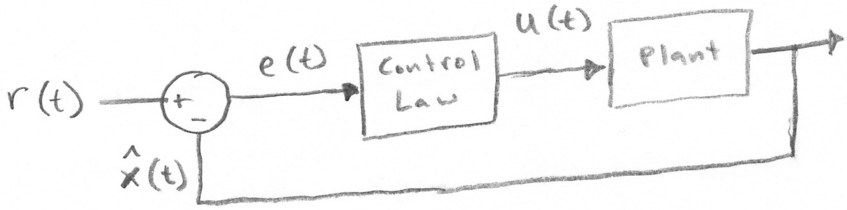
# Balancebot PID Control

## Basic feedback control




$r(t) \Rightarrow$ reference, desired state

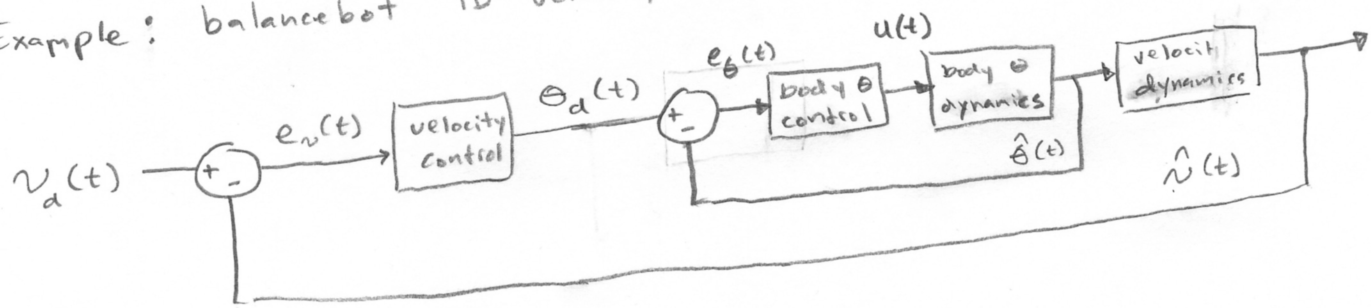
$\hat{x}(t) \Rightarrow$ sensor measurement of state

$e(t) \Rightarrow$ error signal $\quad e(t) = (r(t) - \hat{x}(t))$

$u(t) \Rightarrow$ control signal to plant

Control feedback loops can be nested

Example: balancebot 1D velocity / stabilization → motor signal




What do we use for control law?

PID - Proportional, Integral, Derivative

why? frustratingly simple,
usually works (with careful tuning)

PID control law:

$$u(t) = K_p e(t) + K_I \int_0^t e(t)\,dt + K_D \frac{de(t)}{dt}$$

for a discrete system

$$u(k) = u_P(k) + u_I(k) + u_D(k)$$

$$u_P(k) = K_P\left(x_r(k) - \hat{x}(k)\right)$$

Drives state towards the reference value

$$u_I(k) = u_I(k-1) + K_I\left(x_r(k) - \hat{x}(k)\right)$$

Drive accumulated error to zero to remove offset

$$u_D(k) = K_D\left(\dot{x}_r(k) - \hat{\dot{x}}(k)\right)$$

if $\dot{x}_r = 0$ apply damping to reduce overshoot. otherwise drive @ prescribed velocity

for Integral term:

avoid integral windup!

1. Initialize $u_I(0) = 0$
2. saturate $u_I$ to reasonable bounds
3. Reset $u_I(k) = 0$ each time a <u>new</u> reference state is commanded

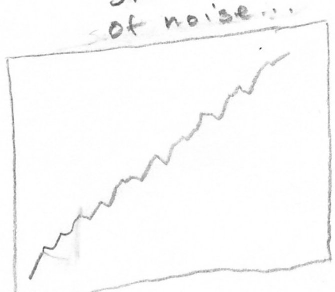
We can also have linear & constant bias states

so the complete control law

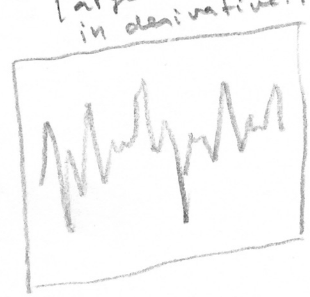
$$u(k) = u_P(k) + u_I(k) + u_D(k) + u_B(x_r) + u_c$$

Also consider noise:

Small amount of noise...



large noise in derivative...



# PID tuning by hand

1. Ensure gains are sensible, i.e. $K_p$ has physical meaning mapping error to motor command!

2. Start with $K_p$, turn up till it oscillates and dial it back. Make sure you measure your bias angle

3. for $\theta$ controller, response should be "stiff" it should "fight back"

4. Damp overshoot with $K_D$, and remember $K_D$ is divided by $\Delta t$. If you get more oscillations with increasing $K_D$, use a better filter

5. $K_I$ only to remove steady state error last. remember $K_I$ is multiplied by $\Delta t$

6. A body angle controller alone will not make the balancebot stable.

Finally if building a model

$$u(t) = K_p e(t) + K_I \int_0^t e(t)\,dt + K_D \frac{de(t)}{dt}$$

$\downarrow$ Laplace transform

$$U(s) = K_p + \frac{K_I}{s} + K_D s$$

$$\downarrow$$

$$= \frac{K_D s^2 + K_p s + K_I}{s}$$

# Balance bot Model

$\frac{\Theta}{\tau}$ & $\frac{\Phi}{\Theta}$ are 2nd order systems w/o damping

Second order system: $$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$\omega_n$ : natural frequency

Zeta: $\zeta$ : damping ratio

refer to document on Google drive