COMPUTATIONAL ALGORITHMS FOR ADAPTIVE ROBOT CONTROL

by

Günter Dieter Niemeyer

B.S., Technische Hochschule Aachen, (1986)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1990

Signature of Author [illegible]
Department of Aeronautics and Astronautics
January 25, 1990

Certified by [illegible]
Professor Jean-Jacques E. Slotine
Thesis Supervisor
Department of Mechanical Engineering

Accepted by [illegible]
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

COMPUTATIONAL ALGORITHMS FOR ADAPTIVE ROBOT CONTROL

by

Günter Dieter Niemeyer

Submitted to the Department of Aeronautics and Astronautics on January 25, 1990 in partial fulfillment of the requirements for the degree of Master of Science in Aeronautics and Astronautics.

Abstract

Effective adaptive controller designs potentially combine high-speed and high-precision in robot manipulation. Furthermore, they can considerably simplify high-level programming by providing consistent performance in the face of large variations in loads and tasks. Globally convergent adaptive manipulator controllers have recently been developed for this purpose. However, computational complexity has so far restricted their applicability to manipulators with few degrees-of-freedom.

This work presents a new recursive algorithm, which effectively implements the simple, yet globally tracking-convergent, adaptive sliding manipulator controller. To further improve efficiency, a procedure is derived to select and evaluate a minimal set of inertial parameters of the manipulator. Since many tasks require direct cartesian control, the algorithm is subsequently expanded to handle cartesian data in real-time and possibly exploit kinematic redundancies. Finally, the application to constrained motion is discussed, introducing impedance control features to guarantee stable contacts to arbitrary passive environments. The complete system allows the effective use of adaptive strategies on multi degree-of-freedom manipulators for a variety of tasks.

The developments are implemented and illustrated experimentally on a four degree-of-freedom articulated robot arm and suggest a wide range of application beyond adaptation to grasped loads.

Thesis Supervisor: Professor Jean-Jacques E. Slotine
Title: Associate Professor of Mechanical Engineering

Acknowledgments

I wish to express my deepest gratitude to my advisor, Professor Jean-Jacques Slotine. He was a constant source of support, ideas, and excellent guidance and truly brought the project to life.

Special thanks go to Professor Kenneth Salisbury and Dr. William Townsend, who allowed me to use their truly fine manipulator and provided many stimulating discussions and much encouragement. Also many thanks to Brian Eberman and Sundar Narasimhan, who helped in the software implementation and were always full of suggestions.

I furthermore deeply appreciate the many friendships I have found here: My long-time office-mates Andre Sharon, Ted Clancy, Harvey Koselka, and Hyun Yang, the members of the Nonlinear Systems and Whole Arm groups, the Martin Design Center and the Artificial Intelligence Laboratory. They all provided numerous pieces of helpful advice and comments and made this time thoroughly enjoyable.

I also gratefully acknowledge the support from the H.A. Perry Foundation, as well as partial support by the MIT Seagrant Office and Grant 8803767-MSM from the National Science Foundation. 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.

Finally, I am most grateful to my family, who supported and encouraged my work in any way they could and, I am sure, without whom I would not be here today.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Adaptive control has been extensively studied, and several globally convergent controllers have been derived for linear time-invariant single-input single-output systems. Recent developments (for example [Slotine and Li, 1986], [Craig, Hsu, and Sastry, 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]) have been able to derive adaptive manipulator controllers with similar global convergence properties. This represents an important extension of adaptive control to a class of nonlinear multi-input systems. Such adaptive manipulator controllers potentially provide consistently high performance in the face of large variations of loads and tasks. Furthermore, this performance is achieved without requiring increased actuator bandwidth.

A simple, yet globally tracking-convergent, direct adaptive sliding manipulator controller was developed in [Slotine and Li, 1986] and demonstrated experimentally in subsequent work [Slotine and Li, 1987b]. Though the controller provides excellent performance and robustness properties, it application is currently limited by the computational burden of the nonlinear control and adaptation laws. Existing recursive algorithms are rendered useless as they do not match the nature of the controller.

The central part of the work presented here develops a new recursive algorithm to effectively implement the adaptive manipulator controller of [Slotine and Li, 1986]. This allows the application to multi degree-of-freedom manipulators, which so far have been excluded from the advanced control concepts.

An additional increase in computational efficiency can be achieved with the use of minimal parameter sets. This work also presents a procedure for selecting such minimal sets and evaluating the involved parameters. It is applicable to all joint types and kinematic structures and presents a more practical alternative to existing theoretical results ([An, Atkeson, and Hollerbach, 1985], [Khosla and Kanade, 1985], [Mayeda, 1988]).

Since many interesting tasks can be addressed with direct cartesian control, this work proceeds by extending the recursive algorithm to handle cartesian data in real-time. It thus provides a cartesian implementation of the adaptive controller. To handle redundant manipulators, a pseudo-inverse method is used with nullspace minimization of a joint position norm. On the basis of this cartesian implementation, the application to constrained motion is then examined. In particular, impedance control features are introduced to guarantee stable contacts with arbitrary passive environments. The complete system then allows the effective use of adaptive strategies on multi degree-of-freedom manipulators for a large variety of tasks.

The developments are implemented and illustrated experimentally on a four degree-of-freedom articulated robot arm. A comparison between P.D. and adaptive controller clearly demonstrates the improvements caused by the adaptation process. Further experiments also suggest a wide range of application beyond the adaptation to grasped loads.

Following a brief review of the direct adaptive controller of [Slotine and Li, 1986] in Chapter 2, Chapter 3 presents the recursive algorithm for the implementation. The minimal parameter sets are derived in Chapter 4, after which cartesian implementation and application to constrained motion are discussed in Chapters 5 and 6 respectively. Finally the experimental results are detailed in Chapter 7.

Chapter 2

Direct Adaptive Sliding Control

The direct adaptive sliding controller considered in this thesis was originally developed and later experimentally verified in [Slotine and Li, 1986, 1987a]. It is applicable to all rigid manipulators, regardless of kinematic structure, and briefly summarized in the following.

In the absence of friction and other disturbances, the dynamics of an n degree of freedom rigid manipulator (with the load considered as part of the last link) can be written as

$$\boldsymbol{H}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2.1}$$

where $\boldsymbol{q}$ and $\boldsymbol{\tau}$ are the $n\times1$ vectors of joint displacements and applied joint torques (or forces) respectively. $\boldsymbol{H}(\boldsymbol{q})$ denotes the $n\times n$ symmetric positive definite (s.p.d.) manipulator inertia matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}$ is the $n\times1$ vector of centripetal and Coriolis torques, and $\boldsymbol{G}(\boldsymbol{q})$ gives the $n\times1$ vector of gravitational torques. A friction model can be added to the above dynamics, using e.g. $\boldsymbol{D}_s\ \mathrm{sgn}(\dot{\boldsymbol{q}})$ and $\boldsymbol{D}_v\ \dot{\boldsymbol{q}}$ as the $n\times1$ vectors of static (Coulomb) and viscous friction torques. Note that the sign operator sgn(.) is performed independently in each component, and the matrices $\boldsymbol{D}_s$ and $\boldsymbol{D}_v$ are diagonal positive definite (or semi-definite). Including friction, the manipulator can thus be modeled as

$$\boldsymbol{H}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{D}_s\,\mathrm{sgn}(\dot{\boldsymbol{q}}) + \boldsymbol{D}_v\,\dot{\boldsymbol{q}} = \boldsymbol{\tau} \tag{2.2}$$

The adaptive controller design problem is stated as follows: given the desired trajectory $\boldsymbol{q}_d(t)$, 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 $\boldsymbol{q}(t)$ closely tracks the desired trajectory $\boldsymbol{q}_d(t)$.

In contrast to other controllers, a *reference* trajectory, which incorporates a first level of feedback, is defined together with an associated error measure as

$$\dot{\boldsymbol{q}}_r = \dot{\boldsymbol{q}}_d - \boldsymbol{\Lambda}\,\tilde{\boldsymbol{q}} \tag{2.3}$$

$$\boldsymbol{s} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_r = \dot{\tilde{\boldsymbol{q}}} + \boldsymbol{\Lambda}\,\tilde{\boldsymbol{q}} \tag{2.4}$$

where $\tilde{\boldsymbol{q}} = \boldsymbol{q} - \boldsymbol{q}_d$ and $\boldsymbol{\Lambda}$ is a s.p.d. matrix (or, more generally, a matrix whose eigenvalues are strictly in the right-half complex plane). Also, $\hat{\boldsymbol{a}}(t)$ is defined as the current estimate of the constant vector $\boldsymbol{a}$ of the manipulator's dynamic parameters, with $\tilde{\boldsymbol{a}}(t) = \hat{\boldsymbol{a}}(t) - \boldsymbol{a}$.

The control and adaptation laws can now be given as

$$\boldsymbol{\tau} = \boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\dot{\boldsymbol{q}}_r,\ddot{\boldsymbol{q}}_r)\,\hat{\boldsymbol{a}} - \boldsymbol{K}_D\,\boldsymbol{s} \tag{2.5}$$

$$\dot{\hat{\boldsymbol{a}}} = -\boldsymbol{P}\,\boldsymbol{Y}^T\boldsymbol{s} \tag{2.6}$$

where $\boldsymbol{K}_D$ and $\boldsymbol{P}$ are symmetric positive definite gain matrices and the matrix $\boldsymbol{Y}(\boldsymbol{q},\dot{\boldsymbol{q}},\dot{\boldsymbol{q}}_r,\ddot{\boldsymbol{q}}_r)$ is defined by

$$Y(q,\dot{q},\dot{q}_r,\ddot{q}_r)\,\hat{a} = \hat{H}(q)\,\ddot{q}_r + \hat{C}(q,\dot{q})\,\dot{q}_r + \hat{G}(q) + \hat{D}_s\,\mathrm{sgn}(\dot{q}_r) + \hat{D}_v\,\dot{q}_r \tag{2.7}$$

The controller thus consists of a P.D. feedback term plus a particular type of feedforward. To be more precise, the feedforward terms do not cancel the natural dynamics, but rather compensates for the torques corresponding to the reference trajectory. This helps the overall convergence (adding negative terms to $\dot{V}(t)$) and avoids the use of velocity measurements for friction compensation. Therefore problems of inaccurate friction models at zero velocity are alleviated and no boundary layer approximation is necessary for Coulomb friction. This is true in general for all friction models, that provide a total friction torque monotone in velocity.

The analysis and stability proof of this controller are done using a Lyapunov-like function

$$V(t) = \frac{1}{2}\, s^T H\, s + \frac{1}{2}\, \tilde{a}^T P^{-1}\, \tilde{a} \tag{2.8}$$

and exploiting fundamental physical properties of the system, such as conservation of energy and the positive-definiteness of the inertia matrix H. Substituting the above control and adaptation laws (2.5) and (2.6) then yields

$$\dot{V}(t) = -\, s^T (\, D_v + K_D\,)\, s \;-\; s^T\, D_s\, (\,\mathrm{sgn}(\dot{q}) - \mathrm{sgn}(\dot{q}_r)\,) \leq 0 \tag{2.9}$$

Using simple functional analysis arguments, this result can be shown to imply that $s \rightarrow 0$ as $t \rightarrow \infty$, which in turn implies that $\tilde{q}$ and $\dot{\tilde{q}}$ both tend to 0. The algorithm is therefore guaranteed global tracking convergence independent of the initial parameter estimates.

It is interesting to note that the use of an adaptive controller, such as the one given above, improves the performance of a manipulator without increasing the actuator bandwidth. This is extremely important when dealing with large loads, as they generally reduce the structural frequencies and limit the available bandwidth considerably, making fixed parameter controllers most sensitive to parameter uncertainties.

Chapter 3

Recursive Implementation

A major obstacle towards the implementation of multi-d.o.f. nonlinear controllers, is their computational complexity. This is particularly true for multi-d.o.f. manipulator controllers using full dynamics compensation, such as the adaptive controller described in Chapter 2. It is therefore important to develop and implement recursive algorithms to preform the necessary calculations. Such algorithms provide the necessary efficiency for multi-d.o.f. controllers, as their execution time remains linear in the number of degrees of freedom, or more precisely in the number of links.

Physical insight in the dynamic equations of motion for rigid manipulators has led to the recursive *Newton-Euler* algorithm. This algorithm exactly calculates the forces and torques necessary for any given position, velocity, and acceleration, as determined by the equations of motion. It thus provides an efficient implementation of "computed torque" type controllers, which attempt to precisely cancel the nonlinear dynamics. However, the adaptive sliding controller does not attempt cancellation, but introduces a reference velocity and slightly modifies the dynamic equations computing the applied torques. It thus can not make use of the Newton-Euler algorithm, which is limited to the exact equations of motion.

A solution to this problem was suggested by [Walker, 1988], who applies the ideas of [Slotine and Li, 1986] directly to the recursive Newton-Euler formulation of the equations of motion. The resulting algorithm is recursive, hence efficient, and has very similar convergence properties as the original adaptive sliding controller. Differences, however, exist and, in particular, there is no joint-space representation of the calculated torques, and therefore no simple "closed-form" of the resulting multi-input controller.

To achieve an exact implementation of the adaptive sliding controller of [Slotine and Li, 1986], this section develops a new recursive algorithm. Though it is derived independently from the Newton-Euler algorithm, it takes a similar form, generalized to accommodate the reference velocity. As the original algorithm, it remains applicable to all rigid manipulators, regardless of kinematic chains and joint structure. To allow a simpler notation, however, the derivation is detailed for open kinematic chains only. Remarks on the closed chain results follow thereafter.

The efficiency of a recursive algorithm is achieved by treating each link as an individual rigid body. The following thus proceeds by relating all the joint quantities of the controller to the rigid-body quantities of each link and then substituting these in the control and adaptation laws. Throughout these developments the standard Denavit-Hartenberg convention for coordinate frame locations and numbering is used (see, *e.g.*, [Asada and Slotine, 1986]). That is, for open kinematic chains, the links are numbered from zero (base) to n (tip) with joint i connecting link $i-1$ to link i.

3.1 The Definitions of Spatial Quantities

As in the computational approach of [Walker, 1988], a simplifying aspect of the derivation is the use of the spatial vector notation of [Featherstone, 1987]. This notation combines corresponding rotational and translational quantities into a single 6-dimensional vector, thus reducing the number and complexity of involved equations. Appendix A gives a summary of this vector notation and its fundamental rules, while this section defines the needed quantities.

Due to the nature of the spatial notation, the relative velocity of two connected links can simply be expressed as the product of a fixed spatial vector $\boldsymbol{d}_k$ and the joint velocity $\dot{q}_k$, regardless of the joint type (for example rotational, translational or even screw-like). The structure of this spatial vector $\boldsymbol{d}_k$, also referred to as the joint axis, determines the type of joint, while its norm specifies the gear ratio. That is

$$\boldsymbol{v}_k - \boldsymbol{v}_{k-1} = \boldsymbol{d}_k \cdot \dot{q}_k \tag{3.1}$$

Therefore the spatial velocity $\boldsymbol{v}_k$, reference velocity $\boldsymbol{w}_k$ and reference acceleration $\dot{\boldsymbol{w}}_k$ as well as the velocity error $\boldsymbol{e}_k$ of each link can be written as

$$\boldsymbol{v}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \, \dot{q}_l \tag{3.2}$$

$$\boldsymbol{w}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \, \dot{q}_{r_l} \tag{3.3}$$

$$\dot{\boldsymbol{w}}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \, \ddot{q}_{r_l} + \boldsymbol{v}_l \times \boldsymbol{d}_l \, \dot{q}_{r_l} \tag{3.4}$$

$$\boldsymbol{e}_k = \boldsymbol{v}_k - \boldsymbol{w}_k = \sum_{l=1}^{k} \boldsymbol{d}_l \, s_l \tag{3.5}$$

Associated with each link is also a spatial inertia matrix $\boldsymbol{I}_k$ containing all ten inertia parameters of the link. Defined as in Appendix-Section A.4, these matrices can be expressed using sparse placement matrices R_i and the ten parameters values a_k^i of the link.

$$I_k = \sum_{i=1}^{10} R_i \, a_k^i \tag{3.6}$$

The local spatial forces f_k caused by each link can also be divided into the effects of all ten parameters.

$$f_k = \sum_{i=1}^{10} f_k^i \, a_k^i \tag{3.7}$$

Summing the local forces of the above links then results in the total force F_k to be applied to a particular link

$$F_k = \sum_{l=k}^{n} f_l \tag{3.8}$$

which can in turn be mapped to the joint torque as

$$\tau_k = d_k^T \, F_k \tag{3.9}$$

3.2 The Relation of Joint and Link Inertia Matrices

Having related joint and link velocities, accelerations, and forces in the above definitions, the next step in the derivation must relate the joint and link inertia matrices. This is best done by examining the kinetic energy, which is independent of the choice of variables, as are all types of energy. In particular, it can be expressed in both joint and link variables as

$$E_{kin} = \sum_{k=1}^{n} \sum_{j=1}^{n} \frac{1}{2} \dot{q}_k \, h_{kj} \, \dot{q}_j \qquad \text{or} \qquad E_{kin} = \sum_{i=1}^{n} \frac{1}{2} v_i^T I_i \, v_i \tag{3.10}$$

Substituting the definition for spatial velocity (3.2) then allows the kinetic energy to be rewritten as

$$\begin{aligned} E_{kin} &= \sum_{i=1}^{n} \frac{1}{2} \sum_{k=1}^{i} d_k^T \dot{q}_k \;\; I_i \; \sum_{j=1}^{i} d_j \dot{q}_j \\ &= \sum_{k=1}^{n} \sum_{j=1}^{n} \frac{1}{2} \dot{q}_k \left[d_k^T \sum_{i=\max(k,j)}^{n} I_i \, d_j \right] \dot{q}_j \end{aligned} \tag{3.11}$$

so that the inertia matrix elements h_{kj} can be written as

$$h_{kj} = d_k^T \sum_{i=\max(k,j)}^{n} I_i \;\, d_j \tag{3.12}$$

The above transformations, as will many others throughout the derivation, rely strongly on changes in the order of summation and on corresponding changes in the summation ranges. In particular, it is true that

$$\sum_{i=1}^{n} \sum_{k=1}^{i} = \sum_{k=1}^{n} \sum_{i=k}^{n} \tag{3.13}$$

as well as

$$\sum_{i=k}^{n} \sum_{j=1}^{i} = \sum_{j=1}^{n} \sum_{i=\max(k,j)}^{n} \tag{3.14}$$

These equalities are best verified geometrically, as in Figure 3-1, where each shaded element is included in the summation.

