Multiple decision trees to diagnose a transient state of dynamic systems. Application to a DC motor.

Antonio J. Suárez, Pedro J. Abad
Dpto de Ingeniería Electrónica,
Sistemas Informáticos y Automática
Universidad de Huelva
abadhe@diesia.uhu.es, asuarez@diesia.uhu.es

Rafael M. Gasca, Juan A. Ortega
Dpto de Lenguajes y Sistemas
Informáticos
Universidad de Sevilla
gasca@lsi.us.es, ortega@lsi.us.es

Abstract

In this paper, a novel methodology is proposed to diagnose a transient state of a dynamic system using supervised learning. It is composed by two steps: one off-line process and another on-line process. The off-line phase begins gathering data from the system, both when it is running free of fault and when the system is running in each fault mode. Also, it is possible to generate these data from Monte Carlo simulations of a system model. A segmentation and normalization algorithm is used to reduce the large amount of gathered data. The final step of the off-line process is the generation of a decision tree by a classification tool. The on-line process of the methodology consists in evaluating a new reading of the system sensors with the generated decision trees. The system diagnosis is the result of this evaluation which has a linear computational cost due to the simplicity of the decision trees. In order to improve diagnosability problems of this methodology, it is proposed a new solution in this work. Instead of generating only one decision tree, a different decision tree is generated for each fault mode and free of fault mode. Therefore multiple possibilities of diagnosis can be offered for a given behaviour of dynamic system. Methodology has been applied to diagnose a DC motor. Eight different faults have been considered and the results have been discussed including diagnosability conflicts.

1 Introduction

Inside the Artificial Intelligence techniques, data mining is about solving problems by analyzing data already present in databases. Data mining is defined as the automatic process of discovering patterns in data. One of the fields of Data Mining is the Machine Learning, which is defined as the ability for a computer system to generate new knowledge based on its past experiences.

A long variety of techniques, coming from Artificial Intelligence, has been applied to diagnosis field from their beginning. Along this time, a great number of approximations have been proposed from different points of view. In [Cordier *et al.*, 2000] a comparative between the communities FDI and DX for model based diagnosis can be found. One of the fields that have been widely investigated is the diagnosis of dynamic systems. These types of systems are very difficult to diagnose due to the great amount of components, the small set of observable variables, the interactions among their components, and the habitual presence of a control system which could hide the presence and identification of the faults. Machine Learning techniques have been applied to this field, from a decade ago [Feng, 1992] to current years [Roverso, 2003].

Induction motors are very common in industry due to their simplicity, rugged structure, cheapness and easy maintainability. It is very usual that these motors were involved into larger industrial systems. Fault detection and diagnosis of these motors are very important when they are working in on-line monitor conditions. Due to dynamical conditions of these systems, it is crucial to diagnose the faults quickly and precisely. Several techniques has been applied in order to diagnose these motors: techniques based on the signal analysis [Schoen *et al.*, 1995], based on the dynamic modelling of the motor [Chan *et al.*, 1999] and knowledge based techniques. Inside the knowledge based techniques, many tools have been used: expert systems [Filippetti *et al.*, 1992], neuronal networks [Liu *et al.*, 2000] or automatic classification [Hajiaghajani *et al.*, 2004].

The aim of this paper is to use the power of the automatic learning to diagnose dynamic systems with a minimal amount of sensors. A complete methodology is proposed for this purpose and it is applied to a dynamic system involving a DC motor. This is an important improvement to the methodology presented in a previous work [Abad *et al.*, 2002].

A improvement in the methodology is proposed with the aim of treating no-diagnosable behaviours. System behaviour is no-diagnosable when it could have more than one possible diagnostic. The improvement proposes to offer all possible diagnostics by generating multiple decision trees.

The complete methodology is illustrated with simulations of a DC motor. Eight different faults have been considered and the results have been discussed including diagnosability conflicts.

2 General methodology for diagnosing dynamic system by classification

2.1 Assumptions

The following assumptions are needed to consider in the present methodology:

- The system will be alternating between steady states and transient states.
- The number of transient states is finite and they are always produced between two known set points.
- When the system begins a transient state, if a fault exists it will be present fully since beginning of the transient to the steady state.
- Faults only will be detected at transient states, independently of the time instant in which the fault occurs.

