Parameter Estimation and Adaptive Control

Robert Stengel
Robotics and Intelligent Systems MAE 345,
Princeton University, 2017

- Parameter estimation
  - after the fact
  - real time
- Simultaneous Location and Mapping (SLAM)
- Reinforcement ("Q") learning
- Gain scheduling
- Adaptive critic (DHADP)
- Failure-tolerant control

Off-Line (i.e., "after the fact") Parameter Estimation

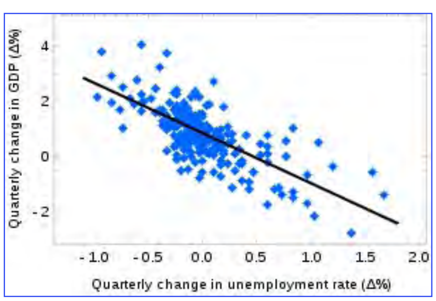
Parameter-Dependent Linear System

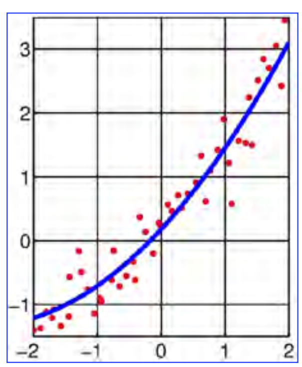
Linear systems contains parameters

$$\boxed{\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{\Phi}(\mathbf{p})\mathbf{x}_k + \mathbf{\Gamma}(\mathbf{p})\mathbf{u}_k \\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{n}_k \end{aligned}}$$

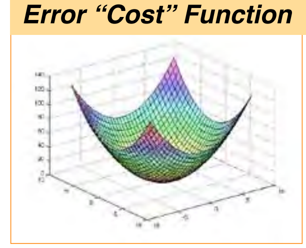
What if the parameter vector, p, is unknown?

Least-Square-Error Estimates of System Parameters





Error "Cost" Function



Trends and higher-degree curve-fitting
Multivariate estimation
Identification of dynamic system parameters

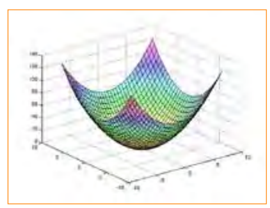
LTI System with Unknown Parameters

$$\boxed{\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{\Phi}(\mathbf{p})\mathbf{x}_k + \mathbf{\Gamma}(\mathbf{p})\mathbf{u}_k + \mathbf{\Lambda}(\mathbf{p})\mathbf{w}_k, \quad \mathbf{x}_0 \text{ given}\\ \mathbf{z}_k &= \mathbf{H}\mathbf{x}_k + \mathbf{n}_k, \quad k = 0, K\end{aligned}}$$

Parameters to be identified from experimental data, p

Known input, $\mathbf{u}_k$, noisy measurements, $\mathbf{x}_k$, made at discrete instants of time

Error Cost Function for Parameter Identification



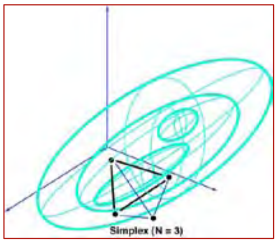
Weighted-square error of difference between measurements and model's estimates

$$J = \sum_{k=0}^{K} \boldsymbol{\varepsilon}_k^T \mathbf{R} \boldsymbol{\varepsilon}_k = \sum_{k=0}^{K} \left[\mathbf{z}_k - \hat{\mathbf{x}}_k\right]^T \mathbf{R}\left[\mathbf{z}_k - \hat{\mathbf{x}}_k\right]$$

$\mathbf{z}_k$: Measurement data set

$\hat{\mathbf{x}}_k$: Estimate propagated by sampled-data model

$\mathbf{R}$: Weighting matrix



Parameter Identification via Search

Error cost minimized by choice of p and x(0)

$$\min_{w.r.t.\,\mathbf{p},\mathbf{x}_0} J = \min_{w.r.t.\,\mathbf{p},\mathbf{x}_0} \sum_{k=0}^{K} \left[\mathbf{z}_k - \hat{\mathbf{x}}_k\right]^T \mathbf{R} \left[\mathbf{z}_k - \hat{\mathbf{x}}_k\right]$$

using search, e.g., Genetic Algorithm, Nelder-Mead (Downhill Simplex) algorithm [MATLAB's *fminsearch*], …

Extended Kalman Filter for Nonlinear State Estimation

Link to #20

Extended Kalman-Bucy Filter

Continuous-Time Nonlinear System

$$\dot{\mathbf{x}}(t) = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t)\right]$$
$$\mathbf{z}(t) = \mathbf{h}\left[\mathbf{x}(t)\right] + \mathbf{n}(t)$$

- **Propagate the state estimate using the continuous-time nonlinear model**
- **Update the state estimate using an optimal continuous-time linear correction in the nonlinear propagation**
- **Calculate optimal filter gain as in previous lecture and *OCE***

