

MSc on Intelligent Critical Infrastructure Systems

Machine Learning

Lecture 13 and 14

Marios Polycarpou

Director, KIOS Research and Innovation Center of Excellence

Professor, Electrical and Computer Engineering

University of Cyprus

funded by:

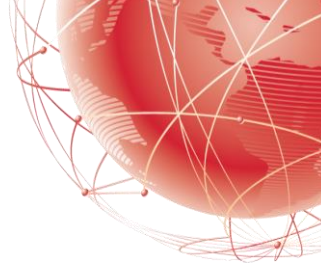




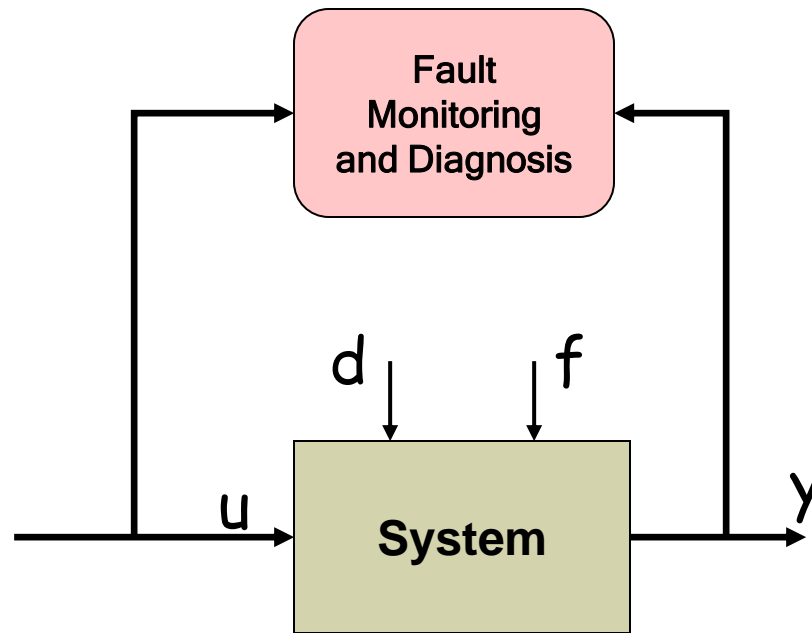
Application to Monitoring and Control – Outline

- Introduction to Monitoring and Control
- Mathematical Modelling of Dynamical Systems
- Adaptation and Learning in Control Systems
- System Identification (Continuous-time & Discrete-time)
- Learning Control using Neural Networks

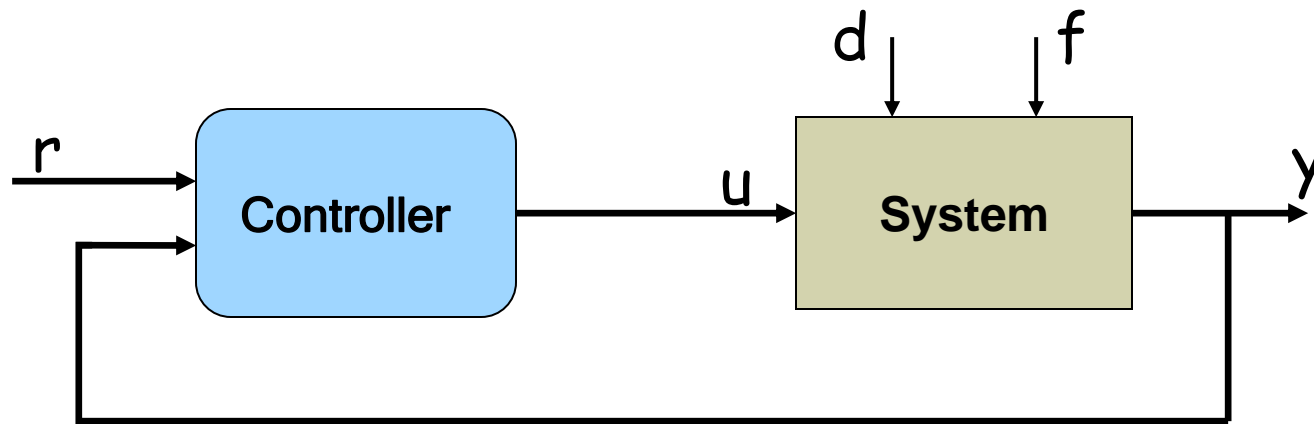
→ CONCLUDING REMARKS FOR THE CLASS



Monitoring and Control

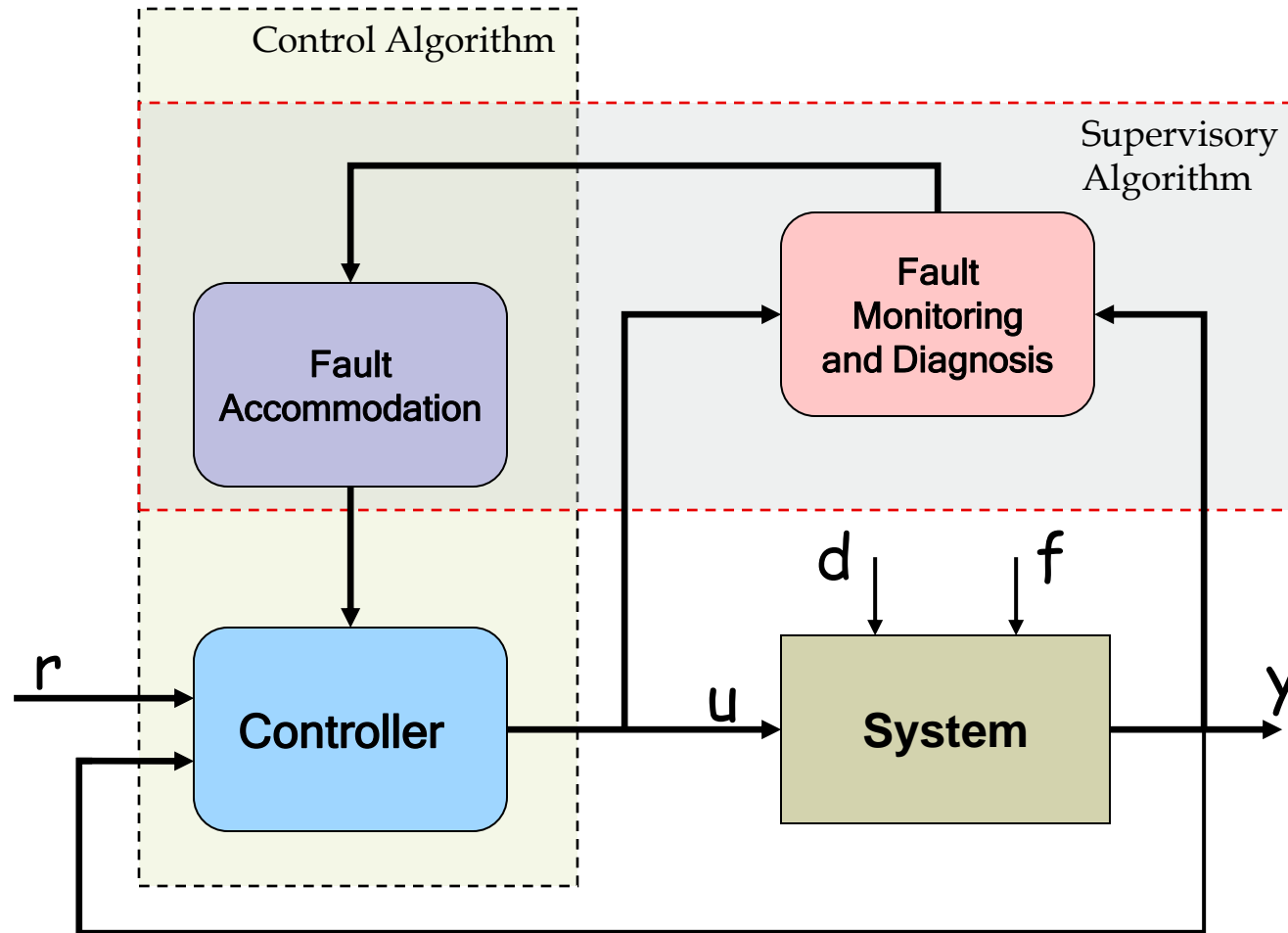


Monitoring and Control





Monitoring and Control



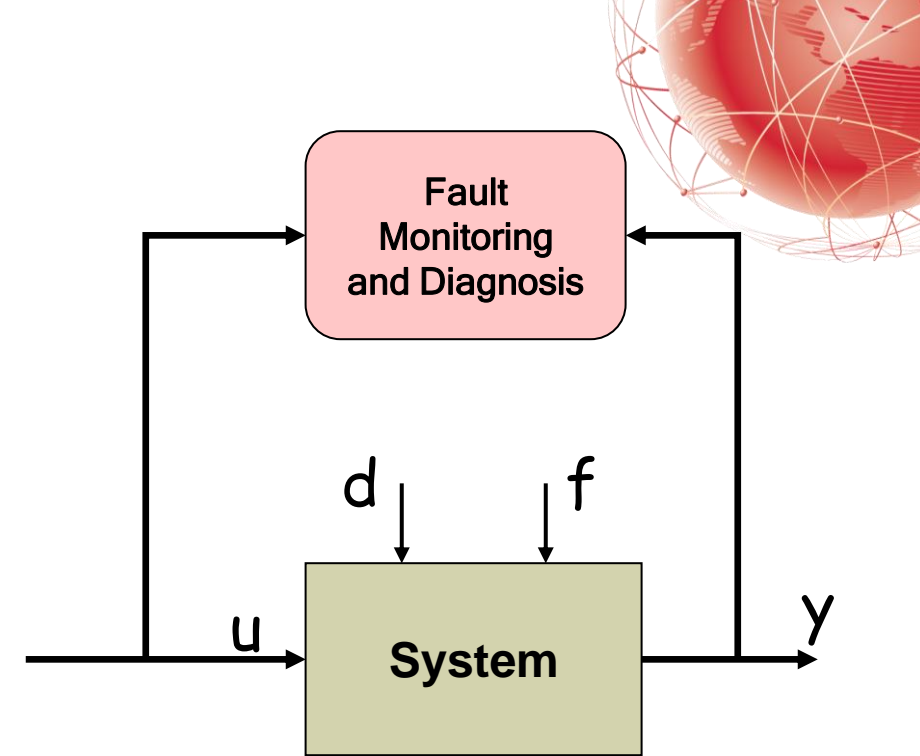


Monitoring and Control Applications

- Power and Energy Systems
- Transportation Systems
- Water Distributions Networks
- Distributed Autonomous Vehicles
- Chemical and Petrochemical Engineering Processes
- Smart Buildings
- Manufacturing Systems
- Biological and Biomedical Engineering Applications
- Environmental Monitoring and Control Applications
- Military and Security Applications

Monitoring Formulation

- Utilize input and output data from the system to determine whether it is behaving as it should
- Operates in real-time during operation of the system
- Acts passively; does not affect the behavior of the system
- If it is detected that something is not working well, then an action may be triggered to affect the behavior of the system
- Typically, a monitoring agent consists of:
 - Mathematical model of the system that is being monitored
 - Filtering algorithms
 - Decision logic algorithms