2.2 Definitions an notation

In order to clarify the phases of the proposed methodology we will need the following definitions:

Definition 1. Trajectory. A *trajectory* can be defined as a function s from a set of time instants T to a set of a system sensor values $V \subseteq I\!R$.

$$s : T \rightarrow V$$

where the values of T are to regular intervals, from the initial instant t_0 to the final instant t_n

$$T = [t_1, t_1 + \Delta T, t_1 + 2\Delta T, ..., t_1 + (n-1)\Delta T, t_n]$$

t_1 denotes the beginning of a transient state, and t_n denotes the establishment point. The *i* element of a *trajectory* will be denoted as s_i. Thus a *trajectory j* will be represented as:

$$j = [j_1, j_2, ..., j_n]$$

Definition 2. Labelled Trajectory. A *labelled trajectory* is defined as a *trajectory* in which a label has been added. This label will be situated as the last element in the *trajectory* and it represents the conditions in which the *trajectory* has been obtained. Thus a *labelled trajectory* jl will be represented as:

$$jl = [jl_1, jl_2, ..., jl_n, LABEL]$$

where label is a string(discrete value).

Definition 3. Trajectories Database. A *trajectories database* is a collection of *trajectories*, belonging to the same system and corresponding to the same time instant. All *trajectories* must have the same numbers of elements. This collection of *trajectories* must be stored in a file, where each *trajectory* is represented by one line.

Definition 4. Labelled Trajectories Database. A *labelled trajectories database* is a *trajectories* database where all *trajectories* are *labelled trajectories*.

2.3 General methodology

The proposed general methodology has two phases clearly different:

1. The first phase (Figure 1) is developed off-line. In this phase the main objective is to obtain a set of decision trees that characterize the system behaviour in the different fault modes which want be diagnosed.

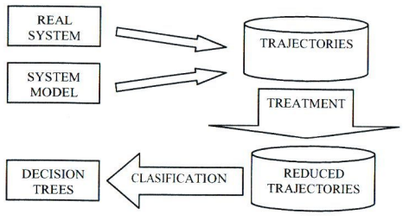

Figure 1: Off-line phase of the proposed methodology

2. The second phase (Figure 2) is the diagnosis phase itself. It is developed on-line, while the system is being monitored. In this phase the generated trees are evaluated with the system sensor measurements in order to obtain a system diagnostic.

Figure 2: On-line phase of the proposed methodology

Off-line phase

The first task of this phase is to select the set of faults that would like to detect, and the transient states at which these faults will be detected. Since the diagnosis process will be performed at known transient states, the process below described must be performed for all transient states in which the diagnosis process must be performed. In this phase there are three consecutive steps: data gathering, data treatment and decision trees generation.

Data gathering In this step the goal is to obtain a collection of system *trajectories*, for each fault mode and for the free fault running. Obtaining the free fault *trajectories* is not a problem usually, but gathering the fault *trajectories* is another matter. For this purpose, two different options are considered:

- When it is possible to generate the modes, without damaging the system, they are provoked and the *trajectories* are gathered. This is, for example, the running mode in which a system component is accidentally disconnected. Also, it is possible to use this method when the system will be built in a series production. In this scenario, a set of prototypes could be damaged to provoke the faults and the *trajectories* could be obtained. This way, the diagnostician will be useful for all products in the series.
- When the fault cannot be provoked in the real system (because the system could be damaged or because it is impossible to stop the running system), a system model must be generated. Then, fault


```
Algorithm Seg_TS = Sliding_Window(T ,max_error)
anchor = 1;
while not finished segmenting time series
  i = 2;
  while (calculate_error(T[anchor: anchor + i ]) < max_error
        and (T[anchor + i ]) < length(T))
      i = i + 1;
  end;
  Seg_TS =concat(Seg_TS,create_segment(T[anchor:anchor+(i-1)]);
  anchor = anchor + i;
end;
```

Table 1: Segmentation on-line algorithm

modes, which cannot be provoked in the real system, are simulated in the model, in order to obtain the corresponding *trajectories*.

These two options are not excluding, and both can be used at the same time. For each collected *trajectory*, a representative label will be added. This label represents the running mode of the system in which the *trajectory* has been obtained: a concrete fault or free fault. This way, each *trajectory* is transformed in a *labelled trajectory*. All *labelled trajectories* are stored in a labelled database. The result is a *labelled trajectories database*.

