1-1-2003

Adaptive control and parameter estimation of dynamically mismatched robotic manipulators

Jonathan Olson
Iowa State University

Adaptive control and parameter estimation of dynamically mismatched robotic manipulators

by

Jonathan Olson

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Greg R. Luecke (Major Professor)
Adrian Sannier
Nicoli Elia

Iowa State University

Ames, Iowa

2003

Graduate College

Iowa State University

This is to certify that the master's thesis of

Jonathan Olson

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION AND LITERATURE REVIEW

Abstract

This thesis presents the development of an adaptive controller based on mismatched dynamic systems. Through the simulated control of the SAM manipulator, an intelligent assist device, the controller's performance and stability are analyzed. Two dynamic models, a simple model based on a revolute-prismatic manipulator and a complex model based on a three-revolute manipulator, are derived to describe the SAM's motion. Both models correctly describe the kinematic motion of the SAM however the complex model better captures the true manipulator dynamics. The adaptive controller is developed for the simple dynamic model of the SAM manipulator. The controller is then applied to drive a more complex SAM dynamic model. The simulation results show that even though differing dynamic terms exist, the adaptive controller is able to adjust the unknown parameters to significantly improve performance when compared to a non-adaptive computed torque controller. A step change in the payload mass was also tracked and properly identified by the adaptive controller.

Introduction to Material Handling by Mechanical Means

Material handling in the workplace is carried out through three distinct methods. These methods include fully automated, such as programmable robots, manual, which is characterized by human work, and a hybrid of the manual-automated methods.

Since their introduction in the 1950s [1], robotic manipulators have spread to a wide variety of applications. Ideally suited for long run highly repetitive tasks, robotic manipulators perform such workplace activities as welding and assembly. By performing these tasks the robotic manipulator can assure a higher degree of accuracy and repeatability than a human, while also providing greater workplace safety by handling heavy or awkward work pieces. However, to justify the cost of a robotic manipulator, long run tasks are needed.

Manual material handling is highlighted by the flexibility and dexterity of a human worker. A person exhibits skills lacking in a programmable robot such as problem solving

and awareness of objects in the workspace. However, injury and mistakes can result when the task requires high speed, becomes repetitive, or requires movement of heavy objects. According to the Bureau of Labor Statistics [2], more than 5,215,600 workers were injured in 2001 resulting in over 1.5 million missed days of work. Over 400,000 injuries to the back were reported. Lifting objects causes many of these injuries. The addition of mechanical aids such as powered lifts to the workplace would help reduce these injuries.

Hybrid manual-automated systems are the third method for material handling in the workplace. Devices such as the powered lift fall into this category. These devices, in their most basic form, augment or replace human strength mechanically yet still require a person to operate. When used correctly hybrid manipulators reduce the stress and fatigue on the worker. A more complex hybrid manipulator referred to as an intelligent assist device (IAD) can become a mechanical extension of a person. These devices easily adapt to different tasks, augment the operator's strength over a wide range of motion, and can also aid in path planning and obstacle avoidance. One such manipulator is the Simple Air Manipulator (SAM) located at Iowa State University.

Review of SAM Configuration

The SAM is a heavy lift assist device. Shown in Figure 1.1, the SAM was initially powered exclusively in the vertical direction by a pneumatic actuator. This removed the strain of lifting the payload from the operator. However, the operator must supply the planar forces needed to move and orient the payload, overcoming both the inertia of the SAM and the payload.

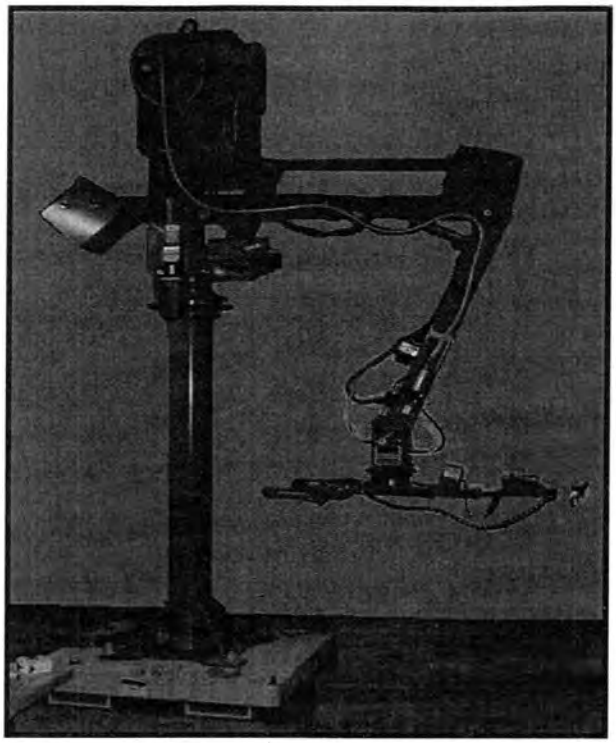

Figure 1.1: Simple Air Manipulator (SAM)

K. L. Tan [3] completed the initial SAM configuration and control design, which led to a fully actuated manipulator. The modifications included adding servomotors at three joint positions; base, elbow, and wrist, as shown in Figure 1.2.

Figure 1.2: Base, Elbow, and Wrist joints

The handle assembly, shown in Figure 1.3, is connected to the end-effector and outfitted with two force transducers, an emergency stop, and an activation switch. An onboard PC executes the control logic in C++. A Servo To Go data acquisition card reads the servo encoder and force transducer signals. The controller then interprets this data and the appropriate control signals are then sent to the servomotors. The operator defines the direction and velocity of the end-effector by applying forces to the handle. The direction and magnitude of the forces are decomposed into desired joint positions by the control logic. A proportional plus derivative controller then drives the error between the actual and desired joint positions to zero.

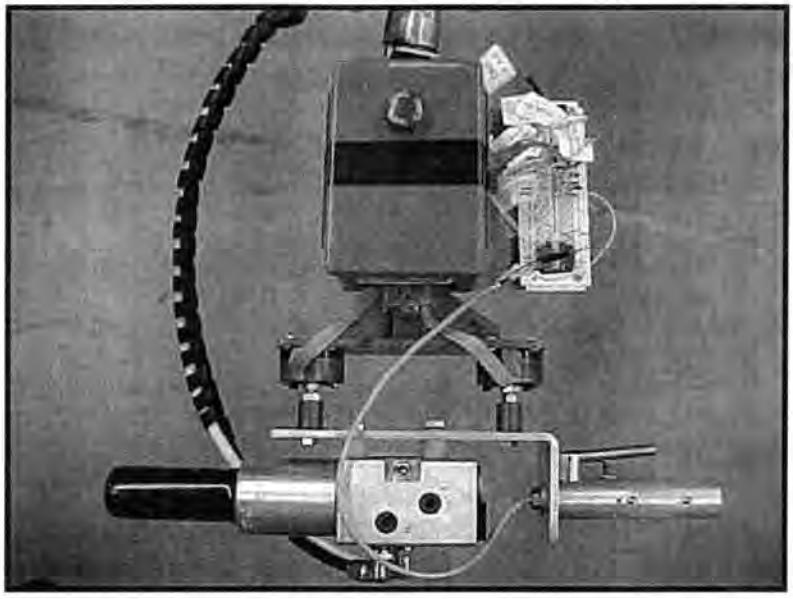

Figure 1.3: Handle Assembly With Force Transducers and Vertical Control

Research Problem and Objectives

By combining the intelligence and the visual and tactile senses of a human with the strength and robustness of a robot, a perfect solution is found to the IAD. However, a complication still exists, how to best control an IAD. The lack of a predefined trajectory and the need to account for a wide variety of payloads makes designing a controller for an IAD difficult. Robotic manipulators operate in a well-defined environment where the trajectory, the weight and location of the work piece, and the task are precisely defined. This allows

optimized control, minimizing the perturbations caused by changes in payload and configuration. With the SAM, operator safety is of greatest importance. While there are several advantages to having the operator in the workspace of the manipulator, they could be seriously injured if the manipulator were to become unstable. Performance is still important and necessary to properly control the manipulator when handling different payloads. The controller must be able to ensure operator safety and at the same time provide smooth predictable motion.

The SAM's current PD controller is optimized for an unloaded state and gain scheduling is used to compensate for the change in inertia seen by the base joint as the end-effector moves away from the base. Using inverse plant control would eliminate the need for gain scheduling by linearizing the change in inertia of the plant. However, to have true linearization the models parameters must be known precisely. This is where the use of an adaptive element in the control structure would be of great advantage. Proper identification of the payload weight and other variables would result in stable, safe, and consistent performance over the entire workspace. The SAM is a complex manipulator with several closed chain linkages. As a result the true dynamic equations are extremely complex and lengthy. Designing the adaptive element based on these equations would be difficult and error prone. Ideally, a simpler model that correctly represents the SAM kinematically while still capturing the basic dynamics could be used to design the controller.

Objectives:

- Design an adaptive controller for the SAM manipulator using a simple model of the SAM that is kinematically correct but dynamically insufficient to completely represent the true manipulator. This controller will then be used to drive a more complex SAM model, which will act as the true plant.
- Illustrate through simulation that the addition of the adaptive element to the controller gives better performance when compared to a non-adaptive computed torque controller.
- Show that the adaptive controller can still adapt when mismatched model dynamics are present. Showing that the mass matrix and the centrifugal and Coriolis vector of

the simplified model converge to the corresponding characteristics of the complex model will prove this.

- Show parameter identification can properly sense and quantify a change in payload weight.

Literature Review

Developing a dynamically true model of the SAM is a difficult task. The development of an adaptive controller using these equations would be harder still. One of the questions this research studies is the feasibility to use a more rudimentary model of the SAM and still accurately control the more complex manipulator safely.

Several methods exist to determine the dynamic equations for a manipulator that contains closed chain linkages [18], [19] as the SAM does. These methods rely on modifications of the Newton-Euler method. The resulting equations give a very accurate description of the manipulator's operation.

The SAM's ideal use makes designing a controller to achieve uniform performance over the entire workspace a daunting task. The operator generates the SAM's trajectory in real time and can change the dynamic characteristics dramatically as a payload is lifted or released. What is needed is a controller that can accurately identify parameter values as well as the change in payload weight. This will allow a simplified dynamic model to be used when designing the controller, one that does not take into account the changes in inertia due to different link configurations and does not truly represent the link connectivity. Adaptive control provides a tool to reach the performance objectives while still maintaining stability when controlling devices such as the SAM.

Adaptive control has two main objectives: estimation of unknown plant parameters and control of the system based on knowledge of the estimated parameters. Many publications and papers have investigated adaptive control. Several types of adaptive control are not properly suited for this research such as schemes based on assuming slow variations in parameters, plant linearization by ignoring higher order terms, and assuming time invariant plant parameters. Adaptive control for robotic manipulators stems from the theory of model reference adaptive control (MARC) [4], [5], [6]. A reference model, which exhibits the

desired dynamic response, is specified. The reference model output and the actual plant output are compared and the error between the two drives parameter changes in the controller. The error is then eliminated as the model adapts to the plant.

Adaptive controllers for robotic manipulators can be classified several different ways depending on the update law or the control structure. Many adaptive controllers for robots use either computed torque or preservation of the passivity properties of the rigid robot to control the robot. Update laws may be driven by either the error between the desired output and the actual output, tracking error, or the error between the estimated parameters and the true parameters, prediction error.

A popular method of robot control is the computed torque, or inverse dynamics, controller. The defining feature of the computed torque controller is feedback of the nonlinear terms. When the model perfectly matches the plant, the nonlinear terms are cancelled causing the system to become linear and decoupled. In [4] Craig presents an adaptive scheme that exploits the nonlinear cancellation of the computed torque controller, which decouples the unknown parameters. The parameter update is driven by the error between the desired output and the actual output. This derivation is globally convergent, meaning for any given initial condition and any trajectory, asymptotic trajectory tracking is ensured. By exploiting the computed torque controller this method is also forced to maintain a positive definite estimated mass matrix and must measure joint acceleration. Because the estimated mass matrix is inverted, upper and lower limits are placed on the parameters to ensure the matrix remains positive definite. Middleton and Goodwin [7] use a similar approach to Craig with two main differences. First, the update law is driven by the error between the estimated parameters and the true parameters rather than the tracking error. Second, the requirement to measure true joint acceleration is removed by utilizing a first-order filter to estimate the acceleration. Spong and Ortega [8] eliminate the restriction of placing bounds on the estimated parameters by not inverting the estimated mass matrix. Rather, they use a method often found in robust non-adaptive control design and replace the estimated mass matrix with a static best guess matrix. This removes the variable estimated parameters from the mass matrix replacing them with fixed estimates. An additional term is

then added to the control signal to drive adaptation. Spong and Ortega's derivation was augment by Dawson and Lewis [9] to further prove stability.

A second group of globally convergent adaptive controllers look to maintain the passivity properties of the rigid robot. This is achieved by exploiting the Hamiltonian structure of rigid robot dynamics. This method does not require inversion of the estimated mass matrix or joint acceleration feedback. However, because this scheme does not linearize the plant the estimated parameters are not decoupled. Sadegh and Horowitz [10] and Slotine and Li [11] have developed controllers based on maintaining passivity. In both cases the update law is based on tracking error. Both the computed torque and passivity-based methods for adaptive control are detailed in [12].

Research in adaptive control also provides insight into controlling manipulators exhibiting flexible links and joints. A slightly different controller from the above methods is presented by Yang et. al. [13]. Here the authors wish to control a flexible manipulator undergoing a step change in load mass with an adaptive pole placement controller. A least squares identification algorithm is used to place the poles at appropriate locations. A critical restriction on the controller requires prior knowledge of when the payload is being picked up and released.