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}\left[\hat{\mathbf{x}}(t), \mathbf{u}(t)\right] + \mathbf{K}(t)\left\{\mathbf{z}(t) - \mathbf{h}\left[\hat{\mathbf{x}}(t)\right]\right\}$$

Hybrid Extended Kalman Filter

Numerical integration for state and covariance propagation

State Estimate (–)

$$\hat{\mathbf{x}}_k(-) = \hat{\mathbf{x}}_{k-1}(+) + \int_{t_{k-1}}^{t_k} \mathbf{f}\left[\hat{\mathbf{x}}(\tau), \mathbf{u}(\tau)\right] d\tau$$

Covariance Estimate (–)

$$\mathbf{P}_k(-)\left[t_k\right] = \mathbf{P}_{k-1}(+) + \int_{t_{k-1}}^{t_k} \left[\mathbf{F}(\tau)\mathbf{P}(\tau) + \mathbf{P}(\tau)\mathbf{F}^T(\tau) + \mathbf{L}(\tau)\mathbf{Q}'_C(\tau)\mathbf{L}^T(\tau)\right] d\tau$$

Jacobian matrices must be calculated

Hybrid Extended Kalman Filter

Incorporate measurements at discrete instants of time

Filter Gain

$$\mathbf{K}_k = \mathbf{P}_k(-)\mathbf{H}_k^T(t_k)\left[\mathbf{H}_k\mathbf{P}_k(-)\mathbf{H}_k^T + \mathbf{R}_{k-1}\right]^{-1}$$

State Estimate (+)

$$\hat{\mathbf{x}}_k(+) = \hat{\mathbf{x}}_k(-) + \mathbf{K}_k\left[\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k(-)\right]$$

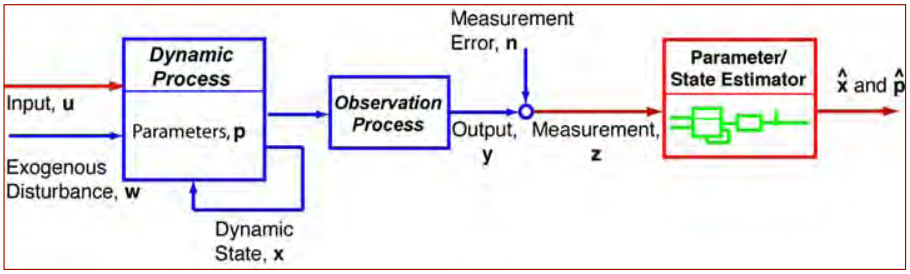
Covariance Estimate (+)

$$\mathbf{P}_k(+) = \left[\mathbf{I}_n - \mathbf{K}_k\mathbf{H}_k\right]\mathbf{P}_k(-)$$

On-Line (i.e., "real-time") Parameter Estimation

Parameter Identification Using an Extended Kalman-Bucy Filter

Augment state to include the parameter



Extend the dynamic model to account for the parameter

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{p}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}_\mathbf{x}\left[\mathbf{x}(t),\mathbf{p}(t),\mathbf{u}(t),\mathbf{w}_\mathbf{x}(t)\right] \\ \mathbf{f}_\mathbf{p}\left[\mathbf{p}(t),\mathbf{w}_\mathbf{p}(t)\right] \end{bmatrix}; \quad \mathbf{z} = \mathbf{h}\left[\mathbf{x}(t)\right] + \mathbf{n}(t)$$

Parameter Vector Must Have a Dynamic Model

Several alternatives

Unknown constant parameter: p(*t*) = constant

$$\dot{\mathbf{p}}(t) = \mathbf{f}_\mathbf{p}\left[\mathbf{p}(t),\mathbf{w}_\mathbf{p}(t)\right] \triangleq \mathbf{0}; \quad \mathbf{p}(0) = \mathbf{p}_o; \quad \mathbf{P}_\mathbf{p}(0) = \mathbf{P}_{\mathbf{p}_o}$$

Random parameter: p(*t*) = Integrated white noise

$$\dot{\mathbf{p}}(t) = \mathbf{f}_\mathbf{p}\left[\mathbf{p}(t),\mathbf{w}_\mathbf{p}(t)\right] \triangleq \mathbf{w}_\mathbf{p}(t); \quad \mathbf{p}(0) = \mathbf{p}_o; \quad \mathbf{P}_\mathbf{p}(0) = \mathbf{P}_{\mathbf{p}_o}$$

$$E\left[\mathbf{w}_\mathbf{p}(t)\right] = \mathbf{0}; \quad E\left[\mathbf{w}_\mathbf{p}(t)\mathbf{w}_\mathbf{p}^T(\tau)\right] = \mathbf{Q}_\mathbf{p}\delta(t-\tau)$$

