Lab 4: Modelling over-causal system

MCHA3400

Semester 1 2024

Introduction

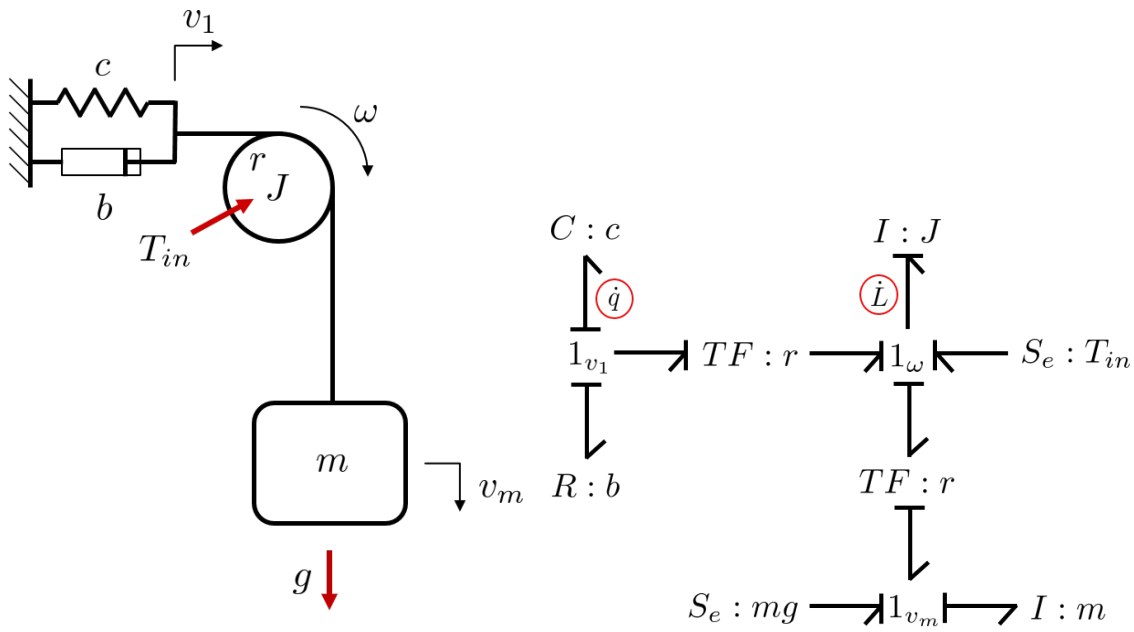
In lab 3, we simulated two physical systems in Matlab by using the native ODE solvers. In this lab, we will continue to develop these skills by constructing a simulation for an over-causal system. Additionally, we will implement an output function which can be used at a later date to interface the simulation with external targets.

This lab is worth 1.5% of your overall grade. Show your results to your tutor by the end of your lab session to be awarded marks. Labs will also be accepted at the beginning of your lab 5 session. After this point in time, no marks will be awarded.

Lab task 1: Simulation of an over-causal system

A simple mechanical system containing a mass-pulley assembly is shown in Figure 1a. An input torque is applied to a pulley wheel which is attached to a spring-damper system and a mass via a rigid connection. The pulley wheel has moment of inertia J and radius r. The mass is influenced by the attached cable and gravity.

The bond graph describing the pulley system is shown in Figure 1b. Causality has been assigned by placing the inertia associated with the pulley into integral causality and propagating. This propagation has forced the inertia associated with the mass into derivative causality, implying that the system is over-causal.



(a) A simple mechanical system.

(b) Bond graph of the mechanical system.

Figure 1: A simple mechanical system with over-causal bond graph.

A state-space model for the over-causal system can be resolved by first assigning states to all energy-storing components in integral causality. Propagation follows the standard procedure, with the exception that the inertia in derivative causality requires application of the inverse of the standard inertia CCRs. The complete bond graph with power variables fully propagated is shown in Figure 2 where

$$\begin{aligned} A &= -\frac{1}{c}q - br\frac{1}{J}L \\ B &= -mg + mr\frac{1}{J}\dot{L}. \end{aligned} \tag{1}$$

