Prelab 0 and Prelab 1A. 2-Link Arm: Trajectory Planning, Velocity Filtering, and Control

Problems 1 and 2 are due as "Prelab 0": 7pm Monday, October 7.
Problems 3, 4, and 5 are due as "Prelab 1A": 7pm Monday, October 14.

Be sure you have the entire zip-file needed for Prelab 1A. The assignment involves modifying an existing Simulink (.slx) file. The Overview is long, simply to walk through some derivations, with the actual assignment at the end, and with Equations 15, 18, 21 and 26 being most important.

Overview

As the title of this Prelab implies, your goals are to design: [1] trajectories for a 2-link arm, [2] a causal velocity filter, and [3] control. Here, each of two motors in the arm will be controlled with a Proportional-plus-Derivative (PD) controller, which means your command signal to the motor will have the following form:

$$p_m = K_p \cdot (\theta_{des} - \theta) + K_d \cdot (0 - \dot{\theta}). \tag{1}$$

Here, p_m is your input signal to the motor, which (for the Lego EV3 system) will consist of integers between -100 and 100. Abstractly, think of -100 to 100 corresponding to -100% to +100% of some maximum voltage input to the motor[1]. Basically, assume p_m is proportional to voltage to the motor.

As will eventually be described in lecture, the transfer function from input p_m to angular *velocity*, ω, of the motor as an output can be fairly well-modeled[2] as first-order:

$$\frac{\Omega(s)}{P_m(s)} = \frac{K_{DC}}{\tau s + 1}, \tag{2}$$

where τ is some positive-valued *time constant*. This is an equation in the Laplace domain, for which it is typical to use a capital letter for the input (denominator) and output (numerator) variables at left, and to include "(s)" to indicate we have a transfer function in the Laplace domain; e.g., Ω is a capital ω. The system is first-order, since the highest term in s in the denominator is $s = s^1$.

Poles of the system are solutions for "s" in which the output could be non-zero, despite having the input constrained to exactly zero. In other words, they are solutions in which the denominator of Eq. 2 is zero, i.e., solutions for:

$$\tau s + 1 = 0. \tag{3}$$

An nth-order system has n poles. Here, we have one pole. It is at

$$s = -\frac{1}{\tau}, \tag{4}$$

and when the input, $p_m(t)$, is simply held at a constant value, say $p_m(t) = V_{pm}$, the output of an nth-order linear system will be of the form:

$$C_0 + C_1 e^{s_1 t} + C_2 e^{s_2 t} + ... + C_n e^{s_n t}, \tag{5}$$

so for our output *velocity*, $\omega(t)$,

$$\omega(t) = C_0 + C_1 e^{s_1 t} = C_0 + C_1 e^{-\frac{t}{\tau}}. \tag{6}$$

[1] For example, from a PWM (pulse-width modulated) signal.
[2] "All models are wrong but some are useful." `https://en.wikipedia.org/wiki/All_models_are_wrong`

Next, let's rewrite Eq. 2 as a differential equation, by interpreting s as d/dt:

$$\tau \frac{d\omega}{dt} + \omega = K p_m = K V_{pm}. \tag{7}$$

Rearrange Eq. 7 with the highest derivative output term, angular acceleration, alone on one side:

$$\frac{d}{dt}\omega = -\frac{1}{\tau}\omega + \frac{K}{\tau} V_{pm} \tag{8}$$

Using the expression in Eq. 6 for $\omega(t)$, we can rewrite Eq. 8, toward solving for C_0 and C_1:

$$-\frac{1}{\tau} C_1 e^{-\frac{1}{\tau}} = -\frac{1}{\tau} C_0 - \frac{1}{\tau} C_1 e^{-\frac{t}{\tau}} + \frac{1}{\tau} K V_{pm}. \tag{9}$$

The terms with C_1 (on the left versus right side) are identical here (as they should be), allowing us to solve for C_0:

$$C_0 = K_{DC} V_{pm}. \tag{10}$$

Because $\tau > 0$, we know that $e^{-\frac{t}{\tau}}$ will decay exponentially toward zero as $t \to \infty$; i.e., $e^{-\infty} = \frac{1}{e^{\infty}} = 0$. Generalizing, all the e^{st} terms from Eq. 5 die away to zero if the system is STABLE (meaning the real part of each pole, s_i, is strictly negative).

