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

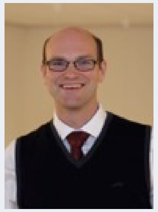
Erik Frisk and Mattias Krysander
{frisk,matkr}@isy.liu.se

Dept. Electrical Engineering
Vehicular Systems
Linköping University
Sweden

September 1, 2015

LINKÖPING UNIVERSITY

Who are we?



Erik Frisk
Associate professor
frisk@isy.liu.se

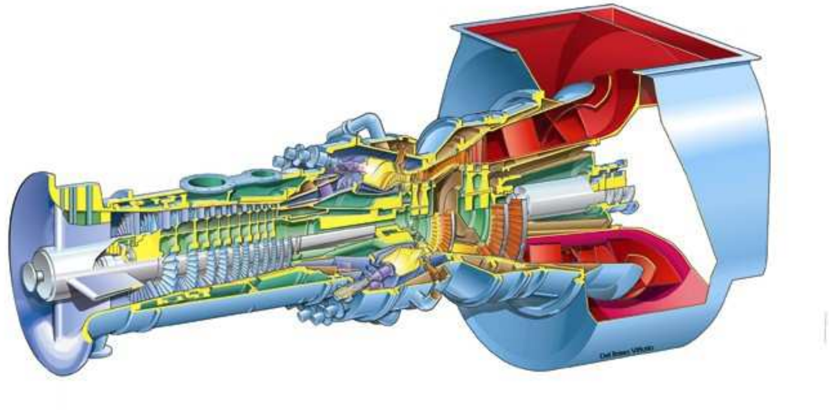
Mattias Krysander
Associate professor
matkr@isy.liu.se

Department of Electrical Engineering
Linköping University
Sweden

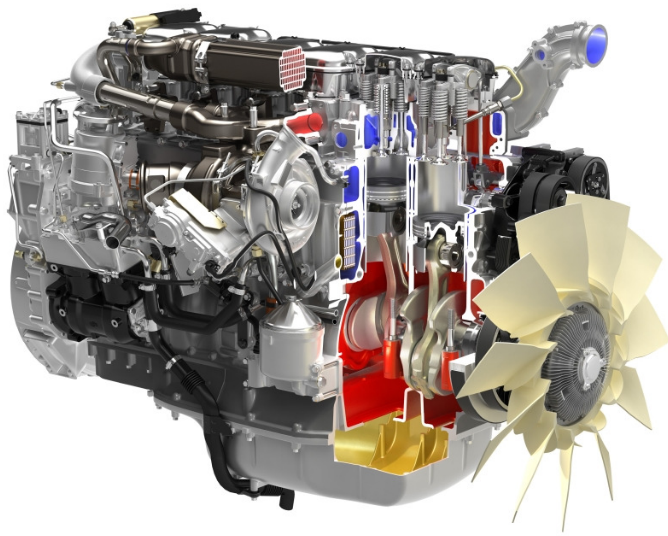
— Introduction —

Outline

Supervision of an industrial gas turbine



Supervision of an automotive engine



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

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

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

1. Formally introduce structural models and fundamental diagnosis definitions
2. 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
3. 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.

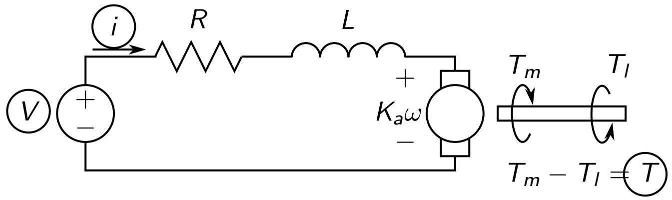
Basic principle - systematic utilization of redundancy

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

$$
\begin{aligned}
x &= g(u) \\
y_1 &= x \\
y_2 &= x \\
y_3 &= x
\end{aligned}
\qquad\qquad
\begin{aligned}
r_1 &= y_1 - g(u) \\
r_2 &= y_2 - g(u) \\
r_3 &= y_2 - y_1 \\
r_4 &= y_3 - g(u) \\
r_5 &= y_3 - y_1 \\
r_6 &= y_3 - y_2
\end{aligned}
$$

- 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



$e_1 : V = iR(1 + f_R) + L\frac{di}{dt} + K_a i\omega$ $\quad e_4 : T = T_m - T_l$ $\quad e_7 : y_i = i + f_i$

$e_2 : T_m = K_a i^2$ $\quad e_5 : \frac{d\theta}{dt} = \omega$ $\quad e_8 : y_\omega = \omega + f_\omega$

$e_3 : J\frac{d\omega}{dt} = T - b\omega$ $\quad e_6 : \frac{d\omega}{dt} = \alpha$ $\quad e_9 : y_T = T + f_T$

Model summary (9 equations)

Known variables(4): V, y_i, y_ω, y_T
Unknown variables(7): i, θ, ω, α, T, T_m, T_l, (i, ω, θ dynamic)
Fault variables(4): f_R, f_i, f_ω, f_T

Structural model

Structural model

A structural model only models that variables are related!

Example relating variables: V, i, ω

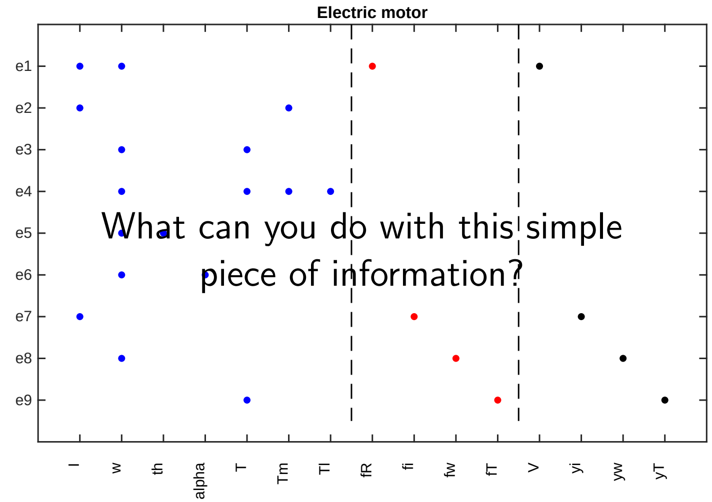
$$e_1 : V = iR(1 + f_R) + L\frac{di}{dt} + K_a i\,\omega$$

	Unknown variables														
	i	θ	ω	α	T	T_m	T_l	f_R	f_i	f_ω	f_T	V	y_i	y_ω	y_T
e_1	X		X					X				X			

- 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!

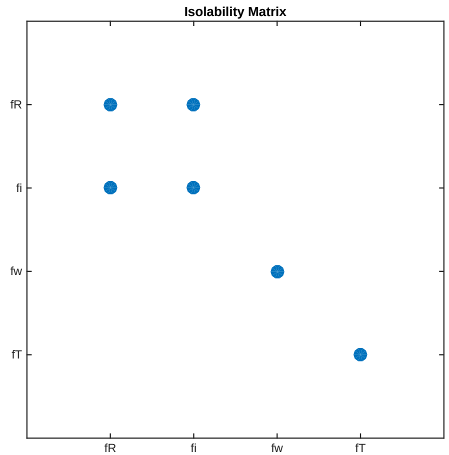
Main drawback: Only best case results!

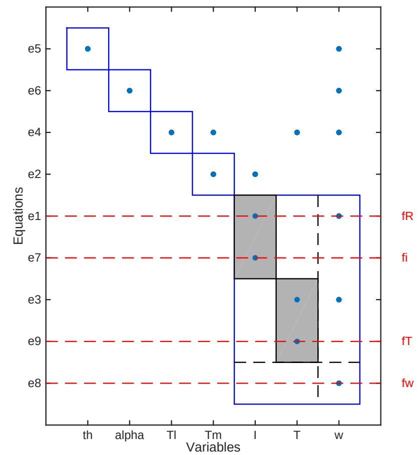
Structural model of the electric motor



- **Known variables**(4): V, y_i, y_ω, y_T
- **Unknown variables**(7): i, θ, ω, α, T, T_m, T_l, (i, ω, θ dynamic)
- **Fault variables**(4): f_R, f_i, f_ω, f_T

Structural isolability analysis of model





Nontrivial result

f_R and f_i can not be isolated from each other, unique isolation of f_ω 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 achieves full isolability are

$$\mathcal{S}_1 = \{i\}$$
$$\mathcal{S}_2 = \{T_m\}$$
$$\mathcal{S}_3 = \{T_l\}$$

Let us add $\mathcal{S}_1$, 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?

$$e_1: V = iR(1 + f_R) + L\frac{di}{dt} + K_a i\omega \qquad e_4: T = T_m - T_l \qquad e_7: y_i = i + f_i$$

$$e_2: T_m = K_a i^2 \qquad e_5: \frac{d\theta}{dt} = \omega \qquad e_8: y_\omega = \omega + f_\omega$$

$$e_3: J\frac{d\omega}{dt} = T - b\omega \qquad e_6: \frac{d\omega}{dt} = \alpha \qquad e_9: y_T = T + f_T$$

$$e_{10}: y_{i,2} = i$$

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 (ω and T)

$$r = J\dot{y}_\omega + by_\omega - y_T$$

Structural redundancy

Determine redundancy by counting equations and unknown variables!

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

$$\mathcal{M}_1 = \{y_i = i, y_{i,2} = i\} \qquad \Rightarrow r_1 = y_i - y_{i,2}$$

$$\mathcal{M}_2 = \{y_\omega = \omega, y_T = T, J\dot{\omega} = T - b\omega\} \qquad \Rightarrow r_2 = y_T - J\dot{y}_\omega - b\omega$$

$$\mathcal{M}_3 = \{V = L\frac{d}{dt}i + i\,R + K_a i\omega, \quad y_\omega = \omega, y_i = i\} \qquad \Rightarrow r_3 = V - L\dot{y}_i + y_i\,R + K_a y_i y_\omega$$

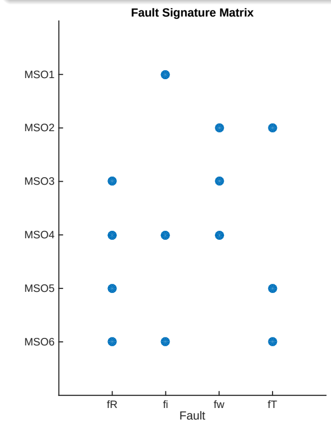
$$\mathcal{M}_4 = \dots$$

$$\mathcal{M}_5 = \dots$$

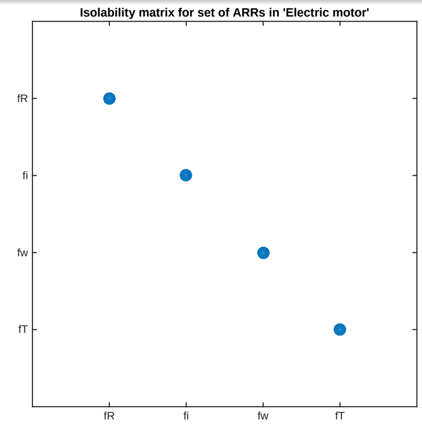
$$\mathcal{M}_6 = \dots$$

Fault signature matrix and isolability for MSOs

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



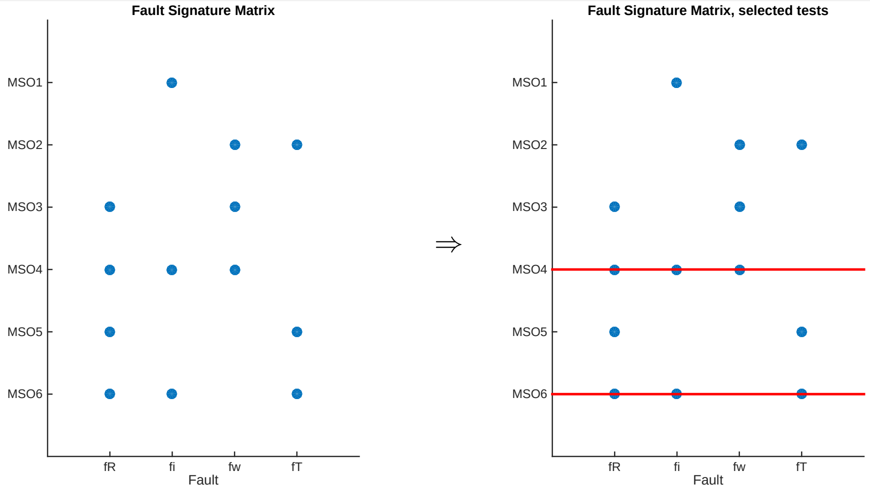
$\Rightarrow$



If I could design 6 residuals based on the MSOs $\Rightarrow$ full isolability

Test selection

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



Code generation supported by structural analysis

Q: Can we automatically generate code for residual generator?

