RICE UNIVERSITY

Derivation and Application of Nonlinear Analytical Redundancy Techniques with Applications to Robotics

by

Martin L. Leuschen

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Approved, Thesis Committee:

Joseph R. Cavallaro, Chair
Associate Professor of Electrical and
Computer Engineering

Richard G. Baraniuk
Professor of Electrical and Computer
Engineering

Fathi Ghorbel
Associate Professor of Mechanical
Engineering and Materials Science

Ian D. Walker
Professor of Electrical and Computer
Engineering, Clemson University

Houston, Texas
October, 2001

ABSTRACT

Derivation and Application of Nonlinear Analytical Redundancy Techniques with Applications to Robotics

by

Martin L. Leuschen

Fault detection is important in many robotic applications. Failures of powerful robots, high velocity robots, or robots in hazardous environments are quite capable of causing significant and possibly irreparable havoc if they are not detected promptly and appropriate action taken. As robots are commonly used because power, speed, or resistances to environmental factors need to exceed human capabilities, fault detection is a common and serious concern in the robotics arena.

Analytical redundancy (AR) is a fault-detection method that allows us to explicitly derive the maximum possible number of linearly independent control model-based consistency tests for a system. Using a linear model of the system of interest, analytical redundancy exploits the null-space of the state space control observability matrix to allow the creation of a set of test residuals. These tests use sensor data histories and known control inputs to detect any deviation from the static or dynamic behaviors of the model in real time.

The standard analytical redundancy fault detection technique is limited mathematically to linear systems. Since analytical redundancy is a model-based technique, it is extremely sensitive to differences between the nominal model behavior and the

actual system behavior. A system model with strong nonlinear characteristics, such as a multi-joint robot manipulator, changes significantly in behavior when linearized. Often a linearized model is no longer an accurate description of the system behavior. This makes effective implementation of the analytical redundancy technique difficult, as modeling errors will generate significant false error signals when linear analytical redundancy is applied. To solve this problem we have used nonlinear control theory to extend the analytical redundancy principle into the nonlinear realm. Our nonlinear analytical redundancy (NLAR) technique is applicable to systems described by nonlinear ordinary differential equations and preserves the important formal guarantees of linear analytical redundancy. Nonlinear analytical redundancy generates considerable improvement in performance over linear analytical redundancy when performing fault detection on nonlinear systems, as it removes all of the extraneous residual signal generated by the modeling inaccuracies introduced by linearization, allowing for lower threshold.

Acknowledgments

A doctorate is a difficult task, and I could not have done it without lots of help. I want to thank my advisors Dr. Joseph Cavallaro and Dr. Ian Walker, who provided essential support and assistance throughout my graduate career. Dr. Fathi Ghorbel deserves thanks not only for being a committee member on both my thesis committees, but for providing helpful pointers and a short course that lead me to a critical component of my doctoral research. I also wish to thank my fourth committee member, Dr. Richard Baraniuk, for his help, his interest, and for being patient with me when I was trying to find a time that four busy professors all had free.

Thanks are also due to my family and friends who have encouraged, supported and inspired me, as well as teaching me many valuable lessons. Listing everything I owe them would be a thesis in its own right.

Deepest thanks and love to Sarah.

This work was supported in part by NASA grant NAG5-9785, NSF/EPSCoR grant EPS-9630167, and DOE contract DE-FG07-97ER 14830.

Contents

Illustrations

Tables

Chapter 1

Introduction

1.1 The Importance of Robotic Fault Detection

One of the most important and fastest growing areas in the robotics industry is robotic reliability [4, 8, 9, 10, 16, 23, 24, 27, 29, 30, 32, 43, 44, 45, 46, 47]. Robots are commonly used where a greater than human level of speed, environmental tolerance, or power is required. However, all of these characteristics make failures of the robot more dangerous. Powerful robots that fail can cause considerable damage to their environment. High velocity robots can react so quickly to faults that stopping them before they cause damage can be a difficult task. Robots in hazardous environments that fail can be almost impossible to remove or repair due to said environment, and in the worst case might even breach containment. A powerful high velocity robot that fails while in a hazardous environment can thus wreak considerable havoc [22, 23, 27, 45].

Dynamic fault detection systems that monitor the robot for faults in real time can contribute significantly to system reliability and safety by reducing these dangers [10, 19, 24, 30, 44, 45, 46]. Quick detection of faults can prevent the damage caused by out of control systems to both themselves and their environments [44, 45]. Fault detection can also improve maintenance and allow for graceful degradation of redundant systems. Thus good fault detection allows for the completion of previously impractical or impossible tasks.

1.2 Analytical Redundancy (AR)

The analytical redundancy (AR) method is an important technique currently used for fault detection [5, 45]. AR is an especially interesting and useful technique as it allows us to explicitly derive the maximum possible number of model-based linearly independent consistency tests for a system [5].

One major advantage of AR is that the technique automatically removes the model-following aspects of the sensor and control data, leaving only the deviations from the model characteristics inherent in the individual AR tests. In addition to allowing better model tuning, this reduces the amount of extraneous data the error detection software has to deal with, making isolating faults much easier. Although some faults can be visible in the raw sensor data streams, AR isolates faulty behavior from expected behavior, making large faults more clear, exposing less obvious faults, and increasing the speed of fault detection. AR tests can also facilitate categorization of faults through the association of particular residuals and signals with particular faults.

Perhaps the most important aspect of analytical redundancy is that it guarantees that the test residuals generated by the techniques will test the entire space of "observable" faults [5]. Using a linear model of the system of interest, AR exploits the left null-space of the control-theoretic observability matrix to allow the creation of a set of test equations. These tests use sensor data histories and known current and past control inputs to detect any observable deviation whatsoever from the static or dynamic behaviors predicted by the model. While there are some minor limitations inherent in the model-based nature of the technique, this is a very useful and broad guarantee. Simply put, AR guarantees that it *will* see any deviation from the model that *can* be seen.

Finally, analytical redundancy generates a minimal span of the space of faults. This means that not only can it make sure that every model deviation is observed, it can also do it with the minimum possible number of tests.

1.3 Nonlinear Issues

The standard AR fault detection technique [5] is effectively limited to linear systems. As AR is a model-based technique, it is extremely sensitive to differences between the nominal model behavior and the actual system behavior. A system model with strong nonlinear characteristics, such as a multi-joint robot manipulator, suffers considerably from linearization. Many of the most energetic dynamic behaviors of the system are nonlinear in nature. This makes effective implementation of the AR technique difficult, as modeling errors will generate significant false error signals when linear AR is applied.

To solve this problem we have used nonlinear control theory [20, 36] to extend the AR principle into the nonlinear realm. Although others have combined linear AR with nonlinear systems [41, 48, 50] none of this work uses nonlinear observability techniques. Our nonlinear analytical redundancy (NLAR) technique is generally applicable to systems described by nonlinear ordinary differential equations and preserves the desirable formal guarantees of AR. It also generates considerable improvement in performance over linear AR when performing fault detection on nonlinear systems.

1.4 Earlier Work

AR for robotic systems has been examined by several researchers in the past. The following references are particularly germane to the nonlinear analytical redundancy techniques discussed in this thesis dissertation.

Visinsky, Cavallaro, and Walker examined robotic AR as a part of a larger multi-layered fault detection system [44, 45]. In this work, standard linear AR was used in concert with other techniques in the creation of a multi-layer fault detection and fault tolerance framework. Problems resulting from the linear nature of AR were compensated for by other methods which recognized when the system was in a stressful or nonlinear situation. This approach coped well with the model accuracy problems of linear AR, but did not solve them, and thus provided an excellent starting point for the research in this thesis.

Nonlinear observers have been used in the same role as AR [4, 15] and it has been shown that AR and observer based methods are equivalent in the linear case [34]. However this only applies to linear systems, and the nonlinear observer based method [15] lacks the span guarantees of NLAR.

Wünnenberg and Frank have investigated methods for thresholding linear robotic AR test residuals to compensate for various modeling inaccuracies [11, 48]. Various techniques, including dynamic analysis and fuzzy logic, were used to predict when a system was in a portion of the dynamic workspace that was poorly modeled by the linear AR fault residuals. In these areas the techniques modified the thresholds which the analytical redundancy residuals had to surpass to trigger a fault condition in these areas.

Isidori and De Persis have derived a geometric residual generator [38] using nonlinear observability. This method is similar in concept to AR, but requires accurate fault modeling, and is thus not useful in the same contexts.

Starosweicki and Comtet-Varga have produced some interesting work describing rigorous nonlinear AR for limited classes of nonlinear systems [41, 42]. This work departs from the traditional observability roots of analytical redundancy and focuses on methods for expressing model-based consistency tests in terms of sensor and control histories. No work on general nonlinear AR firmly based on nonlinear control theory is available.

1.5 Contribution and Overview of Thesis

The main contribution of this thesis is the development of a rigorous method for deriving fully nonlinear analytical redundancy (NLAR) test residuals that can be applied to a wide range of nonlinear systems. The NLAR method presented is as theoretically rigorous about covering all observable faults in nonlinear systems as linear AR is with respect to linear systems. The method is fully applicable to any observable system where the nonlinearities can be mapped onto a smooth manifold and quite useful in situations where a smooth nonlinear manifold is a better approximation of the system than a simple linear system. It maintains linear AR's important theoretical guarantee of spanning the failure space with a minimal number of test residuals. This new NLAR method extends all of the good aspects of linear AR into a whole new area of nonlinear systems. In such systems, good, rigorous fault detection techniques like nonlinear analytical redundancy are especially needed.

Chapter 2 discusses analytical redundancy techniques that are based on linear principles. It reviews and re-derives the standard analytical redundancy technique as originally presented by Chow and Willsky [5]. It also presents a small example. Methods that emulate nonlinear analytical redundancy using linear techniques, developed in this thesis and by others, are also discussed. Chapter 3 presents the new nonlinear analytical redundancy technique. Nonlinear observability techniques are presented and used to derive nonlinear analytical redundancy. A simple example, parallel to the linear example of Chapter 2, is also provided. Chapter 4 presents the Rosie mobile worksystem and describes a project in which nonlinear analytical redundancy was applied to data from a physical testbed emulating one of the robot's hydraulic wheel actuators. Results from the project are given, and issues concerning model accuracy are discussed in the context of this system. Chapter 5 describes the application of nonlinear analytical redundancy to a simulated two joint planar robot arm. This system is well modeled but quite complex mathematically, and so it presents other interesting problems and issues. Chapter 6 offers a summary of this research, and offers probable directions for future work. The whole of this work provides conclusive evidence that the nonlinear analytical redundancy technique presented is a considerable improvement over the linear technique.

Chapter 2

Background and Previous Work

In this chapter we discuss classical linear analytical redundancy (AR) and discuss previous attempts to extend AR into the nonlinear domain.

2.1 Linear Analytical Redundancy

The core idea of analytical redundancy is both intuitive and elegant [5]. It is intuitive because it relies on the core concept of observability, namely that everything that can be learned about the model-based behavior of a system is contained in the observability matrix. It is elegant in that it processes that information in such a way as to generate a theoretically elegant series of residual tests: AR residuals are guaranteed both to be linearly independent and to test for all detectable deviations from the system model. Thus, every residual contains at least some information not contained in other residuals, and every single observable deviation from the system model is tested for by one of the AR residuals. The following is a detailed derivation of the AR method.

2.1.1 Control Theory Modeling Concerns

Consider the canonical state-space linear control model with n states, q control inputs, and m sensors:

$$\begin{aligned} \underline{\dot{x}}(t) &= A\underline{x}(t) + \textstyle\sum_{j=1}^{q} B_j u_j(t) + W \\ y_j(t) &= c_j\underline{x}(t) + v_j, \qquad j = 1, \cdots, m \end{aligned} \tag{2.1}$$

Or, more compactly:

$$\begin{aligned} \underline{\dot{x}}(t) &= A \cdot \underline{x}(t) + B \cdot \underline{u}(t) + W \\ \underline{y}(t) &= C \cdot \underline{x}(t) + V \end{aligned} \tag{2.2}$$

Here W represents modeling error and system disturbances while V represents sensor noise. These vectors represent the deviations from the model that are detected with AR, so they will be henceforth left out of the model equations while AR is derived.

Figure 2.1 is a graphical depiction of the equation 2.1 system. It gives a good intuition of how separable a linear system is mathematically.

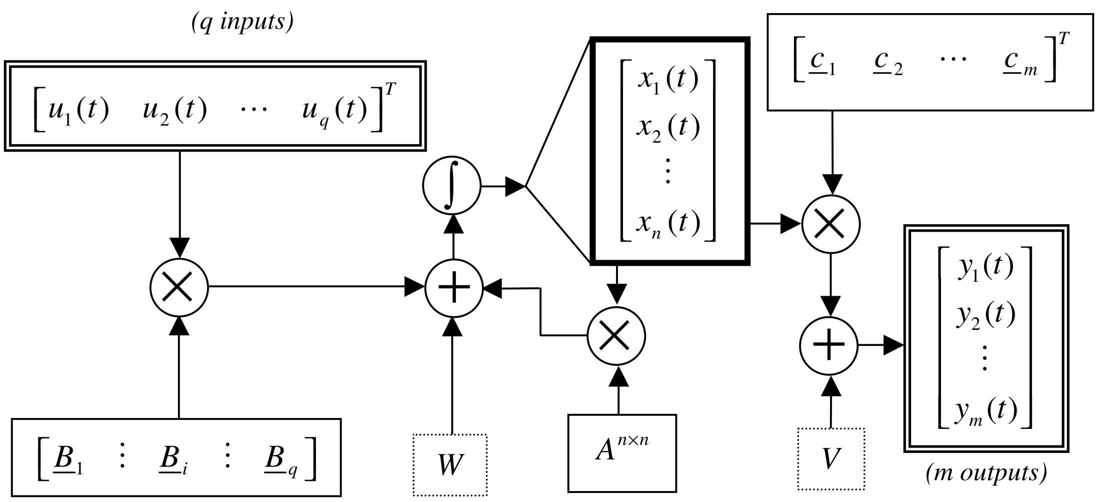


Figure 2.1 Linear state-space control model.

The model parameters (A,B, and C) are constants independent of the state, and all operations are simple products or sums. Compared to the nonlinear system shown later in Figure 3.2 this system is easy to break into individual components.

This characteristic of linear systems is used to derive the classical observability matrix O:

$$O = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix}. \tag{2.3}$$

Note how this construction depends on the separability of A and C from the rest of the system and the same separability leads to a constant, independent of the state and time variables.

The normal use of O is to determine if a system is observable, where observable is defined as "all of the initial states $x_i(t)$ can be determined from known sensor and input values in a finite time interval [1, 14, 36]." It can be shown that this is equivalent to testing the rank of O; if $rank(O) = n$ then the system is observable. Intuitively, this is linked to how effective model-based fault detection is, as the model is based on the states. Mathematically, proper manipulation of O allows explicit listing of not only what is known about every state, but, by separating out the observability associated with individual sensors, how many different ways each state's value can be determined. AR traditionally uses an alternate expression of O that facilitates this:

$$O = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \end{bmatrix}, \quad \text{where} \quad C_j = \begin{bmatrix} \underline{c}_j \\ \underline{c}_j A \\ \underline{c}_j A^2 \\ \vdots \\ \underline{c}_j A^n \end{bmatrix} \quad \text{and } n_j = max(rank(C_j)). \tag{2.4}$$

This amounts to a simple rearrangement of the rows of O. (Care must be taken not to confuse the C_j elements of O with the $\underline{c}_j$ vectors of equation 2.1.) This arrangement allows us to easily examine the observability of every sensor individually, which will prove useful in AR.

The astute observer will note that the AR choice of n_j leads to an overdetermined C_j matrix that possesses one more row than it possesses rank. Each C_j has $n_j + 1$ rows, and the Cayley-Hamilton theorem guarantees that for a C_j of k rows

$$rank(C_j) = \begin{cases} k & k \leq n_j \\ n_j & k > n_j \end{cases}. \tag{2.5}$$

In other words, the rank of C_j increases by one with each additional row until the first row that does not increase the rank. After this, the rank of the matrix will not increase further. In AR, the full rank C_j matrix is used with the addition of one additional row. Although the last row does not increase the rank of the matrix, it does contain useful information. This extra term is used to cover the extra information implied by the model equation independent of the observability. The mathematical background for this will be made more clear in Chapter 3 during the discussion of appropriate observability matrices for nonlinear AR.

2.1.2 Analytical Redundancy Construction

The first expression needed for AR is the left null matrix of the observability matrix O:

$$\Omega O = [0]. \tag{2.6}$$

The Ω left null matrix is perpendicular to O. It is conceptually a space of dynamic behaviors that should not be observable in the system. Mathematically, each row of Ω is a linear combination of sensor results that will result in zero for any state.

As Ω is a set of basis vectors spanning the left null-space of O, it is not unique. This allows the choice of formulations of Ω that facilitate AR computation or clarify the AR test residuals.

The second core equation in AR is a method of expressing O in terms of only sensor outputs and control inputs.

Definition 2.1: "Dynamically Derived Observability," or O_{DD}: *The reformulation of the observability matrix in terms of control inputs and sensor readings as shown below.*

$$\underline{y}(t) = C\underline{x}(t) = \begin{bmatrix} \underline{c}_1 \\ \underline{c}_2 \\ \vdots \\ \underline{c}_n \end{bmatrix} \underline{x}(t) = \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \underline{c}_2 \cdot \underline{x}(t) \\ \vdots \\ \underline{c}_m \cdot \underline{x}(t) \end{bmatrix}. \tag{2.7}$$

Without loss of generality, look at the $\underline{c}_1$ portion of the equation only and take the following series of derivatives:

$$\begin{bmatrix} \underline{y}_1(t) \\ \dot{\underline{y}}_1(t) \\ \ddot{\underline{y}}_1(t) \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \frac{d}{dt}\underline{c}_1 \cdot \underline{x}(t) \\ \frac{d^2}{dt^2}\underline{c}_1 \cdot \underline{x}(t) \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \underline{c}_1 \cdot \dot{\underline{x}}(t) \\ \underline{c}_1 \cdot \ddot{\underline{x}}(t) \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \underline{c}_1 \cdot (A\underline{x}(t) + B\underline{u}(t)) \\ \underline{c}_1 \cdot \frac{d}{dt}\left(A\underline{x}(t) + B\underline{u}(t)\right) \\ \vdots \\ \end{bmatrix}.$$

Recurse:

$$= \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \underline{c}_1 \cdot (A\underline{x}(t) + B\underline{u}(t)) \\ \underline{c}_1 \cdot \left(A\left(A\underline{x}(t) + B\underline{u}(t)\right) + B\dot{\underline{u}}(t)\right) \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \cdot \underline{x}(t) \\ \underline{c}_1 A\underline{x}(t) + \underline{c}_1 B\underline{u}(t)) \\ \underline{c}_1 A^2\underline{x}(t) + \underline{c}_1 AB\underline{u}(t) + \underline{c}_1 B\dot{\underline{u}}(t) \\ \vdots \\ \end{bmatrix}.$$

Reorder:

$$\begin{bmatrix} \underline{c}_1 \\ \underline{c}_1 A \\ \underline{c}_1 A^2 \\ \vdots \\ \underline{c}_1 A^n \end{bmatrix} \underline{x}(t) =$$

$$O_{DD} = \begin{bmatrix} y_1(t) \\ \dot{y}_1(t) \\ \ddot{y}_1(t) \\ \vdots \\ \frac{d^n}{dt^n} y_1(t) \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \underline{c}_1 B & 0 & 0 & \dots & 0 \\ \underline{c}_1 AB & \underline{c}_1 B & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \underline{c}_1 A^{n-1} B & \dots & \underline{c}_1 A^{n-i} B & \dots & \underline{c}_1 B \end{bmatrix} \begin{bmatrix} \underline{u}(t) \\ \dot{\underline{u}}(t) \\ \vdots \\ \frac{d^n}{dt^n} \underline{u}(t) \end{bmatrix} \square \quad (2.8)$$

The dynamically derived observability O_{DD} is identical to the traditional control theoretic $O \cdot \underline{x}(t)$ as long as the system behavior is following the control model.

It is interesting to consider the derivation of the O_{DD} observability above with respect to the concept of observability in general. O_{DD} shows that looking at the rank of the observability matrix is essentially looking at repeated derivatives of the sensor readings and modified derivatives of the control inputs to see if they will generate vectors that span the dimension of the state space. This spanning of the dimension of the state space is in fact exactly what we want observability to show us.

The AR test residuals are simply generated by multiplying equations 2.6 and 2.8 above.

$$\Omega O_{DD} = R \quad (2.9)$$

R is the set of AR residual tests. Each element of R is a linear combination of sensor values and control inputs. Ω supplies constant coefficients and O_{DD} supplies both constant coefficients and the sensor and control signal variable terms. R is also referred to as the *generalized parity vector* and will be equal to a column of zeros as long as the O_{DD} observability is an accurate description of O. ($\Omega O = [0] \Rightarrow \Omega O\underline{x}(t) = \underline{0} \Rightarrow \Omega O_{DD} = \underline{0}$.) Since system models are seldom perfect and often affected by various forms of noise, in practice R is a vector of relatively small nonzero signals, or *residuals*. (This is why the vector is called R.) When the system fails in a manner that changes the system model significantly, however, the O_{DD} equation will no longer be valid and R will produce nonzero results. These large nonzero signals allow for easy and clear fault detection.

In practice, the tests contained in R look like sensor comparisons, model equations, or derivatives of these, although this depends somewhat on the choice of Ω. (As Ω is a set of basis vectors for a space, it is of course not unique.) This is a result of how the AR tests are derived - the model equations and sensors are the "raw material" used in AR. This is also intuitively satisfying for model-based testing, as these are the intuitively obvious tests in this context. AR ends up providing theoretical justification and limits on the fault detection tests one would naturally want to attempt. Chow and Willsky [5] showed that AR tests derived in this manner test all of the possible observable deviations from the model and are also linearly independent, and thus are as small as a complete set of tests can be. These guarantees are very useful in a fault detection system.

2.1.3 Analytical Redundancy Information Flow

Figure 2.2 depicts the flow of information in an AR system. The control model and system have plain black outlines. These represent the system being analyzed and presumably exist prior to the application of AR.

The derivation of the AR test residuals is denoted by a dotted black border in the figure. This section of the diagram represents the derivation of the AR tests R_i using the control system model as discussed above. This is the symbolic stage of AR, where the equations are calculated, and is done off line before the AR is implemented, and need only be done once.

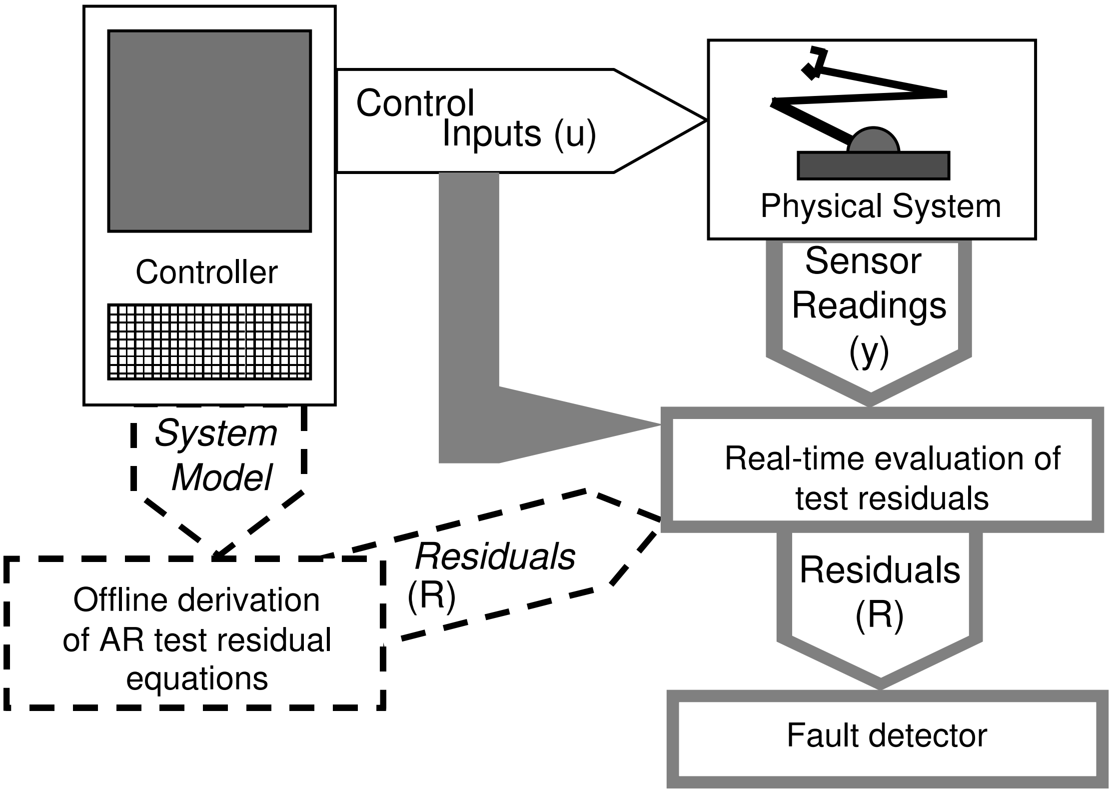


Figure 2.2 Data Flow in AR.

The application of the R_i residuals is shown by a thick gray border in the figure. These residuals are the part of AR that must be done online as the system operates. Every time step the results of the AR test residuals are calculated and the results output to the fault detector. This is the calculation stage of AR, where the previously determined test residual equations are evaluated.

2.1.4 A Geometric Notion of Analytical Redundancy

Figure 2.3 gives a basic geometric interpretation of the internal workings of an analytical redundancy test. The curved surfaces represent the observability spaces O.

For standard AR as discussed in this chapter these are linear spaces, but analytical redundancy for curved nonlinear spaces will be discussed in Chapter 3.

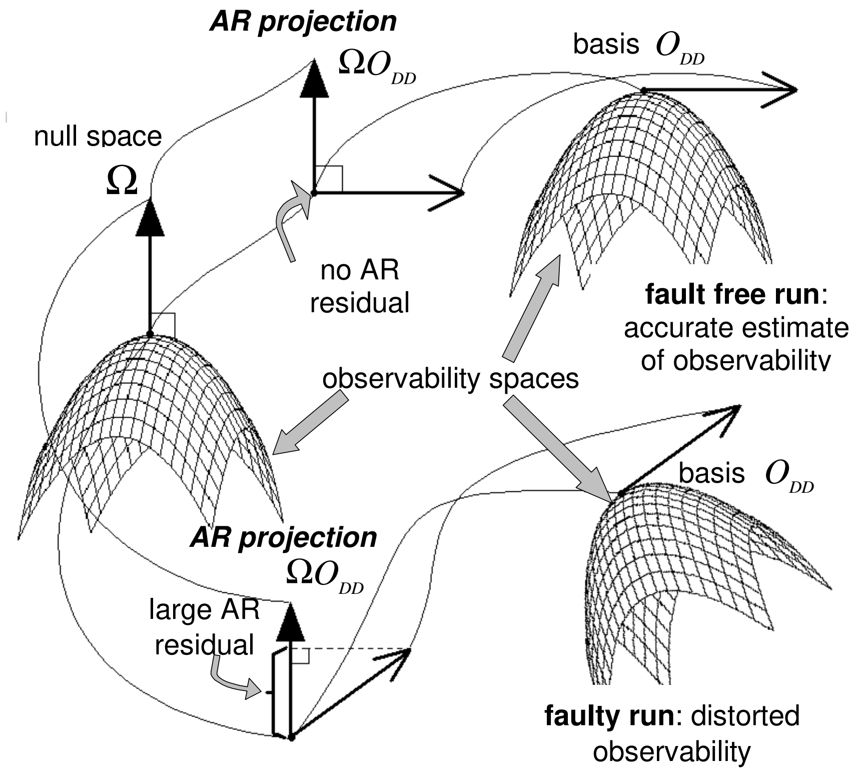


Figure 2.3 A graphical notion of AR.

The observability on the left is used to generate the null matrix Ω, represented here by a single vector. (Note that Ω is a left-null, rather than a right-null.) The right hand observabilities are both dynamically derived observabilities (O_{DD}s). The upper O_{DD} is derived from a working system and is virtually the same as the observability used to generate the null-space Ω. The vector on top of the observability represents a basis for this space. (These spaces are usually multi-dimensional, of course. The single basis vector is for clarity.) The lower observability space is derived from a faulty system. The faults have caused a significant change in the span of the space and the basis vector is skewed to represent this.

Note that the AR equation 2.9 can be viewed as a projection of the dynamically derived observability O_{DD} onto the null-space Ω. The AR projections in the middle of the figure show the effects of the AR equation on the spaces in question.

For the working system, the basis vector of the O_{DD} space are still very nearly perpendicular to the basis vector of the null-space Ω, so the results of the AR test are small. For the failed system, this is not true. The fault has altered O_{DD} sufficiently that its basis vector has a significant component in the Ω null-space. This component shows up as a large residual when the analytical redundancy equation is applied, so the fault is detected.

2.2 Linear AR Example

AR is best appreciated in context, so let us consider AR fault detection for the following simple system:

$$\begin{aligned} \underline{\dot{x}}(t) &= \begin{bmatrix} 0 & \beta \\ \gamma & 0 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) \\ \underline{y}(t) &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{x}(t) = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \underline{x}(t) \end{aligned} . \tag{2.10}$$

This system is mathematically uncomplicated and clearly instrumented to make the example easy to follow. The observability matrix is:

$$O = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \beta \\ \beta\gamma & 0 \\ 0 & 1 \\ \gamma & 0 \\ 0 & \beta\gamma \end{bmatrix}. \tag{2.11}$$

Note that each C_i is rank two, and that three ranks have been kept in each, as discussed in section 2.1.1. Now choose Ω and calculate O_{DD}:

$$\Omega O = [0] \Rightarrow \Omega = \begin{bmatrix} 0 & 1 & 0 & -\beta & 0 & 0 \\ -\beta\gamma & 0 & 1 & 0 & 0 & 0 \\ -\gamma & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -\beta\gamma & 0 & 1 \end{bmatrix} \tag{2.12}$$

$$O_{DD} = \begin{bmatrix} y_1(t) \\ \dot{y}_1(t) - c_1 Bu(t) \\ \ddot{y}_1(t) - c_1 ABu(t) - c_1 B\dot{u}(t) \\ y_2(t) \\ \dot{y}_2(t) - c_2 Bu(t) \\ \ddot{y}_2(t) - c_2 ABu(t) - c_2 B\dot{u}(t) \end{bmatrix}. \tag{2.13}$$

Multiply Ω and O_{DD} together to get the boxed AR test residuals below:

$$\Omega O_{DD} = [R] \Rightarrow$$

$$\boxed{R_1 = -\dot{y}_1(t) + c_1 Bu(t) + \beta y_2(t)} \tag{2.14}$$

$$\boxed{R_2 = -\ddot{y}_1(t) + \beta\gamma y_1(t) + c_1 ABu(t) + c_1 B\dot{u}(t)} \tag{2.15}$$

$$\boxed{R_3 = -\dot{y}_2(t) + \gamma y_1(t) + c_2 Bu(t)} \tag{2.16}$$

$$\boxed{R_4 = -\ddot{y}_2(t) + \beta\gamma y_2(t) + c_2 ABu(t) + c_2 B\dot{u}(t)} \tag{2.17}$$

R_1 and R_3 are clearly the model equations, and R_2 and R_4 are their first derivatives, although they draw on both sensors and model equations. In this way AR tests both the dynamic behavior of the system and the consistency between the sensors. It effectively uses the coupled nature of the model equations to compare the sensors.

2.3 Ad Hoc Extension of Linear AR

Piecewise linear AR (PLAR) and nearly nonlinear AR (NNAR) are ad hoc but useful methods of extending AR tests into the nonlinear realm. (Why this is desirable is presented in more detail in Chapter 3.) They were both developed early in the course of this work to deal with the nonlinearities of the hydraulic actuator described in Chapter 4 [25, 26, 30, 33]. Neither of these methods are truly nonlinear in nature, however, as both use the linearized system to derive the AR tests.

2.3.1 Piecewise Linear AR

One way to deal with a system model that is nonlinear enough to change considerably as it moves through the workspace is to create several sets of AR tests for the system linearized about state vectors located in each region of interest. In the local region each set would be more accurate than a general linearization of the control equations for the entire workspace. An example of this technique, and the improvement it brings, is illustrated below in Figure 2.4. In this case, generated for the hydraulic servovalve system from Chapter 4, the pressure-valve position workspace in which the flow equation is nonlinear is divided up into nine equal regions. The model equation is linearized about a point at the center of each and normal AR tests are derived. (Due to the symmetry of the system, only four linearizations are needed in practice.) During operation, the AR test used is the one that was linearized about a point closest to the current position, with interpolated transitions near the borders of each region.

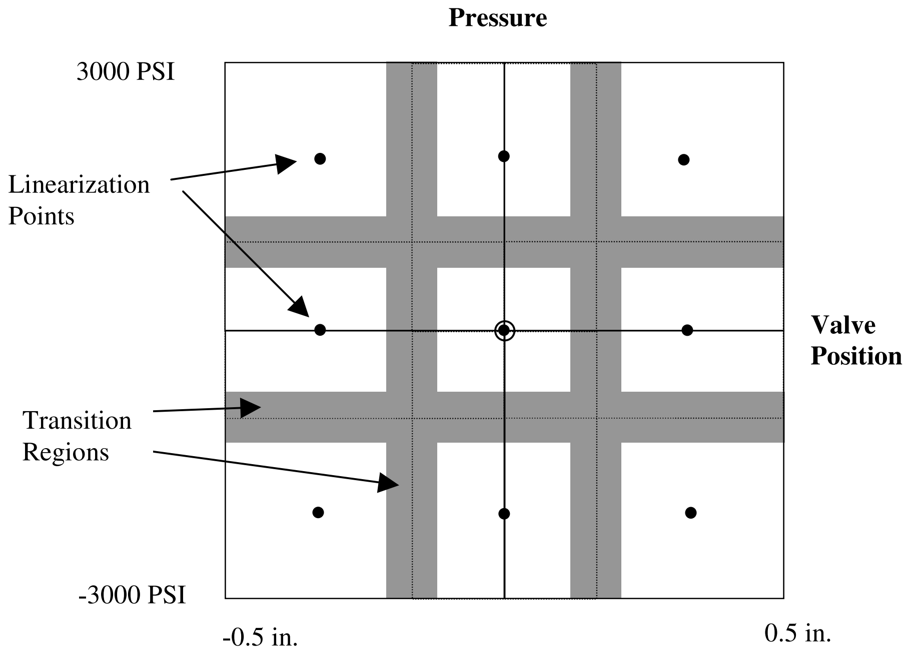


Figure 2.4 PLAR division of hydraulic servovalve workspace.

Figure 2.5 shows the results from a fault free run and a faulty run (the fault was a large leak) of PLAR tests on a simulated hydraulic servovalve [25, 26]. The fault free run shows how PLAR minimizes the drifting away from the model errors of pure linear AR. Before the test residual can drift far from the point about which it was linearized, the system transitions into another, more appropriate AR test linearized about a point closer to the actual state of the system, leading to a saw-toothed residual about the nominally correct zero value. The results in a test run with a large leak fault added to the hydraulic system show that this saw-tooth is about an order of magnitude smaller than the fault signature.

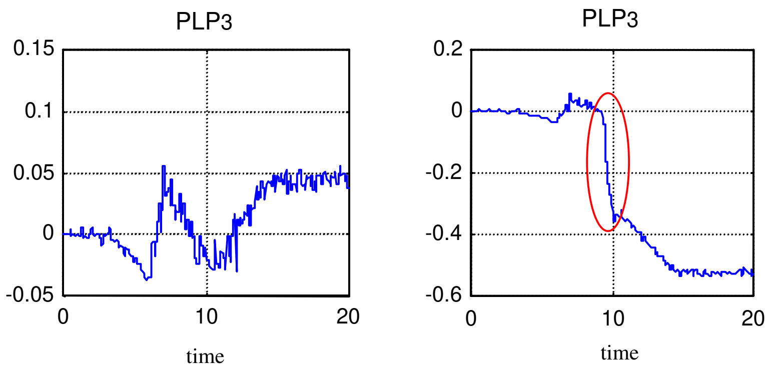


Figure 2.5 PLAR fault free (left) and faulty (right) test results for simulated hydraulic servovalve.

2.3.2 Nearly Nonlinear AR

Nearly nonlinear analytical redundancy is a natural outgrowth of PLAR. Although each PLAR partition uses a different linearization of the control equations, the different linearizations all share the same basic form - they are tangents to the control system at the point of linearization. This means that the AR tests generated by the different linearizations will be, in essence, tangents to some more accurate nonlinear AR test. It is reasonable to approach this test by dividing the workspace into many closely packed regions and taking appropriate linearizations. In the limit of infinitely small regions, this is a nonlinear AR test! However, there is a much simpler method of finding these tests. Recall that AR tests tend to be combinations of the model equations, sensor comparisons, and their derivatives. By performing linear AR on a linearized system and identifying the relationship between the AR residual tests and the control model and sensors, it is possible to find nonlinear AR tests by simply duplicating this relationship to the nonlinear control system and sensors! For example, if the AR tests for the linear system are the linearized model equations, their first derivatives, and direct sensor comparisons between the linearized sensors, the nonlinear model equations, their first derivatives, and comparisons of the nonlinear sensors can be used as NNAR tests.

Why, then, are these tests referred to as merely "nearly nonlinear" rather than "fully nonlinear?" Unfortunately, as they use the linear observability matrix they are still using linear approximations of the observability null-space, Ω. It can be shown that this is different than the nonlinear null-space; in fact the two can have a different rank. The AR guarantees of testing for all of the possible model deviations efficiently are thus not valid in PLAR and NNAR. *Any* AR method that doesn't use the full nonlinear observability space will suffer from this drawback. This is what makes fully nonlinear AR using the nonlinear observability space, as seen in Chapter 3, so desirable, and relegates PLAR and NNAR to secondary roles. These methods do not require extensive nonlinear calculation to use, but they are essentially ad hoc in nature.

2.4 Other Work in Nonlinear and Robotic AR

AR for robotic systems has been examined by several researchers in the past. Visinsky et. al. examined robotic AR as a part of a larger multi-layered fault detection and fault tolerance framework [44, 45]. This work used standard linear AR and used other methods, such as reachable measurement intervals (RMI) and model-based thresholding (ThMB), to deal with the troubling nonlinear behaviors of robotic systems. ThMB and RMI dealt with the model accuracy problems of linear AR by predicting when the model would do poorly and expanding detection thresholds appropriately, rather than reducing the model inaccuracies. The desire to remove these inaccuracies provided motivation for the research in this thesis.

Considerable work has been done using the concept of nonlinear observers in the same role as AR [4, 15]. In fact, it has been shown that AR and observer based methods are equivalent in the linear case [34]. However, the proof in [34] is only applicable to linear systems, so the nonlinear observers are not the same as nonlinear AR. Additionally, the nonlinear observer based method [15] lacks the span guarantees of NLAR.

Zhirabok and Preobragenskaya have presented work with nonlinear AR test residuals based on observer theory [50]. Nonlinear test residuals are generated by following an algorithm for restating the model equations in terms of inputs u_i and outputs y_i. This method has problems similar to NNAR described in section 2.3.2 above, as it does not use the observability to maintain the guarantees that make AR so desirable.

Wünnenberg and Frank have investigated methods for thresholding linear robotic AR test residuals to compensate for various modeling inaccuracies [11, 48]. Instead of adapting the AR tests to the nonlinear systems, this work takes the practical approach of developing a system that runs in parallel to the AR system, predicts when the modeling inaccuracies will likely be large, and increases the thresholds on the AR residual tests appropriately.

Starosweicki and Comtet-Varga have produced some interesting work describing rigorous nonlinear AR limited classes of nonlinear systems [41, 42]. This work discusses several methods of rigorously developing various AR-like test residuals without actually using the nonlinear observability. This work considers the spanning issue, but is limited by its neglect of the core observability issues of AR.

Isidori and De Persis have derived a geometric residual generator using nonlinear observability [38]. These residuals are similar in concept to those generated by AR in that they use the null space of the observability to test the system behavior. However, they are limited to checking for faults where both the disturbance and fault dynamics are known well enough to model accurately, making span guarantees much more troublesome [12, 37]. This limits the Isidori approach to detecting well known and well modeled faults, while AR and NLAR are geared to detect all deviations from the model, as seen in the following chapter.

Chapter 3

Derivation of Nonlinear Analytical Redundancy

3.1 Linear AR to Nonlinear AR, a Difficult Journey

Nonlinear systems in general are difficult to deal with mathematically [20, 21, 25, 33, 36]. They simply lack the many useful properties of linear systems that have been exploited over the years to make them tractable. Nonlinear systems do not obey the superposition principle and can exhibit behaviors such as finite escape times, multiple equilibria, or chaos, that are quite difficult to deal with [21]. Observability, and by extension analytical redundancy, are made much more difficult as a result of theses properites.

However, the very intractability of nonlinear systems makes good fault detection techniques like AR more important. The behavior of a nonlinear system is harder to predict and control, resulting in reduced safety and reliability, which in turn makes fault detection more important. However, model-based fault detection techniques require good models to be effective, and traditional linear AR is generally unsatisfactory for systems with significant nonlinear components. Figure 3.1 shows model error for the hydraulic servovalve robot motor system discussed in Chapter 4.

This system has a simple square root nonlinearity as seen in equation 3.1,

$$Q = x_v K_f \sqrt{\tfrac{2}{\rho}(p_s - p_l)} = d_m \dot{\theta}_m + (c_{im} + c_{em})p_l + \frac{v_t \dot{p}_l}{4\beta_e}. \tag{3.1}$$

(Note that the variable are all defined in Chapter 4.) Casual perusal shows there are significant areas of the workspace where the linear approximation error is over 50%! This kind of behavior is not unusual in robotic systems either. The canonical robotic arm has many nonlinear coupling and friction effects. Even a simple two joint revolute planar manipulator, as discussed in Chapter 5, is nonlinear.

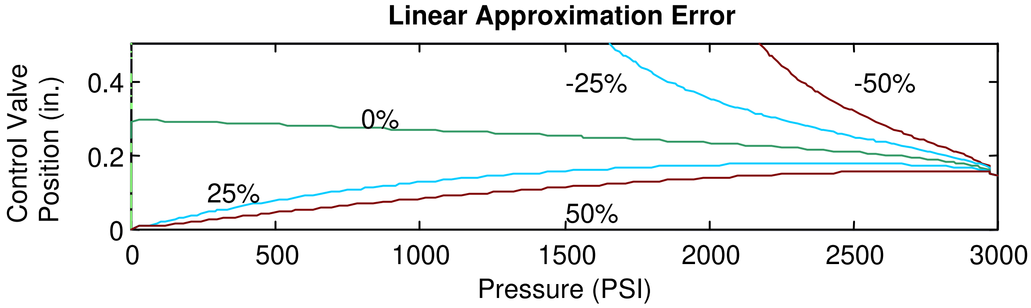


Figure 3.1 Error due to linearization for hydraulic servovalve.

3.1.1 Nonlinear Observability

Consider the following nonlinear state-space control system model analogous to equation 2.1 and Figure 2.1:

$$\begin{aligned} \dot{\underline{x}}(t) &= f(\underline{x}(t)) + g(\underline{x}(t)) \cdot \underline{u}(t) + W \\ \underline{y}(t) &= h(\underline{x}(t)) + V. \end{aligned} \tag{3.2}$$

It is clear that this formulation is much less separable than the linear system. The A, B, and C matrices of constants which defined the linear system are replaced by the $f(\underline{x}(t))$, $g(\underline{x}(t))$, and $h(\underline{x}(t))$ *functions* which are not nearly so tractable. As $f(\underline{x}(t))$ is equivalent to $A \cdot \underline{x}(t)$ rather than A, linear techniques that rely on the use of A alone, such as observability, become much more complex.

Fortunately, recent developments in nonlinear control theory include a notion of local observability applicable to nonlinear systems that are relatively smooth [20, 36]. The local nature of the observability restricts observability to a region of the workspace "near" the current state. Theoretically, "near" means there is an open neighborhood around the current state where the observability is valid. For practical systems, this means that the sampling rate should be high enough that the

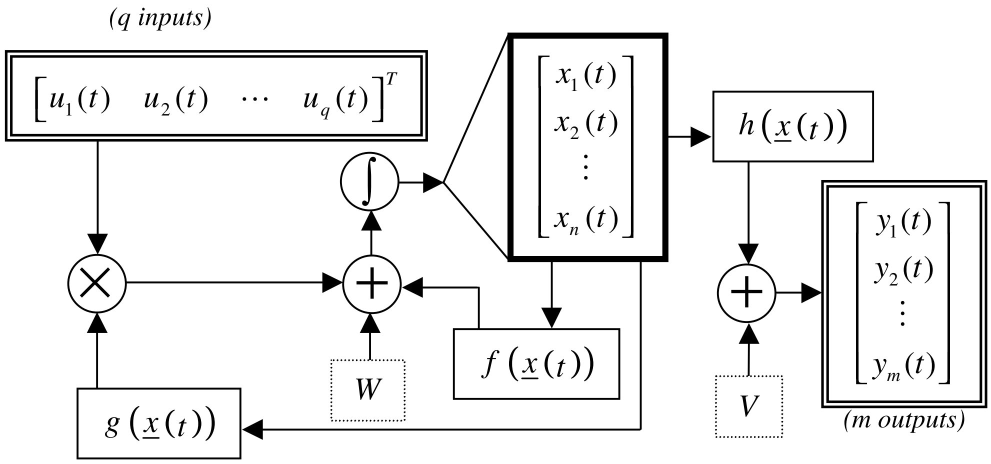


Figure 3.2 Nonlinear state-space control model.

system does not invalidate the current model over the time needed to determine the observability. This is not particularly restrictive in the case of analytical redundancy, as it only uses very short sensor and control histories (as necessary to approximate the derivatives). Any system that is sampled closely enough to allow effective model-based fault detection and close approximations of the derivatives will keep these AR histories in the "near" area, so the AR calculations will be valid. Note that this restriction to a local area is somewhat similar to the restrictions on PLAR and NNAR, but that full nonlinear observability characteristics of the region are used, rather than the linear observability of a model linearized about a nearby state.

The smoothness requirements on this observability are somewhat complex, and discussed fully in Isidori [20]. For the purposes of this thesis it will suffice to say that the system must be a manifold, or that any local area of the workspace should be smooth in the classical sense. This smoothness requirement is more restrictive, as there are real world system behaviors that are not smooth, such as Coulomb friction.

