Failure Detection in Dynamic Systems with Modeling Errors

Dan T. Horak*
Allied-Signal Aerospace Company, Columbia, Maryland

This paper determines the theoretical limits of failure detectability in systems with modeling errors. Such errors are the main obstacle to accurate failure detection in aircraft, jet engines, power generation systems, and most other systems that include mechanical components and/or thermofluid processes. The paper then introduces the reachable measurement intervals (RMI) failure detection method that performs at the limit of detectability. RMI is capable of detecting the smallest theoretically detectable failures in systems with modeling errors, and is implementable in real time on microprocessors. The method is illustrated on a jet engine example.

Nomenclature

a	= entry of the matrix A
A	= system matrix
b	= entry of the matrix B
B	= input matrix
B_I	= input matrix equal to the identity matrix
C	= output matrix
C_1	= output matrix for additional measurements
D	= direct transmission matrix
H	= Pontryagin's function
K	= matrix of gains for additional measurements
L	= vector of Lagrange multipliers
m	= entry of the matrix M
M	= matrix of Lagrange multipliers associated with uncertain entries of A
n	= entry of the matrix N
N	= matrix of Lagrange multipliers associated with uncertain entries of B
p	= discrete-time index corresponding to current time
P	= performance index for optimization
q	= vector of system parameters
t	= time
t_o	= current time
t_s	= length of the processing window
T	= sampling time
u	= input vector
x	= state vector
y	= output vector
z	= vector of additional measurements

Subscripts

d	= discrete
e	= uncertain and uncorrelated
n	= nominal
v	= uncertain

Introduction

FAILURE detection and isolation (FDI) in dynamic systems is accomplished by testing the consistency of a system via hardware and/or analytical redundancy. Hardware redundancy is achieved by using multiple sensors for the measurement of one signal, and is relatively straightforward.

Analytical-redundancy-based FDI uses a model of the dynamic system to generate the redundancy required for failure detection. In some systems, it is possible to measure all of the states and use their values to test the validity of the state equations.[1] Measurement of all of the states allows FDI in a dynamic system to be performed as if the system were a static system, and is relatively straightforward. In many systems, however, all of the states cannot be measured because of cost, weight, and size considerations. FDI schemes for such systems must extract the redundant information from dissimilar sensors, using the differential equations that relate their outputs. The analysis and methods presented in this paper apply to such systems.

Failure detection and isolation are relatively easy in electronic subsystems, because they can be modeled very accurately and all their inputs can be measured. They can be diagnosed accurately by comparing their outputs to outputs of their models excited by the same inputs. The detection of failures in mechanical, electromechanical, and thermofluid systems, on the other hand, is an open problem. Their dynamical models have unavoidable errors, such as the uncertain values of aircraft stability derivatives, and they are excited by unmeasurable noise, such as wind gusts. In the presence of modeling errors and/or noise, failures cannot be detected by comparing the response of the monitored system to "the correct" response because there are infinitely many responses that correspond to the unfailured operation of the system.

The effect of noise on failure detection is well understood; there are several effective techniques for failure detection and isolation in the presence of noise. They all use window averaging to reduce the contribution of noise to the detection function while not reducing the failure signature. These techniques include the sequential probability ratio test (SPRT),[1] the generalized likelihood ratio (GLR) test,[2] and the weighted sum square residual (WSSR) test.[3]

However, in most systems, modeling errors and not noise are the primary obstacle to failure detection. Jet engines, aircraft flight dynamics, servo systems, power generation systems, and others, belong to this category. The importance of modeling errors is reflected in the following quotation from NASA's request for research proposals that followed the study described in Ref. 3: "A fundamental limitation of the performance of this and all similar analytically redundant schemes is the adequacy of the model used to establish the reference upon which the detection/isolation decision is based."[4]

Distinguishing between modeling errors and failures is difficult, because window averaging, which solves the noise problem, does not reduce the effect of the modeling errors. Chow and Willsky[5] recently outlined an ideal solution to the model uncertainties problem in FDI systems: "An ideal simplistic ap-

Received Feb. 6, 1987; revision received July 10, 1987; presented as Paper 87-2604 at the AIAA Guidance, Navigation and Control Conference, Monterey, CA, Aug. 17–19, 1987. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1987 All rights reserved.

*Member, Technical Staff, Aerospace Technology Center.

proach to designing a robust FDI system is to include all uncertainties in the overall problem specification; then a robust design is obtained by optimizing (in some sense) the performance of the entire system with the uncertainties present. However, this generally leads to a complex mathematical problem that is too difficult to solve in practice." The notion that the problem is too complex to be solved accurately in practice is reflected in the literature. Only a few methods have been proposed to deal with the problem, none of which offers an effective solution. They can be classified into three categories.

The simplest approach is to neglect modeling errors in the design of the algorithm, and then to apply an additional threshold to the detection function to prevent false alarms due to the neglected errors. This, however, is a poor solution because the threshold must be large enough to prevent false alarms when the modeling errors effects are largest. Therefore, it is too large most of the time and thus prevents the detection of small failures. Clark[6] applied this approach to a failure detection system for a hydrofoil boat. He showed that for a fixed maneuver his algorithm is robust for the simple case of two separate parameter changes of about 10%. He noted that in a practical situation his detection thresholds would have to be variable and dependent on the activity of the system. However, he did not propose a method for computing the variable thresholds, neither did he address the problem of simultaneous variations of many system parameters.

The second category includes methods that estimate the effect of modeling errors, and compensate the detection algorithm for them. The simplest approach in this category is to model the errors as additive noise and then to apply an available method for FDI in the presence of noise. This, however, is a bad solution to the problem because it neglects the fundamental property of modeling errors, i.e., that they affect the system through products of states and errors or inputs and errors and not as additive disturbances.

Leininger[7] developed a GLR-based scheme in which he first estimates the model mismatch effects during the unfailed operation of the system, and then detects sensor failures while removing the mismatch effect from the computations. His method applies to systems in which the modeling errors are constant and the system inputs are constant set-points. The assumption of constant modeling errors allows him to identify their effect, from which point on he can treat the system model as exact. Unfortunately, this approach does not apply to most practical systems in which both the modeling errors and the inputs can vary.

