Practical Course Matlab/Simulink

Simulink Control Design

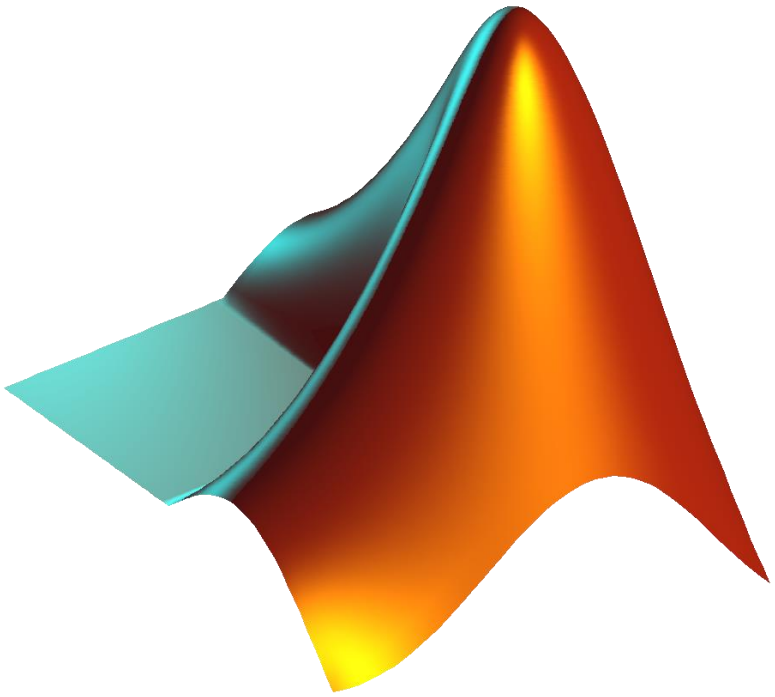

Table of Contents

Table of Figures

0 General Information and Advice

The following exercises cover MATLAB's Simulink Control Design Toolbox. They increase in difficulty and they also build upon each other, so you should start with the first exercise and continue with the second one. The exercises have been designed for MATLAB 2019a.

1 Pendulum Control

Like in previous sessions of the course, we will take the nonlinear pendulum (Figure 1-1) as an example. This time, however, a few more elements are added to the problem.