However, there are many important systems demonstrating smooth nonlinearities that this nonlinear notion of observability can easily deal with. For example, in the hydraulic example of equation 3.1, the nonlinearity is a square root. This is clearly smooth in the classical sense. NLAR using the Isidori notion of nonlinear observability is clearly worthwhile for dealing with such systems.

According to this notion of observability, a system is *locally observable* if:

$$Rank(\nabla\mathcal{O}) = n, \tag{3.3}$$

where:

$$\mathcal{O} = span\{h_j, L_{k_1}L_{k_2}\dots L_{k_p}h_j\},$$

$$j = 1,\dots,m \quad p = 1,2,\dots \quad k_i \in span\{f, g_1,\dots,g_q\},$$

and

$$L_{\underline{k}}h = \sum_{i=1}^{n} \frac{\partial h(\underline{x})}{\partial x_i} k_i(\underline{x}) = \langle \nabla(h(\underline{x})), \underline{k}(\underline{x})\rangle\,,$$

is the "Lie derivative of scalar function h in the direction of vector function $\underline{k}$." Recursed Lie derivatives, as seen in the definition of $\mathcal{O}$ above, are written in one of the following ways:

$$L_{\underline{i}}\left(L_{\underline{j}}\left(L_{\underline{k}}h\right)\right) = L_{\underline{i}}L_{\underline{j}}L_{\underline{k}}h = L_{\underline{ijk}}h.$$

As the direction vector is often a function, and must be a vector, it is common to omit the vector notation for the subscript, like so: $L_f h$.

Lemma: 3.1

1. *If h is constant, $L_{\underline{k}}h = 0$.*

2. *If h is linear and $\underline{k}$ is constant, , $L_{\underline{k}}h$ is also constant.*

3. *If both h and $\underline{k}$ are linear, $L_{\underline{k}}h$ is also linear.* □

These are all trivial consequences of the definition of Lie differentiation, and can be used to show, for a linear system, that the $\mathcal{O}$ nonlinear observability is equivalent to the linear notion O.

In a linear system, f and h are linear, equal to $A\underline{x}$ and $C\underline{x}$ respectively, and all g_i are constants equal to B_i. It is possible to choose $k_i \in span\{f, g_1, \ldots, g_q\}$ to be equal to the $f = A\underline{x}$ and $g_i = B_i$ functions and assume $h = C\underline{x}$ is single element without loss of generality.

From Lemma 3.1 and the Lie derivative definition, it can be seen that any element of O containing a Lie derivative with respect to a $g_i = B_i$ will be constant or zero. As the quantity of interest is the rank of $\nabla\mathcal{O}$, elements of $\mathcal{O}$ that contain a Lie derivative with respect to g_i are ignored. (The gradient of a constant function is zero.) Eliminating these terms, the following terms remain:

$$\mathcal{O} = span\{h, L_f h, L_{ff} h, L_{fff} h, \ldots\} = span\{C\underline{x}, CA\underline{x}, CA^2\underline{x}, CA^3\underline{x}, \ldots\}.$$

Taking the gradient, $\nabla\mathcal{O}$ is the span of the rows of the matrix:

$$Rank(\nabla\mathcal{O}) = Rank\left(\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix}\right).$$

The requirement $Rank(\nabla\mathcal{O}) = n$ is thus clearly identical to the well-known linear observability criterion, based on the linear observability matrix O:

$$Rank(O) = Rank\left(\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix}\right) = n.$$

Note that the linear O and nonlinear $\mathcal{O}$ are equivalent concepts of observability but not identical to each other mathematically. $O\underline{x}$ is equal to $\mathcal{O}$ and O is equivalent to $\nabla\mathcal{O}$. This can be somewhat confusing, but these notations are established [5]. Table 3.1 clarifies this relationship.

3.1.2 $\mathcal{O}$ in Nonlinear Analytical Redundancy

Although a valid notion of nonlinear observability is a necessary prerequisite for nonlinear AR, it is not a sufficient one. One still needs to be able to determine the null-spaces corresponding to those of the observability matrices exploited in linear AR. Linear observability has properties that are exploited in Chapter 2 that does not easily transfer to the nonlinear case.

The core problem involves the elimination of the state from the AR equations. Consider the Ω null matrix. In AR for linear models, O (or $\nabla\mathcal{O}$) is a matrix of constants and thus so is Ω. For a nonlinear system, $\nabla\mathcal{O}$ is an explicit function of the state and by extension so is Ω. Any nonlinear AR method has to come to terms with this state dependency. Additionally, the more complex dependence of the control system elements on the state will make the construction of a modified observability matrix similar to the O_{DD} of equation 2.8 much more difficult. Again, isolating the state is difficult, nor is it immediately obvious how to construct an observability space analogue. Techniques for overcoming these stumbling blocks will be discussed in section 3.2.

Linear Value	Linear Notation	Nonlinear Notation
$[C\underline{x}, CA\underline{x}, CA^2\underline{x}, CA^3\underline{x}, \ldots]^T$	$O\underline{x}$	$\mathcal{O}$
$[C, CA, CA^2, CA^3, \ldots]^T$	O	$\nabla\mathcal{O}$

Table 3.1 Mathematical relationship between linear and nonlinear observability spaces.

3.1.3 Discretization Issues

The original AR work by Chow and Willsky [5] was done in a sampled data system using discrete mathematics. In some ways, this is the natural domain of AR, which in practice will deal with data sampled from real systems. However, during the course of the derivation of the nonlinear analytical redundancy technique that forms the core of this thesis, discretization of the system was delayed until the final step of implementing the NLAR tests. As all of the mathematics in AR apply equally well in the continuous domain, this approach is certainly valid. However, the nonlinear observability techniques discussed in Isidori [20] are continuous in nature, and not trivially convertible to discrete forms in a rigorous manner. In fact, investigation of the interaction of data sampling with the nonlinear system's modeling accuracy is an avenue of research that would be well worth pursuing. For the purposes of this thesis, however, it is assumed that discretization of the continuously derived NLAR tests at a high sampling rate is sufficient to avoid significant error due to this effect.

3.2 NLAR

The novel innovations of this thesis are methods of reshaping and rethinking the nonlinear observability $\mathcal{O}$ presented by Isidori [20] and the AR equation $\Omega O = [0]$ so that the two mesh into a rigorous and useful nonlinear AR technique.

3.2.1 Nonlinear Ω

The obvious approach to converting the $\mathcal{O}$ notion of observability for AR use would be to follow the linear AR model as closely as possible. This requires stacking the elements of the $\nabla\mathcal{O}$ to create a matrix analogous to the linear O matrix and finding its left null, as seen in equation 3.4.

$$\Omega \nabla \mathcal{O}_M = [0] \tag{3.4}$$

where:

$$\nabla \mathcal{O}_M = \nabla \begin{bmatrix} \vdots \\ h_j \\ \vdots \\ L_{k_i} h_j \\ \vdots \\ L_{k_i} L_{k_l} h_j \\ \vdots \end{bmatrix}$$

and

$$j = 1, \ldots, m \quad \{i, l, p\} = 1, 2, \ldots \quad k_i \in span\{f, g_1, \ldots, g_q\}.$$

Unfortunately, this notion of the observability matrix did not lead to any useful AR relations, despite extensive effort expended trying to devise them. The main cause of this is the lack of useful alternate formulations of this observability, as provided by equation 2.8 in the linear case.

The novel grouped nonlinear observability as expressed in equation 3.5 yields a much more tractable formulation of the observability matrix for the purposes of NLAR.

Definition 3.1 : Triangular Nonlinear Observability, or $\mathcal{O}_\triangle$: *The grouped form of the nonlinear observability is expressed below in equation 3.5. This formulation sums the elements of $\mathcal{O}$ that are Lie differentiated to the same degree. It is roughly triangular in form when written out and a vital component of NLAR,*

$$\mathcal{O}_{\triangle} = \begin{bmatrix} h\left(\underline{x}(t)\right) \\ \sum\limits_{j=0}^{q} L(j) \\ \sum\limits_{l=0}^{q}\sum\limits_{j=0}^{q} L(j,l) \\ \sum\limits_{m=0}^{q}\sum\limits_{l=0}^{q}\sum\limits_{j=0}^{q} L(j,l,m) \\ \vdots \end{bmatrix} \tag{3.5}$$

where:

$$L(j,l,m,\ldots) = (u_j u_l u_m \ldots)\, L_{\ldots k(m)k(l)k(j)} h\left(\underline{x}(t)\right)$$

$$u_0 = 1, \quad k(j) = \begin{cases} f & ,j = 0 \\ g_j & ,j \neq 0 \end{cases} \quad \square$$

Note that this formulation introduces the potential of losing rank by summing the elements of $\mathcal{O}$ that share the same degree of Lie differentiation. In some cases, this might lead to underestimating the span of the observability space. Fortunately, this reduction in rank will seldom be an issue in real systems. In sensitive systems where NLAR analysis is appropriate, thorough instrumentation of the system is common, and the minor potential losses of rank caused by equation 3.5 are not likely to reduce the overall observability of the system. As our dynamically derived observability ($\mathcal{O}_{\triangle DD}$) derived in section 3.2.2 requires this particular observability ($\mathcal{O}_{\triangle}$) to function, this kind of thorough instrumentation is reassuring.

Additionally, this formulation multiplies many of the terms by the control inputs (u_i). However, the span of the gradients of these terms is unaffected by multiplying the terms by a scalar independent of $\underline{x}(t)$ such as u_i. This span is what defines the observability, so this modification does not affect the utility of the $\mathcal{O}_{\triangle}$ formulation for NLAR. $\nabla\mathcal{O}_{\triangle}$ is equivalent to the linear observability O, as all the gradients of Lie derivatives with respect to g are zero in the linear case.

Finally, for nonlinear systems $\Omega\nabla\mathcal{O} = [0] \not\equiv \Omega\mathcal{O} = [0]$. This makes taking the gradient of $\mathcal{O}_{\triangle}$ in the AR equation impossible. Thus, in the following sections, the

Ω generated by

$$\Omega \mathcal{O}_{\triangle} = [0] \tag{3.6}$$

is used. Some consequences of this will be seen in the next two sections.

3.2.2 Nonlinear Dynamically Derived Observability ($\mathcal{O}_{\triangle DD}$)

Here a novel nonlinear dynamically derived observability matrix, $\mathcal{O}_{\triangle DD}$, is derived. This reformulation of the observability in terms of control inputs u_i and sensor readings y_i is needed to complete the AR parity equation.

Initially, it is assumed that the sensor function $h(\underline{x}(t))$ is linear and represented by $C \cdot \underline{x}(t)$ and that there is only one control input u. The first step is to examine the stacked time derivatives of y:

$$\begin{bmatrix} \underline{y}(t) \\ \underline{\dot{y}}(t) \\ \underline{\ddot{y}}(t) \\ \vdots \\ \\ \end{bmatrix} = \begin{bmatrix} C \cdot \underline{x}(t) \\ \frac{d}{dt} C \cdot \underline{x}(t) \\ \frac{d^2}{dt^2} C \cdot \underline{x}(t) \\ \vdots \\ \\ \end{bmatrix} = \begin{bmatrix} C \cdot \underline{x}(t) \\ C \cdot \underline{\dot{x}}(t) \\ C \cdot \underline{\ddot{x}}(t) \\ \vdots \\ \\ \end{bmatrix} = \begin{bmatrix} C \cdot \underline{x}(t) \\ C \cdot \left(f\left(\underline{x}(t)\right) + g\left(\underline{x}(t)\right) \underline{u}(t) \right) \\ C \cdot \frac{d}{dt} \left(f\left(\underline{x}(t)\right) + g\left(\underline{x}(t)\right) \underline{u}(t) \right) \\ \vdots \\ \\ \end{bmatrix}.$$

Chaining the derivatives of f and g in the third line and converting the second line to Lie derivative form:

$$= \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t)) \underline{u}(t) \\ C \cdot \left(\nabla f\left(\underline{x}(t)\right) \cdot \underline{\dot{x}}(t) + \nabla g\left(\underline{x}(t)\right) \cdot \underline{\dot{x}}(t) \underline{u}(t) + g\left(\underline{x}(t)\right) \underline{\dot{u}}(t) \right) \\ \vdots \\ \\ \end{bmatrix}.$$

Then expand the $\underline{\dot{x}}$ terms again:

$$= \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t))\underline{u}(t) \\ C \cdot \begin{pmatrix} \nabla f\left(\underline{x}(t)\right) \cdot \left(f\left(\underline{x}(t)\right) + g\left(\underline{x}(t)\right)\underline{u}(t)\right) + g\left(\underline{x}(t)\right)\underline{\dot{u}}(t) \\ +\nabla g\left(\underline{x}(t)\right) \cdot \left(f\left(\underline{x}(t)\right) + g\left(\underline{x}(t)\right)\underline{u}(t)\right)\underline{u}(t) \end{pmatrix} \\ \vdots \\ \end{bmatrix}.$$

Converting the third line to Lie derivative form:

$$\begin{bmatrix} \underline{y}(t) \\ \underline{\dot{y}}(t) \\ \underline{\ddot{y}}(t) \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t))\underline{u}(t) \\ \begin{pmatrix} L_{ff}(C \cdot \underline{x}(t)) + L_{gf}(C \cdot \underline{x}(t))\underline{u}(t) + L_{fg}(C \cdot \underline{x}(t))\underline{u}(t) \\ +L_{gg}(C \cdot \underline{x}(t))\underline{u}^2(t) + Cg\left(\underline{x}(t)\right)\underline{\dot{u}}(t) \end{pmatrix} \\ \vdots \\ \end{bmatrix}.$$

Then simply regroup to get a matrix that looks like $\mathcal{O}_\triangle$.

$$\begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t))\underline{u}(t) \\ \begin{pmatrix} L_{ff}(C \cdot \underline{x}(t)) + L_{gf}(C \cdot \underline{x}(t))\underline{u}(t) \\ +L_{fg}(C \cdot \underline{x}(t))\underline{u}(t) + L_{gg}(C \cdot \underline{x}(t))\underline{u}^2(t) \end{pmatrix} \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \underline{y}(t) \\ \underline{\dot{y}}(t) \\ \underline{\ddot{y}}(t) \\ \vdots \\ \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ Cg(\underline{x}(t))\underline{\dot{u}}(t) \\ \vdots \\ \end{bmatrix}. \tag{3.7}$$

Note that the LHS of equation 3.7 is $\mathcal{O}_\triangle$ for the postulated system; this is the observability reformulation needed for nonlinear AR. This will be the dynamically derived observability for nonlinear systems $\mathcal{O}_{\triangle DD}$. In fact, this construction is valid for multiple control inputs as well. Equation 3.8 gives explicit $\mathcal{O}_{\triangle DD}$ values for a multiple nonlinear input system up to the fourth derivative.

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} \underline{y}(t) - 0 \\ \underline{\dot{y}}(t) - 0 \\ \underline{\ddot{y}}(t) - \sum \underline{\dot{u}}(t) L_g \\ \frac{d^3}{dt^3}\underline{y}(t) - \begin{pmatrix} \sum \underline{\ddot{u}}_i(t) L_{g_i} + \sum \underline{\dot{u}}_i(t) L_{\underline{\dot{x}} g_i} + \sum \underline{\dot{u}}_i(t) L_{g_i f} \\ + \sum \underline{\dot{u}}_i(t) L_{f g_i} + \sum \underline{u}_i(t) \sum \underline{\dot{u}}_j(t) L_{g_i g_j} \\ + \sum \underline{\dot{u}}_i(t) \sum \underline{u}_j(t) L_{g_i g_j} \end{pmatrix} \\ \frac{d^4}{dt^4}\underline{y}(t) - \begin{pmatrix} \sum \frac{d^3}{dt^3}\underline{u}_i(t) L_{g_i} + 2\sum \underline{\ddot{u}}_i(t) L_{\underline{\dot{x}} g_i} \\ + \sum \underline{\dot{u}}_i(t) L_{\underline{\dot{x}}\underline{\dot{x}} g_i} + \sum \underline{\ddot{u}}_i(t) L_{g_i f} \\ + \sum \underline{\dot{u}}_i(t) L_{\underline{\dot{x}} g_i f} + \sum \underline{\ddot{u}}_i(t) L_{f g_i} \\ + \sum \underline{\dot{u}}_i(t) L_{\underline{\dot{x}} f g_i} + 2\sum \underline{\dot{u}}_i(t) \sum \underline{\dot{u}}_j(t) L_{g_i g_j} \\ + \sum \underline{u}_i(t) \sum \underline{\ddot{u}}_j(t) L_{g_i g_j} \\ + \sum \underline{u}_i(t) \sum \underline{\dot{u}}_j(t) L_{\underline{\dot{x}} g_i g_j} \\ + \sum \underline{\ddot{u}}_i(t) \sum \underline{u}_j(t) L_{g_i g_j} \\ + \sum \underline{\dot{u}}_i(t) \sum \underline{u}_j(t) L_{\underline{\dot{x}} g_i g_j} \\ + \sum \dot{u}_i L_{f f g_i} + \sum \dot{u}_i L_{f g_i f} + \sum \dot{u}_i L_{g_i f f} \\ + \sum \dot{u}_i \sum u_j L_{f g_j g_i} + \sum \dot{u}_i \sum u_j L_{g_j f g_i} \\ + \sum \dot{u}_i \sum u_j L_{g_j g_i f} + \sum u_i \sum \dot{u}_j L_{f g_j g_i} \\ + \sum u_i \sum \dot{u}_j L_{g_j f g_i} + \sum u_i \sum \dot{u}_j L_{g_j g_i f} \\ + \sum \dot{u}_i \sum u_j \sum u_k L_{g_k g_j g_i} \\ + \sum u_i \sum \dot{u}_j \sum u_k L_{g_k g_j g_i} \\ + \sum u_i \sum u_j \sum \dot{u}_k L_{g_k g_j g_i} \end{pmatrix} \end{bmatrix} \tag{3.8}$$

(All of the Lie derivatives of equation 3.8 are with respect to $C \cdot \underline{x}(t)$, thus $L_g = L_g(C \cdot \underline{x}(t)) = g \cdot C$.)

The number of additional terms on the RHS of equation 3.8 grows geometrically as the degree of derivation increases. Multiple input systems with high-rank observability sub-matrices will thus be difficult to express with this equation. Additionally, many of these RHS terms contain explicit references to the state $\underline{x}(t)$. However, it does not seem possible to completely remove these references from an arbitrary nonlinear system. For this reason, this work requires that the system is observable, which guarantees the ability to find expressions for these values.

Our $\mathcal{O}_{\triangle DD}$ derivation is much more complicated when linear sensor equation $C \cdot \underline{x}(t)$ cannot be assumed. When the nonlinear sensor equation $h(\underline{x}(t))$ is used the above mathematics becomes even more complicated and cluttered. However, as observability is already required (see above), this complication is not necessary. For an observable system, the states can always be determined and it is possible to emulate a linear sensor matrix.

3.2.3 Rank of $\mathcal{O}_{\triangle DD}$ and the Number of Independent NLAR Tests

The remaining issue for nonlinear AR is to determine how many elements of the dynamically derived observability vector $\mathcal{O}_{\triangle DD}$ must be taken to generate a minimal span of the model-based error space. The linear rank behaviors of AR are reviewed below to establish a baseline. In Chapter 2 it is shown how the Caley-Hamilton theorem is used in linear AR to guarantee our knowledge of the maximum rank of each observability submatrix (C_j) [5]:

$$Rank\left(C_j(k)\right) = Rank\left(\begin{bmatrix} c_j \\ c_jA \\ c_jA^2 \\ \vdots \\ c_jA^k \end{bmatrix}\right) = \begin{cases} k+1 & k < r_j \\ r_j & k \geq r_j \end{cases} \quad , \quad r_j \leq n+1 \; . \tag{3.9}$$

The rank of each observability sub-matrix is equal to the number of rows in that sub-matrix, up to a certain number of rows. After that, the rank never increases, no matter how many more rows are added. In terms of observability, one could say that there are exactly r_j independent dimensions in the observability sub-space, or that

$rank(C_j) = r_j$. This implies that r_j independent AR tests are needed - one for each dimension of the observability sub-space. (Observations are tested to see if they fall in a r_j -dimensional space. Each test covers one dimension.) What dimensions for Ω and $C_j(k)$ accomplish this?

Each sensor c_j has an associated rank, r_j, resulting in a total of $\sum_{j=1}^{m} r_j$ degrees of freedom in the sensor observation subspaces. The dimensions of the various matrices in the canonical AR equation are shown in equation 3.10,.

$$\Omega^{\left(\left(\sum(r_j+1)\right)-n\right)\times\left(\sum(r_j+1)\right)} \begin{bmatrix} \vdots \\ C_j \\ \vdots \end{bmatrix}^{\left(\sum(r_j+1)\right)\times n} \underline{x}(t)^{n\times 1} = \underline{0}^{\left(\left(\sum r_j+1\right)-n\right)\times 1} . \qquad (3.10)$$

(Recall that m is the number of sensors and n is the number of states.) Note the dimensions of Ω are completely determined from the rank of the complete observability matrix by basic algebraic principles. Also, note equation 3.10 is ill posed when $\left(\sum(r_j+1)\right) - n < 1$. This situation can only occur in an unobservable system, however, so it is not an issue for this work. (In practice, this is simply the situation where so little is known about the system that model-based fault detection is impossible.)

Equation 3.10 determines N_{AR}, the number of AR tests generated by linear AR: $N_{AR} = \sum_{j=1}^{m}(r_j(lin)) + (m-n)$. (The notation $r_j(lin)$ is used to distinguish this rank from the rank of the nonlinear observability submatrices.) As $\sum_{j=1}^{m} r_j(lin)$ degrees of freedom exist in the sensor observation subspaces, that part of the N_{AR} equation makes intuitive sense. However, the purpose of the $m-n$ term is not immediately clear unless other issues are considered.

In their original paper defining AR [5], Chow and Willsky explained this by defining two kinds of analytical redundancy. "Direct redundancy" referred to direct sensor redundancy, the ability to compare the sensor to data with data from other sensors. Each sensor contributes one direct redundancy test in this manner. "Temporal redundancy" is the observability based tests that make up the majority of AR. Thus each sensor contributes $r_j + 1$ degrees of freedom to the AR equation, for a total of $N_{AR} = \sum_{j=1}^{m} (r_j) + m$. However, in order to use the model in the model-based testing the state needs to be determined. This uses n of the degrees of freedom, leaving the listed number of test residuals. When $m = n$, there are equal numbers of sensors and states, so there are no extra direct redundancy tests. When $m > n$, the extra redundant sensors are used to generate extra tests. (A redundant sensor measures the same state or combination of states already directly observed by another sensor or sensors.) When $m < n$, some of the model-based tests must be forgone in order to use the observability to determine the value of states that have not been instrumented.

In the nonlinear case all of the rank and span arguments still apply, except for one: the Cayley-Hamilton theorem does not apply. Without this guarantee of predictable rank behavior there are problems proving a nonlinear observability sub-matrix is full rank. Additionally, if it turns out that nonlinear observability matrices can gain rank in rows after a row that does not contribute to the rank, this potential sparseness in the observability will make choosing the correct degree of $C_j(k)$ quite sensitive. Fortunately, the guarantee provided by Cayley-Hamilton in linear systems also applies to nonlinear systems, as shown below.

Start by defining the following notation:

$$\mathcal{O}(N^*) = span\left\{c_j \underline{x}, L_{k_1} L_{k_2} \dots L_{k_l} c_j \underline{x}\right\}, \tag{3.11}$$

where:

$$j = 1, \dots, m, \;\; l = 1, 2, \dots, N^*, \quad k_i \in span\left\{f, g_1, \dots, g_q\right\}.$$

(Note that N^* and n^* are not complex, the $*$ distinguishes them from the number of states n.)

Definition 3.2 : $\nabla\mathcal{O}(N^*)$**:** *a subset of* $\nabla\mathcal{O}$ *containing Lie derivatives of degree* N^* *or less.*

$Rank\,(\nabla\mathcal{O}(N^*)) = n^*$ and $Rank\,(\underline{x}) = n$ $\square$.

The following will be shown to be true:

Prove: If $Rank\left(\nabla\mathcal{O}(N^*)\right) = n^* < n$, and if $Rank\left(\nabla\mathcal{O}(N^*+1)\right) = n^* < n$, then $Rank\left(\nabla\mathcal{O}(\infty)\right) = Rank\left(\nabla\mathcal{O}\right) = n^* < n$.

Proof: General case for N^*:

$$\begin{aligned}\mathcal{O}\left(N^*\right) &= \left\{c\underline{x}, L_{k_1}c\underline{x}, L_{k_2k_1}c\underline{x}, \ldots, L_{k_{N^*}\ldots k_2k_1}c\underline{x}\right\}, \\ &= \left\{c\underline{x}, c\cdot k_1, \nabla\left(c\cdot k_1\right)\cdot k_2, \ldots, \nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{N^*}\right\},\end{aligned}$$

$$\begin{aligned}\mathcal{O}\left(N^*+1\right) &= \left\{c\underline{x}, L_{k_1}c\underline{x}, L_{k_2k_1}c\underline{x}, \ldots, L_{k_{N^*}\ldots k_2k_1}c\underline{x}, L_{k_{(N^*+1)}\ldots k_2k_1}c\underline{x}\right\}, \\ &= \left\{c\underline{x}, c\cdot k_1, \nabla\left(c\cdot k_1\right)\cdot k_2, \ldots, \nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{N^*},,\right. \\ &\quad \left.\nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{(N^*+1)}\right\}, \\ &Rank\left(\nabla\mathcal{O}\left(N^*\right)\right) = Rank\left(\nabla\mathcal{O}\left(N^*+1\right)\right) = n^*.\end{aligned}$$

This requires that the highest order elements of $\nabla\mathcal{O}(N^*+1)$ do not increase the rank:

$$\begin{aligned}&\Rightarrow span\left\{\nabla\left(\nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{(N^*+1)}\right)\right\} \subseteq \\ &\quad span\left\{c, \nabla\left(c\cdot k_1\right), \nabla\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right), \ldots, \nabla\left(\nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{N^*}\right)\right\}.\end{aligned}$$

Define the following symbols for the elements of $\mathcal{O}(N^*+1)$:

$$\begin{aligned}&\theta_i \in \left\{\left(\nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{(N^*+1)}\right)\right\}, \\ &\psi_i \in \left\{c\underline{x}, \left(c\cdot k_1\right), \left(\nabla\left(c\cdot k_1\right)\cdot k_2\right), \ldots, \left(\nabla\left(\ldots\left(\nabla\left(c\cdot k_1\right)\cdot k_2\right)\ldots\right)\cdot k_{N^*}\right)\right\}, \\ &\Theta_i = \nabla\theta_i,\ \ \Psi_i = \nabla\psi_i.\end{aligned}$$

Then by definition of span write the linear combination:

$$\forall\Theta_i\,\exists\left\{\alpha_i\right\}\in\ S.T.\ \sum_i\alpha_i\Psi_i = \Theta_i.$$

In other words, since the span of Θ_i is a subset of the span of Ψ_i, it is always possible to find a linear combination of Ψ_i to represent any Θ_i.

Now consider $\nabla\mathcal{O}(N^*+2)$: By definition $\nabla\mathcal{O}(N^*+2) = \nabla\left\{\mathcal{O}(N^*+1)\right\} \cup \nabla\left\{L_{k_1}\theta_i\right\} = \nabla\left\{\mathcal{O}(N^*+1)\right\} \cup \left\{L_{k_1}\Theta_i\right\}$.

The new elements of $\mathcal{O}(N^*+2)$ are produced by applying Lie derivatives to the highest order elements of $\mathcal{O}(N^*+1)$. Substitute $\nabla\mathcal{O}(N^*+2) = \nabla\left\{\mathcal{O}(N^*+1)\right\} \cup \left\{L_{k_1}\left(\sum_i\alpha_i\Psi_i\right)\right\}$.

Reorder summation $\nabla\mathcal{O}(N^*+2) = \nabla\left\{\mathcal{O}(N^*+1)\right\} \cup \left\{\sum_i \alpha_i L_{k_1}(\Psi_i)\right\}$.

Note that $L_{k_1}(\Psi_i) \in \{\Psi_i\} \cup \{\Theta_i\}$, by construction. Then $\nabla\mathcal{O}(N^*+2) = \nabla\left\{\mathcal{O}(N^*+1)\right\} \cup \left\{\sum_i \alpha_i \phi\right\}, \quad \phi \in \{\Psi_i\} \cup \{\Theta_i\}$.

However $span\{\Theta_i\} \subseteq span\{\Psi_i\} = span\{\nabla\mathcal{O}(N^*+1)\}$.

So $rank\,(span\{\nabla\mathcal{O}(N^*+1)\}) = rank\,(span\{\nabla\mathcal{O}(N^*+2)\}) = n^*$.

Now, since it is possible to write $\nabla\mathcal{O}(N^*+2)$ in terms of Θ_i and Ψ_i, it is also possible to repeat the linear combination argument on $span\{\nabla\mathcal{O}(N^*+3)\}$ and thus recursively insure that $span\{\mathcal{O}(N^*+\beta)\} = n^*, \;\; \beta = 1, 2, \ldots$. Thus repeated application of Lie derivatives cannot introduce new dimensionality once a Lie derivative one order higher fails to do so. $\square$

This is the only remaining requirement needed to use the intuitive linear argument for the dimensionality of the AR residual space on the nonlinear system, and so $N_{NLAR} = \sum_{j=1}^{m}(r_j(nonlin)) + (m-n)$. As the rank of the nonlinear observabilitiy submatrices are always greater than or equal to the ranks of the linear submatrices $(r_j(nonlin) \geq r_j(lin))$, nonlinear analytical redundancy is guaranteed to generate at least as many independent test residuals as linear AR and will possibly generate more.

As the gradient of $\mathcal{O}$ is not taken in NLAR, the rank of the Ω matrix will be different and NLAR residuals that are not independent will be generated. Precisely $n-1$ redundant equations will be created, as implied by equation 3.12

$$\Omega^{\left(\left(\sum(r_j+1)\right)-1\right)\times\left(\sum(r_j+1)\right)} \begin{bmatrix} \vdots \\ C_j \\ \vdots \end{bmatrix}^{\left(\sum(r_j+1)\right)\times 1} = \underline{0}^{\left(\left(\sum r_j+1\right)-n\right)\times 1}. \tag{3.12}$$

It turns out that eliminating the redundant equations from the valid NLAR residuals is trivial (as the equations themselves are usually trivial), so this is a minor concern.

3.2.4 Assembling the Components

Now we have all the components we need to create rigorous nonlinear analytical redundancy test residuals. The triangular nonlinear observability $\mathcal{O}_{\triangle}$ allows calculation of a left null-space Ω compatible with the nonlinear dynamically derived observability $\mathcal{O}_{\triangle DD}$. The rank arguments in the previous section determine how many rows of $\mathcal{O}_{\triangle DD}$ are required and how many of the resulting tests are independent. The full NLAR algorithm is thus:

1. Determine the triangular nonlinear observability $\mathcal{O}_{\triangle}$ and its left null Ω.
2. Determine the nonlinear dynamically derived observability $\mathcal{O}_{\triangle DD}$.
3. Find the rank $r_j(nonlin)$ of each observability submatrix in the observability "matrix" in $\nabla\mathcal{O}_{\triangle DD}$. Keep $r_j(nonlin) + 1$ rows in each subvector.
4. Apply the NLAR equation : $\Omega\mathcal{O}_{\triangle DD} = R$.
5. Use $N_{NLAR} = \sum\limits_{j=1}^{m} (r_j(nonlin)) + (m - n)$ to determine how many independent tests there are. Delete the redundant tests. $\square$

3.3 NLAR Example

Consider the following nonlinear system:

$$\begin{array}{rl} \underline{\dot{x}}(t) = & \begin{bmatrix} 0 & -x_2 \\ -x_1 & 0 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(t) \\ \underline{y}(t) = & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \underline{x}(t) = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \underline{x}(t) \end{array} . \tag{3.13}$$

The nonlinearity is a result of the states in the control matrix. Where the linear example of equation 2.10 has constants α and β, the nonlinear system has state variables $-x_1$ and $-x_2$. Since the control matrix is multiplied by the state vector, the differential equation described by linear example is a linear combination of state variables and the nonlinear example above contains squared state variable terms. Note however that the linear example is in fact the canonical linearization of the nonlinear example.

Beginning the process of NLAR, the nonlinear observability vector $\mathcal{O}_\triangle$ is:

$$\mathcal{O}_\triangle = \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t))\underline{u}(t) \\ \begin{pmatrix} L_{ff}(C \cdot \underline{x}(t)) + L_{gf}(C \cdot \underline{x}(t))\underline{u}(t) \\ +L_{fg}(C \cdot \underline{x}(t))\underline{u}(t) + L_{gg}(C \cdot \underline{x}(t))\underline{u}^2(t) \end{pmatrix} \\ \vdots \\ \end{bmatrix} \begin{bmatrix} x_1 \\ -x_2^2 \\ 2x_1^2x_2 - 2u_1x_2 \\ x_2 \\ -x_1^2 \\ 2x_2^2x_1 - 2u_2x_1 \end{bmatrix}. \quad (3.14)$$

Note that the rank of each observability sub-vector is 2, so 3 terms in each are kept. Now calculate Ω and $\mathcal{O}_{\triangle DD}$:

$$\Omega\mathcal{O}_\triangle = 0 \Rightarrow \Omega = \begin{bmatrix} x_2^2 & x_1 & 0 & 0 & 0 & 0 \\ 2u_1x_2 - 2x_1^2x_2 & 0 & x_1 & 0 & 0 & 0 \\ -x_2 & 0 & 0 & x_1 & 0 & 0 \\ x_1^2 & 0 & 0 & 0 & x_1 & 0 \\ 2u_2x_1 - 2x_2^2x_1 & 0 & 0 & 0 & 0 & x_1 \end{bmatrix}, \quad (3.15)$$

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f(C \cdot \underline{x}(t)) + L_g(C \cdot \underline{x}(t))\underline{u}(t) \\ \begin{pmatrix} L_{ff}(C \cdot \underline{x}(t)) + L_{gf}(C \cdot \underline{x}(t))\underline{u}(t) \\ +L_{fg}(C \cdot \underline{x}(t))\underline{u}(t) + L_{gg}(C \cdot \underline{x}(t))\underline{u}^2(t) \end{pmatrix} \\ \vdots \\ \end{bmatrix},$$

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} \underline{y}(t) \\ \underline{\dot{y}}(t) \\ \underline{\ddot{y}}(t) \\ \vdots \\ \\ \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ Cg(\underline{x}(t))\underline{\dot{u}}(t) \\ \vdots \\ \\ \end{bmatrix},$$

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} y_1 \\ \dot{y}_1 \\ \ddot{y}_1 - \underline{\dot{u}}(t) \\ y_2 \\ \dot{y}_2 \\ \ddot{y}_2 - \underline{\dot{u}}(t) \end{bmatrix}. \tag{3.16}$$

Then apply the NLAR equation:

$$\Omega\mathcal{O}_{\triangle} = \begin{bmatrix} x_2^2 & x_1 & 0 & 0 & 0 & 0 \\ 2u_1x_2 - 2x_1^2x_2 & 0 & x_1 & 0 & 0 & 0 \\ -x_2 & 0 & 0 & x_1 & 0 & 0 \\ x_1^2 & 0 & 0 & 0 & x_1 & 0 \\ 2u_2x_1 - 2x_2^2x_1 & 0 & 0 & 0 & 0 & x_1 \end{bmatrix} \begin{bmatrix} y_1 \\ \dot{y}_1 \\ \ddot{y}_1 - \underline{\dot{u}}(t) \\ y_2 \\ \dot{y}_2 \\ \ddot{y}_2 - \underline{\dot{u}}(t) \end{bmatrix}. \tag{3.17}$$

The following boxed equations are the resulting tests:

$$\boxed{R_1 = y_2^2 + \dot{y}_1} \tag{3.18}$$

$$\boxed{R_2 = 2u_1y_2 - 2y_1^2y_2 + \ddot{y}_1 - \underline{\dot{u}}(t)} \tag{3.19}$$

$$\boxed{R_3 = -y_2 + y_2} \tag{3.20}$$

$$\boxed{R_4 = y_1^2 + \dot{y}_2} \tag{3.21}$$

$$\boxed{R_5 = 2u_2y_1 - 2y_2^2y_1 + \ddot{y}_2 - \underline{\dot{u}}(t)} \tag{3.22}$$

Note that test R_3 is obviously trivial. However, $N_{NLAR} = \sum_{j=1}^{m}(r_j(nonlin)) + (m-n) = (2+2)+(2-2) = 4$, so it is only expected that four out of the five equations will be independent. Since it is known that the NLAR method will introduce an extra redundant equation for this system, simply discarding this test is the optimal course.

Quite similarly to the linear case in Chapter 2, R_1 and R_4 are clearly the model equations and R_2 and R_5 are their first derivatives, although they draw on both sensors and model equations. Thus NLAR tests both the dynamic behavior of the system and the consistency between the sensors in the same way as AR.

For this example, the NLAR tests represent the same components of the system model as the linear AR tests. However, the NLAR tests rigorously follow the nonlinear model. If linear AR had been used to determine the tests for a linearized version of this system, linearization-based modeling inaccuracies would have resulted. A hybrid method like NNAR would generate the same tests as NLAR but would not provide a rigorous guarantee of covering all of the possible tests without NLAR arguments. Another system with this characteristic is discussed in the next Chapter (4) and a system where NNAR would not apply due to a larger nonlinear observability space is discussed in Chapter 5.

3.3.1 Comparison of the Linear and Nonlinear Techniques

Table 3.2 enumerates and compares the basic differences between linear and nonlinear analytical redundancy. The differences in the system model and sensors show the function-oriented nature of the nonlinear system. The new triangular nonlinear nullspace for NLAR requires a different notation and algebra for the NLAR calculation. These changes in calculation cause the extra redundant tests that appear in NLAR.

Concept	Linear AR	Nonlinear AR
System	$\dot{x} = A\underline{x} + Bu$	$\dot{\underline{x}} = f(\underline{x}) + g(\underline{x}) \cdot \underline{u}$
Sensors	$C\underline{x}$	$C\underline{x}$ ($h(\underline{x})$ possible)
Null Space	$\Omega O = [0]$	$\Omega \mathcal{O}_{\triangle} = [0]$
Observability	O_{DD}	$\mathcal{O}_{\triangle DD}$
Total # of Tests	$\sum(r_j(lin)) + (m-n)$	$\sum(r_j(nonlin)) + (m-1)$
# of Independent Tests	$\sum(r_j(lin)) + (m-n)$	$\sum(r_j(nonlin)) + (m-n)$
# of Redundant Tests	0	$n-1$

Table 3.2 Comparison of linear and nonlinear AR techniques.

Chapter 4

Nonlinear Analytical Redundancy for Hydraulics

The Rosie robot discussed in this chapter was the initial motivatoion for the study of nonlinear analytical redundancy. The nonlinearities introduced by the hydraulic servovalve (briefly discussed in the previous chapter, Figure 3.1, and equation 3.1) are a significant issue when dealing with hydraulic robots and systems [3, 7, 17]. These nonlinearities led us to investigate the application of modified AR fault detection techniques to deal in an ad hoc manner with the nonlinear hydraulic system [25, 26, 30, 33]. Eventually, we left the ad hoc limitations of our early techniques behind with increasingly rigorous notions of nonlinear analytical redundancy.

4.1 Rosie and the Hydraulic Testbed

The Rosie mobile worksystem [2, 6, 18] is an important and interesting example robot that is on the cutting edge of hazardous environment robotics. Rosie, a heavy-duty hydraulic robot designed for nuclear reactor decontamination and dismantlement, was developed and is under further development by RedZone Robotics Inc. and Carnegie Mellon University's Field Robotics Center. The robot has four independently steerable wheels powered by rotary hydraulic motors on a chassis sporting a heavy-duty crane/manipulator. Fault detection for Rosie is interesting, challenging, and important.

Failure modes, effects, and criticality analysis (FMECA) and fault tree based reliability analysis by researchers working with Rosie determined that the hydraulic wheel actuator subsystem was a vital component for the reliability of the mobile platform. A failure of a wheel mechanism might prevent the removal of the chassis from the reactor work site. This led to a project where the goal was to develop effective data analysis procedures (in our case AR-based fault detection) for hydraulic wheel actuators and then implement them on a testbed system located at Foster-Miller Inc., a company with considerable experience with hydraulic systems. The reliability of existing and future robots could then be enhanced by the results of this project.

The system under consideration consists of a rotary hydraulic motor connected to a 3000 PSI hydraulic power supply through a nonlinear hydraulic spool valve. Hydraulic systems are vulnerable to many faults that electrical systems do not experience and are much harder to model, in part due to their inherently nonlinear nature. However, a hydraulic system has considerable advantages as an actuator in a radioactive environment, as such systems are rugged and powerful, and much less likely to produce dangerous sparks than an electrical system. Therefore, it is sensible to use one and simply put some extra time and energy into ensurung the hydraulic system is adequately monitored.

4.1.1 The Rosie Mobile Worksystem

The Rosie Mobile Worksystem is a tele-robotically operated, hydraulically driven robot, which provides locomotion and a four degree-of-freedom heavy manipulator arm which can be equipped with various tools and robot manipulators. Figure 4.1 is a photograph of the Rosie worksystem. As described in the literature [2, 6, 18], the robot consists of two main components or modules. The first module is a locomotor or mobile platform upon which is mounted the second module, a heavy manipulator. The locomotor module supports and transports the manipulator, and supplies it with power and control/communications. The locomotor consists of a central spine, or body core, upon which are attached front and rear drive wheel assemblies, an electronics enclosure, a hydraulic power supply system, a hydraulic enclosure for filters and valving, and a tether system. The locomotor platform is 198 cm wide, 107 cm high and 290 cm long (78 x 42 x 114 in.), supports an overall machine weight of 6,350 kg (14,000 lb.), and has a maximum speed of 0.6 m/s (2.0 ft/s).

Each wheel is powered individually by means of a geared, piston-type hydraulic motor, and is independently steered by means of a rotary actuator above that wheel. The front wheels are mounted on beams that can extend to provide additional stability when the manipulator arm is extended, as shown in Figure 4.2. The rear wheels are mounted on a pivoting beam for steering purposes.

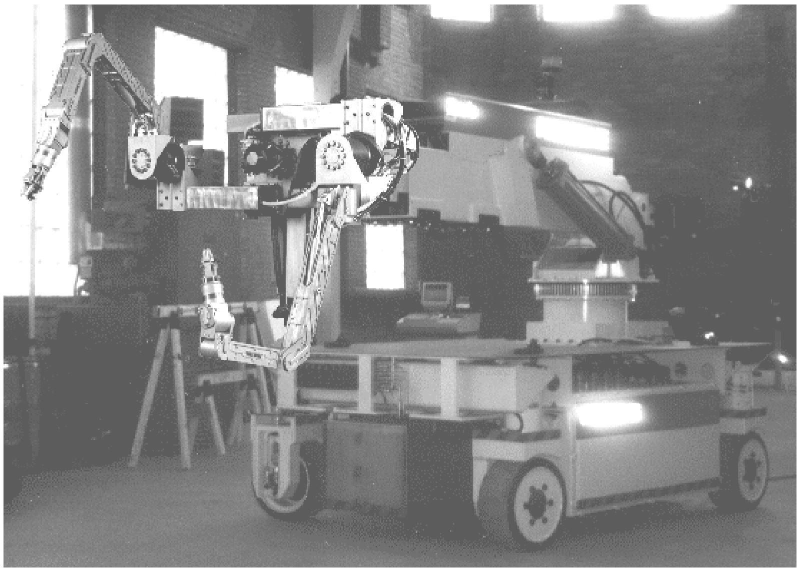


Figure 4.1 The Rosie Mobile Worksystem.

The HPSS consists of a 45 kW (60 hp) supply which provides 114 l/min. (30 GPM) at 20.7 MPa (3,000 psi) for all robot operations. Electrical power and control are provided through a 61 m (200 ft) tether which is wound on a powered reel at the rear of the unit.

The heavy manipulator module supports and positions the tools that actually perform the D&D functions. Shown with a fully-extended boom in Figure 4.2, the heavy

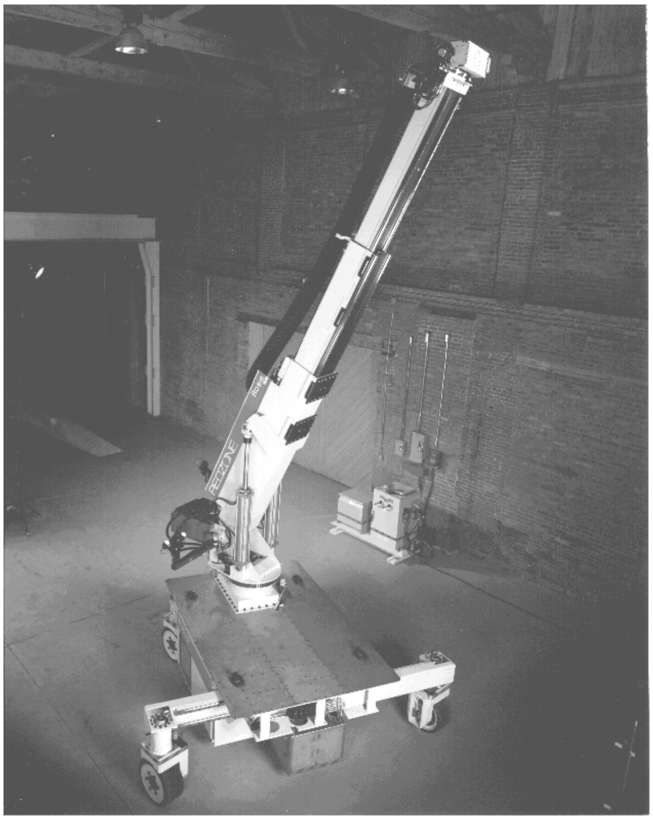


Figure 4.2 Rosie with manipulator arm extended.

manipulator performs the functions of waist rotation, shoulder pitch, outer forearm extension, inner forearm extension, and wrist pitch. The hardware to execute each of these functions is very similar and consists of a flow servovalve, an actuator, and fluid components (tubing, fittings, etc.). The shoulder pitch and forearm extension functions have piston-cylinder actuators and the waist and wrist rotation are achieved through rotary actuators.

4.1.2 The Hydraulic Testbed and Fault Simulation

The testbed was a hydraulic motor controlled by a hydraulic servovalve. (This was the main nonlinear component of the system.) The specific component selected for the test program was the hydraulic wheel motor from Rosie, a Black Bruin Model 404-080-2111 from Valmet Power Transmission Inc. It has a radial piston design and has a maximum power rating of 35 kW (47 hp), a maximum output speed of 185 rpm and can deliver a torque of 2990 Nm at 250 bar (2205 ft-lb at 3600 psi). This motor was capable of driving a wheel directly and therefore accepting a substantial radial load, the exact value of load depending on the axial location of the load with respect to the motor.

