Roles of Knowledge in Motor Learning

by

Christopher Granger Atkeson

A.B., Harvard College (1981)
S.M., Harvard University (1981)

Submitted to the Department of
Brain & Cognitive Sciences
in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY
IN BRAIN & COGNITIVE SCIENCES

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1986

Signature of Author .....
Department of Brain & Cognitive Sciences
September 1, 1986

Certified and Accepted by .....
Professor Emilio Bizzi
Eugene McDermott Professor in the Brain Sciences and Human Behavior
Director, Whitaker College
Chairman, Department of Brain & Cognitive Sciences

Roles of Knowledge in Motor Learning

by

Christopher Granger Atkeson

Submitted to the Department of Brain and Cognitive Sciences on September 1, 1986 in partial fulfillment of the requirements for the degree of PhD.

Abstract

The goal of this thesis is to apply the computational approach to motor learning, i.e., describe the constraints that enable performance improvement with experience and also the constraints that must be satisfied by a motor learning system, describe what is being computed in order to achieve learning, and why it is being computed. The particular tasks used to assess motor learning are loaded and unloaded free arm movement, and the thesis includes work on rigid body load estimation, arm model estimation, optimal filtering for model parameter estimation, and trajectory learning from practice. Learning algorithms have been developed and implemented in the context of robot arm control.

The thesis demonstrates some of the roles of knowledge in learning. Powerful generalizations can be made on the basis of knowledge of system structure, as is demonstrated in the load and arm model estimation algorithms. Improving the performance of parameter estimation algorithms used in learning involves knowledge of the measurement noise characteristics, as is shown in the derivation of optimal filters. Using trajectory errors to correct commands requires knowledge of how command errors are transformed into performance errors, i.e., an accurate model of the dynamics of the controlled system, as is demonstrated in the trajectory learning work. The performance demonstrated by the algorithms developed in this thesis should be compared with algorithms that use less knowledge, such as table based schemes to learn arm dynamics, previous single trajectory learning algorithms, and much of traditional adaptive control.

Thesis Supervisor: Dr. Emilio Bizzi
Eugene McDermott Professor in the Brain Sciences
and Human Behavior
Director, Whitaker College
Chairman, Department of Brain & Cognitive Sciences

Acknowledgements

I would like to thank the many people who have helped me along the way.

Financial support was provided by a National Science Foundation Graduate Fellowship and then a Whitaker Health Sciences Fund Graduate Fellowship. The thesis describes research done at the Whitaker College of Health Sciences, Technology, and Management, the Department of Brain and Cognitive Sciences, and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support was provided in the Whitaker College by grants AM26710 and NS09343 from the National Institutes of Health. Support for research at the Artificial Intelligence Laboratory was provided in part by the Systems Development Foundation and the Defense Advanced Research Projects Agency under Office of Naval Research contracts N00014-80-C-050, N00014-82-K-0334, and N00014-85-K-0124.

Contents

Chapter 1

Introduction

Learning is currently an important distinguishing characteristic between biological systems and machines. Humans and animals tend to improve their motor performance with experience, while machines often repeat the same errors, day in and day out. An important step towards understanding ourselves, and also building more useful machines, is to begin to understand how learning is achieved.

The goal of this thesis is to apply the computational approach (Marr, 1977; Marr, 1982; Hildreth and Hollerbach, 1985) to motor learning. The point of view taken is that learning is an information processing problem: a learning algorithm modifies commands using information from previous experience so as to improve performance. A proposed methodology for computational neuroscience and artificial intelligence includes the following steps: (Marr, 1977; Winston, 1984)

1. Identify a particular information processing problem or task.

2. Formulate a computational theory: Expose constraints that enable (or restrict) performance, describe what is being computed and why, select or devise appropriate representations, define procedures to solve problem.

3. Devise and implement actual algorithms to solve problem.

4. Verify and evaluate the performance of the implementation via experiments.

The particular tasks used to assess motor learning are loaded and unloaded free arm movement. The computational theory developed applies model based feedforward control to the arm control problem, and involves elements of control theory and system identification. In order to effectively use model based control one must have an accurate model, and the first half of the thesis describes how to build models of particular types of loads and arms. One constraint that is heavily relied upon is that the arms and loads described in this thesis can be adequately modelled using rigid body dynamics. This allows us to use models with small numbers of parameters and to make powerful generalizations between dissimilar movements.

The implementation of the model building procedures reveals that good models can be identified quickly and are useful for control. However, the models used to represent the arm and load dynamics have limited degrees of freedom, and cannot represent the full complexity of the true dynamics. The second half of the thesis describes how a command can be refined for a particular movement on the basis of practice, allowing representation of fine details of the dynamics. The computational theory for the learning from practice described here involves making explicit what the learning operator that maps performance errors to command corrections should be. The implementation reveals fast and effective trajectory learning.

A major contribution of this thesis is to demonstrate some of the roles of knowledge in learning. Powerful generalizations can be made on the basis of knowledge of system structure, as is demonstrated in the load and arm model estimation algorithms. Improving the performance of parameter estimation algorithms used in learning involves knowledge of the measurement noise characteristics, as is shown in the derivation of optimal filters. Using trajectory errors to efficiently correct commands requires knowledge of how command errors are transformed into performance errors, ie. an accurate model of the dynamics of the controlled system, as

is demonstrated in the trajectory learning work. The performance demonstrated by the algorithms developed in this thesis should be compared with algorithms that use less knowledge, such as table based schemes to learn arm dynamics, previous based trajectory learning algorithms, and much of traditional adaptive control.

In order to evaluate the computational theory developed in this thesis learning algorithms have been developed and implemented in the context of robot arm control. Although the experimental work in this thesis is restricted to work with robots, the insights gained have a much broader relevance. The adaptive mechanisms used to implement learning may differ in different domains, but at a computational level the issues and principles remain the same.

1.1 Choices and assumptions

1.1.1 Working definition of learning

The working definition of learning for the purposes of this thesis is *improvement of performance with experience*. Since the experimental work involves only machines, motivational levels, development, and other factors that make analysing biological performance improvement difficult are not an issue.

Two kinds of learning are studied. The first involves building and refining internal models of one's self and the external world, and using those models to generate the appropriate actuator commands, and to guide processing of sensory data. The second involves using internal models to transform performance errors into command corrections.

1.1.2 Focus on arm trajectory control

The effector system we will focus on will be a mechanical robot arm, and we will mainly be concerned with improving execution of free arm trajectories, either with or

without a load. The types of learning appropriate for this sort of task are probably different from the types of learning involved in riding a bicycle or juggling. The ideas in this thesis will hopefully generalize and be useful in studying these other tasks, however.

To provide a framework for what follows let us examine a typical robot control architecture. We can divide the robot control problem into deciding what to do (*planning*) and doing it (*execution*). For current robots planning is usually accomplished by a human programmer, although a goal of this and much other research is to incrementally automate robot programming.

Since we are concerned with robot trajectory execution, a plan is a complete specification of the motion of the robot in some coordinate system. Often the plan is expressed in *task coordinates,* and in order to execute this plan we convert task coordinates to arm coordinates. This process is referred to as *inverse kinematics.*

The next step in making the robot follow the desired trajectory is supplying appropriate commands to the actuators. The simplest approach to this is to generate these commands by measuring the difference between where the arm is and where it is supposed to be at any instant in time and using some function (usually a linear function) of this error as the drive signal to the actuators. This is what is known as *feedback control.* This type of control is useful in compensating for unpredicted disturbances.

One problem with pure feedback control is that it requires errors in order to generate any drive signals. We can avoid this problem by calculating our best guess as to the appropriate command signals to apply to the actuators to follow the desired trajectory exactly. This precalculated command is added to the feedback command during execution of the motion. I will refer to applying a precalculated actuator command as *command feedforward control* or simply *feedforward control.* The process of calculating the appropriate actuator command from a specification of the desired robot motion is referred to as *inverse dynamics.*

At this level of detail the same problems of coordinate transformation, prediction of appropriate actuator (muscle) commands, and compensating for unpredicted disturbances must be addressed by biological systems.

As a starting point for our discussion of learning, let us ask the question, "How might we go about building learning into this performance architecture?" or equivalently (from the point of view of this thesis) "What information can be extracted from past experience to improve present and future robot performance?"

Making better plans: Planning can be improved in several ways: the world models used in planning can be refined, better methods for solving the given task can be generated, and the planning methods themselves can be changed. These processes are assumed to be independent of improving execution of a given plan, and will not be addressed in what follows. We will take the plan as fixed.

Improving execution of fixed plans: The thesis will focus on execution and making the effector system obey a given plan more closely. All components of execution can be improved using experience. One way of improving the various coordinate transformations involved in robot control is to refine the kinematic model of the robot using measurements from redundant sensing such as vision of the robot tip and joint angle sensing. To improve feedforward control we could refine the dynamic model of the robot. We could also design a better feedback controller using the past history of controller actions.

Assume kinematics is essentially perfect: We will assume our kinematic model of the robot is essentially perfect, as there is already much research going on in the area of kinematic calibration.

Assume a fixed feedback controller: As there is currently much research in adaptive feedback control we will also take our feedback controller to be fixed.

In a complete robot learning system plan learning, kinematic learning, adaptive feedback control, and the types of learning discussed in this thesis will all occur simultaneously. An important point to keep in mind is that they can be handled by

essentially independent modules.

Focus on dynamics and control: In this thesis we will discuss how to build and refine a model of robot dynamics to be used for predicting the appropriate actuator commands to drive the robot (feedforward control). We will discuss how to handle certain types of loads. We will also discuss how to build a model of noise in sensory data, and how to optimally filter that noise in the model building process. In addition we will show the role of the robot model as the learning operator during movement practice, ie. the robot model transforms trajectory following errors into feedforward command corrections.

1.2 Outline of thesis

1.2.1 Format of thesis

Many of the chapters are designed to be independent papers. The price of trying to write a thesis whose chapters are publishable is some repetition and seeming multiplicity of goals.

1.2.2 Introduction and Previous Work

This chapter attempts to give the reader an idea of where he is going, why, and some of the landmarks to look for. Chapter 2, Previous Work, reviews some of the related research. Other related research is reviewed in the appropriate chapters.

1.2.3 Arm and Load Identification

Chapters 3, 4, and 5 describe building models for one's own arm and any rigid body load. Appendix A describes the use of an identified model for the control of a particular robot arm.

This work demonstrates the power of knowing the system structure for learning. In this case the knowledge of the rigid body dynamics structure dramatically reduces the number of parameters to be learned and increases the learning rate compared to tabular learning schemes (Albus, 1975a, 1975b, Raibert, 1977, 1978, Raibert and Horn, 1978). More importantly, knowing the system structure allows much more powerful generalizations to be made. There are two constraints that can be used as sources of generalization. The first is smoothness of the input/output transformation. This constraint can be used to generalize between similar movements and is relied on heavily by tabular learning approaches. The constraint used in this work, system structure, is more powerful in that information gained from one movement can be used to guide quite dissimilar movements.

The crucial insight of these chapters is that systems dominated by rigid body dynamics can be described by a relatively small set of inertial parameters that appear linearly in the dynamics equations. Estimating these parameters turns out to be simple, as the theory and implementations show. We start with load identification (Chapter 3) because that is the simplest version of the story, and with very little additional work we apply the same ideas to arm identification (Chapter 4).

The reliance on system structure to achieve generalization does have a price, however. The models do not represent well deviations from the assumed structure, which are always present to some degree. Thus, on any particular trajectory execution a rigid body dynamics model will have small errors. Changing the model parameters to fit this trajectory more exactly will degrade performance on other trajectories. What is required is an additional level of modelling that allows representation of fine details of the dynamics. This correction model and how it is learned is described in Chapter 6, Single Trajectory Learning.

Chapter 5, Optimal Filtering For Parameter Estimation, discusses the practical issues of handling noisy measurements. How to optimally process the incoming sensory data is determined by models of the measurement noise. Improving the per-

formance of parameter estimation algorithms used in learning involves knowledge of the measurement noise characteristics, which are estimated from redundant sensing. Thus, the learning elements themselves learn, in this case by building better models of the sensors and their interaction with the world.

An insight that simplifies system identification is to use sensing to decouple different system identification problems. We use a wrist force/torque sensor that might allow us to estimate load parameters independently of the arm dynamics, for example. More generally, covering an arm or any effector system with a sensory barrier such as a tactile sensing skin allows us to separate identifying internal system dynamics and identifying external system dynamics and disturbances. Without a sensory barrier the combined system of arm plus external world may be too complex to identify robustly. Decoupling system identification problems allows different identification procedures to be applied to the different subsystems. In the case of an arm with variable loads, we expect the arm dynamics to be only slowly varying, while load dynamics change rapidly as loads are picked up or put down. Widely different rates and types of system change call for different system identification algorithms to track the changes. Another reason for decoupling arm and load identification is that the arm structure is relatively constant while the dynamics of different loads can have quite different structures. Arm identification can be based on a fixed model structure, while a complete load identification system must handle many different model structures.

It is important to keep in mind the broad relevance of the model building approach suggested here. It is true that other sources of dynamics such as friction and actuator dynamics play an important role in the dynamics of many current robot arms. However, 1) rigid body dynamics is often a source of coupling dynamics between the joints, 2) Since friction and actuator dynamics are usually isolated at particular joints their identification reduces to a single input single output modelling problem rather than the more difficult multiple input multiple output problem

solved here, and 3) the same process of hypothesizing a model structure and estimating parameters can be used to identify friction and actuator dynamics and can even be added to the linear parameter estimation algorithms used in this work.

1.2.4 Single Trajectory Learning

Chapter 6, Single Trajectory Learning, addresses the issue of using practice to improve the execution of a particular trajectory. The crucial insight of this chapter is that one must use an accurate model of the controlled system to make sense of trajectory errors, ie. convert the errors into corrections to feedforward commands. Without an accurate model attempts to improve trajectory performance will most likely degrade performance. This work demonstrates the role of knowledge in analysing past behavior and correcting previous mistakes, and should be compared to the proliferation of ad hoc trajectory learning schemes.