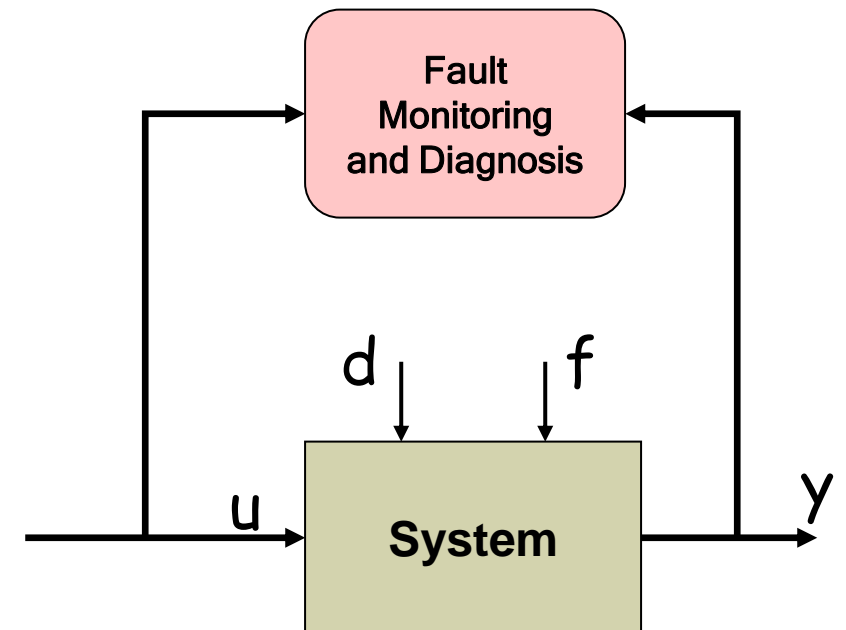




Monitoring Formulation

WHY DO WE NEED TO MONITOR THE SYSTEM? WHAT CAN GO WRONG?

- System/Process Faults
- Actuator Faults
- Sensor Faults
- Communication Faults
- Controller Faults
- Environment Faults
- Malicious Attacks (cyber-security)

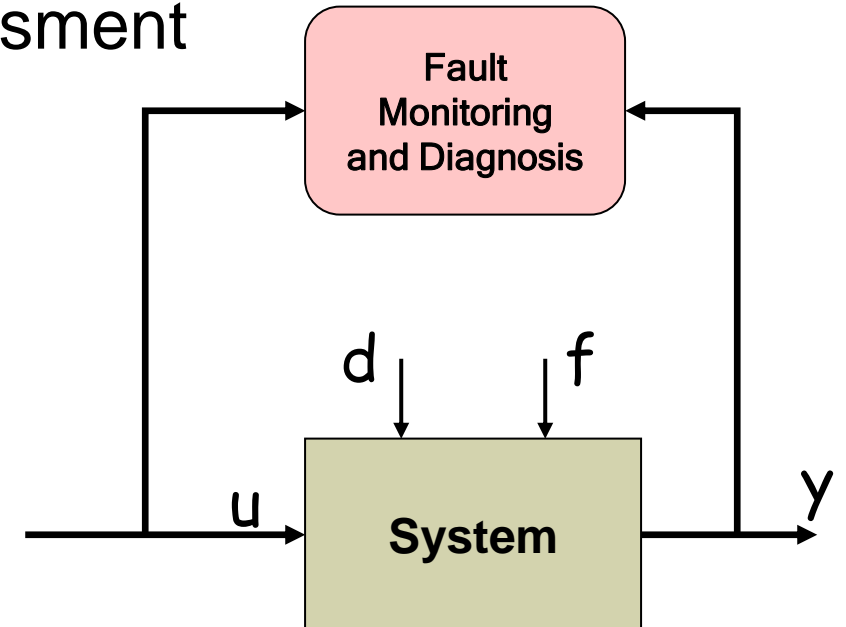




Monitoring Formulation

FAULT/EVENT DIAGNOSTIC STEPS:

- Fault/event detection
- Fault/event isolation
- Fault/event identification and risk assessment
- Fault/event accommodation
 - Active fault accommodation
 - Passive fault accommodation

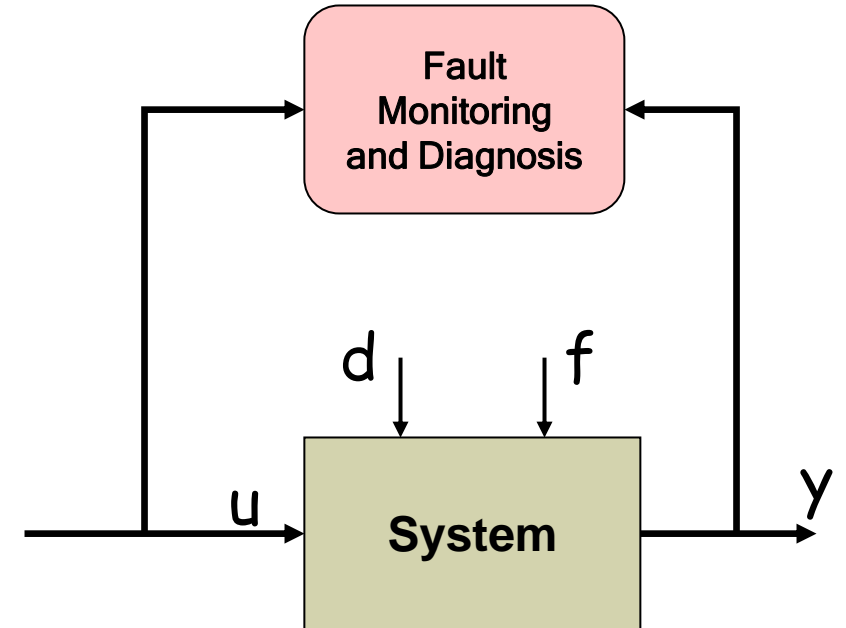




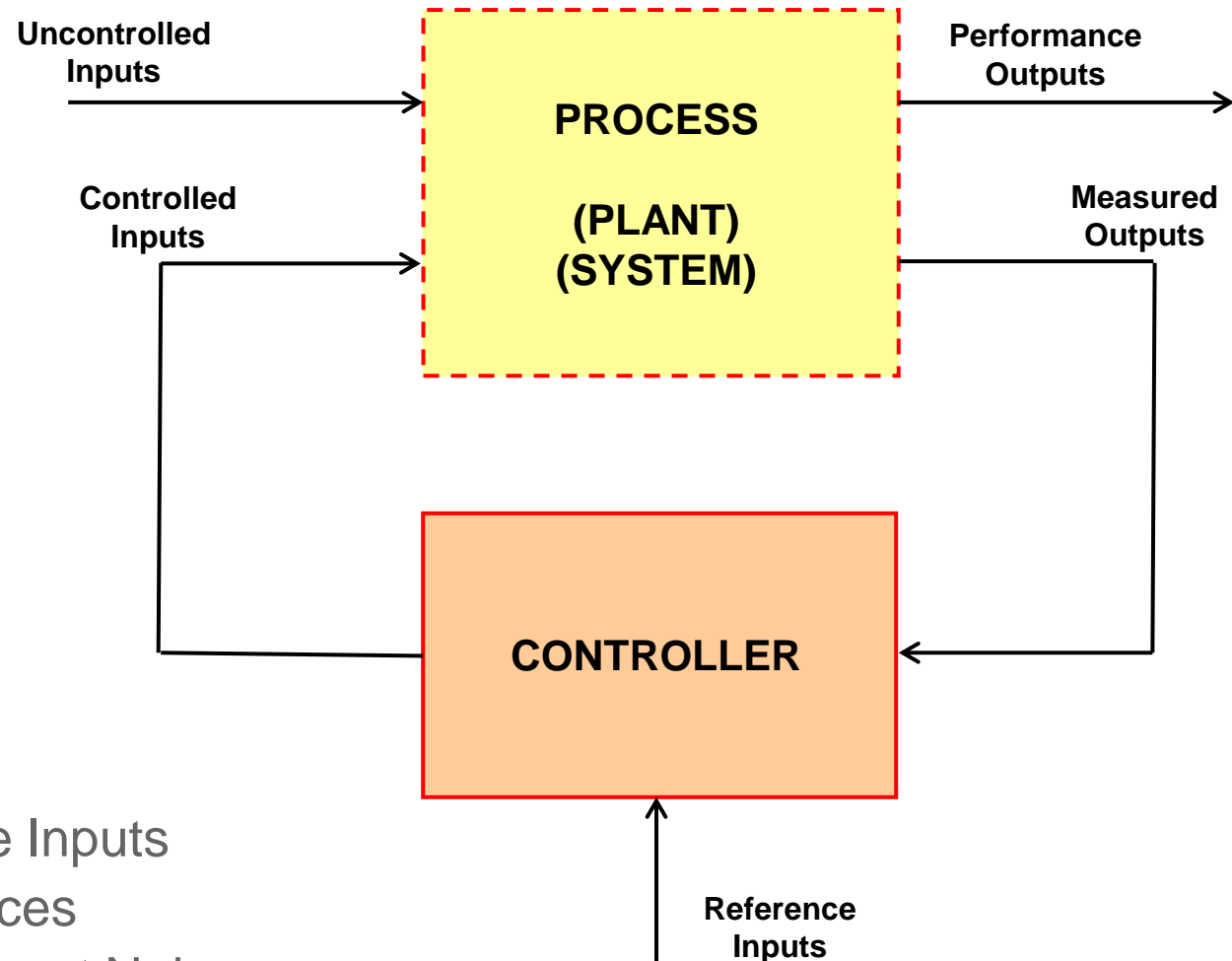
Monitoring Formulation

KEY CHALLENGES

- distinguish between faults and modeling uncertainty or measurement noise
- avoid false positives and false negatives
- handle multiple faults
- isolate faults in a large-scale system (needle in a haystack)
- prevent “small” faults from escalating into a major failure
- reliable fault accommodation
- cyber-physical security



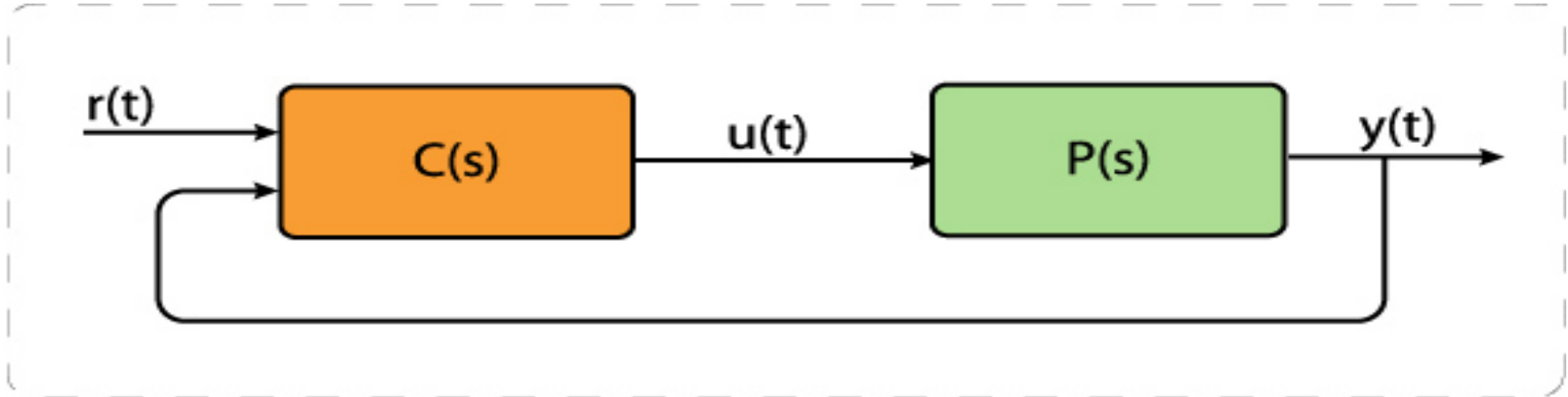
General Control Formulation



Key Issues:

- Sensors
- Actuators
- Reference Inputs
- Disturbances
- Measurement Noise
- ***Feedback***

General Control Formulation

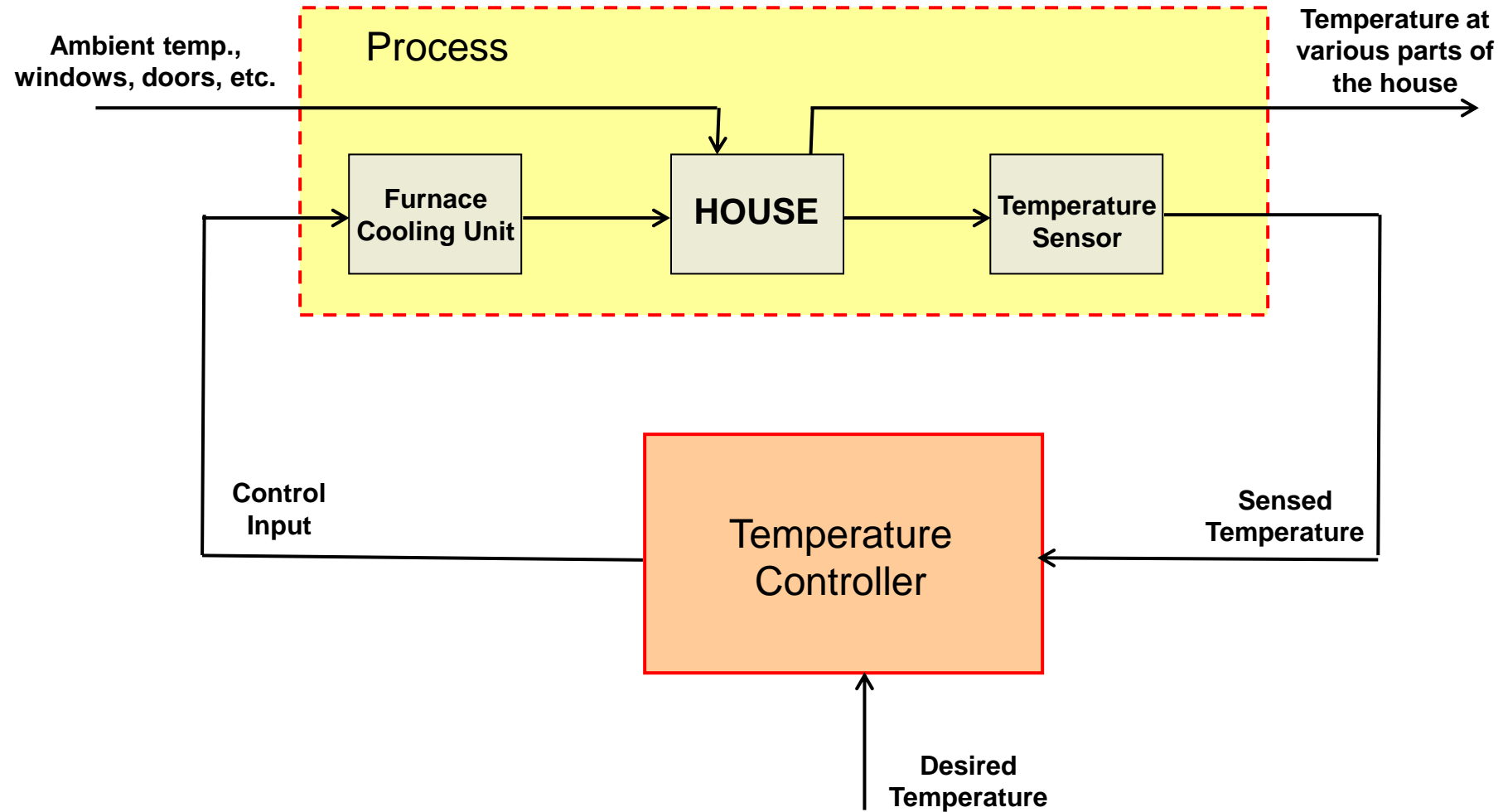


Key Issues:

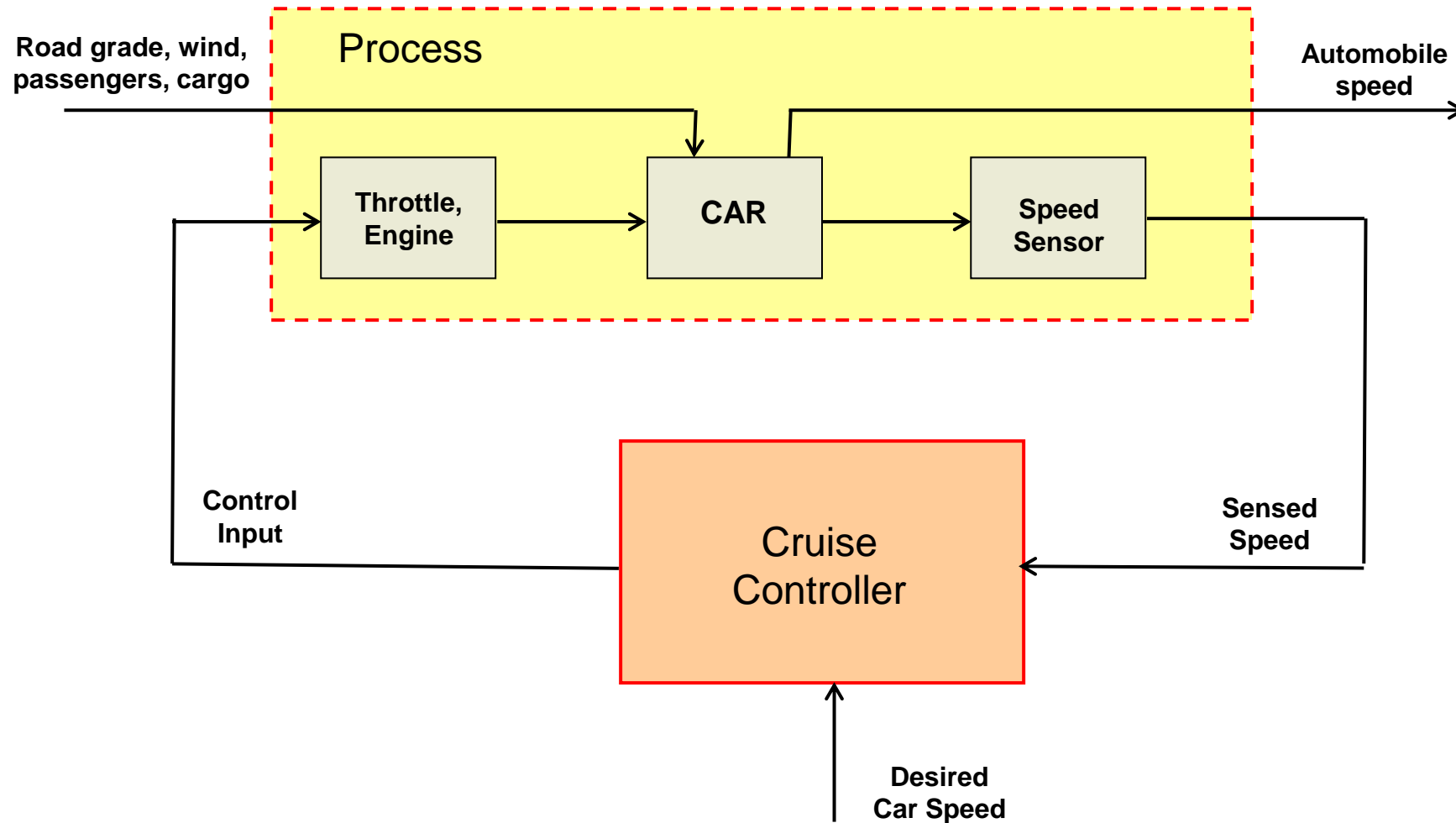
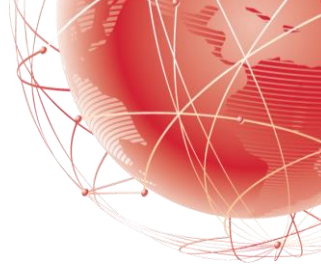
- Sensors
- Actuators
- Reference Inputs
- Disturbances
- Measurement Noise
- ***Feedback***



Simple Example: Temperature Control



Simple Example: Automobile Cruise Control





Mathematical Modelling

- a) Based on first principles of physics (chemistry, biology, economics, etc.)
- b) System identification using real data
- c) Combination of first principles and system identification

WHY DO WE NEED A MATHEMATICAL MODEL?

- Prediction
- Control
- Monitoring
- Design



Mathematical Modelling

HOW ACCURATE DOES A MATHEMATICAL MODEL NEED TO BE?