The concept for the test rig itself is shown in figures 4.3 and 4.4. A hydraulic motor powered by a HPSS was mounted on a machine bed. The output shaft was loaded radially by means of an adapter and a hydraulic jack assembly. Load was applied to the motor by means of an identical motor, used as a pump. The pump loading device was connected to the motor through a flexible coupling and differed from the motor in not having a hardened shaft and not having the "freewheeling with springs" option. The pump was fed through a separate hydraulic supply consisting of a low-pressure pump, cooler and reservoir. Load was controlled by means of a throttling valve, with a relief valve to prevent overpressure. Table 4.1 shows a listing of the motor-servovalve system faults investigated, along with the planned installation methods.

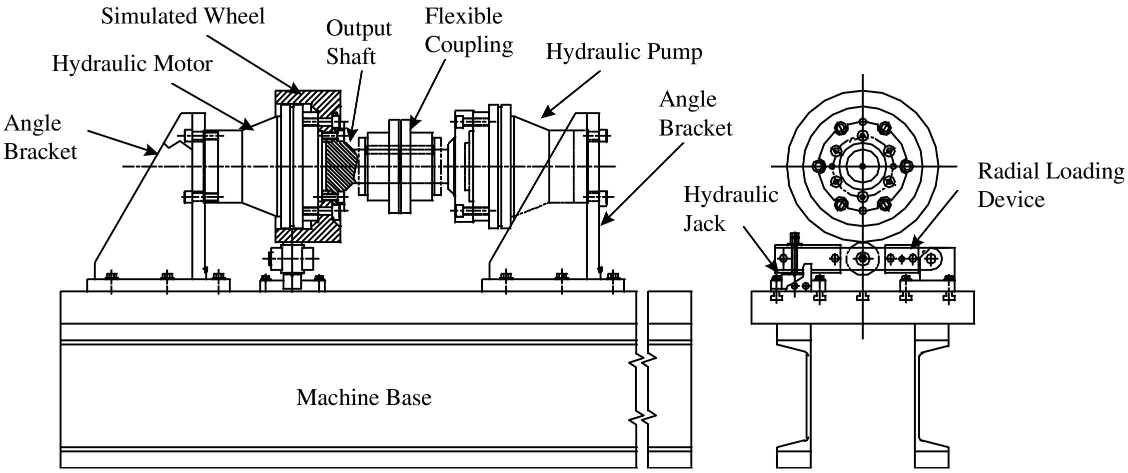


Figure 4.3 Hydraulic test rig schematic.

Problems with the servovalves focused on open windings and sticking of internal valve components. Open winding faults were simulated by inserting a relay in series with the winding. The relay was actuated by means of a bit output of the data acquisition board. This allowed a simple simulation of the fault in software.

A sticking valve was simulated by altering the software control profile for the valve. For example, a new open loop control profile was substituted for the standard PID

Component	Fault	Installation Method
Servovalve	Open winding	Relay
Servovalve	Sticking valve	Change control profile in software
Hydraulic motor	Ruptured line	Tee flow to separate tank
Tachometer	Failed tachometer	Make input zero in controller
Hydraulic motor	Internal leak	External hydraulic short with variable restriction

Table 4.1 Faults for the Foster-Miller hydraulic testbed.

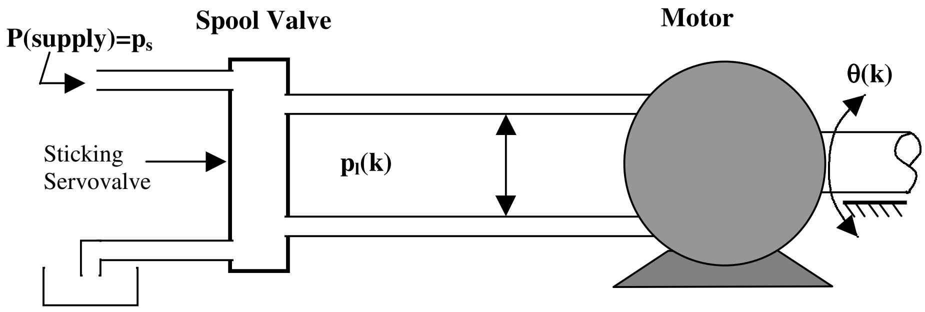


Figure 4.4 Conceptual hydraulic system.

closed loop control algorithm. The new algorithm incorporated stick-slip behavior as needed to simulate the sticking valve.

4.1.3 The Mathematical System Model

Begin by defining the terms used in the model.

Notation:

- A, B, and C are the canonical discrete time state-space system matrices
- B_m is the viscous damping coefficient
- $C_{tm} = c_{em} + c_{im}$ represent total, external, and internal leakage, respectively
- d_m is the volumetric displacement of the motor
- J_t is the inertia of the motor and load
- K_f,k_q and k_c are valve flow coefficients

- $M = k_c + C_{tm}$ is a generalized pressure coefficient
- p_l and $p(k)$ are the (continuous and discrete) pressure drop across the motor
- p_s is the hydraulic power supply; nominal pressure of 3000 PSI
- Q is the net fluid flow into the spool valve
- t is the continuous time variable, k the discrete time variable, Δt is the time step
- T_g is the torque generated by the motor
- T_l is the load torque
- R_1 through R_4 are nonlinear AR tests
- u_v and $u(k)$ are the (continuous and discrete) servovalve positions
- v_t is the volume of fluid within the motor
- $y(k)$ is the state vector
- β_e is the bulk modulus of the hydraulic fluid
- θ_m and $\theta(k)$ are the (continuous and discrete) position of the motor shaft
- ρ is the hydraulic fluid density. □

The following model equations are standard for hydraulic systems [40, 49]:

$$T_g = p_l d_m = J_t \ddot{\theta}_m + B_m \dot{\theta}_m + T_l, \tag{4.1}$$

$$Q = u_v K_f \sqrt{2(p_s - p_l)/\rho} = d_m \dot{\theta}_m + (c_{im} + c_{em}) p_l + \frac{v_t \dot{p}_l}{4\beta_e} \tag{4.2}$$

The state-space control model uses the following state vector:

$$\underline{x} = \begin{bmatrix} \theta \\ \dot{\theta} \\ p_l \end{bmatrix}. \tag{4.3}$$

The second and third state variables are instrumented, but the first (θ) is not. The nonlinear control system formed by these assumptions is as follows:

$$\begin{aligned}
&\dot{\underline{x}} = f(\underline{x}) + g(\underline{x})u \\
&\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{p}_l \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -B_m/J_t & d_m/J_t \\ 0 & -4\beta_e d_m/v_t & -4\beta_e C_{tm}/v_t \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ p_l \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ (4\beta_e K_f/v_t)\sqrt{2(p_s - p_l)/\rho} \end{bmatrix} u_v \\
&\underline{y} = C\underline{x}, \quad C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \underline{y} = \begin{bmatrix} \dot{\theta} \\ p_l \end{bmatrix}.
\end{aligned} \tag{4.4}$$

4.2 Nonlinear Analytical Redundancy for Rosie

Applying our NLAR method to get the grouped observability vector $\mathcal{O}_\triangle$:

$$\mathcal{O}_\triangle = \begin{bmatrix} c_1\underline{x} \\ L_f c_1\underline{x} + L_g c_1\underline{x}u \\ L_{ff} c_1\underline{x} + L_{fg} c_1\underline{x}u + L_{gf} c_1\underline{x}u + L_{gg} c_1\underline{x}u^2 \\ c_2\underline{x} \\ L_f c_2\underline{x} + L_g c_2\underline{x}u \\ L_{ff} c_2\underline{x} + L_{fg} c_2\underline{x}u + L_{gf} c_2\underline{x}u + L_{gg} c_2\underline{x}u^2 \end{bmatrix},$$

$$\mathcal{O}_\triangle = \begin{bmatrix} \dot\theta \\ -B_m\dot\theta/J_t + d_m p_l/J_t \\ \left\{ \begin{array}{l} (B_m^2/J_t^2 + -4\beta_e d_m^2/J_t v_t)\,\dot\theta \\ +(-B_m d_m/J_t^2 + -4\beta_e d_m C_{tm}/J_t v_t)\,p_l \\ +(4\beta_e d_m K_f/J_t v_t)\sqrt{2(p_s-p_l)/\rho}\ u \end{array} \right\} \\ p_l \\ -4\beta_e d_m/v_t\dot\theta + -4\beta_e C_{tm}/v_t p_l + 4\beta_e K_f/v_t\left(\sqrt{2(p_s-p_l)/\rho}\right)\ u \\ \left\{ \begin{array}{l} \left(4B_m\beta_e d_m/J_t v_t + 16\beta_e^2 d_m C_{tm}/v_t^2 + (8\beta_e^2 d_m K_f/v_t^2)\sqrt{2/\rho(p_s-p_l)}\,u\right)\dot\theta \\ +\left(-4\beta_e d_m^2/J_t v_t + 16\beta_e^2 C_{tm}^2/v_t^2 + (8\beta_e^2 K_f C_{tm}/v_t^2)\sqrt{2/\rho(p_s-p_l)}\,u\right)p_l \\ -16\beta_e^2 K_f C_{tm}/v_t^2\left(\sqrt{2(p_s-p_l)/\rho}\right)u + \left(-16\beta_e^2 K_f^2/\rho v_t^2\right)u^2 \end{array} \right\} \end{bmatrix} . \tag{4.5}$$

Note the nonlinear sub-observability matrices $c_i\nabla\mathcal{O}_\triangle$ have no terms containing θ, so the system is rank two and only three terms are needed in each C_j. This also means four independent NLAR tests are expected.

Now calculate the null-matrix Ω:

$$\Omega\mathcal{O}_\triangle = \begin{bmatrix} \Omega_{11} & \dot\theta & 0 & 0 & 0 & 0 \\ \Omega_{21} & 0 & \dot\theta & 0 & 0 & 0 \\ \Omega_{31} & 0 & 0 & \dot\theta & 0 & 0 \\ \Omega_{41} & 0 & 0 & 0 & \dot\theta & 0 \\ \Omega_{51} & 0 & 0 & 0 & 0 & \dot\theta \end{bmatrix}\mathcal{O}_\triangle = 0\ , \tag{4.6}$$

where:

$$\Omega_{11} = \frac{B_m}{J_t}\dot\theta + \frac{-d_m}{J_t}p_l,$$

and

$$\Omega_{12} = \left(\frac{-B_m^2}{J_t^2} + \frac{4\beta_e d_m^2}{J_t v_t}\right)\dot\theta + \left(\frac{B_m d_m}{J_t^2} + \frac{4\beta_e d_m C_{tm}}{J_t v_t}\right)p_l + \frac{-4\beta_e d_m K_f}{J_t v_t}\sqrt{2(p_s-p_l)/\rho}\,u,$$

$$\Omega_{13} = -p_l,$$

$$\Omega_{14} = 4\beta_e d_m/v_t\dot{\theta} + 4\beta_e C_{tm}/v_t p_l + (-4\beta_e K_f/v_t)\sqrt{2(p_s - p_l)/\rho}\ u,$$

$$\begin{aligned}\Omega_{51} = &\Big(-4B_m\beta_e d_m/J_t v_t + -16\beta_e^2 d_m C_{tm}/v_t^2 + (-8\beta_e^2 d_m K_f/v_t^2)\sqrt{2/\rho(p_s - p_l)}\,u\Big)\,\dot{\theta} \\ &+ \Big(4\beta_e d_m^2/J_t v_t + -16\beta_e^2 C_{tm}^2/v_t^2 + (-8\beta_e^2 K_f C_{tm}/v_t^2)\sqrt{2/\rho(p_s - p_l)}\,u\Big)\,p_l \\ &+ (16\beta_e^2 K_f C_{tm}/v_t^2)\sqrt{2(p_s - p_l)/\rho u} + 16\beta_e^2 K_f^2/\rho v_t^2 u^2\end{aligned}.$$

Then calculate $\mathcal{O}_{\triangle DD}$:

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} y_1(t) \\ \dot{y}_1(t) \\ \ddot{y}_1(t) \\ y_2(t) \\ \dot{y}_2(t) \\ \ddot{y}_2(t) - (4\beta_e K_f/v_t)\sqrt{2(p_s - y_2(t))/\rho}\ \dot{u}(t) \end{bmatrix}. \tag{4.7}$$

Then simply apply the NLAR equation:

$$\begin{bmatrix} \Omega_{11} & \dot{\theta} & 0 & 0 & 0 & 0 \\ \Omega_{21} & 0 & \dot{\theta} & 0 & 0 & 0 \\ \Omega_{31} & 0 & 0 & \dot{\theta} & 0 & 0 \\ \Omega_{41} & 0 & 0 & 0 & \dot{\theta} & 0 \\ \Omega_{51} & 0 & 0 & 0 & 0 & \dot{\theta} \end{bmatrix} \begin{bmatrix} y_1(t) \\ \dot{y}_1(t) \\ \ddot{y}_1(t) \\ y_2(t) \\ \dot{y}_2(t) \\ \ddot{y}_2(t) - (4\beta_e K_f/v_t)\sqrt{2(p_s - y_2(t))/\rho}\ \dot{u}(t) \end{bmatrix} = 0. \tag{4.8}$$

to get the NLAR tests.

$$\boxed{R_1 = -\dot{y}_1(t) - \frac{B_m}{J_t}y_1(t) + \frac{d_m}{J_t}y_2(t)} \tag{4.9}$$

$$\boxed{R_2 = -\ddot{y}_1(t) + \left(\frac{B_m^2}{J_t^2} + \frac{-4\beta_e d_m^2}{J_t v_t}\right)y_1(t) + \left(\frac{-B_m d_m}{J_t^2} + \frac{-4\beta_e d_m C_{tm}}{J_t v_t}\right)y_2(t)} \tag{4.10}$$

$$\boxed{R_3 = -y_2(t)y_1(t) + y_1(t)y_2(t)} \tag{4.11}$$

$$\boxed{\begin{aligned} &R_4 = -\dot{y}_2(t) + -4\beta_e d_m y_1(t)/v_t + -4\beta_e C_{tm} y_2(t)/v_t \\ &+ (4\beta_e K_f/v_t)\left(\sqrt{2(p_s - y_2(t))/\rho}\right)\, u \end{aligned}} \tag{4.12}$$

$$\boxed{\begin{aligned} &R_5 = -\ddot{y}_2(t) \\ &+ \left(4B_m\beta_e d_m/J_t v_t + 16\beta_e^2 d_m C_{tm}/v_t^2 + (8\beta_e^2 d_m K_f/v_t^2)\sqrt{2/\rho(p_s - y_2(t))}\ u\right) y_1(t) \\ &+ \left(-4\beta_e d_m^2/J_t v_t + 16\beta_e^2 C_{tm}^2/v_t^2 + (8\beta_e^2 K_f C_{tm}/v_t^2)\sqrt{2/\rho(p_s - y_2(t))}\ u\right) y_2(t) \\ &- (16\beta_e^2 K_f C_{tm}/v_t^2)\sqrt{2/\rho(p_s - y_2(t))}\ u - 16\beta_e^2 K_f^2 u^2/\rho v_t^2 \\ &+ (4\beta_e K_f/v_t)\sqrt{2(p_s - y_2(t))/\rho}\,\dot{u} \end{aligned}} \tag{4.13}$$

Note that R_3 is trivial, so the expected number of independent NLAR tests is generated. R_1 and R_4 correspond to the model equations. R_2 and R_5 correspond to the first derivatives of the model equations.

Additionally, note that since the linear observability O and the nonlinear observability $\mathcal{O}$ span the same spaces for this system, the NNAR method generates test residuals mathematically equivalent to the NLAR test residuals above [28, 30, 31]. This is useful, as the hydraulic testbed data was available before the full NLAR method was. Thus earlier NNAR test residual results need not be recalculated for NLAR. Instead, the NLAR technique can be used to retroactively guarantee that this particular set of NNAR results maintains the span guarantees of analytical redundancy [30].

4.3 Results

Note that these results were generated using the NNAR method. As discussed previously, the tests are in this particular instance equivalent to the NLAR tests. (This is not true in general.) The NNAR notation is somewhat different than the notation system used above for NLAR. $R1 = V1$, $R2 = V2$, $R4 = NV3$, and $R5 = NV4$.

4.3.1 Servovalve, Open Winding Fault

Onset and duration of this fault is clear in all AR tests. The open winding acts like a step input that is not accounted for in the model, provoking a strong AR response.

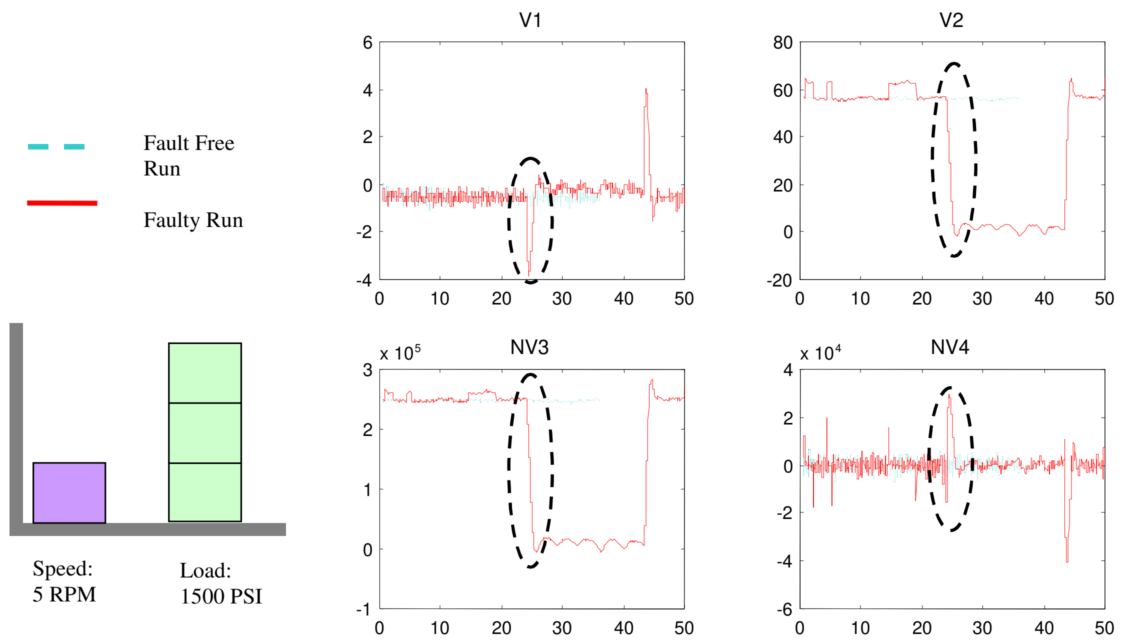


Figure 4.5 Servovalve, open winding fault example run.

4.3.2 Sticking Wheel Motor Servovalve Fault

This fault is evident on all AR tests, although V1 and NV4 show clearest results. During the parts of the run where the system is stuck, it does not follow the dynamic model at all, and is thus easy to detect.

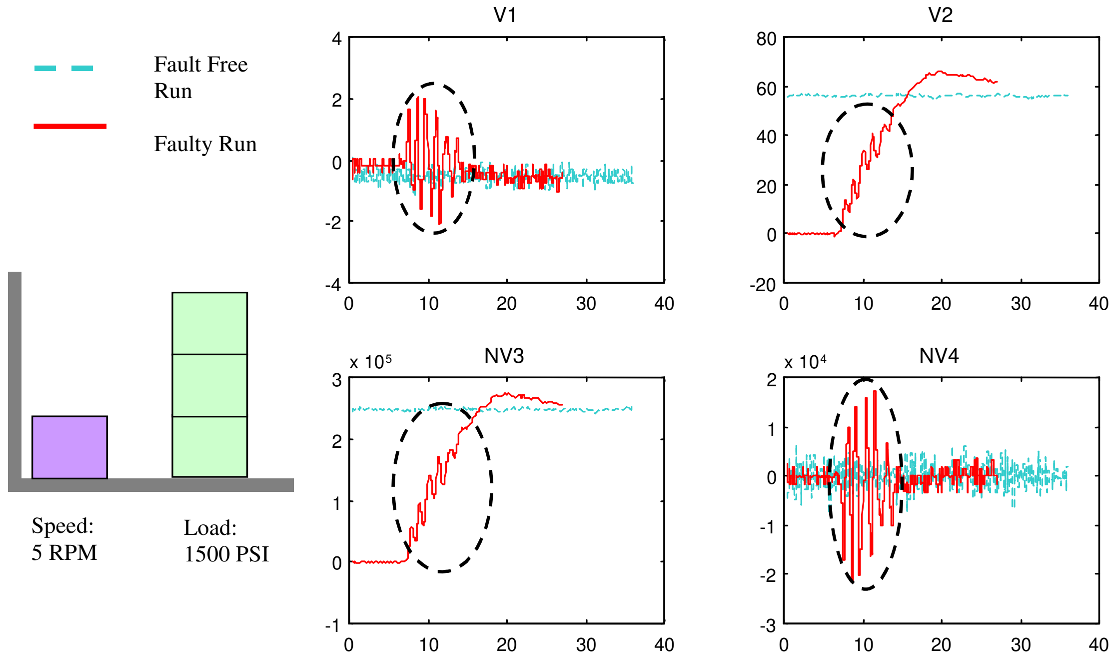


Figure 4.6 Sticking wheel motor servovalve fault example run.

4.3.3 Leak in Motor-Valve System Fault

Our NLAR method does not detect this fault, even for relatively large leaks. (Exactly why is discussed below.) Flow-based AR (AR tests based on flow sensors rather than pressure sensors), if workable, would likely be a better approach to this fault.

4.3.4 Loss of Speed Feedback (Tachometer) Fault

Except for the idling system, AR does well here. (As the idling system is already giving a near-zero value for the zeroed out sensor, this is unsurprising.) If idling, it would alert the user as soon as a motion was attempted. As the failed sensor

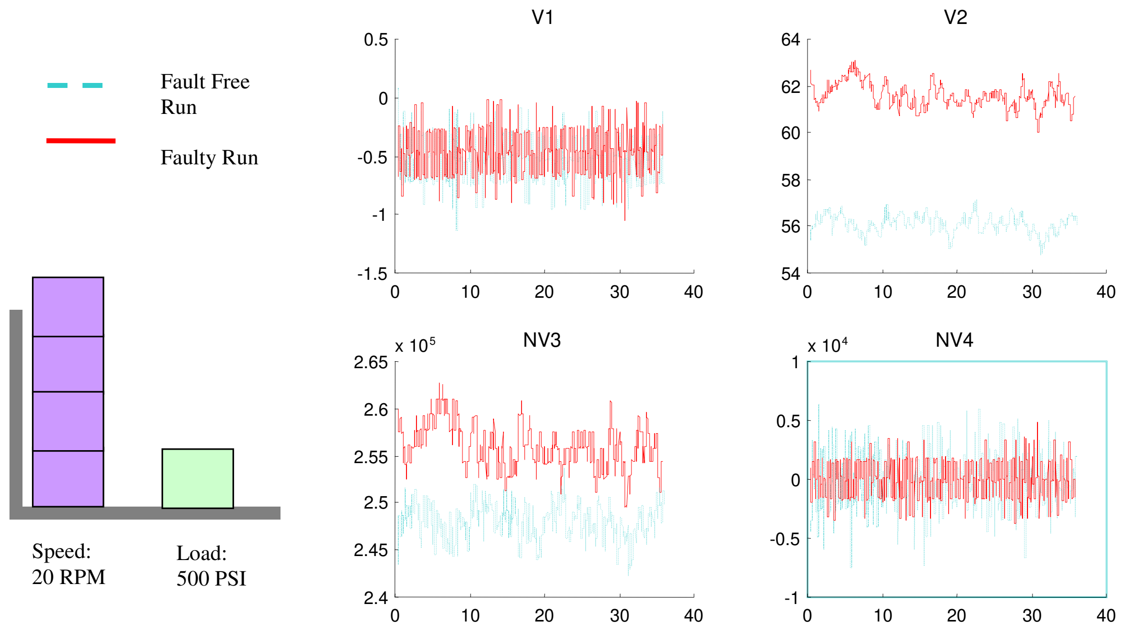


Figure 4.7 Leak in motor-valve system fault.

invalidates the control loop, AR detects this as a deviation from the model-expected behavior.

4.3.5 Motor Internal Leak Fault

The effects of this leak fault are also too small to detect.

4.4 Model Accuracy Issues

Ultimately, the success of the NLAR tests for Rosie was limited by the noise and accuracy of the control model used. It seems that the noise in a powerful hydraulic system such as the testbed is large. Additionally, the system control model was not sufficiently accurate to detect fine changes in the system. Thus NLAR was best

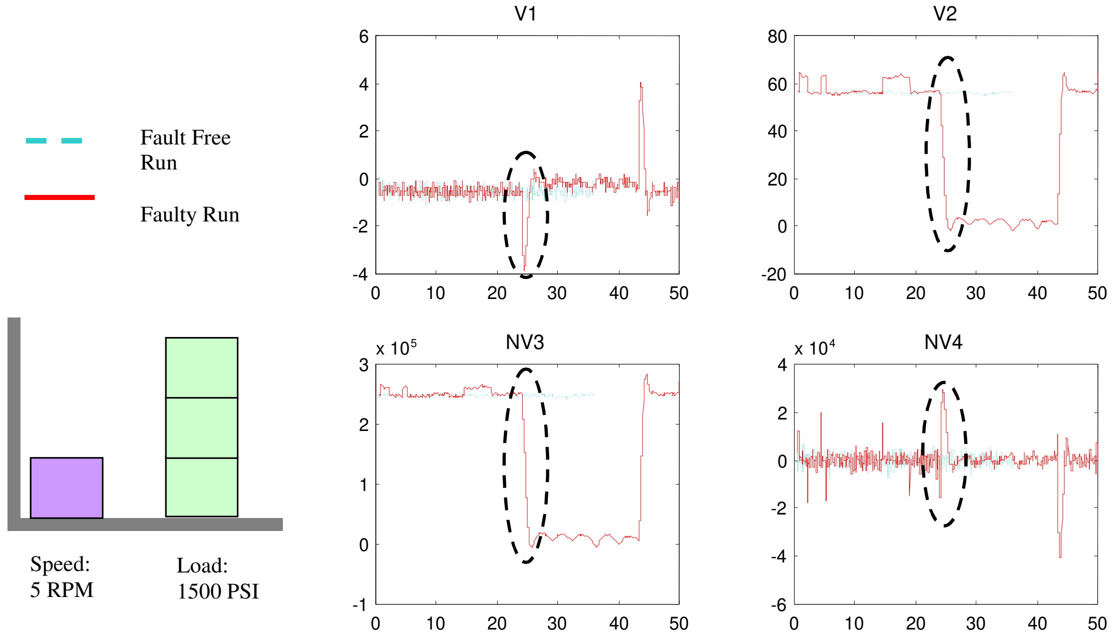


Figure 4.8 Loss of speed feedback (tachometer) fault example run.

at detecting faults that represented very large deviations from the model, but had great difficulty detecting faults such as leaks that had very little effect on the model following behavior of the system.

This is an important lesson about the proper use of AR and NLAR. Only the faults that are severe enough to change the system behavior to a degree significantly greater than the modeling error are detectable. If the system is not well modeled, small and subtle faults will be undetectable.

This is a simple application of the concept of signal to noise ratio. In practice, it makes NLAR most valuable when applied to well modeled systems. As the Rosie system was not sufficiently well modeled, our NLAR test residuals were not exploited to their full potential. In the next chapter, a system that has been modeled exten-

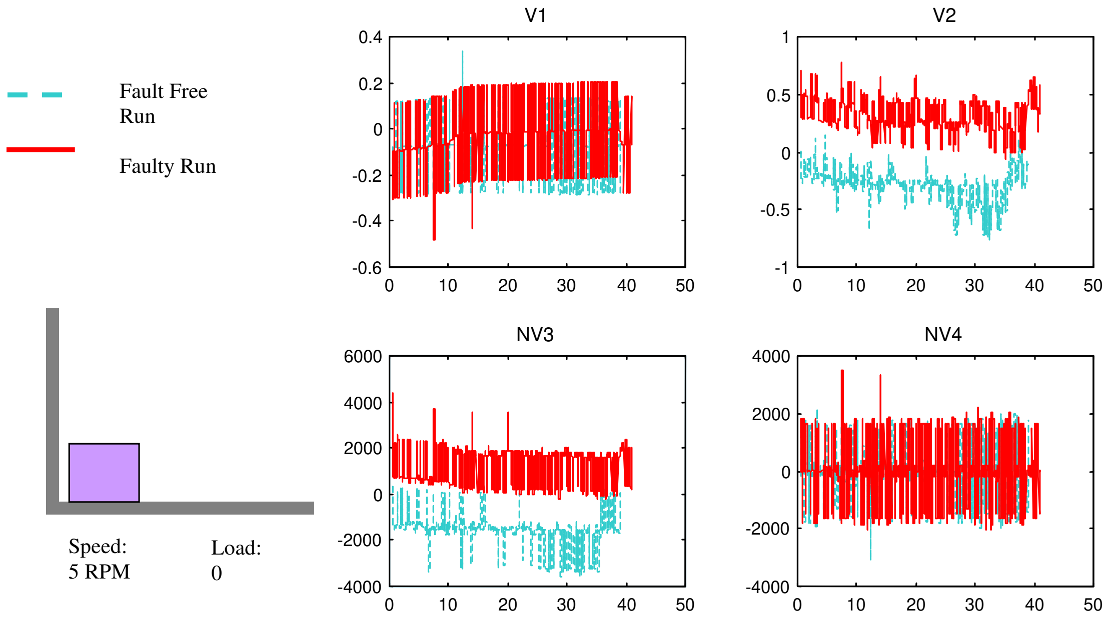


Figure 4.9 Motor internal leak fault example run.

sively is examined via NLAR. The smaller degree of model error will increase the effectiveness of the NLAR technique dramatically.

Chapter 5

Nonlinear Analytical Redundancy in a Planar Robot

5.1 The Integrated Motion Inc. Robot

The Integrated Motion Inc. (IMI) robot at Clemson University as seen in Figure 5.1 is a two-link, revolute, direct drive planar manipulator [13]. The planar two joint manipulator is a canonical platform for testing new robotic techniques, as it has many of the complexities of larger multi-joint systems while remaining relatively compact both physically and mathematically. In the context of NLAR research, this system demonstrates the nonlinear coupling effects needed to test the nonlinear part of NLAR.

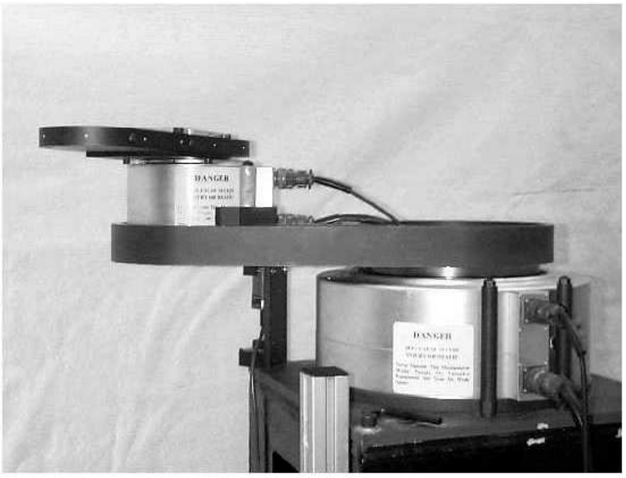


Figure 5.1 The Integrated Motion Inc. Robot.

This robot has been very carefully modeled to facilitate model-based control research, and is thus ideal for investigation of model-based fault detection techniques.

The model seen in equation 5.1 was derived at Clemson [13] for model-based control research.

$$\begin{aligned}
\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &= \begin{bmatrix} p_1 + 2p_3\cos(q_2) & p_2 + p_3\cos(q_2) \\ p_2 + p_3\cos(q_2) & p_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} \\
&+ \begin{bmatrix} -p_3\sin(q_2)\dot{q}_2 & -p_3\sin(q_2)\left(\dot{q}_1 + \dot{q}_2\right) \\ p_3\sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \\
&+ \left(\begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} f_{s1} & 0 \\ 0 & f_{s2} \end{bmatrix} \begin{bmatrix} sign(\dot{q}_1) \\ sign(\dot{q}_2) \end{bmatrix} \right)
\end{aligned} \tag{5.1}$$

Here u_i are the control input torques, q_i are the joint variables, $p_1 = 3.473kg \cdot m^2$, $p_2 = 0.193kg \cdot m^2$, and $p_3 = 0.242kg \cdot m^2$ are rotational inertia terms, $f_{d1} = 1.3N \cdot m \cdot s$ and $f_{d2} = 0.88N \cdot m \cdot s$ model the dynamic (viscous) friction, and $f_{s1} = 1.519N \cdot m$ and $f_{s2} = 0.932N \cdot m$ model the static (Coulomb) friction. The model is in the form of the standard [39] robotic model $M\left(\underline{q}\right)\underline{\ddot{q}} + N(\underline{q}, \underline{\dot{q}}) + \underline{\zeta} = \underline{u}$. Here M contains the inertia terms, N contains the coupling, Coriolis, and dynamic friction effects, while ζ is used for all disturbance terms, including friction

For NLAR the model is reformulated as a state-space control model with the state and output vectors similar to equation 3.2, as seen in equation 5.2. This system is clearly nonlinear and quite complex. The Coulomb friction is not smooth, and will provide behavior in the fault runs that can not be modeled by AR.

$$
\begin{aligned}
\dot{\underline{x}} &= f(\underline{x}) + g_1(\underline{x})u_1 + g_2, \\
f(\underline{x}) &= \begin{bmatrix} f_1(\underline{x}) \\ f_2(\underline{x}) \\ f_3(\underline{x}) \\ f_4(\underline{x}) \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ f_2(q_2, \dot{q}_1, \dot{q}_2) \\ \dot{q}_2 \\ f_4(q_2, \dot{q}_1, \dot{q}_2) \end{bmatrix} = \begin{bmatrix} \dot{q}_1 \\ \alpha\dot{q}_1 + \beta\dot{q}_2 \\ \dot{q}_2 \\ \gamma\dot{q}_1 + \delta\dot{q}_2 \end{bmatrix}, \\
g_1(\underline{x}) &= \begin{bmatrix} 0 \\ -p_2/\varphi(q_2) \\ 0 \\ (p_2 + p_3\cos(q_2))/\varphi(q_2) \end{bmatrix} = \begin{bmatrix} g_{11} \\ g_{12} \\ g_{13} \\ g_{14} \end{bmatrix}, \\
g_2(\underline{x}) &= \begin{bmatrix} 0 \\ (p_2 + p_3\cos(q_2))/\varphi(q_2) \\ 0 \\ (-p_1 - 2p_3\cos(q_2))/\varphi(q_2) \end{bmatrix} = \begin{bmatrix} g_{21} \\ g_{22} \\ g_{23} \\ g_{24} \end{bmatrix}.
\end{aligned}
\tag{5.2}
$$

where:

$$
\begin{aligned}
&\alpha(q_2, \dot{q}_1, \dot{q}_2) = \left(p_3\sin(q_2)\left((p_2 + p_3\cos(q_2))\,\dot{q}_1 + p_2\dot{q}_2\right) - p_2 f_{d1}\right)/\varphi(q_2), \\
&\beta(q_2, \dot{q}_1, \dot{q}_2) = (p_2p_3\sin(q_2)\,(\dot{q}_1 + \dot{q}_2) + (p_2 + p_3\cos(q_2))\, f_{d2})/\varphi(q_2), \\
&\gamma(q_2, \dot{q}_1, \dot{q}_2) = \left(\begin{array}{l} -p_3\sin(q_2)\left((p_1 + 2p_3\cos(q_2))\,\dot{q}_1 + (p_2 + p_3\cos(q_2))\,\dot{q}_2\right) \\ +\,(p_2 + p_3\cos(q_2))\, f_{d1} \end{array} \right)/\varphi(q_2), \\
&\delta(q_2, \dot{q}_1, \dot{q}_2) = (-p_3\sin(q_2)\,(p_2 + p_3\cos(q_2))\,(\dot{q}_1 + \dot{q}_2) - (p_1 + 2p_3\cos(q_2))\, f_{d2})/\varphi(q_2), \\
&\varphi(q_2) = p_1p_2 - p_2^2 - p_3^3\cos^2(q_2).
\end{aligned}
$$

and

$$
\underline{x}(t) = \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix}
$$

is the state vector.

The following analysis assumes the system is fully instrumented with a tachometer and a resolver on each joint:

$$\underline{y} = \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \underline{x} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \underline{x}. \tag{5.3}$$

5.1.1 Robotic Simulation

Time and resource constraints made fault simulation on the physical IMI robot impractical. Instead, the model seen in equation 5.1, derived at Clemson [13] , was used to create a Simulink model of the IMI robot at 1000 Hz. This simulation was used to examine the effect of various simulated faults on the NLAR residuals.

The simulated system was controlled by a PID controller that used only the resolver sensor on each joint. (The redundant tachometers are only used for sensor redundancy in the NLAR tests.) Path tracking was through this standard PID error controller run at 100Hz, one tenth the simulation frequency. This PID controller used the same data (sensor outputs y_i) as the NLAR tests, and provided the control inputs u_i used by the system and NLAR tests. However, the controller was outside of the actual robot simulation and had no direct effect on the model following behavior of the system. Thus it did not need to be considered in the NLAR residual derivation. Analytical redundancy in general is independent of the controller; this can be a useful property.

5.2 Analytical Redundancy for the Simulated Robot

The following sections derive both linear and nonlinear AR tests for the IMI robot. It is worth noting that some of the higher-order NLAR test residuals are too large to depict here and are relegated to Appendix A. These residuals, and large terms associated with them, will be marked as Υ in the equations that follow.

5.2.1 Nonlinear Analytical Redundancy

The first step is to determine $\mathcal{O}$ and Ω:

$$\mathcal{O}_{\triangle} = \begin{bmatrix} x_1 \\ c_1 f + u_1 c_1 g_1 + u_2 c_1 g_2 \\ \Lambda_1 \\ (\Upsilon_1) \\ (\Upsilon_{11}) \\ x_2 \\ c_2 f + u_1 c_2 g_1 + u_2 c_2 g_2 \\ \Lambda_2 \\ (\Upsilon_2) \\ x_3 \\ c_3 f + u_1 c_3 g_1 + u_2 c_3 g_2 \\ \Lambda_3 \\ (\Upsilon_3) \\ x_4 \\ c_4 f + u_1 c_4 g_1 + u_2 c_4 g_2 \\ \Lambda_4 \\ (\Upsilon_4) \end{bmatrix} = \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \\ O_7 \\ O_8 \\ O_9 \\ O_{10} \\ O_{11} \\ O_{12} \\ O_{13} \\ O_{14} \\ O_{15} \\ O_{16} \\ O_{17} \end{bmatrix} \tag{5.4}$$

where:

$$\Lambda_i = \begin{pmatrix} L_f c_i f + u_1 L_{g_1} c_i f + u_2 L_{g_2} c_i f \\ +u_1 L_f c_i g_1 + u_1^2 L_{g_1} c_i g_1 + u_1 u_2 L_{g_2} c_i g_1 \\ +u_2 L_f c_i g_2 + u_1 u_2 L_{g_1} c_i g_2 + u_2^2 L_{g_2} c_i g_2 \end{pmatrix}$$

and the various Υ terms contain third and higher order Lie derivatives and are too large to conveniently fit in this representation. (In fact, some of them are too large to conveniently fit on a single page of text!) The Υ terms will be discussed further in Appendix A.

Only the terms dependent on c_1 make any reference to x_1 - this is because that state does not affect the dynamics of the system, and so can only be observed by the sensor that directly measures it. This makes the associated observability sub-matrix rank four rather than rank three. Since AR preserves one more row in each observability submatrix than rank of that submatrix, this sub-matrix has five rows while the others only have four. The expected number of NLAR tests is $\sum_{j=1}^{m}(r_j) + (m-n) = 11$.

Find the standard null-matrix:

$$\Omega\mathcal{O}_\triangle = 0 \Rightarrow \Omega \begin{bmatrix} O_1 \\ \vdots \\ O_{17} \end{bmatrix} = 0 \Rightarrow \Omega = \begin{bmatrix} \begin{bmatrix} O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \\ O_7 \\ O_8 \\ O_9 \\ O_{10} \\ O_{11} \\ O_{12} \\ O_{13} \\ O_{14} \\ O_{15} \\ O_{16} \\ O_{17} \end{bmatrix}, [\mathrm{I}]^{16\times 16}(-x_1) \end{bmatrix}$$

Where $[\mathrm{I}]^{16\times16}$ is the 16×16 identity matrix. Next, find the alternate observability space using the $\mathcal{O}_{\triangle DD}$ formulation:

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} \underline{y}(t) \\ \underline{\dot{y}}(t) \\ \underline{\ddot{y}}(t) \\ \frac{d^3}{dt^3}\underline{y}(t) \\ \vdots \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ Cg_1\dot{u}_1 + Cg_2\dot{u}_2 \\ (\Upsilon_Y) \\ (\Upsilon_{Y'}) \end{bmatrix} \tag{5.5}$$

Again there are large Υ terms that are discussed in Appendix A.

$$\mathcal{O}_{\triangle DD} = \begin{bmatrix} y_1 \\ \dot{y}_1 \\ \ddot{y}_1 \\ \frac{d^3}{dt^3}y_1 - (\Upsilon_{Y_1}) \\ \frac{d^4}{dt^4}y_1 - (\Upsilon_{Y_{11}}) \\ y_2 \\ \dot{y}_2 \\ \ddot{y}_2 - (c_2g_1\dot{u}_1 + c_2g_2\dot{u}_2) \\ \frac{d^3}{dt^3}y_2 - (\Upsilon_{Y_2}) \\ y_3 \\ \dot{y}_3 \\ \ddot{y}_3 \\ \frac{d^3}{dt^3}y_3 - (\Upsilon_{Y_3}) \\ y_4 \\ \dot{y}_4 \\ \ddot{y}_4 - (c_4g_1\dot{u}_1 + c_4g_2\dot{u}_2) \\ \frac{d^3}{dt^3}y_4 - (\Upsilon_{Y_4}) \end{bmatrix} \tag{5.6}$$

Then use the AR equation:

$$\Omega\mathcal{O}_{\triangle DD} = T.$$

Here a temporary notation of T_i is used, rather than R_i, so that the redundant tests can be eliminated without losing the sequential ordering of test numbers R_i in the results section.

The large tests omitted below (T_3, T_4, T_8, T_{12}, and T_{16}) are discussed in appendix A. The shorter tests (no Υ terms) are as follows:

$$T_1 = +f_1 - \dot{y}_1 \tag{5.7}$$

$$T_2 = \nabla f_1 \cdot f + u_1 \nabla f_1 \cdot g_1 + u_2 \nabla f_1 \cdot g_2 - \ddot{y}_1 \tag{5.8}$$

$$T_5 = x_2 - y_2 \tag{5.9}$$

$$T_6 = c_2 f + u_1 c_2 g_1 + u_2 c_2 g_2 - \dot{y}_2 \tag{5.10}$$

$$T_7 = \begin{pmatrix} L_f c_2 f + u_1 L_{g_1} c_2 f + u_2 L_{g_2} c_2 f \\ +u_1 L_f c_2 g_1 + u_1^2 L_{g_1} c_2 g_1 \\ +u_1 u_2 L_{g_2} c_2 g_1 + u_2 L_f c_2 g_2 \\ +u_1 u_2 L_{g_1} c_2 g_2 + u_2^2 L_{g_2} c_2 g_2 \end{pmatrix} - (\ddot{y}_2 - (c_2 g_1 \dot{u}_1 + c_2 g_2 \dot{u}_2)) \tag{5.11}$$

$$T_9 = x_3 - y_3 \tag{5.12}$$

$$T_{10} = c_3 f + u_1 c_3 g_1 + u_2 c_3 g_2 - \dot{y}_3 \tag{5.13}$$

$$T_{11} = (L_f c_3 f + u_1 L_{g_1} c_3 f + u_2 L_{g_2} c_3 f) - \ddot{y}_3 \tag{5.14}$$

$$T_{13} = x_4 - y_4 \tag{5.15}$$

$$T_{14} = c_4 f + u_1 c_4 g_1 + u_2 c_4 g_2 - \dot{y}_4 \tag{5.16}$$

$$T_{15} = \begin{pmatrix} L_f c_4 f + u_1 L_{g_1} c_4 f + u_2 L_{g_2} c_4 f \\ +u_1 L_f c_4 g_1 + u_1^2 L_{g_1} c_4 g_1 \\ +u_1 u_2 L_{g_2} c_4 g_1 + u_2 L_f c_4 g_2 \\ +u_1 u_2 L_{g_1} c_4 g_2 + u_2^2 L_{g_2} c_4 g_2 \end{pmatrix} - (\ddot{y}_4 - (c_4 g_1 \dot{u}_1 + c_4 g_2 \dot{u}_2)) \tag{5.17}$$

So our tests are as follows:

$$\boxed{T_1 = -f_1 + \dot{y}_1} \tag{5.18}$$

This is the first model equation vs. the shoulder resolver.

$$\boxed{T_2 = f_2 + u_1 g_{12} + u_2 g_{22} - \ddot{y}_1} \tag{5.19}$$

This is the second model equation vs. the shoulder resolver.

$$\boxed{T_5 = -y_2 + y_2} \tag{5.20}$$

This is a trivial equation associated with the first tachometer.

$$\boxed{T_6 = f_2 + u_1 g_{12} + u_2 g_{22} - \dot{y}_2} \tag{5.21}$$

This is the second model equation vs. the shoulder tachometer.

$$\boxed{\begin{aligned} &T_7 = \nabla f_2 \cdot f + u_1 \nabla f_2 \cdot g_1 + u_2 \nabla f_2 \cdot g_2 + u_1 \nabla g_{12} \cdot f + u_1^2 \nabla g_{12} \cdot g_1 + u_1 u_2 \nabla g_{12} \cdot g_2 \\ &+u_2 \nabla g_{22} \cdot f + u_1 u_2 \nabla g_{22} \cdot g_1 + u_2^2 \nabla g_{22} \cdot g_2 + \dot{u}_1 g_{12} + \dot{u}_2 g_{22} - \ddot{y}_2 \end{aligned}} \tag{5.22}$$

Further examination reveals equation 5.22 to be the derivative of the second model equation, with some normal NLAR substitutions from other model equations. It uses the tachometer as the primary sensor.

$$\boxed{T_9 = -y_3 + y_3} \tag{5.23}$$

This is a trivial equation for the elbow resolver.

$$\boxed{T_{10} = -f_3 + \dot{y}_3} \tag{5.24}$$

This is the third model equation vs. the elbow resolver.

$$\boxed{T_{11} = f_4 + u_1 g_{14} + u_2 g_{24} - \ddot{y}_3} \tag{5.25}$$

This is the fourth model equation vs. the elbow resolver.

$$\boxed{T_{13} = -y_4 + y_4} \tag{5.26}$$

This is the trivial test for the elbow tachometer.

$$\boxed{T_{14} = f_4 + u_1 g_{14} + u_2 g_{24} - \dot{y}_4} \tag{5.27}$$

This is the fourth model equation vs. the elbow tachometer.

$$\boxed{\begin{aligned} T_{15} = {} & \nabla f_4 \cdot f + u_1 \nabla f_4 \cdot g_1 + u_2 \nabla f_4 \cdot g_2 + u_1 \nabla g_{14} \cdot f + u_1^2 \nabla g_{14} \cdot g_1 + u_1 u_2 \nabla g_{14} \cdot g_2 \\ & + u_2 \nabla g_{24} \cdot f + u_1 u_2 \nabla g_{24} \cdot g_1 + u_2^2 \nabla g_{24} \cdot g_2 + \dot{u}_1 g_{14} + \dot{u}_2 g_{24} - \ddot{y}_4 \end{aligned}} \tag{5.28}$$

Further examination reveals equation 5.28 to be the derivative of the fourth model equation, with some normal NLAR substitutions from other model equations. Uses the tachometer as the primary sensor.

