Proceedings

1994 IEEE International Conference on
Robotics and Automation

May 8 - 13, 1994
San Diego, California

Sponsored by

IEEE Robotics and Automation Society

IEEE Computer Society Press
Los Alamitos, California

Washington • Brussels • Tokyo

Proceedings of the *IEEE International Conference on Robotics and Automation*, San Diego, CA (May 1994).

New Dynamic Model-Based Fault Detection Thresholds for Robot Manipulators

M. L. Visinsky, I. D. Walker, and J. R. Cavallaro
Department of Electrical & Computer Engineering
Rice University, Houston, TX 77251-1892

Abstract

Autonomous robotic fault detection is becoming increasingly important as robots are used in more inaccessible and hazardous environments. Detection algorithms, however, are adversely effected by the model simplification, parameter uncertainty, and computational inaccuracy inherent in robotic control, leading to an unacceptable number of false alarms and overzealous fault tolerance. The algorithms must use thresholds to mask out these errors. Typically, the thresholds are empirically determined from a specific robot trajectory. The effect of modeling inaccuracy, however, fluctuates dynamically as the robot moves and failures occur. The thresholds need to be dynamic and respond to the changes in the robot system so as to differentiate between real failures and misalignment due to modeling errors. This paper first summarizes the Reachable Measurement Intervals (RMI) method of computing dynamic thresholds and then, learning from the robot-oriented analysis of RMI, presents a more efficient threshold generation method using the manipulator dynamics property of linearity in parameters.

1 Introduction

Fault detection and tolerance are increasingly important for space and hazardous environment robotics. Long distances and inaccessibility complicate a remote operator's ability to quickly detect, cope with, or repair failures internal to the robot systems. Using local fault detection and tolerance algorithms, however, the robot is able to effectively isolate faulty components and continue performing its tasks without the need for immediate human intervention.

Fault detection algorithms monitor systems for failures by comparing measured system outputs with other similar outputs or with the corresponding expected output derived from a model of the system behavior. In previous work [9, 12, 13, 14], we have developed comparison tests for a variety of robots using analytical redundancy, a method which reveals the existing functional redundancy in a given system and allows us to perform fault detection without modifying or expanding the current hardware. In their pure form, the ideal tests derived from analytical redundancy cause an unacceptable number of false alarms due to the sensor and modeling errors which arise from linearization of the robot equations and inaccuracies in model parameters such as link inertia or mass. In addition, the effect of the model errors and sensor noise fluctuates dynamically as the motion of the robot changes and as failures occur. Dynamic thresholds must be used to mask out the modeling errors and cope with the ever changing robot status while remaining small enough so as not to hide real failures.

There are a variety of algorithms which focus on computing thresholds for fault detection in uncertain dynamic systems [1, 2, 6, 7, 15]. We present a summary of our analysis of the Reachable Measurement Intervals (RMI) algorithm developed and analyzed for aircraft systems by Horak, et al, in various publications [3, 4, 5]. This paper then explores the development of a novel threshold generator, tailored to the special needs of robotics, which utilizes the parameter error effect estimation techniques of RMI while creating a faster, more efficient algorithm. Section 2 summarizes the key points of the RMI algorithm. Section 3 discusses the use of RMI in simple robot examples and explores its extensibility to more general robotic problems. Section 4 discusses the derivation of our new method for threshold generation and Section 5 presents the initial results of the algorithm compared to RMI. Our conclusions and future work are presented in Section 6.

2 Reachable Measurement Intervals

We briefly summarize the RMI algorithm below. RMI is designed to monitor dynamic systems with

bounded parameter errors described by the equations:

$$\begin{aligned}\dot{x}(t) &= A_n x(t) + B_n u(t) + A_v x(t) + B_v u(t) + W(t) \\ y(t) &= Cx(t) + V(t),\end{aligned} \quad (1)$$

where x is the state vector for the system, u is the known input (or control) signal, and y is the output measurement signal. A_n and B_n are the nominal (linear) system dynamics and input distribution matrices with A_v and B_v being their corresponding variation matrices. A_n and B_n are constant and represent the chosen model for the system linear dynamics. A_v and B_v are the possible errors in the model and are assumed to be within defined limits. The values of the input and output noise vectors, W and V, are also assumed bounded and represent the non-linear effects of the robot dynamics and gravity as well as the sensor inaccuracy. The matrix C is initially assumed exact, but may also vary between bounds as noted near the end of this discussion.