The robustness of adaptive controllers with respect to uncertainties such as unknown time-varying plant parameters, unmodeled dynamics, and bounded input disturbances is studied in [14]. This work provides stability to the controller by modifying the adaptive law and does not require persistently exciting signals. Fu [15] proposes a decentralized robust adaptive control based on a Lyapunov method by adding a disturbance input to the torque controller. The control method is decentralized meaning each joint has an independently applied "subcontroller". This allows for greater application flexibility and also limits the disturbances caused by variations in parameters or uncertainties in the model dynamics. Asmer et. al. [16] develop an adaptive controller similar to [4] which gives insight into selecting controller gains. This method also provides robustness to signal noise by modifying the regression matrix to depend on the desired trajectory rather that the true trajectory. However, controller requires time invariant unknown parameters, which is not acceptable for application on the SAM. Also, in the case of the SAM, using the desired

trajectory will not eliminate contamination from noise as the desired trajectory is generated through the force transducer signals.

Thesis Overview

Chapter 2 details the derivation of the dynamic equations for both the simple and complex SAM models. Chapter 3 presents a detailed explanation of the adaptive control laws and parameter update scheme and their application to the SAM. Simulation results are presented and discussed in Chapter 4, and Chapter 5 contains the conclusion and suggestions for future investigation.

CHAPTER 2
DYNAMIC MODELING

In this chapter the dynamic models are defined for a simple SAM, modeled as a revolute-prismatic manipulator, and a complex SAM, modeled as a 3-revolute manipulator. The wrist degree of freedom is neglected in both cases. Instead the end-effector and its payload are realized at the distal end of final link. The models will be used to study the performance of an adaptive controller when attempting to control the more complex model with the simple model. These models are the foundation of the controller developed in Chapter 3.

Dynamic Equations for Robotic Manipulators

The dynamic equations for an n-link rigid robotic manipulator can be written as,

$$\tau = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) \qquad 2.1$$

where τ is the nx1 vector of joint torques, $M(\theta)$ is the nxn symmetric positive definite mass matrix, $V(\theta,\dot{\theta})$ is the nx1 vector of torques arising from centrifugal and Coriolis forces, and $G(\theta)$ is the nx1 vector of torques due to gravity.

To determine the values for M, V, and G the iterative Newton-Euler dynamic formulation presented in [21] and first published by Luh, Walker, and Paul in [22] is used. The following equations are valid for both revolute and prismatic links.

Outward iterations: Linear and rotational velocities and accelerations are solved by an iterative process beginning with link 1 and propagating outward link by link to link n.

$$
\begin{aligned}
{}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\omega_i + (1-s_{i+1})\dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\
{}^{i+1}v_{i+1} &= {}^{i+1}_{i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + s_{i+1}\dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \\
{}^{i+1}\dot{\omega}_{i+1} &= {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + (1-s_{i+1})({}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}) \\
{}^{i+1}\dot{v}_{i+1} &= {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i) + s_{i+1}(2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}) \\
{}^{i+1}\dot{v}_{ci+1} &= {}^{i}\dot{\omega}_i \times {}^{i}P_{ci+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{ci+1}) + {}^{i+1}\dot{v}_{i+1}
\end{aligned}
\qquad 2.2
$$

where:

s_i defines the link i as prismatic, s = 1, or revolute, s = 0

${}^{i}\omega_i$ and ${}^{i}\dot{\omega}_i$ are the rotational velocity and acceleration of link i relative to frame i

${}^{i}v_i$ and ${}^{i}\dot{v}_i$ are the linear velocity and acceleration of link i relative to frame i

${}^{i}\dot{v}_{c_i}$ is the linear acceleration for the center of mass of link i relative to frame i

${}^{i}P_{i+1}$ is the position vector describing the location of link i+1 relative to frame i

${}^{i}P_{ci+1}$ is the position vector describing the center of mass of link i+1 relative to frame i

${}^{i+1}_{i}R$ is the rotation matrix describing frame i relative to frame i+1

The inertial force and torque acting at the center of mass of each link are defined as

$$
\begin{aligned}
{}^{i+1}N_{i+1} &= {}^{i+1}I_{ci+1}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i}\omega_i \times {}^{i+1}I_{ci+1}\,{}^{i+1}\omega_{i+1} \\
{}^{i+1}F_{i+1} &= m_{i+1}\,{}^{i+1}\dot{v}_{ci+1}
\end{aligned}
\qquad 2.3
$$

where:

${}^{i}I_{c_i}$ and m_i are the inertia tensor defined at the center of mass and the mass for link i

${}^{i}N_i$ is the moment acting at the center of mass of link i

${}^{i}F_i$ is the force acting at the center of mass of link i

Inward iterations: Inward iterations are used to compute the forces and torques acting on each link.

$$
\begin{aligned}
{}^{i}f_i &= {}_{i+1}^{i}R^{i+1}f_{i+1} + {}^{i}F_i \\
{}^{i}n_i &= {}^{i}N_i + {}_{i+1}^{i}R^{i+1}n_{i+1} + {}^{i}P_{ci} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}_{i+1}^{i}R^{i+1}f_{i+1} \\
\tau_i &= {}^{i}n^{T}{}_{i}{}^{i}\hat{Z}_i
\end{aligned}
\qquad 2.4
$$

where:

${}^{i}f_i$ is the force exerted on link i by link i-1

${}^{i}n_i$ is the torque exerted on link i by link i-1

τ_i is the joint torque provided by an actuator

The solution to these equations results in the necessary joint torques that enable the robot to traverse the trajectory described by joint positions, θ.

The SAM Manipulator

The SAM is a complex and difficult manipulator to model correctly. The existence of several closed chain linkages, which can be seen in Figure 2.1, makes the application of the traditional robotic dynamic equations a nontrivial task. This is part of the motivation behind testing the performance of a simple SAM dynamic model driving a more complex model.

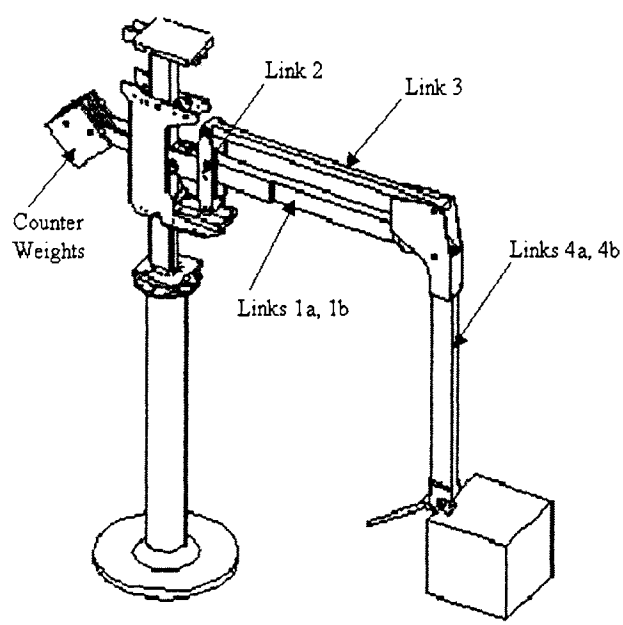

Figure 2.1: Diagram of the SAM

Table 2.1 contains the properties of the links labeled in Figure 2.1

Table 2.1: Dynamic Properties of the SAM

Link Description	Weight (kg)	Mass Moment about CG (kg m^2)	CG from Pivot (m)
Link 1a	68.58	33.2995	0.5513
Link 1b	10.84	3.0213	0.5513
Link 2	9.88	0.2286	NA
Link 3	9.88	2.644	NA
Link 4a	57.32	23.4326	0.5151
Link 4b	10.40	2.9900	0.9342
Counter Weights	202.6	NA	0.6096

Simple SAM Dynamic Model Derivation

The simple model of the SAM is represented as a revolute-prismatic (RP) manipulator. This model is kinematically correct yet does not include the rotational dynamics of links 1a, 1b, 3, 4a, and 4b shown in Figure 2.1. Figure 2.2 diagrams the frame descriptions for the simple SAM model.

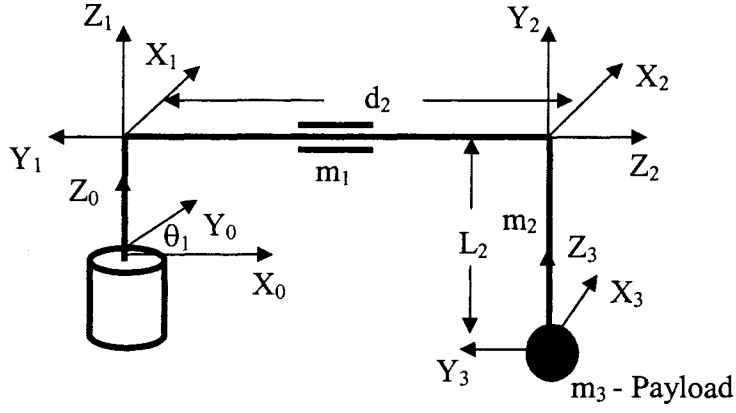

Figure 2.2: Frame Assignment for RP SAM Manipulator

The rotation of the base joint is described by θ_1 while d_2 describes the linear translation of the end effector. The masses of links 1, 2, and the payload are m_1, m_2, and m_3 respectively.

The Denavit-Hartenberg parameters describing the connectivity between links are presented in Table 2.2.

Table 2.2: DH Parameters (Craig)

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1+90°
2	0	90°	d_2	0
3	0	-90°	$-L_2$	0

The definitions of the DH parameters are,

$$a_i = \text{the distance from } \hat{Z}_i \text{ to } \hat{Z}_{i+1} \text{ measured along } \hat{X}_i$$

$$\alpha_i = \text{the angle between } \hat{Z}_i \text{ and } \hat{Z}_{i+1} \text{ measured about } \hat{X}_i$$

$$d_i = \text{the distance from } \hat{X}_{i-1} \text{ to } \hat{X}_i \text{ measured along } \hat{Z}_i$$

$$\theta_i = \text{the angle between } \hat{X}_{i-1} \text{ and } \hat{X}_i \text{ measured about } \hat{Z}_i$$

Using the DH parameters forward kinematics are derived which describe the end effector, frame {3}, in terms of the base, frame {0}.

$$ {}^0_3T = \begin{bmatrix} -s_1 & -c_1 & 0 & d_2c_1 \\ c_1 & s_1 & 0 & d_2s_1 \\ 0 & 0 & 1 & -L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 2.5 $$

The vector, ${}^iP_{ci}$, defines the distance from each link's frame of reference to its center of mass with L_{c1} and L_{c2} defined as the distance from the link joint to the center of gravity for links 1 and 2 respectively.

$$ {}^1P_{c1} = \begin{bmatrix} 0 \\ -L_{c1} \\ 0 \end{bmatrix} \quad {}^2P_{c2} = \begin{bmatrix} 0 \\ -L_{c2} \\ 0 \end{bmatrix} \quad {}^3P_{c3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad 2.6 $$

The inertial tensors are defined as,

$$I_{c1} = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix} \quad I_{c2} = \begin{bmatrix} I_{xx2} & 0 & 0 \\ 0 & I_{yy2} & 0 \\ 0 & 0 & I_{zz2} \end{bmatrix} \tag{2.7}$$

where I_{yy} and I_{zz} are assumed equal and I_{xx} is neglected.

Using the Newton-Euler method, equations 2.2-2.4, the dynamic equations are derived for the RP model of the SAM resulting in the joint torques,

$$\begin{aligned} \tau_1 &= \left(I_{zz1} + I_{yy2} + m_1 L_{c1}^{\ 2} + m_2 d_2^{\ 2} + m_3 d_2^{\ 2}\right)\ddot{\theta}_1 + \left(2m_2 d_2 + 2m_3 d_2\right)\dot{\theta}_1 \dot{d}_2 \\ \tau_2 &= \left(m_2 + m_3\right)\ddot{d}_2 - \left(m_2 d_2 + m_3 d_2\right)\dot{\theta}_1^{\ 2} \end{aligned} \tag{2.8}$$

Dividing equations 2.8 into the resulting inertia matrix and centripetal and gravitational torques vectors gives,

$$M = \begin{bmatrix} I_{zz1} + I_{yy2} + m_1 l_{c1}^{\ 2} + m_2 d_2^{\ 2} + m_3 d_2^{\ 2} & 0.0 \\ 0.0 & m_2 + m_3 \end{bmatrix}$$

$$V = \begin{bmatrix} (2m_2 d_2 + 2m_3 d_2)\dot{\theta}_1 \dot{d}_2 \\ (-m_2 d_2 - m_3 d_2)\dot{\theta}_1^{\ 2} \end{bmatrix} \tag{2.9}$$

$$G = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$$

Complex SAM Dynamic Model Derivation

The complex SAM model will test the ability of the adaptive controller. The complex model of the SAM is represented as a 3-revolute (3R) manipulator. The model is then

constrained to restrict motion to the X-Y plane. This model more accurately captures the rotational dynamics of links 1a, 1b, 3, 4a, and 4b as the end-effector moves towards or away from the base. Figure 2.3 diagrams the frame descriptions for the complex SAM model.