Dynamic Models for the Parameter Vector

Random parameter: p(*t*) = Integral of integrated white noise

$$\dot{\mathbf{p}}_M(t) = \begin{bmatrix} \dot{\mathbf{p}}(t) \\ \dot{\mathbf{p}}_D(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{p}_D(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{w}_\mathbf{p}(t) \end{bmatrix}$$

Parameter vector
Parameter rate of change

Random parameter: p(*t*) = Double integral of integrated white noise

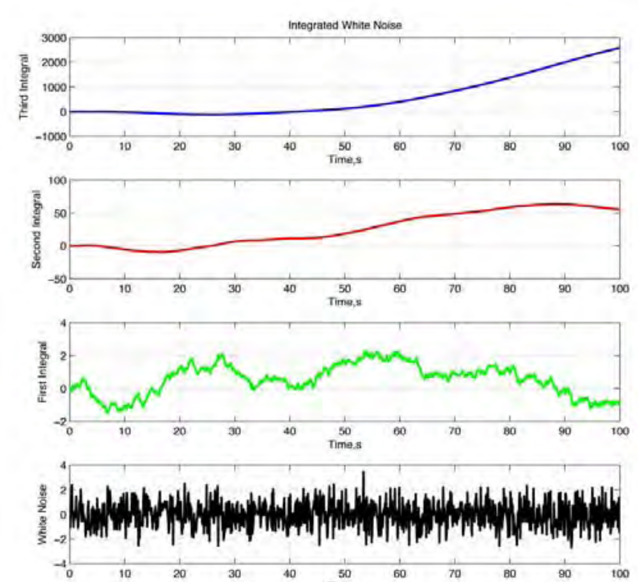
$$\dot{\mathbf{p}}_M(t) = \begin{bmatrix} \dot{\mathbf{p}}(t) \\ \dot{\mathbf{p}}_D(t) \\ \dot{\mathbf{p}}_A(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p}(t) \\ \mathbf{p}_D(t) \\ \mathbf{p}_A(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{w}_\mathbf{p}(t) \end{bmatrix}$$

Parameter vector
Parameter rate of change
Parameter acceleration

Number of parameters and derivatives to be estimated is doubled or tripled

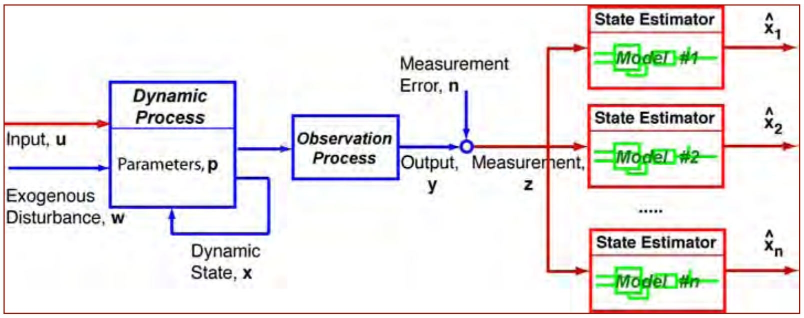
Integrated White Noise Models of a Parameter

- **Third integral models slowly varying, smooth parameter**
- **Second integral is smoother but still has fast changes**
- **First integral of white noise has abrupt jumps, valleys, and peaks**
- **White noise**



Multiple-Model Testing for System Identification

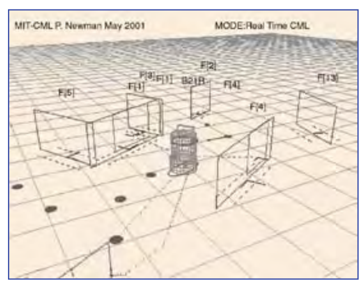
Create a bank of Kalman Filters, one for each hypothetical model, $n = 1, N$



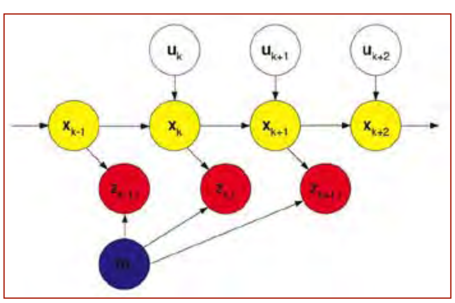
Choose model with minimum error residual

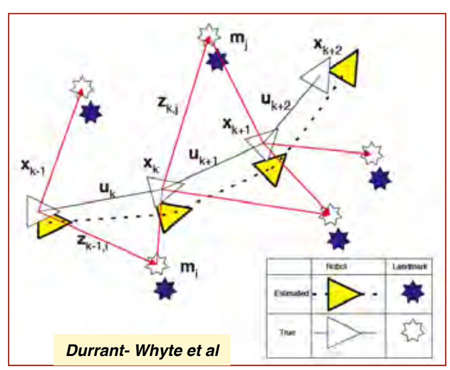
$$J_n = \sum_{k'=k-k_o}^{k} \boldsymbol{\varepsilon}_{n_{k'}}^T \mathbf{R} \boldsymbol{\varepsilon}_{n_{k'}} = \sum_{k'=k-k_o}^{k} \left[\mathbf{z}_{n_{k'}} - \hat{\mathbf{x}}_{n_{k'}} \right]^T \mathbf{R} \left[\mathbf{z}_{n_{k'}} - \hat{\mathbf{x}}_{n_{k'}} \right]$$

