Introduction
Analysis and design of large-scale diagnosis systems

**Definition (Large scale)**

Systems and models that can not be managed by hand; that need computational support.

*We do not mean:* distributed diagnosis, big data, machine learning, classifiers, and other exciting fields

**Scope of tutorial**

- Describe techniques suitable for large scale, non-linear, models based on structural analysis
- Support different stages of diagnosis systems design
- Provide a theoretical foundation

Methods for fault diagnosis

\[
\dot{x} = Ax + Bu \\
\dot{x} = g(x, u) \\
y = Cx \\
y = h(x)
\]

There are many published techniques, elegant and powerful, to address fault diagnosis problems based on, e.g., state-space models like above.

They might involve, more or less, involved mathematics and formula manipulation.

**This tutorial**

This tutorial covers techniques that are suitable for large systems where involved hand-manipulation of equations is not an option.
Main parts of the tutorial

Outline

- Formally introduce structural models and fundamental diagnosis definitions
- Derive algorithms for analysis of models and diagnosis systems
  - Introduction of fundamental graph-theoretical tools, e.g., Dulmage-Mendelsohn decomposition of bi-partite graphs
  - Determination of fault isolability properties of a model
  - Determination of fault isolability properties of a diagnosis system
  - Finding sensor locations for fault diagnosis
- Derive algorithms for design of residual generators
  - Finding all minimal submodels with redundancy
  - Generating residuals based on submodels with redundancy

Objectives

- Understand fundamental methods in structural analysis for fault diagnosis
- Understand possibilities and limitations of the techniques
- Introduce sample computational tools
- Tutorial not intended as a course in the fundamentals of structural analysis, our objective has been to make the presentation accessible even without a background in structural analysis
- Does not include all approaches for structural analysis in fault diagnosis, e.g., bond graphs and directed graph representations are not covered.

Software

Fault Diagnosis Toolbox for Matlab

Some key features

- Structural analysis of large-scale DAE models
- Analysis
  - Find submodels with redundancy (MSO/MTES)
  - Diagnosability analysis of models and diagnosis systems
  - Sensor placement analysis
- Code generation for residual generators
  - based on matchings (ARRs)
  - based on observers

Download – code + documentation

http://www.fs.isy.liu.se/Software/FaultDiagnosisToolbox/

Experimental code

The code is poorly tested, and I’m sure contains a lot of bugs. Still useful and we will continue to develop it.

Basic principle - systematic utilization of redundancy

4 equations, 1 unknown, 6 (minimal) residual generators

\[
\begin{align*}
  x &= g(u) \\
  y_1 &= x \\
  y_2 &= x \\
  y_3 &= x \\
  r_1 &= y_1 - g(u) \\
  r_2 &= y_2 - g(u) \\
  r_3 &= y_2 - y_1 \\
  r_4 &= y_3 - g(u) \\
  r_5 &= y_3 - y_1 \\
  r_6 &= y_3 - y_2
\end{align*}
\]

- Number of possibilities grows exponentially (here \(\binom{n}{2}\) minimal combinations)
- Not just \(y - \hat{y}\)
- Is this illustration relevant for more general cases?
**Example: Ideal electric motor model**

![Electric motor diagram]

- $e_1: V = iR(1 + f_R) + L \frac{di}{dt} + K_a i \omega$
- $e_2: T_m = K_a i^2$
- $e_3: \int \frac{d\omega}{dt} = T - b \omega$

**Model summary (9 equations)**

- **Known variables (4):** $V, y_i, y_\omega, y_T$
- **Unknown variables (7):** $i, \theta, \omega, \alpha, T, T_m, T_I, (i, \omega, \theta$ dynamic)$
- **Fault variables (4):** $f_R, f_i, f_\omega, f_T$

**Structural model**

A structural model only models that variables are related!

Example relating variables: $V, i, \omega$

$$e_1: V = iR(1 + f_R) + L \frac{di}{dt} + K_a i \omega$$

<table>
<thead>
<tr>
<th>Unknown variables</th>
<th>$e_1$</th>
<th>$i$</th>
<th>$\theta$</th>
<th>$\omega$</th>
<th>$\alpha$</th>
<th>$T$</th>
<th>$T_m$</th>
<th>$T_I$</th>
<th>$f_R$</th>
<th>$f_i$</th>
<th>$f_\omega$</th>
<th>$f_T$</th>
<th>$V$</th>
<th>$y_i$</th>
<th>$y_\omega$</th>
<th>$y_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td>$X$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Coarse model description, no parameters or analytical expressions
- Can be obtained early in design process with little engineering effort
- Large-scale model analysis possible using graph theoretical tools
- Very useful!

**Main drawback:** Only best case results!

**Structural isolability analysis of model**

Nontrivial result

$f_R$ and $f_i$ can not be isolated from each other, unique isolation of $f_\omega$ and $f_T$
**Sensor placement - which sensors to add?**

**Q:** Which sensors should we add to achieve full isolability?

Choose among \{i, \theta, \omega, \alpha, T, T_m, T_l\}. Minimal sets of sensors that achieve full isolability are

- \( S_1 = \{i\} \)
- \( S_2 = \{T_m\} \)
- \( S_3 = \{T_l\} \)

Let us add \( S_1 \), a second sensor measuring \( i \) (one current sensor already used),

\[ y_{i,2} = i \]

**Create residuals to detect and isolate faults**

**Q:** Which equations can be used to create residuals?

Analysis shows that there are 6 minimal sets of equations with redundancy, called MSO sets. Three are

- \( M_1 = \{y_i = i, y_{i,2} = i\} \) ⇒ \( r_1 = y_i - y_{i,2} \)
- \( M_2 = \{y_\omega = \omega, y_T = T, J\dot{\omega} = T - b\omega\} \) ⇒ \( r_2 = y_T - J\dot{\omega} - b\omega \)
- \( M_3 = \{V = L\frac{di}{dt} + iR + K_a i \omega, y_\omega = \omega, y_i = i\} \) ⇒ \( r_3 = V - L\dot{y}_i + y_i R + K_a y_i y_\omega \)
- \( M_4 = \ldots \)
- \( M_5 = \ldots \)
- \( M_6 = \ldots \)

**Fault signature matrix and isolability for MSOs**

**Q:** Which isolability is given by the 6 MSOs/candidate residual generators?

<table>
<thead>
<tr>
<th>Fault Signature Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_R )</td>
</tr>
<tr>
<td>MSO1</td>
</tr>
<tr>
<td>( f_R )</td>
</tr>
</tbody>
</table>

**Create residuals to detect and isolate faults**

**Q:** Which equations can be used to create residuals?

- \( e_1 : V = iR(1 + f_R) + L\frac{di}{dt} + K_a i \omega \)
- \( e_2 : T_m = K_a i^2 \)
- \( e_3 : J\frac{d\omega}{dt} = T - b\omega \)
- \( e_4 : T = T_m - T_I \)
- \( e_5 : \frac{d\theta}{dt} = \omega \)
- \( e_6 : \frac{d\omega}{dt} = \alpha \)
- \( e_7 : y_i = i + f_i \)
- \( e_8 : y_\omega = \omega + f_\omega \)
- \( e_9 : y_T = T + f_T \)
- \( e_{10} : y_{i,2} = i \)

Example, equations \{e_3, e_8, e_9\} \{J\dot{\omega} = T - b\omega, y_\omega = \omega, y_T = T\} has redundancy! 3 equations, 2 unknown variables (\( \omega \) and \( T \))

\[ r = J\dot{\omega} + b\omega - y_T \]

**Structural redundancy**

Determine redundancy by counting equations and unknown variables!

**If I could design 6 residuals based on the MSOs ⇒ full isolability**
Test selection

**Q:** Do we need all 6 residuals? No, only 4

Fault Signature Matrix

<table>
<thead>
<tr>
<th>Fault</th>
<th>fR</th>
<th>fi</th>
<th>fw</th>
<th>fT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fault Signature Matrix, selected tests

<table>
<thead>
<tr>
<th>Fault</th>
<th>fR</th>
<th>fi</th>
<th>fw</th>
<th>fT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSO6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Code generation supported by structural analysis

**Q:** Can we automatically generate code for residual generator?

For example, MSO \( M_2 \)

\[
\{ y_\omega = \omega, y_T = T, J_\dot{\omega} = T - b_\omega \}
\]

Automatic generation of code

\[
\begin{align*}
\% & \text{ Initialize state variables} \\
& w = \text{state}.w; \\
& \% \text{ Residual generator body} \\
& T = yT; \% \text{e9} \\
& w = yw; \% \text{e8} \\
& dw = \text{ApproxDiff}(w,\text{state}.w,Ts); \% \text{e11} \\
& r_2 = J_\dot{\omega} + by_\omega - y_T \\
\end{align*}
\]

Design process aided by structural analysis

Modeling \( \rightarrow \) Diagnosability Analysis \( \rightarrow \) Sensor Selection

Residual Generator Analysis \( \rightarrow \) Test Selection \( \rightarrow \) Code Generation

Some history

50’s In mathematics, graph theory. A. Dulmage and N. Mendelsohn, "Covering of bi-partite graphs"

60’s-70’s Structure analysis and decomposition of large systems, e.g., C.T. Lin, "Structural controllability" (AC-1974)

90’s Structural analysis for fault diagnosis, first introduced by M. Staroswiecki and P. Declerck. After that, thriving research area in AI and Automatic Control research communities.

All these topics will be covered in the tutorial

Presentation biased to our own work
**Basic definitions**

### Outline

- **Introduction**
- **Structural models and basic definitions**
  - Diagnosis system design
  - Residual generation
  - Diagnosability analysis
  - Sensor placement analysis
- **Case study and software demonstration**
- **Analytical vs structural properties**
- **Concluding remarks**

### A structural model - the nominal model

**Variables types:**

- **Unknown variables:**
  - $i$, $\omega$, $T$, $T_m$, $T_l$
- **Known variables:**
  - sensor values, known input signals: $V$, $y_i$, $y_\omega$, $y_T$
- **Known parameter values:**
  - $R$, $L$, $K_a$, $J$, $b$

**Common mistakes:**

- Consider $i$ as a known variable since it measured.
- Consider a variable that can be estimated using the model, i.e., $T_m$, to be a known variable.

