PROCEEDINGS
SPIE—The International Society for Optical Engineering

Telemanipulator Technology and Space Telerobotics

Won S. Kim
Chair/Editor

7–9 September 1993
Boston, Massachusetts

Sponsored and Published by
SPIE—The International Society for Optical Engineering

In cooperation with
Automated Imaging Association
Center for Excellence in Optical Data Processing/Carnegie Mellon University
The Advanced Technology Center for Precision Manufacturing/University of Connecticut
ISPRS—International Society for Photogrammetry and Remote Sensing

Volume 2057

Dynamic sensor-based fault detection for robots

M. L. Visinsky and J. R. Cavallaro and I. D. Walker

Dept. of Electrical and Computer Engineering
Rice University, Houston, TX 77251

ABSTRACT

Fault detection and fault tolerance are increasingly important for robots in space or hazardous environments due to the dangerous and often inaccessible nature of these environs. We have previously developed algorithms to enable robots to autonomously cope with failures of critical sensors and motors. Typically, the detection thresholds used in such algorithms to mask out model and sensor errors are empirically determined and are based on a specific robot trajectory. We have noted, however, that the effect of model and sensor inaccuracy fluctuates dynamically as the robot moves and as failures occur. The thresholds, therefore, need to be more dynamic and respond to the changes in the robot system so as to maintain an optimal bound for sensing real failures in the system versus misalignment due to modeling errors. In this paper, we analyze the Reachable Measurement Intervals method of computing dynamic thresholds and explore its applicability to robotic fault detection.

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 and cope with failures internal to the robot systems. The physical dangers of these environments also make it difficult for humans to freely or frequently repair failed components within the robot. 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 the corresponding expected output derived from a model of the system behavior. In previous work,[11–15] we have developed comparison tests for a variety of robots using analytical redundancy which reveals the existing functional redundancy available in a given system model and allows us to perform maximal fault detection without requiring modifications or expansions of the current system hardware. In reality, the ideal tests derived from analytical redundancy cause, in their pure form, an unacceptable number of false alarms due to sensor error and inexact system models being used in the controller comparisons. The model errors arise from linearization of the robot equations as well as from inaccuracies in model parameters such as joint length or mass. Thresholds must be used to mask out these modeling errors but must not be too large so as not to hide any real failures. In addition, the effect of the model errors and sensor noise fluctuates dynamically as the motion of the robot changes and as failures occur. The thresholds need to be dynamic in order to cope with the ever changing robot status.

There are a variety of algorithms which focus on computing thresholds for fault detection in uncertain dynamic systems.[1,2,8,10,16] This paper explores the development of new methods specifically for robots for generating dynamic thresholds which respond to the changing trajectory and failure situations of the robot based on expected error bounds of the system parameters. We focus on the Reachable Measurement Intervals (RMI) algorithm developed and analyzed for aircraft systems by Horak, et al, in various publications[3–7] and explore its applicability to robot systems. Section 2 summarizes the key points of the algorithm and the changes necessary to make RMI suitable for robotics. Section 3 discusses the use of RMI in simple robot examples and compares the results. A discussion of the extensibility of RMI to more general robotic problems is also presented. Section 4 discusses some alternatives to the RMI method for threshold generation. Our conclusions and future work are presented in Section 5.

0-8194-1322-4/93/$6.00

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:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + W(t) \\ y(t) &= Cx(t) + V(t), \end{aligned} \tag{1}$$

where x is the state vector for the system, A is the system dynamics matrix, B is the input distribution matrix, u is the known input (or control) signal, y is the output measurement signal, C is the measurement matrix, and W and V are input and output noise vectors.

The matrix A is defined as $A = A_n + A_v$, the sum of a nominal matrix A_n and a variation A_v. The matrix B is similarly defined as $B = B_n + B_v$. A_n and B_n are constant and represent the chosen model for the system. A_v and B_v are the possible errors in the model and are assumed to be within defined limits. The values of the noise vectors, W and V, are also bounded. For robot systems, W might be a combination of the effects of gravity and the non-linear portions of the robot dynamics. V would be the noise or inaccuracy associated with the sensors. The matrix C is initially assumed exact, but may also vary between bounds as noted near the end of this discussion. The system is thus rewritten as:

$$\begin{aligned} \dot{x}(t) &= A_n x(t) + B_n u(t) + A_v x(t) + B_v u(t) + W(t) \\ y(t) &= Cx(t) + V(t), \end{aligned} \tag{2}$$

The RMI algorithm provides conditions for finding the specific modeling errors which will maximize or minimize the system outputs given bounded parameter variations in A, B, W, and V. RMI thus finds the extremes of each output, y_j, described by the performance index

$$PI_j - y_j - c_j x(t_f), \tag{3}$$

where c_j is the jth row of the output matrix C and $x(t_f)$ is the state at the present time (which is the end of the procedure). The foundation of the RMI algorithm is Pontryagin's Maximum Principle.[7,9] In order to use the Maximum Principle in this manner, the system eigenvalues must have negative real parts so that any transients in the system are damped out in a finite amount of time, t_s, and the outputs of the system at time t_f are assumed to depend only on the inputs which occurred in the last t_s time steps.