Kosut et al.[8] developed a method for computing optimal thresholds for innovations-based failure detection algorithms for systems with model uncertainty. It applies to systems in which the model uncertainty can be modeled by a frequency response matrix with a bounded norm. Their method utilizes a model error filter that operates on the system input and produces a time-varying threshold that is added to the noise threshold in order to prevent false alarms due to the modeling errors. They also derived an estimate of the minimum detectable failure. The main drawback of this approach is its conservativeness due to the use of matrix norms in the specification of the model uncertainty and in the derivation of the model error filter. Different input-output pairs in practical multivariable systems have different nominal responses and different model uncertainties associated with them. The information on these differences is lost once norms are introduced, leading to larger than necessary thresholds and, therefore, larger than necessary smallest detectable failures.

Emami-Naeini et al.[9] presented a similar method for determining the optimal threshold, and combined it with optimally robust filters designed using the internal model principle and frequency shaping. They also developed a computer-aided design technique for computation of thresholds that are optimal for a specific class of inputs, such as steps or ramps. As pointed out by Clark[6] and as explained in detail in the next section of this paper, an effective failure detection algorithm must use varying thresholds that are based on the varying states and inputs of the system. This method[9] cannot compute the thresholds for multivariable systems in real time because the computation is very complicated. Nor can it precompute a table of thresholds to be scheduled in real time, because in practical systems the input signals are complex and not known in advance.

The third approach to failure detection in systems with imperfect models is to use algorithms that are insensitive to the model uncertainty. Robust parity relations[4,5] belong to this category. Parity relations are very effective in detecting failures in dynamic systems with exact models. The robust parity relations are an attempt to use their method in the presence of model uncertainty. This approach suffers from three fundamental drawbacks that limit its use.

The first and the most fundamental problem is that robust parity relations resolve the problem of modeling errors by avoiding rather than by accounting for them. They use for failure detection only that part of the system that can be modeled accurately, and avoid the uncertain parts of the system. This is done by selecting only the most robust parity relations.[4] This may result in only a partial coverage of the failure modes, those observable through measurements that are not affected significantly by the modeling errors. A sensor that is separated from the rest of the system by an uncertain subsystem will be neglected by robust parity relations, because there is nothing in the method to account for the wide range of outputs that the sensor could produce as the uncertain subsystem changes. However, the detection of failures in such sensors is the primary purpose of failure detection methods for uncertain systems.

The second limitation, also observed by Chow and Willsky,[5] is that the parity coefficients should be recomputed at each time step because they depend on the state and the recent history of the system inputs. The computation of the coefficients, however, cannot be done in real time because it involves a complex optimization. Chow and Willsky[5] resolve this problem by suggesting the precomputation of the coefficients for many operating points and scheduling them according to the operating conditions. This approximation implies that the method can only be applied to systems in which the inputs are slowly varying set-points; it cannot be used during maneuvers characterized by inputs that vary at a speed comparable to the speed of response of the system.

The third limitation of robust parity relations is their inability to handle accurately the amount of uncertainty present in practical systems. The jet engine model used as example in Ref. 4 has 63 uncertain parameters. Therefore, there are over 9×10^{18} combinations of system parameters if only the two extreme values of each uncertain parameter are considered. The robust parity relations in Ref. 4, however, are designed using a few "representative" models, e.g., about one model for 10^{18} systems. Since one representative model cannot represent accurately 10^{18} systems, the computed parity check can be very inaccurate. In practice one would have to use a large empirically determined threshold to prevent false alarms caused by the inaccuracy of the method. This large threshold would prevent the detection of small failures.

A common drawback to all the proposed techniques is their inability to distinguish between uncertain parameters that are constant and those which are time-varying. Treating both cases as if they were the same adds even more to the inaccuracy of these techniques, because the constant parameters introduce less uncertainty into the system response than the time-varying parameters.

This paper makes two main contributions to the problem of FDI in systems with modeling errors. It first determines the limits of failure detectability in systems with model uncertainty. The argument is not based on mathematics and, therefore, is not limited to any specific algorithm structure. The second contribution is the development of the reachable measurement intervals (RMI) method for failure detection in systems with model uncertainty. RMI performance is the theoretical limit of failure detectability, and its accuracy and computational efficiency have been demonstrated in real-time laboratory experiments.[10]

Limits of Failure Detectability in Systems with Modeling Errors

Failures in dynamic systems are detected by comparing the response of the monitored system to an estimated response computed using a model of the system. The comparison is performed either directly on the measurements, or indirectly using a computed quantity such as estimator innovations. If an exact model of the system is available, even the smallest difference between the measured and the computed responses is an accurate indication of failure. Since in this ideal case the system model does not contain any uncertainty, the test for failures can use a zero threshold.

If modeling errors are present, the model does not represent the system exactly. Consider the following general description of a dynamic system with model uncertainties.

$$\dot{x}(t) = [A_n + A_e + A(q)]x(t) + [B_n + B_e + B(q)]u(t) \quad \text{(1a)}$$

$$y(t) = [C_n + C_e + C(q)]x(t) + [D_n + D_e + D(q)]u(t) \quad \text{(1b)}$$

The constant matrices A_n, B_n, C_n, and D_n represent the nominal model of the dynamic system. The interval matrices A_e, B_e, C_e, and D_e specify the uncertainty of the uncorrelated entries of the model matrices. Each entry of these matrices is an interval defined by lower and upper limits. Without loss of generality, it is assumed that the intervals are symmetric about zero. Then, if only uncorrelated uncertainties are present, the value of a_{ij}, an uncertain entry of the system matrix, is known to lie in the interval $a_{nij} - a_{eij} \leq a_{ij} \leq a_{nij} + a_{eij}$, where $2a_{eij}$ is the interval width. The same convention applies to the other matrices, too. The matrices $A(q)$, $B(q)$, $C(q)$, and $D(q)$ represent the uncertainty due to the correlated entries of the nominal model. The correlation is through a vector q of system parameter that can affect several entries of the system model. The vector q itself is an interval vector, i.e., each of its terms can vary within a prespecified interval.

In some systems, the uncertainty intervals in Eqs. (1) can be related to component tolerances and their values can be determined analytically. If the system model is not available in symbolic form, the parameter uncertainty intervals must be determined experimentally. One simple way to estimate them is through system identification. The unceratinty intervals are those variations of the identified system parameters that cause only a negligible increase in the minimized cost function. We are currently developing a special identification method that will identify the nominal model and the model uncertainties simultaneously, from experimental data.