Simultaneous Location and Mapping (SLAM)

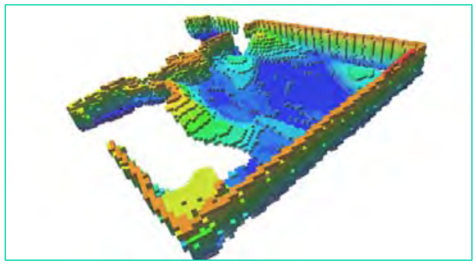


- Build or update a local map within an unknown environment
  - Stochastic map, defined by mean and covariance of many points
  - SLAM Algorithm = State estimation with <u>bank</u> of extended Kalman filters, a form of particle filter
  - Landmark and terrain tracking
  - Multi-sensor integration





SLAM with Ultrasound SONAR, LIDAR, or RADAR

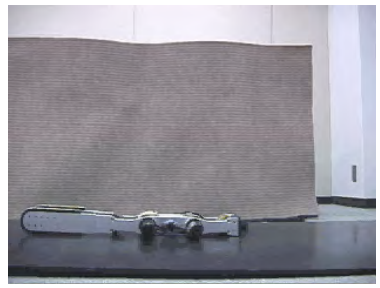


UW-RSE Lab

Adaptive Control

Reinforcement ("Q") Learning

- **Learn from success and failure**
- **Repetitive trials**
  - Reward correct behavior
  - Penalize incorrect behavior
- **Learn to control from a human operator**



http://en.wikipedia.org/wiki/Reinforcement_learning

Adaptive Control System Design

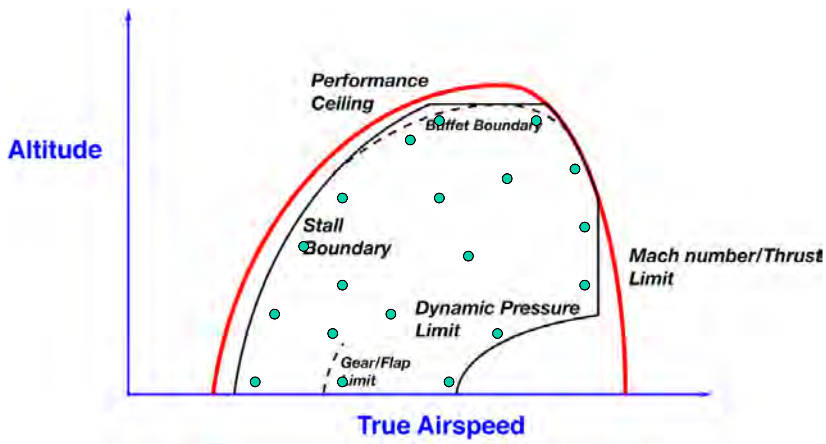
- **Control logic changes to accommodate changes or unknown parameters of the plant**
  - System identification to improve state estimate
  - Gain scheduling to account for environmental change
  - Adaptive Critic (Dual Heuristic Adaptive Dynamic Programming)
  - Learning systems that track performance metrics (e.g., CMAC)
  - Reinforcement learning
- **Control law is nonlinear**

$$\boxed{\mathbf{u}(t) = \mathbf{c}\left[\mathbf{z}(t), \mathbf{a}, \mathbf{y}^*(t)\right]}$$

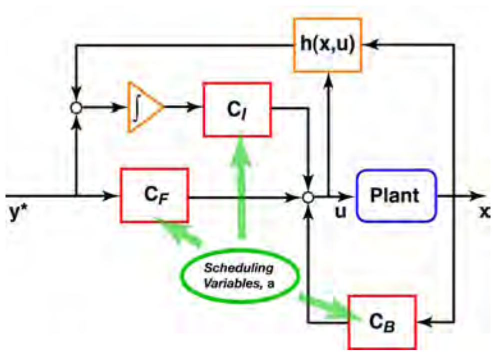
$\mathbf{c}[\bullet]$:	Control law
$\mathbf{x}(t)$:	State
$\mathbf{z}[\mathbf{x}(t)]$:	Measurement of state
$\mathbf{a}$:	Control law parameters
$\mathbf{y}^*(t)$:	Command input

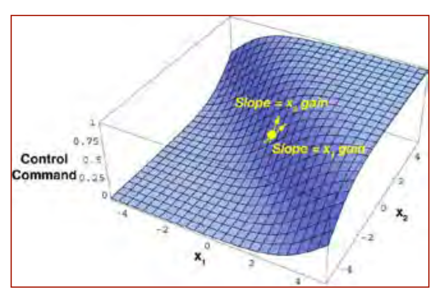
Operating Points Within a Flight Envelope

