Lecture 8.2 - (PID Implementation)

Today:

- More details on integral action and anti-windup
- Implementation of PID control: analog + digital
- Realizing & implementing more general controllers

References

- AM08, Section 11.4, 11.5
- Astrom & Hagglund, Adv PID Control, Ch 3

Outline

I. Integral control & anti-windup compensation
   - A. Properties of integral action
   - B. Windup
   - C. Anti-windup

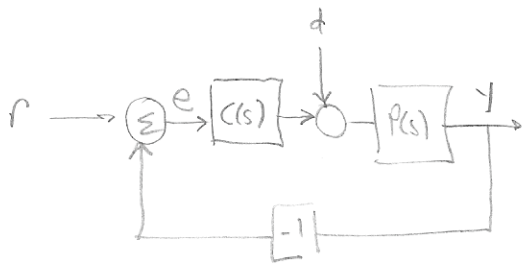
II. Implementing PID controllers
   - A. Analog implementation using op-amps
   - B. Digital implementation

III. Realizing & implementing frequency domain controllers

$$u = C(s)e \xrightarrow{\text{tf2ss}} \dot{x}_c = A_c x_c + B_c e \xrightarrow{\text{c2d}} x_{k+1} = \bar{A}x_k + \bar{B}e_k$$
$$u = C_c x_c + D_c e \qquad u_k = \bar{C}x_k + \bar{D}e_k$$

RCF: $\left[\begin{array}{cccc|c} -a_1 & -a_2 & \cdots & -a_n & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & & & & \vdots \\ \vdots & & 1 & 0 & 0 \\ \hline b_1 & b_2 & \cdots & b_m & 0 \end{array}\right]$ — c2d → sparrow/falcon

Properties of Integral Action



$$u = \left[\frac{k_i}{s} + \tilde{C}(s)\right] e$$

$$G_{er}(0) = \frac{1}{1+PC} = 0$$

$$G_{ed}(0) = \frac{P}{1+PC} = 0$$

Properties:

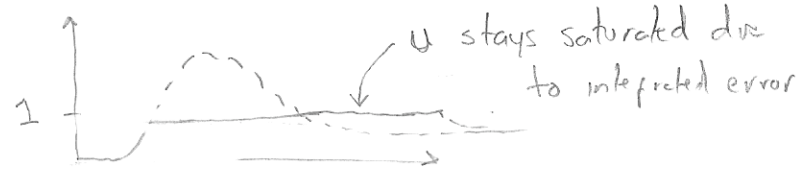
- $u(\infty)$ provides steady state input required to cancel constant reference inputs and constant disturbances
- Compare w/ state space control

$$u = \underbrace{-K(x - x_d)}_{Ke_x} + \underbrace{k_r r}_{\text{Nominal input required to hold output at desired value}} \qquad k_r = -1/(CA^{-1}B)$$

$$u = \tilde{C}(s) e_y + \underbrace{\int_0^t e_y \, dt}_{\text{Steady state input required to hold output}}$$

- Integral feedback automatically computes $k_r r$ w/out knowledge of A, B, C $\Rightarrow$ "self calibrating"