For example, MSO $\mathcal{M}_2$

$$\{y_\omega = \omega, y_T = T, J\dot{\omega} = T - b\omega\}$$

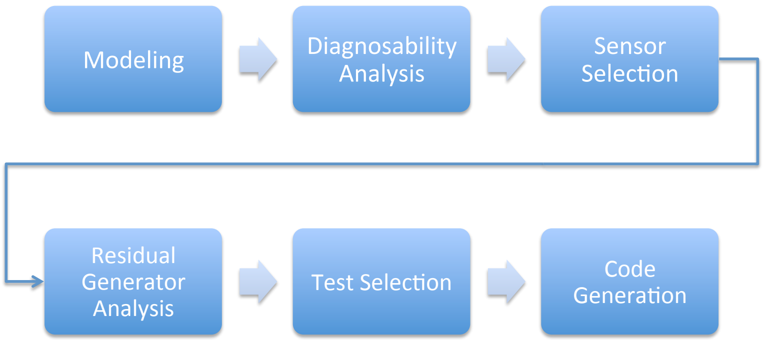
has redundancy and it is possible to generate code for residual generator, equivalent to

$$r_2 = J\dot{y}_\omega + by_\omega - y_T$$

Automatic generation of code

```
% Initialize state variables
w = state.w;

% Residual generator body
T = yT; % e9
w = yw; % e8
dw = ApproxDiff(w,state.w,Ts);  % e11
r2 = J*dw+bw-T; % e3
```


Design process aided by structural analysis



All these topics will be covered in the tutorial

Presentation biased to our own work

Some history

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

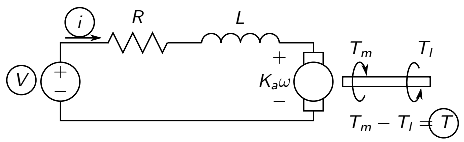
60's-70's Structure analysis and decomposition of large systems, e.g., C.T. Lin, "*Structural controllability*" (AC-1974)

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.

— *Basic definitions* —

Outline

A structural model - the nominal model



$e_1 : V = iR + L\dfrac{di}{dt} + K_a i\omega$

$e_2 : T_m = K_a i^2$

$e_3 : J\dfrac{d\omega}{dt} = T - b\omega$

$e_4 : T = T_m - T_l$

$e_5 : y_i = i$

$e_6 : y_\omega = \omega$

$e_7 : y_T = T$

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.

A structural model - the nominal model

$e_1 : V = iR + L\dfrac{di}{dt} + K_a i\omega$

$e_2 : T_m = K_a i^2$

$e_3 : J\dfrac{d\omega}{dt} = T - b\omega$

$e_4 : T = T_m - T_l$

$e_5 : y_i = i$

$e_6 : y_\omega = \omega$

$e_7 : y_T = T$

Biadjacency matrix:

	i	ω	T	T_m	T_l
e_1	X	X			
e_2	X			X	
e_3		X	X		
e_4			X	X	X
e_5	X				
e_6		X			
e_7			X		

A structural model with fault information

Fault influence can be included in the model

- by fault signals
- by equation assumptions/supports

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

$e_2 : T_m = K_a i^2$

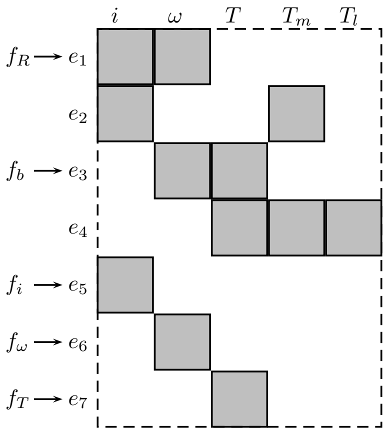
$e_3 : J\frac{d\omega}{dt} = T - (b + f_b)\omega$

$e_4 : T = T_m - T_l$

$e_5 : y_i = i + f_i$

$e_6 : y_\omega = \omega + f_\omega$

$e_7 : y_T = T + f_T$



Structural representation of dynamic systems

Structural representation of dynamic systems can be done in a number of ways.

1. Consider x and $\dot{x}$ to be structurally the same variable.
2. 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

$$x' = \frac{dx}{dt}$$

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

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

$e_2 : T_m = K_a i^2$

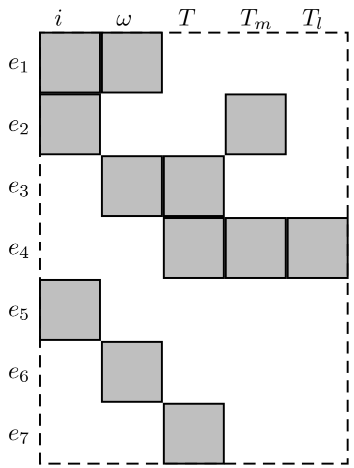
$e_3 : J\frac{d\omega}{dt} = T - b\omega$

$e_4 : T = T_m - T_l$

$e_5 : y_i = i$

$e_6 : y_\omega = \omega$

$e_7 : y_T = T$



- Compact description
- Good for analysis

Dynamics - distinguish derivatives

$e_1 : V = iR + Li' + K_a i\omega$

$e_2 : T_m = K_a i^2$

$e_3 : J\omega' = T - b\omega$

$e_4 : T = T_m - T_l$

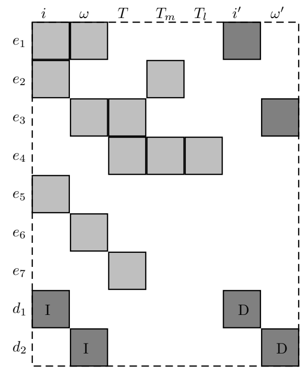
$e_5 : y_i = i$

$e_6 : y_\omega = \omega$

$e_7 : y_T = T$

$d_1 : i' = \frac{di}{dt}$

$d_2 : \omega' = \frac{d\omega}{dt}$



- 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

1. Can a residual generator be derived from M ? or equivalently can the consistency of M be checked?
2. Which faults are expected to influence the residual?

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

Testable equation set?

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

$$e_3 : T = J\frac{d\omega}{dt} + b\omega$$
$$e_5 : i = y_i$$
$$e_6 : \omega = y_\omega$$
$$e_1 : V - iR - L\frac{di}{dt} - K_a i\omega = 0$$

	T	i	ω
e_3	X		X
e_5		X	
e_6			X
e_1		X	X

- Yes! The values of ω, 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.

Fault sensitivity of the residual?

- Model with fault:

$$e_3 : T = J\frac{d\omega}{dt} + (b + f_b)\omega$$
$$e_5 : i = y_i - f_i$$
$$e_6 : \omega = y_\omega - f_\omega$$
$$e_1 : V - i(R + f_R) - L\frac{di}{dt} - K_a i\omega = 0$$

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

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

$$r = V - y_i R + L\frac{dy_i}{dt} - K_a y_i y_\omega =$$
$$= y_i f_R + f_i(K_a f_\omega - R - y_w - f_R) - L\frac{df_i}{dt} - K_a y_i f_\omega$$

- Sensitive to all faults except f_b.
- Not surprising since e_3 was not used in the derivation of the residual!

Strucutural analysis provides the same information

- Model with fault:

$$e_3 : T = J\frac{d\omega}{dt} + (b + f_b)\omega$$
$$e_5 : i = y_i - f_i$$
$$e_6 : \omega = y_\omega - f_\omega$$
$$e_1 : V - i(R + f_R) - L\frac{di}{dt} - K_a i\omega = 0$$

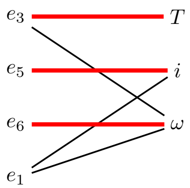
	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

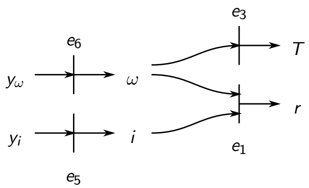
- Structural analysis provides the following useful diagnosis information:
  - residual from $\{e_1, e_5, e_6\}$
  - sensitive to $\{f_i, f_\omega, f_R\}$
- Let's formalize the structural reasoning!

Matching

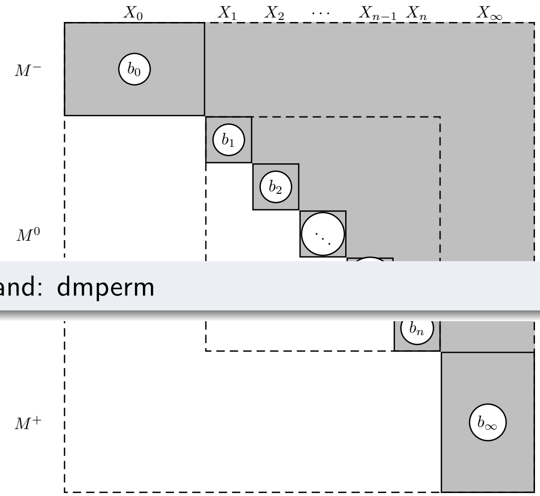
- A *matching* in a bipartite graph is a pairing of nodes in the two sets.
- Formally: set of edges with no common nodes.
- A matching with maximum cardinality is a *maximal matching*.
- Diagnosis related interpretation: which variable is computed from which equation

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R





Dulmage-Mendelsohn decomposition



Matlab command: dmperm

- 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.

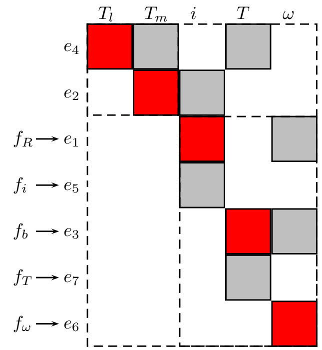
Dulmage-Mendelsohn Decomposition

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

Detectable faults

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

$M^+ = \{e_1, e_5, e_6\}$
$X^+ = \{i, \omega\}$
Faults in M^+: $\{f_i, f_\omega, f_R\}$



$M^+ = \{e_1, e_3, e_5, e_6, e_7\}$
$X^+ = \{i, T, \omega\}$
Faults in M^+: $\{f_R, f_i, f_b, f_T, f_\omega, \}$

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

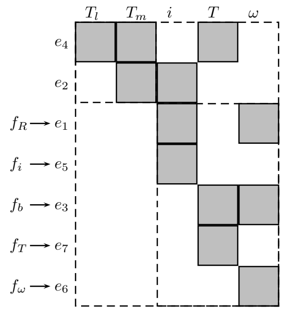
Degree of redundancy

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

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

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

$M^+ = \{e_1, e_5, e_6\}$
$X^+ = \{i, \omega\}$
$\varphi(M) = 3 - 2 = 1$



$M^+ = \{e_1, e_3, e_5, e_6, e_7\}$
$X^+ = \{i, T, \omega\}$
$\varphi(M) = 5 - 3 = 2$

Basic definitions - overdetermined equation sets

Structurally Overdetermined (SO)

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

Minimally Structurally Overdetermined (MSO)

An SO set M is an 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 $E \subset M$

Examples - electrical motor

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

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

- $M = \{e_1, e_3, e_5, e_6\}$ 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*.
- $M^+ = \{e_1, e_5, e_6\}$ 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!

Example - sensor redundancy

$e_1 : y_1 = x$
$e_2 : y_2 = x$
$e_3 : y_3 = x$

$\{e_1, e_2\} : r_1 = y_1 - y_2$
$\{e_1, e_3\} : r_2 = y_1 - y_3$
$\{e_2, e_3\} : r_3 = y_2 - y_3$
$\{e_1, e_2, e_3\} : r_4 = r_1^2 + r_2^2$

- $\{e_1, e_2, e_3\}$ is Structurally Overdetermined (SO) but *not* MSO since
- $\{e_1, e_2\}$, $\{e_1, e_3\}$, $\{e_2, e_3\}$ all are MSO:s
- All above equation sets are PSO since degree of redundancy decreases if an element is removed.

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:

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*

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:

$$
\begin{aligned}
e_1 &: \quad \dot{x}_1 = -x_1 + u + f_1\\
e_2 &: \quad \dot{x}_2 = x_1 - 2x_2 + x_3 + f_2\\
e_3 &: \quad \dot{x}_3 = x_2 - 3x_3\\
e_4 &: \quad y_1 = x_2 + f_3\\
e_5 &: \quad y_2 = x_2 + f_4\\
e_6 &: \quad y_3 = x_3 + f_5
\end{aligned}
$$

	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

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

MSO sets

There are 8 MSO sets in the model

	Equations	Faults
MSO_1	$\{e_3, e_5, e_6\}$	$\{f_4, f_5\}$
MSO_2	$\{e_3, e_4, e_6\}$	$\{f_3, f_5\}$
MSO_3	$\{e_4, e_5\}$	$\{f_3, f_4\}$
MSO_4	$\{e_1, e_2, e_3, e_6\}$	$\{f_1, f_2, f_5\}$
MSO_5	$\{e_1, e_2, e_3, e_5\}$	$\{f_1, f_2, f_4\}$
MSO_6	$\{e_1, e_2, e_3, e_4\}$	$\{f_1, f_2, f_3\}$
MSO_7	$\{e_1, e_2, e_5, e_6\}$	$\{f_1, f_2, f_4, f_5\}$
MSO_8	$\{e_1, e_2, e_4, e_6\}$	$\{f_1, f_2, f_3, f_5\}$

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!

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

Tests sensitive to few faults give more precise isolation.