- Prediction Model vs Control Design Model
- Limitations of mathematical modelling

→ *Everything Should Be Made as Simple as Possible, But Not Simpler*
– A. Einstein

→ *Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.*
– George E. Box

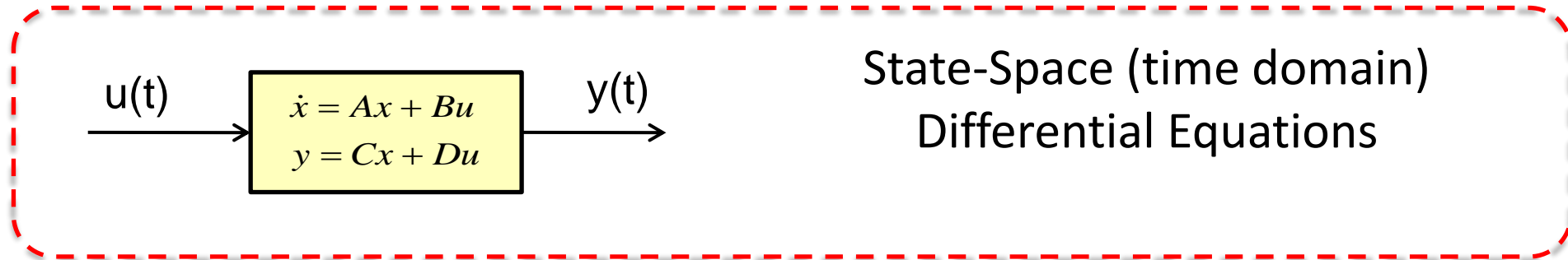
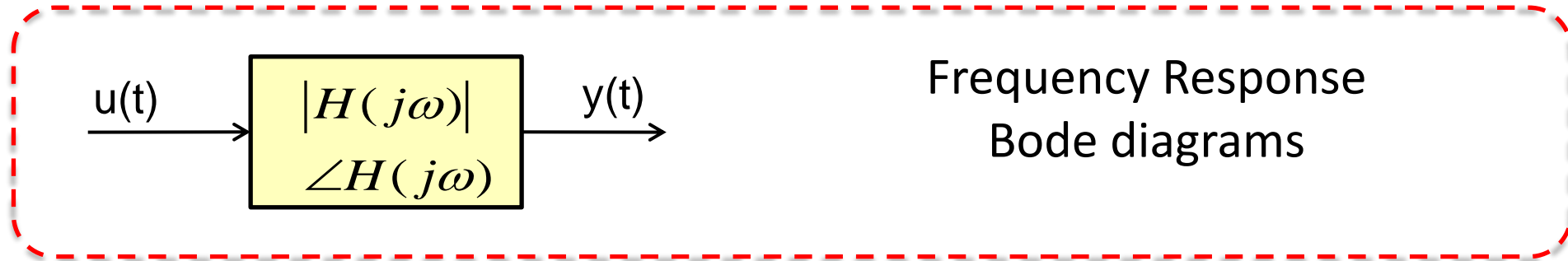
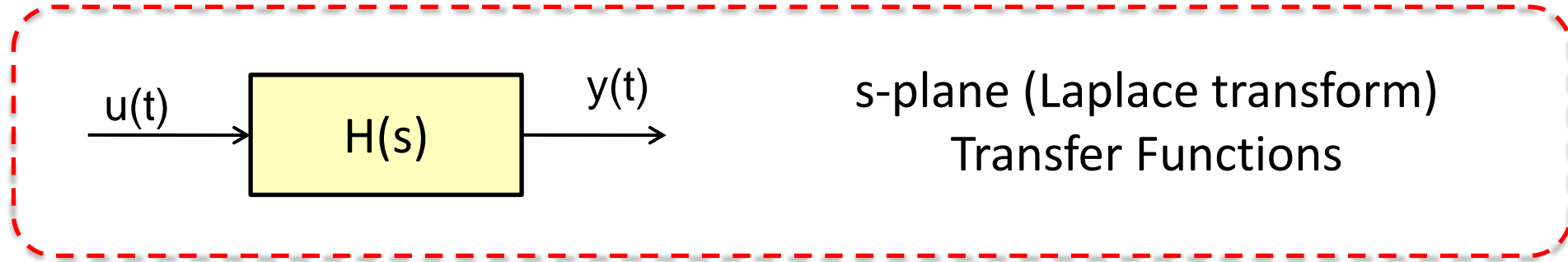
Mathematical Modelling



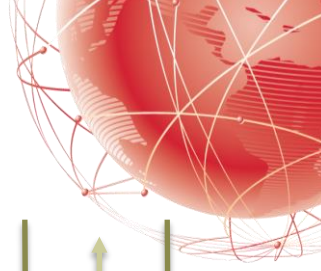
- Differential/dynamic systems vs algebraic systems
 - Systems with memory – the outputs depend not only on the inputs but also on the initial conditions
- Linear vs nonlinear models
- Continuous-time vs discrete-time models
- Time invariant (stationary) vs time-varying systems (non-stationary)



Mathematical Modelling: the 3 domains

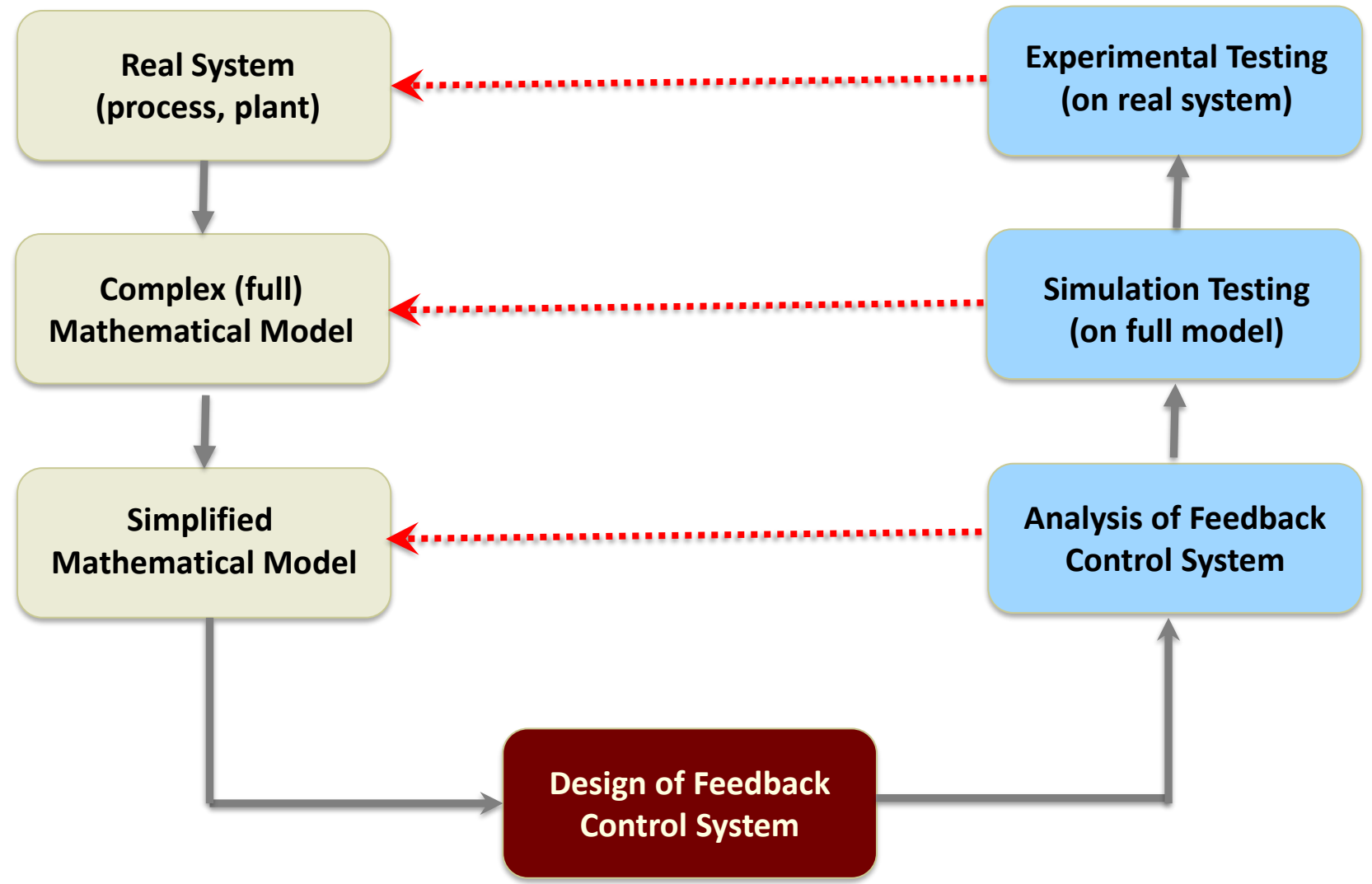


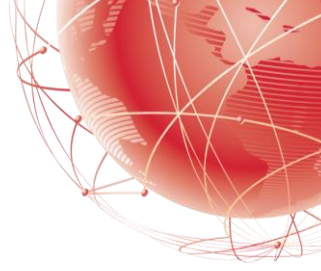
Modelling, Control Design and Evaluation



Evaluation and Testing

Modelling





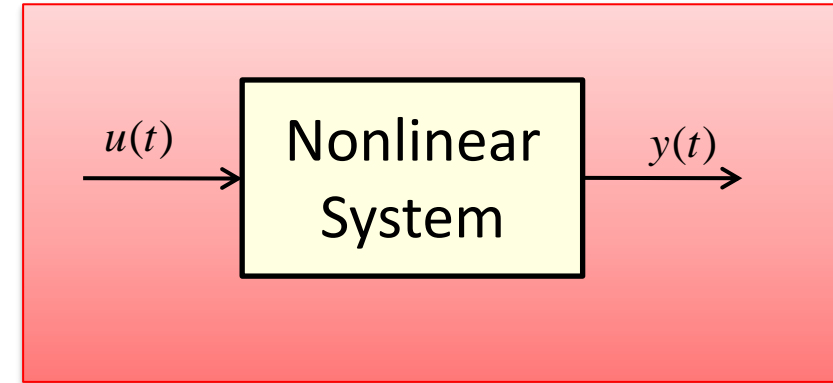
Model-based design vs Data-driven design

- Model-based design methods for monitoring and control are based on developing a model of the system, while data-driven (model-free) methods are based on experimental data
- Some key trade-offs:
 - Model-based design methods facilitates the use of powerful design tools and analysis methods
 - Errors can be detected at various steps of the procedure
 - Developing a model may be time consuming and expensive
 - Inaccurate models may lead to bad performance

Learning Control: Motivation

$$\dot{x}(t) = f_0(x(t)) + \boxed{f(x(t))} + u(t)$$
$$y(t) = x(t)$$

↓
unknown



Feedback control:

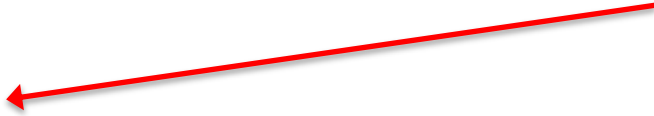
$$u(t) = -f_0(x(t)) + \dot{x}_d(t) - K[x(t) - x_d(t)]$$

Closed-Loop Dynamics:

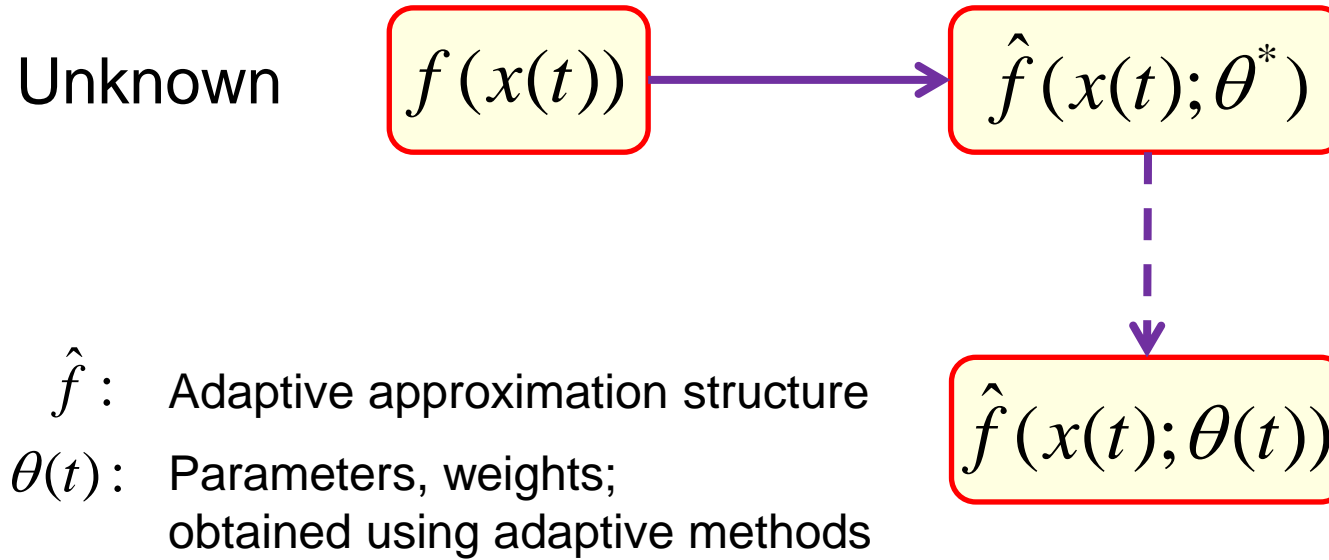
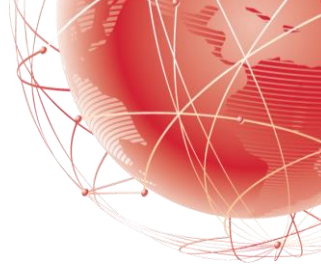
$$\dot{e}(t) = -Ke(t) + \boxed{f(x(t))}$$

$$e(t) = x(t) - x_d(t)$$

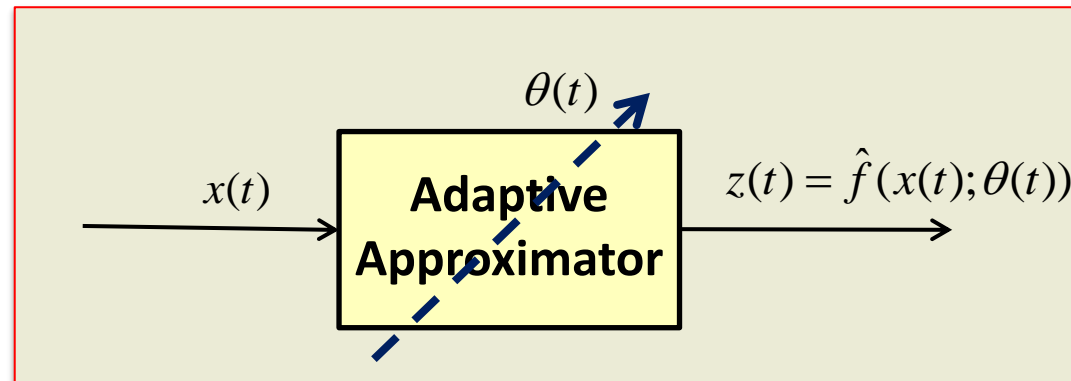
May lead to
instability



Learning Control: Motivation



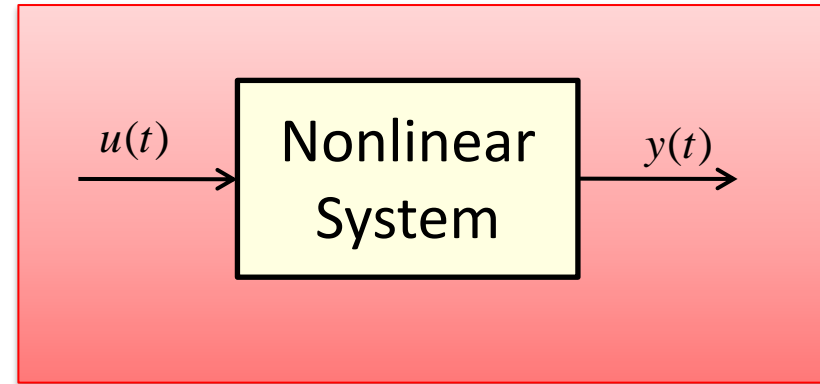
LEARNING THE
UNCERTAINTY
DURING
OPERATION OF
THE SYSTEM



Learning Control: Motivation

$$\dot{x}(t) = f_0(x(t)) + \boxed{f(x(t))} + u(t)$$
$$y(t) = x(t)$$

↓
unknown



Feedback control:

$$u(t) = -f_0(x(t)) + \dot{x}_d(t) - K[x(t) - x_d(t)] - \hat{f}(x(t); \theta(t))$$

Closed-Loop Dynamics:

$$\dot{e}(t) = -Ke(t) + \boxed{f(x(t)) - \hat{f}(x(t); \theta(t))}$$
$$e(t) = x(t) - x_d(t)$$

Improves performance and reduces the possibility of instability



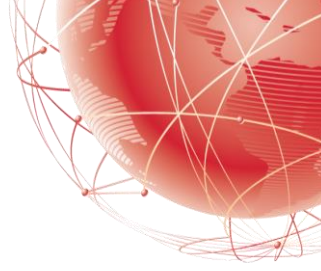
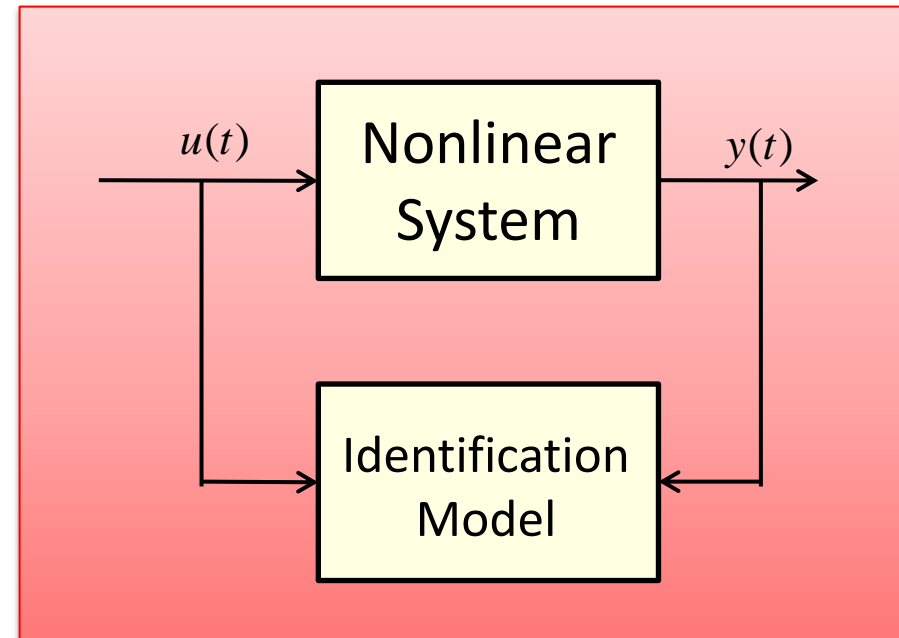
Nonlinear Identification (Learning)

$$\dot{x}(t) = \xi(x(t), u(t), t) + f(x(t), u(t), t)$$

$$y(t) = \zeta(x(t), u(t), t) + h(x(t), u(t), t)$$

$x(t)$: State variable
 $u(t)$: Control input
 ξ, ζ : Known parts
 f, h : Unknown parts

Problem: design an identification model that allows estimation of the unknown f and h .





Start with a first-order linear example

$$\dot{x}(t) = -ax(t) + bu(t) \quad a, b : \text{unknown}$$

[Plant]

$$\dot{\hat{x}}(t) = -a_m \hat{x}(t) + (a_m - \hat{a}(t))x(t) + \hat{b}(t)u(t) \quad a_m > 0$$

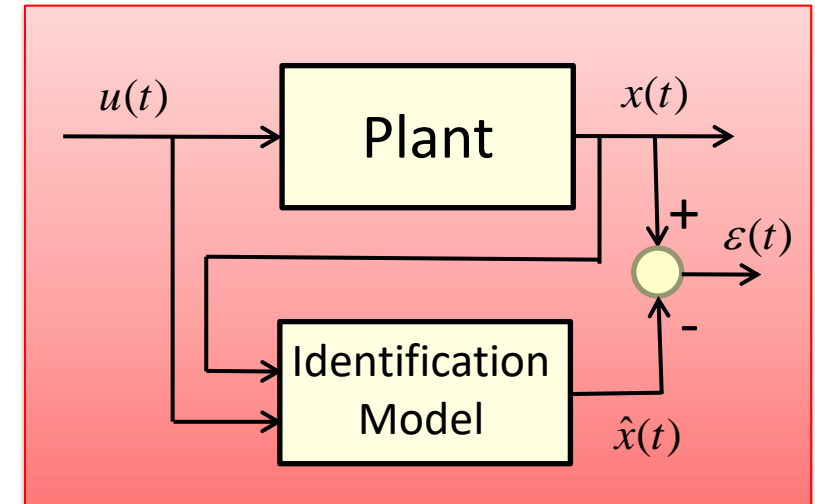
[Estimation Scheme]

$$\dot{\hat{a}}(t) = -\gamma_1 \varepsilon(t)x(t)$$

$$\dot{\hat{b}}(t) = \gamma_2 \varepsilon(t)u(t)$$

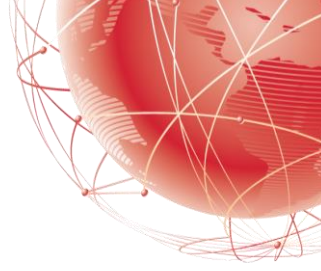
[Adaptive Laws]

$$\varepsilon(t) = x(t) - \hat{x}(t) \quad \text{[Estimation Error]}$$



Design Parameters: $a_m > 0, \gamma_1 > 0, \gamma_2 > 0, \hat{x}(0)$

Nonlinear Systems with Unknown Parameters



$$\dot{x}(t) = -ax(t) + cf(x(t)) + bg(u(t)) \quad a, b, c : \text{unknown}$$
$$f(\cdot), g(\cdot) : \text{known}$$

$$\dot{\hat{x}}(t) = -a_m \hat{x}(t) + (a_m - \hat{a}(t))x(t) + \hat{c}(t)f(x(t)) + \hat{b}(t)g(u(t)) \quad \text{Estimation Scheme}$$

$$\hat{x}(t) = \frac{1}{s + a_m} \left[(a_m - \hat{a}(t))x(t) + \hat{c}(t)f(x(t)) + \hat{b}(t)g(u(t)) \right]$$

Adaptive Laws

$$\dot{\hat{a}}(t) = -\gamma_1 \varepsilon(t)x(t)$$

$$\dot{\hat{b}}(t) = \gamma_2 \varepsilon(t)g(u(t))$$

$$\dot{\hat{c}}(t) = \gamma_3 \varepsilon(t)f(x(t))$$

$$\varepsilon(t) = x(t) - \hat{x}(t) \quad \text{[Estimation Error]}$$

Nonlinear Systems with Unknown Functions



$$\dot{x}(t) = \underbrace{\xi(x(t), u(t))}_{\text{known}} + \underbrace{f(x(t), u(t))}_{\text{unknown}} \quad \text{Plant}$$

$$\dot{\hat{x}}(t) = -a_m \hat{x}(t) + \xi(x(t), u(t)) + a_m x(t) + \hat{f}(x(t), u(t); \hat{\theta}(t)) \quad \text{Estimation Scheme}$$

$$\hat{x}(t) = \frac{1}{s + a_m} \left[\xi(x(t), u(t)) + a_m x(t) + \hat{f}(x(t), u(t); \hat{\theta}(t)) \right]$$