Data treatment *Labelled trajectories database* contains all information gathered from the system sensors or generated by the simulations. Usually, most of these data are not relevant to distinguish among different behaviours. For example, data of *trajectories* which are closer to the steady state will be very similar, even if *trajectories* belong to different behaviours.

In this step, the aim is to reduce the amount of data that represents each *trajectory*. It is performed by means of a segmentation algorithm. The goal of this algorithm is to characterize a *trajectory* by a succession of linear segments. This succession of linear segments approximates the *trajectory* with many less points than the original *trajectory*. In [Keogh *et al.*, 2001] a comparative among segmentation algorithms can be found. In our case, the selected algorithm is the sliding window algorithm. The reason for this selection is that it will be needed perform this segmentation on-line in the next phase of the methodology. The sliding window algorithm can be found in Table 1.

In this algorithm a segment is growing until it exceeds a determine error bound.

$$\sum_{i=1}^{n} |x_i - s_i| < error \qquad (1)$$

In equation 1 x_i is each trajectory point and s_i is each segment point.

When the error bound is exceeded, a new segment begins to grow. The error is calculated using the expression in equation 2.

```
Algorithm Normalized_DB = Normalize(Segmented_DB)
timestamps = {}
trajectory = read from Segmented_DB
while ∃ trajectories in Segmented_DB
   for each element in trajectory
     if timestamp(element) ∉ timestamps
       timestamps = timestamps ⋃ timestamp(element)
     end
     trajectory = read from Segmented_DB
   end
  end
go top of Segmented_DB
trajectory = read from segmented_DB
while ∃ trajectories in Segmented_DB
  for each element in timestamps
    if element ∄ in trajectory timestamps
      generate a new segment with element
    end
    add new element to normalized trajectory
  end
  save normalized trajectory to Normalized_DB
  trajectory = read from Segmented_DB
end
```

Table 2: Normalization algorithm

$$Error = \sqrt{\sum_{i=1}^{n} \frac{(x_i - s_i)^2}{n}} + \lambda \max |x_i - s_i| \qquad (2)$$

The first term of the sum in the expression represents the deviation between the *trajectory* and the segment. The second term ensures that the deviation in any point is further that error. Lambda factor lets pondering each term.

In the the segmentation algorithm, each *trajectory* has been approximated with few segments, but the resulting database is not a *labelled trajectories database*, because all *trajectories* in database have not the same number of elements. After segmentation process, each *trajectory* is represented with a different amount of segments, and these segments start and finish in different time instants. In order to solve this situation, and recover the *labelled trajectories database*, new segments must be generated to homogenize the numbers of elements in each *trajectory*. This will be performed with a normalization algorithm. The normalization algorithm is shown in the table 2.

This algorithm performs two iterations on the *segmented trajectories database*. In the first iteration, all different timestamps of all *trajectories* are saved. In the second iteration, new segments are generated for all timestamps calculated in the first iteration. Each *normalized trajectory* is saved in a new *normalized trajectories* database and the label of *trajectory* is added. Generating new segments from a existing segments is very easy. Only it is

necessary to applied the linear equation $y = ax + b$. a and b are calculated for the current segment, and a new y is generated for the new timestamp x.

Normalization algorithm returns a new database in which all *trajectories* have the same number of elements and they correspond to the same timestamp. This new database fulfils the definition of *labelled trajectories database*.

The cost that must be paid in normalization process is that the new *normalized trajectories* have more segments than the original ones. In any case, the global process of segmentation and normalization has reached to reduce the amount of information to treat.

Decision trees generation Final step in off-line phase is to generate a set of decision trees that can be used in the diagnostic process. The resulting database of the previous step fulfils all conditions to apply to them a classification tool. This way the *normalized and segmented labelled trajectories database* is used as training set with the selected classification tool. Output of this step is the set of generated decision trees.

On-line phase

The on-line phase of the methodology consists in evaluating an observation, of the sensor values of the monitored system, with the set of decision trees obtained in the off-line phase. In order to compare observed data with decision trees, the same treatment must be applied to observed data. This way, it is necessary to perform the on-line segmentation of the *trajectory* which is being observed. This is the reason because on-line segmentation algorithm was selected. Also, it is necessary to generate new segments for the normalized timestamps. After this treatment, observed values will be comparable with decision trees directly.

All process has low computational cost, and due to this the on-line process can be quickly performed.

