ELEN90055 Control Systems
Workshop 0

Introduction to MATLAB and SIMULINK

Prepared by Alejandro Maass and Michael Cantoni; Email: cantoni@unimelb.edu.au

Created: February 28, 2019; Last compiled: March 9, 2019

1 Introduction

The aim in Workshop 0 is to work through a very basic tutorial on MATLAB and SIMULINK. This introduction is tailored to control systems in particular. It includes use of the key MATLAB functions and SIMULINK blocks required for the forthcoming workshops. Specifically, the focus is to build and simulate models; the context is not so important. The interpretation of models and simulations will be the focus of subsequent workshops.

More general and sophisticated tutorials can be found at:

`https://au.mathworks.com/support/learn-with-matlab-tutorials.html`

2 A simple case study

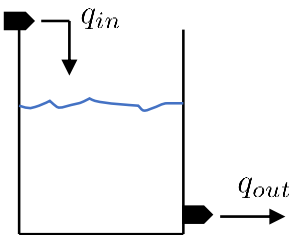

Figure 1: Water tank with imposable inflow q_{in} and outflow q_o.

Consider the water tank in Figure 1, where q_{in} is the inflow and q_{out} is the outflow. The volume V of water in the tank behaves according to the following model:

$$\frac{dv(t)}{dt} = q_{in}(t) - q_{out}(t). \tag{1}$$

For the purpose of this workshop, it is assumed that the inflow and outflow are constant. Therefore, (1) can be written as

$$v(t) = v(0) + \int_0^t (q_{in} - q_{out})dt = V(0) + (q_{in} - q_{out})t, \tag{2}$$

assuming zero as the initial time.

The aim of this workshop is to implement (2) in both MATLAB and SIMULINK. In particular, the required MATLAB functions and SIMULINK blocks are presented and explained.

3 Matlab implementation of the model

3.1 Matlab scripts

MATLAB is an interpreter. The user can enter commands one at a time directly. Each command, which may define and assign a value to a data object, for example, perhaps by function call, is executed as it is entered. There are many built-in data object types (scalars, vectors, systems, etc.) and toolbox extensions of these. A collection of commands can be grouped into a *script*. The script is a file with a `.m` extension. A script is run by entering its name at the command line, once in the same working directory, or by pushing the play button in the in-built script editor. Each command in the script is executed in the line order specified. A script can be created in the editor tab that is part of the MATLAB environment, or using any other text editor. Alternatively, it can be created in the command window by typing `edit scriptfilename`.

- Create a MATLAB script, in which you will record your working for Workshop 0.

3.2 Relevant commands

3.2.1 Tank as a transfer function

To be able to implement (2) in MATLAB, we need to create an integrator. The input q of the integrator is the net flow, being the difference between the inflow and the outflow, and the output v is the water volume. Recall that an integrator has a frequency domain transfer function $V(s)/Q(s) = 1/s$, where $V(s)$ and $Q(s)$ denote the Laplace transform of the signals v and q, respectively. We use the `tf` command to create a system object with specified transfer function.

- Create the transfer function $1/s$ with the `tf` command.

There are several ways to create this transfer function. Some examples are provided below.

```
% example 1
sys = tf(1,[1 0]);
% example 2
s = tf('s');
sys = 1/s;
% example 3
s = tf([1 0],1);
sys = 1/s;
```

Please type `help tf` in the command window for a better understanding of this command. Note, ending a MATLAB command with `;` stops the result from being displayed. To see result of that line when the script is run, remove the `;`. The character `%` marks the start of a comment.

3.2.2 Computing a step response

Given the system object `sys`, we can use the command `step` to plot the water volume over time, given a constant inflow and outflow. The command `step` computes the step response of the integrator and generates a plot of this, assuming zero initial conditions in line with the transfer function specification of the system; i.e., $v(0) = 0$ in (2). A non-zero

initial condition can be added manually as explained further below. Let us assume $q_{in} = 5$ and $q_{out} = 1$, i.e. the tank is filling up with water. Note that the step magnitude is $q = q_{in} - q_{out} = 4$.

- Compute the step response, with amplitude 4, of the transfer function $1/s$.

