Adaptive Control of Manipulators Containing Closed Kinematic Loops[1]

Michael W. Walker

Robotics Research Laboratory
Department of Electrical Engineering and Computer Science
The University of Michigan
Ann Arbor, Michigan 48109

February 15, 1988

[1]This work was supported in part by a TRW Faculty Assistantship Grant to the University of Michigan.

Abstract

This paper describes a new algorithm for the adaptive control of a robot manipulator which may contain closed kinematic loops. The algorithm identifies the mass properties of each link and the viscous friction coefficients for each joint of the manipulator. It is similar to the Newton-Euler inverse dynamics algorithm and hence, obtains its computational efficiency through the recursive nature of the algorithm.

Contents

1 Introduction

The development of computationally efficient inverse dynamics algorithms for manipulators has enabled motion control systems to be designed which theoretically produce motions identical to the desired motions [1,2]. These algorithms are based upon a model which consist of a sequence of rigid bodies interconnected by rotational or translational joints. The mass properties of each link are assumed to be known exactly and nonlinearities such as link flexibility and gearing backlash do not exist. The performance of these controllers is limited to the extent that the actual manipulator exhibits these characteristics.

However, even if the manipulator can be modeled perfectly one must still consider the effects of the unknown properties of an object being moved by a manipulator. For this reason and the fact that the model parameters of existing manipulators are not known exactly, several researchers have investigated the application of adaptive control methods. The approach of initial efforts have been to model the manipulator as a linear system and to apply existing adaptive control methods [3,4,5,6]. The major assumption is that good results can be achieved if the model parameters being identified are not changing rapidly. Stability has always been an issue in this approach and no proof has ever been given.

Another approach to adaptive control of a manipulator has been to base the control on the full nonlinear model of the manipulator which is used by the inverse dynamics algorithms. The significant advantage of these approaches has been the ability of proving global stability. Craig, *et al.*, was the first to use this approach [7]. Their approach is to drive the manipulator using an inverse dynamics model whose parameters are only estimates of the actual parameters. If the model matches the actual manipulator, the manipulator motion will exactly track the desired motion. Any deviation will be due to errors in the parameter estimates. The parameter estimates are continuously modified based upon the deviation of the manipulator from the desired trajectory. The method of parameter modification is chosen such that stability can be shown using a Lyapunov function. The only drawbacks of this method is the need for an initial guess of the unknown parameters which is near to the actual parameters, the need to invert a large matrix at each iteration of the algorithm, and the need to measure the joint acceleration. Of these three, the need for matrix inversion is the main problem.

A more recent extension of these results [8] have shown that the need for matrix inversion and computing the joint accelerations can be eliminated by using a filtered form of the dynamic equations of motion in the derivation of the adaptation law. However, the computational complexity of the method is still relatively high.

Another similar approach to adaptive control has been proposed by Slotine and Li [9]. A Lyapunov function is used to prove stability by showing that the output errors converge to a sliding surface, which in turn implies that the tracking errors converge to zero. The advantages over Craigs method is the elimination of the need for a close initial guess of the unknown parameters and the measurement of acceleration. The main problem is in the computational complexity of the algorithm. This is typical of controllers which are based upon the Lagrangian

formulation of the manipulator dynamics.

In both of these adaptive control methods, there is the practical issue of computational complexity. For this reason, neither have been applied to manipulators with more than two or three degrees of freedom. In addition, both have been formulated for serial link manipulators although many manipulators contain closed kinematic loops.

This paper presents an efficient solution to the problem of the adaptive control of a manipulator containing closed kinematic loops and whose mass and friction parameters are unknown. It adopts the basic method of Slotine and Li and addresses the issues of generality and computational efficiency. Generality is obtained by allowing the manipulator to contain closed kinematic loops. Computational efficiency is obtained by basing the control on an efficient inverse dynamics model of the manipulator.

The body of the paper begins in the next section with a presentation of the notation used in the paper. In addition, a new method of defining the kinematics of manipulators containing closed kinematic loops is presented. The method is a simple extension of the D-H notation [10] which is commonly used for serial link manipulators and naturally leads to the use of a binary tree data structure for storing the manipulator model. Finally, the form of the kinematic constraint equations which are characteristic of manipulators containing closed kinematic loops is presented.

The next section presents a new inverse dynamics algorithm for manipulators containing closed kinematic loops. In this algorithm only inertial and gravitational forces are computed. This improves the computational efficiency over existing methods [11] by eliminating the need for the computation of constraint forces.

The next section presents the adaptive controller. The use of the inverse dynamics model of the manipulator in the formulation of the control directly leads to an efficient algorithm for its implementation. Two important concepts which are introduced are spatial feedback vectors and scaler feedback functions. The character of the control is dictated by the definition of these vectors and functions.

The next section presents the details of the implementation of the controller and the final section draws some conclusions and suggest some areas for further investigation.

2 Preliminaries

This section presents some preliminary material which supports the development of the results in the following sections. First, the data structure used in the controller algorithm is described. Next, the kinematic notation is presented followed by a section on the constraint equations imposed by the closed kinematic loops. Finally, the method used to model the mass distribution of the rigid links of the manipulator is presented.