	Equations	Faults
MSO_1	$\{e_3, e_5, e_6\}$	$\{f_4, f_5\}$
MSO_2	$\{e_3, e_4, e_6\}$	$\{f_3, f_5\}$
MSO_3	$\{e_4, e_5\}$	$\{f_3, f_4\}$
MSO_4	$\{e_1, e_2, e_3, e_6\}$	$\{f_1, f_2, f_5\}$
MSO_5	$\{e_1, e_2, e_3, e_5\}$	$\{f_1, f_2, f_4\}$
MSO_6	$\{e_1, e_2, e_3, e_4\}$	$\{f_1, f_2, f_3\}$
MSO_7	$\{e_1, e_2, e_5, e_6\}$	$\{f_1, f_2, f_4, f_5\}$
MSO_8	$\{e_1, e_2, e_4, e_6\}$	$\{f_1, f_2, f_3, f_5\}$

$$\text{Faults}(MSO_1), \text{Faults}(MSO_4), \text{Faults}(MSO_5) \subset \text{Faults}(MSO_7)$$

$$\text{Faults}(MSO_2), \text{Faults}(MSO_4), \text{Faults}(MSO_6) \subset \text{Faults}(MSO_8)$$

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:

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

Adding equation e_3 does not change the fault sensitivity:

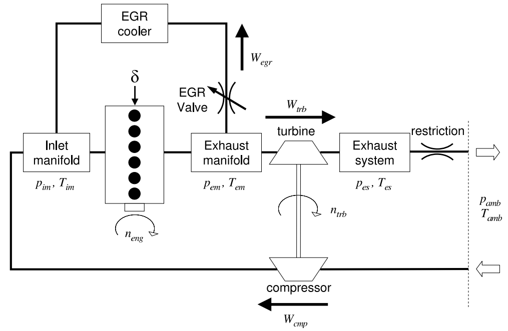
$$Faults(\{e_1, e_2, e_3, e_5, e_6\}) = \{f_1, f_2, f_4, f_5\}$$

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

	Equations	Faults
MSO_1	$\{e_3, e_5, e_6\}$	$\{f_4, f_5\}$
MSO_2	$\{e_3, e_4, e_6\}$	$\{f_3, f_5\}$
MSO_3	$\{e_4, e_5\}$	$\{f_3, f_4\}$
MSO_4	$\{e_1, e_2, e_3, e_6\}$	$\{f_1, f_2, f_5\}$
MSO_5	$\{e_1, e_2, e_3, e_5\}$	$\{f_1, f_2, f_4\}$
MSO_6	$\{e_1, e_2, e_3, e_4\}$	$\{f_1, f_2, f_3\}$
MSO_7	$\{e_1, e_2, e_5, e_6\}$	$\{f_1, f_2, f_4, f_5\}$
MSO_8	$\{e_1, e_2, e_4, e_6\}$	$\{f_1, f_2, f_3, f_5\}$

What distinguish the first 6 MSO sets?

Questions

	Equations	Faults
MSO_1	$\{e_3, e_5, e_6\}$	$\{f_4, f_5\}$
MSO_2	$\{e_3, e_4, e_6\}$	$\{f_3, f_5\}$
MSO_3	$\{e_4, e_5\}$	$\{f_3, f_4\}$
MSO_4	$\{e_1, e_2, e_3, e_6\}$	$\{f_1, f_2, f_5\}$
MSO_5	$\{e_1, e_2, e_3, e_5\}$	$\{f_1, f_2, f_4\}$
MSO_6	$\{e_1, e_2, e_3, e_4\}$	$\{f_1, f_2, f_3\}$
MSO_7	$\{e_1, e_2, e_5, e_6\}$	$\{f_1, f_2, f_4, f_5\}$
MSO_8	$\{e_1, e_2, e_4, e_6\}$	$\{f_1, f_2, f_3, f_5\}$

Is it always MSO sets we are looking for?

Questions

How do we characterize the PSO set $MSO_7 \cup \{e_3\}$, which has the properties

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

Fundamental questions

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

Answers

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

Definition (Test Support)

Given a model $\mathcal{M}$ and a set of faults $\mathcal{F}$, a non-empty subset of faults $\zeta \subseteq \mathcal{F}$ is a test support if there exists a PSO set $M \subseteq \mathcal{M}$ 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 $M' \supsetneq M$ where M' is a PSO set it holds that $F(M') \supsetneq F(M)$.

MSO_7 is not a TES since

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

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

	Equations	Faults
MSO_1	$\{e_3, e_5, e_6\}$	$\{f_4, f_5\}$
MSO_2	$\{e_3, e_4, e_6\}$	$\{f_3, f_5\}$
MSO_3	$\{e_4, e_5\}$	$\{f_3, f_4\}$
MSO_4	$\{e_1, e_2, e_3, e_6\}$	$\{f_1, f_2, f_5\}$
MSO_5	$\{e_1, e_2, e_3, e_5\}$	$\{f_1, f_2, f_4\}$
MSO_6	$\{e_1, e_2, e_3, e_4\}$	$\{f_1, f_2, f_3\}$
MSO_7	$\{e_1, e_2, e_5, e_6\}$	$\{f_1, f_2, f_4, f_5\}$
MSO_8	$\{e_1, e_2, e_4, e_6\}$	$\{f_1, f_2, f_3, f_5\}$

- 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 $\{e_1, e_2, e_3, e_5, e_6\}$, $\{e_1, e_2, e_3, e_4, e_6\}$

Summary

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

TS/TES

- $\zeta \subseteq \mathcal{F}$ is a TS $\Leftrightarrow$ there is a residual sensitive to the faults in ζ
- The TES corresponding to ζ can easliy be computed.

MTES are

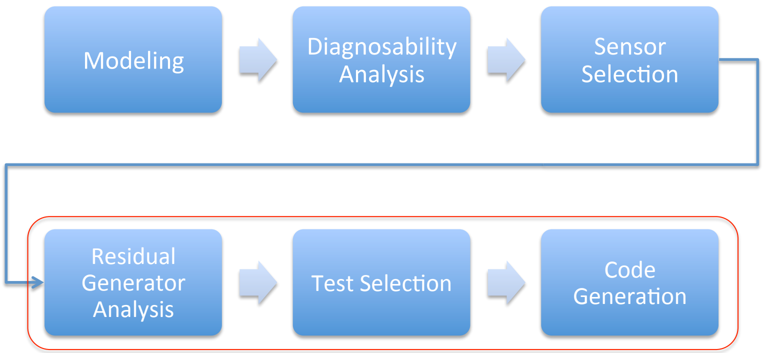
- typically MSO sets.
- fewer than MSO sets.
- sensitive to minimal sets of faults.
- sufficient and necessary for maximum multiple fault isolability

$\Rightarrow$ *candidates for deriving residuals*

— *Diagnosis Systems Design* —

Outline

Design system design supported by structural methods



Diagnosis system design

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

Designing diagnosis system utilizing structural analysis

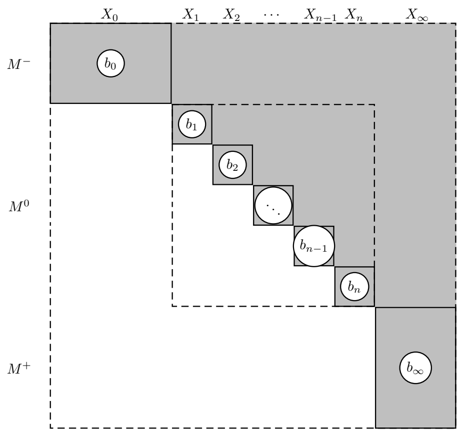
1. Find (all) testable models (MSO/MTES/...)
2. Select a subset of testable models with required fault isolability
3. From each selected testable model generate code for the corresponding residual.

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.

	x_1	x_2	x_3	x_4
~~(1)~~	~~X~~			~~X~~
(2)	X	X		
(3)	X	X		X
(4)			X	
(5)				X
(6)			X	X

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

	x_1	x_2	x_3	x_4
(1)	X			X
(2)	X	X		
(3)	X	X		X
(4)			X	
~~(5)~~			~~X~~	~~X~~
(6)				X
(7)				X

- 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) . . .

Basic algorithm

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

```
function 𝓜_MSO = FindMSO(M)
  if φ(M)=1
    𝓜_MSO := {M}
  else
    𝓜_MSO := ∅
    for each e ∈ M
      M' = (M \ {e})^+
      𝓜_MSO := 𝓜_MSO ∪ FindMSO(M')
    end
  end
```


The same MSO set is found several times

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

	x_1	x_2	x_3	x_4
~~(1)~~	~~X~~			~~X~~
(2)	X	X		
(3)	X	X		X
~~(4)~~			~~X~~	
(5)			X	X
(6)				X
(7)				X

- 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.

	x_1	x_2	x_3	x_4
(1)	X			X
(2)	X	X		
~~(3)~~	~~X~~	~~X~~		~~X~~
(4)			X	
(5)			X	X
(6)				X
(7)				X

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 overdetermined part

	x_1	x_2	x_3	x_4
(1)	X			X
(2)	X	X		
(3)	X	X		X
(4)			X	
(5)			X	X
(6)				X
(7)				X

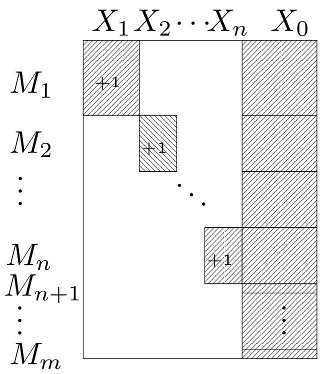
$M_1 = \{(1)(2)(3)\}$ $X_1 = \{x_1, x_2\}$
$M_2 = \{(4)(5)\}$ $X_2 = \{x_3\}$
$M_3 = \{(6)\}$ $X_3 = \varnothing$
$M_4 = \{(7)\}$ $X_4 = \varnothing$
$X_0 = \{x_4\}$

- $|M_i| = |X_i| + 1$
- All MSO sets can be written as a union of equivalence classes, e.g.

$$\{(6)(7)\} = M_3 \cup M_4$$
$$\{(4)(5)(6)\} = M_2 \cup M_3$$

Equivalence classes

Any PSO set can be written on the canonical form



This form will be useful for

1. improving the basic algorithm (now)
2. performing diagnosability analysis (later)

Can be obtained easily with attractive complexity properties

Lumping

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

Original structure:

	x_1	x_2	x_3	x_4
(1)	X			X
(2)	X	X		
(3)	X	X		X
(4)			X	
(5)			X	X
(6)				X
(7)				X

Lumped structure:

	x_4
$M_1 = \{(1)(2)(3)\}$	X
$M_2 = \{(4)(5)\}$	X
$M_3 = \{(6)\}$	X
$M_4 = \{(7)\}$	X

- 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.

Improved algorithm

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

Lets consider this example again

$$
\begin{aligned}
e_1 &: \quad \dot{x}_1 = -x_1 + u + f_1 \\
e_2 &: \quad \dot{x}_2 = x_1 - 2x_2 + x_3 + f_2 \\
e_3 &: \quad \dot{x}_3 = x_2 - 3x_3 \\
e_4 &: \quad y_1 = x_2 + f_3 \\
e_5 &: \quad y_2 = x_2 + f_4 \\
e_6 &: \quad y_3 = x_3 + f_5
\end{aligned}
$$

	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

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

MSO algorithm: We start with the complete model

$\{e_1, e_2, e_3, e_4, e_5, e_6\}$

	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

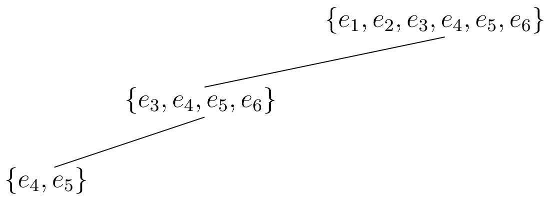
MSO algorithm: Remove e_1 and compute $(M \setminus \{e_1\})^+$

$\{e_1, e_2, e_3, e_4, e_5, e_6\}$

$\{e_3, e_4, e_5, e_6\}$

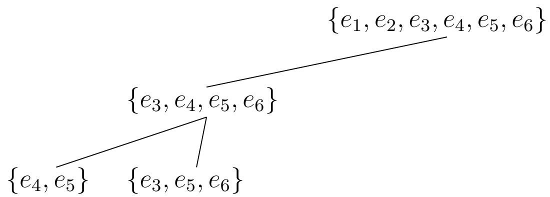
	x_1	x_2	x_3
~~e_1~~	~~X~~		
e_2	X	X	X
e_3		X	X
e_4		X	
e_5		X	
e_6			X

MSO algorithm: Remove e_3



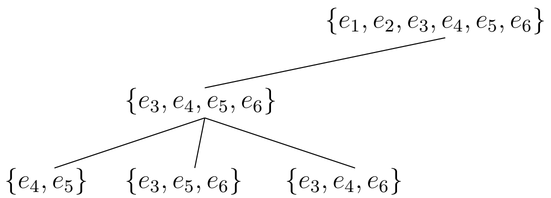
	x_1	x_2	x_3
~~e_1~~	~~X~~		
e_2	X	X	X
~~e_3~~		~~X~~	~~X~~
e_4		X	
e_5		X	
e_6			X

MSO algorithm: Go back and remove e_4



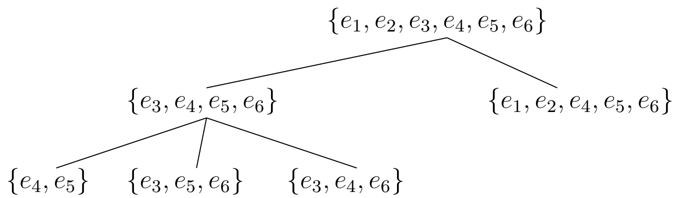
	x_1	x_2	x_3
~~e_1~~	~~X~~		
e_2	X	X	X
e_3		X	X
~~e_4~~		~~X~~	
e_5		X	
e_6			X

MSO algorithm: Go back and remove e_5



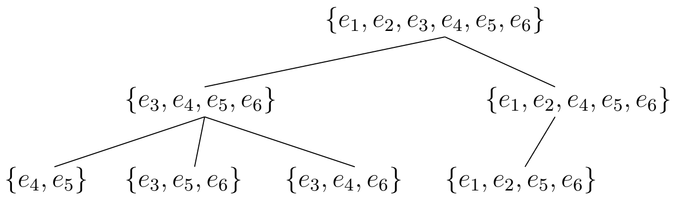
	x_1	x_2	x_3
~~e_1~~	~~X~~		
e_2	X	X	X
e_3		X	X
e_4		X	
~~e_5~~		~~X~~	
e_6			X

MSO algorithm: Go back 2 steps and remove e_3



	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_4



	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

Summary - MSO algorithm

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

An Efficient Algorithm for Finding Minimal Over-constrained Sub-systems for Model-based Diagnosis, Mattias Krysander, Jan Åslund, and Mattias Nyberg. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 38(1), 2008.

