AN APPROACH TO OUTPUT FEEDBACK ADAPTIVE CONTROL
FOR ROBOT MANIPULATORS
H.M. Schwartz \(^1\)^\(^2\) and J.M. Daly

Abstract

This paper presents a practical approach to output feedback adaptive control for robot manipulators. A novel approach is used to design the adaptive controller. Simulations and experimental results are used to demonstrate the effectiveness of the method. The approach to the controller design is based on ease of design and implementation and not on achieving theoretical stability results. Nevertheless, the origin is shown to be locally asymptotically stable.

Key Words

Adaptive control, robot control.

1. Introduction

The goal of this paper is to develop a practical and relatively tractable nonlinear adaptive output feedback adaptive robot controller. The purpose of this work is to facilitate the design and implementation of these controllers. A relatively direct method of controller design is proposed that is accessible to the practitioner.

\(^1\)Howard M. Schwartz, Department of Systems and Computer Engineering, 1125 Colonel By Drive, Carleton University, Ottawa, Ontario, K1S 5B6, Canada

\(^2\)E-mail: schwartz@sce.carleton.ca
During the last two decades a number of output feedback adaptive control algorithms have been proposed. The early seminal work in the field was done by Craig, Hsu and Sastry [1]. They developed an adaptive control law based on techniques associated with feedback linearization [2] and early robot controllers based on the robotics version of feedback linearization, namely, the computed torque technique. The Craig, Hsu and Sastry adaption law was based on Lyapunov methods of adaptive control [3]. The difficulty with the Craig, Hsu and Sastry method was the need for measurement of the acceleration and the velocity for the parameter adaptation law and the control law. Slotine and Li [4] then used passivity techniques and the structural dynamics of the manipulator to develop an adaptive control law that no longer required the acceleration signal. However, one was required to use various auxiliary signals in the algorithms that further complicated the controller design and adaptation law. Furthermore, it has been noted that the algorithm can fail due to parameter drift associated with correlated velocity noise in the adaptation law [5]. To address the issue of parameter estimation drift due to noise Sadegh and Horowitz [6] proposed a method whereby the adaptation regression vector did not depend on the velocity measurements and it only depended on the desired trajectory. However, the requirement to use a relatively noisy velocity signal in the control law limited the performance of these methods.

Other researchers proposed methods of output feedback adaptive control in which the velocity signal was in some way estimated [7]. The method proposed by Candus de Wit and Fixot [8] [9], proposed a nonlinear observer to estimate the joint velocities. These methods were also based on passivity and as such require auxiliary signals and observer or estimator gains that would have to meet Lyapunov based stability conditions. Another approach is to use an extended Kalman filter to estimate both the
state and the inertial parameters [10]. However, such methods require careful tuning of the filter parameters. A number of papers suggested using high gain observers for nonlinear output feedback adaptive control [11]. This method was then applied to the robotics application [12]. The difficulty with using the high gain observers are once again, choosing gains sufficiently high to meet stability criteria and the subsequent implementation is therefore not practical as the high gain observers can excite unmodeled higher order dynamics resulting in a chattering behaviour. Furthermore, it is a difficult design process to find parameters that meet the stability criteria.

The goal of this paper is to present a design procedure for nonlinear output feedback adaptive control for robot manipulators. The design method will be relatively easy to follow and practical to implement. The usefulness of the proposed design method will be demonstrated by use of a simulated robot and by experimental results on a direct drive robot. Finally we will demonstrate local stability. Essentially we are proposing an opposite approach to the prevailing methods. Instead of proposing a relatively complicated controller design to meet stability requirements, we are proposing a relatively simple and practical design but with a restricted local stability argument.

The paper is organized as follows: section 2 presents the algorithm that is being proposed, section 3 presents the simulation and experimental results and finally section 4 presents a stability argument.

2. Design of an Output Feedback Adaptive Controller for Robot Manipulators

The algorithm being proposed is a relatively straightforward extension of the seminal
work of Craig, Hsu and Sastry [1]. Their work is a direct result of work on the computed torque technique and feedback linearization [11]. Therefore the design begins with the classical computed torque linearization and decoupling law. Given the standard formulation of the robot dynamics as,

\[ T = M(q)\ddot{q} + C(q, \dot{q}) + G(q) \]  

(1)

where \( T \) is a \( n \times 1 \) vector of torques applied to the joints, \( M(q) \) is the \( n \times n \) mass (or inertia) matrix, \( q \) is the \( n \times 1 \) vector of joint positions, \( C(q, \dot{q}) \) is the \( n \times 1 \) vector of centrifugal and Coriolis terms, and \( G(q) \) is the \( n \times 1 \) gravity vector.

