A Simple Introduction to Embedded Control Systems (PID Control)

Control System
- Control physical system's output
  - By setting physical system's input
- Tracking
  - E.g.
    - Cruise control
    - Thermostat control
    - Disk drive control
    - Aircraft altitude control
    - Chemical processes
- Difficulty due to
  - Disturbance: wind, road, tire, brake; opening/closing door...
  - Human interface: feel good, feel right...

Acknowledgements
- The material in this lecture is adapted from:
  - T. Wescott, “PID Without a PhD”

Tracking
Open-Loop Control Systems

- **Plant**
  - Physical system to be controlled
  - Car, plane, disk, heater...
- **Actuator**
  - Device to control the plant
  - Throttle, wing flap, disk motor, ...
- **Controller**
  - Designed product to control the plant

Output
- The aspect of the physical system we are interested in
  - Speed, disk location, temperature
- Reference
  - The value we want to see at output
  - Desired speed, desired location, desired temperature
- Disturbance
  - Uncontrollable input to the plant imposed by environment
  - Wind, bumping the disk drive, door opening

Other Characteristics of open loop

- Feed-forward control
- Delay in actual change of the output
- Controller doesn’t know how well thing goes
- Simple
- Best use for predictable systems

Closed Loop Control Systems

- **Sensor**
  - Measure the plant output
- **Error detector**
  - Detect Error
- **Feedback control systems**
- **Minimize tracking error**
Designing Open Loop Control System

- Develop a model of the plant
- Develop a controller
- Analyze the controller
- Consider Disturbance
- Determine Performance
- Example: Open Loop Cruise Control System

Model of the Plant

- May not be necessary
  - Can be done through experimenting and tuning
- But,
  - Can make it easier to design
  - May be useful for deriving the controller
- Example: throttle that goes from 0 to 45 degree
  - On flat surface at 50 mph, open the throttle to 40 degrees
  - Wait 1 "time unit"
  - Measure the speed, let's say 55 mph
  - Then the following equation satisfies the above scenario
    - \( v_{t+1} = 0.7v_t + 0.5u_t \)
    - 55 = 0.7*50+0.5*40
  - If the equation holds for all other scenarios
    - Then we have a model of the plant

Designing the Controller

- Assuming we want to use a simple linear function
  - \( u_t = F(r_t) = P \cdot r_t \)
  - \( r_t \) is the desired speed
- Linear proportional controller
  - \( v_{t+1} = 0.7v_t + 0.5u_t = 0.7v_t + 0.5Pr_t \)
  - Let \( v_{t+1} = v_t \) at steady state = \( v_{ss} \)
  - \( v_{ss} = 0.7v_{ss} + 0.5Pr_t \)
  - At steady state, we want \( v_{ss} = r_t \)
  - \( P = 0.6 \)
    - i.e. \( u_t = 0.6r_t \)

Analyzing the Controller

- Let \( v_0 = 20 \text{ mph}, \ r_0 = 50 \text{ mph} \)
- \( v_{t+1} = 0.7v_t + 0.5(0.6)r_t \)
  - \( = 0.7v_t + 0.3*50 = 0.7v_t + 15 \)
  - Throttle position is 0.6*50=30 degrees
Considering the Disturbance

- Assume road grade can affect the speed
  - From -5 mph to +5 mph
  - $v_{t+1} = 0.7v_t + 10$
  - $v_{t+1} = 0.7v_t + 20$

