Application of Multiple Sliding Time Windows to Fault Detection Based on Interval Models

Joaquim Armengol[1,2], **Josep Vehí**[1], **Louise Travé-Massuyès**[2,3] **and Miguel Ángel Sainz**[4]
[1] Institut d'Informàtica i Aplicacions. Universitat de Girona. Campus de Montilivi. E-17071 Girona, Catalonia, Spain.
[2] LEA-SICA.
[3] LAAS-CNRS. 7, Avenue du Colonel Roche. F-31077 Toulouse, France.
[4] Departament d'Informàtica i Matemàtica Aplicada. Universitat de Girona.
Campus de Montilivi. E-17071 Girona, Catalonia, Spain.
E-mail: armengol@eia.udg.es, vehi@eia.udg.es, louise@laas.fr and sainz@ima.udg.es

Abstract

Interval models may be used in many cases to express the imprecision and the uncertainty related to complex systems. The envelopes may be used to represent the results of the simulation of these models. One of the applications of the envelopes is as reference behaviour for Fault Detection (FD) based on analytical redundancy. In this case, the properties of the envelopes (completeness, soundness) have important consequences on the results of the FD, like missed or false alarms. This paper presents the Modal Interval Simulator (MIS), which approaches the FD problem by means of error-bounded envelopes, i.e. by the simultaneous computation of an overbounded envelope and an underbounded one. Modal Interval Analysis, which provides tools to compute interval extensions of real functions with the adequate semantics, is used for computing these envelopes. The MIS system uses multiple sliding time windows for performing FD. This allows the detection of faults of different kinds avoiding (provided that some assumptions are fulfilled) false alarms.

Introduction

A fault is a deviation of at least one characteristic property or parameter of a system from the acceptable, usual or standard condition (Isermann & Ballé 1997). The consequences of a fault may range from economical losses derived from lower efficiency of the system to danger for the people or the environment. This makes the early detection of faults an important task. Many different techniques have been developed in the recent years to approach this problem (Chen & Patton 1998; Frank, Ding, & Köppen-Seliger 2000). Among them, there are heuristic approaches, which are based on rules or cognitive methods, and analytical approaches, which are based on a model of the system. In the latter case tools like identification or estimation are used.

One technique for detecting faults consists in comparing the behaviour of the real system and a reference one. A fault is detected when there are discrepancies between them (Reiter 1987). This technique is called physical redundancy when the reference behavior is obtained from another system and analytical redundancy when it is obtained from a model of the system. In the latter case, the results of the Fault Detection (FD) are highly dependent on the model. The main problem is that the model is an approximate representation of the system and hence the two behaviours are never exactly the same. This is a consequence of the modeling procedure of systems because it usually involves hypotheses, assumptions, simplifications, linearizations, etc.

Uncertainty and imprecision must be considered in order to avoid this problem. They can be taken into account, for instance, when the comparison between the behaviour of the real system and the one of its model is performed. In this case, a fault is indicated when the difference is larger than a threshold. The difficulty is to determine the size of the threshold, which depends on the uncertainty and the imprecision. If the threshold is too small, there may be false alarms. On the other side, if the threshold is too large, there may be missed alarms, i.e. faults that are not indicated.

Imprecision and uncertainty may also be considered in the modeling procedure. This work is based on this approach. Specifically, interval models, i.e. models with interval-valued parameters, are used. The simulation of this kind of models produces envelopes. In this case, a fault is indicated when the behaviour of the real system is outside the predicted envelope. Then, the properties of the FD results are consequence of the properties of the envelopes. Best FD results are obtained using the exact envelope, but its calculation is a difficult task. This work proposes to approach this problem by generating error-bounded envelopes, which produce the same FD results than the exact envelope requiring much less computations. The computation of error-bounded envelopes is tackled by using modal intervals. The FD results are improved using multiple sliding time windows.

The Modal Interval Simulator (MIS) includes these features, i.e. the computation of error-bounded envelopes by means of Modal Interval Analysis and the use of multiple sliding time windows. This simulator is applied to the detection of faults in an academic example. This example shows the effects of the uncertainty in the measurements. The proposed approach to deal with this problem is used for detecting faults in a real example. Finally, a summary is presented and some conclusions and some orientations for the future work are provided.

Interval Models

A usual model consisting of functions with real-valued parameters is precise so it does not capture imprecision and uncertainty. These can be represented in an interval model, that is, a model where the values of the parameters are intervals. An interval model represents indeed a set of models. For instance, assume that the behaviour of a n-th order dynamic SISO (Single Input, Single Output) discrete system is represented by the following difference equation:

$$y_t = \sum_{i=1}^{m+1} a_i y_{t-iT} + \sum_{j=1}^{p+1} b_j z_{t-jT} \tag{1}$$

in which it can be observed that the output of the system at any time point (y_t) depends on the values of the previous outputs (y_{t-iT}) and inputs (z_{t-jT}), being T the sampling time. This dependency is given by the parameters of the system model (a_i and b_j). Their values can be expressed by means of intervals if they are uncertain or imprecise. Part of the imprecision originates from noise and anticipated disturbances. It is assumed here that these are captured by the interval values.