Adaptive Laws

$$\dot{\hat{\theta}} = \Gamma Z^T \varepsilon$$

$$Z = \frac{\partial \hat{f}}{\partial \hat{\theta}}(x, u, \hat{\theta})$$

$$\varepsilon = x - \hat{x}$$

$$\Gamma > 0 \quad (\text{positive definite})$$

Continuous-Time Learning Using Neural Networks



$$\dot{x}(t) = f(x(t)) + u(t) \quad f : \text{unknown}$$

$$\dot{\hat{x}}(t) = -a_m \hat{x}(t) + a_m x(t) + \hat{f}(x(t); \hat{\theta}(t)) + u(t) \quad a_m > 0$$

$$\dot{\hat{\theta}}(t) = \Gamma Z(t)^T \varepsilon(t)$$

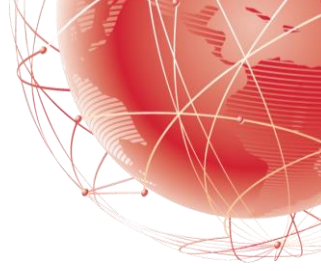
A. Radial Basis Function (RBF) Networks

$$\hat{f}(x; \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_N) = \sum_{i=1}^N \hat{\theta}_i e^{-(x-c_i)^2/\sigma^2}$$

$$Z_i = \frac{\partial \hat{f}}{\partial \hat{\theta}_i} = e^{-(x-c_i)^2/\sigma^2}$$

$$\dot{\hat{\theta}}_i = \gamma_i e^{-(x-c_i)^2/\sigma^2} (x - \hat{x})$$

Continuous-Time Learning Using Neural Networks



Implementation using RBF Networks

$$\dot{x} = f(x) + u$$

$$x(0) = x^0$$

$$\dot{\hat{x}} = -a_m \hat{x} + a_m x + \sum_{i=1}^N \left(\hat{\theta}_i e^{-(x-c_i)^2/\sigma^2} \right) + u$$

$$\hat{x}(0) = \hat{x}^0$$

$$\dot{\hat{\theta}}_i = \gamma_i e^{-(x-c_i)^2/\sigma^2} (x - \hat{x})$$

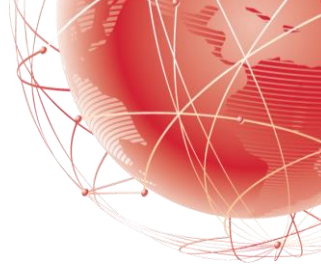
$$\hat{\theta}_i(0) = \hat{\theta}_i^0$$

Design parameters to choose:

$$a_m, \sigma, \hat{x}^0, \gamma_i, c_i, \hat{\theta}_i^0$$

$$i = 1, 2, \dots, N$$

Continuous-Time Learning Using Neural Networks



B. Sigmoidal Neural Networks (SNN)

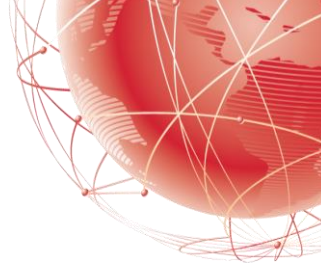
$$\hat{f}(x; \hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_{3M}) = \sum_{i=1}^M \hat{\theta}_i \sigma(\hat{\theta}_{i+M} x + \hat{\theta}_{i+2M})$$

$$\sigma(p) = \frac{1}{1 + e^{-p}} \quad \Rightarrow \quad \sigma(\alpha x + \beta) = \frac{1}{1 + e^{-(\alpha x + \beta)}}$$

$$Z_i = \frac{\partial \hat{f}}{\partial \hat{\theta}_i}$$

$$\dot{\hat{\theta}}_i = \gamma_i Z_i (x - \hat{x})$$

Continuous-Time Learning Using Neural Networks



Implementation of SNN

$$\dot{x} = f(x) + u$$

$$x(0) = x^0$$

$$\dot{\hat{x}} = -a_m \hat{x} + a_m x + \sum_{i=1}^M \hat{\theta}_i \sigma(\hat{\theta}_{i+M} x + \hat{\theta}_{i+2M}) + u$$

$$\hat{x}(0) = \hat{x}^0$$

$$\dot{\hat{\theta}}_i = \gamma_i \frac{\partial \hat{f}}{\partial \hat{\theta}_i} (x - \hat{x})$$

$$\hat{\theta}_i(0) = \hat{\theta}_i^0$$

Design parameters to choose:

$$a_m, \hat{x}^0, \gamma_i, \hat{\theta}_i^0$$

$$i = 1, 2, \dots, 3M$$

Discrete-time Learning in dynamical systems



$$y(k+1) = f\left(y(k), y(k-1), \dots, y(k-n_y), u(k), \dots, u(k-n_u)\right)$$

$$k \in \mathbb{Z}^+ \quad k = 0, 1, 2, \dots$$

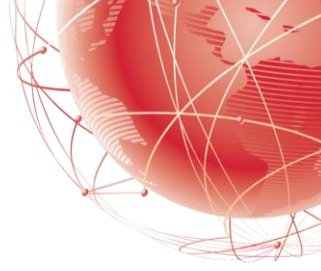
$$f : \mathbb{R}^{n_y+1} \times \mathbb{R}^{n_u+1} \rightarrow \mathbb{R}$$

f is unknown (or partially unknown)

$$\text{Let } z(k) = \left[y(k), y(k-1), \dots, y(k-n_y), u(k), \dots, u(k-n_u) \right]$$

$$\Rightarrow y(k+1) = f(z(k)) \quad (z(k) \text{ is measurable})$$

Discrete-Time Learning



$$y(k+1) = f(z(k))$$

$$\hat{y}(k+1) = \hat{f}(z(k); \hat{\theta}(k))$$

Prediction/Identification Model

Adaptive Laws:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \alpha_0 e(k+1) \xi(k)$$

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \frac{\gamma_0 e(k+1)}{\beta_0 + |\xi(k)|^2} \xi(k)$$

Normalized Gradient Descent

$$\alpha_0 > 0$$

Step size

$$0 < \gamma_0 < 2$$

Learning rate

$$\beta_0 > 0$$

Small design constant

$$\xi(k) = \frac{\partial \hat{f}}{\partial \hat{\theta}}(z(k), \hat{\theta}(k))$$

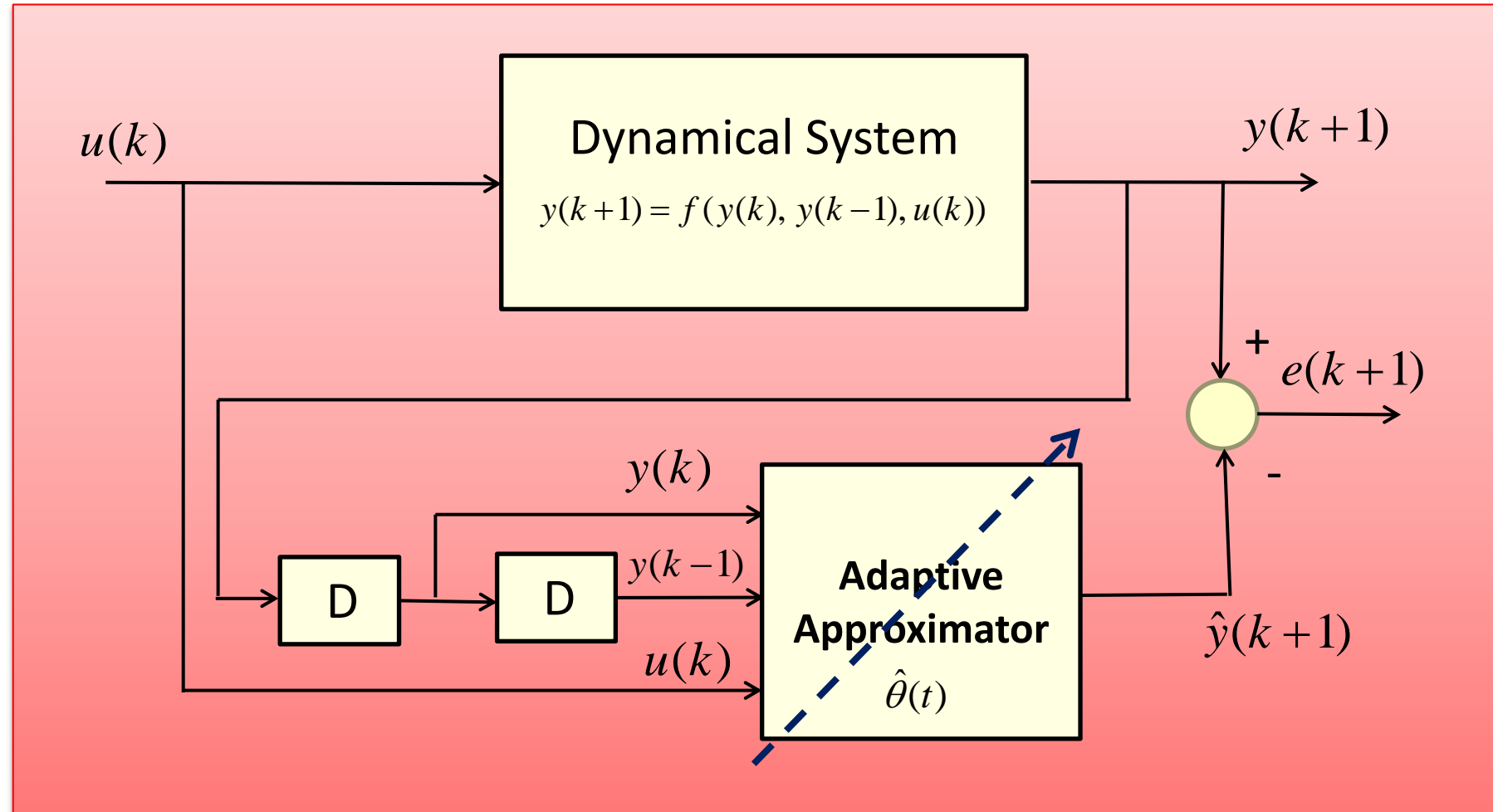
Network sensitivity function

$$e(k) = y(k) - \hat{y}(k)$$

Estimation error

Discrete-Time Learning

Example: $y(k+1) = f(y(k), y(k-1), u(k))$



Discrete-Time Learning

Example: $y(k+1) = f(y(k), y(k-1), u(k))$

Given $y(0), y(-1), \hat{\theta}(0), u(0)$

\begin {for} $k = 0, 1, 2, \dots, N$

1. $z(k) = [y(k), y(k-1), u(k)]$

2. $\xi(k) = \frac{\partial \hat{f}}{\partial \hat{\theta}}(z(k), \hat{\theta}(k))$

3. $y(k+1) = f(z(k))$

4. $\hat{y}(k+1) = \hat{f}(z(k), \hat{\theta}(k))$

5. $\hat{\theta}(k+1) = \hat{\theta}(k) + \frac{\gamma_0 (y(k+1) - \hat{y}(k+1))}{\beta_0 + |\xi(k)|^2} \xi(k)$

\end {for}

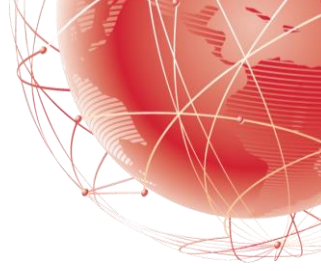


From Learning to Fault Diagnosis



- A similar estimation scheme can also be used for fault diagnosis
- Additionally, in fault diagnosis we require upper bounds (or statistical information) on the uncertainty so that we can distinguish between faults and modelling uncertainties
- Based on the modelling uncertainty bounds, we can compute adaptive threshold signals
- Finally, we also need to design a decision logic algorithm for fault detection and isolation

Learning Control Using Neural Networks



$$\dot{x}(t) = \xi(x(t), u(t), t) + f(x(t), u(t), t)$$

$$y(t) = \zeta(x(t), u(t), t) + h(x(t), u(t), t)$$

$x(t)$: State variable
 $u(t)$: Control input
 ξ, ζ : Known parts
 f, h : Unknown parts

Problem: design a controller:

$$u = \mu(y, \theta, t)$$

$$\dot{\theta} = \lambda(u, y, \theta, t)$$

Such that $y(t)$ follows a desired trajectory $y_r(t)$ as closely as possible.