Horak, et al, assume that the initial conditions on x ($x(t_f - t_s)$) can be set to zero because the effects of the state before time $t_f - t_s$ are assumed negligible. With robotic systems, we modify the initial condition appropriately for a dynamic trajectory. The initial condition at the start of each window is set to the nominal state x_n which is the expected state of the system given that there are no modeling errors or sensor noise. This implies that the robot controller was able to successfully damp out the transient errors (from model and sensor inaccuracies) within t_s time steps so that the state of the system at time $t_f - t_s$ can be thought of as the expected nominal result.

The focus of RMI is the Hamiltonian:

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

where L is the vector of Lagrange multipliers described by the approximation[3,4,6,7]

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

with the terminal boundary condition

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

The RMI procedure solves equations 2 and 5 with the appropriate initial conditions on x and boundary condition 6, while selecting at each time values of A_v, B_v, and W which maximize (and then minimize) H in equation 4. 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. 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_v 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 2 to produce the next x_{max} and x_{min}. 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 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 over 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 (equation 3). The result of the RMI procedure is two numbers per output measurement which represent the extreme achievable values of that output at time t_f given bounded modeling errors. These numbers bracket the desired value for a particular state (position, velocity, etc.) and can be used as near optimal thresholds in fault detection 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 work, we assess the usefulness of the RMI algorithm when applied to robotics. We initially chose to focus on the algorithm applied to a single-joint, pendulum robot to demonstrate the procedure and subsequently explored RMI for a two-joint robot in order to assess the expandability of the algorithm as well as the effects of joint coupling on the size of the derived thresholds. The implementation of the algorithm uses a discretized version of the dynamics (equation 2). 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 joint). 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 are illustrated in Figure 1. 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. For this analysis, 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.

The dynamics equations are written in terms of the uncertain parameters $\hat{m}$ and $\hat{l}$:

$$\underline{x}(k+1) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} 0 \\ \frac{\Delta t}{\hat{m}\hat{l}^2} \end{bmatrix} u + \begin{bmatrix} 0 \\ -\frac{g\Delta t}{\hat{l}} \end{bmatrix} cos(x_1), \tag{7}$$

$$\underline{y}(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{x}(k). \tag{8}$$

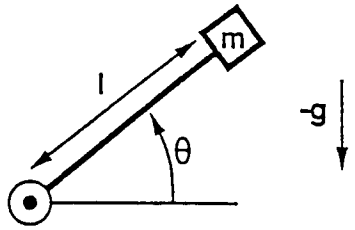

Figure 1: Pendulum Robot System.

The control, u, is derived so as to damp out transient errors in the system using the system gains and to account for the remaining non-linear 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:

$$\begin{aligned} u &= ml^2 u_d - ml^2(K_d\dot{\theta} + K_p\theta) + mglcos(\theta), \\ u_d &= \ddot{\theta}_d + K_d\dot{\theta}_d + K_p\theta_d \end{aligned} \tag{9}$$

Inserting the control into equation 7 and rearranging results in the following RMI state equation (see equation 2):

$$\underline{x}(k+1) = \overbrace{\begin{bmatrix} 1 & \Delta t \\ -K_p\Delta t & 1-K_d\Delta t \end{bmatrix}}^{A_n}\underline{x}(k) + \overbrace{\begin{bmatrix} 0 \\ \Delta t \end{bmatrix}}^{B_n} u_d + g_v, \tag{10}$$

$$g_v = \overbrace{\begin{bmatrix} 0 & 0 \\ (1-\frac{ml^2}{\hat{m}\hat{l}^2})K_p\Delta t & (1-\frac{ml^2}{\hat{m}\hat{l}^2})K_d\Delta t \end{bmatrix}}^{Av}\underline{x}(k) + \overbrace{\begin{bmatrix} 0 \\ (\frac{ml^2}{\hat{m}\hat{l}^2}-1)(\Delta t) \end{bmatrix}}^{Bv} u_d + \overbrace{\begin{bmatrix} 0 \\ (\frac{mlg}{\hat{m}\hat{l}^2}-\frac{g}{\hat{l}})(\Delta t) \end{bmatrix}}^{W} cos(\theta). \tag{11}$$

where Δt is the iteration interval. Note that if the parameters are exact, the variance g_v reduces to zero. At each iteration of the RMI process, four different values of g_v are computed based on the 2^2 combinations of the extremes for uncertain parameters $\hat{m}$ and $\hat{l}$. The maximum and minimum g_v values are then chosen to compute the next x_{max} and x_{min}. For this analysis the sensors are assumed exact so that the output equation is simply $\underline{y}(k) = \underline{x}(k)$. We assume a robot model with two sensors (encoder and tach) per joint whose measurements correspond to y_1 and y_2 respectively. The difference between the RMI results for the two outputs stems from the different Lagrange multipliers used in the maximization process.