Envelopes

The reference behaviour for analytical redundancy can be obtained from the model by simulation. In the case of a real-valued model, the results are the trajectories of the variables of the system across time. In the case of an interval model, as it is a set of models, a set of curves called *envelope* has to be obtained for each variable. The envelope is complete, i.e. it includes all the possible behaviours of the set of models, and sound, i.e. every point inside the envelope belongs to the trajectory of at least one of the models included in the set.

The computation of the envelope is a difficult task, specially (and paradoxically) if the system is considered time invariant, i.e. it is known that the values of the parameters are constant although they are uncertain or imprecise. At each time step of the simulation the maximum and the minimum possible values of the variable have to be determined. This is a range computation problem. The function whose range has to be determined is defined by the interval model of the system and the parameter space is determined by the interval values of the parameters, the inputs and the initial state. This problem can be solved, for instance, using global optimization algorithms (Hansen 1992; Kearfott 1996). This task needs, most of the times, an important computational effort and, even though, the results are usually approximations due to errors of rounding, truncation, etc.

Therefore, the result of the simulation is not the envelope but an approximation of it, which can be complete or sound, but usually is not both. The complete and sound envelope is called *exact envelope*, whereas a complete but not sound envelope is called *overbounded envelope* and a sound but not complete envelope is called *underbounded envelope*. In the worst case, an envelope which is neither complete nor sound is obtained. The properties of the envelopes are summarised in figure 1.

These properties are very important to the FD results. The value of a variable of a non-faulty system must be inside the predicted envelope, hence a fault is detected when it appears outside the envelope. However, the value of a variable of a faulty system can be everywhere, inside or outside the envelope, due to the dynamics of the system among other

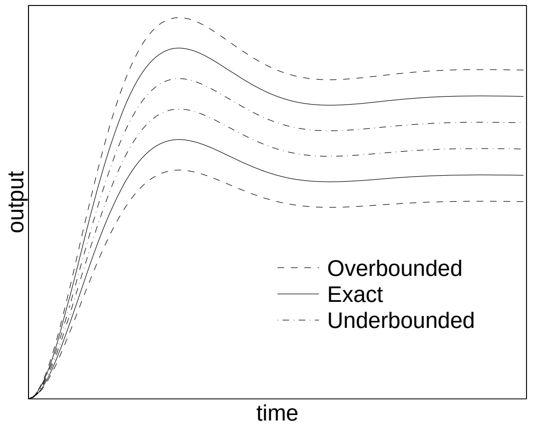

Figure 1: Properties of the envelopes.

causes. Therefore, it is guaranteed that there is a fault when a variable is outside the envelope, but there is no way, using the envelope approach, to guarantee that a system is not faulty. In other words, if the exact envelope could be used for FD, there would not be false alarms (under the stated assumptions about noise and disturbances), but there could be missed alarms.

In the real case the FD system uses envelopes that are not exact. If an overbounded envelope is used, there are not false alarms, like when the exact envelope is used, and the amount of missed alarms depend on the degree of overbounding of the envelope. If the envelope is much overbounded, the amount of missed alarms is larger. On the other side, if an underbounded envelope is used, the amount of missed alarms is smaller, but there may be false alarms. Therefore, the exact envelope is the one that reduces the amount of missed alarms to the minimum allowed by the envelope-based FD method without generating false alarms.

The properties of the envelopes of several simulators are assessed in (Armengol *et al.* 2000).

Error-bounded Envelopes

Even when the envelopes are overbounded (resp. underbounded) the degree of overbounding (resp. underbounding) is generally not known, so the distance with respect to the exact envelope is not known. The determination of this distance is equivalent to the determination of the exact envelope. Nevertheless this distance can be bounded computing simultaneously an overbounded and an underbounded envelope: the exact envelope includes the underbounded one and is included in the overbounded one. The set formed by an overbounded and an underbounded envelope is called *error-bounded envelope*.

The use of error-bounded envelopes for FD compared to the search of the exact envelope is an important improvement as it allows one to obtain the same results with much less computations. This is achieved when the error-bounded envelopes are computed iteratively. Every time a partial result is obtained, it is compared with the measurement of the variable. The FD algorithm is structured along these three cases:

- Case 1. If the measurement is outside the overbounded envelope, the iterative procedure can stop because although better approximations to the exact envelope could be obtained, the measurement would always be outside the overbounded envelope. A fault is indicated.
- Case 2. If the measurement is inside the underbounded envelope, iterations can also stop because even if better approximations to the exact envelope would be obtained, the measurement would remain in that zone. This means that a fault can not be indicated, even if the system is faulty.
- Case 3. If the measurement is between the overbounded and the underbounded envelopes, more iterations have to be done to come back to one of the two previous cases.