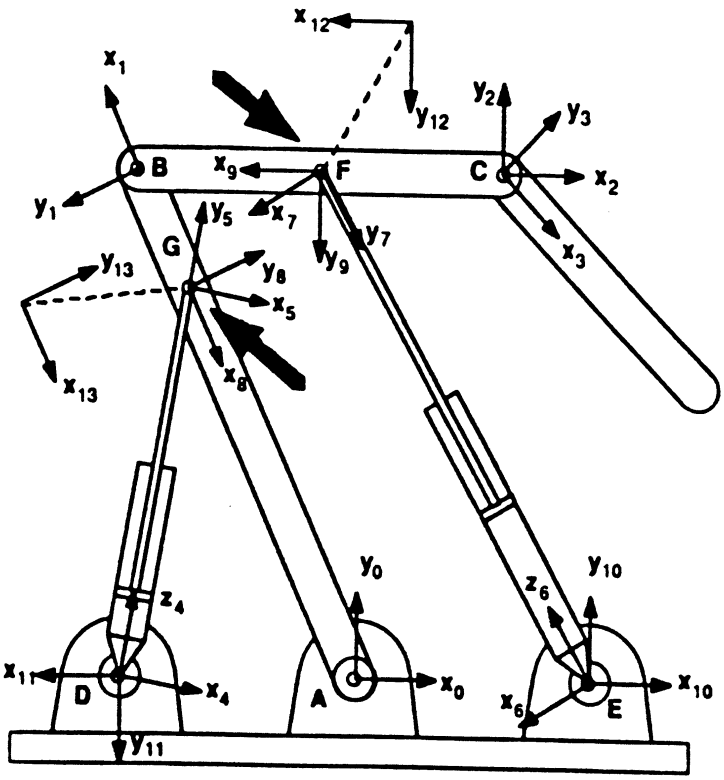

Figure 1: An Example Manipulator

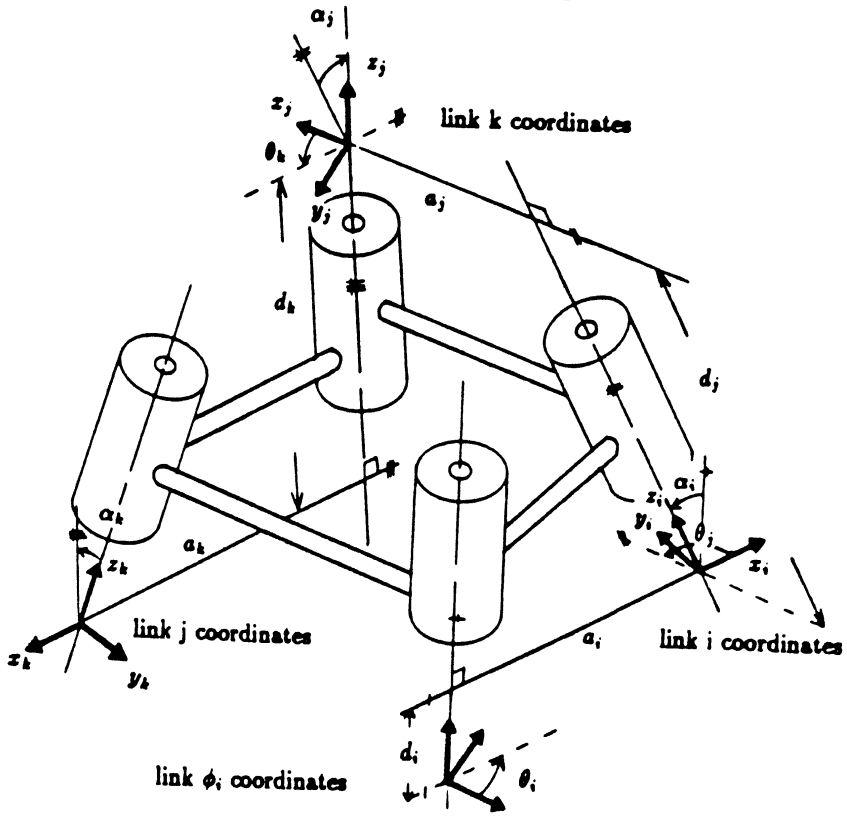

Figure 2: An Example Link, Numbered i

2.1 The Data Structure

In the implementation of the controller, several quantities associated with each link are needed, such as: the link velocity, acceleration, and inertial forces and moments. This section presents the data structure used to store this information. The form of this data structure is important since the design of the control algorithm is directly linked to this data structure. An example manipulator, illustrated in Figure 1, is used as bases for discussion.

In general, the number of rigid links in the manipulator is $m+1$. For the example in Figure 1, $m+1 = 8$. The base link is numbered 0 and all other links are numbered in an arbitrary order from 1 to m. It is apparent that a graph data structure could be used to store the information. Each record or item in the data structure would be associated with a particular link and all of the information needed about that link would be stored in the associated data record. The problem with using this type of data structure is that it does not lead to a particularly simple or efficient algorithm for the controller. A much better data structure is a binary tree data structure. The remainder of this section develops the method of obtaining this type of data structure.

We begin by selecting a set of joints such that if the associated links were disconnected at these joints there would be a unique sequence of links connecting any given link to the base link. An example of two joints which accomplish this are indicated in Figure 1 by the large arrows. The resulting structure is a tree structure and we can now establish relationships between different links in terms of their descendants and predecessors. For example, the descendants of link 1 are links 2 and 3. The predecessors of link 7 are links 0 and 6. The immediate descendant of link 1 is link 2 and the immediate predecessor of link 2 is link 1. We also note that some links may have more than one immediate descendant. For example, link 0 has three: 1, 6, and 4. The fact that a link could have an arbitrary number of immediate descendants cause practical problems in the implementation of the controller. To get around this problem we introduce the concept of connector links.

An example of a link with four joints is illustrated in Figure 2. The joint connecting the immediate predecessor to the link is assigned the same number as the link. One of the other three joints is used to locate the link coordinates. This determines the D-H kinematic parameters a_i, α_i, d_i, and θ_i as illustrated in the Figure 2. To locate the remaining two joints with respect to link i coordinates we insert fictitious links, j and k, called connector links. They are called connector links because they connect another sequence of links to the tree structure. First is link j which is located relative to link i coordinates. The D-H kinematic parameters [10], a_j, α_j, d_j, and θ_j are used to locate this link j coordinates with respect to link i coordinates. Next is link k which is located relative to link j coordinates. Again the D-H kinematic parameters, a_k, α_k, d_k, and θ_k are used to locate this link k coordinates with respect to link j coordinates. Note that all of the kinematic parameters, a, α d, and θ, associated with connector links are constant and that for both the connector and successor links the position and orientation of the descendant with respect to the current link is described using homogeneous transforms parameterized by the D-H kinematic parameters a, α, d and θ. The real links are

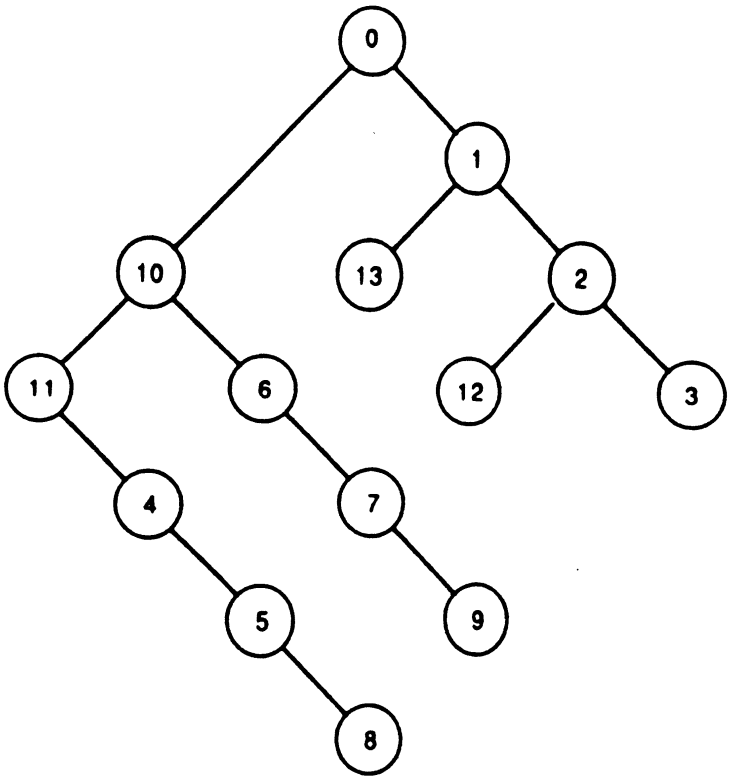

Figure 3: Binary Tree Structure

called successor links to distinguish them from the imaginary connector links.

