A Realistic Digital Twin of a Vehicular Engine System

A Simulation Environment Platform for Fault Diagnosis

Mark Kok Yew Ng

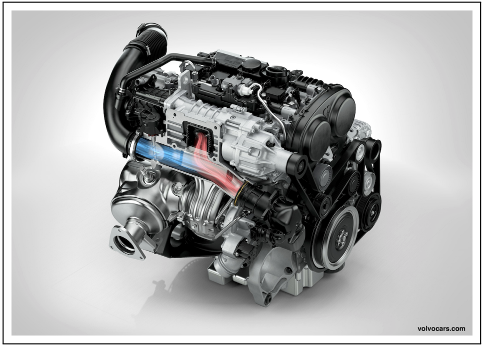
Senior Lecturer, School of Engineering, Ulster University, UK
Adjunct Senior Research Fellow, School of Engineering, Monash University, Malaysia
ILN+ Researcher in Residence, Digital Catapult

mark.ng@ulster.ac.uk
markusng.com

The 8th IEEE Conference on Control Technology and Applications (CCTA) 2024
Workshop TuWH2T3

20th August 2024

Presentation Overview



*Requirements: Matlab R2020b or later

Key Members

Prof. Erik Frisk
Linköping University

Prof. Lars Eriksson
Linköping University

Assoc. Prof. Mattias Krysander
Linköping University

Dr Mark Ng
Ulster University

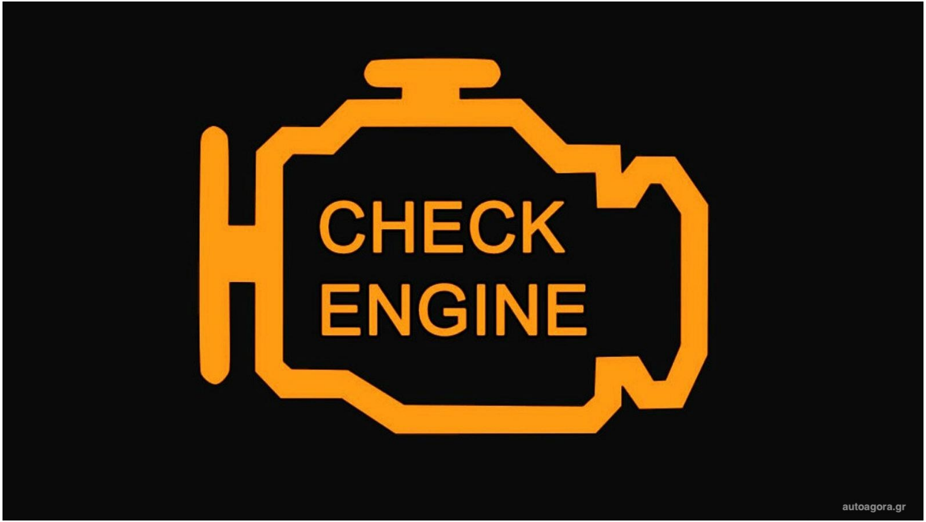
LINKÖPING UNIVERSITY

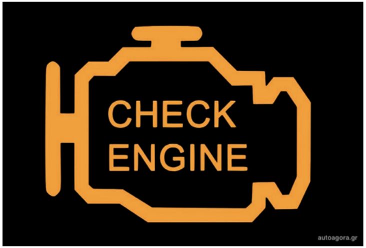
VOLVO

MONASH University

Ulster University

Introduction


CHECK
ENGINE
autoagora.gr



Catalytic Converter

Lambda Sensor

Ignition Coil Pack

Spark Plugs

.
.
.

Challenges

- Methods for root cause detection of faults
- ICEs are not clearly understood
- Diagnostic systems monitor many components independently
  - Faults manifest and trigger other monitors
  - Faults not detected in chronological order
    - Manifested faults trigger faster than root fault
- Physical injection and testing of faults
  - Shortens lifespan of engine and causes permanent damage
- WLTP

Objectives

- Improved fault isolation that vehicle is correctly repaired at first attempt
  - Determine fault isolation capability given sensor set-up
  - Analysis of which sensors needed to fulfil fault isolation requirements
  - Methods for fault detection providing good fault isolation possibilities
  - Design of a Digital Twin/Simulation Testbed to meet the above purposes

What is a Digital Twin?

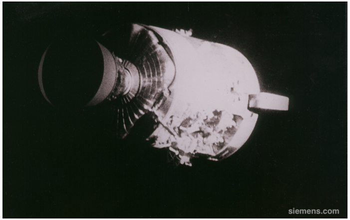
Digital Twin

- A virtual representation of an object or system that spans its lifecycle

  . . . continuously updated with data from its physical counterpart. — MIT Sloan

  . . . uses simulation, ML, and reasoning to help decision-making. — IBM

- First digital twin — Apollo 13



Digital Twin

- A virtual representation of an object or system that spans its lifecycle

  . . . c … *an*

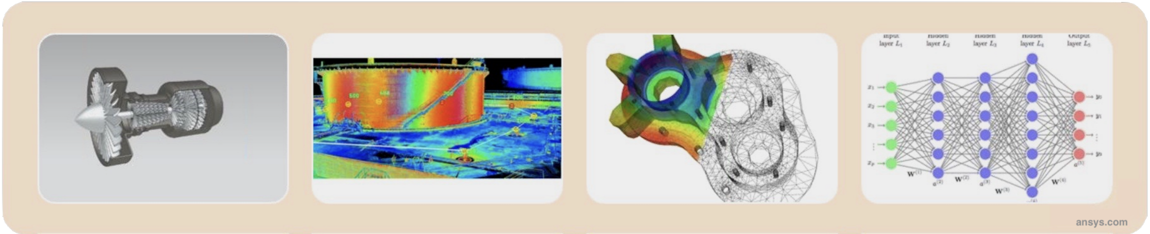
  . . . u

- Firs

What is ***NOT*** a Digital Twin?

Not Digital Twin

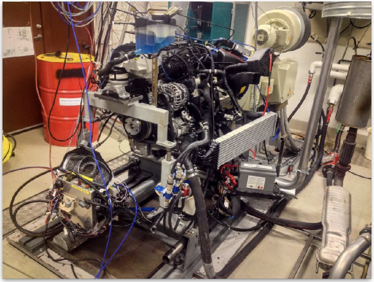
- CAD model
- 3D scan
- Physics model, e.g. 3D, 1D system
- AI/ML model

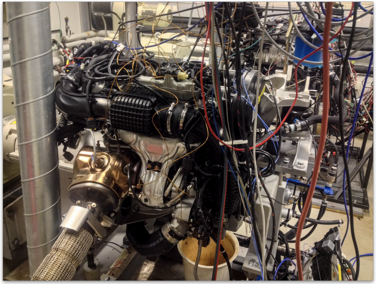


Modeling the Engine

Physical Engine on the Bench

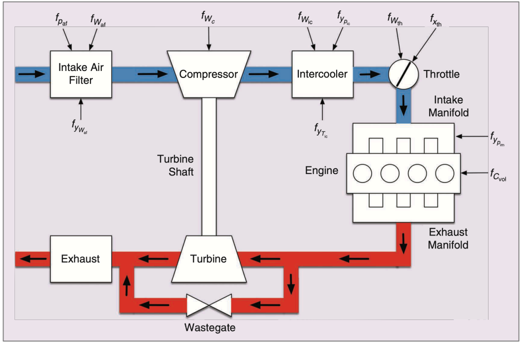
- Volvo Drive-E T5 Engine
  - 2.0L 4-cylinder turbocharged petrol
  - 8-speed automatic gearbox
  - Max Power: 245 bhp
  - Max Torque: 350 Nm
  - 0–62 mph: 6.4 s





Physical Engine on the Bench

- 6 actuators/inputs
- 9 sensors/outputs
- 13th-order system!



System states

- Air filter temperature, T_{af}
- Air filter pressure, p_{af}
- Compressor temperature, T_c
- Compressor pressure, p_c
- Intercooler temperature, T_{ic}
- Intercooler pressure, p_{ic}
- Intake manifold temperature, T_{im}
- Intake manifold pressure, p_{im}
- Exhaust manifold temperature, T_{em}
- Exhaust manifold pressure, p_{em}
- Turbine temperature, T_t
- Turbine pressure, p_t
- Turbine speed, ω_t

Actuators

- Reference engine speed, ω_{eREF}
- Control input for throttle position area, A_{th}
- Control input for wastegate, u_{wg}
- Air–fuel ratio, λ
- Ambient pressure, p_{amb}
- Ambient temperature, T_{amb}

Sensors

- Compressor temperature, T_c
- Compressor pressure, p_c
- Intercooler temperature, T_{ic}
- Intercooler pressure, p_{ic}
- Intake manifold temperature, T_{im}
- Intake manifold pressure, p_{im}
- Air filter mass flow, W_{af}
- Engine torque, Tq_e
- Exhaust manifold pressure, p_{em}

Mathematical Modeling

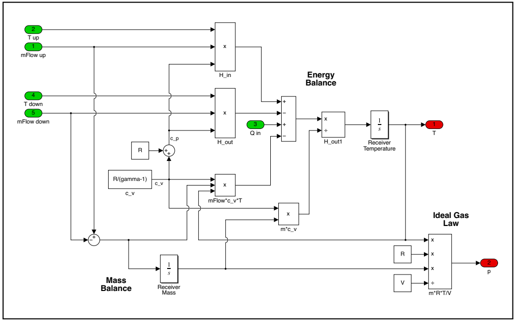
- Modeled using dynamical equations describing air flow (MVEM)
- 62 equations

$$e_1:\quad \dot{T}_{\mathrm{af}} = \frac{R_a T_{\mathrm{af}}}{p_{\mathrm{af}} V_{\mathrm{af}} c_{\mathrm{vi}}}\big((R_a + c_{\mathrm{vi}}) W_{\mathrm{af}} T_{\mathrm{af,in}} - (R_a + c_{\mathrm{vi}}) W_c T_{\mathrm{af}} - (W_{\mathrm{af}} - W_c) c_{\mathrm{vi}} T_{\mathrm{af}}\big),$$

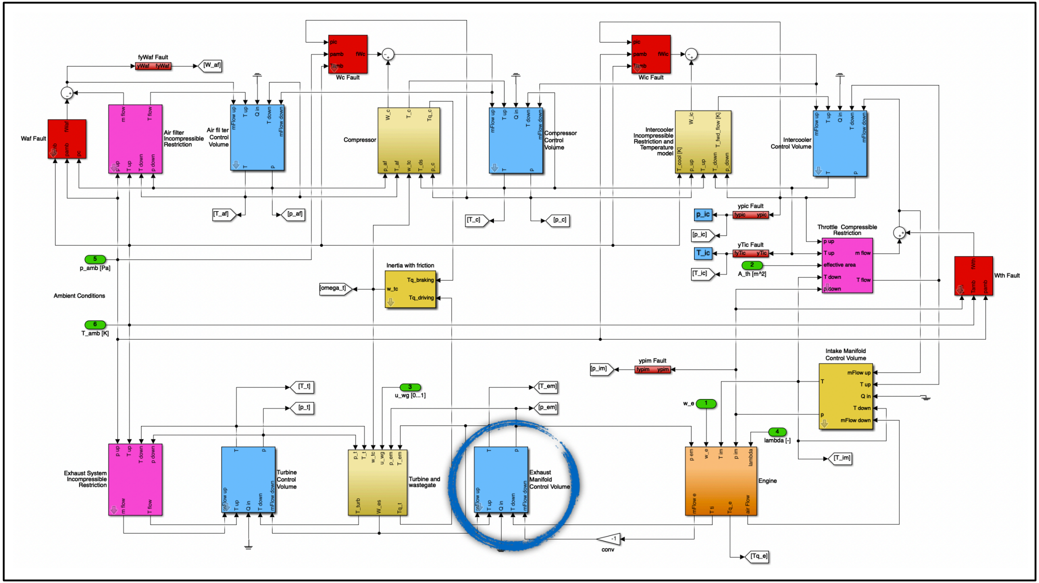
$$e_2:\quad \dot{p}_{\mathrm{af}} = \frac{R_a T_{\mathrm{af}}}{V_{\mathrm{af}}}(W_{\mathrm{af}} - W_c) + \frac{p_{\mathrm{af}}}{T_{\mathrm{af}}}\dot{T}_{\mathrm{af}},$$

$$e_3:\quad \dot{T}_c = \frac{R_a T_c}{p_c V_{\mathrm{ic}} c_{\mathrm{vi}}}\big((R_a + c_{\mathrm{vi}}) W_c T_{c,\mathrm{in}} - (R_a + c_{vi}) W_{\mathrm{ic}} T_c - (W_c - W_{\mathrm{ic}}) c_{\mathrm{vi}} T_c\big),$$

$$e_4:\quad \dot{p}_c = \frac{R_a T_c}{V_{\mathrm{ic}}}(W_c - W_{\mathrm{ic}}) + \frac{p_c}{T_c}\dot{T}_c,$$