Output of the evaluation is the label corresponding to behaviour of the system. This label indicates what is happening in the system.

Because a different set of decision trees have been generated for each transient state, it is necessary to know the current setting point and the new reference, in order to select the appropriate set of decision trees.

2.4 Diagnosability conflicts

From the diagnosability definition in [Console *et al.*, 2000] and the discriminable fault definition in [Trave-Massuyes *et al.*, 2001], we can conclude that a system is fully diagnosable when all faults are discriminable among them. But, what to do when this does not happen? We propose to offer all possible diagnostics. In other words, when two or more faults produce the same sensor readings, we propose to offer all possible causes. With the current methodology this is not possible, because when decision trees are evaluated, only one diagnosis is possible. In order to be able to offer all possible diagnosis for non-discriminable faults, a new step is proposed in off-line phase of the methodology: *Labelled trajectories database* will be replicated as many times as different labels exists. For each new database, only one label will be maintained and the rest of labels are changed to the opposite of the maintained label. This way, each new database has only two opposite labels, and there are as many databases as possible behaviours for diagnosing. Of course, all of these databases are classified individually, and a different decision trees set is obtained for each database (figure 3).

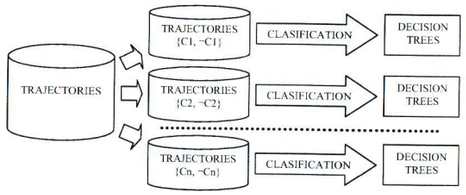

Figure 3: Multiple decision tree sets generation

On-line phase of the methodology must be adapted to the new changes. Now, all decision rule sets must be evaluated with the same sensor readings, and each one returns its own diagnosis. The diagnosis returned by each new rule sets has only two possibilities: positive (belonging to its class) or negative (not belonging to its class). The definitive diagnosis will be all positive evaluations of each class (figure 4).

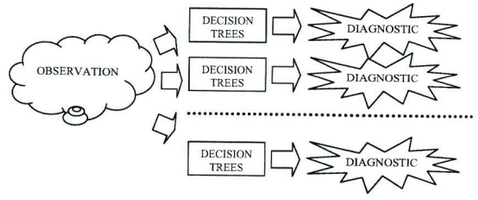

Figure 4: Multiple evaluating of a observation

With this modification in the methodology, three possibilities could occur:

1. Only one set of trees provides a positive diagnosis. This means that is the diagnosis.
2. Some sets of trees provide a positive diagnosis. This means that the correct diagnosis is one of them but cannot be specified which one.
3. No sets of trees provide a positive diagnosis. This means that a fault is present (because the free fault class is negative) but cannot be specified which one.

Third possibility lets to methodology to give a non-incorrect result when a fault mode for which it has not been prepared occurs.

3 Application to a DC motor

3.1 System description

The proposed methodology has been applied to a separately excited DC motor which is supplied by a three phase rectifier circuit. The DC motor is fed by the three phase rectifier through a chopper that consists of a IGBT transistor, and a free-wheeling diode. The motor torque is controlled by the armature current, which is regulated by a current control loop. The motor speed is controlled by an external PI controller, which provides the current reference for the current control loop.

The DC motor drives a fixed load. This motor performs a recurrent work, running from the stop state to reach the reference. This could be the scenario of a motor driving a centrifugal pump. This system has been implemented with Simulink® using components of the SimPowerSystems® Toolbox. (figure 5)

3.2 Faults identification

Faults to diagnose are divided in two types:

Rupture Faults. When these faults occur, the faulty component stops totally. For the present system, next rupture faults will be considered:

- One phase fault. One phase voltage is missing.
- Two phases fault. Two phases voltage are missing.
- IGBT rupture. The IGBT transistor is always on.
- Free-wheeling diode is short-circuiting.
- Free-wheeling diode is open-circuiting.

Tire Faults. This type of faults occurs when components are loosing its properties gradually, and this produces a progressive alteration in the working system. Next tire faults will be considered:

- Some turns in the armature winding are short-circuited. This produces a decrement in the inductance and resistance of the winding. The rate between the number of short-circuited turns and the decrease in resistance and inductance are given by equations 3 and 4.

$$\Omega = K\frac{L}{S} \tag{3}$$

$$L = \frac{\mu_r \mu_0 N^2 A}{L_c} \tag{4}$$

- Some turns in the field winding are short-circuited. The fault is the same that the previous one, but applied to the field winding.
- Bearing friction increasing. An excessive bearing friction, due to stress, produces an increasing in the friction torque.