Therefore, each link can have up to two immediate descendants. One would be a connector link and the other would be a successor link. Only one of each type is allowed. Hence, with the introduction of the connector links we have converted a general tree data structure into a binary tree data structure.

For a given link, the first joint encountered when moving in the direction of the base is numbered the same as the given link. Thus, all of the joints except those where the kinematic loops have been broken have been assigned a number. There are m of these. Assuming there are r independent loops the remaining r joints located at the points where the kinematic loops have been broken are numbered from $m+1$ to $m+r$ making a total of $m+r$ joints. Finally, a fictitious successor link called a terminating link is associated with these joints and are given the same number as the associated joint. The purpose of the terminating links are to allocate a item in the data structure to store all of the information concerning the associated joints, for example, their position and the viscous friction coefficients. The D-H kinematic parameters for the terminating links are chosen so that the loop closure equations can be written in terms of homogeneous transforms. This will be described in more detail in the section concerning the constraint equations.

If link i is a successor link then the associated joint is either translational or rotational. In either case the joint position is denoted by q_i. If the joint is translational then $q_i = d_i$. If the joint is rotational then $q_i = \theta_i$.

Finally, connector links are numbered starting at $m+r+1$ on up to the total number of links, both imaginary and real, contained in the manipulator.

The resulting data structure for the example manipulator is illustrated in Figure 3. The convention we use is that the items in the data structure associated with connector links are drawn to the left of its predecessor and the items associated with successor links are drawn to the right.

2.2 Manipulator Kinematics

¿From Figure 3 one can see that the position of each link can be determined with respect to base by starting at the base and successively computing each link position while moving out from the base. Let ϕ_i denote the number of the immediate predecessor of link i. If the homogeneous transform of link ϕ_i coordinates with respect to link 0 coordinates, $T_0^{\phi_i}$, is known then the homogeneous transform of link i coordinates with respect to the base is T_0^i and can be computing using the following equation.

$$T_0^i = T_0^{\phi_i} T_{\phi_i}^i$$

where $T_{\phi_i}^i$ is a function of the a_i, d_i, α_i, θ_i.

$$T_{\phi_i}^i = \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\theta_i)c(\alpha_i) & -c(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c()$ and $s()$ stand for the sine and cosine functions. Note that the same equation applies if link i is a connector link or a successor link.

This paper uses the spatial notation which was popularized by Featherstone, [12,13]. With this notation transformation matrices are 6×6 matrices. The spatial transformation matrix from link i to link ϕ_i is:

$$X_{\phi_i}^i = \begin{bmatrix} A_{\phi_i}^i & O \\ K(p_{\phi_i}^i)A_{\phi_i}^i & A_{\phi_i}^i \end{bmatrix}$$

where O is the 3×3 null matrix, $A_{\phi_i}^i$ and $p_{\phi_i}^i$ are the upper left 3×3 submatrix and upper right 3×1 submatrix of the homogeneous transformation matrix $T_{\phi_i}^i$, respectively. The 3×3 matrix $K()$ is a skew symmetric matrix such that $K(a)b = a \times b$ for any 3×1 vectors a and b.

As with homogeneous transforms, the spatial transformation from link i coordinates to link 0 coordinates can be computed given that of its immediate predecessor by multiplying the transforms together.

$$X_0^i = X_0^{\phi_i} X_{\phi_i}^i \tag{1}$$

Similarly, the spatial velocity and acceleration of link i can be determined given those of its

immediate predecessor, ϕ_i.

$$v_i = \begin{cases} v_{\phi_i} + s_i \dot{q}_i & \text{if link i is a successor} \\ v_{\phi_i} & \text{if link i is a connector} \end{cases} \tag{2}$$

$$\dot{v}_i = \begin{cases} \dot{v}_{\phi_i} + s_i \ddot{q}_i + v_i \times s_i \dot{q}_i & \text{if link i is a successor} \\ \dot{v}_{\phi_i} & \text{if link i is a connector} \end{cases} \tag{3}$$

where s_i is the third column of $X_0^{\phi_i}$ if joint i is rotational or the sixth column of $X_0^{\phi_i}$ if joint i is translational.

Thus, one starts at the root of the binary tree data structure and works out along the branches using the above equations to successively compute the spatial transformation matrix, velocity and the acceleration of each link.

2.3 Kinematic Constraint Equations

Since, in general, the manipulator is graph structured, there exist constraints in the relative position of each of the joints of the manipulator. These constraints can be represented by a set of equations of the following form:

$$q = U(Q) \tag{4}$$

where Q is an $n \times 1$ vector containing the positions of the joints where the actuators are located. That is, the manipulator has n degrees of freedom corresponding to the n actuators contained in the manipulator and for each Q_i there is a q_j such that $Q_i \equiv q_j$. Given $U(Q)$, $\dot{Q}$, and $\ddot{Q}$, the velocity of all the joints can be determined.

$$\dot{q} = E(Q)\dot{Q} \tag{5}$$

where

$$E(Q) = \frac{\partial U(Q)}{\partial Q} \tag{6}$$

The acceleration is given by:

$$\ddot{q} = E(Q)\ddot{Q} + \dot{E}(Q)\dot{Q} \tag{7}$$

For a given manipulator these equations are usually fairly simple. Often the matrix $E(Q)$ is constant. For example, it is simply the identity matrix for a serial link manipulator. However, for some manipulators these equations can become complex. For these manipulators we have found that the ϵ-algebra is very convenient for evaluating the time derivatives of the joint positions [14]. Using this algebra one only has to program to solution to Equation 4, change the order of the algebra and automatically obtain the solution to equations 5 and 7.

The equations of constraint are obtained from the loop closure equations. For the manipulator shown in Figure 1 there are two independent loops. The two loop closure equations are:

$$T_0^1(q_1)T_1^{13} = T_0^{10}T_{10}^{11}T_{11}^4(q_4)T_4^5(q_5)T_5^8(q_8) \tag{8}$$

and

$$T_0^1(q_1)T_1^2(q_2)T_2^{12} = T_0^{10}T_{10}^6(q_6)T_6^7(q_7)T_7^9(q_9) \tag{9}$$

These two matrix equations consist a total of 32 scaler equations of which only six are independent. Thus, we can determine q_1, q_2, q_4, q_6, q_8, and q_9 as a function of the actuator positions q_5 and q_7. Using the above notation, $q_5 = Q_1$, $q_7 = Q_2$ and $q_3 = Q_3$. These three equations along with the six obtained from equations 8 and 9 give us the function $U(Q)$ defined above.

2.4 Mass Parameters

There are two types of parameters of the manipulator model whose values are estimated by the adaptive controller. One is the viscous friction coefficients of the joints and the other is the parameters associated with the distribution of mass in each of the links.