To do this, we first need to set the amplitude of the step to 4. This is done by entering

```
opt = stepDataOptions('StepAmplitude',4);
```

Then, type

```
step(sys,opt);
```

Note that `sys` is the system object with integrator transfer function created previously. The result of this is a plot that is displayed when the script is run even with the `;` in place. This plot is shown in Figure 2. Sometimes it is useful to access to record the data

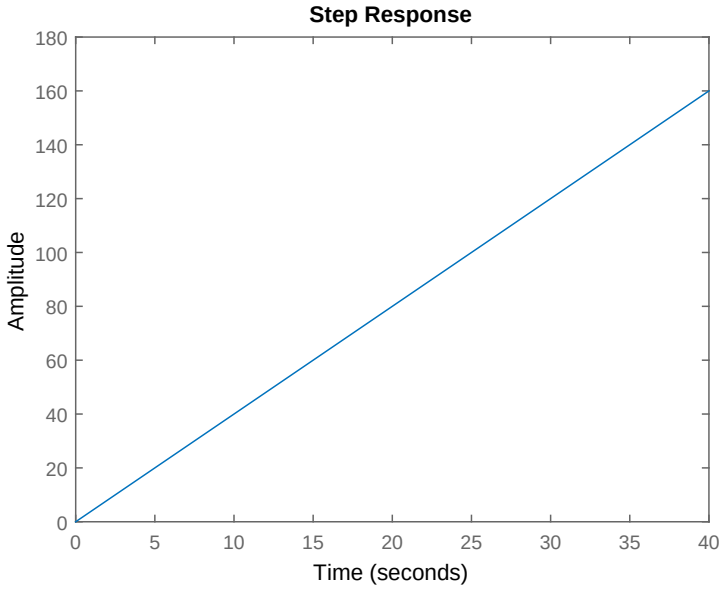

Figure 2: Step response of the integrator $1/s$.

generated by the `step` (and other) command(s) in new data objects. This would allows us to manually add the initial condition $v(0)$, for example. To do this, use

```
[y,t]=step(sys,opt);
```

We can plot the contents of the data vector y versus the time vector t using the `plot` command, which includes options for specifying labels, title, etc. Specifically, to add an initial condition of $v(0) = 5$, type

```
plot(t,y+5);
```

Note that Figure 2 represents the tank getting filled with water as the inflow is greater than the outflow. Furthermore, it is consistent with the ramp function described algebraically in (2).

- Compute the tank step response for different constant values of q_{in}, q_{out} and $v(0)$.

3.2.3 Building the step response using a 'for loop'

We now show how to simulate in a MATLAB script the step response of the integrator using `for` loops. To do this, we need to discretise (1), and we pick the following simple discretisation for the derivative

$$\frac{dv(t)}{dt} \approx \frac{v(k+1) - v(k)}{T} = q_{in} - q_{out}, \tag{3}$$

where T is the discrete time step. For the sake of argument, we take $T = 1$ below. To implement (3) in our MATLAB script use the following code:

```
qin = 5; % Inflow
qout = 1; % Outflow
v(1) = 0; % Initial condition (n.b. vectors are indexed starting from 1)
for i=1:1:40 % This means i ranges from 1 to 40 in steps of 1
  v(i+1) = v(i) + qin - qout; % Discretization (3) of (1) with T=1
end
stem([0:1:40],v); % use of plot instead would join the dots
```

The above code generates the plot in Figure 3.

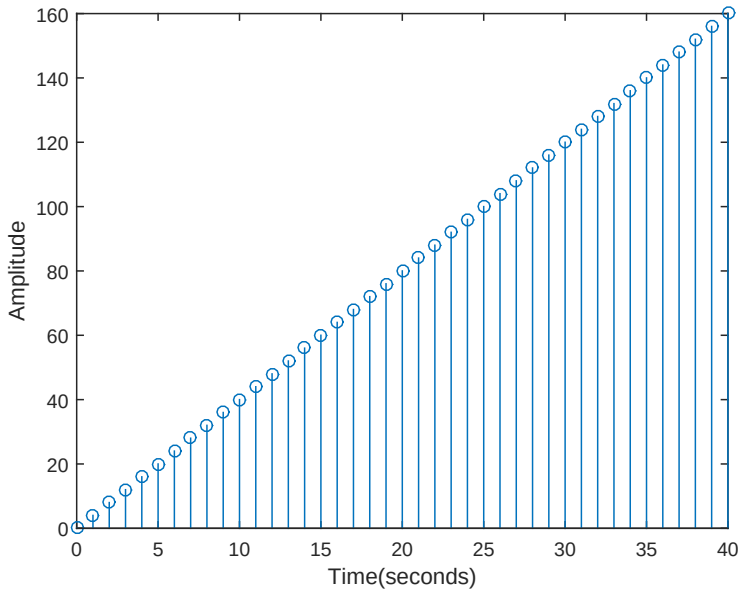

Figure 3: Step response of the discretised integrator.

Note that (3) is a simple (Euler) discretization of the continuous-time equation (1). There are many other more sophisticated methods available as well, which can also be easily implemented with `for` loops.

