1 Activity goal

The purpose of this laboratory activity consists of introducing and becoming familiar with the experimental setup used in the next laboratory activities. At the end of this activity, you should be able of implementing and configuring a Simulink model to perform a “real–time simulation” that envisages a direct interaction with a physical device.

2 Experimental setup

The experimental setup consists of:

1. Quanser SRV–02 servomotor. This is the physical device to be controlled (plant). It consists of a DC motor (1) in Fig. 1c) with a built–in planetary gearbox (2 in Fig. 1c) capable of driving a mechanical load attached to its output shaft (2 in Fig. 1b). Two mechanical loads will be used for the laboratory activities of this course: a disc inertia (i.e. inertial load – 1 in Fig. 1e) and a rigid beam connected to the output shaft through a flexible joint (i.e. resonant load – 1 in Fig. 1f). An incremental optical encoder is directly connected to the output shaft of the SRV–02 unit for measuring the position of the mechanical load (2 in Fig. 1d). A potentiometer (3 in Fig. 1d) is also connected to the shaft through a gear coupling (that includes an anti–backlash mechanism – 3 in Fig. 1b), and can be used as an alternative load position sensor. The DC motor is driven by a linear voltage driver based on a power operational amplifier (1 in Fig. 1d). The simplified electrical schematic of the voltage driver is reported in Fig. 2e. It is rather immediate to deduce that the voltage driver behaves – from “Motor Input” to ”Motor Current A” output – as a first order low–pass filter with DC gain

\[ k_{\text{drv}} = \frac{R_2}{R_1 + R_2} \left( 1 + \frac{R_3}{R_4} \right) \approx 0.6 \]  

and cut–off frequency

\[ f_c = \frac{1}{2\pi \left( \frac{R_1}{R_2} \right) C_1} \approx 1.2 \text{ kHz} \]  

A shunt resistor is connected in series to the motor to sense the armature current (i.e. the current flowing through the rotor windings).


The PCI–6221 board (installed on PCFA20 ÷ PCFA27 workstations) supports
Figure 1: Experimental setup details.
Figure 2: Experimental setup details (cont’d).
- 16 single-ended or 8 differential analog inputs channels with 16 bit ADC resolution, maximum sample rate (per channel) of 250 kS/s and configurable ADC input range (available options: ±0.2 V, ±1 V, ±5 V, ±10 V).
- 2 analog output channels with 16 bit DAC resolution, fixed ±10 V output range and maximum sample rate of 833 kS/s for single–channel output mode and 740 kS/s for dual–channel output mode.
- 24 digital I/O channels.
- 2 general purpose 32–bit counters/timers that can be used for PWM generation, encoder counting, frequency measurement, event counting, etc.

The PCIe–6321 board (installed on PCFA16 ÷ PCFA19 workstations) has similar specifications to PCI–6221, except for minor differences such as
- the DAC maximum sampling rates: 900 kS/s for single–channel output mode and 840 kS/s for dual–channel output mode.
- the increased number of general purpose counters/timers: 4 instead of 2.

3. **National Instruments BNC–2110 terminal board.**

4. **BNC–terminated connection cables** (three single–channel cable + one 5–channel cable for each workbench).

5. **DC power supply** with adjustable voltage output (laptop power adapter).

6. **PC workstation** running Matlab/Simulink. The **Real–Time Windows Target (RTWT)** toolbox is used to perform a “real–time simulation” that involves a direct interaction with the experimental setup.

The experimental setup parameters that are relevant for simulation and control design purposes are listed in Tab. 1. The nominal values of the parameters are deduced from accompanying data–sheets.