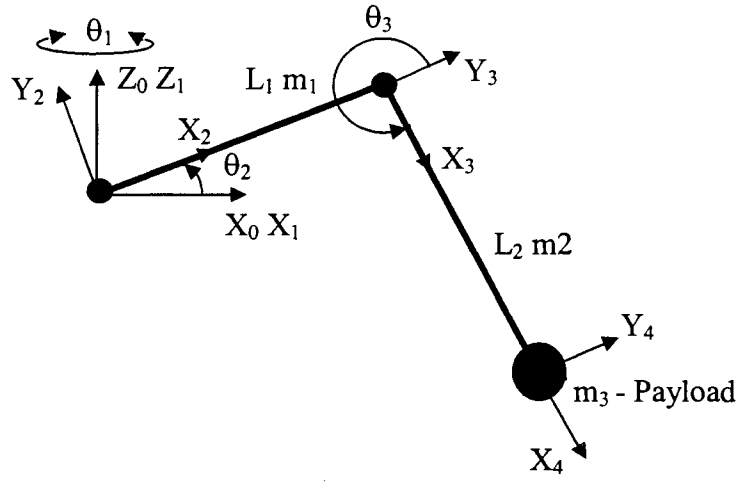

Figure 2.3: Frame Assignment for 3R SAM Manipulator

The rotation of the base joint is described by θ_1 while the combination of θ_2 and θ_3 describes the linear translation of the end-effector. The Denavit-Hartenberg parameters describing the connectivity between links are presented in Table 2.2.

Table 2.2: DH Parameters (Craig)

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	90°	0	θ_2
3	L_1	0	0	θ_3
4	L_2	0	0	0

Using the DH parameters forward kinematics are derived which describe the end-effector, frame {4}, in terms of the base, frame {0}.

$$ {}^{0}_{3}T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & L_2c_1c_{23} + L_1c_1c_2 \\ s_1c_{23} & -s_1s_{23} & -c_1 & L_2s_1c_{23} + L_1s_1c_2 \\ s_{23} & c_{23} & 0 & L_2s_{23} + L_1s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 2.10 $$

As before, ${}^{i}P_{ci}$ and I_{ci} are defined as follows,

$$ {}^{2}P_{c2} = \begin{bmatrix} 0 \\ -L_{c1} \\ 0 \end{bmatrix} \quad {}^{3}P_{c3} = \begin{bmatrix} 0 \\ -L_{c2} \\ 0 \end{bmatrix} \quad {}^{4}P_{c4} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad 2.11 $$

$$ {}^{2}I_{c2} = \begin{bmatrix} I_{xx1} & 0 & 0 \\ 0 & I_{yy1} & 0 \\ 0 & 0 & I_{zz1} \end{bmatrix} \quad {}^{3}I_{c3} = \begin{bmatrix} I_{xx2} & 0 & 0 \\ 0 & I_{yy2} & 0 \\ 0 & 0 & I_{zz2} \end{bmatrix} \qquad 2.12 $$

Once again, by applying the Newton-Euler method the resulting joint torques are determined. The exact results are in Appendix A: Complex SAM Dynamic Model: Initial Results.

$$ \begin{aligned} \tau_1 &= f(\theta_2,\theta_3)\ddot{\theta}_1 + f(\theta_2,\theta_3)\dot{\theta}_1\dot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1\dot{\theta}_2 \\ \tau_2 &= f(\theta_3)\ddot{\theta}_2 + f(\theta_3)\ddot{\theta}_3 + f(\theta_3)\dot{\theta}_2\dot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_3)\dot{\theta}_3^{\,2} + f(\theta_2,\theta_3)g \\ \tau_3 &= f(\theta_3)\ddot{\theta}_2 + C\ddot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_3)\dot{\theta}_2^{\,2} + f(\theta_2,\theta_3)g \end{aligned} \qquad 2.13 $$

Rewriting equation 2.13 in standard robotic form results in,

$$ \tau = \begin{bmatrix} f_{11}(\theta_2,\theta_3) & 0 & 0 \\ 0 & f_{22}(\theta_3) & f_{23}(\theta_3) \\ 0 & f_{32}(\theta_3) & C_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} f(\theta_2,\theta_3)\dot{\theta}_1\dot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1\dot{\theta}_2 \\ f(\theta_3)\dot{\theta}_2\dot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_3)\dot{\theta}_3^{\,2} \\ f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_3)\dot{\theta}_2^{\,2} \end{bmatrix} + \begin{bmatrix} 0 \\ f(\theta_2,\theta_3)g \\ f(\theta_2,\theta_3)g \end{bmatrix} \qquad 2.14 $$

From equation 2.14 it is noted that the torque for joints 2 and 3 are decoupled from joint one.

Constraining the complex SAM dynamic model

The model must be constrained to be kinematically equivalent to the SAM. The controller does not measure the SAM's vertical degree of freedom so this value will be treated as a constant. Joints 2 and 3 are controlled with a linear actuator to provide horizontal motion of the end effector. The joint angles must then be converted to linear variables in the X and Z direction with Z fixed. Figure 2.3 shows the constrained model.

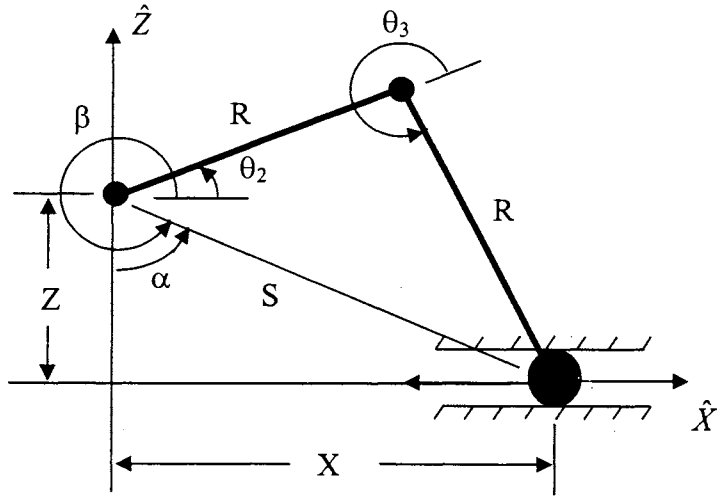

Figure 2.3: Constrained SAM Model

The distance Z in the above figure is fixed at 1.372m. The input to the system is X. The values of α, β, and S are functions of X and Z.

$$\alpha = \tan^{-1}\left(\frac{x}{z}\right)$$
$$\beta = 270^{\circ} + \alpha \qquad 2.15$$
$$S = \sqrt{x^2 + z^2}$$

Equation 2.14 shows that the joint 1 torque is independent from joints 2 and 3. This decoupling of the actuation forces of joint 1 with respect to joints 2 and 3 allows for easy

model constraining. Using equation 2.16, the torques for joints 2 and 3 can be converted into forces in the X and Z directions.

$$F_x = \left(J^T\right)^{-1} T \tag{2.16}$$

To accomplish the conversion, joint velocities are mapped into the Cartesian space using the Jacobian of the manipulator, which is defined as,

$$V_x = J\dot{\theta} \tag{2.17}$$

The linear velocities in the X and Z directions are,

$$\begin{aligned} \dot{x} &= -Rs_2\dot{\theta}_2 - Rs_{23}\left(\dot{\theta}_2 + \dot{\theta}_3\right) \\ \dot{y} &= Rc_2\dot{\theta}_2 + Rc_{23}\left(\dot{\theta}_2 + \dot{\theta}_3\right) \end{aligned} \tag{2.18}$$

Equation 2.18 rewritten in matrix form gives,

$$\begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -rs_2 - rs_{23} & -rs_{23} \\ rc_2 + rc_{23} & rc_{23} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{2.19}$$

From 2.19 the Jacobian as written in frame {0} is,

$$J = \begin{bmatrix} -rs_2 - rs_{23} & -rs_{23} \\ rc_2 + rc_{23} & rc_{23} \end{bmatrix} \tag{2.20}$$

Now the joint torques can be transformed into linear forces in Cartesian space. Expanding equation 2.16,

$$\begin{bmatrix} F_x \\ F_z \end{bmatrix} = \left(J^T\right)^{-1}\left[\begin{bmatrix} 0 & m_{22} \\ 0 & m_{32} \end{bmatrix}\begin{bmatrix} \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} + \begin{bmatrix} g_2 \\ g_3 \end{bmatrix}\right] \tag{2.21}$$

The completed transformation from joint torques to linear forces is,

$$\begin{aligned} F_x &= f(\theta_2,\theta_3)\ddot{\theta}_2 + f(\theta_2,\theta_3)\ddot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_2,\theta_3)\dot{\theta}_2^{\,2} + f(\theta_2,\theta_3)g \\ F_z &= f(\theta_2,\theta_3)\ddot{\theta}_2 + f(\theta_2,\theta_3)\ddot{\theta}_3 + f(\theta_2,\theta_3)\dot{\theta}_1^{\,2} + f(\theta_2,\theta_3)\dot{\theta}_2^{\,2} + f(\theta_2,\theta_3)\dot{\theta}_3^{\,2} + f(\theta_2,\theta_3)g \end{aligned} \tag{2.22}$$

where $f(\theta_2,\theta_3)$ is a function that relies on the manipulator parameters mass, inertia, link lengths, and locations of centers of mass. Because the assumption that the height Z remains fixed, F_z is not needed in the control law and will be removed from the remaining derivations.

Position Constraints. Solve for position constraints using vector loop equations applied to Figure 2.3.

$$\mathrm{Re}^{i\theta_2} + \mathrm{Re}^{i\theta_2+\theta_3} = Se^{i\beta} \tag{2.23}$$

Writing equation 2.23 in its real and imaginary components gives,

$$Rc_2 + Rc_{23} = Sc_\beta \tag{2.24}$$

$$Rs_2 + Rs_{23} = Ss_\beta \tag{2.25}$$

Square and add equations 2.24 and 2.25 to solve for θ_2 and θ_3.

$$\begin{aligned} \theta_2 &= \beta \pm \cos^{-1}\left(\frac{S^2}{2SR}\right) \\ \theta_3 &= a\tan 2\left(\frac{Ss_\beta - Rs_2}{Sc_\beta - Rc_2}\right) - \theta_2 \end{aligned} \tag{2.26}$$

Velocity constraints. To solve for the velocity constraints a new vector loop is written that uses Z and X from Figure 2.3.

$$\mathrm{Re}^{i\theta_2} + \mathrm{Re}^{i\theta_2+\theta_3} = X - Zi \tag{2.27}$$

The derivative of 2.27results in,

$$Ri\dot{\theta}_2 e^{i\theta_2} + Ri\left(\dot{\theta}_2 + \dot{\theta}_3\right)e^{i\theta_2+\theta_3} = \dot{x} \tag{2.28}$$

Separating 2.28 into real and imaginary parts leads to,

$$-Rs_2\dot{\theta}_2 - Rs_{23}\left(\dot{\theta}_2 + \dot{\theta}_3\right) = \dot{x} \tag{2.29}$$

$$Rc_2\dot{\theta}_2 + Rc_{23}\left(\dot{\theta}_2 + \dot{\theta}_3\right) = 0 \tag{2.30}$$

Solving for $\dot{\theta}_2$ and $\dot{\theta}_3$ results in the velocity constraints,

$$\begin{aligned}\dot{\theta}_2 &= -\frac{c_{23}}{R\left(s_2c_{23} - s_{23}c_2\right)}\dot{x} \\ \dot{\theta}_3 &= \frac{\left(c_2 + c_{23}\right)}{R\left(s_2c_{23} - s_{23}c_2\right)}\dot{x}\end{aligned} \tag{2.31}$$

Acceleration constraints. Acceleration constraints are found by taking the derivative of 2.28

$$Ri\ddot{\theta}_2 e^{i\theta_2} - R\dot{\theta}_2{}^2 e^{i\theta_2} + Ri\left(\ddot{\theta}_2 + \ddot{\theta}_3\right)e^{i\theta_2+\theta_3} - R\left(\dot{\theta}_2{}^2 + 2\dot{\theta}_2\dot{\theta}_3 + \dot{\theta}_3{}^2\right)e^{i\theta_2+\theta_3} = \ddot{x} \tag{2.32}$$

Separating 2.32 into real and imaginary parts leads to,

$$-(Rs_2 + Rs_{23})\ddot{\theta}_2 - Rs_{23}\ddot{\theta}_3 = \ddot{x} + (Rc_2 + Rc_{23})\dot{\theta}_2^{\,2} + Rc_{23}\dot{\theta}_3^{\,2} + 2Rc_{23}\dot{\theta}_2\dot{\theta}_3 \quad 2.33$$

$$(Rc_2 + Rc_{23})\ddot{\theta}_2 + Rc_{23}\ddot{\theta}_3 = (Rs_2 + Rs_{23})\dot{\theta}_2^{\,2} + Rs_{23}\dot{\theta}_3^{\,2} + 2Rs_{23}\dot{\theta}_2\dot{\theta}_3 \quad 2.34$$

Solving for $\ddot{\theta}_2$ and $\ddot{\theta}_3$ results in the acceleration constraints,

$$\ddot{\theta}_2 = \frac{\ddot{x}c_{23} + (Rc_3 + R)\dot{\theta}_2^{\,2} + R\dot{\theta}_3^{\,2} + 2R\dot{\theta}_2\dot{\theta}_3}{Rs_3}$$
$$\ddot{\theta}_3 = \frac{-\ddot{x}(c_2 + c_{23}) - (2Rc_3 + 2R)\dot{\theta}_2^{\,2} - (Rc_3 + R)\dot{\theta}_3^{\,2} - (2Rc_3 + 3R)\dot{\theta}_2\dot{\theta}_3}{Rs_3} \quad 2.35$$

Final System

Replacing θ_2 *and* θ_3 and their derivatives with equations 2.26, 2.31, and 2.35 respectively results in the final system describing the complex SAM's dynamics.

$$\tau_1 = f(x,z)\ddot{\theta}_1 + f(x,z)\dot{\theta}_1\dot{x}$$
$$\tau_2 = f(x,z)\ddot{x} + f(x,z)\dot{\theta}_1^{\,2} + f(x,z)\dot{x}^2 + f(x,z)g \quad 2.36$$

The dependence on Z can be removed as it is fixed. The resulting dynamic equations as written in standard robotic form,

$$\tau = \begin{bmatrix} f(x) & 0.0 \\ 0.0 & f(x) \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} f(x)\dot{\theta}_1\dot{x} \\ f(x)\dot{\theta}_1^{\,2} + f(x)\dot{x}^2 \end{bmatrix} + \begin{bmatrix} 0.0 \\ f(x)g \end{bmatrix} \quad 2.37$$

The exact resulting inertia, centripetal, and gravitational terms can be found in APPENDIX A: Complex SAM Dynamic Model: Final Results.