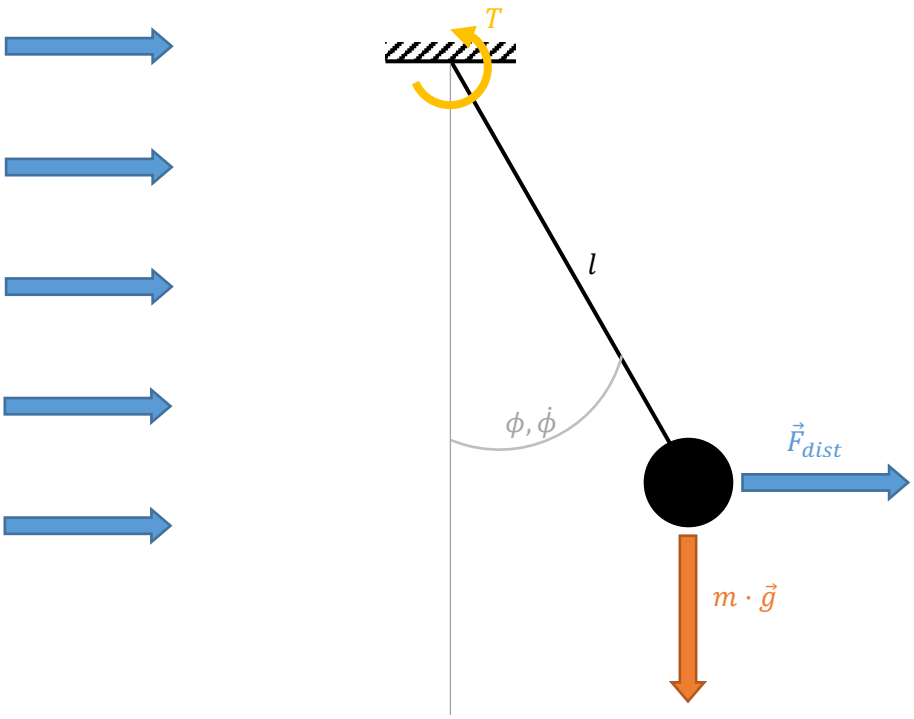

Figure 1-1: Schematic of the pendulum

Suppose that the mass is suspended from the attachment point by a rigid element (e.g., a rod) and that a torque T can be applied at the attachment point. A positive torque T acts counter-clockwise. Moreover, there is a horizontal flow of air that causes a constant horizontal aerodynamic force and thereby alters the pendulum's equilibrium points.

1.1 Operating Points

In this first part of the exercise, you will determine the resulting new (equilibrium) operating points. For the sake of this exercise, suppose you could not easily compute them. Therefore, you will use Simulink Control Design to first take simulation snapshots and then do an optimization-based operating point search.

Exercise

(1) In the folder Exercise 1, an initialization script called init_pendulum_with_disturbance.m is given. Execute it.

(2) Open the model pendulum_with_disturbance.slx and simulate it.

(3) From the Simulink model window, open the Linear Analysis Tool

(4) Take simulation snapshots at 1 s, 5 s and 50 s

(5) Examine the snapshots by double-clicking on the "op_snapshot1" variable that has been created. How do you interpret the values of `phi_dot` for the different snapshots? When can the excitation angle of the pendulum be considered steady-state? Verify that `phi_dot` has the following values for the snapshots:

- 1s-snapshot: 0.50228 rad/s
- 5s-snapshot: 0.53642 rad/s
- 50s-snapshot: 0.0093123 rad/s

(6) Save the 50-s-snapshot as "op_snapshot_50s" in the MATLAB base workspace by clicking on "Initialize model…"

(7) Trim the model using optimization-based operating point search. Import the 50s-snapshot saved before as a starting point. Note that you cannot import the "op_snapshot1" variable, because it contains three snapshots. Search for **steady state** `phi` and `phi_dot`. Force `phi_dot` and `input_torque` **to be zero**.

(8) Verify that the value of `phi` at this trimmed operating point has the value 0.44692 rad. Rename the operating point to "op_trim_A".

(9) Find the second operating point of the pendulum with zero input by specifying a proper initial value for the optimization-based search. Verify its steady-state property. Hint: Think about the two equilibrium states of the undisturbed pendulum. Note that `phi` is given in radians.

(10) Rename the second operating point to "op_trim_B". Copy both operating points that you found with the optimization-based search into the MATLAB base workspace (A and B).

1.2 Linearization

Now that you have found both equilibrium points of the pendulum, you can proceed with linearizing the nonlinear system around those states.

Exercise

(1) Create appropriate Linear Analysis Points in the model, such that the linear transfer from `input_torque` to `phi` can be determined. (Hint: There are multiple possibilities to achieve that. For example, you can use the model interfaces or an input perturbation in combination with an output measurement.)

(2) Linearize the system around operating point A by creating a step plot. What can you observe in the plot? Note that the amplitude of the system represents the deviation from the linearization point. Think about why the amplitude has a nonzero offset.

(3) Using the plot, estimate and note the period of the oscillation. (Hint: You can click on the plotted line to visualize data point values.) T = 2.33 s

(4) Rename the linearized system to "linsys_A".

(5) Linearize the system around its second operating point (B) and create a new step plot. What can you observe? (Hint: The linearization should fail due to the instability of the operating point.)

(6) Rename the second linearized system to "linsys_B".

1.3 Frequency Response Estimation

This part of the exercise will familiarize you with frequency response estimation.

Exercise

(1) To prepare frequency response estimation, create a sinestream input. Initialize frequencies and parameters based on the stable linearized system.

(2) Create a frequency response estimation at the first operating point (A) by generating a Bode plot.

(3) Add the Bode diagram of the stable linearized system to the Bode plot of the frequency response estimation and compare the results.

