ; pressing this icon, one can stop the simulation before stop time. After simulation is done, double click the **Scope** block to display the angular velocity variation with time. Click the autoscale icon 🔍 in the display window to scale the axes as per the variable ranges. Autoscaled scope display is shown in Fig. M3.14. With zoom facility, try zooming the portion of graph between 0.5 to 1 sec, and 20 to 25 unit angular velocity to identify the numerical value of angular velocity at 0.8 seconds.
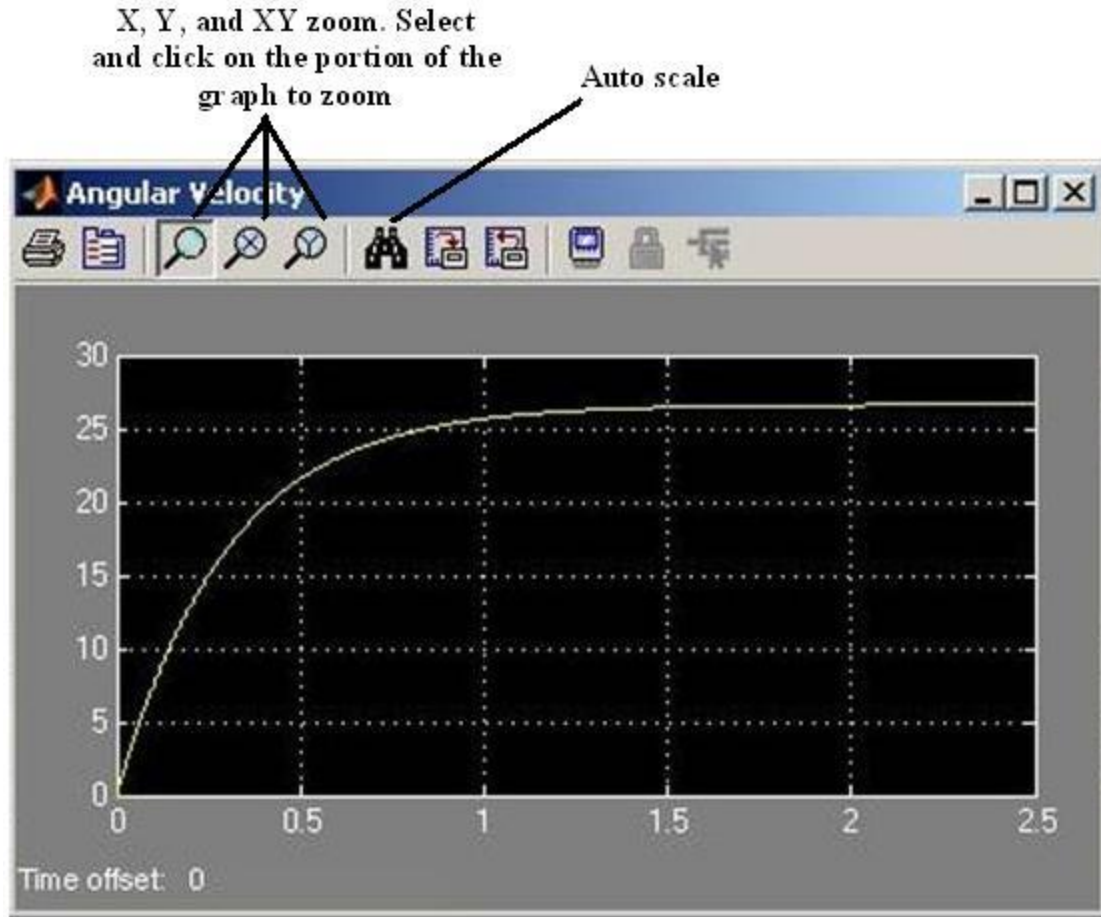
Fig. M3.14 *Scope display of angular velocity*