### 3 Real–time simulation

The Matlab/Simulink environment will be used throughout the course to support all the activities that are generally involved with the typical design flow of a control system. The implementation and experimental testing of a control system will be also done within the same environment, by resorting to the **Real–Time Windows Target (RTWT)** toolbox. This toolbox provides a real-time kernel for executing Simulink models on a computer running Windows. It includes library blocks to interface a Simulink model with several I/O devices that cover a wide range of I/O types, such as analog I/O signals, digital I/O signals, encoder/counter input signals, PWM/frequency outputs and communication protocols (serial, UDP, TCP, etc.). In order to run a Simulink model in real time with the RTWT support, it is first necessary to convert it into an executable code, that is next automatically loaded into memory and run by the RTWT kernel whenever the “real–time simulation” is started. The whole conversion process of a Simulink model into an executable code is managed by Matlab and is transparent to the user. The process basically consists of two stages: in the first stage, the Simulink model is automatically converted into a C language source code by the Real–Time Workshop (RTW) toolbox. Then, in the second stage, the generated C–code is automatically compiled using a supported C compiler.
<table>
<thead>
<tr>
<th>DC gearmotor nominal parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Motor type</strong></td>
</tr>
<tr>
<td><strong>Motor id</strong></td>
</tr>
<tr>
<td><strong>Nominal voltage</strong></td>
</tr>
<tr>
<td><strong>Nominal output power</strong></td>
</tr>
<tr>
<td><strong>Motor efficiency</strong></td>
</tr>
<tr>
<td><strong>Armature resistance</strong></td>
</tr>
<tr>
<td><strong>Armature inductance</strong></td>
</tr>
<tr>
<td><strong>No-load speed</strong></td>
</tr>
<tr>
<td><strong>No-load current</strong></td>
</tr>
<tr>
<td><strong>Stall torque</strong></td>
</tr>
<tr>
<td><strong>Back-EMF constant</strong></td>
</tr>
<tr>
<td><strong>Torque constant</strong></td>
</tr>
<tr>
<td><strong>Rotor inertia</strong></td>
</tr>
</tbody>
</table>

| Gearbox type | planetary gearbox |
| Gearbox id | Micromotor SA 23/1 |
| Gearbox ratio | 14 |
| Gearbox efficiency | 0.80 |

| Moment of inertia of external 72-tooth gear | 1.4 × 10⁻⁶ kg m² |
| Moment of inertia of extra disc | 3.0 × 10⁻⁵ kg m² |

<table>
<thead>
<tr>
<th>Sensors data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensor type</strong></td>
</tr>
<tr>
<td><strong>Sensed quantity</strong></td>
</tr>
<tr>
<td><strong>Sensor id</strong></td>
</tr>
<tr>
<td><strong>Pulses-per-rotation (ppr)</strong></td>
</tr>
</tbody>
</table>

(1) ×4 times larger when adopting a quadrature encoding mode for counting the encoder pulses.

| Sensor type | potentiometer "1" |
| **Sensed quantity** | output shaft angular position (absolute) |
| **Sensor id** | Spectrol 138-0-0-103 |
| **Resistance** | 10 kΩ |
| **Angle range** | ±170 deg |
| **Supply voltage** | 5 V |

| Sensor type | potentiometer "2" |
| **Sensed quantity** | flexible joint angular displacement (absolute) |
| **Sensor id** | Spectrol 357-0-0-103 |
| **Resistance** | 10 kΩ |
| **Angle range** | ±170 deg |
| **Supply voltage** | 5 V |

<table>
<thead>
<tr>
<th>Voltage driver data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Driver type</strong></td>
</tr>
<tr>
<td><strong>Driver gain</strong></td>
</tr>
<tr>
<td><strong>Driver cut-off frequency</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I/O board data</th>
</tr>
</thead>
</table>
| **Board id** | NI PCI-6221 (on PCFA20 ÷ PCFA27 workstations)  
NI PCIe-6321 (on PCFA16 ÷ PCFA19 workstations) |
| **ADC range** | ±0.2 V, ±1 V, ±5 V, ±10 V |
| **ADC resolution** | 16 bits |
| **DAC range** | ±10 V |
| **DAC resolution** | 16 bits |

Table 1: Experimental setup data.
3.1 Simulink and Real–Time Windows Target settings

Below are reported the steps required to configure a Simulink model to run in real–time with the support of the RTWT toolbox:

1. Prepare a working directory/folder (e.g. "Lab[...]") in your Desktop or Documents folders, with [...] denoting an identification label for the lab activity (e.g. "Lab01" for activity 1). Change the current working directory to your new working directory, by using the Matlab commands `pwd` and `cd` or the Current Directory field in the toolbar.

2. Open a new Simulink model, either by selecting New → Model from the Matlab window toolbar, or by invoking `simulink` from the command window and then using the New Button in the toolbar of the Simulink Library Browser window.

