Fault-tolerant control includes real-time fault diagnosis and take actions to avoid severe consequences of faults. Software makes “intelligent” assessment and remedial actions. The need is evident in many areas of application. Autonomous systems clearly need fault-tolerant methods. Ad hoc methods have been practiced for decades, a systematic methodology has evolved during the last.
Areas of particular interest

Fault-tolerant control

Created by the need for safety, reliability and availability.

Touches on fundamental properties of controlled systems and the control architectures.

Diagnosis

The essential means to determine whether faults have occurred and thereafter locate their cause

Material for Socrates Lectures

This short course is based on the book:

*Diagnosis and Fault-tolerant Control.*

By M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki.


And on

*SaTool Users Manual by T. Lorentzen and M. Blanke, Oersted•DTU, Technical University of Denmark, April 2004.*
The system can be changed by events, some of which are faults. We may change the controller and/or the system by action of a supervisor.

A bathtub is a hybrid system. Dynamic relation between input flow $q$, height $h$ and area $A(h)$ are

$$\dot{h}(t) = \frac{1}{A(h)}q(t)$$

Holds only until the edge is reached:

$$\dot{h}(t) = 0 \text{ for } h = h_{\text{edge}}$$
A Hybrid System

Dynamic relation between input flow $q$, height $h$ and area $A(h)$ are:

- For $0 \leq h < h_{\text{edge}}$: 
  \[ h(t) = \frac{1}{A(h)} q(t) \]
- For $h \geq h_{\text{edge}}$: 
  \[ h(t) = 0 \]

Fault-accommodation

Fault accommodation

Change control parameters or structure to avoid the consequences of a fault.

Input-output between controller and plant is unchanged.
Handling of fault - reconfiguration

- Fault reconfiguration: a sensor failure in inner loop.
- Switch to differentiating control when fault is diagnosed

Handling of sensor fault by estimation

Replace faulty measurement by estimated value
Sensor fusion subject to faults

Simple sensor fusion
\[ \hat{y}(t) = S y_{valid}(t) + (1 - S) \hat{y}(t) \]

- Faulty sensor measurement is replaced by an estimate, which is used in the feedback loop

Diagnosis and behaviours
Fault-tolerant versus safety problem

Running example: ship steering

\[ \dot{\psi}_b(t) = b(\delta(t) + H(\omega_2(t))) \]
\[ \psi(t) = \psi_1(t) + \omega_{m1}(t) \]
\[ \psi_{m1}(t) = \psi(t) + f_{\psi 1}(t) \]
\[ \psi_{m2}(t) = \psi(t) + f_{\psi 2}(t) \]
\[ \omega_{m}(t) = \omega_2(t) + \omega_{n}(t) + f_{\omega}(t) \]
Component-based Analysis

Chapter 4 in Diagnosis and Fault-tolerant Control
Generic component model for a process tank (batch process)

Use-modes:
UM0: No operation
UM1: Filling the tank
UM2: Processing batch
UM3: Emptying via normal pipe
UM4: Empty via waste outlet
UM5: Cleaning the tank

Analysis based on structure

Sections 5.1 – 5.3, pp 99-122
This is fun

Design Sequence Overview

M. Blanke 2004
Diagnosis & Fault-tolerant Control

Digraph for linear system

**Example 5.2**

Graph represented as the incidence matrix:

\[
\begin{pmatrix}
1 & u & x_1 & x_2 & \dot{x}_1 & \dot{x}_2 \\
0 & c_1 & 0 & 0 & 1 & 0 \\
c_2 & 0 & 0 & 1 & 1 & 0 \\
c_3 & 0 & 0 & 1 & 0 & 1 \\
c_4 & 1 & 1 & 1 & 0 & 1
\end{pmatrix}
\]

\[c_1 \cdot \dot{x}_1 = \frac{d}{dt} x_1\]

\[c_2 \cdot \dot{x}_1 = a x_2\]

\[c_3 \cdot \dot{x}_2 = \frac{d}{dt} x_2\]

\[c_4 \cdot \dot{x}_2 = b x_1 + c x_2 + d u\]
Bipartite graph

Graph represented as incidence matrix:

\[
\begin{array}{cccc}
\mathbf{u} & x_1 & x_2 & \dot{x}_1 & \dot{x}_2 \\
c_1 & 0 & 1 & 0 & 1 & 0 \\
c_2 & 0 & 0 & 1 & 1 & 0 \\
c_3 & 0 & 0 & 1 & 0 & 1 \\
c_4 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

Or drawn as a bipartite graph:

"A simple graph is one with no parallel edges and loops. A bipartite graph is a simple graph in which the set of vertices can be partitioned into two sets such that every edge is between a vertex in C and a vertex in Z: \( G = (C, Z, E) \)."

Structure Graph

Structural analysis

A graph-based technique where principal relations between variables express the system's properties. Measured and controlled quantities in the system are related to variables through functional relations, which need not be explicitly stated. The user specifies a list of these relations that together describe the functionality of the system considered. A list of such variables and functional relations constitute the system's structure graph.

A Fault is a Violation of a Constraint

Faults
Normal operation means all functional relations are intact for the system. Should faults occur, one or more functional relations cease to be valid. In the structure graph, one or more nodes of the graph will disappear when a fault occurs.

SaTool is an implementation of structural analysis theory. It will analyze a structure graph and provide knowledge about fundamental properties of the system in normal and faulty conditions.