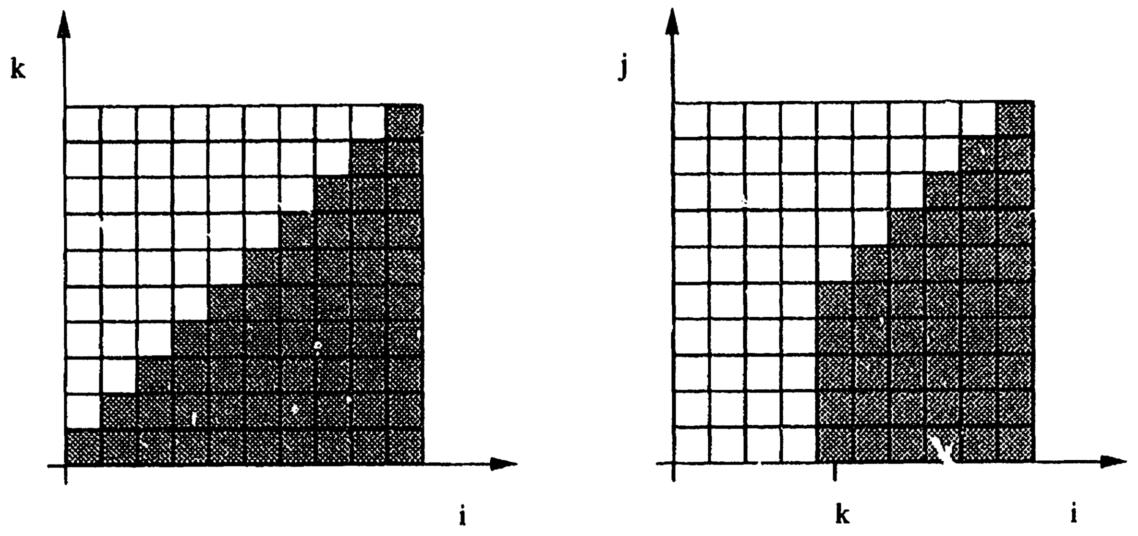


Figure 3-1: Geometric Interpretation of Summations

3.3 Expressing the Coriolis Matrix

The adaptive controller makes use of the relation between the Coriolis matrix C and the inertia matrix derivative $\dot{H}$ by noting the skew-symmetry of $(\dot{H} - 2C)$ or equivalently using $\dot{H} = C + C^T$. In addition, the following derivation uses an explicit expression for each element of the Coriolis matrix, which can be obtained by deriving the equation of motion with a Lagrangian approach [Slotine and Li, 1987e]. That is, Coriolis matrix is given by

$$c_{ki} = \frac{1}{2} \sum_{j=1}^{n} \left[\frac{\partial h_{ki}}{\partial q_j} + \frac{\partial h_{kj}}{\partial q_i} - \frac{\partial h_{ij}}{\partial q_k} \right] \dot{q}_j \tag{3.15}$$

Using this expression and the above result for the inertia matrix (3.12) it is also possible to derive a spatial expression for ‘he Coriolis matrix.

First, it is necessary to find partial differentiation rules for spatial joint axes and inertia matrices. Knowing that for a fixed reference frame the transformation matrices obey

$$\frac{\partial}{\partial q_i} X_k = \begin{cases} d_i \times X_k & \forall\ i \le k \\ 0 & \forall\ i > k \end{cases} \tag{3.16}$$

it then follows that

$$\frac{\partial}{\partial q_i} d_k = \begin{cases} d_i \times d_k & \forall\ i \le k \\ 0 & \forall\ i > k \end{cases} \tag{3.17}$$

$$\frac{\partial}{\partial q_i} I_k = \begin{cases} d_i \times I_k - I_k\, d_i \times & \forall\ i \le k \\ 0 & \forall\ i > k \end{cases} \tag{3.18}$$

The following derivation examines the three parts of c_{ki} according to Equation (3.15) first before connecting them into one larger equation. Thus the first part takes the form

$$\begin{aligned} \sum_{j=1}^{n} \frac{\partial h_{ki}}{\partial q_j} \dot{q}_j &= \sum_{j=1}^{n} \sum_{l=\max(k,i)}^{n} \frac{\partial}{\partial q_j} \left[d_k^T I_l d_i \right] \dot{q}_j \\ &= \sum_{j=1}^{n} \sum_{l=\max(k,i)}^{n} \dot{q}_j\, (d_j \times d_k)^T\, I_l\, d_i \qquad \text{if } j \le k \\ &+ \sum_{j=1}^{n} \sum_{l=\max(k,i)}^{n} \dot{q}_j\, d_k^T\, (d_j \times I_l - I_l d_j \times)\, d_i \qquad \text{if } j \le l \\ &+ \sum_{j=1}^{n} \sum_{l=\max(k,i)}^{n} \dot{q}_j\, d_k^T\, I_l\, d_j \times d_i \qquad \text{if } j \le i \end{aligned}$$

using the partial differential rules, as given above by Equations (3.17) and (3.18). The conditions can be incorporated in the summation boundaries for j and the summations can then be performed creating a spatial velocity.

$$\sum_{j=1}^{n} \frac{\partial h_{ki}}{\partial q_j} \dot{q}_j = - \sum_{l=\max(k,l)}^{n} \boldsymbol{d}_k^T \boldsymbol{v}_k \times \boldsymbol{I}_l \, \boldsymbol{d}_i + \sum_{l=\max(k,i)}^{n} \boldsymbol{d}_k^T \boldsymbol{v}_l \times \boldsymbol{I}_l \, \boldsymbol{d}_i$$

$$- \sum_{l=\max(k,i)}^{n} \boldsymbol{d}_k^T \boldsymbol{I}_l \, \boldsymbol{v}_l \times \boldsymbol{d}_i + \sum_{l=\max(k,i)}^{n} \boldsymbol{d}_k^T \boldsymbol{I}_l \, \boldsymbol{v}_i \times \boldsymbol{d}_i$$

The second part of Equation (3.15) can then be expressed as

$$\sum_{j=1}^{n} \frac{\partial h_{kj}}{\partial q_i} \dot{q}_j = \sum_{j=1}^{n} \sum_{l=\max(k,j)}^{n} \frac{\partial}{\partial q_i} \left[\boldsymbol{d}_k^T \boldsymbol{I}_l \boldsymbol{d}_j \right] \dot{q}_j$$

$$= \sum_{j=1}^{n} \sum_{l=\max(k,j)}^{n} \dot{q}_j \, (\boldsymbol{d}_i \times \boldsymbol{d}_k)^T \, \boldsymbol{I}_l \, \boldsymbol{d}_j \qquad \text{if } \; i \le k$$

$$+ \sum_{j=1}^{n} \sum_{l=\max(k,j)}^{n} \dot{q}_j \, \boldsymbol{d}_k^T \, (\boldsymbol{d}_i \times \boldsymbol{I}_l - \boldsymbol{I}_l \boldsymbol{d}_i \times) \, \boldsymbol{d}_j \qquad \text{if } \; i \le l$$

$$+ \sum_{j=1}^{n} \sum_{l=\max(k,j)}^{n} \dot{q}_j \, \boldsymbol{d}_k^T \, \boldsymbol{I}_l \, \boldsymbol{d}_i \times \boldsymbol{d}_j \qquad \text{if } \; i \le j$$

Changing the order of summation according to

$$\sum_{j=1}^{n} \sum_{l=\max(k,j)}^{n} = \sum_{l=k}^{n} \sum_{j=1}^{l}$$

as was done in Section 3.2 and then incorporating two of the conditions into the summations, noticing that $i \le j$ also implies $i \le l$, results in

$$\sum_{j=1}^{n} \frac{\partial h_{kj}}{\partial q_i} \dot{q}_j = - \sum_{l=k}^{n} \boldsymbol{d}_k^T \boldsymbol{d}_i \times \boldsymbol{I}_l \, \boldsymbol{v}_l \qquad \text{if } \; i \le k$$

$$+ \sum_{l=\max(k,i)}^{n} \boldsymbol{d}_k^T \, (\boldsymbol{d}_i \times \boldsymbol{I}_l - \boldsymbol{I}_l \boldsymbol{d}_i \times) \, \boldsymbol{v}_l$$

$$+ \sum_{l=\max(k,i)}^{n} \boldsymbol{d}_k^T \, \boldsymbol{I}_l \, \boldsymbol{d}_i \times (\boldsymbol{v}_l - \boldsymbol{v}_i)$$

$$\sum_{j=1}^{n} \frac{\partial h_{kj}}{\partial q_i} \dot{q}_j = - \sum_{l=\max(k,i)}^{n} d_k^T d_i \times I_l \, v_l \qquad \text{if } i \leq k$$

$$+ \sum_{l=\max(k,i)}^{n} d_k^T \, d_i \times I_l \, v_l \; - \sum_{l=\max(k,i)}^{n} d_k^T \, I_l \, d_i \times v_i$$

where the change of the summation range in the first sum has no immediate effect, but simplifies the further development.

Finally the third part of Equation (3.15) is skew-symmetric to the second part and can thus be written as

$$\sum_{j=1}^{n} - \frac{\partial h_{ij}}{\partial q_k} \dot{q}_j = \sum_{j=1}^{n} \sum_{l=\max(i,j)}^{n} \frac{\partial}{\partial q_k} \left[- d_i^T I_l d_j \right] \dot{q}_j$$

$$= \sum_{l=\max(k,i)}^{n} d_i^T d_k \times I_l \, v_l \qquad \text{if } k \leq i$$

$$- \sum_{l=\max(k,i)}^{n} d_i^T \, d_k \times I_l \, v_l \; + \sum_{l=\max(k,i)}^{n} d_i^T \, I_l \, d_k \times v_k$$

The spatial expression for an element c_{ki} of the C matrix can now be obtained by connecting the above three parts and canceling several terms thereof.

$$
\begin{aligned}
c_{ki} = \frac{1}{2} \sum_{l=\max(k,i)}^{n} \; & - d_k^T v_k \times I_l d_i + d_k^T v_l \times I_l d_i - d_k^T I_l v_l \times d_i \\
& + d_k^T I_l v_i \times d_i + d_k^T d_i \times I_l v_l - d_k^T I_l d_i \times v_i \\
& - d_i^T d_k \times I_l v_l + d_i^T I_l d_k \times v_k \\
& - d_k^T d_i \times I_l v_l \qquad \text{if } i \le k \\
& + d_i^T d_k \times I_l v_l \qquad \text{if } k \le i \\
= \sum_{l=\max(k,i)}^{n} \; & + \frac{1}{2} d_k^T v_l \times I_l d_i - \frac{1}{2} d_k^T I_l v_l \times d_i + \frac{1}{2} d_k^T d_i \times I_l v_l \\
& + d_k^T I_l v_i \times d_i
\end{aligned}
\tag{3.19}
$$

3.4 The Gravity Vector

An expression for the gravitational torques is quickly obtained when realizing that gravitational forces corresponds to a vertical upward acceleration of the complete system in a gravity-free environment. Therefore

$$
G_k = - d_k^T \sum_{i=k}^{n} I_i \, g_0 \tag{3.20}
$$

where g_0 is the gravitational acceleration pointing downward. In the implementation the gravity terms should also be lumped into a vertical acceleration, as to avoid additional calculations.

3.5 Feedforward of the Inertial Forces

This section computes the actual joint torques to be applied due to the inertial forces of the manipulator. In particular, substituting the above results (3.12), (3.19), and (3.20) into the control law yields

$$\begin{aligned}\tau_k &= \sum_{i=1}^{n} h_{ki}\,\ddot{q}_{r_i} + \sum_{i=1}^{n} c_{ki}\,\dot{q}_{r_i} + G_k \qquad (3.21)\\ &= d_k^T \sum_{l=k}^{n} I_l\,(-g_0)\\ &+ d_k^T \sum_{i=1}^{n} \sum_{l=\max(k,i)}^{n} I_l\,(d_i\,\ddot{q}_{r_i} + v_i \times d_i\,\dot{q}_{r_i})\\ &\qquad + \frac{1}{2}\,(I_l d_i \times v_l \dot{q}_{r_i} + v_l \times I_l d_i \dot{q}_{r_i} + d_i \times I_l v_l \dot{q}_{r_i})\end{aligned}$$

After changing the order and ranges of summations, similar to (3.14), the inertial torques can be written as

$$\tau_k = d_k^T\,F_k = d_k^T \sum_{l=k}^{n} f_l \qquad (3.22)$$

with the individual link forces f_l being calculated as

$$f_l = I_l\,(\dot{w}_l - g_0) + \frac{1}{2}\,v_l \times I_l\,w_l + \frac{1}{2}\,w_l \times I_l\,v_l + \frac{1}{2}\,I_l\,w_l \times v_l \qquad (3.23)$$

Or, splitting the effects of different parameters, that is using Equation (3.7), the inertial torques can also be expressed as

$$\tau_k = d_k^T \sum_{l=k}^{n} \sum_{i=1}^{10} f_l^i\,a_l^i \qquad (3.24)$$

with

$$f_l^i = R_l\,(\dot{w}_l - g_0) + \frac{1}{2}\,v_l \times R_l\,w_l + \frac{1}{2}\,w_l \times R_l\,v_l + \frac{1}{2}\,R_l\,w_l \times v_l \qquad (3.25)$$

3.6 Adaptation Law for Inertial Parameters

To implement the adaptation law given in Equation (2.6), it is necessary to notice that the matrix Y, as defined in (2.7), represents the coefficients of each torque with respect to all parameters. Remembering the above results, this implies a coefficient between torque τ_k and an inertial parameter a_l^i of

$$Y_{kl}^i = \begin{cases} d_k^T\,f_l^i & \forall\ k \le l \\ 0 & \forall\ k > l \end{cases} \qquad (3.26)$$

The product $Y^T s$ can then be computed as

$$(\,Y^T s\,)_l^i = \sum_{k=1}^{l} s_k\,d_k^T\,f_l^i = e_l^T\,f_l^i \qquad (3.27)$$

so that the complete adaptation law for the inertial parameters, given a diagonal gain matrix P, is

$$\dot{\hat{a}}_l^i = -\,P_l^i\,e_l^T\,f_l^i \qquad (3.28)$$

3.7 The Complete Recursive Algorithm

The key equations of the recursive algorithm are given by the local inertial force of each link (3.23) and the spatial version of the adaptation law (3.28). Together with the definitions of the spatial quantities and some joint level computations for friction compensation and P.D. feedback, they form the complete algorithm.

initialize : $\dot{w}_0 = -g_0$

upward :
$$
\begin{aligned}
v_k &= v_{k-1} + d_k \dot{q}_k \\
w_k &= w_{k-1} + d_k \dot{q}_{r_k} \\
\dot{w}_k &= \dot{w}_{k-1} + d_k \ddot{q}_{r_k} + v_{k-1} \times d_k \dot{q}_{r_k} \\
e_k &= v_k - w_k
\end{aligned}
$$

downward :
$$
\begin{aligned}
f_k^i &= \tfrac{1}{2} v_k \times R_i w_k \div \tfrac{1}{2} w_k \times R_i v_k + \tfrac{1}{2} R_i w_k \times v_k + R_i \dot{w}_k \\
F_k &= F_{k+1} + \sum_{i=1}^{10} f_k^i \, a_k^i \\
\tau_k &= d_k^T F_k + \hat{D}_{s_k} \operatorname{sgn}(\dot{q}_{r_k}) + \hat{D}_{v_k} \dot{q}_{r_k} - K_{D_k} s_k \\
\dot{\hat{a}}_k^i &= -\, P_k^i \, e_k^T f_k^i \\
\dot{\hat{D}}_{s_k} &= -\, P_k^s \, s_k \operatorname{sgn}(\dot{q}_{r_k}) \\
\dot{\hat{D}}_{v_k} &= -\, P_k^v \, s_k \, \dot{q}_{r_k}
\end{aligned}
$$

Table 3-1: Complete Equations for the Recursive Implementation of the Direct Adaptive Controller

The algorithm consists of two major parts. During the first, the kinematic structure is traversed from the base upwards to the tip, while computing link velocities and accelerations. Then second part then proceeds in reverse order downwards from the tip to the base, at each link determining the individual forces, the joint torque, and updating the parameter estimates. Table 3-1 summarizes all involved equations for the case of diagonal gain matrices $\boldsymbol{P}$ and $\boldsymbol{K}_D$.

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

- Using the Denavit-Hartenberg convention, 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 summed forces are expressed with respect to the frame beneath them.

- It is important to "customize" the algorithm (similarly to e.g. [Koshla and Kanade, 1985]), so as to avoid multiplications by or additions with zero. These occur mainly in the local force component calculations (due to the sparse placement matrices), which should be evaluated analytically, as is described in Appendix B. Also, several velocity and acceleration components may be restricted to zero (especially close to the base) and can thus be eliminated.

- Several parameters may be redundant, and therefore the algorithm can be optimized by using a minimal parameter set, as is detailed in Chapter 4.

- The gravitational forces are implemented as a vertical acceleration in gravity-free space, by setting the spatial acceleration of the base appropriately.

3.8 Comparison to Newton-Euler and Walker Algorithm

The major difference between the true implementation algorithm developed above and both the Newton-Euler and [Walker, 1988] algorithm lies in the computation of the local force components. For comparison, the equivalent equations of all three algorithms are given by

$$f_{l_{impl.}} = I_l\,(\dot{w}_l - g_0) + \frac{1}{2}\,v_l \times I_l\,w_l + \frac{1}{2}\,w_l \times I_l\,v_l + \frac{1}{2}\,I_l\,w_l \times v_l \tag{3.29}$$

$$f_{l_{N-E}} = I_l\,(\dot{w}_l - g_0) + v_l \times I_l\,v_l \tag{3.30}$$

$$f_{l_W} = I_l\,(\dot{w}_l - g_0) + w_l \times I_l\,w_l \tag{3.31}$$

Quite clearly, the derived algorithm for the actual implementation can be reduced to either algorithm, if reference and actual velocities are identical.

As a consequence of the above, a minor difference can also be found in the adaptation segment of the [Walker, 1988] algorithm.

3.9 Closed Kinematic Chains

The above developments were detailed in the context of manipulators with open kinematic chains. They remain, however, also valid for manipulators with closed kinematic chains.

To implement the algorithm for closed chains, imaginary cuts are placed at several joints to obtain an open but branched kinematic structure. The forces are then simply added at the branch points and the constraint equations are used to determine the torques for the motorized joints. Notice that the link numbering system has to be changed and therefore also the ranges of the summations throughout the algorithm have to be modified. They must be specified as sets of links lying above or below, as is appropriate. The approach of [Walker, 1988] for structuring closed chains and incorporating kinematic constraints can be used straightforwardly.