MTES algorithm

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

A Structural Algorithm for Finding Testable Sub-models and Multiple Fault Isolability Analysis., Mattias Krysander, Jan Åslund, and Erik Frisk (2010). 21st International Workshop on Principles of Diagnosis (DX-10). Portland, Oregon, USA.

It is a slight modification of the MSO algorithm.

Basic idea

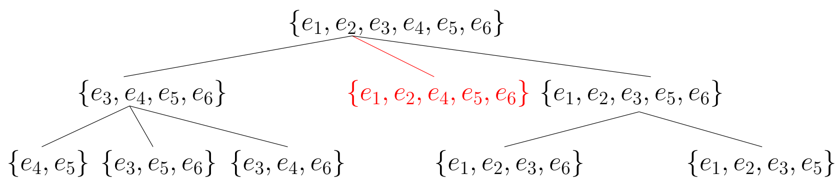
There's no point removing equations that doesn't contain faults, since we are interested in fault sensitivity.

Modification

Stop doing that!

MTES algorithm

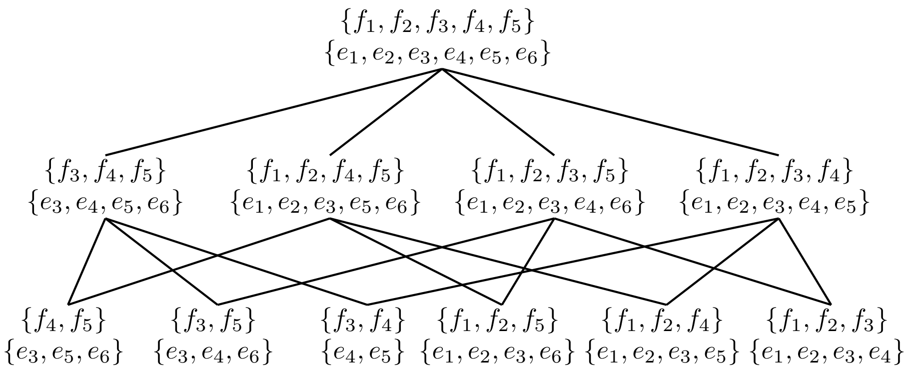
In the example e_3 is the only equation without fault.
We will not remove e_3
We remove e_4 instead.



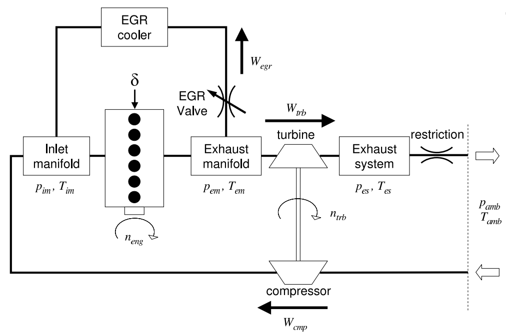
The nodes are TES:s and the leaves are MTES:s.

All TSs and TESs for the model

The algorithm traverses all TESs



Scania truck engine example



Original model:

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

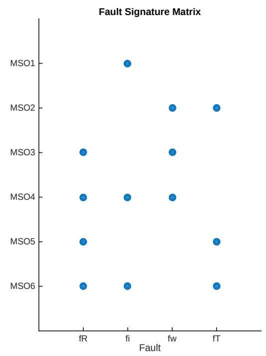
- Reduces the resulting number of testable sets:
  - 1436 MSO sets cmp. to 32 MTESs which all are MSOs.
  - Only 6 needed for full single fault isolation.
- Reduces the computational burden:
  - 1774 PSO sets $\sim$ runtime MSO-alg. (2.5 s)
  - 61 TESs $\sim$ runtime MTES-alg. (0.42 s)
  - Few number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

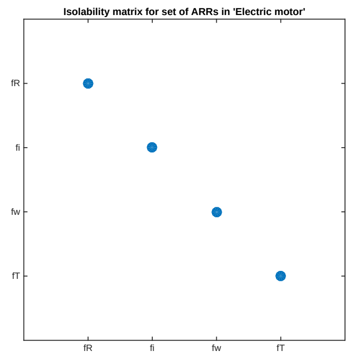
Test selection

- 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.

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)

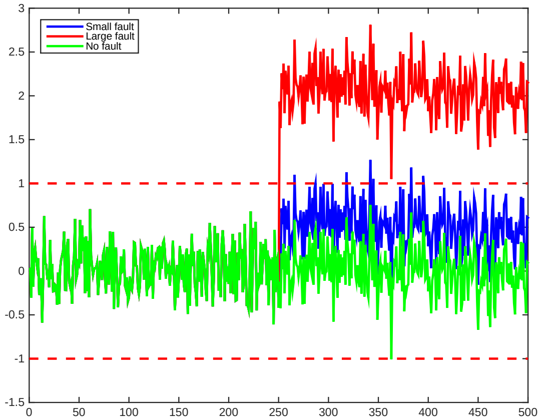
Output: A small set of tests with required isolability

Fault isolability of tests

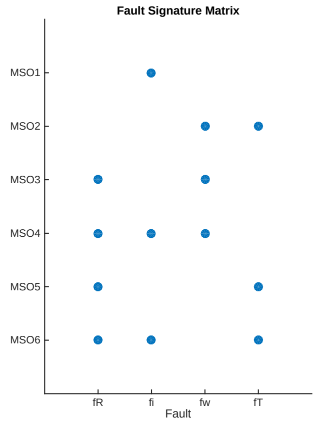
	NF	f_1	f_2
T	0	X	0

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 is a minimal hitting set problem



Requirement for each desired diagnosability property:

Detectability:
f_R: $T_1 = \{3, 4, 5, 6\}$
...

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_ω: $T_4 = \{5, 6\}$
...

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

- Find all minimal test sets with a minimal hitting set algorithm.

  Might easily lead to computationally intractable problems.

J. De Kleer, BC Williams. "Diagnosing multiple faults". Artificial intelligence 32 (1), 97-130, 1987.

- 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.

Cormen, L., Leiserson, C. E., and Ronald, L. (1990). Rivest, "Introduction to Algorithms.", 1990.

- Iterative approach involving both test selection and residual generation.

Test selection

Many more alternatives in for example:

De Kleer, Johan. "Hitting set algorithms for model-based diagnosis." 22th International Workshop on Principles of Diagnosis, DX, 2011.

Example

	NF	f_R	f_i	f_ω	f_T
f_R	$3-6$	–	3, 5	5, 6	3, 4
f_i	1, 4, 6	1	–	1, 6	1, 4
f_ω	$2-4$	2	2, 3	–	3, 4
f_T	2, 5, 6	2	2, 5	5, 6	–

- 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:

	f_R	f_i	f_ω	f_T
f_R	1	1	0	0
f_i	1	1	0	0
f_ω	0	0	1	0
f_T	0	0	0	1

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

Greedy search incorporating residual generation

Basic idea

Select residuals adding the most number of desired diagnosis properties.

	f_1	f_2	f_3
r_1	X	X	
r_2	X		X
r_3		X	X
r_4	X		

	NF	f_1	f_2	f_3
f_1	1, 2, 4	–	2, 4	1, 4
f_2	1, 3	3	–	1
f_3	2, 3	3	2	–

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

— Residual generation —

Outline

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

$$e_1: \quad \dot{x}_1 - x_2 = 0$$
$$e_2: \quad \dot{x}_3 - x_4 = 0$$
$$e_3: \quad \dot{x}_4 x_1 + 2x_2x_4 - y_1 = 0$$
$$e_4: \quad x_3 - y_3 = 0$$
$$e_5: \quad x_2 - y_2 = 0$$

	x_1	x_2	x_4	x_3
e_5		X		
e_1	X	X		
e_3	X	X	X	
e_2			X	
e_4				X

Solve according to order in decomposition:

$$e_4: x_3 := y_3 \qquad e_2: x_4 := \dot{x}_3$$
$$e_3: \dot{x}_1 := x_2 \qquad e_1: x_2 := \frac{-\dot{x}_4 x_1 + y_1}{2x_4}$$

Compute a residual:

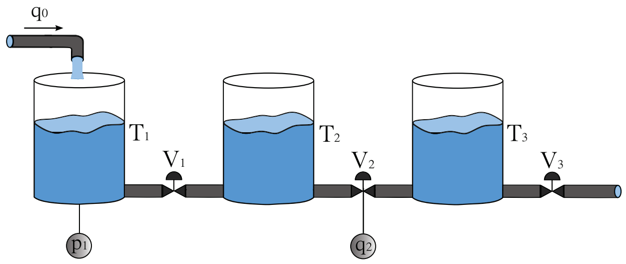
$$e_5: r := y_2 - x_2$$

Basic principle - Sequential residual generation

Basic approach

1. Given a testable set of equations (MSO/MTES/...)
2. Compute a matching (Dulmage-Mendelsohn decomposition)
3. Solve according to decomposition (numerically or symbolically)
4. Compute residuals with the redundant equations

Illustrative example



$$e_1: q_1 = \frac{1}{R_{V1}}(p_1 - p_2) \qquad e_5: \dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2) \qquad e_9: y_3 = q_0$$

$$e_2: q_2 = \frac{1}{R_{V2}}(p_2 - p_3) \qquad e_6: \dot{p}_3 = \frac{1}{C_{T3}}(q_2 - q_3) \qquad e_{10}: \dot{p}_1 = \frac{dp_1}{dt}$$

$$e_3: q_3 = \frac{1}{R_{V3}}(p_3) \qquad e_7: y_1 = p_1 \qquad e_{11}: \dot{p}_2 = \frac{dp_2}{dt}$$

$$e_4: \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \qquad e_8: y_2 = q_2 \qquad e_{12}: \dot{p}_3 = \frac{dp_3}{dt}$$

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 $(q_0, q_1, q_2, p_1, p_2, \dot{p}_1, \dot{p}_2)$

$$e_1: q_1 = \frac{1}{R_{V1}}(p_1 - p_2) \qquad e_7: y_1 = p_1 \qquad e_{10}: \dot{p}_1 = \frac{dp_1}{dt}$$

$$e_4: \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \qquad e_8: y_2 = q_2 \qquad e_{11}: \dot{p}_2 = \frac{dp_2}{dt}$$

$$e_5: \dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2) \qquad e_9: y_3 = q_0$$

Redundant equation

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

Compute a matching

$$e_1: q_1 = \frac{1}{R_{V1}}(p_1 - p_2) \qquad e_7: y_1 = p_1 \qquad e_{10}: \dot{p}_1 = \frac{dp_1}{dt}$$

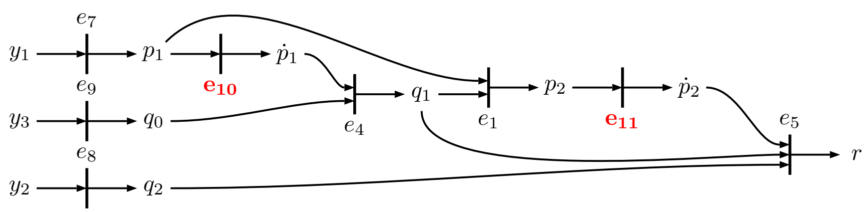
$$e_4: \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \qquad e_8: y_2 = q_2 \qquad e_{11}: \dot{p}_2 = \frac{dp_2}{dt}$$

$$e_9: y_3 = q_0$$

	$\dot{p}_2$	p_2	q_1	$\dot{p}_1$	p_1	q_0	q_2
e_{11}	**X**	X					
e_1		**X**	X		X		
e_4			**X**	X		X	
e_{10}				**X**	X		
e_7					**X**		
e_9						**X**	
e_8							**X**

Computational graph for matching

	$\dot{p}_2$	p_2	q_1	$\dot{p}_1$	p_1	q_0	q_2
e_{11}	**X**	X					
e_1		**X**	X		X		
e_4			**X**	X		X	
e_{10}				**X**	X		
e_7					**X**		
e_9						**X**	
e_8							**X**



Equations e_{10} and e_{11} in **derivative** causality.