The dynamics of (1) may also be expressed in linear regression form for \( p \) robot parameters as,

\[ T = Y(q, \dot{q}, \ddot{q})\theta \]

where \( Y(q, \dot{q}, \ddot{q}) \) is an \( n \times p \) matrix of known functions, and \( \theta \) is a \( p \times 1 \) vector of robot parameters. The exact linearizing equation is,

\[ T_L = M(q)v + C(q, \dot{q}) + G(q) \]  

(2)

where \( v \) is given by,

\[ v = k_p e + k_d \dot{e} + \ddot{q}_d \]  

(3)

and \( e = q_d - q, \dot{e} = \dot{q}_d - \dot{q}, \) where \( q_d, \dot{q}_d \) and \( \ddot{q}_d \) are the desired robot trajectories and \( k_p \) and \( k_d \) are diagonal matrices. Assuming perfect knowledge of the inertial parameters, \( \theta \), and perfect measurements of the robot joint positions, velocities and accelerations, one gets the following well known error equations by substituting, \( T = T_L \) into (1):

\[ M(q)v + C(q, \dot{q}) + G(q) = M(q)\ddot{q} + C(q, \dot{q}) + G(q) \]  

(4)
Then one gets, $\ddot{q} = v$ and $\ddot{q} = k_p e + k_d \dot{e} + \ddot{q}_d$, resulting in the error equation,

$$\dddot{e} + k_d \dot{e} + k_p e = 0$$  \hspace{1cm} (5)

Equation (5) represents the trajectory tracking error dynamics and by choosing appropriate values for the diagonal matrices $k_d$ and $k_p$ one can arbitrarily set the specified closed loop tracking dynamics for each robot joint. Therefore, the first step in the design process is to choose $k_d$ and $k_p$ independently for each robot joint to meet desired performance specifications.

However, given that the velocity signal is not available one designs a separate linear observer to estimate the velocity signal. Assuming that the linearization law based on the computed torque technique is known perfectly; there are no parameter errors and $\theta$ is known, then the dynamics for each joint of the robot appears as a second order system that can be written as:

$$\begin{bmatrix}
\dot{\hat{q}}_i \\
\ddot{\hat{q}}_i
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
\hat{q}_i \\
\dot{\hat{q}}_i
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} v_i$$ \hspace{1cm} (6)

Then design a linear observer for the system defined by (6). The linear observer for each joint becomes:

$$\begin{bmatrix}
\dot{\hat{q}}_i \\
\ddot{\hat{q}}_i
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
\hat{q}_i \\
\dot{\hat{q}}_i
\end{bmatrix} + \begin{bmatrix}
0 \\
1
\end{bmatrix} v + \begin{bmatrix}
l_1 \\
l_2
\end{bmatrix} (q_i - \hat{q}_i)$$ \hspace{1cm} (7)

In matrix form this becomes:

$$\dot{\hat{q}}_i = (A_0 - LH)\hat{q}_i + LH \dot{q}_i + bv$$ \hspace{1cm} (8)

where $H = [1, 0]$. The design procedure for the observer for each joint is to design the observer gain, $L$, such that the characteristic values of $(A_0 - LH)$ are approximately 3 to 5 times faster than the dominant modes of (5). This is a common observer design practice.
Finally, the adaptation law is designed based on the seminal work of Craig, Hsu and Sastry, [1] however, instead of the actual joint velocity and acceleration signals one uses the estimates based on the individual joint observers given by (7). The adaptation law becomes:

\[ \hat{\dot{\theta}} = \Gamma Y(q, \dot{\hat{q}}, \ddot{\hat{q}}) \hat{M}^{-1}(q) E_1 \]  

(9)

where \( E_1 = \dot{e} + \psi e \), for simplicity, choose \( \psi = 1 \), and choose \( \Gamma \) based on desired adaptation rate, and the choice of \( \Gamma = I \), is a suitable initial design choice. Then implement the controller based on the “certainty equivalence principle” as:

\[ T_L = Y(q, \dot{\hat{q}}, v) \hat{\dot{\theta}} = \hat{M}(q)v + \hat{C}(q, \dot{\hat{q}}) + \hat{G}(q) \]  

(10)

where \( v = k_p e + k_d \dot{e} + \ddot{q}_d \), and \( \dot{e} = \dot{q}_d - \dot{\hat{q}} \). Equations (7), (9) and (10) define the nonlinear output feedback adaptive controller. The design is now complete.

3. Simulation and Experiments

Simulations and experiments on a direct drive robot are used to demonstrate the ease of design, the ease of implementation and the excellent performance of the resulting systems. Section 3.1 will discuss the simulation results and section 3.2 will discuss the experimental results.

3.1 Simulation Results

For the simulations in this work, the dynamics of a \( n = 2 \) degree of freedom serial link manipulator are used. The equation of dynamics for this manipulator comes from [13] and is repeated here for convenience. The dynamics take the form of (1) but the robot operates in the horizontal plane, and as such \( G(q) \) is zero. There are \( p = 2 \) parameters to be estimated for the robot, they are:
\[
\theta = \begin{bmatrix}
\theta_1 \\
\theta_2
\end{bmatrix} = \begin{bmatrix}
m_1 l^2 \\
m_2 l^2
\end{bmatrix}
\]

The mass and coriolis matrices are given as:

\[
M(q) = \begin{bmatrix}
\theta_1 + 2\theta_2 + 2\theta_2 \cos q_2 & \theta_2 + \theta_2 \cos q_2 \\
\theta_2 + \theta_2 \cos q_2 & \theta_2
\end{bmatrix} \tag{11}
\]

\[
C(q, \dot{q}) = \begin{bmatrix}
-2\theta_2 \dot{q}_2 \sin q_2 & -\theta_2 \dot{q}_2 \sin q_2 \\
\theta_2 \dot{q}_1 \sin q_2 & 0
\end{bmatrix} \tag{12}
\]