#### Biadjacency matrix:

```
<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>\omega</th>
<th>T</th>
<th>T_m</th>
<th>T_l</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
A structural model with fault information

Fault influence can be included in the model
- by fault signals
- by equation assumptions/supports

\[\begin{align*}
e_1: V &= i(R + f_R) + L \frac{di}{dt} + K_a i \omega \\
e_2: T_m &= K_a i^2 \\
e_3: J \frac{d\omega}{dt} &= T - (b + f_b) \omega \\
e_4: T &= T_m - T_l \\
e_5: y_i &= i + f_i \\
e_6: y_\omega &= \omega + f_\omega \\
e_7: y_T &= T + f_T
\end{align*}\]

- Compact description
- Good for analysis

Structural representation of dynamic systems

Structural representation of dynamic systems can be done in a number of ways.
- Consider \(x\) and \(\dot{x}\) to be structurally the same variable.
- Consider \(x\) and \(\dot{x}\) to be separate variables.
  - If the variable representing the derivative is denoted \(x'\) the model is extended with relations on the form
    \[\dot{x}' = \frac{dx}{dt}\]
    - Often, also extend with some causality constraints (e.g. differential or integral causality)
  - Choice depend on purpose and objective.
  - For analysis purposes, approach 1 is typically most suited.

Dynamics - not distinguish derivatives

\[\begin{align*}
e_1: V &= iR + L \frac{di}{dt} + K_a i \omega \\
e_2: T_m &= K_a i^2 \\
e_3: J \frac{d\omega}{dt} &= T - (b + f_b) \omega \\
e_4: T &= T_m - T_l \\
e_5: y_i &= i + f_i \\
e_6: y_\omega &= \omega + f_\omega \\
e_7: y_T &= T + f_T
\end{align*}\]

- Compact description
- Good for analysis

Dynamics - distinguish derivatives

\[\begin{align*}
e_1: V &= iR + Li' + K_a i \omega \\
e_2: T_m &= K_a i^2 \\
e_3: J \omega' &= T - b \omega \\
e_4: T &= T_m - T_l \\
e_5: y_i &= i \\
e_6: y_\omega &= \omega \\
e_7: y_T &= T
\end{align*}\]

- Add differential constraints
- Used for computing sequential residual generators
- Differential/integral causality
Structural properties interesting for diagnosis

Properties interesting both for residual generation, fault detectability and isolability analysis.

Let \( M = \{ e_1, e_2, \ldots, e_n \} \) be a set of equations.

Basic questions answered by structural analysis

- Can a residual generator be derived from \( M \) ?
  - or equivalently can the consistency of \( M \) be checked?
- Which faults are expected to influence the residual?

Structural results give generic answers. We will come back to this later.

Fault sensitivity of the residual?

- Model with fault:

  \[
  e_3 : T = J \frac{d\omega}{dt} + (b + f_b)\omega
  \]

  \[
  e_5 : i = y_i - f_i
  \]

  \[
  e_6 : \omega = y_\omega - f_\omega
  \]

  \[
  e_1 : V - i(R + f_R) - L \frac{di}{dt} - K_a i\omega = 0
  \]

  Which faults could cause the residual to be non-zero?

  \[
  r = V - y_i R + L \frac{dy_i}{dt} - K_a y_i y_\omega =
  \]

  \[
  = y_i f_R + f_i (K_a f_\omega - R - y_\omega - f_R) - L \frac{df_i}{dt} - K_a y_i f_\omega
  \]

  Sensitive to all faults except \( f_b \).

  Not surprising since \( e_3 \) was not used in the derivation of the residual.

Testable equation set?

- Is it possible to compute a residual from these equations?

  \[
  e_3 : T = J \frac{d\omega}{dt} + b\omega
  \]

  \[
  e_5 : i = y_i
  \]

  \[
  e_6 : \omega = y_\omega
  \]

  \[
  e_1 : V - iR - L \frac{di}{dt} - K_a i\omega = 0
  \]

  Yes! The values of \( \omega, i, \) and \( T \) can be computed using equations \( e_6, e_5, \) and \( e_3 \) respectively. Then there is an additional equation \( e_1 \) a so-called redundant equation that can be used for residual generation

  \[
  V - y_i R + L \frac{dy_i}{dt} - K_a y_i y_\omega = 0
  \]

  Compute the residual

  \[
  r = V - y_i R + L \frac{dy_i}{dt} - K_a y_i y_\omega
  \]

  and compare if it is close to 0.

Structural analysis provides the same information

- Model with fault:

  \[
  e_3 : T = J \frac{d\omega}{dt} + (b + f_b)\omega
  \]

  \[
  e_5 : i = y_i
  \]

  \[
  e_6 : \omega = y_\omega
  \]

  \[
  e_1 : V - i(R + f_R) - L \frac{di}{dt} - K_a i\omega = 0
  \]

- Structural analysis provides the following useful diagnosis information:

  - residual from \( \{ e_1, e_5, e_6 \} \)
  - sensitive to \( \{ f_i, f_\omega, f_R \} \)

  Let’s formalize the structural reasoning!
Matching

- A matching in a bipartite graph is a pairing of nodes in the two sets.
- Formally: set of edges with no common nodes.
- A matching with maximum cardinality is a maximal matching.

Diagnosis related interpretation: which variable is computed from which equation

\[
\begin{array}{c|c|c|c}
T & i & \omega \\
\hline
\text{e}_3 & \times & \times & \text{f}_b \\
\text{e}_5 & \times & & \text{f}_i \\
\text{e}_6 & \times & \times & \text{f}_\omega \\
\text{e}_1 & \times & \times & \text{f}_R \\
\end{array}
\]

Dulmage-Mendelsohn Decomposition

- Find a maximal matching
- Rearrange rows and columns
- Identify the under-, just-, and over-determined parts by backtracking
- Identify the block decomposition of the just-determined part. Erik will explain later.
- Dulmage-Mendelsohn decomposition can be done very fast for large models.

Dulmage-Mendelsohn decomposition

\[
M^+ = \{\text{e}_1, \text{e}_3, \text{e}_5, \text{e}_6\} \\
X^+ = \{i, \omega\} \\
\text{Faults in } M^+: \{f_1, f_\omega, f_R\}
\]

Detectable faults

- \(M^+\) is the overdetermined part of model \(M\).
- \(M^0\) is the exactly determined part of model \(M\).
- \(M^-\) is the underdetermined part of model \(M\).

The overdetermined part contains all redundancy.

**Structurally detectable fault**

Fault \(f\) is structurally detectable in \(M\) if \(f\) enters in \(M^+\).
Basic definitions - degree of redundancy

Degree of redundancy

Let $M$ be a set of equations in the unknowns $X$, then

$$\varphi(M) = |M^+| - |X^+|$$

Examples - electrical motor

Relation between overdetermined part and SO, MSO, and PSO sets.

- $M = \{e_1, e_3, e_6\}$ is SO since
  $$\varphi(M) = |M^+| - |X^+| = 3 - 2 = 1 > 0$$
  A residual can be computed but it is not sensitive to all faults in $M$.

- $M^+ = \{e_1, e_3, e_6\}$ is SO but also
  - PSO since the redundancy decreases if any equation is removed
  - MSO since there is no SO subset.

MSO and PSO sets seem to be more promising!

Basic definitions - overdetermined equation sets

Structurally Overdetermined (SO)

$M$ is SO if $\varphi(M) > 0$

Minimally Structurally Overdetermined (MSO)

An SO set $M$ is MSO if no proper subset of $M$ is SO.

Proper Structurally Overdetermined (PSO)

An SO set $M$ is PSO if $\varphi(E) < \varphi(M)$ for all proper subsets $E \subset M$.

Properties

- $M$ PSO set $\Leftrightarrow$ residual from $M$ sensitive to all faults in $M$
- MSO sets are PSO sets with structural redundancy 1.
- MSO sets are sensitive to few faults, which is good for fault isolation. $\Rightarrow$ MSO sets are candidates for residual generation
Conclusions so far

Structural properties:

- MSO set $\Leftrightarrow$ residual from $M$ sensitive to all faults in $M$
- MSO sets are PSO sets with structural redundancy 1.
- MSO sets are sensitive to few faults which is good for fault isolation.

$\Rightarrow$ MSO sets are candidates for residual generation

MSO and PSO models characterize model redundancy, but faults are not taken into account.

Next we will take faults into account.

Example: A state-space model

To illustrate the ideas I will consider the following small state-space model with 3 states, 3 measurements, and 5 faults:

$e_1 : x_1 = -x_1 + u + f_1$
$e_2 : x_2 = x_1 - 2x_2 + x_3 + f_2$
$e_3 : x_3 = x_2 - 3x_3$
$e_4 : y_1 = x_2 + f_3$
$e_5 : y_2 = x_2 + f_4$
$e_6 : y_3 = x_3 + f_5$

$x_i$ represent the unknown variables, $u$ and $y_i$ the known variables, and $f_i$ the faults to be monitored.

First observation: All MSO sets are not equally "good"

