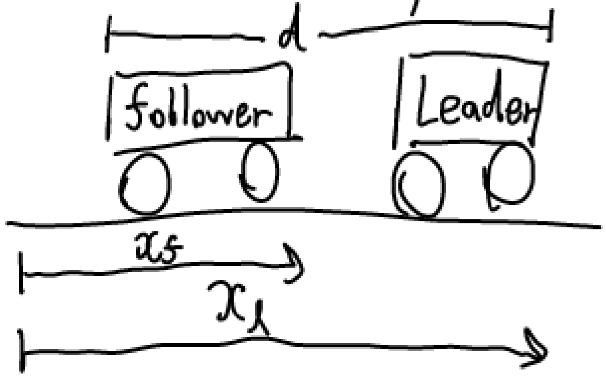
SE380

## 1.2 Control Engineering

(1.1 – 1.3 in course notes) especially

**Ex.** Automated Highway




**Assumptions**

- Cars move in a line
- thru software or the follower we can assign its speed

$$\frac{dx_f}{dt} = u$$

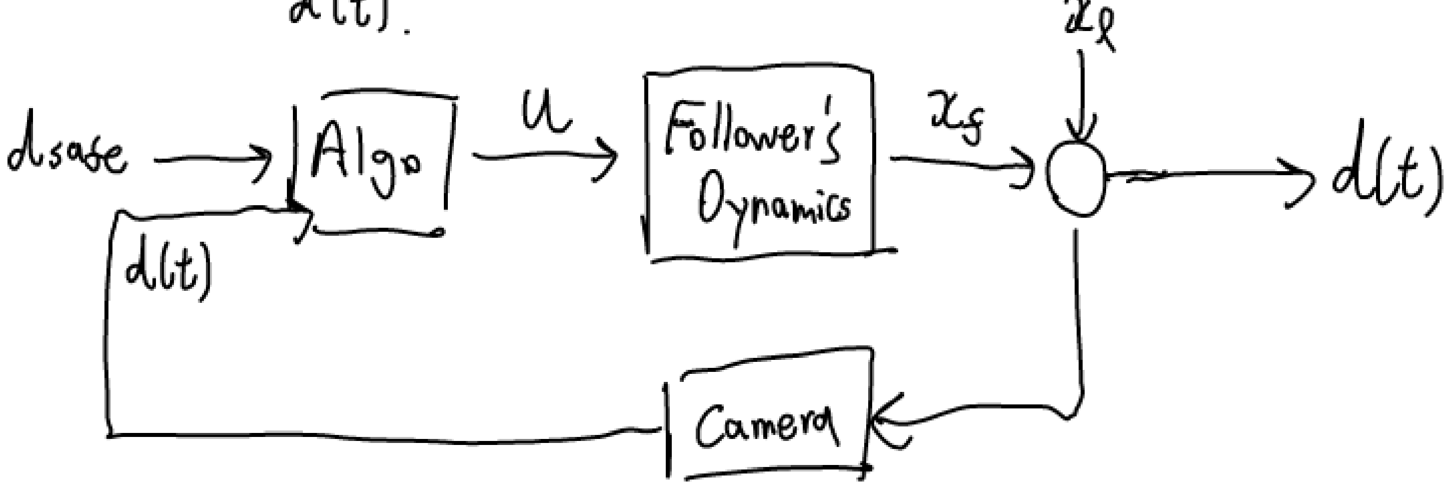
- leader moves @ a constant unknown speed

**Spec:** Assign a velocity to follower in order to maintain a safe distance from leader, i.e. $d \doteq d_{safe}$

**Option #1:** "Open-loop" don't put any sensors on follower (save $$)

Can we meet the specs? Not realistically.


