Günter Niemeyer
Jean-Jacques E. Slotine
Nonlinear Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

Performance in Adaptive Manipulator Control

Abstract

Effective adaptive controller designs potentially combine high speed and high precision in robot manipulation and furthermore can considerably simplify high-level programming by providing consistent performance in the face of large variations in loads or tasks. A simple, globally tracking-convergent, direct adaptive manipulator controller has previously been developed and experimentally demonstrated. In this article, we further explore the performance issues linked to a computationally effective implementation. Specifically, we develop a recursive implementation applicable to both open and closed kinematic chains, as well as rules for obtaining minimal parametrizations. We also discuss implementations of the algorithm directly in Cartesian space, the exploitation of kinematic redundancies, and applications to adaptive compliant motion control. These developments are illustrated experimentally on a four-degrees-of-freedom articulated robot arm and suggest a wide range of application well beyond adaptation to grasped loads.

1. Introduction

Adaptive control of linear time-invariant, single-input, single-output systems has been extensively studied, and a number of globally convergent controllers have been derived. Extensions of the results to nonlinear or multi-input systems have rarely been achieved. However, research in adaptive robot control has been very active in recent years (Slotine and Li 1986; Craig et al. 1986; Middleton and Goodwin 1986; Hsu et al. 1987; Sadegh and Horowitz 1987; Bayard and Wen 1987; Koditschek 1987; Slotine and Li 1987a–e; Li and Slotine 1988; Walker 1988) and has led to the remarkable result that global convergence properties similar to those of single-input linear systems can indeed be obtained for robot manipulators, which represent an important class of nonlinear, time-varying, multi-input, multi-output dynamic systems. Slotine and Li (1986) derived a simple, globally tracking-convergent, direct adaptive manipulator controller, which was demonstrated experimentally in subsequent work (Slotine and Li 1987a). In this article, we study an efficient computational implementation of the adaptive controller applicable to both open and closed kinematic chains. To further improve efficiency, the use of proper minimal parametrizations is also discussed. We then extend the implementation to Cartesian controllers, exploiting possible kinematic redundancies, and finally study applications to constrained motion. The developments are illustrated experimentally on a four-degree-of-freedom articulated robot arm.

Following a brief review of the direct adaptive controller of Slotine and Li (1986), section 2 studies issues of computational efficiency, such as recursive implementation and minimal parametrizations. Section 3 develops an implementation directly in Cartesian space, applicable to both nonredundant and redundant robots, and studies applications to adaptive compliant motion control. Section 4 presents experimental results. Section 5 offers brief concluding remarks.

The International Journal of Robotics Research,
Vol. 10, No. 2, April 1991,

2. Implementation of Adaptive Manipulator Controllers

In this section we discuss the computationally efficient recursive implementation of adaptive manipulator controllers. Section 2.1 briefly reviews the direct adaptive controller of Slotine and Li (1986, 1987d). The recursive implementation is then detailed in section 2.2, and section 2.3 discusses minimal parametrizations.

2.1. The Adaptive Controller

The following briefly summarizes the direct adaptive manipulator control algorithm of Slotine and Li (1986, 1987d). In the absence of friction and other disturbances, the dynamics of a rigid manipulator (with the load considered as part of the last link) can be written as

$$H(q)\ddot{q} + C(q,\dot{q})\,\dot{q} + G(q) = \tau \qquad (1)$$

where q and τ are $n \times 1$ vectors of joint displacements and applied joint torques (or forces), respectively. $H(q)$ is the $n \times n$ symmetric positive-definite (s.p.d.) manipulator inertia matrix, $C(q, \dot{q})\, \dot{q}$ is the $n \times 1$ vector of centripetal and Coriolis torques, and $G(q)$ is the $n \times 1$ vector of gravitational torques. In addition, a friction model can be added to the dynamics in equation (1), using, e.g., $D_s \operatorname{sgn}(\dot{q})$ and $D_v \dot{q}$ as the $n \times 1$ vectors of static (Coulomb) and viscous friction torques, where the sign operator $\operatorname{sgn}(.)$ is performed independently in each component and the matrices D_s *and* D_v are positive definite (or semidefinite). The adaptive controller design problem is as follows: given the desired trajectory $q_d(t)$, with some or all of the dynamic parameters being unknown, and with the joint positions and velocities measured, derive a control law for the actuator torques and an adaptation law for the unknown parameters such that the manipulator joint position $q(t)$ closely tracks the desired trajectory $q_d(t)$.

Define a "reference" trajectory and an associated tracking error measure as

$$\dot{q}_r = \dot{q}_d - \Lambda \tilde{q} \tag{2}$$

$$s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \Lambda \tilde{q} \tag{3}$$

where $\tilde{q} = q - q_d$ and Λ is a s.p.d. matrix (or, more generally, a matrix whose eigenvalues are strictly in the right-half complex plane). Also, define $\hat{a}(t)$ as the current estimate of the constant vector a of the manipulator's dynamic parameters, with $\tilde{a}(t) = \hat{a}(t) - a$. Accordingly, define the known matrix $Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)$ by

$$\begin{aligned} Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{a} = \hat{H}(q)\ddot{q}_r + \hat{C}(q, \dot{q})\dot{q}_r \\ + \hat{G}(q) + \hat{D}_s \operatorname{sgn}(\dot{q}_r) + \hat{D}_v \dot{q}_r. \end{aligned} \tag{4}$$

Then, using the Lyapunov-like function

$$V(t) = \frac{1}{2}\,[s^T H s + \tilde{a}^T P^{-1} \tilde{a}] \tag{5}$$

and exploiting fundamental physical properties of the system, such as conservation of energy and the positive-definiteness of the inertia matrix H, it can be shown that the control and adaptation laws

$$\tau = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{a} - K_D s \tag{6}$$

$$\dot{\hat{a}} = -P Y^T s, \tag{7}$$

where K_D and P are s.p.d. gain matrices, yield $\dot{V}(t) \leq -s^T (\mathrm{K}_D + \mathrm{D}_v)\, s \leq 0$. Using simple functional analysis arguments, this can be shown to imply that $s \to 0$ as $t \to \infty$, which in turn implies that $\tilde{q}$ and $\dot{\tilde{q}}$ both tend to 0. Therefore the algorithm is guaranteed global tracking convergence independently of the initial parameter estimates. Note that the control law is composed of a "P.D." term, $-K_D s$, and a particular type of "feedforward," $Y\hat{a}$.

The friction model incorporated in the above control and adaptation laws consists of Coulomb and viscous friction. Yet, much in the nature of this controller, we do not attempt to cancel either type of friction, but rather compensate for the frictional torques corresponding to the reference trajectory. In addition to helping the convergence by adding negative terms to $\dot{V}(t)$, this avoids the use of velocity measurements for friction compensation and thus alleviates problems of inaccurate friction models at zero velocity. Consequently, we do not in principle need a smoothing approximation for Coulomb friction at small velocities, although in practice it may still be desirable to avoid abrupt switchings of the torque command.

It is also interesting to note that dealing with large, possibly unknown loads generally reduces the lowest structural frequencies considerably and severely limits the available control bandwidth. This further contributes to the sensitivity of nonadaptive controllers to the presence of loads, highlighting the need for developments such as the above.

2.2. *Recursive Implementation*

A well-known obstacle for the implementation of multi-degree-of-freedom manipulator controllers using full dynamics compensation is their computational complexity. The physically motivated Newton-Euler algorithm can significantly reduce this complexity by recursively computing the natural dynamics. This provides an efficient implementation for "computed-torque" controllers. However, adaptive sliding controllers, as described previously, introduce an additional reference velocity and thus use modified dynamic equations, preventing the application of the Newton-Euler algorithm. Walker (1988) suggests a solution to this problem by applying the basic ideas of Slotine and Li (1986) directly to the dynamic equations in recursive Newton-Euler form. The resulting algorithm is recursive, hence computationally efficient, and has essentially the same convergence properties as the original adaptive sliding controller. However, differences exist. In particular, there is no joint space representation of the calculated torques, and therefore no simple "closed form" of the resulting multi-input controller.

In this section we present an exact recursive implementation of the adaptive sliding controller of Slotine and Li (1986). Though the algorithm is derived independently of the Newton-Euler algorithm, it takes a similar form, generalized to accommodate the reference velocity. Although it may be applied to closed chains equally well, it is detailed here in the context of open chains, allowing a simpler notation. The standard Denavit-Hartenberg convention is used (Asada and Slotine 1986), thus numbering the links from 0 (base) to n (tip), with joint i connecting link $i-1$ to link i.