(4) At what frequency do you observe a peak in the Bode Magnitude plot? Compare this frequency with the oscillation period you previously determined. Do they agree? (Hint: Consider $T[\mathrm{s}] = \frac{2 \cdot \pi}{f_{res}[\mathrm{rad/s}]}$ with f_{res} being the resonance frequency.) at 2.69 Hz
2*pi/2.69 = 2.3357, they agree with each other

(5) Select the tab of the Bode plot and print the plot to a new figure. Annotate the peak by inserting a "Text Arrow" in the figure. As text, enter "`X s, determined by Y`", where X is the **oscillation period** of the linear system and Y is your name. Save this figure as "FRE.fig".

(6) Copy both linear system variables ("linsys_A", "linsys_B") to the MATLAB base workspace. What are the poles of both linear systems? Do the poles fit to your observations? (Hint: Use the MATLAB command `pole`)
yes. the poles of A are -0.089 +/- 2.691i, both in left side
the poles of B are 2.6051 and -2.7829, one is in right side and thus unstable

1.4 Cleaning Up

In order to submit the results of your experiment, conduct the following steps.

Exercise

(1) In the exercise folder, you will find a script called export_pendulum_with_disturbance.m. Open it an look at the code, which exports part of your MATLAB base workspace to a file. Execute the script.

(2) Verify that all the variables have been exported correctly. For example, you can use the "Import Data" feature to do that.

(3) Upload both your export (pendulum_export.mat) and the figure (FRE.fig) to Moodle.

2 Moving Robot – Wheel Speed Control

In this last exercise you will repeat the process of tuning a PID controller for the wheel speed of the moving robot. You did this before using the Control System Toolbox. This time, you will use Simulink Control Design. Remember, each of the two wheels of the robot is driven by an electric motor through a set of gears, as shown in Figure 2-1.

The relationship between motor command u and wheel velocity v, as represented by the block diagram in Figure 2-2, has already been implemented in the Simulink template file.

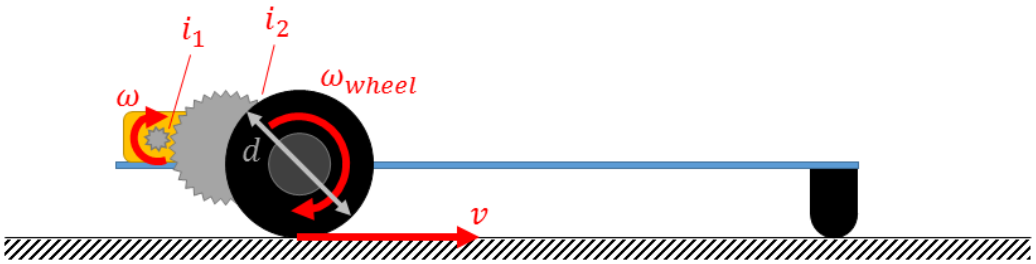

Figure 2-1: Side-view of robot with engine, gears and wheel

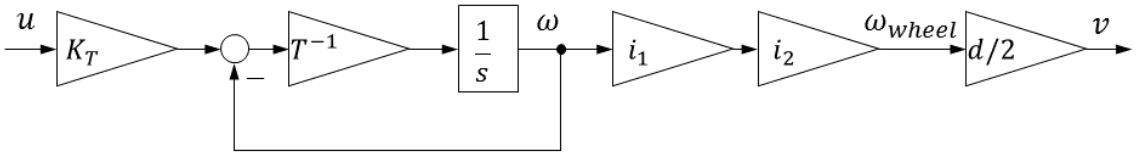

Figure 2-2: Block diagram of the wheel speed transfer function (command path only)

Exercise

(1) In the folder Exercise_2, an initialization script called init_wheel_speed_control.m is given. Execute it.

(2) Open the model wheel_speed_control.slx and familiarize yourself with the model.

(3) Simulate the model. Observe the outputs with the scope. Note the effect of the disturbance at $t = 5\,\mathrm{s}$. Now we want to build a controller that is able to reject the disturbance and ensures that the velocity `v` follows the commanded `v_target`.