Tests sensitive to few faults give more precise isolation.

<table>
<thead>
<tr>
<th>Equations</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MSO_1$</td>
<td>${e_3, e_5, e_6}$</td>
</tr>
<tr>
<td>$MSO_2$</td>
<td>${e_3, e_4, e_5}$</td>
</tr>
<tr>
<td>$MSO_3$</td>
<td>${e_4, e_5}$</td>
</tr>
<tr>
<td>$MSO_4$</td>
<td>${e_1, e_2, e_3, e_6}$</td>
</tr>
<tr>
<td>$MSO_5$</td>
<td>${e_1, e_2, e_3, e_5}$</td>
</tr>
<tr>
<td>$MSO_6$</td>
<td>${e_1, e_2, e_3, e_4}$</td>
</tr>
<tr>
<td>$MSO_7$</td>
<td>${e_1, e_2, e_5, e_6}$</td>
</tr>
<tr>
<td>$MSO_8$</td>
<td>${e_1, e_2, e_4, e_6}$</td>
</tr>
</tbody>
</table>

In the definitions of redundancy, SO, MSO, and PSO we only considered equations and unknown variables.

But who cares about equations?

We are mainly interested in faults!

Conclusion 1

$MSO_7$ and $MSO_8$ are not minimal with respect to fault sensitivity
Second observation: Sometimes there are better test sets

A residual generator based on the equations in MSO\(_7\) will be sensitive to the faults:

\[ \text{Faults}(\{ e_1, e_2, e_5, e_6 \}) = \{ f_1, f_2, f_4, f_5 \} \]

Adding equation \( e_3 \) does not change the fault sensitivity:

\[ \text{Faults}(\{ e_1, e_2, e_3, e_5, e_6 \}) = \{ f_1, f_2, f_4, f_5 \} \]

Conclusion 2
There exists a PSO set larger than MSO\(_7\) with the same fault sensitivity.

Third observation: There are too many MSO sets

Consider the following model of a Scania truck engine

Original model:
- 532 equations
- 8 states
- 528 unknowns
- 4 redundant eq.
- 3 actuator faults
- 4 sensor faults

There are 1436 MSO sets in this model.

Conclusion 3
There are too many MSO sets to handle in practice and we have to find a way to sort out which sets to use for residual generator design.

Questions

<table>
<thead>
<tr>
<th>Equations</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO(_1)</td>
<td>{ e_3, e_5 }</td>
</tr>
<tr>
<td>MSO(_2)</td>
<td>{ e_3, e_4, e_5 }</td>
</tr>
<tr>
<td>MSO(_3)</td>
<td>{ e_4, e_5 }</td>
</tr>
<tr>
<td>MSO(_4)</td>
<td>{ e_1, e_2, e_3, e_6 }</td>
</tr>
<tr>
<td>MSO(_5)</td>
<td>{ e_1, e_2, e_3, e_4 }</td>
</tr>
<tr>
<td>MSO(_6)</td>
<td>{ e_1, e_2, e_3, e_5 }</td>
</tr>
<tr>
<td>MSO(_7)</td>
<td>{ e_1, e_2, e_5, e_6 }</td>
</tr>
<tr>
<td>MSO(_8)</td>
<td>{ e_1, e_2, e_4, e_6 }</td>
</tr>
</tbody>
</table>

What distinguish the first 6 MSO sets?

<table>
<thead>
<tr>
<th>Equations</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO(_1)</td>
<td>{ e_3, e_5 }</td>
</tr>
<tr>
<td>MSO(_2)</td>
<td>{ e_3, e_4, e_5 }</td>
</tr>
<tr>
<td>MSO(_3)</td>
<td>{ e_4, e_5 }</td>
</tr>
<tr>
<td>MSO(_4)</td>
<td>{ e_1, e_2, e_3, e_6 }</td>
</tr>
<tr>
<td>MSO(_5)</td>
<td>{ e_1, e_2, e_3, e_4 }</td>
</tr>
<tr>
<td>MSO(_6)</td>
<td>{ e_1, e_2, e_3, e_5 }</td>
</tr>
<tr>
<td>MSO(_7)</td>
<td>{ e_1, e_2, e_4, e_6 }</td>
</tr>
<tr>
<td>MSO(_8)</td>
<td>{ e_1, e_2, e_4, e_6 }</td>
</tr>
</tbody>
</table>

Is it always MSO sets we are looking for?
Questions

How do we characterize the PSO set $\text{MSO}_7 \cup \{e_3\}$, which has the properties

- It is not an MSO set.
- It has the same fault sensitivity as an MSO set.

Answers

Let $F(M)$ denote the set of faults included in $M$.

Definition (Test Support)

Given a model $M$ and a set of faults $F$, a non-empty subset of faults $\zeta \subseteq F$ is a test support if there exists a PSO set $M \subseteq M$ such that $F(M) = \zeta$.

Definition (Test Equation Support)

An equation set $M$ is a Test Equation Support (TES) if

1. $M$ is a PSO set,
2. $F(M) \neq \emptyset$, and
3. for any $M' \supseteq M$ where $M'$ is a PSO set it holds that $F(M') \supseteq F(M)$.

$\text{MSO}_7$ is not a TES since

$\text{Faults}([e_1, e_2, e_5]) = \text{Faults}([e_1, e_2, e_3, e_5, e_6]) = \{f_1, f_2, f_4, f_5\}$

Fundamental questions

- Which fault sensitivities are possible?
- For a given possible fault sensitivity, which sub-model is the best to use?

Answers

Definition (Minimal Test Support)

Given a model, a test support is a minimal test support (MTS) if no proper subset is a test support.

Definition (Minimal Test Equation Support)

A TES $M$ is a minimal TES (MTES) if there exists no subset of $M$ that is a TES.
### Example

<table>
<thead>
<tr>
<th>Equations</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSO₁</td>
<td>{e₃, e₅, e₆}</td>
</tr>
<tr>
<td>MSO₂</td>
<td>{e₃, e₄, e₆}</td>
</tr>
<tr>
<td>MSO₃</td>
<td>{e₄, e₅}</td>
</tr>
<tr>
<td>MSO₄</td>
<td>{e₁, e₂, e₃, e₆}</td>
</tr>
<tr>
<td>MSO₅</td>
<td>{e₁, e₂, e₃, e₅}</td>
</tr>
<tr>
<td>MSO₆</td>
<td>{e₁, e₂, e₃, e₄}</td>
</tr>
<tr>
<td>MSO₇</td>
<td>{e₁, e₂, e₅, e₆}</td>
</tr>
<tr>
<td>MSO₈</td>
<td>{e₁, e₂, e₄, e₆}</td>
</tr>
</tbody>
</table>

- The MTES are the first 6 MSO sets. (fewer MTESs than MSOs)
- The 2 last not even a TES.
- The TES corresponding to last TS:s are \{e₁, e₂, e₃, e₅, e₆\}, \{e₁, e₂, e₃, e₄, e₆\}

### Summary

Consider a model $M$ with faults $\mathcal{F}$.

**TS/TES**
- $\zeta \subseteq \mathcal{F}$ is a TS $\iff$ there is a residual sensitive to the faults in $\zeta$
- The TES corresponding to $\zeta$ can easily be computed.

**MTES are**
- typically MSO sets.
- fewer than MSO sets.
- sensitive to minimal sets of faults.
- sufficient and necessary for maximum multiple fault isolability

$\Rightarrow$ candidates for deriving residuals

---

### Outline

- Introduction
- Structural models and basic definitions
- Diagnosis system design
- Residual generation
- Diagnosability analysis
- Sensor placement analysis
- Case study and software demonstration
- Analytical vs structural properties
- Concluding remarks
Design system design supported by structural methods

A successful approach to diagnosis is to design a set of residual generators with different fault sensitivities.

Designing diagnosis system utilizing structural analysis

1. Find (all) testable models (MSO/MTES/...)
2. Select a subset of testable models with required fault isolability
3. From each selected testable model generate code for the corresponding residual.

Algorithms covered here

- Basic MSO algorithm
- Improved MSO algorithm
- MTES algorithm

Dulmage-Mendelsohn decomposition

A cornerstone in the MSO-algorithm is the Dulmage-Mendelsohn decomposition.

- In this algorithm we will only use it to find the overdetermined part $M^+$ of model $M$ because
- All MSO sets are contained in the overdetermined part.

Finding MSO sets

- MSO sets are found by alternately removing equations and computing the overdetermined part.

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(3)</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Properties of an MSO:

- A structurally overdetermined part is an MSO set if and only if
  - $\#\text{ equations} = \#\text{ unknowns} + 1$
- The degree of redundancy decreases with one for each removal.
Basic algorithm

- Try all combinations

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- Remove (1)
- Get overdetermined part
  - Remove (4)
  - Get overdetermined part
    - $\Rightarrow \ (6)(7) \ MSO!$  
  - Remove (5)
  - Get overdetermined part
    - $\Rightarrow \ (6)(7) \ MSO!$  
  - Remove (6) . . .
- Remove (2) . . .

The same MSO set is found several times

- Example: Removing (1) and then (4) resulted in the MSO (6)(7).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- Remove (4)
- Remove (1)
- $\ (6)(7) \ MSO!$

- If the order of removal is permuted, the same MSO set is obtained.
  $\Rightarrow$ Permutations of the order of removal will be prevented.

The same MSO set is found several times

- Removal of different equations will sometimes result in the same overdetermined part.

Exploit this by defining equivalence classes on the set of equations

Basic algorithm

The basic algorithm is very easy to implement. In pseudo-code (feed with $M^+$):

