Lab 7: Controls

EECS 16B Spring 2024

Slides: http://links.eecs16b.org/lab7-slides
Lab 7 Overview

● Make your car drive straight!
  ○ Open-Loop Control
    ■ Open loop simulation with/without model mismatch
    ■ Jolt value calculations
  ○ Closed-Loop Control
    ■ Simulation + feedback gain f-value tuning
    ■ Steady-state error correction

● Turning
  ○ Derive equation
  ○ Implement turning in Arduino code
System ID → Open Loop Control

\[ v_L[i] = d_L[i+1] - d_L[i] = \theta_L u_L[i] - \beta_L \]

\[ v_R[i] = d_R[i+1] - d_R[i] = \theta_R u_R[i] - \beta_R \]

- Last week, we:
  - knew \( u \), measured \( v \)
  - calculated \( \theta_{L,R} \) and \( \beta_{L,R} \) from least squares
  - Determined operating velocity point \( v^* \)

- Opposite problem: given some target \( v \), what input \( u \) do we need?
  - Open Loop Control: solve the above equations for \( u \)

\[ u_{OL}^L = \frac{v^* + \beta_L}{\theta_L} \quad u_{OL}^R = \frac{v^* + \beta_R}{\theta_R} \]
Problems with Open Loop

\[ u_{OL}^L = \frac{v^* + \beta_L}{\theta_L} \quad u_{OL}^R = \frac{v^* + \beta_R}{\theta_R} \]

Does open loop work well for systems with disturbances? Why or why not?
Problems with Open Loop

- Will not correct for disturbance/noise (marginally stable)
- Assumes $\theta$, $\beta$ are the actual $\theta$, $\beta$ of the wheels
  - Any error will build up, preventing the car from going straight

\[
 u_{L}^{OL} = \frac{v^* + \beta_L}{\theta_L} \quad u_{R}^{OL} = \frac{v^* + \beta_R}{\theta_R}
\]
Closed Loop Intuition

- Introduce an error term that indicates the car's trajectory
  - Negative feedback allows us to correct for disturbance
- Goal: drive this delta to a zero/constant value!
Closed Loop Equations

- Introduce an error term:
  \[ \delta[i] = d_L[i] - d_R[i] \]

- The wheel/motor models become
  \[
  d_L[i+1] = d_L[i] + \theta_L u_L[i] - \beta_L - f_L \delta[i] \\
  d_R[i+1] = d_R[i] + \theta_R u_R[i] - \beta_R + f_R \delta[i]
  \]

Note: Convention is that \( f > 0 \)

\[
\begin{align*}
  u_L[i] &= u_{OL}^L - \frac{f_L}{\theta_L} \delta[i] \\
  u_R[i] &= u_{OL}^R + \frac{f_R}{\theta_R} \delta[i]
\end{align*}
\]

\[
\begin{align*}
  u_{OL}^L &= \frac{v^* + \beta_L}{\theta_L} \\
  u_{OL}^R &= \frac{v^* + \beta_R}{\theta_R}
\end{align*}
\]
Closed Loop Visualization for finding $u$

\[ d_L[i + 1] = d_L[i] + \theta u_L[i] - \beta_L \]

\[ d_R[i + 1] = d_R[i] + \theta u_R[i] - \beta_R \]
Review: Closed-Loop Control

**Open-Loop Equations**

\[ u_{OL}^L = \frac{v^* + \beta_L}{\theta_L} \]

\[ u_{OL}^R = \frac{v^* + \beta_R}{\theta_R} \]

**Closed-Loop Equations**

\[ u_L[i] = u_{OL}^L - \frac{f_L}{\theta_L} \delta[i] \]

\[ u_R[i] = u_{OL}^R + \frac{f_R}{\theta_R} \delta[i] \]

\[ \delta[i] = d_L[i] - d_R[i] \]
Closed Loop Analysis

- What’s the error after one step?
  - $\delta[i+1] = d_L[i+1] - d_R[i+1]$
  - $\delta[i+1] = v^* - f_L \delta[i] + d_L[i] - (v^* + f_R \delta[i] + d_R[i])$
  - $\delta[i+1] = \delta[i] (1 - f_L - f_R)$
    - This is of the discrete system form $\delta[i+1] = \lambda \delta[i]$, so $\lambda = (1 - f_L - f_R)$