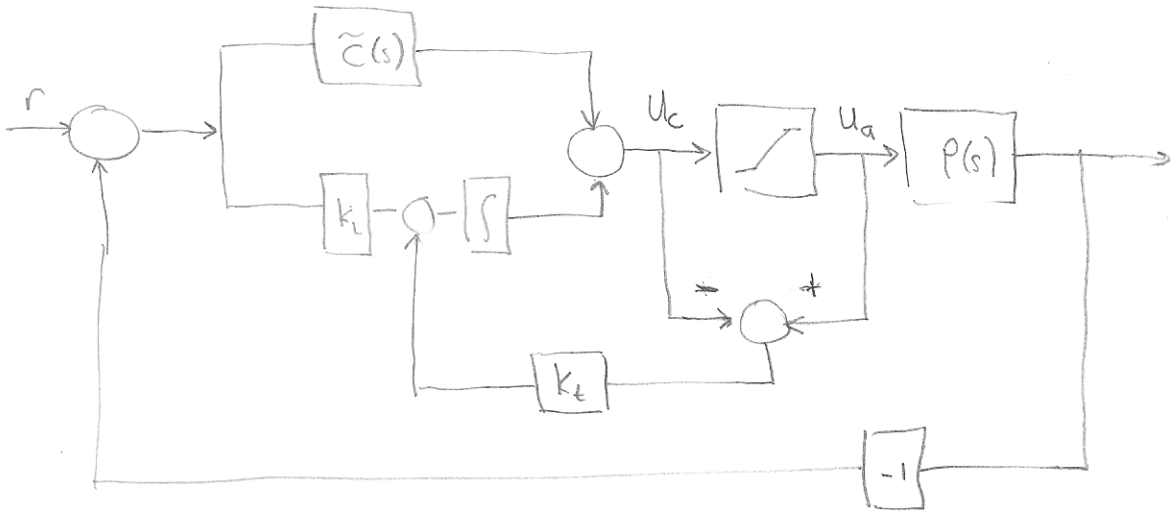
Problem: Integrator windup



$$u = k_p e + k_i \int e \, dt + k_d \dot{e}$$

- Note that system is open loop while in saturation
- Can also get instabilities (oscillations)

Anti-windup compensation

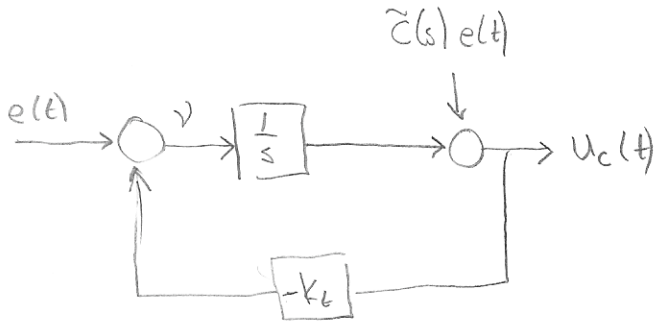


Key idea: "turn off" integration of error when input is saturated

① $|u_c| < u_{lim}$: $u_a = \left[k_i/s + \tilde{C}(s)\right] e$

② $|u_c| \geq u_{lim}$: $u_a = u_{lim} \Rightarrow$ process is open loop

Redraw integrator as



$e(t)$ is independent signal (since process is open loop)

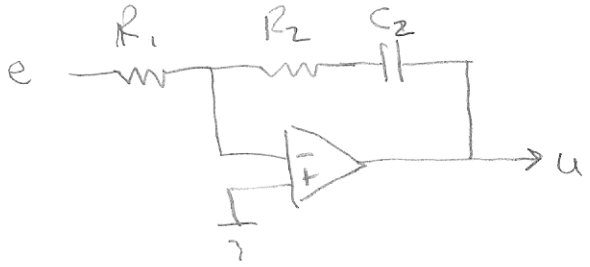
v = integrator input

$H_{ve} = \dfrac{s}{s + k_t} \Rightarrow v \to 0$ as $t \to \infty$

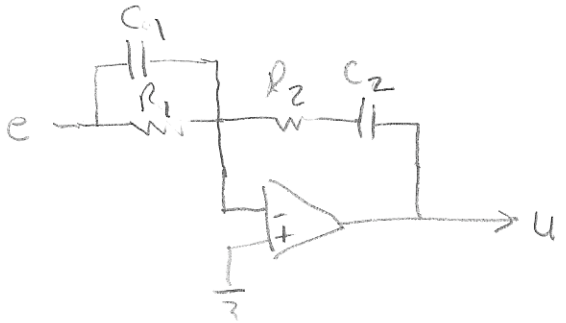
k_t sets rate of convergence

Rule of thumb: set $k_t < \frac{1}{T_i}$ = integrator time constant
$\Rightarrow$ integrator reset "slower" than integration (higher OK, but watch measurement noise ...)