There are ten quantities which specify the distribution of mass in each rigid link of the manipulator. These are the six unique components of the moment of inertia matrix, $\underline{J}_i$, the center of mass vector times the total mass, $\underline{r}_i$ and the total mass, m_i. These are all referred to link i coordinates and combined into a single 6×6 spatial moment of inertia matrix, $\underline{I}_i$.

$$\underline{I}_i = \begin{bmatrix} -K(\underline{r}_i) & m_i I \\ \underline{J}_i & K(\underline{r}_i) \end{bmatrix}$$

It is the ten unique components of this matrix we need to estimate for the adaptive control. The moment of inertia matrix,

$$\underline{J}_i = \begin{bmatrix} \underline{J}_{xxi} & \underline{J}_{xyi} & \underline{J}_{xzi} \\ \underline{J}_{yxi} & \underline{J}_{yyi} & \underline{J}_{yzi} \\ \underline{J}_{zxi} & \underline{J}_{zyi} & \underline{J}_{zzi} \end{bmatrix}$$

the center of mass vector times the mass,

$$\underline{r}_i = [\underline{r}_{xi} \;\; \underline{r}_{yi} \;\; \underline{r}_{zi}]^T$$

and the mass, m_i.

For notational purposes the parameters are combined into a single 10×1 vector, $\boldsymbol{m}_i$.

$$\begin{aligned} \boldsymbol{m}_i &= [m_{i1} \;\; m_{i2} \;\; m_{i3} \;\; m_{i4} \;\; m_{i5} \;\; m_{i6} \;\; m_{i7} \;\; m_{i8} \;\; m_{i9} \;\; m_{i10}]^T \\ &= [\underline{J}_{xxi} \;\; \underline{J}_{yyi} \;\; \underline{J}_{zzi} \;\; \underline{J}_{xyi} \;\; \underline{J}_{yzi} \;\; \underline{J}_{zxi} \;\; \underline{r}_{xi} \;\; \underline{r}_{yi} \;\; \underline{r}_{zi} \;\; m_i]^T \end{aligned}$$

In this way we can easily write $\underline{I}_i$ as a linear function of these parameters.

$$\underline{I}_i = \sum_{k=1}^{10} R_k m_{ik}$$

where the R_k are 6×6 matrices with one's where the associated parameter is located in $\underline{I}_i$ and zero's elsewhere. For example,

$$R_{10} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In the development of the dynamic equations of motion in the following section all quantities are referred to the base coordinate system. To refer $\underline{I}_i$ to the base coordinates one premultiplies and postmultiplies by the link i spatial transformation matrix.

$$I_i = X_0^i \underline{I}_i X_0^{i\prime}$$

The following is an important property of the spatial moment of inertia matrices. If y is a spatial vector and its derivative with respect to time exist then.

$$1/2 \frac{d}{dt}(y' I_i y) = y'(I_i \dot{y} + v_i \times I_i y) \tag{10}$$

where v_i is the spatial velocity of the $i-th$ link. This is easily shown using the definition of I_i and the fact that

$$\frac{dX_0^i}{dt} = v_i \times X_0^i$$

3 The Equations of Motion

The recursive Newton-Euler form of the equations of motion is used to model the manipulator. This form of the equation is especially useful for solving the inverse dynamics problem. In this problem the actuator torques/forces are computed given the joint positions, velocities and accelerations.

The approach used here is similar to that proposed by Luh and Zheng [11] except that the constraint forces are not needed. The method is different in that link inertial forces are used to determine the actuator forces instead of the actual forces on each link. In fact, the approach to modeling is more similar to Kane's [15] formulation than the Newton-Euler formulation.

The algorithm is a three step process.

1. Given the desired positions, velocities, and accelerations of the independent variables, Q, calculate the corresponding positions, velocities, and accelerations of the joints, q, using equations, 4 through 7.

2. For each link, determine ρ_j, the sum of the friction force and the projection of the sum of the inertial forces plus gravity loading on the axis of motion of joint j.

3. Determine the actuator forces, τ_j using the constraint equations and the ρ_j computed in step 2.

$$\tau = \boldsymbol{E}(Q)^T \rho$$

where τ is an $n \times 1$ vector of the actuator forces, and ρ is an $(m+r) \times 1$ vector of the ρ_j

Step 2 is exactly the same procedure which would be used if the manipulator was a serial link or tree structure manipulator. In this case the ρ_j would be identical to the actuator forces. The only difference is in the first step, wherein joint positions, velocities, and accelerations are determined which are consistent with the constraint equations, and the last step, wherein the constraint equations are again used in determining the actuator forces.

Step 2 can be implemented in three steps. First, the position, velocity and acceleration of each link is computed using equations 1 through 3. This is done by starting at the base and working out along the branches of the tree. Next, for each link j we determine $\boldsymbol{f}_j$, the sum of the gravitational and inertial forces of link j and all of its descendants. This is done by starting at the tips of the branches of the tree and working back toward the base of the tree using the following equations. Let $\boldsymbol{F}_j$ denote the inertial force on link j. Then:

$$\boldsymbol{F}_j = \dot{\boldsymbol{M}}_j = \boldsymbol{I}_j \dot{\boldsymbol{v}}_j + \boldsymbol{v}_j \times \boldsymbol{I}_j \boldsymbol{v}_j \tag{11}$$

where $\boldsymbol{M}_j = \boldsymbol{I}_j \boldsymbol{v}_j$ is the spatial momentum vector for the $j-th$ link. Note that $\boldsymbol{F}_j$ for connector and terminating links are zero since they have no mass. Let j be the index of the current link. Let k and l be the index of the successor and connector links. The vector $\boldsymbol{f}_j$ is used to denote the sum of the inertial and gravitational forces for link j and all of it's descendants. Then,

$$\boldsymbol{f}_j = \boldsymbol{F}_j - \boldsymbol{I}_j \boldsymbol{g} + \boldsymbol{f}_k + \boldsymbol{f}_l \tag{12}$$

where $\boldsymbol{g} = [0 \;\; 0 \;\; -9.8 \;\; 0 \;\; 0 \;\; 0]'$ is the spatial acceleration due to gravity vector.

The last step is to determine ρ_j, the force due to friction, η_j, plus the projection of $\boldsymbol{f}_j$ onto the axis of motion of joint j.

$$\rho_j = \boldsymbol{s}_j' \boldsymbol{f}_j + \eta_j \tag{13}$$

where η_j is the component of joint force due to friction.

This recursive form of the dynamic equations of motion can be shown to be a valid formulation by using the principle of virtual work. We begin by defining some sets which facilitate the development. The set Φ_i is the set of all joint numbers encountered while traversing the tree from link i to the base link. For the example manipulator in figure 1 $\Phi_4 = \{4, 10, 11\}$. The set $\Psi_j = \{i | j \in \Phi_i\}$ is the set consisting of i plus all of the indices of link i's descendants. For the example manipulator $\Psi_1 = \{1, 2, 3, 12, 13\}$.