Simulink Model



Oil & Gas Science and Technology – *Rev. IFP*, Vol. 62 (2007), No. 4, pp. 523-538

DOI: 10.2516/ogst:2007042

IFP International Conference
Rencontres Scientifiques de l'IFP

New Trends on Engine Control, Simulation and Modelling
Avancées dans le contrôle et la simulation des systèmes Groupe Moto-Propulseur

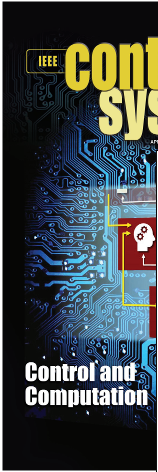
Modeling and Control of Turbocharged SI and DI Engines

L. Eriksson

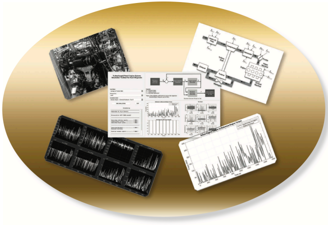
Vehicular Systems, Dept. of Electrical Engineering
Linköping University, SE-58183 Linköping - Sweden
e-mail: larer@isy.liu.se

Résumé — Modélisation et contrôle de moteurs suralimentés à allumage commandé et à injection directe — Une méthodologie pour la modélisation par composants de moteurs suralimentés est décrite et appliquée. Plusieurs modèles à composants sont considérés et évalués. De plus, de nouveaux modèles sont élaborés incluant l'efficacité du compresseur, le flux dans le compresseur, et le flux dans la turbine. Enfin, deux exemples d'application qui utilisent cette méthodologie et ces modèles de composants sont présentés. Les applications sont, d'une part, la conception d'observateurs et le contrôle du rapport air/carburant de moteurs à allumage commandé, et d'autre part la conception du contrôle de moteurs à injection directe incluant un turbocompresseur à géométrie variable et le recyclage de gaz d'échappement.

Abstract* — *Modeling and Control of Turbocharged SI and DI Engines — *A component based modeling methodology for turbocharged engines is described and applied. Several component models are compiled and reviewed. In addition new models are developed for the compressor efficiency, compressor flow, and turbine flow. Two application examples are finally given where the modeling methodology and the component models have been used. The applications are, firstly, observer design and airfuel ratio control of SI engines and, secondly, control design of DI engines with VGT and EGR.*



A Realistic Simulation Testbed of a Turbocharged Spark-Ignited Engine System



A PLATFORM FOR THE EVALUATION OF FAULT DIAGNOSIS ALGORITHMS AND STRATEGIES

KOK YEW NG, ERIK FRISK, MATTIAS KRYSANDER, and LARS ERIKSSON

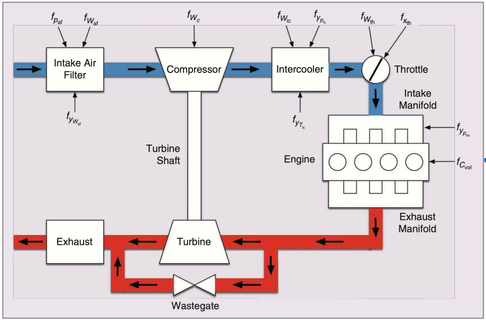
Digital Object Identifier 10.1109/MCS.2019.2961793
Date of current version: 17 March 2020

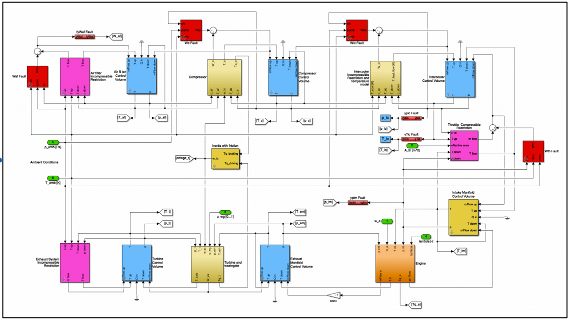
The study of fault diagnosis on automotive engine systems has been an interesting and ongoing topic for many years. Numerous research projects were conducted by automakers and research institutions to discover new and more advanced methods to perform diagnosis for better fault isolation (FI). Some of the research in this field has been reported in [1]–[5].

In most automotive systems today, the diagnostic systems monitor multiple components in the engine and are independent of each other. However,

 1066-033X/20©2020IEEE

Simulink Model

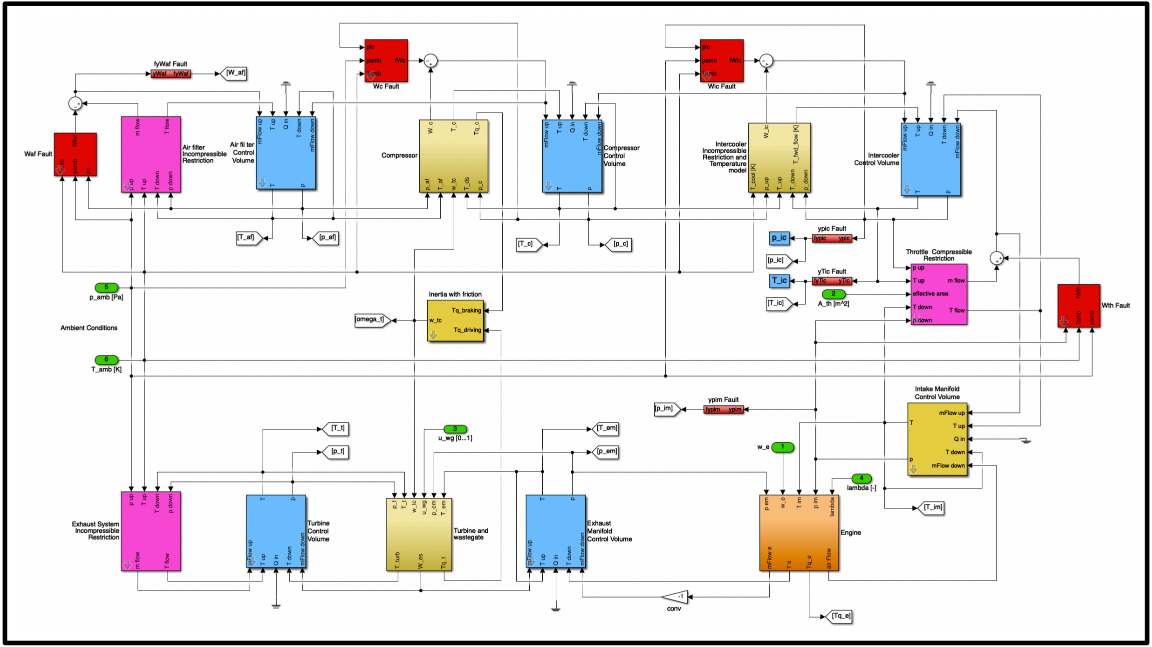




Exercise 1 — Modeling (15–20 minutes)

- Download main workshop exercise file from `markusng.com/assets/Docs/CCTA.zip` and unzip/extract it.
- Head to folder "Ex1 - Modeling".
- Ensure that "Ex1 - Modeling" is the Current Folder in Matlab.
- Run `main.m`.
  - Ensure that all engine data have been loaded to the Workspace. You can use the command '`who`' for this.
  - You will also see 2 Simulink models being opened — `Engine.slx` and `EngineLibrary.slx`.
- The engine model within `Engine.slx` (the amber box) is partly completed. With the help of the `SimulinkLibrary.slx` library file and the figure in the file `Simulink.pdf`, complete the engine model. **Ignore the red blocks!**.

IEEE CCTA
Workshop 2 Exercises


fyWaf Fault
yWaf fyWaf
[W_af]
Waf Fault
Air filter Incompressible Restriction
Air filter Control Volume
Wc Fault
Compressor
Compressor Control Volume
Wic Fault
Intercooler Incompressible Restriction and Temperature model
Intercooler Control Volume
[T_af]
[p_af]
[T_c]
[p_c]
ypic Fault
[p_ic]
yTic Fault
[T_ic]
Throttle Compressible Restriction
Wth Fault
5
p_amb [Pa]
Ambient Conditions
6
T_amb [K]
Inertia with friction
Tq_braking
w_tc
Tq_driving
[omega_t]
2
A_th [m^2]
ypim Fault
[p_im]
Intake Manifold Control Volume
[T_t]
[p_t]
3
u_wg [0...1]
[T_em]
[p_em]
w_e
1
4
lambda [-]
[T_im]
Exhaust System Incompressible Restriction
Turbine Control Volume
Turbine and wastegate
Exhaust Manifold Control Volume
Engine
conv
-1
[Tq_e]

Structural Model

Structural Model

- Analyze fault isolability from the outset without having to run simulations or perform experimentation

Structural Model

- Analyze fault isolability from the outset without having to run simulations or perform experimentation
- Provide general guideline for further development of fault diagnosis schemes

Structural Model

- Analyze fault isolability from the outset without having to run simulations or perform experimentation
- Provide general guideline for further development of fault diagnosis schemes
- Pros
  - Only a mathematical model of the system is needed
  - Simple and fast solution
  - No design of residuals required

Structural Model

- Analyze fault isolability from the outset without having to run simulations or perform experimentation
- Provide general guideline for further development of fault diagnosis schemes
- Pros
  - Only a mathematical model of the system is needed
  - Simple and fast solution
  - No design of residuals required
- Cons
  - Computed based on idealized conditions — unbounded fault scenarios
  - Performance depend entirely on quality of model
  - Actual fault isolability has to be verified and analyzed via simulations or experimentation

Structural Model

Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Control Engineering Practice 14 (2006) 597–608

CONTROL ENGINEERING PRACTICE

www.elsevier.com/locate/conengprac

Structural analysis of fault isolability in the DAMADICS benchmark

Dilek Düştegör[a], Erik Frisk[b], Vincent Cocquempot[a,*], Mattias Krysander[b], Marcel Staroswiecki[c]

[a] *LAGIS, Université des Sciences et Technologies de Lille, Bât. D, Bureau 316, École Polytechnique Universitaire de Lille, 59655 Villeneuve d'Ascq Cedex, France*
[b] *Department of Electrical Engineering Linköping University, SE-584 31 Linköping, Sweden*
[c] *École Polytechnique Universitaire de Lille, 59655 Villeneuve d'Ascq Cedex, France*

Received 15 March 2004; accepted 7 April 2005
Available online 15 June 2005

Abstract

Structural analysis is a powerful tool for early determination of fault detectability/fault isolability possibilities. It is shown how different levels of knowledge about faults can be incorporated in a structural fault isolability analysis and how they result in different isolability properties. The results are evaluated on the DAMADICS valve benchmark model. It is also shown how to determine which faults in the benchmark need further modelling to get desired isolability properties of the diagnosis system.

Keywords: Fault detection and isolation; Structural analysis; Fault modelling

Structural Analysis for Fault Isolation

- Steps to determine fault isolability using structural analysis
  1. Obtain the differential equations
  2. Construct the structural model (full-scale)
  3. Construct the reduced structural model
  4. Obtain the fault isolation matrix (FIM)

Example 1

1. Let's assume that we have a system that can be represented using the following differential equations:

$$e_1 : \dot{x}_1 = 2x_1 + f_1, \tag{1}$$
$$e_2 : \dot{x}_2 = x_2, \tag{2}$$
$$e_3 : y_1 = x_1, \tag{3}$$
$$e_4 : y_2 = x_2 + f_2, \tag{4}$$

   where $\{x_1, x_2\}$ are the states of the system, $\{y_1, y_2\}$ the measured outputs, and $\{f_1, f_2\}$ are the faults acting onto the system. The notations e_i where $i = 1, 2, 3, 4$ are used to enumerate the equations.

Example 1

2. Construct the full-scale structural model including all unmeasurable and measurable variables and faults.

   A structural model explains the relationships among the unknown variables (states), known variables (inputs and outputs), and faults in the system.

   An "x" is placed in the corresponding columns where the variables or faults are used to explain each equation in (1)–(4). For example, the state x_1 and the fault f_1 are used in equation e_1 in (1).

Table 1: Structural model for the system in (1)–(4).

	x_1	x_2	y_1	y_2	f_1	f_2
e_1	X				X	
e_2		X				
e_3	X		X			
e_4		X		X		X

Example 1

3. Construct the reduced structural model by performing canonical decomposition onto the unknown variables.

   Ensure that the table is filled up diagonally towards bottom right.

   i) Complete the equations for first state without fault
   ii) Then complete the equations for first state with fault
   iii) Repeat steps (i)–(ii), cycling through all states of the system

Table 2: Reduced structural model for the system (1)–(4).

	x_1	x_2	
e_3	X		
e_1	X		$\leftarrow f_1$
e_2		X	
e_4		X	$\leftarrow f_2$

Example 1

4. Identify now the fault(s) that could potentially affect each other on the same columns in the reduced structural model.