Note that the second and fourth model equations appear multiple times, but the tests use different sensors. This is equivalent to testing the sensors against each other (the intuitively obvious tachometer vs. resolver tests) but less efficient than comparing the sensors directly. However, the exact formulation of the tests is not unique, because the Ω in the AR equation is not unique. In practice, the choice of Ω made above makes computation of the AR tests much easier, so these comparisons will be developed after the calculation of the AR tests, as seen in the following example tests:

$$T_2 - T_6 = \ddot{y}_1 - \dot{y}_2\,, \tag{5.29}$$

$$T_{11} - T_{14} = \ddot{y}_3 - \dot{y}_4. \tag{5.30}$$

It turns out similar manipulations are possible with the larger tests in Appendix A as well. For the final data analyses this was in fact done for every duplicated test. The resulting tests, relabeled in sequential R_i fashion and described as functions of the model equations, are shown in Table 5.1.

Test Label	Description	T_i Equation
R_1	Second model equation checked against shoulder resolver. Tests the acceleration of the system.	T_1
R_2	Derivative of second model equation checked against shoulder resolver. Tests the jerk of the system.	T_2
R_3	Second derivative of second model equation checked against shoulder resolver. Tests the derivative of the jerk.	T_3
R_4	Sensor comparison of shoulder tachometer with derivative of shoulder resolver.	$T_2 - T_6$
R_5	Derivative of R_4.	$T_3 - T_7$
R_6	Second derivative of R_4.	$T_4 - T_8$
R_7	Fourth model equation checked against elbow resolver. Tests the acceleration of the system.	T_{10}
R_8	Derivative of fourth model equation checked against elbow resolver. Tests the jerk of the system.	T_{11}
R_9	Sensor comparison of elbow tachometer and derivative of elbow resolver.	$T_{11} - T_{14}$
R_{10}	Derivative of R_9.	$T_{12} - T_{15}$
R_{11}	Second derivative of fourth model equation checked against elbow tachometer. Tests the derivative of the jerk.	T_{16}

Table 5.1 NLAR tests for IMI test robot.

5.2.2 Complexity of Nonlinear Tests

The derivation in the previous section skips over the mathematical details for five out of the eleven valid tests because of mathematical complexity. It is a valid concern that this complexity might be a limiting factor in the use of nonlinear analytical redundancy.

The full equations in appendix A are quite large and complex. They represent the results of a geometrically growing number of sums (the number of terms in the summation in the ith row of $\mathcal{O}_\triangle$ is roughly $(q+1)^{i-1}$) of increasingly recursed Lie derivatives of multivariable nonlinear fractions. Test residuals of such complexity are not suitable for calculation by hand, even though they need only be calculated once to determine the NLAR test residuals.

However, the mechanical parts of the NLAR derivation for these tests can be, and was, automated. Computers are quite capable of doing repeated mechanical symbolic manipulations of this order. In fact, all of the test residuals in this section were derived in Matlab [35] as shown in Appendix A. Since this sort of symbolic operation reduces the possibility of human computational error affecting the NLAR results, it is a good idea and makes the derivation of complex NLAR tests practical.

Once determined, the NLAR test residuals were passed directly to Matlab's numeric solver and applied to the data. (In a real time implementation of the tests, a more efficient calculation software would be preferred, of course.) Although complex to initially calculate and large when written out, the NLAR tests in Appendix A are not of excessive computational complexity. Mechanical calculation of the compiled residuals at 10 ms intervals will not be difficult for a modern computer.

5.2.3 Linear Analytical Redundancy

Start with the original nonlinear system model:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} p_1 + 2p_3\cos(q_2) & p_2 + p_3\cos(q_2) \\ p_2 + p_3\cos(q_2) & p_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$
$$+ \begin{bmatrix} -p_3\sin(q_2)\dot{q}_2 & -p_3\sin(q_2)\left(\dot{q}_1 + \dot{q}_2\right) \\ p_3\sin(q_2)\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \qquad ,$$
$$+ \left(\begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} f_{s1} & 0 \\ 0 & f_{s2} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \right)$$

where $p_1 = 3.473kg \cdot m^2$, $p_2 = 0.193kg \cdot m^2$, $p_3 = 0.242kg \cdot m^2$, $f_{d1} = 1.3N \cdot m \cdot s$, $f_{d2} = 0.88N \cdot m \cdot s$, and $f_{s1} = 1.519N \cdot m$, $f_{s2} = 0.932N \cdot m$ [13].

Linearizing about $q_i = \dot{q}_i = 0$, ($\Rightarrow \cos(q_i) = 1, \sin(q_i) = q_i$) and neglecting Coulomb friction we get:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} p_1 + 2p_3 & p_2 + p_3 \\ p_2 + p_3 & p_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \Rightarrow \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} =$$
$$\left\{ (1/(-p_2p_1 + p_2^2 + p_3^2)) \begin{bmatrix} -p_2 & p_2 + p_3 \\ p_2 + p_3 & -p_1 - 2p_3 \end{bmatrix} \right\} \left\{ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \right\} .$$

Solving and substituting in the proper constants:

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} -0.4367 & 0.6663 \\ 0.9844 & -6.0615 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} 0.3360 & -0.7572 \\ -0.7572 & 6.8880 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} . \qquad (5.31)$$

It is interesting to note that a zero A matrix results if friction is neglected. Since this is one of the more difficult to measure physical terms, linear AR for the IMI robot has to deal with extra difficult model accuracy problems.

The sensors include resolvers, which measure q directly, so it is necessary to include the trivial derivative equations. This suggests the following control system:

$$\dot{\underline{x}} = A\underline{x} + \sum Bu \Leftrightarrow$$

$$\begin{bmatrix} \dot{q}_1 \\ \ddot{q}_1 \\ \dot{q}_2 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & 0 & A_{44} \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} \dot{\theta} + \begin{bmatrix} 0 \\ B_{21} \\ 0 \\ B_{41} \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ B_{22} \\ 0 \\ B_{42} \end{bmatrix} u_2 \tag{5.32}$$

where $A_{22} = -0.4367$, $A_{24} = 0.6663$, $A_{42} = 0.9844$, $A_{44} = -6.0615$, $B_{21} = 0.3360$, $B_{41} = -0.7572$, $B_{22} = -0.7572$, $B_{42} = 6.8880$ and

$$\underline{x}(t) = \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix}$$

is the state vector. The following observation vector uses the same sensors as the nonlinear case:

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \\ \underline{c}_2 \\ \underline{c}_3 \\ \underline{c}_4 \end{bmatrix} \begin{bmatrix} q_1 \\ \dot{q}_1 \\ q_2 \\ \dot{q}_2 \end{bmatrix}.$$

To create an observation matrix, A^2 and A^3 are needed:

$$A^2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & 0 & A_{44} \end{bmatrix}^2 = \begin{bmatrix} 0 & A_{22} & 0 & A_{24} \\ 0 & A_{22}^2 + A_{22}A_{42} & 0 & A_{22}A_{24} + A_{24}A_{44} \\ 0 & A_{42} & 0 & A_{44} \\ 0 & A_{22}A_{42} + A_{24}A_{42} & 0 & A_{44}^2 + A_{22}A_{24} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -0.437 & 0 & 0.666 \\ 0 & 0.847 & 0 & -4.32 \\ 0 & 0.984 & 0 & -6.06 \\ 0 & -6.40 & 0 & 37.4 \end{bmatrix} = \begin{bmatrix} 0 & A2_{12} & 0 & A2_{14} \\ 0 & A2_{22} & 0 & A2_{24} \\ 0 & A2_{32} & 0 & A2_{34} \\ 0 & A2_{42} & 0 & A2_{44} \end{bmatrix},$$

$$A^3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & 0 & A_{44} \end{bmatrix}^3$$

$$= \begin{bmatrix} 0 & 0.847 & 0 & -4.33 \\ 0 & -4.63 & 0 & 26.8 \\ 0 & -6.40 & 0 & 37.4 \\ 0 & 39.6 & 0 & -231 \end{bmatrix} = \begin{bmatrix} 0 & A3_{12} & 0 & A3_{14} \\ 0 & A3_{22} & 0 & A3_{24} \\ 0 & A3_{32} & 0 & A3_{34} \\ 0 & A3_{42} & 0 & A3_{44} \end{bmatrix}.$$

So

$$O = \begin{bmatrix} c_i \\ c_iA \\ c_iA^2 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & A2_{12} & 0 & A2_{14} \\ 0 & A3_{12} & 0 & A3_{14} \\ 0 & 1 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} \\ 0 & A2_{22} & 0 & A2_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & A2_{32} & 0 & A2_{34} \\ 0 & A3_{32} & 0 & A3_{34} \\ 0 & 0 & 0 & 1 \\ 0 & A_{42} & 0 & A_{44} \\ 0 & A2_{42} & 0 & A2_{44} \end{bmatrix}, \; O\underline{x} = \begin{bmatrix} q_1 \\ \dot{q}_1 \\ A2_{12}\dot{q}_1 + A2_{14}\dot{q}_2 \\ A3_{12}\dot{q}_1 + A3_{14}\dot{q}_2 \\ \dot{q}_1 \\ A_{22}\dot{q}_1 + A_{24}\dot{q}_2 \\ A2_{22}\dot{q}_1 + A2_{24}\dot{q}_2 \\ q_2 \\ \dot{q}_2 \\ A2_{32}\dot{q}_1 + A2_{34}\dot{q}_2 \\ A3_{32}\dot{q}_1 + A3_{34}\dot{q}_2 \\ \dot{q}_2 \\ A_{42}\dot{q}_1 + A_{44}\dot{q}_2 \\ A2_{42}\dot{q}_1 + A2_{44}\dot{q}_2 \end{bmatrix} \quad (5.33)$$

The linear system is clearly observable. Since there are two sub-matrices of rank 3 and two of rank 2, $\sum(rank + 1) = 14$ ranks of the observability matrix O are needed when calculating the null-matrix:

$$\Omega O\underline{x} = 0 \Rightarrow \Omega \begin{bmatrix} q_1 \\ \dot{q}_1 \\ A2_{12}\dot{q}_1 + A2_{14}\dot{q}_2 \\ A3_{12}\dot{q}_1 + A3_{14}\dot{q}_2 \\ \dot{q}_1 \\ A_{22}\dot{q}_1 + A_{24}\dot{q}_2 \\ A2_{22}\dot{q}_1 + A2_{24}\dot{q}_2 \\ q_2 \\ \dot{q}_2 \\ A2_{32}\dot{q}_1 + A2_{34}\dot{q}_2 \\ A3_{32}\dot{q}_1 + A3_{34}\dot{q}_2 \\ \dot{q}_2 \\ A_{42}\dot{q}_1 + A_{44}\dot{q}_2 \\ A2_{42}\dot{q}_1 + A2_{44}\dot{q}_2 \end{bmatrix} = 0,$$

$$\Omega = \left[\begin{bmatrix} \dot{q}_1 \\ A2_{12}\dot{q}_1 + A2_{14}\dot{q}_2 \\ A3_{12}\dot{q}_1 + A3_{14}\dot{q}_2 \\ \dot{q}_1 \\ A_{22}\dot{q}_1 + A_{24}\dot{q}_2 \\ A2_{22}\dot{q}_1 + A2_{24}\dot{q}_2 \\ q_2 \\ \dot{q}_2 \\ A2_{32}\dot{q}_1 + A2_{34}\dot{q}_2 \\ A3_{32}\dot{q}_1 + A3_{34}\dot{q}_2 \\ \dot{q}_2 \\ A_{42}\dot{q}_1 + A_{44}\dot{q}_2 \\ A2_{42}\dot{q}_1 + A2_{44}\dot{q}_2 \end{bmatrix}, [\mathrm{I}]^{13\times 13}(-q_1) \right]$$

Now calculate the linear dynamically derivable observability O_{DD} :

$$O_{DD}(c_1) = \begin{bmatrix} \underline{c}_1 \\ \underline{c}_1 A \\ \underline{c}_1 A^2 \\ \vdots \\ \underline{c}_1 A^n \end{bmatrix} \underline{x}$$

$$= \begin{bmatrix} y_1 \\ \dot{y}_1 \\ \ddot{y}_1 \\ \vdots \\ \frac{d^n}{dt^n} y_1 \end{bmatrix} - \begin{bmatrix} 0 & 0 \quad 0 \quad \dots & 0 \\ \underline{c}_1 B & 0 \quad 0 \quad \dots & 0 \\ \underline{c}_1 AB & \underline{c}_1 B \quad 0 \quad \dots & 0 \\ \vdots & \vdots & \vdots \\ \underline{c}_1 A^{n-1} B & \dots \ \underline{c}_1 A^{n-i} B \ \dots & \underline{c}_1 B \end{bmatrix} \begin{bmatrix} \underline{u} \\ \dot{\underline{u}} \\ \vdots \\ \frac{d^n}{dt^n} \underline{u} \end{bmatrix}$$

$$O_{DD}\underline{x}(t) =$$

$$\begin{bmatrix} y_1 \\ \dot{y}_1 \\ \ddot{y}_1 - B_{21}u_1 - B_{22}u_2 \\ \frac{d^3}{dt^3}y_1 - \left(A2_{12}B_{21} + A2_{14}B_{41}\right)u_1 - \left(A2_{12}B_{22} + A2_{14}B_{42}\right)u_2 - B_{21}\dot{u}_1 - B_{22}\dot{u}_2 \\ y_2 \\ \dot{y}_2 - B_{21}u_1 - B_{22}u_2 \\ \ddot{y}_2 - \left(A_{22}B_{21} + A_{24}B_{41}\right)u_1 - \left(A_{22}B_{22} + A_{24}B_{42}\right)u_2 - B_{21}\dot{u}_1 - B_{22}\dot{u}_2 \\ y_3 \\ \dot{y}_3 \\ \ddot{y}_3 - B_{41}u_1 - B_{42}u_2 \\ \frac{d^3}{dt^3}y_3 - \left(A2_{32}B_{21} + A2_{34}B_{41}\right)u_1 - \left(A2_{32}B_{22} + A2_{34}B_{42}\right)u_2 - B_{41}\dot{u}_1 - B_{42}\dot{u}_2 \\ y_4 \\ \dot{y}_4 - B_{41}u_1 - B_{42}u_2 \\ \ddot{y}_4 - \left(A_{42}B_{21} + A_{44}B_{41}\right)u_1 - \left(A_{42}B_{22} + A_{44}B_{42}\right)u_2 - B_{41}\dot{u}_1 - B_{42}\dot{u}_2 \end{bmatrix} \tag{5.34}$$

Now take the basic AR equation and solve to get the tests:

$$\Omega O_{DD}\underline{x} = T,$$

$$\begin{bmatrix} \dot{q}_1 y_1 - \dot{y}_1 q_1 \\ \left(A2_{12}\dot{q}_1 + A2_{14}\dot{q}_2\right) y_1 - \left(\ddot{y}_1 - B_{21}u_1 - B_{22}u_2\right) q_1 \\ \left(A3_{12}\dot{q}_1 + A3_{14}\dot{q}_2\right) y_1 - \left(\frac{d^3}{dt^3} y_1 + \begin{Bmatrix} -\left(A2_{12}B_{21} + A2_{14}B_{41}\right) u_1 \\ -\left(A2_{12}B_{22} + A2_{14}B_{42}\right) u_2 \\ -B_{21}\dot{u}_1 - B_{22}\dot{u}_2 \end{Bmatrix}\right) q_1 \\ \left(\dot{q}_1\right) y_1 - \left(y_2\right) q_1 \\ \left(A_{22}\dot{q}_1 + A_{24}\dot{q}_2\right) y_1 - \left(\dot{y}_2 - B_{21}u_1 - B_{22}u_2\right) q_1 \\ \left(A2_{22}\dot{q}_1 + A2_{24}\dot{q}_2\right) y_1 - \left(\ddot{y}_2 + \begin{Bmatrix} -\left(A_{22}B_{21} + A_{24}B_{41}\right) u_1 \\ -\left(A_{22}B_{22} + A_{24}B_{42}\right) u_2 \\ -B_{21}\dot{u}_1 - B_{22}\dot{u}_2 \end{Bmatrix}\right) q_1 \\ \left(q_2\right) y_1 - \left(y_3\right) q_1 \\ \left(\dot{q}_2\right) y_1 - \left(\dot{y}_3\right) q_1 \\ \left(A2_{32}\dot{q}_1 + A2_{34}\dot{q}_2\right) y_1 - \left(\ddot{y}_3 - B_{41}u_1 - B_{42}u_2\right) q_1 \\ \left(A3_{32}\dot{q}_1 + A3_{34}\dot{q}_2\right) y_1 - \left(\frac{d^3}{dt^3} y_3 + \begin{Bmatrix} -\left(A2_{32}B_{21} + A2_{34}B_{41}\right) u_1 \\ -\left(A2_{32}B_{22} + A2_{34}B_{42}\right) u_2 \\ -B_{41}\dot{u}_1 - B_{42}\dot{u}_2 \end{Bmatrix}\right) q_1 \\ \left(\dot{q}_2\right) y_1 - \left(y_4\right) q_1 \\ \left(A_{42}\dot{q}_1 + A_{44}\dot{q}_2\right) y_1 - \left(\dot{y}_4 - B_{41}u_1 - B_{42}u_2\right) q_1 \\ \left(A2_{42}\dot{q}_1 + A2_{44}\dot{q}_2\right) y_1 - \left(\ddot{y}_4 + \begin{Bmatrix} -\left(A_{42}B_{21} + A_{44}B_{41}\right) u_1 \\ -\left(A_{42}B_{22} + A_{44}B_{42}\right) u_2 \\ -B_{41}\dot{u}_1 - B_{42}\dot{u}_2 \end{Bmatrix}\right) q_1 \end{bmatrix} = T.$$

The tests (except for the expected trivial equations) are the model equations and their various derivatives. Five trivial/redundant equations are expected from this series; they are discussed below.

$$\boxed{LT_1 = y_2 - \dot{y}_1} \tag{5.35}$$

This is a tachometer vs. resolver test based on the first model equation. Since $LT_2 - LT_5$ essentially gives this test, it is classified as redundant. (This is a matter of choice.)

$$\boxed{LT_2 = A_{22}y_2 + A_{24}y_4 + B_{21}u_1 + B_{22}u_2 - \ddot{y}_1} \tag{5.36}$$

This is the first model equation checked against the resolver.

$$\boxed{\begin{array}{c} LT_3 = A3_{12}y_2 + A3_{14}y_4 + \left(A2_{12}B_{21} + A2_{14}B_{41}\right)u_1 \\ + \left(A2_{12}B_{22} + A2_{14}B_{42}\right)u_2 + B_{21}\dot{u}_1 + B_{22}\dot{u}_2 - \frac{d^3}{dt^3}y_1 \end{array}} \tag{5.37}$$

This is the derivative of LT_2.

$$\boxed{LT_4 = y_2 - y_2} \tag{5.38}$$

This is a trivial test.

$$\boxed{LT_5 = A_{22}y_2 + A_{24}y_4 + B_{21}u_1 + B_{22}u_2 - \dot{y}_2} \tag{5.39}$$

This is the first model equation checked against the tachometer. It can be combined with LT_2 to create a sensor comparison.

$$\boxed{\begin{array}{c} LT_6 = A2_{22}y_2 + A2_{24}y_4 + \left(A_{22}B_{21} + A_{24}B_{41}\right)u_1 \\ + \left(A_{22}B_{22} + A_{24}B_{42}\right)u_2 + B_{21}\dot{u}_1 + B_{22}\dot{u}_2 - \ddot{y}_2 \end{array}} \tag{5.40}$$

This is the derivative of LT_5. It can be turned into the derivative of a sensor comparison in a similar manner.

$$\boxed{LT_7 = y_4 - y_4} \tag{5.41}$$

This is a trivial test.

$$\boxed{LT_8 = y_4 - \dot{y}_3} \tag{5.42}$$

This is a tachometer vs. resolver test based on the second model equation. Since $LT_9 - LT_12$ gives this test, we classify it as redundant.

$$\boxed{LT_9 = A_{42}y_2 + A_{44}y_4 + B_{41}u_1 + B_{42}u_2 - \ddot{y}_3} \tag{5.43}$$

This is the second model equation tested against the resolver.

$$\boxed{\begin{gathered} LT_{10} = A3_{32}y_2 + A3_{34}y_4 + \left(A2_{32}B_{21} + A2_{34}B_{41}\right)u_1 \\ + \left(A2_{32}B_{22} + A2_{34}B_{42}\right)u_2 + B_{41}\dot{u}_1 + B_{42}\dot{u}_2 - \tfrac{d^3}{dt^3}y_3 \end{gathered}} \tag{5.44}$$

This is the derivative of LT_9.

$$\boxed{LT_{11} = y_4 - y_4} \tag{5.45}$$

This is a trivial test.

$$\boxed{LT_{12} = A_{42}y_2 + A_{44}y_4 + B_{41}u_1 + B_{42}u_2 - \dot{y}_4} \tag{5.46}$$

This is the second model equation tested against the tachometer. It can be converted to a sensor comparison.

$$\boxed{\begin{gathered} LT_{13} = \left(A2_{42}y_2 + A2_{44}y_4\right) + \left(A_{42}B_{21} + A_{44}B_{41}\right)u_1 \\ + \left(A_{42}B_{22} + A_{44}B_{42}\right)u_2 + B_{41}\dot{u}_1 + B_{42}\dot{u}_2 - \ddot{y}_4 \end{gathered}} \tag{5.47}$$

This is the derivative of LT_{12}.

A table of the tests follows (Table 5.2). Tests are labeled LR_i where i is the index of the corresponding nonlinear test.

The linear AR generates three fewer tests than NLAR. Obviously the linearized model's observation space O is not a good model of the real observation space $\mathcal{O}$. In terms of the tests that both types of AR create, the sensor comparisons are the same in the linear and nonlinear residuals, but the test residuals that depend on the model

Test Label	Description
LR_1	Second model equation checked against shoulder resolver. Tests the acceleration of the system.
LR_2	Derivative of second model equation checked against shoulder resolver. Tests the jerk of the system.
LR_4	Sensor comparison of shoulder tachometer with derivative of shoulder resolver.
LR_5	Derivative of LR_4.
LR_7	Fourth model equation checked against elbow resolver. Tests the acceleration of the system.
LR_8	Derivative of fourth model equation checked against elbow resolver. Tests the jerk of the system.
LR_9	Sensor comparison of elbow tachometer and derivative of elbow resolver.
LR_{10}	Derivative of LR_9.

Table 5.2 Linear AR tests for IMI test robot.

are inferior to the NLAR tests, as they compare the sensor data to an inaccurate model of the system.

5.3 Results

Several sets of NLAR and AR residual test results are presented below. The tests were run at 100Hz, sampling every 10ms at the same time as the controller. (Recall the model ran at 1000Hz.) The model was given 300ms to settle before NLAR testing began. Faults were considered detected if the magnitude of the NLAR residual was at least twice the maximum value achieved in a fault free run with the same parameters. More sophisticated detection techniques are certainly possible, but are not developed here.

In the test trajectory each joint's input signal was at a different phase and frequency to maximize the variety of joint coupling effects experienced, as shown in equation 5.48:

$$\begin{aligned} \theta_{d1}(t) &= 0.5\sin(t/0.3), \\ \theta_{d2}(t) &= 0.5\cos(t/0.4) - 0.5. \end{aligned} \tag{5.48}$$

Note that the observability space does not have a standard set of unit dimensions and as a result the output of NLAR test residuals is not standardized either. The tests are in units of velocity, acceleration, jerk, and even the derivative of the jerk. This can be confusing, but for fault detection in practice one is not interested in the units of the test, but only in their relative magnitude. The size of the signal relative to the fault-free "noise" is of interest, but the units or absolute size are of little consequence.

The NLAR output of Figure 5.2 is typical of the results. Before the fault occurs at $t = 6s$, the NLAR tests mostly show noise-like fluctuations around a mean of 0. This is to be expected from fault-free NLAR tests as measurement error and unmodeled effects will prevent perfect test residuals in practice. (It is interesting to note that the noise looks like a superposition of sine waves, which as the input is sinusoidal are likely model-following errors, and step functions, which are a consequence of the discontinuous, unmodeled Coulomb friction.) Once the fault occurs this particular test detects it on the same time step. Close examination of the fault detail on Figure 5.2 shows that the residual at $t = 6s$ is about an order of magnitude larger than the fault free noise.

The table of residuals presented in Figure 5.3 represents the results of our NLAR tests for a frozen sensor fault - the shoulder resolver is frozen at its current value at $t = 6s$. As seen in the plotted results, all of the NLAR tests detect the fault very quickly; many of them detect it on the very next time step. This is a promising result, as a frozen sensor does not immediately cause significant tracking errors.

The next example, shown in Figure 5.4, demonstrates a fault that should be very difficult to detect. This shows sensor drift fault, where the value for a sensor became slowly and smoothly less accurate over time. This kind of fault is much more difficult

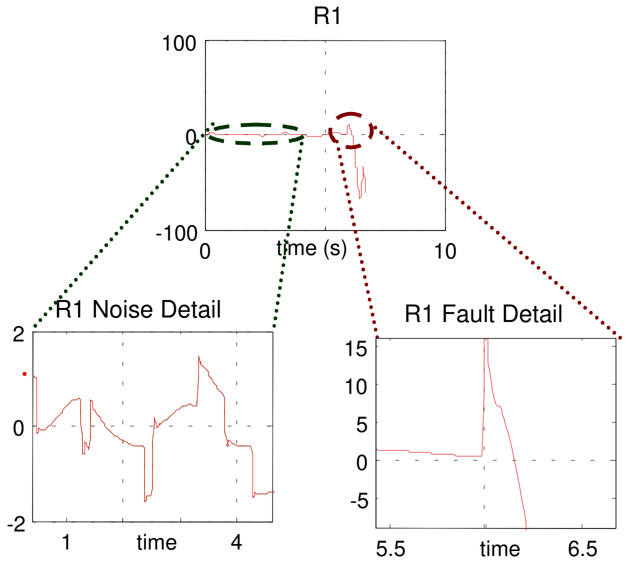
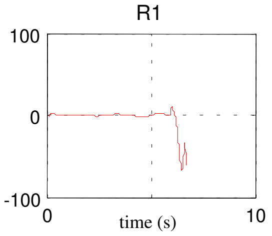
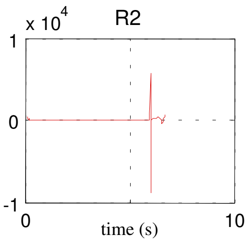
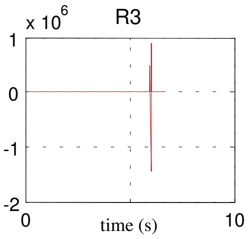
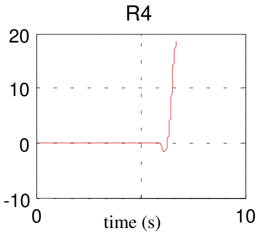
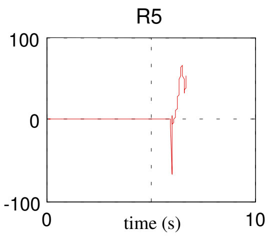


Figure 5.2 Typical IMI NLAR test residual.

to detect than most, since the inconsistencies with respect to the model start at zero and only increase smoothly and slowly to a detectable value. Since most of the information in the model deals with dynamic behaviors, this slow change produces signals smaller than the noise level on all but one of the NLAR tests. The R_1 result in the figure is typical of these. However, R_4, which directly tests sensor redundancy, gives a detectable signal. Comparison with the sensor error plot below it on the same figure shows the discrimination of the NLAR test. Before the tracking error is even visible, NLAR has registered a residual twice the fault-free size.

The final fault examined here is a broken motor/drive train fault. In this fault, input to the shoulder motor stops producing torque in the joint at $t = 6$ seconds. This fault is interesting in that it produces several NLAR residuals that show signals at the time of the fault that are obvious to the human eye but not twice the magnitude of the fault free case. These faults are labeled as “detectable” in Figure 5.5, as they could be detected by a more sophisticated residual analysis system. However, even assuming the “detectable” residuals are not exploited, there are several NLAR residuals that detect the fault in a timely fashion using the basic “twice the magnitude” approach.

Figure 5.6 is a comparison figure showing the improvement in performance of NLAR over linear AR for the same broken motor fault as Figure 5.5. Note that linear AR produces eight rather than eleven tests, and that four of these are sensor comparisons identical to the NLAR tests and therefore not shown. This leaves four tests that are analogous to NLAR tests but linear in nature. For example, there is a linear AR test that examines the linear model of acceleration of each joint that can be compared to the NLAR test of the nonlinear, model of the same. These analogous tests are compared in Figure 5.6 . It is clear that NLAR outperforms linear AR on all four of these tests by about an order of magnitude. This is likely a result of the poor modeling of the nonlinear coupling and dynamical effects of the manipulator in the linear AR tests.

NLAR Residual Description	Results for frozen sensor fault run	Time
R1: Acceleration of the shoulder.		10ms 1 step
R2: Jerk of the shoulder.		20ms 2 steps
R3: Second derivative of the second model equation. (Fourth derivative of shoulder.)		10ms 1 step
R4: Sensor comparison of shoulder tachometer with derivative of shoulder resolver.		30ms 3 steps
R5: Derivative of R4.		10ms 1 step

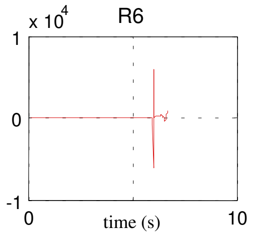
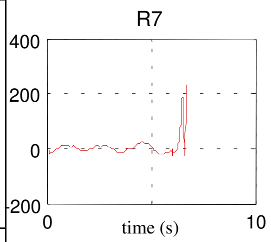
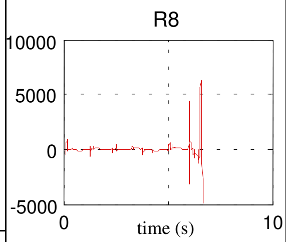
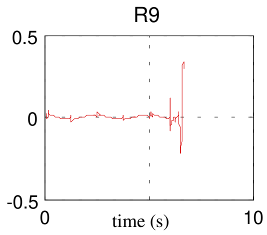
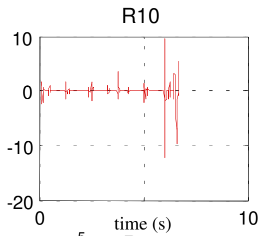
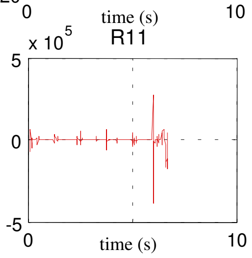
R6: Second derivative of R4.		10ms 1 step
R7: Acceleration of the elbow.		400ms 40 steps
R8: Jerk of elbow.		30ms 3 steps
R9: Sensor comparison of elbow tachometer with derivative of elbow resolver.		40ms 4 steps
R10: Derivative of R9.		30ms 3 steps
R11: Second derivative of the fourth model equation. (Fourth derivative of elbow.)		20ms 2 steps

Figure 5.3 IMI NLAR residuals for a frozen sensor fault.

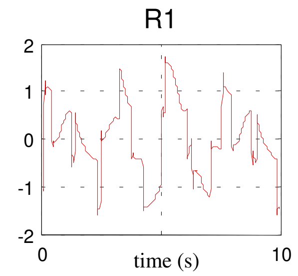
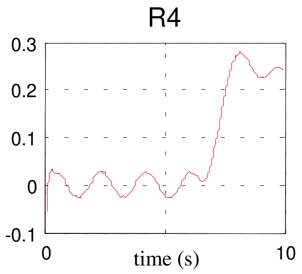
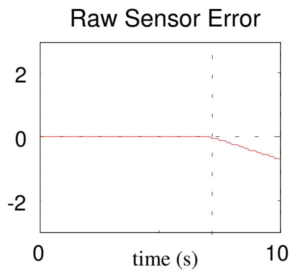
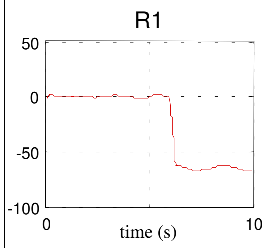
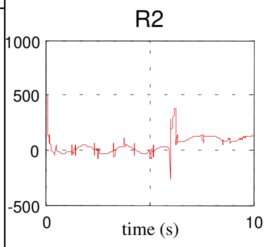
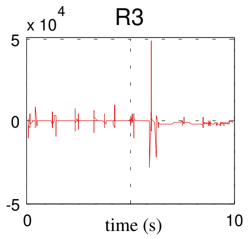
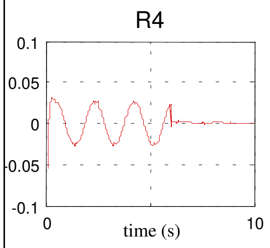
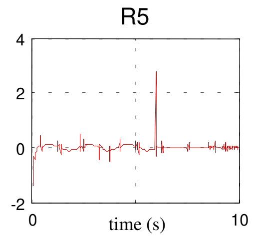
NLAR Residual Description	Selected results for sensor drift fault run	Time
R1: Acceleration of the shoulder.		signal less than noise
R4: Sensor comparison of shoulder tachometer with derivative of shoulder resolver.		1.24s 124 steps
Plot of sensor error vs. time. Vertical line is at 7.24s, time of NLAR fault detection.		(Not a NLAR test.) y-axis is radians

Figure 5.4 Selected IMI NLAR residuals, sensor drift fault.

NLAR Residual Description	Results for motor fault	Time
R1: Acceleration of the shoulder.		30ms 3 steps
R2: Jerk of the shoulder.		detect-able
R3: Second derivative of the second model equation. (Fourth derivative of shoulder.)		10ms 1 step
R4: Sensor comparison of shoulder tachometer with derivative of shoulder resolver.		detect-able
R5: Derivative of R4.		detect-able

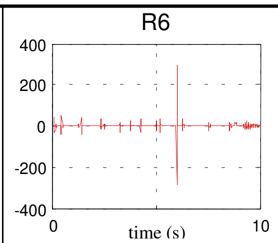
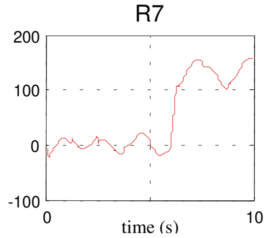
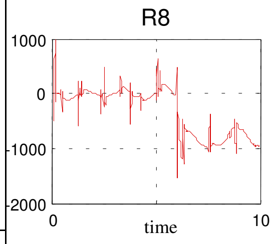
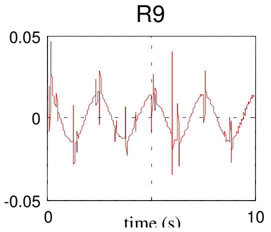
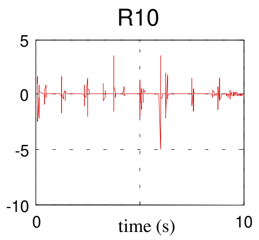
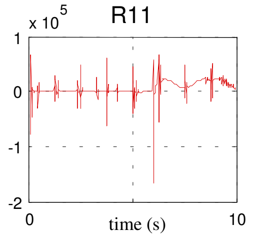
R6: Second derivative of R4.		20ms 2 steps
R7: Acceleration of the elbow.		160ms 16 steps
R8: Jerk of elbow.		detect-able
R9: Sensor comparison of elbow tachometer with derivative of elbow resolver.		detect-able
R10: Derivative of R9.		detect-able
R3: Second derivative of the fourth model equation. (Fourth derivative of elbow.).		30ms 3 steps

Figure 5.5 IMI NLAR residuals, motor fault.

5.3.1 Evaluation of Results

Clearly nonlinear analytical redundancy performs well as a fault detector for the IMI robot. Even subtle faults are detected both clearly and quickly. Additionally, NLAR offers obvious improvements over the linear AR method. More NLAR tests are generated than LAR tests, and these tests are clearly theoretically justified. The NLAR tests also follow the model much more closely than the LAR tests, resulting in more sensitive fault detection. Overall, the performance of NLAR on the IMI robot is very encouraging.

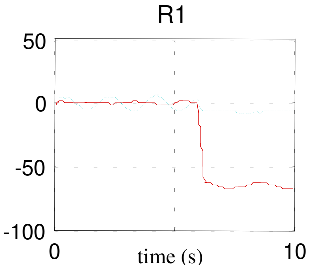
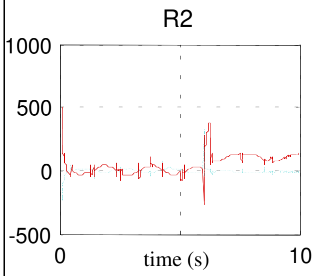
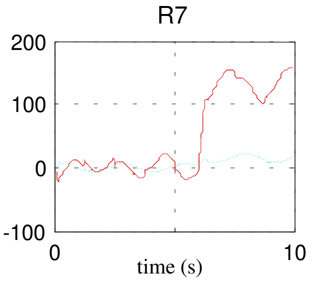
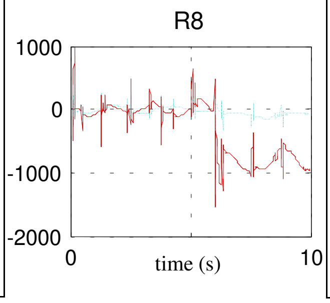
NLAR Residual Description	Results for motor fault	NLAR better because
R1: Acceleration of the shoulder.		less noise, larger signal
R2: Jerk of the shoulder.		larger signal
R7: Acceleration of the elbow.		larger signal
R8: Jerk of elbow.		larger signal

Figure 5.6 IMI NLAR test result (solid dark) vs. linear AR test result (dotted light) for motor fault.

Chapter 6

Conclusions and Future Work

6.1 Contributions of the Thesis

The main contribution of this thesis is the new nonlinear analytical redundancy (NLAR) fault detection technique. NLAR allows rigorous model-based fault detection for nonlinear systems without requiring model-weakening linearizations. NLAR test residuals are guaranteed to see any possible observable deviation from the system model [5, 30, 42], a very desirable trait in any fault detection system. The ability to deal accurately with nonlinear models is very desirable in robotics as well as many other disciplines [30, 33].

The Integrated Motion Inc. (IMI) robot example in this thesis shows excellent residual fault detection results. Its quick detection of serious faults and sensitivity to even the very subtle drifting fault show how effectively NLAR can find deviations from model-following behavior. NLAR also clearly outperforms the linear AR technique. While the synthesis of NLAR and NLAR test residuals can be nontrivial, the test residuals themselves are quite intuitive and once derived can be evaluated in real time. Overall the NLAR technique shows considerable promise for practical implementation.

6.2 Design Complexity and Model Accuracy

There are two main limitations on the application of NLAR. One limitation is the complexity of calculation that can be required to generate test residuals for systems such as the IMI robot. Another is the reliance of NLAR on high degrees of model accuracy.

The complexity issue is easily dealt with by automation of the Lie differentiation process, as was done in Chapter 5 and Appendix A. This is a relatively minor hurdle for potential users of NLAR, as automated equation solvers are common tools. As even the complex NLAR tests are scalar sums, resource usage due to computational complexity is also a minor issue.

The model accuracy issue is the only remaining factor limiting the application of NLAR. Physical systems can be difficult to model accurately, and nonlinear systems are doubly so. However, this is not likely to be a large issue on systems where nonlinear analytical redundancy is appropriate. If the system is important and sensitive enough to require the kind of detailed fault detection provided by NLAR, a relatively accurate model of the system most likely already exists to facilitate other fault detection and fault prevention techniques.

6.3 Directions for Further Research

Future work with NLAR is likely to focus on both testing against data from well-modeled nonlinear physical systems and the production of more elaborate systems for analyzing the residual signals. The IMI robot is a likely candidate for future work with physical data, as it is well-modeled, already has AR tests derived for it, and is accessible to the NLAR research group.

Analysis of the data from the test residuals is potentially a very rich area for further research. Characterizing specific kinds of faults by their AR signals and thus building better fault detectors are immediate and important goals in the application of NLAR.

Theoretically dealing with non-smooth types of nonlinearities is another interesting future project. The $\mathcal{O}$ notion of nonlinear observability is incompatible with discontinuous systems. However systems which are best modeled by mathematically discontinuous models are not uncommon. Dealing with discontinuities robustly would further increase the sensitivity of NLAR on many systems.

Investigation of the interaction of data sampling with the nonlinear system's modeling accuracy is another avenue of potentially useful research. Nonlinear systems are not necessarily as tractable with respect to discretization as linear systems, but certainly guarantees and transformations that would increase the robustness of NLAR applications are possible.

Finally, creating an automated NLAR software package would be a useful and interesting project. Many of the steps of NLAR are quite mechanistic, and creating a software package that generated tests residuals automatically is possible. Such software would greatly facilitate future application of the technique.

Appendix A

Automatic Derivation of NLAR for the IMI Robot

The two degree of freedom planar robot example of Chapter 5 did not explicitly derive some of the larger terms (Υ_i) involved in NLAR for the IMI robot. This appendix contains a brief discussion of why these Υ terms are so large, Matlab code for deriving these terms, and explicitly presents the larger terms.

A.1 Complexity Issues for the Higher Order terms of $\mathcal{O}_\triangle$

Consider the triangular nonlinear observability ($\mathcal{O}_\triangle$) defined in equation 3.5, and reproduced below:

$$\mathcal{O}_\triangle = \begin{bmatrix} h\left(\underline{x}(t)\right) \\ \sum\limits_{j=0}^{q} L(j) \\ \sum\limits_{l=0}^{q}\sum\limits_{j=0}^{q} L(j,l) \\ \sum\limits_{m=0}^{q}\sum\limits_{l=0}^{q}\sum\limits_{j=0}^{q} L(j,l,m) \\ \vdots \end{bmatrix},$$

where:

$$L(j,l,m,\ldots) = \left(u_j u_l u_m \ldots\right) L_{\ldots k(m)k(l)k(j)} h\left(\underline{x}(t)\right)$$

$$u_0 = 1, \quad k(j) = \begin{cases} f & ,j=0 \\ g_j & ,j \neq 0 \end{cases},$$

and q is the number of control inputs.

Note that each successive line of $\mathcal{O}_\triangle$ has $q+1$ times as many terms than the previous line. For the IMI robot, $q=2$, and a progression where each line has 3 times as many terms as the previous results:

$$O_{NL} = \begin{bmatrix} C \cdot \underline{x}(t) \\ L_f C\underline{x} + u_1 L_{g_1} C\underline{x} + u_2 L_{g_2} C\underline{x} \\ \begin{pmatrix} L_{ff} C\underline{x} + u_1 L_{g_1 f} C\underline{x} + u_2 L_{g_2 f} C\underline{x} \\ +u_1 L_{fg_1} C\underline{x} + u_1^2 L_{g_1 g_1} C\underline{x} + u_1 u_2 L_{g_2 g_1} C\underline{x} \\ +u_2 L_{fg_2} C\underline{x} + u_1 u_2 L_{g_1 g_2} C\underline{x} + u_2^2 L_{g_2 g_2} C\underline{x} \end{pmatrix} \\ (27\,\text{terms}) \\ (81\,\text{terms}) \\ \vdots \end{bmatrix}.$$

Additionally, each succesive line increases the degree of Lie differentiation of the component terms by one. Since each Lie differentiation includes a gradient with respect to the state vector, and the components of the IMI system model are of course nonlinear with respect to the state vector, the succesive derivatives can get quite large. Consider the following Lie deriviatives with respect to c_2.