```pseudo
def $M_{MSO}$ = FindMSO($M$)
    if $\varphi(M)=1$
        $M_{MSO} := \{M\}$
    else
        $M_{MSO} := \emptyset$
        for each $e \in M$
            $M' = (M \setminus \{e\})^+$
            $M_{MSO} := M_{MSO} \cup$ FindMSO($M'$)
        end
    end
```

Exploit this by defining equivalence classes on the set of equations.
Equivalence classes

Let $M$ be the model consisting of a set of equations. Equation $e_i$ is related to equation $e_j$ if

$$e_i \notin (M \setminus \{e_j\})^+$$

It can easily be proven that this is an equivalence relation. Thus, $[e]$ denotes the set of equations that is not in the overdetermined part when equation $e$ is removed.

Equivalence classes

The same overdetermined part will be obtained independent on which equation in an equivalence class that is removed.

Unique decomposition of an over-determined part

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$X$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>$X$</td>
<td>$X$</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td></td>
<td>$X$</td>
</tr>
<tr>
<td>(5)</td>
<td></td>
<td></td>
<td>$X$</td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td>$X$</td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$M_1 = \{(1)(2)(3)\}$  $X_1 = \{x_1, x_2\}$
$M_2 = \{(4)(5)\}$     $X_2 = \{x_3\}$
$M_3 = \{(6)\}$        $X_3 = \emptyset$
$M_4 = \{(7)\}$        $X_4 = \emptyset$
$X_0 = \{x_4\}$

- $|M_i| = |X_i| + 1$
- All MSO sets can be written as a union of equivalence classes, e.g.
  $\{(6)(7)\} = M_3 \cup M_4$
  $\{(4)(5)(6)\} = M_2 \cup M_3$

Lumping

- The equivalence classes can be lumped together forming a reduced structure.

Original structure:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>$X$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>$X$</td>
<td>$X$</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
</tr>
<tr>
<td>(4)</td>
<td></td>
<td></td>
<td>$X$</td>
</tr>
<tr>
<td>(5)</td>
<td></td>
<td></td>
<td>$X$</td>
</tr>
<tr>
<td>(6)</td>
<td></td>
<td>$X$</td>
<td></td>
</tr>
<tr>
<td>(7)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Lumped structure:

<table>
<thead>
<tr>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
</tr>
</tbody>
</table>

- There is a one to one correspondence between MSO sets in the original and in the lumped structure.
- The lumped structure can be used to find all MSO sets.

Equivalence classes

Any PSO set can be written on the canonical form

$X_1 X_2 \cdots X_n X_0$

$M_1$

$M_2$

$\vdots$

$M_n$

$M_{n+1}$

$\vdots$

$M_m$

This form will be useful for
- improving the basic algorithm (now)
- performing diagnosability analysis (later)

Can be obtained easily with attractive complexity properties

Lumping

- The equivalence classes can be lumped together forming a reduced structure.
Improved algorithm

- The same principle as the basic algorithm.
- Avoids that the same set is found more than once.
  - Prohibits permutations of the order of removal.
  - Reduces the structure by lumping.

MSO algorithm: We start with the complete model

\[ \{ e_1, e_2, e_3, e_4, e_5, e_6 \} \]

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_1 )</td>
<td>( X )</td>
<td></td>
</tr>
<tr>
<td>( e_2 )</td>
<td>( X )</td>
<td>( X )</td>
</tr>
<tr>
<td>( e_3 )</td>
<td>( X )</td>
<td>( X )</td>
</tr>
<tr>
<td>( e_4 )</td>
<td>( X )</td>
<td></td>
</tr>
<tr>
<td>( e_5 )</td>
<td>( X )</td>
<td></td>
</tr>
<tr>
<td>( e_6 )</td>
<td>( X )</td>
<td></td>
</tr>
</tbody>
</table>

MSO algorithm: Remove \( e_1 \) and compute \( (M \setminus \{ e_1 \})^+ \)

\[ \{ e_1, e_2, e_3, e_4, e_5, e_6 \} \]

\[ \{ e_3, e_4, e_5, e_6 \} \]

$\mathbf{x}_i$ represent the unknown variables, $u$ and $y_i$ the known variables, and $f_i$ the faults to be monitored.

Let's consider this example again

| \( e_1 \) | \( x_1 \) | \( x_2 \) | \( x_3 \) |
|---|---|---|
| \( e_1 \) | \( X \) |
| \( e_2 \) | \( X \) | \( X \) | \( X \) |
| \( e_3 \) | \( X \) | \( X \) |
| \( e_4 \) | \( X \) |
| \( e_5 \) | \( X \) |
| \( e_6 \) | \( X \) |
MSO algorithm: Remove $e_3$

\[
\begin{align*}
\{e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{e_3, e_4, e_5, e_6\} \\
\{e_4, e_5\}
\end{align*}
\]

\[
\begin{array}{ccc}
    x_1 & x_2 & x_3 \\
    e_1 & X & \\
    e_2 & X & X & X \\
    e_3 & X & X \\
    e_4 & X \\
    e_5 & X \\
    e_6 & X \\
\end{array}
\]

MSO algorithm: Go back and remove $e_4$

\[
\begin{align*}
\{e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{e_3, e_4, e_5, e_6\} \\
\{e_4, e_5\}
\end{align*}
\]

\[
\begin{array}{ccc}
    x_1 & x_2 & x_3 \\
    e_1 & X & \\
    e_2 & X & X & X \\
    e_3 & X & X \\
    e_4 & X \\
    e_5 & X \\
    e_6 & X \\
\end{array}
\]

MSO algorithm: Go back and remove $e_5$

\[
\begin{align*}
\{e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{e_3, e_4, e_5, e_6\} \\
\{e_4, e_5\} \\
\{e_3, e_5, e_6\} \\
\{e_3, e_4, e_6\}
\end{align*}
\]

\[
\begin{array}{ccc}
    x_1 & x_2 & x_3 \\
    e_1 & X & \\
    e_2 & X & X & X \\
    e_3 & X & X \\
    e_4 & X \\
    e_5 & X \\
    e_6 & X \\
\end{array}
\]

MSO algorithm: Go back 2 steps and remove $e_3$

\[
\begin{align*}
\{e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{e_3, e_4, e_5, e_6\} \\
\{e_1, e_2, e_4, e_5, e_6\} \\
\{e_3, e_4, e_6\} \\
\{e_3, e_5, e_6\} \\
\{e_3, e_4, e_6\}
\end{align*}
\]

\[
\begin{array}{ccc}
    x_1 & x_2 & x_3 \\
    e_1 & X & \\
    e_2 & X & X & X \\
    e_3 & X & X \\
    e_4 & X \\
    e_5 & X \\
    e_6 & X \\
\end{array}
\]
**Summary - MSO algorithm**

- An algorithm for finding all MSO sets for a given model structure
- Main ideas:
  - Top-down approach
  - Structural reduction based on the unique decomposition of overdetermined parts
  - Prohibit that any MSO set is found more than once.


---

**MTES algorithm**

I will now present the algorithm that finds all MTESs and TESs.


It is a slight modification of the MSO algorithm.

**Basic idea**

There’s no point removing equations that doesn’t contain faults, since we are interested in fault sensitivity.

**Modification**

Stop doing that!

---

In the example $e_3$ is the only equation without fault.

We will not remove $e_3$

We remove $e_4$ instead.

The nodes are TES:s and the leaves are MTES:s.
All TSs and TESs for the model

The algorithm traverses all TESs

\[
\begin{align*}
\{f_1, f_2, f_3, f_4, f_5\} \\
\{e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{f_3, f_4, f_5\} \\
\{f_1, f_2, f_3, f_5\} \\
\{f_1, f_2, f_3, f_4\} \\
{f_1, f_2, f_3, e_4, e_5, e_6} \\
\{f_1, e_1, e_2, e_3, e_4, e_5, e_6\} \\
\{e_1, e_2, e_3, e_4, e_5\} \\
\{e_1, e_2, e_3, e_4, e_5\} \\
\{e_1, e_2, e_3, e_4, e_6\} \\
\{e_1, e_2, e_3, e_4, e_5\} \\
\{e_1, e_2, e_3, e_4, e_5\} \\
\end{align*}
\]

Test selection

- Many candidate residual generators (MSOs/MTESs) can be computed, only a few needed for single fault isolation.
- Realization of a residual generator is computationally demanding.

Careful selection of which test to design in order to achieve the specified diagnosis requirements with few tests.

Scania truck engine example

Original model:
- 532 equations
- 8 states
- 528 unknowns
- 4 redundant eq.
- 3 actuator faults
- 4 sensor faults

- Reduces the resulting number of testable sets:
  - 1436 MSO sets cmp. to 32 MTESs which all are MSOs.
  - Only 6 needed for full single fault isolation.
- Reduces the computational burden:
  - 1774 PSO sets ∼ runtime MSO-alg. (2.5 s)
  - 61 TESs ∼ runtime MTES-alg. (0.42 s)
  - Few number of faults cmp to the number of equations.

Problem formulation

Test selection problem

Given:
- A fault signature matrix (e.g. based on MSO sets/MTES)
- A desired fault isolability (e.g. specified as an isolability matrix)

Output: A small set of tests with required isolability
Fault isolability of tests

<table>
<thead>
<tr>
<th>NF</th>
<th>$f_1$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

- $T$ no alarm $\Rightarrow$ NF, $f_1$, $f_2$ consistent
- $T$ alarm $\Rightarrow f_1$ consistent

- $f_1$ detectable
- $f_1$ isolable from $f_2$
- $f_2$ not isolable from $f_1$

Test selection

- Find all minimal test sets with a minimal hitting set algorithm.
  - Might easily lead to computationally intractable problems.


- Find an approximate minimum cardinality hitting set
  - A greedy search for one small set of tests. Fast with good complexity properties, but cannot guarantee to find the smallest set of tests.


- Iterative approach involving both test selection and residual generation.

Test selection is a minimal hitting set problem

- Requirement for each desired diagnosability property:
