Design for dependability: model/data analysis

Manuel Roveri

Politecnico di Milano, DEIB, Milano,Italy

manuel.roveri@polimi.it

Prof. Manuel Roveri

- Dipartimento di Elettronica, Informazione e Bioingegneria
- manuel.roveri@polimi.it
- http://roveri.faculty.polimi.it
- Research interests:
  - Machine/Deep Learning
  - Internet-of-Things and Cyber-Physical systems
  - Cloud/Edge Computing

Outline of the lectures

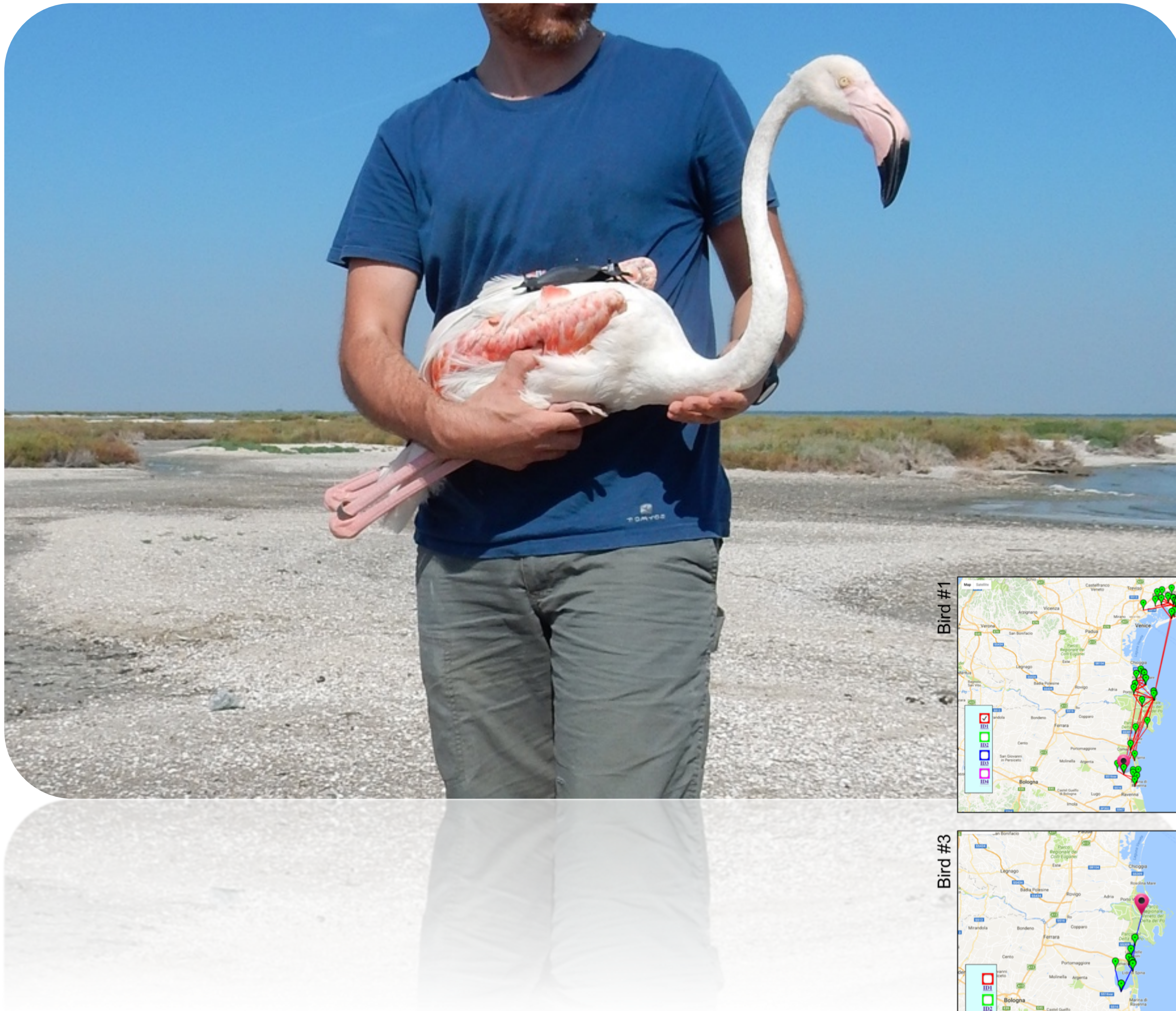
- **March 30 (9.15-12.15)**: Design for dependability: model/data analysis:
  - Introduction to the field
  - Fault Detection
- **March 31 (15.15-18.15)**: Design for dependability: model/data analysis:
  - Fault Diagnosis
  - Fault Mitigation
  - Presentation of the case studies
- **April 1 (10.15-12.15): Discussion on case studies**

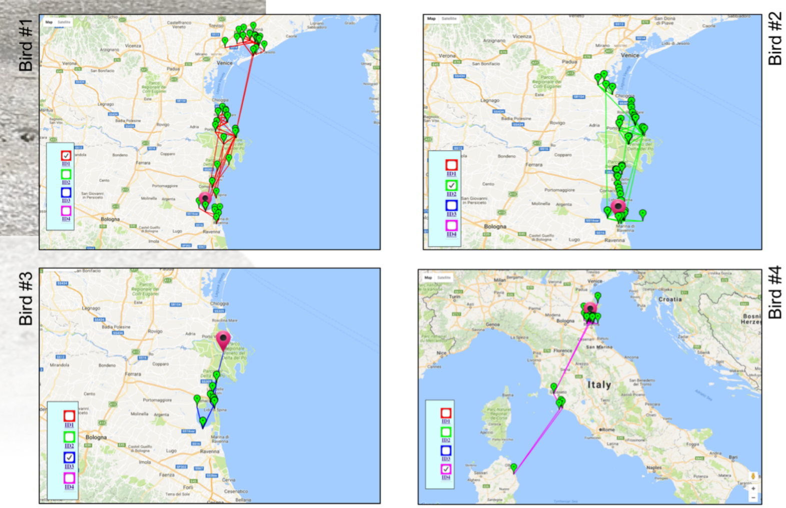
INTRODUCTION

Four examples of complex systems

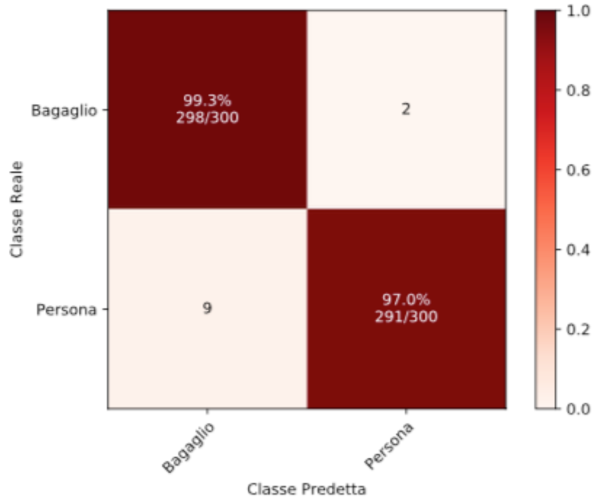
Four examples of complex systems

Four examples of complex systems





Four examples of complex systems



(a) Matrice di confusione

(b) Bagagli classificati come Persone

Detecting faults in complex systems

- In general, unwished **unpredictable situations are the result of faults affecting the sensor/actuator system** and may be either permanent or temporary, developing abruptly or incipiently.
- The **problem becomes more pronounced as sensing/actuation systems get older** since the sensors (along with the electronic chain up to the ADC) and the actuators are no more able to provide the correct functionality (and not always a calibration phase can solve the issue)

Detecting, Isolating and identifying faults by analyzing data:
Why?

Detecting faults by analyzing data

- It is of paramount relevance for all applications involving a decision making process to design methods able to **analyze and interpret incoming data streams** so that faults are
  - detected,
  - isolated,
  - identified as soon as possible and,
  - possibly, accommodated for before decisions or actions are taken on the basis of carried information.

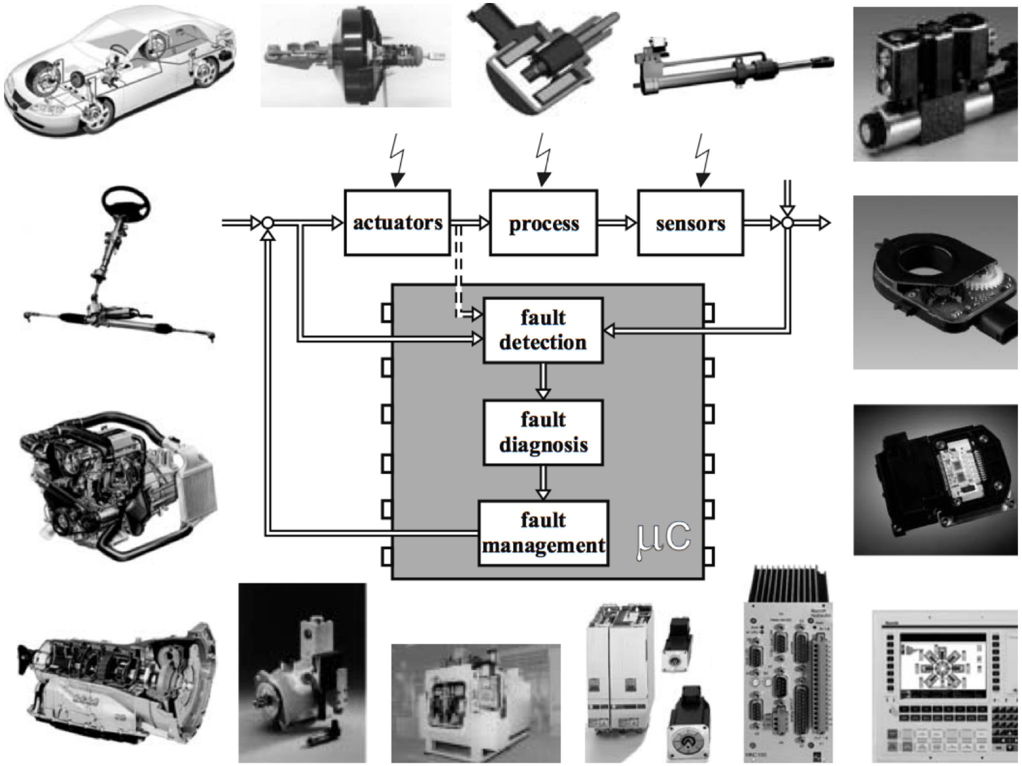
Why analyzing data?

- Despite the fact that **hardware solutions can be envisaged to partly mitigate the problems**, e.g., those based on modular redundancy by replicating the acquired hardware, they are not always able to deal with all types of faults that the sensor might encounter.
- Whereas an abrupt type of fault affecting a specific sensor can be easily detected by setting suitable thresholds, **a drift type of fault would affect all sensors, hence making impossible to detect it with a strict hardware replication schema**.
- **A modular redundancy also implies an increment in cost** that, by scaling linearly with the number of elements, might be acceptable for integrated sensors but not necessarily for more accurate and expensive traditional non silicon-based sensors.

How to detect, isolate and identify faults through data analysis?

Fault Detection and Diagnosis Systems (FDDS)

- Fault Detection and Diagnosis Systems are software applications designed to
  - detect potential insurgence of faults (fault detection),
  - identify them (i.e., determine their type and magnitude),
  - isolate faults (i.e., localize them within the system) and,
  - possibly, mitigate their effects through ad-hoc actions (management step)



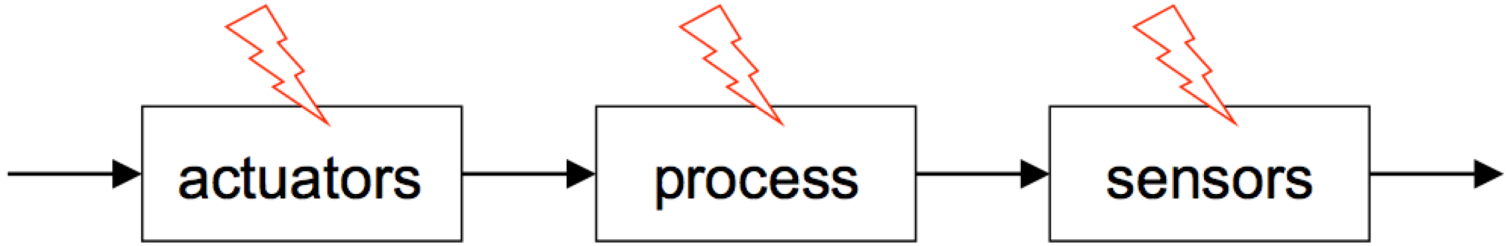
On-line Detection / Off-line Diagnosis

- Detection
  - On-line:
    - Low Complexity
    - Data stream
    - Raise alarms
- Diagnosis (Isolation/Identification)
  - Off-line:
    - High Complexity
    - Info about the system
    - Libraries of Faults
  - On-line:
    - Only when accommodation is considered

FAULTS AND THE FAULT DETECTION TASK

Faults

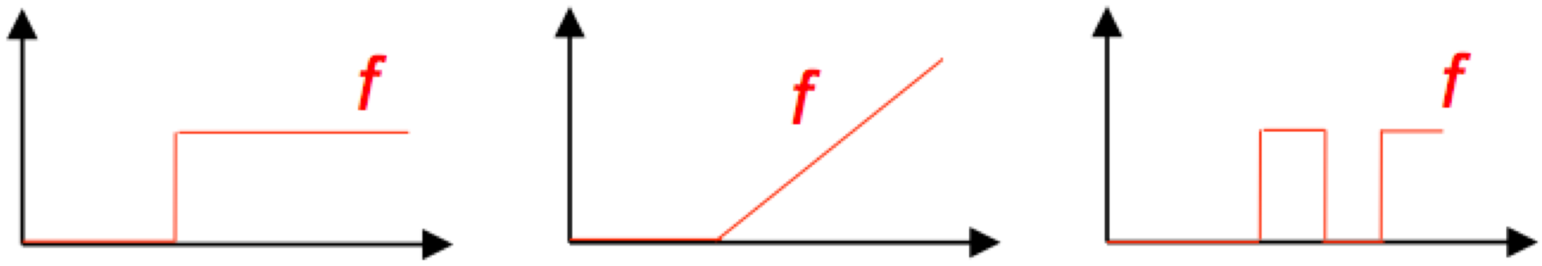
- **Fault** - An unpermitted deviation of at least one characteristic property or parameter of the system form the acceptable / usual / standard condition.
- Depending on the fault location:
  - Faults in actuators
  - Faults in sensors
  - Faults in process components



How to model the effects of faults on data?

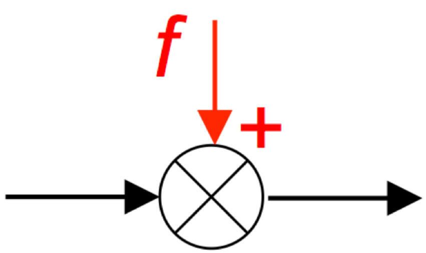
Faults: the temporal evolution

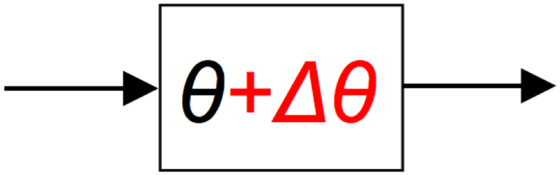
- Depending on the temporal evolution:
  - Abrupt faults – faults that manifest as quick changes in the system, modelled as steps or bias signals.
  - Incipient faults – manifest as slow drifts, modelled as ramps or drift signals.
  - Intermittent faults – manifest as impulse signals of unknown duration and even amplitude.



Faults: the effect on the system

- Depending on the effect on the system:
  - Additive faults – faults that affect system variables in an additive way; the effect of the fault on the system output only depends on the fault magnitude.
  - Multiplicative faults – faults that modify system parameters, their effect on the system outputs depends not only on the fault size but also on the value of the system input.

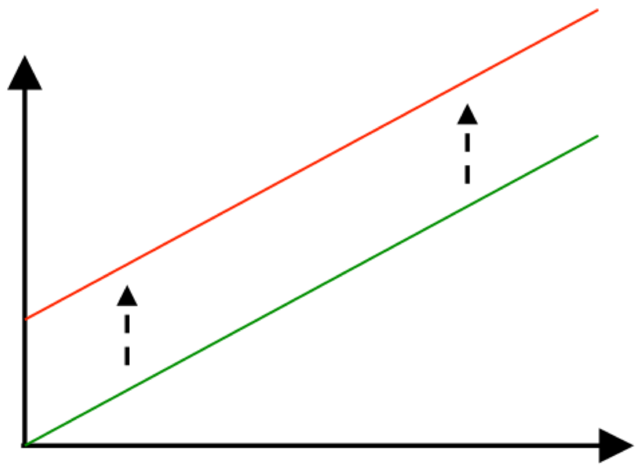




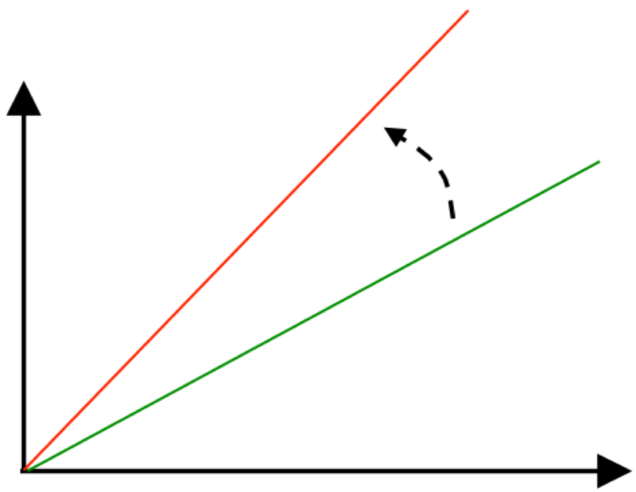
Example of faults

Event	Fault Modeling	Fault Evolution	Fault signature
Sensor Miscalibration	*Permanent*	*Incipient*	*Offset/Drift*
Thermal drift affecting sensors	*Permanent*	*Incipient*	*Drift/Precision degradation*
Electronic fault at the board level	*Permanent/Transient*	*Abrupt*	*Offset/Stuck-at Fault*
Communication error	*Permanent/Intermittent*	*Missing data*	
Software error at the readout system	*Permanent/Intermittent*	*Abrupt*	*Stuck-at Fault*