To generate the trajectory for the robot to follow, a command signal consisting of a square wave with a period of 20 seconds was used. The square wave has peak values of ±1 radians. This signal was pre-filtered using a critically damped second-order linear filter with a bandwidth of \(\omega_n = 2.0\ \text{rad/s}\). The transfer function for this filter is given as:

\[
G(s) = \frac{4}{s^2 + 4s + 4} \tag{13}
\]

Cases free of noise were run, as well as cases having uniformly distributed random noise to simulate the quantization error of a 12-bit position resolver. In all simulations, the controller dynamics were set to have a bandwidth \(\omega_n\) of 2.0 rad/sec, and a damping ration \(\zeta\) of 1, matching the bandwidth of the trajectory pre-filter(13). This resulted in controller feedback gains of \(k_p = 4I_{2 \times 2}\) and \(k_d = 4I_{2 \times 2}\). The sample period for all of the simulations was set at \(T_s = 0.001\), corresponding to a sampling frequency of 1000 Hz, which is well above the bandwidth of the closed-loop system and is faster than the observer time constants. In all cases the robot parameter estimates were initialized to \(\hat{\theta}_1 = 1.5\) and \(\hat{\theta}_2 = 3\). The true parameter values were \(\theta_1 = 1\) and \(\theta_2 = 2\), as such the initial parameter estimates were 1.5 times the true parameter values.
Fig. 1 represents the tracking error for this method when implemented in simulation without any measurement noise added. The value for $\psi$ was set to $\psi = I$. The observer gains were based on (7) and set to $l_1 = 20$ and $l_2 = 300$, placing the poles for the observer at $s = -10 \pm 10\sqrt{2}j$.

The tracking error for this simulation reached maximum values of 0.2115 radians for link 1, and 0.1676 radians for link 2, with no noise in the system. The error on the parameter estimates can be seen in Fig. 2. The adaptation gain was set to $\Gamma = I$, values much larger than this resulted in divergence of the simulation. It can be seen that after 50 seconds of simulation, the parameter estimates are both quite close to their true values.

When noise was added to the system to simulate 12-bit quantization error, the maximum error did not change significantly, and while the error became much noisier, the results were comparable. This is illustrated in Fig. 3. In this case, the maximum tracking error was 0.2040 radians for link 1, and 0.1744 radians for link 2.
Figure 2: Parameter Estimation Error for $\theta_1$ (solid) and $\theta_2$ (dashed).

Figure 3: Error between desired trajectory and simulated trajectory for joint 1 (solid line) and joint 2 (dashed line) with position error equivalent to quantization by a 12-bit resolver.

3.2 Experimental Results

The control algorithm in this work has been implemented experimentally using the Carleton University Direct-Drive Robot. This robot is a $n = 2$ degree of freedom robot, consisting of four links in a parallelogram linkage, which operates in the horizontal plane. The dynamics of this robot are derived in [14] and are now presented.
These dynamics take the form of (1), but since the robot operates in the horizontal plane $G(q)$ is zero. In this case there are $p = 3$ robot parameters to be estimated, they are:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} I_0 + I_2 + m_2 l_4^2 + m_3 l_2^2 \\ I_1 + I_3 + m_2 l_1^2 + m_3 l_3^2 \\ m_2 l_4 + m_3 l_2 l_3 \end{bmatrix}$$

where $I_0$ through $I_3$ represent inertia terms, $l_1$ through $l_4$ are the lengths of each of the links, and $m_2$ and $m_3$ are the masses of links 2 and 3, respectively. The mass and coriolis matrices are given as:

$$M(q) = \begin{bmatrix} \theta_1 & \theta_3 \cos (q_1 - q_2) \\ \theta_3 \cos (q_1 - q_2) & \theta_2 \end{bmatrix}$$

$$C(q, \dot{q}) = \begin{bmatrix} 0 & \theta_3 \dot{q}_2 \sin (q_1 - q_2) \\ -\theta_3 \dot{q}_1 \sin (q_1 - q_2) & 0 \end{bmatrix}$$

The links of the robot are driven by two brushless DC servo motors. Each motor has a resolver for measuring position, and a tachometer for measuring velocity. The
resolver signals are converted by the amplifiers into 12-bit digital words. These 12-bit words are then sent to the computer through digital input/output ports of a digital interface card. The analog tachometer signals are sent to an analog/digital card inside the computer, which samples and digitizes the velocity measurements using 12 bits of resolution.

To drive the motors, the torques computed by the controller are scaled to a 12-bit number, and are then output to the digital/analog converter where they are converted to a $\pm 5$ V signal that is sent to the amplifiers. The torque constant for each motor was measured at 1.7 N-m/V, giving a torque saturation value of 8.5 N-m [16] (based on a $\pm 5$ V maximum control signal). All experiments were run using a sample period of $T_s = 0.002$, giving a sample frequency of 500 Hz.