Dynamic model is a function of altitude and airspeed
Design LTI controllers throughout the flight envelope



Gain Scheduling



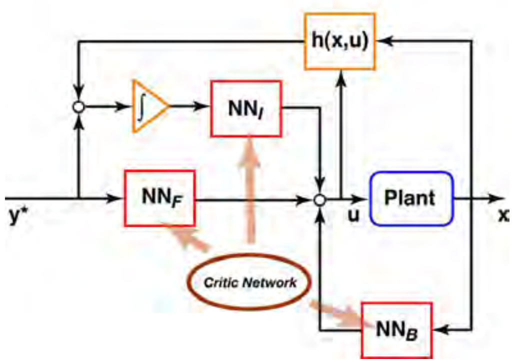


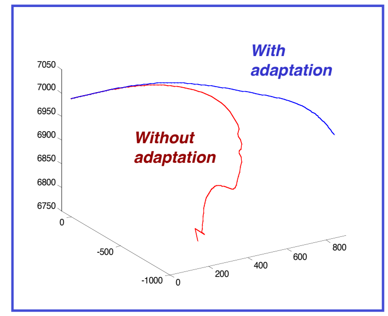
Proportional-integral controller with scheduled gains

$$\mathbf{u}(t) = C_F(\mathbf{a})\mathbf{y}^* + C_B(\mathbf{a})\Delta\mathbf{x} + C_I(\mathbf{a})\int \Delta\mathbf{y}(t)\,dt$$
$$\approx \mathbf{c}\left[\mathbf{x}(t), \mathbf{a}, \mathbf{y}^*(t)\right]$$

Scheduling variables, a, are "slow", *e.g.*, altitude, speed, properties of chemical process, …

Adaptive Critic Neural Network Controller

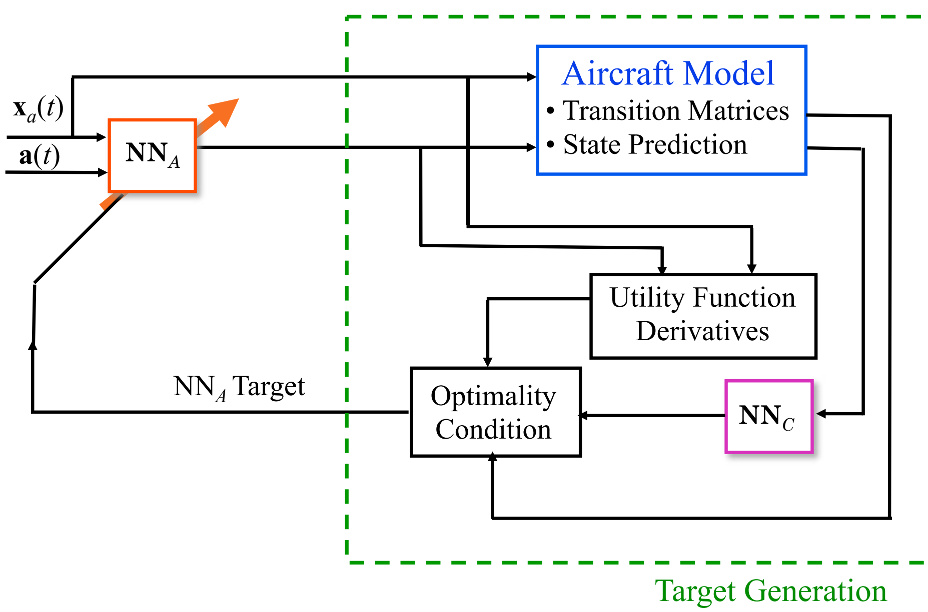




- On-line adaptive critic controller
  - Replace gain matrices by neural networks *(see Lecture 19)*
  - Nonlinear control law implemented as "action network"
  - Performance and control usage evaluated via "critic network"
  - Control network weights adapted to improve performance
  - Cost model adapted to improve critique