We solve for C_1 by requiring Eq. 6 to match the initial condition(s) of the system (i.e., at $t = 0$), again with the same assumption that p_m is a constant (for all $t \geq 0$). Assuming the motor starts at rest (velocity of zero), with $\omega(0) = \omega_0 = 0$, then from Eq. 6:

$$\omega_0 = \omega(t = 0) = C_0 + C_1 e^0 = C_0 + C_1 = K_{DC} \cdot V_{pm} + C_1, \tag{11}$$

so

$$C_1 = \omega_0 - K_{DC} V_{pm} = -K_{DC} V_{pm}. \tag{12}$$

We are more interested in the angle as output, which is the integral of angular velocity, of course. Plugging in for C_0 and C_1 and then integrating Eq. 6, we get:

$$\theta(t) = \int \omega(t))dt = K_{DC} V_{pm} \int \left(1 - e^{-\frac{t}{\tau}}\right) dt = C_i + K_{DC} V_{pm} \left(t + \tau e^{-\frac{t}{\tau}}\right), \tag{13}$$

where C_i is an integration constant. For all labs, the motor encoders are initialized (arbitrarily) at $\theta_0 = 0$ (when $t = 0$), from which we can solve for C_i:

$$C_i = -\tau K_{DC} V_{pm}, \tag{14}$$

so, to summarize: if we start with zero initial conditions (for both motor angle and velocity) and then assume $p_m = V_{pm}$ as a constant value for all $t \geq 0$ (e.g., a step input),

$$\theta(t) = K_{DC} V_{pm} \left(t + \tau e^{-\frac{t}{\tau}} - \tau\right) \tag{15}$$

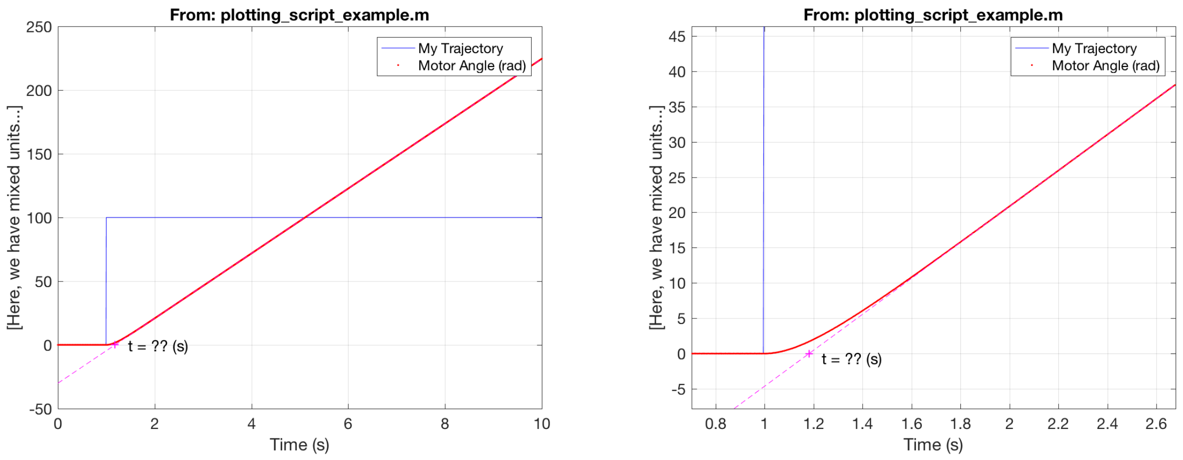
The equation above tells us what to expect, "open loop", when we send a step input command to the motor. In this prelab assignment, you are to solve an inverse problem, by applying a step input to the system (of magnitude V_{pm}) and using the resulting step response to determine the remaining two constants in the Equation 15: K_{DC} and τ. This is a form of system identification.