As in the computational approach of Walker (1988), a key simplifying aspect of the derivation is the use of the spatial vector notation of Featherstone (1987), which allows translational and rotational quantities to be combined in a single six-dimensional vector. This notation, which is summarized briefly in appendix A, considerably reduces the complexity and number of equations. With $\boldsymbol{d}$ being the spatial direction of a joint axis, the spatial velocity, reference velocity, and reference acceleration of a link are defined as follows:

$$\boldsymbol{v}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \dot{q}_l \qquad . \tag{8}$$

$$\boldsymbol{w}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \dot{q}_{r_l} \tag{9}$$

$$\dot{\boldsymbol{w}}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \ddot{q}_{r_l} + \boldsymbol{v}_l \times \boldsymbol{d}_l \dot{q}_{r_l} \tag{10}$$

Also, for each link, we define 10 placement matrices that determine the location of all 10 inertial parameters of the link in its spatial inertia matrix. The 10 parameters represent six inertias (recall that the 3×3-link inertia matrix is symmetric), three parameters for the location of the center of mass (each coordinate of the center of mass is multiplied by the mass), and the mass itself. The placement matrices thus consist only of zeros and ones and are extremely sparse.

$$\boldsymbol{I}_k = \sum_{i=1}^{10} \boldsymbol{R}_i a_k^i \tag{11}$$

with $a_k^i \in \{J_{xx}, J_{yy}, J_{zz}, J_{xy}, J_{xz}, J_{yz}, p_x, p_y, p_z, m\}$ and $p_i = mr_i$.

In addition to the above definitions, the derivation makes use of two observations. First, the matrix $\boldsymbol{C}$ in the manipulator dynamics (equation [1]) is not unique. Examining the Lagrangian derivation of the equations of motion, as in Slotine and Li (1987d), it is chosen to obey $\dot{\boldsymbol{H}} = \boldsymbol{C} + \boldsymbol{C}^T$, that is

$$c_{ki} = \frac{1}{2} \sum_{j=1}^{n} \left(\frac{\partial h_{ki}}{\partial q_j} + \frac{\partial h_{kj}}{\partial q_i} - \frac{\partial h_{ij}}{\partial q_k} \right) \dot{q}_j . \tag{12}$$

Second, the kinetic energy E_{kin} must be independent of the choice of coordinates, establishing an implicit relation between joint and spatial inertia matrices:

$$E_{kin} = \frac{1}{2} \dot{\boldsymbol{q}}^T \boldsymbol{H}(\boldsymbol{q}) \dot{\boldsymbol{q}} = \sum_{l=1}^{n} \frac{1}{2} \boldsymbol{v}_l^T \boldsymbol{I}_l \boldsymbol{v}_l . \tag{13}$$

Substituting equation (8) in the above then results in an explicit expression for the joint inertia matrix. This result is in turn substituted in equation (12) to produce an explicit expression for the Coriolis matrix. We can then express all components of the manipulator equation of motion and thus the control and adaptation laws in spatial quantities:

$$h_{ij} = \boldsymbol{d}_i^T \sum_{k=\max(i,j)}^{n} \boldsymbol{I}_k \boldsymbol{d}_j \tag{14}$$

$$c_{ij} = \boldsymbol{d}_i^T \sum_{k=\max(i,j)}^{n} \boldsymbol{I}_k \boldsymbol{v}_j \times \boldsymbol{d}_j + \frac{1}{2} \boldsymbol{v}_k \times \boldsymbol{I}_k \boldsymbol{d}_j + \frac{1}{2} \boldsymbol{d}_j \times \boldsymbol{I}_k \boldsymbol{v}_k + \frac{1}{2} \boldsymbol{I}_k \boldsymbol{d}_j \times \boldsymbol{v}_k \tag{15}$$

$$G_i = -\boldsymbol{d}_i^T \sum_{k=i}^{n} \boldsymbol{I}_k \boldsymbol{g}_0 \tag{16}$$

where $\boldsymbol{g}_0$ is the gravity vector.

As a final step these expressions are substituted in the control and adaptation laws. The actual algorithm is then obtained by performing all summations recursively. Like the Newton-Euler algorithm, it is divided into an upward and a downward section. Velocities and accelerations are determined during the first part, while the second computes forces and torques and performs the adaptation. Table 1 summarizes the complete algorithm (for diagonal gain matrices), with all vectors expressed in the spatial notation.

Several practical considerations can greatly influence the actual efficiency of the algorithm:

1. Using the Denavit-Hartenberg conventions, the following choice of reference frames minimizes the number of frame transformations: velocities, accelerations, inertias, and local force components are expressed in their own frame (i.e., in the frame attached to their link), while joint axes and

Table 1. Equations for the Recursive Implementation of the Adaptive Controller

Initialize:	$\dot{\boldsymbol{w}}_0 = -g_0$
Upward:	$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} + \boldsymbol{d}_k \dot{q}_k$
	$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \boldsymbol{d}_k \dot{q}_{r_k}$
	$\dot{\boldsymbol{w}}_k = \dot{\boldsymbol{w}}_{k-1} + \boldsymbol{d}_k \ddot{q}_{r_k} + \boldsymbol{v}_{k-1} \times \boldsymbol{d}_k \dot{q}_{r_k}$
	$\boldsymbol{e}_k = \boldsymbol{v}_k - \boldsymbol{w}_k$
Downward:	$\boldsymbol{f}_k^i = \frac{1}{2} \boldsymbol{v}_k \times \boldsymbol{R}_i \boldsymbol{w}_k + \frac{1}{2} \boldsymbol{w}_k \times \boldsymbol{R}_i \boldsymbol{v}_k + \frac{1}{2} \boldsymbol{R}_i \boldsymbol{w}_k \times \boldsymbol{v}_k + \boldsymbol{R}_i \dot{\boldsymbol{w}}_k$
	$\boldsymbol{F}_k = \boldsymbol{F}_{k+1} + \sum_{i-1}^{10} \boldsymbol{f}_k^i a_k^i$
	$\tau_k = \boldsymbol{d}_k^T \boldsymbol{F}_k + \hat{D}_{s_k} \operatorname{sgn}(\dot{q}_{r_k}) + \hat{D}_{v_k} \dot{q}_{r_k} - K_{D_k} s_k$
	$\dot{\hat{a}}_k^i = -P_k^i \boldsymbol{e}_k^T \boldsymbol{f}_k^i$
	$\dot{\hat{D}}_{s_k} = -P_k^s s_k \operatorname{sgn}(\dot{q}_{r_k})$
	$\dot{\hat{D}}_{v_k} = -P_k^v s_k \dot{q}_{r_k}$

summed forces are expressed with respect to the frame beneath them.

2. It is recommended to "customize" the algorithm (similarly to Khosla and Kanade 1985), so as to avoid multiplications by or additions with zero. These occur mainly in the local force component calculations (because of the sparse placement matrices), which should be evaluated analytically. Also, several velocity components may be restricted to zero (especially close to the base) and can thus be eliminated.
3. Several parameters may be redundant, and therefore the algorithm can be optimized by using a minimal parameter set, the choice of which we will detail in section 2.3.
4. The gravitational forces are implemented as a vertical acceleration in gravity-free space, thus eliminating their explicit calculation.

If the algorithm is to be implemented on closed chains, imaginary cuts are placed at different joints, thus creating an open but branched tree type structure. The constraint equations are then used to determine all velocities and again to calculate torques for the motorized joints, while forces are simply summed at branch points. The approach of Walker (1988) for structuring closed chains and incorporating kinematic constraints can be used straightforwardly.

2.3. Minimal Parameterization

As mentioned previously, the mass properties of an arbitrary rigid body can be described by 10 parameters, all of which are included in the spatial inertia matrix. If, however, we connect two rigid bodies through a joint and thus restrict their motion with respect to each other, not all 20 parameters are needed to describe the mass properties of the connected system (An et al. 1985; Khosla and Kanade 1985). Mayeda et al. (1988) studied this redundancy analytically for robots with *rotational* joints whose axes are restricted to be either perpendicular or parallel. They conclude that the minimum number of parameters necessary is $7N - 4B$, where N is the number of links and B the number of parallel joints located at the base (which is at least one). If the first joint is vertical, then another two parameters can be removed.