Faults: example of sensor miscalibration



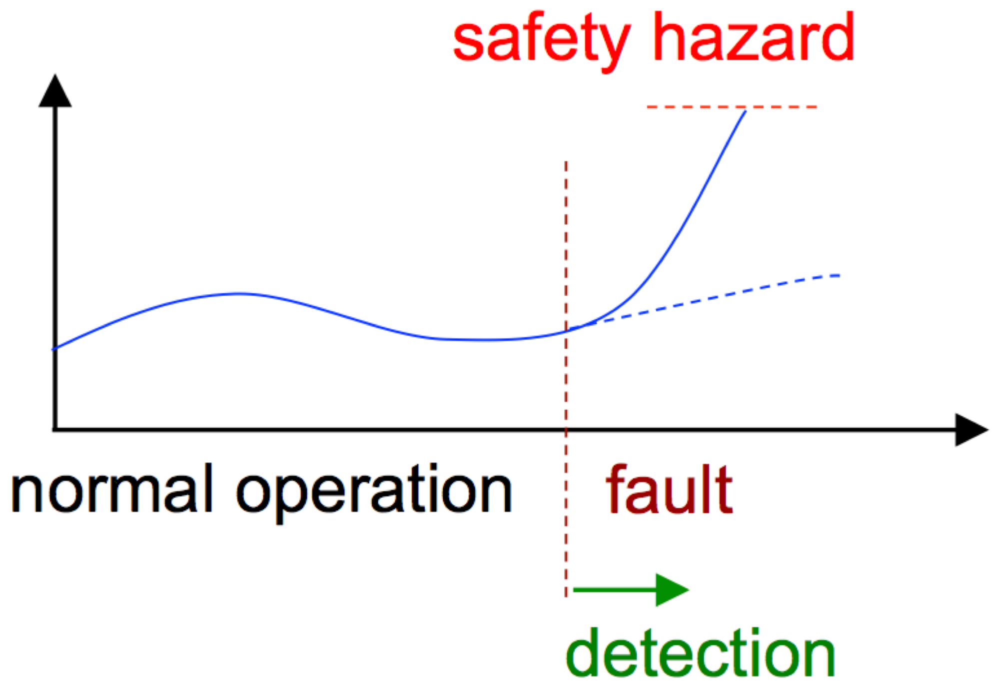
offset (additive)



change in static gain (multiplicative)

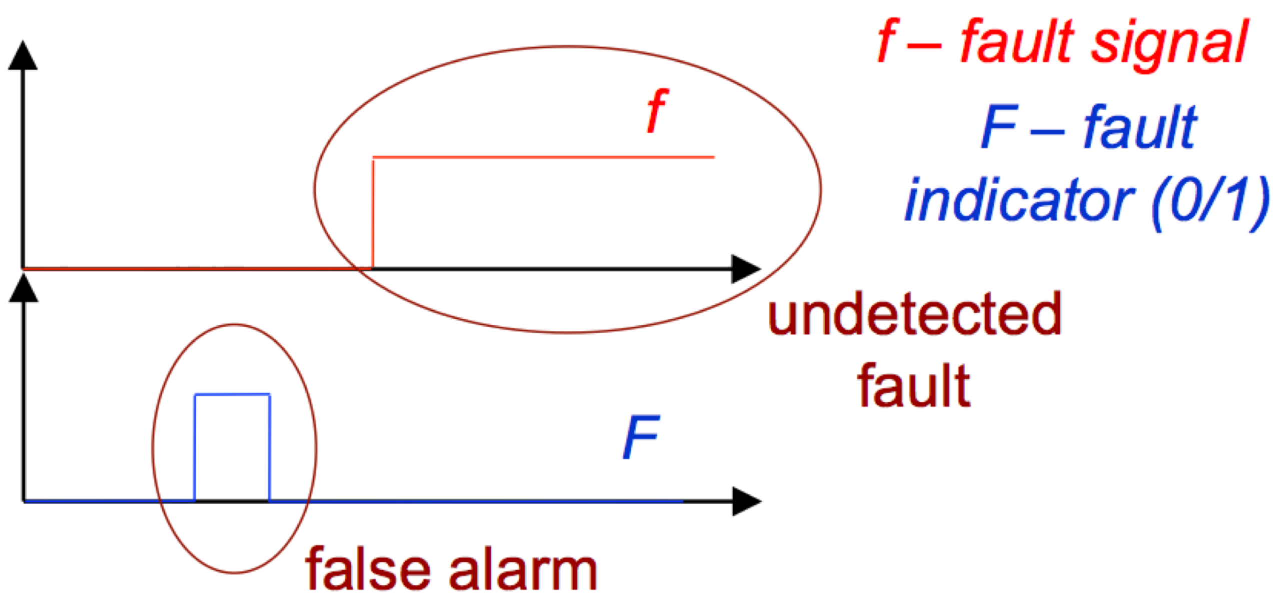
The fault detection task

- <u>Fault detection</u> – determination of the presence (or not) of faults in the system.
- **Goal** – to detect faults as soon as possible, before their future evolution leads to failures or security hazards.



The fault detection task

- Situations to avoid:
  - Undetected faults (false negatives) – faults acting on the plant that are not detected by the FD system.
  - False alarms (false positives) – an alarm is generated by the FD system being the plant fault-free.



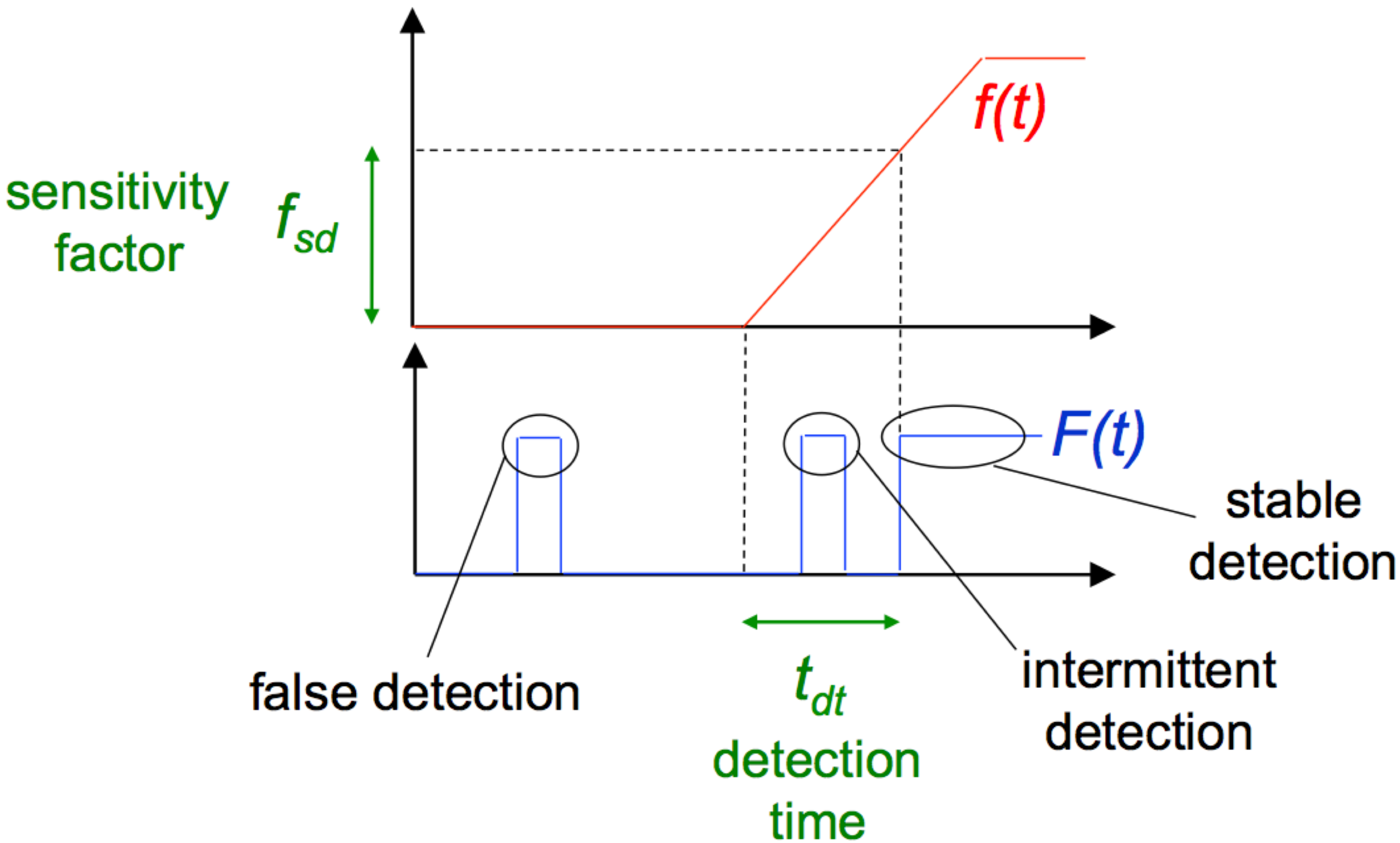
The fault detection task

- Undetected faults vs. false alarms:
  - In practice, the design of a FD system has to consider a compromise between sensitivity to faults and generation of false alarms.
  - **Undetected faults** have to be eliminated in safety-critical systems (examples: aircrafts, nuclear power plants).
  - **False alarms** are undesirable in systems whose shutdown leads to important economic losses.

Any complex system requires a carefully designed FD to reduce both undetected faults and false alarm (the trade-off is application-dependent)

The fault detection task

- Temporal behaviour of the FD system:



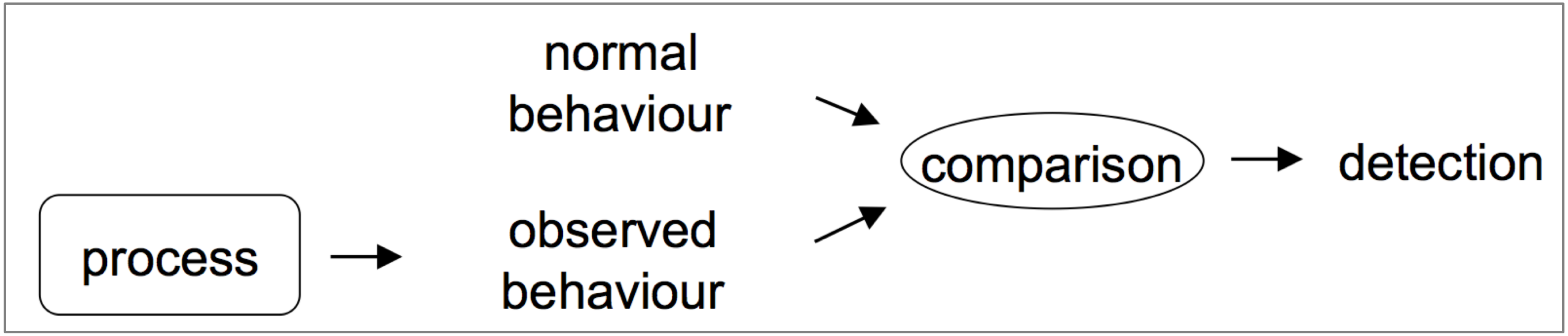
The fault detection task

- Performance indexes to evaluate FD systems:
  - False alarm rate - % of the time being the system in normal operation in which the FD system (incorrectly) indicates a faulty operation.
  - True detection rate - % of the time being the fault present in which the FD system (correctly) indicates the faulty operation.
  - Detection time (dt) – Period of time from the fault time instant up to the moment of the last rising edge of the detection indicator.
  - Sensitivity factor (fsd) – Value of the fault strength in the moment of the last rising edge of the detection indicator.

How to detect faults by analyzing data? Which are the main families of solutions?

Operating principle of FDs

- Operating principle: on-line comparison of the actual system observed behaviour against the known “normal operation behaviour”.

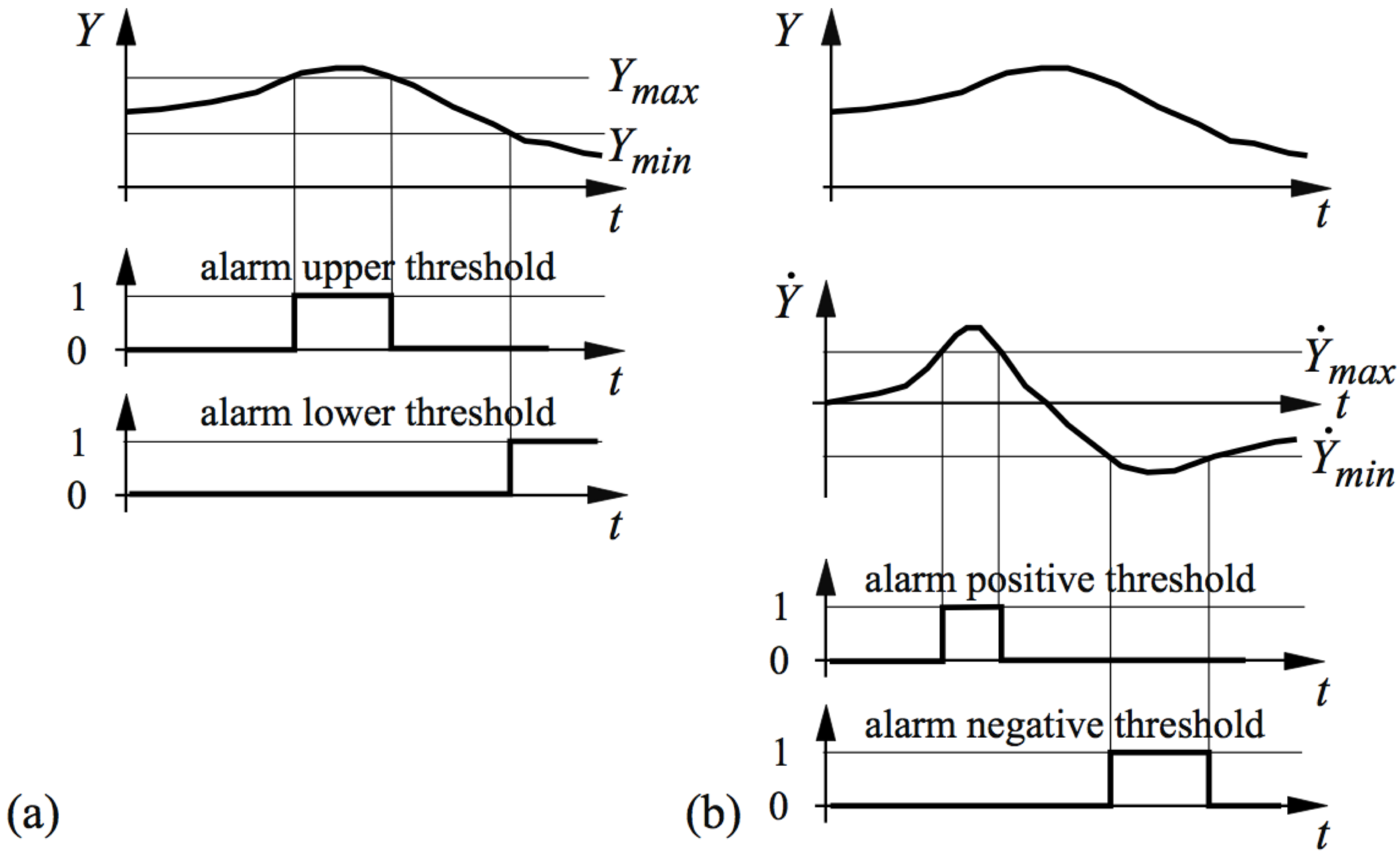


- Knowledge about the “normal operation behaviour”:
  - Empirical knowledge.
  - Extracted from from data.
  - Physical modelling.

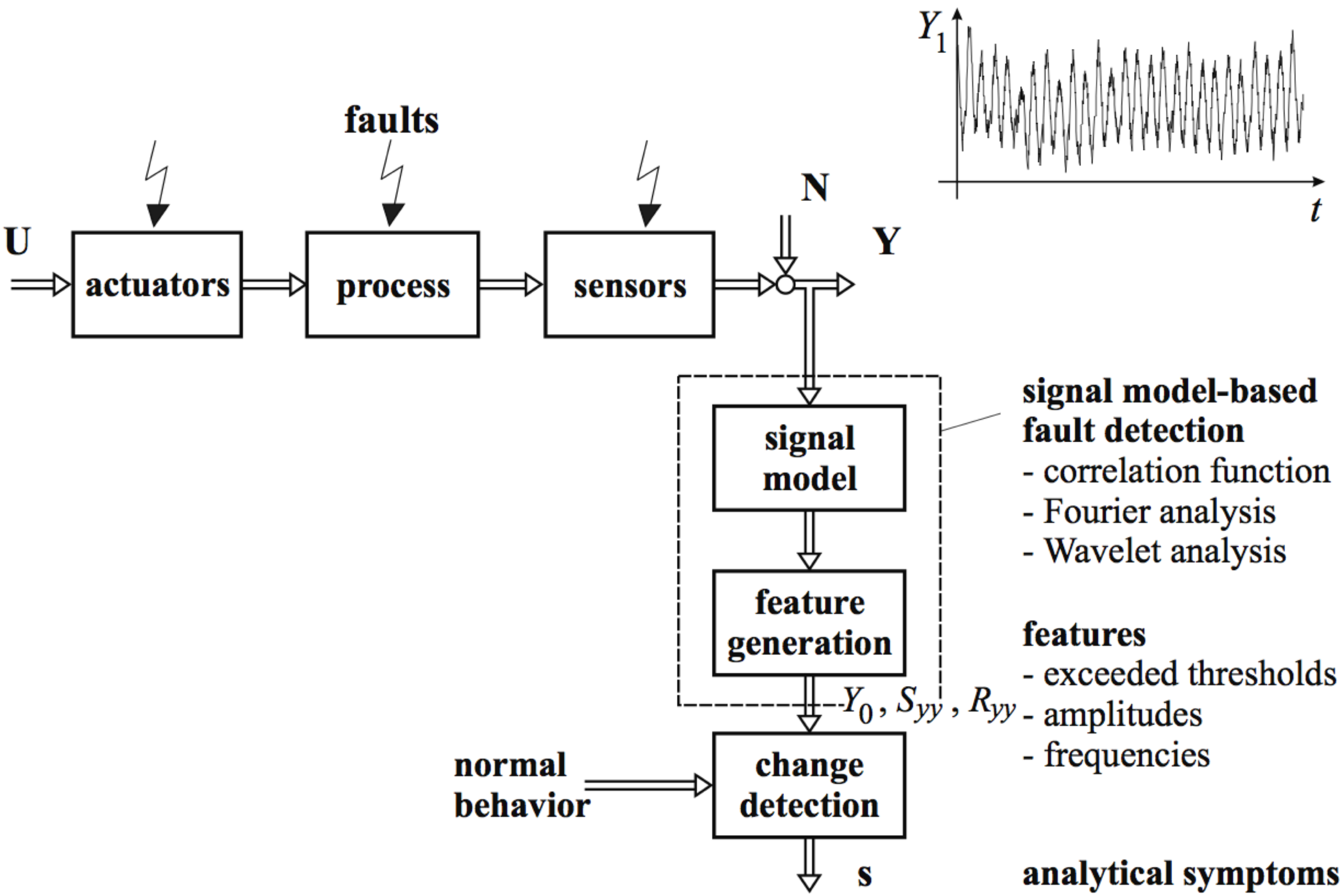
Methods

- Types of FD methods:
  - Traditional methods – simple test based on elementary empirical knowledge about the process.
  - Signal-based methods – Observation of signals whose behaviour in normal operation is known; signal models are characterized using experimental data.
  - Model-based methods – The input-output relation for normal operation is known; process models are obtained by apriori information with experimental data.