It is clear from the above algorithm that the use of error-bounded envelopes guarantees that there are not false alarms and that the amount of missed alarms is reduced to the minimum (corresponding to the exact envelope). In other words, the optimal results are obtained with much less computations than the ones needed to obtain a close approximation of the exact envelope.

Modal Intervals for Producing Error-bounded Envelopes

The computation of error-bounded envelopes is nicely solved by applying the Modal Interval Analysis (MIA) (Armengol *et al.* 1999). MIA (SIGLA/X 1999) is an extension of classical interval analysis (Moore 1966; 1979). The main difference is that interval analysis identifies an interval by a set of real numbers, whereas MIA identifies an interval by the set of predicates that are fulfilled by the real numbers. Then, a modal interval is defined by a pair

$$X := (X', QX) \tag{2}$$

In this pair, X' is called the *extension*, $X' = \left\{[a,b]' \mid a,b \in \mathbb{R},\quad a \leq b\right\} \in I(\mathbb{R})$. QX is the *modality*, $QX \in \{E, U\}$. The existential modality E indicates that at least one element in the interval fulfills a predicate whereas the universal modality U indicates that every element in the interval fulfills it.

The dual formulation of modal intervals results in the definition of two modal interval extensions of functions, noted by $f^*(X)$ and $f^{**}(X)$ respectively, which provide meaning to the interval computations. In the case of the envelopes, the semantics associated to the *-extension is:

$$\begin{aligned} &U(y(0), Y(0))\, U(z(0), Z(0)) \ldots \\ &U(z(n-1), Z(n-1))\, U(p_i, P_i)\, E(y(n), Y(n)) \\ &(y(n) = f(y(0), z(0), \ldots, z(n-1), p_i)) \end{aligned} \tag{3}$$

In this expression, $y(0)$ is the output of the system at the initial state, $z(i)$ are the inputs, p_i are the parameters of the system model and $Y(n)$ is the calculated envelope at the time point n. This expression is read: “For every $y(0)$, $z(i)$ and p_i, $y(n)$ belongs to $Y(n)$”. Therefore, the envelope computed using the *-extension is complete.

On the other side, the semantics associated to the **-extension is:

$$\begin{aligned} &U(y(n), Y(n))\, E(z(0), Z(0)) \ldots \\ &E(z(n-1), Z(n-1))\, E(p_i, P_i)\, E(y(0), Y(0)) \\ &(y(n) = f(y(0), z(0), \ldots, z(n-1), p_i)) \end{aligned} \tag{4}$$

This semantics is dual: “For every $y(n)$ belonging to $Y(n)$, there exist p_i, $z(i)$ and $y(0)$ that produce this output”. Therefore, the envelope computed using the **-extension is sound.

If both extensions are equal, then the result is exact. Unfortunately, the computation of the *- and **-extensions is, in general, a difficult challenge. MIA provides tools to find overbounded computations of $f^*(X)$ and underbounded computations of $f^{**}(X)$ which maintain the semantic interpretations. These computations are made controlling the roundings of the operations and taking into account the multi-incident variables in the functions.

To illustrate multi-incidences, assume that the model of a system is given by the following difference equation:

$$y_n = a y_{n-1} + (a+b) z_{n-1} \tag{5}$$

The functions whose range have to be computed at the first steps are:

$$\begin{aligned} y_1 &= a y_0 + (a+b) z_0 \\ y_2 &= a y_1 + (a+b) z_1 \\ y_3 &= a y_2 + (a+b) z_2 \end{aligned} \tag{6}$$

There are several kinds of multi-incidences in these functions:

- inside a function, for instance, a appears several times in y_1.
- between functions in different time steps, for instance, a appears in y_1 and y_2 and it is a single parameter if the system is considered time invariant.
- hidden multi-incidences, for instance, if a depends on a physical parameter and b depends on the same one.

A naive substitution of the parameters by their interval values in these functions would lead to obtain results defining overbounded envelopes due to the multi-incidences of the variables. This is what happens when interval arithmetic is applied.

The coercion theorems provide the conditions and the way to obtain optimal extensions when there is monotonicity. If the function is not monotonic for each multi-incident component, these theorems can be partially applied in order to reduce the complexity of the problem.

Finally, a way to obtain even better approximations is by splitting the parameter space. This is done by means of a branch-and-bound algorithm. The use of modal intervals decreases computational time because more subspaces are eliminated compared to other techniques.

