ECE 557: Control, Signals, and Systems Laboratory

Notes for Lab 4 (Gain Compensation and Feedback for a DC Servo)

1. Return time-domain system ID pre-lab and give some notes.
   - Notice differentiator.
     \[
     G_\omega(s) = \frac{\Omega(s)}{V_{in}(s)} = \frac{s\Theta(s)}{V_{in}(s)} = sG(s) = \frac{1}{s(0.0026s + 0.1081)} = \frac{1}{0.0026s + 0.1081}
     \]

2. Return DSP lab reports and give some notes.
   - Quantization noise/error/distortion is not directly related to sampling and its effects.
     - An LSB for the lab is
       \[
       \frac{20V}{2^{12 \text{ codes}}} \approx 4.88 \text{ mV/code} \quad \text{or} \quad \frac{20V}{2^{16 \text{ codes}}} \approx 0.31 \text{ mV/code}.
       \]
       So quantization effects are negligible for our experiments (i.e., other noise sources contribute more and we don’t care about signals that small).
     - The term resolution is normally defined so that higher is better (e.g., “dots per inch”, 1/LSB).
   - Triangle wave energy is tightly concentrated — error is less than 1.5% for a single harmonic.

Think of a triangle wave as a square wave after an antialiasing filter (i.e., an integrator).

Troublesome harmonics are already attenuated. Further filtering would yield a pure sine wave. So aliasing distortion is not much of a concern for triangle waves.

Issue observed in lab was the ZOH’s poor filtering ability. ZOH is not a good LPF approximation.

Fast sampling puts separation between aliases, and so ZOH becomes good enough (FOH better).
• Sampling has **phase** effects from **negative** aliases. For example, at 10 S/s (i.e., folds over @ 5 Hz),

\[
\sin(2\pi \times 9.5 \times t) \text{ aliases to } \sin(2\pi \times -0.5 \times t) = -\sin(2\pi \times 0.5 \times t),
\]

\[
\sin(2\pi \times 9.5 \times t) + \sin(2\pi \times 10.5 \times t) \text{ aliases to } -\sin(2\pi \times 0.5 \times t) + \sin(2\pi \times 0.5 \times t) \equiv 0.
\]

So aliasing *can* cause significant distortion!

• Pure **cosine** aliases **add** together because cos is an even function (i.e., all aliases are **positive**).
  - An **even** triangle wave at 1 Hz only has cosine harmonics at odd frequencies. When it is sampled at 4 S/s, *every* harmonic piles on top of 1 Hz and their amplitudes all **add** together.
  - A 4 S/s ideal reconstruction of such a triangle wave is a pure cosine with unity amplitude because aliasing **stretches** the available harmonics so that each sampled point is hit.

3. Page 14 of the lab text gives an example of working with *dSPACE ControlDesk* data in MATLAB.

```matlab
load filename; % Loads file structure into filename
v = filename; % Copies filename structure to v

v = getfield(v.X(1), 'Data'); % Copies X time vector into t
y1 = getfield(v.Y(1), 'Data'); % Copies Y(1) vector into y1
y2 = getfield(v.Y(2), 'Data'); % Copies Y(2) vector into y2

mean( y1( find(t >= 4) ) ) % Average of Y(1) for time after 4 s

mean( y1( t >= 4 ) ) % Equivalent averaging statement
```

The book likes to do a lot of copying. There is no need to copy the whole file structure into `v` if you don’t want to. Additionally, those `getfield` calls are not necessary because `Data` is a simple field name. So an alternative is:

```matlab
load filename; % Loads file into filename

v = filename.X(1).Data; % Copies X time vector into t
y1 = filename.Y(1).Data; % Copies Y(1) vector into y1
y2 = filename.Y(2).Data; % Copies Y(2) vector into y2

mean( y1( t >= 4 ) ) % Average of Y(1) for time after 4 s
```

In fact, we can get rid of all copying and a few lines.

```matlab
load filename; % Loads up filename
mean( filename.Y(1).Data( filename.X(1).Data >= 4 ) ) % A mean shortcut
```
4. Feedback Control for Linear-Time-Invariant (LTI) Systems

- Open-loop control is not robust, and feedback controller can be simple because of loop dynamics.
- Consider unity-gain feedback with controller $G_c$, plant $G_p$, input signal $X$, and output signal $Y$.

$$Y = G_cG_p(X - Y) = G_cG_pX - G_cG_pY \implies Y = \frac{G_cG_p}{1 + G_cG_p} X = \frac{G}{1 + G}$$ where $G \triangleq G_cG_p$

- $G(s)$ is the forward gain (which is also the loop gain for unity-gain feedback).
  * Ideally, for $Y$ to track $X$, $G$ must be large at all frequencies and $G/(1 + G)$ must be stable.
  * In reality, constraints (e.g., limited bandwidth or control energy) introduce design trade-offs (e.g., gain and bandwidth, speed and overshoot).
- System-type number is how many integrators are in forward path. Improves eventual tracking.
  * For $\theta$ constant, $\dot{\theta} = 0$. If driving $\dot{\theta}$ with $\theta$ error, then a constant $\theta$ implies no $\theta$ error.
  * For no $t_\infty$ error, controller’s imaginary-axis poles must match signal’s (“internal model”).
  - With no error, controller gets no input, and so it must naturally behave like signal.