The RMI algorithm provides conditions for finding the extreme possible system outputs $y(t)$ for each input $u(t)$ given bounded parameter variations in A, B, W, and V. Thus, if the measured output exceeds the computed RMI bounds, a fault is inferred since the modeling errors have been accounted for in the bounds. The focus of RMI is the Hamiltonian system produced by:

$$H = L(t)^T [A_n x(t) + B_n u(t) + g_v(t)]. \quad (2)$$

The variance vector, $g_v = A_v x(t) + B_v u(t) + W(t)$, combines the current control and the possible errors in the coefficient matrices. L is a vector of Lagrange multipliers described by the approximation [3, 4, 5]

$$\dot{L}(t) = -A_n^T L(t), \quad (3)$$

with the terminal boundary condition (for the j^{th} sensor thresholds)

$$L(t_f) = c_j^T, \quad (4)$$

where c_j is the j^{th} row of the measurement matrix C.

The RMI procedure solves equations (1) and (3) with appropriate initial conditions on x [10] and boundary condition (4), while selecting at each time values of A_v, B_v, and W which maximize (and then minimize) H in equation (2). Note that in H, the terms $A_n x(t)$ and $B_n u(t)$ are unaffected by the choice of the variations. These terms are essentially constant and can be ignored in the maximization/minimization process. The procedure is started at a known state in the past history of the trajectory and run for a specific window of iterations to reach the current time. The history window must be large enough so that the output of the system at the end of the window (the current time) depends only on the inputs which occurred during the chosen window of time [4]. By calculating the error effects for the full history window every controller cycle, RMI thus takes into account the global effects of the parameter uncertainty [10].

Horak, et al, assume that the parameter variations are uncorrelated so that they can maximize H by maximizing the individual terms $(L^T A_v x), (L^T B_v u)$, and $(L^T W)$ [4]. However, in a robot system, the A_v, B_v and W matrices rely on the same parameters. By maximizing each term separately, the possibility arises that in one instant of time, the variance of a parameter (mass or length in the robot, for example) will be considered positive to maximize one term and negative to maximize another. It is, of course, not possible for the robot to have a parameter be both heavier and lighter (or, for length, longer and shorter) than expected at the same instant in time.

To account for the correlation of the parameters between the matrices, the elements of A_v, B_v, and W must be chosen as a set for the maximization/minimization process based on the same extreme variation for each correlated parameter. Each parameter has an upper and lower bound corresponding to the assumed range of possible error for that parameter. If there are n correlated parameters, then there are 2^n combinations of the parameter extremes which produce 2^n different sets of A_v, B_v and W matrices. These sets can be pre-computed and stored in a look-up table. At each iteration of the RMI procedure, these matrix sets are checked to determine which set in combination with the current x_{max} and u will maximize H and which set with x_{min} and u will minimize H. These two matrix sets are then used in equation (1) to produce the next x_{max} and x_{min}. More details are given in [10].

Note that, while the terms maximum and minimum are used in this discussion, the results do not correspond to the largest and smallest variation of the parameter errors (the smallest error would be zero, hopefully). Rather, the terms correspond to a maximum and minimum output (y_j) found by combining the various parameter extremes in the calculation of the output state. This produces a range bounding the real output which covers any parameter errors within the chosen extremes.

The effect of the noise vector V is included after the process is complete by adding the upper limit of the variation for V to the maximum bound and the lower limit to the minimum bound. The effect of a

variable C matrix (which would imply that the sensor models are not completely accurate) can also be added at this point by chosing the variations so as to maximize/minimize each output y_j. The result of the RMI procedure is two numbers per output measurement which represent the extreme achievable values of that output at each time t_f given bounded modeling errors. These numbers bracket the desired value for a particular state (position, velocity, etc.) and are used as near optimal thresholds which mask out the affects of modeling errors using the smallest thresholds possible with the given approximations and assumptions.

3 Robotic Fault Detection Using RMI

In this section, we assess the problems associated with the RMI algorithm when applied to robotics. We initially focus on the algorithm applied to a single-joint (pendulum) to demonstrate the procedure and subsequently explore RMI for a two-joint robot in order to study both the expandability of the algorithm and the effects of joint coupling on the size of the derived thresholds. The matrices A_n, B_n, A_v, B_v, and W depend on various parameters of the given system (such as mass and length of a robot link). The nominal or expected values of these parameters are given in the robot specifications along with bounded ranges over which each parameter is expected to vary.

3.1 Pendulum RMI

The pendulum and its nominal parameters were chosen as follows. The joint is rotational with the angle of rotation denoted by θ. The mass is assumed to be concentrated at the end of the joint rod. In this study, the nominal mass, m, is chosen as 10 kg and its corresponding uncertainty, $\hat{m}$, varies from 9 kg to 11 kg (i.e. has a 10% error). The nominal length, l, is 1 m with an assumed error of 1% which means $\hat{l}$ has an expected range of 0.99 m to 1.01 m.