3.2. Multi-joint Robots

As the robot size increases, the RMI equations become more complex. Each joint is affected by the rest of the robot through coupling. To keep the RMI structure the same, each joint is handled separately with the effects from other joints treated as a dynamic disturbance in W. While the derivation of the equations becomes more difficult for larger robots, it need only be performed once in the RMI design phase.

Unfortunately, the run-time computational complexity also increases for robots as the number of uncertain parameters, and thus the number of comparisons necessary at each RMI iteration, increases. RMI can be transformed into a dynamically recursive program[3,7] to help alleviate some of the computational burden. However, for full six or eight joint robots, the resulting 2^6 or 2^8 comparisons might still keep the algorithm from being successful in real-time. Note that while the various sets of coefficient matrices can be precomputed as discussed in Section 2 and pulled from a table, g_v must be re-computed each iteration with the changing state, x, and control,u_d.

Our two-joint robot example is displayed in Figure 2. RMI is performed for each joint separately and is easily made more efficient through parallelization. The parameters for joint 1 are chosen as m_1=12.0 kg with $\hat{m}_1$ varying from 11.0 kg to 13.0 kg and l_1=1.5 m with $\hat{l}_1$ ranging between 1.49 m and 1.51 m. We chose the parameters for joint 2 to be the same as those in the pendulum example to enable more useful comparisons of the results. Thus, m_2=10.0 kg, 9.0 kg$\leq \hat{m}_2 \leq$11.0 kg, l_2=1.0 m, and 0.99 m$\leq \hat{l}_2 \leq$1.01 m.

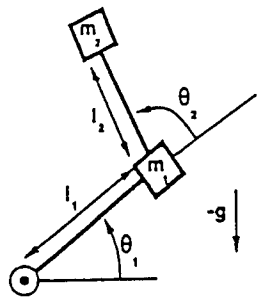

Figure 2: Two-joint Robot System.

One helpful aspect discovered in deriving the RMI equations for this robot is that the coefficient matrices A_v, B_v, and the various W matrices do not all depend on all of the parameters. In joint 2, for example, the parameter m_1 is not present at all. This fact cuts down the number of extreme parameter combinations and thus the comparisons necessary in finding x_{max} and x_{min}. Parameter independence also creates a shortcut in precomputing the matrices since a matrix which does not depend on certain parameters will not change as those parameters change (for constant combinations of the parameters on which the matrix does depend).

Another point of interest is that for both joints in this example, the A_n and B_n matrices are the same as in the pendulum example. The nominal dynamics (without the variance g_v) thus remains the same (between robots varying in the number or size of their joints). We therefore only display the changing g_v equations for the two-joint example. Further, the W matrix has been split into various parts in order to separate the parameter-dependent coefficient sections from the dynamic, state-dependent variables. Letting

$$\begin{aligned} d1 &= \hat{m}_1\tilde{l}_1^2 + \hat{m}_2\tilde{l}_1^2 + \hat{m}_2\tilde{l}_2^2 \\ \text{and } n1 &= \hat{m}_1\hat{l}_1^2 + \hat{m}_2\hat{l}_1^2 + \hat{m}_2\tilde{l}_2^2, \end{aligned}$$

the equation for joint 1 then becomes:

$$\begin{aligned} g_{v1} = {} & \overbrace{\begin{bmatrix} 0 & 0 \\ (1-\frac{n1}{d1})K_p\Delta t & (1-\frac{n1}{d1})K_d\Delta t \end{bmatrix}}^{A_{v1}} \underline{x}(k) + \overbrace{\begin{bmatrix} 0 \\ (\frac{n1}{d1}-1)(\Delta t) \end{bmatrix}}^{B_{v1}} u_d + \\ & \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_1 l_2 - \hat{m}_2 \hat{l}_1 \tilde{l}_2}{d1})\Delta t \end{bmatrix}}^{W_{1a}} \{cos(\theta_2)(2\ddot{\theta}_1+\ddot{\theta}_2) + sin(\theta_2)\dot{\theta}_2(2\dot{\theta}_1+\dot{\theta}_2)\} + \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2\tilde{l}_2^2 - \hat{m}_2\tilde{l}_2^2}{d1})\Delta t \end{bmatrix}}^{W_{1b}} \ddot{\theta}_2 + \\ & \overbrace{\begin{bmatrix} 0 \\ (\frac{(m_1+m_2)l_1 - (\hat{m}_1+\hat{m}_2)\tilde{l}_1}{d1})g\Delta t \end{bmatrix}}^{W_{1c}} cos(\theta_1) . \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_2 - \hat{m}_2 \tilde{l}_2}{d1})g\Delta t \end{bmatrix}}^{W_{1d}} cos(\theta_1+\theta_2). \end{aligned} \tag{12}$$

The equation for the g_v of joint 2 looks very similar to that in the pendulum example. The only difference is in the W matrices where the effects of the coupling to joint 1 come into play.