Now, let $\dot{\tilde{q}}$ be a vector of virtual joint velocities, i.e. any set of joint velocities which are consistent with the constraint equations. The virtual velocity of link i, $\tilde{v}_i$, is written using the above notation as:

$$\tilde{v}_i = \sum_{j \in \Phi_i} s_j \dot{\tilde{q}}_j \tag{14}$$

Also,

$$f_j = \sum_{i \in \Psi_j} (F_i - I_i g) \tag{15}$$

We wish to show that the work done by inertial, gravitational, and friction forces, is equal to the work done by the active forces. That is:

$$dt \sum_{i=1}^{m+r} \left(\tilde{v}_i'(F_i - I_i g) + \dot{\tilde{q}}_i \eta_i \right) = dt \sum_{k=1}^{n} \dot{Q}_k \tau_k$$

Dividing by dt and successively using Equations 14, 30, and 15 on the left side of the above equation gives:

$$\begin{aligned}
\sum_{i=1}^{m+r} \left(\tilde{v}_i'(F_i - I_i g) + \dot{\tilde{q}}_i \eta_i \right) &= \sum_{i=1}^{m+r} [\sum_{j \in \Phi_i} (\dot{\tilde{q}}_j s_j'(F_i - I_i g)) + \dot{\tilde{q}}_i \eta_i] \\
&= \sum_{j=1}^{m+r} [\sum_{i \in \Psi_j} (\dot{\tilde{q}}_j s_j'(F_i - I_i g)) + \dot{\tilde{q}}_j \eta_j] \\
&= \sum_{j=1}^{m+r} (\dot{\tilde{q}}_j s_j' f_j + \dot{\tilde{q}}_j \eta_j) \\
&= \sum_{j=1}^{m+r} \dot{\tilde{q}}_j \rho_j = \sum_{j=1}^{m+r} \sum_{k=1}^{n} \dot{Q}_k e_{jk} \rho_j \\
&= \sum_{k=1}^{n} \dot{Q}_k \tau_k
\end{aligned}$$

Therefore, the recursive set of equations given above constitute a valid representation of the equations of motion of a manipulator.

4 Adaptive Controller

The basic approach used here is the same as presented by Slotine and Li. The differences in the control law and adaptation law are primarily a result of the use of the Newton-Euler formulation of the manipulator dynamics in the derivation.

We begin by defining a class of control laws and then discuss an important property of this class. This class of control laws have the following two characteristics.

1. The feedback control law is implemented through the use of a set of spatial feedback control vectors, $\hat{F}_j$, and functions, $\hat{\eta}_j$, which are defined for each link of the manipulator. These can be any functions of the desired trajectory and the actual trajectory of the manipulator.

2. The τ_j are computed using the $\hat{F}_j$ and $\hat{\eta}_j$ as in the following recursive equations:

$$\begin{aligned} \hat{f}_j &= \hat{F}_j + \hat{f}_k + \hat{f}_l \\ \hat{\rho}_j &= s_j' \hat{f}_j + \hat{\eta}_j \\ \tau &= E(Q)^T \hat{\rho} \end{aligned}$$

   where k and l are the immediate decendants of link j, $\hat{\rho}$ is an $(m+r) \times 1$ vector of the $\hat{\rho}_j$.

These controllers cover a fairly broad class which include classical PID controllers and inverse dynamic controllers. In this paper we show how the $\hat{F}_j$ and $\hat{\eta}_j$ can be chosen to implement an adaptive controller.

The following is an important property of this class of control systems.

$$\sum_{j=1}^{m+r} (\tilde{v}_j'(F_j - I_j g) + \dot{\tilde{q}}_j \eta_j) = \sum_{j=1}^{m+r} (\tilde{v}_j' \hat{F}_j + \dot{\tilde{q}}_j \hat{\eta}_j) \tag{16}$$

where the $\dot{\tilde{q}}_j$ are any joint velocities which are consistent with the constraint equations and $\tilde{v}_j$ is the resulting link velocity. Basically, this is simply a restatement of the principle of virtual work. If the $\dot{\tilde{q}}_j$ are the same as the actual joint velocities, $\dot{q}_j$, then the term on the left is the rate at which energy is absorbed by the manipulator and the term on the right is the rate at which energy is delivered to the arm by the actuators.

We now present the spatial feedback vectors and functions which implement the adaptive controller. First a reference trajectory, denoted by $\dot{Q}(t)$ is defined. It is generated from the desired trajectory and the actual trajectory by solving the following differential equation.

$$\dot{\tilde{Q}} = \dot{Q}_d - \Lambda Q_e$$

where $\dot{Q}_d$ is the preplanned desired trajectory; the tracking error is $Q_e = Q - Q_d$; and Λ is a positive definite diagonal matrix. Rearranging the above equation gives:

$$\dot{Q}_e + \Lambda Q_e = \dot{\tilde{Q}} \tag{17}$$

where $\tilde{Q} = Q - \hat{Q}$.

¿From these we compute the corresponding joint velocities and accelerations:

$$q = U(Q)$$

$$
\begin{aligned}
\dot{q} &= E(Q)\dot{Q} \\
\dot{\hat{q}} &= E(Q)\dot{\hat{Q}} \\
\dot{\tilde{q}} &= \dot{q} - \dot{\hat{q}} \\
\ddot{\hat{q}} &= E(Q)\ddot{\hat{Q}} + \dot{E}(Q)\dot{\hat{Q}}
\end{aligned}
$$

Note that $\dot{E}$ is a function of Q and $\dot{Q}$.

Next we compute the link velocities, v_i and accelerations, $\dot{v}_i$ using Equations 2 and 3. Then we compute the link velocities and accelerations using $\dot{\tilde{q}}$ and $\dot{\hat{q}}$ as joint velocites.

$$
\hat{v}_i = \begin{cases} \hat{v}_{\phi_i} + s_i\dot{\hat{q}}_i & \text{if link i is a successor} \\ \hat{v}_{\phi_i} & \text{if link i is a connector} \end{cases} \tag{18}
$$

$$
\dot{\hat{v}}_i = \begin{cases} \dot{\hat{v}}_{\phi_i} + s_i\ddot{\hat{q}}_i + v_i \times s_i\dot{\hat{q}}_i & \text{if link i is a successor} \\ \dot{\hat{v}}_{\phi_i} & \text{if link i is a connector} \end{cases} \tag{19}
$$

$$
\tilde{v}_i = v_i - \hat{v}_i \tag{20}
$$

The spatial feedback vectors and functions are:

$$
\hat{F}_j = \hat{I}_j\dot{\hat{v}}_j + \hat{v}_j \times \hat{I}_j\hat{v}_j - \hat{I}_j g - \gamma_j I_j^0 \tilde{v}_j \tag{21}
$$

$$
\hat{\eta}_j = \hat{d}_j\dot{q}_j \tag{22}
$$

where γ_j is a positive scaler and I_j^0 and $\hat{I}_j$ are the initial and the current estimates of I_j, respectively. The equations for updating the parameter estimates are:

$$
\dot{\hat{m}}_j = -(1/\alpha_j)w_j \tag{23}
$$

$$
\dot{\hat{d}}_j = -(1/\beta_j)\dot{\tilde{q}}_j\dot{q}_j \tag{24}
$$

where w_j is a 10×1 vector whose $k-th$ component is,

$$
w_{jk} = \underline{\tilde{v}}_j{}'(R_k\dot{\underline{\hat{v}}}_j + \underline{\hat{v}}_j \times R_k\underline{\hat{v}}_j - R_k\underline{g}_j) \tag{25}
$$

and $\underline{\tilde{v}}_j, \dot{\underline{\hat{v}}}_j, \underline{\hat{v}}_j$, and $\underline{g}_j$ are the vectors $\tilde{v}_j, \dot{\hat{v}}_j, \hat{v}_j$, and g referred to link j coordinates, respectively.