We are able to mathematically analyse the effect of various proposed single trajectory learning algorithms, which tells us why one learning operator works better than another. The most important result is that the convergence rates of the algorithms are determined by the quality of the models used. We can put mathematical bounds on acceptable modelling error for the linear case.

The model used in the trajectory learning work is the identified arm model presented in Chapter 4. Thus, starting with only knowledge of system structure, we have demonstrated a system that can build a general model of itself after only three or four movements, and then can learn to execute any particular trajectory to almost the limits of the system repeatability in an additional three or four movements.

1.2.5 Future research

The final chapter discusses what to do next. This work suggests a program of research on the quantitative psychophysics of motor learning. Rigid body load iden-

tification is a first step towards general force perception. The extension of the trajectory learning algorithm to improve performance on a group of similar trajectories is discussed. Also included is a discussion of performance simplification, which reduces the need for detailed internal dynamic models by restricting possible motor performance.

Chapter 2

Previous Work

This chapter spells out the theoretical background and philosophy of our approach, and surveys the relevant engineering literature. This thesis addresses the following questions:

1. How can we generate appropriate actuator command signals to drive a controlled system along some desired trajectory?

2. How can we modify the actuator command signals to improve trajectory execution on the same and similar trajectories, given data from previous movements?

This chapter describes the general approach taken and contrasts it with approaches taken by others. In summary, we

1. Describe three basic types of controllers:

   (a) Feedback

   (b) Disturbance Feedforward

   (c) Command Feedforward

   and their roles,

2. Describe why we have chosen to pursue command feedforward control, and specifically adaptive command feedforward control, and

3. Describe previous work on adaptive command feedforward control.

2.1 Introduction: Three types of control and adaptive control

The motivating control problem: The areas of control theory that one is interested in are often shaped by the systems one desires to control. The emphasis here is on the control of biological and robotic arms during the execution of free trajectories.

Focusing on execution: We are taking a modular approach to motor learning, and have split learning into two parts: making better plans and following plans more closely. In this chapter we assume there is a given plan or desired trajectory and our goal is to execute it as accurately as possible. Thus we focus on the controller, the element which transforms plans into actuator commands, and how we might refine the controller with experience. Following plans more closely is the goal of adaptive control, the branch of control theory that addresses how to improve controllers based on experience.

Three types of control: A control system can be divided into three parts: the feedback controller and two feedforward controllers (Ogata, 1970; Takahashi, Rabins, and Auslander, 1970; Astrom and Wittenmark, 1984). Figure 2.1 shows the general form of such a system.

Feedback control is any control action based on the actual state history of the controlled system, and feedforward control is all other control actions (D'Azzo and Houpis, 1966; Astrom and Wittenmark, 1984). We can distinguish two types of feedforward control: feedforward commands may be based on measurements of other

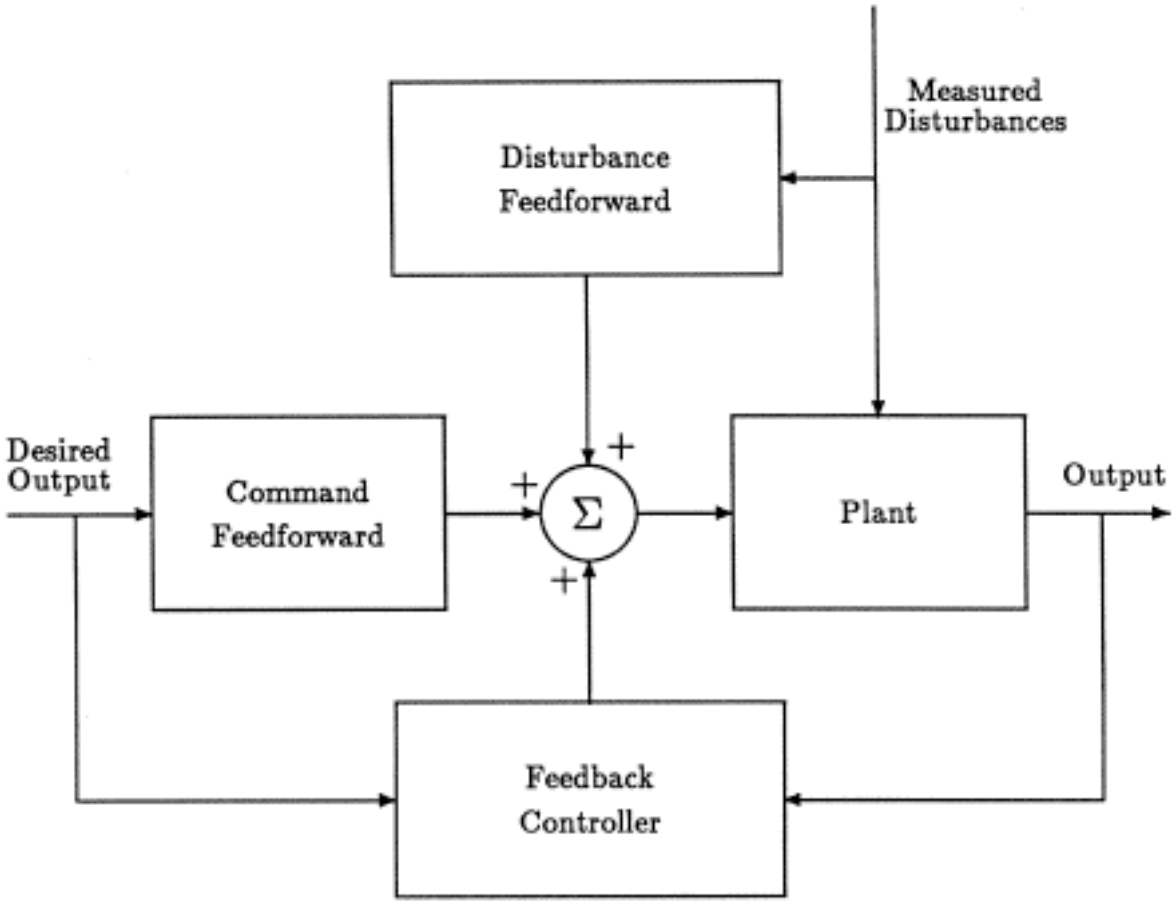


Figure 2.1: The three parts of a control system

variables that affect the controlled system but are not affected by it (disturbance feedforward) or the commands may be based on predictions of the control signals necessary to drive the controlled system along some desired trajectory (command feedforward).

Three types of adaptive control: Corresponding to the different parts of the controller there are different types of adaptive control. Adaptive feedback control involves adjusting the feedback controller. Adaptive feedforward control may involve adjusting gains from measured signals (adaptive disturbance feedforward control) or improving the models used for control signal prediction (adaptive command feedforward control). We will focus on improving the models used for command feedforward control.

2.1.1 Questions that have come up

Generality of approach: Each component of the control system (plant, feedback controller, disturbance feedforward, command feedforward) may be a complex non-linear transformation. We will initially use simple examples (typically linear systems) to present concepts clearly, but the reader should keep in mind that much of what is said applies to more general systems.

Reference and feedforward signals: Note that there are many equivalent ways to think about the command feedforward part of the control system, especially if the system components are linear (Horowitz, 1963). We can think of a reference signal to the feedback controller as another feedforward signal. For conceptual clarity, we want to separate modifiable feedforward actuator commands from reference signals to the feedback controller. The reference signals will always be the desired trajectory.

Terminology: Although the control terminology is not completely standardized the term "feedforward" is used quite often in the literature; for example, feedfor-

ward control for measured disturbances (Grabbe, Ramo, and Wooldridge, 1961; Peschon, 1965; Savas, 1965; Weyrick, 1975; Marshall, 1978; Owens, 1978; Anand, 1984; Schwarzenbach and Gill, 1984; Leigh, 1985), feedforward control for command inputs (Tou, 1959; Webb, 1964; Eveleigh, 1972), feedforward control for disturbances and commands (Macmillan, 1951; Doebelin, 1962; Shinskey, 1963; Woolverton and Murrill, 1967). In the Russian literature the "Theory of Invariance" is often invoked to show how and why feedforward control should be used to make the system error independent of commands and disturbances. This has also been referred to as "Poncelet's principle" (Preminger and Rootenberg, 1964). Command feedforward is often referred to as an input or reference prefilter, especially if this signal is used as the reference to the feedback controller. Combined feedback and feedforward control has been referred to as a "two degree of freedom control system" (Horowitz, 1963). There is occasionally some confusion in the use of the term feedforward. "Feedforward controller" has been used to refer to any element in the forward path of the closed loop controller (Raven, 1978; Saridis, 1977). These elements are clearly part of the feedback controller.

2.2 The Feedback Controller

Feedback control is the only way to handle unmeasured or unpredicted disturbances, modelling errors, and to stabilize unstable plants, and thus plays a critical role in control in general and in robotics in particular. Much effort is going into robot feedback controller design, sometimes to the exclusion of considering other types of control. However, there are limits to what can be done with feedback control, and feedforward control can usefully augment a feedback controller. It should be noted that the feedback controller can be designed to a large extent independently of any feedforward control, separating issues of robustness to disturbances and modelling errors, and command tracking (Horowitz, 1963).

2.2.1 Limits on feedback controller

Control of terminal compliance or, more generally, impedance has been proposed as a goal for robotic control. Such control may require limiting feedback gains, if the desired compliance is implemented as a low gain position servo. For non-redundant robots with no terminal force/torque sensing choosing an impedance specifies the feedback controller completely.

The use of a force sensors at the interface between the robot and its load or environment may allow differential rejection of modelling errors and external forces. In practice, however, this has not yet been shown to work well.

In biological systems there are additional limits on the possible feedback gains due to transmission delays. In general, non minimum phase elements such as delays (which also occur in machines) and right half plane zeros set limits on maximum feedback gains as do modelling error, ignorance of plant dynamics, parameter variations, and measurement or command noise. (Astrom and Wittenmark, 1984)

2.2.2 Plenty of work going on in adaptive feedback control

Approaches to adaptive control have been almost exclusively focused on modifying the feedback controller (W. D. T. Davies, 1970; Tsypkin, 1971; Wittenmark, 1975; Saridis, 1977; Landau, 1979; Harris and Billings, 1981; Isermann, 1982; Astrom, 1983; Landau, Tomizuka, and Auslander, 1983; Astrom and Wittenmark, 1984; Goodwin and Sin, 1984; Voronov and Rutkovsky, 1984; Yale, 1985).

There has also been great interest in adaptive feedback control in robotics. The added complexity of an adaptive controller is typically justified on the basis of the complexity of the robot dynamics, the belief that the parameters in robot models are difficult to estimate, that loads will be unknown, and that there will be significant modelling errors caused by friction (static and dynamic), joint compliance, actuator modelling errors, and link flexibility. Many of the justifications for adaptive con-

trol are not valid, and it is not clear that adaptive feedback controllers have any advantage over well designed non-adaptive controllers. Furthermore, the limitations on feedback control discussed previously limit the performance of adaptive feedback controllers just as they limit fixed gain controller performance.

2.3 Feedforward Control

Feedforward control is any control action that is independent of the previous actual state history of the plant, and is used to improve command following and counteract predictable or measurable disturbances. Feedforward control has been used for a long time (Graham, 1946; Moore, 1951; Preminger and Rootenberg, 1963), and in fact many control systems use feedforward control exclusively and thus are pure open loop controllers.

Aerospace applications: Feedforward control is used extensively in aerospace applications and is related to the use of optimal control there. Typically a rocket trajectory is planned according to some cost criterion and the thrust history necessarily to follow that trajectory is pre-calculated. The feedback controller counteracts deviations from the planned trajectory by modifying the nominal thrusts. Adaptive feedforward control has not been emphasized in these applications as the same rocket is rarely used twice.

Process control applications: Feedforward control has also been used in process control systems (Macmillan, 1951; Shinskey, 1963; Savas, 1965; Weyrick, 1975; Marshall, 1978; Shinskey, 1979). These applications tend to focus on counteracting measured disturbances since desired outputs tend to be constant for long periods of time and outputs during transients or changes in production are usually discarded. Also, process control is characterized by long delays, large time constants (slow processes), and measurable disturbances.

Robotics applications: Feedforward control is an important part of robot control (Orin et al., 1979; Liegeois, Fournier, and Aldon, 1980; Paul, 1981; Brady et al., 1982; Asada, Kanade, and Takeyama, 1983; Gerstenberger, 1985; Hollars and Cannon, 1985; Liegeois, 1985), although there is some variation in the terminology: "Inverse problem" (Paul, 1972); "Computed torque" (Markiewicz, 1973; Bejczy, 1974); "Two stage" control (Vukobratović and Potkonjak, 1982; Vukobratović and Stokić, 1982).

There are several reasons for the emphasis on feedforward control in robotics: 1) Often the task or movement trajectory is specified in advance, and 2) the forces necessary to drive the robot along the specified trajectory are usually dominated by the forces necessary to drive the robot itself, and not the load, and thus are to a large extent predictable. 3) Most robots must interact compliantly (i.e. with low stiffness) with the environment and thus feedback gains are limited. 4) Often the task or trajectory is repetitive and the disturbances are the same on each execution. 5) Computers are typically available for the generation of complex feedforward commands. The goal of robot trajectory control is to minimize deviations from an arbitrary desired trajectory, which with a limited gain feedback controller requires accurate models for the generation of feedforward commands and the ability to refine those feedforward commands on repeated executions of the same trajectory.

Biological applications: Biologists have used the concept of feedforward control to understand commands based on measured signals and commands based on predicted forces (Szentágothai and Arbib, 1975). The vestibular ocular reflex is an example of both disturbance feedforward control and adaptive disturbance feedforward control. Fast limb movement is hypothesized to be driven by command feedforward control, as the delays in feedback control in biological systems limit possible gains and feedback effectiveness.