$$\begin{aligned} g_{v2} = {} & \overbrace{\begin{bmatrix} 0 & 0 \\ (1-\frac{m_2 l_1^2}{\hat{m}_2 \tilde{l}_2^2})K_p\Delta t & (1-\frac{m_2 l_2^2}{\hat{m}_2 \tilde{l}_2^2})K_d\Delta t \end{bmatrix}}^{A_{v2}} \underline{x}(k) + \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_1^2}{\hat{m}_2 \tilde{l}_2^2}-1)(\Delta t) \end{bmatrix}}^{B_{v2}} u_d + \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_2^2}{\hat{m}_2 \tilde{l}_2^2}-1)(\Delta t) \end{bmatrix}}^{W_{2a}} \ddot{\theta}_1 + \\ & \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_1 l_2}{\hat{m}_2 \tilde{l}_2^2} - \frac{\hat{l}_1}{\tilde{l}_2})(\Delta t) \end{bmatrix}}^{W_{2b}} \{cos(\theta_2)\ddot{\theta}_1 + sin(\theta_2)\dot{\theta}_1^2\} + \overbrace{\begin{bmatrix} 0 \\ (\frac{m_2 l_2 g}{\hat{m}_2 \tilde{l}_2^2} - \frac{g}{\tilde{l}_2})(\Delta t) \end{bmatrix}}^{W_{2c}} cos(\theta_1+\theta_2). \end{aligned} \tag{13}$$

3.3. RMI Results

The RMI algorithm produces upper and lower bounds for each measurement output based on the expected extremes in the system parameter errors. To analyze the usefulness of the algorithm for robotics, we ran RMI with several different trajectories and compared the resulting bounds with thresholds derived through experimentation.[11] The RMI bounds are time varying and dynamically change with the trajectory, as expected. RMI allows for tighter thresholds through the course of the robot tasks as RMI includes the current robot state in its computations. Empirically determined thresholds are generally chosen by monitoring several fault-free runs and selecting a threshold larger than the noted maximum deviation from the desired value. The following discussion details some of the more interesting results and observations discovered during our analysis.

In order to better compare the results of the one and two joint robots, we initially opted to apply the same trajectories used in the pendulum analysis to the second joint of the larger robot while keeping the first joint stationary. As noted in Section 3.2, the link parameters and variances for the second joint are the same as those for the pendulum to further the comparison. Keeping the first joint stationary also allows us to separately analyze the coupling effects caused by the motion of the rest of the robot.

The first trajectory studied was a sine wave in which θ (joint position) $= sin(t)$. This motion moved the robot up and down in the vertical plane between ± 1 rads (approximately $\pm 57°$). The following graphs show the responses of the RMI algorithm for the pendulum (Figures 3 and 5) and the second joint of the two-joint robot (Figures 4 and 6).

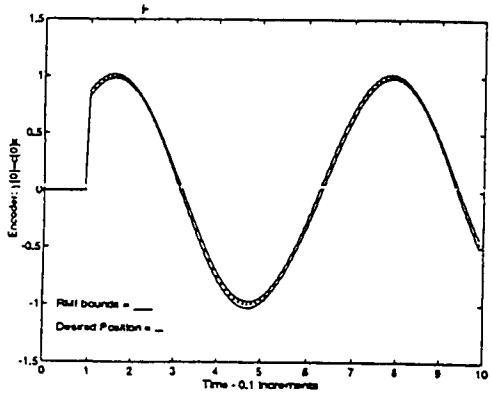

Figure 3: Pendulum Encoder RMI - Sine Wave.

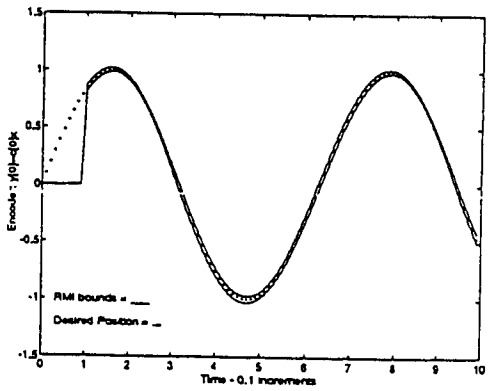

Figure 4: Two-Joint Encoder RMI - Sine Wave for Joint 2

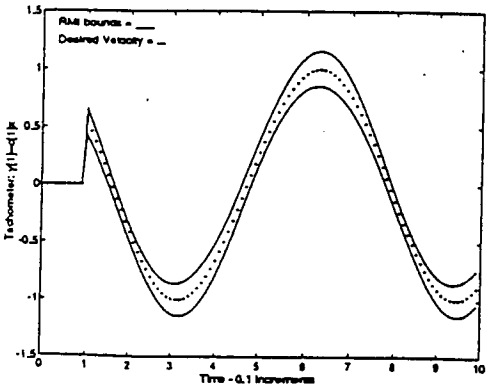

Figure 5: Pendulum Tachometer RMI - Cosine Wave.