Algorithm

This branch-and-bound algorithm computes an external and an internal approximations to the range of the function at each time point. The algorithm applies the coercion theorems to the monotonic variables, which are determined assessing the monotonicity by using only the first derivative. If the function is not totally monotonic and possibly the maximum or the minimum of the function is in a parameter space, this space is split along the edges of the not monotonic variables.

The input arguments of the algorithm are a function f of a set of variables $X = \{x_1 \ldots x_n\}$ and the parameter space determined by the interval values of the variables:

$$Q = \left\{ q = [q_1, q_2, \ldots, q_n]^T \mid q_i \in \left[\underline{q_i}, \overline{q_i}\right], i = 1 \ldots n \right\} \tag{7}$$

The output arguments are the external and internal approximations to the range.

```
function $(external, internal)$ =approx_range$(f, Q)$
    $finish = false$
    $internal$ =inner$(f, Q)$
    save$(Q, read\_list)$
    DO
        $external = internal$
        DO
            $P$ =get$(read\_list)$
            IF not_monotonic$(f, P) = 0$ THEN
                $partial$ =exact$(f, P)$
                $internal = internal \vee partial$
                $external = external \vee partial$
            ELSE
                $partial$ =inner$(f, P)$
                $internal = internal \vee partial$
                $partial$ =outer$(f, P)$
                $external = external \vee \ partial$
                IF not$(partial \subseteq internal)$ THEN
                    $(P_1, P_2)$ =split$(P)$
                    save$(P_1, write\_list)$
                    save$(P_2, write\_list)$
                ENDIF
            ENDIF
        WHILE not(is_empty $(read\_list)$)
        IF is_empty$(write\_list)$ THEN
            $finish = true$
        ENDIF
        IF stop_condition THEN
            $finish = true$
        ENDIF
        exchange$(read\_list, write\_list)$
    WHILE not$(finish)$
```

The functions that are used by the algorithm are described in the following.

function b =**not**(c) returns $true$ if c is $false$ and vice versa.

function save$(Q, list)$ adds the subspace Q to the list of subspaces $list$. The length of the list is increased by one unit.

function Q =**get**$(list)$ picks one of the elements of the list of subspaces $list$. This element is deleted in the list, so the number of elements in the list is decreased by one unit.

function b =**empty**$(list)$ returns $true$ or $false$ depending on whether the list of subspaces $list$ is empty or not, respectively.

function exchange$(list1, list2)$ exchanges the lists of subspaces $list1$ and $list2$.

function a =**not_monotonic**(f, Q) returns the real number a, that is the number of variables with respect to which the function f is not monotonic in the parameter space Q. The test is performed using only the first derivative and using the *-partially optimal coercion theorem.

function Z =**exact**(f, Q) applies the optimal coercion theorem for uni-modal arguments to have the exact range of the totally monotonic function f in the parameter space Q, which is returned as the interval Z.

function Z =**inner**(f, Q) applies the **-partially optimal coercion theorem to have an internal approximation of the range of f in the parameter space Q, which is returned as the interval Z.

function Z =**outer**(f, Q) applies the *-partially optimal coercion theorem to have an external approximation of the range of f in the parameter space Q, which is returned as the interval Z.

function (Q_1, Q_2) =**split**(Q) splits the parameter space Q in two subspaces Q_1 and Q_2 such that $Q_1 \cup Q_2 = Q$ and $Q_1 \cap Q_2 = \emptyset$. Different criteria may be used for the splitting. The implemented function splits along the largest edge among the non-monotonic variables.

The function **stop_condition** compares the measured value of the variable with the partial results of the branch-and-bound algorithm and, according to the three cases described above, returns $true$ if this algorithm can already stop and $false$ otherwise:

```
function $b$ =stop_condition$(external, internal, meas)$
    IF $meas \geq \overline{external}$ THEN
        $b = true$
    ELSEIF $meas \leq \overline{internal}$ AND $meas \geq \underline{internal}$
            THEN
        $b = true$
    ELSEIF $meas \leq \underline{external}$ THEN
        $b = true$
    ELSE
        $b = false$
    END
```

Sliding Time Windows

When the behaviour of a system is simulated, it is assumed that measurements of the variables are not available so the value of each variable at a specific time point is computed with the model of the system, from the initial state (the values of the variables at a past time point) and the inputs to the system from the initial time point to the current one. This implies that each simulation step needs more computations than the previous one and hence the simulation becomes slower and slower. This is the drawback of the simulator presented in (Armengol *et al.* 1999).