The zeroth order Lie derivative is a single variable:

$$c_2\underline{x} = \dot{q}_1.$$

The first order Lie derivative is a large function, equal to f_2:

$$L_f c_2 \underline{x} =$$

$$((p_3 \sin(q_2)\left((p_2 + p_3\cos(q_2))\,\dot{q}_1 + p_2\dot{q}_2\right) - p_2 f_{d1})\,\dot{q}_1)\,/p_1p_2 - {p_2}^2 - {p_3}^2\left(\cos(q_2)\right)^2 \cdot$$
$$+((p_2p_3\sin(q_2)\left(\dot{q}_1 + \dot{q}_2\right) + (p_2 + p_3\cos(q_2))\,f_{d2})\,\dot{q}_2)\,/p_1p_2 - {p_2}^2 - {p_3}^2\left(\cos(q_2)\right)^2$$

The second order Lie derivative is a very large, complex function:

$L_{ff}c_2\underline{x}$

$$= \frac{\begin{pmatrix} -8{p_3}^3\dot{q}_1\cos(q2)p_2{\dot{q}_2}^2 - 3{p_3}^2\dot{q}_2{\dot{q}_1}^2p_1p_2 \\ +3{p_3}^2{\dot{q}_1}^2{p_2}^2\dot{q}_2 - 4{p_3}^3{\dot{q}_1}^2\cos(q2)p_2\dot{q}_2 - 2{p_3}^2{\dot{q}_1}^3p_2p_1 \\ +2{p_3}^2{\dot{q}_1}^3p_2p_1\left(\cos(q2)\right)^2 - {p_3}^4\dot{q}_2{\dot{q}_1}^2\left(\cos(q2)\right)^2 - 4{p_3}^3{\dot{q}_2}^3\cos(q2)p_2 \\ +2{p_3}^2{\dot{q}_1}^3{p_2}^2 - 2{p_3}^2{\dot{q}_1}^3{p_2}^2\left(\cos(q2)\right)^2 - p_2f_{d1}\dot{q}_2f_{d2}p_3\cos(q2) \\ +2p_3{\dot{q}_2}^2\dot{q}_1\cos(q2){p_2}^2p_1 + 6{p_3}^3{\dot{q}_2}^2\dot{q}_1\left(\cos(q2)\right)^3p_2 - {p_2}^2f_{d1}p_3\sin(q2){\dot{q}_2}^2 \\ +4{p_3}^2\dot{q}_2{\dot{q}_1}^2\left(\cos(q2)\right)^2p_1p_2 - p_3\dot{q}_2\cos(q2){\dot{q}_1}^2{p_2}^3 + 2p_3\sin(q2){p_2}^2{\dot{q}_2}^2f_{d2} \\ -6{p_3}^2\sin(q2)p_2{\dot{q}_2}^2f_{d2}\cos(q2) - {p_3}^2\sin(q2){\dot{q}_1}^2\cos(q2)p_2f_{d1} \\ -2p_3\sin(q2){p_2}^2\dot{q}_2\dot{q}_1f_{d1} - p_3\sin(q2){\dot{q}_1}^2{p_2}^2f_{d1} \\ -4{p_3}^2\sin(q2)\dot{q}_1p_2\dot{q}_2f_{d2}\cos(q2) \\ +f_{d2}{p_3}^2\left(\cos(q2)\right)^2\dot{q}_1f_{d1} - f_{d2}{p_3}^2\cos(q2)\sin(q2){\dot{q}_1}^2p_1 \\ -f_{d2}p_2p_3\sin(q2){\dot{q}_1}^2p_1 - 2f_{d2}p_2{p_3}^2\sin(q2){\dot{q}_1}^2\cos(q2) \\ +2f_{d2}p_2\dot{q}_1f_{d1}p_3\cos(q2) - 2{f_{d2}}^2p_2\dot{q}_2p_3\cos(q2) \\ -2f_{d2}{p_3}^3\left(\cos(q2)\right)^2\sin(q2){\dot{q}_1}^2 \\ -2p_3\sin(q2)\dot{q}_1p_2\dot{q}_2f_{d2}p_1 - {f_{d2}}^2p_3\cos(q2)\dot{q}_2p_1 \\ +2{p_3}^4{\dot{q}_1}^3\left(\cos(q2)\right)^2 - 2{p_3}^4{\dot{q}_1}^3\left(\cos(q2)\right)^4 \\ -2{p_3}^3{\dot{q}_2}^2\sin(q2)f_{d2}\left(\cos(q2)\right)^2 + p_3{\dot{q}_2}^3\cos(q2){p_2}^2p_1 \\ +4{p_3}^2\dot{q}_2\dot{q}_1\cos(q2)\sin(q2)p_2f_{d1} + p_3\dot{q}_2\cos(q2){\dot{q}_1}^2{p_2}^2p_1 \\ +3{p_3}^3\dot{q}_2\left(\cos(q2)\right)^3{\dot{q}_1}^2p_2 \\ -4{p_3}^2\dot{q}_2{\dot{q}_1}^2\left(\cos(q2)\right)^2{p_2}^2 - 2p_3{\dot{q}_2}^2\dot{q}_1\cos(q2){p_2}^3 \\ -3p_3{\dot{q}_2}^2\sin(q2)f_{d2}p_1p_2 + 3{p_3}^3{\dot{q}_2}^3\left(\cos(q2)\right)^3p_2 \\ -p_3{\dot{q}_2}^3\cos(q2){p_2}^3 - 2{f_{d2}}^2{p_3}^2\left(\cos(q2)\right)^2\dot{q}_2 \\ +{p_2}^2{f_{d1}}^2\dot{q}_1 - {p_2}^2f_{d1}\dot{q}_2f_{d2} + f_{d2}{p_2}^2\dot{q}_1f_{d1} - {f_{d2}}^2p_2\dot{q}_2p_1 \end{pmatrix}}{\left({p_2}^2 - p_1p_2 + {p_3}^2\left(\cos(q2)\right)^2\right)^2}.$$

Unsurprisingly, the third order Lie derivative $L_{fff}c_2\underline{x}$ is too large to fit on a single page.

One of the terms in the $\mathcal{O}_{\triangle}$ space of the IMI robot is the sum of *eighty-one fourth* order Lie derivatives. Obviously, calculating this term is not a reasonable task for humans. For this reason the determination of the terms was automated using Matlab, as discussed in section A.2.

A.2 Automated Calculation of Large Terms Using Matlab

Due to the mathematically intensive nature of many of the NLAR terms needed for the IMI example of Chapter 5, automatec calculation of these terms was deemed necessary. The Matlab [35] Symbolic Toolbox was used to generate these terms and combine them appropriately to create the NLAR tests. These tests were then transferred to the numerical side of Matlab and combined with the robot simulator to generate the NLAR test residuals of Chapter 5.

A.2.1 The Robot_Definition File

This Matlab file was the first step in automating the NLAR derivation. Robot_Definition serves to define the robot in the symbolic workspace. The various constants, variables, and functions needed for the NLAR derivation are defined in this file.

`Robot_Definition.m`

```
%symbolic constants
syms p1 p2 p3 frd1 frd2;
%state variables
syms q1 q2 q1d q2d q1dd q2dd q1ddd q2ddd q1dddd q2dddd;
%sensors
syms r1 r1d r1dd r1ddd r1dddd tc1 tc1d tc1dd tc1ddd;
syms r2 r2d r2dd r2ddd r2dddd tc2 tc2d tc2dd tc2ddd;
%internal symbols
```

```
syms t alpha beta gamma delta varphi temp;
%control equation functions
syms f1 f2 f3 f4 f g g1 g11 g12 g13 g14 g2 g21 g22 ...
g23 g24 u1 u2 u1d u2d u1dd u2dd u1ddd u2ddd;
%the basis vector
syms basis; basis = [q1; q1d; q2; q2d];
%the observability matrix C
syms C c1 c2 c3 c4;
c1=[1 0 0 0];c2=[0 1 0 0];c3=[0 0 1 0]; c4=[ 0 0 0 1];
C=[c1;c2;c3;c4];

%define the internal symbols
varphi = p1*p2 - p2^2 - p3^2*cos(q2)*cos(q2);

alpha = ( p3*sin(q2)*( (p2 + p3*cos(q2))*q1d + p2*q2d) ...
    - p2*frd1)/varphi;

beta = ( p2*p3*sin(q2)*(q1d+q2d) + (p2 + p3*cos(q2))*frd2 )...
    /varphi;

gamma = ( -p3*sin(q2)*( ( p1 + 2*p3*cos(q2) )*q1d ...
 + ( p2 + p3*cos(q2) )*q2d )+ ( p2 + p3*cos(q2))*frd1 )/varphi;

delta = ( -p3*sin(q2)*( p2 + p3*cos(q2) )*( q1d + q2d ) ...
    -( p1 + 2*p3*cos(q2) )*frd2 )/varphi;

%define the control functions
f1=q1d;
```

```
f2=alpha*q1d+beta*q2d;
f3=q2d;
f4=gamma*q1d+delta*q2d;
g11=0;
g12= p2/varphi;
g13=0;
g14=-( p2 + p3*cos(q2) )/varphi;
g21=0;
g22=-( p2+p3*cos(q2) )/varphi;
g23=0;
g24=( p1 + p3*cos(q2) )/varphi;
f=[f1;f2;f3;f4];
g1=[g11; g12; g13; g14];
g2=[g21; g22; g23; g24];
g=[g1 g2];

%define x-derivatives for use in test residues
xd=[q1d; q1dd; q2d; q2dd]; xdd=[q1dd; q1ddd; q2dd; q2ddd];
```

A.2.2 The Robot_Lie File

This large file used the Matlab symbolic toolbox's symbolic manipulation ability to calculate the various Lie derivatives needed in the NLAR tests. The Robot_Lie file is a large list of recursed Lie derivatives. The liediff and gradf functions do the actual differentiation.

```
Robot_Lie.m

% First order Lie derivatives...
```

```
Lfc1=liediff(f,c1*basis,basis);
Lg1c1=u1*liediff(g1,c1*basis,basis);
Lg2c1=u2*liediff(g2,c1*basis,basis);
L1c1=Lfc1+Lg1c1+Lg2c1;

Lfc2=liediff(f,c2*basis,basis);
Lg1c2=u1*liediff(g1,c2*basis,basis);
Lg2c2=u2*liediff(g2,c2*basis,basis);
L1c2=Lfc2+Lg1c2+Lg2c2;

Lfc3=liediff(f,c3*basis,basis);
Lg1c3=u1*liediff(g1,c3*basis,basis);
Lg2c3=u2*liediff(g2,c3*basis,basis);
L1c3=Lfc3+Lg1c3+Lg2c3;

Lfc4=liediff(f,c4*basis,basis);
Lg1c4=u1*liediff(g1,c4*basis,basis);
Lg2c4=u2*liediff(g2,c4*basis,basis);
L1c4=Lfc4+Lg1c4+Lg2c4;


% Second order Lie derivatives...
% f
Lffc1=liediff(f,Lfc1,basis);
Lfg1c1=liediff(f,Lg1c1,basis);
Lfg2c1=liediff(f,Lg2c1,basis);
```

```
Lffc2=liediff(f,Lfc2,basis);
Lfg1c2=liediff(f,Lg1c2,basis);
Lfg2c2=liediff(f,Lg2c2,basis);

Lffc3=liediff(f,Lfc3,basis);
Lfg1c3=liediff(f,Lg1c3,basis);
Lfg2c3=liediff(f,Lg2c3,basis);

Lffc4=liediff(f,Lfc4,basis);
Lfg1c4=liediff(f,Lg1c4,basis);
Lfg2c4=liediff(f,Lg2c4,basis);

% g1
Lg1fc1=u1*liediff(g1,Lfc1,basis);
Lg1g1c1=u1*liediff(g1,Lg1c1,basis);
Lg1g2c1=u1*liediff(g1,Lg2c1,basis);

Lg1fc2=u1*liediff(g1,Lfc2,basis);
Lg1g1c2=u1*liediff(g1,Lg1c2,basis);
Lg1g2c2=u1*liediff(g1,Lg2c2,basis);

Lg1fc3=u1*liediff(g1,Lfc3,basis);
Lg1g1c3=u1*liediff(g1,Lg1c3,basis);
Lg1g2c3=u1*liediff(g1,Lg2c3,basis);

Lg1fc4=u1*liediff(g1,Lfc4,basis);
Lg1g1c4=u1*liediff(g1,Lg1c4,basis);
Lg1g2c4=u1*liediff(g1,Lg2c4,basis);
```

```
% g2
Lg2fc1=u2*liediff(g2,Lfc1,basis);
Lg2g1c1=u2*liediff(g2,Lg1c1,basis);
Lg2g2c1=u2*liediff(g2,Lg2c1,basis);

Lg2fc2=u2*liediff(g2,Lfc2,basis);
Lg2g1c2=u2*liediff(g2,Lg1c2,basis);
Lg2g2c2=u2*liediff(g2,Lg2c2,basis);

Lg2fc3=u2*liediff(g2,Lfc3,basis);
Lg2g1c3=u2*liediff(g2,Lg1c3,basis);
Lg2g2c3=u2*liediff(g2,Lg2c3,basis);

Lg2fc4=u2*liediff(g2,Lfc4,basis);
Lg2g1c4=u2*liediff(g2,Lg1c4,basis);
Lg2g2c4=u2*liediff(g2,Lg2c4,basis);

% Summations
L2c1=Lffc1+Lfg1c1+Lfg2c1+Lg1fc1+...
Lg1g1c1+Lg1g2c1+Lg2fc1+Lg2g1c1+Lg2g2c1;
L2c2=Lffc2+Lfg1c2+Lfg2c2+Lg1fc2+Lg1g1c2+...
Lg1g2c2+Lg2fc2+Lg2g1c2+Lg2g2c2;
L2c3=Lffc3+Lfg1c3+Lfg2c3+Lg1fc3+Lg1g1c3+...
Lg1g2c3+Lg2fc3+Lg2g1c3+Lg2g2c3;
L2c4=Lffc4+Lfg1c4+Lfg2c4+Lg1fc4+Lg1g1c4+...
Lg1g2c4+Lg2fc4+Lg2g1c4+Lg2g2c4;
```

```
% Third order Lie derivatives...

% c1
% f
Lfffc1 =liediff(f,Lffc1,basis);
Lffg1c1=liediff(f,Lfg1c1,basis);
Lffg2c1=liediff(f,Lfg2c1,basis);

Lfg1fc1 =liediff(f,Lg1fc1,basis);
Lfg1g1c1=liediff(f,Lg1g1c1,basis);
Lfg1g2c1=liediff(f,Lg1g2c1,basis);

Lfg2fc1 =liediff(f,Lg2fc1,basis);
Lfg2g1c1=liediff(f,Lg2g1c1,basis);
Lfg2g2c1=liediff(f,Lg2g2c1,basis);
% g1
Lg1ffc1 =u1*liediff(g1,Lffc1,basis);
Lg1fg1c1=u1*liediff(g1,Lfg1c1,basis);
Lg1fg2c1=u1*liediff(g1,Lfg2c1,basis);

Lg1g1fc1 =u1*liediff(g1,Lg1fc1,basis);
Lg1g1g1c1=u1*liediff(g1,Lg1g1c1,basis);
Lg1g1g2c1=u1*liediff(g1,Lg1g2c1,basis);

Lg1g2fc1 =u1*liediff(g1,Lg2fc1,basis);
Lg1g2g1c1=u1*liediff(g1,Lg2g1c1,basis);
Lg1g2g2c1=u1*liediff(g1,Lg2g2c1,basis);
```

```
% g2
Lg2ffc1 =u2*liediff(g2,Lffc1,basis);
Lg2fg1c1=u2*liediff(g2,Lfg1c1,basis);
Lg2fg2c1=u2*liediff(g2,Lfg2c1,basis);

Lg2g1fc1 =u2*liediff(g2,Lg1fc1,basis);
Lg2g1g1c1=u2*liediff(g2,Lg1g1c1,basis);
Lg2g1g2c1=u2*liediff(g2,Lg1g2c1,basis);

Lg2g2fc1 =u2*liediff(g2,Lg2fc1,basis);
Lg2g2g1c1=u2*liediff(g2,Lg2g1c1,basis);
Lg2g2g2c1=u2*liediff(g2,Lg2g2c1,basis);

L3c1=Lfffc1+Lffg1c1+Lffg2c1+Lfg1fc1+...
Lfg1g1c1+Lfg1g2c1+Lfg2fc1+Lfg2g1c1+Lfg2g2c1 ...
+Lg1ffc1+Lg1fg1c1+Lg1fg2c1+Lg1g1fc1+...
Lg1g1g1c1+Lg1g1g2c1+Lg1g2fc1+Lg1g2g1c1+Lg1g2g2c1 ...
+Lg2ffc1+Lg2fg1c1+Lg2fg2c1+Lg2g1fc1+...
Lg2g1g1c1+Lg2g1g2c1+Lg2g2fc1+Lg2g2g1c1+Lg2g2g2c1;

% c2
% f
Lfffc2 =liediff(f,Lffc2,basis);
Lffg1c2=liediff(f,Lfg1c2,basis);
Lffg2c2=liediff(f,Lfg2c2,basis);

Lfg1fc2 =liediff(f,Lg1fc2,basis);
Lfg1g1c2=liediff(f,Lg1g1c2,basis);
```

```
Lfg1g2c2=liediff(f,Lg1g2c2,basis);

Lfg2fc2 =liediff(f,Lg2fc2,basis);
Lfg2g1c2=liediff(f,Lg2g1c2,basis);
Lfg2g2c2=liediff(f,Lg2g2c2,basis);
% g1
Lg1ffc2 =u1*liediff(g1,Lffc2,basis);
Lg1fg1c2=u1*liediff(g1,Lfg1c2,basis);
Lg1fg2c2=u1*liediff(g1,Lfg2c2,basis);

Lg1g1fc2 =u1*liediff(g1,Lg1fc2,basis);
Lg1g1g1c2=u1*liediff(g1,Lg1g1c2,basis);
Lg1g1g2c2=u1*liediff(g1,Lg1g2c2,basis);

Lg1g2fc2 =u1*liediff(g1,Lg2fc2,basis);
Lg1g2g1c2=u1*liediff(g1,Lg2g1c2,basis);
Lg1g2g2c2=u1*liediff(g1,Lg2g2c2,basis);
% g2
Lg2ffc2 =u2*liediff(g2,Lffc2,basis);
Lg2fg1c2=u2*liediff(g2,Lfg1c2,basis);
Lg2fg2c2=u2*liediff(g2,Lfg2c2,basis);

Lg2g1fc2 =u2*liediff(g2,Lg1fc2,basis);
Lg2g1g1c2=u2*liediff(g2,Lg1g1c2,basis);
Lg2g1g2c2=u2*liediff(g2,Lg1g2c2,basis);

Lg2g2fc2 =u2*liediff(g2,Lg2fc2,basis);
Lg2g2g1c2=u2*liediff(g2,Lg2g1c2,basis);
```

```
Lg2g2g2c2=u2*liediff(g2,Lg2g2c2,basis);

L3c2=Lfffc2+Lffg1c2+Lffg2c2+Lfg1fc2...
+Lfg1g1c2+Lfg1g2c2+Lfg2fc2+Lfg2g1c2+Lfg2g2c2 ...
+Lg1ffc2+Lg1fg1c2+Lg1fg2c2+Lg1g1fc2...
+Lg1g1g1c2+Lg1g1g2c2+Lg1g2fc2+Lg1g2g1c2+Lg1g2g2c2 ...
+Lg2ffc2+Lg2fg1c2+Lg2fg2c2+Lg2g1fc2...
+Lg2g1g1c2+Lg2g1g2c2+Lg2g2fc2+Lg2g2g1c2+Lg2g2g2c2;

% c3
% f
Lfffc3 =liediff(f,Lffc3,basis);
Lffg1c3=liediff(f,Lfg1c3,basis);
Lffg2c3=liediff(f,Lfg2c3,basis);

Lfg1fc3 =liediff(f,Lg1fc3,basis);
Lfg1g1c3=liediff(f,Lg1g1c3,basis);
Lfg1g2c3=liediff(f,Lg1g2c3,basis);

Lfg2fc3 =liediff(f,Lg2fc3,basis);
Lfg2g1c3=liediff(f,Lg2g1c3,basis);
Lfg2g2c3=liediff(f,Lg2g2c3,basis);
% g1
Lg1ffc3 =u1*liediff(g1,Lffc3,basis);
Lg1fg1c3=u1*liediff(g1,Lfg1c3,basis);
Lg1fg2c3=u1*liediff(g1,Lfg2c3,basis);

Lg1g1fc3 =u1*liediff(g1,Lg1fc3,basis);
```

```
Lg1g1g1c3=u1*liediff(g1,Lg1g1c3,basis);
Lg1g1g2c3=u1*liediff(g1,Lg1g2c3,basis);

Lg1g2fc3 =u1*liediff(g1,Lg2fc3,basis);
Lg1g2g1c3=u1*liediff(g1,Lg2g1c3,basis);
Lg1g2g2c3=u1*liediff(g1,Lg2g2c3,basis);
% g2
Lg2ffc3 =u2*liediff(g2,Lffc3,basis);
Lg2fg1c3=u2*liediff(g2,Lfg1c3,basis);
Lg2fg2c3=u2*liediff(g2,Lfg2c3,basis);

Lg2g1fc3 =u2*liediff(g2,Lg1fc3,basis);
Lg2g1g1c3=u2*liediff(g2,Lg1g1c3,basis);
Lg2g1g2c3=u2*liediff(g2,Lg1g2c3,basis);

Lg2g2fc3 =u2*liediff(g2,Lg2fc3,basis);
Lg2g2g1c3=u2*liediff(g2,Lg2g1c3,basis);
Lg2g2g2c3=u2*liediff(g2,Lg2g2c3,basis);

L3c3=Lfffc3+Lffg1c3+Lffg2c3+Lfg1fc3...
+Lfg1g1c3+Lfg1g2c3+Lfg2fc3+Lfg2g1c3+Lfg2g2c3 ...
+Lg1ffc3+Lg1fg1c3+Lg1fg2c3+Lg1g1fc3...
+Lg1g1g1c3+Lg1g1g2c3+Lg1g2fc3+Lg1g2g1c3...
+Lg1g2g2c3+Lg2ffc3+Lg2fg1c3+Lg2fg2c3...
+Lg2g1fc3+Lg2g1g1c3+Lg2g1g2c3+Lg2g2fc3...
+Lg2g2g1c3+Lg2g2g2c3;

% c4
```

```
% f
Lfffc4 =liediff(f,Lffc4,basis);
Lffg1c4=liediff(f,Lfg1c4,basis);
Lffg2c4=liediff(f,Lfg2c4,basis);

Lfg1fc4 =liediff(f,Lg1fc4,basis);
Lfg1g1c4=liediff(f,Lg1g1c4,basis);
Lfg1g2c4=liediff(f,Lg1g2c4,basis);

Lfg2fc4 =liediff(f,Lg2fc4,basis);
Lfg2g1c4=liediff(f,Lg2g1c4,basis);
Lfg2g2c4=liediff(f,Lg2g2c4,basis);
% g1
Lg1ffc4 =u1*liediff(g1,Lffc4,basis);
Lg1fg1c4=u1*liediff(g1,Lfg1c4,basis);
Lg1fg2c4=u1*liediff(g1,Lfg2c4,basis);

Lg1g1fc4 =u1*liediff(g1,Lg1fc4,basis);
Lg1g1g1c4=u1*liediff(g1,Lg1g1c4,basis);
Lg1g1g2c4=u1*liediff(g1,Lg1g2c4,basis);

Lg1g2fc4 =u1*liediff(g1,Lg2fc4,basis);
Lg1g2g1c4=u1*liediff(g1,Lg2g1c4,basis);
Lg1g2g2c4=u1*liediff(g1,Lg2g2c4,basis);
% g2
Lg2ffc4 =u2*liediff(g2,Lffc4,basis);
Lg2fg1c4=u2*liediff(g2,Lfg1c4,basis);
Lg2fg2c4=u2*liediff(g2,Lfg2c4,basis);
```

```
Lg2g1fc4 =u2*liediff(g2,Lg1fc4,basis);
Lg2g1g1c4=u2*liediff(g2,Lg1g1c4,basis);
Lg2g1g2c4=u2*liediff(g2,Lg1g2c4,basis);

Lg2g2fc4 =u2*liediff(g2,Lg2fc4,basis);
Lg2g2g1c4=u2*liediff(g2,Lg2g1c4,basis);
Lg2g2g2c4=u2*liediff(g2,Lg2g2c4,basis);

L3c4=Lfffc4+Lffg1c4+Lffg2c4+Lfg1fc4...
+Lfg1g1c4+Lfg1g2c4+Lfg2fc4+Lfg2g1c4...
+Lfg2g2c4+Lg1ffc4+Lg1fg1c4+Lg1fg2c4...
+Lg1g1fc4+Lg1g1g1c4+Lg1g1g2c4+Lg1g2fc4...
+Lg1g2g1c4+Lg1g2g2c4+Lg2ffc4+Lg2fg1c4...
+Lg2fg2c4+Lg2g1fc4+Lg2g1g1c4+Lg2g1g2c4...
+Lg2g2fc4+Lg2g2g1c4+Lg2g2g2c4;



% Fourth order Lie derivatives...

% c1 only
% ff
Lffffc1 =liediff(f,Lfffc1,basis);
Lfffg1c1=liediff(f,Lffg1c1,basis);
Lfffg2c1=liediff(f,Lffg2c1,basis);

Lffg1fc1 =liediff(f,Lfg1fc1,basis);
```

```
Lffg1g1c1=liediff(f,Lfg1g1c1,basis);
Lffg1g2c1=liediff(f,Lfg1g2c1,basis);

Lffg2fc1 =liediff(f,Lfg2fc1,basis);
Lffg2g1c1=liediff(f,Lfg2g1c1,basis);
Lffg2g2c1=liediff(f,Lfg2g2c1,basis);
% fg1
Lfg1ffc1 =liediff(f,Lg1ffc1,basis);
Lfg1fg1c1=liediff(f,Lg1fg1c1,basis);
Lfg1fg2c1=liediff(f,Lg1fg2c1,basis);

Lfg1g1fc1 =liediff(f,Lg1g1fc1,basis);
Lfg1g1g1c1=liediff(f,Lg1g1g1c1,basis);
Lfg1g1g2c1=liediff(f,Lg1g1g2c1,basis);

Lfg1g2fc1 =liediff(f,Lg1g2fc1,basis);
Lfg1g2g1c1=liediff(f,Lg1g2g1c1,basis);
Lfg1g2g2c1=liediff(f,Lg1g2g2c1,basis);
% fg2
Lfg2ffc1 =liediff(f,Lg2ffc1,basis);
Lfg2fg1c1=liediff(f,Lg2fg1c1,basis);
Lfg2fg2c1=liediff(f,Lg2fg2c1,basis);

Lfg2g1fc1 =liediff(f,Lg2g1fc1,basis);
Lfg2g1g1c1=liediff(f,Lg2g1g1c1,basis);
Lfg2g1g2c1=liediff(f,Lg2g1g2c1,basis);

Lfg2g2fc1 =liediff(f,Lg2g2fc1,basis);
```

```
Lfg2g2g1c1=liediff(f,Lg2g2g1c1,basis);
Lfg2g2g2c1=liediff(f,Lg2g2g2c1,basis);

% g1f
Lg1fffc1 =u1*liediff(g1,Lfffc1,basis);
Lg1ffg1c1=u1*liediff(g1,Lffg1c1,basis);
Lg1ffg2c1=u1*liediff(g1,Lffg2c1,basis);

Lg1fg1fc1 =u1*liediff(g1,Lfg1fc1,basis);
Lg1fg1g1c1=u1*liediff(g1,Lfg1g1c1,basis);
Lg1fg1g2c1=u1*liediff(g1,Lfg1g2c1,basis);

Lg1fg2fc1 =u1*liediff(g1,Lfg2fc1,basis);
Lg1fg2g1c1=u1*liediff(g1,Lfg2g1c1,basis);
Lg1fg2g2c1=u1*liediff(g1,Lfg2g2c1,basis);
% g1g1
Lg1g1ffc1 =u1*liediff(g1,Lg1ffc1,basis);
Lg1g1fg1c1=u1*liediff(g1,Lg1fg1c1,basis);
Lg1g1fg2c1=u1*liediff(g1,Lg1fg2c1,basis);

Lg1g1g1fc1 =u1*liediff(g1,Lg1g1fc1,basis);
Lg1g1g1g1c1=u1*liediff(g1,Lg1g1g1c1,basis);
Lg1g1g1g2c1=u1*liediff(g1,Lg1g1g2c1,basis);

Lg1g1g2fc1 =u1*liediff(g1,Lg1g2fc1,basis);
Lg1g1g2g1c1=u1*liediff(g1,Lg1g2g1c1,basis);
Lg1g1g2g2c1=u1*liediff(g1,Lg1g2g2c1,basis);
% g1g2
```

```
Lg1g2ffc1 =u1*liediff(g1,Lg2ffc1,basis);
Lg1g2fg1c1=u1*liediff(g1,Lg2fg1c1,basis);
Lg1g2fg2c1=u1*liediff(g1,Lg2fg2c1,basis);

Lg1g2g1fc1 =u1*liediff(g1,Lg2g1fc1,basis);
Lg1g2g1g1c1=u1*liediff(g1,Lg2g1g1c1,basis);
Lg1g2g1g2c1=u1*liediff(g1,Lg2g1g2c1,basis);

Lg1g2g2fc1 =u1*liediff(g1,Lg2g2fc1,basis);
Lg1g2g2g1c1=u1*liediff(g1,Lg2g2g1c1,basis);
Lg1g2g2g2c1=u1*liediff(g1,Lg2g2g2c1,basis);

% g2f
Lg2fffc1 =u2*liediff(g2,Lfffc1,basis);
Lg2ffg1c1=u2*liediff(g2,Lffg1c1,basis);
Lg2ffg2c1=u2*liediff(g2,Lffg2c1,basis);
Lg2fg1fc1 =u2*liediff(g2,Lfg1fc1,basis);
Lg2fg1g1c1=u2*liediff(g2,Lfg1g1c1,basis);
Lg2fg1g2c1=u2*liediff(g2,Lfg1g2c1,basis);
Lg2fg2fc1 =u2*liediff(g2,Lfg2fc1,basis);
Lg2fg2g1c1=u2*liediff(g2,Lfg2g1c1,basis);
Lg2fg2g2c1=u2*liediff(g2,Lfg2g2c1,basis);
% g2g1
Lg2g1ffc1 =u2*liediff(g2,Lg1ffc1,basis);
Lg2g1fg1c1=u2*liediff(g2,Lg1fg1c1,basis);
Lg2g1fg2c1=u2*liediff(g2,Lg1fg2c1,basis);
Lg2g1g1fc1 =u2*liediff(g2,Lg1g1fc1,basis);
Lg2g1g1g1c1=u2*liediff(g2,Lg1g1g1c1,basis);
```

```
Lg2g1g1g2c1=u2*liediff(g2,Lg1g1g2c1,basis);
Lg2g1g2fc1 =u2*liediff(g2,Lg1g2fc1,basis);
Lg2g1g2g1c1=u2*liediff(g2,Lg1g2g1c1,basis);
Lg2g1g2g2c1=u2*liediff(g2,Lg1g2g2c1,basis);
% g2g2
Lg2g2ffc1 =u2*liediff(g2,Lg2ffc1,basis);
Lg2g2fg1c1=u2*liediff(g2,Lg2fg1c1,basis);
Lg2g2fg2c1=u2*liediff(g2,Lg2fg2c1,basis);
Lg2g2g1fc1 =u2*liediff(g2,Lg2g1fc1,basis);
Lg2g2g1g1c1=u2*liediff(g2,Lg2g1g1c1,basis);
Lg2g2g1g2c1=u2*liediff(g2,Lg2g1g2c1,basis);
Lg2g2g2fc1 =u2*liediff(g2,Lg2g2fc1,basis);
Lg2g2g2g1c1=u2*liediff(g2,Lg2g2g1c1,basis);
Lg2g2g2g2c1=u2*liediff(g2,Lg2g2g2c1,basis);


L4c1=Lffffc1+Lfffg1c1+Lfffg2c1+Lffg1fc1...
+Lffg1g1c1+Lffg1g2c1+Lffg2fc1+Lffg2g1c1...
+Lffg2g2c1+Lfg1ffc1+Lfg1fg1c1+Lfg1fg2c1...
+Lfg1g1fc1+Lfg1g1g1c1+Lfg1g1g2c1+Lfg1g2fc1...
+Lfg1g2g1c1+Lfg1g2g2c1+Lfg2ffc1+Lfg2fg1c1...
+Lfg2fg2c1+Lfg2g1fc1+Lfg2g1g1c1+Lfg2g1g2c1...
+Lfg2g2fc1+Lfg2g2g1c1+Lfg2g2g2c1...
+Lg1fffc1+Lg1ffg1c1+Lg1ffg2c1+Lg1fg1fc1...
+Lg1fg1g1c1+Lg1fg1g2c1+Lg1fg2fc1+Lg1fg2g1c1...
+Lg1fg2g2c1+Lg1g1ffc1+Lg1g1fg1c1+Lg1g1fg2c1...
+Lg1g1g1fc1+Lg1g1g1g1c1+Lg1g1g1g2c1...
+Lg1g1g2fc1+Lg1g1g2g1c1+Lg1g1g2g2c1...
```

```
+Lg1g2ffc1+Lg1g2fg1c1+Lg1g2fg2c1+Lg1g2g1fc1...
+Lg1g2g1g1c1+Lg1g2g1g2c1+Lg1g2g2fc1...
+Lg1g2g2g1c1+Lg1g2g2g2c1+Lg2fffc1...
+Lg2ffg1c1+Lg2ffg2c1+Lg2fg1fc1...
+Lg2fg1g1c1+Lg2fg1g2c1+Lg2fg2fc1...
+Lg2fg2g1c1+Lg2fg2g2c1+Lg2g1ffc1...
+Lg2g1fg1c1+Lg2g1fg2c1+Lg2g1g1fc1...
+Lg2g1g1g1c1+Lg2g1g1g2c1+Lg2g1g2fc1...
+Lg2g1g2g1c1+Lg2g1g2g2c1+Lg2g2ffc1...
+Lg2g2fg1c1+Lg2g2fg2c1+Lg2g2g1fc1...
+Lg2g2g1g1c1+Lg2g2g1g2c1+Lg2g2g2fc1...
+Lg2g2g2g1c1+Lg2g2g2g2c1;

% Now define the y-dot residues

% y and yd have no residue

% ydd residues
yddc1=-(u1d/u1*Lg1c1+u2d/u2*Lg2c1);
yddc2=-(u1d/u1*Lg1c2+u2d/u2*Lg2c2);
yddc3=-(u1d/u1*Lg1c3+u2d/u2*Lg2c3);
yddc4=-(u1d/u1*Lg1c4+u2d/u2*Lg2c4);

% yddd residues

temp=u1dd/u1*Lg1c1+u2dd/u2*Lg2c1...
   +liediff(xd,u1d/u1*Lg1c1,basis)...
   +liediff(xd,u2d/u2*Lg2c1,basis) ...
```

```
   +u1d/u1*Lg1fc1+u2d/u2*Lg2fc1 ...
   +u1d/u1*Lfg1c1+u2d/u2*Lfg2c1 ...
   +u1d/u1*Lg1g1c1+u2d/u2*Lg1g2c1...
   +u1d/u1*Lg2g1c1+u2d/u2*Lg2g2c1 ...
   +u1d/u1*Lg1g1c1+u1d/u1*Lg1g2c1...
   +u2d/u2*Lg2g1c1+u2d/u2*Lg2g2c1;
ydddc1=-temp;

temp=u1dd/u1*Lg1c2+u2dd/u2*Lg2c2 ...
   +liediff(xd,u1d/u1*Lg1c2,basis)...
   +liediff(xd,u2d/u2*Lg2c2,basis) ...
   +u1d/u1*Lg1fc2+u2d/u2*Lg2fc2 ...
   +u1d/u1*Lfg1c2+u2d/u2*Lfg2c2 ...
   +u1d/u1*Lg1g1c2+u2d/u2*Lg1g2c2...
   +u1d/u1*Lg2g1c2+u2d/u2*Lg2g2c2 ...
   +u1d/u1*Lg1g1c2+u1d/u1*Lg1g2c2...
   +u2d/u2*Lg2g1c2+u2d/u2*Lg2g2c2;
ydddc2=-temp;

%Double checked.
temp=u1dd/u1*Lg1c3+u2dd/u2*Lg2c3 ...
   +liediff(xd,u1d/u1*Lg1c3,basis)...
   +liediff(xd,u2d/u2*Lg2c3,basis) ...
   +u1d/u1*Lg1fc3+u2d/u2*Lg2fc3 ...
   +u1d/u1*Lfg1c3+u2d/u2*Lfg2c3 ...
   +u1d/u1*Lg1g1c3+u2d/u2*Lg1g2c3...
   +u1d/u1*Lg2g1c3+u2d/u2*Lg2g2c3 ...
   +u1d/u1*Lg1g1c3+u1d/u1*Lg1g2c3...
```

```
   +u2d/u2*Lg2g1c3+u2d/u2*Lg2g2c3;
ydddc3=-temp;

temp=u1dd/u1*Lg1c4+u2dd/u2*Lg2c4 ...
   +liediff(xd,u1d/u1*Lg1c4,basis)...
   +liediff(xd,u2d/u2*Lg2c4,basis) ...
   +u1d/u1*Lg1fc4+u2d/u2*Lg2fc4 ...
   +u1d/u1*Lfg1c4+u2d/u2*Lfg2c4 ...
   +u1d/u1*Lg1g1c4+u2d/u2*Lg1g2c4...
   +u1d/u1*Lg2g1c4+u2d/u2*Lg2g2c4 ...
   +u1d/u1*Lg1g1c4+u1d/u1*Lg1g2c4...
   +u2d/u2*Lg2g1c4+u2d/u2*Lg2g2c4;
ydddc4=-temp;

%ydddd residues - only need to do c1,
% but it is quite large!

temp=u1ddd/u1*Lg1c1+u2ddd/u2*Lg2c1 ...
   +2*liediff(xd,u1dd/u1*Lg1c1,basis)...
   +2*liediff(xd,u2dd/u2*Lg2c1,basis) ...
   +liediff(xd,liediff(xd,u1d/u1*Lg1c1,basis),basis)...
   +liediff(xd,liediff(xd,u2d/u2*Lg2c1,basis),basis) ...
   +u1dd/u1*Lg1fc1 + u2dd/u2*Lg2fc1 ...
   +liediff(xd,u1d/u1*Lg1fc1,basis)...
   +liediff(xd,u2d/u2*Lg2fc1,basis) ...
   +u1dd/u1*Lfg1c1 + u2dd/u2*Lfg2c1 ...
   +liediff(xd,u1d/u1*Lfg1c1,basis)...
   +liediff(xd,u2d/u2*Lfg2c1,basis) ...
```

```
+2*u1d^2/u1^2*Lg1g1c2 +2*u1d*u2d/(u1*u2)*Lg1g2c2 ...
+2*u1d*u2d/(u1*u2)*Lg2g1c2 +2*u2d^2/u2^2*Lg2g2c2 ...
+u1dd/u1*Lg1g1c1+u2dd/u2*Lg1g2c1...
+u1dd/u1*Lg2g1c1+u2dd/u2*Lg2g2c1 ...
+u1d/u1*liediff(xd,Lg1g1c1,basis)...
+u2d/u2*liediff(xd,Lg1g2c1,basis)...
+u1d/u1*liediff(xd,Lg2g1c1,basis)...
+u2d/u2*liediff(xd,Lg2g2c1,basis) ...
+u1dd/u1*Lg1g1c1+u1dd/u1*Lg1g2c1...
+u2dd/u2*Lg2g1c1+u2dd/u2*Lg2g2c1 ...
+u1d/u1*liediff(xd,Lg1g1c1,basis)...
+u1d/u1*liediff(xd,Lg1g2c1,basis)
+u2d/u2*liediff(xd,Lg2g1c1,basis)...
+u2d/u2*liediff(xd,Lg2g2c1,basis) ...
+u1d/u1*Lffg1c1+u2d/u2*Lffg2c1 ...
+u1d/u1*Lfg1fc1+u2d/u2*Lfg2fc1 ...
+u1d/u1*Lg1ffc1+u2d/u2*Lg2ffc1 ...
+u1d/u1*Lfg1g1c1+u2d/u2*Lfg1g2c1...
+u1d/u1*Lfg2g1c1+u2d/u2*Lfg2g2c1 ...
+u1d/u1*Lg1fg1c1+u2d/u2*Lg1fg2c1...
+u1d/u1*Lg2fg1c1+u2d/u2*Lg2fg2c1 ...
+u1d/u1*Lg1g1fc1+u2d/u2*Lg1g2fc1...
+u1d/u1*Lg2g1fc1+u2d/u2*Lg2g2fc1 ...
+u1d/u1*Lfg1g1c1+u1d/u1*Lfg1g2c1...
+u2d/u2*Lfg2g1c1+u2d/u2*Lfg2g2c1 ...
+u1d/u1*Lg1fg1c1+u1d/u1*Lg1fg2c1...
+u2d/u2*Lg2fg1c1+u2d/u2*Lg2fg2c1 ...
+u1d/u1*Lg1g1fc1+u1d/u1*Lg1g2fc1...
```

```
   +u2d/u2*Lg2g1fc1+u2d/u2*Lg2g2fc1 ...
   +u1d/u1*Lg1g1g1c1+u2d/u2*Lg1g1g2c1...
   +u1d/u1*Lg1g2g1c1+u2d/u2*Lg1g2g2c1 ...
   +u1d/u1*Lg2g1g1c1+u2d/u2*Lg2g1g2c1...
   +u1d/u1*Lg2g2g1c1+u2d/u2*Lg2g2g2c1 ...
   +u1d/u1*Lg1g1g1c1+u1d/u1*Lg1g1g2c1...
   +u2d/u2*Lg1g2g1c1+u2d/u2*Lg1g2g2c1 ...
   +u1d/u1*Lg2g1g1c1+u1d/u1*Lg2g1g2c1...
   +u2d/u2*Lg2g2g1c1+u2d/u2*Lg2g2g2c1 ...
   +u1d/u1*Lg1g1g1c1+u1d/u1*Lg1g1g2c1...
   +u1d/u1*Lg1g2g1c1+u1d/u1*Lg1g2g2c1 ...
   +u2d/u2*Lg2g1g1c1+u2d/u2*Lg2g1g2c1...
   +u2d/u2*Lg2g2g1c1+u2d/u2*Lg2g2g2c1;
yddddc1=-temp;
```

---

liediff.m

```
function L=liediff(f,h,b);
% L=liediff(f,h,b) is the Lie Derivative of f in
% the direction of h.  b is the basis vector for
% the gradient.  f&b should be column vectors.

gf=gradf(h,b);
L=gf*f;
```

---

gradf.m

```
function G=gradf(fn,basis)

%G=grad(f,b) is the gradient of function f with respect
%to basis vector b.  f and b must be symbolic variables

n=length(basis);
G=[];
for ii=1:n
   bt=basis(ii);
   z=diff(fn,bt);
   G=[G,z];
end
```

A.2.3 The Robot_Constants and AR_Tests Files

The Robot_Constants and AR_Tests files translate the symbolic characters into the numerical realm to generate the test residuals.

Robot_Constants.m

```
%Robot physical constants
p1=3.473;
p2=0.193;
p3=0.242;
frd1=1.3;
frd2=0.88;
frs1=1.519;
frs2=0.932;
```

```
% Time variable
dt=0.001;%001
tmax=10;
deci=10;%10  %Decimation for controller

t=0:dt:tmax;

%PID control system parameters
fp1=5000;%100
fd1=70;%5
fi1=1;%1
fp2=2000;%20
fd2=8;%1
fi2=1;%1

q1des=(sin(t./0.3)).*0.5;
q2des=(cos(t./0.4)-1).*0.5;

%time of fault
tfault=6;
```

AR_Tests.m

```
% Nonlinear AR tests for 2D robot with redundant sensors

%reset sensor variables

clear r1 r1d r1dd r1ddd r1dddd tc1 tc1d tc1dd tc1ddd;
clear r2 r2d r2dd r2ddd r2dddd tc2 tc2d tc2dd tc2ddd;
```

```
clear u1 u1d u1dd u1ddd u2 u2d u2dd u2ddd;

syms r1 r1d r1dd r1ddd r1dddd tc1 tc1d tc1dd tc1ddd;
syms r2 r2d r2dd r2ddd r2dddd tc2 tc2d tc2dd tc2ddd;
syms u1 u1d u1dd u1ddd u2 u2d u2dd u2ddd;

%store undecimated values
tstore=t;
R1s=R1;
R2s=R2;
TC1s=TC1;
TC2s=TC2;
U1s=U1;
U2s=U2;

dts=dt;

%AR test decimation constant:
DDD=deci;

%time decimation
dt=dt*DDD;
t=deci2(t,DDD);

%Sensor decimation:
R1=deci2(R1,DDD);
R2=deci2(R2,DDD);
TC1=deci2(TC1,DDD);
```

```
TC2=deci2(TC2,DDD);
U1=deci2(U1,DDD);
U2=deci2(U2,DDD);

% Define "On"s as in document

O1=q1;
O2=L1c1;
O3=L2c1;
O4=L3c1;
O5=L4c1;
O6=q1d;
O7=L1c2;
O8=L2c2;
O9=L3c2;
O10=q2;
O11=L1c3;
O12=L2c3;
O13=L3c3;
O14=q2d;
O15=L1c4;
O16=L2c4;
O17=L3c4;

% Define Tests:

T1=O2-r1d;%trivial
T2=O3-r1dd+yddc1;
```

```
T3=O4-r1ddd+ydddc1;
T4=O5-r1dddd+yddddc1;
T5=O6-tc1;%trivial
T6=O7-tc1d;
T7=O8-tc1dd+yddc2;
T8=O9-tc1ddd+ydddc2;
T9=O10-r2;%trivial
T10=O11-r2d;%trivial
T11=O12-r2dd+yddc3;
T12=O13-r2ddd+ydddc3;
T13=O14-tc2;%trivial
T14=O15-tc2d;
T15=O16-tc2dd+yddc4;
T16=O17-tc2ddd+ydddc4;

% initialize test result vectors

T2R=[];T3R=[];T4R=[];T6R=[];T7R=[];T8R=[];
T11R=[];T12R=[];T14R=[];T15R=[];T16R=[];
R1dTC1R=[];R1ddTC1dR=[];R1dddTC1ddR=[];
R2dTC2R=[];R2ddTC2dR=[];

% create discrete derivative vectors

R1d=diff(R1)./dt;
R1dd=diff(R1d)./dt;
R1ddd=diff(R1dd)./dt;
R1dddd=diff(R1ddd)./dt;
```

```
TC1d=diff(TC1)./dt;
TC1dd=diff(TC1d)./dt;
TC1ddd=diff(TC1dd)./dt;

R2d=diff(R2)./dt;
R2dd=diff(R2d)./dt;
R2ddd=diff(R2dd)./dt;
R2dddd=diff(R2ddd)./dt;

TC2d=diff(TC2)./dt;
TC2dd=diff(TC2d)./dt;
TC2ddd=diff(TC2dd)./dt;

U1d=diff(U1)./dt;
U1dd=diff(U1d)./dt;

U2d=diff(U2)./dt;
U2dd=diff(U2d)./dt;

%adjust length
R1d=[R1d(1); R1d]';
R1dd=[R1dd(1); R1dd; R1dd(size(R1dd))]';
R1ddd=[R1ddd(1); R1ddd(1);
R1ddd; R1ddd(size(R1ddd))]';
R1dddd=[R1dddd(1); R1dddd(1); R1dddd;
R1dddd(size(R1dddd));R1dddd(size(R1dddd))]';
```

```
R2d=[R2d(1); R2d]';
R2dd=[R2dd(1); R2dd; R2dd(size(R2dd))]';
R2ddd=[R2ddd(1); R2ddd(1);
R2ddd; R2ddd(size(R2ddd))]';
R2dddd=[R2dddd(1); R2dddd(1); R2dddd;
R2dddd(size(R2dddd));R2dddd(size(R2dddd))]';

TC1d=[TC1d(1); TC1d]';
TC1dd=[TC1dd(1); TC1dd; TC1dd(size(TC1dd))]';
TC1ddd=[TC1ddd(1); TC1ddd(1);
TC1ddd; TC1ddd(size(TC1ddd))]';

TC2d=[TC2d(1); TC2d]';
TC2dd=[TC2dd(1); TC2dd; TC2dd(size(TC2dd))]';
TC2ddd=[TC2ddd(1); TC2ddd(1);
TC2ddd; TC2ddd(size(TC2ddd))]';

U1d=[U1d(1); U1d]';
U1dd=[U1dd(1); U1dd; U1dd(size(U1dd))]';

U2d=[U2d(1); U2d]';
U2dd=[U2dd(1); U2dd; U2dd(size(U2dd))]';


% Test loop:

for ii=1:length(R1);
   %grab current values
```

```
q1=R1(ii);q1d=TC1(ii);q1dd=TC1d(ii);
q1ddd=TC1dd(ii);q1dddd=TC1ddd(ii);
   %Note choice of sensor for qidd... is arbitrary
r1=R1(ii);r1d=R1d(ii);r1dd=R1dd(ii);
 r1ddd=R1ddd(ii);r1dddd=R1dddd(ii);
tc1=TC1(ii);tc1d=TC1d(ii);tc1dd=TC1dd(ii);
tc1ddd=TC1ddd(ii);
u1=U1(ii);u1d=U1d(ii);u1dd=U1dd(ii);
q2=R2(ii);q2d=TC2(ii);q2dd=TC2d(ii);
q2ddd=TC2dd(ii);q2dddd=TC2ddd(ii);
r2=R2(ii);r2d=R2d(ii);r2dd=R2dd(ii);
r2ddd=R2ddd(ii);
tc2=TC2(ii);tc2d=TC2d(ii);tc2dd=TC2dd(ii);
tc2ddd=TC2ddd(ii);
u2=U2(ii);u2d=U2d(ii);u2dd=U2dd(ii);

%solve tests
T2R=[T2R double(eval(T2))];
T3R=[T3R double(eval(T3))];
T4R=[T4R double(eval(T4))];
T6R=[T6R double(eval(T6))];
T7R=[T7R double(eval(T7))];
T8R=[T8R double(eval(T8))];
T11R=[T11R double(eval(T11))];
T12R=[T12R double(eval(T12))];
T14R=[T14R double(eval(T14))];
T15R=[T15R double(eval(T15))];
T16R=[T16R double(eval(T16))];
```

```
   R1dTC1R=[R1dTC1R r1d-tc1];
   R1ddTC1dR=[R1ddTC1dR r1dd-tc1d];
   R1dddTC1ddR=[R1dddTC1ddR r1ddd-tc1dd];
   R2dTC2R=[R2dTC2R r2d-tc2];
   R2ddTC2dR=[R2ddTC2dR r2dd-tc2d];

end;

beep;

t=tstore;  %restore originals
dt=dts;
R1=R1s;
R2=R2s;
TC1=TC1s;
TC2=TC2s;
U1=U1s;
U2=U2s;
```

A.3 Results

The symbolic Matlab files above were used to generate the missing NLAR tests (T_3, T_4, T_8, T_{12}, and T_{16}) from Chapter 5. The Matlab representation (using the variable definitions from the equations above) of these NLAR test residuals is reproduced below.