   If there are 2 or more faults 'sitting' on the same column, then those faults are not isolable from each other. If only 1 fault 'sits' on a single column, then that fault is completely isolated from the others.

   In Table 3, both faults f_1 and f_2 each sits on separate columns. Hence, they are isolated from one another.

Table 3: The FIM for the system (1)–(4).

	f_1	f_2
f_1	x	
f_2		x

***Note: Purely diagonal layout of filled elements in the FIM signifies a complete fault isolation performance of this system.**

Example 2

1. Let's consider a dc motor system shown in Figure 3.

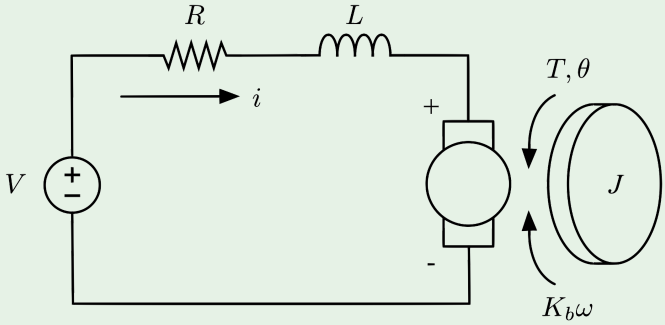


Figure 3: A generic dc motor system commonly used in control engineering studies.

Example 2

1. The system can be modeled using the following equations:

$$e_1 : V = i(R+f_R) + L\frac{di}{dt} + K_a i\omega, \quad (5)$$

$$e_2 : T_m = K_a i^2, \quad (6)$$

$$e_3 : J\frac{d\omega}{dt} = \Delta T - K_b\omega, \quad (7)$$

$$e_4 : \Delta T = T_m - T_L, \quad (8)$$

$$e_5 : \frac{d\theta}{dt} = \omega, \quad (9)$$

$$e_6 : \frac{d\omega}{dt} = \alpha, \quad (10)$$

$$e_7 : y_i = i + f_i, \quad (11)$$

$$e_8 : y_\omega = \omega + f_\omega, \quad (12)$$

$$e_9 : y_\Delta = \Delta T + f_\Delta, \quad (13)$$

where $\{i, \theta, \omega, \alpha, T_m, T_L, \Delta T\}$ are the states of the system, and $\{i, \omega, \Delta T\}$ are the measurable outputs. The input is the voltage V.

Assume that there is a system fault f_R and that all outputs are potentially faulty, i.e. $\{f_i, f_\omega, f_\Delta\}$.

Example 2

2. Construct the full-scale structural model.

Table 4: Structural model for the dc motor system.

	i	θ	ω	α	T_m	T_L	ΔT	V	y_i	y_ω	y_Δ	f_R	f_i	f_ω	f_Δ
e_1	X		X					X				X			
e_2	X				X										
e_3			X				X								
e_4					X	X	X								
e_5		X	X												
e_6			X	X											
e_7	X								X				X		
e_8			X							X				X	
e_9							X				X				X

Example 2

3. Performing canonical decomposition onto the unknown variables yields

Table 5: Reduced structural model for the dc motor system.

	θ	α	T_L	T_m	i	ΔT	ω
e_5	X						X
e_6		X					X
e_4			X	X		X	
e_2				X	X		
e_1				$f_R \longrightarrow$	X		X
e_7				$f_i \longrightarrow$	X		
e_3						X	X
e_9				$f_\Delta \longrightarrow$		X	
e_8				$f_\omega \longrightarrow$			X

Example 2

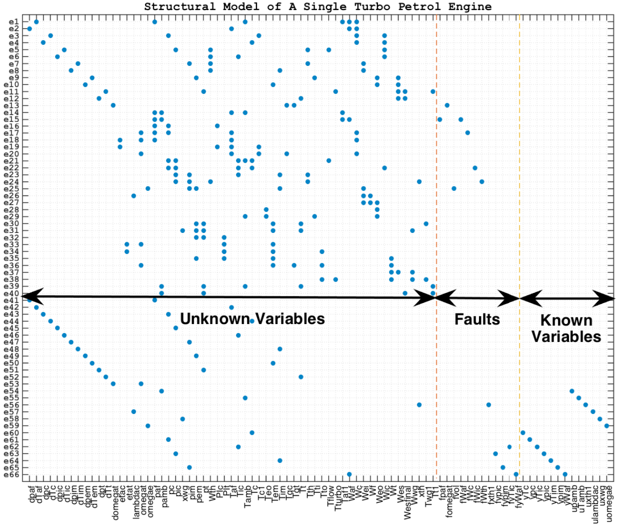
4. The FIM shows that the pair $\{f_R, f_i\}$ are not isolable from each other.

Table 6: The FIM for the dc motor system.

	f_R	f_i	f_Δ	f_ω
f_R	X	X		
f_i	X	X		
f_Δ			X	
f_ω				X

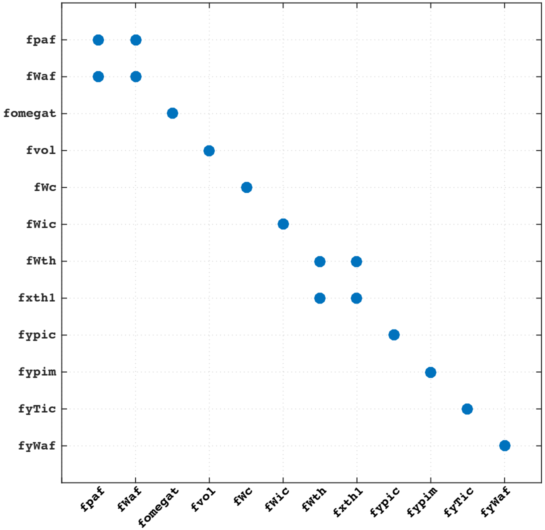
The Engine System

Structural model



The Engine System

FIM



Structural Model

- Analyze fault isolability from the outset without having to run simulations or perform experimentation
- Provide general guideline for further development of fault diagnosis schemes
- Pros
  - Only a mathematical model of the system is needed
  - Simple and fast solution
  - No design of residuals required
- **Cons**
  - **Computed based on idealized conditions — unbounded fault scenarios**
  - **Performance depend entirely on quality of model**
  - **Actual fault isolability has to be verified and analyzed via simulations or experimentation**

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 3287–3293

A Toolbox for Analysis and Design of Model Based Diagnosis Systems for Large Scale Models

Erik Frisk, Mattias Krysander, and Daniel Jung

Department of Electrical Engineering, Linköping University, Sweden
{erik.frisk, mattias.krysander, daniel.jung}@liu.se

Abstract: To facilitate the use of advanced fault diagnosis analysis and design techniques to industrial sized systems, there is a need for computer support. This paper describes a Matlab toolbox and evaluates the software on a challenging industrial problem, air-path diagnosis in an automotive engine. The toolbox includes tools for analysis and design of model based diagnosis systems for large-scale differential algebraic models. The software package supports a complete tool-chain from modeling a system to generating C-code for residual generators. Major design steps supported by the tool are modeling, fault diagnosability analysis, sensor selection, residual generator analysis, test selection, and code generation. Structural methods based on efficient graph theoretical algorithms are used in several steps. In the automotive diagnosis example, a diagnosis system is generated and evaluated using measurement data, both in fault-free operation and with faults injected in the control-loop. The results clearly show the benefit of the toolbox in a model-based design of a diagnosis system. Latest version of the toolbox can be downloaded at `faultdiagnosistoolbox.github.io`.

Keywords: Fault diagnosis, software tool, toolbox, Matlab, automotive engine

Exercise 2 — Structural Model (15–20 minutes)

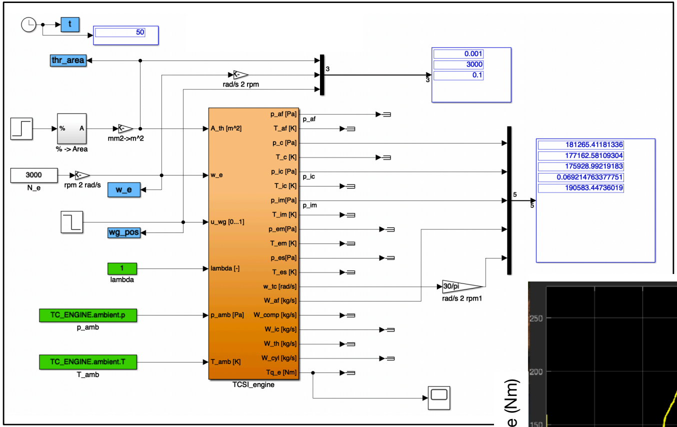
- Still using the unzipped/extracted main workshop exercise file downloaded for Exercise 1.
- Head to folder “Ex2 - Structural Model”.
- Ensure that “Ex2 - Structural Model” is the Current Folder in Matlab.
- Open/edit `main.m`.
- The structural model of the engine has been partly completed (lines 32–71).
- This exercise requires you to complete the definitions for the derivatives (lines 73–85), actuators (line 87 and expand accordingly), and sensors (line 89 and expand accordingly).
- You can refer to pages 49–52 of the file ‘`IEEECSM (arXiv).pdf`’ (preprint of the *IEEE CSM* paper) found in the same folder to help with completing the equations.
- Once you are done, run `main.m`

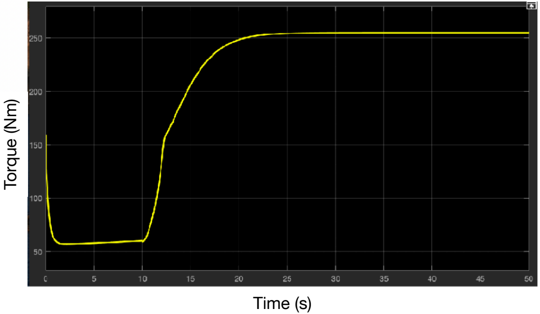
Controller Design

Controller Design

- To follow reference
- Driving Cycle Profiles
- Effect of faults in closed-loop
- Realistic excitation of engine

Open Loop Analysis





Reference Inputs

$$v_{g,1000rpm} = \frac{120\pi r_w}{\text{final gear ratio} \times \text{current gear ratio}},$$

$$\omega_{eREF} = \frac{V i_{gear}(V)}{r_w},$$

$$m_v \dot{V} = F_w - F_d - F_r,$$

$$F_d = \frac{1}{2}\rho_a c_d A_f V^2,$$

$$F_r = m_v c_r g,$$

$$Tq_{eREF} = \frac{Tq_w}{i_{gear}(V)}.$$

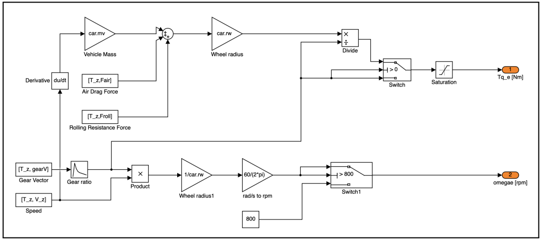
TABLE S1 The key vehicle parameters.

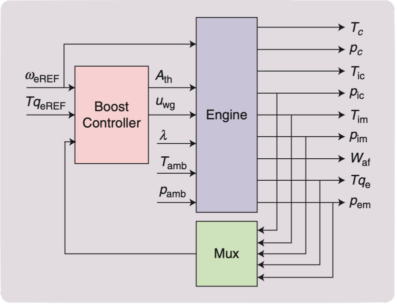
Description	Value	Unit
General vehicle parameters		
Mass, m_v	1700	kg
Drag coefficient, c_d	0.29	[—]
Roll coefficient, c_r	0.013	[—]
Frontal area, A_f	2.28	m^2
Wheel radius, r_w	0.3234	m
(assuming tires rated 215/50R17)		
Gear ratios		
First	5.25	[—]
Second	3.029	[—]
Third	1.95	[—]
Fourth	1.457	[—]
Fifth	1.221	[—]
Sixth	1	[—]
Seventh	0.809	[—]
Eighth	0.673	[—]
Reverse	4.015	[—]
Final drive	2.774	[—]

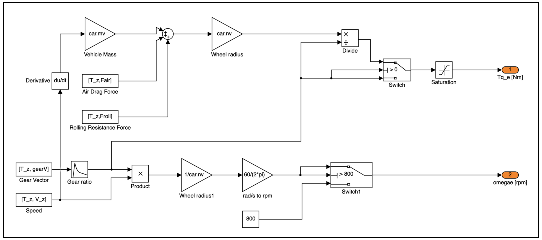
*Speed at 1000 r/min in eighth gear is 62.9 km/h

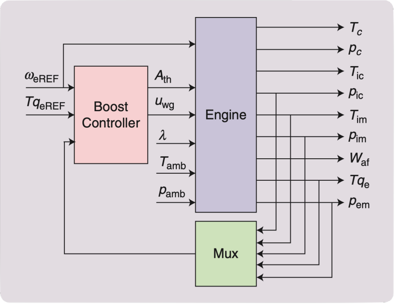
TABLE S2 The gearbox's estimated shift points.