Observations on the Dynamic Models

It is interesting to note the difference between the two SAM models. The simple model does not contain a gravity term due to the use of a prismatic joint where the complex model does as a result of using two revolute joints. The inclusion of the revolute joints also adds an additional term to $V(\theta,\dot{\theta})$ of the second joint. Both base joints should have similar dynamics where as the linear actuated degree of freedom will not match as closely.

Complex model dynamics:

$$\tau = \begin{bmatrix} f(x) & 0.0 \\ 0.0 & f(x) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} f(x)\dot{\theta}_1\dot{x} \\ f(x)\dot{\theta}_1^{\,2} + f(x)\dot{x}^2 \end{bmatrix} + \begin{bmatrix} 0.0 \\ f(x)g \end{bmatrix}$$

Simple model dynamics:

$$\tau = \begin{bmatrix} I_{zz1} + I_{yy2} + m_1 l_{c1}^{\,2} + m_2 d_2^{\,2} + m_3 d_2^{\,2} & 0.0 \\ 0.0 & m_2 + m_3 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} (2m_2 d_2 + 2m_3 d_2)\dot{\theta}_1\dot{d}_2 \\ (-m_2 d_2 - m_3 d_2)\dot{\theta}_1^{\,2} \end{bmatrix}$$

Looking at the resulting values for the mass matrix and Coriolis and centrifugal vector gives insight into how well the models match. One must remember that the exact results in Figure 2.4 are result of a particular trajectory. Figure 2.4 shows the V term for the base joint match for both models as expected. It appears that the mass terms are off by a scaling factor. However, the V term for the elbow joint clearly do not match. This is would appear to be the result of the missing $\dot{x}^2$ term in the simple model's base joint dynamics.

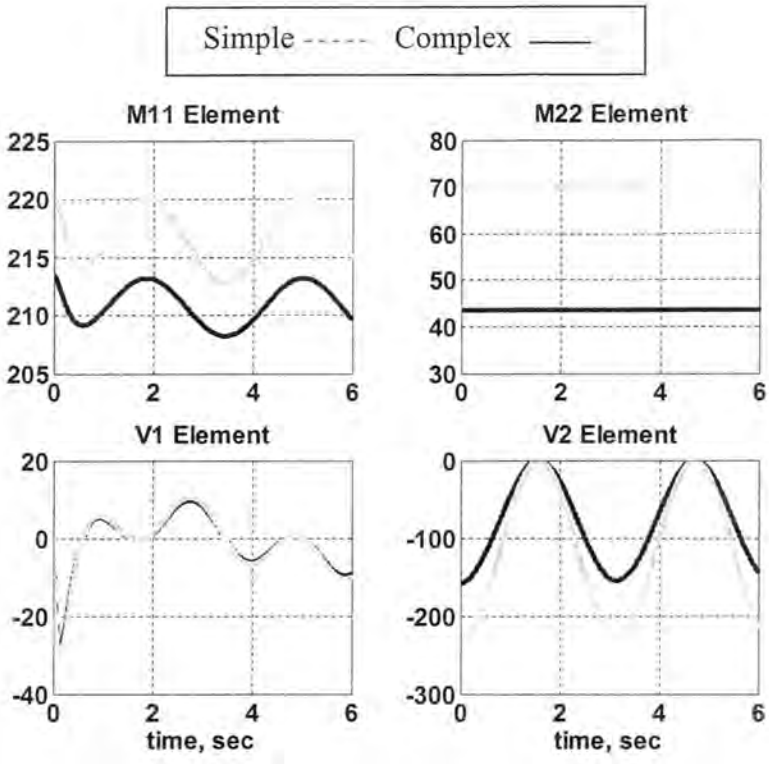

Figure 2.4 Simple vs. Complex Mass Matrix and Coriolis Elements

CHAPTER 3
ADAPTIVE CONTROL OF MANIPULATORS

The adaptive control of manipulators relies on the cancellation of terms between the true plant and the estimated plant by adjusting the unknown parameters in the estimated plant until they converge to the true values. The plant output is compared to the desired states and the error between the two drives the change in the adaptive gains. Figure 3.1 shows a computed torque, or inverse dynamics, based controller with an adaptive element. This scheme is the basis for the adaptive controller developed in this chapter.

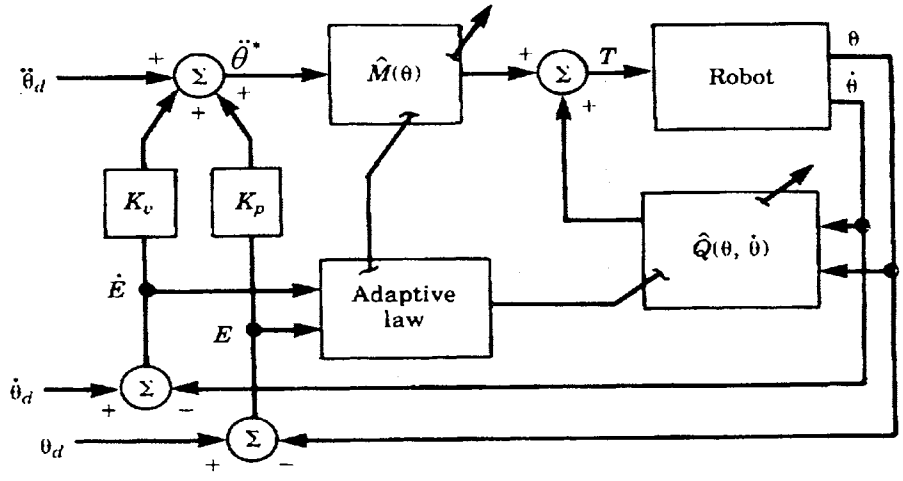

Figure 3.1: Adaptive Control Scheme ([4] p.52)

Derivation of the Control Algorithm

At the core of the adaptive controller exists the computed torque controller. This popular method of manipulator control takes the following form

$$T = \hat{M}(\theta)\ddot{\theta}^* + \hat{V}(\theta,\dot{\theta}) + \hat{G}(\theta) \qquad 3.1$$

where $\hat{M}(\theta), \hat{V}(\theta,\dot{\theta})$, *and* $\hat{G}(\theta)$ are the estimates of $M(\theta), V(\theta,\dot{\theta})$, *and* $G(\theta)$ presented in equation 2.1 in Chapter 2. The acceleration term provides the input to the system depending on the error between the desired trajectory and the actual trajectory.

$$\ddot{\theta}^{*} = \ddot{\theta}_d + K_v\dot{E} + K_pE \tag{3.2}$$

$$\begin{aligned} \dot{E} &= \dot{\theta}_d - \dot{\theta} \\ E &= \theta_d - \theta \end{aligned} \tag{3.3}$$

The vectors $\ddot{\theta}_d(t), \dot{\theta}_d(t), and\ \theta_d(t)$ are the desired joint accelerations, velocities, and positions of the manipulator as a function of time and are assumed known. The gains K_v and K_p are n x n constant diagonal-gain matrices with k_{vi} and k_{pi} on the diagonals. These gains can be selected to place the closed loop poles associated with each joint and provide uniform disturbance rejection over the entire workspace.

Combining equations 3.1 and 3.2 results in the error equation:

$$\begin{aligned} T &= M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) = \hat{M}(\theta)\ddot{\theta}^{*} + \hat{V}(\theta,\dot{\theta}) + \hat{G}(\theta) \\ \hat{M}(\theta)\ddot{\theta}^{*} - M(\theta)\ddot{\theta} &= V(\theta,\dot{\theta}) + G(\theta) - \left(\hat{V}(\theta,\dot{\theta}) + \hat{G}(\theta)\right) \end{aligned} \tag{3.4}$$

Subtract $\hat{M}(\theta)\ddot{\theta}$ from both sides to obtain

$$\hat{M}(\theta)\ddot{\theta}^{*} - \hat{M}(\theta)\ddot{\theta} = \left(M(\theta) - \hat{M}(\theta)\right)\ddot{\theta} + \left(V(\theta,\dot{\theta}) - \hat{V}(\theta,\dot{\theta})\right) + \left(G(\theta) - \hat{G}(\theta)\right) \tag{3.5}$$

Combining equation 3.2 with 3.5 and gathering terms results in the error equation,

$$\ddot{E} + K_v\dot{E} + K_pE = \hat{M}^{-1}(\theta)\left[\tilde{M}(\theta)\ddot{\theta} + \tilde{V}(\theta,\dot{\theta}) + \tilde{G}(\theta)\right] \tag{3.6}$$

where $\tilde{M} = M - \hat{M}, \tilde{V} = V - \hat{V}, and\ \tilde{G} = G - \hat{G}.$

When the estimated plant parameters equal the true plant parameters equation 3.6 reduces to the ideal case,

$$\ddot{E} + K_v\dot{E} + K_pE = 0 \tag{3.7}$$

Rewrite the error equation 3.6 in the form

$$\ddot{E} + K_v \dot{E} + K_p E = \hat{M}^{-1}(\theta) W(\theta, \dot{\theta}, \ddot{\theta}) \Phi \tag{3.8}$$

Here $W(\theta, \dot{\theta}, \ddot{\theta})$ is an n x r matrix of functions and Φ is an r x 1 vector containing the parameter errors. The vector Φ is the difference between the values of the true and estimated parameters.

$$\Phi = P - \hat{P} \tag{3.9}$$

W is composed of manipulator trajectory function and will remain bounded as long as the trajectory is bounded. M must be positive definite and invertible. A method for maintaining a positive definite and invertible mass matrix will be presented later.

Derivation of the Adaptation Algorithm

Following the method presented in [4] the adaptive algorithm is now derived. The method of adaptation is a function of joint parameter error.

$$E_1 = \dot{E} + \Psi E \tag{3.10}$$

Where $\Psi = diag(\Psi_1 \quad \Psi_2 \quad ... \quad \Psi_j)$ and the values of ψ are chosen so the transfer function,

$$\frac{s + \psi_j}{s^2 + k_{vj} s + k_{pj}} \tag{3.11}$$

is strictly real positive (all poles and zeros on left-half plane).

The state space realization for the system can now be written as

$$\begin{aligned} \dot{X} &= AX + B\hat{M}^{-1}W\Phi \\ E_1 &= CX \end{aligned} \tag{3.12}$$

Where $X = \begin{bmatrix} E & \dot{E} \end{bmatrix}^T$.

From the above definitions,

$$\begin{aligned} A^T + PA &= Q \\ PB &= C^T \end{aligned} \tag{3.13}$$

where both P and Q are positive definite.

The adaptation law is derived using Lyapunov theory. Lyapunov global stability requires a function V that satisfies two conditions:

1. All sublevels sets of V are bounded for all z
2. $\dot{V}(z) \leq 0$ for all z

Beginning with the Lyapunov function candidate,

$$V(X,\Phi) = X^T PX + \Phi^T \Gamma^{-1} \Phi \tag{3.14}$$

Differentiation of equation 3.13 with respect to time leads to,

$$\dot{V}(X,\Phi) = -X^T QX + 2\Phi^T \left(W^T \hat{M}^{-1} E_1 + \Gamma^{-1} \dot{\Phi} \right) \tag{3.15}$$

Now choose

$$\dot{\Phi} = -\Gamma W^T \hat{M}^{-1} E_1 \tag{3.16}$$

Replacing $\dot{\Phi}$ in equation 3.13 with equation 3.15, we now have,

$$\dot{V}(X,\Phi) = -X^T Q X \tag{3.17}$$

Equation 3.16 is non-positive because Q is positive definite. This derivation assures stability and error convergence. Since $\Phi = P - \hat{P}$ the derivative is $\dot{\Phi} = -\dot{\hat{P}}$. We now have the adaptation law

$$\dot{\hat{P}} = \Gamma W^T \hat{M}^{-1} E_1 \tag{3.18}$$

Putting the new system in state space results in

$$\begin{bmatrix} \dot{X} \\ \dot{\Phi} \end{bmatrix} = \begin{bmatrix} A & BU^T \\ -\Gamma UC & 0 \end{bmatrix} \begin{bmatrix} X \\ \Phi \end{bmatrix} \tag{3.19}$$

Where

$$\dot{X} = \begin{bmatrix} \dot{E} \\ \ddot{E} \end{bmatrix} \quad X = \begin{bmatrix} E \\ \dot{E} \end{bmatrix} \quad \Phi = P - \hat{P} \quad \dot{\Phi} = -\dot{\hat{P}} \quad U = \left(\hat{M}^{-1} W\right)^T$$

$$A = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \qquad C = \begin{bmatrix} \Psi & I \end{bmatrix} \tag{3.20}$$

Modification to eliminate inverting time varying mass matrix

Following the derivation outlined in [8] the dependence on maintaining a positive definite $\hat{M}$ is removed. This is accomplished by replacing the estimated mass matrix with a static approximated mass matrix. Begin by choosing a new control law,

$$T = M_o(\theta)(a + \delta a) + V_o(\theta, \dot{\theta}) + G_o(\theta) \tag{3.21}$$

where $M_o(\theta)$, $V_o(\theta,\dot{\theta})$, *and* $G_o(\theta)$ are based on static best guess parameter values and

$$a = \ddot{\theta}_d + K_v\dot{E} + K_pE \tag{3.22}$$

Equating 2.1 and 3.20 results in an equation similar to 3.7,

$$M_o(\theta)(a + \delta a - \ddot{\theta}) = \Delta M(\theta)\ddot{\theta} + \Delta V(\theta,\dot{\theta}) + \Delta G(\theta) \equiv Y(\theta,\dot{\theta},\ddot{\theta})\Delta P \tag{3.23}$$

defining $\Delta(\bullet) = (\bullet) - (\bullet)_o$.