Methods: traditional methods

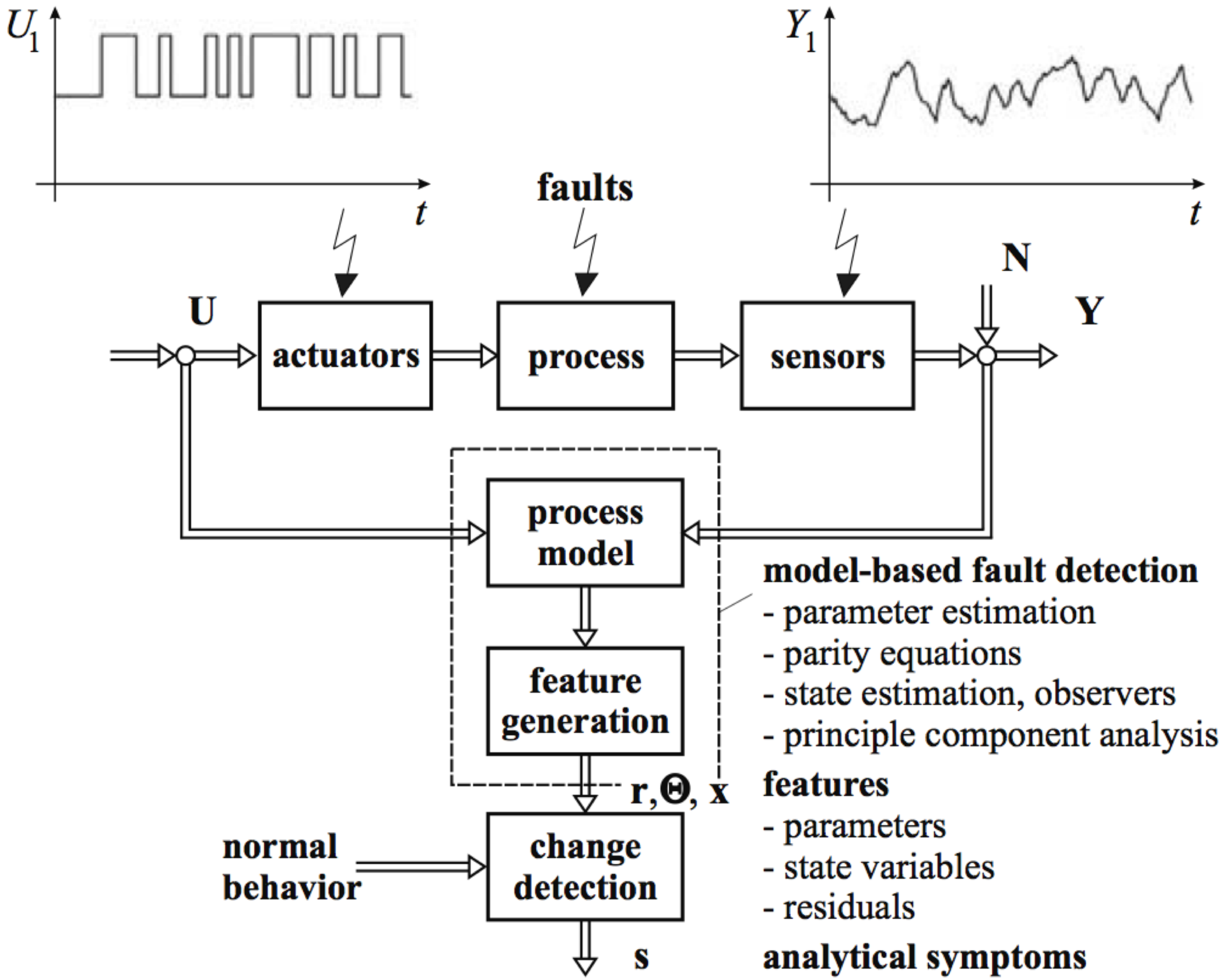


Methods: signal based methods



[Isermann, 2006]

Methods: model based methods



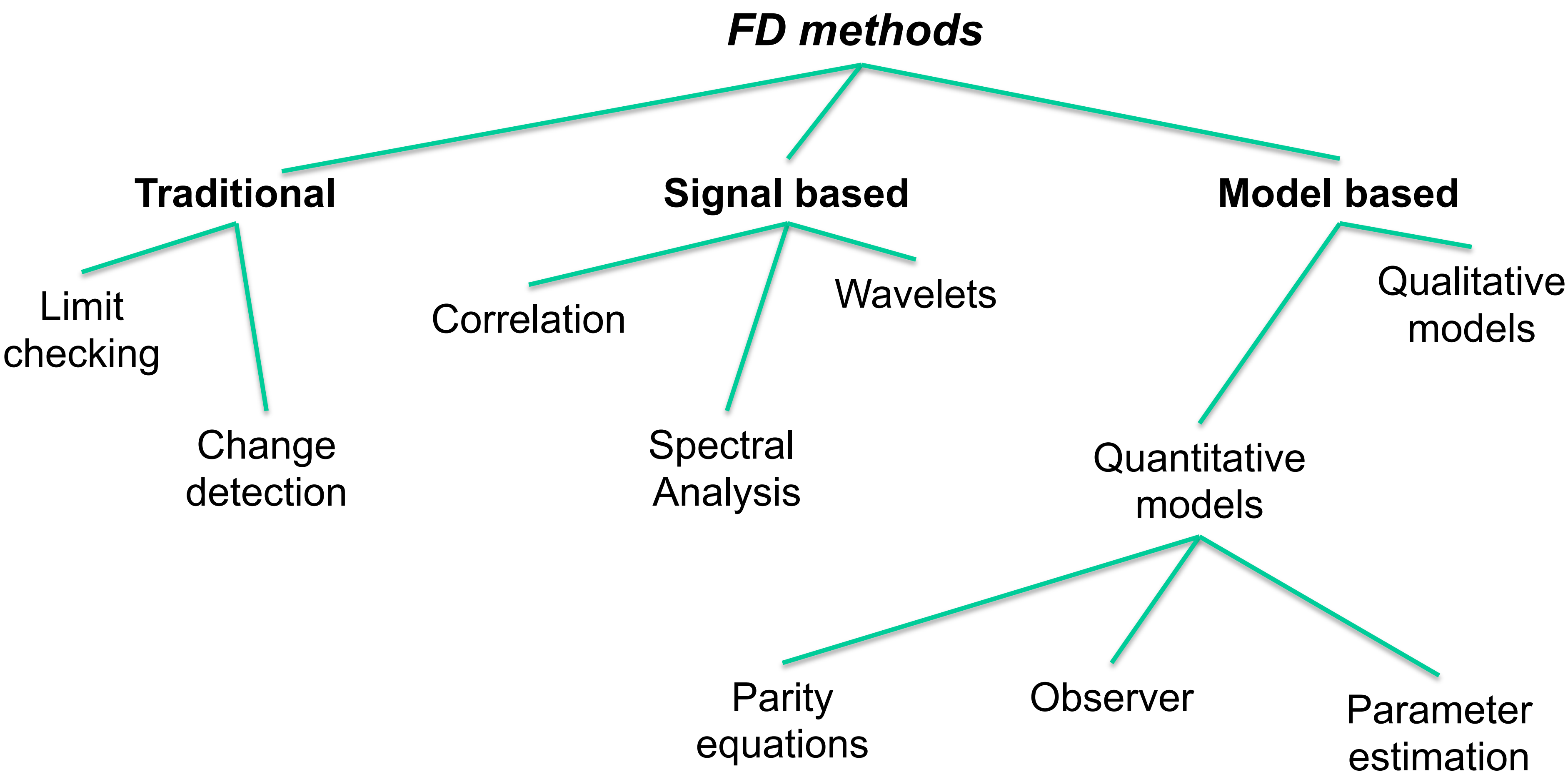
model-based fault detection
- parameter estimation
- parity equations
- state estimation, observers
- principle component analysis

features
- parameters
- state variables
- residuals

analytical symptoms

[Isermann, 2006]

Methods

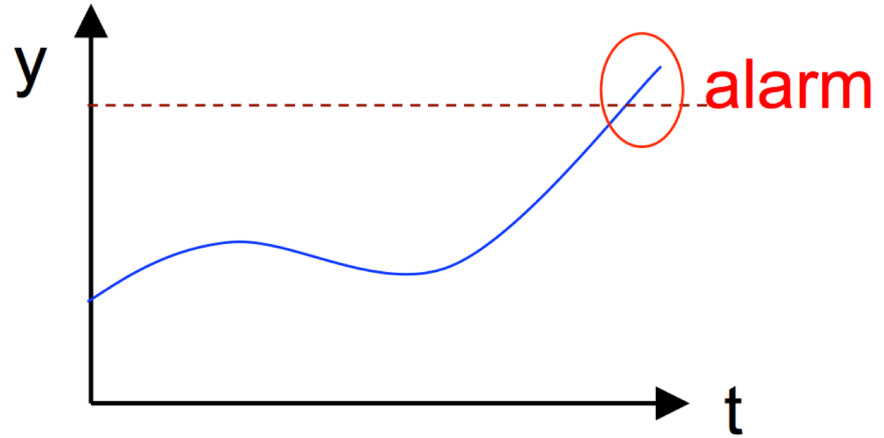

FD methods
Traditional
Signal based
Model based
Limit checking
Change detection
Correlation
Spectral Analysis
Wavelets
Qualitative models
Quantitative models
Parity equations
Observer
Parameter estimation

TRADITIONAL METHODS

Limit checking

- Testing if a given (measured) variable exceeds (indicating of faults) or not a known absolute limit.

$$y(t) \le Y_{\lim} \rightarrow F(t) = 0$$
$$y(t) > Y_{\lim} \rightarrow F(t) = 1$$



- Variants:
  - Two limits, associated to different levels of safety.
  - Use of superior and inferior limits.
- Easy to implement.
- Too conservative (low fault sensitivity).

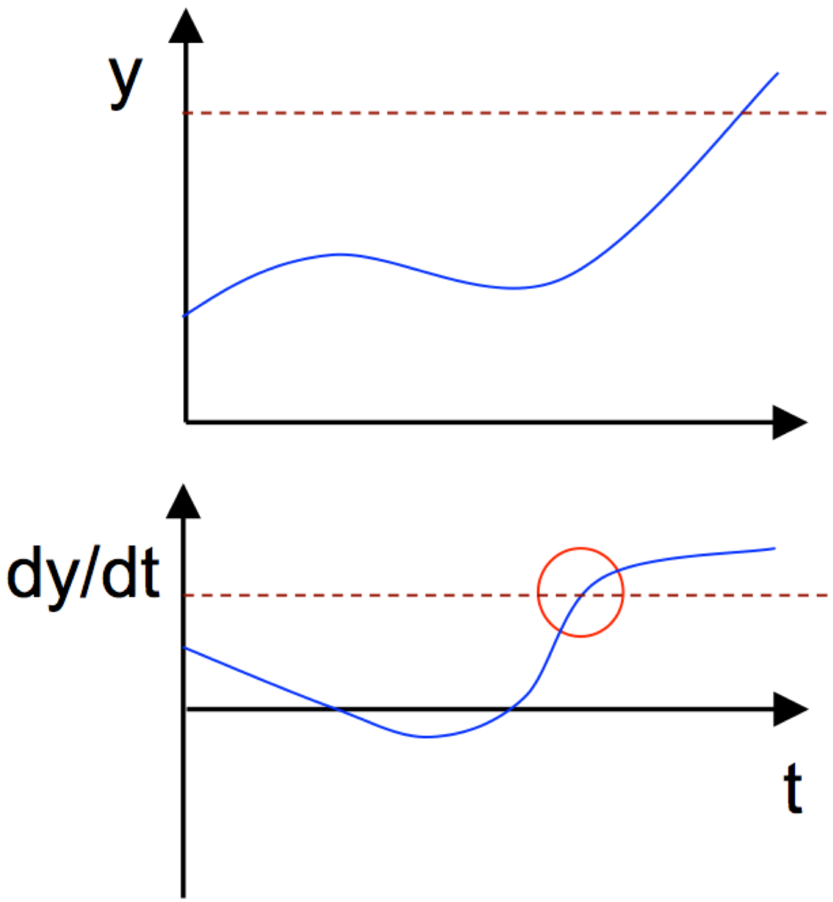
Trend checking

- Testing if the derivative of a given (measured) variable does not exceed a limit (or it is inside an interval).

$$\dot{y}(t) \le \dot{Y}_{\lim} \rightarrow F(t) = 0$$

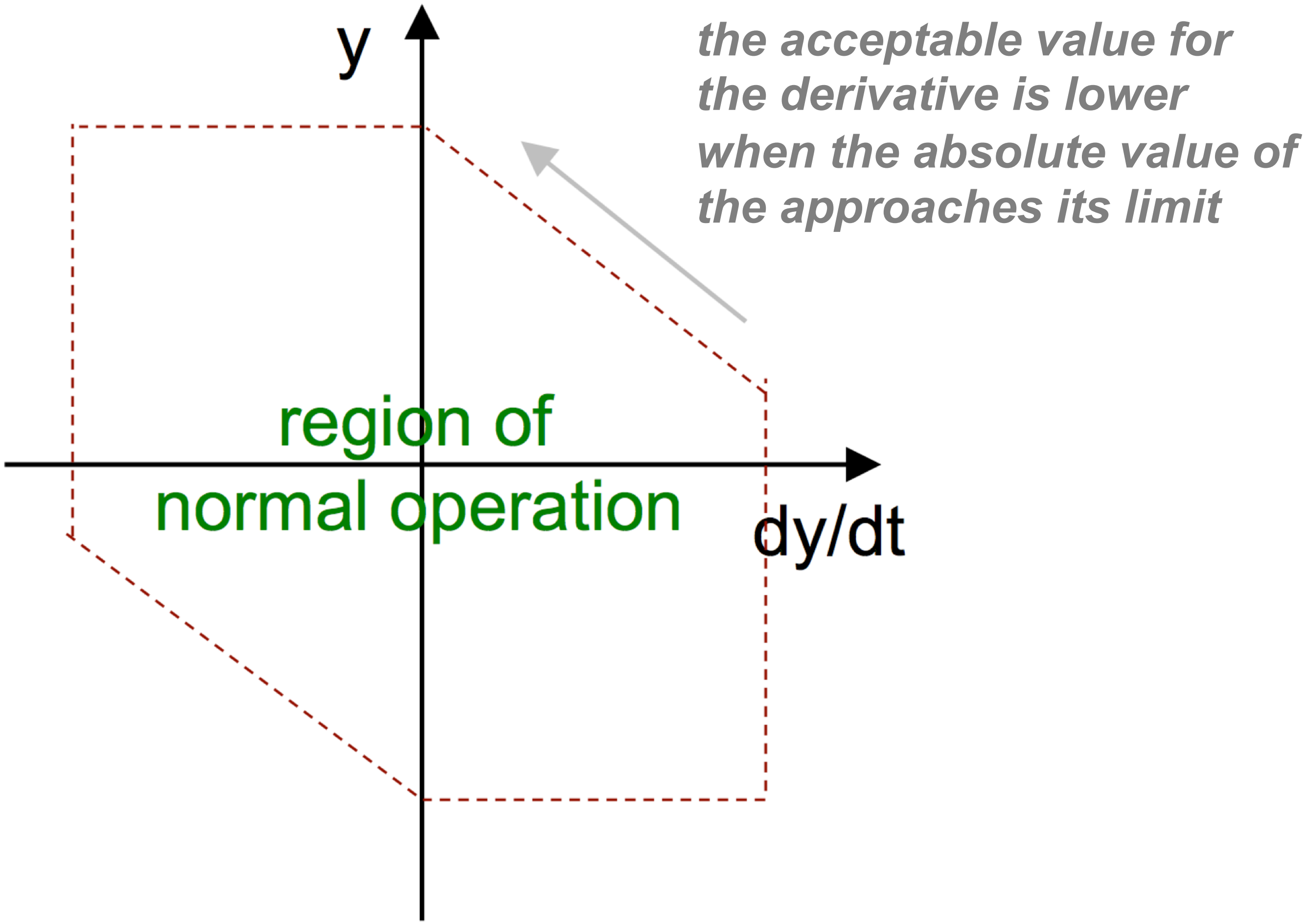
$$\dot{y}(t) > \dot{Y}_{\lim} \rightarrow F(t) = 1$$

- *In some cases, this can lead to a faster detection.*



Combination

- Testing both the absolute value and the value of the derivative.