- Run the above code for different values of inflow, outflow and initial condition.

4 Simulink implementation

Simulink is a graphical extension to Matlab for the modelling and simulation of dynamical systems. In Simulink, system models are captured schematically as block diagrams. Many libraries of elemental building blocks are available, such as transfer function blocks, summing junctions, etc., as well as virtual input and output blocks such as function generators and scopes. These virtual devices allow you to create simulations of the models you will build by configuring elemental building blocks selected into the model from the available libraries, and graphically linking the inputs and outputs of these blocks. Linking is achieved by drawing lines with the mouse between ports. Simulink is integrated with Matlab. Data from the Matlab workspace is available in the Simulink model. This is often used to set configurable parameters in blocks. Data generated in Simulink during

simulation can also be exported for use at the MATLAB command line.

4.1 Starting Simulink

Simulink is started by typing `simulink` in the command window in MATLAB. Alternatively, you can click `New->Simulink Model` at the top left of MATLAB's main window. When it starts, SIMULINK brings up a new blank model, depending on your MATLAB version. This is the window into which a new model can be drawn. From this window, you can click the icon shown in Figure 4 to open the Library Browser, which is the window you use to obtain different modelling objects.

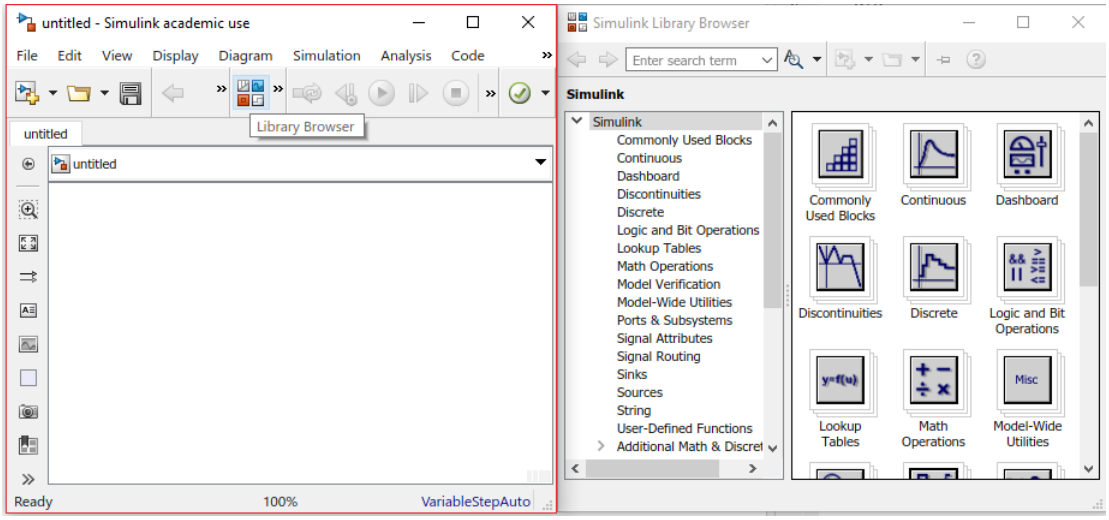

Figure 4: New blank model and corresponding Library Browser.

4.2 Basic elements

There are two major types of item in a SIMULINK model: *blocks* and *lines*. Blocks are used to generate, modify, combine, output, and display signals. Lines are used to transfer signals from one block to another. There are several general classes of blocks, and we list the ones most relevant to our needs below:

- Sources: Used to generate signals
- Sinks: Used to output or display signals

- Continuous: Linear, continuous-time system elements (transfer functions, state-space models, etc.)

- Commonly used: Summing junctions, gains, etc.

4.3 Building the tank model

The tank model expressed mathematically in (2) can be constructed in SIMULINK as shown in Figure 5. We need two step blocks, one integrator block, and one scope block. The step

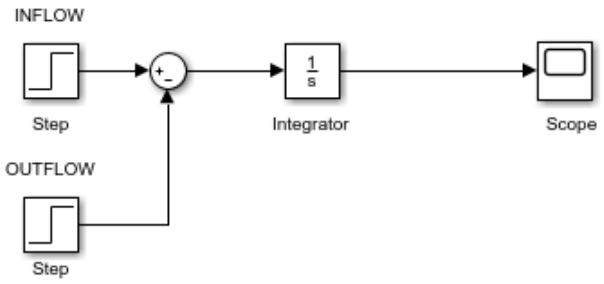

Figure 5: Tank model in SIMULINK.

block can be found in the Library Browser under the 'Sources' tab. The integrator can be found in the 'Continuous' tab, and the scope can be found in the 'Sinks' tab. By double-clicking a block, you can access to its different options. These block parameters can be set to fixed values specified by you in the available fields, or they can be linked to data objects in the MATLAB workspace. To start the simulation you should click the 'Play' (run) button at the top of the model window. To set the initial condition of the integrator, you can double-click the integrator block and modify it accordingly.