Using the spatial notation, it is possible to simply analyze parameter reductions involved with *both* rotational and translational joints, regardless of their intersection angle, as we now show. It will become clear what combinations of parameters affect the dynamic equations and how to eliminate redundant sets of parameters in the implementation.

Let us investigate the combination of two successive links $k-1$ and k. We are interested in finding which variations in the links' parameters have no influence on the dynamic equations and can thus be used to eliminate individual parameters. Examining the dynamic equations, two conditions quickly emerge for a parameter variation to be unobservable:

$$f_{k-1} + f_k = constant, \qquad d_k^T f_k = constant \quad (17)$$

where

$$f_k = \tfrac{1}{2}\boldsymbol{v}_k \times \boldsymbol{I}_k \boldsymbol{w}_k + \tfrac{1}{2}\boldsymbol{w}_k \times \boldsymbol{I}_k \boldsymbol{v}_k + \tfrac{1}{2}\boldsymbol{I}_k \boldsymbol{w}_k \times \boldsymbol{v}_k + \boldsymbol{I}_k \dot{\boldsymbol{w}}_k.$$

To further study these conditions, we introduce an arbitrary inertia variation $\boldsymbol{\delta I}$, which is added to link k and subtracted from link $k-1$, representing the unobservable parameter variation. Noticing that the forces are linear in terms of the inertia matrix and using the velocity and acceleration relation between the two links, we can rewrite these conditions in the following form, which must be satisfied for all allowable $\boldsymbol{\delta I}$.

$$\boldsymbol{u} \times \boldsymbol{\delta I}\, \boldsymbol{d}_k + \boldsymbol{d}_k \times \boldsymbol{\delta I u} + \boldsymbol{\delta I u} \times \boldsymbol{d}_k = \boldsymbol{0}, \qquad \boldsymbol{\delta I}\, \boldsymbol{d}_k = \boldsymbol{0} \quad (18)$$

where $\boldsymbol{u}$ is a generally unrestricted velocity of link $k-1$. We analyze these conditions with respect to coordinate frame $k-1$, permitting the use of a constant $\boldsymbol{d}_k$, dependent only on the type of joint. Thus, given $\boldsymbol{d}_k$, the equations are quickly evaluated for sparse $\boldsymbol{\delta I}$ (e.g., for the placement matrices), and several parameters (or combinations thereof) emerge, which satisfy all conditions.

As a result of the above we have found parameters (combinations) that can be varied freely on link $k-1$ if the appropriate variation is performed on link k. Therefore it is possible to transfer parameters from link $k-1$ to link k, or vice versa, and also to set them to zero on one or the other. Specifically, for rotational or translational joints, the following parameters (combinations) can be varied (and thus set to zero) on link $k-1$:

for rotational joints:	$J_{xx} - J_{yy}, p_z, m$
for translational joints:	$J_{xx}, J_{yy}, J_{zz}, J_{xy}, J_{xz}, J_{yz}$.

To perform the appropriate variation on link k, a reference frame transformation from $k-1$ to k is necessary. Appendix B summarizes this procedure for rotational and translational joints.

Note that for links close to the base, the above conditions (18) are relaxed, because the velocity vectors are restricted. Thus more parameters can be transferred on lower links, and consequently more parameters can be eliminated. For example, for a base rotating about the vertical axis, only the corresponding rotational inertia will affect the dynamics. It is also interesting to note that these formal parameter reductions and transfers often have physical interpretations. In a rotational joint, for instance, a thin rod along the rotational axis (i.e., having no inertia in that direction) can belong to either of the attached links.

3. Cartesian Control

In the previous section we discussed the efficient implementation of a joint space adaptive controller. Many tasks, however, are best described in terms of the end-effector motion or behavior. Thus it is desirable for a controller to directly handle Cartesian (rather than joint) input and output variables (see Luh et al. 1980 and Khatib 1980, 1983, in the nonadaptive case). This section discusses the problems of converting and extending the above adaptive algorithms to handle Cartesian data in real time. The development represents a computational version of Slotine and Li (1987c) and also extends the algorithm to redundant manipulators. Section 3.1 takes a closer look at the inverse kinematics involved, including a discussion of kinematic redundancy. Section 3.2 describes the actual implementation, and section 3.3 discusses extension to compliant motion control.

3.1. Inverse Kinematics

Examining equations (6) and (7), the joint space adaptive controller can be interpreted as using only reference velocity and acceleration as input signals and guaranteeing convergence only to these. It is then the definition of the reference velocity in equation (2) that guarantees the actual convergence to the desired trajectory. Because in the case of Cartesian motion a desired joint trajectory is not given, we now proceed by redefining the joint reference velocity in terms of its Cartesian counterpart. More precisely, with Cartesian reference velocity and acceleration defined as

$$\dot{\boldsymbol{x}}_r = \dot{\boldsymbol{x}}_d - \Lambda\tilde{\boldsymbol{x}} \quad \text{and} \quad \ddot{\boldsymbol{x}}_r = \ddot{\boldsymbol{x}}_d - \Lambda\dot{\tilde{\boldsymbol{x}}} \tag{19}$$

they are related to the joint reference velocity and acceleration by the Jacobian J:

$$\dot{\boldsymbol{x}}_r = J\dot{\boldsymbol{q}}_r \tag{20}$$

$$\ddot{\boldsymbol{x}}_r = J\ddot{\boldsymbol{q}}_r + \dot{J}\dot{\boldsymbol{q}}_r \tag{21}$$

To compute the joint quantities, the above equations are simply inverted. This approach will guarantee convergence of Cartesian motion to the desired trajectory, similarly to the joint space case, as long as singularities are avoided. Thus the actual inverse kinematic solution for desired joint position is not computed directly but rather determined by the dynamics of the system. Notice, however, that it can be retrieved for other purposes (e.g., obstacle avoidance) by exploiting the filter structure of the reference velocity definition. Namely, with p representing the Laplace variable

$$\boldsymbol{q}_d = \frac{1}{p+\lambda}(\dot{\boldsymbol{q}}_r + \lambda\boldsymbol{q}) \tag{22}$$

will provide the exact desired joint trajectory.

In the nonredundant case, the above is sufficient to implement a well-behaved Cartesian controller. In the redundant case, however, the Jacobian inverse does not exist, and further investigation becomes necessary. The method proposed here and implemented in the experimental work uses a pseudo-inverse solution and controls the null space of J to minimize a performance index. The performance index is chosen to be a quadratic norm of joint positions, similar to work done by Klein and Huang (1983) and Klein and Blaho (1987). Such an approach is computationally efficient and intuitively mimics a "flexible beam" clamped to the desired end point trajectory. It thus helps to keep the manipulator away from joint limits, as well as from singular positions. In addition, cyclic Cartesian motions converge to cyclic joint motions (see also Baker and Wampler 1988), avoiding joint trajectory drifts associated with mere psuedo-inverse methods.

In contrast to other controllers, sliding controllers require both reference velocity and reference acceleration. In addition to the pseudo-inverse (or generalized inverse) J^+ of the Jacobian J, we thus also need its time derivative. Assuming a Jacobian of linearly independent rows (i.e., a Jacobian describing a redundant manipulator outside of singularities), we can write explicit expressions for both:

$$J^+ = J^T(JJ^T)^{-1} \tag{23}$$

$$\dot{J}^+ = -J^+\dot{J}J^+ + (1 - J^+J)\dot{J}^T(JJ^T)^{-1}. \tag{24}$$

With these results, we can now specify the kinematic transformations:

$$\dot{\boldsymbol{q}}_r = J^+\dot{\boldsymbol{x}}_r + (1 - J^+J)\boldsymbol{\Psi} \tag{25}$$

$$\ddot{\boldsymbol{q}}_r = J^+(\ddot{\boldsymbol{x}}_r - \dot{J}\dot{\boldsymbol{q}}_r) + (1 - J^+J)[\dot{\boldsymbol{\Psi}} + \dot{J}^T J^{+T}(\dot{\boldsymbol{q}}_r - \boldsymbol{\Psi})] \tag{26}$$

where $\boldsymbol{\Psi}$ is an arbitrary, but continuous, joint space vector, which the joint velocity tracks within the null space. It is used to minimize the performance index and thus is set proportional to the negative gradient thereof. That is,

$$\boldsymbol{\Psi} = -\lambda\nabla f = -\lambda\boldsymbol{q}, \qquad \dot{\boldsymbol{\Psi}} = -\lambda\dot{\boldsymbol{q}} \tag{27}$$

in the case of

$$f = \sum_i \frac{1}{2} q_i^2.$$

An explicit Cartesian P.D. feedback torque $-J^T K_D(\dot{x} - \dot{x}_r)$ may also be added in both redundant and nonredundant cases.