Single tank system example 5.3

\[ c_1 : \dot{h}(t) = q(t) - q_i(t) \]
\[ c_2 : q_i(t) = au(t) \]
\[ c_3 : q_i(t) = k \sqrt{h(t)} \]
\[ c_4 : y(t) = h(t) \]
\[ c_5 : u(t) = \begin{cases} 1 & \text{if } y(t) < h_0 \\ 0 & \text{otherwise} \end{cases} \]
\[ c_6 : \dot{h}(t) = \frac{dh(t)}{dt} \]

Three views

\[
\begin{bmatrix}
    u & y & h & h & q_i & q_o \\
    c_1 & 0 & 0 & 0 & 1 & 1 \\
    c_2 & 1 & 0 & 0 & 0 & 1 \\
    c_3 & 0 & 0 & 1 & 0 & 0 \\
    c_4 & 0 & 1 & 0 & 0 & 0 \\
    c_5 & \\
    c_6 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]
Example 5.3: tank system

Every column in the incidence matrix corresponds to a circle vertex. Every row corresponds to a bar vertex in the bipartite graph.

Complete matching

A matching $M$ is a subset of edges such that no edge have common node (neither in $C$ nor in $X$). Let $|M|$ be number of edges in $M$, then $|M| \leq \min(|C|,|X|)$

A matching is complete in $X$ if $|M| = X$.
It is complete in $C$ if $|M| = |C|$.

Figure: One incomplete and two complete matchings in $X$ for the tank example.
Unmatched constraint

The ranking algorithm to find matchings

1. Mark all known variables
2. Find all constraints with one unmatched variable. Mark these and the corresponding variables.
3. If there exist unmarked constraints with all variables marked, flag these.
4. Continue

The principle of a matching algorithm: start with a known variable. Successively calculate unknown variables using the constraints.
Non invertible constraints

\[ x_2 = c(x_1) \]

Differential and integral constraints

Differential constraint
\[ c_6 : \dot{h} = \frac{dh}{dt} \]

Integral constraint
\[ c_6 : h(t) = \int_0^t \dot{h}(\tau) d\tau + h(0) \]
Development cycle using SaTool

SaTool supports:
- Define system structure
- Analyse structure graph
- Show results
- Modify system
- Load/save models
- Save/print results

SaTool – A tool for Structural Analysis

Features
- Matlab® based (version 6.5 or later)
- Graphical representation of structure as a graph
- Easy point and click manipulation of system structure
- Matching by ranking
- Backtracing
  - Recursive algorithm

Matlab® is a trademark of the Math Works Inc. USA.
SaTool - enter system

SaTool
(ref: Lorentzen and Blanke, 2004)
Analyse system structure
Automatic generation of parity relations for diagnosis

Formulation for structural analysis

Constraints for ship steering example

\begin{align*}
  c_1 & : \dot{\omega}_s(t) = b(\delta(t) + H(\omega_s(t))) \\
  c_2 & : \dot{\psi}(t) = \omega_h(t) + \omega_w(t) \\
  c_3 & : \dot{\psi}_{m1}(t) = \psi(t) \\
  c_4 & : \dot{\psi}_{m2}(t) = \psi(t) \\
  c_5 & : \omega_{h_m}(t) = \omega_3(t) + \omega_w(t) \\
  c_6 & : \frac{d\omega_3}{dt} = \dot{\omega}_s(t) \\
  c_7 & : \frac{d\psi}{dt} = \dot{\psi}(t)
\end{align*}

Known variables
\begin{align*}
  \delta & \text{ rudder angle} \\
  \psi_{m1} & \text{ heading gyro 1} \\
  \psi_{m2} & \text{ heading gyro 2} \\
  \omega_{h_m} & \text{ turn rate gyro} \\
  H & \text{ a non-linear function of } \omega_3
\end{align*}

Unknown variables
\begin{align*}
  \omega_3 & \text{ physical turn rate} \\
  \psi & \text{ physical heading} \\
  \omega_w & \text{ disturbance from waves}
\end{align*}

Known variables
\begin{align*}
  \delta & \text{ rudder angle} \\
  \psi_{m1} & \text{ heading gyro 1} \\
  \psi_{m2} & \text{ heading gyro 2} \\
  \omega_{h_m} & \text{ turn rate gyro} \\
  H & \text{ a non-linear function of } \omega_3
\end{align*}

Unknown variables
\begin{align*}
  \omega_3 & \text{ physical turn rate} \\
  \psi & \text{ physical heading} \\
  \omega_w & \text{ disturbance from waves}
\end{align*}
The Constraint Editor

Graphical and text based output.

Shows fundamental properties:
- Which faults can not be detected
- Which faults can not be isolated
- Which parity relations need be used to diagnose a particular fault
- What can be done if a particular constraint is violated (fault occurred)
View the results (1)

- Select a parity equation from the dropdown menu

- Result
  - Parity Relation
  - Included variables yellow
Viewing the results

- Press **Results:**
  - The matlab editor opens and shows the results in a text file

Results

- **Reachability**
  - Can I reach the unknown variables from the known variables
  - Requirement for observability
- **Controllability**
  - Can I reach the unknown variables from the input variables
- **Parity relations**
  - Which relations exist
- **Detectability**
  - Which faults can be detected with the found parity relations
- **Isolability**
  - Which faults can be isolated with the found parity relations
Investigate the effect of a structural fault

- Faults can be modeled by disabling nodes
  - Select **Disable Node** from the edit menu
  - Click on a node to disable it
  - Perform a new analysis
- The result shows if the system can be reconfigured in case of faults
Results for Diagnosis

SaTool provides automatically generated (nonlinear) parity relations. Use these as residual generators
\[ r(t) = \tilde{g}(c_i(X,K)) \]
- \( r = 0 \) when constraints in \( \tilde{g} \) are valid
- \( r \neq 0 \) when constraints in \( \tilde{g} \) are violated

In order to investigate dynamic properties, we wish to analyse the case of linear parity relations.