Gear	Vehicle Speed (km/h) per 1000 r/min	Engine Shifting Speed (r/min)
First	8.07	2800
Second	14	2700
Third	21.7	2600
Fourth	29	2400
Fifth	34.7	2200
Sixth	42.3	2000
Seventh	52.34	1800
Eighth	62.9	1600


car.mv
Vehicle Mass
car.rw
Wheel radius
Divide
> 0
Switch
Saturation
1
Tq_e [Nm]
Derivative
du/dt
[T_z,Fair]
Air Drag Force
[T_z,Froll]
Rolling Resistance Force
[T_z, gearV]
Gear Vector
Gear ratio
Product
1/car.rw
Wheel radius1
60/(2*pi)
rad/s to rpm
> 800
Switch1
2
omegae [rpm]
[T_z, V_z]
Speed
800

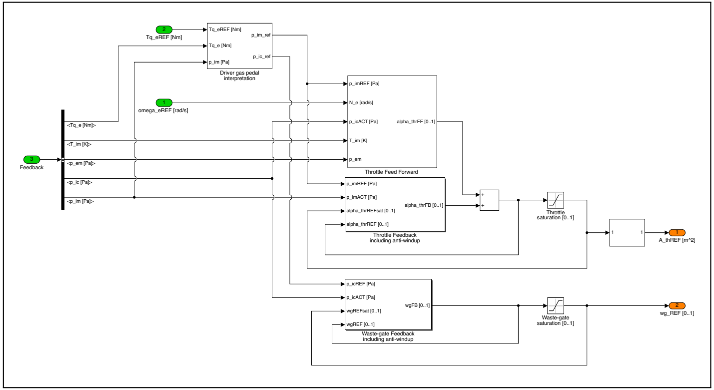

ω_{eREF}
Tq_{eREF}
Boost Controller
A_{th}
u_{wg}
λ
T_{amb}
p_{amb}
Engine
Mux
T_c
p_c
T_{ic}
p_{ic}
T_{im}
p_{im}
W_{af}
Tq_e
p_{em}



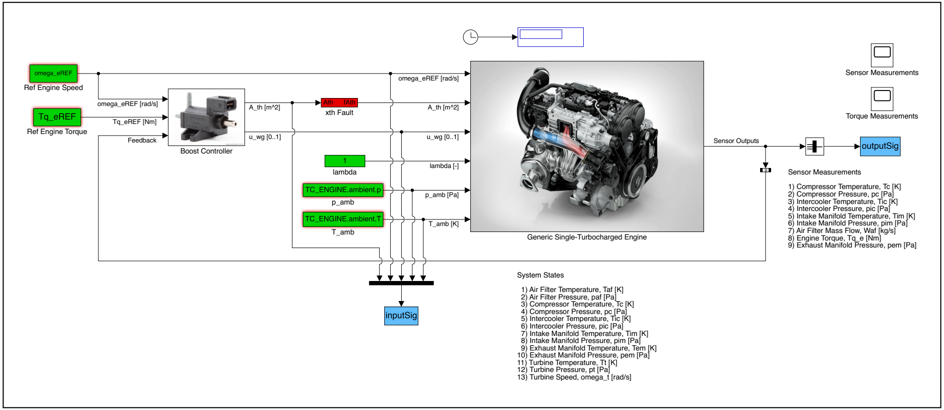


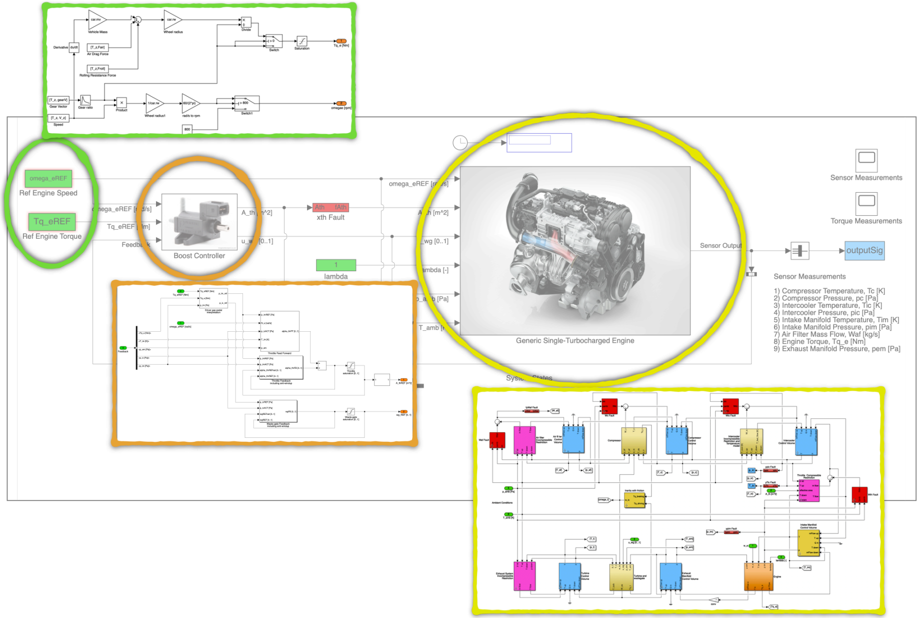
WLTP Driving Cycle

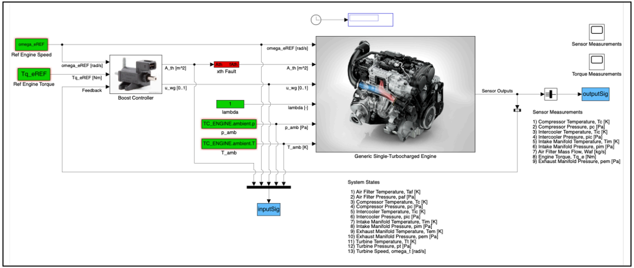
Controller Setup



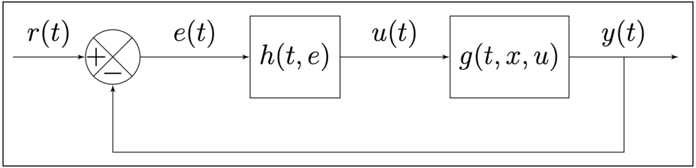
- PI-based with anti-windup
- Control for throttle area and wastegate actuator


omega_eREF
Ref Engine Speed
Tq_eREF
Ref Engine Torque
omega_eREF [rad/s]
Tq_eREF [Nm]
Feedback
Boost Controller
A_th [m^2]
u_wg [0..1]
Ath fAth
xth Fault
1
lambda
TC_ENGINE.ambient.p
p_amb
TC_ENGINE.ambient.T
T_amb
omega_eREF [rad/s]
A_th [m^2]
u_wg [0..1]
lambda [-]
p_amb [Pa]
T_amb [K]
Generic Single-Turbocharged Engine
Sensor Outputs
outputSig
inputSig
Sensor Measurements
Torque Measurements
Sensor Measurements
1) Compressor Temperature, Tc [K]
2) Compressor Pressure, pc [Pa]
3) Intercooler Temperature, Tic [K]
4) Intercooler Pressure, pic [Pa]
5) Intake Manifold Temperature, Tim [K]
6) Intake Manifold Pressure, pim [Pa]
7) Air Filter Mass Flow, Waf [kg/s]
8) Engine Torque, Tq_e [Nm]
9) Exhaust Manifold Pressure, pem [Pa]
System States
1) Air Filter Temperature, Taf [K]
2) Air Filter Pressure, paf [Pa]
3) Compressor Temperature, Tc [K]
4) Compressor Pressure, pc [Pa]
5) Intercooler Temperature, Tic [K]
6) Intercooler Pressure, pic [Pa]
7) Intake Manifold Temperature, Tim [K]
8) Intake Manifold Pressure, pim [Pa]
9) Exhaust Manifold Temperature, Tem [K]
10) Exhaust Manifold Pressure, pem [Pa]
11) Turbine Temperature, Tt [K]
12) Turbine Pressure, pt [Pa]
13) Turbine Speed, omega_t [rad/s]

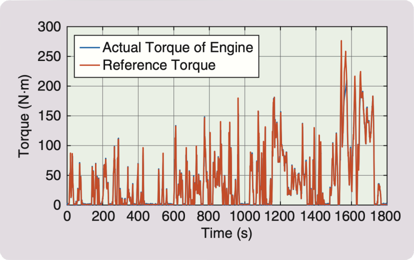

omega_eREF
Ref Engine Speed
Tq_eREF
Ref Engine Torque
Boost Controller
xth Fault
lambda
Generic Single-Turbocharged Engine
Sensor Output
outputSig
Sensor Measurements
Torque Measurements
Sensor Measurements
1) Compressor Temperature, Tc [K]
2) Compressor Pressure, pc [Pa]
3) Intercooler Temperature, Tic [K]
4) Intercooler Pressure, pic [Pa]
5) Intake Manifold Temperature, Tim [K]
6) Intake Manifold Pressure, pim [Pa]
7) Air Filter Mass Flow, Waf [kg/s]
8) Engine Torque, Tq_e [Nm]
9) Exhaust Manifold Pressure, pem [Pa]



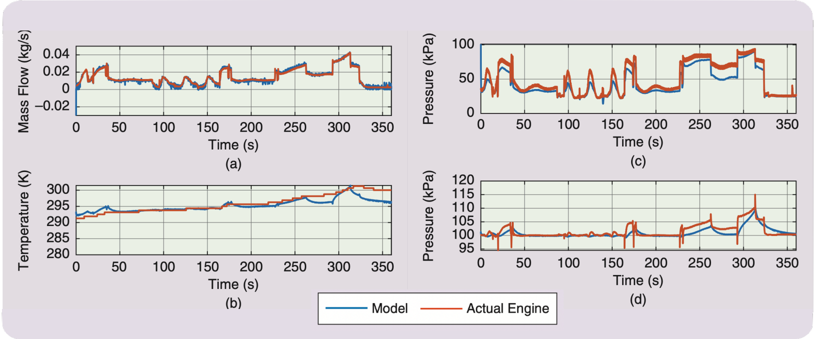
omega_eREF
Ref Engine Speed
Tq_eREF
Ref Engine Torque
omega_eREF [rad/s]
Tq_eREF [Nm]
Feedback
Boost Controller
A_th [m^2]
u_wg [0..1]
xth Fault
1
lambda
TC_ENGINE.ambient.p
p_amb
TC_ENGINE.ambient.T
T_amb
omega_eREF [rad/s]
A_th [m^2]
u_wg [0..1]
lambda [-]
p_amb [Pa]
T_amb [K]
Generic Single-Turbocharged Engine
Sensor Outputs
outputSig
Sensor Measurements
Torque Measurements
inputSig
Sensor Measurements
1) Compressor Temperature, Tc [K]
2) Compressor Pressure, pc [Pa]
3) Intercooler Temperature, Tic [K]
4) Intercooler Pressure, pic [Pa]
5) Intake Manifold Temperature, Tim [K]
6) Intake Manifold Pressure, pim [Pa]
7) Air Filter Mass Flow, Waf [kg/s]
8) Engine Torque, Tq_e [Nm]
9) Exhaust Manifold Pressure, pem [Pa]
System States
1) Air Filter Temperature, Taf [K]
2) Air Filter Pressure, paf [Pa]
3) Compressor Temperature, Tc [K]
4) Compressor Pressure, pc [Pa]
5) Intercooler Temperature, Tic [K]
6) Intercooler Pressure, pic [Pa]
7) Intake Manifold Temperature, Tim [K]
8) Intake Manifold Pressure, pim [Pa]
9) Exhaust Manifold Temperature, Tem [K]
10) Exhaust Manifold Pressure, pem [Pa]
11) Turbine Temperature, Tt [K]
12) Turbine Pressure, pt [Pa]
13) Turbine Speed, omega_t [rad/s]



$r(t)$
+
−
$e(t)$
$h(t, e)$
$u(t)$
$g(t, x, u)$
$y(t)$



Actual Torque of Engine
Reference Torque
Torque (N·m)
300
250
200
150
100
50
0
0 200 400 600 800 1000 1200 1400 1600 1800
Time (s)