Set y-axis limits by right-clicking the axis and choosing **Axes Properties**. In **Y-min,** enter the minimum value for the y-axis. In **Y-max**, enter the maximum value for the y-axis. In **Title,** enter the title of the plot. See Fig. M3.15.
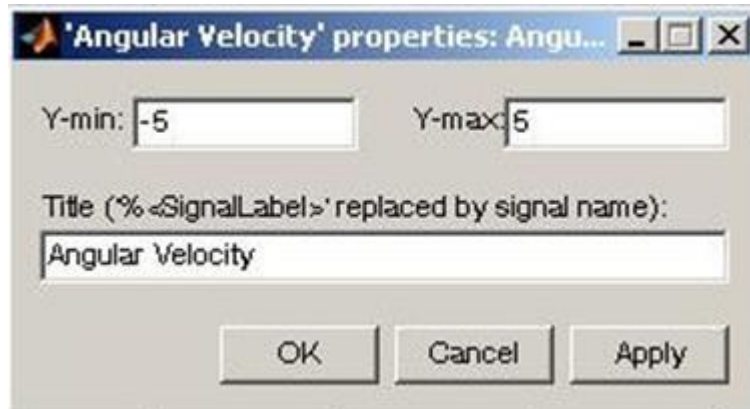
Fig. M3.15 *Scope axis properties editor*

Click the icon shown on the icon bar of Fig. M3.14 to open scope parameter editor (Fig. M3.16). **General** parameters include **Number of axes, Time range**, **Tick labels,** and **Sampling**.

Click on the **Data history** button. If you want input-output data from this scope to be available to MATLAB workspace for further analysis, check the button **Save data to workspace**. In the box **Variable name,** enter the variable name for saving the data. By default it will save the data with variable name **ScopeData**. With the pop-down menu **Format**, select the format in which you want to save the data.

Three specific formats for saving the data are as follows:

1. **Structure with time**: Data will be saved in structured format with time steps. Type the following commands in your MATLAB prompt and observe the outputs.
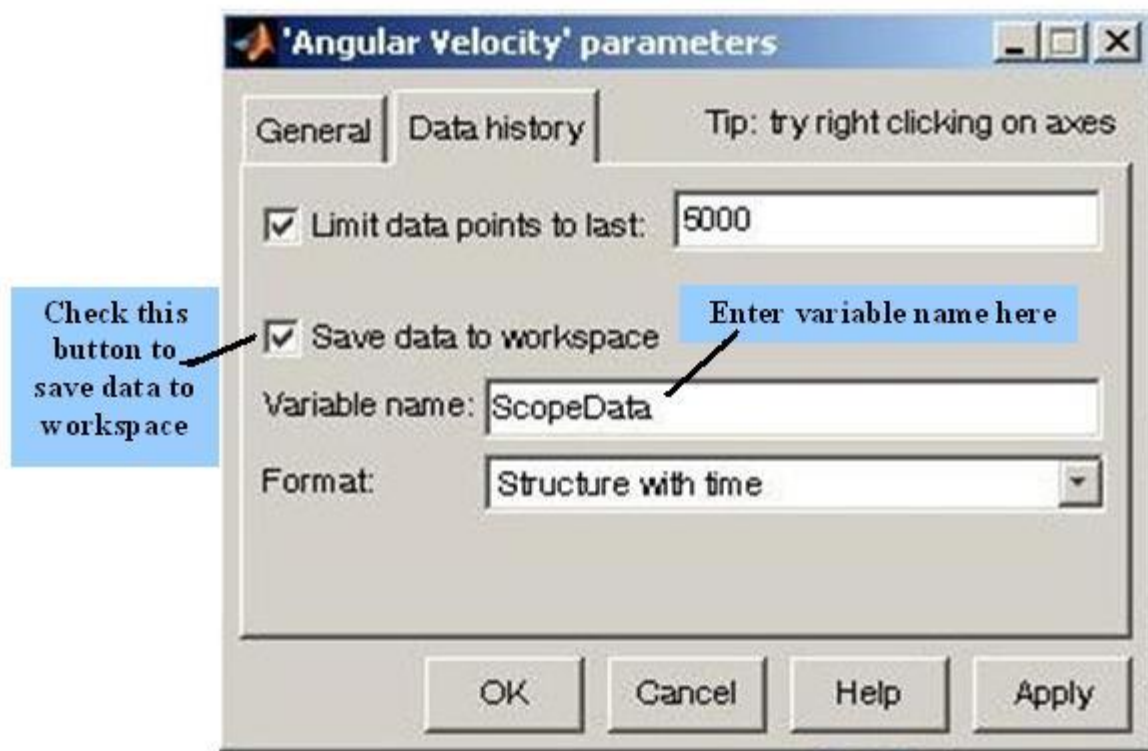
Fig. M3.16 *Scope parameter setting window*