Implementing PID control w/ op amps

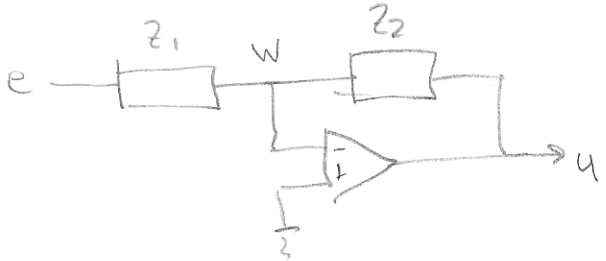


PI control



PID control

To analyze, consider general circuit



Steady state: $\frac{e}{Z_1} = \frac{-u}{Z_2}$

$$\Rightarrow \frac{u}{e} = -\frac{Z_2}{Z_1}$$

PI controller: $Z_1 = R_1$, $Z_2 = R_2 + \frac{1}{C_2 s}$

$$H_{eu}(s) = -\frac{R_2 + \frac{1}{C_2 s}}{R_1} = -\frac{R_2 C_2 s + 1}{R_1 C_2 s}$$

$$H_{PI}(s) = k_p + \frac{k_i}{s} = \frac{k_p s + k_i}{s}$$

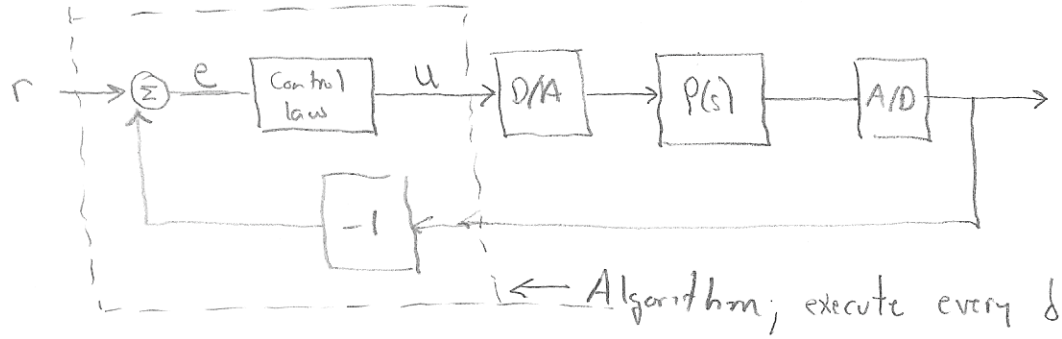
$$k_p = \frac{R_2}{R_1}, \quad k_i = \frac{1}{R_1 C_2}$$

PID controller: $Z_1 = \frac{R_1}{1 + R_1 C_1 s}$, $Z_2 = R_2 + \frac{1}{C_2 s}$, ...

Remarks

1. C_2 gives integrator "state"
2. Can augment to give anti-windup compensation
3. Easy to combine w/ filters to get high frequency rolloff

Implementing PID on a computer



High level: $x_{k+1} = f(x_k, r_k, y_k)$ $\quad u_k = h(x_k, r_k, y_k)$

1. Wait for clock t_k
2. Read inputs: $r(t_k), y(t_k)$
3. Compute control: u_k
4. Send output: $u(t_k) = u_k$
5. Update controller state: x_k

Controller state: $x_k = (e_{i,k-1}, I_k)$

$$x_{k+1} = \begin{bmatrix} r(t_k) - y(t_k) \\ I_k + \underbrace{(r(t_k) - y(t_k))\delta}_{\text{approximates } \int_0^{t_k} e(t)\,dt} \end{bmatrix}$$

$$u_k = k_p(r(t_k) - y(t_k)) + k_i I_k + k_d \underbrace{\frac{(r(t_k) - y(t_k)) - e_{k-1}}{\tau}}_{\text{Approximates } \dot{e}}$$

Remarks

1. Assumes τ is small relative to system speed. Typically require $\tau > 5/\omega_{gc} \Rightarrow$ delay in computation doesn't affect loop
2. See text for more detailed code (including anti-windup)