Mass Flow (kg/s)
0.04
0.02
0
–0.02
0 50 100 150 200 250 300 350
Time (s)
(a)
Pressure (kPa)
100
50
0
0 50 100 150 200 250 300 350
Time (s)
(c)
Temperature (K)
300
295
290
285
280
0 50 100 150 200 250 300 350
Time (s)
(b)
Pressure (kPa)
120
115
110
105
100
95
0 50 100 150 200 250 300 350
Time (s)
(d)
Model
Actual Engine

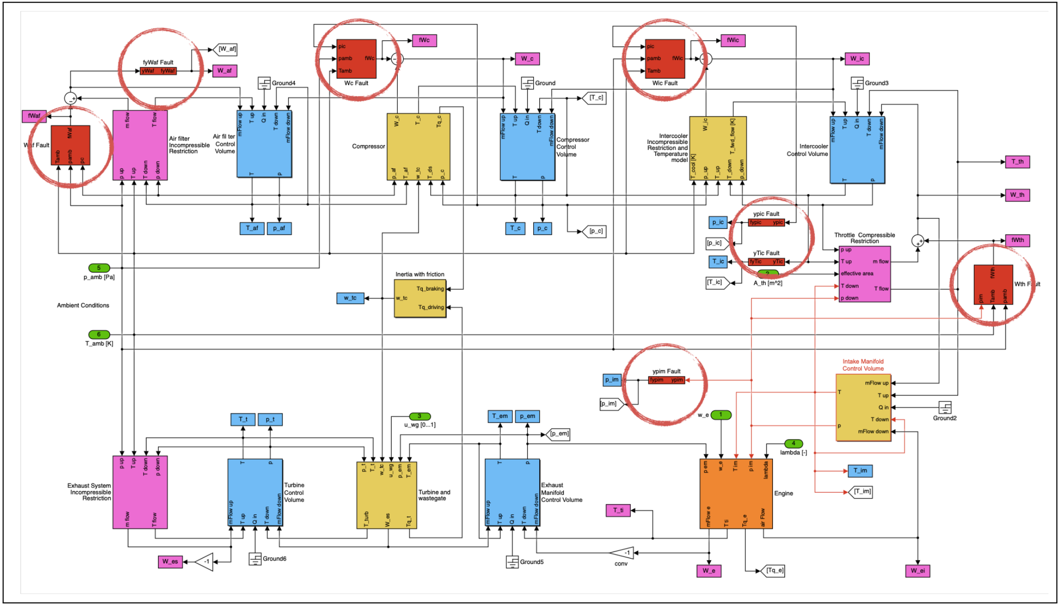
Exercise 3 — Boost Controller (15–20 minutes)

- Still using the unzipped/extracted main workshop exercise file downloaded for Exercise 1.
- Head to folder “Ex3 - Controller”.
- Ensure that “Ex3 - Controller” is the Current Folder in Matlab.
- Run `main.m`.
- You will see 2 Simulink models being opened — `EngineController.slx` and `EngineLibrary.slx`.
- In `EngineController.slx`, click on the Boost Controller block and complete the controller with the help of the figure in `Controller.pdf` and the template blocks in `SimulinkLibrary.slx`.
- Simulate `EngineController.slx` to verify the performance of the boost controller.

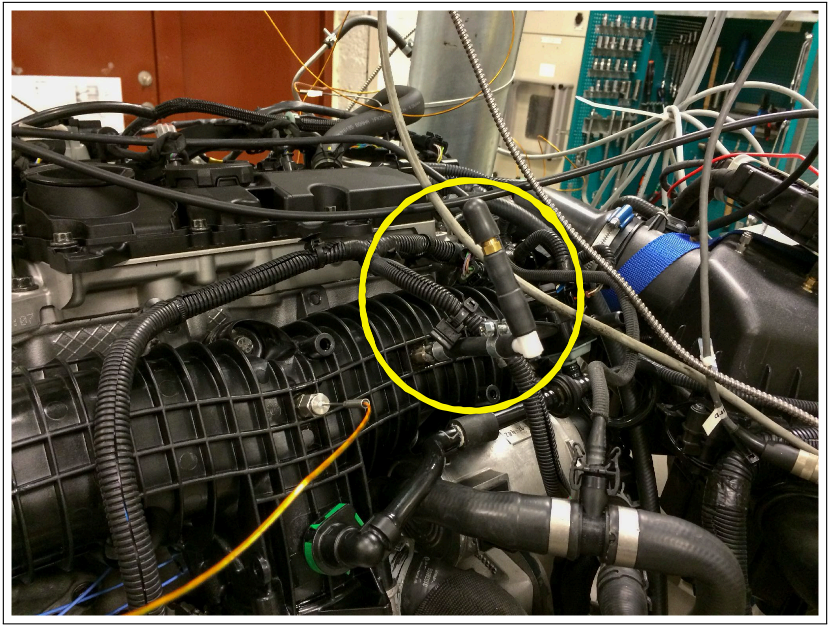
Faults, Classification, and Types of Faults

TABLE 2 **The descriptions of the fault types. T_{DC} represents the duration of the driving cycle: Worldwide Harmonized Light Vehicle Test Procedure (1800 s), New European Driving Cycle (1220 s), Extra-Urban Driving Cycle (400 s), and U.S. Environmental Protection Agency Federal Test Procedure (1874 s).**

Fault	Description	Fault Threshold	Nature of Fault (Active Period)	Severity
$f_{p_{af}}$	Loss of pressure in the air filter	20-kPa pressure drop	Abrupt (from 200 s until T_{DC})	Medium
$f_{C_{vol}}$	Intake-valve timing stuck at an arbitrary position	Stuck at end or middle position	Abrupt pulses (active for 30 s every 150 s)	High
$f_{W_{af}}$	Air leakage between the air filter and the compressor	20% of flow through leakage	Incipient (from 200 s until T_{DC})	Medium
f_{W_c}	Air leakage between the compressor and the intercooler	20% of flow through leakage	Abrupt (from $0.4T_{DC}$ until T_{DC})	High
$f_{W_{ic}}$	Air leakage between the intercooler and the throttle	20% of flow through leakage	Abrupt (from $0.4T_{DC}$ until $0.8T_{DC}$)	High
$f_{W_{th}}$	Air leakage after the throttle in the intake manifold	20% of flow through leakage	Abrupt pulses (active for 40 s every 200 s)	High
$f_{x_{th}}$	Throttle position actuator error	Fault leading to 20% flow error	Abrupt (from $0.4T_{DC}$ until T_{DC})	Medium
$f_{y_{W_{af}}}$	Air filter flow sensor fault	20% flow error	Abrupt pulses (active for 30 s every 150 s)	Low
$f_{y_{p_{im}}}$	Intake manifold pressure sensor fault	20% pressure deviation	Incipient (from 200 s until T_{DC})	Low
$f_{y_{p_{ic}}}$	Intercooler pressure sensor fault	20% pressure deviation	Abrupt pulses (active for 40 s every 200 s)	Low
$f_{y_{T_{ic}}}$	Intercooler temperature sensor fault	20-K offset	Abrupt pulses (active for 30 s every 150 s)	Low



fyWaf Fault
[W_af]
W_af
fWc
Wc Fault
W_c
Ground4
Ground
[T_c]
fWic
Wic Fault
W_ic
Ground3
fWaf
Waf Fault
Air filter
Incompressible
Restriction
Air filter
Control
Volume
Compressor
Compressor
Control
Volume
Intercooler
Incompressible
Restriction and
Temperature
model
Intercooler
Control Volume
T_th
W_th
T_af
p_af
T_c
p_c
[p_c]
ypic Fault
p_ic
[p_ic]
yTic Fault
T_ic
[T_ic]
A_th [m^2]
Throttle Compressible
Restriction
fWth
Wth Fault
p_amb [Pa]
Inertia with friction
w_tc
Ambient Conditions
T_amb [K]
Intake Manifold
Control Volume
ypim Fault
p_im
[p_im]
w_e
Ground2
T_t
p_t
u_wg [0...1]
T_em
p_em
[p_em]
lambda [-]
Exhaust System
Incompressible
Restriction
Turbine
Control
Volume
Turbine and
wastegate
Exhaust
Manifold
Control Volume
Engine
T_im
[T_im]
T_ti
W_es
Ground6
Ground5
conv
W_e
[Tq_e]
W_ei



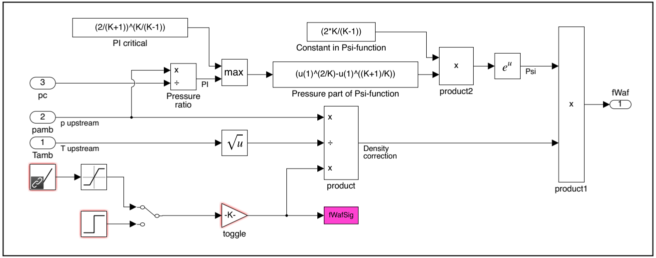
Mass Flow Fault Defined by Size of Leakage Area

- Leakage at specific part of engine system
- Area of leakage (mm^2) represents severity of fault

$$W_{leak} = k_{leak} \frac{p_{high}}{\sqrt{T_{amb}}} \Psi\left(\frac{p_{low}}{p_{high}}\right),$$

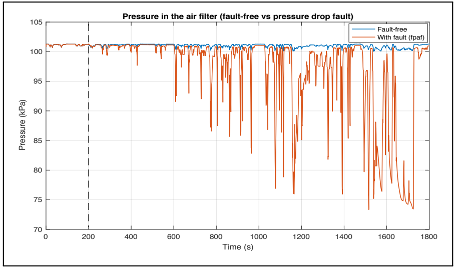
where

$$\Psi\left(\frac{p_{low}}{p_{high}}\right) = \begin{cases} \sqrt{\frac{2\kappa}{\kappa-1}\left\{\left(\frac{p_{low}}{p_{high}}\right)^{2/\kappa} - \left(\frac{p_{high}}{p_{low}}\right)^{(\kappa+1)/\kappa}\right\}}, \\ \qquad \text{if } \left(\frac{p_{low}}{p_{high}}\right) \geq \left(\frac{2}{\kappa+1}\right)^{\kappa/(\kappa-1)}, \\ \sqrt{\kappa\left(\frac{2}{\kappa+1}\right)^{(\kappa+1)/(\kappa-1)}}, \quad \text{otherwise.} \end{cases}$$

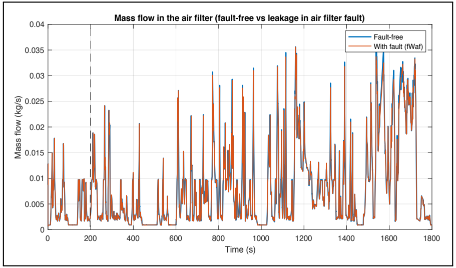


Visualizing Faults and Their Effects on Engine System

- Pressure drop in air filter, $f_{p_{af}}$
- Incipient fault induced at 200 s
- Effects are prominent and easy to detect



- Leakage in the air filter, $f_{W_{af}}$
- Incipient fault induced at 200 s
- Effects are less prominent and difficult to detect



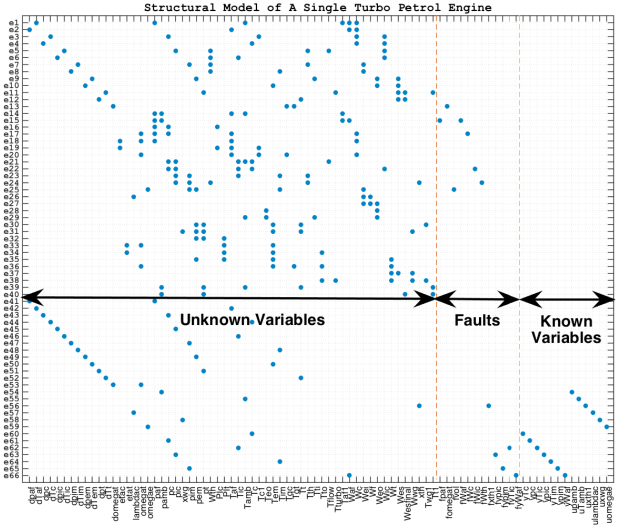
Generation of Residuals

Objectives

- Improved fault isolation that vehicle is correctly repaired at first attempt
  - Determine fault isolation capability given sensor set-up
  - Analysis where fault modes can be isolated with current sensor set-up
  - Analysis of which sensors needed to fulfil fault isolation requirements
  - Methods for fault detection providing good fault isolation possibilities
  - Design of a Digital Twin/Simulation Testbed to meet the above purposes