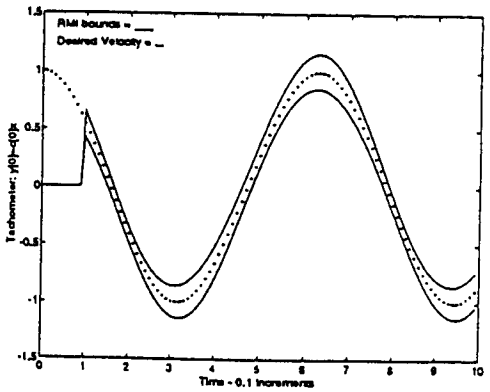

Figure 6: Two-Joint Tachometer RMI - Cosine Wave for Joint 2

Note that the width between the RMI bounds increases near the peaks (especially obvious in the tachometer results of Figures 5 and 6). To explain this, let us first assume that the robot is actually heavier than the controller expects (and the link length is perfect). The controller is sending commands that are two weak to push the robot to the

top of the trajectory and which are not adequate to counteract the increased affects of gravity on the way back down. The robot therefore lags on the way up and leads on the way down. When the trajectory reaches a peak, the controller slows the robot down in order to make the change in direction. If the robot is near the top of the trajectory, the heavier weight aids the slowing down process and causes more of an undershoot of the goal position. If the robot is on the down swing, the weight of the robot causes it to overcome the inadequate control and overshoot the goal slightly. The opposite effects hold true for a robot which is lighter than expected as the control would be more heavy-handed than necessary. Combined, these effects increase the width between the two RMI bounds when the trajectory changes direction.

The width between the RMI bounds is generally larger for the two-joint robot than for the one joint robot due to the addition of coupling effects into the error calculations. The motion of the second joint when combined with the normal modeling errors of the first joint will affect the first joint more strongly than just the modeling errors alone. Although the first joint is not asked to move in most of these trajectories, the expected modeling errors cause some variation in the motion (because the controller is not perfect). The first joint will thus affect the motion of the second joint and vice versa. The width is not always larger, however, as the coupling sometimes helps the robot to get back on course despite the modeling errors (not obvious in the scale of these graphs).

The next trajectory we tested was a simple ramp in which position θ=t. One interesting aspect of this trajectory is that it more clearly shows the affect of gravity on the bounds as the robot spins in a circle in the vertical plane. Figures 7 and 8 show the results for the pendulum robot. (The two-joint robot responds similarly with larger bounds due to coupling.) The oscillation due to gravity is again more obvious in the tachometer RMI (Figure 8). At $\theta = \frac{\pi}{2}$ (the top of the circular motion) and $\frac{3\pi}{2}$ (the bottom), gravity has the least disturbance on the robot while it has the most affect when the robot is extended perpendicular to the direction of gravity (ie: at 0 and π in the rotation).

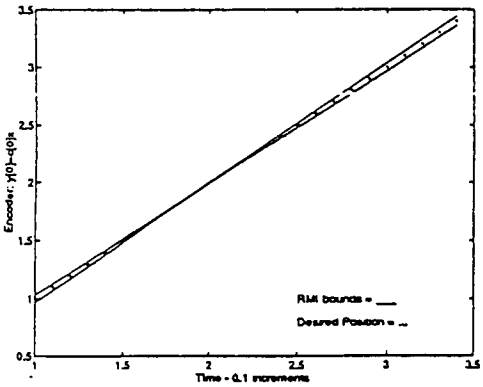

Figure 7: Pendulum Encoder RMI - Ramp Trajectory.

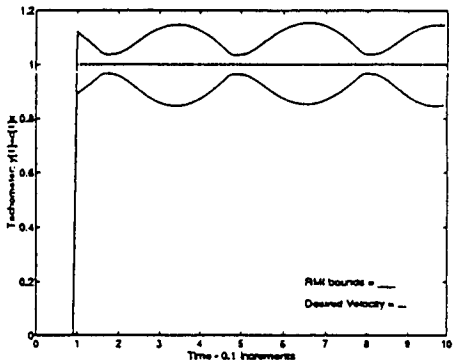

Figure 8: Pendulum Tachometer RMI - Constant Trajectory.

The next test used a composite trajectory with $\theta = sin(t) + cos(0.5 * t) + 0.1 * (15 - t)$. The interesting result of this trajectory is actually in the bounds produced for joint 1 in the two-joint robot (see Figures 9 and 10). While it is not desired that the joint move, modeling errors make the control slightly incorrect and allow the motion of the second joint as well as gravity to take affect. The maximum deviation due to these factors is shown by the RMI bounds in the two figures. Note that the scale of these figures is quite small. Since the downward force induced by gravity will be essentially constant because the joint remains generally perpendicular to the direction of gravity, the figures give a good indication of the coupling effects for this trajectory. It is also interesting to note at this stage that the tachometer (velocity) is generally more affected by modeling errors and coupling (thus creating wider bounds) than the encoder (position).