Technological limitations: To a certain extent the use of feedforward control has been limited in the past by the available technology (Peschon, 1965; Savas,

1965; Isermann, 1981; Isermann, 1982; Astrom and Wittenmark, 1984). The cost of analog electrical, mechanical, pneumatic, fluidic, or hydraulic controller circuits is proportional to their complexity, and implementing derivative operators, nonlinear operations, time delays, or functions such as $\sin(\theta)$ is relatively difficult. These circuits are restricted to be causal processors, and thus derivatives of the command must either be provided to the controller or approximated (An interesting mechanical differentiator is presented in Doebelin, 1962, p.286). These derivatives are typically useful in feedforward control. Furthermore, the reliability of analog circuitry is inversely proportional to its complexity, and this has encouraged the use of simple controllers. Some mechanism for information storage and retrieval is useful for feedforward control, as we shall later demonstrate, and in the past this was restricted to such technology as mechanical cam following. The introduction of computer control and the dramatic drop in the cost of computer hardware has greatly expanded the range of possible controller designs.

2.3.1 Feedforward Controller Design

It is well known that the ideal feedforward controller incorporates a model of the inverse of the plant (Graham, 1946; Moore, 1951; Tou, 1959; Shinskey, 1963; Eveleigh, 1972; Weyrick, 1975; Owens, 1978; Shinskey, 1979; Isermann, 1981; Isermann, 1982; Astrom and Wittenmark, 1984). One implication of this is that adaptive feedforward control involves building better models of the plant.

The need to invert the plant immediately raises the question of what to do when the inverse does not exist or is not realizable. This occurs when the plant has any of the following features:

1. **Time delays:** The inverse of a time delay is a time advance, which is not realizable.

2. **Right half plane zeros:** A causal inverse of a right half plane zero is unstable,

and a stable inverse is not realizable since it requires control actions in the infinite past.

3. **Many to one mappings:** A plant may map many sets of inputs into the same outputs, and thus formally the plant inverse does not exist. This is not typically a real problem, as additional constraints can be added to resolve the many to one mapping, or a particular inverse can be chosen.

4. **Singularities:** Points or sets of points at which a nonlinear or time varying linear plant mapping changes dimension may complicate inverting the plant. We will rely on the planning stage to avoid this problem, at least for now.

We see that realizability of the plant inverse is a critical issue in feedforward command design. This leads us to distinguish disturbance feedforward and command feedforward.

1. **Future commands are known:** A disturbance is assumed to be unpredictable, or at least to have an unpredictable component, while we assume that commands are known as far into the future as necessary. Of course, the future command may change, and this must be accommodated by the command feedforward controller. Some amount of acausality in command feedforward generation is acceptable. The disturbance feedforward controller, however, must be a causal processor.

2. **Commands have negligible noise:** In addition, disturbance measurements may be corrupted by noise, while commands, being internal to the controller, are assumed to be have negligible noise. Thus, derivative operators can accurately take derivatives of the command.

The goal of disturbance feedforward controller design is to design a causal and stable system whose input is disturbance measurements and whose output is ap-

propriate control actions. Approaches to designing such a system are discussed in (Isermann, 1981).

It is possible to treat command feedforward control as a type of disturbance feedforward control and attempt to design a causal command filter. This is not necessary the best approach, however. The goal of command feedforward control is to design the particular actuator command signal for a particular desired trajectory. This processing need not be causal. To handle a plant delay it is possible simply to shift the command backwards in time. To handle a right half plane zero it is possible to design commands that do not grow to infinity. This typically requires that the command starts in the infinite past. Since this is clearly not possible this type of command must be truncated, leading to unavoidable trajectory following errors.

2.4 Previous Approaches To Adaptive Feedforward Control

2.4.1 Self Tuning Adaptive Feedforward Control

A straightforward approach to adaptive feedforward control is to treat command and disturbance feedforward control identically and simply automate existing feedforward controller design procedures. Initially the plant and any disturbance effects must be identified, and then a controller is designed from that. There are typically two steps to identification: choosing a model structure and then estimating the free parameters in that structure. By making different choices in model structures, parameter estimation procedures, and feedforward controller design many different adaptive feedforward control schemes can be derived. This approach is analogous to "Self Tuning" adaptive feedback control (Isermann, 1981; Isermann, 1982; Astrom, 1983; Astrom and Wittenmark, 1984). The perils of right half plane zeros are typically avoided by using a model structure that cannot contain right half plane zeros.

This may lead to an approximation of the true plant, but the inverse of the model will always be stable.

Woolverton and Murrill (1967) suggest choosing initial parameters for a second order feedforward controller from experiments on the controlled process and then a trial and error exploration of the nearby parameter space to find the optimum parameter set.

Bristol (1967) chooses a simple linear model structure for a heat exchanger, $V = a \cdot L + b$, and closes an integrating feedback loop around the free parameters a and b to continuously update this model. The model is concurrently used to generate feedforward commands, V, from load measurements, L. An objective of this scheme is to only include operations easily implementable in analog hardware and the final controller was pneumatic.

Bernard and Lefkowitz (1962) chose a more complicated process model

$$P(s) = \frac{Ke^{-Ds}}{Ls + 1} \tag{2.1}$$

where P(s) is the process transfer function, K is the gain, D is the delay, and L is a time constant. Initially K, D, and L are estimated from previously collected data, and then D and L are continuously updated. This model is used to generate bang-bang control responses to step changes in measured disturbances.

Wittenmark (1973) derives a self-tuning minimum variance feedforward controller.

Schumann and Christ (1979) present the most extensive exploration of self tuning adaptive feedforward controllers. They make explicit three steps:

1. Choose structural parameters of process and disturbance models: model order, time delays, sampling interval, etc.

2. Use some recursive parameter estimation scheme to identify the model parameters. Typically recursive least squares is used although if there is significant

process noise an unbiased scheme such as recursive maximum likelihood is more appropriate.

3. Design a feedforward controller based on the identified model. Here we assume the certainty equivalence principle is valid.

The process and disturbances are modelled by separate ARMA models. Since their model structure allowed right half plane zeros they needed feedforward design procedures that handled this problem. They explored seven feedforward control design procedures.

Elicabe and Meira (1983) extend the adaptive model following control approach (Landau, 1979) to adapt a feedforward controller to handle measurable disturbances.

Widrow and colleagues (Widrow, Walach, and Shaffer, 1983; Widrow and Walach, 1983; Widrow et al., 1981; Anderson and Johnstone, 1981; Widrow, McCool, and Medoff, 1978) model the plant as an all pole process, thus avoiding the right half plane zero problem. They use an LMS (least mean square) algorithm to continuously model the plant. They approximate the true plant in a least squares sense, and are able to drive the plant approximately along a trajectory after a certain delay.

Adaptive feedforward control in robotics

2.4.2 Adaptive Global Models

One approach to adaptive command feedforward control is to attempt to identify a model of the robot dynamics, and invert the model, or to directly identify an inverse model of the plant. In this section we will outline approaches which attempt to model the robot dynamics under all conditions with one model. We refer to this approach as the global modelling approach.

One approach for such global modeling is to derive a rigid body dynamics model of the arm and identify the parameters of that model. Examples of such work are

discussed in Chapter 4 and include Mayeda, Osuka, and Kangawa (1984), Mukerjee (1984), Mukerjee and Ballard (1985), Olsen and Bekey (1985), Neuman and Khosla (1985), An, Atkeson, and Hollerbach (1985), Khosla and Kanade (1985).

Goor (1985a, 1985b) emphasized the actuator dynamics and claimed a decoupled third order model of each joint of the robot was more appropriate. He incorporated a third order command feedforward filter and developed an algorithm that estimated the coefficients of that filter in real time. He also emphasized the necessity of commanded trajectories to be executable, i.e. not require infinite inputs.

A key issue in adaptive global modeling is that the identified model should be constant or at least independent of the state of the plant, even during large motions of the robot. If this is not the case then it is not clear that there is enough data during the movement of the robot to identify a new model at each significantly different state.

Another key issue is to insure that the data used to estimate the robot model cover a wide range of states and inputs. Otherwise structural modelling errors may drive the parameter estimation procedure to seriously distort the estimated parameters to fit a small range of data. This can cause predictions for other ranges of data not used in the estimation to deteriorate, and the advantage of global models, generalization, is lost.

2.4.3 Lookup Table Based Approaches: Local models

Lookup table based approaches have been suggested in the past as a solution to the model evaluation problem: given an existing model can we compute the necessary inputs to achieve the desired outputs fast enough? (Albus, 1975a, 1975b; Raibert, 1977, 1978; Raibert and Horn, 1978) One way to fill the tables is using some form of learning. Tables do provide an easily modifiable representation of a transformation. However, a straightforward implementation leads to huge table sizes, and consider-

able effort must be expended in designing how data is actually stored, and how it is put in and taken out of the table. A major flaw of tabular approaches is that it is difficult to take advantage of the structure of the arm dynamics, so much more data has to be collected to learn to the same level of performance as the algorithms described in this thesis, and generalizations between dissimilar movements are difficult to make.

We will eventually be suggesting table based approaches to form a middle level of models: local models. Global models help us achieve generalization, and single trajectory models allow us to follow a single trajectory with zero error after practice. Lookup table based local models can help us modify commands for a particular trajectory to achieve similar trajectories. An important distinction between our approach and previous approaches is that we propose using a hierarchy of several different types of models to gain the benefits of each and the drawbacks of none.

2.4.4 Iterative Trajectory Learning: Single trajectory models

At the opposite extreme of global modeling is the modeling of the feedforward command for a single trajectory. Through repeated attempts to follow a single specified trajectory the feedforward command is refined, and ultimately the trajectory is followed with zero error. The key issues here are the stability and convergence rate of the iterative process, and how to design the learning operator. One drawback of this approach is that it only produces the appropriate command for a single trajectory. There is little guidance as to how to modify that command for similar trajectories. This approach is discussed in Chapter 6, Single Trajectory Learning.

Chapter 3

Rigid Body Load Inertial Parameter Estimation

3.1 Abstract

A method for estimating the mass, the center of mass, and the moments of inertia of a rigid body load during general manipulator movement is presented. The algorithm is derived from the Newton-Euler equations and incorporates measurements of the force and torque from a wrist force-torque sensor and of the arm kinematics. The identification equations are linear in the desired unknown parameters, which are estimated by least squares. We have implemented this identification procedure on a PUMA 600 robot equipped with an RTI FS-B wrist force-torque sensor, and on the MIT Serial Link Direct Drive Arm equipped with a Barry Wright Company Astek wrist force-torque sensor. Good estimates were obtained for load mass and center of mass, and the forces and torques due to movement of the load could be predicted accurately. The load moments of inertia were more difficult to estimate.

[1]This chapter is a revised version of (Atkeson, An, and Hollerbach, 1985a)

3.2 Introduction

This chapter presents a method of estimating all of the inertial parameters of a rigid body load using a wrist force-torque sensor: the mass, the moments of inertia, the location of its center of mass, and the object's orientation relative to a force sensing coordinate system. The procedure has three steps:

1. A Newton-Euler formulation for the rigid body load yields dynamics equations linear in the unknown inertial parameters, when the moment of inertia tensor is expressed about the wrist sensing force coordinate system origin.

2. These inertial parameters are then estimated using a least squares estimation algorithm.

3. The location of the load's center of mass, its orientation, and its principal moments of inertia can be recovered from the sensor referenced estimated parameters.

In principle, there are no restrictions on the movements used to do this load identification, except that if accurate estimation of all the parameters is desired the motion must be sufficiently rich (ie., occupy more than one orientation with respect to gravity and contain angular accelerations in several different directions). In practice, however, special test movements must sometimes be used to get accurate estimates of moment of inertia parameters.

There are several applications for this load identification procedure. The accuracy of path following and general control quality of manipulators moving external loads can be improved by incorporating a model of the load into the controller, as the effective inertial parameters of the last link of the manipulator change with the load. The mass, the center of mass, and the moments of inertia constitute a complete set of inertial parameters for an object; in most cases, these parameters form a good description of the object, although they do not uniquely define it. The object

may be completely unknown at first and an inertial description of the object may be generated as the robot picks up and moves the object. The robot may also be used in a verification process, in which the desired specification of the object is known and the manipulator examines the object to verify if it is within the tolerances. Recognition, finding the best match of a manipulated object to one among a set of known objects, may also be desired. Finally, the estimated location of the center of mass and the orientation of the principal axis can be used to verify that the manipulator has grasped the object in the desired manner.

A key feature of our approach is that it requires no special test or identification movements and therefore can continuously interpret wrist force and torque sensory data during any desired manipulation. Previous methods of load identification were restricted in their application. Paul (1981) described two methods of determining the mass of a load when the manipulator is at rest, one requiring the knowledge of joint torques and the other forces and torques at the wrist. The center of mass and the load moments of inertia were not identified.

Coiffet (1983) utilized joint torque sensing to estimate the mass and center of mass of a load for a robot at rest. Moments of inertia were estimated with special test motions, moving only one axis at a time or applying test torques. Because of the intervening link masses and domination of inertia by the mass moments, joint torque sensing is less accurate than wrist force-torque sensing.

Olsen and Bekey (1985) assumed full force-torque sensing at the wrist to identify the load without special test motions. Mukerjee (1984), and Mukerjee & Ballard (1985) developed an approach similar to ours, again allowing general motion during load identification. Nevertheless, neither paper simulated or experimentally implemented their procedures to verify the correctness of the equations or to determine the accuracy of estimation in the presence of noise and imperfect measurements.

Our algorithm requires measurements of the force and torque due to a load and measurements or estimates of the position, velocity, acceleration, orientation, angu-

lar velocity, and angular acceleration of the force sensing coordinate system. The algorithm can handle incomplete force and torque measurement by simply eliminating the equations containing missing measurements. The necessary kinematic data can be obtained from the joint angles and, if available, the joint velocities of the manipulator. The inertial parameters of a robot hand can be identified using this algorithm and then the predicted forces and torques due to the hand can be subtracted from the sensed forces and torques.