3. Save the Simulink model in the working directory, by selecting File → Save As in the model window menu (or push the equivalent button in the toolbar of the Simulink model window). Note that RTW generates some auxiliary files in the current working directory during the code generation process. Therefore, ensure to save the Simulink model file in your current working directory, to prevent the generation of undesired junk files in other folders.

4. To set the host and target parameters of the real–time simulation, select Simulation → Configuration Parameters in the Simulink model window menu. The Configuration Parameter window opens.

5. Click Solver in the Select list; in the Solver option box (see Fig. 3a) change Type to Fixed-Step and Solver to `ode3` (Bogacki-Shampine). Set Fixed-step size (fundamental sample time): to the desired sampling time ($\geq 0.001$ s). Set the Stop time: to Inf to enable a continuous running simulation. Press the Apply button to confirm the new simulation settings.

   Note: only fixed-step solvers are allowed for real–time simulations. The choice of the numerical solver only affects how the continuous–time blocks in the model are integrated. If no continuous–time blocks are present in the model, use the discrete solver to reduce the (online) computational load.

6. Click Data Import/Export on the Select list; under Save Options (see Fig. 3b), uncheck Limit data points to last. This enables the Simulink to save all the data during the simulation. Press the Apply button to confirm the new simulation settings.

7. Click Code generation on the Select list (see Fig. 3c); click the Browse button in the Target selection section: the System Target File Browser window opens. Select `rtwin.tlc` in the browser window, and press OK to save the configuration.

8. Press the Ok button in the Configuration Parameter window to close it and confirm all the new simulation parameters settings.

9. In the Simulink model window menu go to Code → External Mode Control Panel: the External Mode Control Panel window opens (see Fig. 4a).

10. In the Configuration section of the Simulink model toolbar, push the Signal & Triggering button: the External Signal & Triggering window opens (see Fig.4b).
Figure 3: Configuration Parameters window: (a) Solver Pane; (b) Data Import/Export Pane; (c) Code Generation Pane.

Figure 4: External Mode settings: (a) External Mode Control Panel; (b) External Signal & Triggering window.

11. In the Trigger section, change Duration to 100000 (with a sample time equal to 0.001, this choice allows to save 10 s of data). Push the Apply button to save changes, and then the Close buttons in the last two windows to close them.

12. In the Simulink model window toolbar, select External instead of Normal (default selection)
in the pull-down menu (see Fig.12). Alternatively, select Simulation → Mode → External.

13. Select File → Save in the Simulink model window menu (or push the equivalent button in the toolbar of the Simulink model window) to save the Simulink model, together with all its settings for real–time simulation.

3.2 Interfacing with the experimental setup

The RTWT blockset that enables the I/O interfacing of a Simulink model with the physical device is located under Real–Time Windows Target in the Simulink Library Browser window. The three blocks that are relevant for all the laboratory activities of this course are the Analog Input, Analog Output and Encoder Input. Below is reported a simple example showing how to configure them to work with the I/O boards installed in the laboratory workstations:

1. In the Simulink Library Browser, select the Real-Time Windows Target blockset (see Fig.6).

2. Drag & Drop the Analog Input, Analog Output and Encoder Input blocks in your Simulink model window.

The Analog Input block gets the current value of a specified ADC input channel (or a list of channels), and makes it available at its output port (i.e. “it imports data from the real world into the Simulink model”). In this example, the block will be used to (simultaneously) acquire the “Motor Current A”, “Motor Current B” signals of the Quanser SRV–02 unit. To monitor
the signals evolution in real time, connect a Scope block at the output of the Analog Input block, as shown in Fig. 7. Since the output of the Analog Input block will be configured as an array of two input channels, use a Demux block to separate the two channels into two individual scalar signals.

The Analog Output block sets a specified DAC output channel (or a list of channels) to the value provided at its input port (i.e. "it exports Simulink data to the real world"). In this example, the block will be used to set a certain voltage value at the "Motor Input" of the Quanser SRV–02 unit. For such purpose, connect a Constant block at the input of the Analog Output block, as shown in Fig. 7.