The model in Eqs. (1) is only aware of the ranges within which the parameters of the unfailed system lie; it does not know their exact values. Thus, the failure detection must be based on comparing the measured response of the monitored system and a very large number of model responses obtainable by varying the uncertain parameters of the system within their acceptable limits. A failure is detected if the measured response cannot be justified by any of the computable responses (and specifically not by the one corresponding to the "worst" parameter deviations), indicating that a change larger than that explainable as modeling errors is present in the system.

The failure detection test must not produce frequent false alarms. Therefore, since the actual values of the uncertain parameters are not known, the test must always assume that the "worst" parameter deviations are present and use an appropriately large detection threshold in order to prevent false alarms. This introduces conservativeness into the test, because the actual parameter deviations are not the worst possible most of the time. Consequently, the use of a threshold that is too large for the actual parameter values of the system results in missed failures. This conservativeness is unavoidable if false alarms are to be prevented. It is the penalty paid for not knowing the parameter values exactly.

The best test for failures in systems with parameter uncertainties is one that does not introduce any additional conservativeness beyond the unavoidable level. It uses at every time step the smallest possible threshold required to prevent false alarms in the presence of the worst possible parameter deviations. The specification of the best test is unique; therefore, there is only one such test and its performance is the limit of failure detectability.

The smallest possible threshold is equal to the largest possible deviation of a measurement (or a function of the measurements) from its value computed with the nominal model parameters. This optimal threshold value guarantees that even the worst parameter variations that are not considered failures, i.e., those that yield the largest measurement deviations, do not produce false alarms. The smallest possible threshold can be computed exactly by simulating the system (starting in the past and ending at the current time), while adjusting the values of the uncertain parameters so as to achieve the maximum possible value of a measurement. The RMI algorithm, described below, determines the maximizing values of the uncertain parameters directly, without searching through the extremely large number of possible combinations. The procedure is then repeated to minimize the values of the measurements. The interval defined by the maximum and the minimum is the reachable measurement interval. It includes all those values for the measurement that the unfailed system could possible generate. The interval defines the smallest possible threshold region; its width is the sum of the smallest possible threshold for positive and negative deviations of the measurement from its nominal value. The smallest possible threshold region represents the unavoidable conservativeness of any failure detection test due to the modeling errors.

Because the smallest possible threshold depends on the past and current system inputs and states, the best test must recompute the smallelst threshold at every time step. Any failure detection method that does not recompute the smallest possible threshold at every time step adds unnecessary conservativeness into the detection process and will underperform the best test. It will not be able to detect failures as small as the best test can without issuing frequent false alarms.

The RMI Algorithm

The RMI algorithm is capable of computing the reachable measurement intervals, which are the smallest possible thresholds, for a system described by Eqs. (1). It computes the thresholds exactly as outlined above, using an optimization procedure based on the Maximum Principle. The use of the reachable intervals for failure detection is straightforward. It consists of applying an RMI algorithm to each sensor and testing whether each measurement lies within its interval, at every time step. A measurement outside of its interval indicates a failure.

Two versions of the RMI algorithm have been developed. Version 1 is appropriate for systems with parameters that can vary faster than the speed of response of the system,[11] such as stability derivatives of aircraft. This algorithm changes parameter values at every time step of the simulation in order to compute the largest possible deviation of the measurement from the nominal value. Version 2 is for systems with uncertain parameters that vary slower than the response of the system, or are constant but unknown.[11] This algorithm keeps the parameter values constant inside its processing window. It is applicable, for example, to different units from a production series for which only one nominal model is available.

Version 1 is a more general algorithm, because slow parameter variations can be considered a special case of rapid variations. It is described here in detail. Version 2 is then described briefly, with reference to the equations describing version 1. For clarity, the algorithm is formulated first for the case of an exact C matrix and no D matrix. It is then extended to account for uncertain C and D matrices.

It is assumed that the system is stable. Therefore, any transients in the system settle to negligible values in a finite-time t_s. This assumption implies that the outputs of the system at time t_o depends only on the inputs that occurred during the last t_s seconds, to an accuracy selectable by the choice of t_s. Therefore, for any time t_o, it is possible to apply Pontryagin's Maximum Principle[13] over a finite-time window starting at time $t_o - t_s$ and ending at t_o. The principle is used to maximize and to minimize the outputs of the system at time t_o with respect to the unknown values of the entire of matrices A and B. The derivation of the algorithm for the case of maximization of output y_j follows.

Define A as the sum of a nominal matrix A_n and a variation matrix A_v, which represents the combined effect of the uncorrelated and the correlated parameter uncertainties. Matrix B is defined similarlty. Note that A_n and B_n are known and constant, while A_v and B_v are not known exactly and are free to vary within their prespecified tolerances. The system in Eqs. (1) can now be rewritten as

$$\dot{x}(t) = A_n x(t) + B_n u(t) + A_v x(t) + B_v u(t) \tag{2a}$$

$$y(t) = Cx(t) \tag{2b}$$

The performance index[13] required for the Maximum-Principle-based maximization of y_j, the jth output, is simply the value of the output

$$P = c_j x(t_o) \tag{3}$$

where c_j designates row j of the output matrix C, and $x(t_o)$ is the state at the present time (which is the end time of the optimization).

The Pontrayagin H function is given by

$$H = L^T(t)[A_n x(t) + B_n u(t) + A_v x(t) + B_v u(t)] \tag{4}$$

The differential equations for the Lagrange multipliers are obtained by partial differentiation of $-H$ in Eq. (4) with respect to the states x, which yields

$$\dot{L}(t) = -A_n^T L(t) - A_v^T L(t) \tag{5}$$

The terminal boundary conditions on L for this fixed terminal time problem are

$$L(t_o) = c_j^T \tag{6}$$

The initial conditions on x, i.e., the values of $x(t_o - t_s)$, can be set to zero, because t_s was selected to make their effect negligible. Therefore,

$$x(t_o - t_s) = 0 \tag{7}$$

The optimization consists of solving Eqs. (2) and (5) with boundary conditions (6) and (7), while selecting at all times those values of A_v and B_v, that maximize H given by Eq. (4). Note that the maximizing values are either the minimum or the maximum values the parameter tolerances allow. The result of the procedure is a single number, $y_j(t_o)$, which is the largest achievable value of output number j at time t_o. The algorithm is then used a second time to minimize the output, which is accomplished if the signs of the performance index (3) and the boundary conditions (6) are changed. One can then maximize/minimize the other outputs and repeat the entire process for the next time step $t_o + T$.