System Identification

Let's look at the behavior of $\theta(t)$ in Eq. 15, both for $t = 0$ and in the limit for $t >> \tau$.

- When $t = 0$: $\quad \theta = 0$ and $\dot{\theta} = 0$. Recall we assumed this, as initial conditions.
- When $t >> \tau$: $\quad e^{-t}\tau \approx 0$, so $\theta(t) \approx K_{DC} \cdot V_{pm} \cdot (t - \tau)$.

This is just a straight line. At $t = \tau$, "$t - \tau$" is zero, so this line goes through the point $(\tau, 0)$. Below is an example; you can generate a similar plot using the MATLAB-based files that go with this prelab.



Above is a step response for the simulated motor system, using $V_{pm} = 100$. The plot at right is simply "zoomed in" on the same data shown at left. We will often refer to a dynamic system to be controlled as a "plant". This is an open-loop step response of our plant.

PD Control of the Plant

In lab, we will be interested in controlling the angle of each of multiple motor output shafts, to perform various tasks. Note the relationship between angle (θ) and angular velocity ($\dot{\theta}$) is simply:

$$\omega = \frac{d\theta}{dt} = \frac{d}{dt}\theta. \tag{16}$$

Replacing $\frac{d}{dt}$ with s, we get:

$$\Omega(s) = s\Theta(s) \quad \rightarrow \quad \frac{\Theta(s)}{\Omega(s)} = \frac{1}{s}. \tag{17}$$

Based on Eq. 2, we can then find the transfer function to angle, θ, as output:

$$\frac{\Theta(s)}{P_m(s)} = \frac{1}{s} \cdot \frac{\Omega(s)}{P_m(s)} = \frac{K_{DC}}{\tau s^2 + s}. \tag{18}$$

Let's rewrite Eq. 18 as a differential equation. First, rearrange the s-domain terms:
$\Theta(s) \cdot (\tau s^2 + s) = P_m(s) \cdot K_{DC}$, $\quad$ and then replace s with d/dt and replace s^2 with d^2/dt^2:

$$\tau\ddot{\theta} + \dot{\theta} = K_{DC} \cdot p_m, \tag{19}$$

and let's consider a CONTROL LAW for p_m: $\quad p_m = -K_p\theta - K_d\dot{\theta}$. This yields:

$$\tau\ddot{\theta} + \dot{\theta} = -K_{DC}\left(K_p\theta + K_d\dot{\theta}\right) \tag{20}$$

which we can rewrite as:

$$\ddot{\theta} + \left(\frac{1}{\tau} + K_{DC}K_d\right)\dot{\theta} + (K_{DC}K_p)\,\theta = 0. \tag{21}$$

The closed-loop system dynamics given in Eq. 21 are those of a classic second-order system:

$$\ddot{\theta} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0. \tag{22}$$

To design a controller, you must select values for the control gains, K_p and K_d. We suggest it is more intuitive to think of this as a design of damping ratio, ζ, and natural freqency, ω_n. Picking a value of ζ between 0.7 and 1.0 (critically damped) would produce little (or no) overshoot in a step response. The larger ω_n is, the better the ability to "track" a reference; however, if K_p gets too large, the motor will just saturate (i.e., recall the magnitude of p_m can never exceed ± 100).