Chapter 4

Minimal Parameterization

The inertial and mass properties of an arbitrary rigid body are completely described by 10 parameters, all of which are included in the spatial inertia matrix. They consist of the mass, the location of center of mass, and the six traditional inertia values. However, when two rigid bodies are connected through some joint and their motion is restricted with respect to each other, not all 20 parameters are needed to describe the behavior of the connected system. Consequently, the dynamics of a rigid manipulator can be described by a reduced set of inertial parameters ([An, Atkeson, and Hollerbach, 1985], [Khosla and Kanade, 1985]).

This parameter redundancy has been studied analytically by [Mayeda, 1988] for robots with *rotational* joints, whose axes are either perpendicular or parallel. He concludes the minimum number of parameters necessary to describe the whole system to be 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.

Parameter redundancy is quite important for both an efficient implementation and the estimation of the parameter values. The following therefore studies parameter reductions in more detail, using the spatial notation to simplify the analysis. In particular, a system of two consecutive links is examined and a procedure is derived, to eliminate redundant parameters therein. Applying this procedure recursively to all connected links allows the elimination of all redundant parameters of the manipulator and simultaneously gives rules for the evaluation of the remaining parameters. The derivation is valid for arbitrary joint types and configurations, including both rotational and translational joints of an arbitrary intersection angles.

4.1 Parameter Redundancies for Two Consecutive Links

The following study focuses on two consecutive links $k-1$ and k connected by the particular joint, as is shown in Figure 4-1. The joint may be of any type and intersect other joint axes at any angle. The derived procedure allows a reduction of the original 20 inertial parameters, by transfering appropriate parameter values. That is, several parameter values can be set to zero, if other values are adjusted accordingly, without effecting the equations of motion.

In the following, the first link is assumed to have an arbitrary velocity and acceleration, that is it can move and turn in all directions. The case of restricted motions is a simple extension and is discussed later. The velocity of the second link can then determined using joint velocity and acceleration as

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1} + \boldsymbol{d}_k \dot{q}_k \tag{4.1}$$

$$\dot{\boldsymbol{v}}_k = \dot{\boldsymbol{v}}_{k-1} + \boldsymbol{d}_k \ddot{q}_k + \boldsymbol{v}_{k-1} \times \boldsymbol{d}_k \dot{q}_k \tag{4.2}$$

where $\boldsymbol{d}$ represents the joint axes, and $\boldsymbol{v}$, $\dot{\boldsymbol{v}}$ are the velocity and acceleration of the particular link.

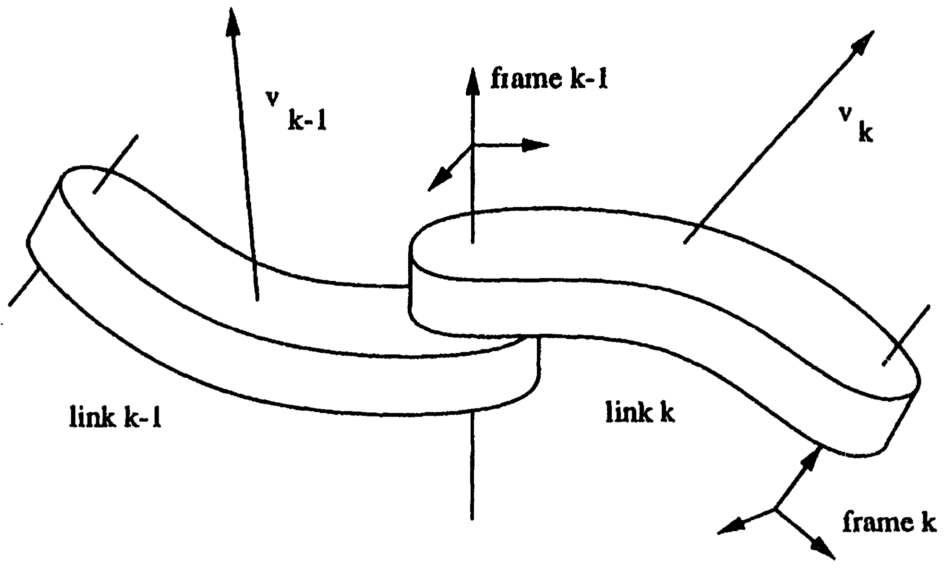


Figure 4-1: Two consecutive links connected by a rotational joint

To assure that the parameter changes have no effect on the equations of motion two conditions have to be satisfied. Given the velocities and accelerations both the total forces and the joint torque must remain unchanged. That is, while changing the parameter values the following must hold for any velocity and acceleration:

$$f_{k-1} + f_k = \text{constant} \tag{4.3}$$

$$d_k^T f_k = \text{constant} \tag{4.4}$$

where the individual link forces can be computed as

$$f_k = I_k \dot{v}_k + v_k \times I_k v_k \tag{4.5}$$

To evaluate these conditions the upper force f_k can be rewritten as a function of the lower link velocity and acceleration using Equations (4.1) and (4.2).

$$\begin{aligned} f_k = \; & I_k \dot{v}_{k-1} + v_{k-1} \times I_k v_{k-1} \\ & + I_k d_k \; \ddot{q}_k \\ & + (I_k v_{k-1} \times d_k + d_k \times I_k v_{k-1} + v_{k-1} \times I_k d_k) \; \dot{q}_k \\ & + d_k \times I_k d_k \; \dot{q}_k^2 \end{aligned} \tag{4.6}$$

To further study the parameter changes, an inertial variation matrix δI is introduced. This matrix is added to the inertia matrix of link k and subtracted from that of link $k-1$, as the total inertia matrix must remain constant. Substituting in the above equations, equivalent conditions can be found for this inertia variation.

$$\delta I \, d_k = 0 \tag{4.7}$$

$$\delta I \, v_{k-1} \times d_k + d_k \times \delta I \, v_{k-1} + v_{k-1} \times \delta I \, d_k = 0 \tag{4.8}$$

$$d_k \times \delta I \, d_k = 0 \tag{4.9}$$

$$d_k^T \, \delta I \, \dot{v}_{k-1} = 0 \tag{4.10}$$

$$d_k^T \, \delta I \, d_k = 0 \tag{4.11}$$

$$d_k^T \, v_{k-1} \times \delta I \, v_{k-1} = 0 \tag{4.12}$$

However these conditions are redundant and in particular Conditions (4.9), (4.10), and (4.11) follow directly from Condition (4.7), while Equation (4.8) automatically implies Equation (4.12). That is, all inertia variations δI, which satisfy the following two conditions, describe parameter changes that have no effect on the equations of motion.

$$\delta I \, d_k = 0 \tag{4.13}$$

$$\delta I \, v_{k-1} \times d_k + d_k \times \delta I \, v_{k-1} + v_{k-1} \times \delta I \, d_k = 0 \tag{4.14}$$

These conditions were derived with no assumptions on the manipulator structure and thus remain general for all joint types and configurations. To achieve further results a particular joint type has to be chosen, which specifies a value for the joint axis d_k. According to the Denavit-Hartenberg convention the joint axis is attached to frame $k-1$, so that the further analysis is best performed in this frame.

Evaluation of the above conditions can be done using Tables 4-1 and 4-2, which give the resulting spatial vector for unit values of v_{k-1} and d_k. Note the numbers enclosed in brackets represent the values of the specified parameter, i.e. (3) is the value of parameter 3 (representing the z-axis inertia). They clearly show the dependence on the different inertia parameters. Given the value of d_k, these tables then allow all necessary parameters and parameter combinations to be identified. The remaining parameters and parameter combinations can therefore be included in the allowable set of inertia variations δI.

Using δI it is now possible to eliminate parameters by subtracting their value from link $k-1$ and adding them to link k. This suggests the interpretation of *transfering* parameters from link $k-1$ to link k, which can also be inverted, i.e. parameters can be transfered from link k to link $k-1$. To transfer the parameters, however, the inertia variations δI has to undergo a reference frame transformation from frame $k-1$, in which it was derived, to frame k. Tables 4-3 and 4-4 describe this transformation for both upward and downward directions.

d_{k_1}	d_{k_2}	d_{k_3}	d_{k_4}	d_{k_5}	d_{k_6}
.	+(9)	−(8)	+(10)	.	.
−(9)	.	+(7)	.	+(10)	.
+(8)	−(7)	.	.	.	+(10)
+(1)	+(4)	+(5)	.	−(9)	+(8)
+(4)	+(2)	+(6)	+(9)	.	−(7)
+(5)	+(6)	+(3)	−(8)	+(7)	.

Table 4-1: Evaluation of $\delta I\, d_k$

	d_{k_1}	d_{k_2}	d_{k_3}	d_{k_4}	d_{k_5}	d_{k_6}
v_{k-1_1}	. −2(8) −2(9) . −2(5) +2(4)	. +2(7) . +2(5) . +(3)+(2)−(1)	. . +2(7) −2(4) −(3)−(2)+(1) .	. . . . . .	. . +2(10) +2(8) −2(7) .	. −2(10) . +2(9) . −2(7)
v_{k-1_2}	+2(8) . . . −2(6) −(3)+(2)−(1)	−2(7) . −2(9) +2(6) . −2(4)	. . +2(8) +(3)−(2)+(1) +2(4) .	. . −2(10) −2(8) +2(7) .	. . . . . .	+2(10) . . . +2(9) −2(8)
v_{k-1_3}	+2(9) . . . −(3)+(2)+(1) +2(6)	. +2(9) . +(3)−(2)−(1) . +2(5)	−2(7) −2(8) . −2(6) −2(5) .	. +2(10) . −2(9) . +2(7)	−2(10) . . . −2(9) +2(8)	. . . . . .
v_{k-1_4}	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .
v_{k-1_5}	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .
v_{k-1_6}	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .

Table 4-2: Evaluation of $v_{k-1} \times \delta I\, d_k \; + \; d_k \times \delta I\, v_{k-1} \; + \; \delta I\, v_{k-1} \times d_k$

	$J_{xx}\|^{k-1}$	$J_{yy}\|^{k-1}$	$J_{zz}\|^{k-1}$	$J_{xy}\|^{k-1}$	$J_{xz}\|^{k-1}$	$J_{yz}\|^{k-1}$	$mr_x\|^{k-1}$	$mr_y\|^{k-1}$	$mr_z\|^{k-1}$	$m\|^{k-1}$
$J_{xx}\|^k$	$c\theta^2$	$s\theta^2$	0	$2s\theta c\theta$	0	0	0	0	$-2d$	d^2
$J_{yy}\|^k$	$s\theta^2 c\alpha^2$	$c\theta^2 c\alpha^2$	$s\alpha^2$	$-2s\theta c\theta c\alpha^2$	$-2s\theta s\alpha c\alpha$	$2c\theta s\alpha c\alpha$	$-2ac\theta$ $-2ds\theta s\alpha c\alpha$	$-2as\theta$ $+2dc\theta s\alpha c\alpha$	$-2dc\alpha^2$	$a^2+d^2c\alpha^2$
$J_{zz}\|^k$	$s\theta^2 s\alpha^2$	$c\theta^2 s\alpha^2$	$c\alpha^2$	$-2s\theta c\theta s\alpha^2$	$2s\theta s\alpha c\alpha$	$-2c\theta s\alpha c\alpha$	$-2ac\theta$ $+2ds\theta s\alpha c\alpha$	$-2as\theta$ $-2dc\theta s\alpha c\alpha$	$-2ds\alpha^2$	$a^2+d^2s\alpha^2$
$J_{xy}\|^k$	$-s\theta c\theta c\alpha$	$s\theta c\theta c\alpha$	0	$-(s\theta^2-c\theta^2)c\alpha$	$c\theta s\alpha$	$s\theta s\alpha$	$-as\theta c\alpha$ $+dc\theta s\alpha$	$+ac\theta c\alpha$ $+ds\theta s\alpha$	$as\alpha$	$-ads\alpha$
$J_{xz}\|^k$	$s\theta c\theta s\alpha$	$-s\theta c\theta s\alpha$	0	$(s\theta^2-c\theta^2)s\alpha$	$c\theta c\alpha$	$s\theta c\alpha$	$+as\theta s\alpha$ $+dc\theta c\alpha$	$-ac\theta s\alpha$ $+ds\theta c\alpha$	$ac\alpha$	$-adc\alpha$
$J_{yz}\|^k$	$-s\theta^2 s\alpha c\alpha$	$-c\theta^2 s\alpha c\alpha$	$s\alpha c\alpha$	$2s\theta c\theta s\alpha c\alpha$	$s\theta(s\alpha^2-c\alpha^2)$	$-c\theta(s\alpha^2-c\alpha^2)$	$ds\theta(s\alpha^2-c\alpha^2)$	$-dc\theta(s\alpha^2-c\alpha^2)$	$2ds\alpha c\alpha$	$-d^2s\alpha c\alpha$
$mr_x\|^k$	0	0	0	0	0	0	$c\theta$	$s\theta$	0	$-a$
$mr_y\|^k$	0	0	0	0	0	0	$-s\theta c\alpha$	$c\theta c\alpha$	$s\alpha$	$-ds\alpha$
$mr_z\|^k$	0	0	0	0	0	0	$s\theta s\alpha$	$-c\theta s\alpha$	$c\alpha$	$-dc\alpha$
$m\|^k$	0	0	0	0	0	0	0	0	0	1

Table 4-3: Upwards Transformation of a Spatial Inertia Matrix

	$J_{xx}\|^k$	$J_{yy}\|^k$	$J_{zz}\|^k$	$J_{xy}\|^k$	$J_{xz}\|^k$	$J_{yz}\|^k$	$mr_x\|^k$	$mr_y\|^k$	$mr_z\|^k$	$m\|^k$
$J_{xx}\|^{k-1}$	$c\theta^2$	$s\theta^2c\alpha^2$	$s\theta^2s\alpha^2$	$-2s\theta c\theta c\alpha$	$2s\theta c\theta s\alpha$	$-2s\theta^2s\alpha c\alpha$	$2as\theta^2$	$+2as\theta c\theta c\alpha$ $+2ds\alpha$	$-2as\theta c\theta s\alpha$ $+2dc\alpha$	$a^2s\theta^2+d^2$
$J_{yy}\|^{k-1}$	$s\theta^2$	$c\theta^2c\alpha^2$	$c\theta^2s\alpha^2$	$2s\theta c\theta c\alpha$	$-2s\theta c\theta s\alpha$	$-2c\theta^2s\alpha c\alpha$	$2ac\theta^2$	$-2as\theta c\theta c\alpha$ $+2ds\alpha$	$+2as\theta c\theta s\alpha$ $+2dc\alpha$	$a^2c\theta^2+d^2$
$J_{zz}\|^{k-1}$	0	$s\alpha^2$	$c\alpha^2$	0	0	$2s\alpha c\alpha$	$2a$	0	0	a^2
$J_{xy}\|^{k-1}$	$s\theta c\theta$	$-s\theta c\theta c\alpha^2$	$-s\theta c\theta s\alpha^2$	$-(s\theta^2-c\theta^2)c\alpha$	$(s\theta^2-c\theta^2)s\alpha$	$2s\theta c\theta s\alpha c\alpha$	$-2as\theta c\theta$	$a(s\theta^2-c\theta^2)c\alpha$	$-a(s\theta^2-c\theta^2)s\alpha$	$-a^2s\theta c\theta$
$J_{xz}\|^{k-1}$	0	$-s\theta s\alpha c\alpha$	$s\theta s\alpha c\alpha$	$c\theta s\alpha$	$c\theta c\alpha$	$s\theta(s\alpha^2{-}{-}c\alpha^2)$	$-dc\theta$	$-ac\theta s\alpha$ $+ds\theta c\alpha$	$-ac\theta c\alpha$ $-ds\theta s\alpha$	$-adc\theta$
$J_{yz}\|^{k-1}$	0	$c\theta s\alpha c\alpha$	$-c\theta s\alpha c\alpha$	$s\theta s\alpha$	$s\theta c\alpha$	$-c\theta(s\alpha^2-c\alpha^2)$	$-ds\theta$	$-as\theta s\alpha$ $-dc\theta c\alpha$	$-as\theta c\alpha$ $+dc\theta s\alpha$	$-ads\theta$
$mr_x\|^{k-1}$	0	0	0	0	0	0	$c\theta$	$-s\theta c\alpha$	$s\theta s\alpha$	$ac\theta$
$mr_y\|^{k-1}$	0	0	0	0	0	0	$s\theta$	$c\theta c\alpha$	$-c\theta s\alpha$	$as\theta$
$mr_z\|^{k-1}$	0	0	0	0	0	0	0	$s\alpha$	$c\alpha$	d
$m\|^{k-1}$	0	0	0	0	0	0	0	0	0	1

Table 4-4: Downwards Transformation of a Spatial Inertia Matrix

4.2 Parameter Transfers through Rotational Joints

This section turns its attention to the particular case of rotational joints. However, the intersection angle between joint axes still remains arbitrary. Rotational joints are described by a unit rotation along the z-axis, so that

$$d_k = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.15}$$

Therefore, assuming an unrestricted ν_{k-1} and using Tables 4-1 and 4-2, the equation of motion depends on the following parameters and combinations.

$$(1)-(2) \quad (3) \quad (4) \quad (5) \quad (6) \quad (7) \quad (8) \tag{4.16}$$

The inertia variation thus has three degrees of freedom and contains

$$\begin{aligned} \delta I\,|^{k-1} &= \{\ (1)=(2)\ ,\ (9)\ ,\ (10)\ \} \\ &= \{\ (\ \delta J_{xx} = \delta J_{yy}\ )\ ,\ \delta p_z\ ,\ \delta m\ \} \end{aligned} \tag{4.17}$$

To eliminate and transfer parameters between links, the parameter values must undergo the reference frame transformation, given for rotational joint in Table 4-5. The Denavit-Hartenberg parameters (a,α,d,θ) correspond to joint *k*.