When running the experiments, a trajectory consisting of a combination of sines and cosines at different frequencies is selected. It is based on a trajectory used in [15] and is given as:

\[
y_1(t) = \cos(2\omega t) - \cos(4\omega t) \quad (16)
\]
\[
y_2(t) = \left(\frac{\pi}{2}\right) - 1.9 + \sin(\omega t) + \sin(2\omega t) \quad (17)
\]

where $\omega$ was chosen to be 1.257 rad/s.

This signal was pre-filtered using a critically damped second-order linear filter with a bandwidth of $\omega_n = 2.0$ rad/s. A filter with this bandwidth was selected to ensure that the torques required to track the trajectory were kept within the torque saturation value of 8.5 N-m. The transfer function for this filter is given by (13). The filtered trajectories used for each link can be seen in Fig. 4.

In each of the experiments, the controller closed-loop dynamics were set to be critically damped and given a bandwidth $\omega_n = 6.0$ rad/s. This yielded feedback gains
of $k_p = 36I_{2 \times 2}$ and $k_d = 12I_{2 \times 2}$. For all experiments, the robot parameter estimates were initialized to $\hat{\theta}_1 = 0.2$, $\hat{\theta}_2 = 0.2$, and $\hat{\theta}_3 = 0.05$.

Filtering the original trajectory provided access to the desired velocity and desired acceleration signals corresponding to the filtered trajectory. As well, the range of the filtered positions matched well with the achievable range of motion for each link. This resulted in a path that the robot was able to follow without encountering points of singularity or obstacles such as its own aluminum structure.

In the implementation of all controllers, a Proportional-Derivative (PD) controller was used initially to move the robot links close to their origin (zero radians). This was done to minimize initial position error for the adaptive controllers.

The observer gains were set to $l_1 = 20$ and $l_2 = 500$, placing the observer poles at $s = -10 \pm 20j$. These poles are much faster than the dynamics of the closed loop system. It was determined experimentally that using underdamped observer poles allowed for smaller observer gains and yielded performance comparable to a set of critically damped observer poles placed farther into the negative real portion of the $s$-plane. An experiment was performed with observer poles at $s = -25$, giving observer gains of $l_1 = 50$ and $l_2 = 625$, and observer error was not significantly different than with the observer poles placed at $s = -10 \pm 20j$. For all experiments, the value of $\psi$ was set to $\psi = I$. Several values were used for the adaptation gain $\Gamma$ to compare performance.

Fig. 5 shows the tracking error for this method when implemented with $\Gamma = 0.1$. When run experimentally, values for $\Gamma$ any larger than this result in unacceptable performance of the robot during the experiment. The tracking error in this case reached maximum values of $-0.3810$ radians for link 1, and $-0.3583$ radians for link 2. After convergence of the parameter estimates, the error remained bounded between 0 radians and $-0.045$ radians for link 1. For link 2, the tracking error remained bounded
Figure 5: Position Error for joint 1 (solid line) and joint 2 (dotted line) after 180 seconds with $\Gamma = 0.1$ on the Direct-Drive Robot.

between 0.080 radians and $-0.066$ radians. The parameter estimates converged to steady-state values as seen in Fig. 6.

While the performance of this experiment is quite good, the choice of $\Gamma$ is close to the largest possible value before unacceptable performance is observed. As a result, another experiment was performed with an adaptation gain of $\Gamma = 0.05$, half of the previous value. The tracking error for this experiment is shown in Fig. 8. In this case, the tracking error reached maximum values of $-0.4234$ radians for link 1, and $-0.3282$ radians for link 2. After 200 seconds, the tracking error remained bounded between 0.005 radians and $-0.040$ radians for link 1. For link 2, the tracking error remained bounded between 0.080 radians and $-0.060$ radians. This is a positive result, since very similar performance is achieved over time with a lower adaptation gain. Using this lower adaptation gain keeps the system much further from the point of divergence of the experiment.
Figure 6: Parameter estimates over time for $\theta_1$ (solid), $\theta_2$ (dotted), and $\theta_3$ (dashed) with $\Gamma = 0.1$ on the Direct-Drive Robot.

Figure 7: Computed torques used to drive the robot links for link 1 (dashed) and link 2 (solid).

4. Stability

In this section we will theoretically demonstrate that the equilibrium point of the system is locally, asymptotically stable. The controller and adaptation law has already been completely designed, as such, unlike the prevailing literature, we do not seek a Lyapunov function and then choose parameters to satisfy a Lyapunov stability argu-
Figure 8: Position Error for joint 1 (solid line) and joint 2 (dotted line) after 180 seconds with $\Gamma = 0.05$ on the Direct-Drive Robot.