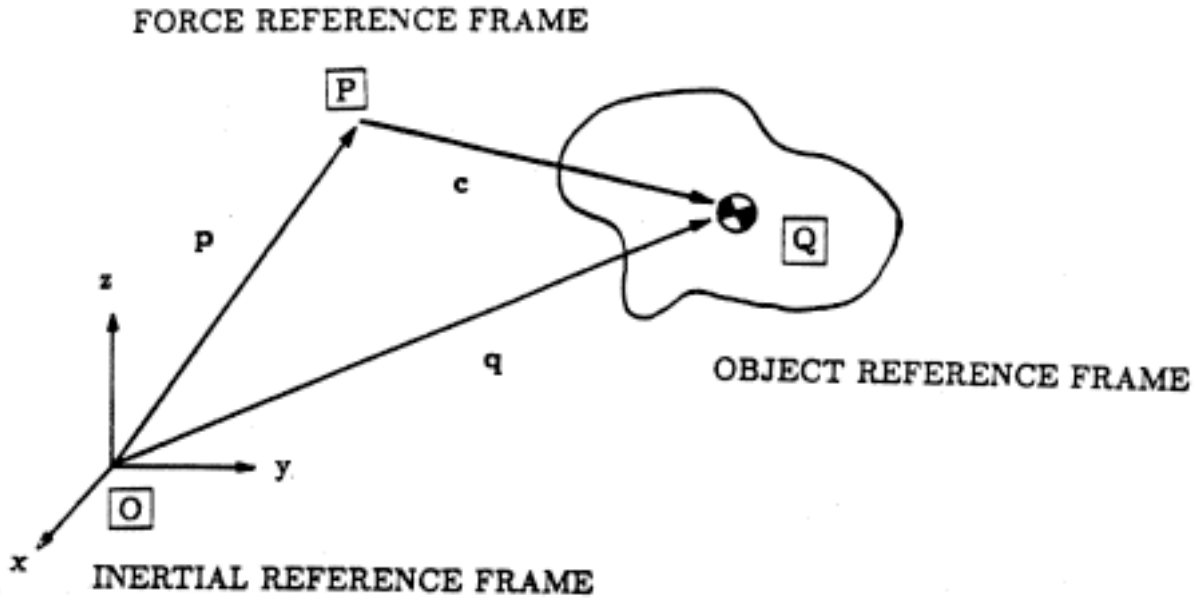
This inertial parameter estimation algorithm was implemented using a PUMA 600 robot equipped with an RTI FS-B wrist force-torque sensor, and on the MIT Serial Link Direct Drive Arm (DDA) equipped with a Barry Wright Company Astek FS6-120A-200 6-axis wrist force-torque sensor.

3.3 The Newton-Euler Approach To The Load Identification Problem

3.3.1 Deriving The Parameter Equation

To derive equations for estimating the unknown inertial parameters, the coordinate systems in Figure 3.1 are used to relate different coordinate frames and vectors. **O** is an inertial or base coordinate system, which is fixed in space with gravity pointing along the $-z$ axis. **P** is the force reference coordinate system of a wrist force-torque sensor rigidly attached to the load. **Q** represents the principal axis of the rigid body load located at the center of mass. The x axis of **Q** is along the largest principal moment of inertia, and the z axis along the smallest. **Q** is unique up to a reflection in bodies with 3 distinct principal moments of inertia. In the derivation that follows all vectors are initially expressed in the base coordinate system **O**.

The mass, moments of inertia, location of the center of mass, and orientation of the body (a rotation ${}_{QP}\mathbf{R}$ from the principal axes to the force reference system) are



p: position vector from the origin of the base coordinate frame to the origin of the wrist sensor coordinate frame.

q: position vector from the origin of the base coordinate frame to the center of the mass of the load.

c: position vector from the origin of the wrist sensor coordinate frame to the center of the mass of the load.

Figure 3.1: Coordinate Frames.

related to the motion of the load and the forces and torques exerted on it by the Newton-Euler equations. The net force ${}_{q}\mathbf{f}$ and the net torque ${}_{q}\mathbf{n}$ acting on the load at the center of mass are:

$$ {}_{q}\mathbf{f} = \mathbf{f} + m\mathbf{g} = m\ddot{\mathbf{q}} \tag{3.1} $$

$$ {}_{q}\mathbf{n} = \mathbf{n} - \mathbf{c} \times \mathbf{f} = {}_{q}\mathbf{I}\dot{\omega} + \omega \times ({}_{q}\mathbf{I}\omega) \tag{3.2} $$

where:

- $\mathbf{f}$ = the force exerted by the wrist sensor on the load at the point $\mathbf{p}$,
- m = the mass of the load,
- $\mathbf{g}$ = the gravity vector ($\mathbf{g} = [0, 0, -9.8\,\text{meters/sec}^2]$),
- $\ddot{\mathbf{q}}$ = the acceleration of the center of mass of the load,
- $\mathbf{n}$ = the torque exerted by the wrist sensor on the load at the point $\mathbf{p}$,
- $\mathbf{c}$ = the unknown location of the center of mass relative to the force sensing wrist origin $\mathbf{P}$,
- ${}_{q}\mathbf{I}$ = the moment of inertia tensor about the center of mass,
- ω = the angular velocity vector, and
- $\dot{\omega}$ = the angular acceleration vector.

We need to express the force and torque measured by the wrist sensor in terms of the product of known geometric parameters and the unknown inertial parameters. Although the location of the center of mass and hence its acceleration $\ddot{\mathbf{q}}$ are unknown, $\ddot{\mathbf{q}}$ is related to the the acceleration of the force reference frame $\ddot{\mathbf{p}}$ by

$$ \ddot{\mathbf{q}} = \ddot{\mathbf{p}} + \dot{\omega} \times \mathbf{c} + \omega \times (\omega \times \mathbf{c}) \qquad (3\,[7.40]^2) $$

Substituting (3) into (3.1),

$$ \mathbf{f} = m\ddot{\mathbf{p}} - m\mathbf{g} + \dot{\omega} \times m\mathbf{c} + \omega \times (\omega \times m\mathbf{c}) \tag{3.4} $$

[2]Equation numbers in brackets refer to equations in Symon, 1971.

Substituting (3.4) into (3.2),

$$\mathbf{n} = {}_{q}\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ({}_{q}\mathbf{I}\boldsymbol{\omega}) + m\mathbf{c} \times (\dot{\boldsymbol{\omega}} \times \mathbf{c}) + m\mathbf{c} \times (\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{c})) + m\mathbf{c} \times \bar{\mathbf{p}} - m\mathbf{c} \times \mathbf{g} \quad (5)$$

Although the terms $\mathbf{c} \times (\dot{\boldsymbol{\omega}} \times \mathbf{c})$ and $\mathbf{c} \times (\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{c}))$ are quadratic in the unknown location of the center of mass $\mathbf{c}$, these quadratic terms are eliminated by expressing the moment of inertia tensor about the force sensor coordinate origin (${}_{p}\mathbf{I}$) instead of about the center of mass (${}_{q}\mathbf{I}$). Rewriting (5) as:

$$\mathbf{n} = {}_{q}\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ({}_{q}\mathbf{I}\boldsymbol{\omega}) + m[(\mathbf{c}^T\mathbf{c})\mathbf{1} - (\mathbf{c}\mathbf{c}^T)]\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (m[(\mathbf{c}^T\mathbf{c})\mathbf{1} - (\mathbf{c}\mathbf{c}^T)]\boldsymbol{\omega}) + m\mathbf{c} \times \bar{\mathbf{p}} - m\mathbf{c} \times \mathbf{g} \quad (6)$$

and using the three dimensional version of the parallel axis theorem

$${}_{p}\mathbf{I} = {}_{q}\mathbf{I} + m[(\mathbf{c}^T\mathbf{c})\mathbf{1} - (\mathbf{c}\mathbf{c}^T)] \qquad (7\ [10.147]).$$

to simplify it results in:

$$\mathbf{n} = {}_{p}\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times ({}_{p}\mathbf{I}\boldsymbol{\omega}) + m\mathbf{c} \times \bar{\mathbf{p}} - m\mathbf{c} \times \mathbf{g} \qquad (3.8)$$

($\mathbf{1}$ is the 3 dimensional identity matrix). We now express all the vectors in the wrist sensor coordinate system $\mathbf{P}$, since then the quantities $\mathbf{c}$ and ${}_{p}\mathbf{I}$ are constant. Moreover, the wrist force-torque sensor measures forces and torques directly in the $\mathbf{P}$ coordinate frame.

In order to formulate the above equations as a system of linear equations, the following notation is used:

$$\boldsymbol{\omega} \times \mathbf{c} = \begin{bmatrix} 0 & -\omega_x & \omega_y \\ \omega_x & 0 & -\omega_z \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \triangleq [\boldsymbol{\omega}\times]\,\mathbf{c} \qquad (3.9)$$

$$\mathbf{I}\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z & 0 & 0 & 0 \\ 0 & \omega_x & 0 & \omega_y & \omega_z & 0 \\ 0 & 0 & \omega_z & 0 & \omega_y & \omega_z \end{bmatrix} \begin{bmatrix} I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \triangleq [\bullet\omega] \begin{bmatrix} I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \tag{3.10}$$

where

$$\mathbf{I} = \mathbf{I}^T = \begin{bmatrix} I_{11} & I_{12} & I_{13} \\ I_{12} & I_{22} & I_{23} \\ I_{13} & I_{23} & I_{33} \end{bmatrix} \tag{3.11}$$

Using these expressions, Eqs. (3.4) and (3.8) can be written as a single matrix equation in the wrist sensor coordinate frame:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} \ddot{\mathbf{p}} - \mathbf{g} & [\dot{\omega}\times] + [\omega\times][\omega\times] & \mathbf{0} \\ \mathbf{0} & [(\mathbf{g} - \ddot{\mathbf{p}})\times] & [\bullet\dot{\omega}] + [\omega\times][\bullet\omega] \end{bmatrix} \begin{bmatrix} m \\ mc_x \\ mc_y \\ mc_z \\ I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \tag{3.12}$$

or more compactly,

$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{\phi} \tag{3.13}$$

where $\boldsymbol{w}$ is a 6 element wrench vector combining both the force and torque vectors, $\boldsymbol{A}$ is a 6×10 matrix, and $\boldsymbol{\phi}$ is the vector of the 10 unknown inertial parameters. Note that the center of mass cannot be determined directly, but only as the mass

moment $m\mathbf{c}$. But since the mass m is separately determined, its contribution can be factored from the mass moment later.

3.3.2 Estimating The Parameters

The quantities inside the $\boldsymbol{A}$ matrix are computed by direct kinematics computation (Luh, Walker, and Paul, 1980b) from the measured joint angles. The elements of the $\boldsymbol{w}$ vector are measured directly by the wrist force sensor. Since (3.13) represents 6 equations and 10 unknowns, at least two data points are necessary to solve for the $\boldsymbol{\phi}$ vector, i.e. the force and the position data sampled at two different configurations of the manipulator. For robust estimates in the presence of noise, we actually need to use a larger number of data points. Each data point adds 6 more equations, while the number of unknowns, the elements of $\boldsymbol{\phi}$, remain constant. $\boldsymbol{w}$ and $\boldsymbol{A}$ can be augmented with n data points:

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{A}[1] \\ \vdots \\ \boldsymbol{A}[n] \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} \boldsymbol{w}[1] \\ \vdots \\ \boldsymbol{w}[n] \end{bmatrix} \tag{3.14}$$

where each $\boldsymbol{A}[i]$ and $\boldsymbol{w}[i]$ are matrix and vector quantities described in (3.12). Formulated this way, any linear estimation algorithm can be used to identify the $\boldsymbol{\phi}$ vector. A simple and popular method is the least squares method. The estimate for $\boldsymbol{\phi}$ is given by:

$$\hat{\boldsymbol{\phi}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{w} \tag{3.15}$$

Equation (3.15) can also be formulated in a recursive form (Ljung and Soderstrom, 1983) for on-line estimation.

3.3.3 Recovering Object And Grip Parameters

The estimated inertial parameters (m, $m\mathbf{c}$, ${}_p\mathbf{I}$) are adequate for control, but for object recognition and verification we also require the principal moments of inertia

I_1, I_2, I_3, the location of the center of mass $\mathbf{c}$, and the orientation ${}_{QP}\mathbf{R}$ of $\mathbf{Q}$ with respect to $\mathbf{P}$.

The parallel axis theorem is used to compute the inertia terms translated to the center of mass of the load.

$$\hat{\mathbf{c}} = \frac{\widehat{m\mathbf{c}}}{\hat{m}} \tag{3.16}$$

$${}_{\mathbf{c}}\hat{\mathbf{I}} = {}_{\mathbf{p}}\hat{\mathbf{I}} - \hat{m}[(\hat{\mathbf{c}}^T\hat{\mathbf{c}})\mathbf{1} - (\hat{\mathbf{c}}\hat{\mathbf{c}}^T)] \tag{3.17}$$

The principal moments are obtained by diagonalizing ${}_{\mathbf{c}}\hat{\mathbf{I}}$.

$${}_{\mathbf{c}}\hat{\mathbf{I}} = {}_{QP}\hat{\mathbf{R}} \begin{bmatrix} \hat{I}_1 & 0 & 0 \\ 0 & \hat{I}_2 & 0 \\ 0 & 0 & \hat{I}_3 \end{bmatrix} {}_{QP}\hat{\mathbf{R}}^T \tag{3.18}$$

This diagonalization can always be achieved since ${}_{\mathbf{c}}\hat{\mathbf{I}}$ is symmetric, but when two or more principal moments are equal the rotation matrix, ${}_{QP}\mathbf{R}$, is no longer unique.