The Encoder Input block gets the pulse count of a specified encoder input channel. In this example, the block will be used to get the pulse count of the encoder installed in the Quanser SRV–02 unit. To monitor the encoder pulse count in real time, connect a Scope block at the output of the Encoder Input block, as shown in Fig. 7.

3. Double-click on the Analog Output block previously imported in the Simulink model. The Block Parameters: Analog Output dialog window opens.

4. In the Data acquisition board section, open the drop–down list (by pushing the "▼" button) and verify whether the PCI–6221 board (or PCIe–6321 board, depending on your workstation) appears in the list of I/O boards already installed and configured to work with Simulink + RTWT. Then:

   - if the I/O board is present (see Fig. 8a), select it for later use. Then push the Board setup button to verify whether the board has been installed with the correct settings. A new windows opens (see Fig. 9b).

   - if the board is not present, push the Install new board button: a pop-up menu opens (see Fig. 9a). Search the National Instrument PCI–6221 I/O board (or PCIe–6321 board, depending on your workstation) in the menu, under National Instruments (or National Instruments University of Padova 9/17
In the **Counter 0 mode** drop-down list, select the **Quadrature encoder** option. This enables the quadrature encoding mode for counting the encoder pulses (note: this setting is only relevant for the Encoder Input block that will be configured later). Next, push the button **Test** for testing the board installation: if the board has been properly installed, the **Board Test OK** window should open (see Fig. 9c). Finally, push the **OK** button to confirm the board selection and configuration.

6. In the **Timing** section of the Analog Output dialog window, set **Sample Time** to the desired value in seconds, or leave “−1” to inherit the sample time specified in the configuration parameters of the Simulink model. Set the **Maximum missed ticks** to zero.

7. In the **Input/Output** section, specify the list of DAC channels to use in the **Output channels** field. Set “1” in the field for using the first DAC channel, namely the channel marked as “AO 0” in the BNC–2110 terminal board, to generate the desired voltage reference to be passed to the DC motor voltage driver. The DAC output will be later connected to the voltage driver input in the SRV–02 unit via the BNC–2110 terminal board.

Set both the **Initial value** and **Final value** fields to zero, in order to turn the DC motor off when the simulation is not running.

8. Push the **Apply** button to confirm the new settings; then, push the **OK** button to close the window.

9. Double-click on the **Encoder Input** block previously imported in the Simulink model. The **Block Parameters: Encoder Input** dialog window opens.

10. In the **Data acquisition board** section, open the drop-down list (by pushing the “▼” button) and select the I/O board (PCI–6221 or PCIe–6321 board, depending on your workstation).
11. In the Timing section, set Sample Time to the desired value in seconds, or leave “−1” to inherit the sample time specified in the configuration parameters of the Simulink model. Set the Maximum missed ticks to zero.

12. In the Input/Output section, specify the list of encoder counters to read. Set “1” in the field for interfacing with the counter 0 of the I/O board (note: this is the counter that has been configured to count using a quadrature encoding mode during the board installation and setup). The counter inputs will be later connected to the two quadrature outputs of the SRV–02 encoder via the BNC–2110 terminal board.

In the Quadrature mode drop–down list, select the option quadruple to enable the counter to count using the quadrature encoding mode.

In the reset input function drop–down list, select the option reset to reset the counter register at the beginning of the real–time simulations. With this setting, the zero position reference is assumed as the actual position of the encoder shaft at the simulation startup.
13. Push the Apply button to confirm the new settings; then, push the OK button to close the window.

14. Double-click on the Analog Input block previously imported in the Simulink model. The Block Parameters: Analog Input dialog window opens (see Fig. 8c).

15. In the Data acquisition board section, open the drop–down list (by pushing the “▼” button) and select the I/O board (PCI–6221 or PCIe–6321 board, depending on your workstation) installed in the previous steps.

16. In the Timing section, set Sample Time to the desired value in seconds, or leave “−1” to inherit the sample time specified in the configuration parameters of the Simulink model. Set the Maximum missed ticks to zero.

17. In the Input/Output section, specify the list of ADC channels to use in the Input channels field. Set “[1,2]” in the field for using the first two ADC input channels, namely those marked as “AI 0” and “AI 1” in the BNC–2110 terminal board. The two ADC inputs will be later connected to the “Motor Current A” and “B” outputs of the SRV–02 unit, via the BNC–2110 terminal board.