3.3 Off-line phase

In order to obtain the *labelled trajectories database*, a set of simulations have been done with the above described system. Transient state simulated is from stop state to a reference of 50 rad/sec. Parameters for simulations are shown in table 3

Nominal Power	3.9	KW
Armature resistance	0.6	Ω
Armature inductance	0.0012	H
Field resistance	240	Ω
Field inductance	120	H
Mutual inductances	1.8	H
Friction torque	0.5	N.m
Load torque	10	N.m

Table 3: DC Motor specifications

A tolerance interval is considered in nominal values of each component. These components are considered free of fault if its value persists inside this interval. This represents the little alterations that can be produced in a system without a fault be considered. Tolerance rate has been established in a 4% of the nominal value of the components.

Each rupture fault simulation has been performed by changing the system model to produce the fault.

For tire faults, the nominal values of the faulty component are changed. For short-circuiting of turns in windings, has been considered a decrease of inductance and resistance until 40%. Friction torque rising to 10 N.m has been considered for bearing fault.

30 simulations per behaviour have been done by Montecarlo selection of interval values. Data gathered corresponds to readings of the angular speed of the rotor (Wm) and the control signal (Cv).

After simulations, the segmentation and normalization process has been applied to *labelled trajectories database*. With the aim of increase the available information, the value of the derivate of each segment has been added to the *segmented trajectories database*.

Finally, the classification tool C5.0® has been applied to *segmented trajectories database*. Both options, a single decision tree and multiples decision trees have been generated. The size of the single tree is 9 nodes. Sizes of multiples trees are between 1 and 6 nodes.

3.4 On-line Phase: Results

Results of table 4 have been obtained by 10 simulations per behaviour. Each simulation has been appropriately treated and evaluated with both sets of trees.

It is important to highlight that faults F2 and DS are non-discriminable because both give the same sensor readings.

As can be found in results table, the rupture faults are clearly diagnosed with the single tree, except for the non-discriminable faults F2 and DS. Single tree gives an erroneous diagnosis for F2 fault in all simulations. Tire faults are more imprecisely diagnosed, and a few simulations offer an erroneous diagnosis. Motive for this is that sensors readings are very similar when tire values are very closer to the correct ones. By using multiples trees, all results are more ambiguous. Various possible diagnosis are offered in some fault modes which are correctly diagnosing with the single tree. However, with non-discriminable faults DS and F2, both diagnosis are offered, avoiding an erroneous diagnostic.

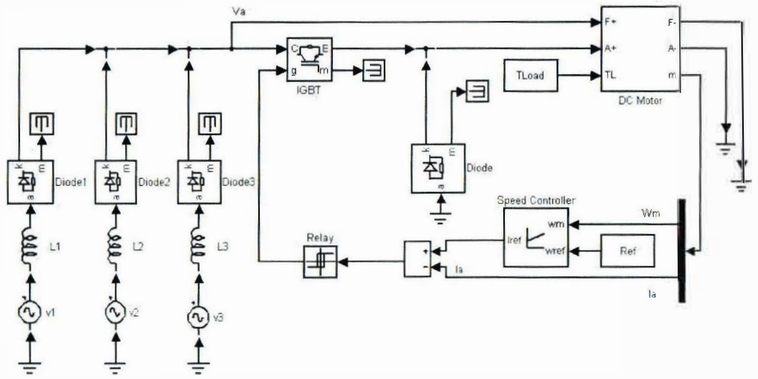

Figure 5: Simulink® model of the system

Behaviour	Single tree diagnosis	Multiple trees diagnosis
(OK) No Fault	100% OK	80% OK 20% OK or AW
(AW) Armature Wind.	80% AW 20% OK	60% AW 20% AW or OK 20% AW or BF
(FW) Field Windings	90% FW 10% BF	80% FW 10% FW or F1 10% ??
(BF) Bearing Friction	90% BF 10% FW	80% BF 20% BF or FW
(F1) 1 Phase Fault	100% F1	90% F1 10% ??
(F2) 2 Phase Fault	100% DS	100% F2 or DS
(IG) IGBT Fault	100% IG	100% IG
(DS) Diode Short-cir.	100% DS	100% DS or F2
(DO) Diode Open-cir.	100% DO	100% DO

Table 4: Diagnosis results

4 Conclusions and further works

A complete methodology to diagnose dynamic systems by classification has been proposed.

