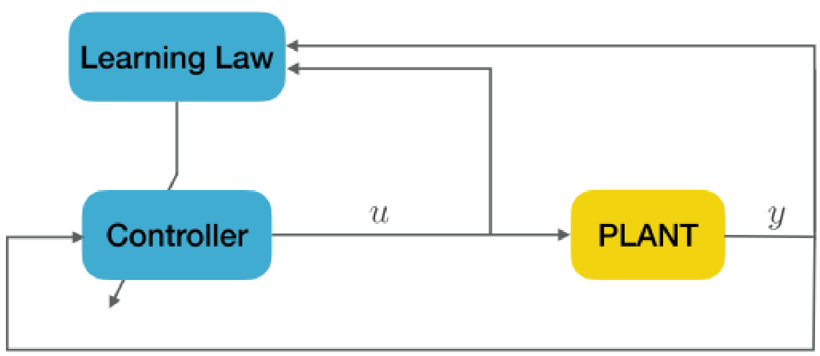
06_03_Passive FTC: Recent Advances in Adaptive Methods

1. Structure of Passive Fault Tolerant Control

A Passive, Adaptive FTC block diagram



Motivation

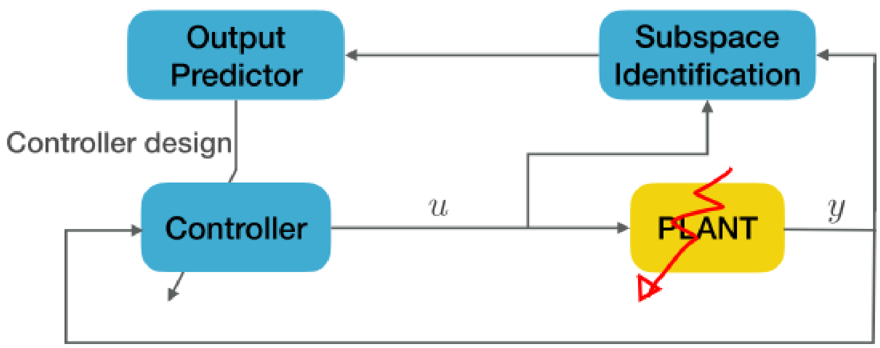
- **Active Approaches** need an **explicit** FDI decision and fault estimation (and of course model)
    - Fault Classes for FDI needs to be designed **by experts**
    - FDI adds **computational complexity'**
- **Passive adaptive approaches** do not need FDI (do not need model or automatically learn a model)
    - less computational complexity
    - work in healthy and faulty conditions, even for unknown faults
- **distinction between Active and Passive is blurry**

Example Methods

> **Examples**:

> > Neural Networks + Back Propagation
> > Direct/Indirect Adaptive Control
> > Polynomial/Wavelets etc. controllers
> > Reinforcement Learning
> > Active Inference
> > Other Machine Learning approaches
> > …

2. An Example of PFTC Method: Subspace Predictive FTC



Introduction

Subspace Predictive Fault-Tolerant Control = Subspace Identification + Output Predictor + Controller design law

- subspace-identification: learning model of the plan
    - Fault Happen → learn faulty model
- model-free, data-driven approach

Process

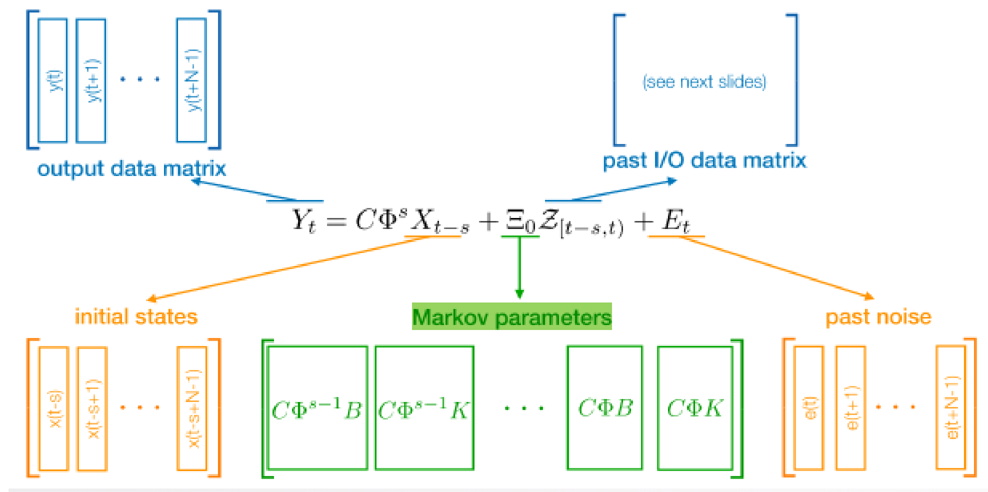
The following discrete-time **state space model** in innovation form is **assumed** (but not known!)