It is interesting to note that the parameter reductions often have physical interpretations. In this example the parameters correspond to a thin rod (i.e. having no inertia in that direction) along the rotational axis, which can belong to either of the attached links.

$$\delta J_{xx}\,|^{k-1} = J$$

$$\delta J_{yy}\,|^{k-1} = J$$

$$\delta p_z\,|^{k-1} = p_z$$

$$\delta m\,|^{k-1} = m$$

$$\delta J_{xx}\,|^{k} = J + m\,d^2 - 2\,d\,p_z$$

$$\delta J_{yy}\,|^{k} = (J + m\,d^2 - 2\,d\,p_z)\cos^2(\alpha) + m\,a^2$$

$$\delta J_{zz}\,|^{k} = (J + m\,d^2 - 2\,d\,p_z)\sin^2(\alpha) + m\,a^2$$

$$\delta J_{xy}\,|^{k} = (a\,p_z - m\,a\,d)\sin(\alpha)$$

$$\delta J_{xz}\,|^{k} = (a\,p_z - m\,a\,d)\cos(\alpha)$$

$$\delta J_{yz}\,|^{k} = -(J + m\,d^2 - 2\,d\,p_z)\sin(\alpha)\cos(\alpha)$$

$$\delta p_x\,|^{k} = -m\,a$$

$$\delta p_y\,|^{k} = (p_z - m\,d)\sin(\alpha)$$

$$\delta p_z\,|^{k} = (p_z - m\,d)\cos(\alpha)$$

$$\delta m\,|^{k} = m$$

Table 4-5: Transformation of the inertia parameters for rotational joints

4.3 Parameter Transfers through Translational Joints

Translational joints are characterized by a unit translational vector along the z-axis. Therefore

$$d_k = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.18}$$

As in the previous example using the Tables 4-1 and 4-2, the equation of motion depends on the following parameters and combinations.

$$(7) \quad (8) \quad (9) \quad (10) \tag{4.19}$$

The inertia variation thus has six degrees of freedom and contains

$$\begin{aligned} \delta I\,|^{k-1} &= \{\ (1)\ ,\ (2)\ ,\ (3)\ ,\ (4)\ ,\ (5)\ ,\ (6)\ \} \\ &= \{\ \delta J_{xx}\ ,\ \delta J_{yy}\ ,\ \delta J_{zz}\ ,\ \delta J_{xy}\ ,\ \delta J_{xz}\ ,\ \delta J_{yz}\ \} \end{aligned} \tag{4.20}$$

The reference frame transformation for these particular parameters is given in Table 4-6.

4.4 Parameter Reductions for Links with Restricted Motion

For links with restricted motion, as for example the base link, many more parameters can be eliminated. The analysis should proceed as before, however ignoring any conditions arising from the restricted directions. Also, only forces in the unrestricted directions need to be examined. As such situations only arise with very simple kinematics, the analysis is typically straight-forward. For example, a base rotating about a vertical axis only requires the corresponding rotational inertia as parameters.

$$
\begin{aligned}
\delta J_{xx}\,|^{k-1} &= J_{xx} \\
\delta J_{yy}\,|^{k-1} &= J_{yy} \\
\delta J_{zz}\,|^{k-1} &= J_{zz} \\
\delta J_{xy}\,|^{k-1} &= J_{xy} \\
\delta J_{xz}\,|^{k-1} &= J_{xz} \\
\delta J_{yz}\,|^{k-1} &= J_{yz}
\end{aligned}
$$

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

Table 4-6: Transformation of the inertia parameters for translational joints

Chapter 5

Cartesian Implementation

The previous chapters analyzed the efficient implementation of a joint-space adaptive controller. Many tasks, however, are best described in terms of the end-effector motion or behavior. While it is possible to perform the inverse kinematics on a desired end-effector trajectory a-priori and thus to use a joint-space controller to achieve specified end-effector motions, such a procedure prevents the implementation of an arbitrary end-effector behavior. Therefore, tasks that require interactions with the end-effector also necessitate a controller capable of utilizing cartesian data directly as an input (e.g., [Luh, Walker, and Paul, 1980], [Khatib, 1980, 1983], in the non-adaptive case). This chapter discusses the problems of converting and extending the joint-space 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.

5.1 On-Line Inverse Kinematics

Taking a closer look at the joint-space controller, that is examing the control and adaptation laws (2.5) and (2.6), it 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 (2.3) which guarantees the actual convergence to the desired trajectory. Since in the case of cartesian motion a desired joint trajectory is not given, it is possible to redefine the joint reference variables in terms of its cartesian counterparts. More precisely, with cartesian reference velocity and acceleration defined as

$$\dot{x}_r = \dot{x}_d - \Lambda \tilde{x} \tag{5.1}$$

$$\ddot{x}_r = \ddot{x}_d - \Lambda \dot{\tilde{x}} \tag{5.2}$$

where $\tilde{x} = x - x_d$ and Λ is again a s.p.d. matrix, the joint versions can be defined using the manipulator Jacobian J.

$$\dot{x}_r = J \; \dot{q}_r \tag{5.3}$$

$$\ddot{x}_r = J \; \ddot{q}_r + \dot{J} \; \dot{q}_r \tag{5.4}$$

For non-redundant manipulators, the joint reference velocity and acceleration are then obtained by inverting the Jacobian.

$$\dot{q}_r = J^{-1} \; \dot{x}_r \tag{5.5}$$

$$\ddot{q}_r = J^{-1} (\ddot{x}_r - \dot{J} \, \dot{q}_r) \tag{5.6}$$

This simple extension then provides a complete cartesian adaptive controller, which guarantees convergence of cartesian motion to the desired trajectory, similarly to the joint-space case, as long as singularities are avoided.

Note that the actual inverse kinematic solution for desired joint position is never computed directly but rather determined by the dynamics of the system. It can, however, be retrieved for other purposes by exploiting the filter structure of the reference velocity definition. Namely, implementing a filter of the form

$$\dot{q}_d + \lambda\, q_d = \dot{q}_r + \lambda\, q \tag{5.7}$$

will provide the exact desired joint trajectory. This also remains valid for redundant manipulators.

Also, to provide a more direct feedback or to achieve a certain behavior, it is also possible to add an explicit cartesian P.D. control component of the form

$$\tau = -J^T K_D\,(\dot{x} - \dot{x}_r) \tag{5.8}$$

Finally, note that many representations can be used for the end-effector orientation. For instance, defining the end-effector orientation as in [Luh, Walker, and Paul, 1980], with (n , o , a) representing the actual orientation vector triplet, (n_d , o_d , a_d) representing the desired orientation vector triplet, and ω 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

$$\dot{x}_r^{\text{orient}} = \omega_d + \frac{\lambda}{2}\,(\, n \times n_d + o \times o_d + a \times a_d \,) \tag{5.9}$$

guarantees that the orientation error and the angular velocity error both go to zero (with $\dot{x} - \dot{x}_r$). 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.

5.2 Redundancy Solution

For redundant manipulators the inverse Jacobian J^{-1}, as was used above, is no longer uniquely defined. To resolve this non-uniqueness, a pseudo-inverse J^+ can be used instead, which automatically minimizes the joint velocities corresponding to any cartesian velocities. In addition, the nullspace, i.e. the space of all joint motions which produce no cartesian motion, can be used to minimize any performance index.

Similar to work done by [Klein and Huang, 1983], [Klein and Blaho, 1987], the performance index is chosen to be a quadratic norm of joint positions. Such an approach is computationally efficient and, intuitively, mimics a "flexible beam" clamped to the desired endpoint 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 pseudo-inverse methods.

In contrast to other controllers, sliding controllers require both reference velocity and acceleration, so that besides the pseudo-inverse (or generalized inverse) J^+ of the Jacobian J, its time-derivative is also needed. Assuming a Jacobian of linearly independent rows, that is a Jacobian describing a redundant manipulator outside of singularities, we can write explicit expressions for both as

$$J^+ = J^T \, (J J^T)^{-1} \tag{5.10}$$

$$\dot{J}^+ = -J^+ \dot{J} J^+ + (I - J^+ J)\, \dot{J}^T (J J^T)^{-1} \tag{5.11}$$

With these results the inverse kinematic transformation for a redundant manipulator can be specified as

$$\dot{q}_r = J^+ \dot{x}_r + (1 - J^+ J) \psi \tag{5.12}$$

$$\ddot{q}_r = J^+ (\ddot{x}_r - \dot{J} \dot{q}_r) + (1 - J^+ J) (\dot{\psi} + \dot{J}^T J^{+T} (\dot{q}_r - \psi)) \tag{5.13}$$

where ψ is an arbitrary, but continuous, joint-space vector, which the joint velocity attempts to track within the null-space. It is used to minimize the performance index and thus is set proportional to the negative gradient thereof. That is

$$\psi = - \lambda \nabla f = - \lambda q \quad , \quad \dot{\psi} = - \lambda \dot{q} \tag{5.14}$$

in the case of $f = \sum_i \frac{1}{2} q_i^2$

5.3 Implementation Aspects of Inverse Kinematics

Implementing the kinematic transformations directly as written above can be computationally intensive. Therefore, after recursively computing the Jacobian and its derivative, the pseudo-inverse should be obtained using an orthogonalization algorithm equivalent to Gramm-Schmidt's. That is, if the Jacobian J is decomposed into a lower-triangular matrix R and a row-orthogonal matrix Q as

$$J = R \, Q \tag{5.15}$$

the pseudo-inversion is then achieved by transposing Q and using backsubstitution to invert R.

$$J^+ = Q^T R^{-1} \tag{5.16}$$

The pseudo-inverse J^+ is never explicitly computed, but rather substituted directly into the transformation equations, resulting in

$$\dot{q}_r = \psi + Q^T \left(R^{-1} \dot{x}_r - Q \psi \right) \tag{5.17}$$

$$\ddot{q}_r = p + Q^T \left(R^{-1} (\ddot{x}_r - \dot{J} \dot{q}_r) - Q p \right) \tag{5.18}$$

with

$$p = \dot{\psi} + \dot{J}^T R^{-T} Q \left(\dot{q}_r - \psi \right) \tag{5.19}$$

where again none of the matrix products should be computed explicitly.

Chapter 6

Applications to Constrained Motion

Many useful tasks involve some interaction with a fixed environment. Controlling the end-effector impedance is an attractive option for such tasks (e.g., [Hogan, 1985]), since it imitates a passive mechanism and thus yields stable interaction with any passive environment. However, using impedance control for tracking substantially limits the overall performance, as it does not compensate for the nonlinear dynamics of a rigid manipulator, as is the case for all P.D. controllers.

This chapter introduces impedance control features within the framework of adaptive control. This allows the application to constrained motion, while preserving the performance improvements achieved by adaptive control. No model of the environment is assumed, besides the passivity thereof, and no force measurements are required. Only position and velocity signals are needed for feedback.

6.1 The Basic Passivity Concept

Passivity theory is a mathematical formalization of the physically intuitive concepts of power and energy. It defines the scalar product between input $\boldsymbol{u}$ and output $\boldsymbol{v}$ of a particular system to be the "power" entering the system. If, in addition, there exists a lower bounded "system-energy" function E and a non-negative "power dissipation" function P_{diss}, which obey

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \boldsymbol{u}^T\, \boldsymbol{v} - P_{diss} \tag{6.1}$$

the system is called passive. A strictly passive system must have a positive power dissipation, as long as the system-energy has not reached its lower bound.

As a consequence of the above, a passive system with no input is stable, because no further energy can be obtained, and a strictly passive system is asymptotically stable, because energy is constantly lost. This corresponds to a Lyapunov-type argument and thus passive systems are guaranteed stability without any explicit analysis of possibly complicated dynamics.

A mayor advantage of passivity theory is its ability to connect different passive subsystems into an overall passive system. This closure property allows many small and well understood components to interact without worrying about complicated dynamics caused by the interaction. Connections, however, can not be chosen completely arbitrarily but must take either a negative feedback or parallel type structure, as shown in Figure 6-1.

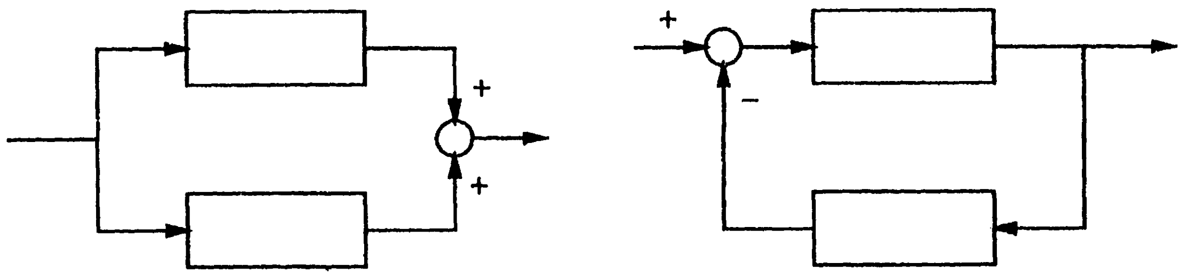


Figure 6-1: Negative feedback and parallel connections maintaining passivity

6.2 Passivity Interpretation of the Adaptive Controller

As noticed by many researchers (e.g., [Ortega and Spong, 1988], [Landau and Horowitz, 1988], [Kelly and Carelli, 1988]), the Lyapunov-like derivation of the adaptive manipulator controller of [Slotine and Li, 1986] can easily be translated in terms of passivity arguments. This offers several advantages, such as the ease of adding new subsystems, of which the following sections will make use.

A rigid manipulator itself obeys the conservation of energy, and thus generates a passive mapping between input torques and output velocities. This mapping itself, however, can also be viewed as a composition of two passive sub-parts, corresponding to the kinetic and potential energy of the manipulator. These two subsystems are then connected in a feedback configuration, as in Figure 6-2, verifying the passivity of the overall manipulator. Equivalently, the passivity can be shown by

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[E_{pot} + \frac{1}{2}\dot{q}^T H \dot{q} \right] = \nabla E_{pot}\, \dot{q} + \dot{q}^T H \ddot{q} + \frac{1}{2}\, \dot{q}^T \dot{H} \dot{q} = \dot{q}^T \tau \tag{6.2}$$

since $\dot{H} = C + C^T$. Note, that actual gravitational forces are given by $(-G)$. If friction forces are included, these can again be interpreted in their own feedback block, and the complete system is then strictly passive.

After introducing the reference velocity, the exact (ideal) feedforward component of

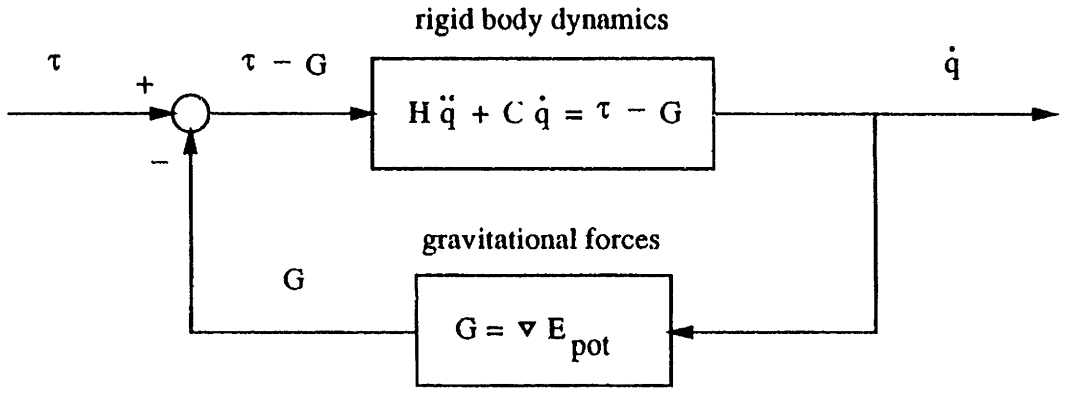


Figure 6-2: Passive Interpretation of a rigid manipulator

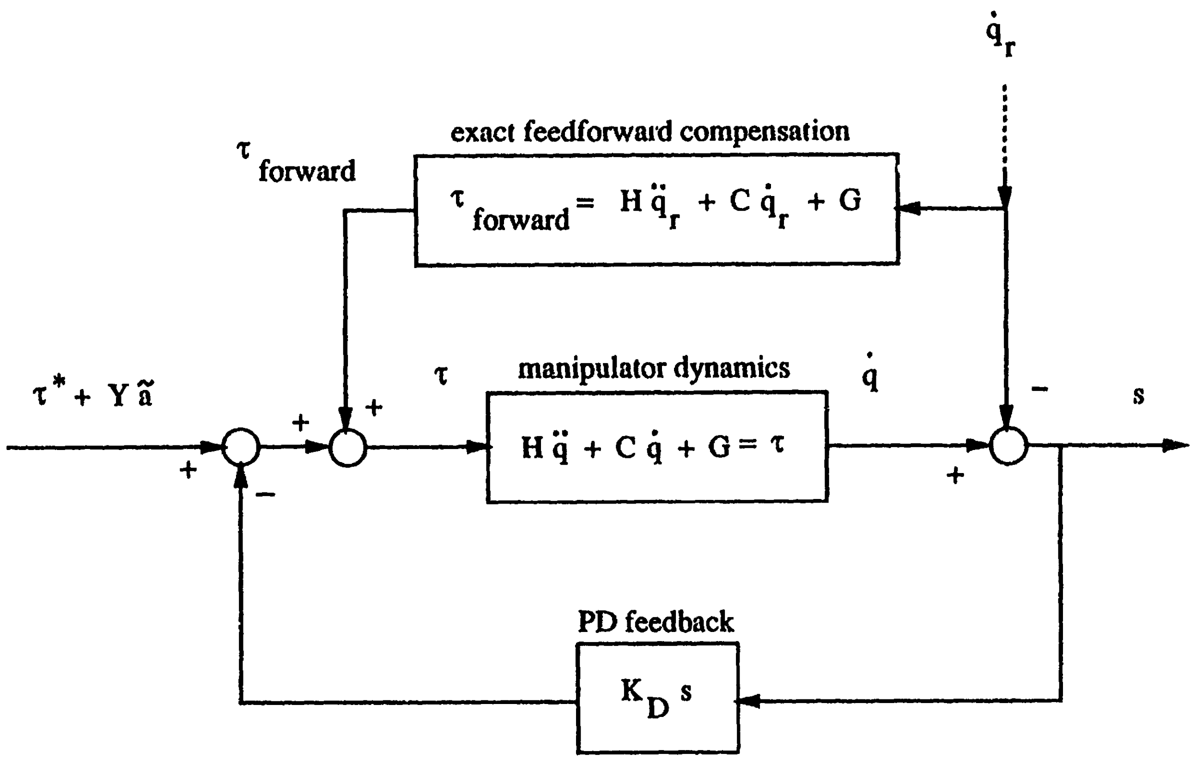
the control law (2.5) then modifies input and output variables of this mapping. That is, a passive mapping is created between any additional torque inputs and reference velocity error $s = \dot{q} - \dot{q}_r$. Due to the parametric uncertainty, the additional torque inputs correspond not only to additional controller torques or external forces, denoted by τ^*, but also to errors $Y\tilde{a}$ in the feedforward compensation. The P.D. component of the control law, in turn, adds a dissipative element to the system. As Figure 6-3 illustrates, the closed-loop system thus represents a dissipative mapping. From the Lyapunov analysis, the control law indeed yields

$$\frac{d}{dt}\left[\frac{1}{2}s^T H s\right] = s^T\ (\ \tau^* + Y\tilde{a}\ ) - s^T K_D\, s \tag{6.3}$$