Causal Velocity Filter of Encoder Data

As mentioned at the end of Lecture 1, the Lego EV3 motors output encoder data which gives the angle of the motor, rounded off (i.e., "quantitized") to the nearest degree. Assume we get encoder data at discrete times, with sampling time T between samples. At step $(n+1)$, a simple estimate, v_{n+1} of angular VELOCITY would be:

$$\frac{d}{dt}\theta = \omega_{n+1} \approx v_{n+1} = \frac{\theta_{n+1} - \theta_n}{T}. \tag{23}$$

As illustrated in the Lecture 1 m-file on GauchoSpace [3], we can instead generate a more smooth estimate by using a weighted average of Eq. 23 and the previous velocity estimate, v_n:

$$v_{n+1} = f \cdot v_n + (1-f)\cdot\left(\frac{\theta_{n+1} - \theta_n}{T}\right). \tag{24}$$

Equation 23 corresponds to a choice of $f = 0$ (in Eq. 24).

Just as we can rewrite a continuous-time differential equation using Laplace notation (with the s operator), we can rewrite a (discrete-time) *difference* equation using the discrete-time operator: z. Here, z^n refers to the value of a signal at the n^{th} time step; e.g., $y_{n+3} \rightarrow z^3 Y(z)$, in representing a discrete-time equation like Eq. 24. Let's rewrite this equation with our desired "output", v_n, on the lefthand side, and with all terms with the input, θ_n, on the righthand side:

$$V(z)\cdot(z-f) = \Theta(z)\cdot\frac{1-f}{T}(z-1). \tag{25}$$

Rearranging the terms, we get the following discrete-time transfer function:

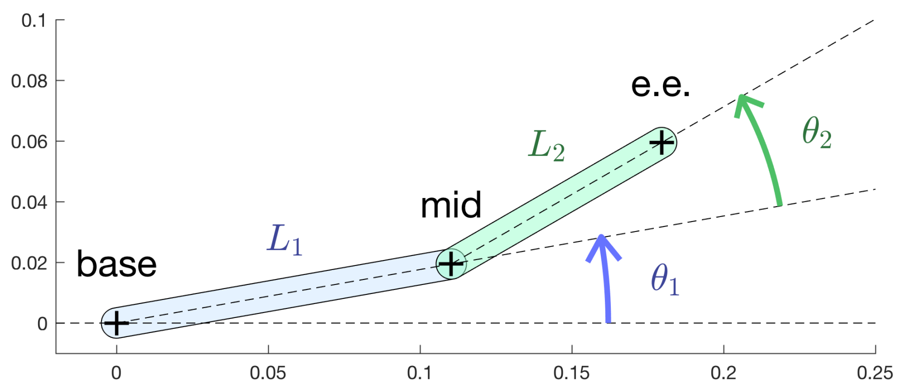
$$\frac{V(z)}{\Theta(z)} = \left[\frac{1-f}{T}\right]\cdot\frac{(z-1)}{(z-f)} = \frac{(Az-A)}{(z-f)}. \tag{26}$$

Here, T is the sample time you set for your Simulink model, so the only open parameter to choose is f. To implement this with a "Discrete Transfer Fcn" block in MATLAB, it may simplify things to use define $A \equiv (1-f)/T$, as illustrated at right in the equation above.

Trajectory Planning Finally, you will need to generate (and test) some interesting reference trajectories for the 2-link arm to track. For Lab 1A, you are to design a system to enable the end effector of the arm to track simple shapes, such as a triangle and a circle.

[3] Look for "Lecture 1 – MATLAB code", which when saved should be named "vel_causal_example.m"

The forward kinematics are very straight-forward. For Lab 1, θ_1 and θ_2 will be the angles of the motor outputs. Therefore, θ_1 measures an absolute angle of the first link, and θ_2 measure a relative angle, between the first and second links. Referring to the figure above, where the base is arbitrarily set to



coordinates $(0, 0)$, the end effector (point e.e.) is at:

$$x_{ee} = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \tag{27}$$

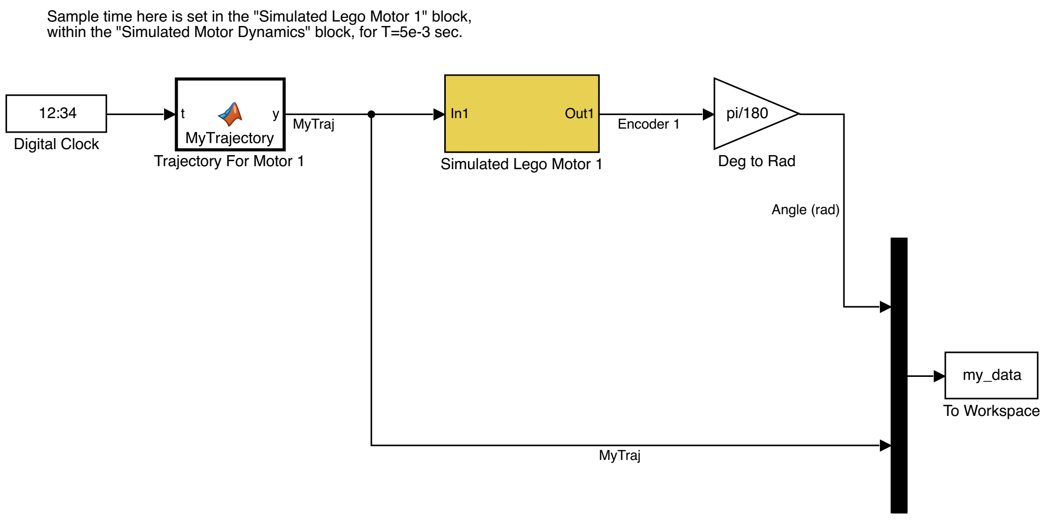
$$y_{ee} = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \tag{28}$$

That is, these equations define the "forward kinematics" of the system. If we assume $\theta_2 \geq 0$, only (so the elbow does not "bend backwards"), then we can solve for the inverse kinematic (IK) solution, to convert a desired (x_{ee}, y_{ee}) point into corresponding θ_1 and θ_2 angles for the arm. Of course, the arm has a limited reachable workspace (RWS), and there will only be feasible IK solutions for (x_{ee}, y_{ee}) points within this region.

The Actual Assignment

There are several files you should have, to complete this assignment, which should all be in the compressed zip-file that contained this pdf. Start with this Simulink model file:
→ Prelab1a_simulink_starter.slx

Here is what the Simulink model you will start with looks like:



• Open this file. Run it, by pressing the "run" button (green circle, with a black triangle), which should be an icon along the top (and is also a option in the "Simulation" pull-down menu).

• Then, run the MATLAB plotting script, called: plotting_script_example.m

• You can use the data from the open-loop step response for system identification, to find τ and K_{DC}.

The yellow block is a simulated plant, similar to a Lego motor system. Double-clicking on "Trajectory For Motor 1" should open a MATLAB function, which is currently set to send a step of magnitude 100, starting at t=1 second. When estimating τ, don't forget the step did not start until t=1!

To complete the assignment, modify this Simulink model and write any required additional MATLAB scripts and/or functions to do the following. An example of a modified Simulink model appears at the end of the assignment.

1. Produce reference signals to be tracked, to trace an equilateral triangle that is 5 cm on each side. This requires writing code for the inverse kinematics of the system, to solve for required joint angles of the motors over time. Assume $L_1 = 11.2\ cm$ and $L_2 = 8\ cm$, but keep these as variables in the code (which may change somewhat for the actual lab).

2. Estimate velocity (rad/s), from encoders (which output degrees), via a causal filter. Assume a sampling time of $T = 0.005$ seconds. You just need to pick "reasonable" values for f and A, in Equation 26; experiment to do so. (Once f is chosen, A is deterministically a function of f.) These will be entered into a "Discrete Transfer Fcn" blue, labeled "Causal Velocity Filter" on the next page.