The final demonstrated trajectory was one specifically for the two-joint robot in which both joints are in motion. The first link is commanded to move along a sine wave with $\theta = sin(0.2t)$ (Figures 11 and 13). The second joint (Figures 12 and 14) moves along the composite wave mentioned above. From the analysis of the sine/cosine wave trajectory for both robots (see Figures 3 - 6), we know that there is generally little oscillation in width between the

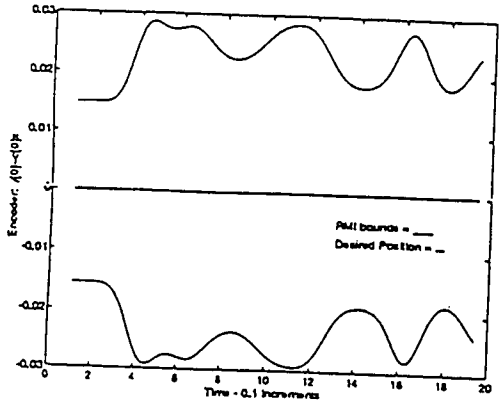

Figure 9: Two-Joint Encoder RMI - Composite Trajectory for Joint 1.

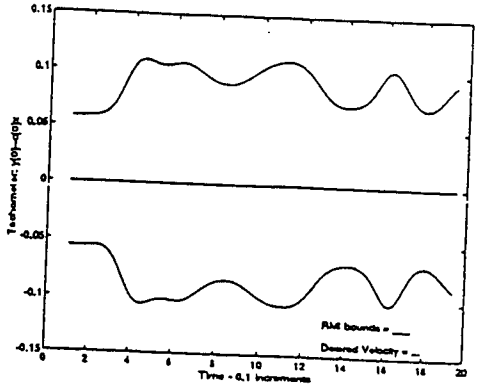

Figure 10: Two-Joint Tachometer RMI - Composite Trajectory for Joint 1.

bounds except for near the peaks. In Figure 11, however, there are more variations apparent (around Time=5, 12, etc.) which are directly due to the pull of the other joint as it moves. The encoder RMI for joint 2 has also been affected although less obviously. The tachometer RMIs (Figures 13 and 14) are more dramatically affected by the coupling between the joints when both joints are in motion.

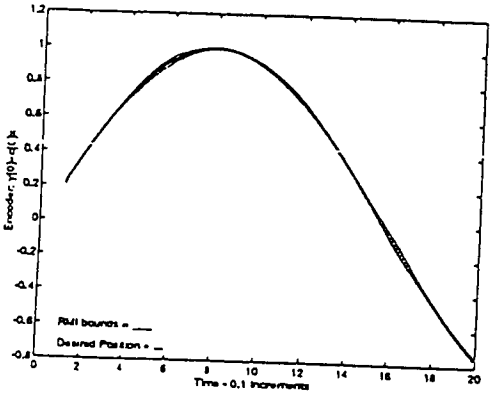

Figure 11: Two-Joint Encoder RMI - Sine Wave for Joint 1.

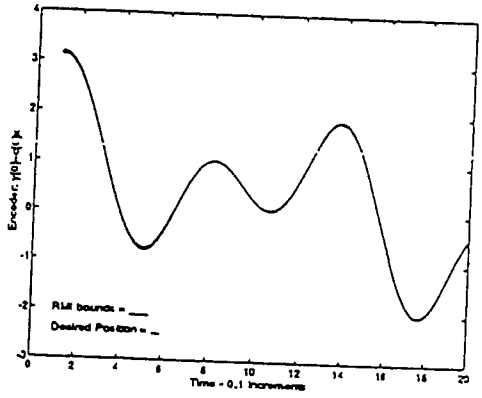

Figure 12: Two-Joint Encoder RMI - Composite Wave for Joint 2.

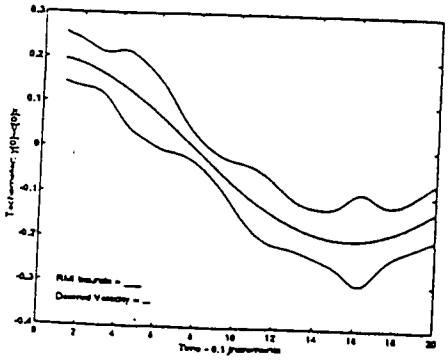

Figure 13: Two-Joint Tachometer RMI - Cosine Wave for Joint 1.

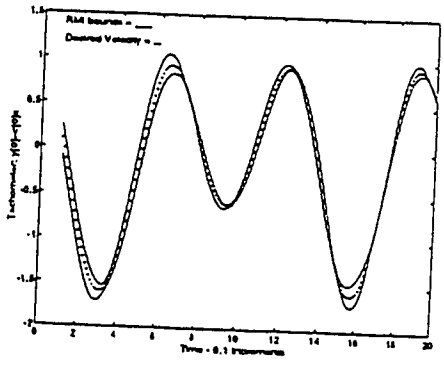

Figure 14: Two-Joint Tachometer RMI - Composite Wave for Joint 2.