  - **Detectability:** $f_R$: $T_1 = \{3, 4, 5, 6\}$
  - $\ldots$
  - **Isolability:**
    - $f_R$ isol. from $f_i$: $T_2 = \{3, 5\}$
    - $f_i$ isol. from $f_R$: $T_3 = \{1\}$
    - $f_R$ isol. from $f_\omega$: $T_4 = \{5, 6\}$
    - $\ldots$

**Test selection $T$**

A minimal set of tests $T$ is a solution if $T \cap T_i \neq \emptyset$ for all desired diagnosability properties $i$.

Test selection

- Many more alternatives in for example:

Minimal test sets for full single fault isolability: \{1, 2, 4, 5\}, \{1, 2, 3, 5\}, \{1, 2, 3, 6\}

Assume that we do not care to isolate \(f_R\) and \(f_i\), i.e., the desired isolability can be specified as:

\[
\begin{array}{c|cccc}
 & f_R & f_i & f_\omega & f_T \\
\hline
f_R & 3 & 6 & 3, 5 & 5, 6 & 3, 4 \\
f_i & 1, 4, 6 & 1 & 1, 6 & 1, 4 \\
f_\omega & 2 & 4 & 2, 3 & 3, 4 \\
f_T & 2, 5, 6 & 2 & 2, 5 & 5, 6 & 5, 4 \\
\end{array}
\]

Minimum cardinality solution: \{2, 4, 6\}

---

Greedy search incorporating residual generation

**Basic idea**

Select residuals adding the most number of desired diagnosis properties.

<table>
<thead>
<tr>
<th>(r_1)</th>
<th>(f_1)</th>
<th>(f_2)</th>
<th>(f_3)</th>
<th>(f_R)</th>
<th>(f_i)</th>
<th>(f_\omega)</th>
<th>(f_T)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(r_2)</td>
<td>(X)</td>
<td>(X)</td>
<td></td>
<td>1, 2, 4</td>
<td>2, 4</td>
<td>1, 4</td>
<td></td>
</tr>
<tr>
<td>(r_3)</td>
<td>(X)</td>
<td>(X)</td>
<td></td>
<td>1, 3</td>
<td>3, 2</td>
<td>1, 4</td>
<td></td>
</tr>
<tr>
<td>(r_4)</td>
<td>(X)</td>
<td></td>
<td></td>
<td>2, 3</td>
<td>3, 2</td>
<td>1, 4</td>
<td></td>
</tr>
</tbody>
</table>

- Select residual generator 1. Realization pass.
- Select residual generator 2. Realization fails.
- Select residual generator 3. Realization pass.
- Select residual generator 4. Realization pass.

---

Outline

- Introduction
- Structural models and basic definitions
- Diagnosis system design
- Residual generation
- Diagnosability analysis
- Sensor placement analysis
- Case study and software demonstration
- Analytical vs structural properties
- Concluding remarks
Residual generation and structural analysis

- Structural analysis of model can be of good help
- A matching gives information which equations can be used to (in a best case) compute/estimate unknown variables
- Careful treatment of dynamics
- Again, not general solutions but helpful approaches in your diagnostic toolbox

Two types of methods covered here
- Sequential residual generation
- Observer based residual generation

Sequential residual generation

Basic idea

**Given**: A set of equations with redundancy
**Approach**: Choose computational sequence for the unknown variables and check consistency in redundant equations

- Popular in DX community
- Easy to automatically generate residual generators from a given model
- Choice how to interpret differential constraints, derivative/integral causality
- Interesting, but not without limitations

Sequential residual generation

5 equations, 4 unknowns

<table>
<thead>
<tr>
<th>Equation</th>
<th>x1</th>
<th>x2</th>
<th>x4</th>
<th>x3</th>
</tr>
</thead>
<tbody>
<tr>
<td>e1</td>
<td>x1  - x2 = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e2</td>
<td>x3  - x4 = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e3</td>
<td>x4x1 + 2x2x4 - y1 = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e4</td>
<td>x3  - y3 = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e5</td>
<td>x2  - y2 = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solve according to order in decomposition:

\[
\begin{align*}
e_4 & : x_3 := y_3 \\
e_3 & : \dot{x}_1 := x_2 \\
e_2 & : x_4 := \dot{x}_3 \\
e_1 & : \dot{x}_2 := \frac{-x_4x_1 + y_1}{2x_4}
\end{align*}
\]

Compute a residual:

\[
e_5 : r := y_2 - x_2
\]
Illustrative example

Find overdetermined sets of equations
There are 6 MSO sets for the model, for illustration, use
\[ \mathcal{M} = \{ e_1, e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11} \} \]
Redundancy 1: 8 eq., 7 unknown variables \((q_0, q_1, q_2, p_1, p_2, \dot{p}_1, \dot{p}_2)\)

\[
\begin{align*}
    e_1 &: q_1 = \frac{1}{R_{V_1}}(p_1 - p_2) \\
    e_2 &: q_2 = \frac{1}{R_{V_2}}(p_2 - p_3) \\
    e_3 &: q_3 = \frac{1}{R_{V_3}}(p_3) \\
    e_4 &: \dot{p}_1 = \frac{1}{C_{T_1}}(q_0 - q_1) \\
    e_5 &: \dot{p}_2 = \frac{1}{C_{T_2}}(q_1 - q_2) \\
    e_6 &: \dot{y}_2 = q_2 \\
    e_7 &: y_1 = p_1 \\
    e_8 &: y_2 = q_2 \\
    e_9 &: y_3 = q_0 \\
    e_{10} &: \dot{p}_1 = \frac{dp_1}{dt} \\
    e_{11} &: \dot{p}_2 = \frac{dp_2}{dt}
\end{align*}
\]

Redundant equation
For illustration, choose equation \(e_5\) as a redundant equation, i.e., compute unknown variables using \((e_1, e_4, e_7, e_8, e_9, e_{10}, e_{11})\)

\[
\begin{array}{cccccccc}
    e_1 & e_{11} & e_4 & e_7 & e_8 & e_{10} & e_9 & e_2 \\
    \dot{p}_2 & \dot{p}_2 & q_1 & \dot{p}_1 & p_1 & q_0 & q_2
\end{array}
\]

Computational graph for matching

Equations \(e_{10}\) and \(e_{11}\) in derivative causality.
Residual generator code

Fairly straightforward to generate code automatically for this case

Code

\[
\begin{align*}
q_2 &= y_2; \quad \% e_8 \\
q_0 &= y_3; \quad \% e_9 \\
p_1 &= y_1; \quad \% e_7 \\
dp_1 &= \text{ApproxDiff}(p_1, \text{state}.p1, Ts); \quad \% e_{10} \\
q_1 &= q_0 - CT_1 * dp_1; \quad \% e_4 \\
p_2 &= p_1 - Rv_1 * q_1; \quad \% e_1 \\
dp_2 &= \text{ApproxDiff}(p_2, \text{state}.p2, Ts); \quad \% e_{11} \\
r &= dp_2 - (q_1 - q_2) / CT_2; \quad \% e_5
\end{align*}
\]

Causality of sequential residual generators

- Derivative causality
  + No stability issues
  - Numerical differentiation highly sensitive to noise
- Integral causality
  - Stability issues
  + Numerical integration good wrt. noise
- Mixed causality - a little of both

Not easy to say which one is always best, but generally integration is preferred to differentiation

Matching and Hall components

Here the matching gives a computational sequence for all variables

Important!

This is generally not true
Hall components & Dulmage-Mendelsohn decomposition

- The blocks in the exactly determined part is called Hall components
- If a Hall component is of size 1; compute variable \(x_i\) in equation \(e_i\)
- If Hall component is larger (always square) than 1 \(\Rightarrow\) system of equations that need to be solved simultaneously

Observer based residual generation

The basic idea in observer based residual generation is the same as in sequential residual generation

- Estimate/compute unknown variables \(\hat{x}\)
- Check if model is consistent with \(\hat{x}\)

With an observer the most basic setup model/residual generator is

\[
\dot{x} = g(x, u) \\
y = h(x, u)
\]

\[
\dot{\hat{x}} = g(\hat{x}, u) + K(y - h(\hat{x}, u))
\]

Design procedures typically available for state-space models

- pole placement
- EKF/UKF/Monte-Carlo filters
- Sliding mode
- ...

Hall components and computational loops

- Two Hall components of size 1 and one of size 2
  \((x_3, e_4) \rightarrow (x_4, e_2) \rightarrow \{\{x_1, x_2\}, \{e_1, e_5\}\})\]

- If only algebraic constraints \(\Rightarrow\) algebraic loop
- If differential constraint \(\Rightarrow\) loop in integral causality

A matching finds computational sequences, including identifying computational loops

DAE models

DAE model

An MSO/submodel consists of a number of equations \(g_i\), a set of dynamic variables \(x_1\), and a set of algebraic variables \(x_2\)

\[
g_i(dx_1, x_1, x_2, z, f) = 0 \quad i = 1, \ldots, n
\]

\[
dx_1 = \frac{d}{dt}x_1
\]

- A DAE model where you can solve for highest order derivatives \(dx_1\) and \(x_2\), is called a low-index, or low differential-index, DAE model.
- Essentially equivalent to state-space models

For structurally low-index problems, code for observers can be generated

Submodels like MSE/MTES are not typically in state-space form!
**Example: Three Tank example again**

\[
e_1 : q_1 = \frac{1}{R_{V1}}(p_1 - p_2) \quad e_5 : \dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2) \quad e_8 : y_2 = q_2
\]

\[
e_4 : \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \quad e_7 : y_1 = p_1 \quad e_9 : y_1 = q_0
\]

The MSO \(\mathcal{M} = \{e_1, e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}\}\)

This is not a state-space form, suitable for standard observer design techniques. But it is low-index so it is close enough.

**Partition model using structure**

<table>
<thead>
<tr>
<th>Dynamic equations</th>
<th>Algebraic equations</th>
<th>Redundant equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[\dot{p}<em>1 = \frac{1}{C</em>{T1}}(q_0 - q_1)]</td>
<td>[0 = q_0 - y_3]</td>
<td>[r = y_1 - p_1]</td>