This optimization problem is extremely difficult to solve. It is a two-point boundary value problem in which the entries of A_v, which appears in Eqs. (2), (4), and (5), frequently switch values between their upper and lower limits. The switching times are determined by the Lagrange multipliers L, for which the initial conditions are not known. The iterative solution of this problem is very time consuming, because at every time step in the processing window the number of possible parameter combinations can be very large. The computational capability required for an exact real-time solution of this optimization problem in typical aerospace applications is orders of magnitude beyond that of today's microprocessors.

At this point an approximation is introduced that makes the problem much easier to solve. The second term in Eq. (5) is neglected to yield

$$\dot{L}(t) = -A_n^T L(t) \tag{8}$$

The reachable measurement intervals computed with the approximate algorithm, with Eq. (8) replacing Eq. (5), are accurate to within a few percent if the entries of A_v are small compared to the entries of A_n (e.g., 30% or less). Fortunately, most systems that are critical enough to require a failure detection system have relatively accurate models. If the unmodeled parameter variations in a system were large enough to produce responses that are significantly different from the nominal response, failure detection in it would be impossible. A better model would have to be derived before any attempt to detect failures in it, with RMI or with any other technique.

In the approximate algorithm, defined by Eqs. (2), (4), (8), (6), and (7), Eq. (8) is independent of the other equations. It can be solved off-line once, by backward integration from the terminal conditions given by Eq. (6). The precomputed time history of the Lagrange multipliers L can be stored in tables. Thus, the maximization/minimization of the outputs can be performed by forward integration of Eqs. (2) from initial conditions Eq. (7), while selecting values of A_v and B_v that maximize the Pontryagin function H defined in Eq. (4). The Lagrange multipliers required in Eq. (4) are taken from the precomputed tables.

The Pontryagin function defined by Eq. (4) is a sum of products of parameters, Lagrange multipliers, and states or inputs. The uncorrelated part of a maximizing parameter value is determined completely by the sign of the Lx or Lu term that multiplies it. The parameter takes on the largest value allowed by the tolerances if the term is positive, and the smallest value otherwise. The correlated parts of the parameter values depend on the optimal values of the correlating system parameters q [see Eqs. (1)]. The values of q are determined first, so as to maximize H. Once q is known, the contribution of the correlated parameters to A_v and B_v can be evaluated. Note that this process is straightforward if $A(q)$ and $B(q)$ are linear functions of q, but can be complicated if the relationships are nonlinear.

The effects of the parameter variations of matrices C and D are added to the measurements after they have been maximized with respect to A and B. The parameters are determined according to the signs of the states and the inputs at time t_o, so as to maximize $C_v x(t_o) + D_v u(t_o)$.

The RMI algorithm described above can be reformulated to be recursive without decreasing its accuracy significantly. The recursive algorithm is computationally 20 to 60 times faster than the nonrecursive one, and is suitable for real-time implementation. The details of the recursive formulation are in Appendix A.

The RMI algorithm for slowly varying parameters (version 2 of RMI) keeps the uncertain matrices A_v and B_v constant in the processing window. The system equations given by Eqs. (2) are augmented by

$$\dot{A}_v(t) = 0 \tag{9a}$$

$$\dot{B}_v(t) = 0 \tag{9b}$$

to reflect the fact that A_v and B_v are unknown constants.

Equations (2-7) apply to this problem too. In addition to the conventional Lagrange multipliers given by Eq. (5), there are also multipliers associated with the unknown entries of matrices A_v and B_v. They are obtained by partial differentiation of $-H$ with respect to the entries of these matrices. The multipliers associated with A_v are

$$\dot{M}(t) = -L(t)x^T(t) \tag{10}$$

and the multipliers associated with B_v are

$$\dot{N}(t) = -L(t)u^T(t) \tag{11}$$

Their terminal boundary conditions are

$$M(t_o) = 0 \tag{12a}$$

$$N(t_o) = 0 \tag{12b}$$

The optimal values of A_v and B_v are determined by maximizing

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{vij}m_{ij}(t_o - t_s) + \sum_{i=1}^{n}\sum_{k=1}^{m} b_{vik}n_{ik}(t_o - t_s) \tag{13}$$

The proof of this condition is in Ref. 13.

The optimization procedure consists of solving Eqs. (2), (9), (5), (10), and (11) with boundary conditions given by Eqs. (6), (7), and (12). The solution process includes the selection of the optimal values of A_v and B_v, which are kept constant in the processing window by maximizing Eq. (13). This problem is also very difficult to solve in the exact form, but can be simplified by the introduction of two approximations. The first is Eq. (8), just as in version 1 of RMI. The second is the modification of Eq. (10) to

$$\dot{M}(t) = -L(t)x_n^T(t) \tag{14}$$

where x_n are the system states as estimated with a model based on the nominal system. The approximate algorithm based on Eqs. (2), (9), (8), (14), and (11) is easier to solve and can be executed recursively, as detailed in Ref. 12.

Both versions of the RMI algorithm consist entirely of backward and forward simulations of a stable system, and are not susceptible to numerical problems other than those involved in the simulations. Therefore, the RMI algorithm can be applied even to very large order systems.

The performance of the RMI algorithm is illustrated on NASA's Hypothetical Turbofan Engine.[14] The engine model and the operating condition are specified in Appendix B. Uncertainty tolerances of ±10% on the entries of the A and B matrices of the engine model were used in the simulations. These tolerances account for the fact that this fourth-order linear model[3] is an approximation to a sixteenth-order nonlinear model of the engine, which itself is only an approximation of the engine behavior. Because of the model reduction and linearization, the parameter tolerances are very complex functions of the uncertainties of the parameters of the sixteenth-order model. In addition to these errors, there are also small differences due to manufacturing and wear between different units from the same production series. It is assumed in this example that the engine and the sensors are noise-free. Furthermore, it is assumed that the engine is operating exactly at its scheduled operating point, so that only errors due to the dynamic portion of the model are present.