Problem: given a linear system
Determine: a residual generator

Design Sequence Overview

Many good approaches here - let’s pick one!
Design of residual generators

Chapter 6.1-6.3

Design of residual generator - notation and general model

\[ \dot{x}(t) = g(x(t), u(t), d(t), f(t)), \quad x(0) = x_0 \]
\[ y(t) = h(x(t), u(t), d(t), f(t)) \]

- \( x \) be a state vector - velocity position, turn rates and angles
- \( u \) be a control signal, e.g. rudder angle, propeller thrust
- \( y \) be a vector of measured quantities
- \( f \) faults represented as a vector and
- \( d \) be disturbances
Detailed design of fault diagnosis

Linear case:
Let variables without a bar denote deviation from the point of linearization, \( x(t) = \bar{x}(t) - \bar{x}_0(t) \)

\[
\dot{x}(t) = Ax(t) + Bu(t) + E_f d(t) + F_f f(t), \quad x(0) = 0
\]

\[
y(t) = Cx(t) + Du(t) + E_f d(t) + F_f f(t)
\]

Fault detection - detectability

Problem (6.1)
Given a model of the process, determine a stable residual generator such that:
- With no fault \( (f(t)=0) \) \( r(t) \rightarrow 0 \)
- \( r(t) \) is affected by \( f(t) \)
Or:
\[
\forall t, u(t), d(t), x(0), z(0): \quad f(t)=0 \Rightarrow r(t) \rightarrow 0
\]
\[
\exists t, r(t) \neq 0 \iff \exists t, f(t) \neq 0
\]
Residual generator

\[ r(s) = V_y u(s) + V_y y(s) \]

\[ r(s) = (V_y u(s) + V_y H_{dyy}(s)) u(s) + V_y H_{did}(s) d(s) \]

\[ + V_y H_{dyx}(s) x + V_y H_{df}(s) f(s) \]

**Design goal:** make \( r(s) \) independent of \( u(s) \) and \( d(s) \)

We desire \( V_y H_{dy}(s) \neq 0 \) but it is not guaranteed that this can be achieved with this design goal.
Residual generator - parity space approach

The problem is to make \( r(s) \) independent of \( u(s) \) and \( d(s) \):

\[
\begin{align*}
(V_{ru}(s) + V_{ry}(s)H_{ru}(s))u(s) + V_{ry}(s)H_{yd}(s)d(s) &= 0 \\
\equiv \\
(V_{ru}(s) V_{ry}(s))\begin{pmatrix} H_{ru}(s) & H_{yd}(s) \\ I & 0 \end{pmatrix} &= 0
\end{align*}
\]

This problem has the form:

given \( A(s) \), find all \( x(s) \) that satisfy \( x(s)A(s) = 0 \)

Nullspace design of residual generator

Let \( \begin{pmatrix} H_{ru}(s) & H_{yd}(s) \\ I & 0 \end{pmatrix} = H(s) = \frac{1}{h(s)} \bar{H}(s) \)

Algorithm:
1. Find the nullspace \( N_r \) of \( \bar{H}(s) \)
2. Determine \( (\tilde{V}_{ru}(s) \quad \tilde{V}_{ry}(s)) \subset N_r(\bar{H}(s)) \)
3. Let \( F(s) = (\tilde{V}_{ru}(s) \quad \tilde{V}_{ry}(s)) \)
4. Let a residual generator be \( r(s) = \frac{1}{p(s)}Q(s)F(s)\begin{pmatrix} y(s) \\ u(s) \end{pmatrix} \)

where \( p(s) \) is a stable, scalar polynomial to make \( \frac{1}{p(s)}Q(s)F(s) \) causal.

\( Q(s) \) is "arbitrary", nonzero.
Ship example: gyro fault diagnosis

\[ \dot{\omega}_3 = b(\delta + H(\omega_3, \delta)) \]
\[ \dot{\psi} = \omega_3 + \omega_{3w} \]
\[ \psi_m = \psi + f \psi \]
\[ \omega_{3m} = \omega_3 + \omega_{3w} + f \delta \]
\[ \delta_m = \delta \]

and \[ \frac{\partial H}{\partial r} = H_1 < 0 \]

\[ r \rightarrow 0 \Rightarrow \text{stable} \]
Example: Heading and rate measurements

\[
H(s) = \frac{1}{s(s-bh)} \begin{pmatrix} bs & s(s-bh) \\ b & (s-bh) \\ s(s-bh) & 0 \end{pmatrix}, \quad bh < 0
\]

has the left nullspace basis

\[
F(s) = \begin{pmatrix} -1 & s & 0 & 0 \\ 0 & -1 & 0 & s & 0 \end{pmatrix}
\]

Ship example continued

Measurements are rate gyro + gyro angle

\[
\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} \omega_{3m} - \hat{\omega}_3 \\ \psi_m - \hat{\psi} \end{bmatrix} = \begin{bmatrix} 1 \\ s^{-1} \end{bmatrix} \omega_{2\omega} + \begin{bmatrix} f_\theta \\ f_\psi \end{bmatrix}
\]

Insensitivity to wave input:

\[
r(s) = W(s) \begin{bmatrix} 1 \\ s^{-1} \end{bmatrix} = 0 \Rightarrow W(s) = \begin{bmatrix} -s^{-1} & 1 \end{bmatrix}
\]

and

\[
r(s) = f_\psi - s^{-1}f_\theta
\]
Residual generator

Residual generator
\[ r(s) = \frac{h(s)}{p(s)} Q(s) F(s) \begin{bmatrix} y(s) \\ u(s) \end{bmatrix} = \begin{bmatrix} V_y(s) \\ V_u(s) \end{bmatrix} \begin{bmatrix} y(s) \\ u(s) \end{bmatrix} \]

Residual generator for the specific case
\[
\begin{pmatrix} r_1(s) \\ r_2(s) \end{pmatrix} = \begin{pmatrix} 1 \\ s \\ 0 \\ 1 \\ -1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} \omega_{sy}(s) \\ \psi_{sy1}(s) \\ \psi_{sy2}(s) \\ \delta(s) \end{pmatrix}
\]