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + Ke(k) \\ y(k) = Cx(k) + e(k) \end{cases}$$

zero-mean white noise (innovation)

Assumption: $\Phi = A - KC$ is stable and the system is minimal

<u>Step 1: online recursive subspace identification</u>

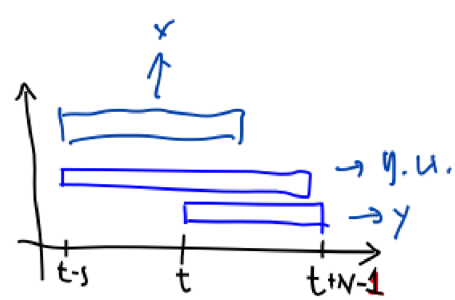


$$
\mathcal{Z}_{[t-s,t)} = \begin{bmatrix}
u(t-s) & u(t-s+1) & \cdots & u(t-s+N-1) \\
y(t-s) & y(t-s+1) & \cdots & y(t-s+N-1) \\
u(t-s+1) & u(t-s+2) & \cdots & u(t-s+N) \\
y(t-s+1) & y(t-s+2) & \cdots & y(t-s+N) \\
\vdots & \vdots & & \vdots \\
u(t-1) & u(t) & \cdots & u(t+N-2) \\
y(t-1) & y(t) & \cdots & y(t+N-2)
\end{bmatrix}
$$

- Initial states is unknown
- Past noise is unknown

Then we can estimate the system parameter by estimate the **Markov Parameters**(by solving a least square question)

$$\hat{\Xi}_0 = Y_t \cdot \mathcal{Z}^{+}_{[t-s,t)}$$



Illustration

$$
\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) \end{cases}
$$

$$
y(k) = C\left[A^{k-k_0} x\left(k_0\right) + + \sum_{h=k_0}^{k-1} A^{k-1-h} Bu(h)\right]
$$

Step 2: Output Predictor

- Future Outputs can be approximated by:

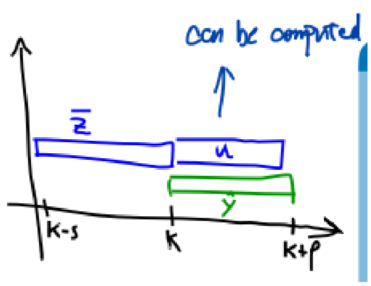
$$\hat{\mathbf{y}}_{[k,k+f)} \approx \begin{bmatrix} \Xi_0 \\ \Xi_1 \\ \vdots \\ \Xi_{f-1} \end{bmatrix} \bar{Z}_{[k-s,k)} + \begin{bmatrix} 0 & & & \\ \Psi_1 & 0 & & \\ \vdots & \vdots & \ddots & \\ \Psi_{f-1} & \Psi_{f-2} & \cdots & \Psi_1 \; 0 \end{bmatrix} \cdot \begin{bmatrix} u(k) \\ y(k) \\ \vdots \\ u(k+f-1) \\ y(k+f-1) \end{bmatrix} + \underbrace{\begin{bmatrix} C\Phi^{s} x(k-s) \\ C\Phi^{s+1} x(k-s) \\ \vdots \\ C\Phi^{s+f-1} x(k-s) \end{bmatrix}}_{\mathbf{b}_x}$$

$$\bar{Z}_{[k-s,k]} = \begin{bmatrix} u(k-s) \\ y(k-s) \\ \vdots \\ u(k-1) \\ y(k-1) \end{bmatrix}$$

$$\Psi_\tau \triangleq C\Phi^{\tau-1}[B\ K],\ \tau = 1, \cdots, f-1;$$

$$\Xi_i = [0_{l\times i(m+l)}\ C\Phi^{s-1}B\ C\Phi^{s-1}K\ \cdots C\Phi^{i}B\ C\Phi^{i}K]$$

  - However, actually we do not know the real u and y after timestamp k, but can be computed
  - The process can be illustrated in the following graph:



Step 3: Fault Tolerant Control Law

- Assume a fault occurs at time T_0
- The **faulty dynamics get learned** by the Subspace Identification (Step 1) and the predictor is updated to match them
- Then we can use a lot of method to design a controller
  - For example use MPC

Future reference $\quad \mathbf{r}_{[k+1,k+f)} = [r^T(k+1)\ r^T(k+2)\ \cdots\ r^T(k+f-1)]^T,$

Cost function $\quad J(k+1) = \|\mathbf{r}_{[k+1,k+f)} - \hat{\mathbf{y}}_{[k+1,k+f)}\|_Q^2 + \|\Delta \mathbf{u}_{[k,k+f-1)}\|_R^2,$

Input change $\quad \Delta u(k+i) = u(k+i) - u(k+i-1)$

Weighting matrices $\quad Q, R \succ 0$

$$S_{\Delta} = \begin{bmatrix} I_m & & & \\ -I_m & I_m & & \\ & \ddots & \ddots & \\ & & -I_m & I_m \end{bmatrix}, \quad S_{k-1} = \begin{bmatrix} 0_{m \times (s-1)(m+l)} & I_m & 0_{m \times l} \\ 0_{(f-2)m \times (s-1)(m+l)} & 0_{(f-2)m \times m} & 0_{(f-2)m \times l} \end{bmatrix}$$

$$\mathbf{u}_{[k,k+f-1)} = \arg \min_{\mathbf{u}_{[k,k+f-1)}} J(k+1),$$

$$\mathbf{u}^{*}_{[k,k+f-1)} = \left[\Lambda^T Q \Lambda + S_{\Delta}^T R S_{\Delta}\right]^{-1} \left[\Lambda^T Q \left(\mathbf{r}_{[k+1,k+f)} - \bar{\Gamma} Z_{[k-s,k)}\right) + S_{\Delta}^T R S_{k-1} Z_{[k-s,k)}\right]$$

Properties

<u>Benefits</u>

- no requirement for FDI
- works both during healthy and faulty conditions
- conventional control design laws such as LQR, MPC, LPV-based, etc. can be used

<u>Limitations</u>

- there is **no guarantee** of closed-loop stability
- adapting to a fault can be **slower** than in active approaches
  - **abrupt faults** are more difficult to accommodate quick enough

Summary

- Passive Fault Tolerant Control
  - No offline model needed
  - May have delay
- Subspace Predictive FTC
  - Learn model by **subspace identification**
  - **Predict** future output and future state
  - Use control law design control input based on learnt system model and prediction (**such as MPC controller**)