Finally, note that many representations can be used for the end-effector orientation. For instance, defining the end-effector orientation as in Luh et al. (1980), with $(\boldsymbol{n}, \boldsymbol{o}, \boldsymbol{a})$ representing the actual orientation vector triplet, $(\boldsymbol{n}_d, \boldsymbol{o}_d, \boldsymbol{a}_d)$ representing the desired orientation vector triplet and $\boldsymbol{\omega}$ representing the angular velocity, and using the results of Yuan (1988), one can easily show that letting the orientation components $\dot{x}_r^{\text{orient}}$ of $\dot{x}_r$ be

$$\dot{x}_r^{\text{orient}} = \omega_d + \frac{\lambda}{2}(\boldsymbol{n} \times \boldsymbol{n}_d + \boldsymbol{o} \times \boldsymbol{o}_d + \boldsymbol{a} \times \boldsymbol{a}_d) \quad (28)$$

guarantees that the orientation error and the angular velocity error both go to zero (as $\dot{x} - \dot{x}_r$ goes to zero). Following Yuan (1988), it is also possible to use Euler parameters to represent the end-effector orientation, avoiding possible singularities of Euler angles and rotations.

3.2. *Implementation of Kinematic Transformations*

Implementing the kinematic transformations directly as written above can be computationally intensive, in particular as separate transformations of velocity and acceleration are required. Remembering that exact differentiation can be performed on filtered signals, a crude and unprovable approximation would be to transform only the velocity signal and then use a filter to differentiate and obtain the acceleration signal. Using this motivation, we now present a provable "non-linear" filter:

$$\ddot{q}_r + \lambda \dot{q}_r = J^+(\ddot{x}_r + \lambda \dot{x}_r - \dot{J}\dot{q}_r) + (1 - J^+J)[\dot{\Psi} + \lambda\Psi]. \quad (29)$$

After transients this filter allows the joint reference quantities to *exactly* follow any possible Cartesian reference quantities. However, it does approximate the null-space components, which nevertheless prevent drift of the joint reference trajectory and limit the joint displacements, as was originally intended. Thus implementation of this filter achieves the proper behavior and performance, while quite drastically reducing the computational burden of the inverse kinematics.

Computation can also be reduced by using effective algorithms for the pseudo-inversion. After (recursively) computing the Jacobian, the pseudo-inverse is best obtained using an orthogonalization algorithm equivalent to Gramm-Schmidt's. That is, the Jacobian $\boldsymbol{J}$ is decomposed into a lower triangular matrix $\boldsymbol{R}$ and a row orthogonal matrix $\boldsymbol{Q}$ as

$$J = RQ. \quad (30)$$

The pseudo-inversion is then achieved by transposing $\boldsymbol{Q}$ and using backsubstitution to invert $\boldsymbol{R}$:

$$J^+ = Q^T R^{-1}. \quad (31)$$

Note that the pseudo-inverse itself is never computed but rather is substituted directly into the transformation equations, resulting in

$$\ddot{q}_r + \lambda \dot{q}_r = q^* + Q^T(R^{-1}x^* - Qq^*) \quad (32)$$

with

$$x^* = \ddot{x}_r + \lambda \dot{x}_r - \dot{J}\dot{q}_r$$

$$q^* = \dot{\Psi} + \lambda\Psi.$$

3.3. *Constrained Motion Control*

Many useful tasks are not only described in Cartesian space, but also involve interaction with some fixed environment. Controlling the end-effector impedance is an attractive option for such tasks (Hogan 1985), as it imitates a passive mechanism and thus yields stable interaction with any passive environment. However, using impedance control for tracking limits performance, because it does not include dynamic effects (as is the case for all P.D. controllers). In this section, we introduce impedance control features within the framework of adaptive control. We assume the environments to be passive, without further models thereof. Furthermore, force measurements are not required. Only the joint positions and velocities are needed for feedback, as in the earlier developments.

In the next section we interpret our feedback control law and adaptation law in terms of the creation of passive mappings. This passivity formulation (Popov 1973; Desoer and Vidyasagar 1975) will allow our analysis to account for interactions with the environment in a straightforward fashion, as detailed in the next section.

Passivity Interpretation of the Adaptive Controller

As noticed by many researchers (Ortega and Spong 1988; Landau and Horowitz 1988; Kelly and Carelli 1988), the Lyapunov-like derivation of the direct adaptive manipulator controller of Slotine and Li (1986) can easily be translated in terms of passivity arguments. Conservation of energy (which motivates the original design),

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}[\dot{q}^T H \dot{q}] + \dot{q}^T G = \dot{q}^T \tau, \quad (33)$$

can be interpreted as reflecting a passive mapping $\tau \rightarrow \dot{q}$ for the open-loop manipulator. The exact (ideal) feedforward component of the control law (6) then modifies input and output variables of this mapping to create a passive mapping between additional torque inputs (or external forces reflected at the joints) and reference velocity error $\mathbf{s} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_r$. Because of the parametric uncertainty, the actual additional torque inputs correspond not only to additional controller torques or external forces, denoted by $\boldsymbol{\tau}^*$, but also to errors $\mathbf{Y}\tilde{\mathbf{a}}$ in the feedforward compensation. The P.D. component of the control law, in turn, adds a dissipative element to the system. As Figure 1 illustrates, the closed-loop system thus represents a dissipative mapping. From the Lyapunov analysis, the control law indeed yields

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{s}^T\mathbf{H}\mathbf{s}] + \mathbf{s}^T\mathbf{K}_D\mathbf{s} = \mathbf{s}^T(\boldsymbol{\tau}^* + \mathbf{Y}\tilde{\mathbf{a}}) \qquad (34)$$

Furthermore, using the adaptation law (7) then corresponds to inserting a passive feedback block between $\mathbf{s}$ and $(-\mathbf{Y}\tilde{\mathbf{a}})$, because

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}[\tilde{\mathbf{a}}^T\mathbf{P}^{-1}\tilde{\mathbf{a}}] = -\mathbf{s}^T\mathbf{Y}\tilde{\mathbf{a}}. \qquad (35)$$

This can also be shown directly by noticing that the integrator structure

$$\dot{\tilde{\mathbf{a}}} = \dot{\hat{\mathbf{a}}} = -\mathbf{P}\mathbf{Y}^T\mathbf{s} \qquad (36)$$

implies a passive map $(-\mathbf{Y}^T\mathbf{s}) \to \tilde{\mathbf{a}}$, and thus also a passive map $\mathbf{s} \to (-\mathbf{Y}\tilde{\mathbf{a}})$.

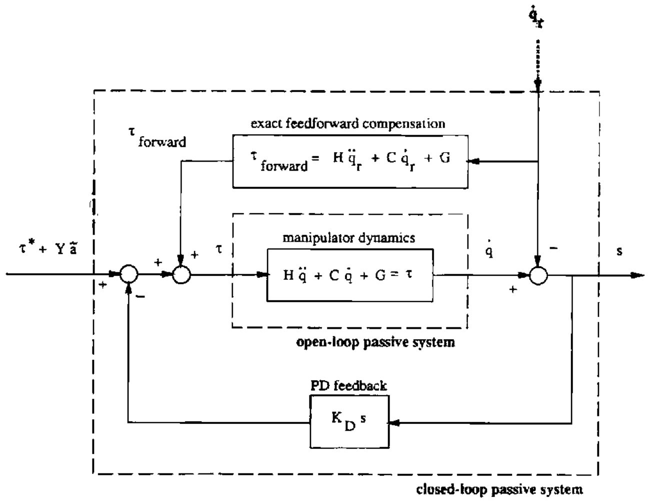
Elementary passivity results (Slotine and Li 1990) then show that such a configuration, as shown in Figure 2, is passive as a whole and therefore globally stable. They further allow new passive blocks to be added to the system in a simple fashion while preserving the overall stability and convergence. Correspondingly, they always guarantee the existence of a Lyapunov-like function V, upper bounding the original V and such that $\dot{V}(t) = -\mathbf{s}^T(\mathbf{D}_v + \mathbf{K}_D)\mathbf{s} \leq 0$, thus preserving overall stability and the convergence of $\mathbf{s}$ to zero.