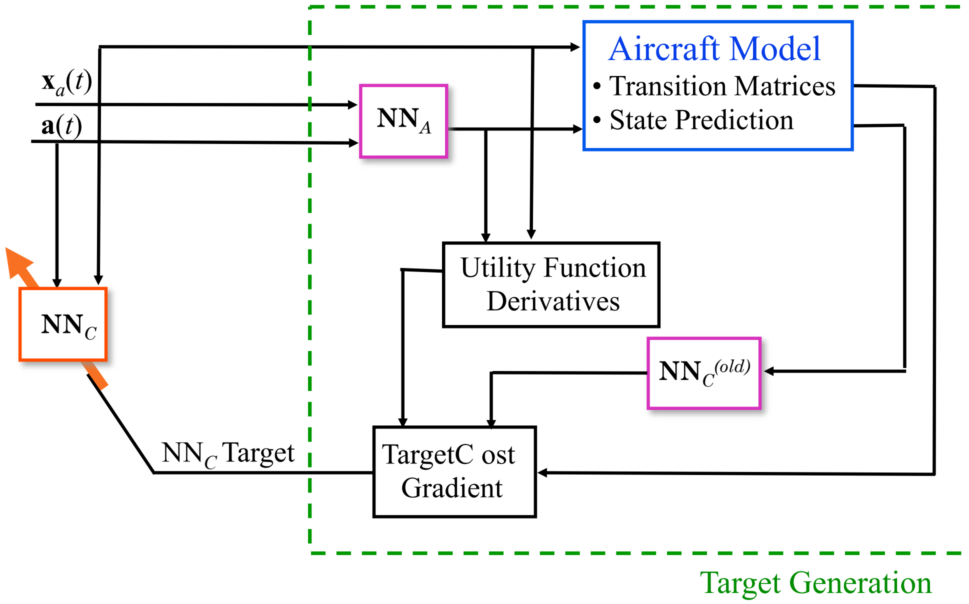
Action Network On-line Training

Train action network, at time *t*, holding the critic parameters fixed



Critic Network On-line Training

Train critic network, at time *t*, holding the action parameters fixed



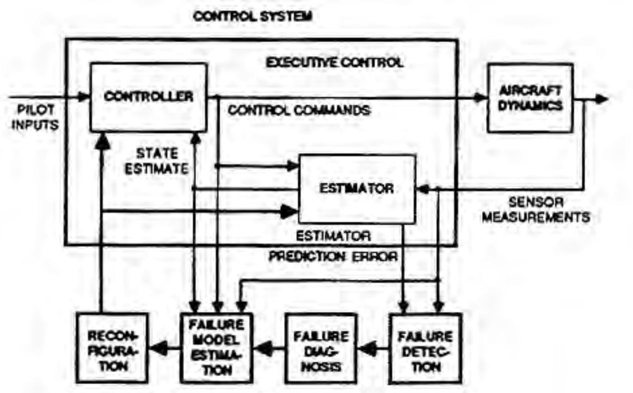
Real-Time Implementation of Rule-Based Control System

Control system knowledge-base contents

Task	Parameters	Rules	Major subtasks
Executive control	18	23	Kalman filter and linear-quadratic regulator
Failure detection	9	15	Normalized innovations monitor
Failure diagnosis	135	147	Signal dependency search
Failure model estimation	15	23	Multiple-model algorithm
Reconfiguration	32	39	Weighted left pseudoinverse

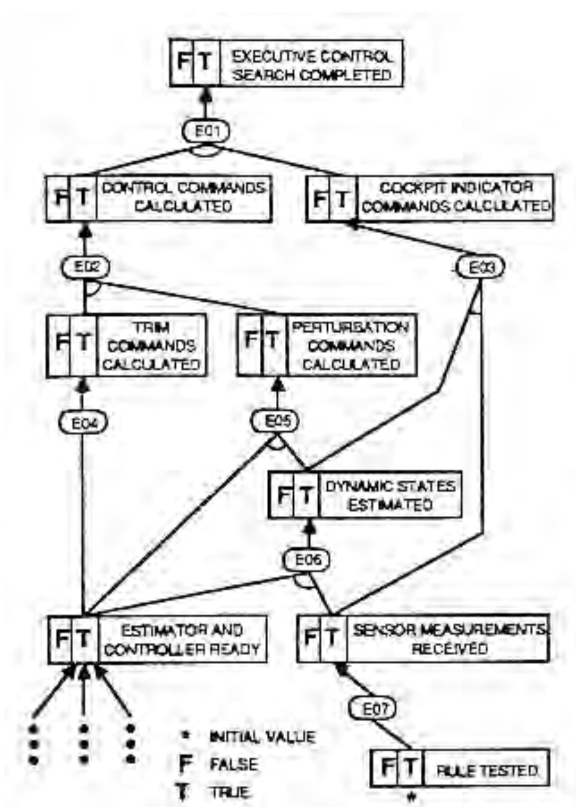
Rule-Based Control System

(Handelman and Stengel, 1989)