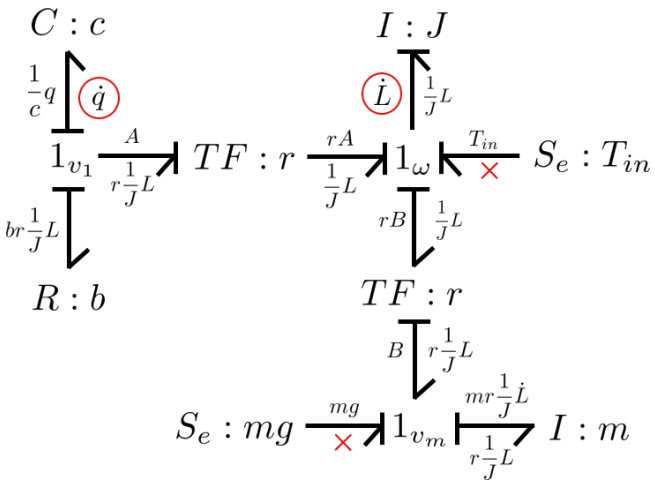


Figure 2: Bond graph of the mechanical system with propagated states.

Tasks:

a) An implicit expression describing $\dot{L}$ can be extracted from the bond graph shown in Figure 2, given by

$$\dot{L} = r\left[-\frac{1}{c}q - br\frac{1}{J}L\right] + T_{in} - r\left[-mg + mr\frac{1}{J}\dot{L}\right]. \tag{2}$$

Rearrange this expression to make $\dot{L}$ the subject.

b) Create a new file in Matlab and include the commands `clear` and `clc` at the top of your script. Create a parameters structure called `params` that will store all of the system parameters. Add the following parameters to the structure.

$$\begin{aligned} r &= 0.3 & b &= 36.7 \\ m &= 6 & J &= 2.6 \\ c &= 2.6 \times 10^{-2} & g &= 9.8. \end{aligned} \tag{3}$$

c) Create two anonymous functions that describe the behaviour $\dot{L}$ and $\dot{q}$ as a function of states and inputs. Once completed, combine all states into a single state equation of the form

$$\dot{x} = f(x, u), \tag{4}$$

where $x = [L, q]$ is the state of the system and $u = T_{in}$ is the input.

> **Note**
>
> For causality-constrained systems, it is difficult to implement a simulation directly from the CCRs and SSRs as we have done with previous simulations. For this task, resolve a set of state-space equations and implement them directly into Matlab.

d) A PD controller is to be applied to the system using the torque input after a time of 10 seconds has elapsed. The definition of the control signal is given in (5). Create an explicit function at the bottom of your script that describes the input torque as a function of time and states.

$$T_{in}(t,\omega,q) = \begin{cases} 0, & t < 10 \\ -k_p(q-q^\star) - k_d\omega, & t \geq 10. \end{cases} \tag{5}$$

where $k_p = 200, k_d = 10$, $q^\star = 0.5$ and $\omega = \frac{1}{J}L$.

e) Using the `ode45` solver with a relative error tolerance of 1×10^{-6}, simulate the mechanical system for 20 seconds from the initial conditions $L(0) = 0, q(0) = 0$. Plot the angular velocity ω and spring displacement vs time. Your results should agree with Figure 3.

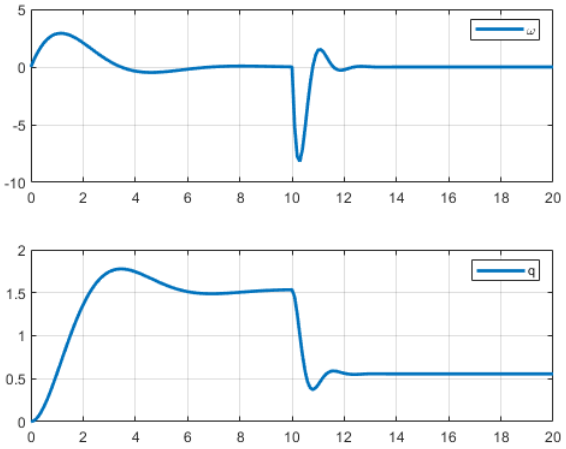


Figure 3: Expected simulation results.

f) Adjust your simulation code to use a fixed output step size of 0.1 seconds. Implement an output function that displays the simulation times at which the output function was called. The display in the console should be similar to Figure 4.

```
Time in output function
    0.1000

Time in output function
    0.2000

Time in output function
    0.3000

Time in output function
    0.4000

Time in output function
    0.5000

Time in output function
    0.6000
```

Figure 4: Expected console output from output function.