More powerful techniques need to be considered

Statistical tests

- off-line: fixed length sequence (after storing all data)
- on-line: at each time instant

- **Statistical hypothesis tests:**
  - Off-line
  - Control of FPs
- **Change detection tests**
  - On-line
  - No control of FPs

Hypothesis tests: the literature

	Test family	*Type (P/NP)*	*Change (Ab/Dr)*	*Entity under test*	*1D/ ND*	*On-line/ Off-line*	*Training Set /A priori information*	*Notes*
Z-test	Statistical Hypothesis testing	Parameteric	Abrupt	Mean	1D	Off-line	Parameters	Assume normality and known variance
t-test	Statistical Hypothesis testing	Parameteric	Abrupt	Mean	1D	Off-line	None	Assume normality
Mann-Whitney U test	Statistical Hypothesis testing	Non Parameteric	Abrupt	Median	1D	Off-line	None	Rank Test
Kolmogorov-Smirnov test	Statistical Hypothesis testing	Non Parameteric	Abrupt	Pdf	1D	Off-line	None	Also goodness of fit test
Kruskal-Wallis test	Statistical Hypothesis testing	Non Parameteric	Abrupt	Median	1D	Off-line	None	Mann-Whitney based, Multiple subsets

Change-detection tests

- *Change detection tests are methods designed to detect variations in the pdf of the process generating the data*

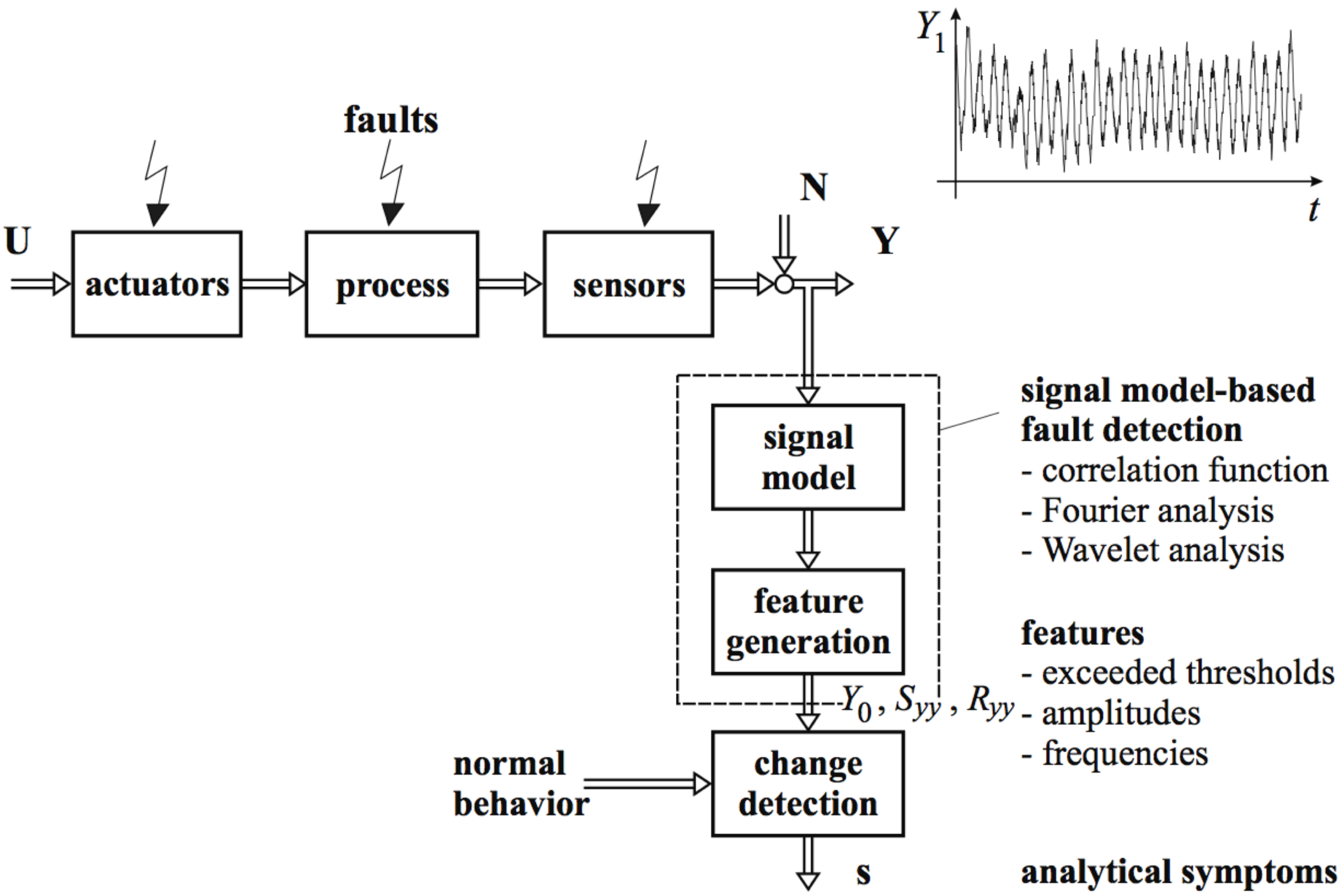
- **Parametric approach**: knowledge of the pdf before and after the change
  - CUSUM test
  - Shiryaev-Robert test
- **Nonparametric approach:**
  - CI-CUSUM test, NPCUSUM test
  - ICI-based change detection test

SIGNAL-BASED METHODS

Signal model-based fault-detection methods

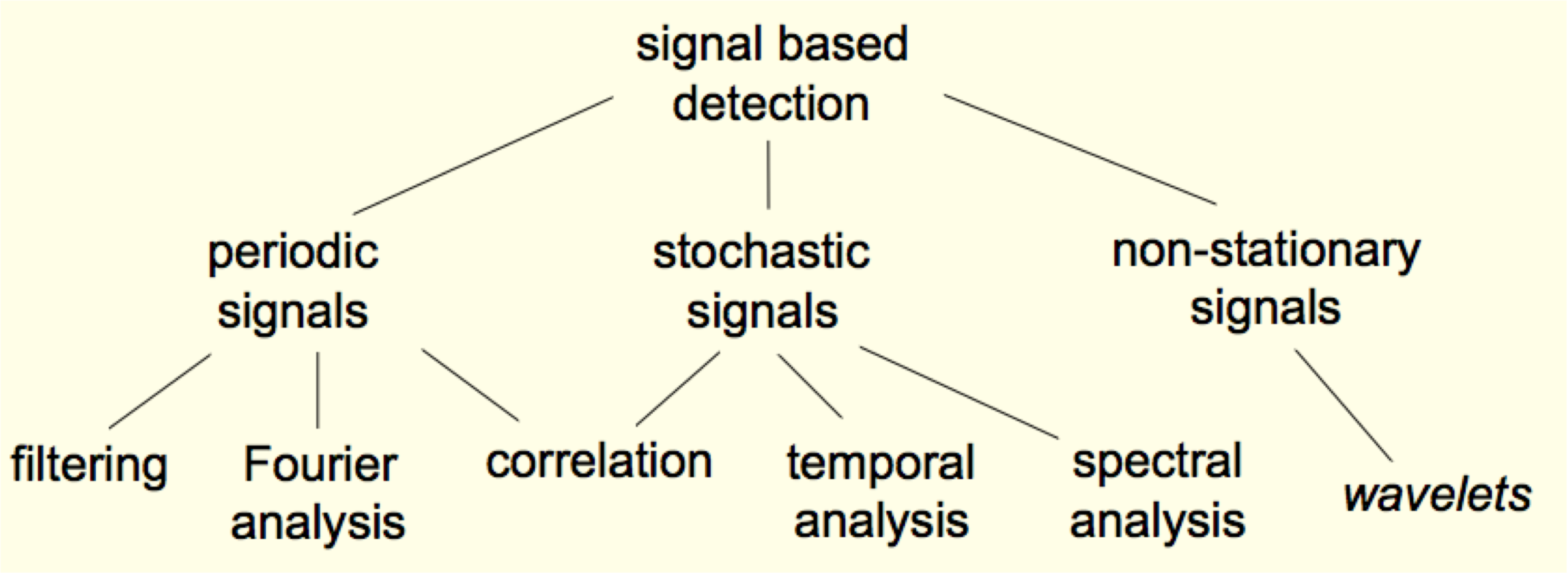
- Many **measured signals of processes show oscillations** that are either of harmonic or stochastic nature, or both
- If changes of these signals are related to faults in the actuators, the processes and sensors, **signal model-based fault-detection methods can be applied**
- Especially for **machine vibration**, the measurements of position, speed or acceleration allows to detect imbalance or bearing faults, knocking or chattering.

Scheme for the fault detection with signal models



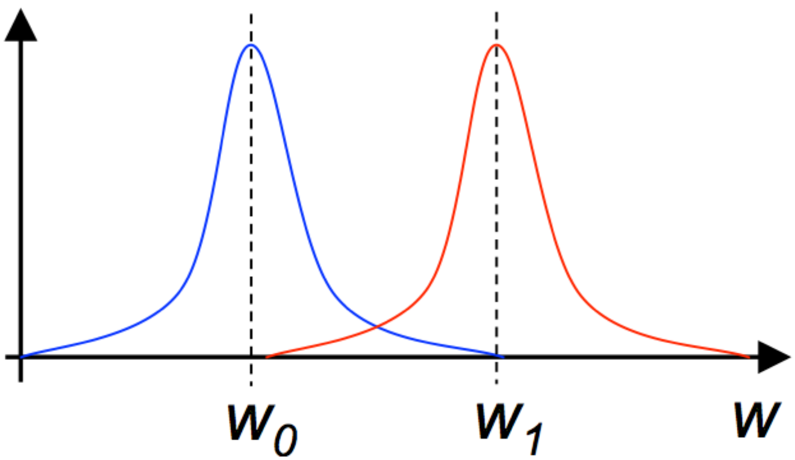
Signal based methods

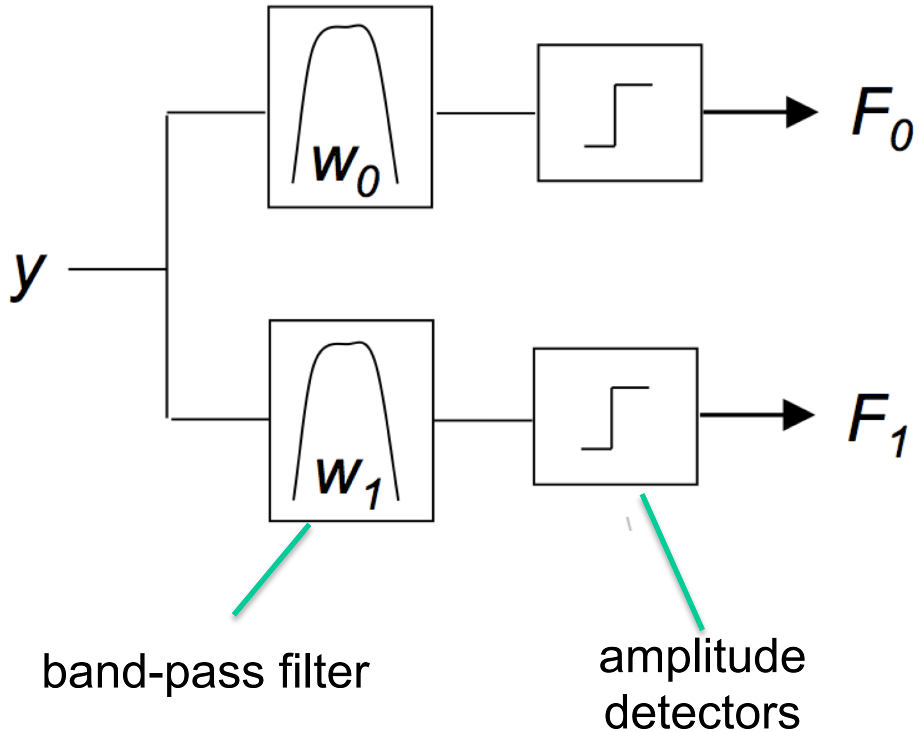
- Some signals present a known behaviour that is changed by the presence of faults.
- Types of signal and methods:



Filtering

- It can be used when faults modify the spectrum of a given signal in such a way each fault leads to a different bandwidth for the signal.

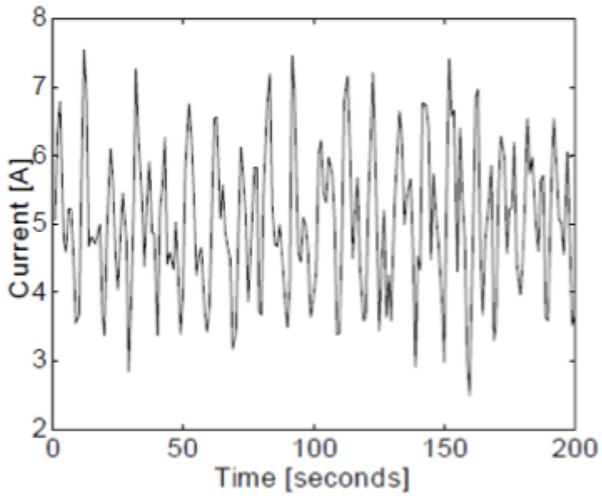




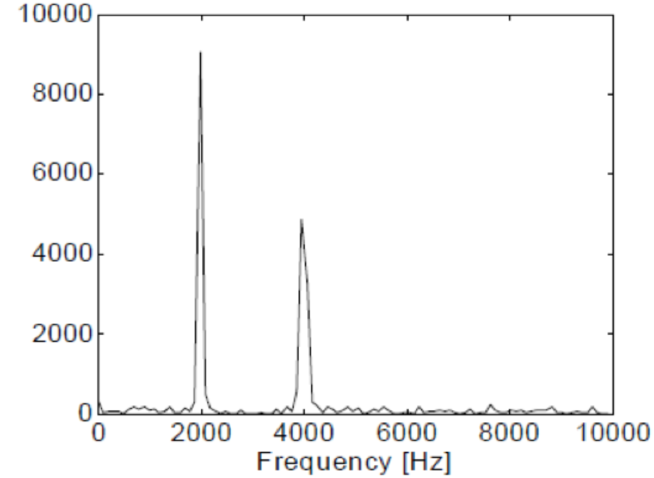
Fourier analysis

- It can be used when faults modify the signal spectrum

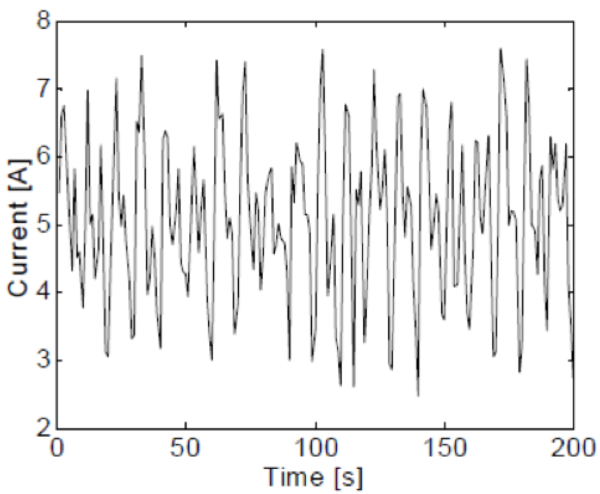
Normal operation



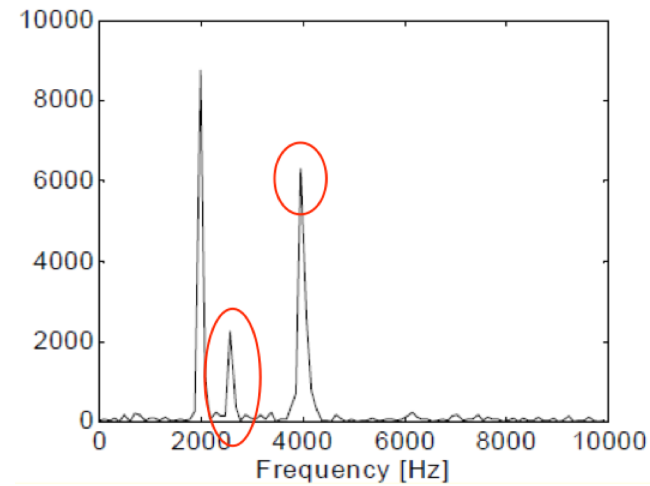
FFT →



Fault



FFT →

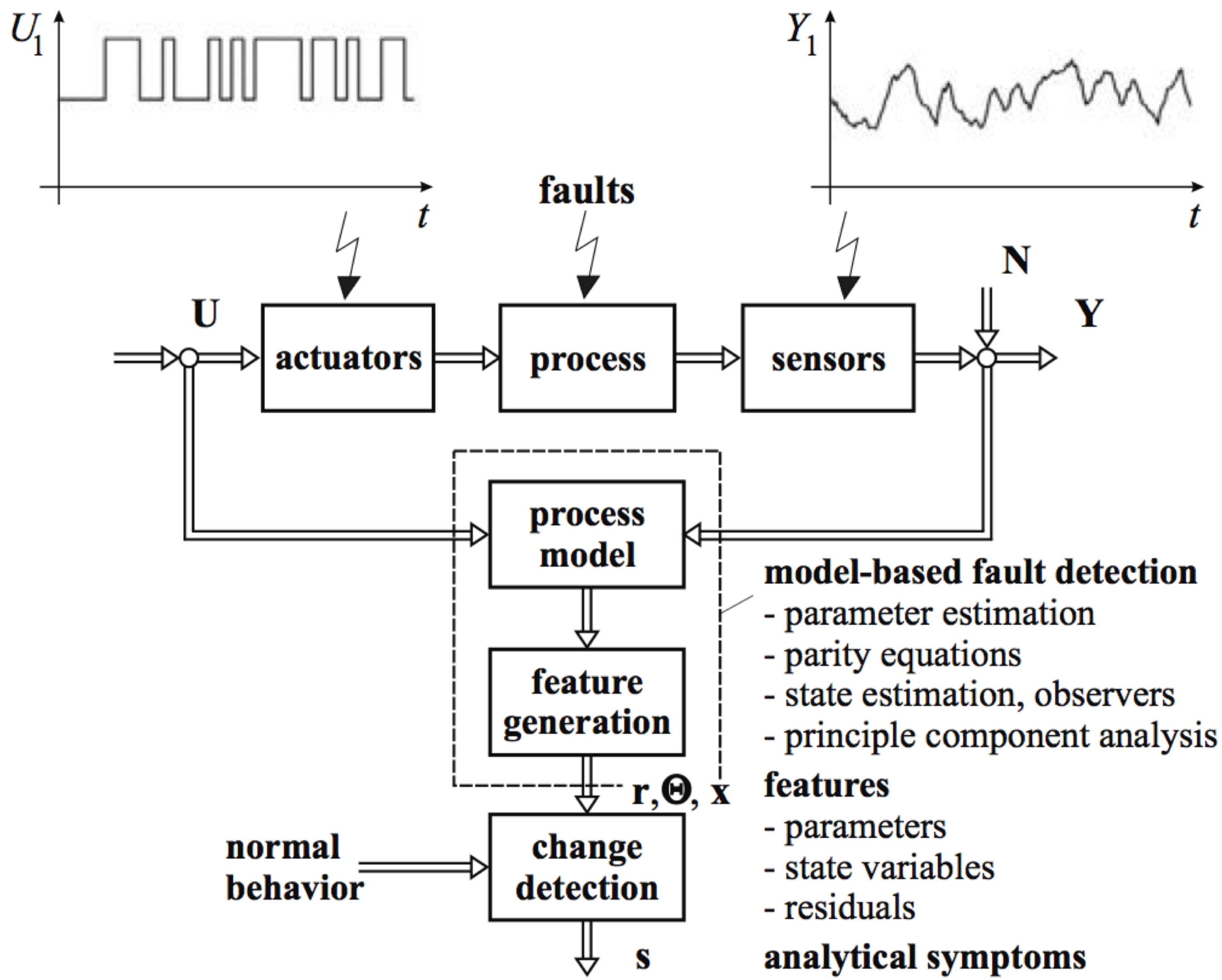


MODEL-BASED METHODS

Fault detection with process-identification methods

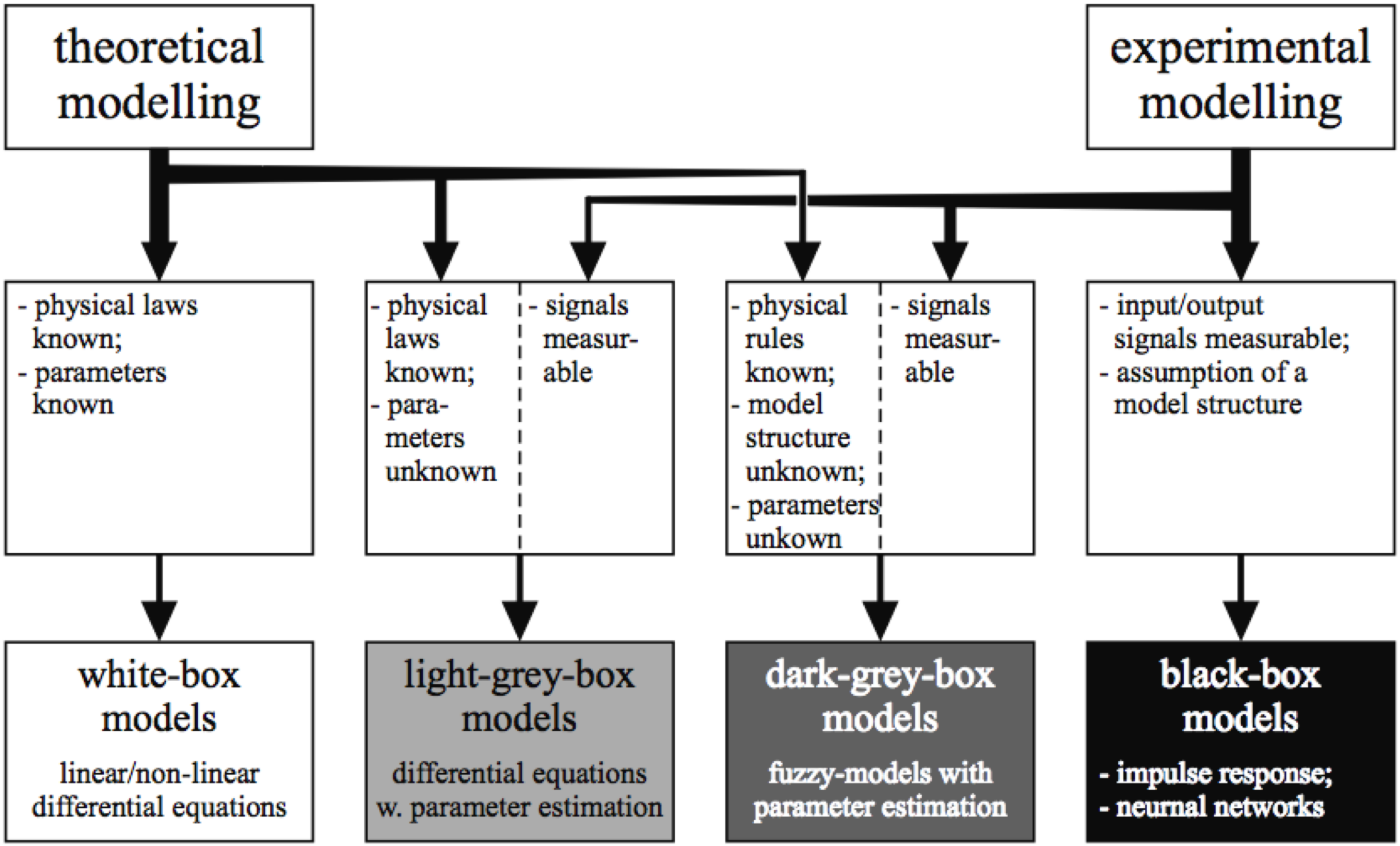
- **Mathematical process models describe the relationships between input signals and output signals**
- In many cases the **process models is unknown or some parameters are known**
- Model must be **precise** in order to express deviations results of process faults
- **Process-identification methods must be applied before applying any model-based fault-detection method**