In the case of FD, the measurements of the variables (generally a subset) at each time point are needed in order to compare them with the corresponding envelopes. Hence these measurements are available and any measurement belonging to a past time point can be used as initial state to compute the envelopes at the current time point. The time interval from this *initial* time point to the current one is called *time window*. If the window used at each simulation step has always the same length, then a *sliding time window* is being considered. In this case, the time needed to compute the envelopes at each simulation step remains constant.

The resulting envelopes depend on the length of the sliding time window that is being used, so the indication of faults using different window lengths can also be different. This can be due to different reasons.

One possible reason is the dynamics of the system. The output of a faulty system may be inside the envelope, at least during some time interval, so there is a detection time. Experimentally, it has been seen that longer windows have longer detection times.

Then it could seem that shorter windows are better. This is not true. For instance, shorter windows do not detect slow drifts because the envelope ”follows” the measurement. This gives another possible reason for having different results when different window lengths are used: the kind of fault, its duration, etc. The experiments show that short duration faults are not detected by longer windows because the duration of the fault is shorter than the detection time. These questions will be deeply studied in the future.

And yet another reason is that there are combinations of parameters outside their respective intervals that give outputs inside the envelope. This is shown with an example in the following.

The example is based on a generic first order system:

$$y_n = \left(1 - \frac{T}{\tau}\right) y_{n-1} + \frac{kT}{\tau} z_{n-1} \qquad (8)$$

with the following parameters:

- static gain: $k = [0.95, 1.05]$.
- time constant: $\tau = [10, 20]$ s.
- initial state: $y_0 = [1.5, 1.5]$.
- sampling time: $T = 1$ s.
- input: $z_n = 2, 1, 2, 1...$ for $n = 0, 1, 2, 3...$ s.

The exact envelope at $t = 1$ s is computed. Then, many combinations of k and τ are simulated and their output is checked against the envelope. The result is shown in figure 2a: the combinations belonging to the interval model (the rectangle) give outputs inside the envelope, as expected, but many others (the shadowed regions) also do.

The same procedure is done at $t = 2$ s and $t = 3$ s and the obtained results are shown in figures 2b and 2c, respectively.

Given that error-bounded envelopes guarantee that there are not false alarms but there can be missed alarms, if a fault

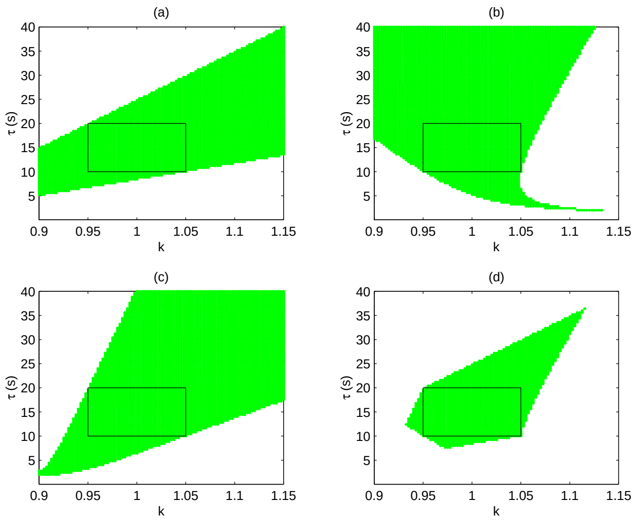

Figure 2: FD using different window lengths.

is indicated by any of the envelopes obtained using different window lengths, then it is guaranteed that there is a fault:

$$\begin{aligned} &\text{IF } y_m(t) \notin Y(w_1, t) \text{ OR } \ldots \\ &\text{OR } y_m(t) \notin Y(w_n, t) \text{ THEN } fault = 1 \end{aligned} \qquad (9)$$

$y_m(t)$ is the measured value of the variable y at time t, $Y(w_i, t)$ is the value of the variable y at time t predicted by the window length w_i and the value of $fault$ is 1 if a fault has been detected and 0 otherwise.

Therefore, the detection results that are obtained using multiple sliding time windows are better than the ones obtained using any single sliding time window, because it increments the amount of detected faults and moreover guarantees that there are not false alarms. This is confirmed in figure 2d which has a shadowed region smaller than the ones in figures 2a, 2b and 2c because it is the intersection of them.

The results that are obtained using multiple sliding time windows are also better than the ones that are obtained by simulation, i.e. computing the envelopes at any time point starting always from the same initial one, like in (Armengol *et al.* 1999).

Modal Interval Simulator

The Modal Interval Simulator (MIS) implements these features:

- Computation of error-bounded envelopes. It uses a branch-and-bound algorithm based on modal intervals. This algorithm calculates the error-bounded envelopes iteratively, tightening the overbounded envelope and widening the underbounded one at each iteration. The iterations stop when the computation of better approximations can not change the FD results.
- Use of multiple sliding time windows.