$$\ddot{E} + K_v\dot{E} + K_pE = M_o^{-1}(\theta)Y(\theta,\dot{\theta},\ddot{\theta})\Delta P - \delta a \tag{3.24}$$

By choosing,

$$\delta a = M_o^{-1}(\theta)Y(\theta,\dot{\theta},\ddot{\theta})\Delta\hat{P} \tag{3.25}$$

where $\Delta\hat{P} = \hat{P} - \hat{P}_o$ *and* $\hat{P}_o = P_o$, a result similar to equation 3.8 is reached with equation 3.9 replacing $\Delta\hat{P}$.

$$\ddot{E} + K_v\dot{E} + K_pE = M_o^{-1}(\theta)Y(\theta,\dot{\theta},\ddot{\theta})\Phi \tag{3.26}$$

The resulting system uses only fixed parameters when calculating M, V, and G thus avoiding placing bounds on the estimated parameters. The new control signal is

$$\ddot{\theta}^* = K_pE + K_v\dot{E} + \ddot{\theta}_d + M_o^{-1}(\theta)Y(\theta,\dot{\theta},\ddot{\theta})\Delta\hat{P} \tag{3.27}$$

Design of Adaptive Controller for the Simple SAM

Referring to the dynamic solution derived for the simple SAM in Chapter 2 equation 2.9 is rewritten for reference,

$$M = \begin{bmatrix} I_{zz1} + I_{yy2} + m_1 l_{c1}^{\ 2} + m_2 d_2^{\ 2} + m_3 d_2^{\ 2} & 0.0 \\ 0.0 & m_2 + m_3 \end{bmatrix}$$

$$V = \begin{bmatrix} (2m_2 d_2 + 2m_3 d_2)\dot{\theta}_1 \dot{d}_2 \\ (-m_2 d_2 - m_3 d_2)\dot{\theta}_1^{\ 2} \end{bmatrix}$$

$$G = \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$$

The only known parameter is d_2, which defines the linear translation of the end-effector. The unknown parameters are the link masses, inertias, and positions of the centroids. Placing these unknowns into the estimated parameter vector $\hat{P}$.

$$\hat{P} = \begin{bmatrix} \hat{I}_{zz1} + \hat{I}_{yy2} \\ \hat{m}_1 \hat{l}_{c1}^{\ 2} \\ \hat{m}_2 \\ \hat{m}_3 \end{bmatrix} \qquad 3.28$$

With knowledge of the M, V, and G terms and the unknown parameter list the resulting Y matrix is,

$$Y = \begin{bmatrix} \ddot{\theta}_1 & \ddot{\theta}_1 & d_2^{\ 2}\ddot{\theta}_1 + 2d_2\dot{\theta}_1\dot{d}_2 & d_2^{\ 2}\ddot{\theta}_1 + 2d_2\dot{\theta}_1\dot{d}_2 \\ 0 & 0 & \ddot{d}_2 - d_2\dot{\theta}_1^{\ 2} & \ddot{d}_2 - d_2\dot{\theta}_1^{\ 2} \end{bmatrix} \qquad 3.29$$

Resulting system equations used in implementing the controller.

CHAPTER 4
SIMULATION RESULTS

With the dynamic equations developed in Chapter 2 and the adaptive controller designed in Chapter 3 the control of the SAM is simulated. The simple model of the SAM is used as the plant in the controller and the complex model realizes the "true" SAM dynamics.

Trajectory Generation

In the operation of the SAM the user defines the end-effector trajectory by applying a force at the handle. The end-effector displacement is proportional to the force; the harder the user pushes the faster the manipulator moves. This proportionality is defined in equation 4.1.

$$\Delta X = K_{force} F \tag{4.1}$$

The Jacobian then maps the Cartesian displacement into the joint space.

$$\Delta X = J \Delta \theta \tag{4.2}$$

Combining equations 4.1 and 4.2 defines the relationship between input force and joint displacement.

$$K_{force} F = J \Delta \theta \tag{4.3}$$

Expanding equation 4.3 results in,

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta d \end{bmatrix} = {}^0J^{-1} \begin{bmatrix} k_f & 0 \\ 0 & k_f \end{bmatrix} {}^0_3R \begin{bmatrix} f_x \\ f_y \end{bmatrix}^3 \tag{4.4}$$

where the Jacobian in frame $\{0\}$ is determined by,

$$\dot{x} = \dot{d}\cos(\theta_1) - d\sin(\theta_1)\dot{\theta}_1$$
$$\dot{y} = \dot{d}\sin(\theta_1) + d\cos(\theta_1)\dot{\theta}_1 \qquad 4.5$$

$${}^0J = \begin{bmatrix} -ds_1 & c_1 \\ dc_1 & s_1 \end{bmatrix} \qquad 4.6$$

Using equation 4.4 the joint displacement is determined by the input forces. The displacement is then used to generate the desired position. The derivative of the position results in the desired velocity and the derivative of the velocity results it the desired acceleration.

Simulation Parameters and Methodology

The initial estimated parameters used on the simple model and the true parameters used on the complex model for the simulations are presented in Table 4.1.

Table 4.1: Initial True and Estimated Values for SAM Properties

	m_1	m_2	m_3	L_{c1}	L_{c2}	I_{yy1}	I_{zz1}	I_{yy2}	I_{zz2}
True	89.3	67.7	10.0	0.60	0.75	45.0	45.0	35.0	35.0
Estimated	120.0	25.0	10.0	1.20	1.20	45.0	45.0	35.0	35.0

The values of the gains K_v and K_p were selected to provide settling time of 1 second, a damping ratio of 0.707 and a natural frequency of 4Hz when perfect model matching is achieved.

Pseudo code used to simulate the adaptive controller in MATLAB.

1. Set true and estimated plant values.
2. Set values for gains (K_v K_p ψ Γ).
3. Set manipulator's initial position, velocity and acceleration to the initial desired position, velocity, and acceleration.
4. Begin control loop

a. Calculate desired trajectory, velocity, and acceleration.

b. Calculate control input $\ddot{\theta}^* = K_p E + K_v \dot{E} + \ddot{\theta}_d + M_o^{-1}(\theta) Y(\theta, \dot{\theta}, \ddot{\theta}) \Delta \hat{P}$.

c. Calculate joint torque $T = M_o(\theta)\ddot{\theta}^* + V_o(\theta, \dot{\theta}) + G_o(\theta)$.

d. Calculate true joint positions, velocities, and accelerations from true model values using $\ddot{\theta} = M^{-1}(T - V(\theta, \dot{\theta}) - G(\theta))$ and integration.

e. Calculate Y based on true trajectory, velocity, and acceleration and known plant parameters.

f. Calculate M_o based on estimated parameters and true trajectory, velocity, and acceleration.

g. Calculate $\dot{\hat{P}} = \Gamma(M_o^{-1} Y)^T E_1$ where $E_1 = \dot{E} + \Psi E$.

h. Calculate $\hat{P} = \dot{\hat{P}} \Delta T + \hat{P}$.

i. Update the estimated parameters.

5. End control loop

Simulation Results

The simulation results when driving the complex SAM using a controller designed for the simple SAM are presented for three cases. The first case runs the manipulator with an initial set of estimated parameters. The simulation is allowed to run until the parameters converge resulting in the determination of a final set of more accurate estimated parameters for the remaining case runs. The final two case studies compare the performance of the adaptive controller to that of a non-adaptive computed torque controller. The second case is for an unloaded manipulator where all true parameters are held constant. The final case simulates a step increase in payload weight at two seconds.

The Cartesian end-effector trajectory and the desired joint positions, velocities, and accelerations are shown in Figure 4.1 and Figure 4.2 respectively.

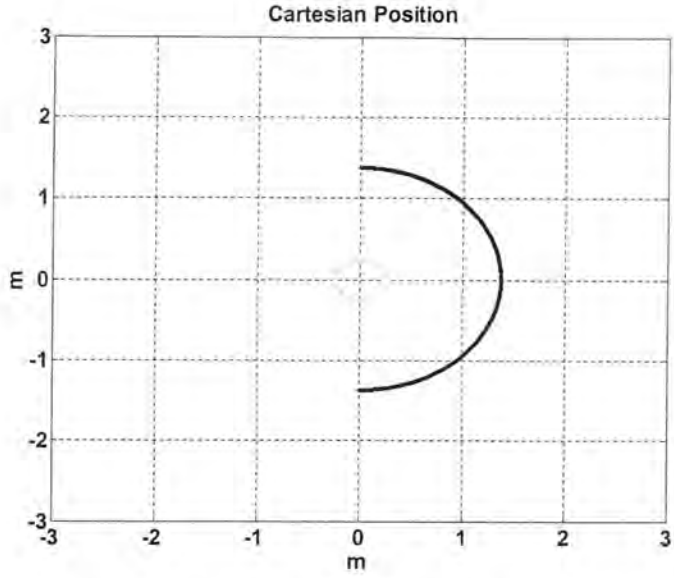

Figure 4.1: Cartesian Trajectory of End Effector

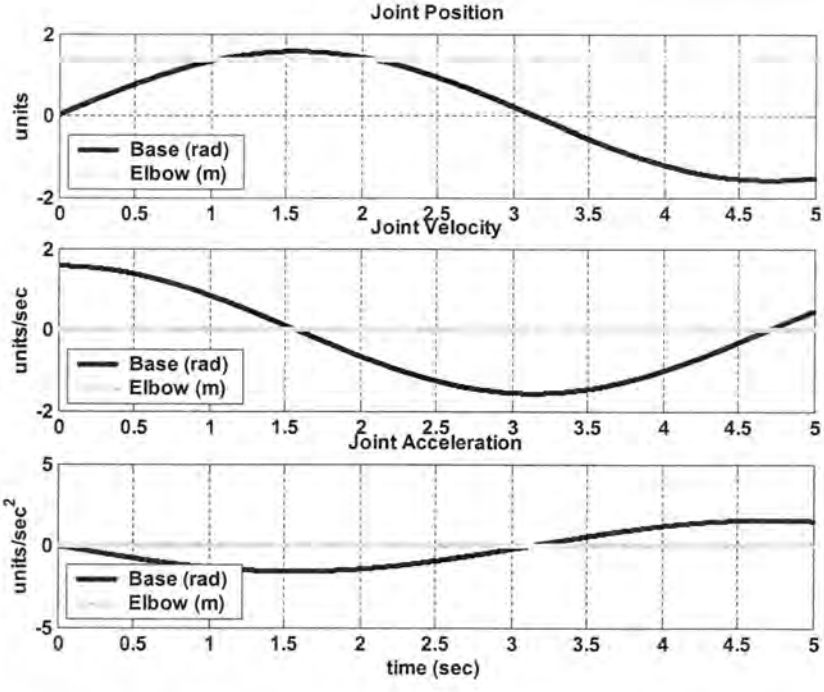

Figure 4.2: Joint 1 and 2 Positions, Velocities, and Accelerations

Case 1: Estimating parameters

Figure 4.3 shows the estimated simple model parameters converging to better represent the complex model parameters.

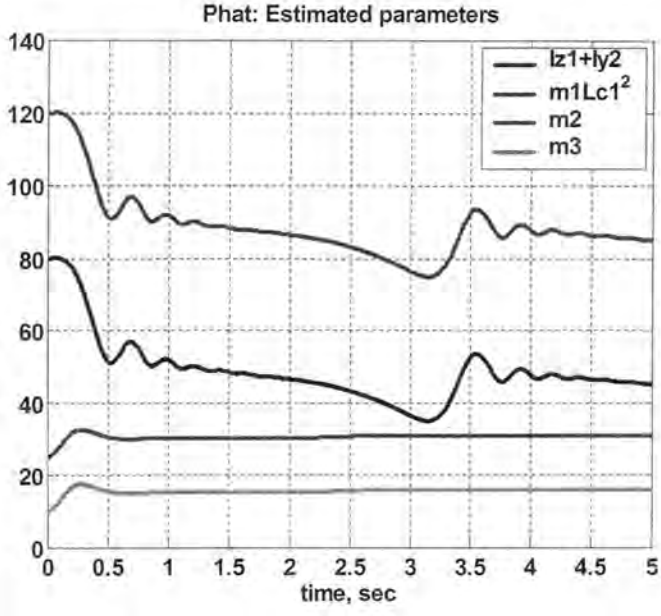

Figure 4.3: Estimated Parameters for Adaptive Case

The final values help identify a new set of estimated parameters which will be used in the final two simulations.

Table 4.2: Final True and Estimated Values for SAM Properties

	m_1	m_2	m_3	L_{c1}	L_{c2}	I_{yy1}	I_{zz1}	I_{yy2}	I_{zz2}
True	89.3	67.7	10.0	0.60	0.75	45.0	45.0	35.0	35.0
Estimated	90.0	40.0	10.0	0.60	0.75	45.0	45.0	35.0	35.0

Figure 4.4 results show how the estimated plant dynamics converge to approximate the true dynamics further verifying the accuracy of the estimated parameters.