>> ScopeData

ScopeData =

time: [4663x1 double]

signals: [1x1 struct]

blockName: 'armature_dcmotor/Angular Velocity'

Structures are used in MATLAB to store mixed mode data types, and individual fields of the structure can be accessed by '***dot*** $(\bullet)$' operator. To see the information stored in the field **signals,** type:

>> ScopeData.signals

ans =

values: [4663x1 double]

dimensions: 1

label: ''

title: ''

plotStyle: 0

It indicates that the field **signals** contains subfield **values**, which is of 4663 x 1 size vector containing the values of angular velocity. Try accessing the field **time** of**ScopeData**.

Exercise M3.5

Plot the angular velocity against time using the **plot** command. Give suitable title to the plot and labels to x and y axes.

Hint: You need to plot **ScopeData.signals.values** against **ScopeData.time**.

---

2. Structure : This is the same as **Structure with time**; the only difference is that the time steps will not be saved.

Exercise M3.6

Run the simulation with scope data to be saved as **Structure** format. Verify that the time field of ScopeData structure is actually an empty matrix.

---

3. **Array** : Array format is simply a two   column matrix with number of data points being equal to number of rows. The maximum number of data points limits to the number entered in the box **Limit data points to last**. In Fig. M3.16, the limit is 5000 data points.

Exercise M3.7

Run the simulation with scope data to be saved as Array format. Repeat Exercise M3.5

with saved data matrix.

We have used an example to show how to build the simulink diagram, how to enter data and carry out a simulation in the SIMULINK environment. The reader will agree that this is a very simple process.

Solving the following exercises will make the readers more confident in solving the control system design and analysis problems through SIMULINK.

Exercise M3.8

This problem requires some modification in the above considered armature-controlled d.c. motor SIMULINK example.

1. The angular position is obtained by integrating the angular velocity. Implement this in the model and display angular position in a different scope.

2. Remove the constant applied voltage block. Obtain the Step, Ramp, and Sinusoidal responses of the system.

3. Simulate a closed-loop position control system assuming a proportional controller of gain K_P.

Hint: Add a reference input signal for angular position and add a summing block which calculates the error between the angular position reference and the measured angular position. Multiply this error by the gain K_P and let this signal be the applied voltage E_a.

Exercise M3.9

Consider a dynamic system consisting of a single output y, and two inputs r and w :

$$Y(s) = G(s)R(s) + N(s)W(s)$$

where

$$G(s) = \frac{2e^{-5s}}{50s^2 + 15s + 1} \,(process)$$

$$N(s) = \frac{0.3e^{-5s}}{15s + 1} \,(disturbance)$$

Model the system in SIMULINK.

For r and w both step signals, obtain the system output y. Display the output on scope. Also, using a SIMULINK block, etch the output to the workspace.

Hint: To implement deadtime, use the block **Transport delay** from **Continuous,** and identify suitable block from **Sinks** library to fetch the variable directly to the workspace.

Exercise M3.10

Assign 0.5, $\sqrt{2}$, and 60^0 to y_0, ω_n, and θ, respectively.

Simulate
$$y(t) = \frac{y_0}{\sqrt{1-\zeta^2}} e^{-\zeta\omega_n t} \sin\left(\omega_n \sqrt{1-\zeta^2}\, t + \theta\right)$$
for 10 seconds.

Display plots for $\zeta = 1/\sqrt{2}, \zeta = 1,$ and $\zeta = \sqrt{2}$ on scope, and also save $y(t)$ to the MATLAB file **Yt.mat**.

Hint: Use **Math Function**, **Add**, **Divide**, and **Trigonometric Function** from **Math Operations** block libraries.

