Control Systems

- **def**: collection of mechanical and electrical devices that controls the operation of a *physical plant*
- requires $\geq 1$ output devices (open loop) and $\geq 1$ input devices (closed loop)
- originally: elec. circuits & mech. devices
  now: µP-based with control algorithm in software
Processor-based Control Systems

"Control Theory"†

- various models
  - finite state machines
  - fuzzy logic / neural networks
  - linear systems
  - PID (classical)

- our main interest here is with PID


Block Diagram

- text...
Components - 1

1. **Real state variables,** \( X(t) \) – properties of the physical plant being controlled
   - requires sensor(s) and state estimator to produce *estimated state variables*, \( X'(t) \)

2. **Desired state variables,** \( X^*(t) \) – what the system seeks for \( X(t) \)

Components - 2

3. **Control outputs,** \( U(t) \) – output commands to devices that affect the physical plant
   - typically control actuators which produce driving forces, \( V(t) \)

4. **Control algorithm** – generates \( U(t) \)
   - based on error: \( E(t) = X^*(t) - X'(t) \)
   - goal of a control system: minimize \( E(t) \)
Control System Effectiveness

- determined by 3 properties:
  1. **steady-state error**: average value of $E(t)$
  2. **transient response**: how long it takes system to reach 99% of final output after $X'$ or $X^*$ is changed
  3. **stability**: if a steady state is reached after a change in $X'$ or $X^*$
     - i.e., no oscillations

Open-loop Systems

- no feedback, \( \therefore \) no state estimator

- examples:
  - toaster – fixed amount of time
  - traffic light – ditto. Must be closed-loop if goal is to maximize traffic flow!
  - OL stepper motor controller – no encoder
Closed-loop Systems

- includes feedback/state estimator
- examples:
  - robot arm incremental control – $U(t)$ is incremented, decremented, or left alone
  - temperature control – add heat or not (bang bang), assumes passive heat loss
    - can add hysteresis with 2 set points: $T_{Hr}$, $T_{L}$

Example Closed-loop System†: a "bang-bang" temperature controller

† from "Embedded Microcomputer Systems", J. Valvano, 2007
PID Systems

- from linear control theory
- faster, more accurate, better control

\[ U(t) = k_p \cdot E(t) + k_i \cdot \int_0^t E(\tau) d\tau + k_d \cdot \frac{dE(t)}{dt} \]

- the k's are design parameters, aka gains
  - determined using control theory and Laplace / Laplace\(^{-1}\) transforms
  - the I term addresses steady state error, the D term addresses transient error

PID Controller Block Diagram

- ideal / parallel form:

  \[ E(t) = SP - PV \quad (\text{desired} - \text{current}) \]
  \[ U(t) = k_p \cdot E(t) + k_i \cdot \int_0^t E(\tau) d\tau + k_d \cdot \frac{dE(t)}{dt} \]
Types of PID Controllers

- consider values of the gains:

<table>
<thead>
<tr>
<th>Type</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Digital PID Implementation

- break $U(t)$ into separate terms & convert to \textit{discrete time}
- \textit{discrete time}: fixed period, like DSP ($\Delta t$)

\[ U(t) = P(t) + I(t) + D(t) \]

\[ P(t) = k_p \cdot E(t) \quad \rightarrow \quad P(n) = k_p \cdot E(n) \]
Digital PID Implementation

\[ I(t) = k_i \int_0^t E(\tau) \, d\tau \rightarrow I(n) = k_i \sum_{i=1}^\infty E(i) \cdot \Delta t \]

\[ D(t) = k_d \frac{dE(t)}{dt} \rightarrow D(n) = k_d \frac{E(n) - E(n-1)}{\Delta t} \]

- noise-prone, better to use average of 2 derivatives over different time spans

---

Digital PID Implementation

- combining, we now have:

\[ U(n) = k_p \cdot E(n) + k_i \cdot \sum_{i=1}^\infty E(i) \cdot \Delta t + k_d \cdot \left[ \frac{E(n) - E(n-1)}{\Delta t} \right] \]

- \( \Delta t \) can be factored into \( k_i \) & \( k_d \), so:

\[ U(n) = k_p \cdot E(n) + k'_i \cdot \sum E(n) + k'_d \cdot \left[ E(n) - E(n-1) \right] \]
Digital PID Implementation

\[ U(n) = k_p \cdot E(n) + k_i \cdot \sum E(n) + k_d \cdot [E(n) - E(n-1)] \]

- this can easily be implemented in code based on the 3 gains & 2 "history" (state) variables:
  1. accumulated error ("istate"): \( I(n) = I(n-1) + E(n) \)
  2. previous error ("dstate"): \( E(n-1) \)

- however: this requires precision in acquiring \( y(t) \) and processing \( u(n) \) at exact \( \Delta t \) intervals
  - hence: our task scheduler!

PID Implementation: Data structure

```c
/*
 * Abstraction of a PID controller as a C structure
 */
typedef struct {
    // gain factors
    float pGain, // proportional
    iGain, // integral
    dGain; // derivative
    // allowable range of integrator state values to prevent windup
    float iMax, // max allowed
    iMin; // min allowed
    // running state variables
    float iState; // integrator state
    float dState; // derivative state
} Spid;
```
Processor-based Control Systems

PID Implementation: "update" function

```c
/* implementation of discretized version of parallel form of PID controller */
float PID_Update(Spid *pid, float current, float desired)
{
    float error, pTerm, iTerm, dTerm;
    // calculate error
    error = desired - current;
    // calculate the proportional term
    pTerm = pid->pGain * error;
    // calculate integral of error as running total
    pid->iState += error;
    // perform limiting of integral state to prevent windup
    if (pid->iState < pid->iMin) pid->iState = pid->iMin;
    if (pid->iState > pid->iMax) pid->iState = pid->iMax;
    // calculate the integral term
    iTerm = pid->iGain * pid->iState;
    // calculate the derivative term from delta-error
    dTerm = pid->dGain * (error - pid->dState);
    // remember current value for next update
    pid->dState = error;
    // calculate & return PID result
    return pTerm + iTerm + dTerm;
}
```

References

- PID process control, a “Cruise Control” example
  - http://www.codeproject.com/Articles/36459/PID-process-control-a-Cruise-Control-example

- PID Theory
  - http://pcbheaven.com/wikipages/PID_Theory

- Understanding PID Control (toilet ex.)

- PID Controller