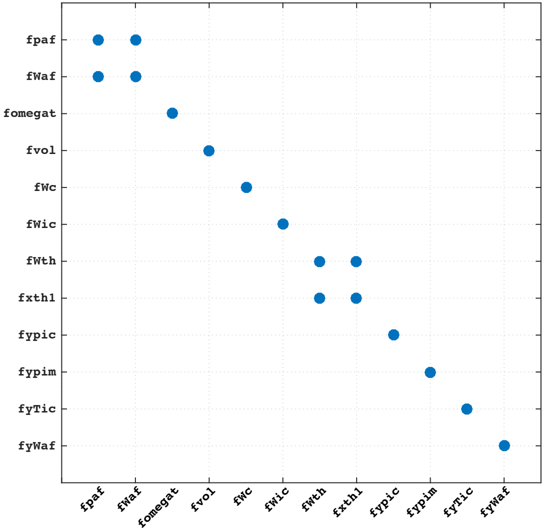
Initial Fault Isolation Analysis

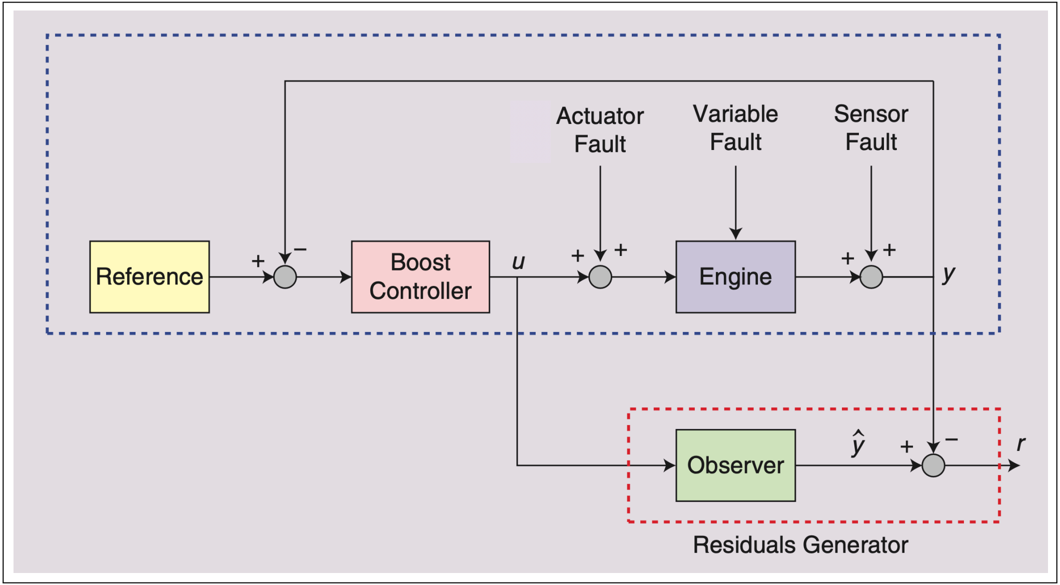
Structural model



Initial Fault Isolation Analysis

FIM





Reference
Boost Controller
Actuator Fault
Variable Fault
Sensor Fault
Engine
Observer
u
y
ŷ
r
+
−
Residuals Generator

Generation of Residuals

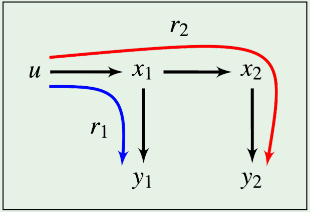
Theoretical Background

Assume the system

$$
\begin{aligned}
\dot{x}_1 &= -x_1 + u + f_u,\\
\dot{x}_2 &= x_1 - x_2,\\
y_1 &= x_1 + f_1,\\
y_2 &= x_2 + f_2.
\end{aligned}
$$

The following residuals can be generated:

$$
\begin{aligned}
r_1 &= \hat{y}_1 - y_1,\\
r_2 &= \hat{y}_2 - y_2.
\end{aligned}
$$



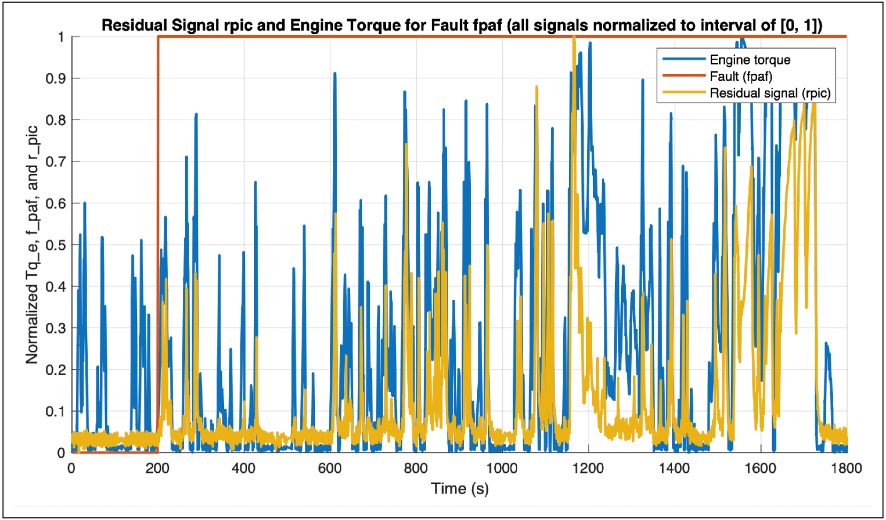
Engine System

TABLE 3 **The default residuals ("Original 9") for fault detection given the sensor setup in Table 1.**

Residual	Description
r_{T_c}	Residual for the compressor temperature sensor
r_{p_c}	Residual for the compressor pressure sensor
$r_{T_{ic}}$	Residual for the intercooler temperature sensor
$r_{p_{ic}}$	Residual for the intercooler pressure sensor
$r_{T_{im}}$	Residual for the intake manifold temperature sensor
$r_{p_{im}}$	Residual for the intake manifold pressure sensor
$r_{W_{af}}$	Residual for the air filter mass flow sensor
r_{Tq_e}	Residual for the engine torque sensor
$r_{p_{em}}$	Residual for the exhaust manifold pressure sensor

Simulation Results

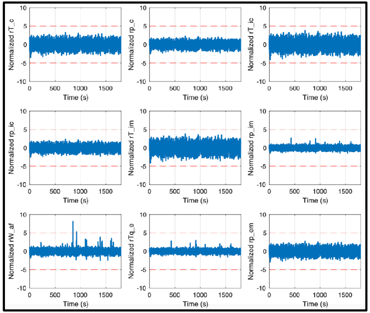
Effects of Engine Dynamics on Residuals



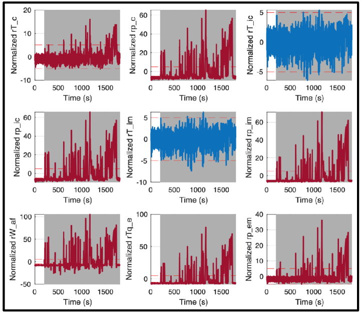
Suggested Fault Detection Requirements

- **Time for fault detection**: $\boxed{|r_i| > J \;\; \& \;\; t_f > 3\,\mathrm{s}}$
- **Missed detections**: Testbed is designed so that amplitudes of faults are large enough that they should be detected.

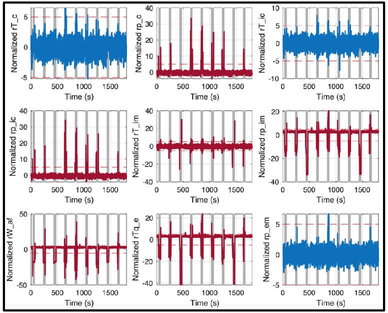
Simulation Results



Fault-free



With fault f_{paf}



With fault f_{Wth}

Non-triggered
Triggered

Simulation Results

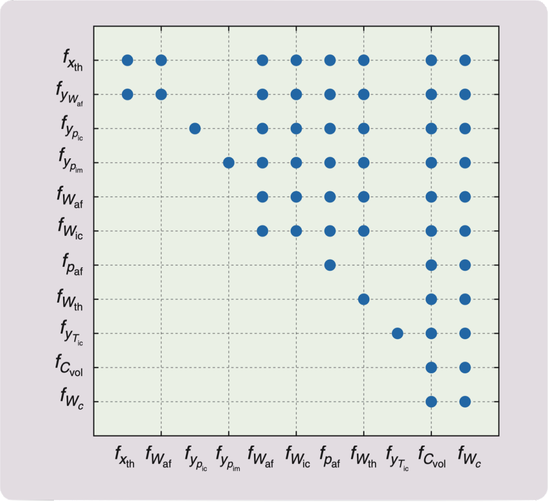
TABLE 4 The fault sensitivity matrix of the "Original 9" residuals.

Residual	$f_{p_{af}}$	$f_{C_{vol}}$	$f_{W_{af}}$	f_{W_c}	$f_{W_{ic}}$	$f_{W_{th}}$	$f_{x_{th}}$	$f_{y_{p_{ic}}}$	$f_{y_{p_{im}}}$	$f_{y_{T_{ic}}}$	$f_{y_{W_{af}}}$
r_{T_c}	1	1	0	1	0	0	0	0	0	0	0
r_{p_c}	1	1	1	1	1	1	0	0	0	0	0
$r_{T_{ic}}$	0	1	0	1	0	0	0	0	0	1	0
$r_{p_{ic}}$	1	1	1	1	1	1	0	1	0	0	0
$r_{T_{im}}$	0	1	0	1	0	1	0	0	0	0	0
$r_{p_{im}}$	1	1	1	1	1	1	0	0	1	0	0
$r_{W_{af}}$	1	1	1	1	1	1	1	0	0	0	1
r_{Tq_e}	1	1	1	1	1	1	0	0	0	0	0
$r_{p_{em}}$	1	1	0	1	0	0	0	0	0	0	0

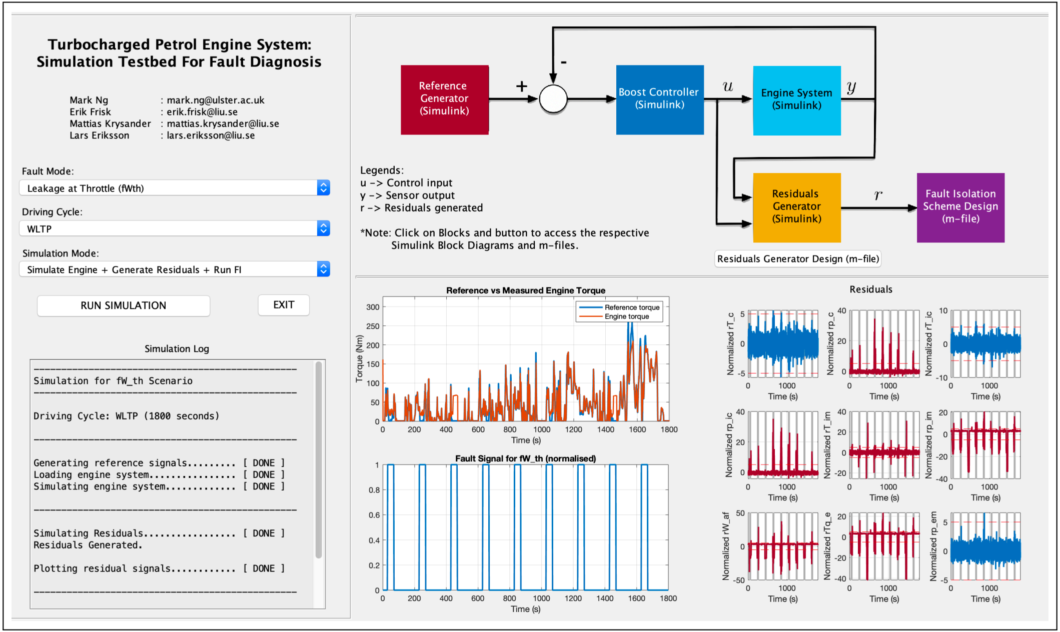
Simulation Results

TABLE 4 **The fault sensitivity matrix of the "Original 9" residuals.**

Residual	$f_{p_{af}}$	$f_{C_{vol}}$	$f_{W_{af}}$	f_{W_c}	$f_{W_{ic}}$	$f_{W_{th}}$	$f_{x_{th}}$	$f_{y_{p_{ic}}}$	$f_{y_{p_{im}}}$	$f_{y_{T_{ic}}}$	$f_{y_{W_{af}}}$
r_{T_c}	1	1	0	1	0	0	0	0	0	0	0
r_{p_c}	1	1	1	1	1	1	0	0	0	0	0
$r_{T_{ic}}$	0	1	0	1	0	0	0	0	0	1	0
$r_{p_{ic}}$	1	1	1	1	1	1	0	1	0	0	0
$r_{T_{im}}$	0	1	0	1	0	1	0	0	0	0	0
$r_{p_{im}}$	1	1	1	1	1	1	0	0	1	0	0
$r_{W_{af}}$	1	1	1	1	1	1	1	0	0	0	1
r_{Tq_e}	1	1	1	1	1	1	0	0	0	0	0
$r_{p_{em}}$	1	1	0	1	0	0	0	0	0	0	0