Global convergence of the tracking can be shown by first realizing that Equation 17 is a linear differential equation in Q_e and $\tilde{Q}$ and that we have chosen the gain matrix Λ to be symmetric and positive definite. Thinking of this as a system with $\tilde{Q}$ as the input and Q_e as the output we know that this system is bibo stable. Therefore, if we can show that the input, $\dot{\tilde{Q}}$, is bounded and $\dot{\tilde{Q}} \to 0$ as $t \to \infty$ then we are assured that $Q_e \to 0$ and that global tracking has been realized. To show that $\dot{\tilde{Q}} \to 0$ as $t \to \infty$ we use the following Lyapunov function:

$$
V(t) = 1/2 \sum_{j=1}^{m+r} (\tilde{v}_j' I_j \tilde{v}_j + \alpha_j \tilde{m}_j^T \tilde{m}_j + \beta_j \tilde{d}_j^2) \tag{26}
$$

where the α_j and β_j are positive scalers, $\tilde{m}_j = m_j - \hat{m}_j$, and $\tilde{d}_j = d_j - \hat{d}_j$. Since $V(t)$ is bounded from below, by showing that $\dot{V}(t) \leq 0$ we have shown that $V(t) \rightarrow$ *constant* and therefore, $\dot{V}(t) \rightarrow 0$.

First we consider the derivative of the first term on the right hand side of equation 26. ¿From Equation 10 we get:

$$\begin{aligned} 1/2\frac{d}{dt}(\tilde{\boldsymbol{v}}_j' \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) &= \tilde{\boldsymbol{v}}_j'(\boldsymbol{I}_j \dot{\tilde{\boldsymbol{v}}}_j + \boldsymbol{v}_j \times \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) \\ &= \tilde{\boldsymbol{v}}_j'(\dot{\tilde{\boldsymbol{M}}}_j - \boldsymbol{I}_j(\tilde{\boldsymbol{v}}_j \times \boldsymbol{v}_j)) \end{aligned}$$

where

$$\dot{\tilde{\boldsymbol{M}}}_j = \frac{d(\boldsymbol{I}_j \tilde{\boldsymbol{v}}_j)}{dt} = \boldsymbol{I}_j \dot{\tilde{\boldsymbol{v}}}_j + \boldsymbol{v}_j \times \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j + \boldsymbol{I}_j(\tilde{\boldsymbol{v}}_j \times \boldsymbol{v}_j)$$

However, since $\tilde{\boldsymbol{v}}_j = \boldsymbol{v}_j - \hat{\boldsymbol{v}}_j$, then

$$\begin{aligned} \dot{\tilde{\boldsymbol{M}}}_j &= \dot{\boldsymbol{M}}_j - \dot{\hat{\boldsymbol{M}}}_j \\ &= \boldsymbol{F}_j - (\boldsymbol{I}_j \dot{\hat{\boldsymbol{v}}}_j + \boldsymbol{v}_j \times \boldsymbol{I}_j \hat{\boldsymbol{v}}_j + \boldsymbol{I}_j(\hat{\boldsymbol{v}}_j \times \boldsymbol{v}_j)) \end{aligned}$$

where

$$\dot{\hat{\boldsymbol{M}}}_j = \frac{d(\boldsymbol{I}_j \hat{\boldsymbol{v}}_j)}{dt}$$

Therefore,

$$\sum_{j=1}^{m+r} 1/2\frac{d}{dt}(\tilde{\boldsymbol{v}}_j' \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) = \sum_{j=1}^{m+r} \tilde{\boldsymbol{v}}_j'(\boldsymbol{F}_j - (\boldsymbol{I}_j \dot{\hat{\boldsymbol{v}}}_j + \boldsymbol{v}_j \times \boldsymbol{I}_j \hat{\boldsymbol{v}}_j))$$

and since $\boldsymbol{v}_j = \tilde{\boldsymbol{v}}_j + \hat{\boldsymbol{v}}_j$ then,

$$\sum_{j=1}^{m+r} 1/2\frac{d}{dt}(\tilde{\boldsymbol{v}}_j' \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) = \sum_{j=1}^{m+r} \tilde{\boldsymbol{v}}_j'(\boldsymbol{F}_j - (\boldsymbol{I}_j \dot{\hat{\boldsymbol{v}}}_j + \hat{\boldsymbol{v}}_j \times \boldsymbol{I}_j \hat{\boldsymbol{v}}_j))$$

Using Equation 16 gives:

$$\begin{aligned} \sum_{j=1}^{m+r} 1/2\frac{d}{dt}(\tilde{\boldsymbol{v}}_j' \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) &= \sum_{j=1}^{m+r} \tilde{\boldsymbol{v}}_j'(\hat{\boldsymbol{F}}_j - (\boldsymbol{I}_j \dot{\hat{\boldsymbol{v}}}_j + \hat{\boldsymbol{v}}_j \times \boldsymbol{I}_j \hat{\boldsymbol{v}}_j + \boldsymbol{I}_j g)) \\ &\quad + \sum_{j=1}^{m+r} \dot{\tilde{q}}_j(\hat{\eta}_j - \eta_j) \end{aligned}$$

Substituting in the equations for $\hat{\boldsymbol{F}}_j$, η_j and $\hat{\eta}_j$ gives:

$$\sum_{j=1}^{m+r} 1/2\frac{d}{dt}(\tilde{\boldsymbol{v}}_j' \boldsymbol{I}_j \tilde{\boldsymbol{v}}_j) = \sum_{j=1}^{m+r} \tilde{\boldsymbol{v}}_j'(\tilde{\boldsymbol{I}}_j \dot{\hat{\boldsymbol{v}}}_j + \hat{\boldsymbol{v}}_j \times \tilde{\boldsymbol{I}}_j \hat{\boldsymbol{v}}_j - \tilde{\boldsymbol{I}}_j g)$$

$$
\begin{aligned}
&- \sum_{j=1}^{m+r} \gamma_j \tilde{v}_j I_j^0 \tilde{v}_j + \sum_{j=1}^{m+r} \tilde{d}_j \dot{\tilde{q}}_j \dot{q}_j \\
&= \sum_{j=1}^{m+r} \tilde{m}_j^T w_j - \sum_{j=1}^{m+r} \gamma_j \tilde{v}_j I_j^0 \tilde{v}_j \\
&\quad + \sum_{j=1}^{m+r} \tilde{d}_j \dot{\tilde{q}}_j \dot{q}_j
\end{aligned} \tag{27}
$$