A direct method of local stability about the equilibrium point is demonstrated. The objective of this section is to formulate the overall system dynamics in terms of the error states. Therefore, define the error state as $\zeta = [e, \dot{e}, e_o, \dot{e}_o]^T$. Then write the error equation as:

$$\dot{\zeta} = f(\zeta) \tag{18}$$

Set $q_d = \dot{q}_d = \ddot{q}_d = 0$ and then compute:

$$A = \frac{\partial F(\zeta)}{\partial \zeta}|_{\zeta=0}, \tag{19}$$

If the matrix $A$ is a stability matrix, then the origin is asymptotically stable ([17], Theorem 3.7 p.127). Essentially, we are simply showing that if the robot is at rest at the origin and there is some small change in the state, the robot will return to the origin. It is admittedly a weak stability argument, however, the goal of the paper is to develop a simple, tractable design and control method, and not to develop a theoretical stability proof. As such we are in some sense sacrificing mathematical theoretical development for practical methods of design and implementation. The analysis starts by deriving the standard error equations for a robot manipulator. The
linearization law given by (10), is:

\[ T_L = Y(q, \dot{q}, v) \dot{\theta} = \hat{M}(q)v + \hat{C}(q, \dot{q}) \]  

(20)

(For ease of development we will consider that the matrix \( C \) contains the gravity terms.) The error state is derived by the standard method of substituting (20) into the equation of dynamics given by (1):

\[ M(q)\ddot{q} + C(q, \dot{q}) = \hat{M}(q)\ddot{q}_d + \hat{M}(q)k_pe + \hat{M}(q)k_d\dot{e} + \hat{C}(q, \dot{q}) \]  

(21)

Define the following error states; the trajectory error is given by, \( e = q_d - q \), the observer error is given by, \( e_o = q - \hat{q} \) and \( \dot{e} = \dot{q}_d - \dot{\hat{q}} \). Substituting for \( \dot{q} = \dot{\hat{q}} - \dot{e}_o \), we get, \( \dot{\hat{e}} = \dot{e} + \dot{e}_o \). There are two sources of errors that need to be accounted for; they are the parameter estimation error and the observer error. Similar to the literature (for example [1]) define the error in estimating the mass matrix as:

\[ \tilde{M}(q) = M(q) - \hat{M}(q) \]  

(22)

The error in the estimate of the coriolis/gravity matrix is affected by both the parameter error and the estimation error. The error in the coriolis/gravity matrix due to parameter error is given by:

\[ \tilde{C}_\theta(q, \dot{q}) = C(q, \dot{q}) - \hat{C}(q, \dot{q}) \]  

(23)

where \( \tilde{C}_\theta(q, \dot{q}) \), represents the estimation error of the matrix \( C(q, \dot{q}) \) due only to the parameter estimation error. Then the difference between the coriolis/gravity matrix based on the parameter estimates at the true velocity and the estimated velocity is given by:

\[ \tilde{C}(q, \dot{q}, \dot{\hat{q}}) = \tilde{C}(q, \dot{q}) - \hat{C}(q, \dot{q}) \]  

(24)

Rearranging (24) and substituting for \( \tilde{C}(q, \dot{q}) \) from (23) one gets:

\[ \hat{C}(q, \dot{q}) = C(q, \dot{q}) - \tilde{C}_\theta(q, \dot{q}) - \hat{C}(q, \dot{q}, \dot{\hat{q}}) \]  

(25)
The error in estimating the coriolis/gravity terms due to both parameter estimation error and observer velocity estimation error is given as:

\[ F(q, \dot{q}, \dot{\theta}, \theta, \dot{\theta}) = \tilde{C}_p(q, \dot{q}) + \tilde{C}(q, \dot{q}, \dot{\theta}) \]

\[ = C(q, \dot{q}) - \tilde{C}(q, \dot{q}) \quad (26) \]

One can now write the error equations by making the appropriate substitution of (22) and (26) into (21) and one gets:

\[ \dot{\hat{M}}(q)\dddot{q} + \hat{M}(q)k_pe + \hat{M}(q)k_d\dot{\hat{e}} + \hat{M}(q)k_d\dot{e}_o = \tilde{M}(q)\ddot{q} + \tilde{M}(q)\dot{q} + C(q, \dot{q}) - \tilde{C}(q, \dot{q}) \quad (27) \]

Substituting for \( \dot{\hat{e}} = \dot{\hat{e}} + \dot{\hat{e}}_o \) and writing \( \tilde{C}(q, \dot{q}) = C(q, \dot{q}) - F(\cdot) \) one gets:

\[ \dot{\hat{M}}(q)\ddot{q} - \hat{M}(q)\dddot{q} + \hat{M}(q)k_pe + \hat{M}(q)k_d\dot{\hat{e}} + \hat{M}(q)k_d\dot{e}_o = \tilde{M}(q)\ddot{q} + F(q, \dot{q}, \dot{\theta}, \theta) \quad (28) \]

Then the trajectory error equation is given as:

\[ \ddot{\hat{e}} + k_d\dot{\hat{e}} + k_pe = \tilde{M}^{-1}(q)(\tilde{M}(q)\ddot{q} + F(q, \dot{q}, \dot{\theta}, \theta)) - k_d\dot{e}_o \quad (29) \]