Application: **Failure-tolerant flight control for *CH-47 Chinook* helicopter**

Control is a side effect from expert system perspective



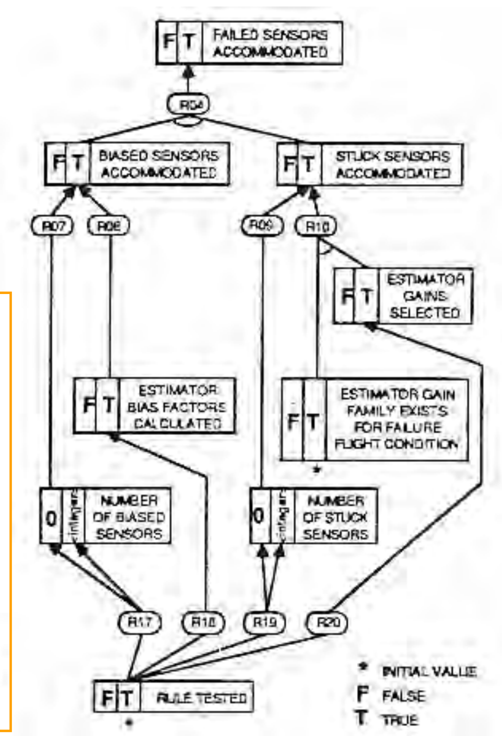
Rule-Based Control Logic

- **Search until root node is solved**
  - **Initiates lower-level functions to declare leaf node is TRUE**

Rule-Based Reconfiguration Logic

Example of a Failure-Diagnosis Rule

Rule-I41:
IF control failure candidates are determined
AND forward collective pitch control is a candidate
AND the largest element of the normalized innovations rms is pitch rate
AND the ratio of pitch rate (rad/s) to vertical velocity (m/s) normalized innovations rms is within 10% of 6.01
THEN hypothesize forward collective pitch control stuck at sign (pitch rate innovations average) × [(35.8 × pitch rate innovations rms) − 1.48] cm at (−2.85 × pitch rate innovations rms) + 0.890 s prior to failure detection

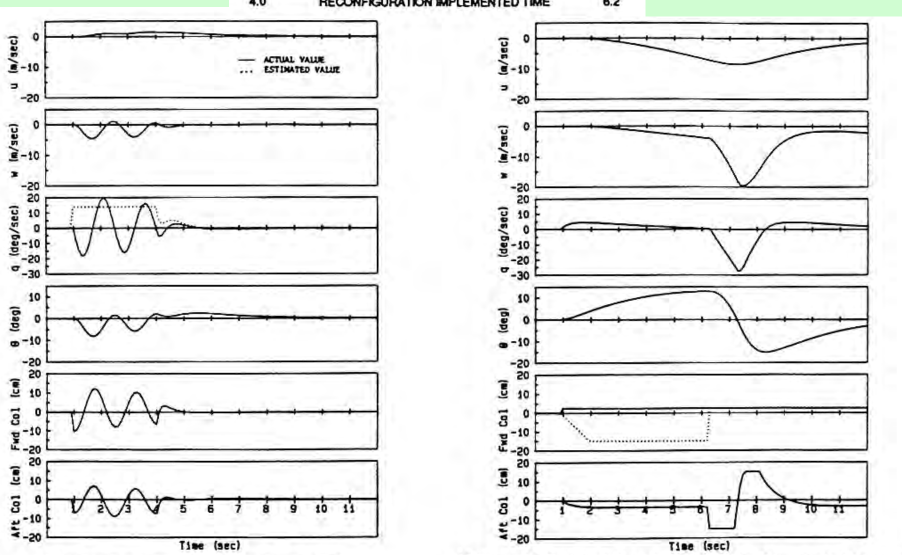


Failure Response

Response to Stuck Pitch-Rate Sensor

1.0	FAILURE TIME	1.0
1.2	FAILURE DETECTED TIME	1.8
1.9	FAILURE DIAGNOSED TIME	2.4
3.8	FAILURE MODEL ESTIMATED TIME	6.0
4.0	RECONFIGURATION IMPLEMENTED TIME	6.2

Response to Stuck Forward-Collective Pitch Actuator

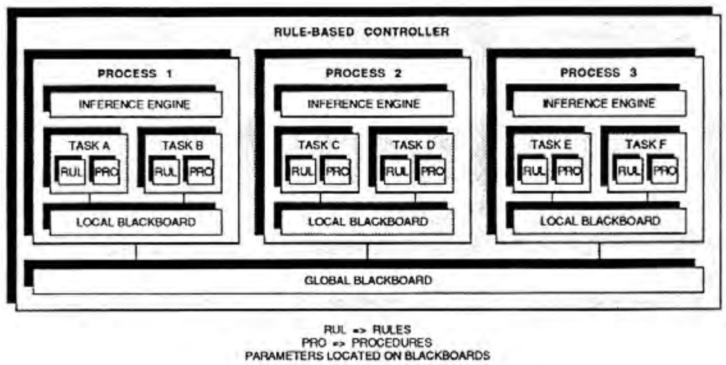


a) Pitch rate sensor stuck 14 deg/s from nominal

b) Forward collective pitch control stuck 2.5 cm from nominal (controls saturate at ± 15 cm)

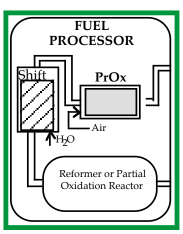
Real-Time Implementation of Rule-Based Control System

- Original code written in LISP
- Automatic procedural code generation (LISP to Pascal)
- Real-time execution on three *i386* processors in *Multibus™* architecture
- External PC used for code development, testing, and helicopter simulation