3.4 Experimental Results

3.4.1 Estimation on the PUMA Robot

The inertial parameter estimation algorithm was originally implemented on a PUMA 600 robot equipped with an RTI FS-B wrist force-torque sensor (Figure 3.2), which measures all six forces and torques. The PUMA 600 has encoders at each joint to measure joint angles, but no tachometers. Thus, to obtain the joint velocities and accelerations, the joint angles are differentiated and double-differentiated, respectively, by a digital differentiating filter (Figure 3.3). The cutoff frequency of 33 Hz for the filter was determined empirically to produce the best results. Both the encoder data and the wrist sensor data were initially sampled at 1000 Hz. It was later determined that a sampling rate of 200 Hz was sufficient, and the data were resampled at the lower rate to reduce processing time. A least squares identification algorithm was

Figure 3.2: Puma with a test load.

implemented as an off-line computation, but an on-line implementation would have been straightforward.

Static Estimation Using The PUMA

To test the calibration of the force sensor and the kinematics of the PUMA arm a static identification was performed. The forces and torques are now due only to the gravity acting on the load, and equations (3.4) and (3.8) simplify to

$$\mathbf{f} = -m\mathbf{g} \tag{3.19}$$

$$\mathbf{n} = -m\mathbf{c} \times \mathbf{g} \tag{3.20}$$

As seen in (3.19) and (3.20), only the mass and the center of mass can be identified while the manipulator is stationary.

To avoid needing to determine the gripper geometric parameters, the center of mass estimates are evaluated by the estimates of the changes in the center of

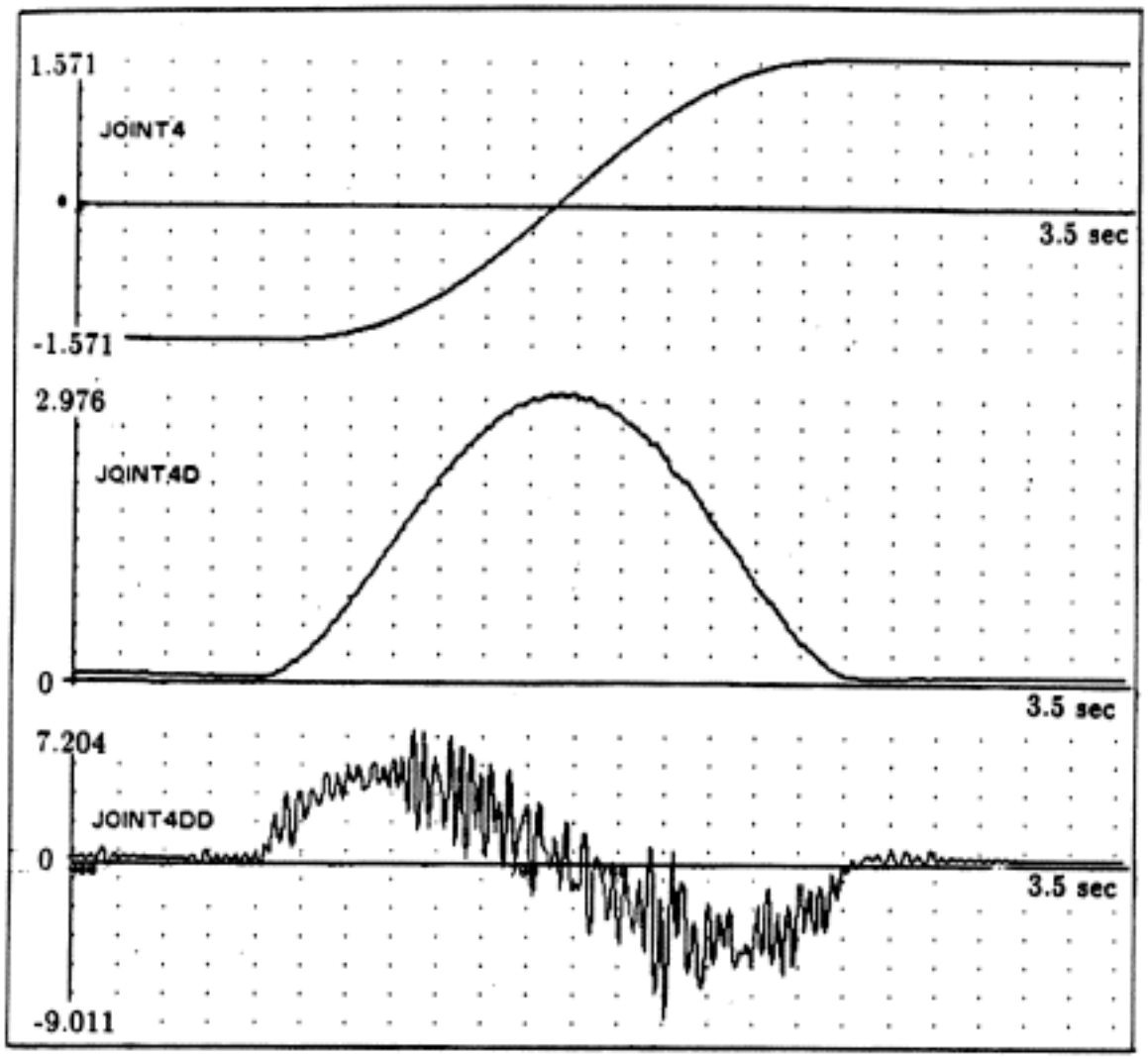


Figure 3.3: Measured angle θ, calculated angular velocity $\dot{\theta}$, and calculated angular acceleration $\ddot{\theta}$ for joint 4.

Parameters	*Actual Values*	*Static Estimates*	*Dynamic Estimates*
Mass (kg)	1.106	1.103	1.067
Change in $c_y(m)$	0.037	0.037	0.039
Mass (kg)	1.106	1.107	1.084
Change in $c_y(m)$	-0.043	-0.043	-0.042
Mass (kg)	1.106	1.100	1.073
Change in $c_y(m)$	-0.021	-0.020	-0.021
Mass (kg)	1.106	1.099	1.074
Change in $c_y(m)$	0.018	0.018	0.020

Table 3.1: Mass and the center of mass estimates

mass as the load is moved along the y-axis from the reference position by known amounts. The results of estimation are shown in the second column of Table 3.1 for an aluminum block ($2 \times 2 \times 6in.$) with a mass of $1.106Kg$. Only the changes in c_y are shown in Table 3.1; the estimates of c_x and c_z remained within $1mm$ of the reference values ($c_x = 1mm$ and $c_z = 47mm$). Each set of estimates were computed from 6 sets of data, i.e. data taken at 6 different positions and orientations of the manipulator, where each data point is averaged over 1000 samples to minimize the effects of noise. The results show that in the static case the mass of the load can be estimated to within $10g$ of the actual mass. The center of mass can be estimated to within $1mm$ of the actual values for this load.

Static load estimation only tests the force sensor calibration and the position measurement capabilities of the robot the sensor is mounted on. In order to assess the effects of the dynamic capabilities of the robot on load estimation and to be able to estimate the moments of inertia of the load we must assess parameter estimation

during general movement.

Dynamic Estimation Using The PUMA

In the dynamic case, the joint position encoder and the wrist sensor data are sampled while the manipulator is in motion. A fifth order polynomial trajectory (a minimum-jerk time function) in joint space was used to minimize the mechanical vibrations at the beginning and the end of the movement, and to improve the signal to noise ratio (SNR) in the acceleration data (Figure 3.3). For more popular bang-coast-bang type trajectories, the joint accelerations are zero except at the beginning and the end of the movements, resulting in a poor signal to noise ratio in the acceleration data for most of the movement.

We found that the PUMA robot lacked the acceleration capacity necessary to estimate the moments of inertia of the load. It also lacked true velocity sensors at the joints, which made estimation of the acceleration of the load difficult. The dynamic estimates of mass and center of mass for the previous load are shown in the last column of Table 3.1. The data used in these estimates were sampled while the manipulator was moving from $[0, 0, 0, -90, 0, 0]$ to $[90, -60, 90, 90, 90, 90]$ degrees on a straight line in joint space in 2 seconds. Joint 4 of the PUMA has a higher maximum acceleration than the other joints, and thus, a longer path was given for it. This movement was the fastest the PUMA can execute using the fifth order trajectory without reaching the maximum acceleration for any of its joints. The estimates used all 400 data points sampled during the 2 second movement. The results show slight deterioration in these estimates when compared to the static estimates; but they are still within $40g$ and $2mm$ of the actual mass and displacement, respectively. However, the signal to noise ratios in the acceleration and the force-torque data were too low for accurate estimates of the moments of inertia for this load ($0.00238 Kg \cdot m^2$ in the largest principal moment). In this case, the torque due to gravity is approximately 40 times greater than the torque due to the maximum angular acceleration

Parameters $(kg \cdot m^2)$	*Actual Values*	PUMA[1] *Estimates*	PUMA[2] *Estimates*	DDA[1] *Estimates*
I_{11}	0.0244	0.0192	0.0246	0.0230
I_{12}	0	-0.0048	0.0006	0.0006
I_{13}	0	0.0019	0.0008	0.0005
I_{22}	0.0007	0.0021	0.0036	-0.0002
I_{23}	0	-0.0016	-0.0004	-0.0002
I_{33}	0.0242	0.0176	0.0199	0.0241

Table 3.2: Actual and estimated moments of inertia, either for all joints moving[1] or special test motions[2].

of the load. Thus, even slight noise in the data would result in poor estimates of **I**.

Therefore, experiments with a larger rotational load were performed for the estimates of the moments of inertia. The new experimental load is shown in Figure 3.2. This load has large masses at the two ends of the aluminum bar, resulting in large moments of inertia in two directions ($\sim 0.024 kg \cdot m^2$) and a small moment in the other. A typical set of estimates of the moments of inertia at the center of mass frame for the load with the gripper subtracted out are shown in Table 3.2 for the above all-joints-moving trajectory. They contain some errors but are fairly close to the actual values.

Special Test Movements Using The PUMA

In order to improve the estimates by maximizing the rotational accelerations in the trajectories, a series of special test movements were generated. The data was sampled while the robot was following three separate 2-second rotational trajectories around the principal axes of the load. Such trajectories used joints 4 and 6 only, and resulted

in higher acceleration data than the previous trajectory, thus improving the signal to noise ratio in both the acceleration and the force-torque data. Typical estimates for these special movements show improvements over the estimates with the previous trajectory (Table 3.2). Although the estimate of I_{22} is slightly worse than before, all the other terms have improved; the cross terms, especially, are much smaller than before. However, these estimates of **I** are not as accurate as the estimates of the mass and the center of mass shown in Table 3.1.

Figure 3.4 shows the comparison of the measured forces and torques, and the computed forces and torques from the estimated parameters and the measured joint data using the simulator for the original trajectory. The two sets of figures match very well even in the mechanical vibrations, verifying qualitatively the accuracy of the estimates. This suggests that for control purposes even poor estimation of moment of inertia parameters will allow good estimates of the total force and torque necessary to achieve a trajectory. This makes good sense in that the load forces with the PUMA are dominated by gravitational components, and angular accelerations experienced by the load are small relative to those components.

The effect of the errors causing poor estimates of moment of inertia parameters could be alleviated by increasing the angular acceleration experienced by the load. Since we had reached the sustained acceleration capacity of even an unloaded PUMA robot, we decided to explore this issue using the MIT Serial Link Direct Drive Arm. This robot can achieve higher sustained accelerations than the PUMA and in addition is also equipped with tachometers at the joints, making estimation of acceleration much easier.

3.4.2 The MIT Serial Link Direct Drive Arm

The inertial parameter estimation algorithm was next implemented on the MIT Serial Link Direct Drive Arm (DDA), equipped with a Barry Wright Company

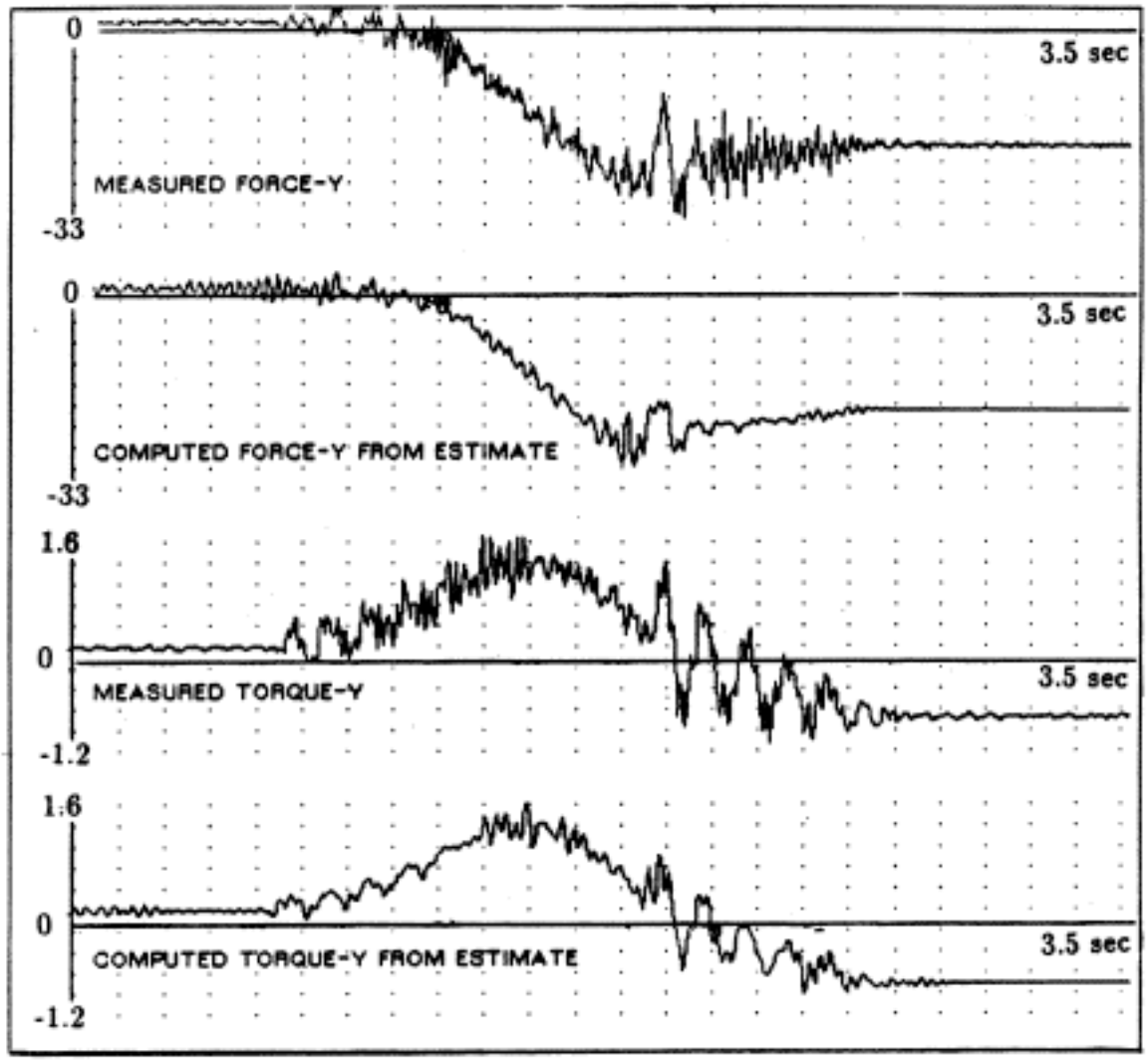


Figure 3.4: Measured force-torque data and computed force-torque data from the estimates using the PUMA.