The derivative of the second term on the right hand of Equation 26 is:

$$
\begin{aligned}
\sum_{j=1}^{m+r} 1/2 \alpha_j \frac{d}{dt}(\tilde{m}_j^T \tilde{m}_j) &= \sum_{j=1}^{m+r} \alpha_j \tilde{m}_j^T \dot{\tilde{m}}_j = \sum_{j=1}^{m+r} \alpha_j \tilde{m}_j^T \dot{\tilde{m}}_j \\
&= - \sum_{j=1}^{m+r} \tilde{m}_j^T w_j
\end{aligned} \tag{28}
$$

The derivative of the third term on the right hand of Equation 26 is:

$$
1/2 \sum_{j=1}^{m+r} \frac{d\tilde{d}_j^2}{dt} = \sum_{j=1}^{m+r} \beta_j \tilde{d}_j \dot{\tilde{d}}_j = - \sum_{j=1}^{m+r} \tilde{d}_j \dot{\tilde{q}}_j \dot{q}_j \tag{29}
$$

Therefore, adding Equations 27, 28, and 29 gives:

$$
\dot{V}(t) = - \sum_{j=1}^{m+r} \gamma_j \tilde{v}_j{}' I_j^0 \tilde{v}_j \leq 0
$$

¿From the above, we now know that $\dot{V}(t) \rightarrow 0$. It is easy to choose the I_j^0 such that $I_j^0 > 0$. Therefore, $\dot{\tilde{Q}} \rightarrow 0$ and hence, global stability of the tracking is assured. That is, $Q_e \rightarrow 0$.

5 Computational Algorithm

The purpose of this section is to present the exact steps used in the implementation of the adaptive control algorithm. The controller is organized so that changing the control algorithm only involves the changing a two procedures within the controller program.

As in most inverse dynamics type algorithms computational efficiency is improved by computing all velocities and accelerations referred to their own link coordinates. This change of coordinates will be denoted by underlining the variable in what follows. For example,

$$
\underline{v}_i = (X_0^i)' v_i
$$

The spatial vector s_i is referred to link ϕ_i coordinates since they are constant in that coordinate system. That is,

$$\underline{s}_i = (X^i_{\phi_0})' s_i$$

Another method used to speed up the computations is to add the acceleration due to gravity to the base coordinate acceleration at the beginning of the computation. This avoids having to explicitly compute the gravity loading for each link as is implied by the previous equations.

The following three procedures **traverse**, **forward**, and **backward** are used in the implementation of the controller. The procedure **traverse** implements a traversal of the binary tree data structure where all of the computed quantities are stored. This is a basic procedure which is used in the laboratory regardless of the particular control algorithm being implemented. The input to the procedure is a link number and the output is $\hat{l}_i$ which is equal to $\hat{f}_i$ referred to link ϕ_i coordinates. This ordering of the traversal of the tree directly corresponds to the ordering of the computations implied by the above equations. Each time a new node in the tree is visited on the trip down through the data structure the procedure **forward** is called. This procedure is used to update any state variable information associated with the control law. The input to this procedure is the current link number and there is no output returned from the procedure. Only data local to this procedure and the **backward** procedure is changed. On the trip back up through the data structure procedure **traverse** calls the procedure **backward**. The input to this procedure is the current link number and the output is the spatial feedback vector, $\underline{\hat{F}}_i$, and the scaler feedback function, $\hat{\eta}_i$ for the current link.

The procedure **traverse** is initially called with $i = 0$, the number of the base link.

PROCEDURE traverse(i,$\hat{l}_i$)
begin

1. If $i = 0$ then set

$$\begin{aligned} q &= U(Q) \\ \dot{q} &= E(Q)\dot{Q} \\ \underline{v}_0 &= 0 \end{aligned}$$

   else if link i is a successor link set

$$\underline{v}_i = (X^i_{\phi_i})'(\underline{v}_{\phi_i} + \underline{s}_i \dot{q}_i)$$

   else if link i is a connector link set

$$\underline{v}_i = (X^i_{\phi_i})' \underline{v}_{\phi_i}$$

2. Call **forward**(i).

3. If a connector link for link i exist and its number is k then call **traverse**(k,$\hat{l}_k$), else set $\hat{l}_k = 0$.

4. If a successor link for link i exist and its number is j then call **traverse**(j,$\hat{\underline{l}}_j$), else set $\hat{\underline{l}}_j = 0$.

5. Call **backward**(i,$\hat{\underline{F}}_i$,$\hat{\eta}_i$).

6. Set:

$$\begin{aligned}\hat{\underline{f}}_i &= \hat{\underline{F}}_i + \hat{\underline{l}}_k + \hat{\underline{l}}_j \\ \hat{\underline{l}}_i &= X^i_{\phi_i}\hat{\underline{f}}_i \\ \hat{\rho}_i &= \underline{s}_i\hat{\underline{l}}_i + \hat{\eta}_i\end{aligned}$$

7. If $i = 0$ then set:

$$\tau = E(Q)^T\hat{\rho}$$

and output τ to the manipulator.

end;

PROCEDURE forward(i)
begin

1. If $i = 0$ then set

$$\begin{aligned}\dot{\tilde{Q}} &= \dot{Q}_d - \Lambda Q_e \\ \ddot{\tilde{Q}} &= \ddot{Q}_d - \Lambda \dot{Q}_e \\ \dot{\hat{q}} &= E(Q)\dot{\hat{Q}} \\ \ddot{\hat{q}} &= E(Q)\ddot{\hat{Q}} + \dot{E}(Q)\dot{\hat{Q}} \\ \dot{\tilde{q}} &= \dot{q} - \dot{\hat{q}} \\ \underline{\tilde{v}}_0 &= \underline{\hat{v}}_0 = 0 \\ \dot{\underline{\hat{v}}}_0 &= -g\end{aligned}$$

else if link i is a successor link set

$$\begin{aligned}\underline{\hat{v}}_i &= (X^i_{\phi_i})'(\underline{\hat{v}}_{\phi_i} + \underline{s}_i\dot{\hat{q}}_i) \\ \dot{\underline{\hat{v}}}_i &= (X^i_{\phi_i})'(\dot{\underline{\hat{v}}}_{\phi_i} + \underline{v}_{\phi_i} \times \underline{s}_i\dot{\hat{q}}_i + \underline{s}_i\ddot{\hat{q}}_i) \\ \underline{\tilde{v}}_i &= \underline{v}_i - \underline{\hat{v}}_i\end{aligned}$$

and compute w_i using Equation 25,
else if link i is a connector link set

$$\begin{aligned}\underline{\hat{v}}_i &= (X^i_{\phi_i})'\underline{\hat{v}}_{\phi_i} \\ \dot{\underline{\hat{v}}}_i &= (X^i_{\phi_i})'\dot{\underline{\hat{v}}}_{\phi_i} \\ \underline{\tilde{v}}_i &= \underline{v}_i - \underline{\hat{v}}_i\end{aligned}$$

end;