We will use this passivity interpretation in the following to study interactions with an environment and to combine the adaptive controller with an impedance controller. Note that changes of coordinates do not affect passivity, so that corresponding coordinate changes may be performed simultaneously on both input and output of a passive system. That is, we can view the input and output of the above system either as joint space variables or as Cartesian end-effector variables, the latter of which will be more convenient in the next section. This is quickly proven by noting that

$$\boldsymbol{\tau}^{*T}\mathbf{s} = (\mathbf{J}^T\mathbf{F})^T\mathbf{s} = \mathbf{F}^T(\mathbf{J}\mathbf{s}) = \mathbf{F}^T(\dot{\mathbf{x}} - \dot{\mathbf{x}}_r). \qquad (37)$$

Adaptive Impedance Control

An arbitrary passive environment can be interpreted as a passive map between contact forces and displacement velocity, as it can only provide a finite amount of energy. To account for the adaptive controller contacting an environment, we add a further element, representing the contact forces, to the above passive system. Note, however, that the environment's passivity is not guaranteed between the Cartesian tracking error $(\dot{\mathbf{x}} - \dot{\mathbf{x}}_r)$ and contact forces in general, a fact reflecting the motion constraint imposed by the envi-



$\dot{q}_r$
$\tau_{forward}$
exact feedforward compensation
$\tau_{forward} = H\ddot{q}_r + C\dot{q}_r + G$
manipulator dynamics
$H\ddot{q} + C\dot{q} + G = \tau$
τ
$\dot{q}$
$\tau^* + Y\tilde{a}$
s
open-loop passive system
PD feedback
$K_D s$
closed-loop passive system

Fig. 1. Open- versus closed-loop passive mapping of the manipulator. The open loop mapping $\tau \to \dot{q}$ is modified by the feedforward compensation to create a mapping between additional torque inputs and reference velocity error. The P.D. feedback torques add an additional dissipative element.

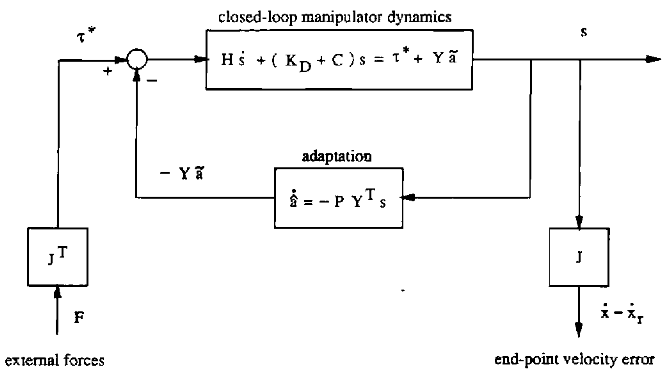


Fig. 2. Passivity interpretation of the adaptive controller. The adaptation accounts for the errors in feedforward compensation resulting from parametric uncertainty. The resulting system can also be viewed in Cartesian space with appropriate Jacobian transformations.

ronment. It is thus necessary to redefine the Cartesian reference velocity vector $\dot{x}_r$ in order to maintain global stability.

Given the surface orientation, we can divide the end point reference frame into subsets parallel and perpendicular to the surface. Along the parallel directions, motion remains unrestricted, and no contact forces (beside contact friction) can occur. Therefore in these directions no changes have to be made to the definition of $\dot{x}_r$. Along perpendicular directions, however, motion is restricted, and tracking is impossible. The components of the reference velocity $\dot{x}_r$ along these directions must consequently be ignored and are simply set to zero. That is

$$\dot{x}_{r_i} = \begin{cases} \dot{x}_{d_i} - \lambda\tilde{x}_i & \parallel \text{surface} \\ \mathbf{0} & \perp \text{surface} \end{cases} \tag{38}$$

Having redefined $\dot{x}_r$ in such a fashion, contact with the environment using an adaptive sliding controller is now possible. However, because position feedback was completely removed in directions perpendicular to the surface, no guarantee is given that contact will actually be maintained. This problem can easily be solved by adding an impedance control (Cartesian P.D.) component restricted to the perpendicular directions that imitates a spring (and possibly damper) system along these directions. Note that damping is still provided by the adaptive controller itself:

$$\mathrm{F}_{imp} = -\boldsymbol{K}(\boldsymbol{x} - \boldsymbol{x}_0)\perp. \tag{39}$$

Because impedance controllers also have the property of being passive (from end point force to velocity), the whole system, as illustrated in Figure 3, is stable and combines the advantages of its individual components (i.e., adaptation, precise tracking, and stable contact). Furthermore, only position and velocity measurements are necessary, and no force measurement is required.

To implement this complete system, only $\dot{x}_r$ needs to be modified to lie parallel to the surface. Therefore the same algorithms can be used for both free and compliant motion. The surface normal can be computed either a priori or on line, based on the direction of the error and the velocity. One may also add adaptive compensation for the contact friction in the same manner as for joint friction.

4. Experimental Results

The above developments were tested on a 4-D.O.F. cable-driven "whole-arm" manipulator (WAM) designed at the M.I.T. Artificial Intelligence Laboratory (Townsend 1988; Salisbury et al. 1988). The manipulator's geometry is shown in Figure 4. The torques are generated by four pulse-width modulated motors capable of delivering a maximum of 1.5 N-m each. They are located close to the base and connected to the lightweight links via cables, introducing a transmision ratio varying between 1:20 and 1:30. Reduction is done at the joints. Position measurements are made at the motor shafts using resolvers. In order to exploit the wide dynamic range that the manipulator can potentially achieve, the arm is connected to a VME-Bus-based multiprocessor system (Narasimhan et al. 1988) consisting of up to six 68020-based processor boards, D/A and A/D boards, a parallel interface, and other boards. This system is interfaced with the net-

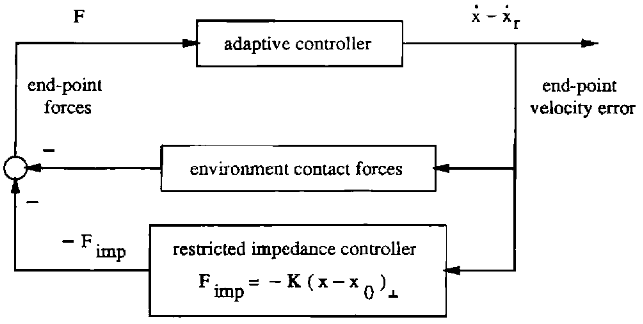


Fig. 3. Complete control system used for compliant motion. The control system combines adaptation, precise tracking, and stable contact be setting the reference velocity components in directions perpendicular to the contact surface to zero.

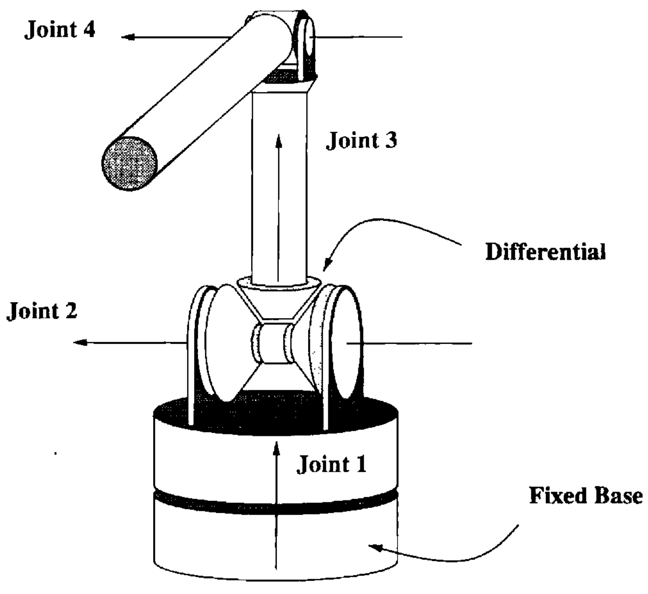


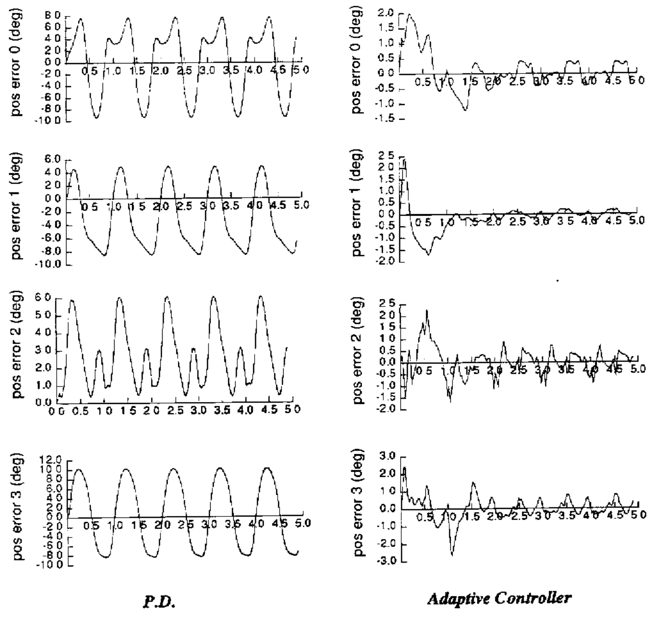
Fig. 4. The WAM manipulator. The manipulator has an extended length of 1 m and is comparable in geometry to the human arm.