**Opt. #2:** "Closed-loop" Equip follower w/ a camera to measure $d(t)$.




The simplest algo. is proportional error feedback.

$$u(t) = -K_p\left(d_{safe} - d(t)\right), \quad K_p > 0$$

$$\frac{dx_s}{dt} = -K_p(d_{safe} - d(t))$$

---

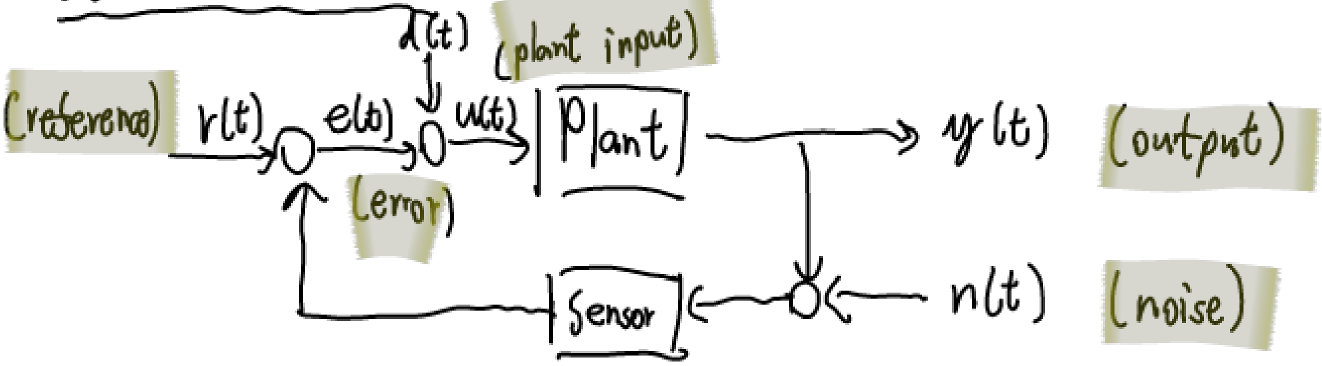
A better controller is proportional-integral error feedback

$$u(t) = -K_p(d_{safe} - d(t)) - K_i \int_0^t (d_{safe} - d(\tau)) d\tau, \quad K_p, K_i > 0$$

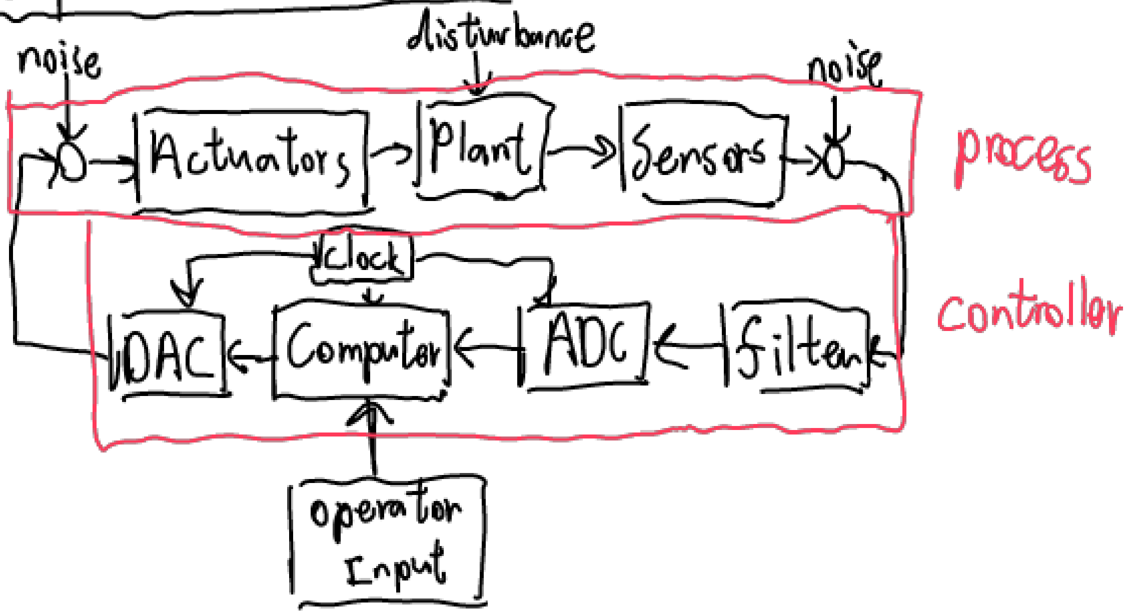
Control Engineering attempts to change the behaviour of a system ("plant") in a desireable way in spite of model uncertainty and despite uncontrolled influences ("disturbances")

We usually change the plant behaviour by connecting it in feedback w/ another system ("controller")

Essential Picture:




Implementation Details:



## Control Cycle:

1) sense the operation of system
2) compare it against desired behaviour
3) compute corrective actions informed by a model of the system's response to external inputs.
4) actuate the system to effect desired change

sense $\longrightarrow$ compute $\longrightarrow$ activate

## 1.3 Control Engineering Design Cycle

1. study system to be controlled; decide on actuators and sensors
2. model system
   - by "model" we mean a mathematical model
   - often one or more differential equations

     e.g. $\frac{dx_f}{dt} = u$
   - based on physics or experiments (system identification)
3. Simplify model if necessary
   - classical control (this course, e.g. P.I.D.) requires that we have a transfer function (TF) of the plant.

     e.g. $\mathcal{L}\left\{\frac{dx_f}{dt}\right\} = \mathcal{L}\{u\} \Rightarrow sX_f(s) - x_f(0) = U(s)$

     $$\Rightarrow \frac{X_f(s)}{U(s)} = \frac{1}{s} + \frac{x_f(0)}{s}$$

     $\frac{1}{s}$ is the TF model of the follower.

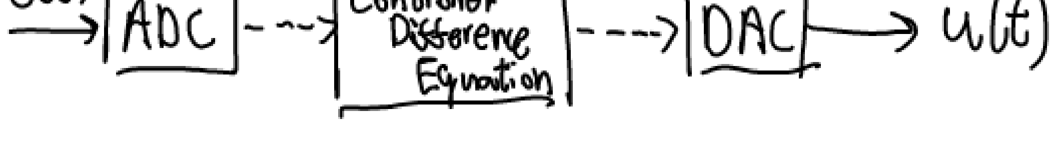
<u>Note</u>: a system has a TF iff it is linear & time invariant.

4. Analyse resulting model
5. Determine specifications
   - stability
   - good steady-state tracking (e.g. $\lim_{t\to\infty} d_{safe} - d(t) = 0$)
   - robust to modelling errors
   - good transient behaviour (e.g. follower can't hit the leader while making $d$ approach $d_{safe}$).
6. Decide on type of controller
7. Design Controller
   - in this course, the controller itself is a TF

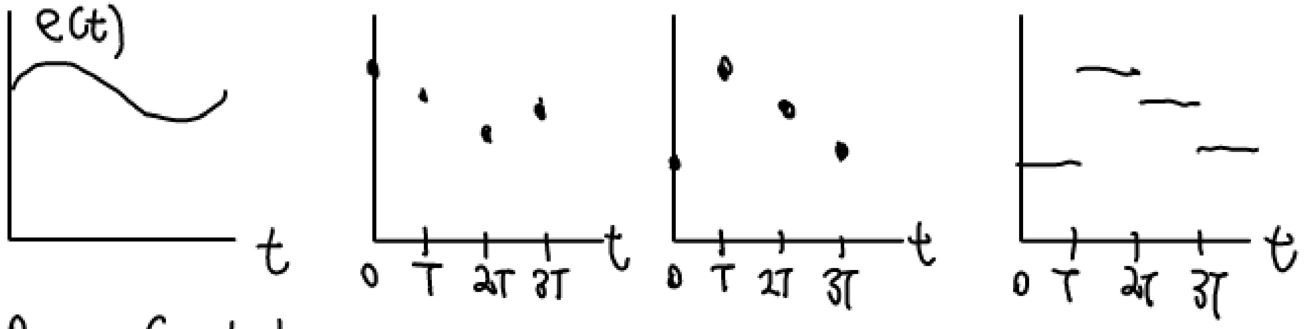

e.g. $C(s) = K_p$

$C(s) = K_p + K_i/s$

8. Simulate closed-loop system
9. Iterate if necessary (goto step 1)
10. Implement Controller
    - You can actually build a system with the same TF from step 7
    - more common: the ODE from step 7 is approximated as a <u>difference equation</u> and implemented on a computer.



$$\xrightarrow{e(t)} \boxed{\text{ADC}} \dashrightarrow \boxed{\text{CDE}} \dashrightarrow \boxed{\text{DAC}} \longrightarrow u(t)$$