Residual generator code

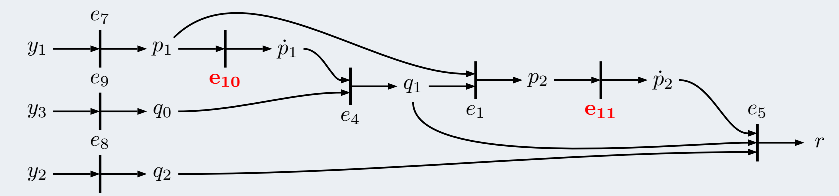
Fairly straightforward to generate code automatically for this case

Code

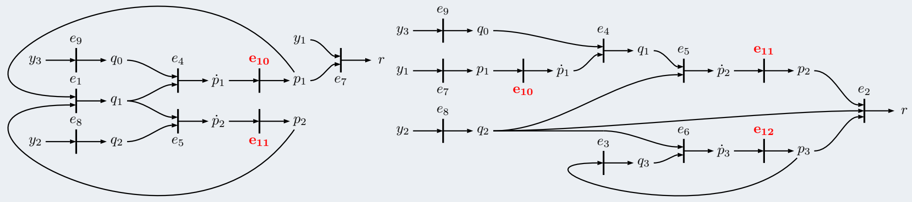
```
q2 = y2; % e8
q0 = y3; % e9
p1 = y1; % e7
dp1 = ApproxDiff(p1,state.p1,Ts);  % e10
q1 = q0-CT1*dp1; % e4
p2 = p1-Rv1*q1; % e1
dp2 = ApproxDiff(p2,state.p2,Ts);  % e11
r = dp2-(q1-q2)/CT2; % e5
```


Causality of sequential residual generators

Derivative causality



Integral and mixed causality



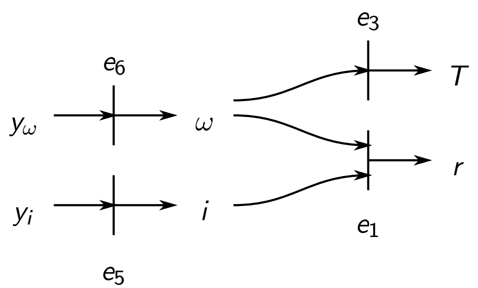
Causality of sequential residual generators

- 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

	T	i	ω	
e_3	X		X	f_b
e_5		X		f_i
e_6			X	f_ω
e_1		X	X	f_R

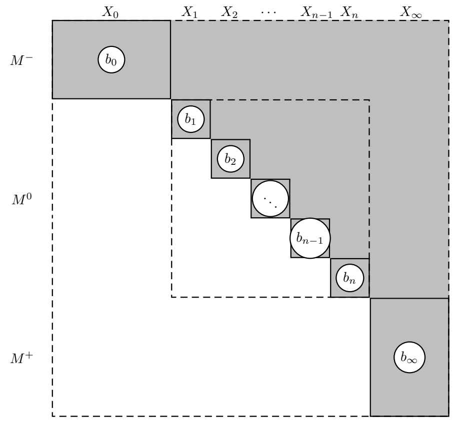


Here the matching gives a computational sequence for all variables

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

Hall components and computational loops

5 equations, 4 unknowns

$$e_1: \quad \dot{x}_1 - x_2 = 0$$
$$e_2: \quad \dot{x}_3 - x_4 = 0$$
$$e_3: \quad \dot{x}_4 x_1 + 2x_2x_4 - y_1 = 0$$
$$e_4: \quad x_3 - y_3 = 0$$
$$e_5: \quad x_2 - y_2 = 0$$

	x_1	x_2	x_4	x_3
e_5		X		
e_1	**X**	X		
e_3	X	**X**	X	
e_2			**X**	
e_4				**X**

- 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 identifing computational loops

Observer based residual generation

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

1. Estimate/compute unknown variables $\hat{x}$
2. Check if model is consistent with $\hat{x}$

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

$$\dot{x} = g(x, u) \qquad \dot{\hat{x}} = g(\hat{x}, u) + K(y - h(\hat{x}, u))$$
$$y = h(x, u) \qquad r = y - h(\hat{x}, u)$$

Design procedures typically available for state-space models

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

Submodels like MSE/MTES are not typically in state-space form!

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

$$g_i(dx_1, x_1, x_2, z, f) = 0 \qquad i = 1, \ldots, n$$
$$dx_1 = \frac{d}{dt}x_1$$

- 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

Example: Three Tank example again

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

$$e_4: \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \qquad e_7: y_1 = p_1 \qquad e_9: y_3 = q_0$$

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

Dynamic equations	Algebraic equations	Redundant equation
$\dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1)$	$0 = q_0 - y_3$	$r = y_1 - p_1$
$\dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2)$	$0 = q_1 R_{V1} - (p_1 - p_2)$	
	$0 = q_2 - y_2$	

Partition to DAE observer

Partition model using structure

Dynamic equations	Algebraic equations	Redundant equation
$\dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1)$	$0 = q_0 - y_3$	$r = y_1 - p_1$
$\dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2)$	$0 = q_1 R_{V1} - (p_1 - p_2)$	
	$0 = q_2 - y_2$	

DAE observer

$$\dot{\hat{p}}_1 = \frac{1}{C_{T1}}(\hat{q}_0 - \hat{q}_1) + K_1 r \qquad 0 = \hat{q}_0 - y_3$$

$$\dot{\hat{p}}_2 = \frac{1}{C_{T2}}(\hat{q}_1 - \hat{q}_2) + K_2 r \qquad 0 = \hat{q}_1 R_{V1} - (\hat{p}_1 - \hat{p}_2)$$

$$0 = \hat{q}_2 - y_2$$

$$0 = r - y_1 + \hat{p}_1$$

Models with low differential index

A low-index DAE model

$$g_i(dx_1, x_1, x_2, z, f) = 0 \qquad i = 1, \ldots, n$$

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

has the property

$$\left.\begin{pmatrix} \frac{\partial g}{\partial dx_1} & \frac{\partial g}{\partial x_2} \end{pmatrix}\right|_{x=x_0,\ z=z_0} \text{ full column rank}$$

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

$$\dot{\hat{x}}_1 = g_1(\hat{x}_1, \hat{x}_2, z) + K(\hat{x}, z) g_r(\hat{x}_1, \hat{x}_2, z)$$

$$0 = g_2(\hat{x}_1, \hat{x}_2, z)$$

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

DAE observer for low-index model

The observer

$$\begin{aligned}\dot{\hat{x}}_1 &= g_1(\hat{x}_1, \hat{x}_2, z) + K(\hat{x}, z) g_r(\hat{x}_1, \hat{x}_2, z)\\ 0 &= g_2(\hat{x}_1, \hat{x}_2, z)\\ r &= g_r(\hat{x}_1, \hat{x}_2, z)\end{aligned}$$

corresponds to the standard setup DAE

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

where the mass matrix M is given by

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

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.

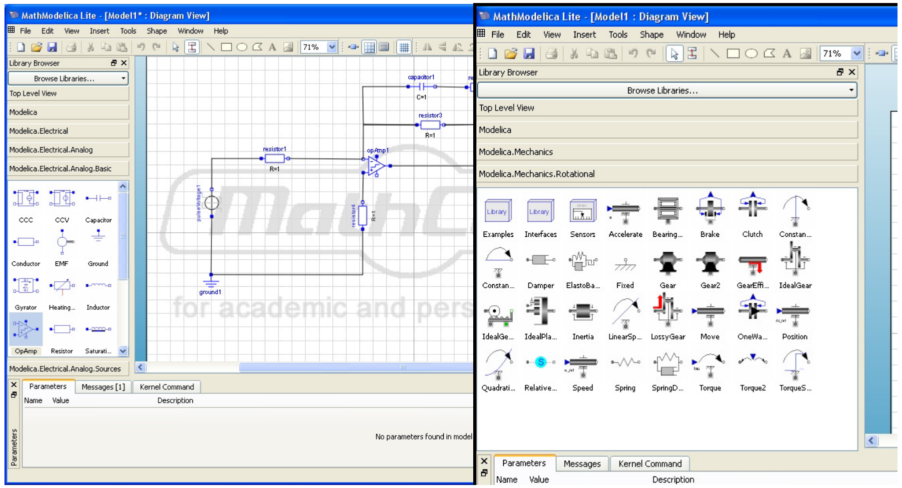
- Fairly straightforward, details not included, to generate code for function $f(x)$ above for low-index problems
- Code generation similar to the sequential residual generators, but only for the highest order derivatives
- Utilizes efficient numerical techniques for integration

— *Diagnosability analysis* —

Outline

Problem formulation

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



Diagnosability analysis

Diagnosability analysis

Determine for a

1. model
2. 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.

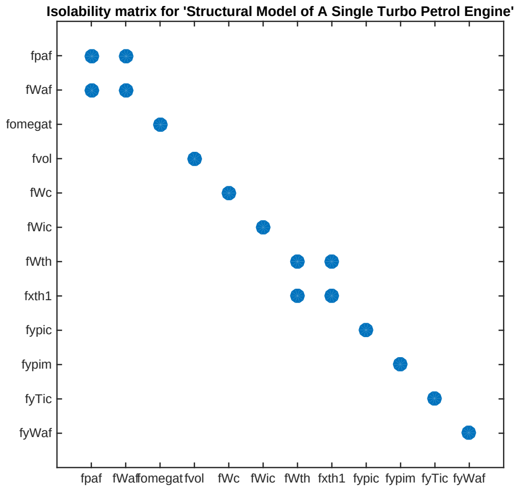
Better way

Utilize steps in the MSO algorithm; equivalence classes!

Isolability matrices

Interpretation

A X in position (i,j) indicates that fault f_i can not be isolated from fault f_j



Diagnosability analysis for a set of tests/model

A test/residual with fault sensitivity

	f_1	f_2
r	X	0

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

	f_1	f_2	f_3
r_1	X	X	0
r_2	0	X	X

The corresponding isolability matrix is then

	f_1	f_2	f_3
f_1	X	X	0
f_2	0	X	0
f_3	0	X	X

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,

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

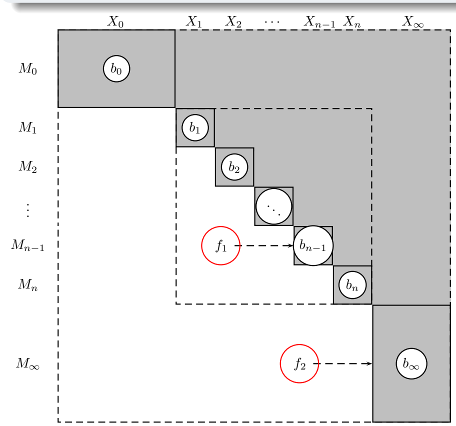
1. Introduce new unknown variable x_f
2. Add new equation $x_f = f$

Now, the model fulfills the assumption.

Structural detectability and Dulmage-Mendelsohn

Detectability

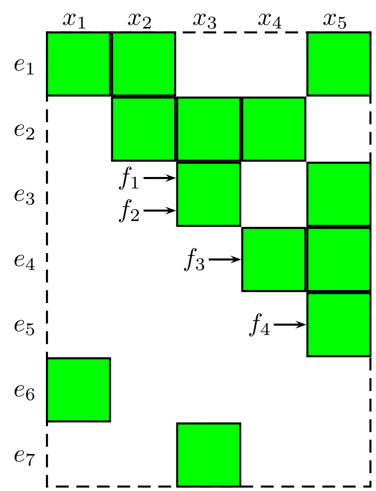
A fault f is structurally detectable if $e_f \in M^+$.



- Fault f_1 not detectable
- Fault f_2 detectable

Detectability in small example

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



Structural isolability

Isolability

A fault F_i is isolable from fault F_j if $\mathcal{O}(F_i) \not\subseteq \mathcal{O}(F_j)$

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_{f_i} but not e_{f_j}

	F_i	F_j
r	X	0

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

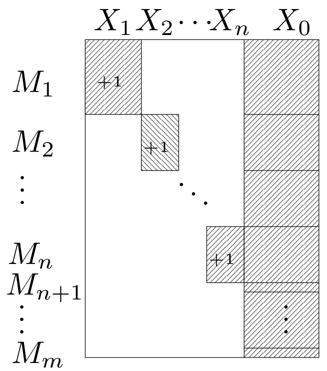
Structural isolability

F_i structurally isolable from F_j iff $e_{f_i} \in \left(M \setminus \{e_{f_j}\}\right)^+$

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



- 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

$e_1: \quad \dot{x}_1 = -x_1 + x_2 + x_5$

$e_2: \quad \dot{x}_2 = -2x_2 + x_3 + x_4$

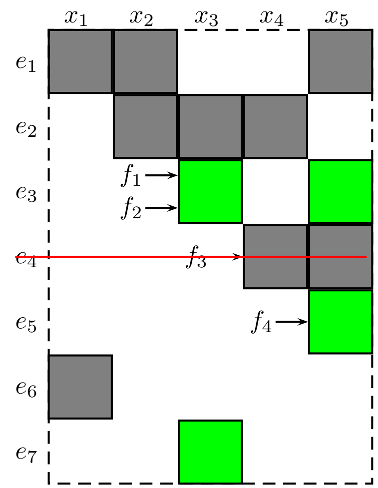
$e_3: \quad \dot{x}_3 = -3x_3 + x_5 + f_1 + f_2$

$e_4: \quad \dot{x}_4 = -4x_4 + x_5 + f_3$

$e_5: \quad \dot{x}_5 = -5x_5 + u + f_4$

$e_6: \quad y_1 = x_1$

$e_7: \quad y_2 = x_3$



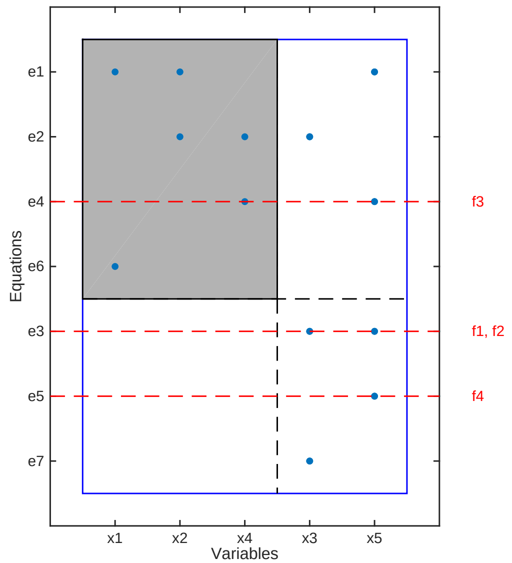
***Equivalence class* $[e_4]$**

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

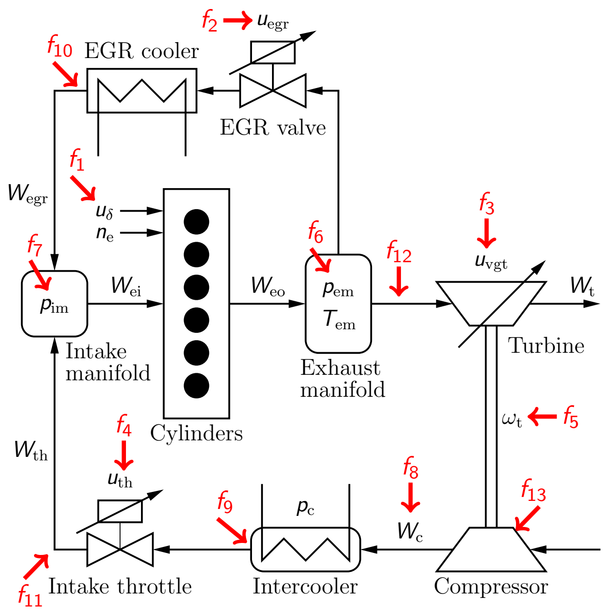
Method - Diagnosability analysis of model