For this analysis, $x = [\theta\ \dot{\theta}]^T$ and the sensors are assumed exact so that the output equation is simply $y(k) = x(k)$. We assume a robot model with two sensors (encoder and tach) per joint whose measurements correspond to y_1 and y_2 respectively. The control, u, is derived so as to damp out errors in the system using PD gains and to compensate for the remaining nonlinear portion of the dynamics (gravity in this case). The control is based upon the nominal estimates, m and l, and in this example is written as:

$$\begin{aligned} u &= ml^2 u_d - ml^2(K_d\dot{\theta} + K_p\theta) + mgl\cos(\theta), \\ u_d &= \ddot{\theta}_d + K_d\dot{\theta}_d + K_p\theta_d \end{aligned} \tag{5}$$

The dynamic equations (1) are in terms of the true parameters $\hat{m}$ and $\hat{l}$ and, when combined with (5), result in the following RMI equation. Note that if the parameters are exact, the variance g_v (which contains the ratios of nominal parameters to true parameters) reduces to zero.

$$\begin{aligned} \dot{x} &= \overbrace{\begin{bmatrix} 0 & 1 \\ -K_p & -K_d \end{bmatrix}}^{A_n} x + \overbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}^{B_n} u_d + g_v, \\ g_v &= \overbrace{\begin{bmatrix} 0 & 0 \\ (1-\frac{ml^2}{\hat{m}\hat{l}^2})K_p & (1-\frac{ml^2}{\hat{m}\hat{l}^2})K_d \end{bmatrix}}^{A_v} x + \\ &\quad \overbrace{\begin{bmatrix} 0 \\ (\frac{ml^2}{\hat{m}\hat{l}^2}-1) \end{bmatrix}}^{B_v} u_d + \overbrace{\begin{bmatrix} 0 \\ (\frac{mlg}{\hat{m}\hat{l}^2} - \frac{g}{\hat{l}}) \end{bmatrix}}^{W} \cos(\theta). \end{aligned} \tag{6}$$

Implementing RMI for the pendulum is relatively straightforward (see [10] for more details). With only two parameters, four different g_v values were possible and four comparisons were made each iteration to find the maximum and minimum output value. The bounds proved to be dynamically variant (see Figure 1) and were generally smaller than the previous empirically determined thresholds which were a constant width of $\pm 0.25 rad/s$. With no coupling effects present in the pendulum, we were also able to see clearly the effects of gravity on the bounds [10].

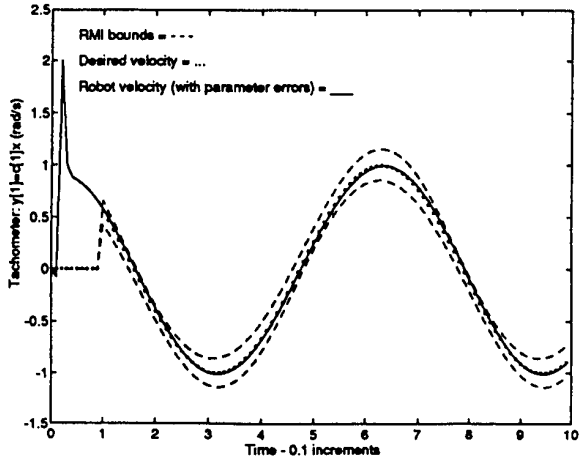

Figure 1: Pendulum Tachometer RMI and robot with parameters: $\hat{m} = 10.7kg, \hat{l} = 1.009m$.

3.2 Multi-joint Robots

For a multi-joint robot, each joint must be handled separately with the effects from other joints treated as

a dynamic disturbance in W in order to keep the RMI algorithm structure the same. RMI is thus performed for each sensor in each joint separately. The two-joint RMI example we implemented performs similarly to the pendulum algorithm with increased variations in the thresholds due to the effects of coupling (Figure 2). When the base joint is held motionless while the other joint moves through a complex trajectory, the effects of coupling can be readily observed without the complications from changing gravity effects. More detailed results are shown in [10].