</tr>
<tr>
<td>[\dot{p}<em>2 = \frac{1}{C</em>{T2}}(q_1 - q_2)]</td>
<td>[0 = q_1 R_{V1} - (p_1 - p_2)]</td>
<td></td>
</tr>
</tbody>
</table>

**Models with low differential index**

A low-index DAE model

\[
g_i(dx_1, x_1, x_2, z, f) = 0 \quad i = 1, \ldots, n
\]

\[
dx_1 = \frac{d}{dt}x_1 \quad i = 1, \ldots, m
\]

has the property

\[
\begin{pmatrix}
\frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2}
\end{pmatrix}_{x = x_0, \ z = z_0}
\]

full column rank

Structurally, this corresponds to a maximal matching with respect to \(dx_1\) and \(x_2\) in the model structure graph.

Model can be transformed into the form

\[
\dot{x}_1 = g_1(x_1, x_2, z, f)
\]

\[
0 = g_2(x_1, x_2, z, f), \quad \frac{\partial g_2}{\partial x_2} \text{ is full column rank}
\]

\[
0 = g_r(x_1, x_2, z, f)
\]

**DAE observer for low-index model**

For a model in the form

\[
\dot{x}_1 = g_1(x_1, x_2, z, f)
\]

\[
0 = g_2(x_1, x_2, z, f), \quad \frac{\partial g_2}{\partial x_2} \text{ is full column rank}
\]

\[
0 = g_r(x_1, x_2, z, f)
\]

a DAE-observer can be formed as

\[
\dot{\hat{x}}_1 = g_1(\hat{x}_1, \hat{x}_2, z) + K_1 r
\]

\[
\dot{\hat{x}}_2 = g_2(\hat{x}_1, \hat{x}_2, z)
\]

\[
0 = g_r(\hat{x}_1, \hat{x}_2, z)
\]

The observer estimates \(x_1\) and \(x_2\), and then a residual can be computed as

\[
r = g_r(\hat{x}_1, \hat{x}_2, z)
\]

**Important:** Very simple approach, no guarantees of observability of performance
The observer

\[
\dot{\hat{x}}_1 = g_1(\hat{x}, \hat{x}_2, z) + K(\hat{x}, z) g_r(\hat{x}, \hat{x}_2, z) \\
0 = g_2(\hat{x}, \hat{x}_2, z) \\
r = g_r(\hat{x}_1, \hat{x}_2, z)
\]

corresponds to the standard setup DAE

\[
M \dot{\hat{w}} = \begin{pmatrix}
g_1(\hat{x}_1, \hat{x}_2, z) + K(\hat{x}, z) g_r(\hat{x}_1, \hat{x}_2, z) \\
g_2(\hat{x}_1, \hat{x}_2, z) \\
r - g_r(x_1, x_2, z)
\end{pmatrix} = F(w, z)
\]

where the mass matrix \( M \) is given by

\[
M = \begin{pmatrix}
I_{n_1} & 0_{n_1 \times (n_2 + n_r)} \\
0_{(n_2 + n_r) \times n_1} & 0_{(n_2 + n_r) \times (n_2 + n_r)}
\end{pmatrix}
\]

Diagnozability analysis

Run the residual generator

Low-index DAE models and ODE solvers

A dynamic system in the form

\[
M \dot{x} = f(x)
\]

with mass matrix \( M \) possibly singular, can be integrated by (any) stiff ODE solver capable of handle low-index DAE models. 

**Example:** ode15s in Matlab.

- Fairly straightforward, details not included, to generate code for function \( f(x) \) above for low-index problems
- Code generation similar to the sequential residual generators, but only for the highest order derivatives
- Utilizes efficient numerical techniques for integration

Outline

- Introduction
- Structural models and basic definitions
- Diagnosis system design
- Residual generation
- Diagnozability analysis
- Sensor placement analysis
- Case study and software demonstration
- Analytical vs structural properties
- Concluding remarks
Problem formulation

Given a dynamic model: What are the fault isolability properties?

Diagnosability analysis

Diagnosability analysis

Determine for a
- model
- diagnosis system
which faults that are structurally detectable and what are the structural isolability properties.

MSO based approach

Since the set of MSOs characterize all possible fault signatures, the MSOs can be used to determine structural isolability of a given model. Often computationally intractable. Just too many.

Better way

Utilize steps in the MSO algorithm; equivalence classes!

Isolability matrices

Interpretation

A X in position (i,j) indicates that fault $f_i$ cannot be isolated from fault $f_j$.

Diagnosability analysis for a set of tests/model

A test/residual with fault sensitivity

\[
\begin{array}{c|cc}
\text{r} & f_1 & f_2 \\
\hline
X & 0 \\
\end{array}
\]

makes it possible to isolate fault $f_1$ from fault $f_2$. Now, consider single fault isolability with a diagnosis system with the fault signature matrix

\[
\begin{array}{ccc}
r_1 & f_1 & f_2 & f_3 \\
X & X & 0 \\
r_2 & 0 & X & X \\
\end{array}
\]

The corresponding isolability matrix is then

\[
\begin{array}{ccc}
f_1 & f_2 & f_3 \\
X & X & 0 \\
f_2 & 0 & X \\
f_3 & 0 & X & X \\
\end{array}
\]
**Structural fault modelling**

**Assumption**
A fault $f$ only violates 1 equation, referred to by $e_f$.

If a fault signal $f$ appears in more than one position in the model,

$$
e_1 : 0 = g_1(x_1, x_2) + x_f$$
$$e_2 : 0 = g_2(x_1, x_2) + x_f$$
$$e_3 : x_f = f$$

- Introduce new unknown variable $x_f$
- Add new equation $x_f = f$

Now, the model fulfills the assumption.

**Detectability in small example**

\[
\begin{align*}
e_1 : & \quad x_1 = -x_1 + x_2 + x_3 \\
e_2 : & \quad x_2 = -2x_2 + x_3 + x_4 \\
e_3 : & \quad x_3 = -3x_3 + x_5 + f_1 + f_2 \\
e_4 : & \quad x_4 = -4x_4 + x_5 + f_3 \\
e_5 : & \quad x_5 = -5x_5 + u + f_4 \\
e_6 : & \quad y_1 = x_1 \\
e_7 : & \quad y_2 = x_3
\end{align*}
\]

**Structural detectability and Dulmage-Mendelsohn**

**Detectability**
A fault $f$ is structurally detectable if $e_f \in M^+$.

- Fault $f_1$ not detectable
- Fault $f_2$ detectable

**Structural isolability**

**Isolability**
A fault $F_i$ is isolable from fault $F_j$ if $O(F_i) \nsubseteq O(F_j)$

Meaning, there exists observations from the faulty mode $F_i$ that is not consistent with the fault mode $F_j$.

- Structurally, this corresponds to the existence of an MSO that include $e_i$ but not $e_j$

\[
\begin{array}{c|c|c}
F_i & F_j & r \\
\hline
F_i & 0 & 1 \\
\end{array}
\]

- or equivalently, fault $F_i$ is detectable in the model where fault $F_j$ is decoupled

**Structural isolability**

$F_i$ structurally isolable from $F_j$ iff $e_i \in (M \setminus \{e_j\})^+$

Structural single fault isolability can thus be determined by $n_f^2$ $M^+$-operations. For single fault isolability, we can do better.
Equivalence classes and isolability

From before we know that $M^+$ of a model can be always be written on the canonical form

\[
\begin{array}{cccc}
X_1 X_2 \cdots X_n & \color{red}{X_0} \\
\hline
M_1 & +1 \\
M_2 & +1 \\
\vdots & \ddots \\
M_n & +1 \\
M_{n+1} & \ddots \\
M_m & \\
\end{array}
\]

- Equivalence classes $M_i$ has the defining property: remove one equation $e$, then none of the equations are members of $(M \setminus \{ e \})^+$
- Detectable faults are isolable if and only if they influence the model in different equivalence classes

Isolability from fault $f_3$ in small example

\[
e_1: \quad \dot{x}_1 = -x_1 + x_2 + x_5 \\
e_2: \quad \dot{x}_2 = -2x_2 + x_3 + x_4 \\
e_3: \quad \dot{x}_3 = -3x_3 + x_6 + f_1 + f_2 \\
e_4: \quad \dot{x}_4 = -4x_4 + x_5 + f_3 \\
e_5: \quad \dot{x}_5 = -5x_5 + u + f_4 \\
e_6: \quad y_1 = x_1 \\
e_7: \quad y_2 = x_3
\]

Equivalence class $[e_4]$

\[[e_4] = \{e_1, e_2, e_4, e_6\}\]

Example system - A automotive engine with EGR/VGT

Method - Diagnosability analysis of model

- Determine equivalence classes in $M^+$:
  \[M_{e_i} = M \setminus \{ e_i \}\]
  \[[e_i] = M^+ \setminus M_{e_i}^+\]
- Faults appearing in the same equivalence class are not isolable
- Faults appearing in separate equivalence classes are isolable
Fault isolation matrix for engine model

Diagnosability analysis for a fault signature matrix

**Isolability properties of a set of residual generators**

Previous results: structural diagnosability properties of a model, what about diagnosability properties for a diagnosis system

A test with fault sensitivity

\[
\begin{array}{c|cc}
\toprule
i & f_i & f_j \\
\midrule
f_1 & X \\
\bottomrule
\end{array}
\]

isolates fault \( f_i \) from \( f_j \).

For example, MSO2 isolates

- Fault \( f_w \) from \( f_R \) and \( f_i \),
- Fault \( f_T \) from \( f_R \) and \( f_i \)
Diagnosability analysis for a fault signature matrix

Rule: Diagnosability properties for a FSM

Fault \( f_i \) is isolable from fault \( f_j \) if there exists a residual sensitive to \( f_i \) but not \( f_j \)

Fault isolation and exoneration

Fault \( f_3 \) occurs at \( t = 2 \) sec.

<table>
<thead>
<tr>
<th>( M_1 )</th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\( f_1 \) | \( f_2 \) | \( f_3 \) | \( f_4 \) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Q: Why is not the isolability matrix diagonal when all columns in FSM are different?
A: We do not assume exoneration (= ideal residual response), exoneration is a term from consistency based diagnosis, here isolation by column matching

CBD diagnosis

\( r_1 > J \Rightarrow f_3 \) or \( f_4 \) \( \Rightarrow D_1 = \{ f_3 \}, D_2 = \{ f_1, f_4 \} \)

Diagnosis result

No exoneration assumption | With exoneration assumption
---|---
\( 0 - 2.5 \) : No fault | \( 0 - 2.5 \) : No fault