<table>
<thead>
<tr>
<th>Time (t)</th>
<th>$v_t$</th>
<th>$v_{t+}$ for $w = 15$</th>
<th>$v_{t+}$ for $w = -15$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20.00</td>
<td>30.00</td>
<td>10.00</td>
</tr>
<tr>
<td>1</td>
<td>26.00</td>
<td>36.00</td>
<td>16.00</td>
</tr>
<tr>
<td>2</td>
<td>32.00</td>
<td>42.00</td>
<td>22.00</td>
</tr>
<tr>
<td>3</td>
<td>38.00</td>
<td>48.00</td>
<td>28.00</td>
</tr>
<tr>
<td>4</td>
<td>44.00</td>
<td>54.00</td>
<td>34.00</td>
</tr>
<tr>
<td>5</td>
<td>44.00</td>
<td>54.00</td>
<td>34.00</td>
</tr>
<tr>
<td>6</td>
<td>46.00</td>
<td>58.00</td>
<td>36.00</td>
</tr>
<tr>
<td>7</td>
<td>47.53</td>
<td>59.24</td>
<td>37.52</td>
</tr>
<tr>
<td>8</td>
<td>48.57</td>
<td>61.56</td>
<td>39.02</td>
</tr>
<tr>
<td>9</td>
<td>49.78</td>
<td>63.90</td>
<td>40.78</td>
</tr>
<tr>
<td>10</td>
<td>50.43</td>
<td>66.05</td>
<td>42.43</td>
</tr>
<tr>
<td>11</td>
<td>51.13</td>
<td>68.24</td>
<td>44.13</td>
</tr>
<tr>
<td>12</td>
<td>51.68</td>
<td>70.52</td>
<td>45.68</td>
</tr>
</tbody>
</table>

Determining Performance

- $v_{t+1} = 0.7v_t + 0.5P(r_t - w_t)$
- $v_1 = 0.7v_0 + 0.5P(r_0 - w_0)$
- $v_2 = 0.7(0.7v_0 + 0.5P(r_0 - w_0)) + 0.5P(r_0 - w_0)$
- $v_t = 0.7v_{t-1} + (0.7 + 1.0)(0.7 + 1.0)(0.7 + 1.0)w_t$
- $v(t) = 0.7v_{t-1} + (0.7 + 1.0)w_t$
- $v_t = 0.7v_{t-1} + (0.7 + 1.0)w_t$

Coefficient of $v_t$ determines rate of decay of $v_0$
- Bigger coefficient will result in slower response
- $> 1$ or $< -1$, $v_t$ will grow without bound
- $< 0$, $v_t$ will oscillate

Designing Closed Loop Control System

- $u_t = P * (r_t - v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P(r_t - v_t) - w_t$
- $0.7 - 0.5P \geq 0$
- $P \leq 1.4$

Stability

- Stability constraint (i.e. convergence) requires:
  - $0.7 - 0.5P < 1$
  - $-1 \leq -0.7 - 0.5P \leq 1$
  - $0.6 - P \leq 3.4$

To avoid oscillation:
- $(0.7 - 5P) \geq 0$
- $P \leq 1.4$
Reducing effect of $v_0$

- $u_t = P \cdot (r_t-v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P(r_t-v_t)-w_t$
- $v_t = (0.7-0.5P)v_{t-1} + (0.7-0.5P)^2v_{t-2} + \ldots + 0.7-0.5P + 1.0)(0.5P r_{t0} - w_0)$
- To reduce the effect of initial condition
  - $0.7-0.5P$ as small as possible
  - $P = 1.4$

Avoid Oscillation

- $u_t = P \cdot (r_t-v_t)$
- $v_{t+1} = 0.7v_t + 0.5u_t - w_t = 0.7v_t + 0.5P(r_t-v_t)-w_t$
- $v_t = (0.7-0.5P)v_{t-1} + (0.7-0.5P)^2v_{t-2} + \ldots + 0.7-0.5P + 1.0)(0.5P r_{t0} - w_0)$
- To avoid oscillation
  - $0.7-0.5P \geq 0$
  - $P \leq 1.4$

Perfect Tracking

- $u_t = P \cdot (r_t-v_t)$
- $v_{ss} = (0.7-0.5P)v_{ss} + 0.5P r_0 - w_0$
- $(1-0.7+0.5P)v_{ss} = 0.5P r_0 - w_0$
- $v_{ss} = (0.5P/(0.3+0.5P)) \cdot r_0 - (1.0/(0.3+0.5P)) \cdot w_0$
- To make $v_{ss}$ as close to $r_0$ as possible
  - $P$ should be as large as possible