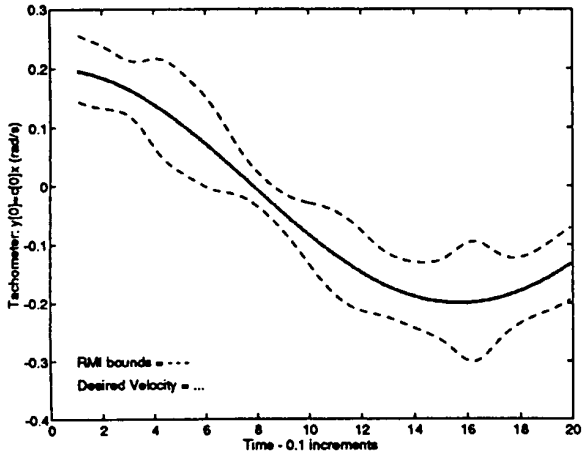

Figure 2: Two-Joint Encoder RMI for Joint 1 - both joints in motion.

The two-joint example revealed, however, that the RMI equations become much more complex as the number of joints increases. The derivation of the equations becomes more difficult for larger robots. In addition, these complex equations must be repeated for each iteration in the history window and for each sensor at each joint. The window used in our implementation of RMI takes one hundred internal iterations to produce the thresholds at each robot controller cycle in order to damp out transients. The algorithm can be parallelized at the joint level, but the internal history iterations greatly increase the time complexity of the algorithm.

Unfortunately, the run-time computational complexity also increases as the number of uncertain parameters, and thus the number of comparisons necessary at each RMI iteration, increases. In moving from the single to the two-joint example, the number of parameters increased from 2 to 4 and the number of comparisons per sensor jumped geometrically from 4 to 16. RMI can be transformed into a dynamically recursive program [3] to help alleviate some of the computational burden. However, even the recursive algorithm requires a full 2^n comparisons (n= total number of parameters) per state variable per joint. For full six or eight joint robots, the resulting 2^{6p} or 2^{8p} comparisons (where p is the number of parameters per joint) might thus keep the algorithm from being successful in real-time. Note that while the various sets of coefficient matrices can be precomputed and pulled from a table, g_v must be re-computed each iteration for each set of coefficient matrices with the changing state, x, and control,u_d.

4 Dynamic Threshold Alternatives

RMI is designed for systems with primarily linear dynamics, requiring special tailoring to fit the nonlinear dynamics of robot manipulators. As noted in Section 3.2, RMI is also not readily scalable to larger robots. The time needed to compute bounds for robots with many uncertain parameters or multiple joints prohibits a real-time response to failures for the detection algorithms. Even the recursive algorithm requires too many comparisons per iteration to be real-time. If the robot cannot make a fast decision about a possible failure, the errors rapidly escalate and endanger the robot and the environment. Previous analysis [9] has shown that with even a simple stuck sensor failure, it only takes a few iterations of the controller before the robot begins to swing wildly out of control (see also Figures 6 and 7).

RMI further requires a lot of memory to maintain the various matrix sets used in finding the maximum and minimum variances. Increasing the number of joints in the robot geometrically increases the number of sets needed during the run. Some savings are possible due to repetition within the matrices, but the memory requirements may be too large to justify the autonomy needed for quick, on-board fault detection, especially in costly space robotics.

To overcome these concerns, we are investigating possible alternatives to the RMI concept which still utilize the idea of finding the maximum variance possible due to bounded parameter errors at each state iteration. Towards this end, we have exploited the similarities between RMI and the analytical redundancy (AR) analysis we performed in deriving our detection tests [9, 14]. Several other threshold generation methods have a similar AR format [7, 15]. With AR, we found that only two time steps of history were needed for adequate fault detection in robots. Using only two steps to compute the thresholds would greatly reduce the computation time over the hundred iterations per controller cycle needed in our implementation of RMI.

A useful facet of rigid link robot dynamics is the linearity with respect to parameters in the nonlinear equations of motion. As in numerous adaptive control strategies [8], we can separate the parameter-based coefficients from the joint variables to produce a linear relationship in the form

$$\tau = \left[\hat{M}(q)\right]\ddot{q} + \hat{N}(q,\dot{q}) = [Y(q,\dot{q},\ddot{q})]\,\hat{p} \qquad (7)$$

where $\hat{M}$ is the inertia matrix, $\hat{N}$ contains the centripetal and coriolis terms, and $q = \theta$. The matrix Y contains known functions of the state variables $(q,\dot{q},\ddot{q})$ and $\hat{p}$ is a vector of the unknown but bounded real robot parameter coefficients, $\hat{m}$ and $\hat{l}$. As a simple example, the torque equations for a pendulum would be transformed from

$$\begin{aligned} \tau &= \hat{m}\hat{l}^2\ddot{\theta} + \hat{m}g\hat{l}\cos(\theta) \\ \text{to } \tau &= \begin{bmatrix} \ddot{\theta} & \cos(\theta) \end{bmatrix} \begin{bmatrix} \hat{m}\hat{l}^2 \\ \hat{m}g\hat{l} \end{bmatrix}. \end{aligned}$$