Start with a simple case: linear adaptive control

$$\dot{y}(t) = ay(t) + bu(t)$$

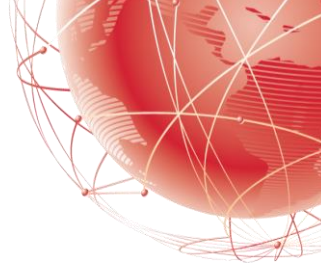
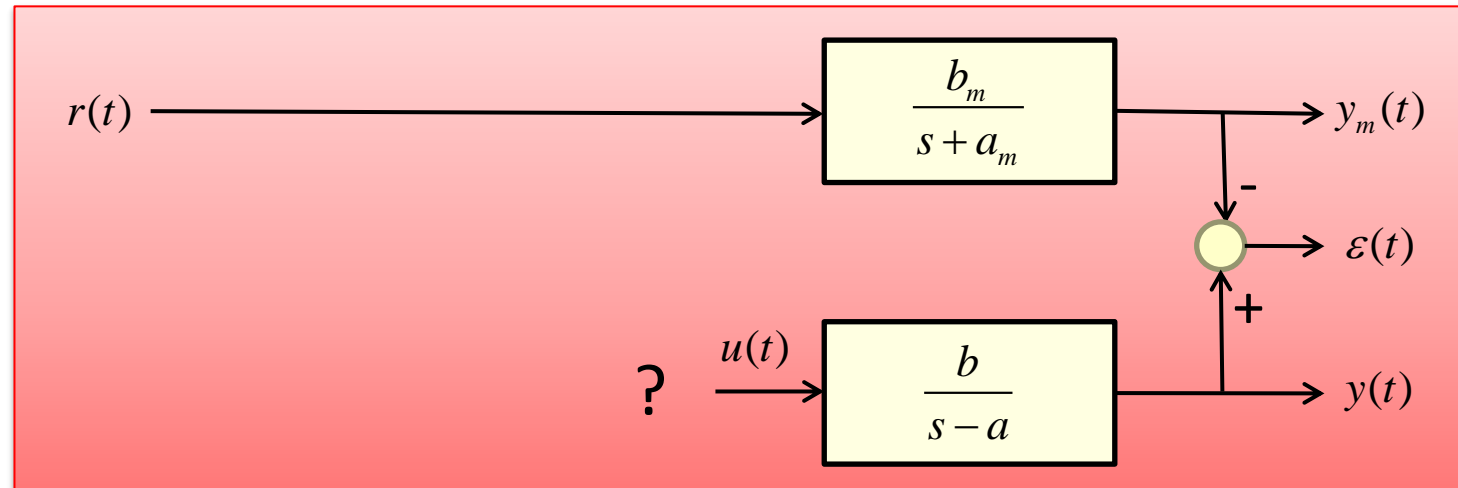
a, b : unknown; $\text{sign}(b)$ is known

Model Reference Adaptive Control (MRAC) Problem:

Choose an appropriate control law $u(t)$ such that all signals in the closed loop plant are bounded and the plant output $y(t)$ tracks the output $y_m(t)$ of the reference model:

$$\dot{y}_m(t) = -a_m y_m(t) + b_m r(t)$$

where $r(t)$ is a bounded, piecewise continuous signal, referred to as the reference or command input.





Start with a simple case: linear adaptive control

How would we solve the MRAC problem if we knew a, b?

$$u = -k^* y(t) + l^* r(t)$$

$$\Rightarrow k^* = \frac{a_m + a}{b} \quad l^* = \frac{b_m}{b}$$

Assuming that b is not equal to 0
(plant is controllable)

$$\frac{y(s)}{r(s)} = \frac{y_m(s)}{r(s)} \Rightarrow |y(t) - y_m(t)| \rightarrow 0 \text{ (exponentially fast for all } r(t)\text{)}$$

Direct Adaptive Control (Direct MRAC)

$$\dot{y}(t) = ay(t) + bu(t)$$

$$\dot{y}_m(t) = -a_m y_m(t) + b_m r(t)$$

$$u(t) = -k(t)y(t) + l(t)r(t)$$

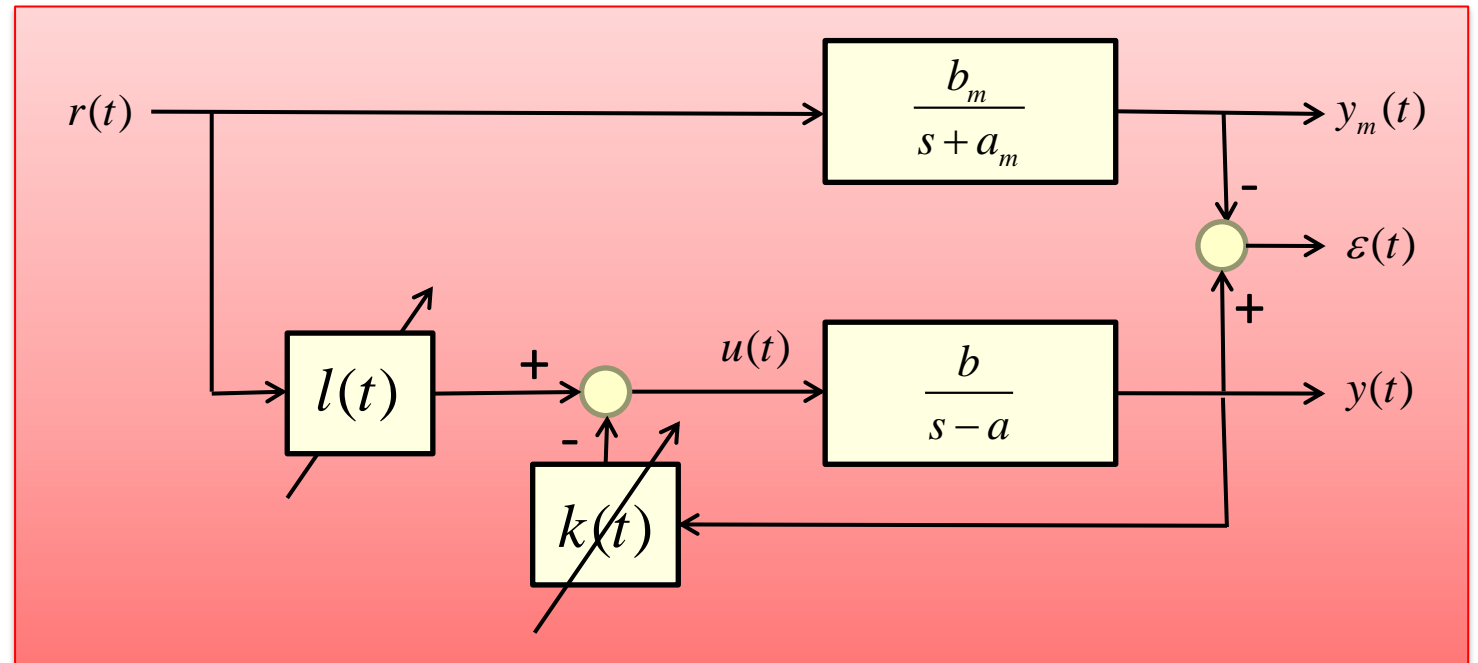
$$\varepsilon(t) = y(t) - y_m(t)$$

$$\dot{k}(t) = \gamma_1 \varepsilon(t) y(t) \operatorname{sgn}(b)$$

$$\dot{l}(t) = -\gamma_2 \varepsilon(t) r(t) \operatorname{sgn}(b)$$

$$\dot{y}(t) = (a - bk(t))y(t) + bl(t)r(t)$$

$$\dot{y}_m(t) = -a_m y_m(t) + b_m r(t)$$



Indirect Adaptive Control (Indirect MRAC)

If a, b were known, then we would use:

$$u = -k^* y(t) + l^* r(t) \quad \left\{ \begin{array}{l} k^* = \frac{a_m + a}{b} \\ l^* = \frac{b_m}{b} \end{array} \right.$$

Replace the unknown a, b by their estimates $\hat{a}(t), \hat{b}(t)$ where they are generated by identification techniques:

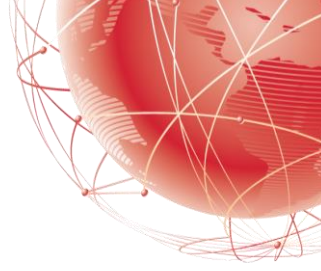
$$u(t) = -\frac{a_m + \hat{a}(t)}{\hat{b}(t)} y(t) + \frac{b_m}{\hat{b}(t)} r(t) \quad (\text{IAC})$$

$$u(t) = -k(t) y(t) + l(t) r(t) \quad (\text{DAC})$$

Need to make sure that: $\hat{b}(t) \neq 0$ (otherwise $u(t) \rightarrow \infty$)



Indirect Adaptive Control (Indirect MRAC)



Identification Model (Estimation Scheme):

$$\dot{\hat{y}}(t) = -a_m \hat{y}(t) + (a_m + \hat{a}(t))y(t) + \hat{b}(t)u(t)$$

Identification Error: $\varepsilon_i(t) = y(t) - \hat{y}(t)$

Adaptive Laws:

$$\dot{\hat{a}}(t) = \gamma_1 \varepsilon_i(t) y(t)$$

$$\dot{\hat{b}}(t) = \gamma_2 \varepsilon_i(t) u(t)$$

Indirect Adaptive Control (Indirect MRAC)



How do we guarantee that: $\hat{b}(t) \neq 0$

Instead of: $\dot{\hat{b}}(t) = \gamma_2 \varepsilon_i(t) u(t)$

Use:

$$\dot{\hat{b}}(t) = \begin{cases} \gamma_2 \varepsilon_i(t) u(t) & \text{if } \hat{b} > \bar{b} \\ \gamma_2 \varepsilon_i(t) u(t) & \text{if } \hat{b} = \bar{b} \text{ and } \gamma_2 \varepsilon_i u > 0 \\ 0 & \text{if } \hat{b} = \bar{b} \text{ and } \gamma_2 \varepsilon_i u \leq 0 \end{cases}$$

This will guarantee that $\hat{b}(t) \geq \bar{b}$ and also $u \in L_\infty$

Need to assume that we know the sign of b and also that we know a lower bound on b .