$T_3 =$

```
- (r1ddd*p3^4*cos(q2)^4 - u2*p2^2*frd1 + r1ddd*p1^2*p2^2
```

```
- p2^2*frd1^2*q1d + u1*frd2*p2^2 - 3*p3^2*q2d*q1d^2*p2^2
- 3*p3^3*q2d^3*cos(q2)^3*p2 + p3*q2d^3*cos(q2)*p2^3
+ 2*frd2^2*p3^2*cos(q2)^2*q2d + u1*frd2*p3^2*cos(q2)^2
- u2*frd2*p3^2*cos(q2)^2 + 2*r1ddd*p2^2*p3^2*cos(q2)^2
- u1d*p2*p3^2*cos(q2)^2 + u2d*p2*p3^2*cos(q2)^2
+ u2d*p3*cos(q2)*p2^2 + u2d*p2^3 - u1d*p2^3 + r1ddd*p2^4
+ p2^2*frd1*q2d*frd2 - frd2*p2^2*q1d*frd1
+ frd2^2*p2*q2d*p1 - u2*frd2*p2*p1 + u1*p2^2*frd1
- 2*r1ddd*p1*p2^3 + u1d*p2^2*p1 - u2d*p1*p2^2
+ u2d*p3^3*cos(q2)^3 + 3*p3^2*q2d*q1d^2*p1*p2
+ 2*frd2^2*p2*q2d*p3*cos(q2)
+ frd2*p3^2*cos(q2)*sin(q2)*q1d^2*p1
+ 2*frd2*p3^3*cos(q2)^2*sin(q2)*q1d^2
- frd2*p3^2*cos(q2)^2*q1d*frd1
- 2*p3*q2d^2*q1d*cos(q2)*p2^2*p1
+ 2*p3*q2d^2*q1d*cos(q2)*p2^3
- p3*q2d^3*cos(q2)*p2^2*p1
- 4*p3^2*q2d*q1d*cos(q2)*sin(q2)*p2*frd1
+ 3*p3*q2d^2*sin(q2)*frd2*p1*p2
+ 2*p3^3*q2d^2*sin(q2)*frd2*cos(q2)^2
- 6*p3^3*q2d^2*q1d*cos(q2)^3*p2
+ p3*q2d*cos(q2)*q1d^2*p2^3
 - p3*q2d*cos(q2)*q1d^2*p2^2*p1
- 3*p3^3*q2d*cos(q2)^3*q1d^2*p2
+ 4*p3^2*q2d*q1d^2*cos(q2)^2*p2^2
+ 2*p3*sin(q2)*p2^2*q2d*q1d*frd1
- 2*p3*sin(q2)*p2^2*q2d^2*frd2
+ 6*p3^2*sin(q2)*p2*q2d^2*frd2*cos(q2)
+ p2^2*frd1*p3*sin(q2)*q2d^2
```

```
+ p2*frd1*q2d*frd2*p3*cos(q2)
+ p3*sin(q2)*q1d^2*p2^2*frd1
+ 4*p3^2*sin(q2)*q1d*p2*q2d*frd2*cos(q2)
 - u2*frd2*p3*cos(q2)*p1 + frd2^2*p3*cos(q2)*q2d*p1
- 3*u2*p3*sin(q2)*q2d*p1*p2 + 3*u2*p3*sin(q2)*q2d*p2^2
- u2*p3^3*sin(q2)*q2d*cos(q2)^2
- 2*u2*p3^2*sin(q2)*q2d*cos(q2)*p2
+ 2*u1*frd2*p2*p3*cos(q2)
+ 2*p3*sin(q2)*q1d*p2*q2d*frd2*p1
+ frd2*p2*p3*sin(q2)*q1d^2*p1
+ 2*frd2*p2*p3^2*sin(q2)*q1d^2*cos(q2)
- 2*r1ddd*p1*p2*p3^2*cos(q2)^2 - u2d*p3*cos(q2)*p1*p2
+ 2*u2*p2*p3^2*sin(q2)*q1d*cos(q2)
+ 2*u2*p3^3*sin(q2)*q1d*cos(q2)^2
- u2*p2*frd1*p3*cos(q2) - 2*u2*p3*sin(q2)*q1d*p2*p1
- u2*frd2*p2*p3*cos(q2) - 2*frd2*p2*q1d*frd1*p3*cos(q2)
- 2*p3^2*q1d^3*p2^2 - 2*p3^4*q1d^3*cos(q2)^2
+ 2*p3^4*q1d^3*cos(q2)^4 + 2*p3^2*q1d^3*p2^2*cos(q2)^2
+ 4*p3^3*q2d^3*cos(q2)*p2 + p3^4*q2d*q1d^2*cos(q2)^2
+ 8*p3^3*q1d*cos(q2)*p2*q2d^2 + 4*p3^3*q1d^2*cos(q2)*p2*q2d
+ 2*p3^2*q1d^3*p2*p1 - 2*p3^2*q1d^3*p2*p1*cos(q2)^2
+ p3^2*sin(q2)*q1d^2*cos(q2)*p2*frd1
- 4*p3^2*q2d*q1d^2*cos(q2)^2*p1*p2
+ 4*u1*p2*p3^2*cos(q2)*sin(q2)*q2d
+ 2*u2*p3*sin(q2)*q1d*p2^2)/( - p1*p2 + p2^2
+ p3^2*cos(q2)^2)^2
```

$T_4 =$

```
(14*u1*p3^3*q2d*q1d*cos(q2)^3*p2^2
```

```
- 6*u1*p3^2*sin(q2)*p2^2*q2d*cos(q2)*frd2
+ 4*u1*p3^2*q2d*q1d*cos(q2)^2*p2^3
- 4*u2*p3^3*cos(q2)^2*frd2*sin(q2)*q1d*p2
- 6*u2*p3*q2d*q1d*cos(q2)*p1^2*p2^2
- 2*u1*p3*q2d*q1d*cos(q2)*p2^4
+ 6*u1*p3*sin(q2)*q2d*frd2*p2^3
- 4*u1*frd2*p2^2*frd1*p3*cos(q2)
+ u1*p3*q2d^2*cos(q2)*p2^3*p1
- 2*u1*frd2^2*p2^2*p3*cos(q2)
- 2*u1*frd2*p2*frd1*p3^2*cos(q2)^2
- u1*p3*q1d^2*cos(q2)*p2^4
+ 8*u1*p3^3*q1d^2*cos(q2)^3*p1*p2
- u1*p3^2*q1d^2*cos(q2)^2*p2^2*p1
- 2*u1*frd2^2*p2*p1*p3*cos(q2)
- 2*u2*frd2*p1*p3^3*sin(q2)*cos(q2)^2*q1d
+ u2*frd2*p1*p2*frd1*p3*cos(q2)
- 6*u1*p3^4*sin(q2)*cos(q2)^3*q2d*frd2
- u1*p3^3*q1d^2*cos(q2)^3*p2^2
+ u1*p3*q1d^2*cos(q2)*p2^3*p1
- 3*r1dddd*p2^4*p3^2*cos(q2)^2
- 3*r1dddd*p1^2*p2^2*p3^2*cos(q2)^2
- 4*p3^3*p1*p2^2*q2d^4*sin(q2)
- 4*u2d*p3*sin(q2)*q1d*p2^3*p1
- r1dddd*p3^6*cos(q2)^6
+ 20*u1*p3^4*q2d*q1d*cos(q2)^4*p2
- 22*u1*p2*p3^4*cos(q2)^2*q1d*q2d
- u1*p3^5*cos(q2)^3*q1d^2
- 28*u1*p2*p3^4*cos(q2)^2*q2d^2
+ 7*u1*p3^3*q2d^2*cos(q2)^3*p2^2
```

```
+ 2*u2*frd2*p1*p3^2*sin(q2)*cos(q2)*q1d*p2
+ 28*p3^4*cos(q2)^2*q1d*p2*frd1*q2d^2
+ 2*u2d*p3*sin(q2)*q1d*p2^4
- 8*u1*p2^2*p3^3*cos(q2)*q2d^2
+ 2*u2d*p3*sin(q2)*q1d*p2^2*p1^2
+ 5*p3^2*q2d^2*q1d^2*cos(q2)*p2^4*sin(q2)
+ p3*q2d*q1d^2*cos(q2)*p2^4*frd1
- 14*p3^2*q2d^2*q1d*cos(q2)^2*p2^3*frd2
- 44*p3^5*cos(q2)^2*q1d^3*p2*q2d*sin(q2)
- 8*p3*q2d^2*q1d*cos(q2)*p2^3*frd2*p1
- u2*p3^4*sin(q2)*cos(q2)^3*q1d*frd1
- 6*p3^5*cos(q2)^3*q2d^2*frd2*q1d
- 19*u1*p2*p3^4*cos(q2)^2*q1d^2
+ 5*p3^4*cos(q2)*q1d^2*p2^2*q2d^2*sin(q2)
- 2*frd2*p2^2*p3^2*q1d^3*p1*cos(q2)^2
+ 2*frd2*p2^2*p3^2*q1d^3*p1
+ 4*frd2*p2^2*p3^3*cos(q2)*q1d^3
- 4*frd2*p2^2*p3^3*cos(q2)^3*q1d^3
+ 2*p3*q1d^2*cos(q2)*p1^2*p2^2*q2d*frd2
- 8*frd2*p2^3*frd1*q1d*p3*sin(q2)*q2d
+ 2*u1dd*p2^3*p3^2*cos(q2)^2
- 8*p3^3*q1d^4*cos(q2)^2*p1*p2^2*sin(q2)
- p3*q1d^3*cos(q2)*p1*p2^3*frd1
+ 3*p3^2*q1d^3*cos(q2)^2*p1*p2^2*frd1
- 7*u2*p3^3*q1d^2*cos(q2)^3*p1*p2
+ u2*p3^3*cos(q2)^3*frd2*frd1
+ 4*p3^3*q1d^4*cos(q2)^2*p1^2*p2*sin(q2)
+ 12*p3^2*q1d^2*cos(q2)^2*p1^2*p2*q2d*frd2
+ p3^2*q1d^4*cos(q2)*p1^2*p2^2*sin(q2)
```

```
- 12*p3^2*q1d^3*cos(q2)*p1*p2^3*sin(q2)*q2d
- 8*p3^3*q1d^3*cos(q2)^3*p1*p2*frd1
+ 36*p3^3*q1d^2*cos(q2)^3*p1*p2*q2d*frd2
- 20*p3^3*q1d^2*cos(q2)^3*p2^2*q2d*frd2
+ 36*p3^5*q1d^3*cos(q2)^4*sin(q2)*p2*q2d
+ u1dd*p2*p3^4*cos(q2)^4
4*u2*p3^2*cos(q2)^2*frd2*frd1*p2
+ 10*u2*p3*q2d*q1d*cos(q2)*p1*p2^3
+ 6*p3^4*q1d^2*cos(q2)^4*p1*q2d*frd2
4*p3^3*q1d^3*cos(q2)^2*p2^3*sin(q2)*q2d
- 2*p3^4*q1d^4*cos(q2)^3*p2^2*sin(q2)
- 3*p3^2*q1d^3*cos(q2)^2*p2^3*frd1
+ p3^3*q1d^3*cos(q2)^3*p2^2*frd1
- 19*p3^4*q1d^3*cos(q2)^4*p2*frd1
+ 5*p2^3*frd1*p3^2*q1d^3
+ 18*u2*p3^4*cos(q2)^2*q1d^2*p2
- 4*p3^2*q1d^2*p2^3*q2d*frd2
- 26*p3^3*q2d^3*cos(q2)^2*p2^3*q1d*sin(q2)
+ 11*p3^2*q2d^2*cos(q2)^2*p2^3*q1d*frd1
+ 20*p3^2*q2d^3*cos(q2)^2*p2^2*frd2*p1
- 16*p3^2*q2d^2*cos(q2)*p2^3*sin(q2)*q1d^2*p1
- 3*p3^6*cos(q2)^3*q1d^2*q2d^2*sin(q2)
- 3*p3*sin(q2)*p2^2*q2d^2*frd1*frd2*p1
- 14*p3^3*q2d^3*cos(q2)^3*p2^2*frd2
+ 16*p3^5*q1d^2*cos(q2)^5*q2d*frd2
+ 58*p3^4*q2d^2*q1d*cos(q2)^4*p2*frd2
- 2*p3*sin(q2)*p2^3*q2d*frd1^2*q1d
- 14*p3^2*sin(q2)*p2^2*q2d^2*frd1*cos(q2)*frd2
- 3*p3^4*q2d^2*cos(q2)^3*p2^2*sin(q2)*q1d^2
```

```
- 15*p3^3*q2d^4*cos(q2)^2*p2^3*sin(q2)
- 9*p3^3*sin(q2)*cos(q2)^2*q2d^2*p2*frd1*frd2
+ 10*p3^2*sin(q2)*cos(q2)*q2d*p2^2*frd1^2*q1d
+ 30*p3^3*q2d^3*cos(q2)^3*p2*frd2*p1
+ 40*p3^4*q2d^3*cos(q2)^4*p2*frd2
6*p3^5*q2d^3*cos(q2)^5*frd2
+ 48*p3^5*q2d^2*cos(q2)^4*p2*sin(q2)*q1d^2
+ 12*p3^4*q2d^3*cos(q2)^3*p2^2*q1d*sin(q2)
+ 11*p3^3*q2d^2*cos(q2)^3*p2^2*q1d*frd1
+ 14*u2*p3^2*q1d^2*cos(q2)^2*p1*p2^2
+ 21*p3^4*q2d^2*cos(q2)^3*p2*sin(q2)*q1d^2*p1
+ 3*p3^4*q2d^4*cos(q2)^3*p2^2*sin(q2)
- p3*cos(q2)*frd2^2*p2*frd1*q2d*p1
- 5*p3^2*cos(q2)^2*frd2^2*p2*frd1*q2d
+ 4*p3*sin(q2)*q2d^2*frd2^2*p2^2*p1
+ 14*p3^2*sin(q2)*q2d^2*frd2^2*p2^2*cos(q2)
+ 4*p3*cos(q2)*frd2*p2^2*frd1^2*q1d
+ 11*u2*p3^5*cos(q2)^3*q1d^2
+ frd2*p2^3*frd1*p3*sin(q2)*q2d^2
- 5*frd2^2*p2^2*frd1*q2d*p3*cos(q2)
- u2d*p3^4*cos(q2)^4*frd2
- 12*p3^2*q1d^2*cos(q2)^2*p1*p2^2*q2d*frd2
+ 5*p3^4*q1d^4*cos(q2)^3*p1*p2*sin(q2)
+ 6*p3^4*sin(q2)*cos(q2)^3*q2d*frd2*q1d*frd1
- 8*p3^5*cos(q2)^3*q2d^3*frd2
- 7*p3^3*q2d^2*cos(q2)^2*p2^3*sin(q2)*q1d^2
- p3^2*q2d^4*cos(q2)*p2^4*sin(q2)
- 20*p3^2*q2d^3*cos(q2)^2*p2^3*frd2
- u2dd*p3^5*cos(q2)^5
```

```
+ 8*p3^5*q2d^2*q1d*cos(q2)^5*frd2
+ 26*p3^3*q2d^3*cos(q2)^2*p1*p2^2*q1d*sin(q2)
- p2^3*frd1^2*p3*sin(q2)*q1d^2
+ 4*p3*sin(q2)*p1*q2d*frd2*p2^2*q1d*frd1
- 4*p3^3*cos(q2)^3*frd2^3*q2d
- 14*p3^4*q2d*q1d^2*cos(q2)^4*p2*frd1
- 4*p3^2*q2d^3*cos(q2)*p2^4*q1d*sin(q2)
- p3*q2d^2*cos(q2)*p2^4*q1d*frd1
- 10*p3*q2d^3*cos(q2)*p2^3*frd2*p1
- 2*p3^2*sin(q2)*cos(q2)*q2d*frd2*p2^2*q1d*frd1
- 11*p3^2*q2d^2*cos(q2)^2*p1*p2^2*q1d*frd1
+ 2*p3^3*p2^3*q2d^3*q1d*sin(q2)
+ 8*p3^2*q2d*q1d^2*cos(q2)^2*p1*p2^2*frd1
- 4*u2*p3^4*cos(q2)^3*frd2*sin(q2)*q1d
+ 15*p3^3*q2d^4*cos(q2)^2*p1*p2^2*sin(q2)
+ 8*p3^4*q2d*q1d^3*cos(q2)^3*p1*p2*sin(q2)
+ 6*p3*q2d^3*cos(q2)*p2^2*p1^2*frd2
+ 8*p3^2*q2d*q1d^3*cos(q2)*p1^2*p2^2*sin(q2)
- 4*p3^3*q2d*q1d^3*cos(q2)^2*p1*p2^2*sin(q2)
+ 6*p3^2*q2d^2*q1d*p2^3*frd2
+ 11*p3^3*p2^2*q2d^2*q1d^2*p1*sin(q2)
- 20*p3^4*cos(q2)*q1d^3*p2*q2d*p1*sin(q2)
+ 11*p3^2*q2d^2*cos(q2)*p2^2*p1^2*sin(q2)*q1d^2
+ p3*q2d^2*cos(q2)*p2^3*p1*q1d*frd1
- 12*p3^3*q1d^3*p2^3*q2d*sin(q2)
- 14*u2*p3^2*p1*p2^2*q1d^2
+ 22*p3^3*q2d^2*q1d*cos(q2)^3*p1*p2*frd2
+ p3^2*q2d^4*cos(q2)*p2^3*p1*sin(q2)
- u2*p3*sin(q2)*p2^3*q1d*frd1
```

```
- 7*u2*p3^2*sin(q2)*p1*p2*q1d*cos(q2)*frd1
+ 2*u2*p3^3*cos(q2)^3*frd2^2
+ 14*p3^2*q1d^2*p1*p2^2*q2d*frd2
- 5*p3^2*q1d^3*p1*p2^2*frd1
+ 7*p3^3*p2^2*q2d^2*q1d^2*p1*sin(q2)*cos(q2)^2
+ 6*u2*p3^2*p1^2*p2*q1d^2
- 4*p3^2*p2^3*q2d^2*q1d*frd1
+ 6*p3^6*q1d^4*cos(q2)^5*sin(q2)
+ p3^5*cos(q2)^3*q1d^3*frd1
- 10*u2*p3^5*q1d^2*cos(q2)^5 + 10*p3^2*q2d^3*p2^3*frd2
+ p3*q2d^4*sin(q2)*p2^5
- 32*frd2^2*p1*p3^2*cos(q2)*sin(q2)*p2*q2d^2
- 8*frd2^2*p1*p3^3*cos(q2)^2*sin(q2)*q2d^2
- 14*p3^4*cos(q2)^3*frd2^2*sin(q2)*q2d^2
- 4*p3^2*cos(q2)^2*frd2^3*q2d*p1
- frd2^2*p1^2*p3^2*cos(q2)*sin(q2)*q1d^2
- 30*p3^3*cos(q2)^2*frd2^2*sin(q2)*p2*q2d^2
- 8*frd2^2*p1*p3^2*cos(q2)*q1d*sin(q2)*p2*q2d
+ 2*frd2^2*p1*p3*cos(q2)*q1d*p2*frd1
+ frd2^2*p1*p3^2*cos(q2)^2*q1d*frd1
+ 12*p3^2*sin(q2)*cos(q2)*q1d*p2*frd1*q2d*frd2*p1
+ 28*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1*q2d*frd2
- 4*p3^4*cos(q2)^3*frd2^2*sin(q2)*q1d^2
- 8*p3^3*cos(q2)^2*frd2^2*q1d*sin(q2)*p2*q2d
+ 4*p3^2*cos(q2)^2*frd2^2*q1d*p2*frd1
+ 2*p3^3*cos(q2)^3*frd2^2*q1d*frd1
- frd2*p1*p3^2*sin(q2)*cos(q2)*q1d^2*p2*frd1
+ 2*p3^2*cos(q2)^2*frd2*frd1^2*q1d*p2
+ 10*p3^3*q2d^2*q1d*cos(q2)^3*p2^2*frd2
```

```
- p3^3*cos(q2)^3*frd2^2*frd1*q2d
- 5*p3^3*cos(q2)^2*frd2*sin(q2)*q1d^2*p2*frd1
- 4*p3^3*cos(q2)^2*frd2^2*sin(q2)*q1d^2*p1
+ 14*p3^2*q2d^2*q1d*cos(q2)^2*p1*p2^2*frd2
+ 4*p3^4*q2d*q1d^3*cos(q2)^3*p2^2*sin(q2)
- 8*p3^2*q2d*q1d^2*cos(q2)^2*p2^3*frd1
- 5*p3^3*q2d*q1d^2*cos(q2)^3*p2^2*frd1
+ 2*u2*p3^4*p1*cos(q2)^2*q1d^2
- p3*q2d*q1d^2*cos(q2)*p1*p2^3*frd1
+ 6*p3*q2d^2*q1d*cos(q2)*p1^2*p2^2*frd2
- 10*p3^2*p1*q2d^3*frd2*p2^2
+ p3*q2d^3*p2^3*frd1*cos(q2)*p1
+ 9*p3^3*q2d^3*p2^2*frd1*cos(q2)^3
- 4*p3*q2d^3*q1d*sin(q2)*p2^4*p1
+ p3*q2d^4*sin(q2)*p2^3*p1^2
- p2^3*frd1^2*p3*sin(q2)*q2d^2
- p2^2*frd1^2*q2d*p3*cos(q2)*frd2
+ 2*p3*q2d^3*q1d*sin(q2)*p2^3*p1^2
- 3*u2^2*p3^2*sin(q2)*cos(q2)*p1*p2
+ 4*u2d*p3^2*sin(q2)*q2d*p1*cos(q2)*p2^2
- u2^2*p3^2*sin(q2)*cos(q2)*p2^2
+ 4*u2d*p3*sin(q2)*q2d*p1^2*p2^2
- 8*u2d*p3*sin(q2)*q2d*p1*p2^3
+ 4*u2d*p3*sin(q2)*q2d*p2^4
- 2*u2d*p3^5*sin(q2)*q2d*cos(q2)^4
+ 2*u1d*p3*cos(q2)*frd2*p2^3
+ 2*u1d*p3^3*cos(q2)^3*frd2*p2
- 6*u1d*p3^2*sin(q2)*p2^2*q2d*cos(q2)*p1
+ 6*u1d*p3^2*sin(q2)*p2^3*q2d*cos(q2)
```

```
+ 6*u1d*p3^4*sin(q2)*p2*q2d*cos(q2)^3
+ 2*u1d*p3^2*cos(q2)^2*frd2*p2^2
- 2*u1dd*p1*p2^2*p3^2*cos(q2)^2
+ u1d*p2^2*frd1*p3^2*cos(q2)^2
+ 2*u2dd*p3*cos(q2)*p1*p2^3
+ 2*u2dd*p3^3*cos(q2)^3*p1*p2
+ 2*u2dd*p1*p2^2*p3^2*cos(q2)^2
- u2d*frd2*p1*p3^3*cos(q2)^3
- u2d*p3^2*cos(q2)^2*frd2*p2^2
+ 2*u2d*p3^5*sin(q2)*cos(q2)^4*q1d
- u2dd*p3*cos(q2)*p1^2*p2^2
- u1d*frd2*p1*p2*p3^2*cos(q2)^2
- 2*u1d*frd2*p1*p3*cos(q2)*p2^2
- u2d*p2^2*frd1*p3^2*cos(q2)^2
- u2d*p2^3*frd1*p3*cos(q2)
- u2d*p2*frd1*p3^3*cos(q2)^3
+ 4*p3^6*cos(q2)^3*q1d^3*q2d*sin(q2)
- 4*u2d*p3^2*sin(q2)*p2^3*q2d*cos(q2)
- 4*u2d*p3^4*sin(q2)*p2*q2d*cos(q2)^3
+ u1d*p3^4*cos(q2)^4*frd2
+ 2*u2d*p3^3*sin(q2)*cos(q2)^2*q2d*p2^2
+ u2d*frd2*p1^2*p3*cos(q2)*p2
+ 4*u2d*p3^3*sin(q2)*q1d*p2^2*cos(q2)^2
- 2*u2d*p3^3*sin(q2)*cos(q2)^2*q2d*p1*p2
- 4*u2d*p3^3*sin(q2)*q1d*p1*p2*cos(q2)^2
+ 2*u2d*p3^2*sin(q2)*cos(q2)*q1d*p2^3
+ 2*u2d*p3^4*sin(q2)*cos(q2)^3*q1d*p2
+ u2d*p2^2*frd1*p3*cos(q2)*p1
- 2*u2d*p3^2*sin(q2)*cos(q2)*q1d*p2^2*p1
```

```
- 2*u2*p3^5*q2d^2*cos(q2)^5
- 11*u2*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1
+ u2*p3*sin(q2)*p1*p2^2*q1d*frd1
+ 10*u2*p3^4*q2d*q1d*cos(q2)^2*p2
+ 4*p3^4*q1d^3*p2^2*q2d*cos(q2)*sin(q2)
- 4*u2*p3^3*q2d*q1d*p2^2*cos(q2)
+ 16*p3^3*cos(q2)*q2d^3*frd2*p2^2
+ 29*u2*p3^3*q2d^2*p1*p2*cos(q2)
- 6*u2*p3^2*q2d*q1d*p1*p2^2
+ 6*u2*p3^2*q2d*q1d*p2^3
+ 20*u2*p3^3*q2d*q1d*cos(q2)*p1*p2
+ 17*u2*p3^3*q2d^2*cos(q2)^3*p2^2
- 21*u2*p3^3*q2d^2*cos(q2)*p2^2
+ 3*u2*p3^2*q2d^2*p1*p2^2
- 4*u2*p3^5*q2d*q1d*cos(q2)^3
+ 6*u2*p3^2*q2d^2*cos(q2)^2*p2^3
- 10*u2*p3^3*q2d*q1d*cos(q2)^3*p1*p2
- 10*p3^2*p1^2*q2d*frd2*p2*q1d^2
+ 3*u1*p3^2*p1*p2^2*q1d^2
+ 9*u1*u2*p3^3*sin(q2)*p2*cos(q2)^2
- 2*u1*p3^3*cos(q2)^3*frd2^2
+ 6*u1*p3^2*p1*p2^2*q1d*q2d
- 6*u2*p3^2*q2d^2*cos(q2)^2*p1*p2^2
+ 7*u1*u2*p3^2*sin(q2)*p2*cos(q2)*p1
- 24*u2*p3^3*q2d^2*cos(q2)^3*p1*p2
+ 9*p2^3*frd1*p3^2*q1d^2*q2d
- 14*p3^5*cos(q2)^2*q1d^4*p2*sin(q2)
+ 10*u2*p3^4*q2d*sin(q2)*frd2*cos(q2)^3
- 2*u2^2*p3*sin(q2)*p2^3
```

```
+ 15*u2*p3^4*q2d^2*cos(q2)^2*p2

- 6*u2*p3*q2d^2*cos(q2)*p2^2*p1^2

+ 10*u2*p3*q2d*sin(q2)*frd2*p1^2*p2

- 12*u2*p3^4*q2d*q1d*cos(q2)^4*p2

- 12*u2*p3^4*q2d^2*cos(q2)^4*p2

+ 8*u2*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1

- 8*u2*p3^2*q2d*q1d*cos(q2)^2*p2^3

- u2dd*p3^4*cos(q2)^4*p2 - 4*u1*p3^2*p1*p2^2*q2d^2

- 3*u1*p3^2*p2^3*q1d^2 - 6*u1*p3^2*p2^3*q1d*q2d

+ 7*u2*p3^3*q2d*cos(q2)^2*sin(q2)*p2*frd1

- 2*u2dd*p3^2*cos(q2)^2*p2^3

+ u2*p3^3*cos(q2)*q1d^2*p2^2

- 3*r1dddd*p2^2*p3^4*cos(q2)^4

+ 8*u2*p3^3*q2d*cos(q2)^2*sin(q2)*frd2*p1

- 4*frd2^2*p2*p1*p3^2*cos(q2)*sin(q2)*q1d^2

- 11*u2*p3*q2d*sin(q2)*frd2*p2^2*p1

- u2*p3*q1d^2*cos(q2)*p1^2*p2^2

+ u2*p2^2*frd1^2*p3*cos(q2)

+ 4*u2*p3*cos(q2)*frd2*p2^2*frd1

- 7*u2*p3^2*q1d^2*cos(q2)^2*p1^2*p2

- u2*p3^4*q1d^2*cos(q2)^4*p1

+ 3*u2*p3^2*cos(q2)^2*frd2^2*p1

+ 15*p3^4*cos(q2)^2*q1d^2*p2*frd1*q2d

+ u2*p3*q2d*sin(q2)*frd2*p2^3

+ 8*u2*p3^2*q2d*q1d*cos(q2)^2*p1*p2^2

+ 11*u2*p3*q2d^2*cos(q2)*p2^3*p1

- 2*u2dd*p3^3*cos(q2)^3*p2^2

+ 4*u1*p3^2*p2^3*q2d^2

- 4*frd2^3*p2*p3^2*cos(q2)^2*q2d
```

```
- 3*u2*p3*q2d*sin(q2)*frd1*p2^3
+ 3*u2*p3*q2d*sin(q2)*frd1*p1*p2^2
- 5*u2*p3*q2d^2*cos(q2)*p2^4
- 3*u2*p3^2*q2d^2*p2^3
+ 6*p3^3*cos(q2)*q1d^2*p2^2*frd1*q2d
+ p3*q2d^2*sin(q2)*q1d^2*p2^5
- 3*u2^2*p3*sin(q2)*p1^2*p2
- u2^2*p3^3*sin(q2)*p1*cos(q2)^2
- 4*frd2^3*p2*p1*q2d*p3*cos(q2)
+ 19*u2*p3^3*sin(q2)*cos(q2)^2*q2d*frd2*p2
- 6*u2^2*p3^3*sin(q2)*p2*cos(q2)^2
- 30*p3^3*p1*q2d^2*cos(q2)*frd2*q1d*p2
- 12*u2*p3^2*q2d*sin(q2)*frd2*p2^2*cos(q2)
+ 5*u2^2*p3*sin(q2)*p1*p2^2
+ 28*u2*p3^2*q2d*sin(q2)*frd2*p1*p2*cos(q2)
- frd2^2*p2*p1^2*p3*sin(q2)*q1d^2
- p3^2*q1d^4*cos(q2)*p2^3*sin(q2)*p1
+ 7*p3^3*cos(q2)*q1d^3*p2*frd1*p1
- u2dd*p3*cos(q2)*p2^4 + 2*p3*q2d^2*frd2^2*p2^3*sin(q2)
+ 2*frd2^2*p2^2*p3*cos(q2)*q1d*frd1
- 4*p3^4*cos(q2)*p2^2*q2d^4*sin(q2)
- 4*frd2^2*p2*p3^3*cos(q2)^2*sin(q2)*q1d^2
- 12*p3^3*cos(q2)*q1d*p2^2*frd1*q2d^2
- 3*u2*p3^2*sin(q2)*cos(q2)*q1d*p2^2*frd1
- 11*p3^3*q1d^2*p2^3*q2d^2*sin(q2)
+ 8*u2*p3^2*q1d^2*p2^3
+ 21*p3^4*cos(q2)^2*q1d^3*p2*frd1
+ 36*p3^4*q2d*frd2*p2*q1d^2*cos(q2)^4
+ 4*frd2*p2*p3^3*q1d^3*p1*cos(q2)^3
```

```
- 4*frd2*p2*p3^3*q1d^3*p1*cos(q2)
+ u2*frd2*p2^3*frd1 - frd2^2*p2^3*frd1*q2d
- frd2^3*p2*p1^2*q2d - u1*frd2^2*p2^2*p1
+ u2*frd2^2*p1^2*p2
- 67*p3^5*cos(q2)^2*p2*q2d^2*q1d^2*sin(q2)
- 34*p3^3*cos(q2)*q2d*frd2*p2*q1d^2*p1
- 50*p3^4*cos(q2)^2*p2*q2d^3*frd2
- 20*p3^5*cos(q2)^3*q2d*frd2*q1d^2
- 8*p3^4*cos(q2)^2*q2d*frd2*q1d^2*p1
+ 7*u2*p3^3*p1*p2*cos(q2)*q1d^2
+ frd2^2*p2^2*p1*q1d*frd1
- 16*p3^4*cos(q2)*p2^2*q2d^3*q1d*sin(q2)
+ p2^3*frd1^3*q1d + u1d*p2^4*frd1
- u2d*frd2*p2^3*p1 + r1dddd*p1^3*p2^3
- 3*r1dddd*p1^2*p2^4 + 3*r1dddd*p1*p2^5
+ u1d*frd2*p2^4 - r1dddd*p2^6 - u2dd*p2^5
+ u2*frd2*p1*p2^2*frd1 + u2*p2^3*frd1^2
- frd2^2*p2^2*frd1*q2d*p1
+ 4*p3^2*p1*q1d*p2^2*frd1*q2d^2
- 6*p3^2*p1*q2d^2*frd2*p2^2*q1d
+ 2*frd2*p2^3*frd1^2*q1d - p2^3*frd1^2*q2d*frd2
- 2*u1*frd2*p2^3*frd1 - u1*p3^3*q1d^2*p2^2*cos(q2)
- u2dd*p1^2*p2^3 + 2*u2dd*p1*p2^4 - u2d*p2^4*frd1
+ u1dd*p1^2*p2^3 - 2*u1dd*p1*p2^4 - u1d*p2^3*frd1*p1
- u1d*frd2*p1*p2^3 + u2d*p2^3*frd1*p1
+ u2d*frd2*p1^2*p2^2 + u1dd*p2^5
- u1*p2^3*frd1^2
- 5*p3^2*sin(q2)*cos(q2)*q1d^2*p2^2*frd1^2
- 21*p3^4*q2d^2*cos(q2)^4*q1d*p2*frd1
```

```
- 4*p3^3*sin(q2)*cos(q2)^2*q1d^2*p2*frd1^2
+ 12*p3^5*q2d^4*cos(q2)^4*sin(q2)*p2
+ p3*q2d^2*sin(q2)*q1d^2*p1^2*p2^3
- 2*p3*q2d^2*sin(q2)*q1d^2*p1*p2^4
+ 4*p3*q2d^3*cos(q2)*p2^4*frd2
+ 8*u1*p2^2*p3^2*cos(q2)*sin(q2)*q1d*frd1
- 30*u1*p2*p3^3*cos(q2)^2*sin(q2)*q2d*frd2
+ 8*u1*p2*p3^3*cos(q2)^2*sin(q2)*q1d*frd1
+ 9*p2^2*p3^3*q1d^4*p1*sin(q2)
+ 4*p3^5*cos(q2)^3*frd2*q1d^3
- 4*p3^5*cos(q2)^5*frd2*q1d^3
- 6*p3^3*q1d^4*p2^3*sin(q2)
+ 2*p3^4*cos(q2)^2*frd2*q1d^3*p1
- 2*p3^4*cos(q2)^4*frd2*q1d^3*p1
- 4*p2*p3^4*q1d^4*p1*cos(q2)*sin(q2)
- 2*p3*q2d^4*sin(q2)*p2^4*p1
+ 2*p3*q2d^3*q1d*sin(q2)*p2^5
+ u2*frd2^2*p1^2*p3*cos(q2)
- 3*p2*p3^3*q1d^4*p1^2*sin(q2)
- 12*u1*p2*p3^2*cos(q2)*sin(q2)*q2d*frd2*p1
+ 12*p2^2*p3^3*q1d^3*p1*q2d*sin(q2)
- 40*p3^4*cos(q2)^2*frd2*q1d^2*p2*q2d
- frd2^3*p1^2*p3*cos(q2)*q2d
- 2*frd2*p2*p3^2*q1d^3*p1^2
+ 2*frd2*p2*p3^2*q1d^3*p1^2*cos(q2)^2
- p3*q2d^3*p2^4*frd1*cos(q2)
- 7*u1*p2*p3^3*cos(q2)*q1d^2*p1
- 16*u1*p2^2*p3^3*cos(q2)*q1d*q2d
- 11*u1*p2^3*p3^2*q2d^2*cos(q2)^2
```

```
+ 3*u2*p3^5*cos(q2)^3*q2d^2
+ 21*u1*p2*p3^4*q2d^2*cos(q2)^4
- 2*frd2^2*p1^2*q1d*p3*sin(q2)*p2*q2d
+ p2^2*frd1*p3^3*q1d^3*cos(q2)
+ 4*p3^2*q2d^3*cos(q2)*p2^3*p1*q1d*sin(q2)
- 3*u2^2*p3^4*sin(q2)*cos(q2)^3
- 6*p3^3*cos(q2)*q2d^2*frd2*p2^2*q1d
- 66*p3^4*cos(q2)^2*q2d^2*frd2*p2*q1d
- 7*frd2^2*p1^2*p3*sin(q2)*p2*q2d^2
- 9*p3^2*p1*q1d^2*p2^2*frd1*q2d
- 10*p3^3*cos(q2)*q2d^3*p2^2*frd1
- 62*p3^5*cos(q2)^2*p2*q2d^3*q1d*sin(q2)
- 29*p3^4*cos(q2)*p2*q2d^2*q1d^2*p1*sin(q2)
- 8*p3^6*cos(q2)^3*q1d^4*sin(q2)
+ 16*frd2*p2^2*p3^3*cos(q2)*q1d^2*q2d
- 4*u1^2*p2^2*p3^2*sin(q2)*cos(q2)
+ 4*p3^3*p2^3*q2d^4*sin(q2)
- p3^5*cos(q2)^2*q1d^4*p1*sin(q2)
- 2*p3^3*q2d^3*p1*p2^2*q1d*sin(q2)
- 4*u1^2*p2*p3^3*sin(q2)*cos(q2)^2
- 34*p3^3*cos(q2)*p2*q2d^3*frd2*p1
- 24*p3^5*cos(q2)^2*p2*q2d^4*sin(q2)
+ 2*u2*frd2^2*p2*p3^2*cos(q2)^2
- 7*u2*p3^2*q1d^2*cos(q2)^2*p2^3
- 4*u2*frd2*p2^2*p3^2*cos(q2)*sin(q2)*q1d
- 17*u2*p3^4*q1d^2*cos(q2)^4*p2
- 2*p3*q2d*frd2*p2^3*q1d^2*cos(q2)*p1
+ 2*u2*frd2*p2*p3*sin(q2)*q1d*p1^2
- 5*frd2*p2^2*frd1*p3^2*cos(q2)*sin(q2)*q1d^2
```

```
- p3^4*cos(q2)^3*frd2*frd1*sin(q2)*q1d^2
- 2*u2*frd2*p2^2*p3*sin(q2)*q1d*p1
- frd2*p2^2*p3*sin(q2)*q1d^2*p1*frd1
+ 4*p3^2*q2d*q1d^3*cos(q2)*p2^4*sin(q2)
+ u1*u2*p3^4*sin(q2)*cos(q2)^3
- u1*u2*p3*sin(q2)*p2^3
+ 2*p3*q2d^2*q1d*cos(q2)*p2^4*frd2
+ 11*u1*p2^2*p3^2*q2d^2*cos(q2)^2*p1
+ 36*p3^5*q2d^3*cos(q2)^4*q1d*sin(q2)*p2
- u2d*frd2*p2^3*p3*cos(q2)
+ 6*r1dddd*p1*p2^3*p3^2*cos(q2)^2
- u2d*frd2*p2*p3^3*cos(q2)^3
+ 3*r1dddd*p1*p2*p3^4*cos(q2)^4
- u1*p3^2*cos(q2)^2*frd2^2*p1
+ 17*u1*p3^4*q1d^2*cos(q2)^4*p2
+ u1*p3^2*q1d^2*cos(q2)^2*p2^3
- 4*u1*frd2^2*p2*p3^2*cos(q2)^2
- frd2*p2^3*frd1*p3*sin(q2)*q1d^2
- 6*u1*p2^2*p3*sin(q2)*q2d*frd2*p1
+ p3*q1d^3*cos(q2)*p2^4*frd1
+ 4*p3^3*q1d^4*cos(q2)^2*p2^3*sin(q2)
- u1*p3*q2d^2*cos(q2)*p2^4
- 10*u1*p3^2*sin(q2)*p2^2*q2d*frd1*cos(q2)
+ 2*u1*p3*q2d*q1d*cos(q2)*p2^3*p1
+ u2*p3*q1d^2*cos(q2)*p2^3*p1
- 4*u1*p3^2*q2d*q1d*cos(q2)^2*p1*p2^2
+ 12*p3^5*q1d^4*cos(q2)^4*p2*sin(q2)
+ 3*u2*frd2^2*p2*p1*p3*cos(q2)
- 4*u2*p3*q2d*q1d*cos(q2)*p2^4
```

```
- 4*u2*p3^3*q2d*q1d*cos(q2)^3*p2^2
+ u1*u2*p3*sin(q2)*p1*p2^2
+ u1*u2*p3^2*sin(q2)*p2^2*cos(q2))/...
(cos(q2)^2*p3^2 - p1*p2 + p2^2)^3
```