PROCEDURE backward(i,$\hat{\underline{F}}_i$,$\hat{\eta}_i$)
begin

1. If link i is a successor link then set:

$$\begin{aligned}
\hat{\underline{F}}_i &= \hat{\underline{I}}_i\dot{\hat{\underline{v}}}_i + \hat{\underline{v}}_i \times \hat{\underline{I}}_i\hat{\underline{v}}_i - \gamma_i I_i^0 \tilde{\underline{v}}_i \\
\hat{\eta}_i &= \hat{d}_i\dot{q}_i \\
\dot{\hat{m}}_i &= -1/\alpha_i w_i \\
\dot{\hat{d}}_i &= -1/\beta_i \dot{\tilde{q}}_i\dot{q}_i
\end{aligned}$$

and update the parameter estimates,
else if link i is a terminating link then set:

$$\begin{aligned}
\hat{\underline{F}}_i &= 0 \\
\hat{\eta}_i &= \hat{d}_i\dot{q}_i \\
\dot{\hat{d}}_i &= -1/\beta_i \dot{\tilde{q}}_i\dot{q}_i
\end{aligned}$$

and update the parameter estimates,
else set $\hat{\underline{F}}_i = 0$, $\hat{\eta}_i = 0$.

end;

6 Conclusion

An efficient algorithm for the adaptive control of a manipulator containing closed kinematic loops has been presented. The issues that have been addressed are computational efficiency and generality. Generality was obtained by formulating the algorithm for manipulators containing closed kinematic loops. Computational efficiency was obtained with the use of a new inverse dynamics model for manipulators with closed kinematic loops. This model achieves improved computational efficiency by eliminating the need to compute the constraint forces. Only inertial and gravitational forces are actually computed by the algorithm.

The algorithm used in the implementation of the adaptive controller directly follows from the binary data structure used in modeling the manipulator. Three procedures were presented for its implementation. The first is the procedure **traverse** which is used in the traversal of the binary tree data structure. Each node in the data structure corresponds to a different link in the manipulator. As each node is visited the controller calls a second procedure called **forward**. This procedure is used to update any state information associated with the control law for the particular node being visited. On return steps of the traversal algorithm the controller

calls a third procedure called **backward**. The function of this procedure is to compute the spatial feedback vectors and functions which define the control law being implemented. After traversing the entire tree the computed torques are outputted to the manipulator to effect the control. It is difficult to quantify the computational complexity of the algorithm since a general closed form solution to the constraint equations does not exist. However, it is clear that for a serial link manipulator the number of computations is linear in the number links contained in the manipulator.

Manipulator controllers should be designed to perform a variety of different classes of compliant motion task. An example of a class could be opening door. Every door is basically the same. Only the size, friction, and mass properties change. It might seem that the adaptive controller presented in this paper could be applied to the compliant motion control problem. However, the model used by this controller is special in that the number of degrees of freedom of the manipulator is the same as the number of actuators. This fact allows us to simplify the model and the formulation of the controller. However, there is little difference between the models presented in this paper and the model for a manipulator preforming a compliant motion task. The main difference is the number of degrees of freedom of the combined manipulator and task are now less than the number of actuators. Thus, we have an infinite number of choices of actuator forces which produce the same motion of the end-effector. As in the hybrid compliant motion control scheme one would need to control both force and position. In addition, compensating for modeling errors in the task kinematics is likely to play an important role in future adaptive compliant motion controllers.

7 Appendix

The following identity is used: For any function, $f(i,j)$ of the indices i and j:

$$\sum_{i=1}^{m+r} \sum_{j \in \Phi_i} f(i,j) = \sum_{j=1}^{m+r} \sum_{i \in \Psi_j} f(i,j) \tag{30}$$

References

[1] J. Y. S. Luh, M. W. Walker, and R. P. Paul, "On-Line Computational Scheme for Mechanical Manipulators", *Trans. AME Journal of Dynamic Systems, Measurement, and Control*, vol. 12(2), pp. 69–76, 1980.

[2] D.E. Orin, R.B. McGhee, M. Vukobratovic, and G. Hartoch, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods", *Mathematical Biosciences*, vol. 43 No. 1/2, February 1979.

[3] A. J. Koivo and T. Guo, "Adaptive Linear Controller for Robotic Manipulators", *IEEE Trans. on Auto. Contr.*, vol. AC-28, 1983.

[4] A. J. Koivo, Force-position-velocity control with self-tuning for robotic manipulators, In *I.E.E.E. Int. Conf. Robotics and Automation*, San Francisco, 1986.

[5] G. Leininger, "Self-tuning Control of Manipulators", *Int'l Symp. on Advanced Software in Robotics*, 1983.

[6] S. Dubowsky and D. T. Desforges, "The Application of Model-Reference Adaptive Control to Robotic Manipulators", *J. Dyn. Syst. Meas. Contr.*, vol. 101, 1979.

[7] J. J. Craig, P. Hsu, and S. S. Sastry, Adaptive control of mechanical manipulators, In *IEEE Int'l Conf. on Robotics and Automation*, San Francisco, CA, 1986.

[8] P. Hsu, M. Bodson, S. Sastry, and B. Paden, Adaptive identification and control for manipulators without using joint accelerations, In *IEEE Int'l Conf. on Robotics and Automation*, Raleigh, North Carolina, 1987.

[9] J. E. Slotine and W. Li, "On the Adaptive Control of Robot Manipulators", *International Journal of Robotics Research*, vol. 6(3), 1987.

[10] J. Denavit and R. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *AME Journal of Applied Mechanics*, , pp. 215–221, June 1955.

[11] J. Y. S. Luh and Y. F. Zheng, "Computation of Input Generalized Forces for Robotics with Closed Kinematic Chain Mechanisms", *IEEE J. of Robotics and Automation*, vol. 4, pp. 95–103, 1985.

[12] R. Featherstone, "The Calculation of Robot Dynamics Using Articulated-Body Inertias", *The Int. J. of Robotics Res.*, vol. 2, no. 1, pp. 13–30, Spring 1983.

[13] R. Lathrop, Constrained (closed-loop) robot simulation by local constraint propagation, In *I.E.E.E. International Conference on Robotics and Automation*, San Francisco, CA, 1986.

[14] M. W. Walker, Manipulator kinematics and the epsilon algebra, In *I.E.E.E. International Conference on Robotics and Automation*, Raleigh, North Carolina, 1987.

[15] T.R. Kane and D. A. Levinson, "The use of Kane's dynamic equations in robotics", *The Int. J. of Robotics Res.*, vol. 2, no. 3, pp. 3–21, 1983.