3.4. RMI in Fault Detection

In general, the thresholds found using RMI are comparable to or smaller than the empirically determined constant thresholds. At times, however, RMI produces larger bounds. This implies that the original experiments did not completely cover all the possible trajectory/modeling error situations. A robot might not generally need the larger thresholds, but at some time the modeling error and coupling conditions may be just right to break through a smaller bound and trigger a false alarm. The actions taken to deal with this non-existent failure would use up resources needed to tackle real failures and limit the robot's chances of surviving. Assuming that the feedback controller is capable of dealing with the smaller failures that might not trip the threshold boundary immediately (such as a slight drift in a sensor), the RMI method produces dynamic and reasonably optimal thresholds for robotic fault detection.

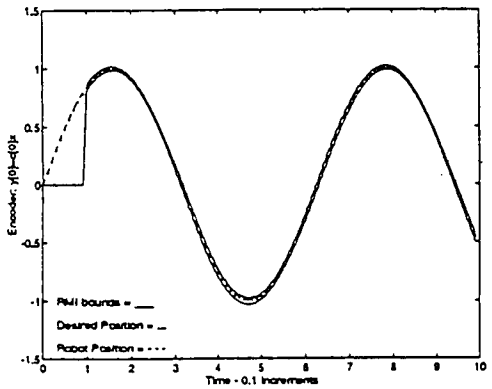

Figure 15: Two-joint Encoder RMI for Joint 2 and Robot Trajectory with $\hat{m}_{(1,2)} = 0.08$ and $\hat{l}_{(1,2)} = -0.005$.

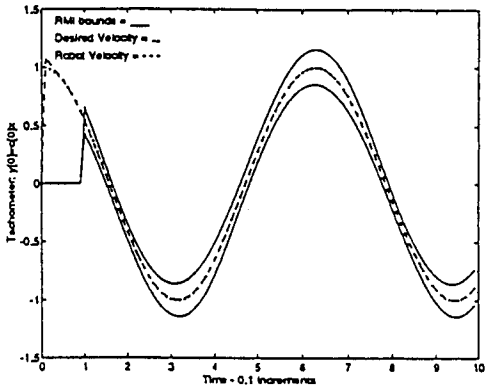

Figure 16: Two-joint Tachometer RMI for Joint 2 and Robot Trajectory with $\hat{m}_{(1,2)} = 0.08$ and $\hat{l}_{(1,2)} = -0.005$.

Dynamically changing thresholds allow the robot to detect most failures faster than the constant threshold found through observation. Figures 15 and 16 show the RMI bounds for the sin wave trajectory. The real path of the second joint deviates from the desired path slightly due to randomly chosen model errors (within the assumed bounds).

Figures 17 and 18 show a stuck sensor failure in Joint 1 at Time=3s. (The sensed value stops changing and the velocity and derived acceleration drop to zero.) Because the joint is not directed to move, it takes a while for the modeling error to accumulate enough power to break loose from the controller and exceed the RMI bounds. The effect on Joint 2 is negligible until the error in the first joint's position becomes much larger than the RMI bounds. Between Time 8.1 and 9.3, Joint 2 is pulled off course. If the detection routines where in place, they would detect the error immediately upon passing the RMI bound (Time=4.8s) and correct the error before it affected Joint 2.

Figures 19 and 20 show a similar sensor failure occurring in Joint 2. (Note that only part of the real robot paths are shown as they quickly escalate towards infinity.) The failure must be caught quickly because it drastically affects both joints very rapidly. The error in the robot path almost immediately passes through the bounds provided by RMI and would thus trigger the fault detection and tolerance routines in time to tolerate the failure and prevent any serious deviation from the desired path.

4. DYNAMIC THRESHOLD ALTERNATIVES

As noted earlier, RMI is 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 a full 2^n comparisons (n=number of parameters) per state variable per joint as mentioned in Section 3.2. 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[11] 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.

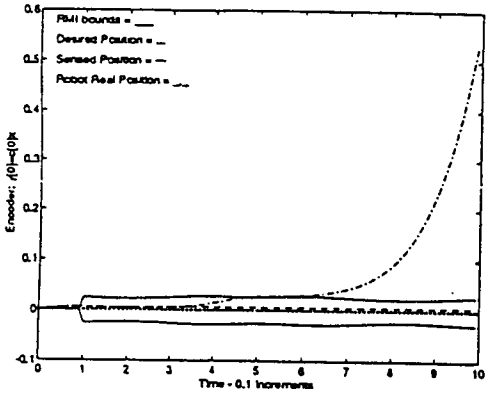

Figure 17: Two-joint Encoder RMI for Joint 1 and Robot Trajectory with Joint 1 sensor failure at Time=3.0s.