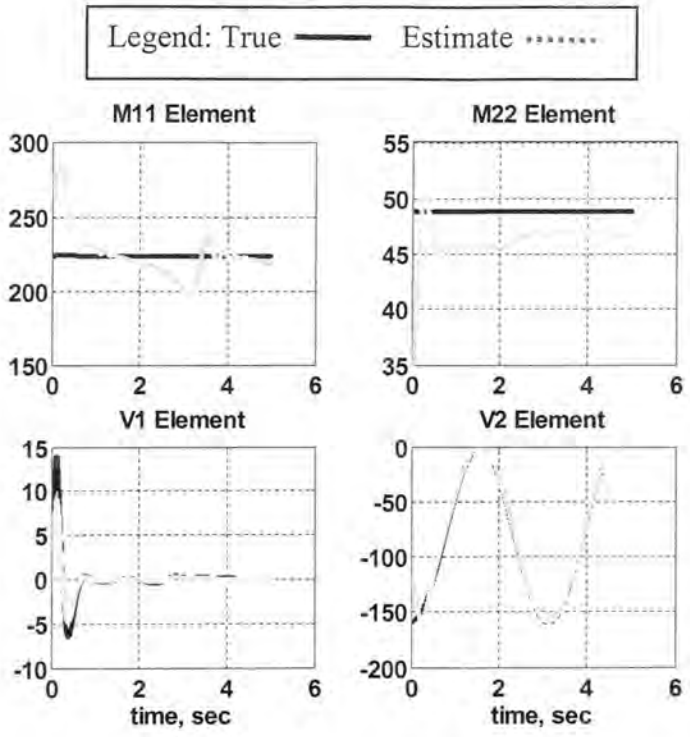

Figure 4.4: Dynamic Terms for Adaptive Case

Case 2: Constant parameters

Figure 4.5 shows the joint error for the computed torque controller. When compared to the error of the adaptive controller in Figure 4.6 it can be seen that the error of the computed torque controller is a factor of ten greater. However, both joint errors are extremely small and would not lead to a noticeable difference in performance.

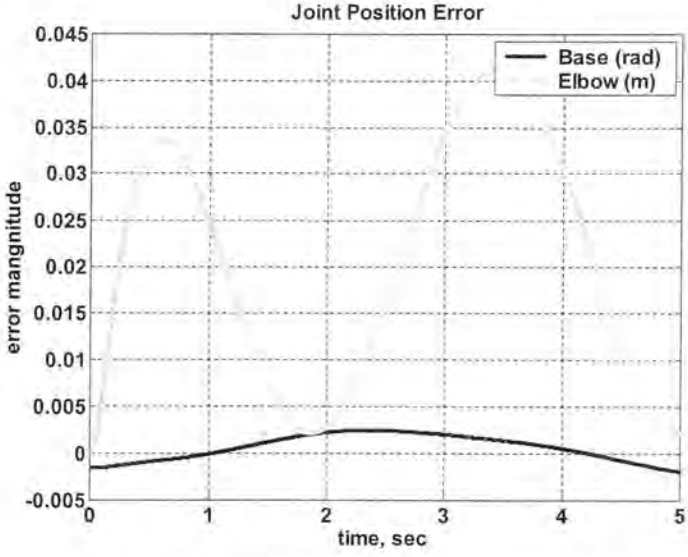

Figure 4.5: Joint Error for Non-adaptive Case

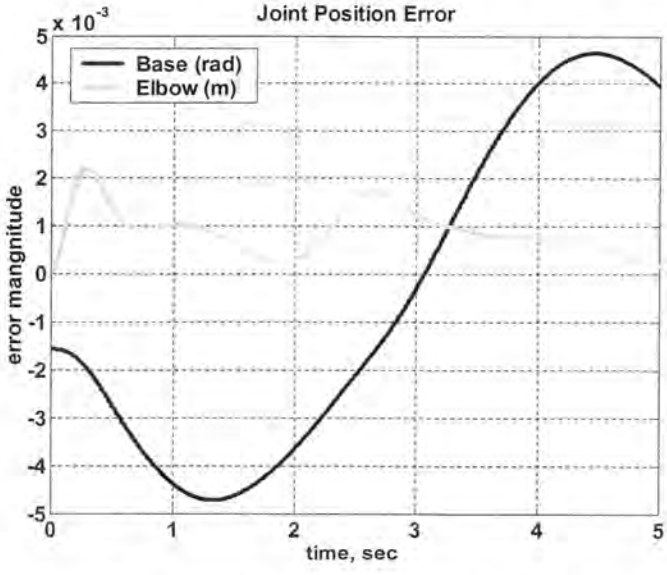

Figure 4.6: Joint Error for Adaptive Case

The result of the larger error can be attributed to the adjusting of parameters in the model. Figure 4.7 shows only slight changes in the parameters indicating that the estimated parameters closely match the actual plant.

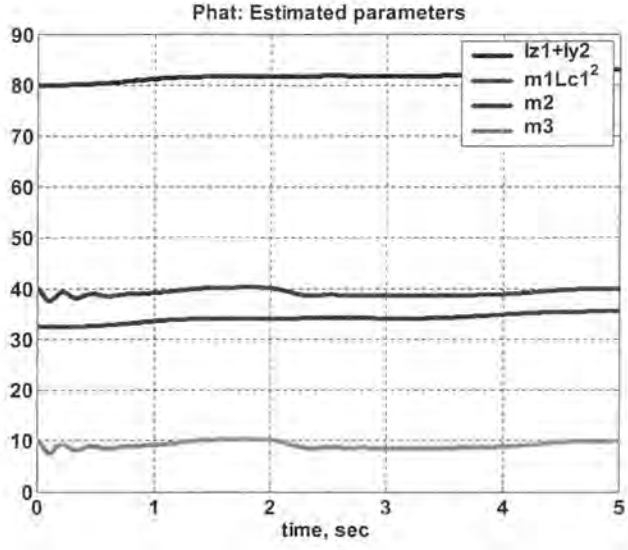

Figure 4.7: Estimated Parameters for Adaptive Case

The dynamic terms in Figures 4.8 and 4.9 show that the adaptive control is adjusting to match the actual plant.

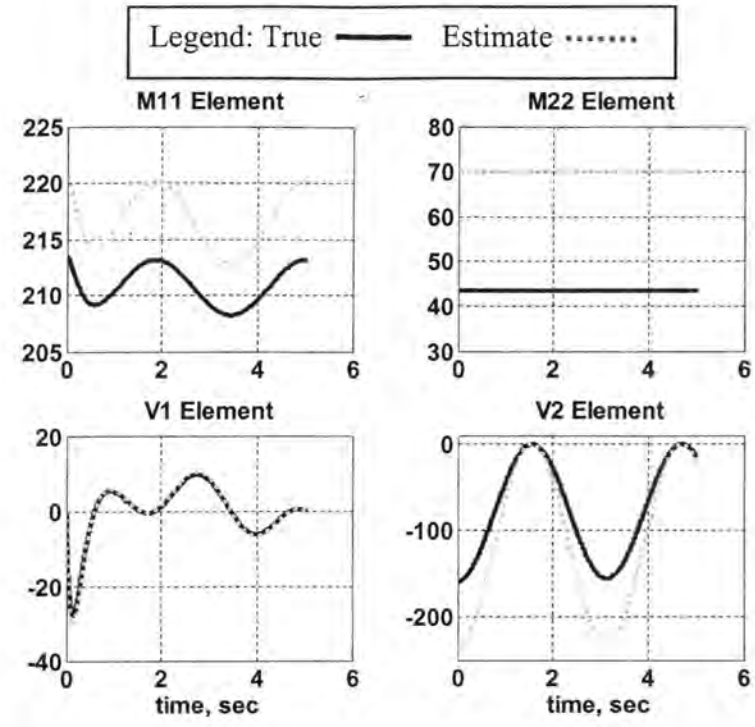

Figure 4.8: Dynamic Terms for Non-adaptive Case

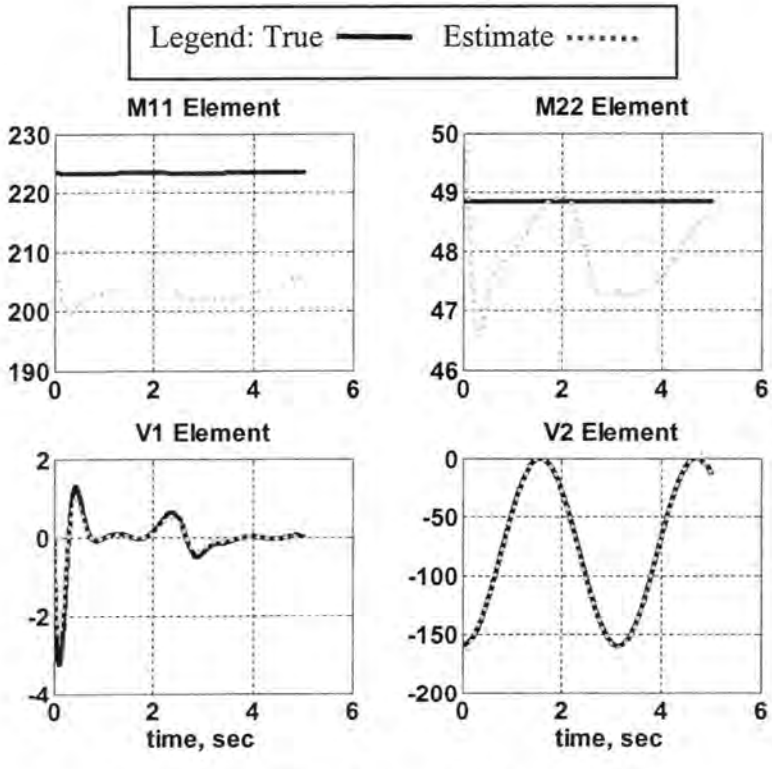

Figure 4.9: Dynamic Terms for Adaptive Case

Case 3: Step change in payload mass

Case 3 consists of simulating a step change in payload mass at two seconds. The mass increases from 10kg to 225kg to simulate the lifting of the maximum rated load for the SAM. The resulting trajectory of the computed torque controlled manipulator has large errors as illustrated in Figure 4.10. These errors are as high as one-half a meter.

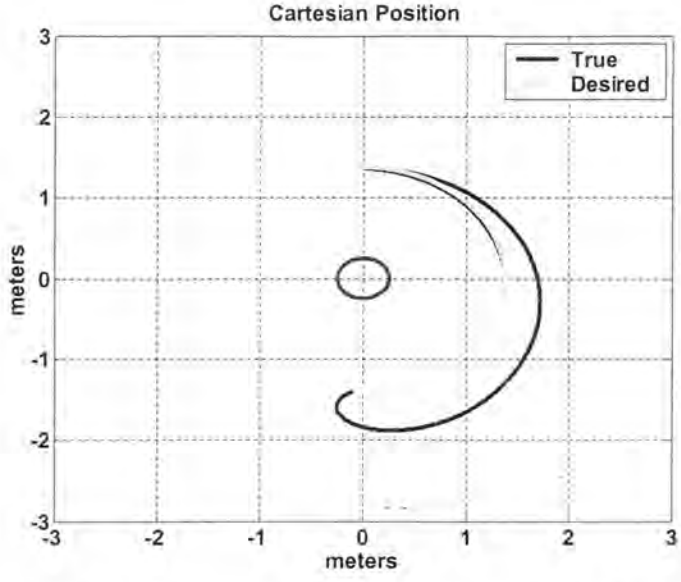

Figure 4.10: Trajectory for Non-adaptive Case Step Change

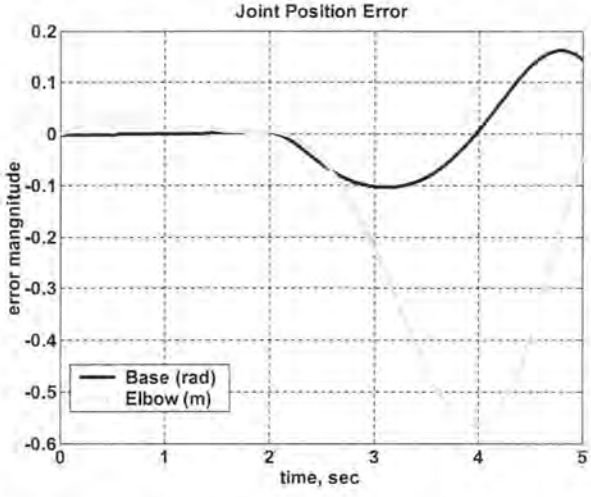

Figure 4.11: Joint Error for Non-adaptive Case Step Change

The large joint error Figure 4.11 displays is understandable after studying Figure 4.12. At the moment the SAM lifts the payload the dynamic terms for the actual manipulator change dramatically while the model does not.

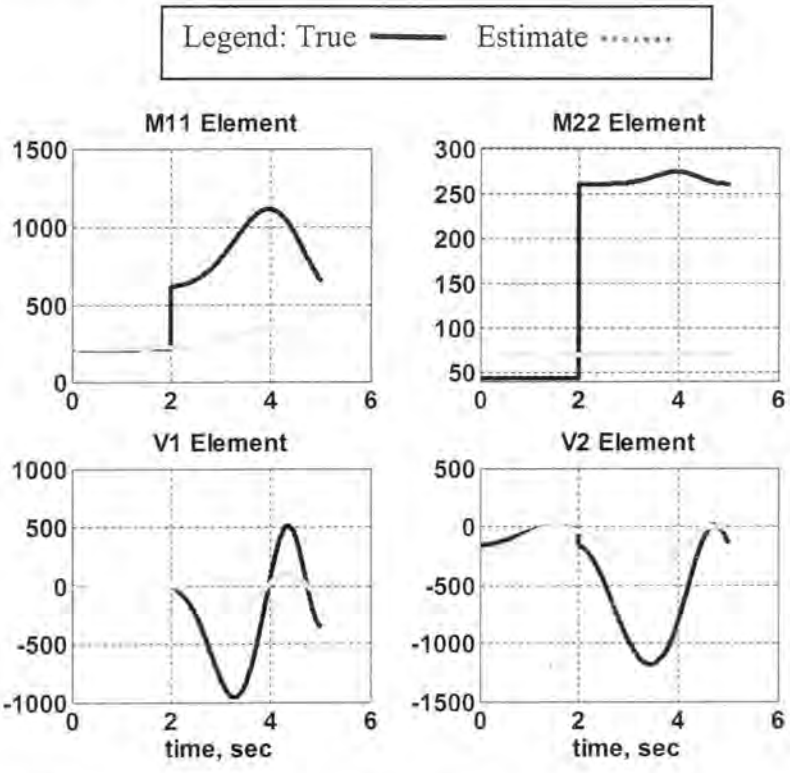

Figure 4.12: Dynamic Terms for Non-adaptive Case Step Change

The adaptive case shows superior performance to the computed torque controller when the parameters do not match. The adaptive control driven manipulator is able to accurately follow the desired trajectory. The improved performance is easily seen in Figures 4.13 and 4.14. The joint error for the adaptive controller is roughly 100 times better that the computed torque controller.