The engine model was excited by a correlated Gaussian fuel flow input signal, to simulate the control inputs generated by the controller during flight at a constant flight condition. The reachable fan speed intervals are shown in Fig. 1, as computed with the recursive form of the version 1 RMI algorithm. The sampling time was $T = 20$ ms, and the processing window length was $t_s = 6$ s. At every time step the test for failures consists of checking whether the fan speed measurement is inside its reachable interval. A measurement outside of the interval indicates a failure.

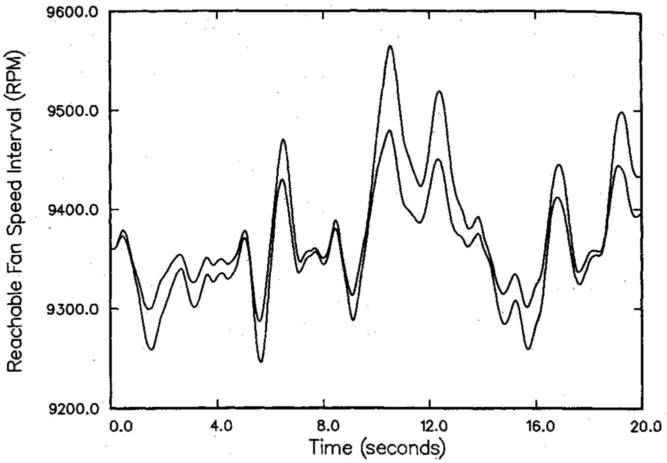

Fig. 1 Reachable fan speed measurement intervals.

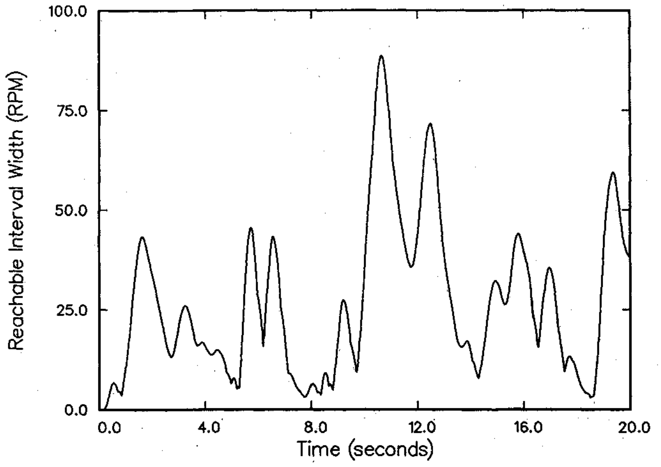

Fig. 2 Width of the reachable fan speed measurement intervals.

The width of the reachable interval, which is equal to the smallest possible threshold region, is shown in Fig. 2. The variable width of the interval reflects the fact that a time-varying threshold is required for effective failure detection in jet engines. If a fixed threshold was used instead, its value would have to be ±44.5 rpm in order to prevent a false alarm in the presence of the worst possible parameter variations at time 10.7 s. At time 18.5 s, when the measurement uncertainty is small, this threshold would have been 25 times too large, leading to poor detection capability. Note that the interval width is closely related to the shape of the fan speed signal. This happens because the modeling errors affect the system through products of errors and states or erorrs and inputs, all of which have a shape similar to the fan speed. However, despite the similarity in shape, the relationship between the fan speed and the reachable interval width is not simple. For example, the nominal fan speed signal in Fig. 1 is 9450 rpm at both time 10.0 and 11.0 s. The reachable interval width at time 11.0 s, however, is twice that at time 10.0 s.

The RMI algorithm for systems with slowly varying parameters (version 2 of RMI) produces intervals that are up to 5% narrower than those in Fig. 2. The small difference between the two algorithms is due to the eigenvalues of the system; two are real and the other two are a complex pair with a damping ratio of 0.96. Because of the eigenvalues, the four Lagrange multipliers have only a few sign changes in the processing window. Therefore, the version 1 algorithm changes parameter only infrequently in the window and uses most of the time the same parameter values as the version 2 algorithm. In more oscillatory systems, such as aircraft flight dynamics,[10,11] the intervals due to rapidly varying parameters can be up to five times wider than those due to slowly varying parameters. In those cases it is important to determine the nature of the modeling errors, and to use that version of the algorithm which describes them best.

The algorithm was implemented in structured Fortran on a Multibus single-board computer based on the 80286/287 processor.[10] It required about 20 ms for the execution of one time step for one sensor. It is estimated that the execution time can be reduced to about 2.5 ms if the 80386/387 processor is used, and if the code is customized to the specific application. Thus, the RMI algorithm, although complex, can be implemented on today's microprocessors and used for failure detection in complex systems.

Including Additional Measurements in the RMI Algorithm

The two versions of the RMI algorithm described above were formulated to verify one measurement against the uncertain model and the system inputs. In this form they are capable of monitoring even single-sensor systems. If more than one sensor is available, it is advantageous to include the additional information in the RMI algorithm. In this form RMI verifies one measurement against the model, the inputs, and the other sensors. The additional sensor information reduces the size of the reachable intervals, because it reduces the uncertainty in the system. Smaller reachable intervals make detection of smaller failures possible.

During the maximization/minimization of a measurement the RMI algorithm selects parameter values that determine the time histories of all the model states. If additional measurements are available but not considered by RMI, discrepancies will exist between the computed states and the measurements. The algorithm will assume that the worst possible parameter variations occur and produce the worst possible time histories of the model states. The measurements, on the other hand, reflect the fact that neither the worst parameter values nor the worst states occur most of the time. If an additional available measurement is a system state, this problem can be resolved by removing the state and the corresponding row and column from the system matrix, and adding a new input term defined as the product of the removed column and the measured state. The RMI algorithm can then be applied to the reduced system, provided it is stable.

In the general case, however, not all the measurements are states, and the reduced system is often unstable. For these cases, an RMI algorithm that utilizes an observer-like structure to account for the additional measurements and is applicable to all systems has been developed. Two versions of this algorithm are available. One is applicable to the rapidly varying parameter case and the other to the slowly varying parameter case. The reachable intervals in both versions are computed using the model

$$\dot{x}(t) = Ax(t) + Bu(t) + K[y(t) - z(t)] \tag{15a}$$

$$y(t) = C_1 x(t) \tag{15b}$$

where the matrix C_1 includes only those rows of the matrix C from Eqs. (2) that correspond to the additional measurements, and z is the vector of measurements corresponding to the computed measurements y. The gain matrix K assures that the system outputs are close to the measurements during the optimization, thus preventing the optimization routine from using state values that disagree with the measurements. The primary concern in selecting the gains is the closeness of the measured and the estimated outputs and not the location of the poles of the closed-loop system. A wide range of gains that produced good results was found for all of the systems to which this approach was applied.