Method

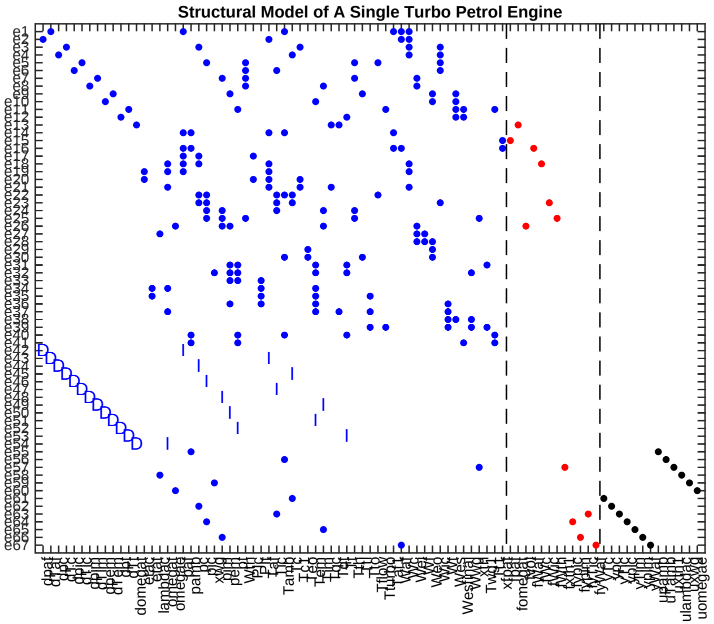
1. Determine equivalence classes in M^+
   - $M_{e_f} = M \setminus \{e_f\}$
   - $[e_f] = M^+ \setminus M^+_{e_f}$
2. Faults appearing in the same equivalence class are not isolable
3. Faults appearing in separate equivalence classes are isolable



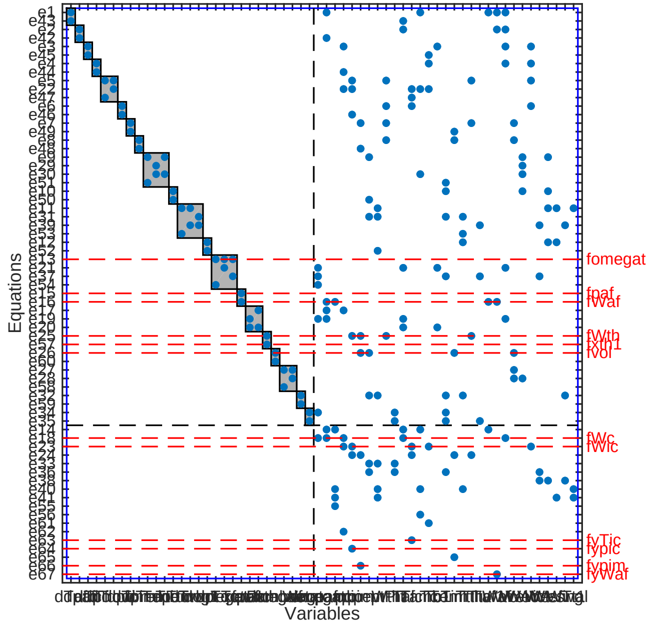
Example system - A automotive engine with EGR/VGT



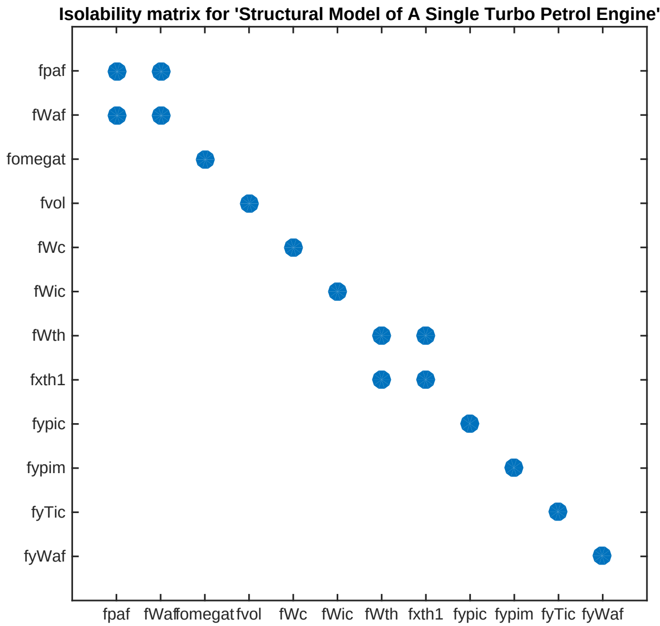
Model structure



Dulmage-Mendelsohn with equivalence classes



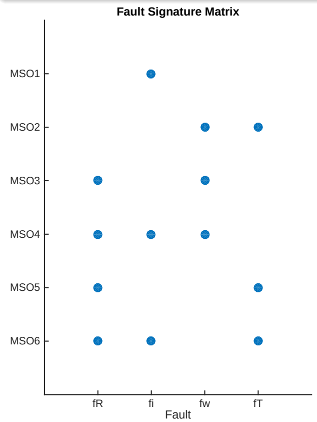
Fault isolation matrix for engine model



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

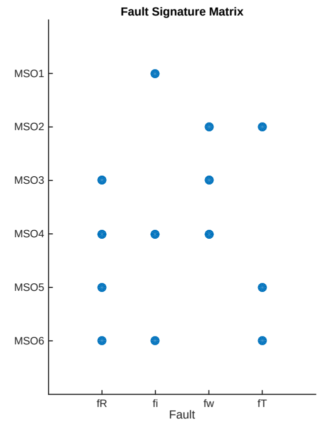
	f_i	f_j
r_1	X	

isolates fault f_i from f_j.

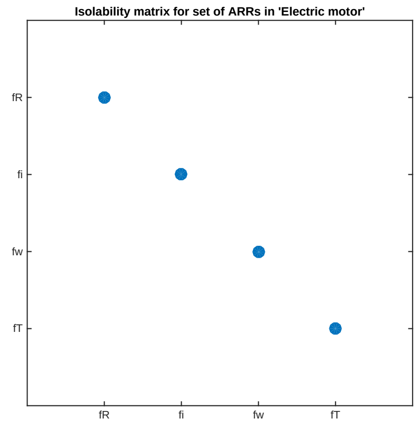
For example, MSO2 isolates

1. Fault f_w from f_R and f_i,
2. Fault f_T from f_R and f_i

Diagnosability analysis for a fault signature matrix



$\Rightarrow$



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

A word on fault isolation and exoneration

	f_1	f_2	f_3	f_4
$\mathcal{M}_1$	0	0	1	1
$\mathcal{M}_2$	1	0	1	0
$\mathcal{M}_3$	1	1	0	1

$\Rightarrow$

	f_1	f_2	f_3	f_4
f_1	1	0	0	0
f_2	1	1	0	1
f_3	0	0	1	0
f_4	0	0	0	1

Q: Why is not the isolability matrixdiagonal 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

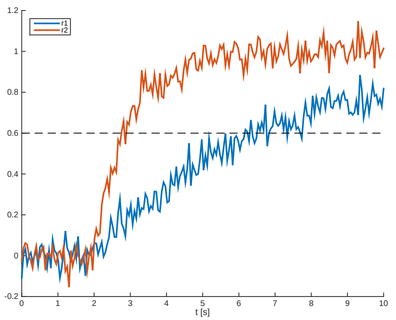
$r_2 > J \Rightarrow f_1$ or f_3

$\Rightarrow$ Minimal consistency based diagnoses with no exoneration assumption: $\mathcal{D}_1 = \{f_3\}$, $\mathcal{D}_2 = \{f_1, f_4\}$

Fault isolation and exoneration

Fault f_3 occurs at $t = 2$ sec.

	f_1	f_2	f_3	f_4
$\mathcal{M}_1$	0	0	1	1
$\mathcal{M}_2$	1	0	1	0
$\mathcal{M}_3$	1	1	0	1



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

— *Sensor Placement Analysis* —

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 motivating example and problem formulation

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

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 $\mathcal{P}$ of possible sensor locations
- A detectability and isolability performance specification

Minimal Sensor Set

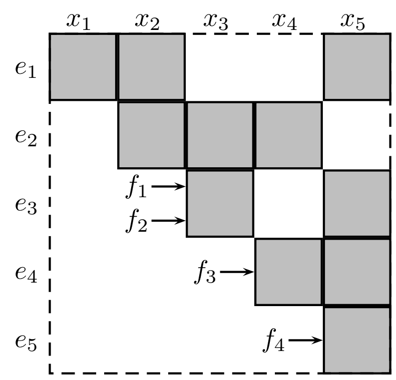
A multiset S, defined on $\mathcal{P}$, is a minimal sensor set if the specification is fulfilled when the sensors in S are added, but not fulfilled when any proper subset is added.

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

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

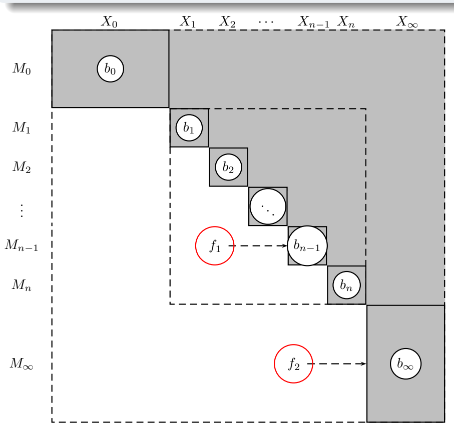


Detectability

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

Detectability

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

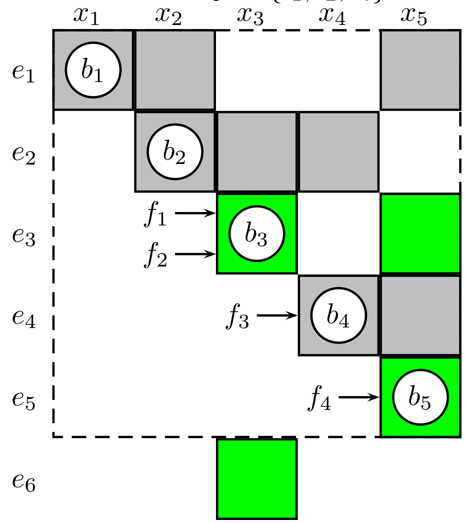


- Fault f_1 not detectable
- Fault f_2 detectable

Sensor Placement for Detectability

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

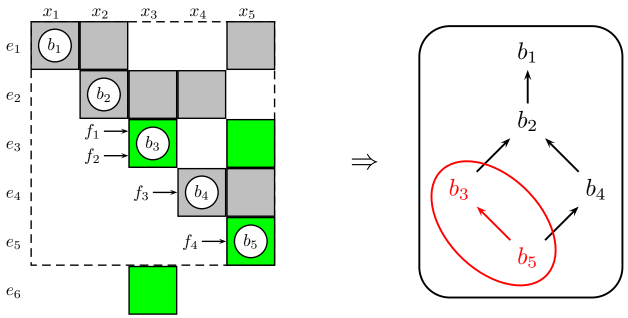
Measure $x_3 \rightarrow \{f_1, f_2, f_4\}$



Define a Partial Order on b_i

Partial Order on b_i

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



Lemma

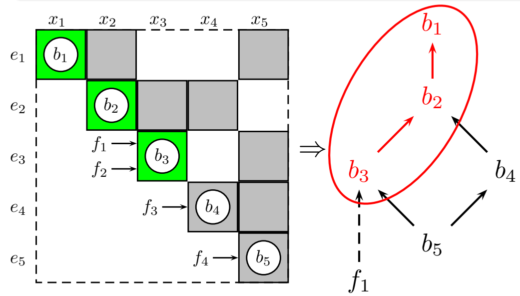
Let e_i measure a variable in b_j then

all equal and lower ordered blocks are included in the overdetermined part.

Minimal Sensor Sets - Detectability

Detectability Set

$$D([f_i]) = \text{measurements that give detectability of fault } f_i$$
$$= \text{all variables in equal and higher ordered blocks}$$



$\Rightarrow$

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

Minimal Sensor Sets - Detectability

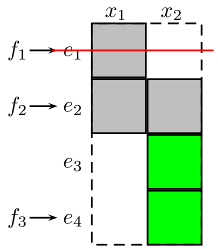
Sensor set for detectability

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.