- Stability Analysis:
  - $|\lambda| < 1$: system is stable (error decreases over time)
  - $|\lambda| > 1$: system is unstable (error increases over time)
  - $\lambda < 0$: system is oscillatory (overcorrection, f-values are too large)
Exploiting Delta for Turning

- What happens during turning?
  - One wheel moves more than the other
  - \( + \) delta \( \rightarrow d_L > d_R \rightarrow \) turn right
  - \( - \) delta \( \rightarrow d_L < d_R \rightarrow \) turn left

- Idea: Add artificial offset value to \( \delta[i] \)
  - Car “thinks” its turning
  - “corrects” it by driving \( \delta \rightarrow 0 \)
  - Naive implementation: add a constant offset?

Closed-Loop Equations

\[
\begin{align*}
  u_L[i] &= u_{OL}^L - \frac{f_L}{\theta_L} \delta[i] \\
  u_R[i] &= u_{OL}^R + \frac{f_R}{\theta_R} \delta[i] \\
  \delta[i] &= d_L[i] - d_R[i]
\end{align*}
\]
Exploiting Delta for Turning

● Naive implementation: add a constant offset?
  ○ Car tries to turn very suddenly
  ○ Large offset -> wheels leave controllable range
  ○ Isn’t really “aesthetic”
    ■ car will turn once and then drive straight rather than sweeping an angle

Closed-Loop Equations

\[
\begin{align*}
  u_L[i] & = u_{OL}^L - \frac{f_L}{\theta_L} \delta[i] \\
  u_R[i] & = u_{OL}^R + \frac{f_R}{\theta_R} \delta[i] \\
  \delta[i] & = d_L[i] - d_R[i]
\end{align*}
\]
Exploiting Delta for Turning

- Goal: gradual, circular turn
  - delta is a distance function!
  - Idea: add offset as a variable dependent on time
- In the case of a circular turn, what should $\delta[i]$ be at time $i$?
  - Function of $r$ (turn radius), $l$ (car width), $v^*$, and time $i$
  - Use arc length formula!
  - Relate distance to velocity and time
  - Check your derivation with staff
Implementing turning.ino

- Code the function for $\delta[i]$ you found
  - Control loop and the data collection have different periods
  - Account for different sampling rates of data collection and controller
- (Optionally) apply a straight correction for any lingering turning due to mechanical errors
Mic Board Verification

- As a final step, verify that your biasing circuits and front-end circuitry still work as expected.
  - we will be using the mic board next week for the SVD/PCA lab!
- You will run a quick Arduino + Python program to see if the Arduino is successfully reading data from the mic board
  - You should see the Python script create a graph that displays the waveform
  - The window may appear frozen. If that happens, try dragging it around and it should work
Lab 7 Checkoff

- Our definition of “straight” is based on the floor tiling in Cory:
  - Inside Cory 125 (1x4 tiles)
  - Outside Cory 125 (3 x 11 on black)
  - Side entrance hallway, from the pink line to the red line (2 x 7 tiles)
Common Bugs

● Double check all equations!
  ○ For both closed loop and turning, one term is positive and one term is negative

● If a wheel jolts and stops moving:
  1. Double check that all pins (motor and encoder) you are using are correctly defined in the Arduino code
  2. Rerun encoder tests from System ID to make sure encoders are still working

● If motors are no longer running, rerun the encoder tests
  ○ If you suspect your Arduino pin is broken, try another pin
  ○ DO NOT USE PIN 9
Tips and Common Errors

- Don’t guess f-values, this will take you forever!
  - Make educated decisions on how to change your f values from iterations of testing.
  - If your car is turning left, how should you change $f_L$ and $f_R$ to fix it?
- Data is stored in RAM, just like last lab, so make sure you keep the 9V plugged in when you plug the USB into your computer.
- You can manipulate the turn radius and run times (in ms!) of the turning sequence.
- Ensure you’ve replaced $v^*$ with $v^*/m$ ONLY in delta_reference function.
- Do not cut the power supply cable and cause a firework. please.
Important Forms/Links

- Help request form: https://eecs16b.org/lab-help
- Checkoff request form: https://eecs16b.org/lab-checkoff
- Slides: http://links.eecs16b.org/lab7-slides
- Anon Feedback: https://eecs16b.org/lab-anon-feedback
- Lab Grades error: https://links.eecs16b.org/lab-checkoff-error