Astek FS6-120A-200 6-axis force-torque sensor which again measures all three forces and three torques about a point. The DDA is described in An, Atkeson, Hollerbach (1985), and is capable of higher tip velocities and accelerations than the PUMA. The DDA has tachometers on each of its three joints so that numerical differentiation of positions is unnecessary, but we still had to digitally differentiate the velocities to find the accelerations using a cutoff frequency of 30Hz. The positions and velocities of the robot were initially sampled at 1kHz but were later down-sampled to match the sampling frequency of the force-torque sensor of 240 Hz. The identification procedure was again implemented off-line.

Dynamic Estimation Using The Direct Drive Arm

The data used for estimating the inertial parameters of the load were sampled while the manipulator was moving from $(280, 269.1, -30)$ to $(80, 19.1, 220)$ in one second. Again a fifth order polynomial straight line trajectory in joint space was used. The resulting estimates for the moment of inertia parameters are shown in the last column of Table 3.2. The estimates for the mass and the location of the center of mass were as good as the PUMA results and are not shown. We see that the moment of inertia parameters estimated are on the whole better than the PUMA results.

Parameters estimated for a set of special test movements using the Direct Drive Arm were not substantially different. Our special test movements for the DDA were not substantially faster than the movement of all joints, and thus probably contained the same amount of information.

Finally, Figure 3.5 shows the comparison of typical measured forces and torques with computed forces and torques from the estimated parameters and the measured joint data using the simulator for the original trajectory. Once again we have a very good match between the measured and predicted forces and torques. Thus we see that the combination of higher angular accelerations and good velocity sensing results in better parameter estimates, as hoped.

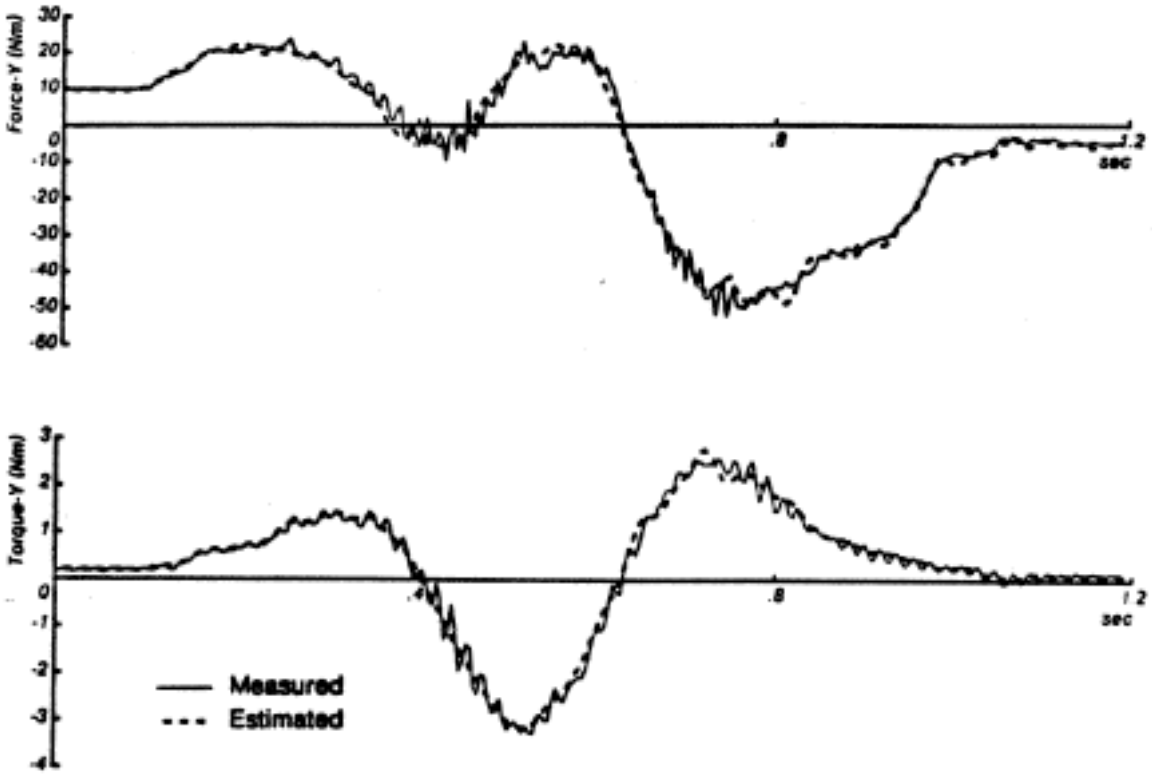


Figure 3.5: Measured force-torque data and computed force-torque data from the estimates using the Direct Drive Arm.

3.5 Discussion

3.5.1 Did The Algorithm Work?

This chapter describes an attempt to characterize the usefulness of wrist force-torque sensing for estimating the inertial parameters of rigid body loads for control and recognition/verification/grasping. Our conclusion is that prediction of forces for control can be good and seems to work well in our implementations. Identifying parameters well enough for recognition of the object may require large accelerations or special test movements in order to accurately identify the moment of inertia parameters.

It is important to realize that there are two distinct uses of an identified model. For control what matters is matching the input-output behavior of the model (in this case the relationship of load trajectory to load forces and torques) to reality, while for recognition/verification what matters is matching estimated parameters to a set of parameters postulated for reality. We find that both of our implementations of load inertial parameter estimation successfully match the input output behavior of the load (see Figures 3.4 and 3.5), although we have not yet used this information in a control scheme. However, we find that the limited acceleration capacity of the PUMA robot and its limited sensing do not permit us to estimate the moments of inertia of the load accurately without the use of special test motions. In all cases the mass and the location of the center of mass could be accurately estimated from both a series of static measurements, and measurements taken during movements.

This work is preliminary in that an adequate statistical characterization of the errors of the estimated parameters or the predicted forces has not been attempted. Nevertheless, we have gained insight into the sources of such errors.

3.5.2 Sources of Error

The ultimate source of error is the random noise inherent in the sensing process itself. The noise levels on the position and velocity sensing are probably negligible, and could be further reduced by appropriate filtering using a model based filter such as the Extended Kalman Filter (Gelb, 1974). The force and torque measurement process involve measuring the strains of structure members in the sensor with strain gages. The random noise involved in such measurements is also probably negligible.

Bias Errors

However, strain gages are notoriously prone to drift. We feel that periodic recalibration of the offsets (very often) and the strain to force calibration matrix (often) may be necessary to reduce load parameter estimation errors further. Before using the force sensors we allowed the system to warm up and we recalibrated the offsets before each data collection session and checked for a change in the calibrated offsets afterward.

Unmodelled Dynamics

A further source of noise is unmodelled structural dynamics. Neither the robot links nor the load itself are perfectly rigid bodies. A greater source of concern is the compliance of the force sensor itself. In order to generate structural strains large enough to be reliably measured with strain gages, a good deal of compliance is introduced into the force sensor. The load rigidly attached to the force sensor becomes a relatively undamped spring mass system. The response of the Astek force sensor to a tap on an attached load is shown in the "undamped impulse response" record of Figure 3.6.

The effect of robot movement on this spring mass system is shown in the "undamped movement response" record.

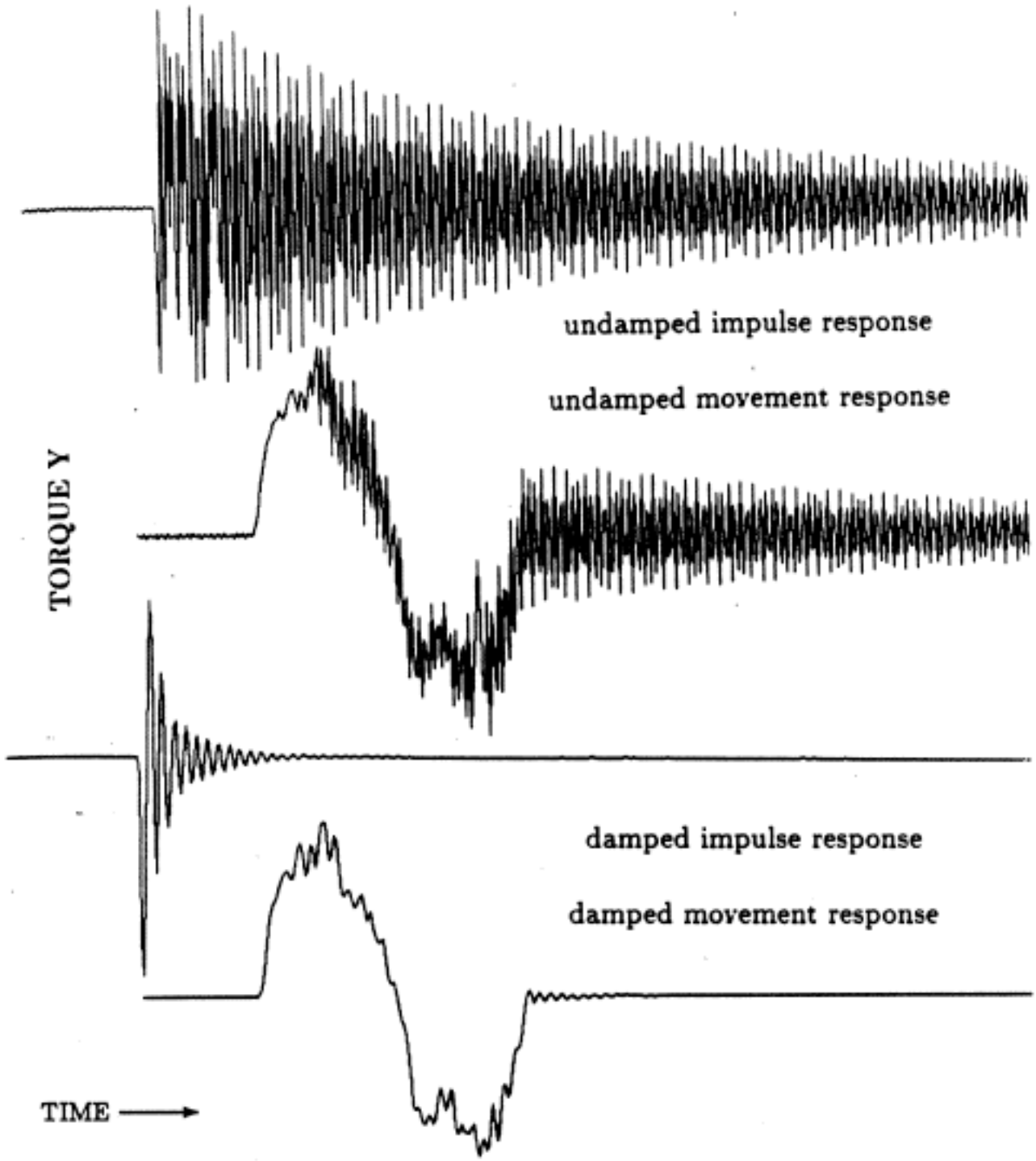


Figure 3.6: Vibrations of load on force sensor.

There are several approaches to deal with this problem of unmodelled dynamics. One approach is to attempt to identify the additional dynamics. This greatly increases the complexity of the identification process, and greatly increases the amount of data that needs to be collected to get reliable estimates of any parameter. We feel such an approach should only be taken as a last resort.

Another approach is to try to avoid exciting the unmodelled dynamics by choosing robot trajectories that are as smooth as possible. This is one reason why we chose 5th order polynomial trajectories, so that we could maintain continuity of velocities and accelerations. Using higher order polynomials would result in even greater smoothness. However, we found that with the PUMA a smooth commanded trajectory did not result in a smooth actual trajectory, in that the control methods used and the actual hardware of the robot still introduced substantial vibration. One way to tell if the PUMA is turned on is to touch it and feel if it is vibrating. Vibrations were less of a problem with the Direct Drive Arm, although still present.

The most successful approach is to mechanically damp out the vibrations by introducing some form of energy dissipation into the structure. We added hard rubber washers between the force sensor and the load, and the "damped impulse response" of Figure 3.6 illustrates the response of the force sensor to a tap on the load. We see that the oscillations decay much faster. The "damped movement response" indicates that this mechanical damping greatly reduces the effect of movement on the resonant modes of the force sensor plus load. The conclusion we draw is that appropriate damping should be built into force sensors, just as accelerometers are filled with oil to provide a critically damped response for a specified measurement bandwidth. Failing that, energy dissipation must be introduced either into the structural components of the robot or into the gripper either structurally or as a viscous skin. Appropriate mechanical damping may also be useful when using a force sensor in closed loop force control.