Next Time: Task Planning and Multi-Agent Systems

Supplementary Material



Preferential Oxidizer (PrOx)

- **Proton-Exchange Membrane Fuel Cell converts hydrogen and oxygen to water and electrical power**
- **Steam Reformer/Partial Oxidizer-Shift Reactor converts fuel (e.g., alcohol or gasoline) to H_2, CO_2, H_2O, and CO. Fuel flow rate is proportional to power demand**

- **CO "poisons" the fuel cell and must be removed from the reformate**
- **Catalyst promotes oxidation of CO to CO_2 over oxidation of H_2 in a Preferential Oxidizer (PrOx)**
- **PrOx reactions are nonlinear functions of catalyst, reformate composition, temperature, and air flow**

Reinforcement ("Q") Learning Control of a Markov Process

- Q: Quality of a state-action function
- Heuristic value function
- <u>One-step philosophy</u> for heuristic optimization

$$Q\left[\mathbf{x}(t_{k+1}),\mathbf{u}(t_{k+1})\right]=Q\left[\mathbf{x}(t_k),\mathbf{u}(t_k)\right]+\alpha(t_k)\left\{\left[L_{\mathbf{u}(t_k)}\left[\mathbf{x}(t_k)\right]+\gamma(t_k)\max_{\mathbf{u}} Q\left[\mathbf{x}(t_{k+1}),\mathbf{u}\right]\right]-Q\left[\mathbf{x}(t_k),\mathbf{u}(t_k)\right]\right\}$$

$$\alpha(t_k): \text{ learning rate}, 0<\alpha(t_k)<1$$

- Various algorithms for computing best control value

$$\mathbf{u}_{best}(t_k)=\arg\max_{\mathbf{u}} Q\left[\mathbf{x}(t_k),\mathbf{u}\right]$$

Q-Learning Snail
https://www.youtube.com/watch?v=UbwlPDaMlvY

Q-Learning, Ball on Plate
https://www.youtube.com/watch?v=04MLqINZwHY&feature=related

Q Learning Control of a Markov Process is Analogous to LQG Control in the LTI Case

$$Q\left[\mathbf{x}(t_{k+1}),\mathbf{u}(t_{k+1})\right]=Q\left[\mathbf{x}(t_k),\mathbf{u}(t_k)\right]+\alpha(t_k)\left\{\left[L_{\mathbf{u}(t_k)}\left[\mathbf{x}(t_k)\right]+\gamma(t_k)\max_{\mathbf{u}} Q\left[\mathbf{x}(t_{k+1}),\mathbf{u}\right]\right]-Q\left[\mathbf{x}(t_k),\mathbf{u}(t_k)\right]\right\}$$

$$\alpha(t_k): \text{ learning rate}, 0<\alpha(t_k)<1$$

Controller

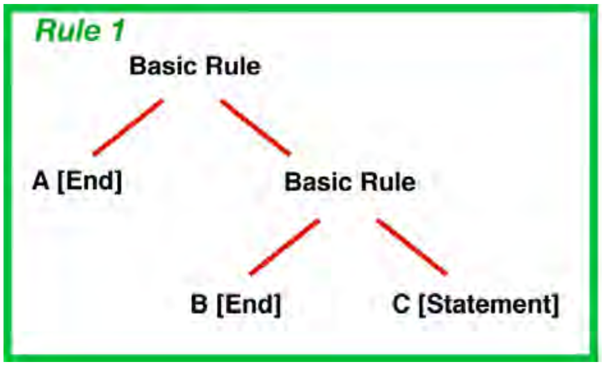
$$\mathbf{x}_{k+1}=\mathbf{\Phi}\mathbf{x}_k+\mathbf{\Gamma}\mathbf{C}\left(\hat{\mathbf{x}}_k-\mathbf{x}_k{}^*\right)$$

Estimator

$$\hat{\mathbf{x}}_k=\mathbf{\Phi}\hat{\mathbf{x}}_{k-1}-\mathbf{\Gamma}\mathbf{C}\left(\hat{\mathbf{x}}_{k-1}-\mathbf{x}_{k-1}{}^*\right)+\mathbf{K}\left\{\mathbf{z}_k-\mathbf{H}_{\mathbf{x}}\left[\mathbf{\Phi}\hat{\mathbf{x}}_{k-1}-\mathbf{\Gamma}\mathbf{C}\left(\hat{\mathbf{x}}_{k-1}-\mathbf{x}_{k-1}{}^*\right)\right]\right\}$$

More on Rules

- Example of a pre-formed compound rule



Rule1 Rule1(A,B,C)
A
B
C

- Once rule is defined, it has a fixed, ordered frame or argument list

- Side effects: Actions triggered by inference
  - If A = TRUE, … but what is A?
  - Execute a function to find out, and return to the rule
  - … then B = C, … but what is C?
  - Execute a function …