The branch-and-bound algorithm is programmed in C++ because the modal interval arithmetic must control the rounding of the operations by the microprocessor in order to maintain the semantics. The simulator MIS, however, is implemented in Matlab. It also uses Maple for some symbolic computations (Armengol *et al.* 1999).

FD in an Academic Example

MIS has been used to detect faults in the generic first order system introduced above. Now the initial state is $y_0 = [0, 0]$ and the input applied to that system is a sequence of steps of different lengths and heights shown in figure 3.

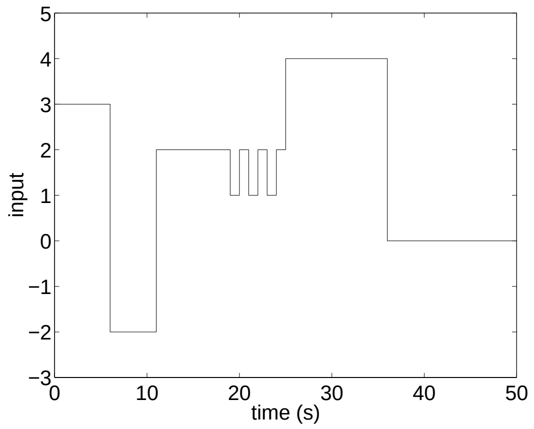

Figure 3: Input to the first order system.

Case A: Clean Data

This case shows the FD results when the system is faulty:

- static gain: $k = 1.15$.
- time constant: $\tau = 5$ s.

It is assumed that there is not uncertainty in the measurements, so their values are precise and correspond to the exact values of the output variable:

$$y_m(t) = y(t) \tag{10}$$

Figure 4 shows the results of the simulation: the error-bounded envelopes (the overbounded envelope in solid line and the underbounded envelope in dashed line).

It may be observed that there are several time points for which the overbounded envelope is much overbounded. It is because the measurement has been already detected inside the underbounded envelope at one of the first iterations. So it is not necessary to obtain better (less overbounded) approximations of it. In these cases, the results are obtained with a small computational effort.

Figure 5 represents the FD results for different window lengths. This figure shows that, although the system is faulty during the whole time interval, it is detected only at some time points. It shows also that the results using different window lengths are different, so the use of multiple sliding time windows enhances the overall results.

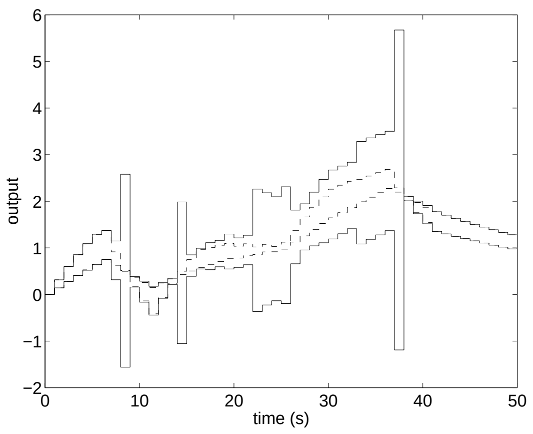

Figure 4: Error-bounded envelopes for window length $w = 5$ s.

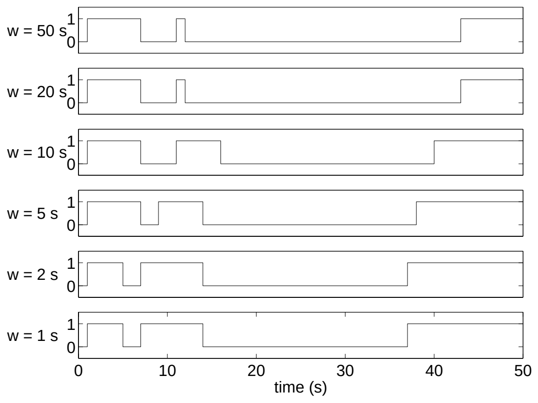

Figure 5: FD results in case A.

Case B: Uncertain Data

If measurements are uncertain (noisy, for instance), it is clear that this uncertainty must be taken into account in the decision procedure, otherwise it may result in false alarms.

The proposed approach is to bound the uncertainty. It is taken into account converting the real-valued measurements into interval measurements.

The decision procedure described above has to be modified, because it was based on the comparison between a real number and an error-bounded envelope. The new decision procedure is:

- Case 1. If the intersection between the measurement and the overbounded envelope is empty, the iterative procedure can stop because although better approximations to the exact envelope could be obtained, this intersection would always be empty so it is guaranteed that there is a fault.
- Case 2. If the intersection between the measurement and the underbounded envelope is not empty, iterations can also stop because even if better approximations to the exact envelope could be obtained, this intersection would never be empty. This means that possibly the actual value

of the variable is inside the underbounded envelope and hence a fault can not be indicated, even if the system is faulty.