\( 2.5 - 6 \) : \( f_3 \) or \( f_4 \) | \( 2.5 - 6 \) : Unknown
\( 6 - \) : \( f_3 \) | \( 6 - \) : \( f_3 \)
A motivating example and problem formulation

\[ e_1 : \dot{x}_1 = -x_1 + x_2 + x_5 \]
\[ e_2 : \dot{x}_2 = -2x_2 + x_3 + x_4 \]
\[ e_3 : \dot{x}_3 = -3x_3 + x_5 + f_1 + f_2 \]
\[ e_4 : \dot{x}_4 = -4x_4 + x_5 + f_3 \]
\[ e_5 : \dot{x}_5 = -5x_5 + u + f_4 \]

Question: Where should I place sensors to make faults \( f_1, \ldots, f_4 \) detectable and isolable, as far as possible?

For example:
- \( \{ x_1 \}, \{ x_2 \}, \{ x_3, x_4 \} \Rightarrow \) detectability of all faults
- \( \{ x_1, x_3 \}, \{ x_1, x_4 \}, \{ x_2, x_3 \}, \{ x_2, x_4 \}, \{ x_3, x_4 \} \Rightarrow \) maximum, not full, fault isolability of \( f_1, \ldots, f_4 \)
- \( \{ x_1, x_1, x_3 \} \Rightarrow \) Possible to isolate also faults in the new sensors

More than one solution, how to characterize all solutions?

Minimal sensor sets and problem formulation

Given:
- A set \( P \) of possible sensor locations
- A detectability and isolability performance specification

**Minimal Sensor Set**

A multiset \( S \), defined on \( P \), is a minimal sensor set if the specification is fulfilled when the sensors in \( S \) are added, but not fulfilled when any proper subset is added.

**Problem Statement**

Find all minimal sensor sets with respect to a required isolability specification and possible sensor locations for any linear differential-algebraic model

A Structural Model

\[ e_1 : \dot{x}_1 = -x_1 + x_2 + x_5 \]
\[ e_2 : \dot{x}_2 = -2x_2 + x_3 + x_4 \]
\[ e_3 : \dot{x}_3 = -3x_3 + x_5 + f_1 + f_2 \]
\[ e_4 : \dot{x}_4 = -4x_4 + x_5 + f_3 \]
\[ e_5 : \dot{x}_5 = -5x_5 + u + f_4 \]
**Detectability**

- Assume that a fault $f$ only violate 1 equation, $e_f$.

**Detectability**

A fault $f$ is structurally detectable if $e_f \in M^+$. 

- Fault $f_1$ not detectable
- Fault $f_2$ detectable

**Define a Partial Order on $b_i$**

**Partial Order on $b_i$**

$b_i \geq b_j$ if element $(i, j)$ is shaded

**Lemma**

Let $e_i$ measure a variable in $b_i$ then all equal and lower ordered blocks are included in the overdetermined part.

**Sensor Placement for Detectability**

**Measure $x_3 \to \{f_1, f_2, f_4\}$**

$e_1 : \dot{x}_1 = -x_1 + x_2 + x_5$
$e_2 : \dot{x}_2 = -2x_2 + x_3 + x_4$
$e_3 : \dot{x}_3 = -3x_3 + x_5 + f_1 + f_2$
$e_4 : \dot{x}_4 = -4x_4 + x_5 + f_3$
$e_5 : \dot{x}_5 = -5x_5 + u + f_4$
$e_6 : y = x_3$

**Minimal Sensor Sets - Detectability**

**Detectability Set**

$D([f_i]) = \text{measurements that give detectability of fault } f_i$

- $D(f_1) = \{x_1, x_2, x_3\}$
- $D(f_2) = \{x_1, x_2, x_3\}$
- $D(f_3) = \{x_1, x_2, x_3\}$
- $D(f_4) = \{x_1, x_2, x_3, x_4, x_5\}$
Sensor set for detectability

S is a sensor set achieving detectability if and only if $S$ has a non-empty intersection for all $D(f_i)$.

A standard minimal hitting-set algorithm can be used to obtain the minimal sensor sets.

$$D(f_1) = \{x_1, x_2, x_3\}$$
$$D(f_2) = \{x_1, x_2, x_3\}$$
$$D(f_3) = \{x_1, x_2, x_4\}$$
$$D(f_4) = \{x_1, x_2, x_3, x_4, x_5\}$$

Sensor placement for maximal isolability

- detectability necessary for isolability
- minimal sensor sets: $\{x_1\}$, $\{x_2\}$, $\{x_3, x_4\}$
- add e.g. measurement $x_1$
- all faults are detectable

Making faults isolable from $f_1$

- Which faults are isolable from $f_1$ with existing sensors? 
  $\Rightarrow$ no faults are isolable from $f_1$
- Applying the detectability algorithm gives detectability sets
  $$D(f_3) = \{x_3, x_4\}$$
  $$D(f_4) = \{x_3, x_4, x_5\}$$
Achieving maximum isolability

- Detectability sets for maximum isolability
  
  \[
  \{x_1, x_2\} : \{x_3, x_4\} \\
  \{x_3, x_4\} \quad \Rightarrow \quad \{x_3\}, \{x_4\}
  \]

- Measurement \(x_1\) was added to achieve detectability

- Maximal isolability is obtained for
  \[
  \{x_1, x_3\}, \{x_1, x_4\}
  \]

- This is not all minimal sensor sets!

How about faults in the new sensors?

"Sloppy" versions of two results

**Lemma**

*Faults in the new sensors are detectable*

This is not surprising, a new sensor equation will always be in the over determined part of the model, that was its objective.

**Lemma**

Let \(F\) be a set of detectable faults in a model \(M\) and \(f_s\) a fault in a new sensor. Then it holds that \(f_s\) is isolable from all faults in \(F\) automatically.

This result were not as evident to me, but it is nice since it makes the algorithm for dealing with faults in the new sensors very simple.

Method summary

- For each detectability and isolability requirement, compute detectability sets
  - Dulmage-Mendelsohn decomposition + identify partial order
- Apply a minimal hitting-set algorithm to all detectability sets to compute all minimal sensor sets

The minimal sensor sets is a characterization of all sensor sets.
Example: An electrical circuit

A small electrical circuit with 5 components that may fail

\[
\begin{align*}
L & \quad R_1 \\
R_2 & \quad C \\
4 & \quad 2 \\
3 & \quad 5
\end{align*}
\]

\[
\begin{align*}
\text{v}_1 = \text{v}_5 & \quad \text{v}_5 = \text{v}_2 + \text{v}_3 \\
\text{i}_1 = \text{i}_2 + \text{i}_5 & \quad \text{i}_1 = \text{i}_3 + \text{i}_4 + \text{i}_5 \\
\text{v}_1 = \text{z} & \quad \text{v}_2 = R_1 \text{i}_2 \\
\text{v}_4 = L \frac{\text{d}}{\text{dt}} \text{i}_4 & \quad \text{i}_5 = C \frac{\text{d}}{\text{dt}} \text{v}_5 \\
\text{v}_3 = \text{v}_4 & \quad \text{v}_3 = R_2 \text{i}_3
\end{align*}
\]

- 10 equations, 2 states, 5 faults, 1 known signal
- Possible measurements: currents and voltages

Examples of results of the analysis

Example run 2

Objective: Achieve full isolability
Possible measurement: voltages and currents

5 minimal solutions

\[
\{i_1, i_3\}, \{i_1, i_4\}, \{i_2, i_3, i_5\}, \{i_2, i_4, i_5\}, \{i_3, i_4, i_5\}
\]

Outline

- Introduction
- Structural models and basic definitions
- Diagnosis system design
- Residual generation
- Diagnosability analysis
- Sensor placement analysis
- Case study and software demonstration
- Analytical vs structural properties
- Concluding remarks
Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics

Example 1: Automotive engine
Analysis of an automotive engine model where only structural information is used
Shows examples on what can be done very early in the design process

Example 2: Three tank system
Analysis of a three-tank system model
Shows examples on what can be done with structural analysis and code generation

Software
http://www.fs.isy.liu.se/Software/FaultDiagnosisToolbox/

Abstract: A mean-value model of a diesel engine with a variable-geometry turbocharger (VGT) and exhaust gas recirculation (EGR) is developed, parameterized, and validated. The intended model applications are system analysis, simulation, and development of model-based control systems. The goal is to construct a model that describes the gas flow dynamics including the dynamics in the manifold, compressor, EGR, and scavenging with focus on the inlet manifold. The EGR model consists of a mass flow model, a pressure model, and a volume model. The VGT model consists of a mass flow model, a pressure model, and a non-minimum phase behaviour, an overshoot, and a sign reversal in the VGT angle.

The model is parameterized using weighted least-squares optimization and it is shown that it is important to tune the submodels and the complete model and not only the submodels or not only the complete model.

The cylinder-out temperature is modelled as

\[ W_{\text{ei}} = \frac{\eta_{\text{vol}} p_{\text{in}} n_{\text{cyl}} V_{\text{d}}}{120 R T_{\text{ei}}} \]  

where \( p_{\text{in}} \) and \( T_{\text{ei}} \) are the pressure and temperature respectively in the intake manifold, \( n_{\text{cyl}} \) is the engine speed, and \( V_{\text{d}} \) is the displaced volume. The volumetric efficiency is in its turn modelled as

\[ \eta_{\text{vol}} = \sqrt{p_{\text{in}}} + c_{\text{vol}} + c_{\text{vol}} + c_{\text{vol}} \]  

The fuel mass flow \( W_{\text{f}} \) into the cylinders is controlled by \( u_{\text{f}} \), which gives the injected mass of fuel in milligrams per cycle and cylinder as

\[ W_{\text{f}} = \frac{10^{-4} n_{\text{cyl}}}{120} u_{\text{f}} \]  

where \( n_{\text{cyl}} \) is the number of cylinders. The mass flow \( W_{\text{out}} \) out from the cylinder is given by the mass balance as

\[ W_{\text{out}} = W_{\text{f}} + W_{\text{sys}} \]  

The oxygen-to-fuel ratio \( \lambda_{\text{o}} \) in the cylinder is defined as

\[ \lambda_{\text{o}} = \frac{W_{\text{c}} X_{\text{h}}}{W_{\text{f}} / (O_{\text{2}}/V_{\text{f}})} \]
Structural modelling