The dynamics of the Lagrange multipliers L are given by Eq. (5) even when the RMI algorithm is based on Eqs. (15). Although the system in Eqs. (15) does have a closed-loop system matrix $(A + KC_1)$, its response remains similar to the open-loop system described by Eqs. (1), provided the gains K are properly designed and $y(t) \approx z(t)$. Therefore, the Lagrange multipliers L must be based on the open-loop system matrix A.

The effect of the additional measurements on the reachable intervals is illustrated on the jet engine model used above. Figure 3 shows the reachable fan speed intervals for the case when the compressor speed is measured and known to be correct. The matrix of additional measurements was $C_1 = [0\ 1\ 0\ 0]$, and the gain matrix was $K = [0\ -10\ 0\ 0]^T$. The width of the interval is shown in Fig. 4. The average width reduction with respect to Fig. 2 is 28%, and it is as high as 62% at some points. Since smaller thresholds result in better detection capability, this algorithm should be used whenever additional measurements are available.

Isolation of Sensor Failures

In most industrial applications, the failure detection and isolation system has to monitor several sensors simultaneously. The RMI algorithm is a powerful building block for such systems in applications with modeling errors.

The simplest multiple-sensor RMI failure detection and isolation system uses one RMI algorithm (without additional

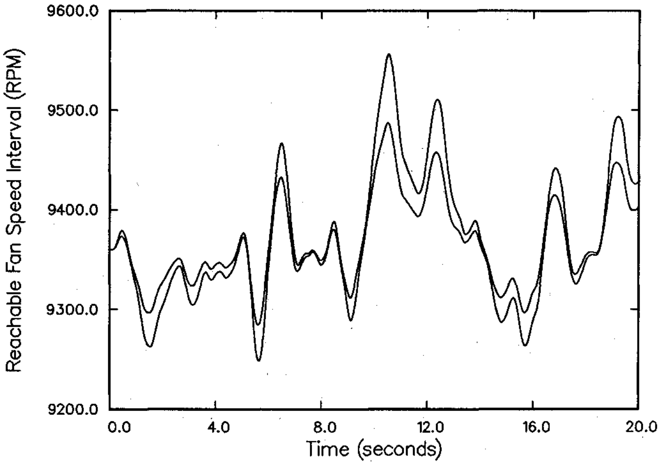

Fig. 3 Reachable fan speed measurement intervals (compressor speed measurement considered).

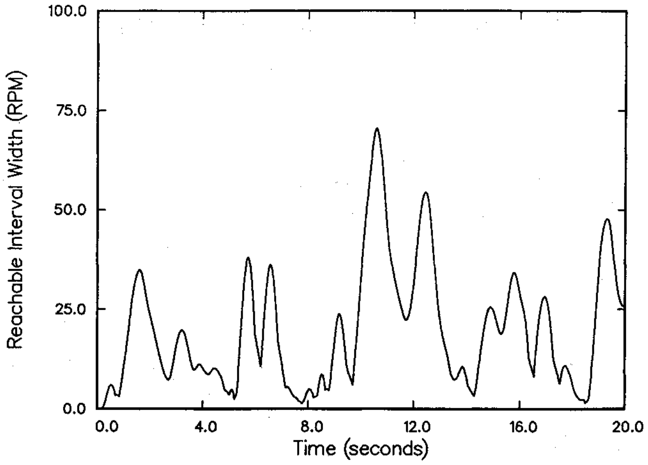

Fig. 4 Width of the reachable fan speed measurement intervals (compressor speed measurement considered).

measurement information) for each sensor. This arrangement provides the coverage of all the sensors and can handle multiple sensor failures without any additional processing. The status of each sensor is determined by its RMI algorithm. The disadvantage of this approach is its conservativeness due to neglecting the other available measurements, as explained in the previous section.

A more complex system uses one RMI algorithm with additional measurement information for each sensor. This system uses narrower reachable measurement intervals and, therefore, can detect smaller failures than the system that neglects the other measurements. However, the additional measurements open the possibility for an incorrect failures isolation decision. In the jet engine example in the previous section, for example, a failed compressor speed sensor could cause the RMI algorithm for the fan speed sensor to indicate a failure even if the fan speed sensor has not failed.

This problem can be resolved by applying more than one RMI alogrithm to each sensor. In the most general case, in a system with N sensors, $N-1$ RMI algorithms can be used for each sensor, each algorithm using all but one additional available measurement. The failure of sensor j is isolated if the $N-1$ algorithms applied to it indicate a failure, and the $N-1$ RMI algorithms in the system that neglect sensor j do not. This logic can be extended to handle multiple sensor failures.

The primary disadvantage of this strategy is the need to execute $N(N-1)$ RMI algorithms simultaneously, in the general case. In practice, however, this number is lower, because sensors that do not reduce the width of the reachable intervals significantly are not used as sources of additional information. The problem can be simplified even further by exploiting the fact that over one-half of the required computations in RMI algorithms, which differ only in the additional measurements they use, are identical. Therefore, a routine that executes N simultaneous RMI algorithms for a sensor requires only about $N/2$ times the CPU times required for a single algorithm, if the common computations are performed only once. It is estimated that the monitoring of five jet engine sensors, as done in Ref. 3, would require only about eight times as many computations as a single RMI algorithm, in spite of the fact that $N(N-1)=20$. Such a system could be implemented on any one of the recently introduced 32-bit/20 MHz microprocessors with algorithm cycle time of 20 ms, provided the code is efficient and application-specific.

Handling Noise

It has been assumed up to this point that the monitored dynamic system is noise-free. This is an accurate approximation for systems in which the effect of modeling errors is much larger than the effect of noise. In many systems, however, sensor and process noise cannot be neglected. Two approaches are possible for including the effect of noise in RMI-based failure detection methods.