Furthermore, using the adaptation law (2.6) corresponds to inserting a passive feedback block between s and $(-Y\tilde{a})$, since

$$\frac{d}{dt}\left[\frac{1}{2}\tilde{a}^T P^{-1}\tilde{a}\right] = -\ s^T\ Y\tilde{a} \tag{6.4}$$

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



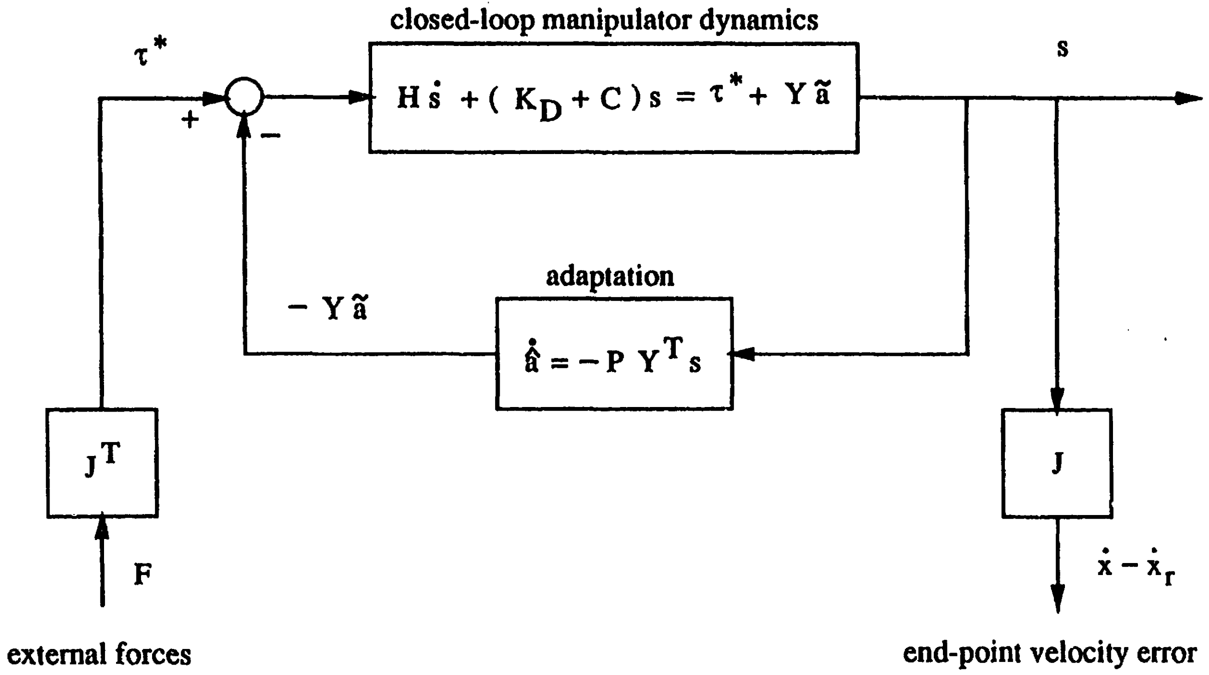
$\dot{q}_r$
exact feedforward compensation
$\tau_{forward}$
$\tau_{forward} = H\ddot{q}_r + C\dot{q}_r + G$
$\tau^* + Y\tilde{a}$
τ
manipulator dynamics
$\dot{q}$
$H\ddot{q} + C\dot{q} + G = \tau$
s
+
+
+
-
+
-
PD feedback
$K_D\, s$

Figure 6-3: Open versus closed loop passive mapping of the manipulator

$$\dot{\tilde{a}} = \dot{\hat{a}} = -\,P\,Y^T s \tag{6.5}$$

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

Passivity results verify, that the complete system, as shown in Figure 6-4, remains passive and hence globally stable. They further allow new passive blocks to be added to the system, while preserving the overall stability and convergence. This property is used to analyze the interactions with an environment and to combine the adaptive controller with an impedance controller. Note, that changes of variables does not affect passivity, so that a corresponding coordinate change may be performed simultaneously on both input and output of a passive system. That is, the above joint-space system can also be connected to cartesian subsystems after appropriate Jacobian transformations. This is easily shown by noting that the power flow to any cartesian subsystems is given as



closed-loop manipulator dynamics
τ^*
+
−
$H\dot{s} + (K_D + C)s = \tau^* + Y\tilde{a}$
s
adaptation
$-Y\tilde{a}$
$\dot{\hat{a}} = -PY^T s$
J^T
J
F
$\dot{x} - \dot{x}_r$
external forces
end-point velocity error

Figure 6-4: Passivity interpretation of the adaptive controller

$$\tau^{*T} s = (J^T F)^T s = F^T (J\dot{s}) = F^T (\dot{x} - \dot{x}_r) \tag{6.6}$$

6.3 Adaptive impedance control

An arbitrary passive environment can be interpreted as a passive mapping between contact forces and displacement velocity, since it can only provide a finite amount of energy. To account for the adaptive controller contacting such an environment, an additional element must be added to the above passive system, representing the contact forces. However, the environment's passivity is not guaranteed between the cartesian tracking error $(\dot{x} - \dot{x}_r)$ and contact forces in general, reflecting the motion constraint imposed by the environment. Therefore, it is necessary to redefine the cartesian reference velocity $\dot{x}_r$ so as 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 set to zero. That is

$$\dot{x}_{r_i} = \begin{cases} \dot{x}_{d_i} - \lambda\,\tilde{x}_i & \parallel \text{ surface} \\ 0 & \perp \text{ surface} \end{cases} \tag{6.7}$$

or if $\boldsymbol{n}$ represents the surface normal

$$\dot{x}_r = (I - n\,n^T)\,(\dot{x}_d - \Lambda\,\tilde{x}) \tag{6.8}$$

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

$$F_{imp} = -\,K\,(x - x_0)_{\perp} \tag{6.9}$$

Also, a nonlinear impedance can be chosen, for example by saturating the spring at a given value to achieve a constant contact force.

As impedance controllers are also passive (from endpoint force to velocity), the whole system, as illustrated in Figure 6-5, is stable, and combines the advantages of its individual components, that is adaptation, precise tracking, and stable contact. Furthermore, only position and velocity feedback is used, and thus no force measurements are required

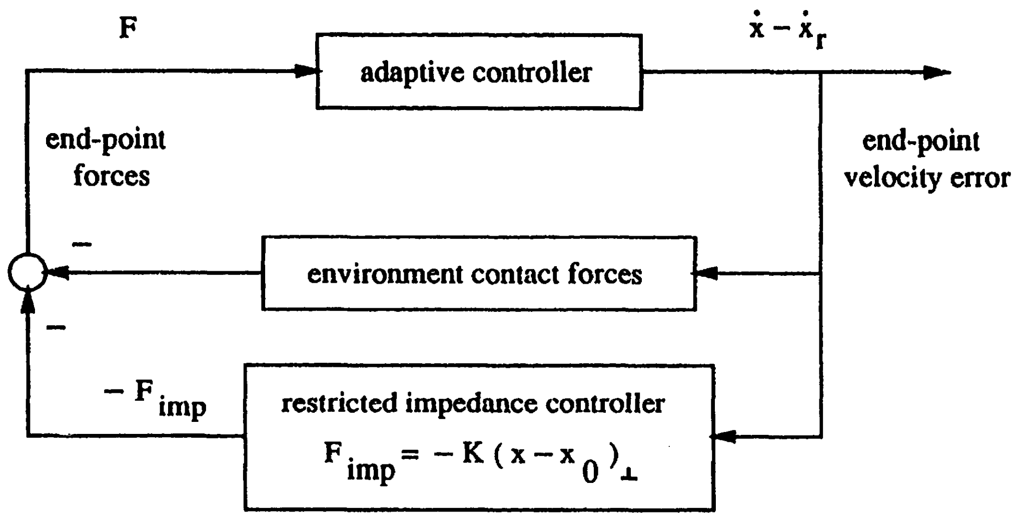


Figure 6-5: Complete control system for constrained motion

To implement this 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 constrained 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.

Chapter 7

Experimental Results

The recursive implementation of the adaptive controller as well as the on-line inverse kinematics were thoroughly tested on a 4 degree-of-freedom cable-driven "whole-arm" manipulator (WAM) designed at the M.I.T. Artificial Intelligence Laboratory [Townsend, 1988], [Salisbury, *et al.*, 1988]. This manipulator is capable of achieving high speeds and thus requires precise control to achieve accurate tracking performance. Furthermore, it has a very efficient force transmission between motors and links, so that open-loop force control can provide good results and circumvent problems associated with direct foice feedback.

7.1 The Experimental Setup

The 4 degree-of-freedom whole-arm manipulator, as is shown in Figure 7-1, has a geometry comparable to the human arm with an extended length of approximately 1 meter. It is powered by four pulse-width modulated motors, capable of delivering a maximum of 1.5 Nm each. The motors are located close to the base and are connected to the lightweight links via two-stage cable transmissions incorporating velocity reductions between 1:20 and 1:30. To maintain high transmission stiffness, these reductions are performed close to the joints. Position measurements are obtained at the motor shaft using 12 bit resolvers. While the joints have a range between ± 90° and ± 135°, the joint velocities without payloads can exceed 720° per second.

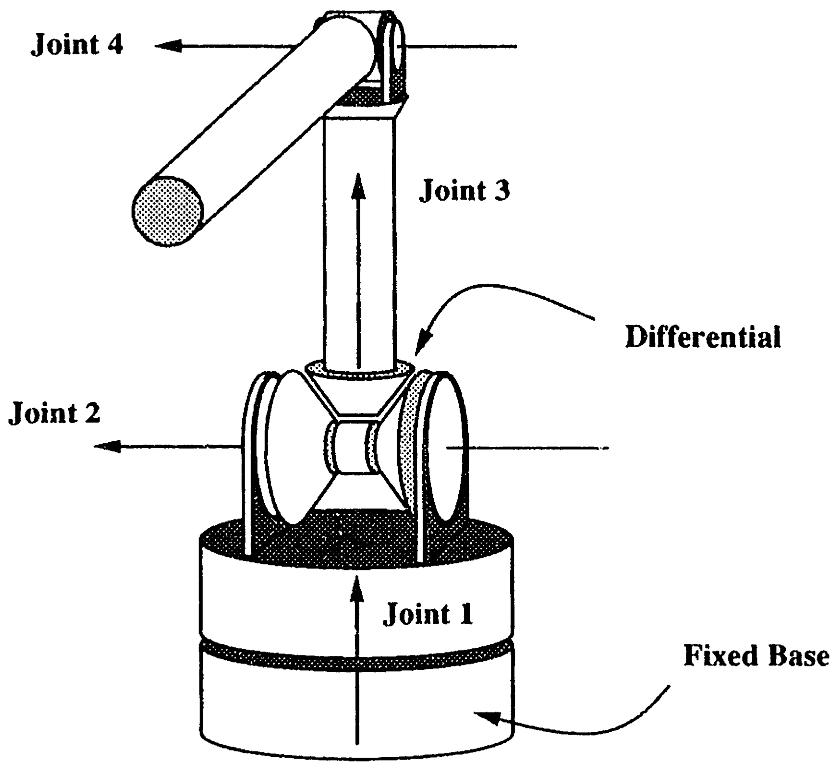


Figure 7-1: The WAM manipulator

In order to exploit the wide dynamic range that the manipulator can achieve, the arm is connected to a VME-Bus based multiprocessor system [Narasimhan, Siegel, and Hollerbach, 1988], consisting of up to six 68020 based processor boards, D/A and A/D boards, a parallel interface, as well as other boards. This system is interfaced with the network via a Sun-3 Workstation of 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. 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 integrations in the adaptation algorithm are performed using a 2nd order Adams-Bashforth scheme. All programs are written in C.

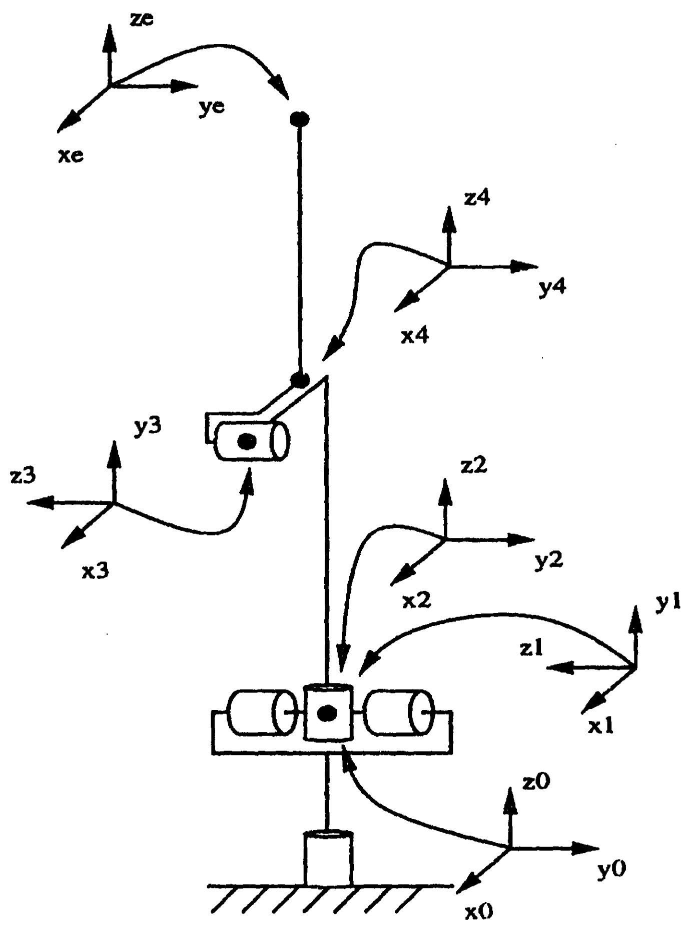


Figure 7-2: The coordinate frames of the WAM manipulator

The coordinate frame locations for the various links are determined by the Denavit-Hartenberg standard and are shown in Figure 7-2. The appropriate parameter values are given by Table 7-1. Given the structure of the manipulator, a minimal parameter set can be obtained, as was discussed in Chapter 4. In these experiments, however, a single parameter redundancy is tolerated, to assure that all ten parameters of the last link are used. This allows the adaptation to an unknown load to be restricted to just these ten parameters. In particular, Table 7-2 specifies the 23 inertial parameters of the manipulator. The friction model consists of viscous and direction-dependent Coulomb friction, so that the manipulator is described by a total of 35 unknown parameters. While dealing with unknown loads only requires the adaptation to ten parameters, the capability of effectively adapting to all 35 parameters allows the range of application to be considerably extended.

Joint	d	θ	a	α
1	0	q_1	0	+90°
2	0	q_2	0	–90°
3	0.5588 m	q_3	0.0406 m	+90°
4	0	q_4	–0.0279 m	–90°
e	0.4598 m	0	0	0

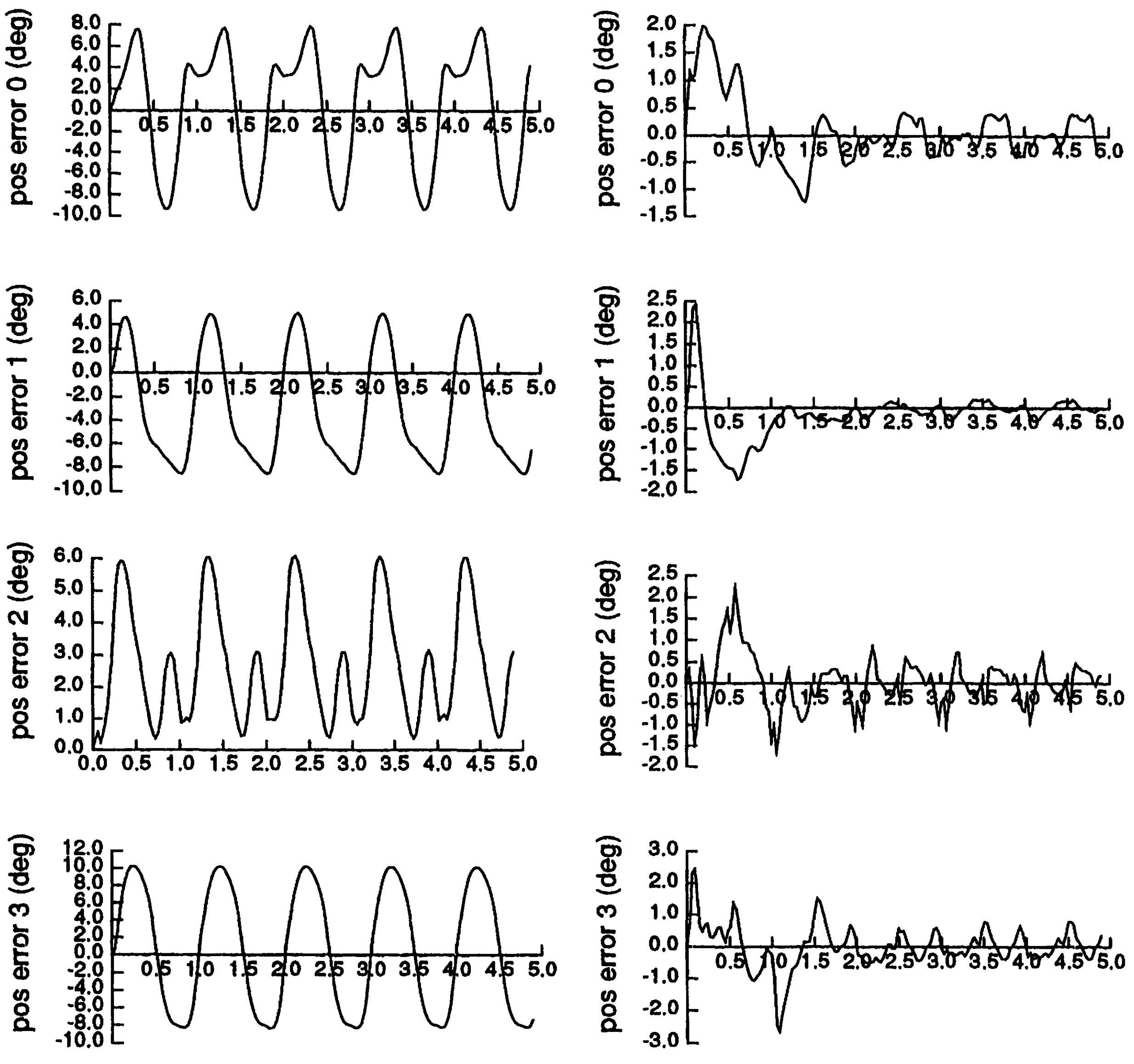
Table 7-1: Denavit-Hartenberg Parameters of the WAM manipulator

Link	Inertial Parameters									
1	.	J_{yy}	.	.	.	.	.	.	.	.
2	J_{xx}	.	.	J_{xy}	J_{xz}	J_{yz}	p_x	.	.	.
3	J_{xx}	.	J_{zz}	J_{xy}	J_{xz}	J_{yz}	p_x	p_y	.	.
4	J_{xx}	J_{yy}	J_{zz}	J_{xy}	J_{xz}	J_{yz}	p_x	p_y	p_z	m

Table 7-2: The necessary inertial parmaters of the WAM manipulator

7.2 A Comparison between P.D. and Adaptive Control

To compare a simple P.D. controller to the adaptive controller, the manipulator was commanded to follow a sinusoidal joint trajectory. This trajectory had a period of 1 second and an amplitude of approximately $\pm 45°$ per joint, thus allowing the tip to travel 5 meters per period, with maximum tip velocities and accelerations of 8.5m/s and 10g. This was done first with a simple P.D. controller, whose performance is helped by the presence of transmission ratios, and second with the adaptive scheme starting as a P.D., that is with an initial $\hat{\boldsymbol{a}} = \boldsymbol{0}$. The plots in Figure 7-3 clearly demonstrate a factor 10 to 20 improvement in tracking error after transients of about 1 second 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 5 times smaller than that of the P.D.. Nevertheless, the generated joint torques, shown in Figure 7-4, are very similar in both smoothness and amplitude.



