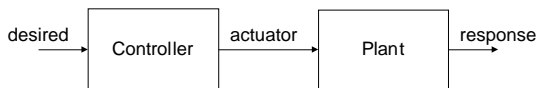


## Control Applications of Neural Nets

## Control Systems in General

- Vocabulary:
  - "Plant" or "Process": The thing being controlled
  - "Controller": The thing doing the controlling
  - Generally both are functions responding to time-varying input.

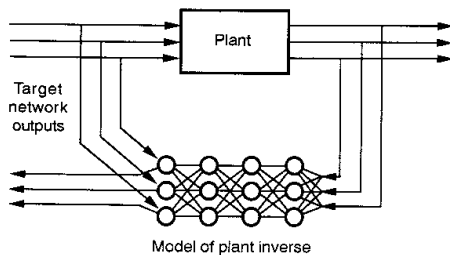
## Open-Loop Control



## Open-Loop Control

- Description of plant (for all time  $t$ ):  
 $response(t) = plant(actuator(t))$
- Description of controller:  
 $actuator(t) = control(desired(t))$
- Thus  
 $response(t) = plant(control(desired(t)))$
- To make  $response(t) = desired(t)$ , make  $control = plant^{-1}$  (the inverse function) modulo a multiplicative constant

## Training a neural net to compute $plant^{-1}$



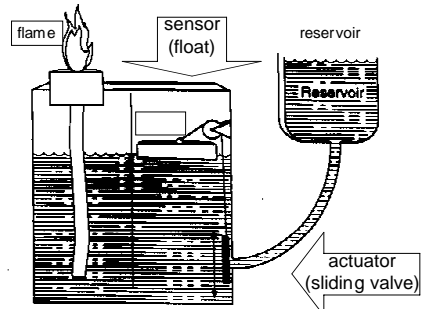
## Problems with "inverse" form of control

- plant function might not be known analytically
- plant function might not be invertible
- plant function could be time-varying (which would require a time-varying net)
- The open-loop approach might be unstable.
- "Feedback control" can help with issue of stability.

## Feedback Control

- Earliest known example?
- How to keep the oil in a reservoir (such as a lamp) at a specific level?
- Answer: Monitor the level; when it drops below the desired level, add more oil, until it reaches the desired level.
- Philon's lamp regulator (220 b.c.)

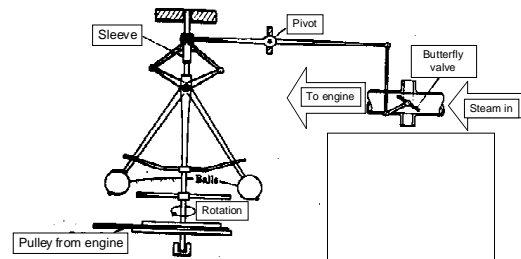
## Philon's Lamp Regulator



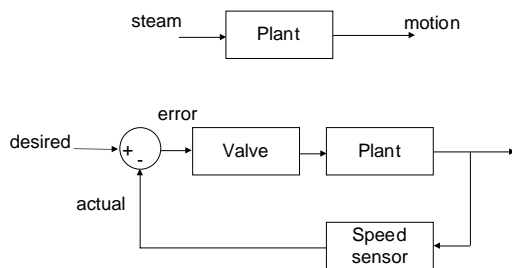
## Steam-Engine Governor

James Watt, 1748

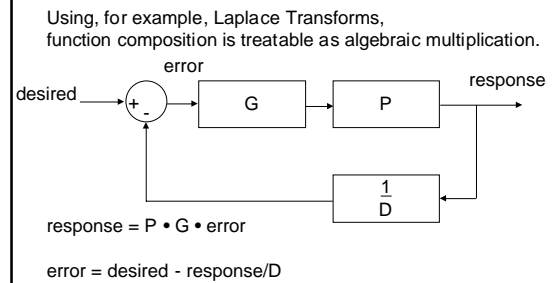
## Flyball Governor (James Watt, 1748)