work via a Sun-3 Workstation (Sun Microsystems, Inc.). The different processors are used to implement high-speed input/output routines, the basic controller algorithm, the adaptation algorithm, the inverse kinematics, and the trajectory generation, as well as other higher level tasks. The i/o processor performs both the input acquisition and filtering at 4 kHz and P.D. torque calculation and output at 2 kHz. The controller, adaptation, inverse kinematics, and trajectory generation algorithms are executed in synchronism on separate processors at 200 Hz. The velocity signals are created on the i/o processor by filtered differentiation of the 16-bit position signals (12-bit resolver signals plus rotation count). The integration in the adaptation algorithm is performed using a second-order Adams-Bashforth scheme. All programs are written in C.

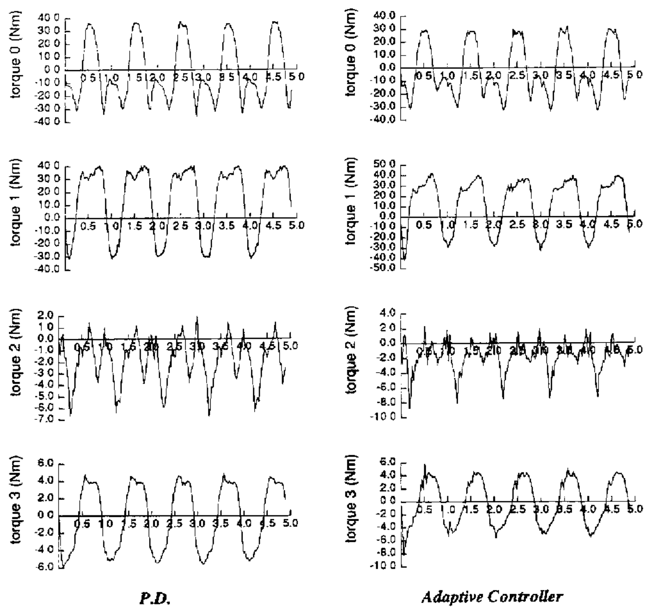
The friction model used for the manipulator consists of viscous and Coulomb friction, where Coulomb friction is allowed to be direction dependent. We therefore deal with 12 friction parameters in addition to 24 inertial parameters, and thus we are adapting to 36 unknown values. While dealing with unknown loads in principle only requires adaption to 10 unknown parameters, the capability of effectively adapting to all 36 parameters allows the range of application to be considerably extended, as discussed later. Also, a closed chain is used in the kinematics, as the cable transmission involves a differential.

In the experiments of Figure 5, the manipulator was commanded to follow a sinusoidal joint trajectory of period 1 and amplitude approximately equal to $\pm 45°$ per joint, thus allowing the tip to travel 5 m per period, with maximum tip velocities and accelerations of 8.5 m/s and 10 g. This was done in (i) with a simple P.D. with the highest possible gain matrix $\mathbf{K}_D$ (whose performance, of course, is helped by the transmission ratios) and in (ii) with the adaptive scheme starting as a P.D. (i.e., with an initial $\hat{\mathbf{a}} = \mathbf{0}$). The plots clearly demonstrate an improvement of 10 to 25 times in tracking error after transients of about 1 s when adapting to all 36 parameters. Furthermore, although both controllers start identical, the maximum tracking error of the adaptive controller during transients remains 2 to 10 times smaller than that of the P.D. The generated joint torques, nevertheless, are very similar in both smoothness and amplitude.

The residual tracking error is mostly caused by the unmodeled dynamics of the actuator, the cable transmissions, and the link flexibility. Also, the motors produce torque ripple of about 5%. However, parameter drift was not observed to be significant during these experiments, so that adaptation dead zones were not used. Nevertheless, it is possible to incorporate such dead zones to avoid long term parameter drift



A



B

Fig. 5. A, *Tracking errors $\tilde{q}$ (in degrees). These tracking errors were obtained on a high-speed sinusoidal trajectory of period 1 s. The comparison between simple P.D. control and adaptive control starting with no initial parameter information* (i.e., $\hat{\mathbf{a}} = \mathbf{0}$) *clearly shows an improvement factor of 10 to 25 after transients of approximately 1 s. Even during transients the tracking error remains 2 to 10 times smaller under adaptive control than under P.D. control.* B, *Joint torques τ (in N = m). The joint torques commanded by both P.D. and adaptive control are very similar in magnitude as well as smoothness. Thus the adaptive controller does not require higher control authority or activity but simply uses the available torques more effectively.*

and enhance robustness. The constant adaptation gains were set according to the following equation, which is intuitively motivated by least-squares identification rules. With Y defined in equation (4),

$$P \sim \textbf{Diagonal}\left[\int_0^T Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d)^T Y(q_d, \dot{q}_d, \dot{q}_d, \ddot{q}_d)\, dt\right]^{-1}. \quad (40)$$

These values were used in all further experiments along various trajectories, and variations thereof were found to have little effect on the system performance.

In other experiments, the adaptive controller was used to push a large box (weight about 10 times that of the last link, and volume about 0.2 m^3) over the floor at speeds of about .75 m/s. This exploited the kinematic redundancy of the arm to maintain line contact with the box while pushing. Despite the lack of accurate models for the relative motion between the box and the arm or for the friction between the box and the floor, this strategy allowed accurate "whole-arm" manipulation of the box. The presence of the box was simply interpreted by the algorithm as a change in inertial and friction coefficients. We believe that this robustness is quite remarkable for a reasonably complex high-performance algorithm and that it should extend the range of applications for adaptive manipulator control well beyond adaptation to grasped loads. Also, the accuracy of the manipulator controller allows us to make full use of the "geometric" stability properties of the task (Mason 1986, in the case of pushing), thus allowing large uncertainties on the initial position of the box to be tolerated and high-speed performance to be achieved without slow transition (contact) phases.

The inverse kinematics and adaptive Cartesian control were tested on trajectories describing a square in the vertical plane. The results, shown in Figure 6, again clearly demonstrate the improvement caused by the adaptation process. Furthermore, the adaptive impedance controller was demonstrated to have excellent performance in making and maintaining stable contact with an unknown curved surface while performing adaptive tracking convergence in the unconstrained directions.

5. Concluding Remarks

We believe that advanced control concepts have exceptional potential in the development of robust, reliable, high-performance robotic systems. We hope studies such as the one presented here will accelerate such implementations in common industrial and scientific practice and thus allow the full potential of ever-increasing computational capabilities to be exploited effectively.

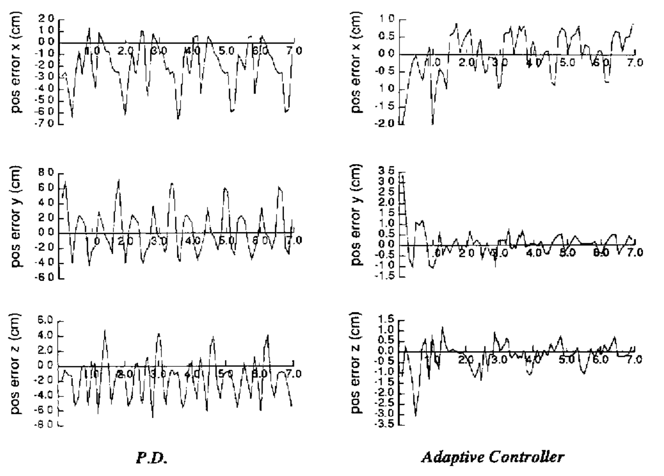


Fig. 6. Cartesian tracking errors $\tilde{x}(t)$ (in cm). These Cartesian tracking errors were obtained on a trajectory describing a square in the vertical plane. The sides are 1 m in length and require 0.4 s each. Again the comparison between simple P.D. control and adaptive control shows a clear improvement in both steady state and transients, even though adaptation starts with no prior information.