$T_8 =$

```
( - 2*frd2*p2*p3^2*q1d^3*p1^2
- 11*u1*p3^2*q2d^2*cos(q2)^2*p2^3
- p2^2*frd1^2*q2d*p3*cos(q2)*frd2
- 2*p2^3*frd1^2*q1d*p3*sin(q2)*q2d
+ 4*frd2*p2^2*p3^3*cos(q2)*q1d^3
- 5*p2^2*frd1^2*p3^2*cos(q2)*sin(q2)*q1d^2
- p2^3*frd1^2*p3*sin(q2)*q2d^2
- 5*p3^2*cos(q2)^2*frd2^2*frd1*q2d*p2
- 9*p3^3*cos(q2)^2*frd2*frd1*sin(q2)*p2*q2d^2
+ 2*p3^2*cos(q2)^2*frd2*frd1^2*q1d*p2
+ 28*p3^3*cos(q2)^2*frd2*frd1*q1d*sin(q2)*p2*q2d
- p3^4*cos(q2)^3*frd2*frd1*sin(q2)*q1d^2
- p3^3*cos(q2)^3*frd2^2*frd1*q2d
- 8*p3^6*cos(q2)^3*q1d^4*sin(q2)
- 2*p3^2*sin(q2)*q1d*p2^2*frd1*q2d*cos(q2)*frd2
- 10*p3^2*p2*q1d^2*p1^2*q2d*frd2
+ 4*p3^3*q2d^4*p2^3*sin(q2)
+ 2*frd2*p2*p3^2*q1d^3*p1^2*cos(q2)^2
- 5*p3^3*cos(q2)^2*frd2*frd1*sin(q2)*q1d^2*p2
+ 2*p3*q2d^2*q1d*cos(q2)*p2^4*frd2
+ p3*q2d*q1d^2*cos(q2)*p2^4*frd1
+ 4*p3^3*q2d*q1d^3*cos(q2)^2*p2^3*sin(q2)
+ 4*p3^2*q2d*q1d^3*cos(q2)*p2^4*sin(q2)
```

```
- 14*p3^2*sin(q2)*p2^2*q2d^2*frd1*cos(q2)*frd2
- u2dd*p3^5*cos(q2)^5
- 3*p3^4*q2d^2*cos(q2)^3*p2^2*sin(q2)*q1d^2
+ 12*p3^4*q2d^3*cos(q2)^3*p2^2*q1d*sin(q2)
- 30*p3^3*sin(q2)*p2*q2d^2*cos(q2)^2*frd2^2
+ 14*p3^2*sin(q2)*p2^2*q2d^2*cos(q2)*frd2^2
- 2*u2^2*p3*sin(q2)*p2^3
+ 10*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1^2*q1d
+ 7*u1*p3^3*q2d^2*cos(q2)^3*p2^2
+ 18*u2*p3^4*cos(q2)^2*p2*q1d^2
+ 26*p3^3*q2d^3*q1d*cos(q2)^2*p1*p2^2*sin(q2)
+ 22*p3^3*q2d^2*q1d*cos(q2)^3*p1*p2*frd2
- 3*tc1ddd*p2^4*p3^2*cos(q2)^2
- 7*p3^3*q2d^2*cos(q2)^2*p2^3*sin(q2)*q1d^2
+ 14*p3^2*q2d^2*q1d*cos(q2)^2*p2^2*p1*frd2
+ 8*p3^2*q2d*q1d^2*cos(q2)^2*p1*p2^2*frd1
+ 8*p3^4*q2d*q1d^3*cos(q2)^3*p1*p2*sin(q2)
+ 20*p3^2*q2d^3*cos(q2)^2*p2^2*p1*frd2
- 10*p3*q2d^3*cos(q2)*p2^3*p1*frd2
+ p3^2*q2d^4*cos(q2)*p2^3*p1*sin(q2)
+ p3*q2d^2*cos(q2)*p2^3*p1*q1d*frd1
+ 7*p3^3*q2d^2*cos(q2)^2*p2^2*p1*sin(q2)*q1d^2
- 26*p3^3*q2d^3*q1d*cos(q2)^2*p2^3*sin(q2)
- 8*p3^2*q2d*q1d^2*cos(q2)^2*p2^3*frd1
+ 10*p3^3*q2d^2*q1d*cos(q2)^3*p2^2*frd2
- 8*p3*q2d^2*q1d*cos(q2)*p2^3*p1*frd2
+ 4*p3^2*q2d^3*q1d*cos(q2)*p2^3*p1*sin(q2)
- 3*tc1ddd*p2^2*p3^4*cos(q2)^4
- p3*q2d*q1d^2*cos(q2)*p2^3*p1*frd1
```

```
+ 4*p3^4*q2d*q1d^3*cos(q2)^3*p2^2*sin(q2)
- 16*p3^2*q2d^2*q1d^2*cos(q2)*p2^3*p1*sin(q2)
- 4*p3^3*q2d*q1d^3*cos(q2)^2*p2^2*p1*sin(q2)
- 12*p3^2*q2d*q1d^3*cos(q2)*p2^3*p1*sin(q2)
- 5*p3^3*q2d*q1d^2*cos(q2)^3*p2^2*frd1
- 14*p3^2*q2d^2*q1d*cos(q2)^2*p2^3*frd2
- u2d*p3^4*cos(q2)^4*frd2
- 6*u1*p3^2*sin(q2)*p2^2*q2d*cos(q2)*frd2
- p3^2*cos(q2)*frd2*sin(q2)*q1d^2*p1*p2*frd1
+ 4*frd2*p2^2*frd1^2*p3*cos(q2)*q1d
+ 2*u2*p3^4*cos(q2)^2*q1d^2*p1
- 4*u2*p3^5*cos(q2)^3*q1d*q2d
- 7*u2*p3^3*p1*p2*cos(q2)^3*q1d^2
- 8*frd2^2*p2*p3^3*cos(q2)^2*sin(q2)*q1d*q2d
- 8*frd2^2*p2*p3^2*sin(q2)*q1d*p1*q2d*cos(q2)
- u1*p3*q2d^2*cos(q2)*p2^4 + 7*u2*p3^3*p1*p2*cos(q2)*q1d^2
- frd2*p2^2*p3*sin(q2)*q1d^2*p1*frd1
+ 4*p3*sin(q2)*p2^2*q2d^2*frd2^2*p1
- 4*u2*p3^3*p2^2*q1d*q2d*cos(q2) - 6*p3^3*q1d^4*p2^3*sin(q2)
- 32*p3^2*sin(q2)*p2*q2d^2*frd2^2*p1*cos(q2)
+ 4*p3*sin(q2)*p2^2*q2d*frd2*p1*q1d*frd1
+ 21*p3^4*q2d^2*cos(q2)^3*p2*sin(q2)*q1d^2*p1
+ 11*p3^2*q2d^2*cos(q2)^2*p2^3*q1d*frd1
- 15*p3^3*q2d^4*cos(q2)^2*p2^3*sin(q2)
+ 10*u2*p3^4*cos(q2)^2*q1d*p2*q2d
- u2*p3*sin(q2)*p2^3*q1d*frd1
- 6*u2*p3^2*q2d^2*cos(q2)^2*p1*p2^2
+ 6*u2*p3^2*q2d^2*cos(q2)^2*p2^3
+ 11*u2*p3*q2d^2*cos(q2)*p1*p2^3
```

```
- 4*frd2*p2^2*p3^3*cos(q2)^3*q1d^3
+ 8*u1*p3^3*q1d^2*cos(q2)^3*p1*p2
- 6*u2*p3*q2d^2*cos(q2)*p1^2*p2^2
- u1*p3^2*q1d^2*cos(q2)^2*p2^2*p1
+ u1*p3*q1d^2*cos(q2)*p2^3*p1
- 5*u2*p3*q2d^2*cos(q2)*p2^4
- 12*u1*p3^2*sin(q2)*p2*q2d*frd2*p1*cos(q2)
- 2*u1*frd2^2*p2*p1*p3*cos(q2)
+ 11*u1*p3^2*q2d^2*cos(q2)^2*p2^2*p1
- 6*u1*p3^4*sin(q2)*cos(q2)^3*q2d*frd2
+ 8*u1*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1
- 4*u1*frd2*p2^2*frd1*p3*cos(q2)
+ 6*u1*p3*sin(q2)*q2d*frd2*p2^3
- 2*u1*p2*p3^2*cos(q2)^2*frd2*frd1
+ 20*u1*p2*p3^4*q2d*q1d*cos(q2)^4
- 6*u1*p3*sin(q2)*p2^2*q2d*frd2*p1
- 10*u1*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1
- 4*u1*p3^2*q2d*q1d*cos(q2)^2*p1*p2^2
+ u1*p3*q2d^2*cos(q2)*p2^3*p1
+ 4*u1*p3^2*q2d*q1d*cos(q2)^2*p2^3
+ 2*u1*p3*q2d*q1d*cos(q2)*p2^3*p1
+ 14*u1*p3^3*q2d*q1d*cos(q2)^3*p2^2
- 2*u1*p3*q2d*q1d*cos(q2)*p2^4
+ 8*u1*p3^2*sin(q2)*cos(q2)*q1d*p2^2*frd1
+ 15*p3^4*q2d*q1d^2*cos(q2)^2*p2*frd1
+ 6*p3^4*sin(q2)*cos(q2)^3*q2d*frd2*q1d*frd1
+ 12*p3^2*sin(q2)*cos(q2)*q1d*p2*frd1*q2d*frd2*p1
+ u1d*p3^4*cos(q2)^4*frd2
- 4*p3^3*sin(q2)*cos(q2)^2*q1d^2*p2*frd1^2
```

```
+ 30*p3^3*q2d^3*cos(q2)^3*p2*frd2*p1
- 21*p3^4*q2d^2*cos(q2)^4*p2*q1d*frd1
+ 12*p3^5*q2d^4*cos(q2)^4*p2*sin(q2)
+ 15*p3^3*q2d^4*cos(q2)^2*p2^2*p1*sin(q2)
+ 6*p3*q2d^3*cos(q2)*p2^2*p1^2*frd2
- 11*p3^2*q2d^2*cos(q2)^2*p2^2*p1*q1d*frd1
- 2*u2dd*p3^3*cos(q2)^3*p2^2
+ 6*p3*q2d^2*q1d*cos(q2)*p2^2*p1^2*frd2
+ 11*p3^2*q2d^2*cos(q2)*p2^2*p1^2*sin(q2)*q1d^2
+ 10*p3^2*q2d^3*frd2*p2^3
- 3*p3*sin(q2)*p2^2*q2d^2*frd1*frd2*p1
+ 8*p3^2*q2d*q1d^3*cos(q2)*p2^2*p1^2*sin(q2)
- 8*p3^3*sin(q2)*cos(q2)^2*q2d^2*frd2^2*p1
- 4*frd2^2*p2*p1*p3^2*cos(q2)*sin(q2)*q1d^2
- frd2^2*p2*p1^2*p3*sin(q2)*q1d^2
- 4*frd2^3*p2*p1*q2d*p3*cos(q2)
+ 6*p3^4*q1d^2*cos(q2)^4*q2d*frd2*p1
+ 16*p3^5*q1d^2*cos(q2)^5*q2d*frd2
- 4*p3^2*q2d^3*cos(q2)*p2^4*q1d*sin(q2)
- p3*q2d^2*cos(q2)*p2^4*q1d*frd1
- frd2^2*p2*frd1*p3*cos(q2)*q2d*p1
- 7*p3*sin(q2)*p2*q2d^2*frd2^2*p1^2
- 2*frd2^2*p2*p3*sin(q2)*q1d*p1^2*q2d
- 4*frd2^2*p2*p3^3*cos(q2)^2*sin(q2)*q1d^2
+ 2*p3*q1d^2*cos(q2)*p2^2*p1^2*q2d*frd2
- u2dd*p3*cos(q2)*p2^4
+ 3*p3^2*q1d^3*cos(q2)^2*p2^2*p1*frd1
+ 4*p3^3*q1d^4*cos(q2)^2*p1^2*p2*sin(q2)
+ p3^2*q1d^4*cos(q2)*p2^2*p1^2*sin(q2)
```

```
- p3*q1d^3*cos(q2)*p2^3*p1*frd1
+ p3^2*cos(q2)^2*frd2^2*p1*q1d*frd1
- p3*cos(q2)*frd2^3*p1^2*q2d
- 4*frd2^3*p2*p3^2*cos(q2)^2*q2d
- p3^2*cos(q2)*frd2^2*p1^2*sin(q2)*q1d^2
+ 2*frd2^2*p2^2*p3*cos(q2)*q1d*frd1
+ 36*p3^4*q1d^2*cos(q2)^4*p2*q2d*frd2
- 12*p3^2*q1d^2*cos(q2)^2*p2^2*q2d*frd2*p1
- 2*p3^4*q1d^4*cos(q2)^3*p2^2*sin(q2)
- 20*p3^3*q1d^2*cos(q2)^3*p2^2*q2d*frd2
- 8*p3^3*q1d^4*cos(q2)^2*p2^2*sin(q2)*p1
+ 12*p3^5*q1d^4*cos(q2)^4*p2*sin(q2)
+ 36*p3^3*q1d^2*cos(q2)^3*p2*q2d*frd2*p1
+ p3^3*q1d^3*cos(q2)^3*p2^2*frd1
+ 4*p3^2*cos(q2)^2*frd2^2*q1d*p2*frd1
+ 5*p3^4*q1d^4*cos(q2)^3*p2*sin(q2)*p1
- 3*p3^2*q1d^3*cos(q2)^2*p2^3*frd1
- 2*p3*q1d^2*cos(q2)*p2^3*q2d*frd2*p1
+ 4*p3^3*q1d^4*cos(q2)^2*p2^3*sin(q2)
+ p3*q1d^3*cos(q2)*p2^4*frd1
- 4*p3^2*cos(q2)^2*frd2^3*q2d*p1
- p3^2*q1d^4*cos(q2)*p2^3*sin(q2)*p1
+ 2*p3^3*cos(q2)^3*frd2^2*q1d*frd1
- 4*p3^3*cos(q2)^2*frd2^2*sin(q2)*q1d^2*p1
- 30*u1*p3^3*sin(q2)*p2*q2d*cos(q2)^2*frd2
+ 2*frd2^2*p2*p1*q1d*frd1*p3*cos(q2)
- 4*p3^4*cos(q2)^3*frd2^2*sin(q2)*q1d^2
+ 8*u2*p3^2*q1d^2*p2^3
- 12*u2*p3^2*sin(q2)*p2^2*q2d*cos(q2)*frd2
```

```
+ 28*u2*p3^2*sin(q2)*p1*p2*q2d*cos(q2)*frd2
- 7*u2*p3^2*sin(q2)*p1*p2*q1d*frd1*cos(q2)
- 3*u2*p3^2*sin(q2)*p2^2*q1d*frd1*cos(q2)
+ 10*u2*p3*sin(q2)*p1^2*p2*q2d*frd2
- 2*u2dd*p3^2*cos(q2)^2*p2^3 - u2dd*p3^4*cos(q2)^4*p2
- u2*p3^4*sin(q2)*cos(q2)^3*q1d*frd1
+ 10*u2*p3^4*sin(q2)*cos(q2)^3*q2d*frd2
+ u2*p3*sin(q2)*p1*p2^2*q1d*frd1
- 11*u2*p3*sin(q2)*p2^2*q2d*frd2*p1
- 8*p3^3*q1d^3*cos(q2)^3*p1*p2*frd1
+ 12*p3^2*q1d^2*cos(q2)^2*p1^2*p2*q2d*frd2
- 4*p3*q2d^3*sin(q2)*q1d*p2^4*p1
+ 2*p3*q2d^2*frd2^2*p2^3*sin(q2)
+ p3*q2d^4*sin(q2)*p2^3*p1^2
+ 3*u2*p3*sin(q2)*p2^2*q2d*frd1*p1
- 2*p3*q2d^4*sin(q2)*p2^4*p1
+ 2*p3*q2d^3*sin(q2)*q1d*p2^3*p1^2
+ 2*p3*q2d^3*sin(q2)*q1d*p2^5
+ 2*u2*frd2*p2*p3*sin(q2)*q1d*p1^2
- 24*u2*p3^3*q2d^2*cos(q2)^3*p2*p1
+ u2*frd2*p2*frd1*p3*cos(q2)*p1
- u2*p3*q1d^2*cos(q2)*p2^2*p1^2
- 7*u2*p3^2*q1d^2*cos(q2)^2*p1^2*p2
- 4*u2*p2*p3^3*cos(q2)^2*frd2*sin(q2)*q1d
- 6*u2*p3*q2d*q1d*cos(q2)*p2^2*p1^2
- p3*q2d^3*p2^4*frd1*cos(q2)
- 2*u2*p3^3*cos(q2)^2*frd2*sin(q2)*q1d*p1
- 4*u2*p3^4*cos(q2)^3*frd2*sin(q2)*q1d
+ 7*u2*p3^3*q2d*cos(q2)^2*sin(q2)*p2*frd1
```

```
- 4*p3^3*q2d^4*p2^2*p1*sin(q2)
- 10*u2*p3^3*q2d*q1d*cos(q2)^3*p1*p2
+ 2*u1dd*p2^3*p3^2*cos(q2)^2 - u2*p3^4*q1d^2*cos(q2)^4*p1
+ u2*p2^2*frd1^2*p3*cos(q2) + u2*p3*cos(q2)*frd2^2*p1^2
+ u2*p3*q1d^2*cos(q2)*p2^3*p1 + u2*p3^3*cos(q2)^3*frd2*frd1
+ 2*u2*p2*p3^2*cos(q2)*frd2*sin(q2)*q1d*p1
+ 3*u2*frd2^2*p2*p1*p3*cos(q2)
- 11*u2*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1
+ 4*u2*frd2*p2^2*frd1*p3*cos(q2)
+ 4*u2*p2*p3^2*cos(q2)^2*frd2*frd1
- 4*u2*frd2*p2^2*p3^2*cos(q2)*sin(q2)*q1d
+ u1dd*p2*p3^4*cos(q2)^4
- 2*u2*frd2*p2^2*p3*sin(q2)*q1d*p1
+ 8*u2*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1
+ 10*u2*p3*q2d*q1d*cos(q2)*p2^3*p1
- 4*u2*p3^3*q2d*q1d*cos(q2)^3*p2^2
- 4*u2*p3*q2d*q1d*cos(q2)*p2^4
+ 2*u2*p3^2*cos(q2)^2*frd2^2*p2
+ 4*p3^2*q2d^2*q1d*p2^2*frd1*p1
+ 3*u2*p3^2*cos(q2)^2*frd2^2*p1
- 17*u2*p3^4*q1d^2*cos(q2)^4*p2
- 7*u2*p3^2*q1d^2*cos(q2)^2*p2^3
+ 17*u2*p3^3*q2d^2*cos(q2)^3*p2^2
- 3*u2*p3*sin(q2)*p2^3*q2d*frd1
+ 19*u2*p3^3*sin(q2)*cos(q2)^2*p2*q2d*frd2
+ 8*u2*p3^3*sin(q2)*cos(q2)^2*q2d*frd2*p1
- 6*u1d*p3^2*sin(q2)*p2^2*q2d*cos(q2)*p1
+ 6*u1d*p3^4*sin(q2)*p2*q2d*cos(q2)^3
+ u1d*p2^2*frd1*p3^2*cos(q2)^2
```

```
+ 6*u1d*p3^2*sin(q2)*p2^3*q2d*cos(q2)
+ 2*u2dd*p3*cos(q2)*p2^3*p1
+ 2*u2dd*p3^3*cos(q2)^3*p1*p2
+ 2*u2dd*p3^2*cos(q2)^2*p2^2*p1
- u2dd*p3*cos(q2)*p2^2*p1^2
- 4*u1^2*p3^3*sin(q2)*p2*cos(q2)^2
- 2*u1dd*p2^2*p3^2*cos(q2)^2*p1
- 12*u2*p3^4*q2d*q1d*cos(q2)^4*p2
- 4*u1^2*p3^2*sin(q2)*p2^2*cos(q2)
+ u1*u2*p3*sin(q2)*p1*p2^2
+ u2*p3*q2d*sin(q2)*frd2*p2^3
- 6*u2^2*p3^3*sin(q2)*p2*cos(q2)^2
- 8*u2*p3^2*q2d*q1d*cos(q2)^2*p2^3
+ 5*u2^2*p3*sin(q2)*p1*p2^2
- u2^2*p3^2*sin(q2)*cos(q2)*p2^2
- u2^2*p3^3*sin(q2)*cos(q2)^2*p1
- 3*u2^2*p3^2*sin(q2)*cos(q2)*p1*p2
+ u1*u2*p3^4*sin(q2)*cos(q2)^3
- 3*u2^2*p3*sin(q2)*p2*p1^2
+ u1*u2*p3^2*sin(q2)*cos(q2)*p2^2
+ 9*u1*u2*p3^3*sin(q2)*p2*cos(q2)^2
+ 7*u1*u2*p3^2*sin(q2)*cos(q2)*p1*p2
- 2*p3*q2d^2*sin(q2)*q1d^2*p2^4*p1
- u1*u2*p3*sin(q2)*p2^3 + p3*q2d^2*sin(q2)*q1d^2*p2^5
+ p3*q2d^2*sin(q2)*q1d^2*p2^3*p1^2
- 14*p3^5*cos(q2)^2*q1d^4*p2*sin(q2)
+ 9*p3^3*q2d^3*p2^2*frd1*cos(q2)^3
+ p3*q2d^3*p2^3*frd1*cos(q2)*p1
- 2*u2d*p3^5*sin(q2)*q2d*cos(q2)^4
```

```
- 4*u2d*p3^2*sin(q2)*q2d*cos(q2)*p2^3
- 4*u2d*p3^4*sin(q2)*q2d*cos(q2)^3*p2
- 2*u2d*p3^3*sin(q2)*p2*q2d*p1*cos(q2)^2
+ 3*tc1ddd*p1*p2*p3^4*cos(q2)^4
+ 4*u2d*p3^2*sin(q2)*q2d*cos(q2)*p2^2*p1
+ u2d*p3*cos(q2)*frd2*p1^2*p2
+ 6*tc1ddd*p1*p2^3*p3^2*cos(q2)^2
+ 4*u2d*p3*sin(q2)*p2^2*q2d*p1^2
- 4*u2d*p3*sin(q2)*q1d*p2^3*p1
+ 2*u2d*p3*sin(q2)*q1d*p2^2*p1^2
- 3*tc1ddd*p1^2*p2^2*p3^2*cos(q2)^2
+ u2d*p2^2*frd1*p3*cos(q2)*p1
- 8*u2d*p3*sin(q2)*p2^3*q2d*p1
+ 2*u2d*p3^3*sin(q2)*p2^2*q2d*cos(q2)^2
- 4*u2d*p3^3*cos(q2)^2*sin(q2)*q1d*p1*p2
+ 2*u2d*p3^2*sin(q2)*q1d*p2^3*cos(q2)
+ 2*u2d*p3^4*sin(q2)*q1d*p2*cos(q2)^3
+ 4*u2d*p3^3*sin(q2)*q1d*p2^2*cos(q2)^2
- 2*u2d*p3^2*sin(q2)*q1d*p2^2*cos(q2)*p1
- u2d*p2^3*frd1*p3*cos(q2)
+ 2*u2d*p3^5*cos(q2)^4*sin(q2)*q1d
+ 4*u2d*p3*sin(q2)*p2^4*q2d
- u2d*p2*frd1*p3^3*cos(q2)^3
- u2d*p3^3*cos(q2)^3*frd2*p1 - u2d*p2^2*frd1*p3^2*cos(q2)^2
+ 2*u2d*p3*sin(q2)*q1d*p2^4 - u2d*frd2*p2*p3^3*cos(q2)^3
- u2d*frd2*p2^2*p3^2*cos(q2)^2 - u2d*frd2*p2^3*p3*cos(q2)
- u1d*p3^2*cos(q2)^2*frd2*p1*p2
+ 8*u2*p3^2*q2d*q1d*cos(q2)^2*p2^2*p1
+ 2*u1d*frd2*p2^3*p3*cos(q2) + 2*u1d*frd2*p2*p3^3*cos(q2)^3
```

```
+ 2*u1d*frd2*p2^2*p3^2*cos(q2)^2
- 2*u1d*frd2*p2^2*p3*cos(q2)*p1 - 3*u1*p3^2*q1d^2*p2^3
- p3^5*cos(q2)^2*q1d^4*p1*sin(q2)
+ 16*p3^3*q1d^2*p2^2*q2d*cos(q2)*frd2
+ 5*p3^2*q1d^3*p2^3*frd1 - 11*p3^3*q1d^2*p2^3*q2d^2*sin(q2)
+ 21*p3^4*cos(q2)^2*q1d^3*p2*frd1
- 12*p3^3*q1d^3*p2^3*q2d*sin(q2)
- 4*p3^2*q1d^2*p2^3*q2d*frd2 + 11*u2*p3^5*cos(q2)^3*q1d^2
+ 6*u2*p3^2*p2^3*q1d*q2d + u2*p3^3*p2^2*cos(q2)*q1d^2
- 20*p3^5*cos(q2)^3*q1d^2*q2d*frd2
- 6*p3^3*q1d*p2^2*q2d^2*cos(q2)*frd2
- 44*p3^5*cos(q2)^2*q1d^3*p2*q2d*sin(q2)
+ p3^5*cos(q2)^3*q1d^3*frd1 - 3*p3^3*p2*q1d^4*p1^2*sin(q2)
- 40*p3^4*cos(q2)^2*q1d^2*q2d*frd2*p2
- 67*p3^5*cos(q2)^2*q1d^2*p2*q2d^2*sin(q2)
+ 4*p3^4*q1d^3*p2^2*q2d*cos(q2)*sin(q2)
+ 8*p3^5*q2d^2*q1d*cos(q2)^5*frd2
+ 6*p3^6*q1d^4*cos(q2)^5*sin(q2) + 4*u1*p2^3*p3^2*q2d^2
- 2*u2*p3^5*q2d^2*cos(q2)^5 - 2*u1*p3^3*cos(q2)^3*frd2^2
+ 2*p3^3*q1d*p2^3*q2d^3*sin(q2)
- 3*p3^6*cos(q2)^3*q1d^2*q2d^2*sin(q2)
+ 9*p3^2*q1d^2*p2^3*q2d*frd1 + 6*p3^2*q1d*p2^3*q2d^2*frd2
- 4*p3^4*cos(q2)*p2^2*q2d^4*sin(q2)
- 14*p3^3*q2d^3*cos(q2)^3*p2^2*frd2
+ 5*p3^4*cos(q2)*q1d^2*p2^2*q2d^2*sin(q2)
+ 6*p3^3*cos(q2)*q1d^2*p2^2*q2d*frd1
+ 6*p3^5*q2d^3*cos(q2)^5*frd2 + p3*q2d^4*sin(q2)*p2^5
+ 3*p3^4*q2d^4*cos(q2)^3*p2^2*sin(q2)
- 66*p3^4*cos(q2)^2*q1d*p2*q2d^2*frd2
```

```
+ 2*u2*p3^3*cos(q2)^3*frd2^2
- 12*p3^3*cos(q2)*q1d*p2^2*frd1*q2d^2
- 50*p3^4*cos(q2)^2*p2*q2d^3*frd2
+ p3^3*q1d^3*p2^2*frd1*cos(q2)
- 20*p3^2*q2d^3*cos(q2)^2*p2^3*frd2
+ 4*p3*q2d^3*cos(q2)*p2^4*frd2
- p3^2*q2d^4*cos(q2)*p2^4*sin(q2)
- 10*u2*p3^5*q1d^2*cos(q2)^5
- 4*p3^3*cos(q2)^3*frd2^3*q2d - 4*p3^2*q1d*p2^3*frd1*q2d^2
+ 12*p3^3*q2d*q1d^3*p1*p2^2*sin(q2)
+ 11*p3^3*q2d^2*q1d^2*p1*p2^2*sin(q2)
- 20*p3^4*q2d*q1d^3*p1*p2*cos(q2)*sin(q2)
- 2*p3^3*q2d^3*q1d*p1*p2^2*sin(q2)
- frd2*p2^3*frd1*p3*sin(q2)*q1d^2
- 6*p3^2*q2d^2*q1d*p1*p2^2*frd2
- 30*p3^3*q2d^2*q1d*p1*p2*cos(q2)*frd2
- 10*p3^3*q2d^3*cos(q2)*p2^2*frd1
- 4*p3^4*p2*q1d^4*p1*cos(q2)*sin(q2)
- 3*u2^2*p3^4*sin(q2)*cos(q2)^3
- 5*p3^2*p2^2*q1d^3*p1*frd1
+ 14*p3^2*p2^2*q1d^2*p1*q2d*frd2
- 9*p3^2*q2d*q1d^2*p1*p2^2*frd1
+ 16*p3^3*p2^2*q2d^3*cos(q2)*frd2
- u1*p3*q1d^2*cos(q2)*p2^4
- 5*frd2^2*p2^2*frd1*q2d*p3*cos(q2)
+ 21*u1*p3^4*q2d^2*cos(q2)^4*p2
+ frd2*p2^3*frd1*p3*sin(q2)*q2d^2
+ 4*p3^6*q2d*q1d^3*cos(q2)^3*sin(q2)
- 62*p3^5*q2d^3*q1d*cos(q2)^2*p2*sin(q2)
```

```
- 8*frd2*p2^3*frd1*q1d*p3*sin(q2)*q2d
- p2^3*frd1^2*p3*sin(q2)*q1d^2
- 5*frd2*p2^2*frd1*p3^2*cos(q2)*sin(q2)*q1d^2
+ 40*p3^4*q2d^3*cos(q2)^4*p2*frd2
- 6*p3^5*q2d^2*q1d*cos(q2)^3*frd2
+ 17*u1*p3^4*q1d^2*cos(q2)^4*p2
- u1*p3^2*cos(q2)^2*frd2^2*p1
+ 14*u2*p3^2*p2^2*q1d^2*p1*cos(q2)^2
- 2*u1*frd2^2*p2^2*p3*cos(q2)
- u1*p3^3*q1d^2*cos(q2)^3*p2^2
- 16*u1*p3^3*cos(q2)*q1d*p2^2*q2d - 14*u2*p3^2*p2^2*q1d^2*p1
- 16*p3^4*cos(q2)*q1d*p2^2*q2d^3*sin(q2)
+ u1*p3^2*q1d^2*cos(q2)^2*p2^3 - 12*u2*p3^4*q2d^2*cos(q2)^4*p2
+ 2*p3^4*cos(q2)^2*frd2*q1d^3*p1
- 2*p3^4*cos(q2)^4*frd2*q1d^3*p1 - 6*u1*p3^2*q1d*p2^3*q2d
+ 9*p3^3*p2^2*q1d^4*p1*sin(q2) - 4*u1*p3^2*cos(q2)^2*frd2^2*p2
- 19*p3^4*q1d^3*cos(q2)^4*p2*frd1
+ 48*p3^5*q2d^2*q1d^2*cos(q2)^4*sin(q2)*p2
- 14*p3^4*sin(q2)*cos(q2)^3*q2d^2*frd2^2
- 7*u1*p3^3*p2*q1d^2*p1*cos(q2)
+ 36*p3^5*q2d*q1d^3*cos(q2)^4*sin(q2)*p2
- 14*p3^4*q2d*q1d^2*cos(q2)^4*p2*frd1
+ 36*p3^5*q2d^3*q1d*cos(q2)^4*sin(q2)*p2
+ 5*p3^2*q2d^2*cos(q2)*p2^4*sin(q2)*q1d^2
+ 58*p3^4*q2d^2*q1d*cos(q2)^4*frd2*p2
+ 3*u2*p3^2*q2d^2*p1*p2^2
+ 20*u2*p3^3*p1*p2*q1d*q2d*cos(q2)
+ 6*u2*p3^2*p1^2*p2*q1d^2 + 2*frd2*p2^3*frd1^2*q1d
- frd2^2*p2^3*frd1*q2d - p2^3*frd1^2*q2d*frd2
```

```
- 2*u1*frd2*p2^3*frd1 - u1*frd2^2*p2^2*p1
- frd2^3*p2*p1^2*q2d + u2*frd2*p2^3*frd1
+ u2*frd2^2*p2*p1^2 + p2^3*frd1^3*q1d
- u1*p2^3*frd1^2 + u1dd*p2^3*p1^2 - 2*u1dd*p2^4*p1
- u2dd*p1^2*p2^3 + 2*u2dd*p1*p2^4 + u1d*p2^4*frd1
+ u1d*frd2*p2^4 - u2d*p2^4*frd1 + tc1ddd*p1^3*p2^3
- 3*tc1ddd*p1^2*p2^4 + 3*tc1ddd*p1*p2^5
- tc1ddd*p3^6*cos(q2)^6 + u2*p2^3*frd1^2
- frd2^2*p2^2*frd1*q2d*p1 - 6*u2*p3^2*p1*p2^2*q1d*q2d
+ frd2^2*p2^2*p1*q1d*frd1 + u1dd*p2^5 - u2dd*p2^5
+ u2*frd2*p2^2*frd1*p1 - u1d*p2^3*frd1*p1
- u1d*frd2*p2^3*p1 + u2d*p2^3*frd1*p1 - u2d*frd2*p2^3*p1
+ u2d*frd2*p2^2*p1^2 - tc1ddd*p2^6
+ 11*p3^3*q2d^2*cos(q2)^3*p2^2*q1d*frd1
- u1*p3^5*cos(q2)^3*q1d^2 + 4*p3^5*cos(q2)^3*frd2*q1d^3
- 4*p3^5*cos(q2)^5*frd2*q1d^3 + 3*u2*p3^5*q2d^2*cos(q2)^3
- 8*p3^5*cos(q2)^3*q2d^3*frd2 + 2*frd2*p2^2*p3^2*q1d^3*p1
- 2*frd2*p2^2*p3^2*q1d^3*p1*cos(q2)^2
- 34*p3^3*p2*q1d^2*p1*q2d*cos(q2)*frd2
- 10*p3^2*p2^2*q2d^3*frd2*p1
+ 7*p3^3*cos(q2)*q1d^3*p2*frd1*p1
- 8*p3^4*cos(q2)^2*q1d^2*q2d*frd2*p1
- 4*u1*p2^2*p3^2*q2d^2*p1 + 15*u2*p3^4*q2d^2*cos(q2)^2*p2
- 21*u2*p3^3*q2d^2*cos(q2)*p2^2
+ 29*u2*p3^3*q2d^2*cos(q2)*p1*p2
- 28*u1*p3^4*cos(q2)^2*p2*q2d^2 - u1*p3^3*q1d^2*p2^2*cos(q2)
- 22*u1*p2*p3^4*q2d*q1d*cos(q2)^2
- 19*u1*p2*p3^4*cos(q2)^2*q1d^2
- 4*frd2*p2*p3^3*q1d^3*p1*cos(q2)
```

```
+ 4*frd2*p2*p3^3*q1d^3*p1*cos(q2)^3
+ 3*u1*p3^2*p2^2*q1d^2*p1 + 6*u1*p3^2*q2d*q1d*p1*p2^2
- 3*u2*p3^2*q2d^2*p2^3 - 8*u1*p3^3*cos(q2)*p2^2*q2d^2
- 34*p3^3*cos(q2)*p2*q2d^3*frd2*p1
- 29*p3^4*cos(q2)*p2*q2d^2*q1d^2*p1*sin(q2)
- 24*p3^5*cos(q2)^2*p2*q2d^4*sin(q2)
+ 28*p3^4*cos(q2)^2*q1d*p2*frd1*q2d^2)/...
(cos(q2)^2*p3^2 - p1*p2 + p2^2)^3
```

$T_{12} =$

```
 - (u2*p2^2*frd1 + p2^2*frd1^2*q1d - u2d*p1*p2^2
- u1*p2^2*frd1 - u1d*p2^2*p1 - u2d*p3^3*cos(q2)^3
- u1*frd2*p1*p2 + frd2*p1*q1d*p2*frd1 - frd2^2*p1^2*q2d
+ frd2*p1*q1d*frd1*p3*cos(q2) + 3*p3^3*q2d*cos(q2)^3*q1d^2*p1
+ 4*u2*p3^2*sin(q2)*q2d*cos(q2)*p1 - frd2*p1^2*p3*sin(q2)*q1d^2
+ 6*p3^4*q2d^2*q1d*cos(q2)^4 + r2ddd*cos(q2)^4*p3^4
- p3^2*q2d^3*p1*p2 + u2d*p1^2*p2 + p3^2*p2^2*q2d^3
+ r2ddd*p1^2*p2^2 - 2*r2ddd*p1*p2^3 + r2ddd*p2^4
+ u2*frd2*p1^2 - 2*u1*p3*sin(q2)*q1d*p2^2
- 2*u1*p3^3*sin(q2)*q1d*cos(q2)^2 - u1*p2*frd1*p3*cos(q2)
+ 2*u1*p2*p3*sin(q2)*q1d*p1 - 2*r2ddd*p1*p2*cos(q2)^2*p3^2
- u1*p3*sin(q2)*q2d*p1*p2 + u1*p3*sin(q2)*q2d*p2^2
- 2*p3^2*sin(q2)*q1d*p1*q2d*frd2*cos(q2)
- 2*p3*sin(q2)*q1d^2*p1*p2*frd1 + u1d*p3^3*cos(q2)^3
+ 2*p3^2*q2d^3*cos(q2)^2*p1*p2 + 2*p3^4*q2d^3*cos(q2)^4
+ u1d*p3*cos(q2)*p2^2 - frd1*p3^2*cos(q2)*sin(q2)*p2*q2d^2
+ frd1*p3^3*cos(q2)^2*sin(q2)*q1d^2 + frd1^2*p3*cos(q2)*q1d*p2
- 4*p3^3*sin(q2)*q1d*cos(q2)^2*q2d*frd2 - u1d*p3*cos(q2)*p1*p2
- 3*u1*p3^3*sin(q2)*q2d*cos(q2)^2 - u1*frd2*p1*p3*cos(q2)
```

```
+ 2*p3^2*q2d^2*q1d*cos(q2)^2*p1*p2
- frd1*p3^2*cos(q2)^2*q2d*frd2
+ 3*p3^3*q2d*q1d*sin(q2)*frd1*cos(q2)^2
- 2*p3^2*q2d^2*q1d*cos(q2)^2*p2^2
+ p3*q2d*cos(q2)*q1d^2*p1^2*p2
- 5*p3^2*q2d^2*cos(q2)*sin(q2)*frd2*p1
+ p3*q2d*q1d*sin(q2)*frd1*p1*p2
+ 6*p3^4*q2d*q1d^2*cos(q2)^4 - u2d*p1*cos(q2)^2*p3^2
+ 2*r2ddd*p2^2*cos(q2)^2*p3^2 - 2*p3^2*q2d^3*cos(q2)^2*p2^2
+ u2*frd1*p3^2*cos(q2)^2 - 3*p3^4*q2d^3*cos(q2)^2
+ u2*p3^3*sin(q2)*q2d*cos(q2)^2 + u1d*p2*p3^2*cos(q2)^2
- 4*frd2^2*p3^2*cos(q2)^2*q2d + p3*q2d^3*cos(q2)*p2^2*p1
- p2^2*frd1*p3*sin(q2)*q2d^2 - p3*q2d^3*cos(q2)*p2^3
- p3*q2d*cos(q2)*q1d^2*p2^2*p1 - 4*frd2^2*p3*cos(q2)*q2d*p1
+ 2*frd2*p3^2*cos(q2)^2*q1d*frd1 - 2*p2*frd1*q2d*frd2*p3*cos(q2)
+ p3*sin(q2)*q1d^2*p2^2*frd1 - p2^2*frd1*q2d*frd2
- u2d*p3*cos(q2)*p2^2 + u1d*p2^3 + 2*u2*frd2*p3^2*cos(q2)^2
- 2*u1*frd2*p3^2*cos(q2)^2 - 4*p3^3*q2d^3*cos(q2)*p2
- 4*frd2*p3^2*cos(q2)*sin(q2)*q1d^2*p1
- 8*p3^4*q2d^2*q1d*cos(q2)^2 + 2*u2*p2*frd1*p3*cos(q2)
+ 2*p3^2*q2d*q1d^2*cos(q2)^2*p1*p2
- 4*u1*p2*p3^2*cos(q2)*sin(q2)*q2d
+ 2*frd2*p2*q1d*frd1*p3*cos(q2) - 8*p3^3*q1d*cos(q2)*p2*q2d^2
- 2*u1*frd2*p2*p3*cos(q2) - 2*p3*sin(q2)*q1d*p2*q2d*frd2*p1
- 4*p3^2*sin(q2)*q1d*p2*q2d*frd2*cos(q2)
+ 3*u2*p3*sin(q2)*q2d*p1*p2 - 3*u2*p3*sin(q2)*q2d*p2^2
- 3*p3*sin(q2)*p2^2*q2d*q1d*frd1
+ 4*p3*sin(q2)*p2^2*q2d^2*frd2 - 2*p3*q2d^2*q1d*cos(q2)*p2^3
+ 2*p3^2*q2d*q1d*cos(q2)*sin(q2)*p2*frd1
```

```
- 5*p3*q2d^2*sin(q2)*frd2*p1*p2
- 6*p3^3*q2d^2*sin(q2)*frd2*cos(q2)^2
+ 6*p3^3*q2d^2*q1d*cos(q2)^3*p2 - 4*p3^3*q2d*q1d^2*cos(q2)*p1
+ u2d*p3*cos(q2)*p1*p2 + 3*u2*frd2*p3*cos(q2)*p1
+ 2*p3*q2d^2*q1d*cos(q2)*p2^2*p1
- 2*p3^2*q2d*q1d^2*cos(q2)^2*p2^2
- 2*p3^2*sin(q2)*p2*q2d^2*frd2*cos(q2)
- 2*u2*p3^2*sin(q2)*q2d*cos(q2)*p2
- 2*u2*p2*p3^2*sin(q2)*q1d*cos(q2)
- 2*u2*p3^3*sin(q2)*q1d*cos(q2)^2
- 4*frd2*p3^3*cos(q2)^2*sin(q2)*q1d^2
- 2*p3^2*sin(q2)*q1d^2*cos(q2)*p2*frd1
- 8*p3^4*q2d*q1d^2*cos(q2)^2
+ 3*p3^3*q2d^3*cos(q2)^3*p2)/...
( - p1*p2 + p2^2 + p3^2*cos(q2)^2)^2
```