Model-based fault detection

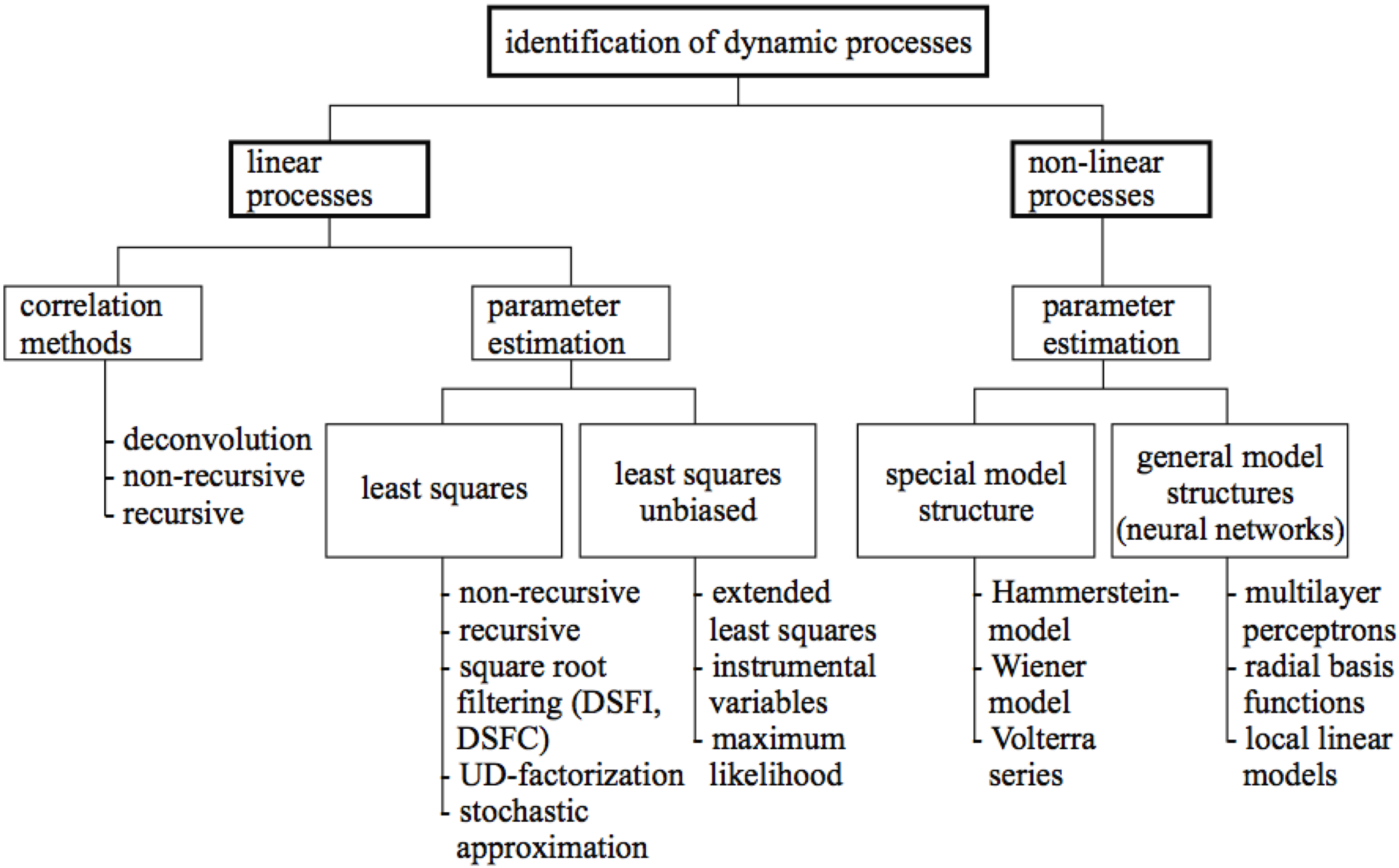


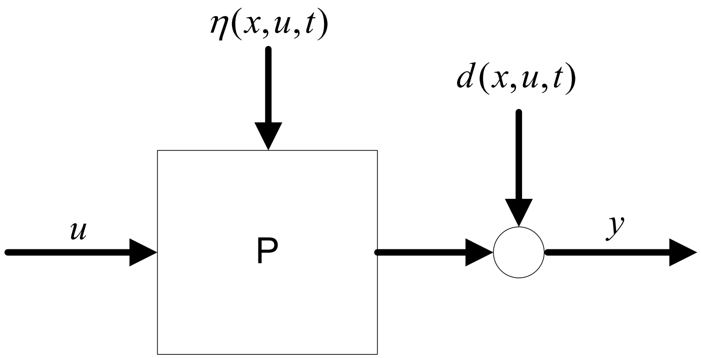
Process models

- Mathematical models of dynamic processes are primarily obtained by
  - <u>Theoretical/physical modelling</u>
    - the model is set up on the basis of mathematically formulated laws of nature
    - simplified assumptions about the process
  - <u>Identification methods (experimentally)</u>
    - mathematical model of a process from measurements
    - parametric or nonparameteric models



theoretical modelling
experimental modelling
- physical laws known;
- parameters known
- physical laws known;
- para-meters unknown
- signals measur-able
- physical rules known;
- model structure unknown;
- parameters unkown
- signals measur-able
- input/output signals measurable;
- assumption of a model structure
white-box models
linear/non-linear differential equations
light-grey-box models
differential equations w. parameter estimation
dark-grey-box models
fuzzy-models with parameter estimation
black-box models
- impulse response;
- neurnal networks


identification of dynamic processes
linear processes
non-linear processes
correlation methods
parameter estimation
parameter estimation
deconvolution
non-recursive
recursive
least squares
least squares unbiased
special model structure
general model structures (neural networks)
non-recursive
recursive
square root filtering (DSFI, DSFC)
UD-factorization
stochastic approximation
extended least squares
instrumental variables
maximum likelihood
Hammerstein-model
Wiener model
Volterra series
multilayer perceptrons
radial basis functions
local linear models



$$\begin{aligned} x(k+1) &= f(x(k), u(k)) + \eta(k), \\ y(k) &= h(x(k), u(k)) + d(k) \end{aligned}$$

- P is described through the input-output representation:

$$y(k) = h(y(k-1), y(k-2), \ldots, y(k-k_y), u(k), u(k-1), \ldots, u(k-k_u)) + d(k)$$

- Linear input-output models represent a further specific subcase:

$$A(z)y(k) = \sum_{i=1}^{m} \frac{B_i(z)}{F_i(z)} u_i(k) + \frac{C(z)}{D(z)} d(k)$$

$A(z)$, $B_i(z)$, $F_i(z)$, $C(z)$, $D(z)$: z-transfer functions

Some specific input-output models

- AR system model: linear autoregressive model

$$A(z)y(k) = \sum_{i=1}^{m} \cancel{\frac{B_i(z)}{F_i(z)}u_i(k)} + \frac{\cancel{C(z)}}{\cancel{D(z)}}d(k) \Rightarrow y(k) = \sum_{i=1}^{k_y} a_i y(k-i) + d(k).$$

- ARX system model: linear autoregressive model with exogenous inputs

$$A(z)y(k) = \sum_{i=1}^{m} \frac{B_i(z)}{\cancel{F_i(z)}}u_i(k) + \frac{\cancel{C(z)}}{\cancel{D(z)}}d(k)$$

$$y(k) = \sum_{i=1}^{k_y} a_i y(k-i) + \sum_{j=0}^{k_u} b_j u(k-j) + d(k).$$

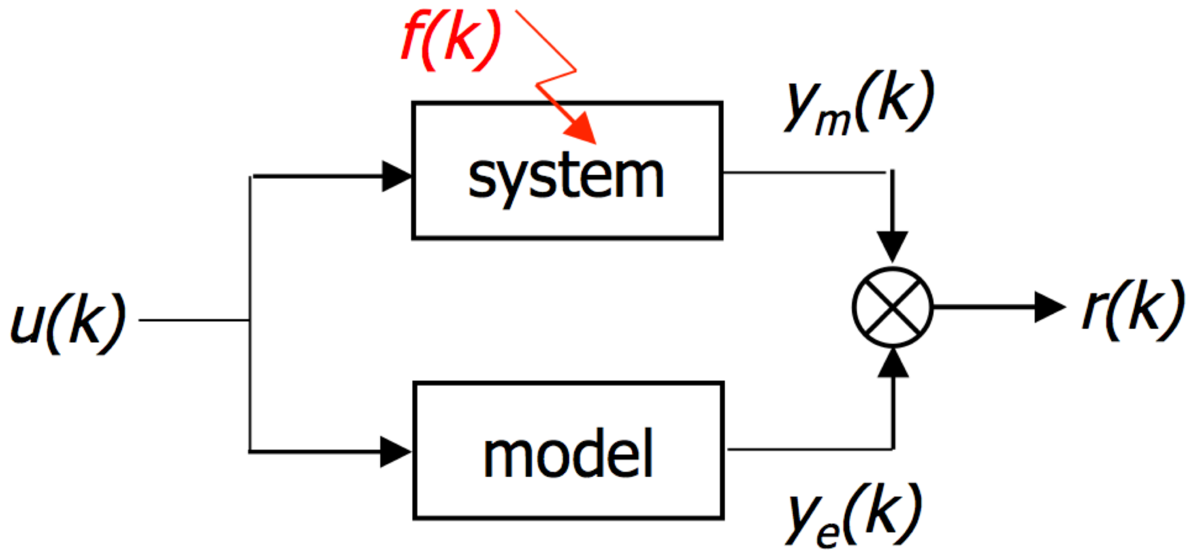
- Non-linear models:
  - Recurrent Neural Networks
  - LSTM/ESNs
  - Non-linear ARX

Model-based fault detection: preliminaries

- Analytical redundancy – existence of two or more different ways to determine the value of a given variable, at least one of them using the system model (for normal operation).
- Residual – Difference in the evaluation of a given variable by two different ways; residuals (with large values) are indicative of faults.
- Analytical Redundancy Relation (ARR) – relation between known or measured variables that is satisfied in absence of faults (moreover, it is expected that is not satisfied when a fault appears); ARRs are normally expressed in the form $f(u,y)=0$

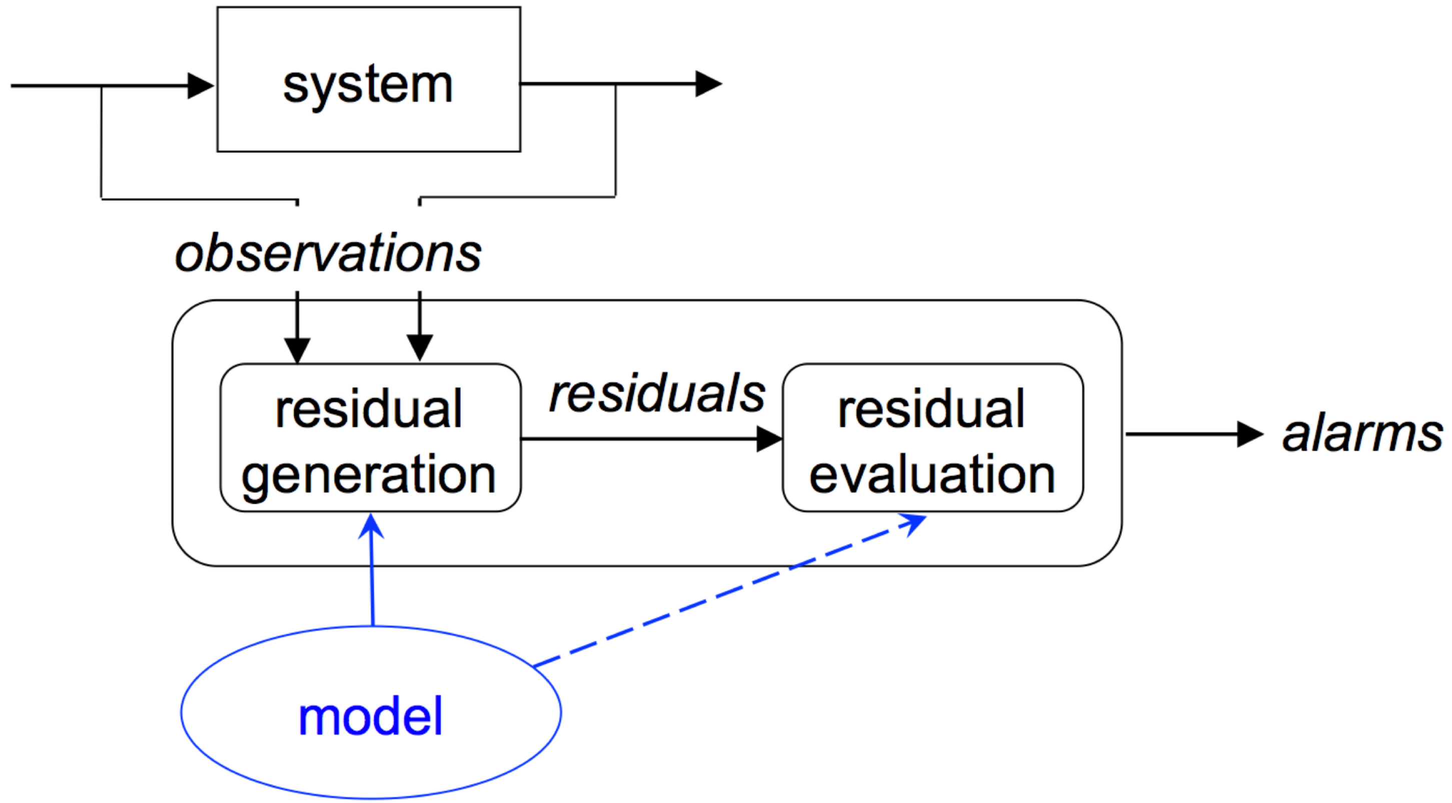
Analytical redundancy

- Example:
  - Model: $y(k) = ay(k-1)+bu(k-1)$
  - Measured output: $y_m(k)$
  - Output estimation (model): $y_e(k) = ay_e(k-1)+bu(k-1)$
  - Residual: $r(k) = y_m(k)- y_e(k)$



Architecture of MBFD systems

- Architecture of model-based fault detection (MBFD) systems:



Stages in MBFD

1. Residual generation:
   - Combined use of model and measurements to obtain fault indicators.
   - If the model is perfect then all the residuals are zero during normal plant operation.
   - At least one of the residuals deviates from zero when a fault to be detected is acting on the system.

$$\boldsymbol{r}(t)=0 \rightarrow FD(t)=0$$
$$r_{\mathrm{i}}(t)\neq 0 \rightarrow FD(t)=1$$

- One residual can be sufficient for FD, several residuals are needed for FDI.