P.D. ***Adaptive Controller***

Figure 7-3: Joint Tracking Errors $\tilde{q}$ (in degrees)

The residual tracking error is mostly due to actuator and other unmodeled dynamics. In particular, the motors produce torque ripple of about 5%. Yet, 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 parameter drift and enhance robustness. The constant adaptation gains were set according to the following

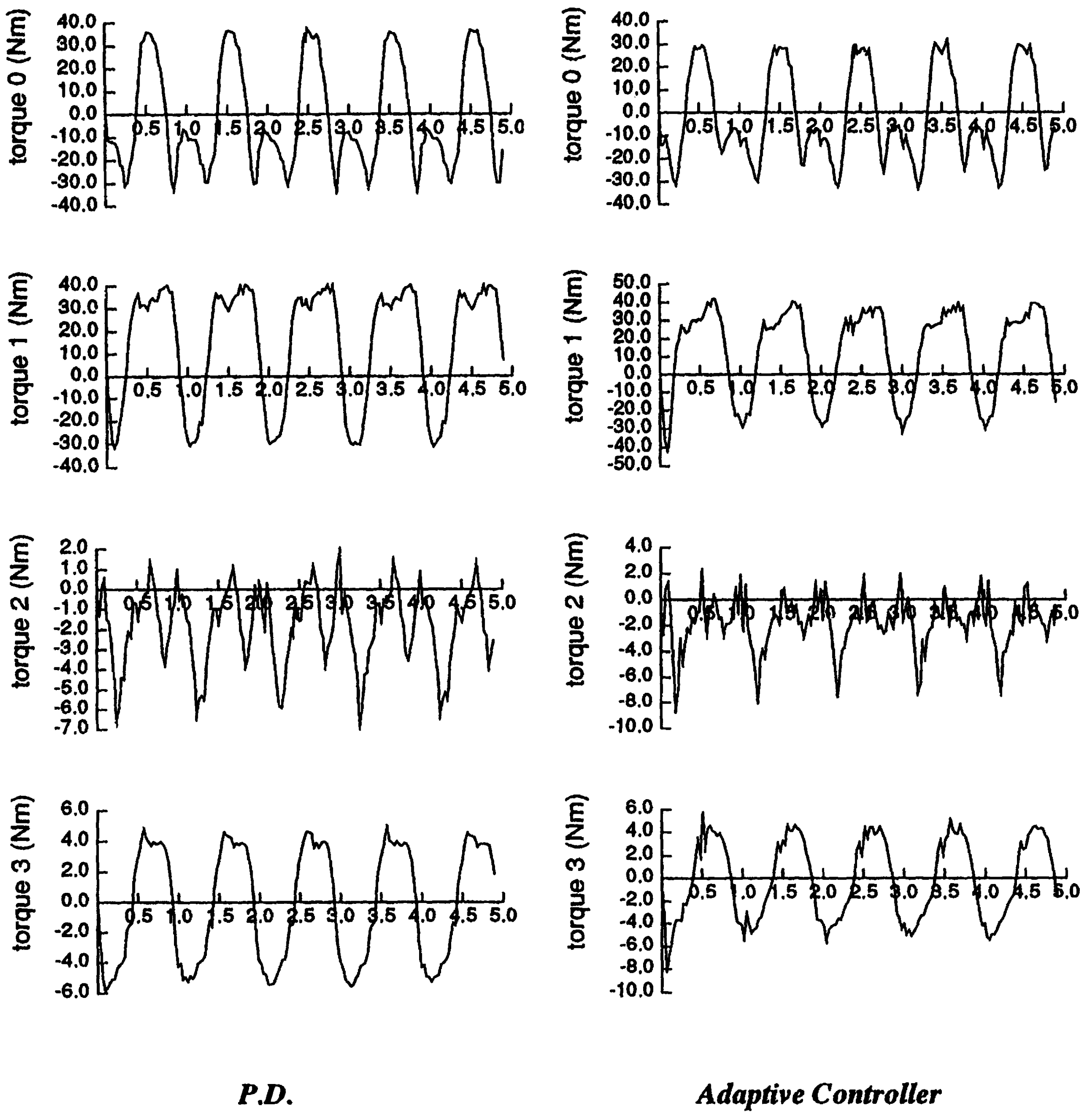


Figure 7-4: Joint Torques τ (in N.m.)

equation, which is intuitively motivated by least-squares identification rules. With Y defined in Equation (2.7)

$$P \sim \text{Diagonal}\left(\int_0^T Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d)^T\, Y(q_d,\dot{q}_d,\dot{q}_d,\ddot{q}_d)\ dt \right)^{-1} \tag{7.1}$$

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

7.3 Cartesian Control Experiments

The inverse kinematics and adaptive cartesian control were tested on trajectories describing a square in the vertical plane. Each side measures 1 meter and was completed in 0.4 seconds. Figure 7-5 shows the resulting cartesian tracking errors under both P.D. and cartesian control, which again clearly demonstrates the improvement caused by the adaptation process both in transients and steady-state. Also the adaptive impedance controller was shown to have excellent performance in making and maintaining stable contact with an unknown curved surface while performing adaptive tracking in the unconstrained directions.

7.4 Whole Arm Experiments

In other experiments, the adaptive controller was used to push a large and heavy box over the floor, at speeds of about 0.75 m/s, by exploiting the kinematic redundancy to maintain line contact with the box. 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 interpreted by the algorithm as a change in inertial and friction coefficients. This robustness is quite remarkable for a reasonably complex high-performance algorithm, and should extend the range of applications for adaptive manipulator control well beyond adaptation to grasped loads. Also, the accuracy of the manipulator controller allows the use of the "geometric" stability properties of the task ([Mason, 1986] in the case of pushing), thus permitting large uncertainties on the initial position of the box, as well as high speed transition (contact) phases.

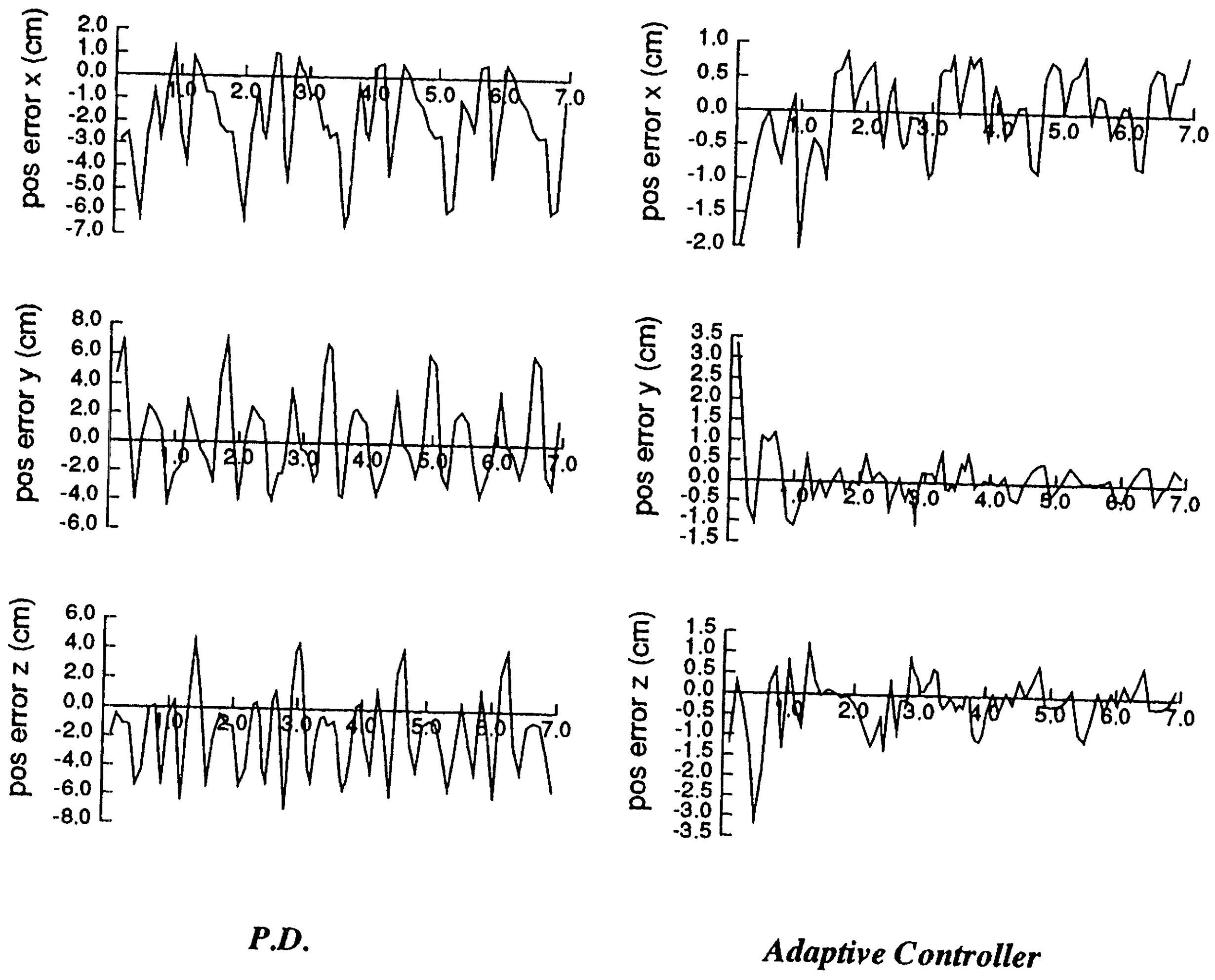


Figure 7-5: Cartesian Tracking Errors $\tilde{x}(t)$ (in cm)

Appendix A

The Spatial Vector Notation

Individual points in space have three degrees of freedom and are well described by the traditional three dimensional vectors. Rigid bodies, however, have six degrees of freedom, and require a set of two traditional vectors to describe both linear and angular quantities. This leads to complex equations and notation, as the relation between the two corresponding vectors must always be treated explicitly.

A more compact description of rigid bodies is given by the *spatial notation* of [Featherstone, 1987], which combines the corresponding linear and angular quantities into a single six dimensional vector. This reduces the number of equations and variables, and also simplifies many expressions, automatically coupling the three dimensional subparts. Due to this coupling, however, the spatial vectors obey a slightly different set of rules than do traditional vectors. Intuitively very similar, these rules are discussed in the following. Traditional vectors are denoted by arrows ($\vec{v}$) to avoid confusion with bold spatial vectors ($\boldsymbol{v}$).

A.1 The Spatial Vector Definition

In general spatial vectors are defined to be a combination of a 3-dimensional *line vector* and a 3-dimensional *free vector*. Line vectors are referenced along a particular line in space, and can be shifted only along this line. Physical examples thereof are the force applied to or the angular velocity of a rigid body. In contrast, free vectors have no point of reference and can be shifted in any direction, as is the case for the applied torque or the translational velocity. To shift a line vector perpendicular to its reference line, the free vector has to change value. This implements the coupling between linear and angular vectors.

When combined, the line vector is located in the upper half and the free vector in the lower half of the spatial vector, as for example in the spatial velocity or force vectors

$$\boldsymbol{v} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{F} = \begin{bmatrix} \vec{F} \\ \vec{\tau} \end{bmatrix} \tag{A.1}$$

Therefore, in all spatial vectors of motion (i.e. velocity, acceleration) the angular quantity takes the upper half, while for spatial vectors of force and momentum the angular quantity takes the lower half. Though initially somewhat confusing, this difference actually unifies the spatial vector rules.

A.2 Spatial Vector Algebra

To describe a linear mapping between spatial vectors, a spatial matrix is introduced. This 6x6 matrix obeys the standard matrix rules, with the exception of the transpose operator. Splitting the spatial matrix into standard 3x3 submatrices and using the standard operator for these subparts, the spatial transpose operator can be defined as

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^T = \begin{bmatrix} D^T & B^T \\ C^T & A^T \end{bmatrix} \tag{A.2}$$

Accordingly spatial vectors transpose as

$$\begin{bmatrix} \vec{l} \\ \vec{f} \end{bmatrix}^T = \begin{bmatrix} \vec{f}^T & \vec{l}^T \end{bmatrix} \tag{A.3}$$

These rules give the transpose operator the same properties as its standard counterpart, in particular the product of velocity and force results in power.

The spatial notation also makes use of a vector product, defined by the following matrix product

$$(s\times)\, q = \begin{bmatrix} \vec{l} \\ \vec{f} \end{bmatrix} \times q = \begin{bmatrix} \vec{l}\times & 0 \\ \vec{f}\times & \vec{l}\times \end{bmatrix} \cdot q \tag{A.4}$$

where $(\vec{l}\times)$ and $(\vec{f}\times)$ denote standard 3x3 skew-symmetric matrices, such that the matrix product $(\vec{l}\times)\,\vec{x}$ equals the standard 3x3 vector product $\vec{l} \times \vec{x}$.

The spatial vector product has the following properties, analogous to its 3 dimensional counterpart.

$$s \times q = -\, q \times s \tag{A.5}$$

$$(s\times)^T = -\,(s\times) \tag{A.6}$$

A.3 Reference Frame Transformations

As spatial vectors have 6 dimensions, the reference frames must also have 6 degrees of freedom. That is, in addition to the orientation of the reference frame, the location of its origin needs to be specified. This point is arbitrary and for rigid body quantities does not have to coincide with the center of mass.

Accordingly, a transfomation of reference frame consists of two parts, the movement of the origin and the rotation of the frame. The first step is specific to spatial vectors and takes care of the coupling between angular and translational quantities. If a spatial vector *s* is composed of line vector $\vec{l}$ and free vector $\vec{f}$, a movement of the reference frame origin from point *a* to point *b* must be accounted for as follows

$$s\,|^b = \begin{bmatrix} \vec{l}\,|^b \\ \vec{f}\,|^b \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \vec{r}_a{}^b\,|^b \times & 1 \end{bmatrix} \cdot \begin{bmatrix} \vec{l}\,|^a \\ \vec{f}\,|^a \end{bmatrix} \qquad (A.7)$$

where $\vec{r}_a{}^b = \vec{r}_a - \vec{r}_b$ is the vector from point *b* to point *a*. Note that superscripts after a vertical bar represent the reference frame and $(\vec{r}\times)$ denotes a skew-symmetric matrix, such that the matrix product $(\vec{r}\times)\,\vec{l}$ equals the vector product $\vec{r}\times\vec{l}$.

The second step accounts for the rotation of the frame and is equivalent to standard 3-dimensional vector rotations. Thus if the reference frame is rotated from orientation *a* into orientation *b* with the standard 3-dimensional rotation matrix defined as $\boldsymbol{R}_a\,|^b$, then a spatial vector changes according to

$$s\,|^b = \begin{bmatrix} \vec{l}\,|^b \\ \vec{f}\,|^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_a\,|^b & 0 \\ 0 & \boldsymbol{R}_a\,|^b \end{bmatrix} \cdot \begin{bmatrix} \vec{l}\,|^a \\ \vec{f}\,|^a \end{bmatrix} \qquad (A.8)$$

Combining the above, the complete spatial reference frame transformation can be written as

$$s|^b = \begin{bmatrix} \vec{l}\,|^b \\ \vec{f}\,|^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_a|^b & \boldsymbol{0} \\ \vec{r}_a{}^b|^b \times \boldsymbol{R}_a|^b & \boldsymbol{R}_a|^b \end{bmatrix} \cdot \begin{bmatrix} \vec{l}\,|^a \\ \vec{f}\,|^a \end{bmatrix} = \boldsymbol{X}_a|^b\, s|^a \qquad (A.9)$$

with the spatial transformation matrix $\boldsymbol{X}_a|^b$. Such a transformation matrix has properties similar to the standard rotation matrices. In particular,

$$\boldsymbol{X}_a|^c = \boldsymbol{X}_b|^c \cdot \boldsymbol{X}_a|^b \qquad (A.10)$$

$$\boldsymbol{X}_a|^{b\,-1} = \boldsymbol{X}_a|^{b\,T} = \boldsymbol{X}_b|^a \qquad (A.11)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{X}_a|^b = \boldsymbol{v}_a{}^b|^b \times \boldsymbol{X}_a|^b = (\boldsymbol{v}_a - \boldsymbol{v}_b)|^b \times \boldsymbol{X}_a|^b \qquad (A.12)$$

$$\boldsymbol{X}_a|^b\,(s|^a \times) = (s|^b \times)\,\boldsymbol{X}_a|^b \qquad (A.13)$$

A.4 The Spatial Inertia Matrix

To analyze rigid body dynamics in the spatial notation, a spatial inertia matrix has to be defined, which maps the spatial velocity to momentum. This inertia matrix contains all ten inertial parameters of a rigid body, that is it consists of the standard 3x3 inertia matrix $\boldsymbol{J}$, the mass m, and the product of mass and location of center of mass $\vec{p}$.

$$\boldsymbol{I} = \begin{bmatrix} -\vec{p} & m \cdot \boldsymbol{1} \\ \boldsymbol{J} & \vec{p} \end{bmatrix} \qquad (A.14)$$

where $\boldsymbol{1}$ denotes the 3x3 unity matrix. Note the reference frame, which is stationary with respect to the link, can be placed at any point with any orientation and does not need to be located at the center of mass or be oriented according to the principle inertia axes.