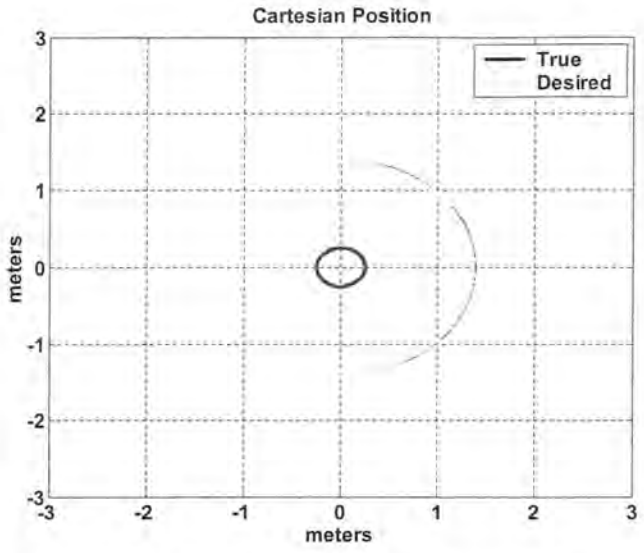

Figure 4.13: Trajectory for Adaptive Case Step Change

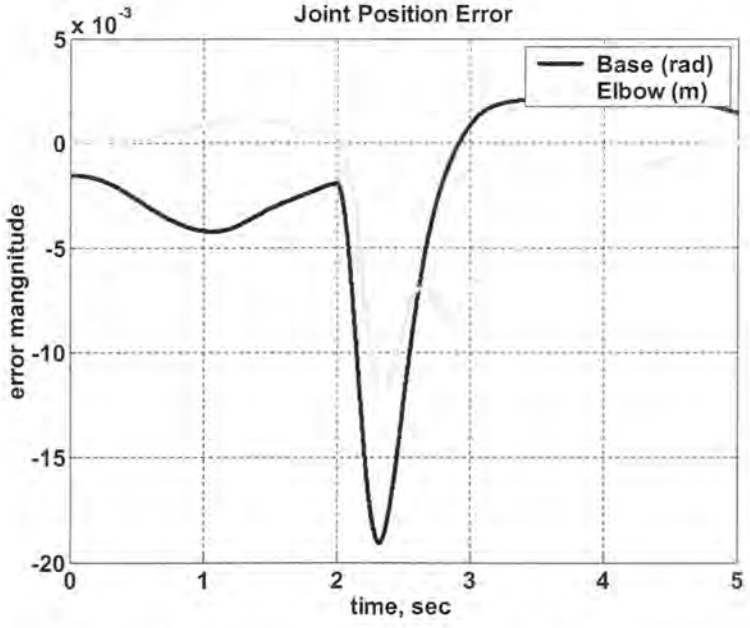

Figure 4.14: Joint Error for Adaptive Case Step Change

When viewing Figure 4.15 the improvement in performance is easily understood. Each element of the modeled dynamic terms converges to the actual terms. This shows that despite the mismatched models.

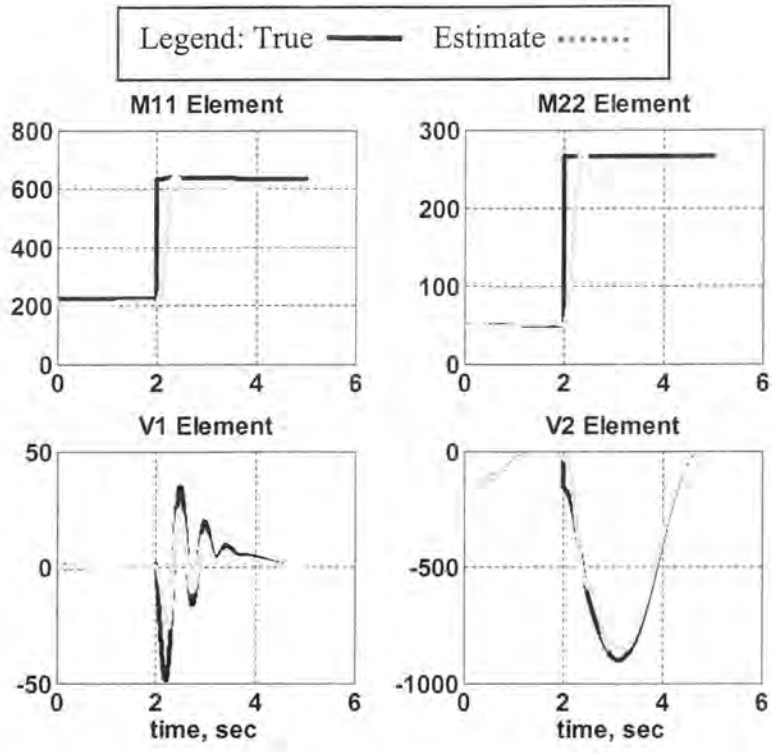

Figure 4.15: Dynamic Terms for Adaptive Case Step Change

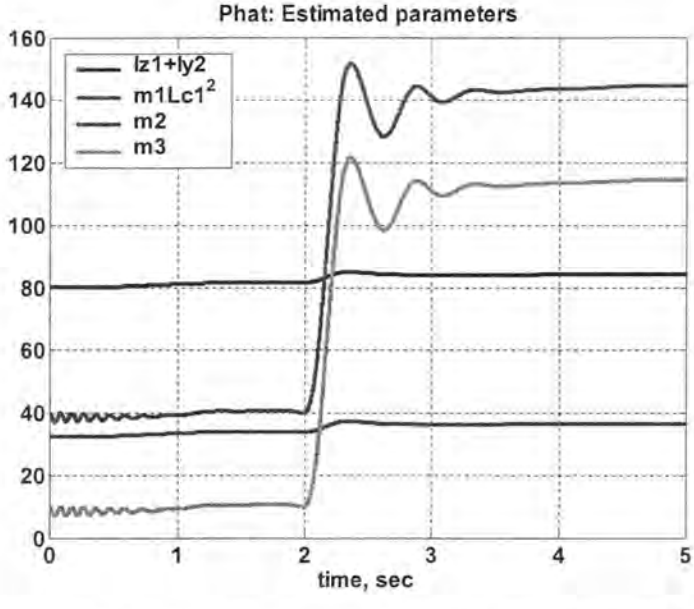

Figure 4.16: Parameter Convergence for Adaptive Case Step Change

The combined change in m_2 and m_3 is roughly 215, as read from Figure 4.16, which matches the step change in payload mass of 215. However, both m_2 and m_3 adapt together allowing

only the change in payload weight and not the actual value of m_3 to be estimated. From this it can be concluded that the adaptive controller is able to rapidly and accurately track and identify changes in parameters.

CHAPTER 5
CONCLUSIONS AND FUTURE INVESTIGATION

The design and implementation of adaptive controllers for robotic manipulators provides a wide variety of solutions when more conventional controllers fail to reach the desired objectives. This thesis studied the application of an adaptive controller based on mismatched model dynamics. Two different dynamic models for the SAM were developed. The simple model, representing the controller model, was based on a revolute-prismatic manipulator model and the true dynamic model was represented as a three-revolute manipulator. The simulation results show that the adaptive controller is able to account for the missing dynamic terms and provide superior performance when compared to a non-adaptive computed torque controller. Especially impressive was the adaptive controller's ability to track and identify a step change in a parameter. From these conclusions the utilization of more simple models to represent complex manipulators in the design process of the controller is possible thus shortening and simplifying the process.

Further investigation regarding the application of the adaptive controller on the SAM manipulator should include robustness to noise in the input signal. The force transducer signal that is used to generate the desired trajectory is subject to noise. In the existing SAM controller the readings from the force transducers need to be filtered to avoid servomotor chatter. One would need to explore if this filtering is adequate for the adaptive controller as well. The presence of noise has been shown to cause parameter drift in adaptive controllers. Methods have been developed to make the controller more robust to signal noise ensuring convergent parameters. Finally the controller should be implemented on the actual SAM manipulator. Then the performance could be evaluated again to determine if the adaptive controller performs as the simulations predict.

APPENDIX

Complex SAM Dynamic Model: Initial Results

Joint 1:

M_{11} = (1/2*m3*L1^2+1/2*m2*L1^2+1/2*Lc2^2*m2+1/2*Lc1^2*m1+1/2*Iyy2+1/2*Iyy1+ L2*m3*L1*cos(th3)+L2*m3*L1*cos(th3+2*th2)+Lc2*m2*L1*cos(th3)+Lc2*m2*L 1*cos(th3+2*th2)+1/2*Iyy1*cos(2*th2)+1/2*Iyy2*cos(2*th2+2*th3)+1/2*L2^2*m3+ 1/2*Lc2^2*m2*cos(2*th2+2*th3)+1/2*L2^2*m3*cos(2*th2+2*th3)+1/2*Lc1^2*m1* cos(2*th2)+1/2*m3*L1^2*cos(2*th2)+1/2*m2*L1^2*cos(2*th2))***thdd1**

$M_{12} = 0$
$M_{13} = 0$

V_1 = (-L2*m3*L1*sin(th3+2*th2)-Lc2^2*m2*sin(2*th2+2*th3)-L1*m3*L2*sin(th3)- Lc2*m2*L1*sin(th3+2*th2)-Iyy2*sin(2*th2+2*th3)-L1*m2*Lc2*sin(th3)- L2^2*m3*sin(2*th2+2*th3)+Ixx2*sin(2*th2+2*th3))***thd1*thd3** + (-m3*L1^2*sin(2*th2)-2*L2*m3*L1*sin(th3+2*th2)+Ixx2*sin(2*th2+2*th3)- Iyy2*sin(2*th2+2*th3)+Ixx1*sin(2*th2)-m2*L1^2*sin(2*th2)- 2*Lc2*m2*L1*sin(th3+2*th2)-L2^2*m3*sin(2*th2+2*th3)-Lc1^2*m1*sin(2*th2)- Lc2^2*m2*sin(2*th2+2*th3)-Iyy1*sin(2*th2))***thd1*thd2**

$G_1 = 0$

Joint 2:

$M_{21} = 0$

M_{22} = (Izz1+2*L1*m3*L2*cos(th3)+Izz2+m3*L1^2+m2*L1^2+Lc1^2*m1+L2^2*m3+Lc2 ^2*m2+2*L1*m2*Lc2*cos(th3))***thdd2**

M_{23} = (Izz2+L1*m3*L2*cos(th3)+L1*m2*Lc2*cos(th3)+Lc2^2*m2+L2^2*m3)***thdd3**

V_2 = (1/2*m3*L1^2*sin(2*th2)+1/2*Lc2^2*m2*sin(2*th2+2*th3)+1/2*Lc1^2*m1*sin(2*th2)+ Lc2*m2*L1*sin(th3+2*th2)+1/2*L2^2*m3*sin(2*th2+2*th3)+1/2*Iyy1*sin(2*th2)+1/2* m2*L1^2*sin(2*th2)-1/2*Ixx2*sin(2*th2+2*th3)+1/2*Iyy2*sin(2*th2+2*th3) - 1/2*Ixx1*sin(2*th2)+L2*m3*L1*sin(th3+2*th2))***thd1^2** +(-L1*sin(th3)*m2*Lc2- L1*sin(th3)*m3*L2)***thd3^2** +(-2*L1*sin(th3)*m2*Lc2- 2*L1*sin(th3)*m3*L2)***thd2*thd3**

G_2 = (Lc2*m2*cos(th2+th3)+L2*m3*cos(th2+th3)+Lc1*m1*cos(th2)+L1*m2*cos(th2)+L1 *m3*cos(th2))***g**

Joint 3:

$M_{21} = 0$

M_{22} = (Izz2+Lc2*(m2*Lc2+m2*cos(th3)*L1)+L2*(m3*L2+m3*cos(th3)*L1))***thdd2**

M_{23} = (Izz2+Lc2^2*m2+L2^2*m3)***thdd3**

V_2 = (Lc2*(1/2*m2*Lc2*sin(2*th2+2*th3)+1/2*m2*L1*sin(th3+2*th2)+1/2*m2*sin(th3)*L1)+
L2*(1/2*m3*L2*sin(2*th2+2*th3)+1/2*m3*L1*sin(th3+2*th2)+1/2*m3*sin(th3)*L1)+1/2
*Iyy2*sin(2*th2+2*th3)-1/2*Ixx2*sin(2*th2+2*th3))***thd1^2**
+(L1*sin(th3)*m2*Lc2+L1*sin(th3)*m3*L2)***thd2^2**

G_2 = (Lc2*m2*cos(th2+th3)+L2*m3*cos(th2+th3))***g**

Complex SAM Dynamic Model: Final Results

Make the following substitutions to replace θ_2 *and* θ_3.