In order to develop a threshold generation algorithm, we need to determine the bounds due to modeling inaccuracies on the tracking error ($e = q_d - q$, where q_d is the desired trajectory) for the robot system. The parameter adaptive control methods are more conducive to this goal [6] because of the separation of the parameters in the equations. The inverse dynamics control law uses the given estimates of the system parameters, m and l, in M and N:

$$\tau = M(q)[\ddot{q}_d + K_d\dot{e} + K_p e] + N(q,\dot{q}). \qquad (8)$$

The real parameters in the robot vary from these estimates but are within the range given by the parameter extremes: $m_{min} \le \hat{m} \le m_{max}$ and $l_{min} \le \hat{l} \le l_{max}$. The resulting robot equation is

$$\ddot{q} = \hat{M}^{-1}\tau - \hat{M}^{-1}\hat{N}. \qquad (9)$$

Substituting the torque from (8) into (9), we get

$$\hat{M}\ddot{q} = M[\ddot{q}_d + K_d\dot{e} + K_p e] + (N - \hat{N}). \qquad (10)$$

Changing $\ddot{q}_d$ into $\ddot{e}$ by adding and subtracting $M\ddot{q}$ on the right hand side and rearranging the equation leads to:

$$\ddot{e} + K_d\dot{e} + K_p e = M^{-1}(\tilde{M}\ddot{q} + \tilde{N}), \qquad (11)$$

where $\tilde{M} = \hat{M} - M$ and $\tilde{N} = \hat{N} - N$. Using (7), we can modify the right hand side to separate the parameters from the state variables:

$$\ddot{e} + K_d\dot{e} + K_p e = M^{-1}\,[Y]\,\tilde{p}. \qquad (12)$$

Equation (12) represents the focus of our new Threshold , Model-Based, (ThMB) algorithm. Here, all the unknown portions of the dynamics have been shifted into the parameter error vector $\tilde{p}$. By selecting appropriate combinations of the parameter variations $\tilde{m}$ and $\tilde{l}$ present in $\tilde{p} = \hat{p} - p$, we can calculate the possible extreme errors in the positive and negative direction for the next iteration of both $\dot{e}$ and e. Notice that in the ThMB method, the inherent nonlinearity of the robot dynamics is explicitly contained in the derivation, in contrast to RMI. This simplifies not only the derivation, but also the resulting equations.

In order to limit the history to two time steps as indicated by the AR analysis, we assume that the error of the last iteration was correctly within the bounds (ie: not a failure). We then compute the new extreme errors based on the measured error, $e_r = q_d - q$, of the last iteration and the current state information (in Y). The equation therefore finally becomes:

$$\ddot{e} = -K_d\dot{e}_r - K_p e_r + M^{-1}\,[Y]\,\tilde{p}. \qquad (13)$$

For the pendulum example, the error equation is

$$\begin{aligned} \ddot{e} &= -K_d(\dot{\theta}_d - \dot{\theta}) - K_p(\theta_d - \theta) \\ &\quad + \overbrace{\begin{bmatrix} \ddot{\theta} & \cos(\theta) \end{bmatrix}}^{Y} \overbrace{\begin{bmatrix} (\frac{\tilde{m}\tilde{l}^2}{ml^2} - 1) \\ (\frac{\tilde{m}\tilde{l}}{ml^2} - \frac{1}{l})g \end{bmatrix}}^{M^{-1}\tilde{p}}. \end{aligned} \qquad (14)$$

We have found that the acceleration that appears in Y can be the desired acceleration or a measured acceleration, if it is available, with little change in the results. Because equation (13) is based on the real (measured) error, however, we expect the thresholds to become invalid after the failure has occurred. This is verified in the results below but is not a major concern as the fault detection and tolerance algorithms of [9] will catch the failure, begin ignoring the faulty data, and find valid data for the controller so the robot will continue on its desired path.

5 Results

This section discusses some of the preliminary results for the new algorithm and compares ThMB to RMI. The nominal parameter estimates and bounds used in the examples below are the same as in the RMI examples of Section 2. We show the results when the parameters in the simulated robot are $\hat{m} = 10.7$kg and $\hat{l} = 1.009$m. Figure 3 displays the computed extremes of $\dot{e}$, the velocity tracking error. The values for the velocity and position errors are derived from the extreme values of $\ddot{e}$ in equation (14) by taking $\dot{e} = \dot{e}_r + \Delta t\ddot{e}$ and $e = e_r + \Delta t\dot{e}$. If the real parameters are within the assumed bounds (that is, the error in the mass is

$< 10\%$ and the length error is $< 1\%$), the resulting error between the desired velocity and the measured robot velocity will be between the two extremes as shown in the figure.