Stages in MBFD

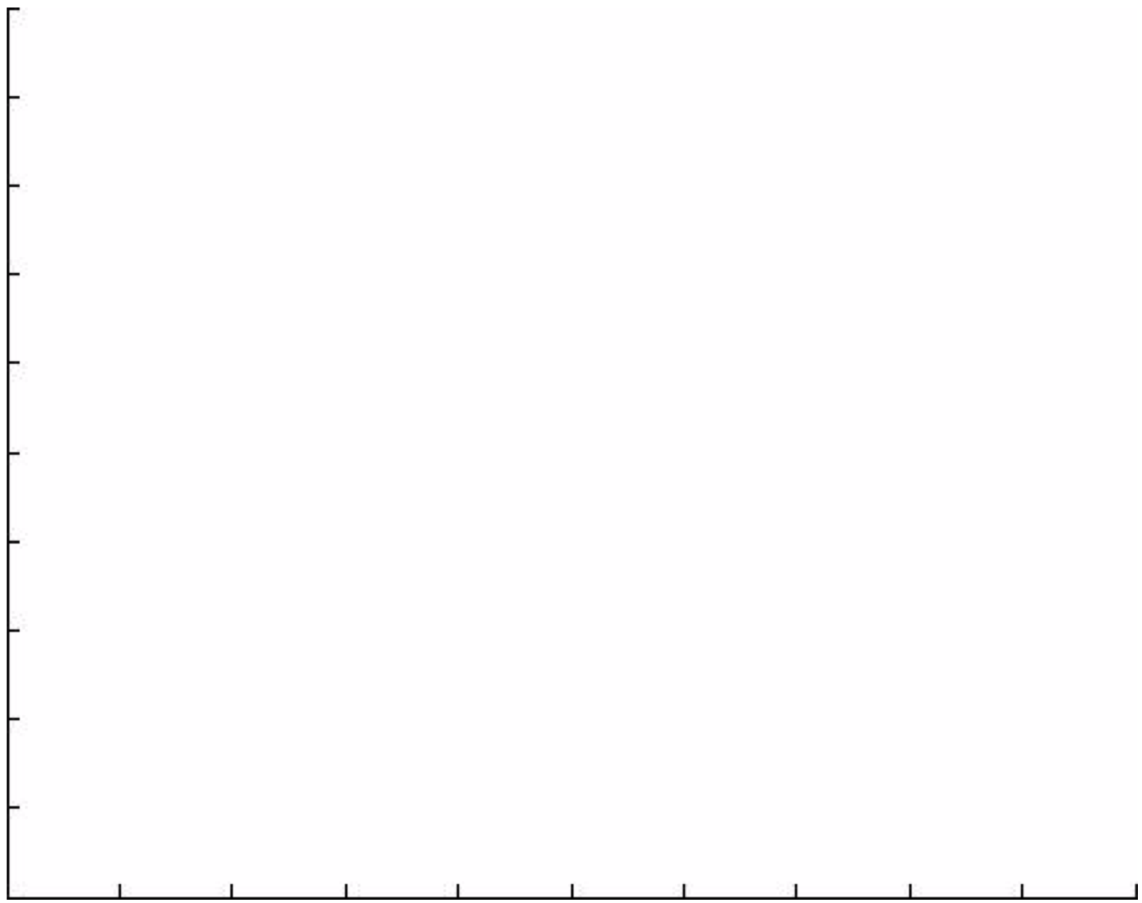
2. Residual evaluation:
   - Modelling errors lead to non-zero residuals even during fault-free operation.
   - The evaluation of residuals aims at determining if the magnitudes of the residuals are significant.
   - In its simplest form, comparison of the actual value of the residual against a fixed threshold.

$$\boldsymbol{r}(t) = 0 \rightarrow FD(t) = 0$$

$$r_i(t) \neq 0 \rightarrow FD(t) = 1$$

- The thresholds values are selected to satisfy some criterion related to false alarms and undetected faults.

An Example of FD based on parameter estimation



FAULT DIAGNOSIS

How to diagnose (isolate and identify) faults by analyzing data?
Which are the main solutions?

Fault detection, isolation, identification

- Tasks related to faults:
  - Detection – Determination of the faults present in a system and the time of detection.
  - Isolation – Determination of the kind, location and time of detection of a fault. Follows fault detection.
  - Identification – Determination of the size and time-variant behaviour of a fault. Follows fault isolation.

Diagnosis

- FD = Fault Detection, FDI = Fault Detection and Isolation, FDD = Fault Detection and Diagnosis.
- In practice, the use of the term diagnosis is very general: sometimes, it can include detection; on the other hand, identification can be considered or not.

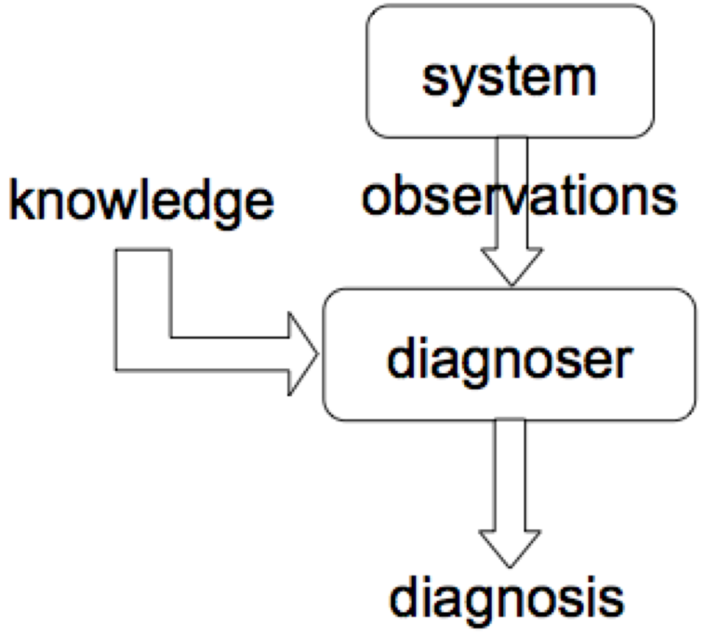
Fault detection, isolation, identification

- The importance of the previous tasks (detection, isolation, identification) is in general decreasing, but the importance of each task is given by the particular application.
- Fault detection is normally of great importance and it has to be implemented on-line; fault isolation can be less important and it may be sufficient its implementation off-line; fault identification can be implemented or not.
- Fault detection is related to safety, fault isolation is related to availability, fault identification is related to predictive maintenance and fault-tolerant control.

Operating principle

- Fault diagnosis relies on comparing the observed behaviour of the monitored system with the a-priori knowledge about it.

✓ *Remarks:*
  - ✓ *Real-time operation.*
  - ✓ *Knowledge for normal and faulty operations is needed (knowledge about normal operation is enough for FD)*
  - ✓ *All faults have to affect the observed variables (detectable faults) in a different way (isolable faults).*

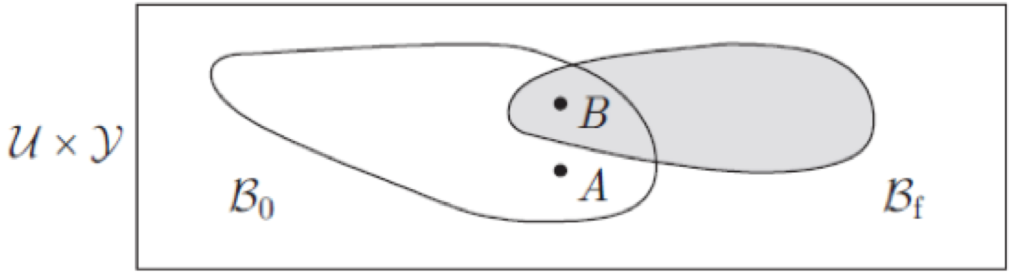


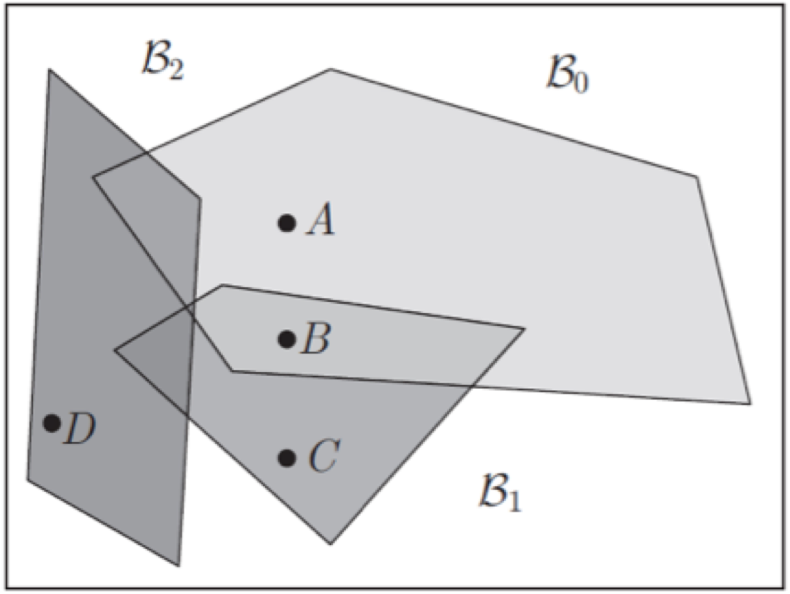
General formulation

- **Notation**:
  - (U,Y) is the sequence of inputs/outputs.
  - B_0 is the normal system behaviour, B_{fi} is the system behaviour under the effect of fault f_i.
- Detection – if the sequence (U,Y) is inconsistent with the behaviour B_0 then the presence of a fault is concluded: (U,Y) not in B_0 $\rightarrow$ fault
- Isolation – if the sequence (U,Y) is consistent with the behaviour B_{fi} then the presence of fault f_i is concluded. $(U,Y) \in B_{fi} \rightarrow$ fault f_i

Detectability and isolability

- Normal and faulty behaviours can be distinguishable for some sequences of system inputs and undistinguishable for others.



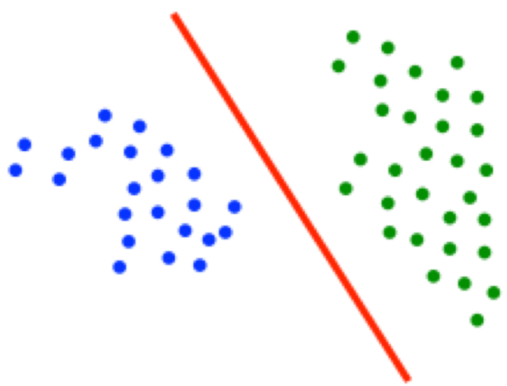


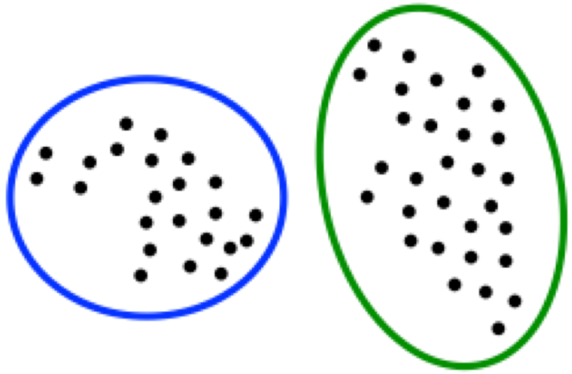
Research in fault diagnosis

- Several independent research communities have addressed the fault diagnosis problem, considering different types of systems, using different problem setups, different nomenclature, different tools,...
  - **AI community** - using techniques such as expert systems, qualitative models, consistency-based diagnosis, ...
  - **FDI community** - using mathematical models, statistical techniques, signal processing, ...
  - **Others**, e.g., chemical engineering community
- Current trend - Approaching between communities, interchange of ideas, combined use of techniques.

General classification of methods (I)

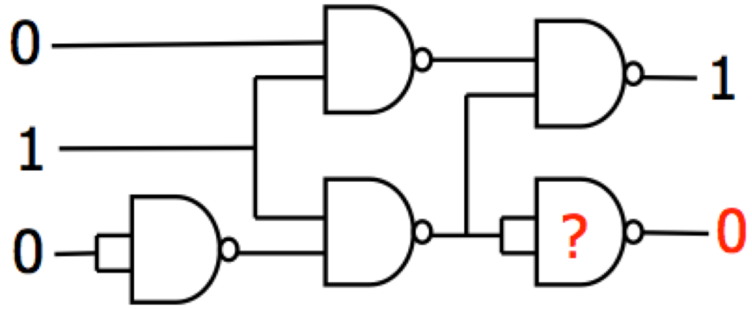
- Classification of diagnosis methods according to the a-priori available knowledge about the system:
  - Data-driven methods:
    - Data about system operation is available, classified according to the faults or not.
    - It is used to train a classifier: statistical classifier, neural network, support vector machine,...
    - On-line: classifier fed with incoming data...

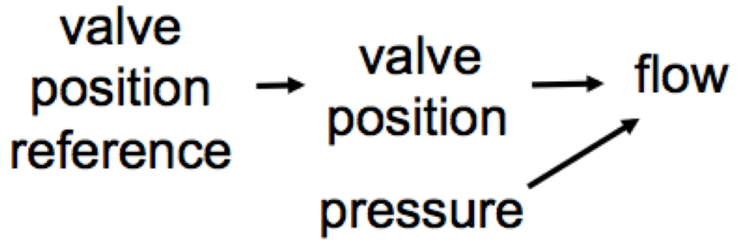




General classification of methods (II)

- Model-based methods:
  - A deeper knowledge of the system is captured in a model that supports diagnostic reasoning.
  - Structure and behaviour, causality, first principles Qualitative, quantitative, analytical, statistical
  - Compare predictions and measurements to trigger the diagnostic reasoning.

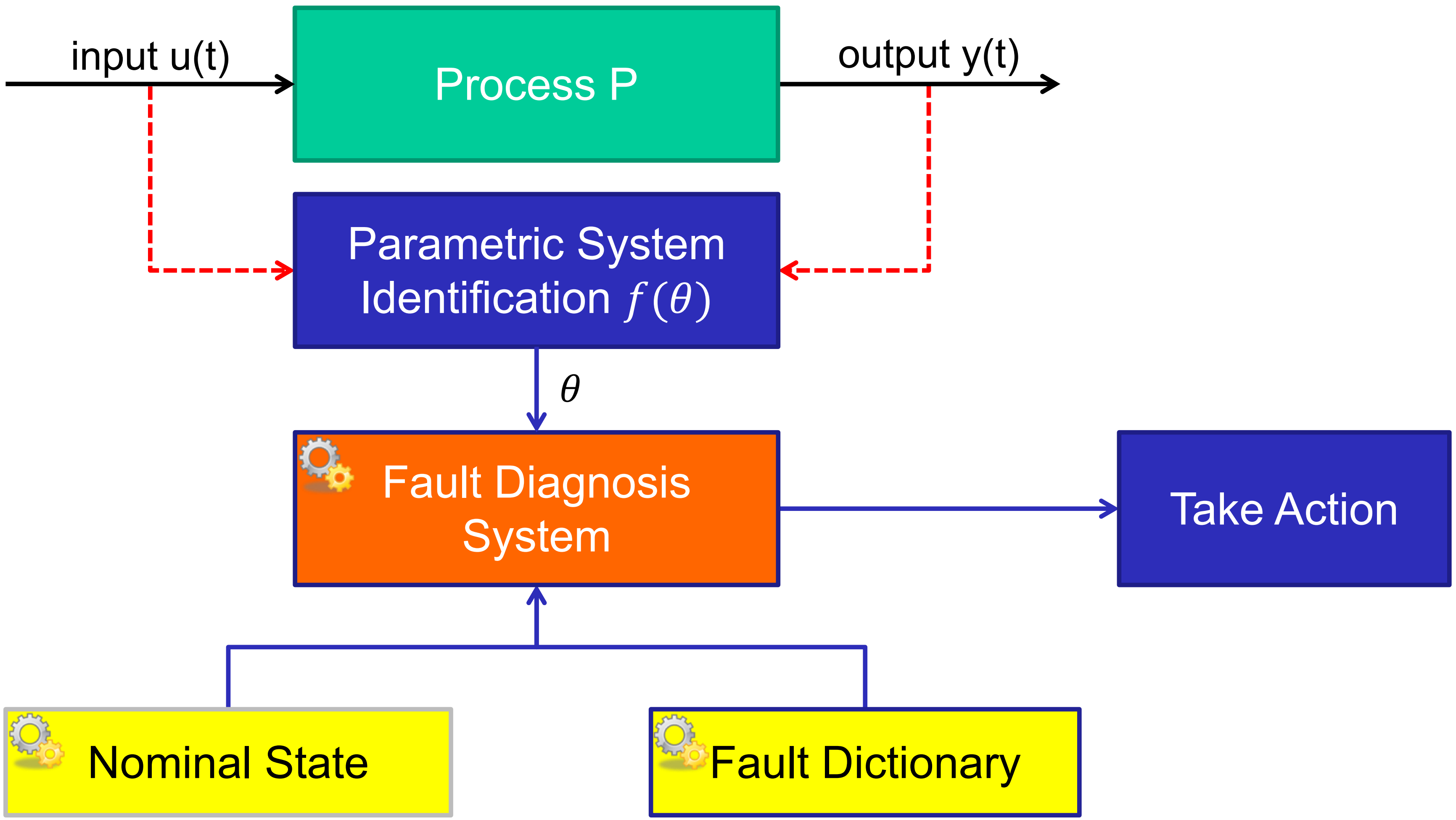




Data-driven vs. model-based

- **Data-driven vs. model-based**:
  - Data-driven methods:
    - Requirement of real data for faulty operation.
    - Are not able to manage new situations.
    - Applicable to real complex and large scale systems (non-linear, many measured variables).
  - Model-based methods:
    - Availability of real data in advance is not required.
    - They are capable to face not previously experimented situations, including multiple faults.
    - Limitations in their application to complex systems.