The first method includes the effect of additive process and sensor noise in the general model of the dynamic system given by Eqs. (1), by adding to it noise vectors.[11] Each entry of these vectors is an interval that bounds the magnitude of the noise. If the noise is known to be Gaussian, the intervals are set at $\pm 3\sigma$. The RMI algorithm then automatically computes reachable intervals that are wide enough to prevent false alarms due to the noise. The RMI algorithm for a slowly varying parameters represents the noise as a bias that is constant in the processing window. The algorithm for rapidly varying parameters repesents it as a constant-amplitude square wave with time-varying period. This approach is simple, but too conservative. It produces reachable intervals that are wide enough for the largest possible noise peak, but too wide most of the time. This conservativeness results in missed failures. Reference 11 presents more details about this approach, including an example.

The second approach for handling noise is appropriate for systems in which a more accurate treatment of noise statistics is required. It is based on the fact that sensor noise covariances computed with the nominal model are a good approximation to the covariances of the monitored system if its parameter variations are relatively small. The failure detection method consists of computing the reachable intervals using the RMI algorithm as if noise were not present, and computing the sensor noise covariances as if modeling errors were not present. The test of failures then reduces to determining whether the fact that a measurement exceeds its reachable interval boundary is statistically significant in the presence of the noise. For example, if half the reachable interval width and the standard deviation of the noise are equal, the crossing of the interval boundary by the measurement at a single time step is not a statistically significant indication of failure. But if the measurement remains outside of the interval for a long enough period of time, it is an indication of failure. A statistically rigorous decision algorithm for this problem, suitable for real-time implementation, is currently under development.

Conclusions

This paper presents an analysis of the failure detection problem in systems with modeling errors. It introduces the concept of the reachable measurement interval, which represents the worst possible effect the modeling errors can have on the system response. The reachable measurement interval was shown to be the smallest possible threshold that can be applied to the difference between a measurement and its estimate in order to determine the presence of failures without causing frequent false alarms due to the modeling errors. Therefore, this threshold allows the detection of the smallest reliable detectable failures.

The RMI failure detection method for systems with modeling errors, introduced in the paper, is a practical realization of the developed concepts. The method utilizes a Maximum-Principle-based optimization algorithm to compute the reachable intervals. The algorithm can be executed recursively, and is computationally efficient enough to allow real-time monitoring of complex multisensor systems when implemented on today's microprocessors. A jet engine example was used to illustrate the method.

The RMI method, combined with a modeling errors identification method currently under development, form an integrated tool for analysis, design, and implementation of failure detection systems in applications with modeling errors.

Appendix A: Recursive RMI Algorithm

The nonrecursive RMI algorithm, described above, performs a real-time simulation of a multivariable system over the entire processing window at every time step. Since up to 500 steps may be required in the time window in order to achieve fine enough resolution of inputs, states, and Lagrange multipliers, the computational load required for monitoring large systems can be significant. However, the algorithm can be simplified significantly through the introduction of an approximation that makes its recursive reformulation possible. In recursive form, the algorithm computes the reachable interval at time t_o+T from its value at t_o, without resimulating the system over the entire processing window.

To facilitate the recursive solution, Eq. (2a) is modified to

$$\dot{x}(t)=A_n x(t)+B_n u(t)+B_I u_v(t) \tag{A1a}$$

$$u_v(t)=A_v x_n(t)+B_v u(t) \tag{A1b}$$

where B_I is the identity matrix. In Eq. (A1b) the term $A_v x_n(t)$ replaced the term $A_v x(t)$ from Eq. (2a). With $x_n(t)$, which is the state of the nominal system at that time as estimated by the nominal system model, recursive computation becomes possible. As with the approximations used above, the errors introduced here are small if the parameter variations are relatively small and do not cause significant changes in the response of the system.

The recursive algorithm is based on the following discrete version of Eq. (A1):

$$\mathbf{x}(k+1)=A_{nd}\mathbf{x}(k)+B_{nd}\mathbf{u}(k)+B_{Id}\mathbf{u}_v(k) \tag{A2a}$$

$$\mathbf{u}_v(k)=A_v\mathbf{x}_n(k)+B_v\mathbf{u}(k) \tag{A2b}$$

The matrices A_{nd}, B_{nd}, and B_{Id} are generated through a zero-order-hold transformation of A_n, B_n, and B_I from Eq. (A1a). Note that the parameter variations A_v and B_v remain in their continuous-time form.

The state of the system at the terminal time during the maximization of an output can be computed by repetitive application of Eq. (A2a) and is given by

$$\mathbf{x}(p)=\sum_{k=m+1}^{p} A_{nd}^{p-k}[B_{nd}\mathbf{u}(k-1)+B_{Id}\mathbf{u}_v(k-1)] \tag{A3}$$

The discrete time mT corresponds to time $t=t_o-t_s$, and the discrete time pT corresponds to time $t=t_o$, the current time. The initial state $\mathbf{x}(m)$ in Eq. (A3) was set to zero according to Eq. (7). There exists a recursive relation between $\mathbf{x}(p)$ and $\mathbf{x}(p+1)$, the maximized state one time step later. It is derived from Eq. (A3) and is given by

$$\mathbf{x}(p+1)=A_{nd}\{\mathbf{x}(p)-A_{nd}^{p-m-1}[B_{nd}\mathbf{u}(m)+B_{Id}\mathbf{u}_v(m)]\}$$

$$+B_{nd}\mathbf{u}(p)+B_{Id}\mathbf{u}_v(p)+\sum_{q\epsilon R}A_{nd}^{p-q}B_{Id}\mathbf{u}_c(q) \tag{A4}$$

The first term in the equation removes from the summation in Eq. (A3) the contribution of the oldest inputs, which are now outside of the processing window. The second and third terms add to the summation the contribution of the latest inputs that entered the window at this time step. The challenging part of the recursive solution is accounting correctly for the varying parameters. This is done by the last term in Eq. (A4), which is now explained.