Figure 4 shows the derived extremes for e, the position tracking error. These values are smaller than the errors expected for the tachometer implying that the tachometer is more easily affected by the modeling errors. This result was also apparent in the RMI examples of [10].

Another feature of the ThMB algorithm is that the thresholds are not centered around zero, the desired error (see Figures 3 and 4). Because the algorithm uses the measured error in its computations, the computed error is biased in the direction of the real error. In these examples, the real robot is heavier than expected so its position initially lags behind the desired value because not enough torque is applied to counteract gravity. When this positive error (remember e=desired - measured) is used in the computation, it increases the expected maximum positive extreme and decreases the maximum negative extreme thus shifting the errors up from zero as in Figure 4. If the real robot were lighter than expected (causing overshoot), the extreme errors would be shifted down. The closer the real robot parameters are to the estimates of the parameters in the controller (and ThMB algorithm), the closer the thresholds will be to being centered around the desired zero error.

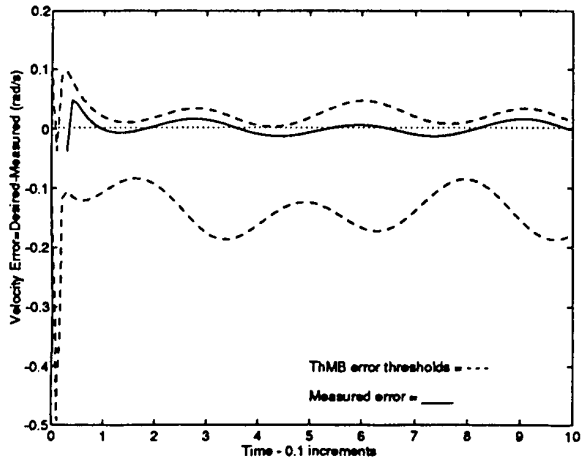

Figure 3: Thresholds for Tachometer with robot parameters: $\hat{m} = 10.7kg, \hat{l} = 1.009m$.

Figure 5 displays the expected velocity errors in a different format by plotting $\dot{q}_d - \dot{e}$ in order to show, as in RMI (see Figure 1), the extreme output velocities achievable with the given parameter variations.

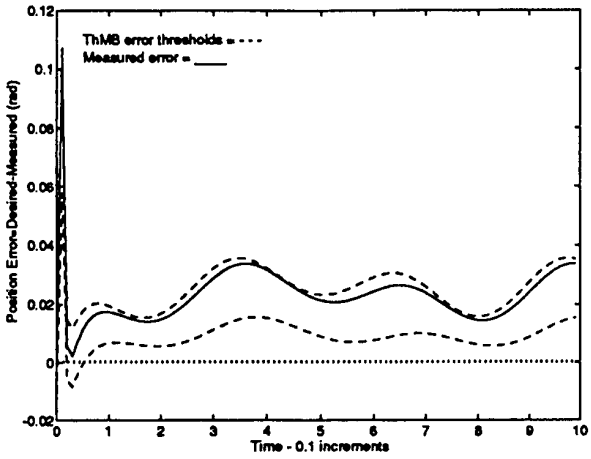

Figure 4: Thresholds for Encoder with robot parameters: $\hat{m} = 10.7kg, \hat{l} = 1.009m$.

Again, we note that the bounds are shifted (in this format, they shift up for the velocity) instead of being centered around the desired values. This reflects the fact that the real measured errors are fed back into the ThMB equations as mentioned earlier.

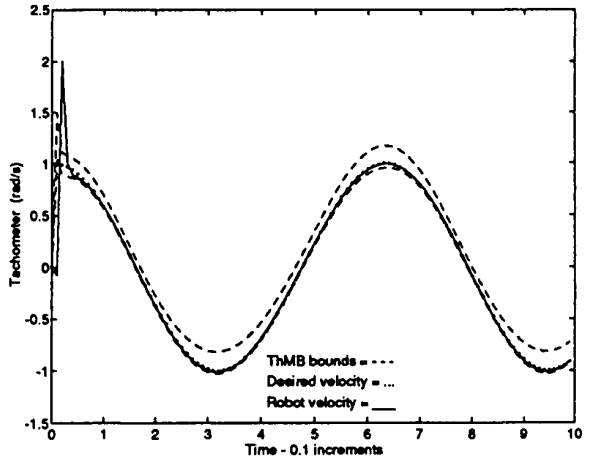

Figure 5: Tachometer ThMB bounds and robot with parameters: $\hat{m} = 10.7kg, \hat{l} = 1.009m$.