$$\alpha = \tan^{-1}\left(\frac{x}{z}\right) \qquad \theta_2 = \beta \pm \cos^{-1}\left(\frac{s^2}{2sr}\right)$$

$$\beta = 270^\circ + \alpha \qquad \theta_{3'} = a\tan 2\left(\frac{s\sin(\beta) - r\sin(\theta_2)}{s\cos(\beta) - r\cos(\theta_2)}\right)$$

$$S = \sqrt{x^2 + z^2} \qquad \theta_3 = \theta_{3'} - \theta_2$$

Joint 1:

M_{11} = (1/2*Iyy3+1/2*m2*L1^2+1/2*Iyy1+1/2*Ixx3+1/2*Iyy3*cos(2*th2+2*th3)+1/2*Iyy2*cos(2*th2+2*th3)+1/2*m2*L1^2*cos(2*th2) +1/2*Lc2^2*m2*cos(2*th2+2*th3) +1/2*m3*L1^2*cos(2*th2)+1/2*Lc2^2*m2+1/2*m3*L1^2+1/2*Iyy2-1/2*Ixx3*cos(2*th2+2*th3)+L1*m3*L2*cos(th3)+L1*m3*L2*cos(th3+2*th2) +L1*m2*Lc2*cos(th3+2*th2)+L1*m2*Lc2*cos(th3)+1/2*Iyy1*cos(2*th2)+ 1/2*L2^2*m3*cos(2*th2+2*th3)+1/2*Lc1^2*m1*cos(2*th2)+1/2*Lc1^2*m1 +1/2*L2^2*m3)

$M_{12} = 0$

V_1 = ((-L1*m2*Lc2*sin(th3)-Lc2*m2*L1*sin(th3+2*th2)+Ixx3*sin(2*th2+2*th3)-Iyy2*sin(2*th2+2*th3)-Lc2^2*m2*sin(2*th2+2*th3)-Iyy3*sin(2*th2+2*th3)-L1*m3*L2*sin(th3)-L2^2*m3*sin(2*th2+2*th3)-L2*m3*L1*sin(th3+2*th2))*(cos(th2)+cos(th2+th3))/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))-(-L2^2*m3*sin(2*th2+2*th3)-Lc2^2*m2*sin(2*th2+2*th3)-m2*L1^2*sin(2*th2)-2*Lc2*m2*L1*sin(th3+2*th2)-Iyy1*sin(2*th2)-Iyy2*sin(2*th2+2*th3)-Lc1^2*m1*sin(2*th2)-Iyy3*sin(2*th2+2*th3)-m3*L1^2*sin(2*th2)+Ixx3*sin(2*th2+2*th3)-2*L2*m3*L1*sin(th3+2*th2))*cos(th2+th3)/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2)))***thd1*xd**

$G_1 = 0$

Linear Actuator - X direction :

$M_{21} = 0$

M_{22} = m22 = (1/r*(cos(th2)+cos(th2+th3))/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2))*((Izz3+Lc2^2*m2+Izz2+L2^2*m3)*(-cos(th2+th3)-cos(th2))/r/sin(th3) +(Izz2+Izz3+Lc2*(m2*Lc2+m2*cos(th3)*L1)+L2*(m3*L2 +m3*cos(th3)*L1))* cos(th2+th3)/r/sin(th3))-1/r*cos(th2+th3)/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2)) *((Izz1+Lc2^2*m2+m2*L1^2+Lc1^2*m1+2*L1*m3*L2*cos(th3)+2*L1*m2*Lc2* cos(th3)+L2^2*m3+m3*L1^2+Izz2+Izz3)*cos(th2+th3)/r/sin(th3)+(Izz3+L1*m2*Lc2*cos(th3)+Izz2+L1*m3*L2*cos(th3)+Lc2^2*m2+L2^2*m3)*(-cos(th2+th3)-cos(th2))/r/sin(th3)

V_2 = (1/r*cos(th2+th3)^2/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2))*(-2*L1*m2*
Lc2*sin(th3)-2*L1*m3*L2*sin(th3))*(cos(th2)+cos(th2+th3))/(r*sin(th2)*cos(th2+th3
)-sin(th2+th3)*cos(th2))^2+1/r*(cos(th2)+cos(th2+th3))/(cos(th2+th3)*sin(th2)-
sin(th2+th3)*cos(th2))*(L1*m3*L2*sin(th3)+L1*m2*Lc2*sin(th3))*cos(th2+th3)^2/(r
*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2+1/r*(cos(th2)+cos(th2+th3))/(cos(th2
+th3)*sin(th2)-sin(th2+th3)*cos(th2))*((Izz3+Lc2^2*m2+Izz2+L2^2*m3)*(2*r*
cos(th2+th3)/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2*(cos(th2)+cos (th2
+th3))-2*r*cos(th2+th3)^2/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2*cos(th3)
-2*r*cos(th2+th3)^2/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2-
r*(cos(th2)+cos(th2+th3))^2/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2-
r*(cos(th2)+cos(th2+th3))^2/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2
*cos(th3)+2*r*cos(th2+th3)/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2*
(cos(th2)+cos(th2+th3))*cos(th3))/r/sin(th3)+(Izz2+Izz3+Lc2*(m2*Lc2+m2*cos(th3)*
L1)+L2*(m3*L2+m3*cos(th3)*L1))*(-2*r*cos(th2+th3)/(r*sin(th2)*cos(th2+th3)-
sin(th2+th3)*cos(th2))^2*(cos(th2)+cos(th2+th3))+r*cos(th2+th3)^2/(r*sin(th2)*cos(t
h2+th3)-sin(th2+th3)*cos(th2))^2*cos(th3)+r*cos(th2+th3)^2/(r*sin(th2)
*cos(th2+th3)-sin(th2+th3)*cos(th2))^2+r*(cos(th2)+cos(th2+th3))^2/(r*sin(th2)
*cos(th2+th3)-sin(th2+th3)*cos(th2))^2)/r/sin(th3))-1/r*cos(th2+th3)/(cos(th2
+th3)*sin(th2)-sin(th2+th3)*cos(th2))*(-L1*m3*L2*sin(th3)-L1*m2*Lc2*sin(th3))
*(cos(th2)+cos(th2+th3))^2/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2-
1/r*cos(th2+th3)/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2))*((Izz1+Lc2^2*m2
+m2*L1^2+Lc1^2*m1+2*L1*m3*L2*cos(th3)+2*L1*m2*Lc2*cos(th3)+L2^2*m3+
m3*L1^2+Izz2+Izz3)*(-2*r*cos(th2+th3)/(r*sin(th2)*cos(th2+th3)-
sin(th2+th3)*cos(th2))^2*(cos(th2)+cos(th2+th3))+r*cos(th2+th3)^2/(r*sin(th2)*cos(t
h2+th3)-sin(th2+th3)*cos(th2))^2*cos(th3)+r*cos(th2+th3)^2/(r*sin(th2)*cos(th2+th3)
-sin(th2+th3)*cos(th2))^2+r*(cos(th2)+cos(th2+th3))^2/(r*sin(th2)*cos(th2+th3)-
sin(th2+th3)*cos(th2))^2)/r/sin(th3) +(Izz3+L1*m2*Lc2*cos(th3)+Izz2+L1*m3
*L2*cos(th3)+Lc2^2*m2+L2^2*m3)*(2*r*cos(th2+th3)/(r*sin(th2)*cos(th2+th3)
-sin(th2+th3)*cos(th2))^2*(cos(th2)+cos(th2+th3))-2*r*cos(th2+th3)^2/(r*sin(th2)*
cos(th2+th3)-sin(th2+th3)*cos(th2))^2*cos(th3)-2*r*cos(th2+th3)^2/
(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2-r*(cos(th2)+cos(th2+th3))^2
/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2-r*(cos(th2)+cos(th2+th3))^2
/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2*cos(th3)+2*r*cos(th2+th3)
/(r*sin(th2)*cos(th2+th3)-sin(th2+th3)*cos(th2))^2*(cos(th2)
+cos(th2+th3))*cos(th3))/r/sin(th3)))***xd^2**
+(-1/r*cos(th2+th3)/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2))*(1/2*Iyy2*
sin(2*th2+2*th3)+1/2*Iyy1*sin(2*th2)+1/2*Iyy3*sin(2*th2+2*th3)-1/2*Ixx3*
sin(2*th2+2*th3)+1/2*m2*L1^2*sin(2*th2) +1/2*Lc2^2*m2* sin(2*th2+2*th3)
+1/2*L2^2*m3*sin(2*th2+2*th3)+1/2*Lc1^2*m1*sin(2*th2)+1/2*m3*L1^2
*sin(2*th2)+L2*m3*L1*sin(th3+2*th2)+Lc2*m2*L1*sin(th3+2*th2))
+1/r*(cos(th2)+cos(th2+th3))/(cos(th2+th3)*sin(th2)-sin(th2+th3)*cos(th2))*
(1/2*Iyy3*sin(2*th2+2*th3)+Lc2*(1/2*m2*Lc2* sin(2*th2+2*th3)+
1/2*m2*L1*sin(th3+2*th2)+1/2*m2*sin(th3)*L1)-1/2*Ixx3*sin(2*th2+2*th3)
+1/2*Iyy2*sin(2*th2+2*th3)+L2*(1/2*m3*L2*sin(2*th2+2*th3)+1/2*m3*L1*sin(th3
+2*th2)+1/2*m3*sin(th3)*L1)))***thd1^2;**

G_2 = ((-cos(th2+th3)-cos(th2))/r/sin(th3)*(L2*m3*cos(th2+th3)+Lc2*m2*cos(th2+th3))
+1/r*cos(th2+th3)/sin(th3)*(L1*m3*cos(th2)+L2*m3*cos(th2+th3)+Lc2*m2*cos(th2+th
3)+L1*m2*cos(th2)+Lc1*m1*cos(th2)))*g

REFERENCES

[1] Asada H., Slotine J., "Robot Analysis and Control," John Wiley & Sons, New York, 1986.

[2] Injuries, Illnesses, and Fatalities, Bureau of Labor Statistics, U.S. Department of Labor. http://www.bls.gov/iif/#tables.

[3] Tan, K.L., "Intelligent Assist Device," Ph.D. dissertation, Iowa State University, 2001.

[4] Craig, John J., "Adaptive Control of Mechanical Manipulators," Addison-Wesley, MA, 1988.

[5] Astrom K.J., Wittenmark B., "Adaptive Control," Addison-Wesley, MA, 1989.

[6] Landau I.D., Lozand R., M'Saad M., "Adaptive Control," Springer, Great Briton, 1998.

[7] Middleton R.H., Goodwin G.C, "Adaptive Computed Torque Control for Rigid Link Manipulators," Systems and Control Letters, Vol. 10, pp. 9-16, 1988.

[8] Spong M.W., Ortega R., "On Adaptive Inverse Dynamics Control of Rigid Robots," IEEE Transactions on Automatic Control, Vol. 35, No. 1, pp. 92-95, January 1990.

[9] Dawson D.M., Lewis F.L., "Comments on "On Adaptive Inverse Dynamics Control of Rigid Robots," IEEE Transactions on Automatic Control, Vol. 36, No.10, October 1991.

[10] Sadegh N., Horowitz R., "Stability Analysis of an Adaptive Controller for Robotic Mantiulators," IEEE International Conference on Robotics and Automation, Raleigh, NC., 1987.

[11] Slotine J.J.E., Li W., "On the Adaptive Control of Robot Manipulators," ASME Winter Annual Meeting, Anaheim, CA, 1986.

[12] Ortega R., Spong M., "Adaptive Motion Control of Rigid Robots: A Tutorial," 1988 IEEE Proceedings of the 27th Conference on Decision and Control, pp1575-1584, December 1988.

[13] Yang T.C, Yang J.C.S., Kudva P., "Load-Adaptive Control of a Single-Link Flexible Manipulator," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 1, pp. 85-90, January/Febuary 1992.

[14] Reed J.S., Ioannou P.A., "Instability Analysis and Robust Adaptive Control of Robotic Manipulators," 1989 IEEE Transactions on Robotics and Automations, Vol. 5, No. 3, pp. 381-386, June 1989.

[15] Fu, Li-Chen, "Robust Adaptive Decentralized Control of Robot Manipulators," 1992 IEEE Transactions on Automatic Control, Vol. 37, N0. 1, pp.106-110, January 1992.

[16] Asmer H., Schwartz H.M., Warshaw G.D., "A Direct Adaptive Control Method With Desired Compensation For Robotic Manipulators," 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 105-113, 7-10 July 1992.

[17] Ghorbel F., Hung J.Y., Spong M.W., "Adaptive Control of Flexible Joint Manipulators," 1989 IEEE, pp 1188-1193, 1989.

[18] Xiang-Rong Xu, Won-Jee Chung, Young-Hyu Choi, Xiang-Feng Ma, "A New Dynamic Formulation for Robot Manipulators Containing Closed Kinematic Chains," Robotica, vol.17 pp.261-267, 1999.

[19] Murry J.J., Lovell G.H., "Dynamic Modeling of Closed-Chain Robotic Manipulators and Implications for trajectory Control," IEEE Transactions on Robotics and Automation, Vol. 5, No. 4, August 1989.

[20] Guo L.S., Zhang Q., "Adaptive Trajectory Control of a Two DOF Closed-Chain Robot," Proceedings of the American Control Conference, June, 2001.

[21] Craig, John J., "Introduction to Robotics: Mechanics and Controls," Addison-Wesley, MA, 1989.

[22] Luh J.Y.S., Walker M.W., Paul R.P., "On-Line Computational Scheme for Mechanical Manipulators," Transactions of the ASME Journal of Dynamics System, Measurement, and Control, 1980.