- Create the block diagram in Figure 5, and simulate the response for different values of inflow, outflow and initial volume. Consider linking the source block parameters to appropriate variables in the MATLAB workspace set up by running the script developed above. The value of a parameter can be changed by changing the corresponding assignment command in the script and re-running the script, or just that command.

4.4 Closing the loop

In this section we feed the volume signal back to the inflow with a constant gain as shown in Figure 6. To do this we need to add a 'Gain' block and a 'summing junction' block, both

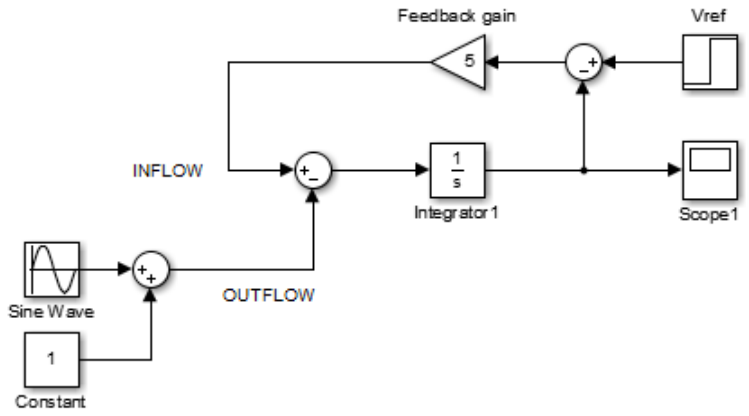

Figure 6: Tank model in Simulink under a constant gain feedback control of inflow to compensate for outflow disturbance.

found under the 'Commonly Used Blocks' tab. We have also replaced the constant outflow by a sinusoid with offset, and we included a reference volume level. The 'Sine Wave' can be found under the 'Sources' tab, and the 'Constant' can be found under 'Commonly Used Blocks' tab.

- Construct the block diagram in Figure 6 and simulate the response for different values of the feedback gain, integrator initial condition, sine wave frequency, offset, and reference volume level.

4.5 Exporting data from Simulink to Matlab

Sometimes, we would like to use the results of a Simulink simulation in the Matlab command window for further calculations and plotting. In fact, doing this is helpful for your reports as screenshots of a Simulink scope are hard to analyse. Exporting the signal data to Matlab will help you get improved plots which can be easily analysed. We can achieve this by using the 'To Workspace' block, found in the 'Sinks' tab (see Figure 7). You can double-click this block in order to edit the signal name and save format.

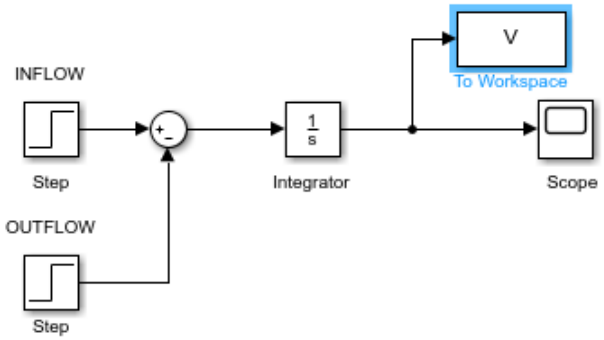

Figure 7: To Workspace block shown in the tank model.

- Include a 'To Workspace' block in your open-loop tank model to export the output of the integrator to MATLAB. Note that you need to run the simulation for this data to be saved.
- Plot the data, which is now in your workspace, using MATLAB and include appropriate labels to it. (This should look exactly as Figure 2, which was previously plotted using a MATLAB commands instead of SIMULINK models).

4.6 Saving a plot from a scope in Simulink

Sometimes it is useful to save the plot directly from the Scope block. To do this, double-click the scope after the simulation has stopped, and click `File->Save to Figure`. A MATLAB figure will open, and this can be saved in a large range of formats including `.eps, .jpg, .png`, etc. This can also be done by using the MATLAB command `print`, which you are encouraged to explore due to its multiple printing options (type `help print` in the command window).

5 Summary of main points

By the end of Workshop 0, you should have gained an improved understanding of MATLAB and SIMULINK, within the context of analysing a simple but insightful example. The tools learned in this workshop are key for the forthcoming workshops, in which more complicated systems are modelled, simulated and analysed.