- Stability margins
  - Poles and zeros in $G$ cause some frequencies to be delayed (i.e., phase shifted).
    * It’s possible that one frequency $\omega_c$ will be delayed by $180^\circ$ (i.e., multiplied by $-1$).
    * For that one frequency $\omega_c$, negative feedback becomes positive feedback.
    * If $|G(\omega_c)| > 1$, positive feedback is constructive (i.e., closed-loop system is unstable).
    * Consider adjacent microphone and speaker — one tone rings and grows.
      - To fix, sound engineer adjusts equalizer to attenuate speaker response at that frequency.
      - For frequency-dependent gain flexibility, we use lead–lag compensation (next week).
  - The gain margin is distance from 0 dB (i.e., where $|G| = 1$) at $\omega_c$ (i.e., where $\angle G = 180^\circ$).
    * It is how far we can increase gain before instability.
    * Must be positive for closed-loop stability.
  - The phase margin is distance from 180° where gain crosses 0 dB (i.e., where $|G| = 1$).
    * It is how much extra “delay” (phase) that can be added before instability.
    * Must be positive for closed-loop stability.
  - Wide margins ensure closed-loop stability will be robust.
    * Margins also represent distance between closed-loop poles and imaginary axis.
    * So margins have observable consequences.
    - Open-loop systems with 2 poles or less never cross 180° of phase shift/delay.
      * These systems are always stable — gain margin is $\infty$ and phase margin is always positive.

- Root locus: closed-loop pole position for every possible forward gain
  - Define $K$, $N(s)$, and $D(s)$ so that $G(s) = K N(s)/D(s)$. So open-loop (OL) zeros come from $N(s)$ and OL poles come from $D(s)$. Gain $K$ has no impact on OL dynamics.
  - The closed-loop (CL) transfer function
    $$\frac{G}{1 + G} = \frac{K N(s)}{D(s)} \text{ as } K \to \infty, 1$$
    * OL zeros and CL zeros are the same for all $K$.
    * For $K = 0$, CL poles match OL poles. As $K \to \infty$, CL poles move to OL (or CL) zeros.
      - $1/|s + a|$ $\to 0$, and so “zeros @ $\infty$” anchor “tent” for unmatched OL poles.
    - As long as CL poles are on left-hand side of complex plane, CL system is stable.
      - Use “tent” analogy with stable sliding CL poles to get snapshot of CL Bode magnitude.
    * As $K$ grows, system can react faster (“bandwidth” increases). ¿Settling time/damping?
5. Complete the Gain Compensation and Feedback for a DC Servo lab

- Implement gain compensation for position regulation of DC servo.
  - In Simulink, the Summing Junction (or sum) component is in the Math section of the library.
  - After placing it on your model, double-click on it.
  - Change the |++| to |+-| to make one of the inputs negative.
    - The | | is a placeholder for no input. It changes the spacing of the inputs.
    - The |+| means the input is positive, and the |−| means the input is negative.
- Initially, use your pre-lab controller design.
  - If you wish, wire up a simulated system for comparison. Capture its output as well.
    * You might relate this to using an estimator (a subject of ECE 650 and ECE 750).
- Using dSPACE ControlDesk, tune gain for fastest speed (i.e., highest gain) with < 5% overshoot.
  - Remember to press Enter (or ←֓) to commit numeric input changes.
  - Set plotter’s Y-axis for a Fixed scale with 0.95 minimum and 1.06 maximum to zoom in on important region of response.
  - Ideally, this type-1 system (i.e., system with one integrator) should have no steady-state error.
    * Unfortunately, nonlinearities and time variance in the system couple with quantization noise in the DAC and make the system far less than ideal. Unless gain is very high, steady-state error will be nonzero (e.g., motor stalls if it doesn’t overcome starting friction).
    * Between the end of one run and the start of the next (i.e., when the motor is not being driven), manually reset the motor to its “zero position.” This step improves system consistency (i.e., it helps with nonlinearities and time variance (e.g., “spots” of friction)).
    * Make sure gain is high enough for steady-state error to be less than 2%.
    * If you are having difficulty meeting both the 2% steady-state requirement and the 5% overshoot requirement, decrease gain substantially (i.e., critically damp or even overdamp your system with a gain lower than your pre-lab prediction).
- Save data and record working gain choice.
- AT ANY TIME, IF YOUR MOTOR STARTS CLICKING BACK AND FORTH VERY QUICKLY, STOP THE EXPERIMENT AND DISCONNECT THE MOTOR IF NECESSARY!! The high-frequency switching can cause permanent damage to the motor. (consider source of oscillations)

- Tips:
  - Do work out of directory on local hard drive — use as MATLAB working directory.
  - In Simulink, the hotkey for building a model is Ctrl-B.
  - Start dSPACE ControlDesk before doing Simulink builds.
  - In MATLAB, change Termination settings for DAC block.
    - Under Termination tab, check box to set 0 V stop value.
  - In dSPACE add a simState control.
    (i) Wire simState to 2-option radio button.
    (ii) Label one option Run and give value 2.
    (iii) Label other option Stop and give value 0.
    (iv) Set Capture Settings to automatically restart and set capture time to simulation time. Restart simulation as needed by using simState control (i.e., no need to change modes).
    (i) To stop early, change simState to Stop.
    (ii) Before restarting, tell Capture Settings to Stop and then Start.
      - This step clears the plotter data on the next run.
      - It will wait for you to change simState back to Run.
    (iii) When you’re ready to start (e.g., after changing gains), set simState to Run.