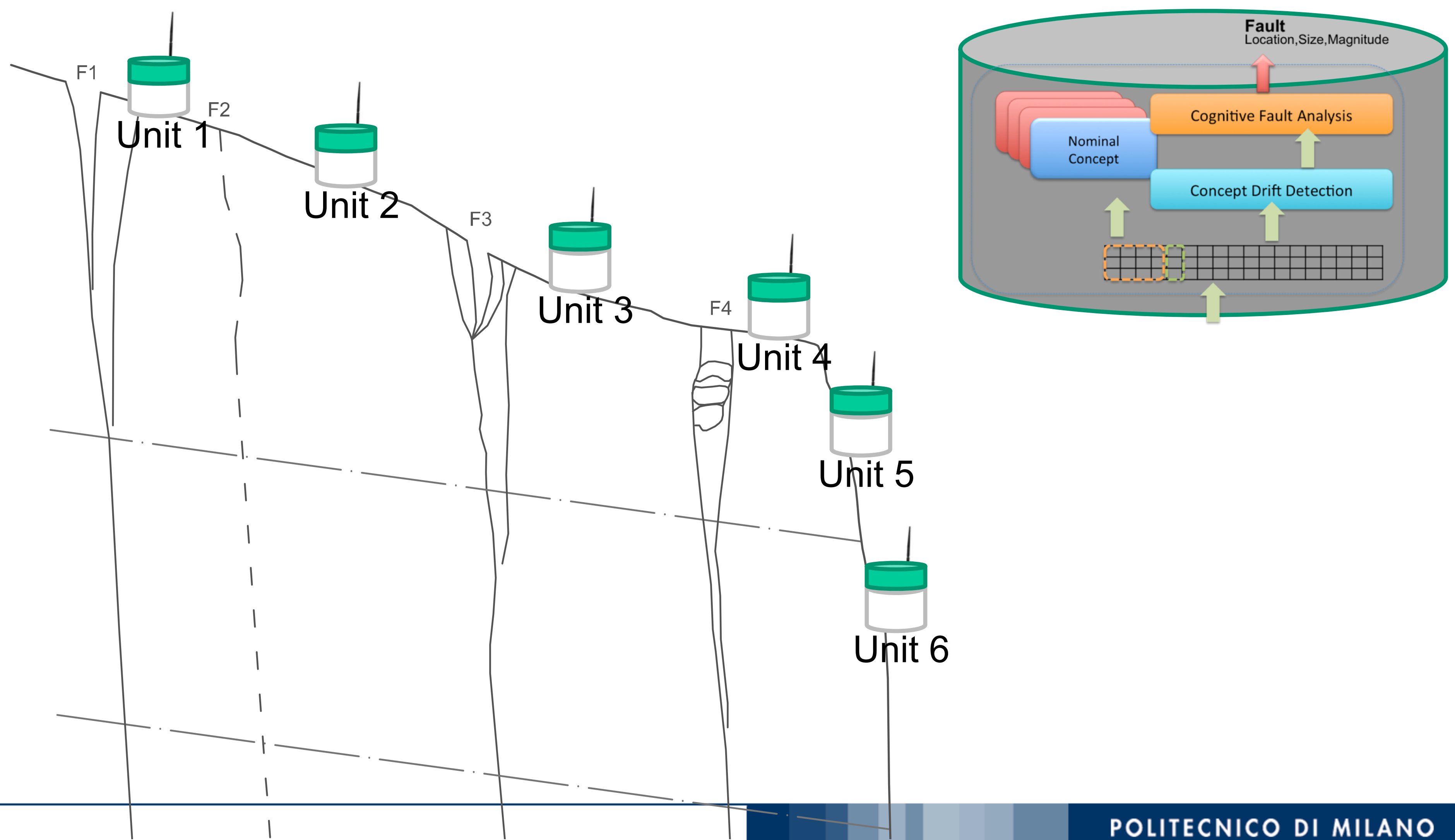
Fault Isolation and Identification



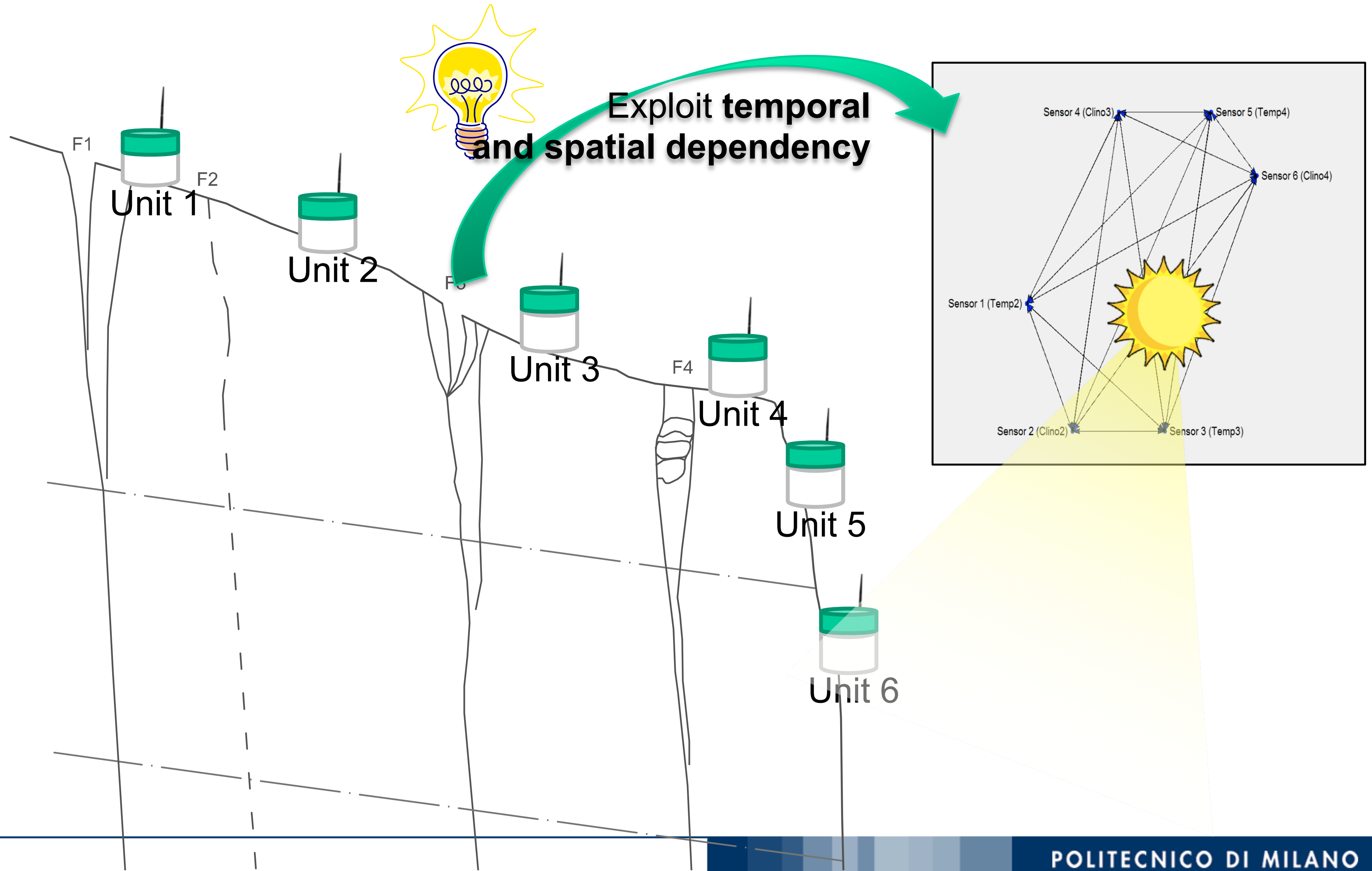
COGNITIVE FAULT DETECTION AND DIAGNOSIS SYSTEMS

Which are the advantages of
cognitive mechanisms
for fault detection/diagnosis?

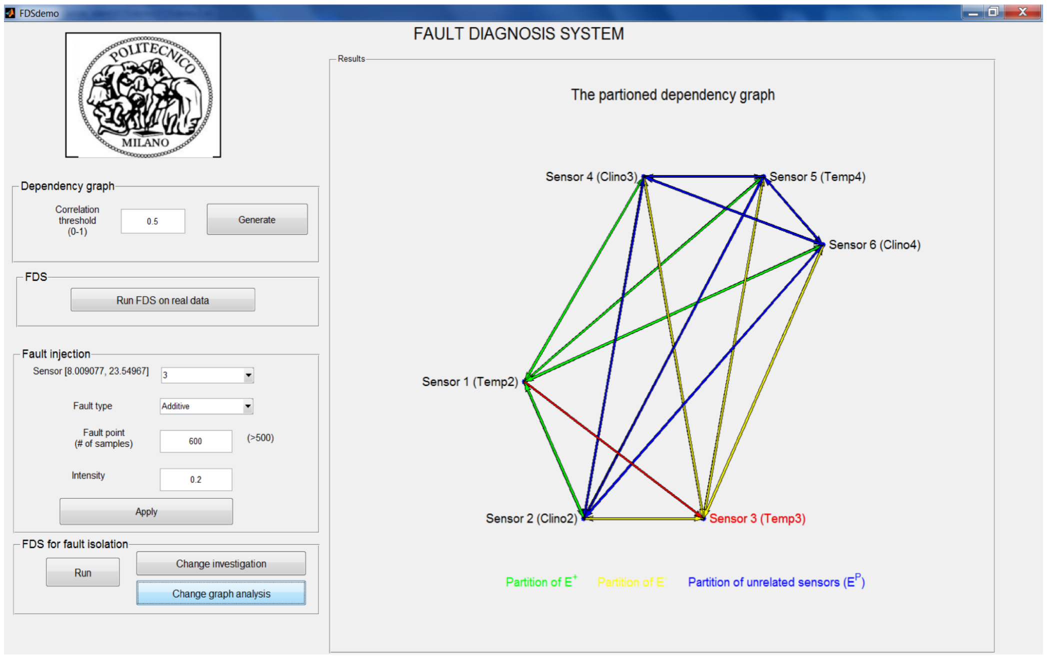
A cognitive fault detection and diagnosis system for distributed sensor networks

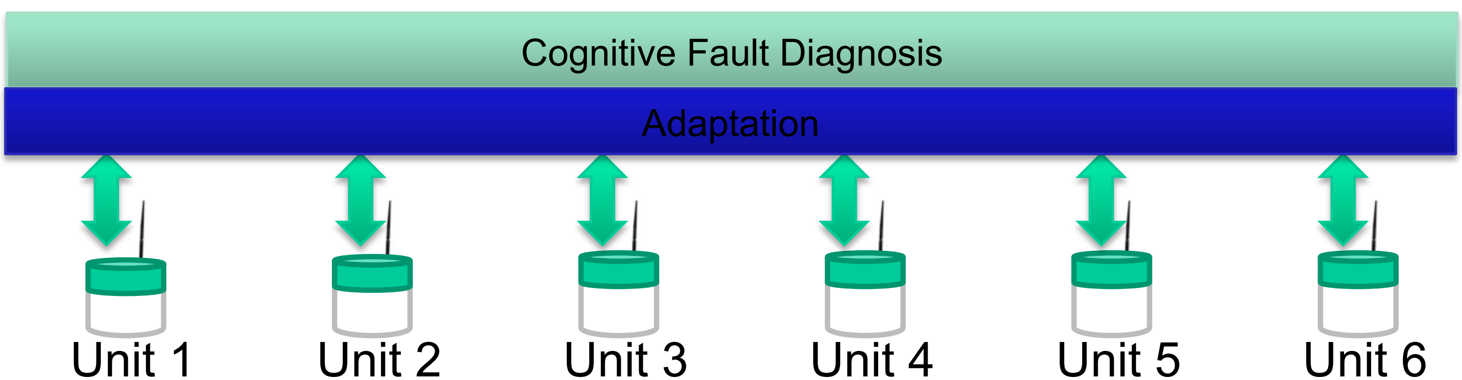


A cognitive fault detection and diagnosis system for distributed sensor networks

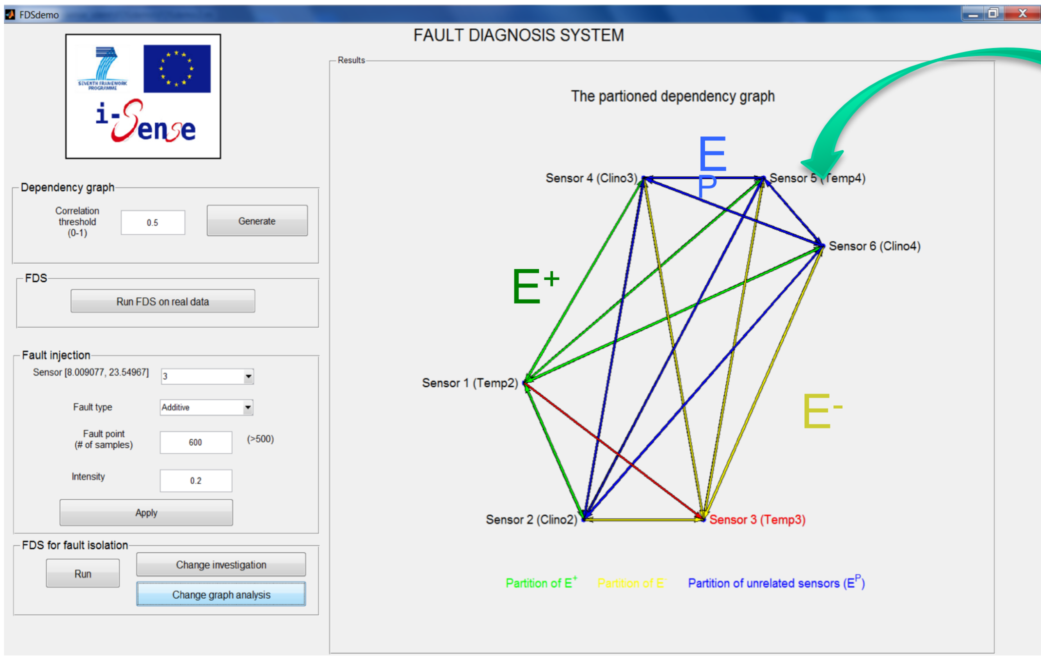


A cognitive fault detection and diagnosis system for distributed sensor networks





Cognitive Fault Detection and Isolation: HMM-based and the dependency graph

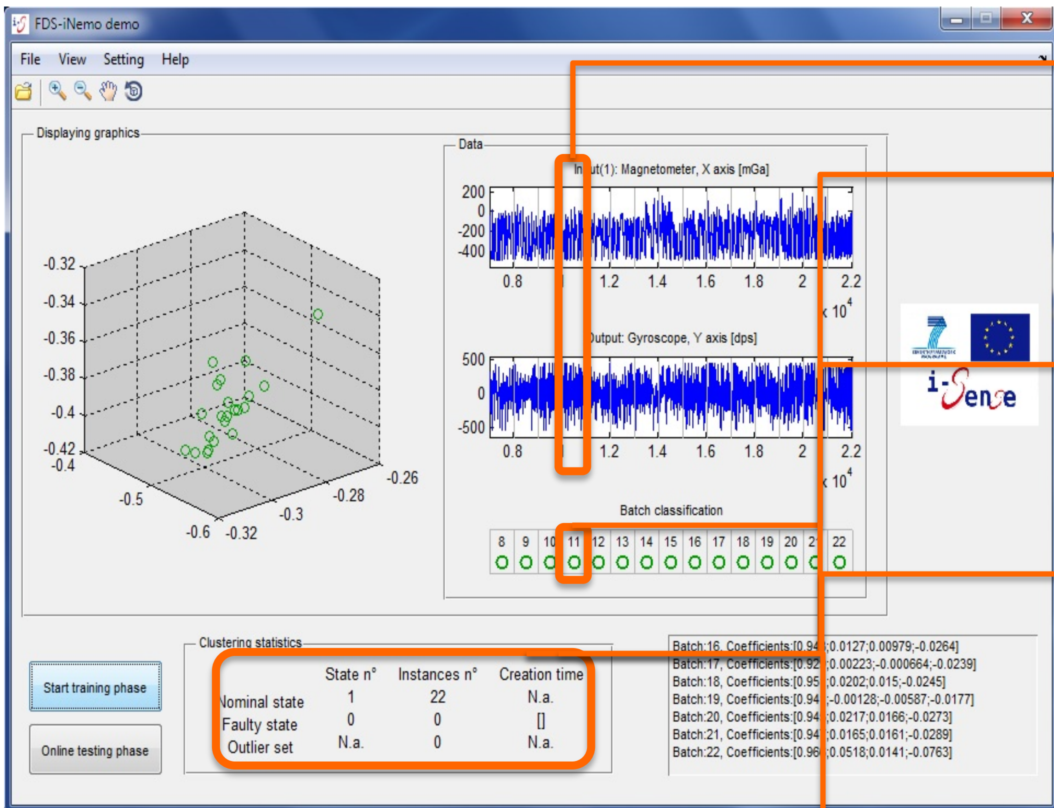


HMM-CDT associated with $\mathcal{H}_{\{(i,j),(u,v)\}}$ detected a change in the $\bar{s}$-th data window;
Partition the graph into sets E^+, E^- and $E^{\mathcal{P}}$ according to Eq. (7), (8), (9) ;
Compute S^+, S^-, $S^{\mathcal{P}}$ according to Eq. (10), (11) and (12);
Compute T^+, T^-, $T^{\mathcal{P}}$ according to Eq. (13), (14) and (15);
if $S^{\mathcal{P}} < T^{\mathcal{P}}$ **then**
  Change in $\mathcal{P}$ detected;
else
  if $S^+ < T^+$ *and* $S^- < T^-$ **then**
    Model Bias in $\mathcal{H}_{\{(i,j),(u,v)\}}$ detected;
  else
    if $S^+ < T^+$ **then**
      Fault at sensor $X_{(i,j)}$ detected;
    else
      Fault at sensor $X_{(v,u)}$ detected;
    end
  end
end

$$S^+ = \frac{1}{\sum_{E^+} w_{\{(i,j),(u,v)\}}} \sum_{E^+} w_{\{(i,j),(u,v)\}} l_{\{(i,j),(u,v)\}}(\bar{s}); \quad (10)$$
$$S^- = \frac{1}{\sum_{E^-} w_{\{(i,j),(u,v)\}}} \sum_{E^-} w_{\{(i,j),(u,v)\}} l_{\{(i,j),(u,v)\}}(\bar{s}); \quad (11)$$
$$S^{\mathcal{P}} = \frac{1}{\sum_{E^{\mathcal{P}}} w_{\{(i,j),(u,v)\}}} \sum_{E^{\mathcal{P}}} w_{\{(i,j),(u,v)\}} l_{\{(i,j),(u,v)\}}(\bar{s}). \quad (12)$$

$$T^+ = \frac{1}{\sum_{E^+} w_{\{(i,j),(u,v)\}}} \sum_{E^+} w_{\{(i,j),(u,v)\}} T_{h,\{(i,j),(u,v)\}}; \quad (13)$$
$$T^- = \frac{1}{\sum_{E^-} w_{\{(i,j),(u,v)\}}} \sum_{E^-} w_{\{(i,j),(u,v)\}} T_{h,\{(i,j),(u,v)\}}; \quad (14)$$
$$T^{\mathcal{P}} = \frac{1}{\sum_{E^{\mathcal{P}}} w_{\{(i,j),(u,v)\}}} \sum_{E^{\mathcal{P}}} w_{\{(i,j),(u,v)\}} T_{h,\{(i,j),(u,v)\}}. \quad (15)$$

Learning the Fault Dictionary



Evolving Clustering Algorithm

```
Train a cluster for each nominal state Ψ = {Ψ_j}_{j=1}^ψ
Set Φ = ∅ (φ = 0) and O = ∅ (o = 0)
Set α_s, ε_t
while A new θ̂_{N,i} is calculated do
  j* ← S(θ̂_{N,i}, Ψ, α_s)
  if j* ≠ 0 then
    if T(i, Ψ, j*, η_t) = 1 then
      Ψ_{j*} ← I_t(θ̂_{N,i}, i, Ψ_{j*})
    else
      Ψ_{j*} ← I_s(θ̂_{N,i}, i, Ψ_{j*})
    end if
  else
    if φ > 0 then
      j* ← S(θ̂_{N,i}, Φ, α_s)
      if j* ≠ 0 then
        if T(i, Φ, j*, η_t) = 1 then
          Φ_{j*} ← I_t(θ̂_{N,i}, i, Φ_{j*})
          for e ∈ P(O), e = (θ, k, 1) do
            if j* ← S(θ, Φ, α_s) then
              if j* ≠ 0 then
                if T(k, Φ, j*, η_t) = 1 then
                  Φ_{j*} ← I_t(θ, k, Φ_{j*})
                else
                  Φ_{j*} ← I_s(θ, k, Φ_{j*})
                end if
                O ← D_o(θ, k, O)
              end if
            end if
          end for
          for j ∈ {1, ..., j* − 1, j* + 1, ..., φ} do
            Φ ← I_me(Φ, Φ_{j*}, Φ_j)
          end for
        end if
      else
        Φ_{j*} ← I_s(θ̂_{N,i}, i, Φ_{j*})
      end if
    else
      O ← I_o(θ̂_{N,i}, i, O)
      if S_pr(Ψ ∪ Φ, O, α_c) = 1 then
        O' ← PS(O)
        if S_po(O') = 1 then
          (O, Φ) ← NC(O, O', Φ)
        end if
      end if
    end if
  end if
end while
```

Nominal cluster characterization

Faulty dictionary creation

Faulty dictionary characterization

Faulty dictionary management

FAULT ACCOMMODATION

How to manage and react to a fault?

