DESIGN FOR CPS SHORT COURSE
ASSIGNMENT 1: INVERTED PENDULUM CONTROLLER

EDWARD A. LEE

Due date: Wednesday, May 11, 2022, AOE

1. Reading

(1) Lingua Franca documentation: https://lf-lang.org
(2) Lohstroh, et al., "Toward a Lingua Franca for Deterministic Concurrent Systems," TECS, May 2021.
(3) Chapter 1 of Lee & Seshia, Introduction to Embedded Systems, MIT Press, 2017.
(4) (optional) Lee, "The Past, Present and Future of Cyber-Physical Systems," *Sensors*, 2015.

2. Programming Assignment

Your task is to implement a controller for an inverted pendulum in Lingua Franca using the C or Python target. Figure 2 illustrates the mechanism you will be controlling (see also a YouTube video of one in action). It has a horizontal arm driven by a motor, and at the end of that arm is a free-swinging pendulum. You can download an LF reactor that simulates the pendulum from here:

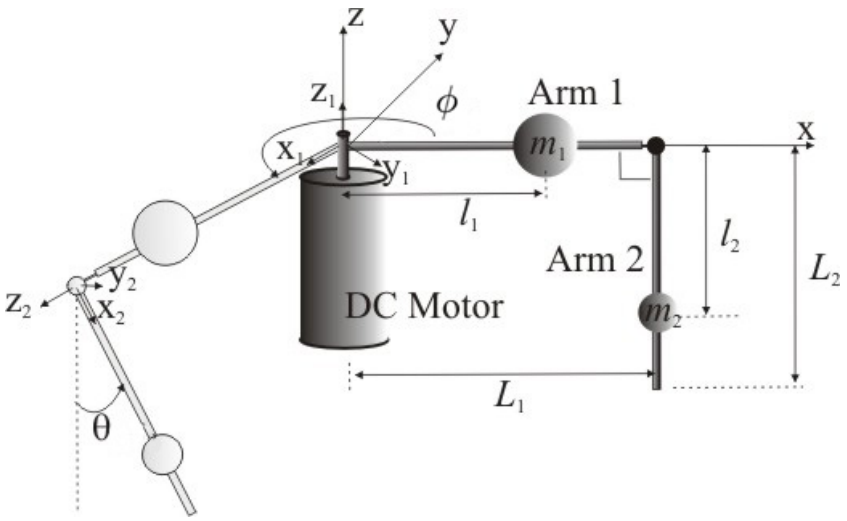

Figure 1. Schematic of the Furuta pendulum from Wikipedia by Benjamin Cazzolato — CC BY 3.0.

- **C version:**
  https://ptolemy.berkeley.edu/∼eal/cps/code/c/PendulumSimulation.lf
- **Python version:**
  https://ptolemy.berkeley.edu/∼eal/cps/code/python/PendulumSimulation.lf

Save this file to your disk as a Lingua Franca file, for example as `PendulumSimulation.lf`. You will want to import the `PendulumSimulation` reactor into your own Lingua Franca program. This simulation has input u, a control signal that applies torque to the horizontal arm to vary its angle ϕ. It has four outputs, θ, $\dot{\theta}$, ϕ, $\dot{\phi}$, which are, respectively, the angle θ of the pendulum (where 0 is pointing straight up), the angular velocity $\dot{\theta}$ of the pendulum, the angle ϕ of the arm, and the angular velocity $\dot{\phi}$ of the arm.

By default, the pendulum's initial angle is $\theta = -\pi$ radians. In the code, this is given by the parameter `initial_theta`, which has default value -3.14159. This means that the pendulum is pointing straight down. To make your job easier, you should override this so that the pendulum is initially already almost balanced vertically, such as `initial_theta = 0.1`.

Your job has been made much easier by your colleagues, who are experts in control systems and have provided you with a design for the controller:

$$u(t) = 1.7\theta + 0.3\dot{\theta} + 0.03\phi + 0.06\dot{\phi}.$$

(Feel free to experiment with these numbers to try to get a better controller.) Your only task is to realize this controller in software using Lingua Franca and verify that it works.

The simulator reactor, by default, outputs its sensor readings (the two angles and angular velocities) every 5 milliseconds, for a sample rate of 200 Hz. You should create a reactor that realizes the above controller and provides the input u to the simulation. To demonstrate that it works, you should plot the angle θ of the pendulum as a function of time for four or five seconds of operation, at least.

Second part: Once you have your pendulum stabilized, it is time to introduce a disturbance. The pendulum simulator has a second input `d` that, when it receives an event, injects an instantaneous change in the angular velocity of the pendulum, as if the pendulum had been tapped by a hard object. Your task is to introduce a disturbance at some time after your pendulum has stabilized and show that your pendulum recovers. How big a disturbance can it tolerate?

Specific deliverables: Send by email to eal@berkeley.edu:

(1) Your plots as PDF or image files.
(2) Your well-commented Lingua Franca program(s).
(3) Your estimate of the maximum initial angle *theta* and disturbance d that this controller can handle without the pendulum falling.

If you use the Python target, you can create your plot using the `matplotlib` library. If you use the C target, you can create your plot by printing time-value pairs to a file and using `gnuplot` to create the plot from those.

NOTE: If you use the C version, you may need to add the following target property in order for your program to compile:

```
target C {
    flags: "-lm"
}
```

This is because the imported file uses the standard C math library, and currently there is no way to specify this requirement in the imported file. The above seems to not be required on Macs, but is required on Linux. Moreover, if you are not using gcc or clang as your compiler, even this won't work. In that case, create a file called "math-include.cmake" with the following contents:

```
// file: math-include.cmake
target_link_libraries( ${LF_MAIN_TARGET} m )
```

and then use the following target property:

```
target C {
    cmake-include: "math-include.cmake?
}
```

There is an open issue in the Lingua Franca repo to fix this problem.