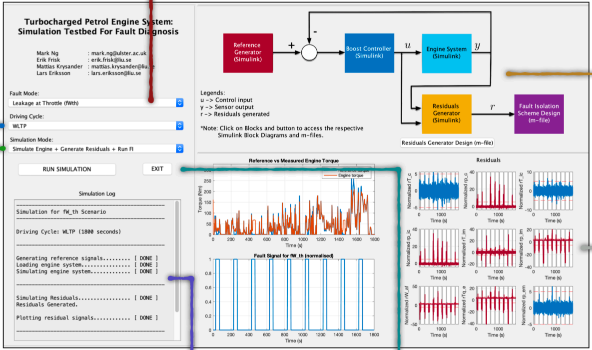
The Simulation Testbed


Turbocharged Petrol Engine System:
Simulation Testbed For Fault Diagnosis
Mark Ng : mark.ng@ulster.ac.uk
Erik Frisk : erik.frisk@liu.se
Mattias Krysander : mattias.krysander@liu.se
Lars Eriksson : lars.eriksson@liu.se
Fault Mode:
Leakage at Throttle (fWth)
Driving Cycle:
WLTP
Simulation Mode:
Simulate Engine + Generate Residuals + Run FI
RUN SIMULATION
EXIT
Simulation Log
Simulation for fW_th Scenario
Driving Cycle: WLTP (1800 seconds)
Generating reference signals......... [DONE]
Loading engine system................ [DONE]
Simulating engine system............. [DONE]
Simulating Residuals................. [DONE]
Residuals Generated.
Plotting residual signals............ [DONE]
Reference Generator (Simulink)
Boost Controller (Simulink)
Engine System (Simulink)
Residuals Generator (Simulink)
Fault Isolation Scheme Design (m-file)
u
y
r
Legends:
u -> Control input
y -> Sensor output
r -> Residuals generated
*Note: Click on Blocks and button to access the respective Simulink Block Diagrams and m-files.
Residuals Generator Design (m-file)
Reference vs Measured Engine Torque
Reference torque
Engine torque
Torque (Nm)
Time (s)
Fault Signal for fW_th (normalised)
Time (s)
Residuals
Normalized rT_c
Normalized rp_c
Normalized rT_ic
Normalized rp_ic
Normalized rT_im
Normalized rp_im
Normalized rW_af
Normalized rTq_e
Normalized rp_em
Time (s)

Driving Cycle Profile:
- WLTP
- NEDC
- EUDC
- FTP-75

Fault Mode:
- Fault-free
- 11 faults

Blocks to access M-files and Simulink Models:
- Reference signals generator (Simulink)
- Controller and engine model (Simulink)
- Residuals generator (m-file and Simulink)
- Fault diagnosis scheme design (m-file)



Simulation Mode:
- Simulate engine only
- Simulate engine with residuals generation and FD schemes

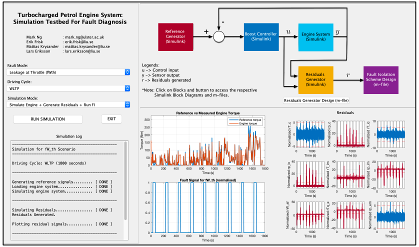
Simulation log to show progress (and results)

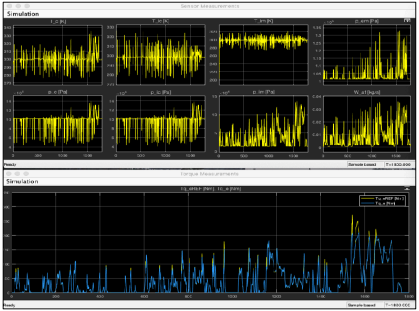
Buttons to Run simulation and to Exit the GUI

Graphical plots:
- Tq_eREF vs Tq_e
- Fault signal
- Residuals

Features of Simulation Testbed

- Simulation settings
- Realistic engine simulation and telemetry sensors readings
- Design and test residuals generation and fault diagnosis algorithms
- Present simulation results
- Customizable





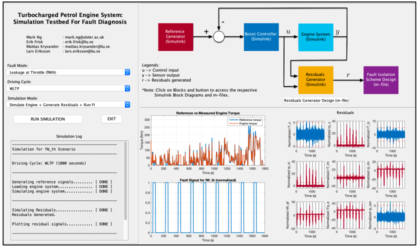
Features of Simulation Testbed

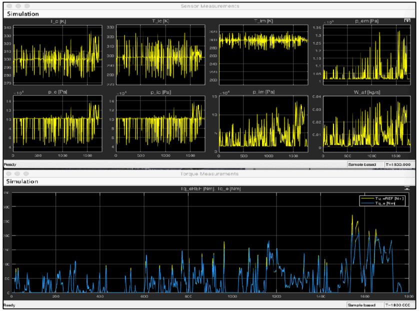
- Simulation settings
- Realistic engine simulation and telemetry sensors readings
- Design and test residuals generation and fault diagnosis algorithms
- Present simulation results
- Customizable

GitHub

Open-source for Research (and Teaching)

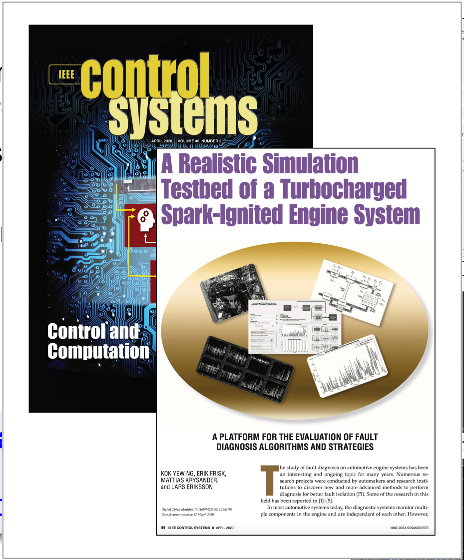
markusng.com/downloads/TCSI

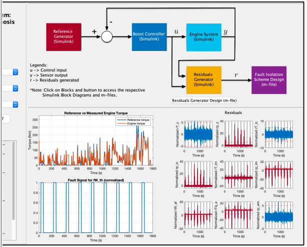


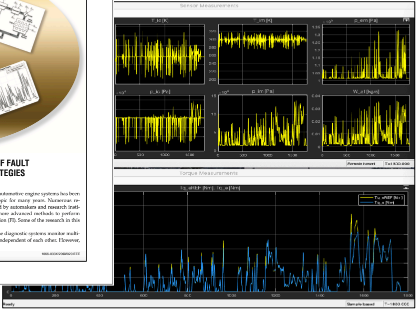


Features of Simulation Testbed

- Simulation settings
- Realistic engine sim
  telemetry sensors r
- Design and test res
  and fault diagnosis
- Present simulation r
- Customizable







Open-source for Resea

`markusng.com/do`

Demo

Generation of Additional Residuals

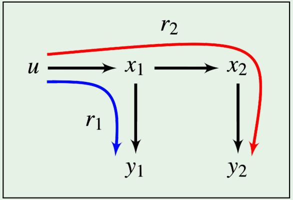
Theoretical Background

Let's revisit the earlier system

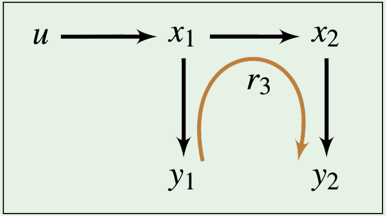
$$\begin{aligned}
\dot{x}_1 &= -x_1 + u + f_u, \\
\dot{x}_2 &= x_1 - x_2, \\
y_1 &= x_1 + f_1, \\
y_2 &= x_2 + f_2.
\end{aligned}$$

The following residuals were generated:

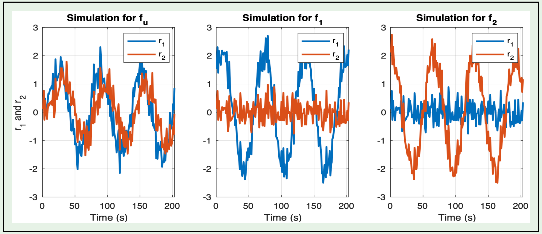
$$\begin{aligned}
r_1 &= \hat{y}_1 - y_1, \\
r_2 &= \hat{y}_2 - y_2.
\end{aligned}$$



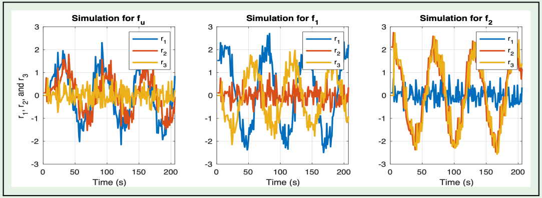
A new path traced to y_2 to generate r_3.



Omission of u from this path removes sensitivity of r_3 towards f_u.



(a) Sensitivity of only $\{r_1, r_2\}$.



(b) Sensitivity of $\{r_1, r_2, r_3\}$.

FSM for $\{r_1, r_2\}$ (unshaded rows), and r_3 (shaded row).

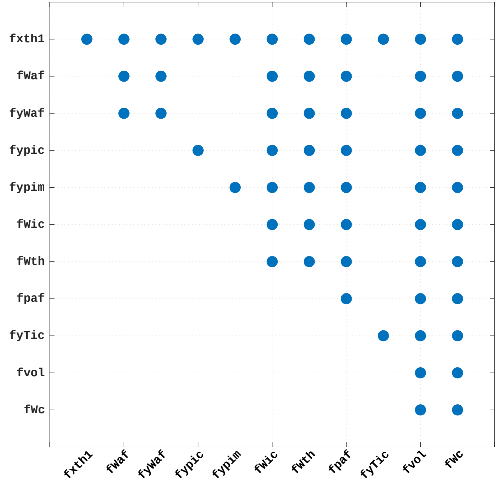
Residual	f_1	f_2	f_u
r_1	1	0	1
r_2	0	1	1
r_3	1	1	0

The triggered residuals and the diagnosis decisions for the isolated faults.

Triggered Residuals	**Diagnosis Decision**
r_1 and r_2	f_u
r_1 and r_3	f_1
r_2 and r_3	f_2

Application to Engine System

- Consider only 7 out of the “Original 9” (remove $r_{T_{qe}}$ and $r_{p_{em}}$)



Application to Engine System

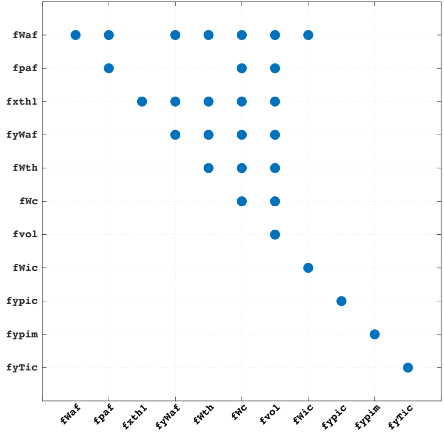
- Consider only 7 out of the “Original 9” (remove $r_{T_{qe}}$ and $r_{p_{em}}$)
- Algorithm returns 14 additional Residuals

Residuals	$f_{p_{af}}$	$f_{C_{vol}}$	$f_{W_{af}}$	f_{W_c}	$f_{W_{ic}}$	$f_{W_{th}}$	$f_{x_{th}}$	$f_{y_{p_{ic}}}$	$f_{y_{p_{im}}}$	$f_{y_{T_{ic}}}$	$f_{y_{W_{af}}}$
r_{T_c}	1	1	0	1	0	0	0	0	0	0	0
r_{p_c}	1	1	0	1	1	1	0	0	0	0	0
$r_{T_{ic}}$	0	1	0	1	0	0	0	0	0	1	0
$r_{p_{ic}}$	1	1	0	1	1	1	0	1	0	0	0
$r_{T_{im}}$	0	1	0	1	0	0	0	0	0	0	0
$r_{p_{im}}$	1	1	0	1	1	1	0	0	1	0	0
$r_{W_{af}}$	1	1	1	1	1	1	0	0	0	0	1
p_{im}-W_{af}	0	1	0	1	0	1	0	0	1	0	1
T_{im}-W_{af}	0	1	0	1	0	1	0	0	0	0	0
p_{ic}-W_{af}	0	1	0	1	0	1	1	1	0	0	1
T_{ic}-W_{af}	0	1	0	1	0	1	0	0	0	1	0
W_{af}-p_{im}	1	1	1	1	1	1	1	0	1	0	1
T_{im}-p_{im}	1	1	0	1	1	0	0	0	1	0	0
p_{ic}-p_{im}	1	1	0	1	1	1	0	1	1	0	0
T_{ic}-p_{im}	0	1	0	0	0	0	0	0	0	1	0
W_{af}-T_{im}	1	1	1	1	1	1	0	0	0	0	1
W_{af}-$T_{ic}p_{ic}T_{im}p_{im}$	1	1	1	1	1	1	1	1	1	0	1
W_{af}-$p_cT_{im}p_{im}$	1	1	1	1	1	1	1	1	1	0	1
W_{af}-$p_cT_{ic}p_{ic}$	1	1	1	1	1	1	1	1	0	1	1
p_{ic}-$T_cp_cT_{im}p_{im}$	0	0	0	0	1	0	0	1	1	0	0
T_{ic}-$T_cp_cT_{im}p_{im}$	0	0	0	0	1	0	0	0	1	1	0