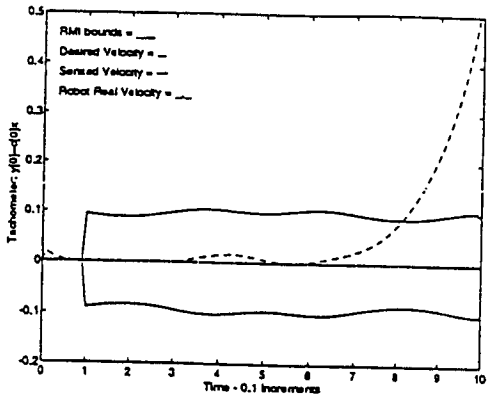

Figure 18: Two-joint Tachometer RMI for Joint 1 and Robot Trajectory with Joint 1 sensor failure at Time=3.0s.

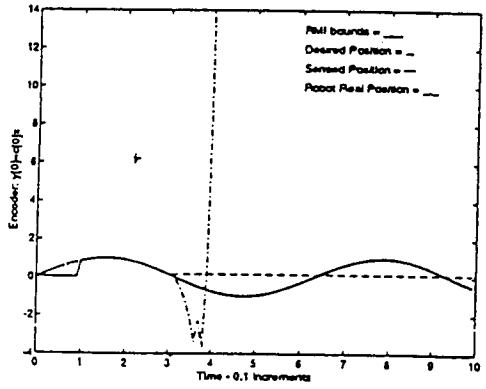

Figure 19: Two-joint Encoder RMI for Joint 2 and Robot Trajectory with Joint 2 sensor failure at Time=3.0s

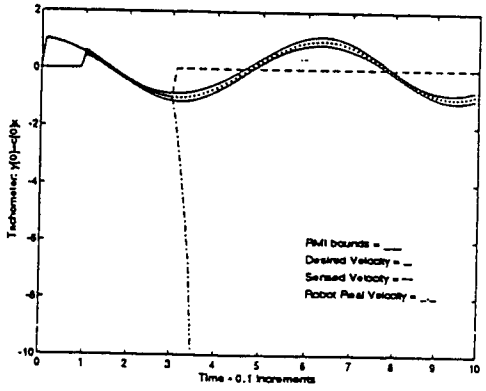

Figure 20: Two-joint Tachometer RMI for Joint 2 and Robot Trajectory with Joint 2 sensor failure at Time=3.0s

RMI also 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 beginning to look at possible alternatives to the RMI concept which still utilize the idea of finding the maximum variance possible from bounded parameter errors at each state iteration. Towards this end, we have noted the similarities between RMI and the analytical redundancy (AR) analysis we have performed in deriving our detection tests.[11,15] Several other threshold generation methods have a similar AR format.[10,16] With AR, we found that only two time steps of history were needed for adequate fault detection in our robots. Using only two steps to compute the thresholds would greatly reduce the computation time (the window used here is a hundred iterations). We have also noted that many control methods are parameter based and may be more efficient and conducive to threshold computations dealing with bounded parameter errors.[8]

5. CONCLUSIONS/FUTURE WORK

Our fault detection and fault tolerance algorithms developed in previous work enable robots to quickly and autonomously cope with failures of critical sensors and motors. In this work, we have analyzed an algorithm for producing more dynamic thresholds than the typical empirically determined and trajectory specific bounds. The thresholds produced by a robotic version of the RMI algorithm derived by Horak, et al, proved capable of dynami-

cally responding to the changes in the robot system so as to maintain an optimal bound for sensing real failures in the system versus false alarms due to modeling errors. While the basic concept behind the algorithm (finding the maximum variation produced by bounded parameter errors) is sound, RMI is not conducive to real-time operation for robotic applications due to the complexity of the robotic equations and the quantity of correlated parameter matrices necessary for the computations. Alternatives are proposed and will be explored in future work.

6. ACKNOWLEDGEMENTS

This work was supported in part by the National Science Foundation under grants MIP-8909498 and MSS-9024391, by 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.

7. 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 Dectection, 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. *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. *Journal of Guidance, Control, and Dynamics*, 12(5):653–658, September-October 1989.

[5] D. T. Horak. System Failure Isolation in Dynamic Systems. *Journal of Guidance, Control, and Dynamics*, 13(6):1075–1082, November-December 1990.

[6] 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.

[7] D. T. Horak and D. M. Goblirsch. Reachable Outputs in Systems with Bounded Parameter Uncertainties: Application to Failure Detection. In *Proceedings of the 1986 American Control Conference*, pages 301–308, Seattle, WA, June 1986.

[8] 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.

[9] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical theory of optimal processes*. Macmillan, New York, 1964. Translated by D. E. Brown.

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

[11] 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.

[12] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Fault Detection and Fault Tolerance in Robotics. In *Proceedings of NASA Space Operations, Applications, Research Symposium*, pages 262–271, Houston, TX, July 1991.

[13] M. L. Visinsky, I. D. Walker, and J. R. Cavallaro. Expert System Framework of Fault Detection and Fault Tolerance for Robots. In *Proceedings of the International Symposium on Robotics and Manufacturing*, pages 793–800, Santa Fe, NM, Nov 1992.

[14] 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., 1993. In Press.

[15] 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.

[16] 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.