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

Sensor placement for isolability



f_i is isolable from f_1 if there exists a residual r such that

	f_i	f_1
r	X	0

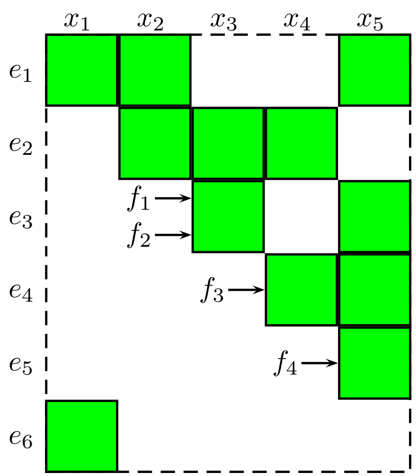
Isolability characterization: f_i is structurally isolable from f_1 if $e_{f_i} \in (M \setminus \{e_{f_1}\})^+$.

f_3 is isolable from f_1 in $M = \{e_1, \dots, e_4\}$ and f_3 is detectable in $M \setminus \{e_1\}$

The sensor placement problem of achieving isolability from f_1 in M is transformed to the problem of achieving detectability in $M \setminus \{e_1\}$.

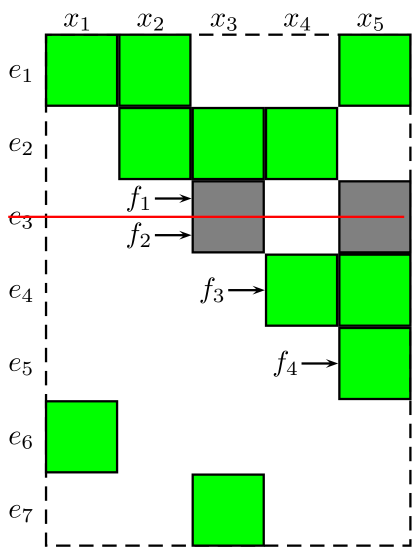
Proceed as in the linear case to achieve isolability.

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

  isolate from $\{f_1, f_2\} : \{x_3, x_4\}$

  isolate from $f_3 : \{x_3, x_4\}$ $\Rightarrow \{x_3\}, \{x_4\}$

  isolate from $f_4 : \{x_2, x_3, x_4, x_5\}$

- 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!

Achieving maximum isolability

- Minimal sensor sets for full detectability
  $$\{x_1\}, \{x_2\}, \{x_3, x_4\}$$
- The first set $\{x_1\}$ was selected, iterate for all!
- Minimal sensor sets for maximum isolability:
  $$\{x_1, x_3\}, \{x_1, x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}$$

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 $\mathcal{F}$ 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 $\mathcal{F}$ 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.

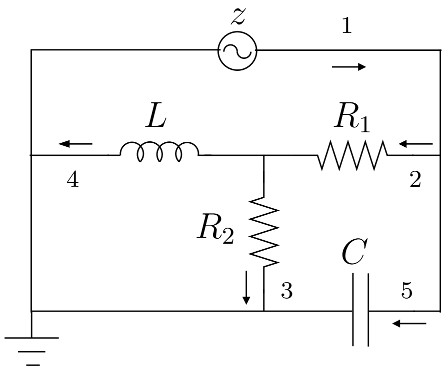
Method summary

1. For each detectability and isolability requirement, compute detectability sets
   - Dulmage-Mendelsohn decomposition + identify partial order
2. 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

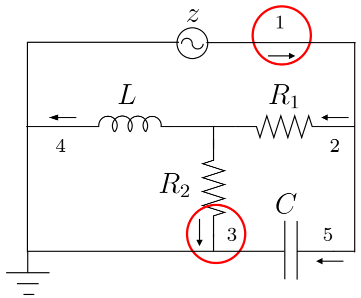
A small electrical circuit with 5 components that may fail



$$v_1 = v_5 \qquad v_5 = v_2 + v_3$$
$$i_1 = i_2 + i_5 \qquad i_1 = i_3 + i_4 + i_5$$
$$v_1 = z \qquad v_2 = R_1 i_2$$
$$v_4 = L\frac{d}{dt}i_4 \qquad i_5 = C\frac{d}{dt}v_5$$
$$v_3 = v_4 \qquad v_3 = R_2 i_3$$

- 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

$\{i_1, i_3\}$, $\{i_1, i_4\}$, $\{i_2, i_3, i_5\}$, $\{i_2, i_4, i_5\}$, $\{i_3, i_4, i_5\}$

— *Case study and software demo* —

Outline

Two examples

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

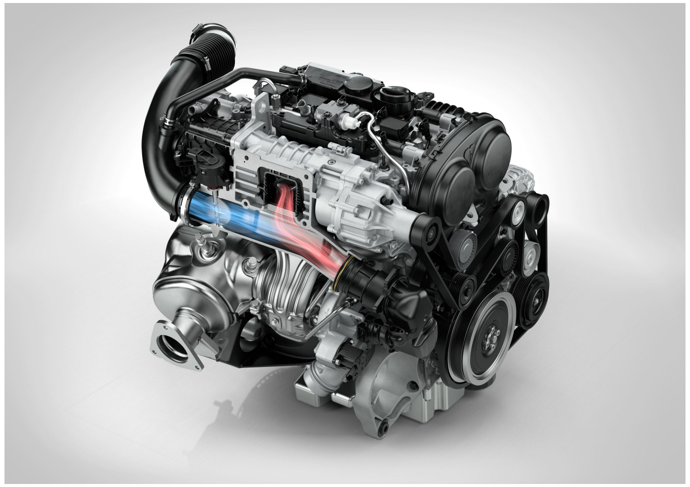
Analysis of a three-tank system model

Shows examples on what can be done with structural analysis and code generation

Software

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

Automotive engine



Example objective

Show how non-trivial results can be obtained using only structural information of a complex system

Modelling of automotive engines

960

Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics

J Wahlström* and **L Eriksson**
Department of Electrical Engineering, Linköping University, Linköping, Sweden

The manuscript was received on 12 February 2010 and was accepted after revision for publication on 4 January 2011.

DOI: 10.1177/0954407011398177

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

Modelling of automotive engines, non-linear equations

$$W_{ei} = \frac{\eta_{vol}\, p_{im}\, n_e\, V_d}{120\, R_a\, T_{im}} \quad (11)$$

where p_{im} and T_{im} are the pressure and temperature respectively in the intake manifold, n_e is the engine speed, and V_d is the displaced volume. The volumetric efficiency is in its turn modelled as

$$\eta_{vol} = c_{vol1}\sqrt{p_{im}} + c_{vol2}\sqrt{n_e} + c_{vol3} \quad (12)$$

The fuel mass flow W_f into the cylinders is controlled by u_δ, which gives the injected mass of fuel in milligrams per cycle and cylinder as

$$W_f = \frac{10^{-6}}{120}\, u_\delta\, n_e\, n_{cyl} \quad (13)$$

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

$$W_{eo} = W_f + W_{ei} \quad (14)$$

The oxygen-to-fuel ratio λ_O in the cylinder is defined as

$$\lambda_O = \frac{W_{ei}\, X_{Oim}}{W_f\, (O/F)_s} \quad (15)$$

the initialization is that the cylinder mass flow model has a mean absolute relative error of 0.9 per cent and a maximum absolute relative error of 2.5 per cent. The parameters are then tuned according to the method in section 8.1.

4.2 Exhaust manifold temperature

The exhaust manifold temperature model consists of a model for the cylinder-out temperature and a model for the heat losses in the exhaust pipes.

4.2.1 Cylinder-out temperature

The cylinder-out temperature T_e is modelled in the same way as in reference [**23**]. This approach is based upon ideal-gas Seliger cycle (or limited pressure cycle [**1**]) calculations that give the cylinder-out temperature as

$$T_e = \eta_{sc}\, \Pi_e^{1-1/\gamma_a}\, r_c^{1-\gamma_a}\, x_p^{1/\gamma_a - 1} \times \left(q_{in}\left(\frac{1-x_{cv}}{c_{pa}} + \frac{x_{cv}}{c_{Va}}\right) + T_1\, r_c^{\gamma_a - 1}\right) \quad (17)$$

where η_{sc} is a compensation factor for non-ideal cycles and x_{cv} the ratio of fuel consumed during constant-volume combustion. The rest of the fuel, i.e. $(1 - x_{cv})$ is used during constant-pressure com-

Structural modelling

```
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

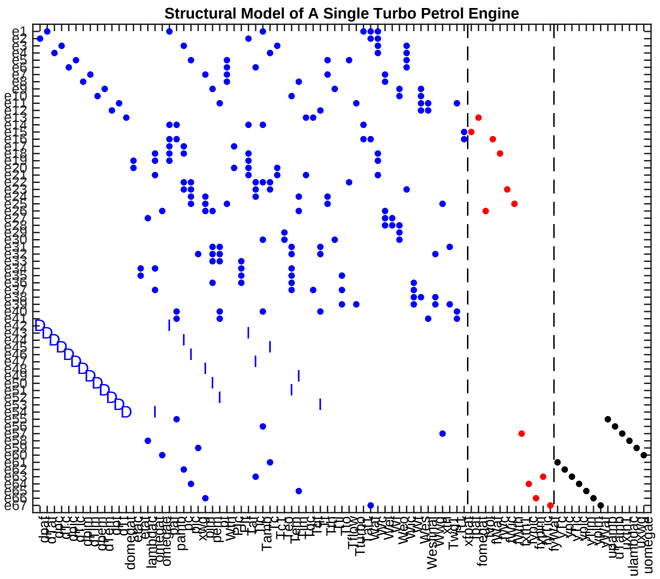
Check model for problems

- Number of known/unknown/fault variables
- Are all signals included in the model
- Degree of redundancy
- Do the model have underdetermined parts

```
>> sm.Lint();
Model: Structural Model of A Single Turbo Petrol Engine

  Type: Structural, dynamic

  Variables and equations
    60 unknown variables
    13 known variables
    12 fault variables
    67 equations, including 13 differential constraints

  Degree of redundancy: 7

  Model validation finished with 0 errors and 0 warnings.
```

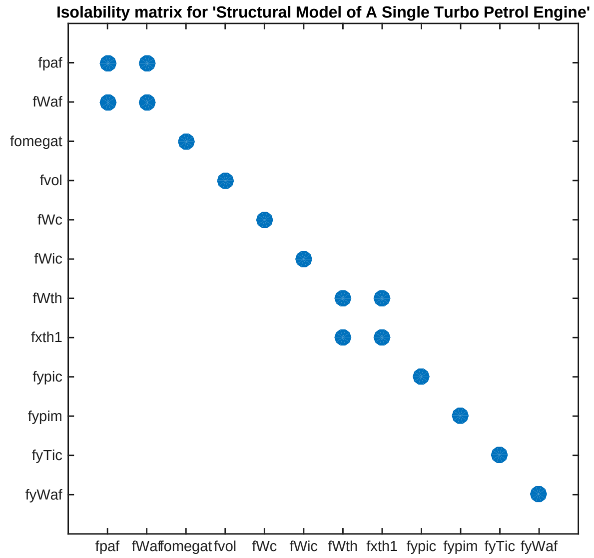

Plot model structure

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



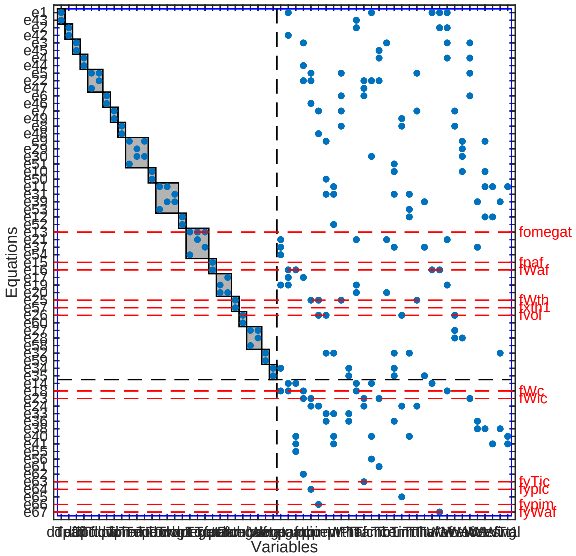
Isolability analysis

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



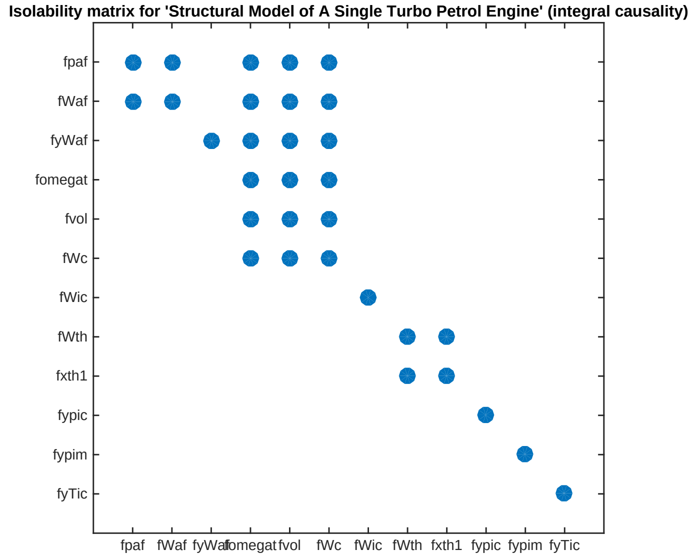
Isolability analysis – Dulmage-Mendelsohn decomp.

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



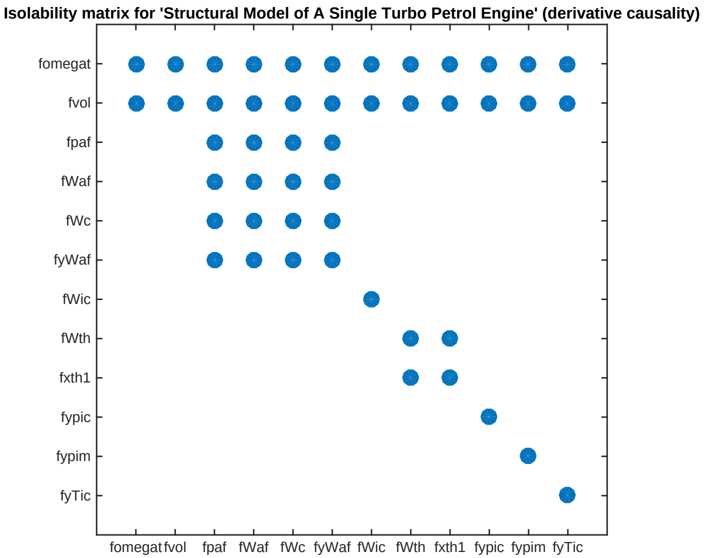
Isolability analysis – integral causality

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



Isolability analysis – derivative causality

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



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

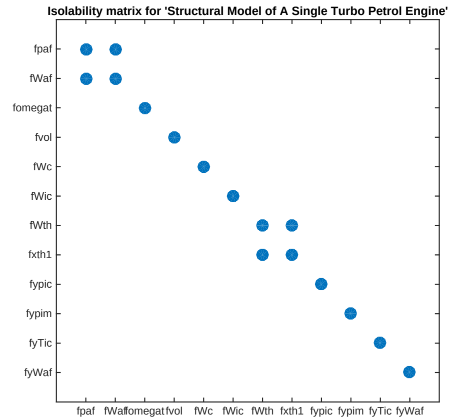
- Finds all possible fault signatures (159)
- For each fault signature, we know which constraints are needed to compute a residual

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

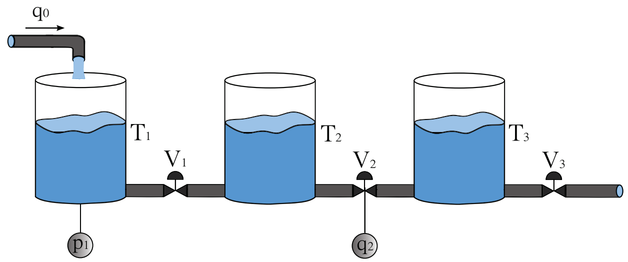
- We have here 159 candidate residual generators
- Do we really need all of them?

Test selection – all 159 is not needed