$T_{16} =$

```
 - ( - 4*u2*p3^3*cos(q2)^2*frd2*sin(q2)*q1d*p2
- 2*u2*p3*q2d*q1d*cos(q2)*p1^2*p2^2
- 2*u2*frd2*p1*p3^3*sin(q2)*cos(q2)^2*q1d
+ 4*u2*frd2*p1*p2*frd1*p3*cos(q2)
- 7*p3^3*p1*p2^2*q2d^4*sin(q2)
- 7*p3^3*p1^2*p2*q2d^2*q1d^2*sin(q2)
+ u2*p3^2*cos(q2)^2*frd1^2*p2
+ 2*u2*frd2*p1*p3^2*cos(q2)^2*frd1
- 2*u2*frd2*p1*p3^2*sin(q2)*cos(q2)*q1d*p2
+ u1dd*p3*cos(q2)*p2^4 + 7*p3^4*cos(q2)^2*frd1*q1d^3*p1
+ 12*p3^4*cos(q2)^2*q1d*p2*frd1*q2d^2
- 2*p3^2*q2d^2*q1d^2*cos(q2)*p2^4*sin(q2)
```

```
+ 5*p3*q2d*q1d^2*cos(q2)*p2^4*frd1
- 28*p3^2*q2d^2*q1d*cos(q2)^2*p2^3*frd2
- 32*p3^5*cos(q2)^2*q1d^3*p2*q2d*sin(q2)
- 15*p3^6*q2d^4*cos(q2)^3*sin(q2)
- 4*p3*q2d^2*q1d*cos(q2)*p2^3*frd2*p1
- 2*u2*p3^4*sin(q2)*cos(q2)^3*q1d*frd1
- 52*p3^5*cos(q2)^3*q2d^2*frd2*q1d
- 2*p3^4*cos(q2)*q1d^2*p2^2*q2d^2*sin(q2)
+ 21*p3^5*cos(q2)^3*frd1*q1d^2*q2d
- 2*p3^2*q2d^3*frd1*p1*p2^2
- 2*p3*q1d^2*cos(q2)*p1^2*p2^2*q2d*frd2
- 14*frd2*p2^3*frd1*q1d*p3*sin(q2)*q2d
+ 2*u1dd*p2^3*p3^2*cos(q2)^2
- 4*p3^3*q1d^4*cos(q2)^2*p1*p2^2*sin(q2)
+ p3*q1d^3*cos(q2)*p1*p2^3*frd1
- 2*p3^2*q1d^3*cos(q2)^2*p1*p2^2*frd1
- p3*q2d^2*q1d*cos(q2)*frd1*p1^2*p2^2
+ p3*q2d^2*sin(q2)*q1d^2*p1^3*p2^2
- u2d*p3^4*cos(q2)^4*frd1
+ 4*p3^2*q2d^4*cos(q2)*sin(q2)*p1^2*p2^2
- 6*u2*p3^3*q1d^2*cos(q2)^3*p1*p2
- u2*p3*q1d^2*cos(q2)*p1^3*p2 + 3*u2*p3^3*cos(q2)^3*frd2*frd1
+ 4*p3^3*q1d^4*cos(q2)^2*p1^2*p2*sin(q2)
- p3*q1d^3*cos(q2)*p1^2*p2^2*frd1
- 2*p3^2*q1d^3*cos(q2)^2*p1^2*p2*frd1
- 12*u1*p3^5*cos(q2)^3*q1d^2
+ 12*p3^5*q1d^4*cos(q2)^4*p1*sin(q2)
+ 2*p3*q1d^2*cos(q2)*p1^3*p2*q2d*frd2
+ 18*p3^2*q1d^2*cos(q2)^2*p1^2*p2*q2d*frd2
```

```
- p3^2*q1d^4*cos(q2)*p1^2*p2^2*sin(q2)
- 4*p3^2*q1d^3*cos(q2)*p1*p2^3*sin(q2)*q2d
- 13*p3^3*q1d^3*cos(q2)^3*p1*p2*frd1
+ 28*p3^3*q1d^2*cos(q2)^3*p1*p2*q2d*frd2
+ 3*p3^4*q1d^4*cos(q2)^3*p1^2*sin(q2)
+ p3^2*p1^2*q1d^3*p2*frd1
- 28*p3^3*q1d^2*cos(q2)^3*p2^2*q2d*frd2
+ 24*p3^5*q1d^3*cos(q2)^4*sin(q2)*p2*q2d
+ u1dd*p2*p3^4*cos(q2)^4 + 6*u2*p3^2*cos(q2)^2*frd2*frd1*p2
- 2*u2*p3^2*q2d*q1d*cos(q2)^2*p1^2*p2
+ 2*u2*p3*q2d*q1d*cos(q2)*p1*p2^3
- 14*u2*p3^4*q2d*q1d*cos(q2)^4*p1
+ 16*p3^3*q1d^2*cos(q2)^3*p1^2*q2d*frd2
+ 54*p3^4*q1d^2*cos(q2)^4*p1*q2d*frd2
- 8*p3^3*q1d^3*cos(q2)^2*p2^3*sin(q2)*q2d
- 4*p3^4*q1d^4*cos(q2)^3*p2^2*sin(q2)
+ 4*p3^2*q1d^3*cos(q2)^2*p2^3*frd1
+ 6*p3^3*q1d^3*cos(q2)^3*p2^2*frd1
- 12*p3^4*q1d^3*cos(q2)^4*p2*frd1 - 2*p2^3*frd1*p3^2*q1d^3
- 42*p3^3*q2d^3*cos(q2)^2*p2^3*q1d*sin(q2)
+ 14*p3^2*q2d^2*cos(q2)^2*p2^3*q1d*frd1
- 2*p3^2*q2d^3*cos(q2)^2*p2^2*frd2*p1
- 2*p3^2*q2d^2*cos(q2)*p2^3*sin(q2)*q1d^2*p1
- 66*p3^6*cos(q2)^3*q1d^2*q2d^2*sin(q2)
- 6*p3*sin(q2)*p2^2*q2d^2*frd1*frd2*p1
- 54*p3^3*q2d^3*cos(q2)^3*p2^2*frd2
+ 44*p3^5*q1d^2*cos(q2)^5*q2d*frd2
+ 44*p3^4*q2d^2*q1d*cos(q2)^4*p2*frd2
- 4*p3*sin(q2)*p2^3*q2d*frd1^2*q1d
```

```
- 10*p3^2*sin(q2)*p2^2*q2d^2*frd1*cos(q2)*frd2
- 30*p3^4*q2d^2*cos(q2)^3*p2^2*sin(q2)*q1d^2
- 19*p3^3*q2d^4*cos(q2)^2*p2^3*sin(q2)
- 6*p3^2*sin(q2)*cos(q2)*q2d^2*p2*frd1*frd2*p1
- 20*p3^3*sin(q2)*cos(q2)^2*q2d^2*p2*frd1*frd2
+ 8*p3^2*sin(q2)*cos(q2)*q2d*p2^2*frd1^2*q1d
+ 8*p3^3*sin(q2)*cos(q2)^2*q2d*p2*frd1^2*q1d
+ 70*p3^3*q2d^3*cos(q2)^3*p2*frd2*p1
+ 22*p3^4*q2d^3*cos(q2)^4*p2*frd2
+ 24*p3^5*q2d^3*cos(q2)^5*frd2
+ 36*p3^5*q2d^2*cos(q2)^4*p2*sin(q2)*q1d^2
- 26*p3^4*q2d^3*cos(q2)^3*p2^2*q1d*sin(q2)
+ 30*p3^3*q2d^2*cos(q2)^3*p2^2*q1d*frd1
- 5*u1*p3^2*cos(q2)^2*frd2*frd1*p2
- u1*frd2*p1*p2*frd1*p3*cos(q2)
+ 6*u2*p3^2*q1d^2*cos(q2)^2*p1*p2^2
- 2*u1dd*p3*cos(q2)*p1*p2^3
- 3*tc2ddd*p1*p2*p3^4*cos(q2)^4
+ 48*p3^4*q2d^2*cos(q2)^3*p2*sin(q2)*q1d^2*p1
- 15*p3^4*q2d^4*cos(q2)^3*p2^2*sin(q2)
- 4*p3*cos(q2)*frd2^2*p2*frd1*q2d*p1
- 8*p3^2*cos(q2)^2*frd2^2*p2*frd1*q2d
+ p3^2*q1d^4*cos(q2)*p1^3*p2*sin(q2)
- 6*p3^4*q1d^3*cos(q2)^4*p1*frd1
+ 14*p3*sin(q2)*q2d^2*frd2^2*p2^2*p1
+ 28*p3^2*sin(q2)*q2d^2*frd2^2*p2^2*cos(q2)
+ 5*p3*cos(q2)*frd2*p2^2*frd1^2*q1d
+ 12*u2*p3^5*cos(q2)^3*q1d^2
+ 4*frd2*p2^3*frd1*p3*sin(q2)*q2d^2
```

```
- 4*frd2^2*p2^2*frd1*q2d*p3*cos(q2)
- 2*u2d*p3^4*cos(q2)^4*frd2
- 18*p3^2*q1d^2*cos(q2)^2*p1*p2^2*q2d*frd2
+ 4*p3^4*q1d^4*cos(q2)^3*p1*p2*sin(q2)
+ 18*p3^4*sin(q2)*cos(q2)^3*q2d*frd2*q1d*frd1
- 34*p3^5*cos(q2)^3*q2d^3*frd2
- 12*p3^3*q2d^2*cos(q2)^2*p2^3*sin(q2)*q1d^2
+ 3*p3^2*q2d^4*cos(q2)*p2^4*sin(q2)
- 14*p3^2*q2d^3*cos(q2)^2*p2^3*frd2
+ 12*p3^5*q2d*q1d^3*cos(q2)^4*sin(q2)*p1
- u2dd*p3^5*cos(q2)^5 + 44*p3^5*q2d^2*q1d*cos(q2)^5*frd2
+ 19*p3^3*q2d^2*cos(q2)^2*p1^2*p2*sin(q2)*q1d^2
+ 42*p3^3*q2d^3*cos(q2)^2*p1*p2^2*q1d*sin(q2)
+ 12*u1*p3^5*q2d^2*cos(q2)^5 + 4*p2^3*frd1^2*p3*sin(q2)*q1d^2
+ 2*p3*sin(q2)*p1^2*q1d*p2*frd1*q2d*frd2
+ 8*p3*sin(q2)*p1*q2d*frd2*p2^2*q1d*frd1
- 8*p3^3*cos(q2)^3*frd2^3*q2d
- p3^2*sin(q2)*p1*q1d^2*p2*frd1^2*cos(q2)
- 5*p3*sin(q2)*p1*q1d^2*p2^2*frd1^2
- 3*p3^4*cos(q2)^3*frd1^2*sin(q2)*q1d^2
+ 24*p3^6*q2d*q1d^3*cos(q2)^5*sin(q2)
- 5*p3^4*q2d*q1d^2*cos(q2)^4*p2*frd1
- 16*p3^5*q2d*q1d^2*cos(q2)^5*frd1
- 2*p3^2*q2d^3*cos(q2)*p2^4*q1d*sin(q2)
- 2*p3*q2d^2*cos(q2)*p2^4*q1d*frd1
- 14*p3*q2d^3*cos(q2)*p2^3*frd2*p1
- 26*p3^2*sin(q2)*cos(q2)*q2d*frd2*p2^2*q1d*frd1
+ 38*p3^4*q2d^3*cos(q2)^3*p1*p2*q1d*sin(q2)
- 14*p3^2*q2d^2*cos(q2)^2*p1*p2^2*q1d*frd1
```

```
+ p3^3*cos(q2)^3*frd2*frd1^2*q1d
+ 14*p3^3*p2^3*q2d^3*q1d*sin(q2)
+ 16*p3^2*q2d^3*cos(q2)^2*p1^2*p2*frd2
- 7*p3^2*q2d*q1d^2*cos(q2)^2*p1*p2^2*frd1
+ 6*p3^3*q2d*q1d^2*cos(q2)^3*p1*p2*frd1
+ 6*p3^6*q2d^4*cos(q2)^5*sin(q2)
- 4*u2*p3^4*cos(q2)^3*frd2*sin(q2)*q1d
- u2*p3*sin(q2)*p1^2*q1d*p2*frd1
- 7*u2*p3^3*q1d^2*cos(q2)^3*p1^2
- 19*p3^3*q2d^2*cos(q2)^3*p1*p2*q1d*frd1
+ 19*p3^3*q2d^4*cos(q2)^2*p1*p2^2*sin(q2)
+ 18*p3^4*q2d^4*cos(q2)^3*p1*p2*sin(q2)
+ 4*p3^3*q2d*q1d^3*cos(q2)^2*p1^2*p2*sin(q2)
+ 20*p3^4*q2d*q1d^3*cos(q2)^3*p1*p2*sin(q2)
+ 18*p3^5*cos(q2)^3*frd1*q1d*q2d^2
+ 8*p3*q2d^3*cos(q2)*p2^2*p1^2*frd2
+ 4*p3^2*q2d*q1d^3*cos(q2)*p1^2*p2^2*sin(q2)
+ 4*p3^3*q2d*q1d^3*cos(q2)^2*p1*p2^2*sin(q2)
+ tc2ddd*p3^6*cos(q2)^6 + 20*p3^2*q2d^2*q1d*p2^3*frd2
- 26*u1*p3^3*cos(q2)*q1d*p2^2*q2d
- 18*u1*p3^5*cos(q2)^3*q2d^2
+ 7*p3^3*p2^2*q2d^2*q1d^2*p1*sin(q2)
- 16*p3^4*cos(q2)*q1d^3*p2*q2d*p1*sin(q2)
+ 4*p3^2*q2d^2*cos(q2)*p2^2*p1^2*sin(q2)*q1d^2
+ 3*p3*q2d^2*cos(q2)*p2^3*p1*q1d*frd1
- 3*u2*p3^2*p1*p2^2*q1d^2
+ 4*p3^2*q2d^2*q1d*cos(q2)^2*p1^2*p2*frd2
+ 60*p3^3*q2d^2*q1d*cos(q2)^3*p1*p2*frd2
- 7*p3^2*q2d^4*cos(q2)*p2^3*p1*sin(q2)
```

```
+ 5*u2*p3*sin(q2)*p2^3*q1d*frd1
- 14*u2*p3^2*sin(q2)*p1*p2*q1d*cos(q2)*frd1
+ 4*u2*p3^3*cos(q2)^3*frd2^2
- 7*u1*p3^4*cos(q2)^2*p1*q1d^2
+ 10*p3^2*q1d^2*p1*p2^2*q2d*frd2
+ 6*u1*p3^3*cos(q2)*p2^2*q2d^2
+ p3^2*q1d^3*p1*p2^2*frd1
- 7*p3^3*p2^2*q2d^2*q1d^2*p1*sin(q2)*cos(q2)^2
+ 3*u2*p3^2*p1^2*p2*q1d^2
- 8*p3^2*p2^3*q2d^2*q1d*frd1
+ 12*p3^6*q1d^4*cos(q2)^5*sin(q2)
+ 12*p3^5*cos(q2)^3*q1d^3*frd1
+ 10*u1*p3*q2d*q1d*cos(q2)*p1*p2^3
- 6*u1*p3*q2d*q1d*cos(q2)*p2^4
- 10*u2*p3^5*q1d^2*cos(q2)^5
+ 10*p3^2*q2d^3*p2^3*frd2
- 10*u1*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1
- 8*u1*p3^3*q2d*cos(q2)^2*sin(q2)*p2*frd1
+ 2*u1*p3*q2d*sin(q2)*frd1*p2^3
- 4*u1*p3^3*cos(q2)^3*frd2^2
+ 14*u1*p3^3*q2d*q1d*cos(q2)^3*p2^2
+ 20*u1*p3^5*q2d*q1d*cos(q2)^5
- 2*u1*p3*q2d*sin(q2)*frd1*p1*p2^2
- 12*u1*p3^3*q2d^2*cos(q2)^3*p2^2
- 4*u1*p3*q2d*q1d*cos(q2)*p1^2*p2^2
+ p3*q2d^4*sin(q2)*p2^5
+ 32*p3^4*q2d^2*q1d*cos(q2)^4*frd2*p1
+ 16*u2*p3^4*p1*cos(q2)^2*q1d*q2d
+ 18*u1*p3^2*q2d^2*cos(q2)^2*p1*p2^2
```

```
- 18*u1*p3^2*q2d^2*cos(q2)^2*p2^3
+ 22*u1*p3^4*q2d^2*cos(q2)^4*p2
- 2*u1*p3*sin(q2)*p1^2*p2*q2d*frd2
+ u1*p3*q2d^2*cos(q2)*p1^2*p2^2
- u1*p3*q2d^2*cos(q2)*p1*p2^3
- 32*frd2^2*p1*p3^2*cos(q2)*sin(q2)*p2*q2d^2
- 46*frd2^2*p1*p3^3*cos(q2)^2*sin(q2)*q2d^2
+ 2*u1*p3^2*sin(q2)*p1*p2*q1d*cos(q2)*frd1
- 32*p3^4*cos(q2)^3*frd2^2*sin(q2)*q2d^2
- 12*p3^2*cos(q2)^2*frd2^3*q2d*p1
- 6*frd2^2*p1^2*p3^2*cos(q2)*sin(q2)*q1d^2
- 4*p3^3*cos(q2)^2*frd2^2*sin(q2)*p2*q2d^2
- 8*frd2^2*p1*p3^2*cos(q2)*q1d*sin(q2)*p2*q2d
- 8*frd2^2*p1*p3^3*cos(q2)^2*q1d*sin(q2)*q2d
+ 4*frd2^2*p1*p3*cos(q2)*q1d*p2*frd1
+ 4*frd2^2*p1*p3^2*cos(q2)^2*q1d*frd1
+ 32*p3^2*sin(q2)*cos(q2)*q1d*p2*frd1*q2d*frd2*p1
+ 14*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1*q2d*frd2
- 8*p3^4*cos(q2)^3*frd2^2*sin(q2)*q1d^2
- 8*p3^3*cos(q2)^2*frd2^2*q1d*sin(q2)*p2*q2d
- 8*p3^4*cos(q2)^3*frd2^2*q1d*sin(q2)*q2d
+ 4*p3^2*cos(q2)^2*frd2^2*q1d*p2*frd1
+ 4*p3^3*cos(q2)^3*frd2^2*q1d*frd1
- 8*frd2*p1*p3^2*sin(q2)*cos(q2)*q1d^2*p2*frd1
+ 5*p3^2*cos(q2)^2*frd2*frd1^2*q1d*p2
- 6*p3^4*cos(q2)^3*frd2*frd1*sin(q2)*q2d^2
- 28*p3^3*q2d^2*q1d*cos(q2)^3*p2^2*frd2
- 2*p3^2*cos(q2)^2*frd2^2*frd1*q2d*p1
- 4*p3^3*cos(q2)^3*frd2^2*frd1*q2d
```

```
- 8*p3^3*cos(q2)^2*frd2*sin(q2)*q1d^2*p2*frd1
- 12*p3^3*cos(q2)^2*frd2^2*sin(q2)*q1d^2*p1
+ 24*p3^2*q2d^2*q1d*cos(q2)^2*p1*p2^2*frd2
- 8*p3^4*q2d*q1d^3*cos(q2)^3*p2^2*sin(q2)
+ 7*p3^2*q2d*q1d^2*cos(q2)^2*p2^3*frd1
- 11*p3^3*q2d*q1d^2*cos(q2)^3*p2^2*frd1
+ 21*u2*p3^4*p1*cos(q2)^2*q1d^2
- 9*p3*q2d*q1d^2*cos(q2)*p1*p2^3*frd1
+ 4*p3*q2d^2*q1d*cos(q2)*p1^2*p2^2*frd2
+ 10*u1*p3^5*q1d^2*cos(q2)^5
- 10*p3^5*q1d^3*cos(q2)^5*frd1
- 2*p3^2*p1*q2d^3*frd2*p2^2
+ 8*u2*p3^3*p1^2*cos(q2)*q1d^2
+ p3*q2d^3*p2^3*frd1*cos(q2)*p1
+ 9*p3^3*q2d^3*p2^2*frd1*cos(q2)^3
- 3*p3^2*q2d^3*cos(q2)^2*frd1*p2^3
+ 7*p3^4*q2d^3*cos(q2)^4*frd1*p2
- 4*p3*q2d^3*q1d*sin(q2)*p2^4*p1
+ 3*p3^2*q2d^3*cos(q2)^2*frd1*p2^2*p1
+ p3*q2d^4*sin(q2)*p2^3*p1^2
+ 2*u1*frd2*p1^2*p3*sin(q2)*q1d*p2
+ 4*u1*frd2*p1*p3^2*sin(q2)*cos(q2)*q1d*p2
- 2*frd2*p1^2*p3*sin(q2)*q1d^2*p2*frd1
+ p3*cos(q2)*frd1^3*p2^2*q1d
- p3^2*cos(q2)*frd1^2*p2^2*sin(q2)*q2d^2
- p3^2*cos(q2)^2*frd1^2*p2*q2d*frd2
+ 22*u1*p3^4*q2d*q1d*cos(q2)^4*p2
- p2^3*frd1^2*p3*sin(q2)*q2d^2
- 2*p2^2*frd1^2*q2d*p3*cos(q2)*frd2
```

```
+ 4*p3*q2d*q1d^2*cos(q2)*p1^2*p2^2*frd1
+ 2*p3^2*q2d^3*q1d*cos(q2)*p1^2*p2^2*sin(q2)
+ 2*p3*q2d*sin(q2)*frd1^2*p1*p2^2*q1d
+ 2*p3*q2d^3*q1d*sin(q2)*p2^3*p1^2
- 10*u1*p3^2*q2d*q1d*cos(q2)^2*p2^3
- 2*u1*frd2*p1*p3^3*sin(q2)*cos(q2)^2*q1d
+ frd2*p1*p3*cos(q2)*frd1^2*q1d*p2
- u1*p2^2*frd1^2*p3*cos(q2)
- 5*u1*p3*cos(q2)*frd2*p2^2*frd1
- 4*u1*p3^2*cos(q2)^2*frd2^2*p2
+ 12*u1*p3^4*q1d^2*cos(q2)^4*p2
- 2*u1*frd2*p2^2*p3*sin(q2)*q1d*p1
+ 2*u1*p3^2*q1d^2*cos(q2)^2*p1^2*p2
- 4*u1*frd2*p2^2*p3^2*sin(q2)*cos(q2)*q1d
- u1*p3*q1d^2*cos(q2)*p1*p2^3
- 4*u1*frd2^2*p1*p3*cos(q2)*p2
- u2^2*p3^2*sin(q2)*cos(q2)*p1*p2
+ 10*u1*p3^2*q2d*q1d*cos(q2)^2*p1*p2^2
- 8*u2d*p3^2*sin(q2)*q2d*p1*cos(q2)*p2^2
- 6*u2d*p3^4*sin(q2)*q2d*p1*cos(q2)^3
+ u2^2*p3^2*sin(q2)*cos(q2)*p2^2
- u1*p3^3*cos(q2)^3*frd2*frd1
- u1*frd2^2*p1^2*p3*cos(q2)
+ 6*u1*p3^4*q1d^2*cos(q2)^4*p1
+ 4*u2d*p3*sin(q2)*q2d*p1^2*p2^2
- 8*u2d*p3*sin(q2)*q2d*p1*p2^3
+ 4*u2d*p3*sin(q2)*q2d*p2^4
- 2*u2d*p3^5*sin(q2)*q2d*cos(q2)^4
+ 4*u1d*p3^5*sin(q2)*cos(q2)^4*q2d
```

```
+ u1d*frd2*p1*p3^3*cos(q2)^3
+ 2*u1d*p3*cos(q2)*frd2*p2^3
+ 2*u1d*p3^3*cos(q2)^3*frd2*p2
- 4*u1d*p3^3*sin(q2)*q1d*p1*p2*cos(q2)^2
- 6*u1d*p3^2*sin(q2)*p2^2*q2d*cos(q2)*p1
+ 6*u1d*p3^2*sin(q2)*p2^3*q2d*cos(q2)
+ 6*u1d*p3^4*sin(q2)*p2*q2d*cos(q2)^3
+ u1d*p2^3*frd1*p3*cos(q2)
+ u1d*p2*frd1*p3^3*cos(q2)^3
+ 2*u1d*p3*sin(q2)*q1d*p2^4
- 4*u1d*p3*sin(q2)*q1d*p1*p2^3
+ 2*u1d*p3^5*sin(q2)*cos(q2)^4*q1d
+ 2*u1d*p3^2*cos(q2)^2*frd2*p2^2
+ 6*u2d*p3^2*sin(q2)*q2d*p1^2*cos(q2)*p2
- 4*u1*p3^4*cos(q2)^3*frd2*sin(q2)*q1d
- 6*u1*p3^3*q1d^2*cos(q2)^3*p2^2
- 4*u1*p3^2*cos(q2)^2*frd2^2*p1
+ u1*p3*q1d^2*cos(q2)*p1^2*p2^2
+ 2*u1*p3^2*q1d^2*cos(q2)^2*p1*p2^2
+ 13*u1*p3^3*q1d^2*cos(q2)^3*p1*p2
- 4*u1*p3^2*q1d^2*cos(q2)^2*p2^3
+ 4*u1d*p3^3*sin(q2)*q1d*p2^2*cos(q2)^2
- 2*u1d*p3^3*sin(q2)*cos(q2)^2*q2d*p1*p2
- 2*u1d*p3*sin(q2)*q2d*p2^4
- 2*u1dd*p1*p2^2*p3^2*cos(q2)^2
+ u1dd*p3*cos(q2)*p1^2*p2^2
+ u1d*p2^2*frd1*p3^2*cos(q2)^2
+ 2*u2dd*p3*cos(q2)*p1*p2^3
+ 2*u2dd*p3^3*cos(q2)^3*p1*p2
```

```
- 2*u1dd*p3^3*cos(q2)^3*p1*p2
- 2*u1d*p3*sin(q2)*q2d*p1^2*p2^2
+ 3*tc2ddd*p1^2*p2^2*p3^2*cos(q2)^2
+ 2*u2dd*p1^2*p2*p3^2*cos(q2)^2
- 2*u2dd*p1*p2^2*p3^2*cos(q2)^2
- 3*u2d*frd2*p1*p3^3*cos(q2)^3
- 2*u2d*p3^2*cos(q2)^2*frd2*p2^2
- u2d*frd2*p1^2*p3^2*cos(q2)^2
+ 2*u2d*p3^5*sin(q2)*cos(q2)^4*q1d
- 6*tc2ddd*p1*p2^3*p3^2*cos(q2)^2
- u2dd*p3*cos(q2)*p1^2*p2^2
+ 4*u1d*p3*sin(q2)*q2d*p1*p2^3
- u1d*p2^2*frd1*p3*cos(q2)*p1
+ 2*u1d*p3^3*sin(q2)*cos(q2)^2*q2d*p2^2
- u1d*frd2*p1*p2*p3^2*cos(q2)^2
- u1d*frd2*p1^2*p3*cos(q2)*p2
- u1d*frd2*p1*p3*cos(q2)*p2^2
+ 2*u1d*p3*sin(q2)*q1d*p1^2*p2^2
- 2*u2d*p2^2*frd1*p3^2*cos(q2)^2
- 2*u2d*p2^3*frd1*p3*cos(q2)
- 2*u2d*p2*frd1*p3^3*cos(q2)^3
- 32*p3^6*cos(q2)^3*q1d^3*q2d*sin(q2)
- 3*u2d*frd2*p1*p3*cos(q2)*p2^2
+ 2*u2d*p3^2*sin(q2)*p2^3*q2d*cos(q2)
+ 2*u2d*p3^4*sin(q2)*p2*q2d*cos(q2)^3
+ 2*u1d*p3^4*cos(q2)^4*frd2
+ 2*u2d*p3^3*sin(q2)*cos(q2)^2*q2d*p2^2
+ 2*u2d*frd2*p1*p2*p3^2*cos(q2)^2
+ 3*u2d*frd2*p1^2*p3*cos(q2)*p2
```

```
+ 2*u2d*p3^3*sin(q2)*q1d*p2^2*cos(q2)^2
- 2*u2d*p3^3*sin(q2)*cos(q2)^2*q2d*p1*p2
- 2*u2d*p3^3*sin(q2)*q1d*p1*p2*cos(q2)^2
+ 2*u2d*p3^2*sin(q2)*cos(q2)*q1d*p2^3
+ 2*u2d*p3^4*sin(q2)*cos(q2)^3*q1d*p2
+ u2d*p3^2*cos(q2)^2*frd1*p1*p2
+ 2*u2d*p2^2*frd1*p3*cos(q2)*p1
- 2*u2d*p3^2*sin(q2)*cos(q2)*q1d*p2^2*p1
+ u1dd*p3^5*cos(q2)^5 - 26*u1*p3^5*cos(q2)^3*q1d*q2d
- 16*u1*p3^3*sin(q2)*cos(q2)^2*p1*q2d*frd2
- 2*u2*p3^5*q2d^2*cos(q2)^5
+ 6*u1*p3^4*sin(q2)*cos(q2)^3*q1d*frd1
- 3*u1^2*p3*sin(q2)*p1*p2^2
- 5*u1^2*p3^3*sin(q2)*p2*cos(q2)^2
- 3*u1^2*p3^2*sin(q2)*cos(q2)*p2^2
- u1^2*p3^2*sin(q2)*cos(q2)*p1*p2
- 5*u2*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1
+ 6*u1*p3^2*sin(q2)*cos(q2)*q1d*p2^2*frd1
- 4*u2*p3*sin(q2)*p1*p2^2*q1d*frd1
- 7*u2*p3^3*sin(q2)*p1*cos(q2)^2*q1d*frd1
- 4*u2*p3^4*q2d*q1d*cos(q2)^2*p2
- 2*u2*p3^3*q2d*q1d*p2^2*cos(q2)
+ 50*p3^3*cos(q2)*q2d^3*frd2*p2^2
- 3*u1^2*p3^4*sin(q2)*cos(q2)^3
+ 36*u2*p3^3*q2d^2*p1*p2*cos(q2)
+ 4*u2*p3^2*q2d*q1d*p1*p2^2 - 8*p3^2*p1^2*p2*q2d^3*frd2
- 4*u2*p3^2*q2d*q1d*p2^3 + 18*u2*p3^3*q2d*q1d*cos(q2)*p1*p2
+ 25*u2*p3^4*q2d^2*cos(q2)^2*p1
+ 25*u2*p3^3*q2d^2*cos(q2)^3*p2^2
```

```
- 28*u2*p3^3*q2d^2*cos(q2)*p2^2 - 6*u2*p3^2*q2d^2*p1*p2^2
- 2*u2*p3^5*q2d*q1d*cos(q2)^3
- 2*u2*p3^2*q2d^2*cos(q2)^2*p2^3
+ 19*u1*p3^3*q2d^2*cos(q2)^3*p1*p2
- 14*u2*p3^3*q2d*q1d*cos(q2)^3*p1*p2
+ 5*u2*p3^2*q2d*sin(q2)*frd1*p1*p2*cos(q2)
- 10*p3^2*p1^2*q2d*frd2*p2*q1d^2 - u1*p3^2*p1*p2^2*q1d^2
- 6*u1*p3^2*p1*p2^2*q1d*q2d
+ 16*u2*p3^2*q2d^2*cos(q2)^2*p1*p2^2
- 32*u2*p3^3*q2d^2*cos(q2)^3*p1*p2
- 12*u1*p3^3*p1*p2*cos(q2)*q1d^2
- 5*p2^3*frd1*p3^2*q1d^2*q2d
+ 16*u2*p3^4*q2d*sin(q2)*frd2*cos(q2)^3
- 18*u2*p3^4*q2d^2*cos(q2)^4*p1
- 9*u2*p3^4*q2d^2*cos(q2)^2*p2
- 6*u2*p3*q2d^2*cos(q2)*p2^2*p1^2
+ 16*u2*p3*q2d*sin(q2)*frd2*p1^2*p2
- 14*u1*p3^4*cos(q2)^2*q1d^2*p2
+ 2*u2*p3^4*q2d*q1d*cos(q2)^4*p2
+ 6*u2*p3^4*q2d^2*cos(q2)^4*p2
+ 3*u2*p3^2*q2d*cos(q2)*sin(q2)*p2^2*frd1
+ 2*u2*p3^2*q2d*q1d*cos(q2)^2*p2^3 - 10*u1*p3^2*p1*p2^2*q2d^2
+ 2*u1*p3^2*p2^3*q1d^2 + 6*u1*p3^2*p2^3*q1d*q2d
+ 13*u2*p3^3*q2d*cos(q2)^2*sin(q2)*p2*frd1
+ 4*u1*p3^3*p2^2*cos(q2)*q1d^2 - 4*u2*p3^3*cos(q2)*q1d^2*p2^2
- 14*u2*p3^2*q2d^2*cos(q2)^2*p1^2*p2
+ 38*u2*p3^3*q2d*cos(q2)^2*sin(q2)*frd2*p1
- 16*u2*p3*q2d*sin(q2)*frd2*p2^2*p1
+ u2*p3*q1d^2*cos(q2)*p1^2*p2^2
```

```
+ 2*u2*p2^2*frd1^2*p3*cos(q2) + 3*u2*p3*cos(q2)*frd2*p2^2*frd1
- 6*u2*p3^2*q1d^2*cos(q2)^2*p1^2*p2
- 18*u2*p3^4*q1d^2*cos(q2)^4*p1
+ 6*u2*p3^3*q1d^2*cos(q2)^3*p2^2
+ 8*u2*p3^2*cos(q2)^2*frd2^2*p1
+ 7*p3^4*cos(q2)^2*q1d^2*p2*frd1*q2d
+ 11*u2*p3*q2d^2*cos(q2)*p2^3*p1
- 2*u2dd*p3^3*cos(q2)^3*p2^2 + 10*u1*p3^2*p2^3*q2d^2
- 26*u1*p3^4*cos(q2)^2*q1d*p2*q2d
- 30*u1*p3^4*cos(q2)^2*p2*q2d^2
+ 5*u2*p3^4*q2d*sin(q2)*frd1*cos(q2)^3
- 5*u2*p3*q2d*sin(q2)*frd1*p2^3
+ 5*u2*p3*q2d*sin(q2)*frd1*p1*p2^2
- 5*u2*p3*q2d^2*cos(q2)*p2^4 - u2*p3^2*q2d^2*p2^3
+ 21*p3^3*cos(q2)*q1d^2*p2^2*frd1*q2d
+ 2*p3^2*q2d^3*frd1*p2^3 - 5*u2*u1*p3*sin(q2)*p2^3
- 3*u2^2*p3*sin(q2)*p1^2*p2
- 4*u2^2*p3^2*sin(q2)*cos(q2)*p1^2
- 5*u2^2*p3^3*sin(q2)*p1*cos(q2)^2
- 36*p3^4*p1*q2d^2*cos(q2)^2*frd2*q1d
- 4*u2*p3^3*sin(q2)*cos(q2)^2*q2d*frd2*p2
- 2*u2^2*p3^3*sin(q2)*p2*cos(q2)^2
- 56*p3^3*p1*q2d^2*cos(q2)*frd2*q1d*p2
- 13*p3^4*p1*p2*q2d^4*cos(q2)*sin(q2)
+ 18*u2*p3^2*q2d*cos(q2)*sin(q2)*frd2*p1^2
- 20*u2*p3^2*q2d*sin(q2)*frd2*p2^2*cos(q2)
+ 3*u2^2*p3*sin(q2)*p1*p2^2
+ 18*u2*p3^2*q2d*sin(q2)*frd2*p1*p2*cos(q2)
+ 12*p3^3*cos(q2)*q1d^3*p2*frd1*p1
```

```
+ 8*u1*p3*sin(q2)*p1*p2^2*q1d*frd1

- 8*u1*p3*sin(q2)*p2^3*q1d*frd1 - u2dd*p3*cos(q2)*p2^4

- 34*u1*p3^2*sin(q2)*p1*p2*q2d*cos(q2)*frd2

- 10*u1*p3*sin(q2)*p1*p2^2*q2d*frd2 + 3*u1^2*p3*sin(q2)*p2^3

+ 9*p3^4*cos(q2)*p2^2*q2d^4*sin(q2)

- 26*p3^3*cos(q2)*q1d*p2^2*frd1*q2d^2

+ 4*u2*p3^2*sin(q2)*cos(q2)*q1d*p2^2*frd1

+ 16*u1*p3^2*sin(q2)*p2^2*q2d*cos(q2)*frd2

+ 14*p3^4*cos(q2)^2*q1d^3*p2*frd1

+ 12*u1*p3*sin(q2)*p2^3*q2d*frd2

+ 8*u1*p3^3*sin(q2)*cos(q2)^2*q1d*p2*frd1

+ 2*u2*u1*p3^4*sin(q2)*cos(q2)^3 + 7*u2*p3^2*q2d^2*p1^2*p2

- 48*p3^5*cos(q2)^2*p2*q2d^2*q1d^2*sin(q2)

- 20*p3^3*cos(q2)*q2d*frd2*p2*q1d^2*p1

- 26*p3^4*cos(q2)^2*p2*q2d^3*frd2

- 52*p3^5*cos(q2)^3*q2d*frd2*q1d^2

- 62*p3^4*cos(q2)^2*q2d*frd2*q1d^2*p1

- 24*u1*p3^3*sin(q2)*cos(q2)^2*q2d*frd2*p2

- 20*u1*p3^4*sin(q2)*cos(q2)^3*q2d*frd2

- 18*p3^3*p1^2*q2d*cos(q2)*frd2*q1d^2

+ 4*u2*p3^3*p1*p2*cos(q2)*q1d^2

- 2*p3^4*cos(q2)*p2^2*q2d^3*q1d*sin(q2)

- frd2^3*p1^3*q2d + p2^3*frd1^3*q1d

+ u1d*p2^4*frd1 + frd2^2*p1^2*q1d*p2*frd1

+ 2*u2*frd2*p1*p2^2*frd1 + u2*p2^3*frd1^2

- 34*p3^4*p1*q2d^3*cos(q2)^2*frd2

+ 14*p3^3*p1*q1d*p2*frd1*cos(q2)*q2d^2

- 2*frd2^2*p2^2*frd1*q2d*p1 + u2d*frd2*p1^3*p2

+ 8*p3^2*p1*q1d*p2^2*frd1*q2d^2 + u2*frd2^2*p1^3
```

```
- 20*p3^2*p1*q2d^2*frd2*p2^2*q1d + frd2*p2^3*frd1^2*q1d
- p2^3*frd1^2*q2d*frd2 - u1*frd2*p2^3*frd1
- u1*frd2^2*p1^2*p2 + tc2ddd*p2^6 - u2dd*p1^3*p2^2
+ 2*u2dd*p1^2*p2^3 - u2dd*p1*p2^4 - u2d*p2^4*frd1
- tc2ddd*p1^3*p2^3 + 3*tc2ddd*p1^2*p2^4 - 3*tc2ddd*p1*p2^5
+ u1dd*p1^2*p2^3 - 2*u1dd*p1*p2^4 - u1d*p2^3*frd1*p1
- u1d*frd2*p1^2*p2^2 + u1d*frd2*p1*p2^3 + u2d*p2^3*frd1*p1
- u2d*frd2*p1^2*p2^2 - u1*frd2*p1*p2^2*frd1 + u1dd*p2^5
+ u2*u1*p3*sin(q2)*p1^2*p2
+ 14*u2*u1*p3^2*sin(q2)*cos(q2)*p1*p2
+ frd2*p1*p2^2*frd1^2*q1d - u1*p2^3*frd1^2
- 5*p3^2*sin(q2)*cos(q2)*q1d^2*p2^2*frd1^2
+ 24*p3^6*q2d^3*cos(q2)^5*q1d*sin(q2)
- 6*p3^4*q2d^2*cos(q2)^4*q1d*p2*frd1
+ 4*u2*u1*p3*sin(q2)*p1*p2^2
+ 7*u2*u1*p3^3*sin(q2)*p1*cos(q2)^2
+ 16*p3^3*sin(q2)*p1*q2d*cos(q2)^2*frd2*q1d*frd1
- 12*p3^5*q2d^2*cos(q2)^5*q1d*frd1
- 4*p3^3*sin(q2)*cos(q2)^2*q1d^2*p2*frd1^2
+ 12*p3^5*q2d^4*cos(q2)^4*sin(q2)*p2
+ 26*p3^4*q2d^3*cos(q2)^4*frd2*p1
- 2*p3*q2d^2*sin(q2)*q1d^2*p1^2*p2^3
+ p3*q2d^2*sin(q2)*q1d^2*p1*p2^4
+ 6*p3*q2d^3*cos(q2)*p2^4*frd2
- 50*p3^6*q2d^3*cos(q2)^3*q1d*sin(q2)
- 2*p3*q2d^4*sin(q2)*p2^4*p1
+ 2*p3*q2d^3*q1d*sin(q2)*p2^5
+ 5*u2*frd2^2*p1^2*p3*cos(q2)
+ 3*u2*u1*p3^3*sin(q2)*p2*cos(q2)^2
```

```
+ 3*tc2ddd*p2^2*p3^4*cos(q2)^4
- 6*frd2^3*p1^2*p3*cos(q2)*q2d + 3*tc2ddd*p2^4*p3^2*cos(q2)^2
- p3*q2d^3*p2^4*frd1*cos(q2) + 4*u2*p3^5*cos(q2)^3*q2d^2
+ frd2^2*p1^2*q1d*p3*cos(q2)*frd1
- 2*frd2^2*p1^2*q1d*p3^2*sin(q2)*cos(q2)*q2d
- u2dd*p1*p3^4*cos(q2)^4 - frd2^2*p1^3*p3*sin(q2)*q1d^2
- 2*frd2^2*p1^2*q1d*p3*sin(q2)*p2*q2d
+ 2*u1dd*p3^3*cos(q2)^3*p2^2 - 4*p2^2*frd1*p3^3*q1d^3*cos(q2)
+ 12*p3^5*q2d^2*cos(q2)^4*sin(q2)*q1d^2*p1
- 3*u2^2*p3^4*sin(q2)*cos(q2)^3
+ 20*p3^3*cos(q2)*q2d^2*frd2*p2^2*q1d
- 52*p3^4*cos(q2)^2*q2d^2*frd2*p2*q1d
- 15*frd2^2*p1^2*p3*sin(q2)*p2*q2d^2
- 15*frd2^2*p1^2*p3^2*sin(q2)*cos(q2)*q2d^2
- 14*u1*p3^3*p1*p2*cos(q2)*q2d^2
+ 5*p3^2*p1*q1d^2*p2^2*frd1*q2d
- 10*p3^3*cos(q2)*q2d^3*p2^2*frd1
- 66*p3^5*cos(q2)^2*p2*q2d^3*q1d*sin(q2)
- 22*p3^4*cos(q2)*p2*q2d^2*q1d^2*p1*sin(q2)
- 16*p3^6*cos(q2)^3*q1d^4*sin(q2)
+ 20*frd2*p2^2*p3^3*cos(q2)*q1d^2*q2d
- 4*p3^4*p1^2*cos(q2)*q1d^4*sin(q2)
- 16*p3^5*cos(q2)^2*q2d*q1d^3*p1*sin(q2)
- 8*p3^4*cos(q2)^2*q2d^3*p2*frd1 + 7*p3^3*p2^3*q2d^4*sin(q2)
- 16*p3^5*cos(q2)^2*q1d^4*p1*sin(q2) - u1*p3^2*p1^2*p2*q1d^2
- 14*p3^4*p1*p2*q2d^3*q1d*cos(q2)*sin(q2)
- 14*p3^3*q2d^3*p1*p2^2*q1d*sin(q2)
- 68*p3^3*cos(q2)*p2*q2d^3*frd2*p1
- 25*p3^5*cos(q2)^2*p2*q2d^4*sin(q2)
```

```
+ 10*u1*p3^3*p1*p2*q1d*cos(q2)*q2d
- 25*p3^5*q2d^2*cos(q2)^2*q1d^2*p1*sin(q2)
+ 36*p3^6*q2d^2*cos(q2)^5*sin(q2)*q1d^2
+ 36*p3^5*q2d^3*cos(q2)^4*q1d*sin(q2)*p2
- 15*p3^3*p1*q1d^2*p2*frd1*cos(q2)*q2d
- 6*u2*u1*p3^2*sin(q2)*cos(q2)*p2^2)/...
(cos(q2)^2*p3^2 - p1*p2 + p2^2)^3
```

A.3.1 Utilizing Large Tests

Obviously, translating the above tests into valid LaTeXequations is impractical due to their size. The resulting equations would be much larger than a single page. However, this does not make implementing the tests difficult, as the software package use to derive them (Matlab [35]) can also evaluate the results.

This separation between derivation and implementation is important. The derivation of the NLAR tests is a symbolic derivation problem based on the model equations, and is performed off-line. The application of the tests derived by the symbolic system requires only simple algebraic and trigonometric manipulations and must be performed repeatedly in real time. Larger NLAR tests will usually be automatically translated into a more efficient compiled language such as C before implementation to maximize efficiency.

Bibliography

[1] Karl J. Åström and Björn Wittenmark. *Computer Controlled Systems.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] L. C. Bares, L. S. Conley, and B.R. Thompson. Rosie: A Mobile Worksystem for D & D: Overview of System Capabilities and CP-5 Reactor Application . In *Proceedings of the ANS 7th Topical Meeting on Robotics and Remote Systems*, pages 471–477, Augusta, GA, April-May 1997.

[3] Fanping Bu and Bin Yao. Nonlinear Model Based Coordinated Adaptive Robust Control of Electro-hydraulic Robotic Arms via Overparametrizing Method. In *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[4] Fabrizio Caccavale and Ian D. Walker. Observer-Based Fault Detection for Robot Manipulators. In *IEEE International Conference on Robotics and Automation*, pages 2881–2887, Albuquerque, NM, April 1997.

[5] E. Y. Chow and A. S. Willsky. Analytical Redundancy and the Design of Robust Failure Detection Systems. *IEEE Transactions on Automatic Control*, AC-29(7):603–614, July 1984.

[6] L. Conley, Dr. W.R. Hamel, and B.R. Thompson. Rosie: A Mobile Worksystem for Decontamination and Dismantlement Operations. In *Proceedings of the ANS 6th Topical Meeting on Robotics and Remote Systems*, pages 231–238, Monterey, CA, February 1995.

[7] B Daachi, A. Benallegue, and N. K. M'Sirdi. A Stable Neural Adaptive Force Controller for a Hydraulic Actuator. In *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[8] Department of Energy, Washington, D.C. *Environmental Restoration and Waste Management 5-Year Plan, Fiscal Years 1994-1998*, January 1993. DOE/S-00097P, Vol. 1-2.

[9] Department of Energy, Federal Energy Technology Center, Morgantown, WV. *Environmental Waste Management Project Fact Sheet*, October 1997. http://www.fetc.doe.gov/publications/factsheets/ewm/index.html.

[10] B.S. Dhillon. *Robot Reliability and Safety.* Springer-Verlag, New York, NY, 1991.

[11] 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.

[12] X. Ding, L. Guo, and T. Jeinsch. A Characterization of Parity Space and Its Application to Robust Fault Detection. *IEEE Transactions on Automatic Control*, 44(2):337–343, February 1999.

[13] W.E. Dixon, Darren M. Dawson I.D. Walker, and J.P. Hartranft. Fault Detection for Robot Manipulators with Parametric Uncertainty: A Prediction-Error-Based Approach. *IEEE Transactions on Robotics and Automation*, 16(6):689–699, 2001.

[14] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems.* Addison-Wesley, Reading, MA, 1995.

[15] H. Hammouri, M. Kinnaert, and E. H. El Yaagoubi. Observer-Based Approach to Fault Detection and Isolation for Nonlinear Systems. *IEEE Transactions on Automatic Control*, 44(10):1879–1884, October 1999.

[16] B. M. Harpel, J. B. Dugan, I. D. Walker, and J. R. Cavallaro. Analysis of Robots for Hazardous Environments. In *Proc. IEEE Annual Reliability and Maintainability Symposium*, pages 111–116, Philadelphia, PA, January 1997.

[17] Marcel Honegger and Peter Corke. Model-Based Control of Hydraulically Actuated Manipulators. In *IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[18] RedZone Robotics Inc. Rosie. http://www.redzone.com/downloads/Rosie.pdf.

[19] 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.

[20] A. Isidori. *Nonlinear Control Systems.* Springer-Verlag, London,UK, 1995.

[21] Hassan K. Khalil. *Nonlinear Systems, Second Edition.* Prentic-Hall, Upper Saddle River, NJ, USA, 1996.

[22] M. H. E. Larcombe and J. R. Halsall. *Robotics in Nuclear Engineering.* Graham & Trotman Ltd., London, 1984.

[23] M. L. Leuschen. Robot Reliability Through Fuzzy Markov Models. Master's thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, 1997.

[24] M. L. Leuschen, I. D. Walker, and J. R. Cavallaro. Robot Reliability Through Fuzzy Markov Models. In *Proc. IEEE Annual Reliability and Maintainability Symposium*, pages 209–214, Anaheim, CA, January 1998.

[25] M. L. Leuschen, I. D. Walker, and J. R. Cavallaro. Investigation of Reliability for Hydraulic Robots Using Analytical Redundancy. In *Proc. IEEE Annual Reliability and Maintainability Symposium*, pages 122–128, Washington, DC, January 1999.

[26] M. L. Leuschen, I. D. Walker, and J. R. Cavallaro. Monitoring and Diagnostics for a Hydraulic Robot in Hazardous Environments. In *Proceedings of the ANS 8th Topical Meeting on Robotics & Remote Systems*, Pittsburgh, PA, April 1999.

[27] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Robot Reliability Using Fuzzy Fault Trees and Markov Models. In *SPIE Proceedings on Sensor Fusion and Distributed Robotic Agents*, pages 73–91, Boston, MA, 1996.

[28] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Analytical redundancy data analysis of a hydraulic robot testbed. Technical Report TR9911, Department of Electrical and Computer Engineering, Rice University, Houston, TX, 1999.

[29] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Evaluating the Reliability of Prototype Degradable Systems. *Reliability Engineering and System Safety*, 72(1):9–20, 2001.

[30] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Experimental AR Fault Detection Methods for a Hydraulic Robot. In *Proceedings of the ANS 9th Topical Meeting on Robotics and Remote Systems*, page F131, Seattle, WA, March 2001.

[31] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Nonlinear analytical redundancy tests for a hydraulic manipulator. Technical Report TREE0101, Depart-

ment of Electrical and Computer Engineering, Rice University, Houston, TX, 2001.

[32] M.L. Leuschen, I.D. Walker, and J.R. Cavallaro. Robotic Fault Detection Using Nonlinear Analytical Redundancy. In *(To appear) IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002.

[33] M.L. Leuschen, I.D. Walker, J.R. Cavallaro, R. W. Gamache, and M. Martin. Experimental AR Fault Detection Methods for a Hydraulic Robot. In *Proceedings of the 18th International Systems Safety Conference*, pages 402–409, Fort Worth, TX, Sept 2000.

[34] Jean-François Magni and Philippe Mouyon. On Residual Generation by Observer and Parity Space Approaches. *IEEE Transactions on Automatic Control*, 39(2):441–447, February 1994.

[35] MathWorks. Matlab. http://www.mathworks.com/products/matlab/.

[36] H. Nijmeijer and A.J. Van der Schaft. *Nonlinear Dynamical Control Systems.* Springer-Verlag, New York, USA, 1990.

[37] Mattias Nyberg and Lars Nielsen. A Universal Chow-Willsky Scheme and Detectability Criteria. *IEEE Transactions on Automatic Control*, 45(1):152–156, January 2000.

[38] Claudio De Parsis and Alberto Isidori. A Geometric Approach to Nonlinear Fault Detection and Isolation. *IEEE Transactions on Automatic Control*, 46(6):853–865, June 2001.

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

[40] Wolfram Stadler. *Analytical Robotics and Mechatronics.* Mc-Graw-Hill, Inc., New York, NY, 1995.

[41] M. Staroswiecki, J. P. Cassar, and G. Comtet-Varga. Analytic Redundancy Relations for State Affine Systems. In *Proceedings of the Fourth European Control Conference*, Brussels, Belgium, 1997.

[42] M. Staroswiecki and G. Comtet-Varga. Analytical Redundancy Relations for Fault Detection and Isolation in Algebraic Dynamic Systems. *Automatica*, 37(5):687–699, 2001.

[43] Alexander B. Trunov and Marios M. Polycarpou. Robust Nonlinear Fault Diagnosis: Application to Robotic Systems. In *IEEE International Conference on Control Applications*, pages 1424–1429, Piscataway, NJ, April 1999.

[44] M. L. Visinsky, J. R. Cavallaro, and I. D. Walker. Robotic Fault Detection and Fault Tolerance: A Survey. *Reliability Engineering and System Safety*, 46(2):139–158, 1994.

[45] M.L. Visinsky, J.R. Cavallaro, and I.D. Walker. A Dynamic Fault Tolerance Framework for Remote Robots. *IEEE Transactions on Robotics and Automation*, 11(4):477–490, 1995.

[46] I. D. Walker and J. R. Cavallaro. The Use of Fault Trees for the Design of Robots for Hazardous Environments. In *Proc. IEEE Annual Reliability and Maintainability Symposium*, pages 229–235, Las Vegas, NV, January 1996.

[47] I. D. Walker, J. R. Cavallaro, and M. L. Leuschen. Keeping the Analog Genie in the Bottle: A Case for Digital Robots. In *IEEE International Conference on Robotics and Automation*, pages 1063–1070, Detroit, MI, May 1999.

[48] 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.

[49] F. M. White. *Fluid Mechanics*. McGraw-Hill, Inc., USA, 1994.

[50] A.N. Zhirabok and O.V. Preobragenskaya. Instrument Fault Detection in Nonlinear Dynamic Systems. In *Proceedings of the 1993 IEEE International Conference on Systems, Man, and Cybernetics*, pages 114–119, Le Touquet, France, Oct 1993.