(4) Replace the `Gain` block (on the top level of the model) with a feedback loop of `v`, containing a PI-controller structure like the one shown in Figure 2-3. Note that for the scope of this exercise, one gain (i.e. one degree of freedom) of the controller remains fixed, whereas the other gain will be tuned. Give the `Tunable_Gain` an initial value of 1000.

(5) Open the "Control System Designer" (Analysis → Control Design → Control System Designer…)

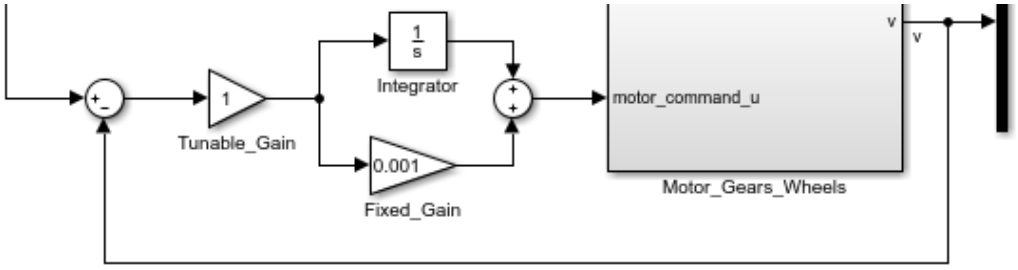

Figure 2-3: Feedback loop

(6) In the new window, click “Add blocks…”, select the block `Tunable_Gain` as the only block to tune and click “OK”. (Note: You can reopen the initial window with “Edit Architecture”.)

(7) Under the tab “Signals” and “Add Locations…”, add the signals `v` and `v_target`. As the output of the tunable gain is not required, remove the respective signal from the list.

(8) Under the tab “Linearization Options”, choose `Model Initial Condition` as operating point. Finally accept the architecture by clicking “OK”.

(9) For analyzing the system behavior, add a “New Step” plot. Choose “New Input-Output Transfer Response”, add `v_target` as input signal and `v` as output signal. Give this response a name, e.g. `v_response`, and click on “Plot”. Additionally, add a “New Bode” plot as second analysis plot, and select `v_response` as response. Note that this is a closed-loop Bode diagram showing the response of `v_target` to the input `v`. (You may have to zoom out in order to have a better look at the plot.)

(10) Now we want to tune the `Tunable_Gain`. Under “Tuning Methods”, add a “Bode Editor” and specify the signal `v` as location for the open-loop response. Note that this is an open-loop Bode diagram of `v`. You can display the plots next to each other with “View” and “Left/Right”.

(11) In the open-loop bode plot, drag the blue magnitude line and observe what happens with the step response of the system and the closed-loop Bode diagram. You effectively tune the gain by dragging the line.

(12) As design goal for our controller, we want the frequency response bandwidth to be $9\,\mathrm{rad/s}$. This ensures that the velocity `v` properly follows the commanded `v_target` while suppressing possible sensor noise. The bandwidth is defined as the frequency, where the closed-loop Bode magnitude crosses $-3\,\mathrm{dB}$, which corresponds to a damping of $1 - 1/\sqrt{2} \approx 29\,\%$. You can see the bandwidth in the closed-loop Bode diagram and it is displayed in the bottom left corner of the open-loop Bode plot. Additionally, you can open a “Closed-Loop Bode Editor” from the tuning methods, select `v_response`, and observe the markings in the Bode diagram.

(13) Adjust the blue open-loop magnitude line to reach a frequency response bandwidth of $9\,\mathrm{rad/s}$ and set the value of `Tunable_Gain` by clicking on “Update Blocks”. What is the resulting value of the tunable gain? 5028.5

(14) Simulate the model. Observe the outputs with the scope and see how the controller rejects the disturbance now.

(15) Try out other tuning methods. When using the PID Tuner, ensure to select “P” as controller type, since we are only tuning a proportional gain. Feel free to also modify the architecture of the controller and the blocks to be tuned.

(16) Finally, save the Simulink model with your controller (wheel speed control.slx) and upload it to Moodle.