In the Input range drop–down list, select the most appropriate voltage range for the ADC input channels. The “−5 to 5 V” input range would be sufficient if the motor armature voltage is limited below 5 V (note: the ADC output will be clamped to 5 V if the motor armature voltage exceeds that limit). However, since the motor armature voltage can be actually set up to reach the motor nominal voltage of 6 V, consider to use an extended range of “−10 to 10 V” to avoid any voltage clamping, at the expense however of a reduced ADC resolution.

18. Push the Apply button to confirm the new settings; then, push the OK button to close the window.

19. Save the Simulink model before proceeding (NOTE: IT IS GOOD PRACTICE TO SAVE THE MODEL FROM TIME TO TIME, TO PREVENT ANY DATA LOSS IN CASE OF A MATLAB CRASH).

3.3 Data displaying and saving

The ADC signals and the encoder pulse count can be visualised in real—time using the Scope blocks. To save the signals in the Matlab workspace for later post–processing, use either the To Workspace blocks (located under Simulink → Sinks in the Simulink Browser window), or configure the Scopes to save the displayed data in the Matlab workspace, as explained below:

1. Double-click on a Scope block whose logged signals has to be imported in the Matlab workspace at the end of simulation.

2. Push the Parameters button (see Fig. 10b) on the scope toolbar. The Parameters dialog window opens.

3. Click the History tab. Uncheck the Limit data points to last checkbox.
Check the *Save data to workspace* checkbox. In the *Variable name* field, specify the name of the variable onto which the data logged by the scope will be saved. The variable will be created in the workspace at the end of the simulation.

4. In the *Format* drop-down list, select the most appropriate format for the variable to be generated in the workspace. The *Structure with time* format is recommended, since it stores both time and data values into a single data structure. The time and data values can be retrieved by simply referencing the appropriate fields of the data structure: for example, if `simdata` is the name of the variable, use the following lines (in the Matlab workspace) to access the time and data fields:

- \( t = \text{simres}.\text{time} \) to retrieve the time values
- \( y = \text{simres}.\text{signals}(n).\text{data}(\cdot, m) \) to retrieve the \( n^{th} \) signal in the \( n^{th} \) set of axes within the scope. If a single signal is logged into a scope with a single set of axes, then the form `simres.signals.data` can be used as a shorthand.

### 3.4 Connecting the cables

Use the BNC-terminated cables to connect the SRV–02 unit with the BNC–2110 terminal board:

1. Connect the “AO 0” analog output channel of the BNC–2110 terminal board to the “Motor Input” of the SRV–02 unit.

2. Connect the “Motor Current A” and “Motor Current B” outputs of the SRV–02 unit to the “AI 0” and “AI 1” analog input channels of the BNC–2110 terminal board.

3. Connect the “Encoder” and “Encoder 90°” outputs of the SRV–02 unit to the “USER 1” and “USER 2” inputa of the BNC–2110 terminal board.

A direct connection between the two user-defined inputs “USER 1” and “USER 2” and the two (quadrature) inputs of counter 0 is established in the spring terminal block located in the *Digital and timing I/O* section of the BNC–2110 terminal board.

4. **VERIFY THAT THE REGULATED OUTPUT VOLTAGE IN THE POWER SUPPLY ADAPTER IS SET TO 12 V BEFORE PROCEEDING.** Connect the power supply to the SRV–02 unit by using the plug–in connector.
The final connection outline should look like that reported in Fig. 11.

**IMPORTANT PRECAUTION**: once the SRV–02 is turned on, DO NOT touch the voltage driver board with any conductive material (pens, metal clips, coins, etc.), in order to prevent the occurrence of short circuits that would produce an unrecoverable damage to the electronic circuit.

### 3.5 Building and running

In order to execute the Simulink model in real–time with the support of the RTWT toolbox, it is first necessary to compile it for generating an executable code, which will be next loaded into memory by the RTWT kernel at runtime. To run the real–time simulation, proceed as follows:

1. Push the *Build* button in the Simulink model window toolbar to generate the C–code, and compile/link it with the supported compiler/linker (see Fig. 12a).