The values of $\mathbf{u}_v(k)$ in Eq. (A2) depend on the signs of $\mathbf{L}(k)$, because A_v and B_b are determined so as to maximize $\mathbf{L}^T(k)\mathbf{u}_v(k)$ in Eq. (4). As the time window moves forward, $\mathbf{L}$ moves with it, while $\mathbf{x}_n$ and $\mathbf{u}$ do not. Therefore, some values of $\mathbf{x}_n$ and $\mathbf{u}$ that correspond to positive values of $\mathbf{L}$ in the processing window ending at time pT may correspond to negative values of $\mathbf{L}$ in the window that ends at time $(p+1)T$, and vice versa. The number of such changes is equal to the number of sign changes in $\mathbf{L}$, and their locations in the window are stored in the set R. This effect can be corrected for in the following way. When $\mathbf{u}_v(k)$ is computed to maximize $\mathbf{L}^T(k)\mathbf{u}_v(k)$, a correction term $\mathbf{u}_c(k)$ is also computed and stored. The correction term is defined so that $\mathbf{u}_v(k)+\mathbf{u}_c(k)$ gives that value of $\mathbf{u}_v$ that would have been used if the signs of $\mathbf{L}(k)$ had been different. The correction for the Lagrange multiplier sign changes then consists of adding the stored correction terms, delayed by the appropriate powers of A_{nd}, to the expression for the new maximized state $\mathbf{x}(p+1)$. Once a correction is applied, the sign of its stored value is changed to correspond to the new value of $\mathbf{u}_v$ that is now in effect.

The eigenvalues corresponding to the backward solution of the Lagrange multiplier dynamic equations, given by Eq. (8), are the same as those of the nominal system, given by Eqs. (2). For relatively well-behaved systems the number of sign changes (oscillations) in $\mathbf{L}$ is small. Therefore, the recursive algorithm replaces the simulation of Eqs. (2), which must be performed over the entire processing window for every time step, by an update and a few corrections, as shown in Eq. (A4).

The use of the nominal states in the system matrix variation term in Eq. (A2b) makes the reachable output intervals symmetrical with respect to the nominal outputs in cases where the parameter tolerances are symmetrical with respect to the nominal parameters. Therefore, once the maximum reachable output is computed, the minimum output can be computed without another optimization, reducing the total computational load to half. In numerous examples, which included models of jet engines, flight dynamics, and nuclear reactors, the CPU time requirements of the recursive algorithm were 20 to 60 times less than those of the nonrecursive one, and small enough to make real-time implementation of the RMI algorithm possible.[10]

Appendix B: Jet Engine Model

The program in Ref. 14 generates the following linearized model for the flight condition 9200m/0.6M, PLA = 83 deg. The states of the model are: fan speed (rpm), compressor speed (rpm), burner exit slow response temperature (K), and fan turbine inlet slow response temperature (K). The inputs are: main burner fuel flow (kg/s), nozzle jet area (m^2), fan guide vane position (degrees), high compressor variable stator vane angle (degrees), and customer compressor bleed flow (percent). The outputs are: fan speed (rpm) and compressor speed (rpm).

Matrix A_n:

−6.067	4.746	−0.378	1.596
0.110	−3.304	1.441	−4.196
1.689*E*−2	−5.514*E*−3	−0.708	5.606*E*−3
0.188	−4.213*E*−2	−3.805*E*−2	−2.126

Matrix B_n:

1.412*E*−4	0.639	−1.588*E*−2	1.498*E*−2	−1.811
7.091*E*−5	0.145	−1.199*E*−3	−1.079*E*−2	−0.631
4.515*E*−6	9.292*E*−3	−8.005*E*−5	−1.084*E*−5	4.382*E*−2
5.491*E*−6	4.575*E*−3	2.057*E*−4	−7.878*E*−5	0.180

Matrix C_n:

1.000	0.000	0.000	0.000
0.000	1.000	0.000	0.000

Matrix D_n:

0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000

References

[1] Deckert, J.C., Desai, M.N., Deyst, J.J., and Willsky, A.S., "F-8 DFBW Sensor Failure Identification Using Analytic Redundancy," *IEEE Transactions on Automatic Control,* Vol. AC-22, No. 5, Oct. 1977, pp. 795-803.

[2] Willsky, A.S., "A Survey of Design Methods for Failure Detection," *Automatica,* Vol. 12, Nov. 1976, pp. 601-611.

[3] Beattie, E.C., Laprad, R.F., Akhter, M.M., and Rock, S.M., "Sensor Failure Detection for Jet Engines," NASA CR-168190, May 1983.

[4] Weiss, J.L., Pattipati, K.R., Willsky, A.S., Eterno, J.S., and Crawford, J.T., "Robust Detection/Isolation/Accommodation for Sensor Failures," NASA CR-174797, Sept. 1985.

[5] Chow, E.Y. and Willsky, A.S., "Analytical Redundancy and the Design of Robust Failure Detection Systems," *IEEE Transactions on Automatic Control,* Vol. AC-29, July 1984, pp. 603-614.

[6] Clark, R.N., "Instrument Fault Detection," *IEEE Transactions on Aerospace and Electronic Systems,* Vol. AES-14, No. 3, May 1978, pp. 456-462.

[7] Leininger, G.G., "Model Degradation Effects on Sensor Failure Detection," *Proceedings of the 1981 Joint Automatic Control Conference,* Vol. II, June 1981.

[8] Kosut, R.L., Walker, R.A., and Shah, S.C., "Robust Fault Detection, Isolation, and Accommodation to Support Integrated Aircraft Control," *Proceedings of the 1983 AIAA Guidance and Control Conference,* AIAA, New York, Aug. 1983, pp. 14-21.

[9] Emami-Naeini, A., Akhter, M.M., and Rock, S.M., "Robust Detection, Isolation, and Accommodation for Sensor Failures," NASA CR-174825, July 1986.

[10] Horak, D.T. and Allison, B.H., "Experimental Implementation and Evaluation of the RMI Failure Detection Algorithm," *Proceedings of the 1987 American Control Conference,* Vol. 3, June 1987, pp. 1803-1810.

[11] Horak, D.T. and Goblirsch, D.M., "Reachable Outputs in Systems with Bounded Parameter Uncertainties: Application to Failure Detection," *Proceedings of the 1986 American Control Conferenc,* Vol. 1, June 1986, pp. 301-308.

[12] Horak, D.T. and Goblirsch, D.M., "A Failure Detection Method for Dynamic Systems with Imperfect Models," American Society of Mechanical Engineers Paper 86-WA/DSC-21, Dec. 1986.

[13] Fan, L.-T, *The Continuous Maximum Principle,* J. Wiley, New York, 1966, pp. 12-28.

[14] Merrill, W.C., "HYTESS-A Hypothetical Turbofan Engine Simplified Simulation," NASA TM-83561, Jan. 1984.