A "natural" choice that is independent of ship's dynamics

Response to faults

Residual response to faults and initial condition
\[ r(s) = \frac{1}{p(s)} Q(s) F(s) \begin{bmatrix} H_{yf}(s) & H_{xf}(s) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} f(s) \\ x_0 \end{bmatrix} \]

In the specific case
\[
\begin{pmatrix} r_1(s) \\ r_2(s) \end{pmatrix} = \begin{pmatrix} 1 \\ s \\ 0 \\ 1 \\ -1 \end{pmatrix} f_{wf}(s) + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} \omega_f(0) \\ \psi_f(0) \end{pmatrix}.
\]
Fault estimation given isolation

\[ H_\omega(s) = \begin{pmatrix} \frac{1}{s} & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \]

\( f_\omega \) isolated \( \Rightarrow \hat{f}_\omega(s) = s r_\omega(s) \)
\( f_{\psi_1} \) isolated \( \Rightarrow \hat{f}_{\psi_1}(s) = 0.5(\hat{r}_1(s) - r_1(s)) \)
\( f_{\psi_2} \) isolated \( \Rightarrow \hat{f}_{\psi_2}(s) = r_2(s) \)

Change Detection

Diagnosis and fault-tolerant control
Section 6.4
Design Sequence Overview

A classical approach

Fault-tolerant Controller Design

Change detection

Residual generator design

Structural Analysis

Component-based Analysis

Overview

Log-likelihood ratio

Log likelihood ratio for an observation $r(i)$:

$$s(r(i)) = \ln \frac{p(r(i) | \theta_1)}{p(r(i) | \theta_0)}$$

The two distributions are given because $\theta_1$ and $\theta_0$ are our hypotheses for fault and no fault, respectively.
Cusum detection

The cumulative sum (CUSUM) is a summation of the log-likelihood ratio

\[ S(j) = \sum_{k=1}^{j} s(k) \]

The CUSUM will integrate up when \( s > 0 \) and down for \( s < 0 \).

When \( S \) pass a threshold, a decision can be taken about the hypotheses: normal condition - faulty condition.

The decision to decide between \( \theta_0 \) and \( \theta_i \) is

- accept \( \theta_0 \) when \( S \leq a \)
- accept \( \theta_i \) when \( S \geq h \)
- continue to observe and test when \( a < S < h \)

Recursive cusum – gaussian residual

Assume a Gaussian distribution if the residual \( r = N(\mu_0, \sigma_0^2) \), i.e. with mean \( \mu_0 \) and variance \( \sigma_0^2 \), when there is no fault, hypothesis \( \theta_0 \)

\[ p(r_i \mid \theta_0) = \frac{1}{\sigma_0 \sqrt{2\pi}} \exp \left( -\frac{(r_i - \mu_0)^2}{2\sigma_0^2} \right) \]

The faulty condition, hypothesis \( \theta_i \), has \( r_i = N(\mu_i, \sigma_i^2) \)

The log likelihood ratio is then

\[ S_j = \ln \left( \frac{p(r_j \mid \theta_1)}{p(r_j \mid \theta_0)} \right) = \ln \frac{\sigma_0}{\sigma_1} + \frac{(r_j - \mu_0)^2}{2\sigma_0^2} - \frac{(r_j - \mu_i)^2}{2\sigma_i^2} \]
Recursive cusum – specific change

Problem 1. Given \( r \sim \mathcal{N}(\mu, \sigma^2) \), let a fault imply a change in mean from \( \mu_0 \) to \( \mu_1 \), with unchanged variance \( \sigma \) before and after the fault, the log-likelihood ratio is then

\[
S_i = \frac{(\mu_1 - \mu_0)^2}{\sigma^2} \left( r - \frac{\mu_0 + \mu_1}{2} \right)
\]

Problem 2: Given \( r \sim \mathcal{N}(\mu, \sigma^2) \), let a fault imply a change in variance from \( \sigma_0^2 \) to \( \sigma_1^2 \), with unchanged mean \( \mu \) before and after the fault, the log-likelihood ratio is then

\[
S_i = \ln \frac{\sigma_0}{\sigma_1} + \frac{\sigma_1^2 - \sigma_0^2}{\sigma_1^2 \sigma_0^2} (r - \mu)^2
\]

Different uses of the CUSUM tests

CUSUM testing can be made in a number of ways. We treat sequential and recursive approaches (2 and 3 are equivalent).

Observe that condition “fault” means \( g(k) \) will increase, “no-fault” will make \( g(k) \) decrease.

1. Compute \( g(k) \) until an upper or lower limit is reached. No hypothesis can be made until either limit is reached. This is a sequential test.
2. Compute \( g(k) \) and store \( g_{\min} = \min(g(k)) \). If \( g(k) - g_{\min} > h \) hypothesis \( \theta_1 \) is assumed. Otherwise \( \theta_0 \).
3. Let \( g(k) = \max(g(k), 0) \) and stop test when a threshold \( h \) is reached. Assume “no-fault” until \( h \) is reached. Recursive test.
Sequential test and the cusum function

Sequential cusum test for change in mean:

\[ g(k) = g(k-1) + \frac{\mu_1 - \mu_0}{\sigma^2} (r(k) - \frac{\mu_1 + \mu_0}{2}) \]

Algorithm for sequential test:

\[ g(k) = g(k-1) + \frac{\mu_1 - \mu_0}{\sigma^2} (r(k) - \frac{\mu_1 + \mu_0}{2}) \]

if \((g(k) < a)\), \(\theta_0\) is true; \(g(k) = 0\); end;

if \((g(k) > h)\), \(\theta_0\) is true; \(g(k) = 0\); end;

Sequential test

Test between two hypotheses:

\(\theta_0\): mean value is \(\mu_0\), \(\theta_1\): mean value is \(\mu_1\)

Algorithm for sequential test:

\[ g(k) = g(k-1) + \frac{\mu_1 - \mu_0}{\sigma^2} (r(k) - \frac{\mu_1 + \mu_0}{2}) \]

if \((g(k) < a)\), \(\theta_0\) is true; \(g(k) = 0\); end;

if \((g(k) > h)\), \(\theta_0\) is true; \(g(k) = 0\); end;
Choice of limits for hypothesis test

With two limits for the hypothesis test $a$ and $h$, these can be chosen to give specified probabilities for false alarm and missed detection.