```matlab
model.type = 'VarStruc';

% Unknown variables
% 59 variables, 13 are states, 13 are d terms, 6 are inputs
model.x = { 'dpaf', 'dTaf', 'dpc', 'dTc', 'dpic', ...
% Known variables
% 7 output sensors and 6 input sensors
model.z = { 'yTc', 'ypc', 'yTic', 'ypic', 'yTim', ...
% Faults
% 12 faults (7 variable faults and 5 sensor faults)
model.f = { 'fpaf', 'fomegat', 'fvol', 'fWaf', 'fWc', ...

% Define structure
% Each line represents a model relation and lists all involved variables.
% Total 66 equations for all variables, inputs and sensors
model.rels = { ...
    { 'dTaf' 'Wc' 'Waf' 'Tamb' 'paf' 'Taf1' },...
    { 'dpaf' 'Taf' 'Wc' 'Waf' },...
    { 'dTc' 'Wc' 'Wic' 'Tc1' 'pc' },...
    sm = DiagnosisModel( model );
```

Check model for problems

```matlab
sm.Lint();
```

Model validation finished with 0 errors and 0 warnings.

Plot model structure

```matlab
>> sm.PlotModel();
```

Isolability analysis

```matlab
>> sm.IsolabilityAnalysis();
```
Isolability analysis – Dulmage-Mendelsohn decomp.

```
>> sm.PlotDM('eqclass', true, 'fault', true);
```

![PlotDM('eqclass', true, 'fault', true)](image)

Isolability analysis – integral causality

```
>> sm.IsolabilityAnalysis('causality', 'int');
```

![Isolability matrix for 'Structural Model of A Single Turbo Petrol Engine' (integral causality)](image)

Isolability analysis – derivative causality

```
>> sm.IsolabilityAnalysis('causality', 'der');
```

![Isolability matrix for 'Structural Model of A Single Turbo Petrol Engine' (derivative causality)](image)

Overdetermined set of equations

Degree of redundancy for the model is 7, there are 394,546 MSO sets, instead compute the set of MTES.

```
>> mtes = sm.MTES();
```

In a second on my laptop, finds 159 MTES
- Finds all possible fault signatures (159)
- For each fault signature, we know which constraints are needed to compute a residual

```
>> FSM = sm.FSM( mtes );
```

- We have here 159 candidate residual generators
- Do we really need all of them?
Test selection – all 159 is not needed

>> ts = sm.TestSelection( FSM, 'method', 'amin')
>> sm.IsolabilityAnalysisFSM(FSM(ts,:));

Modelling

model.type = 'Symbolic';
model.x = {'p1','p2','p3','q0','q1','q2','q3','dp1','dp2','dp3'};
model.f = {'fV1','fV2','fV3','fT1','fT2','fT3'};
model.z = {'y1','y2','y3'};
model.rels = {q1==1/Rv1*(p1-p2) + fV1,...
q2==1/Rv2*(p2-p3) + fV2, ... 
q3==1/Rv3*p3 + fV3,...
dp1==1/CT1*(q0-q1) + fT1,...
dp2==1/CT2*(q1-q2) + fT2, ... 
dp3==1/CT3*(q2-q3) + fT3, ... 
y1==p1, y2==q2, y3==q0,...
DiffConstraint('dp1','p1'),...
DiffConstraint('dp2','p2'),...
DiffConstraint('dp3','p3')};

sm = DiagnosisModel( model );

Example with symbolic equations and code generation

Example with symbolic equations and code generation

Structure is automatically computed

>> sm.PlotModel();
Methods for structural models directly available

```matlab
>> sm.IsolabilityAnalysis();
```

![Isolability matrix for 'Three tank system']

```matlab
>> sm.PlotDM('eqclass',true,'fault',true);
```

Variables

\[
\begin{align*}
p_3 & \\
q_3 & \\
dp_3 & \\
q_0 & \\
dp_1 & \\
dp_2 & \\
p_1 & \\
p_2 & \\
q_1 & \\
q_2 & \\
\end{align*}
\]

Equations

\[
\begin{align*}
e_2 & \\
e_3 & \\
e_4 & \\
e_5 & \\
e_6 & \\
e_7 & \\
e_8 & \\
e_9 & \\
e_{10} & \\
e_{11} & \\
e_{12} & \\
\end{align*}
\]

```

Code generation: Sequential residual generator

MSO \( M = \{e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\} \), with \( e_2 \) as residual equation.

To generate code for the sequential residual generator, 1) compute a matching to compute unknown variables, 2) use residual equation for detection.

```matlab
Gamma = sm.Matching([3, 4, 5, 6, 7, 8, 9, 10, 11, 12]);
sm.SeqResGen( Gamma, 2,'ResGen');
```

Generated code (slightly cropped)

```matlab
function [r, state] = ResGen(z,state,params,Ts)
% Known variables
y1 = z(1);
y2 = z(2);
y3 = z(3);

% Residual generator body
q2 = y2; % e8
q0 = y3; % e9
q3 = p3/Rv3; % e1
dp3 = (q2-q3)/CT3; % e2
p3 = ApproxInt(dp3,state.p3,Ts); % e3
p1 = y1; % e7
dp1 = ApproxDiff(p1,state.p1,Ts); % e10
q1 = q0-CT1*dp1; % e4
dp2 = (q1-q2)/CT2; % e5
p2 = ApproxInt(dp2,state.p2,Ts); % e11
r = q2-(p2-p3)/Rv2; % e2 -- residual equation
end
```
Analytical vs structural properties

- Structural analysis, applicable to a large class of models without details of parameter values etc.
- One price is that only best-case results are obtained
- Relations between analytical and structural results and properties an interesting, but challenging area
- Have not seen much research in this area

Book with a solid theoretical foundation in structural analysis


Basic assumptions for structural analysis

- Structural rank $\text{sprank}(A)$ is equal to the size of a maximum matching of the corresponding bipartite graph.
- $\text{rank}(A) \leq \text{sprank}(A)$
- Structural analysis can give wrong results when a matrix or a sub-matrix is rank deficient, i.e., $\text{rank}(A) \leq \text{sprank}(A)$.
- Example

\[
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\
x_2
\end{bmatrix}
\]

Structural matrix just-determined

\[
A_{\text{str}} = \begin{bmatrix} X & X \\ X & X \end{bmatrix}
\]

Redundancy relation $y_1 - y_2 = 0$. ⇒ no redundancy

Wrong structural results because $A$ is rank deficient:

$\text{rank}(A) = 1 < 2 = \text{sprank}(A)$
Exercise

a) Compute the fault isolability of the model below.

b) Eliminate $T$ in the model by using equation $e_4$. The resulting model with 6 equations is of course equivalent with the original model. Compute the fault isolability for this model and compare it with the isolability obtained in (a).

\[ e_1 : V = i(R + f_R) + L \frac{di}{dt} + K_a i \omega \]
\[ e_2 : T_m = K_a i^2 \]
\[ e_3 : J \frac{d\omega}{dt} = T - (b + f_b) \omega \]
\[ e_4 : T = T_m - T_l \]
\[ e_5 : y_i = i + f_i \]
\[ e_6 : y_\omega = \omega + f_\omega \]
\[ e_7 : y_T = T + f_T \]
Some take home messages

**Structural models**
- Coarse models that do not need parameter values etc.
- Can be obtained early in the design process
- Graph theory; analysis of large models with no numerical issues
- Best-case results

**Analysis**
- Find submodels for detector design
- Not just $y - \hat{y}$, many more possibilities
- Diagnosability, Sensor placement, ...

**Residual generation**
- Structural analysis supports code generation for residual generators
- Sequential residual generators based on matchings
- Observer based residual generators

Thanks for your attention!

---

Structural methods for analysis and design of large-scale diagnosis systems

Erik Frisk and Mattias Krysander
{frisk,matkr}@isy.liu.se

Dept. Electrical Engineering
Vehicular Systems
Linköping University
Sweden

September 1, 2015

Some publications on structural analysis from our group

**Overdetermined equations, MSO, MTES**

Mattias Krysander, Jan Åslund, and Mattias Nyberg.
An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis.

Mattias Krysander, Jan Åslund, and Erik Frisk.
A structural algorithm for finding testable sub-models and multiple fault isolability analysis.
21st International Workshop on Principles of Diagnosis (DX-10), Portland, Oregon, USA, 2010.
Some publications on structural analysis from our group

Sensor placement and diagnosability analysis

Mattias Krysander and Erik Frisk.
Sensor placement for fault diagnosis.

Erik Frisk, Aníbal Bregon, Jan Åslund, Mattias Krysander, Belarmino Pulido, and Gautam Biswas.
Diagnosability analysis considering causal interpretations for differential constraints.

Residual generation supported by structural analysis

Carl Svärd and Mattias Nyberg.
Residual generators for fault diagnosis using computation sequences with mixed causality applied to automotive systems.

Carl Svärd, Mattias Nyberg, and Erik Frisk.
Realizability constrained selection of residual generators for fault diagnosis with an automotive engine application.

Publications on Structural Analysis from our group

Application studies

Dilek Dusteğör, Erik Frisk, Vincent Coquempot, Mattias Krysander, and Marcel Staroswiecki.
Structural analysis of fault isolability in the DAMADICS benchmark.

Carl Svärd and Mattias Nyberg.
Automated design of an FDI-system for the wind turbine benchmark.

Carl Svärd, Mattias Nyberg, Erik Frisk, and Mattias Krysander.
Automotive engine FDI by application of an automated model-based and data-driven design methodology.