```
>> ts = sm.TestSelection( FSM, 'method', 'aminc')
ts =
    12    22    29    55   111   113   150
% 7 tests
>> sm.IsolabilityAnalysisFSM(FSM(ts,:));
```



Example with symbolic equations and code generation



$$e_1: q_1 = \frac{1}{R_{V1}}(p_1 - p_2) \qquad e_5: \dot{p}_2 = \frac{1}{C_{T2}}(q_1 - q_2) \qquad e_9: y_3 = q_0$$

$$e_2: q_2 = \frac{1}{R_{V2}}(p_2 - p_3) \qquad e_6: \dot{p}_3 = \frac{1}{C_{T3}}(q_2 - q_3) \qquad e_{10}: \dot{p}_1 = \frac{dp_1}{dt}$$

$$e_3: q_3 = \frac{1}{R_{V3}}(p_3) \qquad e_7: y_1 = p_1 \qquad e_{11}: \dot{p}_2 = \frac{dp_2}{dt}$$

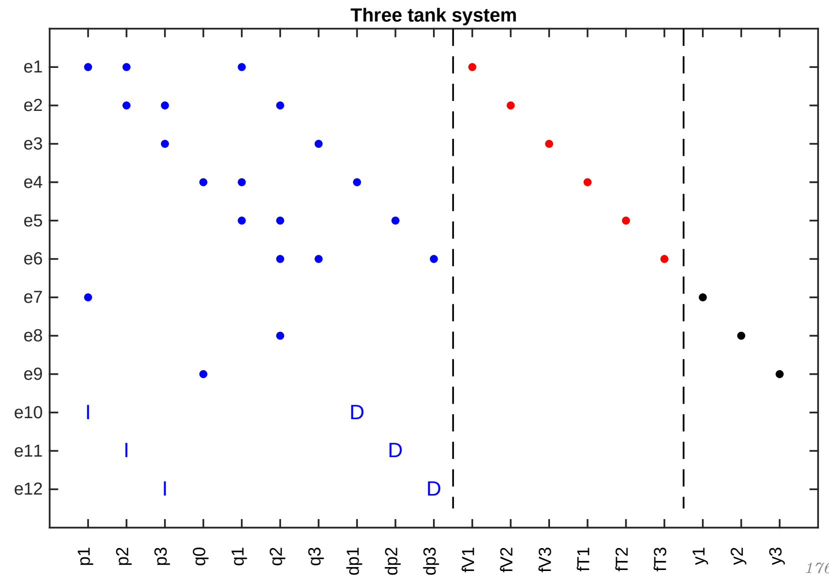
$$e_4: \dot{p}_1 = \frac{1}{C_{T1}}(q_0 - q_1) \qquad e_8: y_2 = q_2 \qquad e_{12}: \dot{p}_3 = \frac{dp_3}{dt}$$

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 );
```

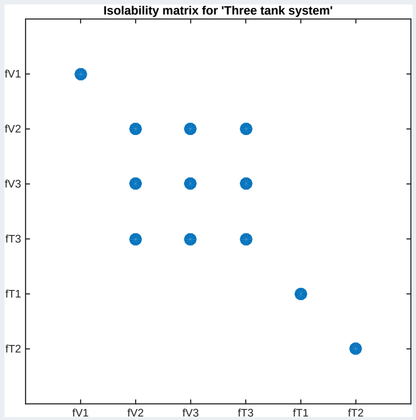

Structure is automatically computed

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

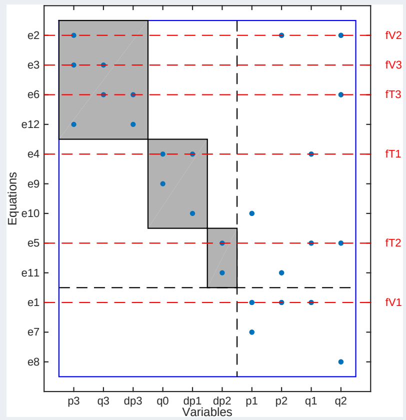


Methods for structural models directly available

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



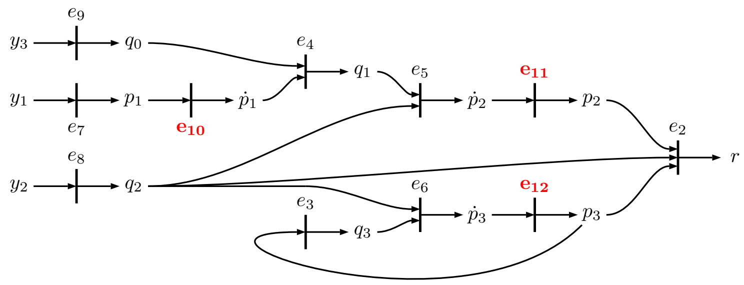

```
>> sm.PlotDM('eqclass',true,'fault',true);
```




```
>> sm.MSO()
ans =
 [1x11 double] [1x8 double] [1x9 double] [1x10 double] [1x11 double] [1x11 double]
```


Code generation: Sequential residual generator

MSO $\mathcal{M} = \{e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}\}$, 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

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


Generated code (slightly cropped)

```
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* —

Outline

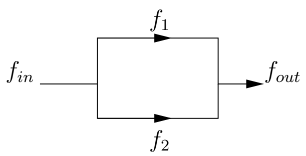
Analytical vs structural properties

- 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

Murota, Kazuo. "*Matrices and matroids for systems analysis*". Springer, 2009.

You have to be careful



$$f_{in} = x$$
$$f_{out} = x$$
$$y_1 = f_{in}$$
$$y_2 = f_{out}$$
$$x = f_1 + f_2$$

	x	f_{in}	f_1	f_2	f_{out}
c_1	X	X			
c_2	X				X
c_3		X			
c_4					X
c_5	X		X	X	

Now, a leak is structurally detectable!

For structural methods to be effective, do as little manipulation as possible. Modelica/Simulink is a quite good representation of models for structural analysis.

Basic assumptions for structural analysis

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

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}}^{A=} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Redundancy relation $y_1 - y_2 = 0$.

$$A_{str} = \begin{bmatrix} X & X \\ X & X \end{bmatrix}$$

Structual matrix just-determined $\Rightarrow$ no redundancy

Wrong structural results because A is rank deficient:

$$rank(A) = 1 < 2 = sprank(A)$$

Exercise

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 orignal model. Compute the fault isolability for this model and compare it with the isolability obtained in (a).

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

$$e_2 : T_m = K_a i^2$$

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

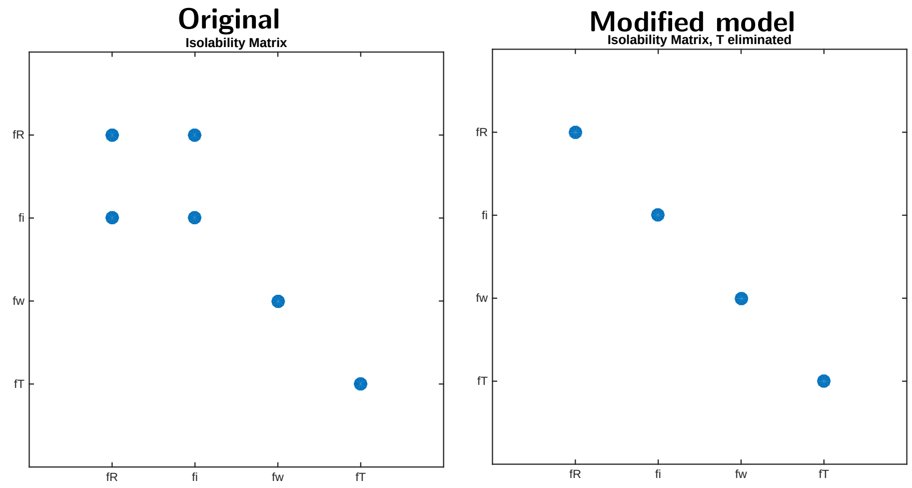
$$e_4 : T = T_m - T_l$$

$$e_5 : y_i = i + f_i$$

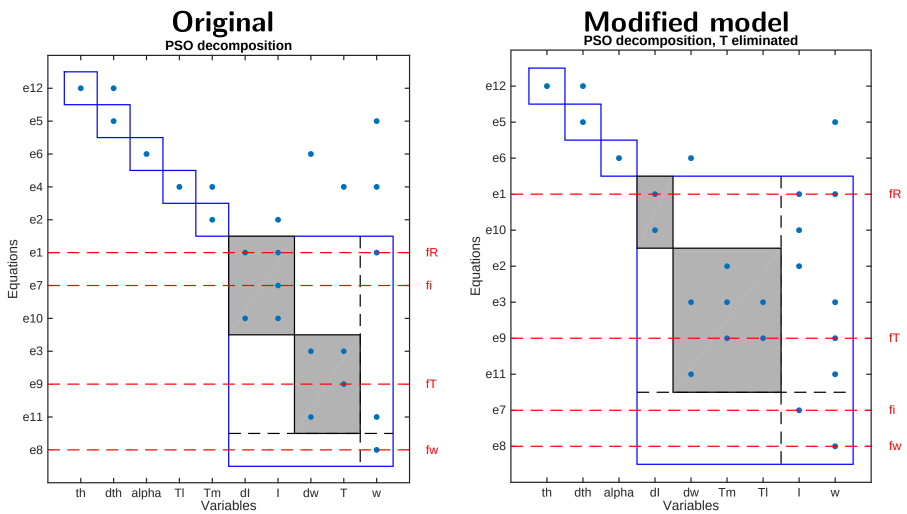
$$e_6 : y_\omega = \omega + f_\omega$$

$$e_7 : y_T = T + f_T$$

Isolability properties depends on model formulation



Isolability properties depends on model formulation



— *Concluding remarks* —

Some take home messages

Structural models

- Coarse models that do not need paramerer 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

LINKÖPING UNIVERSITY

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.
IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 38(1), 2008.

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.
  IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 38(6):1398–1410, 2008.
- Erik Frisk, Anibal Bregon, Jan Åslund, Mattias Krysander, Belarmino Pulido, and Gautam Biswas.
  Diagnosability analysis considering causal interpretations for differential constraints.
  IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 42(5):1216–1229, September 2012.

Some publications on structural analysis from our group

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.
  IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, 40(6):1310–1328, 2010.
- Carl Svärd, Mattias Nyberg, and Erik Frisk.
  Realizability constrained selection of residual generators for fault diagnosis with an automotive engine application.
  IEEE Transactions on Systems, Man, and Cybernetics: Systems, 43(6):1354–1369, 2013.

Publications on Structural Analysis from our group

Application studies

- Dilek Dustegör, Erik Frisk, Vincent Coquempot, Mattias Krysander, and Marcel Staroswiecki.
  Structural analysis of fault isolability in the DAMADICS benchmark.
  Control Engineering Practice, 14(6):597–608, 2006.
- Carl Svärd and Mattias Nyberg.
  Automated design of an FDI-system for the wind turbine benchmark.
  Journal of Control Science and Engineering, 2012, 2012.
- Carl Svärd, Mattias Nyberg, Erik Frisk, and Mattias Krysander.
  Automotive engine FDI by application of an automated model-based and data-driven design methodology.
  Control Engineering Practice, 21(4):455–472, 2013.