- Case 3. In any other case, more iterations have to be done to come back to one of the two previous cases.

In this case a system which is not faulty has been used, but the measurements have been made uncertain (noise, analog to digital conversion errors, etc.) by adding a random number between -0.1 and 0.1 to the exact value of the output variable:

$$y_m(t) = y(t) + rnd([-0.1, 0.1]) \tag{11}$$

As the difference between the measurements and the actual values of the variables is known to be in the interval $[-0.1, 0.1]$, the real-valued measurements are converted into interval measurements by adding this interval:

$$y_{im}(t) = y_m(t) + [-0.1, 0.1] \tag{12}$$

When the above algorithm is applied to these data the indication of fault is always ”0”, so there are not false alarms.

Case C

This case shows the results of FD using the faulty system of case A and assuming that there is an uncertainty in the measurements like the one described for case B. So, interval measurements are used. The results are shown in figure 6. There are not false alarms so if any of the different window lengths indicate a fault at a specific time point, it is guaranteed that there is a fault at that time point.

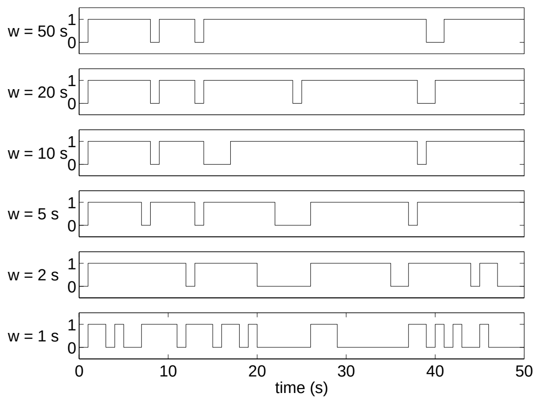

Figure 6: FD results in case C.

FD in a Real Example

This example, the TIGER 2nd stage nozzles system, comes from the Esprit project TIGER (Milne *et al.* 1994). It is a real example with real data.

The goal of that project was to detect and diagnose faults in gas turbines. These devices are very complex and include many subsystems: compressor, fuel system, cooling air, cooling water, hydraulic system, inlet guide vanes, lubrication oil system, steam injection system, etc.

Model

One of the subsystems of two shafts gas turbines is the 2nd stage nozzles system. It is formed by the nozzles, which are used to open and close the air flow through the turbine, and their associated actuator. The variables are:

- $TSNZ$: the position of the nozzles, hence the output of this subsystem.
- $TANZ$: the input to the actuator of the nozzles, hence the input of this subsystem.

The set formed by the nozzles and their actuator is modeled by means of the following difference equation:

$$TSNZ_n = TSNZ_{n-1} + ns_1 TANZ_{n-1} + ns_2 \tag{13}$$

where ns_i are the parameters of the model. The values of these parameters are intervals because they include the uncertainty. These values are not given here for confidentiality reasons.

Data

The values of about 600 variables are collected once per second, so there are many data from the gas turbine. Most of these data belong to situations of normal behaviour and hence are not very interesting. But sometimes there are abnormal situations, unusual events or incidents that are more interesting. Data from these situations are saved and called *scenarios*. Among them, there are scenarios with vibration problems, oil leaks, sensor failures, bad positioned or calibrated sensors, poorly tuned controllers, problems with valves, problems of gas supply, etc.

The example shown in this section uses data from a scenario in which the system is faulty. The nozzles are given a reference to close for a while and open again. Due to a lack of power of the actuator, the nozzles close, but less than expected, as it can be seen in figure 7.

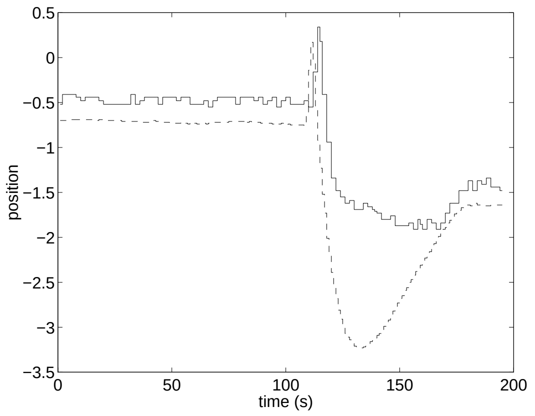

Figure 7: Reference (dashed line) and actual (solid line) position of the nozzles.

Fault Detection Results

The measurements are quantized, as it can be seen in figure 7, so the noise in the measuring procedure can make that a specific analog measure is converted to a digital number which is not the closer one. For this reason, the interval measurements that are used in this case are:

$$y_{im}(t) = y_m(t) + [-0.055, 0.055] \tag{14}$$

The results of MIS using real (noisy) data and windows of length 1, 2, 5, 10, 20, 50, 100 and 200 s are shown in figure 8.