- $P_f$: false detection probability
- $P_m$: missed detection probability

Decker’s result: Choose $h = \ln\left(\frac{1 - P_m}{P_f}\right)$, $a = \ln\left(\frac{P_m}{1 - P_f}\right)$

Recursive form of CUSUM

Cumulative sum for recursive change detection:

$$g(k) = \max\left(0, g(k-1) + \frac{\mu_i - \mu_0}{\sigma^2} (r(k) - \frac{\mu_i + \mu_0}{2})\right)$$
Cusum test – change of test value

The test quantity is $\frac{1}{2}(\mu_0 + \mu_1)$. The "gain" is $\frac{\mu_1 - \mu_0}{\sigma^2}$.

Fig. 6.11 shows sensitivity to change in test quantity.

Time to detect and time between false alarms - Average Run Length (ARL)

Let a signal $z(k)$ have mean $\mu_s$ and variance $\sigma^2$.

The time it takes to reach $h$ is

$$L(\mu_s, \sigma, h) = \frac{\sigma^2}{2\mu_s^2} \left( \exp \left( - \left( 2.0 \frac{\mu_s h}{\sigma^2} + 2.2232 \frac{\mu_s}{\sigma} \right) \right) + \left( 2.0 \frac{\mu_s h}{\sigma^2} + 2.2232 \frac{\mu_s}{\sigma} \right) - 1 \right), \quad \mu_s \neq 0$$

$$\left( \frac{h}{\sigma} + 1.1166 \right)^2, \quad \mu_s = 0$$

With $\mu_s = \mu_t$, the ARL is the average time to detect:

With $\mu_s = \mu_0$, the ARL is the average time between false alarms.
Mean and variance for cusum test

The CUSUM test for a mean value change was

\[ s_i = \frac{\mu_1 - \mu_0}{\sigma^2} \left( r_i - \frac{\mu_1 + \mu_0}{2} \right) \]

If \( E\{r_i\} = \mu_0 \),

\[
\begin{align*}
\mu_s &= E_{r_i} \{ s(r_i) \} = \frac{(\mu_1 - \mu_0)^2}{2\sigma^2} \\
\sigma_s^2 &= E \{ (s(r_i) - \mu_s)^2 \} = \frac{(\mu_1 - \mu_0)^2}{\sigma^2}
\end{align*}
\]

If \( E\{r_i\} = \mu_1 \),

\[
\begin{align*}
\mu_s &= E_{r_i} \{ s(z) \} = \frac{(\mu_1 - \mu_0)^2}{2\sigma^2} \\
\sigma_s^2 &= E \{ (s(z) - \mu_s)^2 \} = \frac{(\mu_1 - \mu_0)^2}{\sigma^2}
\end{align*}
\]

ARL for the CUSUM

For the case of fault

\[
\hat{\tau}_{\text{detect}} = L(\mu_s, \sigma_s, h) = L\left( \frac{(\mu_1 - \mu_0)^2}{2\sigma^2}, \frac{(\mu_1 - \mu_0)^2}{\sigma^2}, h \right)
\]

is mean time to detect.

For the case of no-fault

\[
\hat{\tau}_{\text{false alarm}} = L(\mu_s, \sigma_s, h) = L\left( \frac{(\mu_1 - \mu_0)^2}{2\sigma^2}, \frac{(\mu_1 - \mu_0)^2}{\sigma^2}, h \right)
\]

is mean time between false alarms.

This result is a key design parameter.
ARL for CUSUM

Design procedure:
Given $\mu_i - \mu_0$ and $\sigma^2$,
- determine $h$ to give small low $r_{\text{detect}}$ (3-20 samples)
- check that false alarm time is $10^5 - 10^8$ samples or more.
If design objective is not met,
- decrease $\sigma^2$ (filter) and/or
- increase $\mu_i$
The last step means to make the detector less sensitive, larger size of fault is deemed for detection.

Design Sequence Overview

Finally!
Fault-tolerant control

Chapter 7.1-7.3

A system with control is a hybrid entity. The system can be changed by events, some of which are faults. We may change the controller and/or the system by action of a supervisor.
Diagnosis -> Fault-tolerant

- **Diagnosis**
  - Fault detection
  - Fault isolation
  - Evaluation to get confirmed hypothesis
  - Fault estimation

- **Fault-tolerant methods**
  - Assess estimated structure
  - Is diagnosis unambiguous,
    - determine fault-accommodation action
  - If not,
    - react according to highest severity
    - accommodate both/all possible scenarios
  - Reconfigure in time

---

**Standard control problem - example**

Objective:
Obtain bandwidth > 2 rad/s, loop damping > 0.2
load suppression better than 0.5 of nominal in range [0, 0.5] rad/s,

\[
\theta(s) = \frac{k_p k_f}{s^2 + k_p k_f s + k_p k_f} \theta_{\text{ref}}(s)
\]

Compare with standard form

\[
\theta(s) = \frac{\omega_0^2}{s^2 + 2\omega_0 s + \omega_0^2} \theta_{\text{ref}}(s)
\]
Standard control problem - example

Solution:
Bandwidth >= 2 rad/s, loop damping >= 0.2
load suppression better than 0.5 of nominal in range [0, 0.5] rad/s,

Steady state error due to fixed value of $Q$ is

$$\lim_{r \to \infty} \left( \theta - \theta_{ref} \right) = \frac{1}{k_p k_i k_q} Q_i$$