The inertia matrix obeys the following properties, for changing the reference frame and determining its time derivatives.

$$I_a|^b = X_a|^b \cdot I_a|^a \cdot X_b|^a \tag{A.15}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} I_a|^b = v_a{}^b|^b \times I_a|^b - I_a|^b \, v_a{}^b|^b \times \tag{A.16}$$

To separate the effect of the different parameters, the inertia matrix can also be written as a product of ten placement matrices and the ten parameter values

$$I = \sum_{i=1}^{10} R_i \, a^i \tag{A.17}$$

where the parameters are given as

$$\begin{array}{llll} a^1 = J_{xx} & a^4 = J_{xy} & a^7 = p_x = m\, r_x & a^{10} = m \\ a^2 = J_{yy} & a^5 = J_{xz} & a^8 = p_y = m\, r_y & \\ a^3 = J_{zz} & a^6 = J_{yz} & a^9 = p_z = m\, r_z & \end{array} \tag{A.18}$$

and the placement matrices consist only of ones and zeros, as for example

$$R_7 = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & +1 & 0 \end{array}\right] \tag{A.19}$$

To facilitate the analysis in some sections, the inertia matrix can also be explicitly written as a function of the ten parameters. Interpreting the bracketed numbers as the values of the specified parameters

$$\boldsymbol{I} = \begin{bmatrix} . & +(9) & -(8) & +(10) & . & . \\ -(9) & . & +(7) & . & +(10) & . \\ +(8) & -(7) & . & . & . & +(10) \\ \\ +(1) & +(4) & +(5) & . & -(9) & +(8) \\ +(4) & +(2) & +(6) & +(9) & . & -(7) \\ +(5) & +(6) & +(3) & -(8) & +(7) & . \end{bmatrix} \tag{A.20}$$

A.5 The Dynamics of an Arbitrary Rigid Body

With the above tools it is possible to write the equation of motion for an arbitrary rigid body in the spatial notation. This single equation includes both Newton's and Euler's equation and accounts for their coupling automatically. With $\boldsymbol{a}$ denoting the acceleration, the equation of motion is given by

$$\boldsymbol{F} = \frac{\mathrm{d}}{\mathrm{d}t}\, \boldsymbol{I}\, \boldsymbol{v} = \boldsymbol{I}\, \boldsymbol{a} + \boldsymbol{v} \times \boldsymbol{I}\, \boldsymbol{v} \tag{A.21}$$

Similarly the spatial notation simplifies the total kinetic energy including both rotational and translational energy as well as the power input.

$$E_{\mathrm{kin}} = \frac{1}{2}\, \boldsymbol{v}^T \boldsymbol{I}\, \boldsymbol{v} \tag{A.22}$$

$$P_{\mathrm{in}} = \boldsymbol{v}^T \boldsymbol{F} \tag{A.23}$$

Appendix B

Evaluating the Link Force Coefficients

The key part of the recursive algorithm is the computation of the local force coefficients at each link, given by Equation (3.25). Multiplying these force coefficients with the parameter values determines the forces (see Equation (3.7)) and, furthermore, multiplying them with the velocity errors determines the adaptation values (see Equation (3.28)).

The placement matrices, used to evaluate the force coefficients, are extremely sparse and therefore it is important to customize the algorithm by evaluating these coefficients analytically and eliminating multiplications and additions with zero. The gravitational forces are implemented as a vertical acceleration and can be ignored, so that the link force coefficients are given by

$$f^i = R_i \dot{w} + \frac{1}{2} v \times R_i w + \frac{1}{2} w \times R_i v + \frac{1}{2} R_i w \times v \qquad (B.1)$$

where, for simplicity, the subscripts determining the link are removed from all quantities.

Similarly as in Section 4.1, the above expression can be determined using Tables B-1 and B-2. These Tables give the results for unity vectors $\boldsymbol{v}$, $\boldsymbol{w}$, and $\dot{\boldsymbol{w}}$, where the bracketed numbers should be understood as

$$(x) = \begin{cases} 1 & \text{if } x = i \\ 0 & \text{if } x \neq i \end{cases} \tag{B.2}$$

$\dot{w}_1$	$\dot{w}_2$	$\dot{w}_3$	$\dot{w}_4$	$\dot{w}_5$	$\dot{w}_6$
.	+(9)	−(8)	+(10)	.	.
−(9)	.	+(7)	.	+(10)	.
+(8)	−(7)	.	.	.	+(10)
+(1)	+(4)	+(5)	.	−(9)	+(8)
+(4)	+(2)	+(6)	+(9)	.	−(7)
+(5)	+(6)	+(3)	−(8)	+(7)	.

Table B-1: Evaluation of $\boldsymbol{R}_i\,\dot{\boldsymbol{w}}$

Also, in Table B-3 the force coefficients are specified separately as an analytic function of the components of velocities and acceleration. Examining these functions, it can be noted that many terms repeat. Therefore, it is advantageous to define "velocity combinations" μ, that is products and sums of velocities and accelerations, in an intermittent step and to use these to compute the force coefficients. Tables B-4 and B-5 give the appropriate expressions, which represent a very efficient way to implement the local force computations.

	v_1	v_2	v_3	v_4	v_5	v_6
w_1	. −2(8) −2(9) . −2(5) +2(4)	. +2(7) . +2(5) . +(3)+(2)−(1)	. . +2(7) −2(4) −(3)−(2)+(1) .	. . . . . .	. . +2(10) +2(8) −2(7) .	. −2(10) . +2(9) . −2(7)
w_2	+2(8) . . . −2(6) −(3)+(2)−(1)	−2(7) . −2(9) +2(6) . −2(4)	. . +2(8) +(3)−(2)+(1) +2(4) .	. . −2(10) −2(8) +2(7) .	. . . . . .	+2(10) . . . +2(9) −2(8)
w_3	+2(9) . . . −(3)+(2)+(1) +2(6)	. +2(9) . +(3)−(2)−(1) . +2(5)	−2(7) −2(8) . −2(6) −2(5) .	. +2(10) . −2(9) . +2(7)	−2(10) . . . −2(9) +2(8)	. . . . . .
w_4	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .
w_5	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .
w_6	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .

Table B-2: Evaluation of $w \times R_i v + v \times R_i w + R_i w \times v$

$$f^1 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ -\frac{1}{2}w_3v_2 + \frac{1}{2}w_2v_3 + \dot{w}_1 \\ +\frac{1}{2}w_3v_1 + \frac{1}{2}w_1v_3 \\ -\frac{1}{2}w_2v_1 - \frac{1}{2}w_1v_2 \end{bmatrix}, \quad f^2 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ -\frac{1}{2}w_3v_2 - \frac{1}{2}w_2v_3 \\ +\frac{1}{2}w_3v_1 - \frac{1}{2}w_1v_3 + \dot{w}_2 \\ +\frac{1}{2}w_2v_1 + \frac{1}{2}w_1v_2 \end{bmatrix}, \quad f^3 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ +\frac{1}{2}w_3v_2 + \frac{1}{2}w_2v_3 \\ -\frac{1}{2}w_3v_1 - \frac{1}{2}w_1v_3 \\ -\frac{1}{2}w_2v_1 + \frac{1}{2}w_1v_2 + \dot{w}_3 \end{bmatrix}$$

$$f^4 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ -w_1v_3 \qquad + \dot{w}_2 \\ +w_2v_3 \qquad + \dot{w}_1 \\ +w_1v_1 - w_2v_2 \end{bmatrix}, \quad f^5 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ +w_1v_2 \qquad + \dot{w}_3 \\ -w_1v_1 + w_3v_3 \\ -w_3v_2 \qquad + \dot{w}_1 \end{bmatrix}, \quad f^6 = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ +w_2v_2 - w_3v_3 \\ -w_2v_1 \qquad + \dot{w}_3 \\ +w_3v_1 \qquad + \dot{w}_2 \end{bmatrix}$$

$$f^7 = \begin{bmatrix} -w_2v_2 - w_3v_3 \\ +w_1v_2 \qquad + \dot{w}_3 \\ +w_1v_3 \qquad - \dot{w}_2 \\ \cdot \\ -w_1v_5 + w_2v_4 - \dot{w}_6 \\ -w_1v_6 + w_3v_4 + \dot{w}_5 \end{bmatrix}, \quad f^8 = \begin{bmatrix} +w_2v_1 \qquad - \dot{w}_3 \\ -w_1v_1 - w_3v_3 \\ +w_2v_3 \qquad + \dot{w}_1 \\ +w_1v_5 - w_2v_4 + \dot{w}_6 \\ \cdot \\ -w_2v_6 + w_3v_5 - \dot{w}_4 \end{bmatrix}, \quad f^9 = \begin{bmatrix} +w_3v_1 \qquad + \dot{w}_2 \\ +w_3v_2 \qquad - \dot{w}_1 \\ -w_1v_1 - w_2v_2 \\ +w_1v_6 - w_3v_4 - \dot{w}_5 \\ +w_2v_6 - w_3v_5 + \dot{w}_4 \\ \cdot \end{bmatrix}$$

$$f^{10} = \begin{bmatrix} +w_2v_6 - w_3v_5 + \dot{w}_4 \\ -w_1v_6 + w_3v_4 + \dot{w}_5 \\ +w_1v_5 - w_2v_4 + \dot{w}_6 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

Table B-3: Analytic functions for the individual force coefficients

$$\mu_1 = -\, w_2 v_2 - w_1 v_1$$
$$\mu_2 = -\, w_1 v_1 - w_3 v_3$$
$$\mu_3 = -\, w_3 v_3 - w_2 v_2$$

$$\mu_4 = -\, w_2 v_2 + w_1 v_1$$
$$\mu_5 = -\, w_1 v_1 + w_3 v_3$$
$$\mu_6 = -\, w_3 v_3 + w_2 v_2$$

$$\mu_7 = w_2 v_6 - w_3 v_5 + \dot{w}_4$$
$$\mu_8 = w_3 v_4 - w_1 v_6 + \dot{w}_5$$
$$\mu_9 = w_1 v_5 - w_2 v_4 + \dot{w}_6$$

$$\mu_{10} = w_3 v_2 - \dot{w}_1$$
$$\mu_{11} = w_2 v_3 + \dot{w}_1$$
$$\mu_{12} = \frac{1}{2}(w_2 v_3 + w_3 v_2)$$
$$\mu_{13} = \frac{1}{2}(w_2 v_3 - w_3 v_2) + \dot{w}_1$$

$$\mu_{14} = w_1 v_3 - \dot{w}_2$$
$$\mu_{15} = w_3 v_1 + \dot{w}_2$$
$$\mu_{16} = \frac{1}{2}(w_3 v_1 + w_1 v_3)$$
$$\mu_{17} = \frac{1}{2}(w_3 v_1 - w_1 v_3) + \dot{w}_2$$

$$\mu_{18} = w_2 v_1 - \dot{w}_3$$
$$\mu_{19} = w_1 v_2 + \dot{w}_3$$
$$\mu_{20} = \frac{1}{2}(w_1 v_2 + w_2 v_1)$$
$$\mu_{21} = \frac{1}{2}(w_1 v_2 - w_2 v_1) + \dot{w}_3$$

Table B-4: Velocity combinations for the force coefficient evaluation

$i =$	1	2	3	4	5	6	7	8	9	10
	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$+\mu_3$	$+\mu_{18}$	$+\mu_{15}$	$+\mu_7$
	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$+\mu_{19}$	$+\mu_2$	$+\mu_{10}$	$+\mu_8$
	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$+\mu_{14}$	$+\mu_{11}$	$+\mu_1$	$+\mu_9$
$f^i =$	$+\mu_{13}$	$-\mu_{12}$	$+\mu_{12}$	$-\mu_{14}$	$+\mu_{19}$	$+\mu_6$	$\cdot$	$+\mu_9$	$-\mu_8$	$\cdot$
	$+\mu_{16}$	$+\mu_{17}$	$-\mu_{16}$	$+\mu_{11}$	$+\mu_5$	$-\mu_{18}$	$-\mu_9$	$\cdot$	$+\mu_7$	$\cdot$
	$-\mu_{20}$	$+\mu_{20}$	$+\mu_{21}$	$+\mu_4$	$-\mu_{10}$	$+\mu_{15}$	$+\mu_8$	$-\mu_7$	$\cdot$	$\cdot$

Table B-5: Individual force coefficients given by velocity combinations

Appendix C

The Implementation Code

This Appendix gives the key segments of the actual implementation code. In particular, two routines are listed, which perform the recursive computations at each link for both the controller and adaptation portions of the system. The Denavit-Hartenberg parameters are left general, so that the routines can easily be customized for any particular kinematic structure. First, however, the type and variable definitions are given.

C.1 Type and Variable Definitions

```
/*
**  typedef.h
**
**  (c) Copyright 1989.  All rights reserved.
**   Nonlinear Systems Lab and Artificial Intelligence Lab, MIT
**
**
**   Type Dimensions :   SPTL , VCMB , PRMS_INRTL
**
**   Type Definitions :  spatl , vccmb , param
**
*/

#ifndef TYPEDEF
#define TYPEDEF
```

```
/*
**    Type Dimensions
*/

#define  SPTL         ( 6 )               /* dimension of spatial vectors */
#define  VCMB         ( 21 )              /* number of velocity combinations */
                                          /*  per link needed in traversing  */
#define  PRMS_INRTL   ( 10 )              /* number of parameters per link */


/*
**    Type Definitions
*/

typedef  float    spatl[SPTL] ;           /* spatial vector (6 elements) */

typedef  float    vcomb[VCMB] ;           /* link cartesian velocity    */
                                          /*  combinations (21 choices) */

typedef  struct                           /* parameters of a link including: */
{  float  inrtl[PRMS_INRTL] ;             /*  10 inertial parameters         */
   float  f_vis ;                         /*  viscous friction               */
   float  f_col ;                         /*  coulomb friction               */
   float  f_off ;                         /*  friction offset                */
 }                param ;


#endif


/*
**  (c) Copyright 1989.  All rights reserved.
**   Nonlinear Systems Lab and Artificial Intelligence Lab, MIT
**
**
**   The following gives the variable definitions used in the controller
**    and adaptation routines.
**
*/


/* cartesian data */

spatl    vel[LNKS] ;                      /* Link velocities */
spatl  r_vel[LNKS] ;                      /* Link reference velocities */
spatl  r_acc[LNKS] ;                      /* Link reference accelerations */
spatl  force[LNKS] ;                      /* Link forces */
spatl  v_err[LNKS] ;                      /* Link velocity error */
vcomb  v_com[LNKS] ;                      /* Link velocity combinations */
param    prm[LNKS] ;                      /* Link parameters       */
                                          /*  (including friction) */


/* joint data */

float          q[JNTS] ;                  /* Joint position */
float      q_dot[JNTS] ;                  /* Joint velocity */
float    q_r_dot[JNTS] ;                  /* Joint reference velocity */
float   q_r_ddot[JNTS] ;                  /* Joint reference acceleration */
float        tau[JNTS] ;                  /* Joint torque */


/* Denavit-Hartenberg Parameters */

float      d[LNK] ;                       /* Parmeter d */
float   sinq[LNK] ;                       /* sine and cosine values of */
float   cosq[LNK] ;                       /*  the angle theta */
float      a[LNK] ;                       /* Parmeter a */
float   sina[LNK] ;                       /* sine and cosine values of */
float   cosa[LNK] ;                       /*  the angle alpha */
```