Appendix A: A Brief Introduction to the Spatial Vector Notation

Individual points in space have three degrees of freedom and are well described by three-dimensional vectors, denoted in this appendix by arrows ($\vec{v}$). Rigid bodies, however, have six degrees of freedom and are normally described by sets of two three-dimensional vectors corresonding to linear and angular quantities. The spatial notation of Featherstone (1987) combines these two vectors into a single six-dimensional vector. Although this obviously reduces the number of equations, it also reduces their complexity, as the coupling between linear and angular quantities can be taken care of automatically. Spatial vectors will be denoted as bold characters, for instance

$$\mathbf{v} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \quad \text{and} \quad \mathbf{F} = \begin{bmatrix} \vec{F} \\ \vec{\tau} \end{bmatrix}. \quad (41)$$

Note that in vectors of motion (velocity, acceleration, etc.) the angular quantity resides in the upper half, whereas in vectors of force (force, momentum, etc.) the angular quantity resides in the lower half of the spatial vector. This is because both angular velocities and linear forces are dependent on their point of attack.

The spatial velocity $\mathbf{v}|^1$ combines the angular velocity and linear velocity measured at the origin of and expressed in coordinate frame 1. Changing coordinate frames from 1 to 2 involves shifting the origin from $\vec{r}_1$ to $\vec{r}_2$, as well as rotating according to the 3×3 rotation matrix $R_1|^2$. As in standard vector notation, this rotation matrix consists of the base vectors of frame 1 expressed in frame 2. The resulting transformation matrix $X_1|^2$ is defined in spatial notation by

$$v|^2 = \begin{bmatrix} \vec{\omega}|^2 \\ \vec{v}|^2 \end{bmatrix} = \begin{bmatrix} R_1|^2 & 0 \\ \vec{r}_1{}^2|^2 \times R_1|^2 & R_1|^2 \end{bmatrix} \cdot \begin{bmatrix} \vec{\omega}|^1 \\ \vec{v}|^1 \end{bmatrix} = X_1|^2 v|^1, \tag{42}$$

where $\vec{r}_1{}^2 = \vec{r}_1 - \vec{r}_2$.

The notation $\vec{r}\times$ denotes a matrix that, multiplied by a vector $\vec{s}$, equals the cross product $\vec{r} \times \vec{s}$. Similarly, the spatial cross product is defined as

$$a \times b = (a\times)b = \begin{bmatrix} \vec{m}\times & 0 \\ \vec{n}\times & \vec{m}\times \end{bmatrix} \cdot b \quad \text{with} \quad a = \begin{bmatrix} \vec{m} \\ \vec{n} \end{bmatrix}. \tag{43}$$

The transpose operator is defined differently from what one might expect, so that the product $F^T \cdot v$ is equal to power.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^T = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix}, \quad \begin{bmatrix} \vec{m} \\ \vec{n} \end{bmatrix}^T = [\vec{n}^T \quad \vec{m}^T]. \tag{44}$$

Nevertheless both operators have the same properties as their counterparts in standard vector notation. With these algebraic operators, the transformation matrices X have the following properties:

$$X_a|^c = X_b|^c \cdot X_a|^b \tag{45}$$

$$(X_a|^b)^{-1} = (X_a|^b)^T = X_b|^a \tag{46}$$

$$\frac{d}{dt} X_a|^b = v_a{}^b|^b \times X_a|^b = (v_a - v_b)|^b \times X_a|^b \tag{47}$$

To express rigid-body dynamics we also need to define an inertia matrix I that is a 6×6 matrix mapping spatial velocity to spatial momentum. It consists of the standard 3×3 inertia matrix J, the mass m, and the product of the mass with the position of the center of mass $\vec{p} = m\vec{r}$.

$$I_c|^c = \begin{bmatrix} -\vec{p} & m \cdot \mathbf{1} \\ J & \vec{p} \end{bmatrix} \quad \text{with } \mathbf{1} \text{ being the } 3 \times 3 \text{ unity matrix.} \tag{48}$$

This spatial inertia thus has the following properties:

$$I_c|^a = X_c|^a \cdot I_c|^c \cdot X_a|^c \tag{49}$$

$$\frac{d}{dt} I_c|^a = v_c{}^a|^a \times I_c|^a - I_c|^a v_c{}^a|^a \times. \tag{50}$$

These tools now allow us to write the equation of motion for an arbitrary rigid body. It includes both Newton's and Euler's equations and accounts for any coupling automatically. With $a = \frac{d}{dt} v$ as the acceleration, we have

$$F = \frac{d}{dt} Iv = Ia + v \times Iv. \tag{51}$$

Similarly, the spatial notation simplifies the total kinetic energy and power input, including both rotational and translational components:

$$E_{\text{kin}} = \frac{1}{2} v^T I v \tag{52}$$

$$P_{\text{in}} = v^T F. \tag{53}$$

Appendix B: Transformations for Parameter Reductions

In section 2.3 we introduced an inertia variation matrix δI that is subtracted from link $k-1$ and added to link k. This matrix represents unobservable parameter combinations and allows parameters to be transferred from link $k-1$ to link k, or vice versa. Therefore by choosing the correct variation matrix δI, we can set several parameter values to zero and lump their contents into other parameters.

To transfer parameters, and equivalently the inertia variation δI, from one link to another, a reference frame transformation is necessary. This transformation is given below for the appropriate structure of inertia variation, possible for arbitrary rotational or translational joints. Note that this transformation is spatial and that the Denavit-Hartenberg parameters (a, α, d, θ) correspond to joint k, which is attached to frame $k-1$ and connects links $k-1$ and k.

Rotational Joints

$$\delta I|^{k-1} \quad \delta J_{xx} = \delta J_{yy} = J, \quad \delta p_z = p_z, \quad \delta m = m$$

$$\begin{aligned}
\delta I|^k \quad \delta J_{xx} &= J + md^2 - 2dp_z \\
\delta J_{yy} &= (J + md^2 - 2dp_z)\cos^2(\alpha) + ma^2 \\
\delta J_{zz} &= (J + md^2 - 2dp_z)\sin^2(\alpha) + ma^2 \\
\delta J_{xy} &= (ap_z - mad)\sin(\alpha) \\
\delta J_{xz} &= (ap_z - mad)\cos(\alpha) \\
\delta J_{yz} &= -(J + md^2 - 2dp_z)\sin(\alpha)\cos(\alpha) \\
\delta p_x &= -ma \\
\delta p_y &= (p_z - md)\sin(\alpha) \\
\delta p_z &= (p_z - md)\cos(\alpha)
\end{aligned}$$