Closed-Loop Design

- $u_t = P \cdot (r_t-v_t)$
- Finally, setting $P = 3.3$
  - Stable, track well, some oscillation
    $u_t = 3.3 \cdot (r_t-v_t)$
Analyze the controller

- \( v_0 = 20 \text{ mph}, r_0 = 50 \text{ mph}, w = 0 \)
- \( v_{t+1} = 0.7v_t + 0.5P(r_t - v_t) - w \)
  \( = 0.7v_t + 0.5 \times 3.3(50 - v_t) \)
- \( u_t = P(r_t - v_t) \)
  \( = 3.3(50 - v_t) \)

- But valid \( u_t \) ranges from 0-45
- Controller saturates

Analyzing the Controller

- Set \( P = 1.0 \) to void oscillation
  - Terrible SS performance
Minimize the effect of disturbance

- \( v_{t+1} = 0.7v_t + 0.5 \times 3.3 (r_t - v_t) - w \)
  - \( w = -5 \) or \( +5 \)

- 39.74
  - Close to 42.31
  - Better than
    - 33
    - 66

- Cost
  - SS error
  - Oscillation

General Control System

- Objective
  - Causing output to track a reference even in the presence of
    - Measurement noise
    - Model error
    - Disturbances

- Metrics
  - Stability
    - Output remains bounded
  - Performance
    - How well an output tracks the reference
    - Disturbance rejection
    - Robustness
      - Ability to tolerate modeling error of the plant

Performance (generally speaking)

- Rise time
  - Time it takes from 10% to 90%
- Peak time
- Overshoot
  - Percentage by which Peak exceed final value
- Settling time
  - Time it takes to reach 1% of final value

Plant modeling is difficult

- May need to be done first
- Plant is usually on continuous time
  - Not discrete time
    - E.g. car speed continuously react to throttle position, not at discrete interval
  - Sampling period must be chosen carefully
    - To make sure “nothing interesting” happen in between
    - I.e. small enough
- Plant is usually non-linear
  - E.g. shock absorber response may need to be 8th order differential
- Iterative development of the plant model and controller
  - Have a plant model that is “good enough”
Controller Design: P

- Proportional controller
  - A controller that multiplies the tracking error by a constant
    - \( u_t = P \cdot (r_t - v_t) \)
  - Closed loop model with a linear plant
  - E.g. \( v_{t+1} = (0.7 - 0.5P) v_t + 0.5P r_t - w_t \)
- \( P \) affects
  - Transient response
    - Stability, oscillation
  - Steady state tracking
    - As large as possible
  - Disturbance rejection
    - As large as possible

Controller Design: PD

- Proportional and Derivative control
  - \( u_t = P \cdot (r_t - v_t) + D \cdot ((r_t - v_t) - (r_{t-1} - v_{t-1})) \)
- Consider the size of the error over time
- Intuitively
  - Want to “push” more if the error is not reducing fast enough
  - Want to “push” less if the error is reducing really fast

PD Controller

- Need to keep track of error derivative
- E.g. Cruise controller example
  - \( v_{t+1} = 0.7v_t + 0.5u_t - w_t \)
  - Let \( u_t = P \cdot e_t + D \cdot (e_t - e_{t-1}) \), \( e_t = r_t - v_t \)
  - \( v_{t+1} = (0.7 - 0.5(P+D))v_t + 0.5Dv_{t-1} + 0.5(P+D)r_t - 0.5Dv_{t-1} - w_t \)
  - Assume reference input and disturbance are constant, the steady-state speed is
    - \( V_{ss} = (0.5P/(1-0.7+0.5P)) \cdot r \)
    - Does not depend on \( D \)!!!
- \( P \) can be set for best tracking and disturbance control
- Then \( D \) set to control oscillation/overshoot/rate of convergence

PD Control Example
### PI Control

- Proportional plus integral control
  - $u_t = P \cdot e_t + I \cdot (e_0 + e_1 + \ldots + e_t)$
- Sum up error over time
  - Ensure reaching desired output, eventually
  - $v_u$ will not be reached until $e_u = 0$
- Use $P$ to control disturbance
- Use $I$ to ensure steady state convergence and convergence rate