When one examines (29) one finds that the right hand side of the equation is not in terms of errors states. Therefore, one must convert all the parameters on the right hand side to error states. Taking the equation:

\[ \ddot{\hat{e}} + k_d\dot{\hat{e}} + k_pe = \tilde{M}^{-1}(q)(\tilde{M}(q)\ddot{q} + F(q, \dot{q}, \dot{\theta}, \theta)) \quad (30) \]

and writing, \( \dot{\hat{e}} = \dot{\hat{e}} + \dot{\hat{e}}_o \) and recalling that \( \ddot{q} = 0 \), so that \( \dddot{q} = \ddot{q}_d - \ddot{\hat{e}} = -\ddot{\hat{e}} \), one can then rewrite (30) as:

\[ \ddot{\hat{e}} + k_d\dot{\hat{e}} + k_d\dot{e}_o + k_pe = -\tilde{M}^{-1}(\zeta)\tilde{M}(\zeta)\ddot{\hat{e}} + \tilde{M}^{-1}(\zeta)F(\zeta) \quad (31) \]

Then, collecting terms and for convenience dropping the arguments, one gets the trajectory error equation as:

\[ \ddot{\hat{e}} + (I + \tilde{M}^{-1}\tilde{M})^{-1}k_d\dot{\hat{e}} + (I + \tilde{M}^{-1}\tilde{M})^{-1}k_pe = (I + \tilde{M}^{-1}\tilde{M})^{-1}F - (I + \tilde{M}^{-1}\tilde{M})^{-1}\dot{e}_o \quad (32) \]
Now one formulates the error dynamics for the observer. Rewriting the equation for the manipulator dynamics by substituting the control law (10) into the dynamics (1), one gets:

$$M(q)\ddot{q} + C(q, \dot{q}) = \dot{M}(q)v + \dot{C}(q, \dot{q})$$

(33)

Then write:

$$\ddot{\hat{q}} = M^{-1}(q)(\dot{M}(q)v - F(\cdot))$$

(34)

Similarly, one can write the observer error equation from (7) and (34) as:

$$\begin{bmatrix} \dot{\hat{q}} \\ \ddot{\hat{q}} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v + \begin{bmatrix} 0 \\ I \end{bmatrix} M^{-1}(q)(\dot{M}(q)v - F(\cdot))$$

(35)

Which can be written in the form:

$$\begin{bmatrix} \dot{\hat{e}}_o \\ \ddot{\hat{e}}_o \end{bmatrix} = (A_o - LH) \begin{bmatrix} e_o \\ \dot{e}_o \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} M^{-1}(q)(\dot{M}(q)v - F(\cdot))$$

(36)

where:

$$A_o = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$$

(37)

and $H = [I, 0]$.

One can similarly write out the observer error dynamics, from (36), in terms of the error states as:

$$\begin{bmatrix} \dot{\hat{e}}_o \\ \ddot{\hat{e}}_o \end{bmatrix} = (A_o - LH) \begin{bmatrix} e_o \\ \dot{e}_o \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} M^{-1}(\zeta)(\dot{M}(\zeta)v - F(\cdot))$$

(38)

Substituting for $v = k_pe + k_d\dot{e} + \ddot{q}_d = k_p + k_d\dot{e}$ and $\dot{\hat{e}} = \dot{\hat{e}} + \dot{\hat{e}}_o$, and dropping the
arguments for convenience, one gets the observer error equation as:

\[
\begin{bmatrix}
\dot{e}_o \\
\ddot{e}_o
\end{bmatrix} = (A_o - LH) \begin{bmatrix}
e_o \\
\dot{e}_o
\end{bmatrix} + \begin{bmatrix} 0 \\
I
\end{bmatrix} M^{-1} \ddot{M}k_p e + \begin{bmatrix} 0 \\
I
\end{bmatrix} M^{-1} \ddot{M}k_d \dot{e}_o + \begin{bmatrix} 0 \\
I
\end{bmatrix} M^{-1} F
\]

(39)

The overall error state for the trajectory error and the observer error is a combined equation of (32) and (39).

Now linearize the trajectory errors (32) and (39) about the origin. Clearly, writing out \( \ddot{e} = f(\zeta) \) and then computing \( \partial f(\zeta)/\partial \zeta \) would be an onerous exercise, instead we simply write out (32) and (39) and we neglect second order terms and higher, this results in the correct linearization. As an example and as a case study we will examine the stability of the robot manipulator described in section 3.

4.1 Stability Case Study of Simulated Manipulator

This section will investigate the linearization of the manipulator simulated in section 3. Then show that the origin is locally asymptotically stable. The dynamics take the form of (1) but the robot operates in the horizontal plane, and as such \( G(q) \) is zero and the mass and coriolis matrices are given by (11). Reiterating the equations of dynamics for convenience:

\[
\theta = \begin{bmatrix}
\theta_1 \\
\theta_2
\end{bmatrix} = \begin{bmatrix}
m_1 l^2 \\
m_2 l^2
\end{bmatrix}
\]

The mass and coriolis matrices are given as:

\[
M(q) = \begin{bmatrix}
\theta_1 + 2\theta_2 + 2\theta_2 \cos q_2 & \theta_2 + \theta_2 \cos q_2 \\
\theta_2 + \theta_2 \cos q_2 & \theta_2
\end{bmatrix}
\]