Nominal solution: $U : i_{com} = k_p \left( \theta_{ref} - \theta \right) - n_a$;

Choose $\zeta=0.5$ as nominal value.

$$k_p \geq \frac{4}{k_i} \quad \text{and} \quad \frac{1}{k_i} \geq 1 \Rightarrow k_i^2 = \frac{4}{k_p} \quad \therefore \quad k_p = k_i$$

Control error is \( \theta - \theta_{ref} \to \frac{1}{4} \left( \frac{1}{\omega_k^2} \right) Q \) for \( t \to \infty \)

Handling of fault - reconfiguration

• Fault reconfiguration: a sensor failure in inner loop.
• Switch to differentiating control when fault is diagnosed
Fault-tolerance against sensor faults

Chapter 7.1 - 7.2

Handling of sensor faults by reconfiguration

Reconfiguration: Failed sensor measurement is replaced by an estimate, which is used in the feedback loop

\[ y_k = (1 - H(f_k))y_k + H(f_k) \hat{y}(f_k) \]

Condition for observer \( k \) to exist

\( (A, c, B, c, C_1, \ldots, C_n) \) is observable where \( c_i \) is a column of \( C \)

If not fully observable, the unobservable subsystem must at least be stable
Handling of sensor fault by output estimation

Application - marine navigation

Position and velocity sensor input for NavCom
Application to marine navigation

Navigation computer to make fault-tolerant sensor fusion for
Inertial navigation units
GPS receivers (L1,L2) and diff. GPS and L1 only types
Ship’s log
Conventional gyro units
Vertical gyro
Output of position data: 1Hz, attitude data: 16Hz.
A commercial product from is available with algorithms by the author.

Fault-tolerant sensor fusion
- main issues

- Data arrive asynchronously
- Data have different sampling rates
- Each new data package need be validated before it is included in the sensor fusion solution
- Data from a particular sensor may stop without warning
- Faults can develop arbitrarily (abrupt, incipient, intermittent)
- Faults on certain sensors may be correlated (GPS)
- Noise on different sensor types is often uncorrelated
Optimal sensor fusion

1. Compute inverse covariance:
\[ P_k^{-1} = (P_k)\cdot^{-1} + C_k^T R_k^{-1} C_k \]
\[ P_{k+1} = (P_{k+1})\cdot^{-1} \]
\[ K_k = P_{k+1} C_k R_k^{-1} \]
2. Update estimate
\[ \hat{x}_{k+1} = \hat{x}_k + K_k \cdot (y_k - C_k \cdot \hat{x}_k) \]
3. Predict in time
\[ \hat{x}_{k+1} = \Phi_k \cdot \hat{x}_k \]
\[ P_{k+1} = \Phi_k \cdot P_k \cdot \Phi_k^T + Q_k \]

Measurements that are uncorrelated may be updated when independent sensor packages arrive:
\[ P_{k+1}^{-1} = (P_{k+1})^{-1} + \begin{bmatrix} R_{11}^{-1} & 0 & 0 & C_1 \\ C_1^T & R_{22}^{-1} & 0 & C_2 \\ 0 & 0 & R_{33}^{-1} & C_3 \end{bmatrix} = (P_k)^{-1} + C_1^T R_1^{-1} C_1 + C_2^T R_2^{-1} C_2 + C_3^T R_3^{-1} C_3 \]
This allows asynchronous or missing data.
State logic – asynchronus events

- Treatment of sensor data as random events in time – shown as state-event diagrams for 3 sensors

Consistent implementation

State event logic in consistent implementation:
- change state: \( s(k+1) \leftarrow S(s(k),e(k)) \)
- computations: \( r(t) \leftarrow R(s(k),e(k)) \)

\( s \in \{0,1,2,3,4,5,6\} \),
\( e \in \{v_1, v_2, v_3, \neg v_1, \neg v_2, \neg v_3\} \)

Three sets of sensor events:
- valid and not-valid

State transition matrix

\[
S(s,e) = \begin{bmatrix}
1 & 2 & 3 & 0 & 0 & 0 \\
1 & 3 & 5 & 0 & 1 & 1 \\
3 & 2 & 6 & 2 & 0 & 2 \\
5 & 6 & 4 & 4 & 4 & 0 \\
3 & 7 & 3 & 2 & 1 & 3 \\
5 & 7 & 5 & 4 & 5 & 1 \\
7 & 6 & 6 & 6 & 4 & 2
\end{bmatrix}
\]
Application in Space

- Fault-tolerant ideas applied to the Danish Ørsted satellite - in operation in space since 23 Feb 1999 - and still active.
- Sensor fusion between
  - Star imager
  - Sun sensors
  - Magnetometer and Earth-B-field model
- Gives
  - Attitude
  - Angular velocity estimate

Actuator Faults
Actuator faults

Let \( \beta_i (u_i(t), \theta_i) \) describe the action of the faulty actuator:
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t) + \sum_{i \in I_f} \beta_i (u_i(t), \theta_i)
\]

Two solutions:
1. We may use an estimate \( \hat{\beta} \) and redesign using
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t) + \sum_{i \in I_f} \hat{\beta}_i (u_i(t), \theta_i)
\]
2. We may use the non-faulty actuators only, and redesign using
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t)
\]

Actuator fault - fault estimation (1)

Let \( \beta_i (u_i(t), \theta_i) \) describe the action of the faulty actuator:
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t) + \sum_{i \in I_f} \beta_i (u_i(t), \theta_i)
\]

Two solutions:
1. We may use an estimate \( \hat{\beta} \) and compensate, if possible using
\[
\dot{x}(t) = Ax(t) + \left( \sum_{i \in I_f} b_i u_i(t) - \sum_{i \in I_f} \hat{\beta}_i (u_i(t), \theta_i) \right) + \sum_{i \in I_f} \beta_i (u_i(t), \theta_i)
\]
2. We may use the non-faulty actuators only, and redesign using
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t)
\]
Actuator fault - fault estimation (2)

Let $\beta_i (u_i(t), \theta_i)$ describe the action of the faulty actuator. Then
\[
\sum_{i \in I_x} b_i u_i(t) + \sum_{i \in I_y} \beta_i (u_i(t), \theta_i) = 0
\]
would outbalance the fault.