Exercise M3.11

This problem is to study the effects of Proportional (P), Proportional + Integral (PI), and Proportional + Derivative (PD) control schemes on the temperature control system. A temperature control system has the block diagram given in Fig.M3.17. The input signal is a voltage and represents the desired temperature θ_r. Simulate the control system using SIMULINK and find the steady-state error of the system when θ_r is a unit-step function and (i) $D(s) = 1$ (ii) $D(s) = 1 + \frac{0.1}{s}$ and (iii) $D(s) = 1 + 0.3s$.
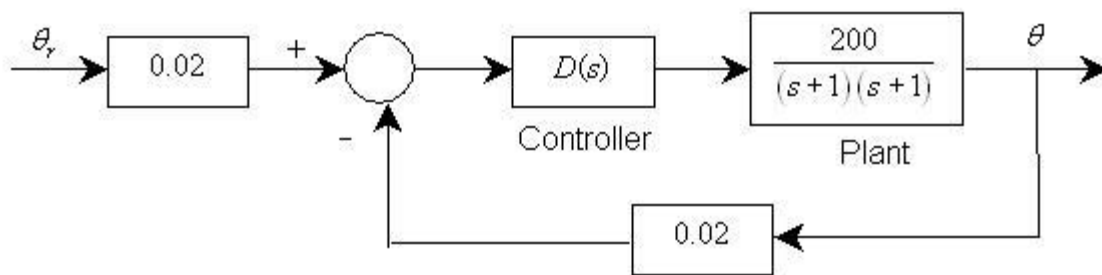
Fig. M3.17

Hint: Use **PID Controller** block from **Simulink Extras** $\rightarrow$ **Additional Linear** to implement the PID controller.

Exercise M3.12

The block diagram in Fig. M3.18 shows the effect of a simple on-off controller on second-order process with deadtime.
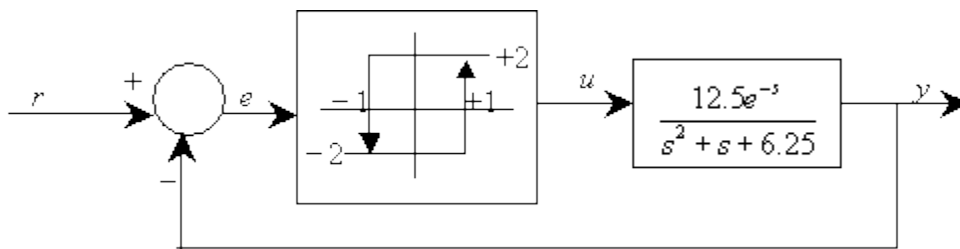

Fig. M3.18

Implement and test the model in SIMULINK for   step inputs of 5.0 and 10.0. Display the control signal $u(t)$ and output the $y(t)$ on separate scopes and also fetch both the signals with time information to MATLAB workspace. Using MATLAB **plot** function, plot the control signal and the output; both against time on single graph.

Hint: To implement on-off controller, use **Relay** from **Discontinuous** block library.

Exercise M3.13

Simulate the Van der Pol oscillator, described by the following nonlinear differential equation: $\dfrac{d^2x}{dt^2} - k\left(1 - x^2\right)\dfrac{dx}{dt} + x = w(t)$,

where $w(t)$ is the disturbance (forcing function) given by $w(t) = 10 * rect(2t)$. Assign $k = 2$.

Hint: Rewrite the second-order Van der Pol differential equation as a system of coupled first-order differential equations. Let,

$$x_1 = x$$

$$\frac{dx}{dt} = \frac{dx_1}{dt} = x_2, x_1(0) = x_{10}$$

$$\frac{d^2x}{dt^2} = \frac{dx_2}{dt} = -x_1 + kx_2 - kx_1^2 x_2 + w(t), x_2(0) = x_{20}$$

The SIMULINK block diagram is to be built using **Signal Generator**, **Gain**, **Integrator**, **Add**, and **Scope**. The output of **Add** block is $\dot{x}_2$. Integrating it once will lead you to x_2 and second integration will lead you to x_1. Double click the **Integrator** block to add the initial conditions $\begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$.