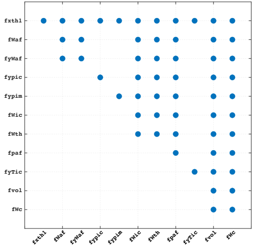
Application to Engine System

- Consider only 7 out of the “Original 9” (remove $r_{T_{qe}}$ and $r_{p_{em}}$)
- Algorithm returns 14 additional Residuals

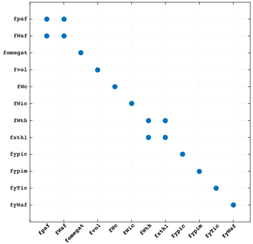
“Original 7” + 14 Additional Residuals



“Original 7”



Best Case

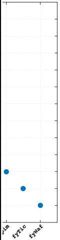


Application to Engine System

- Consider only 7 out of the “Original 9” (remove $r_{T_{qe}}$ and $r_{p_{em}}$)
- Algorithm returns 14 additional Residuals

“Or

fWa
fpa
fxth
fyWa
fWt
fW
fvo
fWi
fypi
fypi
fyTi



2020 7th International Conference on Control, Decision and Information Technologies (CoDIT’20) | Prague, Czech Republic / June 29 - July 2, 2020

Design and Selection of Additional Residuals to Enhance Fault Isolation of a Turbocharged Spark Ignited Engine System*

Kok Yew Ng[1,3], Erik Frisk[2], and Mattias Krysander[2]

***Abstract*— This paper presents a method to enhance fault isolation without adding physical sensors on a turbocharged spark ignited petrol engine system by designing additional residuals from an initial observer-based residuals setup. The best candidates from all potential additional residuals are selected using the concept of sequential residual generation to ensure best fault isolation performance for the least number of additional residuals required. A simulation testbed is used to generate realistic engine data for the design of the additional residuals and the fault isolation performance is verified using structural analysis method.**

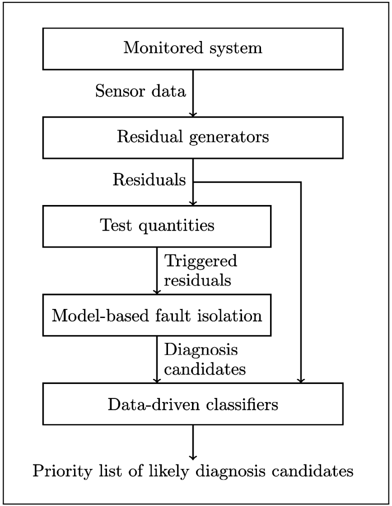
these techniques is that the number of sensors available would affect the quality of the diagnosis, i.e. more sensors (and hence, residuals) would lead to better fault isolation performance [8].

This paper proposes to use the concept of sequential residual generation reported in [9] to design and select additional residuals for a vehicular turbocharged spark ignited engine system with data obtained using the simulation testbed in [10]. The purpose is to improve fault isolation without adding physical sensors onto the engine system.

Fault Isolation

Possible Solutions

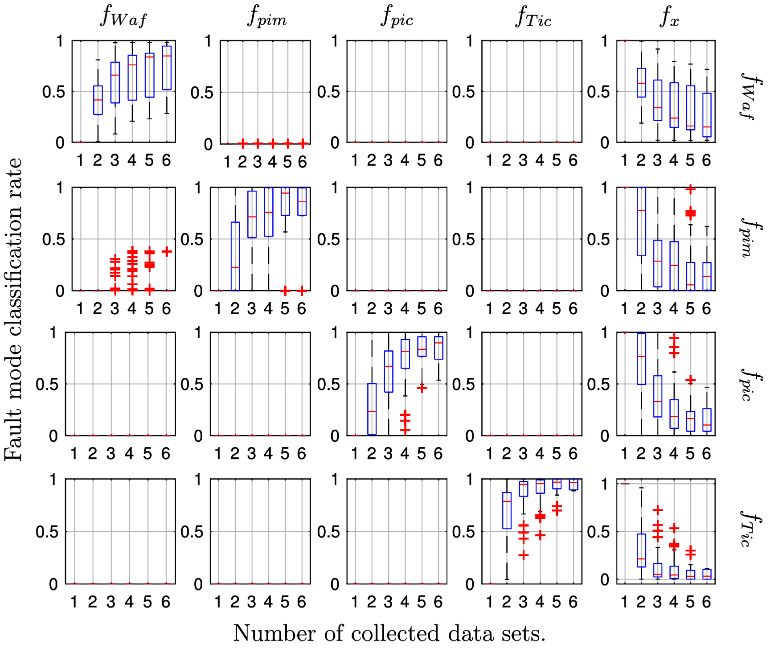
- Hybrid FI method



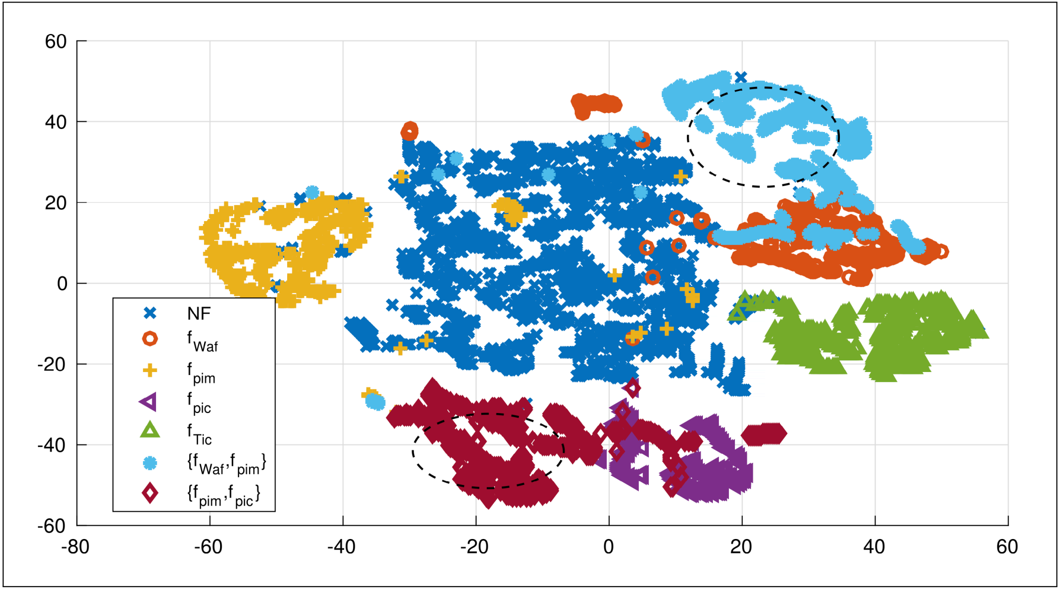
- From set of triggered residuals, compute set of minimal diagnosis candidates using CBD.
- Rank each diagnosis candidate by evaluating residual outputs using SVDD classifier trained with fault mode data.

$$rank(F_l) = \frac{1}{N}\sum_{k=1}^{N} C^{R}_{F_l}(r_k)$$

- Able to cater for unknown faults and multiple-faults modes.
- When true fault mode has been identified, collected data from fault scenario used to update SVDD classifiers.



f_{Waf}
f_{pim}
f_{pic}
f_{Tic}
f_x
Fault mode classification rate
Number of collected data sets.


60
40
20
0
-20
-40
-60
-80
-60
-40
-20
0
20
40
60
NF
f_{Waf}
f_{pim}
f_{pic}
f_{Tic}
$\{f_{Waf}, f_{pim}\}$
$\{f_{pim}, f_{pic}\}$

Control Engineering Practice 80 (2018) 146–156

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation

Daniel Jung [a,*], Kok Yew Ng [b,c], Erik Frisk [a], Mattias Krysander [a]

[a] *Vehicular Systems, Linköping University, Linköping, Sweden*
[b] *School of Engineering, Ulster University, Newtownabbey, BT37 0QB UK*
[c] *Electrical and Computer Systems Engineering, School of Engineering, Monash University Malaysia, Malaysia*

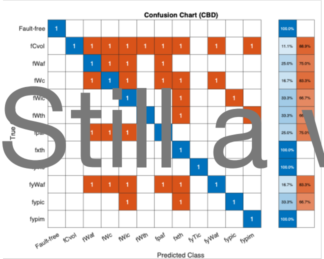
A R T I C L E I N F O

Keywords:
Fault diagnosis
Fault isolation
Machine learning
Artificial intelligence
Classification

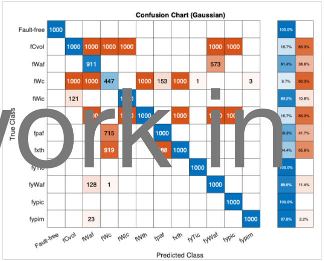
A B S T R A C T

Machine learning can be used to automatically process sensor data and create data-driven models for prediction and classification. However, in applications such as fault diagnosis, faults are rare events and learning models for fault classification is complicated because of lack of relevant training data. This paper proposes a hybrid diagnosis system design which combines model-based residuals with incremental anomaly classifiers. The proposed method is able to identify unknown faults and also classify multiple-faults using only single-fault training data. The proposed method is verified using a physical model and data collected from an internal combustion engine.

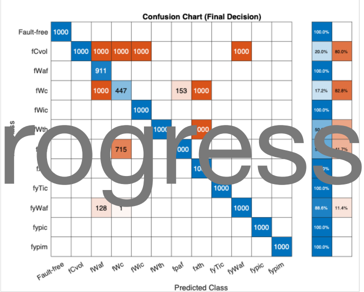
Still a work in progress



Method A



Method B



Final Decision

Select Key Bibliographies

- K. Y. Ng, E. Frisk, M. Krysander, and L. Eriksson, “A Realistic Simulation Testbed of a Turbocharged Spark-Ignited Engine System: A Platform for the Evaluation of Fault Diagnosis Algorithms and Strategies,” *IEEE Control Systems Magazine*, 2020.
- K. Y. Ng, E. Frisk, and M. Krysander, “Design of additional residuals to enhance fault isolation of a TCSI petrol engine system,” *IEEE CoDIT*, Prague, Czech Republic, 2020.
- D. Jung, K. Y. Ng, E. Frisk, and M. Krysander, “Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation,” *Control Engineering Practice*, 2018.
- E. Frisk, M. Krysander, and D. Jung, “A toolbox for analysis and design of model based diagnosis systems for large scale models.” *IFAC-PapersOnLine*, 2017.
- D. Jung, K. Y. Ng, E. Frisk, and M. Krysander, “A combined diagnosis system design using model-based and data-driven methods,” *IEEE SysTol*, Barcelona, Spain, 2016.
- K. Y. Ng, “Design and Development of A Simulation Environment and A Fault Isolation Scheme on A Volvo VEP4 MP Engine,” *Internal Research Technical Report*, Research and Development Centre, Volvo Car Corporation, Gothenburg, Sweden, 2015.
- L. Eriksson, “Modeling and control of turbocharged SI and DI engines.” *Oil & Gas Science and Technology-Revue de l’IFP*, 2007.
- D. Düştegör, E. Frisk, V. Cocquempot, M. Krysander, and M. Staroswiecki, “Structural analysis of fault isolability in the DAMADICS benchmark.” *Control Engineering Practice*, 2006.

Thank you!

*Acknowledgement: This research was supported by Volvo Cars, Gothenburg, Sweden.

A Realistic Digital Twin of a Vehicular Engine System

A Simulation Environment Platform for Fault Diagnosis

Mark Kok Yew Ng

Senior Lecturer, School of Engineering, Ulster University, UK
Adjunct Senior Research Fellow, School of Engineering, Monash University, Malaysia
ILN+ Researcher in Residence, Digital Catapult

mark.ng@ulster.ac.uk
markusng.com

The 8th IEEE Conference on Control Technology and Applications (CCTA) 2024
Workshop TuWH2T3

20th August 2024