Nonlinear Systems (with unknown functions)



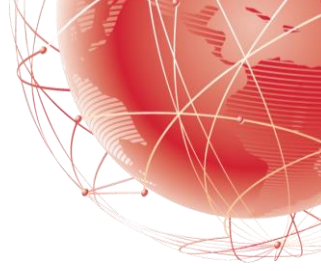
$$\dot{y} = f(y) + u$$

Objective: Choose an appropriate control law $u(t)$ such that all the signals in the closed-loop plant are bounded and the plant output $y(t)$ tracks the output $y_m(t)$ of the reference model

$$\dot{y}_m = -a_m y_m + b_m r$$

for all $r(t)$ that are bounded and piecewise continuous.

Nonlinear Systems (with unknown functions)



If we knew $f(y)$ then what would we do?

$$u = -f(y) - a_m y + b_m r$$

$$\dot{y} = f(y) - f(y) - a_m y + b_m r$$

$$\dot{y} = -a_m y + b_m r$$

$$\dot{y}_m = -a_m y_m + b_m r$$

$$\lim_{t \rightarrow \infty} \|y(t) - y_m(t)\| = 0$$

Exponentially fast

Learning Control with Neural Networks (indirect approach)

$$u = -\hat{f}(y, \hat{\theta}) - a_m y + b_m r$$

Identification Model (Estimation Scheme):

$$\dot{\hat{y}} = -a_m \hat{y} + a_m y + \hat{f}(y, \hat{\theta}) + u$$

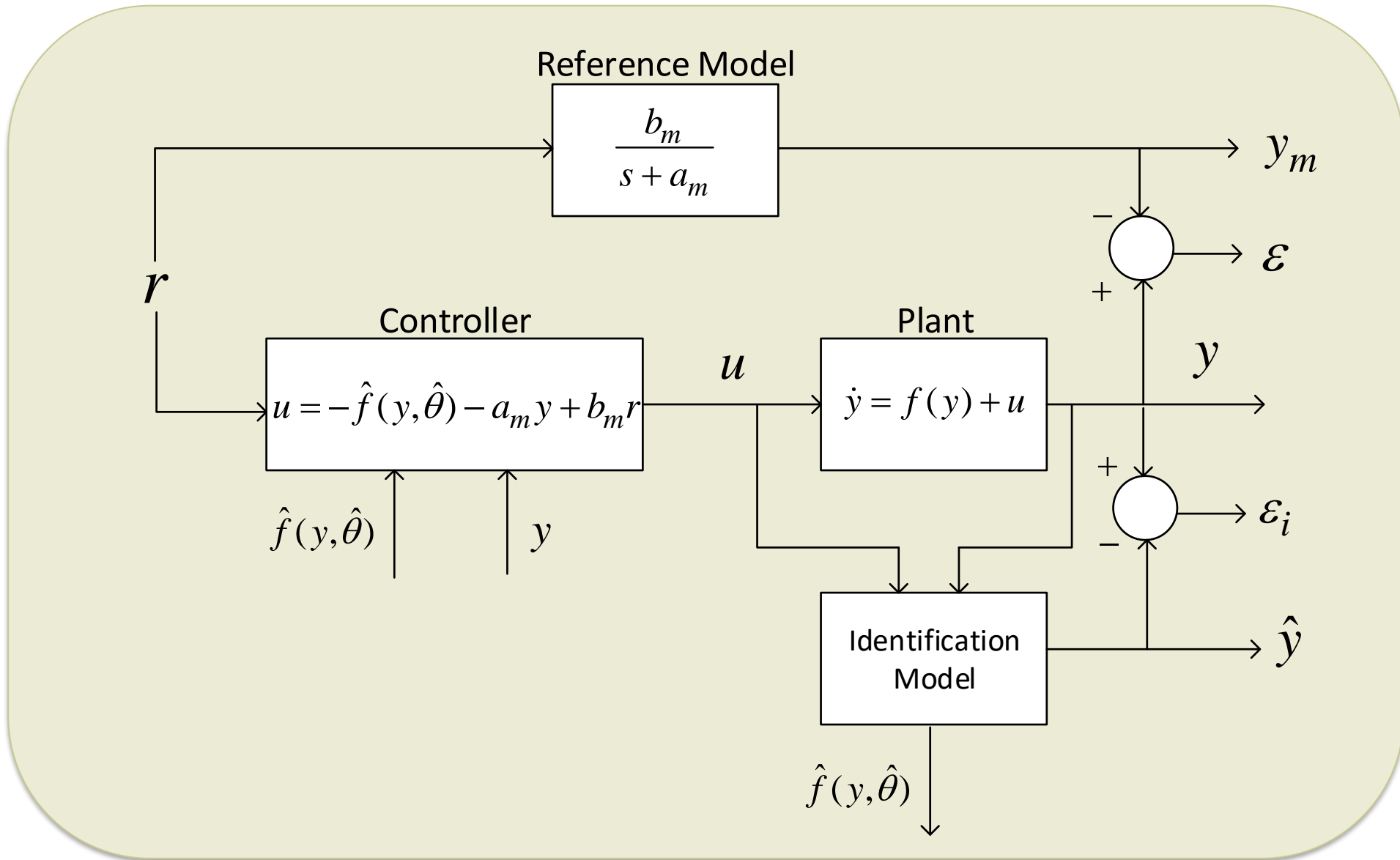
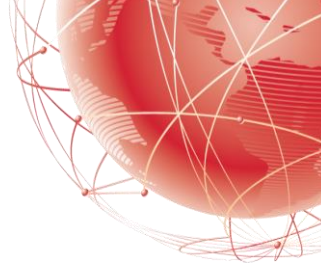
$$\dot{\hat{\theta}} = \Gamma Z^\top e_i$$

$$e_i = y - \hat{y}$$

$$Z = \frac{\partial \hat{f}(\hat{y}, \hat{\theta})}{\partial \hat{\theta}}$$



Learning Control with Neural Networks (indirect approach)



Learning Control with Neural Networks (indirect approach)



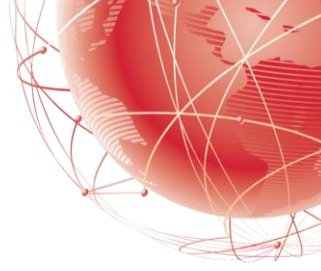
$$\dot{y} = f(y) - \underbrace{\hat{f}(y, \hat{\theta}) - a_m y + b_m r}_u, \quad y(0) = y_o$$

$$\dot{\hat{y}} = -a_m \hat{y} + a_m y + \hat{f}(y, \hat{\theta}) + u, \quad \hat{y}(0) = \hat{y}_o$$

$$\dot{y}_m = -a_m y + b_m r, \quad y_m(0) = y_m^o$$

$$\dot{\hat{\theta}} = \Gamma \frac{\partial \hat{f}}{\partial \hat{\theta}} \underbrace{(y - \hat{y})}_{e_i}, \quad \hat{\theta}(0) = \hat{\theta}^o$$

Discrete-time Learning Control



$$y(k+1) = f(y(k), y(k-1), \dots, y(k-n_y)) + u(k)$$

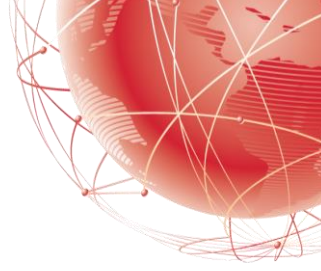
Objective: Choose an appropriate control law $u(k)$ such that all the signals in the closed-loop plant are bounded and the plant output $y(k)$ tracks the output $y_m(k)$ of the reference model $y_m(k+1) = -a_m y_m(k) + b_m r(k)$ for all $r(k)$ that are bounded.

f is unknown (or partially unknown)

$$\text{Let } z(k) = [y(k), y(k-1), \dots, y(k-n_y)]$$

$$\Rightarrow y(k+1) = f(z(k)) + u(k) \quad (z(k) \text{ is measurable})$$

Discrete-Time Learning Control



$$y(k + 1) = f(z(k)) + u(k)$$

$$\hat{y}(k + 1) = \hat{f}(z(k); \hat{\theta}(k)) + u(k)$$

← Prediction/Identification Model

$$u(k) = -\hat{f}(z(k); \hat{\theta}(k)) - a_m y_m(k) + b_m r(k)$$

← Learning Control Law

Adaptive Laws:

$$\hat{\theta}(k + 1) = \hat{\theta}(k) + \alpha_0 e(k + 1) \xi(k)$$

$$\alpha_0 > 0$$

Step size

$$0 < \gamma_0 < 2$$

Learning rate

$$\beta_0 > 0$$

Small design constant

$$\hat{\theta}(k + 1) = \hat{\theta}(k) + \frac{\gamma_0 e(k + 1)}{\beta_0 + |\xi(k)|^2} \xi(k)$$

$$\xi(k) = \frac{\partial \hat{f}}{\partial \hat{\theta}}(z(k), \hat{\theta}(k))$$

Network sensitivity function

↑ Normalized Gradient Descent

$$e(k) = y(k) - \hat{y}(k)$$

Estimation error

Discrete-Time Learning Control

Example: $y(k+1) = f(y(k), y(k-1)) + u(k)$

Given $y(0), y(-1), \hat{\theta}(0), y_m(0)$

\begin {for} $k = 0, 1, 2, \dots, N$

1. $z(k) = [y(k), y(k-1)]$

2. $u(k) = -\hat{f}(z(k), \hat{\theta}(k)) - a_m y_m(k) + b_m r(k)$

3. $\xi(k) = \frac{\partial \hat{f}}{\partial \hat{\theta}}(z(k), \hat{\theta}(k))$

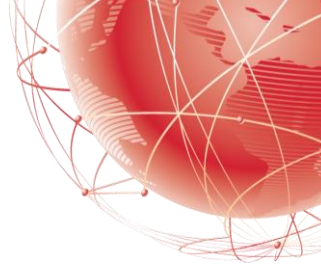
4. $y(k+1) = f(z(k)) + u(k)$

5. $y_m(k+1) = -a_m y_m(k) + b_m r(k)$

6. $\hat{y}(k+1) = \hat{f}(z(k), \hat{\theta}(k))$

7. $\hat{\theta}(k+1) = \hat{\theta}(k) + \frac{\gamma_0 (y(k+1) - \hat{y}(k+1))}{\beta_0 + |\xi(k)|^2} \xi(k)$

\end {for}



Some Bibliography on Monitoring and Control Using Learning Methods



- J.A. Farrell and M.M. Polycarpou, “*Adaptive Approximation Based Control*”, J. Wiley, 2006.
- M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, 2016.

Some Final Conclusions

- We have learned a lot
- A lot that we have not learned!
- The importance of gaining intuition
- Never lose the big picture and the applications