(40)

\[
C(q, \dot{q}) = \begin{bmatrix}
-2\theta_2 \dot{q}_2 \sin q_2 & -\theta_2 \dot{q}_2 \sin q_2 \\
\theta_2 \dot{q}_1 \sin q_2 & 0
\end{bmatrix}
\]

(41)
where, $cq_i = \cos q_i$ and $sq_i = \sin q_i$ and the estimation error of the inertia matrix is given by:

$$
\tilde{M}(q) = \begin{bmatrix}
\hat{\theta}_1 + 2\tilde{\theta}_2 + 2\tilde{\theta}_2cq_{d2}ce_{d2} - 2\tilde{\theta}_2sq_{d2}se_{d2} & \tilde{\theta}_2 + \tilde{\theta}_2cq_{d2}ce_{d2} - \tilde{\theta}_2sq_{d2}se_{d2} \\
\tilde{\theta}_2 + \tilde{\theta}_2cq_{d2}ce_{d2} - \tilde{\theta}_2sq_{d2}se_{d2} & \tilde{\theta}_2
\end{bmatrix}
$$

(42)

Setting the desired trajectory to zero, $q_d = \dot{q}_d = \ddot{q}_d = 0$ and linearizing about small trajectory errors by setting $\sin e_i = e_i$ and $\cos e_i = 1$ one gets:

$$
\delta \tilde{M}(\zeta)_{|\zeta=0} = \begin{bmatrix}
\hat{\theta}_1 + 4\tilde{\theta}_2 & 2\tilde{\theta}_2 \\
2\tilde{\theta}_2 & \tilde{\theta}_2
\end{bmatrix}
$$

(43)

Similarly, we will compute $\hat{M}(\zeta)$ as follows:

$$
\hat{M}(q) = \begin{bmatrix}
\hat{\theta}_1 + 2\tilde{\theta}_2(1 + cq_{q}) & \hat{\theta}(1 + cq_q) \\
\hat{\theta}(1 + cq_{q}) & \tilde{\theta}_2
\end{bmatrix}
$$

(44)

Writing $\hat{\theta}_i = \theta_i - \tilde{\theta}_i$ and linearizing about $q_d = \dot{q}_d = \ddot{q}_d = 0$, and assuming small trajectory errors one gets:

$$
\delta \hat{M}(\zeta)_{|\zeta=0} = \begin{bmatrix}
\theta_1 - \hat{\theta}_1 + 4\tilde{\theta}_2 - 4\dot{\theta}_2 & 2\theta_2 - 2\tilde{\theta}_2 \\
2\theta_2 - 2\tilde{\theta}_2 & \theta_2 - \tilde{\theta}_2
\end{bmatrix}
$$

(45)

One can then compute:

$$
\delta \hat{M}^{-1}(\zeta)_{|\zeta=0} = \frac{1}{\theta_1\theta_2 - \hat{\theta}_1\theta_2 - \tilde{\theta}_2\theta_1} \begin{bmatrix}
\theta_2 - \tilde{\theta}_2 & 2\tilde{\theta}_2 - 2\theta_2 \\
2\tilde{\theta}_2 - 2\theta_2 & \theta_1 - \tilde{\theta}_1 + 4\theta_2 - 4\tilde{\theta}_2
\end{bmatrix}
$$

(46)

and:

$$
\delta \hat{M}^{-1}(\zeta)\delta \hat{M}(\zeta)_{|\zeta=0} = \begin{bmatrix}
\frac{\dot{\theta}_1\theta_2}{\theta_1\theta_2 - \theta_1\theta_2 - \theta_2\theta_1} & 0 \\
0 & \frac{\theta_1\dot{\theta}_2}{\theta_1\theta_2 - \theta_1\theta_2 - \theta_2\theta_1}
\end{bmatrix}
$$

(47)