One might use an estimate of $\hat{\beta}$ and redesign using
\[
\dot{x}(t) = Ax(t) + \sum_{i \in I_x} b_i u_i(t) - \sum_{i \in I_y} \hat{\beta}_i (u_i(t), \theta_i) + \sum_{i \in I_y} \beta_i (u_i(t), \theta_i)
\]
\[
\dot{x}(t) = Ax(t) + B_f u_f(t) + B_f u_{\text{comp}}(t) + \sum_{i \in I_y} \beta_i (u_i(t), \theta_i) \Rightarrow
\]

\[
B_f u_{\text{comp}}(t) = -\sum_{i \in I_y} \hat{\beta}_i (u_i(t), \theta_i)
\]

Actuator fault - fault estimation (3)

If the pseudoinverse of $B_n$ exists, then a solution to compensator design is
\[
u_{\text{comp}}(t) = -\left(B_f^T B_f\right)^{-1} B_f^T \sum_{i \in I_y} \hat{\beta}_i (u_i(t), \theta_i)
\]

However, this does not necessarily give a trajectory of $x(t)$ close to the desired - when $B_f \neq B$
Actuator faults – optimal control approach

- Linear quadratic optimal control gives the "LQ solution" to the state feedback problem.
- If not all states are measured, a state-observer is employed.

Let the system be:

$$\dot{x}(t) = Ax(t) + Bu(t) + \sum_{i=1}^{n} B_i u_i(t)$$

LQ optimal control: minimize the index

$$J((0, \infty), u, x_0) = \frac{1}{2} \int_0^\infty \left( u^T R u + x^T Q x \right) dt \quad Q \succeq 0, \quad R > 0, \quad x(0) = x_0$$

LQ optimal control – nominal system

A solution to the LQ problem requires $(A, B)$ is controllable.

The steady state controller $L$ is obtained as:

$$u(t) = -R^{-1} B^T S x(t) = -L x(t)$$

where $S$ is the (stable) solution to the steady state Riccati equation

$$0 = A^T S + S A - S B R^{-1} B^T S + Q$$

The closed-loop system

$$\dot{x}(t) = \left( A - BR^{-1} B^T S \right) x(t)$$

is stable.

The value of the "cost" index is:

$$J((0, \infty), x_0) = x_0^T S x_0$$
All stabilizing state feedback controllers

Let a stabilizing LQ feedback law be
\[ u_i(t) = -R^{-1}B^iS_i x(t) = -L_i x(t) \]
the feedback gain matrix is \( L_i = R^{-1}B^i S_i \)
and let \( S_i \) be the solution to
\[ 0 = A^i S_i + S_i A - S_i B R^{-1}B^i S_i + Q \]
Then
\[ J((t_i, \infty), x_i) = x_i^T S_i x_i \]
is the measure of "cost" starting at \( x_i \) at time \( t_i \).

Actuator fault at time \( t_f \)

Given: No actuator fault \( t \subset [0, t_f] \), an actuator fault \( t \subset [t_f, \infty] \)
Assume the faulty actuators can be described by a linear model:
\[ \hat{\beta}(u_i(t), \tilde{\theta}_i) = \hat{b}_i u_i(t), i \in I_f \]
Then the model of the faulty system is
\[ \dot{x}(t) = Ax(t) + \sum_{i \in I_f} b_i u_i(t) + \sum_{i \in I_f} \hat{b}_i (u_i(t), \tilde{\theta}_i) \]
\[ \dot{x}(t) = Ax(t) + B_f u(t) \text{ with } x(t_f) = x_f \text{ at the time the fault occurs} \]
Actuator fault at time $t_f$

With $\dot{x}(t) = Ax(t) + B_fu(t)$ with $x(t_f) = x_f$ at $t_f$

the trajectory for $t \in [t_f, \infty]$ is optimised using

$$\dot{x}(t) = Ax(t) + B_fu(t)$$

$u(t) = R^{-1}B_f^TS_f x_f(t)$.

where $S_f$ be the solution to

$$0 = A^TS_f + S_fA - S_fB_fR^{-1}B_f^TS_f + Q$$

and $S_f$ is symmetric ($S_f = S_f^T$)

---

Actuator fault at time $t_f$ – total cost

If the index "consumed" from $t \in [0, t_f]$ is $J\left((0, t_f), x_o\right)$

Then $J\left((0, t_f), x_o\right) \left((t_f, \infty), x_f\right) = J\left((0, t_f), x_o\right) + x_f^TS_fx_f$

instead of $J\left((0, \infty), x_o\right) = x_o^TS_o x_o$

$J\left((0, \infty), x_o\right) = J\left((0, t_f), x_o\right) + x_f^TS_fx_f$

$\Rightarrow J\left((0, t_f), x_o\right) = x_o^TS_o x_o - x_f^TS_fx_f$

$\Rightarrow J\left((0, t_f), x_o\right) \left((t_f, \infty), x_f\right) = x_o^TS_o x_o + x_f^T(S_f - S)x_f$
Ship steering - example (1)

Given (ship with $H_i=0$ and two rudders)
\[
\begin{bmatrix}
\frac{dx_1}{dt} \\
\frac{dx_2}{dt}
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
u_1 \\
u_2
\end{bmatrix} + \begin{bmatrix}
k \end{bmatrix}
\]