This methodology is able to diagnose dynamic systems when they are running in transient states that have been previously trained.

Models are not necessary when experimental data are available; they are only used when it is impossible to acquire real data.

It is not necessary to add new sensors to the system. When diagnosis is not clear, all possibilities are offered.

When the trees have been generated, the diagnosis process is very simple, and it can be performed with a very little computational time. This allows that it can be implanted with low cost components.

Currently this methodology has been applied to a real implementation of the modelled system. Some different faults are being added, as eccentricity fault or brush fault. Also, improvements in data treatment and classification step are being tested.

Acknowledgments

This work have been partiality funded by the *Ministerio de Ciencia y Tecnologia de España* (DPI2003-07146-C02-01), the European Regional Development Fund (ERDF/FEDER) and the *Modelización Matemática Redes y Multimedia* research group of the University of Huelva.

References

[Abad *et al.*, 2002] Pedro J. Abad, Antonio J. Suárez, Rafael M. Gasca, and Juan A. Ortega. Using supervised learning techniques for diagnosis of dynamic systems. In *13th International Workshop on Principles of Diagnosis*, Semmering, Austria, 2002. M.Stumptner & F.Wotawa Ed.

[Chan *et al.*, 1999] C. W. Chan, K. C. Cheung, H. Y. Zhang, and Y. Wang. Fault detection of dc-motors using nonlinear observer based on current b-spline neurofuzzy network. In *Proc 14th IFAC World Congress*, volume B, Beijing, China, 1999.

[Console *et al.*, 2000] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and diagnosability using pepa. In *14th European Conference in A. I.* IOs Press, 2000.

[Cordier *et al.*, 2000] M. 0. Cordier, P. Dague, M. Dumas, F. Lévy, J. Montmain, M. Staroswiecki, and L. Travé-Massuyès. A comparative analysis of ai and control theory approches to model-based diagnosis. In *International Workshop on Principles of Diagnosis DX'00*, pages 33--40, Morelia, Mexique, June 2000. Morgan Kaufmann.

[Feng, 1992] C. Feng. Inducting temporal fault diagnostic rules from a qualitative model. In *Inductive Logic Programming*. S. Muggleton, editor. Academic Press, 1992.

[Filippetti *et al.*, 1992] F. Filippetti, M. Martelli, G. Francesschini, and C. Tassoni. Development of expert system knowledge base to on-line diagnosis of rotor electrical faults of induction motors. In *Conf. Rec. 27th IEEE-IAS Annu. Meeting*, Oct 1992.

[Hajiaghajani *et al.*, 2004] Massoud Hajiaghajani, Hamid A. Toliyat, and Issa M. S. Panahi. Advanced fault diagnosis of a dc motor. *IEEE Trans. On Energy Conversion*. 19(1), March 2004.

[Keogh *et al.*, 2001] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. *IEEE Intl. Conf. Data Mining*, pages 289–296, May 2001.

[Liu *et al.*, 2000] Xiang Qun Liu, Hong Yue Zhang. Jun Liu, and Jing Yang. Fault detection and diagnosis of permanent-magnet dc motor based on parameter estimation and neural network. *IEEE Trans. On Industrial Electronics*, 47(5), October 2000.

[Roverso, 2003] Davide Roverso. Fault diagnosis with the aladdin transient classifier. In *System Diagnosis and Prognosis: Security and Condition Monitoring Issues III*. AeroSense2003, Aeroespace and Defense Sensing and Control Thecnologies Symposium, 2003.

[Schoen *et al.*, 1995] R. R. Schoen, T. G. Habetler, F. Kamran, and R. G. Bartheld. Motor bearing damage detection current monitoring. *IEEE Trans. Ind. Applicat*, vol. 31, Nov-Dec 1995.

[Trave-Massuyes *et al.*, 2001] L. Trave-Massuyes, T. Escobe, and R. Milne. Model based diagnosability and sensor placement application to a frame 6 gas turbine subsystem. In *12th international Workshop on principles of diagnosis (DX01)*, pages 205–212, Sansicario, Via Lattea, Italy, 2001.