Optimal Filtering

The need to numerically differentiate the velocity to find the acceleration, or worse yet, double differentiate positions, greatly amplifies whatever noise is present. One can avoid the need to explicitly calculate accelerations by symbolically integrating equations (3.4) and (3.8). One approach to integrating the equations is presented in the appendix, and we can express the resulting estimation equations in matrix form as:

$$\begin{bmatrix} \int_t^{t+T} \mathbf{f} \\ \int_t^{t+T} \mathbf{n} \end{bmatrix} = \left[\int_t^{t+T} \mathbf{A}\, d\tau\right] \begin{bmatrix} m \\ mc_z \\ mc_y \\ mc_x \\ I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \tag{3.21}$$

where the first row of $\left[\int_t^{t+T} \mathbf{A}\, d\tau\right]$ is

$$\left[\dot{\mathbf{p}}\Big|_t^{t+T} + \int_t^{t+T} \boldsymbol{\omega} \times \dot{\mathbf{p}}\, d\tau - \left(\int_t^{t+T} \mathbf{R}\, d\tau\right) {}^{\circ}\mathbf{g} \right. \quad \left. \left[\left(\boldsymbol{\omega}\Big|_t^{t+T}\right)\times\right] + \int_t^{t+T} [\boldsymbol{\omega}\times][\boldsymbol{\omega}\times]\, d\tau \qquad \mathbf{0}\right] \tag{22}$$

and the second row is

$$\left[\mathbf{0} \qquad \left[\left(-\dot{\mathbf{p}}\Big|_t^{t+T} - \int_t^{t+T} \boldsymbol{\omega} \times \dot{\mathbf{p}}\, d\tau + \left(\int_t^{t+T} \mathbf{R}\, d\tau\right) {}^{\circ}\mathbf{g}\right)\times\right] \right. \quad \left. \left[\bullet\left(\boldsymbol{\omega}\Big|_t^{t+T}\right)\right] + \int_t^{t+T} [\boldsymbol{\omega}\times][\bullet\boldsymbol{\omega}]\, d\tau\right] \tag{23}$$

However, we feel that an effort to characterize the various noise sources and an attempt to "optimally" filter and differentiate/integrate the data will result in better estimates. If there is substantial low frequency noise or bias in the data the integration will amplify that noise relative to the signal frequencies of the data. What is needed is a problem formulation that gives us guidance as to how to design data processing filters for estimation and how to handle the need to differentiate or integrate some of the data to supply missing measurements. We are presently investigating such an approach.

3.5.3 Kinematic Errors

Part of the error may be due to inaccuracies in the current kinematic parameters of the manipulator. Experiments have shown that the actual orientation of the robot can be up to 4° off from the orientation computed from the encoder data.

3.5.4 Why Estimating Moment of Inertia Parameters is Hard

One of the factors that makes it difficult to accurately identify moments of inertia is the typically large contribution of gravitational torque, which depends only on the mass and the relative location of the center of mass to the force sensing coordinate origin. A point mass rotated at a radius of $5cm$ from a horizontal axis must complete a full 360° rotation in 425 milliseconds for the torque due to angular acceleration to be equal to the gravitational torque, if a 5th order polynomial trajectory is used. A way to avoid gravitational torques is to rotate the load about a vertical axis, or to have the point of force-torque sensing close to the center of mass.

A simple example will illustrate the difficulty of recovering principle moments of inertia, given the moment of inertia tensor about the force sensing origin. The principle moment of inertia of a uniform sphere is only 2/7 of the total moment of inertia when it is rotated about an axis tangent to its surface, so that the effects of

any errors in estimating the mass, the location of the center of mass, and the grip moments of inertia are amplified when the principle moment of inertia is calculated. This problem can be reduced by having the point of force sensing as close to the center of mass as possible.

It still may be difficult to find the orientation of the principle moments of inertia even when the moment of inertia tensor about the center of mass has been estimated fairly accurately. This occurs when two or more principle moments of inertia are approximately equal. Finding the orientation of the principle axis is equivalent to diagonalizing a symmetric matrix, which becomes ill-conditioned when some of the eigenvalues are almost equal. A two dimensional example illustrates the problem: Consider the diagonalized matrix

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.24}$$

with eigenvalues $\{\lambda_1, \lambda_2\}$ and whose first principle axis is oriented at an angle θ with respect to the x axis. By substituting $\lambda_1 - \lambda_2 = \epsilon$ into the matrix (3.24),

$$\begin{bmatrix} \lambda_2 + \epsilon\cos^2(\theta) & \epsilon\cos(\theta)\sin(\theta) \\ \epsilon\cos(\theta)\sin(\theta) & \lambda_2 + \epsilon\sin^2(\theta) \end{bmatrix} \tag{3.25}$$

we see that when the two eigenvalues are almost equal, the terms of the matrix dependent on the angle, θ, become very small. All terms that contain angle information are multiplied by the difference of the principle moments of inertia, ϵ. With a fixed amount of noise in each of the entries of the identified moment of inertia matrix, the orientation of the principle axis, θ, will become more and more difficult to recover as the principle moments of inertia approach equality and therefore ϵ approaches zero.

3.6 Open Questions

3.6.1 Data Processing Issues

Many of the issues discussed below will be addressed in conjuction with the continuation of optimal filtering research described in Chapter 5.

Statistical Characterization of Errors

What is the variance of our force and torque predictions? Does this level of error correspond to the variability of force and torque measurements? What is the variance of the estimated parameters, for a given amount of data? What measurement noise does this correspond to? How close are the estimated measurement noise to the specifications for the force sensor? Linear least squares estimation is based on a particular statistical model which includes formulas for estimated parameter variance and predictor variances. However, it is not clear that the assumed statistical model is valid in this application.

Most Efficient Movement

Find the most efficient movement (or just a good movement) to improve the load parameter estimates given what is already known about the load and which estimates are to be improved. This is part of a general problem in quantifying the "richness" of the input. The statistical model assumed for linear least squares estimation provides a good starting point for answering this question.

Quantify desirable sensor properties

What is the stiffness of the wrist sensor? The Aztec wrist force-torque sensor has an internal samping rate of 480Hz., a 120Hz. analog low-pass filter, and a 120 Hz

digital low-pass filter also, for an output sampling rate of 240Hz. What are more desirable sensor filtering properties?

Role of Velocity Sensing

We claim that tachometers helped with the Asada Arm. We could repeat the load identification without using the tachometer measurements, and see how much difference the tachometers made.

Wrist sensing vs. joint sensing

We could compare the load estimation with a wrist sensor with load estimation using joint sensors, and see if there is a substantial difference in accuracy.

Separate Control and Recognition Modules

Since the goals of identification for control (predict input/output behavior) and recognition (get good object parameter estimates) are different there should probably be different modules with different filtering and data handling properties.

Effect of kinematic errors

How robust is the load parameter estimation procedure to kinematic errors caused by joint measurement error and unmeasured link deflection?

3.6.2 A Complete Load Handling System

Use rigid body load inertial parameter estimation as a single module in a more complete load identification system. Handle other model structures: What are useful structures? What are useful parameter estimation procedures? Provide a test or measure for verifying that a load is actually a rigid body. What does this mean? More a rigid body than anything else?

Note that the prediction errors (residuals) of the estimated model provide a natural measure for whether the body is rigid or not.

3.6.3 Real–Time Implementation

Implement load parameter estimation as on-line estimation procedure. It is appropriate to address here the use of apriori load estimates.

3.7 Appendix: Integrating The Forces And Torques In The Force Sensor Frame

In this appendix we will show how to integrate the load identification equations (3.4) and (3.8). Throughout this derivation we will use the superscript notation ^{o} and ^{p} to indicate what coordinate frame a vector is expressed in, if there is any confusion. If there is no such superscript, the vector is expressed in the load frame.

The term $\ddot{\mathbf{p}}$ is integrated using equation 7.22 of (Symon, 1971):

$$\mathbf{R}\frac{d}{dt}({}^{o}\dot{\mathbf{p}}) = \frac{d}{dt}(\mathbf{R}\cdot{}^{o}\dot{\mathbf{p}}) + \mathbf{R}({}^{o}\boldsymbol{\omega}\times{}^{o}\dot{\mathbf{p}}) \tag{3.26}$$

were $\mathbf{R}$ is the rotation matrix representing the rotation from an inertial coordinate system to the force sensing coordinate system that continuously moves with the load. To get an integral form of this equation we simply integrate it:

$$\int_{t}^{t+T}\mathbf{R}\cdot{}^{o}\ddot{\mathbf{p}}\,d\tau = (\mathbf{R}\cdot{}^{o}\dot{\mathbf{p}})\Big|_{t}^{t+T} + \int_{t}^{t+T}\mathbf{R}({}^{o}\boldsymbol{\omega}\times{}^{o}\dot{\mathbf{p}})\,d\tau \tag{3.27}$$

and performing the indicated rotations leaves us with

$$\int_{t}^{t+T}{}^{p}\ddot{\mathbf{p}}\,d\tau = {}^{p}\dot{\mathbf{p}}\Big|_{t}^{t+T} + \int_{t}^{t+T}{}^{p}\boldsymbol{\omega}\times{}^{p}\dot{\mathbf{p}}\,d\tau \tag{3.28}$$

Similarly

$$\int_{t}^{t+T}{}^{p}\dot{\boldsymbol{\omega}}\,d\tau = {}^{p}\boldsymbol{\omega}\Big|_{t}^{t+T} + \int_{t}^{t+T}{}^{p}\boldsymbol{\omega}\times{}^{p}\boldsymbol{\omega}\,d\tau = {}^{p}\boldsymbol{\omega}\Big|_{t}^{t+T} \tag{3.29}$$

since $\boldsymbol{\omega} \times \boldsymbol{\omega} = 0$. We also remember that

$$\int_t^{t+T} {}^p\mathbf{g}\, d\tau = \left(\int_t^{t+T} \mathbf{R}\, d\tau \right) {}^o\mathbf{g} \tag{3.30}$$

Each of the matrices $[{}^p\dot{\boldsymbol{\omega}}\times]$ and $[{}^\bullet\dot{\boldsymbol{\omega}}]$ can be integrated element by element to show that

$$\int_t^{t+T} [{}^p\dot{\omega}\times]\, d\tau = \left[\left(\int_t^{t+T} {}^p\dot{\omega}\, d\tau \right) \times \right] = \left[\left({}^p\omega \Big|_t^{t+T} \right) \times \right] \tag{3.31}$$

$$\int_t^{t+T} [{}^\bullet{}^p\dot{\omega}]\, d\tau = \left[\bullet \left(\int_t^{t+T} {}^p\dot{\omega}\, d\tau \right) \right] = \left[\bullet \left({}^p\omega \Big|_t^{t+T} \right) \right] \tag{3.32}$$

The matrix $[(\mathbf{g} - \ddot{\mathbf{p}})\times]$ can be integrated element by element in the same way. Each matrix element of the terms $[{}^p\boldsymbol{\omega}\times][{}^p\boldsymbol{\omega}\times]$ and $[{}^p\boldsymbol{\omega}\times][{}^\bullet{}^p\boldsymbol{\omega}]$ are numerically integrated by adding values at each time step. The result of this integration is given in Equation (3.21).

Chapter 4

Manipulator Link Inertial Parameter Estimation

4.1 Abstract

A method of estimating the mass, the location of center of mass, and the moments of inertia of each rigid body link of a robot during general manipulator movement is presented. The algorithm is derived from the Newton-Euler equations, and uses measurements of the joint torques as well as the measurement and calculation of the kinematics of the manipulator while it is moving. The identification equations are linear in the desired unknown parameters, and a modified least squares algorithm is used to obtain estimates of these parameters. Some of the parameters, however, are not identifiable due to the restricted motion of proximal links and the lack of full force/torque sensing. The algorithm was implemented on the MIT Serial Link Direct Drive Arm. A good match was obtained between joint torques predicted from the estimated parameters and the joint torques computed from motor currents.

[1]This chapter is a revised version of (An, Atkeson, and Hollerbach, 1985a)

4.2 Introduction

This chapter presents a method of estimating all of the inertial parameters, the mass, the center of mass, and the moments of inertia of each rigid body link of a robot manipulator using joint torque sensing. Determining these parameters from measurements or computer models is generally difficult and involves some approximations to handle the complex shapes of the arm components. Typically, even the manufacturers of manipulators do not know accurate values of these parameters.

The degree of uncertainty in inertial parameters is an important factor in judging the robustness of model-based control strategies. A common objection to the computed torque methods, which involve full dynamics computation (e.g., Luh, Walker, and Paul, 1980b), is their sensitivity to modelling errors, and a variety of alternative robust controllers have been suggested (Samson, 1983; Slotine, 1985; Spong, Thorp, and Kleinwaks; 1984, Gilbert and Ha, 1984). Typically these robust controllers express modelling errors as a differential inertia matrix and coriolis and gravity vectors, but in so doing, no rational basis is provided for the source of errors or the bounds on errors. The error matrices and vectors combine kinematic and dynamic parameter errors, but kinematic calibration is sufficiently developed so that very little error can be expected in the kinematic parameters (Whitney, Lozinski, and Rourke, 1984).

One aim of this work is to place similar bounds on inertial parameter errors by explicitly identifying the inertial parameters of each link that go into the making of the inertia matrix and coriolis and gravity vectors. Our work in load identification (Atkeson, An, and Hollerbach, 1985a) suggests, for example, that mass can be accurately identified to within 1%. Therefore, an assumption of 50% error in link mass in verifying a robust control formulation (Spong, Thorp, and Kleinwaks, 1984) is an unreasonable basis for argument. Slotine (1985) suggests that errors of only a few percent in inertial parameters make his robust controller superior to the computed torque method, but it may well be that these parameters can be identified more

accurately than his assumptions.