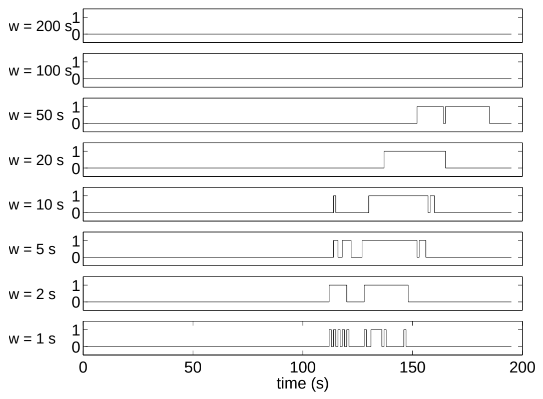

Figure 8: FD results in a subsystem of the TIGER gas turbine.

It can be seen in this figure that there are not false alarms and that some window lengths ($w = 100$ s and $w = 200$ s) never detect this kind of fault which lasts only for a short time.

Conclusions

This paper presents an approach to the detection of faults based on an interval model of the system, i.e. a model which express the uncertainty and the imprecision. The simulation of this model produces envelopes. Error-bounded envelopes produce the same results than the exact envelope with much less computations. The computation of these envelopes is performed using modal interval analysis in a branch-and-bound algorithm. This allows to obtain envelopes with the required semantics in an iterative procedure that stops as soon as the results are optimal.

Different sliding time window lengths may be used to generate the envelopes. Error-bounded envelopes guarantee that whenever one of them detects a fault then there is a fault. Therefore, the use of multiple sliding time windows enhance the results. This has been implemented in the simulator MIS. It has been used to detect faults in academic and real examples.

MIS is also a useful tool to get better models. If the model of the system is too imprecise, i.e. the intervals are too wide, MIS can be used to obtain models with tighter intervals in an iterative procedure: adjustment of the interval parameters, simulation in different scenarios, new adjustment according to the simulation results, and so on.

When the measurement is outside the overbounded envelope, a fault is detected. Its value is either larger or smaller than predicted. A research topic for the future is the use of this information to diagnose some faults, i.e. to indicate which is the faulty parameter of the model.

When a fault is detected, there are some window lengths that detect the fault and some others do not detect it. Another research topic is the use of this information to identify the fault, i.e. to indicate which is the type of the fault (an abrupt one, a short duration one, a drift one, etc.).

Acknowledgements

LEA-SICA is an associated European laboratory constituted by three Spanish and two French research groups.

This work has been funded by the autonomous government of Catalonia, Generalitat de Catalunya, through Grup de Recerca Consolidat ref. 1999SGR00161 and by the Spanish government through co-ordinated research projects CICYT TAP98-0955-C03-02 and TAP99-1079-C03-03.

References

Armengol, J.; Travé-Massuyès, L.; Vehí, J.; and Sainz, M. Á. 1999. Generation of error-bounded envelopes using modal interval analysis. *10th International Workshop on Principles of Diagnosis (DX 1999). Loch Awe, Scotland, UK* 20–26.

Armengol, J.; Travé-Massuyès, L.; Vehí, J.; and de la Rosa, J. L. 2000. A survey on interval model simulators and their properties related to fault detection. *Annual Reviews in Control* 24(1):31–39.

Chen, J., and Patton, R. 1998. *Robust model-based fault diagnosis for dynamic systems*. Kluwer.

Frank, P. M.; Ding, S. X.; and Köppen-Seliger, B. 2000. Current developments in the theory of FDI. *4th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS 2000). Budapest, Hungary* 16–27.

Hansen, E. 1992. *Global optimization using interval analysis*. Marcel Dekker.

Isermann, R., and Ballé, P. 1997. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Engineering Practice* 5:707–719.

Kearfott, R. 1996. *Rigorous global search: continuous problems*. Kluwer Academic Publishers.

Milne, R.; Nicol, C.; Ghallab, M.; Travé-Massuyès, L.; Bousson, K.; Dousson, C.; Quevedo, J.; Aguilar, J.; and Guasch, A. 1994. Tiger: real-time situation assessment of dynamic systems. *Intelligent Systems Engineering* 3(3):103–124.

Moore, R. E. 1966. *Interval analysis*. Prentice-Hall.

Moore, R. E. 1979. *Methods and applications of interval analysis*. Studies in Applied Mathematics (SIAM).

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32(1):57–95.

SIGLA/X. 1999. *Applications of Interval Analysis to Systems and Control. Proceedings of MISC 1999*. Universitat de Girona. chapter Modal Intervals, 157–227.