In comparing Figure 5 to the RMI results (Figure 1) for the same robot situation, we see that the ThMB bounds are smaller. The new method is less conservative than RMI (and may thus incur more false alarms) because it does not take into account the global history of the parameter error effects which RMI computes every cycle using the history window discussed earlier. Essentially, ThMB is a local optimization as compared

to the global optimizations of the RMI method. It is possible that a smaller error at the current iteration will ultimately result in a larger error later in the trajectory than the error currently used in the calculations. Despite this problem, the use of only local history allows ThMB to perform much fewer calculations at each iteration and it is thus able to produce a threshold much faster than RMI. This fast response is essential for robotic fault detection. ThMB can easily be made less prone to false alarms by assuming slightly larger bounds for the parameter errors than necessary.

The final figures (6 and 7) show the response of the new threshold generation algorithm to a stuck encoder failure at time 3.3. The ThMB bounds are plotted after the failure to demonstrate that, unlike RMI, the ThMB thresholds are drastically affected by the failure because the corrupted measured errors (from the faulty sensor) are used in the threshold calculations.

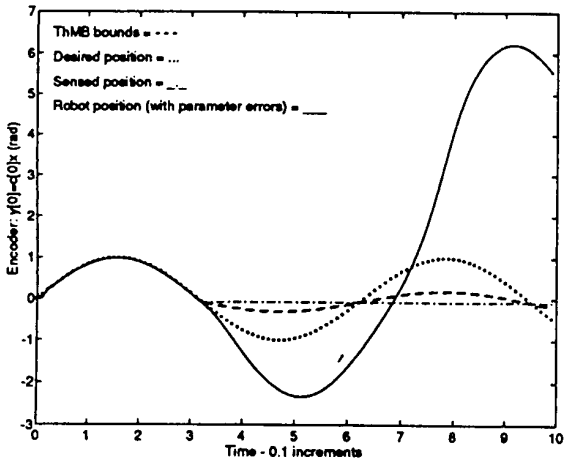

Figure 6: ThMB Encoder bounds when encoder fails at Time 3.3.

With the fault detection algorithms in place, however, the failure would be detected and appropriately tolerated immediately upon exceeding the bounds (which occurs in the next iteration after the failure). To tolerate the failure, the invalid data is ignored and replaced with valid data from surviving sensors [9] so that the erroneous bounds that would occur after the failure time are avoided. The bounds in Figure 6 would actually no longer be computed after the failure as the erroneous data from the failed sensor is ignored. The bounds in Figure 7 would simply return to their normal (fault-free) values as in Figure 5.

We are currently applying the ThMB method to a general robot and investigating the sensitivity of the thresholds to joint coupling. More detailed comparisons of the time complexity of both RMI and ThMB are also underway. These results will be presented in [11].

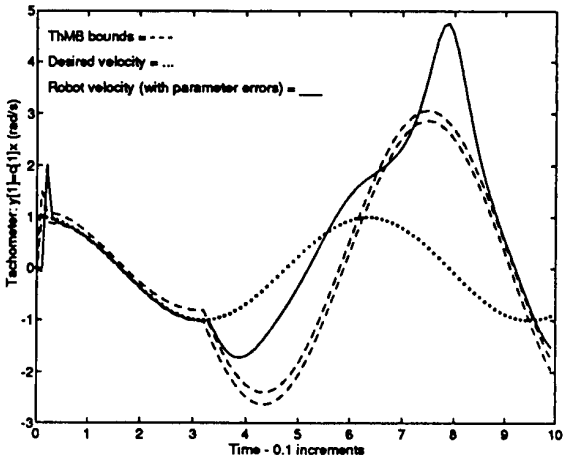

Figure 7: ThMB Tachometer bounds when encoder fails at Time 3.3.

6 Conclusions/Future Work

In this work, we focused on improving fault detection by generating more dynamic thresholds than the typical empirically determined and trajectory specific bounds. We briefly discussed the RMI method and the problems associated with implementing RMI for robotics. While the basic concept behind the RMI algorithm (finding the maximum possible variation produced by bounded parameter errors) is sound, RMI is not conducive to real-time operation for robotic applications due to the nonlinearity and complexity of the robot dynamic equations and the quantity of correlated parameter matrices necessary for the computations. The new algorithm proposed in this paper for generating dynamic, model-based thresholds (ThMB) proved capable of dynamically responding to the changes in the robot system so as to maintain a threshold that differentiates between real failures in the system versus false alarms due to modeling errors.