$$
\approx \begin{bmatrix}
\frac{\dot{\theta}_1}{\theta_1} & 0 \\
0 & \frac{\dot{\theta}_2}{\theta_2}
\end{bmatrix}
$$
Then:
\[ (I + \delta \hat{M}^{-1}(\zeta)\delta \hat{M}(\zeta)) |_{\zeta=0} \approx I \]  
(48)
as such:
\[ (I + \delta \hat{M}^{-1}(\zeta)\delta \hat{M}(\zeta))^{-1} |_{\zeta=0} \approx I \]  
(49)
The term \( F(q, \dot{q}, \dot{q}, \theta, \dot{\theta}) \), which represents the error in the estimation of the coriolis terms due to parameter estimation error and observer error becomes:
\[
F(q, \dot{q}, \dot{q}, \theta, \dot{\theta}) = \begin{bmatrix}
-2\hat{\theta}_2 \dot{q}_1 \dot{q}_2 s q_2 - \hat{\theta}_2 \hat{q}_2^2 s q_2 \\
\hat{\theta}_2 \hat{q}_1^2 s q_2
\end{bmatrix} + \begin{bmatrix}
-2\hat{\theta}_2 (\dot{q}_1 \dot{q}_2 - \dot{\theta}_1 \hat{q}_2) s q_2 - \hat{\theta}_2 (\hat{q}_2^2 - \hat{\theta}_2^2) s q_2 \\
\hat{\theta}_2 (\hat{q}_1^2 - \hat{\theta}_1^2) s q_2
\end{bmatrix}
\]  
(50)
When one formulates \( F(\cdot) \) as \( F(\zeta) \), writing \( F(\zeta) \) in terms of the error states, we find that every term in \( F(\zeta) \) is of second order or higher. As such, linearizing \( F(\zeta) \) about \( \zeta = 0 \) is \( \delta F(\zeta)|_{\zeta=0} = 0 \). Then the trajectory error (32) reduces to:
\[
\ddot{e} + k_d \dot{e} + k_p e = -k_d \dot{e}_o
\]  
(51)
A similar approach can be used to determine the linearized observer error dynamics, where the observer error dynamics are given by (39). Once again, writing out \( \delta M^{-1}(\zeta)\delta \hat{M}(\zeta) \) in terms of the error and linearizing by neglecting second order terms and higher one gets:
\[
\delta M^{-1}(\zeta)\delta \hat{M}(\zeta) = \begin{bmatrix}
\frac{\dot{\theta}_1}{\theta_1} & 0 \\
-2\frac{\theta_1 \theta_2 + 2\theta_2 \theta_1}{\theta_1 \theta_2} & \frac{\hat{\theta}_2}{\theta_2}
\end{bmatrix}
\]  
(52)
Then, when one linearizes (39), one is only left with:
\[
\begin{bmatrix}
\dot{e}_o \\
\ddot{e}_o
\end{bmatrix} = \left( A_o - LH \right) \begin{bmatrix}
e_o \\
\dot{e}_o
\end{bmatrix}
\]  
(53)
Now examine the parameter estimation algorithm. The parameter estimation algorithm is:
\[
\dot{\hat{\theta}} = \hat{\theta} - \hat{\theta} = -\Gamma Y_L^T(q, \dot{q}, \dot{q}) \hat{M}^{-1}(q) B^T \hat{P} \dot{e}_1
\]  
(54)
where the filtered error is $\hat{e}_1 = \hat{\dot{e}} + \psi e$. The parameter estimation error is then given as:

$$\dot{\hat{\theta}} = -\Gamma Y^T_L(q, \dot{q}, \ddot{q})M^{-1}(\zeta)\hat{\dot{e}} - \Gamma Y^T_L(q, \dot{q}, \ddot{q})M^{-1}(\zeta)e$$

(55)

The terms in the matrix $Y^T_L(q, \dot{q}, \ddot{q})$ when linearized about the origin of the error states are all of second order or higher. As such, all the terms on the right hand side of (55) are of second order or higher. Therefore the parameter adaptation law is neglected when evaluating the stability of the system about the origin. The dynamics associated with the parameter adaptation law is very slow in comparison to the dynamics of the trajectory errors or the observer errors. Furthermore, the parameter estimation errors have no effect on the linearized trajectory and observer errors. The parameter error states are effectively decoupled from the observer and trajectory error states when linearized about the origin.

Then the linearized error state equations for the simulated system in section 3 becomes:

$$\begin{bmatrix}
\dot{e}_1 \\
\ddot{e}_1 \\
\dot{e}_2 \\
\ddot{e}_2 \\
\dot{e}_{1o} \\
\ddot{e}_{1o} \\
\dot{e}_{2o} \\
\ddot{e}_{2o}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-k_{p1} & -k_{d1} & 0 & 0 & 0 & -k_{d1} & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -k_{p2} & -k_{d2} & 0 & 0 & 0 & -k_{d2} \\
0 & 0 & 0 & 0 & -l_{11} & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -l_{12} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -l_{22} & 0
\end{bmatrix}
\begin{bmatrix}
e_1 \\
\dot{e}_1 \\
e_2 \\
\dot{e}_2 \\
e_{1o} \\
\dot{e}_{1o} \\
e_{2o} \\
\dot{e}_{2o}
\end{bmatrix}
$$

(56)

Now substituting in the values used in the simulation we have, $k_{p1} = k_{p2} = 4, k_{d1} = k_{d2} = 4, l_{11} = l_{21} = 20, l_{12} = l_{22} = 300$, then the eigenvalues of the matrix in (56) are $-2, -2, -2, -2, -10 \pm 14.14j, -10 \pm 14.14j$, which are the exact design values and the origin is asymptotically stable.
This stability analysis has analyzed the particular case simulated in section 3, however the results are as expected for a linearized system about the origin. The theoretical issue regarding the region of stability is an interesting theoretical question, however it is beyond the scope of this paper which focusses on the design and implementation issues. Furthermore, a general result on the region of stability would have limited practical and design significance due to the complexity and theoretical results that are almost always so overly conservative as to be irrelevant as design tools. The simulation and experimental results indicate a wide region of practical stability.

5 Conclusion

A method of nonlinear output feedback adaptive control for robot manipulators has been proposed. The proposed method is both conceptually easy to design and implement. The simulation results indicate that the algorithm works according to design specifications. Experimental results further validate the usefulness of the proposed method. The proposed method is based on ease of design and implementation and not on a stability argument, as such a separate stability argument is done to show the stability of the origin on the case simulated in the paper. The stability argument shows that the stability of the origin follows the design specifications.

References