C.2 Controller Recursive Routine

```
/*
**  ctr_traverse_general.c
**
**  (c) Copyright 1989.  All rights reserved.
**   Nonlinear Systems Lab and Artificial Intelligence Lab, MIT
**
**
**   The following is a general controller traversing routine for
**    an arbitrary link with the Denavit-Hartenberg parameters given.
**    As part of a recursive implementation it calculates the local
**    cartesian velocity, reference velocity, reference acceleration,
**    and forces of the link.  After having traversed the remaining
**    links, it then propagates the forces downward and computes the
**    control torque for the corresponding joint due to inertial
**    effects.  It also calculates the cartesian velocity error and
**    several velocity combinations, which are used by the adaptation
**    routines.
**
**   The joint motion is here assumed to be rotational, but could
**    equally well be of any other form.
**
**   Note that all calculations are down according to the spatial
**    vector notation.
**
**   Also note that this routine is not stand-alone and thus no
**    variable definitions are given.
**
*/


/*
**   Link LNK
**  ----------
*/

traverse_LNK()

{  register float  sq   = sinq[LNK] ;   /* sine and cosine values of */
   register float  cq   = cosq[LNK] ;   /*  the Denavit-Hartenberg   */
   register float  sa   = sina[LNK] ;   /*  angles theta and alpha   */
   register float  ca   = cosa[LNK] ;
```

```
/*
**   Transform all data from previous link to this link after adding
**    the appropriate joint motion.
*/

/* Transform velocity after adding joint velocity */

{  register float   *pv = vel[LNK-1] ;    /* pointer to previous vel. */
   register float   *v  = vel[LNK] ;      /* pointer to this links vel. */

   register float  t1 , t2 , t4 ;         /* temporary variables */

   t2 = pv[2] + q_dot[LNK] ;

   v[0] =    cq * pv[0] + sq * pv[1] ;
    t1  = - sq * pv[0] + cq * pv[1] ;
   v[1] =    ca *   t1  + sa *   t2  ;
   v[2] = - sa *   t1  + ca *   t2  ;

   v[3] =    cq * pv[3] + sq * pv[4] + d[LNK] *  t1  ;
    t4  = - sq * pv[3] + cq * pv[4] - d[LNK] * v[0] ;
   v[4] =    ca *   t4  + sa * pv[5] + a[LNK] * v[2] ;
   v[5] = - sa *   t4  + ca * pv[5] - a[LNK] * v[1] ;
 }


/* Transform reference velocity after adding joint reference velocity */

{  register float   *prv = r_vel[LNK-1] ;/* pointer to previous r_vel. */
   register float   *rv  = r_vel[LNK] ;  /* pointer to this links r_vel. */

   register float  t1 , t2 , t4 ;        /* temporary variables */

   t2 = prv[2] + q_r_dot[LNK] ;

   rv[0] =    cq * prv[0] + sq * prv[1] ;
     t1  = - sq * prv[0] + cq * prv[1] ;
   rv[1] =    ca *    t1  + sa *    t2  ;
   rv[2] = - sa *    t1  + ca *    t2  ;

   rv[3] =    cq * prv[3] + sq * prv[4] + d[LNK] *   t1  ;
     t4  = - sq * prv[3] + cq * prv[4] - d[LNK] * rv[0] ;
   rv[4] =    ca *    t4  + sa * prv[5] + a[LNK] * rv[2] ;
   rv[5] = - sa *    t4  + ca * prv[5] - a[LNK] * rv[1] ;

   /* note that the second half of this vector is only used to */
   /*  compute the velocity error for the adaptation routine.  */
 }


/* compute the velocity error for the adaptation routine */

{  register float   *v  =   vel[LNK] ;   /* pointer to velocity */
   register float   *rv = r_vel[LNK] ;   /* pointer to ref_velocity */
   register float   *ve = v_err[LNK] ;   /* pointer to velocity error */

   ve[0] = v[0] - rv[0] ;
   ve[1] = v[1] - rv[1] ;
   ve[2] = v[2] - rv[2] ;
   ve[3] = v[3] - rv[3] ;
   ve[4] = v[4] - rv[4] ;
   ve[5] = v[5] - rv[5] ;
 }


/* Transform reference acceleration after adding both joint ref. acc. */
/*  and cross-product of velocity and direction of joint ref. vel. */

{  register float  *pra = r_acc[LNK-1] ;/* pointer to previous r_acc. */
   register float  *ra  = r_acc[LNK] ;  /* pointer to this links r_acc. */
   register float  *pv  = vel[LNK-1] ;  /* pointer to previous vel. */

   register float  t0 , t1 , t2 ;       /* temporary variables */
   register float  t3 , t4 ;            /* temporary variables */

   t0 = pra[0] + pv[1] * q_r_dot[LNK] ;
   t1 = pra[1] - pv[0] * q_r_dot[LNK] ;
   t2 = pra[2]                        + q_r_ddot[LNK] ;
   t3 = pra[3] + pv[4] * q_r_dot[LNK] ;
   t4 = pra[4] - pv[3] * q_r_dot[LNK] ;

   ra[0] =   cq * t0 + sq * t1 ;
     t1  = - sq * t0 + cq * t1 ;
   ra[1] =   ca * t1 + sa * t2 ;
   ra[2] = - sa * t1 + ca * t2 ;

   ra[3] =   cq * t3 + sq *    t4  + d[LNK] *   t1  ;
     t4  = - sq * t3 + cq *    t4  - d[LNK] * ra[0] ;
   ra[4] =   ca * t4 + sa * pra[5] + a[LNK] * ra[2] ;
   ra[5] = - sa * t4 + ca * pra[5] - a[LNK] * ra[1] ;
 }


/*
**   Compute the velocity combinations, which are later needed
**    for force calculations and for adaptation purposes.
*/

{  register float  *v  = vel[LNK] ;     /* pointer to velocity */
   register float  *rv = r_vel[LNK] ;   /* pointer to ref_velocity */
   register float  *ra = r_acc[LNK] ;   /* pointer to ref_acceleration */
   register float  *vc = v_com[LNK] ;   /* pointer to vel_combinations */

   register float  half = 0.5 ;         /* constant 1/2 */
   register float  t0 , t1 , t2 ;       /* temporary variables */


   t0 = - rv[0] * v[0] ;
   t1 = - rv[1] * v[1] ;
   t2 = - rv[2] * v[2] ;

   vc[ 0] = t1 + t0 ;                   /* combinations 0-5 */
   vc[ 1] = t0 + t2 ;
   vc[ 2] = t2 + t1 ;

   vc[ 3] = t1 - t0 ;
   vc[ 4] = t0 - t2 ;
   vc[ 5] = t2 - t1 ;
   vc[ 6] = rv[1] * v[5] - rv[2] * v[4] + ra[3] ;      /* combinations */
   vc[ 7] = rv[2] * v[3] - rv[0] * v[5] + ra[4] ;      /*   6-8        */
   vc[ 8] = rv[0] * v[4] - rv[1] * v[3] + ra[5] ;


            t0 = ra[0] ;                          /* combinations 9-12 */
   vc[ 9] = t1 = rv[2] * v[1] - t0 ;
   vc[10] = t0 = rv[1] * v[2] + t0 ;

   vc[11] = t0 = half * ( t0 + t1 ) ;
   vc[12] =                t0 - t1   ;

            t0 = ra[1] ;                          /* combinations 13-16 */
   vc[13] = t1 = rv[0] * v[2] - t0 ;
   vc[14] = t0 = rv[2] * v[0] + t0 ;

   vc[15] = t0 = half * ( t0 + t1 ) ;
   vc[16] =                t0 - t1   ;

            t0 = ra[2] ;                          /* combinations 17-20 */
   vc[17] = t1 = rv[1] * v[0] - t0 ;
   vc[18] = t0 = rv[0] * v[1] + t0 ;

   vc[19] = t0 = half * ( t0 + t1 ) ;
   vc[20] =                t0 - t1   ;
 }


/*
**   Calculate the local forces as a product between the above vel.
**    combinations and the inertial parameter values.
*/

{  register float  *f  = force[LNK] ;      /* pointer to force */
   register float  *p  = prm[LNK].inrtl ; /* pointer to parameters */
   register float  *vc = v_com[LNK] ;      /* pointer to vel_combinations */

   f[0] =   vc[ 2] * p[6] + vc[17] * p[7] + vc[14] * p[8] + vc[ 6] * p[9] ;
   f[1] =   vc[18] * p[6] + vc[ 1] * p[7] + vc[ 9] * p[8] + vc[ 7] * p[9] ;
   f[2] =   vc[13] * p[6] + vc[10] * p[7] + vc[ 0] * p[8] + vc[ 8] * p[9] ;

   f[3] =   vc[12] * p[0] - vc[11] * p[1] + vc[11] * p[2] - vc[13] * p[3]
          + vc[18] * p[4] + vc[ 5] * p[5] + vc[ 8] * p[7] - vc[ 7] * p[8] ;

   f[4] =   vc[15] * p[0] + vc[16] * p[1] - vc[15] * p[2] + vc[10] * p[3]
          + vc[ 4] * p[4] - vc[17] * p[5] - vc[ 8] * p[6] + vc[ 6] * p[8] ;

   f[5] = - vc[19] * p[0] + vc[19] * p[1] + vc[20] * p[2] + vc[ 3] * p[3]
          - vc[ 9] * p[4] + vc[14] * p[5] + vc[ 7] * p[6] - vc[ 6] * p[7] ;
 }


/*
**   Connect upwards to next link
*/

traverse_LNK+1() ;


/*
```

```
**   Propagate the forces downward and compute the joint torque.
*/

{  register float  *pf = force[LNK-1] ; /* pointer to previous force */
   register float  *f  = force[LNK] ;   /* pointer to force */

   register float  t0 , t1 , t2 ;       /* temporary variables */
   register float       t4 , t5 ;       /* temporary variables */

    t1  =   ca * f[1] - sa * f[2] ;
    t2  =   sa * f[1] + ca * f[2] ;
    t4  =   ca * f[4] - sa * f[5] - a[LNK] * t2 ;
    t5  =   sa * f[4] + ca * f[5] + a[LNK] * t1 ;

   pf[0] += t0 =   cq * f[0] - sq * t1 ;
   pf[1] += t1 =   sq * f[0] + cq * t1 ;
   pf[2] += t2 ;
   pf[3] +=        cq * f[3] - sq * t4 - d[LNK] * t1 ;
   pf[4] +=        sq * f[3] + cq * t4 + d[LNK] * t0 ;
   pf[5] += t5 ;


   tau[LNK] = t5 ;                      /* set joint torque */
 }


   return;
 }
```

C.3 Adaptation Recursive Routine

```
/*
**   adp_traverse_general.c
**
**   (c) Copyright 1989.  All rights reserved.
**    Nonlinear Systems Lab and Artificial Intelligence Lab, MIT
**
**
**    The following is a general adaptation traversing routine for
**     an arbitrary link with the Denavit-Hartenberg parameters given.
**     As part of a recursive implementation it calculates the inertial
**     parameter derivatives and updates the parameter estimates.
**
**    The joint motion is here assumed to be rotational, but could
**     equally well be of any other form.
**
**    Note that all calculations are down according to the spatial
**     vector notation.
**
**    Also note that this routine is not stand-alone and thus no
**     variable definitions are given.
**
*/

/*
**    Link LNK
**  ----------
*/

traverse_LNK()

{

/*
**   Compute the inertial parameter derivatives
*/

{  register float  *pd = prm_dot[LNK].inrtl ; /* pointer to the prm deriv. */
   register float  *vc = v_com[LNK] ;         /* pointer to vel. comb. */
   register float  *ve = v_err[LNK] ;         /* pointer to v_errors */

   pd[0] =   vc[12] * ve[0] + vc[15] * ve[1] - vc[19] * ve[2] ;
   pd[1] = - vc[11] * ve[0] + vc[16] * ve[1] + vc[19] * ve[2] ;
   pd[2] =   vc[11] * ve[0] - vc[15] * ve[1] + vc[20] * ve[2] ;

   pd[3] = - vc[13] * ve[0] + vc[10] * ve[1] + vc[ 3] * ve[2] ;
   pd[4] =   vc[18] * ve[0] + vc[ 4] * ve[1] - vc[ 9] * ve[2] ;
   pd[5] =   vc[ 5] * ve[0] - vc[17] * ve[1] + vc[14] * ve[2] ;

   pd[6] =                  - vc[ 8] * ve[1] + vc[ 7] * ve[2]
           + vc[ 2] * ve[3] + vc[18] * ve[4] + vc[13] * ve[5] ;
   pd[7] =   vc[ 8] * ve[0]                  - vc[ 6] * ve[2]
           + vc[17] * ve[3] + vc[ 1] * ve[4] + vc[10] * ve[5] ;
   pd[8] = - vc[ 7] * ve[0] + vc[ 6] * ve[1]
           + vc[14] * ve[3] + vc[ 9] * ve[4] + vc[ 0] * ve[5] ;

   pd[9] =   vc[ 6] * ve[3] + vc[ 7] * ve[4] + vc[ 8] * ve[5] ;
 }


/*
**   Update the inertial parameter estimates
*/

{  register float  *pm  = prm[LNK].inrtl ;           /* parameters */
   register float  *gn  = gain[LNK].inrtl ;          /* adaptation gains */
   register float  *dot = prm_dot[LNK].inrtl ;       /* prm. derivative */
   register float  *old = prm_old_dot[LNK].inrtl ;   /* prev. prm. der. */

   register float  t ;                               /* temporary variable */

   register float  cn =   1.5 / servo_loop_rate ;    /* integration */
   register float  co = - 0.5 / servo_loop_rate ;    /*  constants  */


   t = dot[0] ; pm[0] -= gn[0] * (cn * t + co * old[0]) ; old[0] = t ;
   t = dot[1] ; pm[1] -= gn[1] * (cn * t + co * old[1]) ; old[1] = t ;
   t = dot[2] ; pm[2] -= gn[2] * (cn * t + co * old[2]) ; old[2] = t ;

   t = dot[3] ; pm[3] -= gn[3] * (cn * t + co * old[3]) ; old[3] = t ;
   t = dot[4] ; pm[4] -= gn[4] * (cn * t + co * old[4]) ; old[4] = t ;
   t = dot[5] ; pm[5] -= gn[5] * (cn * t + co * old[5]) ; old[5] = t ;

   t = dot[6] ; pm[6] -= gn[6] * (cn * t + co * old[6]) ; old[6] = t ;
   t = dot[7] ; pm[7] -= gn[7] * (cn * t + co * old[7]) ; old[7] = t ;
   t = dot[8] ; pm[8] -= gn[8] * (cn * t + co * old[8]) ; old[8] = t ;
   t = dot[9] ; pm[9] -= gn[9] * (cn * t + co * old[9]) ; old[9] = t ;
 }


/*
**   Connect upwards to next link
*/

traverse_LNK+1() ;


   return;
 }
```

References

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

Asada, H., and Slotine, J.J.E. 1986. Robot Analysis and Control. Wiley.

Baillieul, J. 1985. Kinematic Programming Alternatives for Redundant Manipulators. *IEEE Int. Conf. on Robotics and Automation,* St. Louis.

Baker, D.R., and Wampler, C.W. II 1988. On the Inverse Kinematics of Redundant Manipulators. *Int. J. Robotics Res.* 7(2)

Bayard, D.S., and Wen, J.T. 1987. 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. Adaptive Control of Mechanical Manipulators. *I.E.E.E. Int. Conf. Robotics and Automation,* San Francisco.

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

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

Hogan, N. 1985. Impedance Control: An Approach to Manipulation: Part I - Theory. *J. Dynamic Systems, Measurements and Control.* 107(1):1-7.

Hsu, P. et al. 1987. Adaptive Identification and Control of Manipulators Without

Joint Acceleration Measurements. *I.E.E.E. Int. Conf. Robotics and Automation*, Raleigh, NC.

Kelly, R., and Carelli, R. 1988. Unified Approach to Adaptive Control of Robotic Manipulators *Proc. 27th Conf. on Dec. and Contr.*, Austin. pp 1598-1603.

Khatib, O. 1980. Commande Dynamique dans L'Espace Operationnel des Robots Manipulateurs en Presence D'obstacles. Docteur Ingenieur Thesis. Ecole Nationale Superieure de L'Aeronautique et de L'Espace, Toulouse, France.

Khatib, O. 1983. Dynamic Control of Manipulators in Operational Space. *6th IFTOMM Congress on Theory of Machines and Mechanisms.* New Dehli

Khosla, P., and Kanade, T. 1985. Parameter Identification of Robot Dynamics. *I.E.E.E. Conf. Decision and Control*, Fort Lauderdale.

Klein, C.A., and Blaho, B.E. 1987. Dexterity Measures for the Design and Control of Kinematically Redundant Manipulators. *Int. J. Robotics 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. on Systems, Man and Cybernetics.* 13(2):245-250.

Koditschek, D.E. 1987. Adaptive Techniques for Mechanical Systems. *Proceedings of the Fifth Yale Workshop on the Applications of Adaptive Systems Theory.*

Landau, I., and Horowitz, R. 1988. *I.E.E.E. Int. Conf. Robotics and Automation*, Philadelphia, PA.

Li, W., and Slotine, J.J.E. 1987. Parameter Estimation Strategies for Robotic Applications. *A.S.M.E. Winter Annual Meeting*, Boston, MA.

Li, W., and Slotine, J.J.E. 1988a. Indirect Adaptive Robot Control. *5th I.E.E.E. Int. Conf. Robotics and Automation*, Philadelphia, PA.

Luh, J.Y.S., Walker, M., and Paul, R.P.C. 1980. Resolved Acceleration Control of Mechanical Manipulators. *IEEE Trans. on Automatic Control.* 25(3):468-474.

Mason, M.T. 1986. Mechanics and Planning of Manipulator Pushing Operations. *Int. J. Robotics Res.* 5(3):53-71.

Mayeda, H., Yoshida, K., and Osuka, K. 1988. Base Parameters of Manipulator Dynamic Models. *5th I.E.E.E. Int. Conf. Robotics and Automation*, Philadelphia, PA.

Middleton, R.H. and Goodwin, G.C. 1986. Adaptive Computed Torque Control for Rigid Link Manipulators. *25th I.E.E.E. Conf. on Dec. and Contr., Athens, Greece.*

Narasimhan, S., Siegel, D.M., and Hollerbach, J.M. 1989. Condor: An Architecture for Controlling the Utah-MIT Dexterous Hand. *IEEE Trans. on Robotics and Automation.* 5(5):616-627.

Niemeyer, G., and Slotine, J.J.E. 1988. Performance in Adaptive Manipulator Control. *Proc. 27th Conf. on Dec. and Contr.*, Austin.

Niemeyer, G., and Slotine, J.J.E. 1989. Computational Algorithms for Adaptive Compliant Motion. *I.E.E.E. Int. Conf. Robotics and Automation*, Scottsdale, AZ.

Ortega, R., and Spong, M. 1988. *I.E.E.E. Int. Conf. Decision and Control*, Austin, TX.

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

Sadegh, N., and Horowitz, R. 1987. Stability Analysis of an Adaptive Controller for Robotic Manipulators. *I.E.E.E. Int. Conf. Robotics and Automation*, Raleigh, NC.

Salisbury, J.K. et al. 1988. Preliminary Design of a Whole-Arm Manipulator System. *I.E.E.E. Int. Conf. Robotics and Automation*, Philadelphia, PA.

Slotine, J.J.E., and Li, W. 1986. On The Adaptive Control of Robot Manipulators. *A.S.M.E. Winter Annual Meeting*, Anaheim, CA.

Slotine, J.J.E., and Li, W. 1987a. 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. 1987b. Adaptive Robot Control, A Case Study. *I.E.E.E. Int. Conf. Robotics and Automation*, Raleigh, NC.

Slotine, J.J.E., and Li, W. 1987c. Adaptive Strategies in Constrained Manipulation. *I.E.E.E. Int. Conf. Robotics and Automation*, Raleigh, NC.

Slotine, J.J.E., and Li, W. 1987d. Adaptive Robot Control - A New Perspective. *I.E.E.E. Conf. Decision and Control* L.A., CA.

Slotine, J.J.E., and Li, W. 1987e. On the Adaptive Control of Robot Manipulators. *Int. J. Robotics Res.* 6(3).

Slotine, J.J.E., and Li, W. 1988. Adaptive Manipulator Control: A Case Study. *I.E.E.E. Trans. Autom. Control*, 33, 11.

Slotine, J.J.E., and Li, W. 1989. Composite Adaptive Robot Control. *Automatica*, 25(4).

Slotine, J.J.E. and Li, W. 1990. Applied Nonlinear Control. Prentice-Hall.

Townsend, W.T. 1988. The Effect of Transmission Design on Force-Controlled

Manipulator Performance. Technical Report AI-TR1054. Cambridge, MA: Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Walker, M.W. 1988. An Efficient Algorithm for the Adaptive Control of a Manipulator. *Proc. 5th Int. Conf. Robotics and Automation*, Philadelphia.

Yuan, J.S. 1988. Closed-Loop Manipulator Control using Quatemion Feedback. *J. Robotics and Automation*, 4(4):434-440.