### PID Controller

- Combine Proportional, integral, and derivative control
  - $u_t = P \cdot e_t + I \cdot (e_0 + e_1 + \ldots + e_t) + D \cdot (e_t - e_{t-1})$
- Available off-the-shelf

### Block Diagram View of PID Controller

### Software Coding

- Main function loops forever, during each iteration
  - Read plant output sensor
    - May require A2D
  - Read current desired reference input
  - Call PidUpdate, to determine actuator value
  - Set actuator value
    - May require D2A
Software Coding (continued)

- $P_{gain}$, $D_{gain}$, $I_{gain}$ are constants
- $\text{sensor\_value\_previous}$
  - For $D$ control
- $\text{error\_sum}$
  - For $I$ control

typedef struct PID_DATA {
  double $P_{gain}$, $D_{gain}$, $I_{gain}$;
  double $\text{sensor\_value\_previous}$; // find the derivative
da double $\text{error\_sum}$; // cumulative error
} PID_DATA

Computation

- $u_t = P_{gain}e_t + I(e_0 + e_1 + ... + e_t) + D(e_t - e_{t-1})$

```c
double PIDUpdate(PID_DATA *pid_data, double sensor_value, double reference_value) {
  double error, derivative, integral;
  error = reference_value - sensor_value;
  derivative = pid_data->$D_{gain}$ * error; // proportional term
  integral += pid_data->$I_{gain}$ * error; // current = cumulative
  pid_data->$P_{gain}$ += derivative;
  pid_data->$I_{gain}$ += integral;
  pid_data->$D_{gain}$ += derivative;
  return (error + derivative + integral);
}
```

PID tuning

- Analytically deriving $P$, $I$, $D$ may not be possible
  - E.g. plant not is not available, or too costly to obtain
- Ad hoc method for getting “reasonable” $P$, $I$, $D$
  - Start with a small $P$, $I$=0
  - Increase $D$, until seeing oscillation
    - Reduce $D$ a bit
  - Increase $P$, until seeing oscillation
    - Reduce $P$ a bit
  - Increase $I$, until seeing oscillation
  - Iterate until can change anything without excessive oscillation

Excellent Reference for PID Control

- PID Without a PhD by Tim Wescott
  0feat3.htm
Practical Issues with Computer-Based Control

- Quantization
- Overflow
- Aliasing
- Computation Delay

Quantization & Overflow

- Quantization
  - E.g. can't store 0.36 as 4-bit fractional number
  - Can only store 0.75, 0.50, 0.25, 0.00, -0.25, -0.50, -0.75, -1.00
  - Choose 0.25
  - Results in quantization error of 0.11
- Sources of quantization error
  - Operations, e.g. 0.50 * 0.25 = 0.125
  - Can use more bits until input/output to the environment/memory
  - ADC converters
- Overflow
  - Can't store 0.75 + 0.50 = 1.25 as 4-bit fractional number
- Solutions:
  - Use fix-point representation/operations carefully
  - Time-consuming
  - Use floating-point co-processor
  - Costly

Aliasing

- Quantization/overflow
  - Due to discrete nature of computer data
- Aliasing
  - Due to discrete nature of sampling

Aliasing Example

- Sampling at 2.5 Hz, period of 0.4, the following are indistinguishable
  - y(t) = 1.0 * sin(6πt), frequency 3 Hz
  - y(t) = 1.0 * sin(πt), frequency of 0.5 Hz
- In fact, with sampling frequency of 2.5 Hz
  - Can only correctly sample signal below Nyquist frequency 2.5/2 = 1.25 Hz
Computation Delay

- Inherent delay in processing
  - Actuation occurs later than expected
- Need to characterize implementation delay to make sure it is negligible
- Hardware delay is usually easy to characterize
  - Synchronous design
- Software delay is harder to predict
  - Should organize code carefully so delay is predictable and minimized
  - Write software with predictable timing behavior (be like hardware)
    - Time Trigger Architecture
    - Synchronous Software Language and/or RTOS