Optimization criterion
\[
J = \int_0^\infty \left( x^T Q x + u^T R u \right) dt,
\]
\[
Q = \begin{bmatrix}
1 & 0 \\
0 & 100
\end{bmatrix},
R = 100 \times \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

Find the optimal controller for
- the nominal case ($k/I = 1/26.8 \text{ N/kgm}^2$)
- case when one actuator is defect $B = \begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}$

Ship steering – example (2)

Define problem for Matlab $lqr$ function:
\[
A = \begin{bmatrix}
0 & 0 \\
1 & 0
\end{bmatrix},
B_s = \begin{bmatrix}
1 \\
0
\end{bmatrix},
\]
\[
Q = \begin{bmatrix}
1 & 0 \\
0 & 100
\end{bmatrix},
R = 100 \times \begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\]

Find the optimal controller for
1) the nominal case:
\[
[K, S, \lambda] = \text{lqr}(A, B_s, Q, R)
\]
2) Case when one actuator is defect $B_s = \begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix}$
\[
[K_s, S_s, \lambda] = \text{lqr}(A, B_s, Q, R)
\]
Ship steering – example (3)

Results:
Nominal case (no fault)
\[ \mathbf{K}_0 = \begin{bmatrix} 4.35 & 0.71 \\ 4.35 & 0.71 \end{bmatrix}; \quad \mathbf{S}_0 = \begin{bmatrix} 11668 & 1895 \\ 1895 & 732 \end{bmatrix}; \quad \lambda_{ci} = -0.162 \pm j0.162 \]

Fault on actuator 2:
\[ \mathbf{K}_2 = \begin{bmatrix} 7.32 & 1.0 \\ 0 & 0 \end{bmatrix}; \quad \mathbf{S}_2 = \begin{bmatrix} 19623 & 2680 \\ 2680 & 616 \end{bmatrix}; \quad \lambda_{ci} = -0.137 \pm j0.137 \]

Test response from initial condition in closed loop
\[ A_{ci} = A - BK_0 \quad (\text{case 0}) \quad A_{ci2} = A - B_2 K_2 \]
\[ \text{sys} = \text{ss}(A_{ci}, A_{ci2}, C, D) \quad \text{with} \quad C = \text{eye}(2,2); \quad D = 0*\text{eye}(2,2) \]
\[ \text{step(sys)} \]

Model-matching state-feedback

The nominal (no fault) system in closed state feedback loop:
\[ \dot{x}(t) = (A - BK) x(t) \]
\[ y(t) = Cx(t) \]

after the fault occurs,
\[ \dot{x}(t) = A_f x(t) - B_f u(t) \]
\[ y(t) = C_f x(t) \]
with new state feedback controller:
\[ \dot{x}(t) = (A_f - B_f K_f) x(t) \]
\[ y(t) = C_f x(t) \]
Model-matching state-feedback (2)

Ideal if we could obtain \( A - BK = A_f - B_f K_f \)
This is only rarely possible (requires redundant actuators).
Consider the relaxed condition:
\[
\exists L_f \subset \{ L_{stab} \} : \min \| (A - BK) - (A_f - B_f K_f) \| ?
\]
If the pseudo-inverse of \( B_f \) exists
\[
B_f K_f = A_f - (A - BK) \Rightarrow L_f = \left( B_f^T B_f \right)^{-1} B_f^T (A_f - A + BK)
\]

Model-matching output-feedback

Let the controller be
\[
u(t) = -Ky(t) \Rightarrow
\]
\[
\dot{x}(t) = \left( A - BK \right) x(t)
\]
\[
y(t) = Cx(t)
\]
After the a sensor fault occurs,
\[
\dot{x}(t) = \left( A - BK, C_f \right) x(t)
\]
\[
y(t) = C_f x(t)
\]
Exact model matching if \( K_f C_f = KC \).
If \( C = LC_f \) then \( K_f = KL \) gives exact model matching
and we use \( u(t) = -KL y(t) \) after the fault occurs.
Model-matching output-feedback actuator fault

Let the controller be
\[ u(t) = -K y(t) \to \]
\[ x(t) = (A - BK) x(t) \]
\[ y(t) = C x(t) \]

After the sensor fault occurs,
\[ \dot{x}(t) = (A - B_f K_f C) x(t) \]
\[ y(t) = C x(t) \]

Exact model matching if \( B_f K_f = BK \).

Possible if \( B_f M = B \), then \( K_f = MK \) gives exact model matching
and we use \( u(t) = -MK y(t) \) after the fault has occurred.

Ship steering example – example (4)

\[
B = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} ; \text{ normal case}
\]

\[
B_f = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} ; \text{ case of fault on rudder 2}
\]

\[
M = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \Rightarrow B_f M = B \Rightarrow \text{model matching}
\]

hence switch to \( u(t) = -MK y(t) \) after the fault is diagnosed

or
\[
\begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = - \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 4.35 & 0.71 \\ 0.71 & 0.71 \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = -2 \begin{bmatrix} 4.35 & 0.71 \\ 0.71 & 0.71 \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}
\]
a rather obvious choice when one out of two parallel actuators fail!
Summary

- Component-based and Structural analysis
  - Coping with complexity and nonlinear behaviour
- Stringent methods for linear case
  - Fault diagnosis
  - Sensor fusion
- Fault-tolerant Control
  - Sensor faults
  - Actuator faults
- Full scale applications
  - Inverter for crane, Satellite attitude control, Sensor fusion for navigation

Acknowledgements

- Michel Kinnaert (residual generator design)
- Michelle Basseville and Igor Nikiforov (change detection)
- Marcel Staroswiecki (structural analysis and LQ-FTC)
- Jan Lunze (fault-tolerant design ideas)
- Roozbeh Izadi-Zanamabadi (SA & applications)
- Claus Thybo (Inverter application)
- Torsten Lorentzen (SaTool)
That's all folkes