## Block Diagram of Steam Engine



## Symbolic Reasoning (aside)



## Solve for response = f(desired)

$$\text{response} = P \cdot G \cdot \text{error}$$

$$\text{error} = \text{desired} - \text{response}/D$$

Substituting,

$$\text{response} = P \cdot G \cdot (\text{desired} - \text{response}/D)$$

$$\text{response}(1+PG/D) = PG \cdot \text{desired}$$

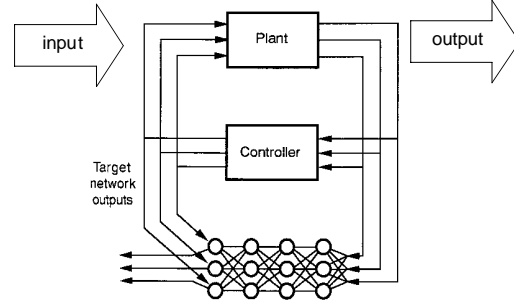
$$\text{response} = \text{desired} \cdot PG / (1+PL)$$

where  $L = G/D$  is the "open loop gain"

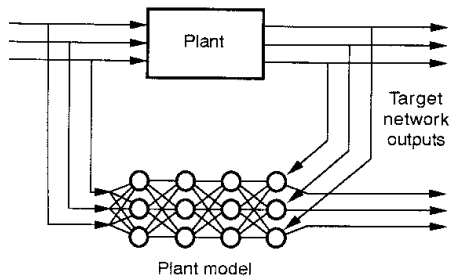
If we can make  $L \gg 1$ ,  $\text{response} = \text{desired} \cdot PG / PL = \text{desired} \cdot G/L$

i.e.  $\text{response} = \text{desired} \cdot D$ .

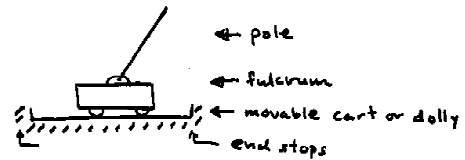
## "Copying" an Existing Feedback Controller with a Neural Net



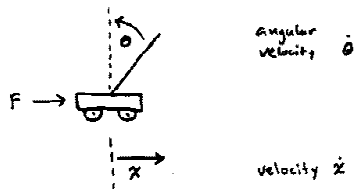
## Copying a Plant as a Neural Net



## Pole-Balancing (pole-cart, broom balancing)



## Pole-Cart Dynamics



$$F(t) = a \ddot{x}(t) + b \dot{x}(t) + c \theta(t) + d \dot{\theta}(t)$$

## Approaches

- Design a controller using classical approach (example: matlab pendemo)
- Develop parameterized controller, then learn parameters using neural net by training (Widrow's approach #1)
- Widrow #2 (Tolat & Widrow)

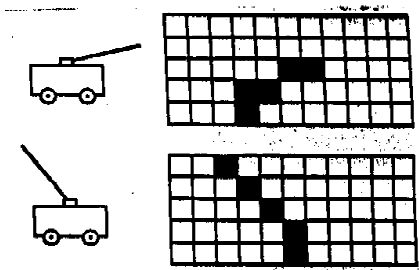
## Widrow #1

- Used Adaline to “observe” an operating controller, with input values  $x, x', \theta, \theta'$
- Train Adaline to emit appropriate force on cart

## Widrow #2 (Tolat & Widrow)

- **Pairs of images** of pole position were input, taken at successive points 100ms apart
- 5x11 pixel image used
- Equations were not used; instead a “teacher’s input” was given indicating force on cart based on a given image pair. Only the sign of this force was actually used (“bang-bang” control).

## States & Images (sample)



## Tolat & Widrow conclusions

- 4225 image pairs used in training
- Adaline with 110 weights + threshold was sufficient to balance the pole
- Error 3.4% after only 1000 training cycles

## Demos

- Jeff Lawson '99 and Chris Lewis '99 demos, using
  - AHC (Adaptive Heuristic Critic)
  - SANE (Symbiotic Adaptive Neuro-Evolution)
- matlab/broom2: run dbroom\_ndemo (2-broom demo by Wan, et al.; does not show BPTT training phase)

## Looking Forward

- Lots of interest in
  - neural control
  - neuro-fuzzy control (neural networks + fuzzy logic)
  - neuro-evolutionary control (neural networks + evolution programs)