Prop. Control

$$u(t) = K_p\left(d_{safe}(nT) - d(nT)\right), \quad nT \leq t < (n+1)T$$

## 1.5 Optimization algorithms as feedback systems

$$\min_{x \in \mathbb{R}^3} f(x), \quad f: \mathbb{R}^n \to \mathbb{R} \quad \text{(possibly nonlinear)}$$

Gradient descent w/ fixed step size: $\boxed{x_{k+1} = x_k - \alpha \nabla f(x_k)}$

$\alpha \in \mathbb{R}$ (constant)

$$\nabla f(x) = \left(\left.\frac{\partial f}{\partial x}\right|_{x=x_k}\right)^T \quad \text{(gradient = transpose of Jacobian)}$$

$$\text{e.g. } f(x) = x_1 x_2 - x_3^2 \Rightarrow \frac{\partial f}{\partial x} = \begin{bmatrix} x_2 & x_1 & -2x_3 \end{bmatrix},$$

$$\nabla f(x) = \begin{bmatrix} x_2 \\ x_1 \\ -2x_3 \end{bmatrix}$$

Initialize at some $x_0 \in \mathbb{R}^n$. At each step algorithm produces new values:

$$x_1 = x_0 - \alpha \nabla f(x_0)$$

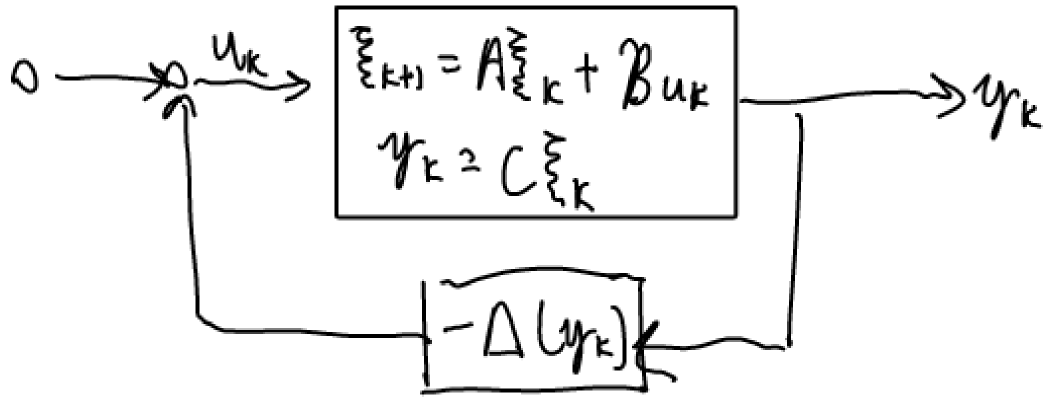
$$x_2 = x_1 - \alpha \nabla f(x_1)$$

$$\vdots$$

Under mild assumptions $f(x_0) \geq f(x_1) \geq f(x_2) \geq \cdots$ and algo converges to local minimum of $f$.

Heavy-ball method: $\boxed{x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta (x_k - x_{k-1})}$

$\beta \in \mathbb{R}$ constant, like friction.




A, B, C constant matrices

Gradient Descent:

$$\xi_k := x_k, \quad A = I_n \in \mathbb{R}^{n \times n}, \quad B = -\alpha I_n, \quad C = I_n, \quad \Delta(y_k) = \nabla f(y_k)$$

Heavy Ball:

$$\xi_k = \begin{bmatrix} \xi_k^{(1)} \\ \xi_k^{(2)} \end{bmatrix} = \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} \in \mathbb{R}^{2n}, \quad A = \begin{bmatrix} (1+\beta) I_n & -\beta I_n \\ I_n & 0_{n \times n} \end{bmatrix} \in \mathbb{R}^{2n \times 2n},$$

$$B = \begin{bmatrix} -\alpha I_n \\ 0_{n \times n} \end{bmatrix} \in \mathbb{R}^{2n \times n} \qquad C = \begin{bmatrix} I_n & 0 \end{bmatrix} \in \mathbb{R}^{n \times 2n}$$

$\Delta(y_k) = \nabla f(x_k)$. $\qquad \xi_{k+1}^{(2)} = \xi_k^{(1)}$

The power of viewing this algo. as a feedback system is that the above block diagram has been extensively studied (Lure problem)

By using these results we can systematically study convergence, robustness to rounding error and even suggest new algos.