The advantages of using ThMB over RMI in robotic applications are that ThMB is real-time, less complex, and produces adequate, if not completely optimal, thresholds. ThMB equations are less involved than those in RMI and are readily derivable from manipulator dynamics. Combined with the fact that ThMB focuses only on the local history, the decrease in complexity enables us to generate a faster algorithm for

producing fault detection thresholds. As mentioned earlier, speed is important because the robot will react to a failure quickly.

Because the ThMB algorithm is derived straight from the dynamic equations fundamental for robotic control, ThMB will be better able to perform in the real-time environment of the robot controller. ThMB still has multiple comparisons to make in determining the extremes of equation (13), but the comparisons are faster because of the simpler equations and also are only done once (instead of one hundred times) in determining each threshold. ThMB produces slightly smaller thresholds than RMI but the increase in possible false alarms can be avoided by being more conservative in estimating the parameter errors. ThMB is, therefore, a viable alternative for generating detection thresholds in robotic applications.

7 Acknowledgements

This work was supported in part by the National Science Foundation under grants MSS-9024391 and DDM-9202639, and by DOE Sandia National Laboratory Contract #18-4379A. The research was further supported by a Mitre Corporation Graduate Fellowship and NSF Graduate Fellowship RCD-9154692.

References

[1] X. Ding and P. M. Frank. Frequency Domain Approach and Threshold Selector for Robust Model-Based Fault Detection and Isolation. In *Proceedings IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 271–276, Baden-Baden, Germany, 1991.

[2] A. Emami-Naeini, M. M. Akhter, and S. M. Rock. Effect of Model Uncertainty on Failure Detection: the Threshold Selector. *IEEE Transactions on Automatic Control*, 33(12):1106–1115, December 1988.

[3] D. T. Horak. Failure Detection in Dynamic Systems with Modeling Errors. *AIAA Journal of Guidance, Control, and Dynamics*, 11(6):508–516, November-December 1988.

[4] D. T. Horak. Experimental Estimation of Modeling Errors in Dynamic Systems. *AIAA Journal of Guidance, Control, and Dynamics*, 12(5):653–658, September-October 1989.

[5] D. T. Horak and B. H. Allison. Experimental Implementation and Evaluation of the RMI Failure Detection Algorithm. In *Proceedings of the 1987 American Control Conference*, pages 1803–1810, Green Valley, AZ, June 1987.

[6] R. Isermann, W. Appel, B. Freyermuth, A. Fuchs, W. Janik, D. Neumann, Th. Reiss, and P. Wanke. Model Based Fault Diagnosis and Supervision of Machines and Drives. In *Proceedings IFAC 11th Triennial World Congress*, pages 1–12, Tallinn, Estonia, USSR, August 1990.

[7] R. Rückwald. A Model-Based Fault Detection Concept for Mechanical Systems. In *Proceedings IFAC Fault Detection, Supervision and Safety for Technical Processes*, pages 205–210, Baden-Baden, Germany, 1991.

[8] M. W. Spong and M. Vidyasagar. *Robot Dynamics and Control*. John Wiley & Sons, Inc., New York, NY, 1989.

[9] M. L. Visinsky. Fault Detection and Fault Tolerance Methods for Robotics. Master's thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, December 1991.

[10] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Dynamic Sensor-Based Fault Detection for Robots. In *1993 SPIE Conference on Telemanipulator Technology and Space Robotics*, volume 2057, pages 385–396, Boston, MA, September 1993.

[11] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Adaptive Fault Detection and Tolerance for Robots. In *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications. Proceedings of the Fifth International Symposium on Robotics and Manufacturing*, Maui, HI, August 1994. To appear.

[12] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Expert System Framework of Fault Detection and Fault Tolerance in Robotics. *International Journal on Computers and Electrical Engineering*, 1994. To appear.

[13] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Chapter 3, Robotic Fault Tolerance: Algorithms and Architectures. In *Robotics and Remote Systems for Hazardous Environments, Volume 1*, pages 53–73. Prentice Hall Publishing Co., Englewood Cliffs, NJ, 1993.

[14] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Layered Dynamic Fault Detection and Tolerance for Robots. In *1993 IEEE International Conference on Robotics and Automation*, pages 180–187, Atlanta, GA, May 1993.

[15] J. Wünnenberg and P. M. Frank. Dynamic Model Based Incipient Fault Detection Concept for Robots. In *Proceedings IFAC 11th Triennial World Congress*, pages 61–66, Tallinn, Estonia, USSR, August 1990.