   While the model is under building, you can check the progress of the building process by clicking the *View diagnostic* link at the bottom of the Simulink model window (see Fig. 12a). The *Diagnostic Viewer* window opens (see Fig. 12b). This window is also used by Simulink as the standard error dialog window, to inform the user of any error present in the Simulink model (or occurring during the building process).

2. Once the model has been builded, push the *Run* button in the Simulink model window toolbar to start the real–time simulation (note: it can take some time before the simulation actually starts).

   While the simulation is running, you can modify the DC motor armature voltage by changing the value of the Constant block connected to the Analog Output block. Since the modification takes effect immediately, a real–time interaction with the physical device is possible.
Because of the presence of the static friction, the motor does not spin for small values of the armature voltage. Slowly increase the value of the armature voltage until the motor starts to spin. Use the scope to monitor the encoder pulse count and voltage drop across the shunt resistor.

3. To stop the simulation, push the Stop button in the Simulink model toolbar.
4 Laboratory assignment

Modify the Simulink model of previous Sec. 3 to obtain both the motor current and speed measures. The motor (armature) current can be determined as

\[ i_a = \frac{\Delta V}{R_s} \]  

(3)

where \(\Delta V\) is the voltage drop across the shunt resistor \(R_s\). Regarding the motor speed, it can be ideally obtained by differentiation of the measured position. In practice, however, an ideal derivative block cannot be implemented, and one has to resort to “approximations” of it based on high-pass filters \(^1\). Compare the results obtained with three different high-pass filters:

a) a continuous–time first–order high–pass filter of the type

\[ H_1(s) = \frac{s}{T_1 s + 1} \]  

(4)

Select \(ode1\ (Euler)\) as the solver to use for the integration of (4) in the real–time simulation. This choice corresponds to discretise the continuous–time system (4) with the forward Euler method (see next point). Choose \(T_1\) such that the filter cut–off frequency is equal to 200 Hz (i.e. \((2\pi T_1)^{-1} = 200\) Hz). Set the solver fixed step size equal to \(T_s = 1\) ms.

b) a discrete–time high–pass filter obtained by discretising (4) with the forward Euler method, namely the IIR filter with transfer function

\[ H_2(z) = H_1(s) \bigg|_{s=z^{-1}T_s} = \frac{z^{-1}}{T_1 z + (T_s - T_1)} \]  

(5)

Set \(T_1\) and \(T_s\) as specified in the previous point.

c) the \(N\)–steps finite backward difference (FIR filter)

\[ H_3(z) = \frac{1 - z^{-N}}{N T_s} \]  

(6)

which corresponds to a discrete–time filtered derivative (with \(N = 1\), it is the conventional discrete–time derivative based on a first–order backward difference). Set \(N = 5\) as a first trial, and \(T_s = 1\) ms as in the previous points.

The resulting Simulink model should look similar to that reported in Fig. 13. Verify that

1. the two filters (4) and (5) produce the same results, thus confirming that the continuous–time blocks in the Simulink model are integrated by adopting the forward Euler method.

2. compared to (4) and (5), the filter (6) provides a higher rejection to the encoder quantisation noise. Explain why, by comparing the frequency responses of the three filters. Which is the one introducing a higher attenuation at high frequency ?

\(^1\)Another important method consists of using observers (e.g. Luenberger observers), through which a “zero phase lag” speed estimate can be obtain (note that by using high–pass filters, a phase lag is always present in the speed estimate). This approach will be consider later in the course.
Optional extra assignments:

4. discretise (4) by using the Tustin’s method; the discretised filter has the transfer function

\[ H_4(z) = H_1(s) \bigg|_{s=\frac{2z^{-1}}{T}} = \frac{2(z-1)}{(2T_1 + T)z + (T - 2T_1)} \]  \hspace{1cm} (7)

(note: obtain (7) by using the c2d routine of the Control System Toolbox, specifying ‘tustin’ as the discretisation method). Compare the response of (7) with those of the previous filters.

5. (More challenging) change the solver to (fixed–step) ode3. Verify that the response of the continuos–time filter is apparently inconsistent with those produced by the discrete–time filters. Try to explain the reason of such inconsistency, possibly with the aid of numerical simulations performed off–line with a variable–step solver (with adequate numerical accuracy and max step size).