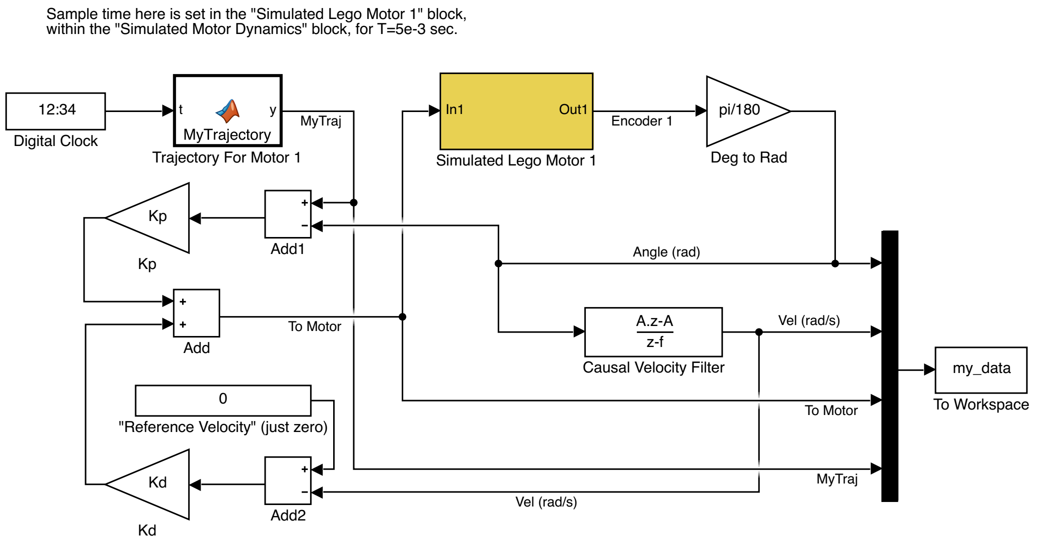
3. Design and include a PD controller for the model. This requires choosing K_p and K_d. First, identify the open-loop plant. Experiment to choose ζ and ω_n that track your reference signal fairly well. Plotting both the reference and actual angle signals will help. If your K_p gain gets too high, the motor will simply saturate, since it cannot exceed a magnitude of 100.

4. Put it all together, to generate "simulated data" for:

a) A step response, in joint space, for one motor. For the reference trajectory, units should be in radians; aim for a step of magnitude 0.3 radians.
b) A triangle trajectory, in task space, for the end effector. This requires controlling TWO motors. Once you have the system modified for a single motor, you can select all blocks to "copy and paste" everything, for a second motor. Just be sure each motor gets its own, appropriate, reference trajectory to follow.

5. Write code to calculate the forward kinematics, given in Equations 27 and 28, and use this to produce a plot of the reference and actual trajectory for 4.b), above, in task space. The reference should look like an equilateral triangle. The actual shape will be somewhat different and is based on your simulated motor angles, which will not track the reference perfectly.

Note: You can command your desired references for 4.a) and 4.b) by modifying the "MyTrajectory" subsystem block. Double-click on the Simulink block to open the matlab function MyTrajectory.m. This block is to be used for generating a step response, and later for making the end effector track a path - just by modifying the if/else statements and code.



To Submit: That was a long document, but we'll review much of it in Lecture 2.

• To summarize your key results, turn in a single-page pdf, called Prelab1a_summary.pdf with 2 subplots and several variables:
- Put a step response in joint space, from 4.a), on the top subplot.
- Draw the reference and actual shapes, from 4.b), on the bottom subplot.
- Near the top of the page, list your values for: K_{DC}, τ, K_p, K_d, f, and A.

• Also submit all your MATLAB files: Simulink model, plus any scripts and/or functions used.

Final comments: Simulink includes a search function, to look for blocks within the Simulink Library Browser. You'll want the "Discrete Transfer Fcn", some "gain" blocks, and some summation blocks. The Simulation model given with the prelab sends some signals to the workspace in MATLAB, inside a variable called "my_data". The plotting m-file illustrates how to access these data, and you can include additional signals by double-clicking on the MUX just before the "To Workspace" block.