Translational Joints

$$\boldsymbol{\delta I}|^{k-1} \quad \begin{array}{lll} \delta J_{xx} = J_{xx}, & \delta J_{yy} = J_{yy}, & \delta J_{zz} = J_{zz} \\ \delta J_{xy} = J_{xy}, & \delta J_{xz} = J_{xz}, & \delta J_{yz} = J_{yz} \end{array}$$

$$\begin{aligned} \boldsymbol{\delta I}|^{k} \quad \delta J_{xx} &= J_{xx} \cos^2(\theta) + 2J_{xy} \sin(\theta) \cos(\theta) + J_{yy} \sin^2(\theta) \\ \delta J_{yy} &= (J_{xx} \sin^2(\theta) - 2J_{xy} \sin(\theta) \cos(\theta) \\ &\quad + J_{yy} \cos^2(\theta)) \cos^2(\alpha) \\ &\quad - 2(J_{xz} \sin(\theta) - J_{yz} \cos(\theta)) \sin(\alpha) \cos(\alpha) \\ &\quad + J_{zz} \sin^2(\alpha) \\ \delta J_{zz} &= (J_{xx} \sin^2(\theta) - 2J_{xy} \sin(\theta) \cos(\theta) \\ &\quad + J_{yy} \cos^2(\theta)) \sin^2(\theta) \\ &\quad + 2(J_{xz} \sin(\theta) - J_{yz} \cos \theta)) \sin(\alpha) \cos(\alpha) \\ &\quad + J_{zz} \cos^2(\alpha) \\ \delta J_{xy} &= (-(J_{xx} - J_{yy}) \sin(\theta) \cos(\theta) \\ &\quad + J_{xy}(\cos^2(\theta) - \sin^2(\theta))) \cos(\alpha) \\ &\quad + (J_{xz} \cos(\theta) + J_{yz} \sin(\theta)) \sin(\alpha) \\ \delta J_{xz} &= ((J_{xx} - J_{yy}) \sin(\theta) - J_{xy}(\cos^2(\theta) \\ &\quad - \sin^2(\theta))) \sin(\alpha) \\ &\quad + (J_{xz} \cos(\theta) + J_{yz} \sin(\theta)) \cos(\alpha) \\ \delta J_{yz} &= -(J_{xx} \sin^2(\theta) - 2J_{xy} \sin(\theta) \cos(\theta) \\ &\quad + J_{yy} \cos^2(\theta)) \sin(\alpha) \cos(\alpha) \\ &\quad - (J_{xz} \sin(\theta) + J_{yz} \cos(\theta))(\cos^2(\alpha) \\ &\quad - \sin^2(\alpha)) + J_{zz} \sin(\alpha) \cos(\alpha) \end{aligned}$$

Acknowledgments

We are most grateful to Ken Salisbury and Bill Townsend for allowing us to test the above results on the fine manipulator they developed and for stimulating discussions. We would also like to thank Brian Eberman and Sundar Narasimhan for their help in the implementation. This work was supported in part by a student grant from the H. A. Perry Foundation and in part by grant 8803767-MSM from the National Science Foundation. Partial support was also provided by the M.I.T. Seagrant Office. Development of the arm and control hardware was supported by grants N00014-86-K-0685 and N00015-85-K-0214 from the Office of Naval Research.

References

An, C. H., Atkeson, C. G., and Hollerbach, J. M. 1985 (Fort Lauderdale). Estimation of inertial parameters of rigid body links of manipulators. *IEEE Conf. Decision and Control.*

Asada, H., and Slotine, J. J. E. 1986. *Robot Analysis and Control.* New York: John Wiley.

Baker, D. R., and Wampler, C. W. II. 1988. On the inverse kinematics of redundant manipulators. *Int. J. Robot. Res.* 7(2):3-21.

Bayard, D. S., and Wen, J. T. 1987 (New Haven). Simple adaptive control laws for robotic manipulators. *Proceedings of the Fifth Yale Workshop on the Applications of Adaptive Systems Theory.*

Craig, J. J., Hsu, P., and Sastry, S. 1986 (San Francisco). Adaptive control of mechanical manipulators. *IEEE Int. Conf. Robotics and Automation.*

Desoer, C. A., and Vidyasagar, M. 1975. *Feedback Systems: Input-Output Properties.* New York: Academic Press.

Featherstone, R. 1987. Robot Dynamics Algorithms. Boston: Kluwer Academic Publishers.

Hogan, N. 1985. Impedance control: An approach to manipulation: Part I—theory. *J. Dyn. Syst. Measurements Control.* 107(1):1-7.

Hsu, P., Bodson, M., Sastry, S., and Paden, B. 1987 (Raleigh, N.C.). Adaptive identification and control of manipulators without joint acceleration measurements. *IEEE Int. Conf. Robotics and Automation.*

Kelly, R., and Carelli, R. 1988 (Austin, Tex.). Unified approach to adaptive control of robotic manipulators. *Proc. 27th Conf. on Dec. and Contr.*, pp. 1598-1603.

Khatib, O. 1980. Commande dynamique dans l'espace operationnel des robots manipulateurs en presence d'obstacles. Docteur ingènieur thesis. Toulouse, France, Ecole Nationale Superieure de L'Aeronautique et de L'Espace.

Khatib, O. 1983 (New Delhi). Dynamic control of manipulators in operational space. *6th IFTOMM Congress on Theory of Machines and Mechanisms.*

Khosla, P., and Kanade, T. 1985 (Fort Lauderdale). Parameter identification of robot dynamics. *IEEE Conf. Decision and Control.*

Klein, C. A., and Blaho, B. E. 1987. Dexterity measures for the design and control of kinematically redundant manipulators. *Int. J. Robot. Res.* 6(2):72-83.

Klein, C. A., and Huang, C. H. 1983. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. Sys. Man Cybernet.* 13(2):245-250.

Koditschek, D. E. 1987 (New Haven). Adaptive techniques for mechanical systems. *Proceedings of the Fifth Yale Workshop on the Applications of Adaptive Systems Theory.*

Landau, I., and Horowitz, R. 1988 (Philadelphia). Synthesis of adaptive controllers for robot manipulators using a passive feedback systems approach. *IEEE Int. Conf. Robotics and Automation.*

Li, W., and Slotine, J. J. E. 1988 (Philadelphia). Indirect adaptive robot control. *IEEE Int. Conf. Robotics and Automation.*

Luh, J. Y. S., Walker, M., and Paul, R. P. C. 1980. Resolved acceleration control of mechanical manipulators. *IEEE Trans. Auto. Control.* 25(3):468-474.

Mason, M. T. 1986. Mechanics and planning of manipulator pushing operations. *Int. J. Robot. Res.* 5(3):53-71.

Mayeda, H., Yoshida, K., and Osuka, K. 1988 (Philadelphia). Base parameters of manipulator dynamic models. *IEEE Int. Conf. Robotics and Automation.*

Middleton, R. H., and Goodwin, G. C. 1986 (Athens). Adaptive computed torque control for rigid link manipulators. *IEEE Conf. on Decision and Control.*

Narasimhan, S., Siegel, D. M., and Hollerbach, J. M. 1989. Condor: An architecture for controlling the Utah-MIT dexterous hand. *IEEE Trans. Robot. Automat.* 5(5):616-627.

Ortega, R., and Spong, M. 1988 (Austin, Tex.). Adaptive motion control of rigid robots: A tutorial. *IEEE Int. Conf. Decision and Control.*

Popov, V. M. 1973. *Hyperstability of Control Systems.* New York: Springer-Verlag.

Sadegh, N., and Horowitz, R. 1987 (Raleigh, N.C.). Stability analysis of an adaptive controller for robotic manipulators. *IEEE Int. Conf. Robotics and Automation.*

Salisbury, J. K., Townsend, W., Eberman, B., and DiPietro, D. 1988 (Philadelphia). Preliminary design of a whole-arm manipulator system. *IEEE Int. Conf. Robotics and Automation.*

Slotine, J. J. E., and Li, W. 1986 (Anaheim, Calif.). On the adaptive control of robot maniulators. *ASME Winter Annual Meeting.*

Slotine, J. J. E., and Li, W. 1987a (Raleigh, N. C.). Adaptive robot control, a case study. *IEEE Int. Conf. Robotics and Automation.*

Slotine, J. J. E., and Li, W. 1987b (Los Angeles). Adaptive robot control—a new perspective. *IEEE Conf. Decision and Control.*

Slotine, J. J. E., and Li, W. 1987c (Raleigh, N.C.). Adaptive strategies in constrained manipulation. *IEEE Int. Conf. Robotics and Automation.*

Slotine, J. J. E., and Li, W. 1987d. On the adaptive control of robot manipulators. *Int. J. Robot. Res.* 6(3):49–59.

Slotine, J. J. E., and Li, W. 1987e (New Haven). Theoretical issues in adaptive manipulator control. *Proceedings of the Fifth Yale Workshop on Applications of Adaptive Systems Theory.*

Slotine, J. J. E., and Li, W. 1990. *Applied Nonlinear Control.* Englewood Cliffs, N.J.: Prentice-Hall.

Townsend, W. T. 1988. The effect of transmission design on force-controlled manipulator performance. Technical report AI-TR1054. Cambridge, Mass.: Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Walker, M. W. 1988 (Philadelphia). An efficient algorithm for the adaptive control of a manipulator. *IEEE Int. Conf. Robotics and Automation.*

Yuan, J. S. 1988. Closed-loop manipulator control using quaternion feedback. *J. Robot. Automat.* 4(4):434–440.