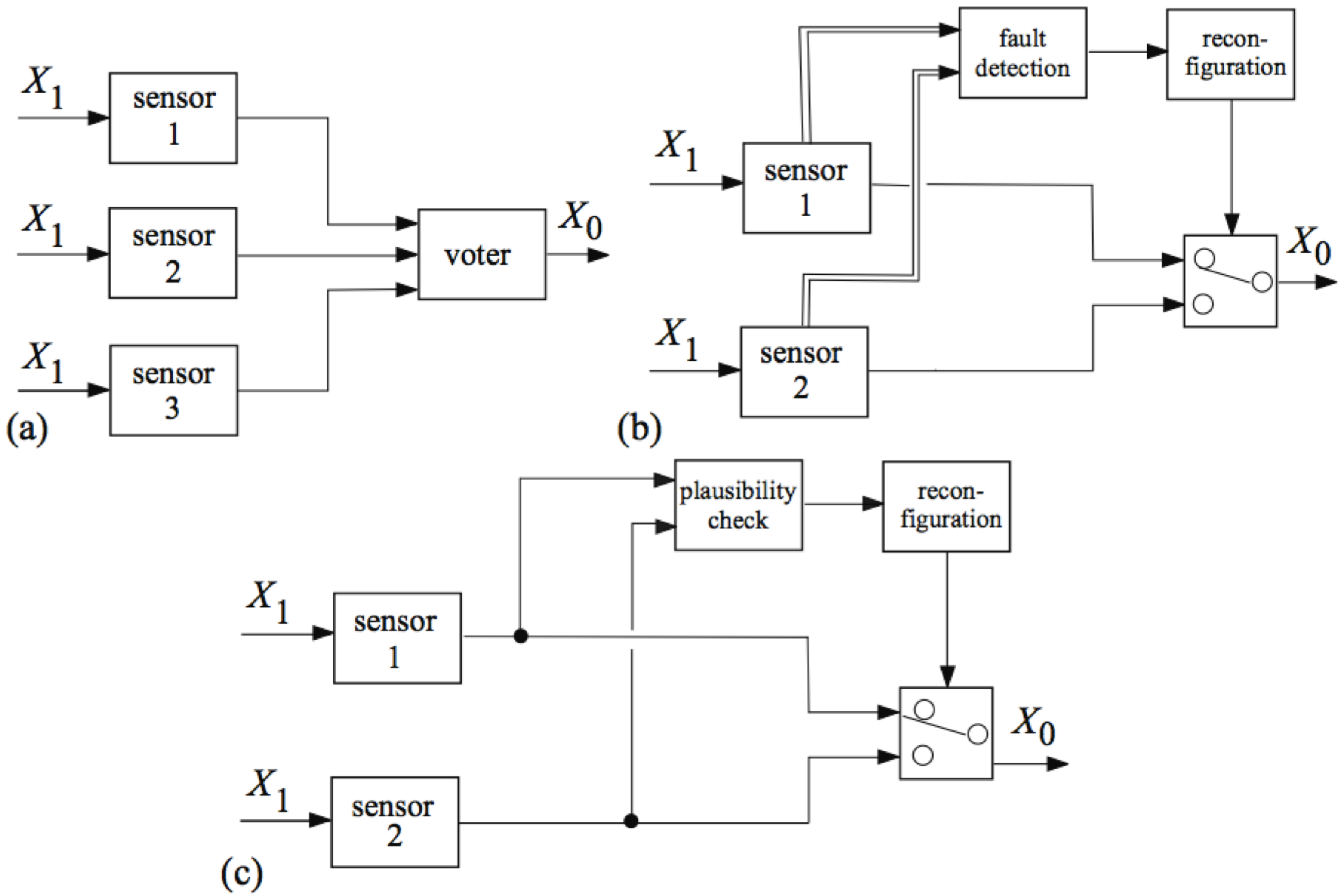
Fault-tolerant components

- High-integrity systems require a comprehensive overall fault-tolerance by **fault-tolerant components** and control
- This means the design of fault-tolerant
  - ***sensors***
  - actuators
  - process parts
  - computers
  - communications
  - control algorithm
- Examples of components with multiple redundancy are known for aircraft, space and nuclear power systems

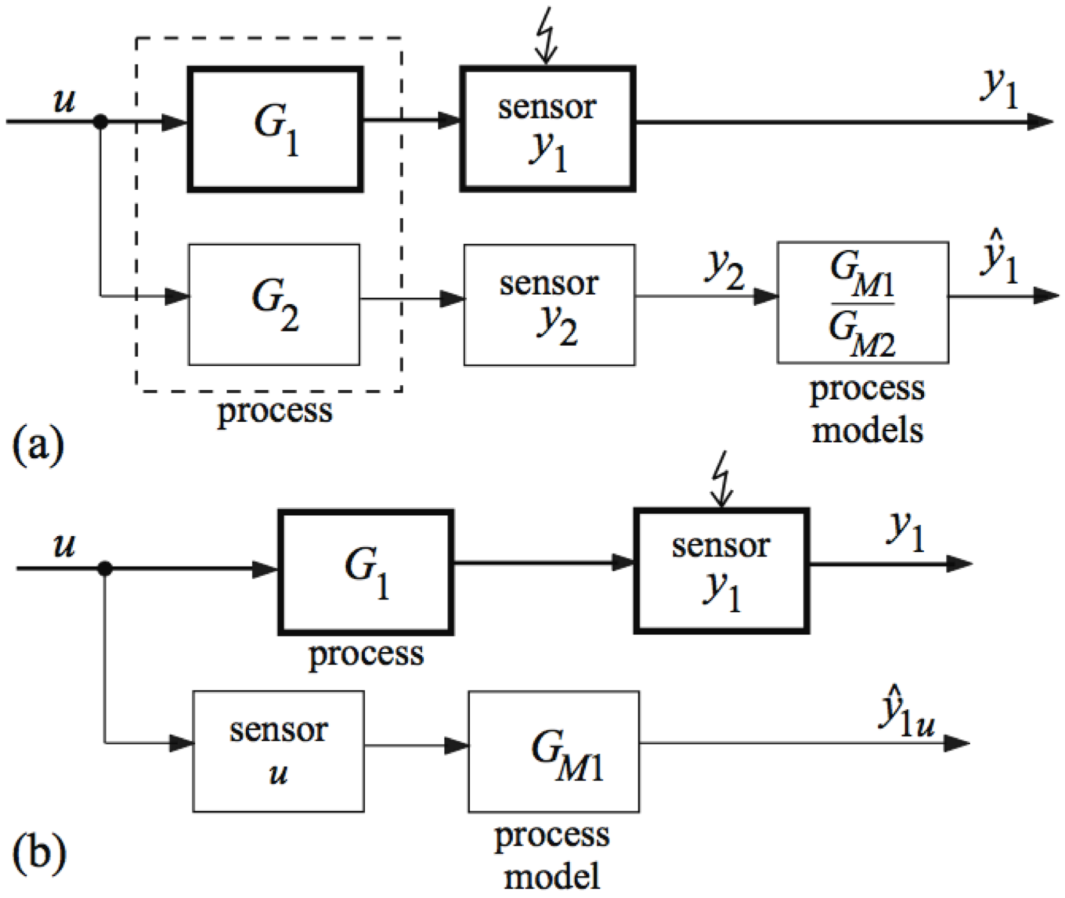
Fault-tolerant sensors

- A fault-tolerant sensor configuration should be at least fail-operation for one sensor fault
- This can be obtained by applying
  - Hardware redundancy with the same type of sensor
  - Analytical redundancy with different sensors and process models

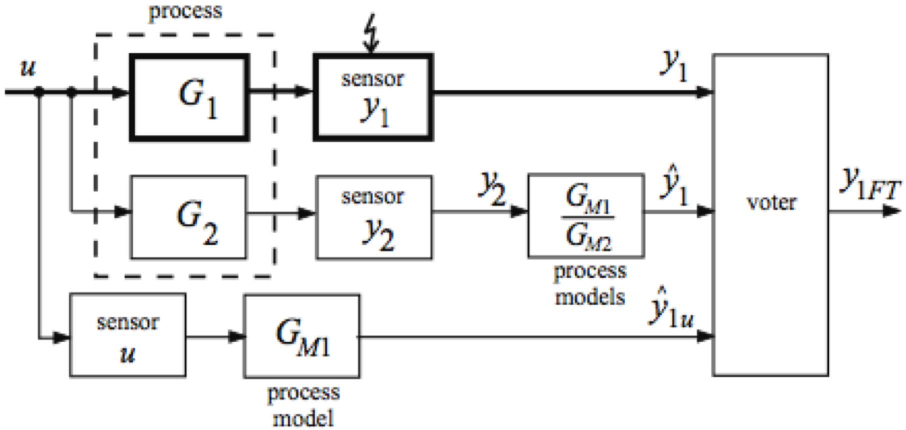
Hardware sensor redundancy



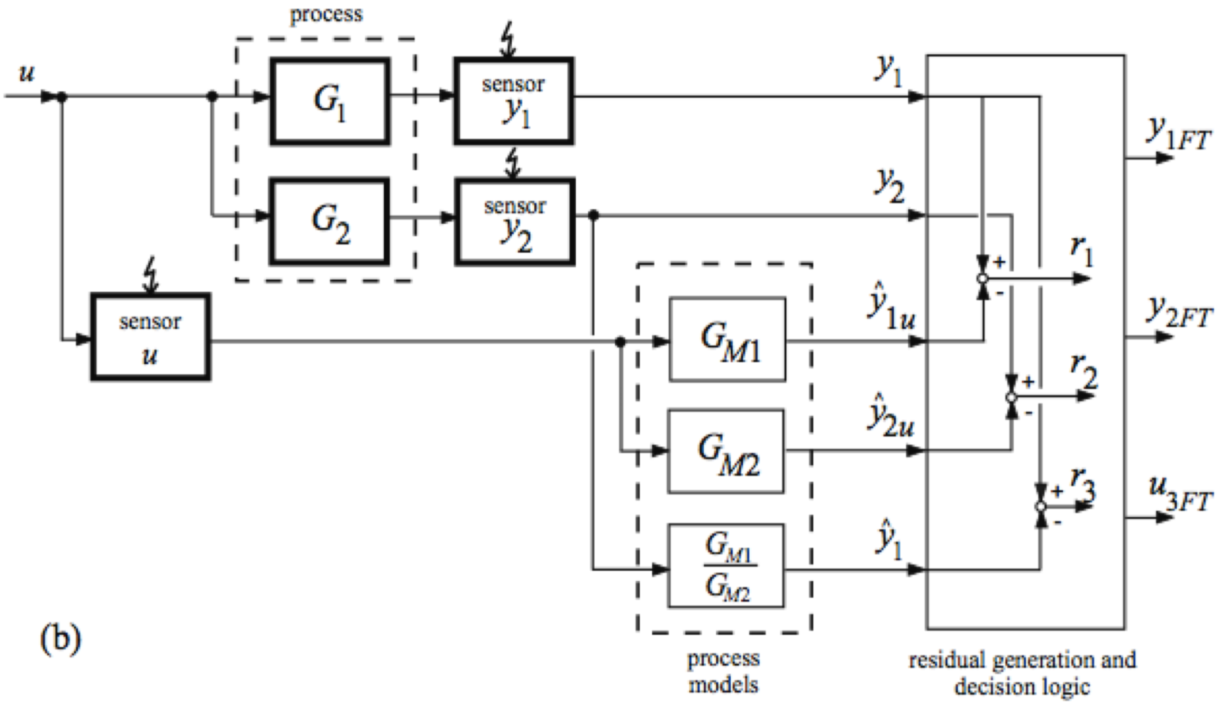
Analytical sensor redundancy (1)



Analytical sensor redundancy (2)



(a)



(b)

Comments

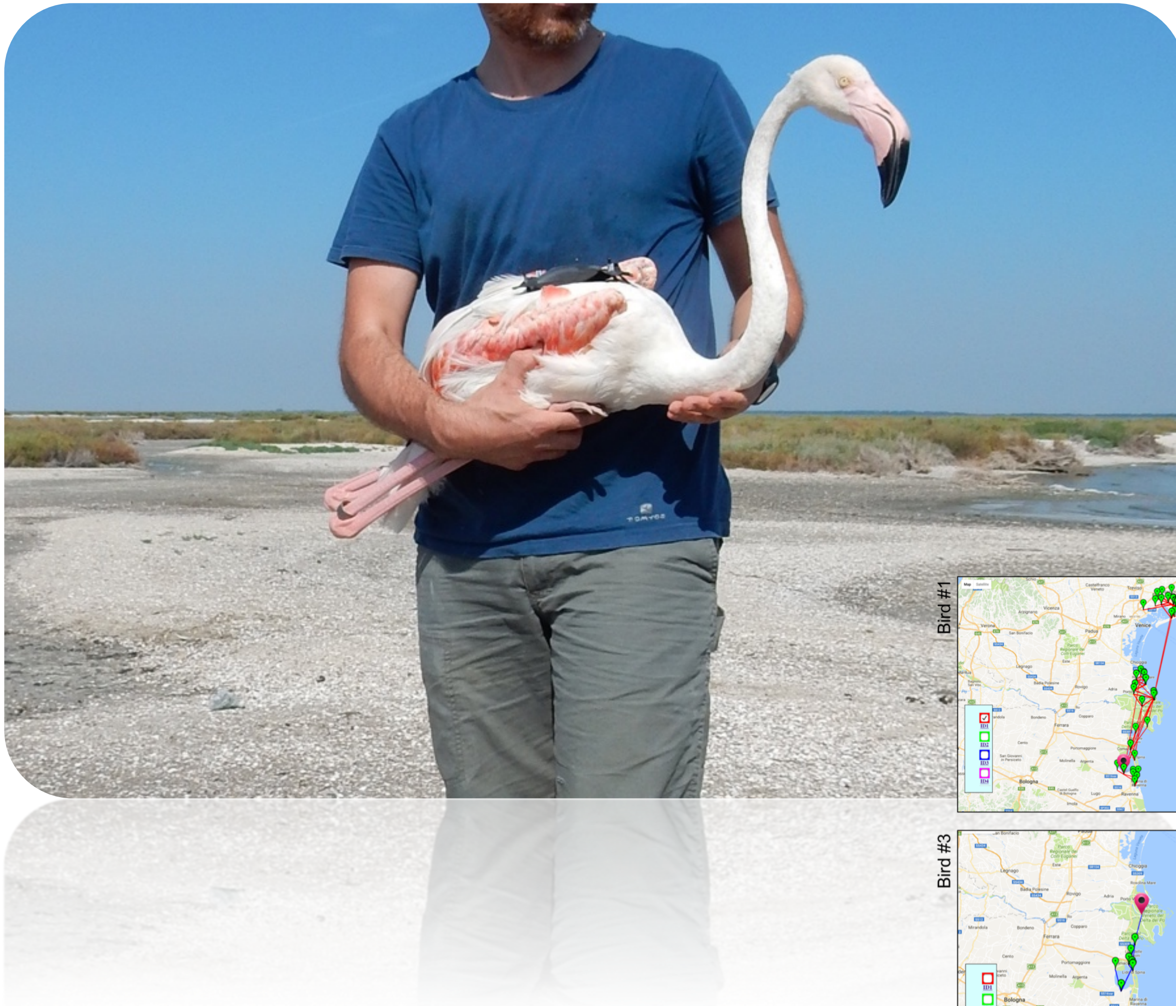
- **For both hardware and analytical sensor redundancy without fault detection** for individual sensors**, at least three measurements must be available** to make one sensor fail-operation
- If the sensor (system) has built-in fault detection mechanisms (integrated self-test or self-validating), two measurements are enough
- **This means that by methods of fault detection, one element can be saved**

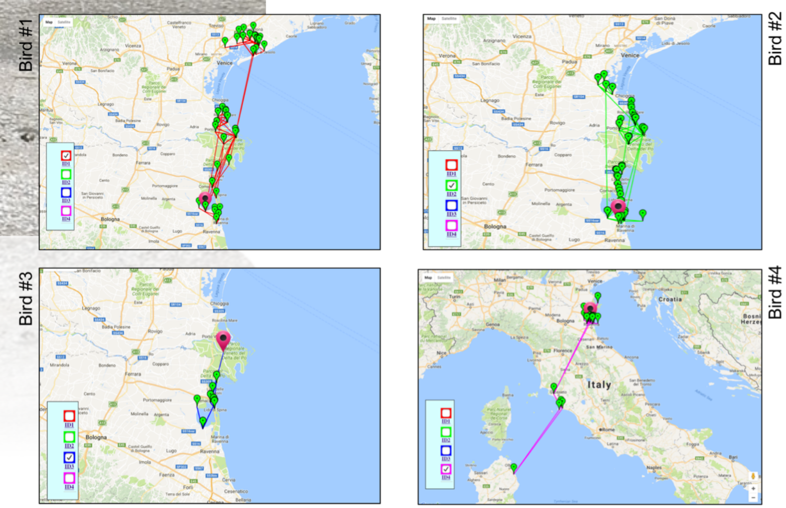
CASE STUDY:
MODEL/DATA ANALYSIS

Case Study #1: Rock collapse monitoring system

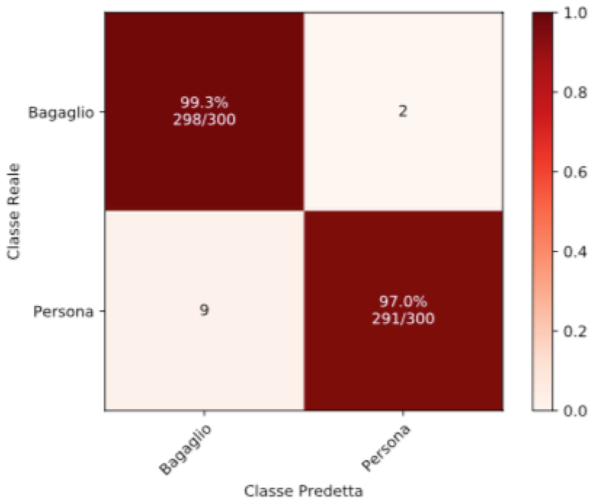
Case Study #2: Monitoring a datacenter

Case Study #3: Tracking wildlife animals





Case Study #4: detecting the presence of people in the airport conveyor belt



(a) Matrice di confusione

(b) Bagagli classificati come Persone

Points to be discussed (1/3)

Detecting, Isolating and identifying faults by analyzing data

1. Which is the information that is collected for the system and how is it processed for the application purposes?
2. Which are the types of faults that could affect the system?
3. How can these faults affect acquired data? Which is the effect on the application?

Points to be discussed (2/3)

How to detect faults by analyzing data?

4. Which are the techniques that could be considered for fault detection?
5. How does the technological implementation of the system influence the selected fault detection technique? Where will it be executed?

Points to be discussed (3/3)

Detecting, Isolating and identifying faults by analyzing data

6. How critical is fault isolation? Why?
7. Fauld identification and the need of a fault dataset. How to deal with it in the considered case study?
8. Is fault accommodation relevant? Why? How to implement it?

Available theses in the field

Four available theses

1. **Spacecraft subsystems** anomaly detection with machine learning techniques
2. **On-board machine learning system** for filtering relevant information
3. **Privacy-preserving** machine and deep learning
4. **Embedded and Edge AI** for Internet-of-Things and Machine Learning