As an alternative approach we propose estimating the inertial parameters on the basis of direct dynamic measurements. The same algorithms used to identify load inertial parameters (Atkeson, An, and Hollerbach, 1985a) can be modified to find link inertial parameters of a robot arm made up of rigid parts. The Newton-Euler dynamic equations are used to express the measured forces and torques at each joint in terms of the product of the measured movements of the rigid body links and the unknown link inertial parameters. These equations are linear in the inertial parameters. However, unlike load estimation, the only sensing is one component of joint torque, inferred from motor current. Coupled with restricted movement near the base, it is, therefore, not possible to find all the inertial parameters of the proximal links. As will be seen, these missing parameters have no effect on the control of the arm.

In this chapter, manipulators with only revolute joints are discussed since handling prismatic joints requires only trivial modifications to the algorithm. The proposed algorithm was verified by implementation on the MIT Serial Link Direct Drive Arm.

4.2.1 Previous Work

Mayeda, Osuka, and Kangawa (1984) required three sets of special test motions to estimate the coefficients of a closed-form Lagrangian dynamics formulation. The 10 inertial parameters of each link are lumped into these numerous coefficients, which are redundant and susceptible to numerical problems in estimation. On the other hand, every coefficient is identifiable since these coefficients represent the actual degrees of freedom of the robot. By sensing torque from only one joint at a time, their algorithm is more susceptible to noise from transmission of dynamic effects of distant links to the proximal measuring joints. For efficient dynamics computation,

the recursive dynamics algorithms require the link parameters explicitly. Though recoverable from the Lagrangian coefficients, it is better to estimate the inertial parameters directly. Though this algorithm was implemented on a PUMA robot, it is difficult to interpret the results because of dominance of the dynamics by the rotor inertia and friction.

Mukerjee and Ballard (1985) directly applied their load identification method to link identification, by requiring full force-torque sensing at each joint. Instrumenting each robot link with full force-torque sensing seems impractical, and is actually unnecessary given joint torque sensing about the rotation axis. Partially as a result, he does not address the issue of unidentifiability of some inertial parameters. Also, he did not verify his algorithm by simulation or by implementation.

Olsen and Bekey (1985) presented a link identification procedure using joint torque sensing and special test motions with single joints. The unidentifiability of certain inertial parameters was not resolved, and the least squares estimation procedure written as a generalized inverse would fail because of linear dependence of some of the inertial parameters. Again, their procedure was not tested by simulation or by actual implementation on a robot arm.

Neuman and Khosla (1985) developed a hybrid estimation procedure combining a Newton-Euler and a Lagrange-Euler formulation of dynamics. Simulation results for a three degree-of-freedom cylindrical robot were presented, and the unidentifiability of certain inertial components was addressed. For some reason they state link mass must be known for a linear estimation procedure, but such a restriction does not exist with our method. Though planning to work with the CMU DDArm II, they have not yet presented experimental results.

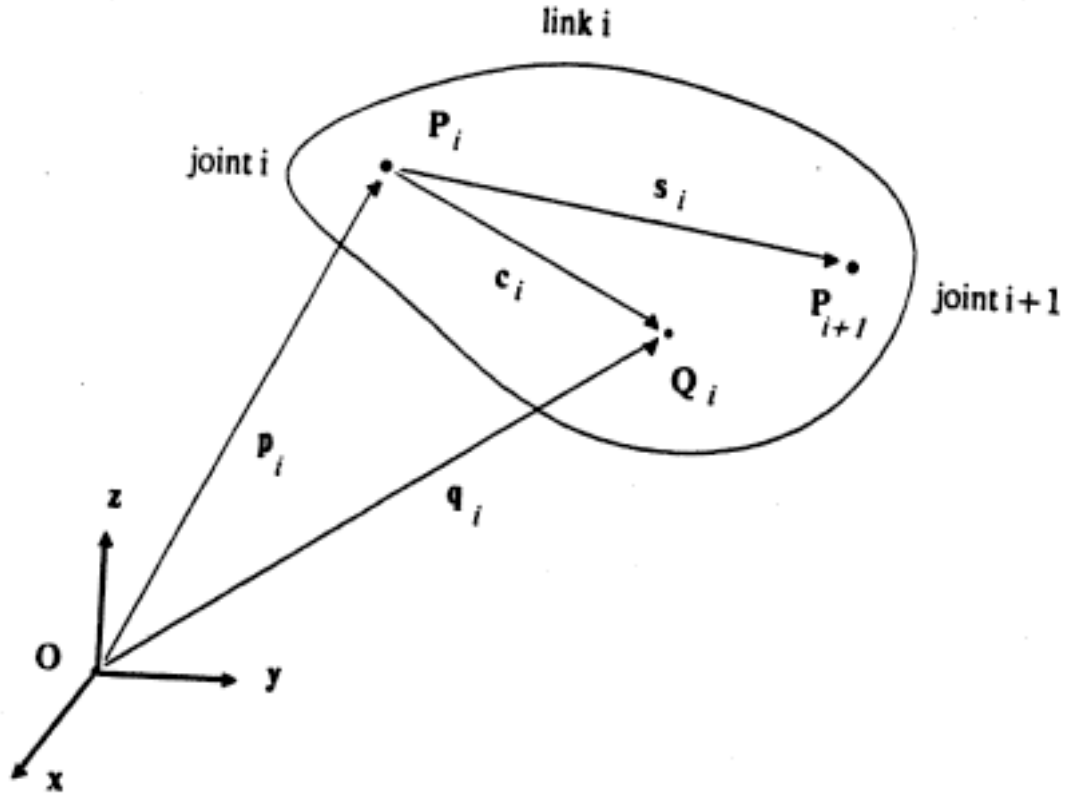


Figure 4.1: Coordinate origins and location vectors for link identification.

4.3 Estimation Procedure

4.3.1 Formulation of Newton-Euler Equations

In our work in load identification (Atkeson, An, and Hollerbach, 1985a), the Newton-Euler equations for a rigid body load were formulated to be linear in the unknown inertial parameters. Then the simple linear least squares method was used to estimate those parameters. By treating each link of a manipulator as a load, this formulation can be extended to the link estimation problem. The differences in the equations are that only one component of force or torque is sensed and that the forces and torques from distal links are summed and transmitted to the proximal joints.

Consider a manipulator with n joints (Figure 4.1). Each link i has its own local

coordinate system $\mathbf{P}_i$ fixed in the link with its origin at joint i. The joint force and torque due to the movement of its own link can be expressed by simply treating the link as a load and applying the previously developed equations for load identification (Atkeson, An, and Hollerbach, 1985a):

$$\begin{bmatrix} \mathbf{f}_{ii} \\ \mathbf{n}_{ii} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{p}}_i - \mathbf{g} & [\dot{\omega}_i\times] + [\omega_i\times][\omega_i\times] & \mathbf{0} \\ \mathbf{0} & [(\mathbf{g} - \tilde{\mathbf{p}}_i)\times] & [\bullet\dot{\omega}_i] + [\omega_i\times][\bullet\omega_i] \end{bmatrix} \begin{bmatrix} m_i \\ m_i c_{x_i} \\ m_i c_{y_i} \\ m_i c_{z_i} \\ I_{xx_i} \\ I_{xy_i} \\ I_{xz_i} \\ I_{yy_i} \\ I_{yz_i} \\ I_{zz_i} \end{bmatrix}$$

or more compactly,

$$\mathrm{w}_{ii} = \mathbf{A}_i \boldsymbol{\phi}_i \tag{4.1}$$

where w_{ij} is the wrench (vector of forces and torques) at joint i due to movement of link j alone. $\mathbf{A}_i$ is the kinematic matrix that describes the motion of link i and $\boldsymbol{\phi}_i$ is the vector of unknown link inertial parameters. All of the quantities are expressed in the local joint i coordinate system. The formulation of the above Newton-Euler equations were already presented in the load identification paper (Atkeson, An, and Hollerbach, 1985a).

The total wrench $\mathbf{w}_i$ at joint i is the sum of the wrenches w_{ij} for all links j distal to joint i:

$$\mathbf{w}_i = \sum_{j=i}^{N} \mathrm{w}_{ij} \tag{4.2}$$

Each wrench w_{ij} at joint i is determined by transmitting the distal wrench w_{jj} across intermediate joints. This is a function of the geometry of the linkage only. The forces

and torques at neighboring joints are related by

$$\begin{bmatrix} \mathbf{f}_{i,i+1} \\ \mathbf{n}_{i,i+1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & \mathbf{0} \\ [\mathbf{s}_i\times]\cdot\mathbf{R}_i & \mathbf{R}_i \end{bmatrix} \begin{bmatrix} \mathbf{f}_{i+1,i+1} \\ \mathbf{n}_{i+1,i+1} \end{bmatrix} \tag{4.3}$$

or more compactly

$$\mathbf{w}_{i,i+1} = \mathbf{T}_i\, \mathbf{w}_{i+1,i+1} \tag{4.4}$$

where

$\mathbf{R}_i$= the rotation matrix rotating the link $i+1$ coordinate system to the link i coordinate system,

$\mathbf{s}_i$= a vector from the origin of the link i coordinate system to the link $i+1$ coordinate system, and

$\mathbf{T}_i$= a wrench transmission matrix.

To obtain the forces and torques at the i^{th} joint due to the movements of the j^{th} link, these matrices can be cascaded:

$$\begin{aligned} \mathbf{w}_{ij} &= \mathbf{T}_i\mathbf{T}_{i+1}\cdots\mathbf{T}_j\mathbf{w}_{jj} \\ &= \mathbf{U}_{ij}\boldsymbol{\phi}_j \end{aligned} \tag{4.5}$$

where $\mathbf{U}_{ij} = \mathbf{T}_i\mathbf{T}_{i+1}\cdots\mathbf{T}_j\mathbf{A}_i$ and $\mathbf{U}_{ii} = \mathbf{A}_i$. A simple matrix expression for a serial kinematic chain (in this case a six joint arm) can be derived from (4.2) and (4.5):

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \\ \mathbf{w}_4 \\ \mathbf{w}_5 \\ \mathbf{w}_6 \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \mathbf{U}_{13} & \mathbf{U}_{14} & \mathbf{U}_{15} & \mathbf{U}_{16} \\ \mathbf{0} & \mathbf{U}_{22} & \mathbf{U}_{23} & \mathbf{U}_{24} & \mathbf{U}_{25} & \mathbf{U}_{26} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_{33} & \mathbf{U}_{34} & \mathbf{U}_{35} & \mathbf{U}_{36} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_{44} & \mathbf{U}_{45} & \mathbf{U}_{46} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_{55} & \mathbf{U}_{56} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{U}_{66} \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_3 \\ \boldsymbol{\phi}_4 \\ \boldsymbol{\phi}_5 \\ \boldsymbol{\phi}_6 \end{bmatrix} \tag{4.6}$$

This equation is linear in the unknown parameters, but the left side is composed of a full force-torque vector at each joint. Since only the torque about the joint axis can usually be measured, each joint wrench must be projected onto the joint rotation axis (typically $[0,0,1]$ in internal coordinates), reducing (4.6) to

$$\boldsymbol{\tau} = \mathbf{K}\boldsymbol{\psi} \tag{4.7}$$

where $\tau_i = [0,0,0,0,0,1]\cdot\mathbf{w}_i$ is the joint torque of the i^{th} link, $\boldsymbol{\psi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3, \boldsymbol{\phi}_4, \boldsymbol{\phi}_5, \boldsymbol{\phi}_6]^T$, and $\mathbf{K}_{ij} = [0,0,0,0,0,1]\cdot\mathbf{U}_{ij}$ when the corresponding entry in (4.6) is nonzero. For an n-link manipulator, $\boldsymbol{\tau}$ is a $n \times 1$ vector, $\boldsymbol{\psi}$ is a $10n \times 1$ vector, and $\mathbf{K}$ is a $n \times 10n$ matrix.

4.3.2 Estimating the Link Parameters

Equation (4.7) represents the dynamics of the manipulator for one sample point. For extra data, (4.7) is augmented as:

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}(1) \\ \cdot \\ \cdot \\ \mathbf{K}(N) \end{bmatrix} \quad \boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}(1) \\ \cdot \\ \cdot \\ \boldsymbol{\tau}(N) \end{bmatrix} \quad N = \text{number of data points}$$

Unfortunately, one cannot apply simple least squares estimate:

$$\boldsymbol{\psi}_{estimate} = (\mathbf{K}^T\mathbf{K})^{-1}\mathbf{K}^T\boldsymbol{\tau} \tag{4.8}$$

because $\mathbf{K}^T\mathbf{K}$ is not invertible due to loss of rank from restricted degrees of freedom at the proximal links and the lack of full force-torque sensing.

There are several ways to resolve this problem. One way to resolve this problem is to use the pseudo inverse to get a solution $\hat{\boldsymbol{\psi}}$ to $\boldsymbol{\tau} = \mathbf{K}\boldsymbol{\psi}$. But since $\mathbf{K}$ is a potentially large $nN \times 10n$ matrix, the pseudo-inverse is computationally inefficient. Another simple method similar to the pseudo inverse is to use ridge regression (Marquardt

and Snee, 1975). Ridge regression makes $\mathbf{K}^T\mathbf{K}$ invertible by adding a small number to the diagonal elements:

$$\hat{\psi} = (\mathbf{K}^T\mathbf{K} + d\mathbf{I}_{10n})^{-1}\mathbf{K}^T\tau \tag{4.9}$$

The estimates are nearly optimal if $d << \lambda_{min}(\mathbf{K}^T\mathbf{K})$, where λ_{min} is the smallest non-zero eigenvalue of $\mathbf{K}^T\mathbf{K}$. Other methods of solution, fundamentally different from the above two, are presented in the discussion section.

4.4 Experimental Results

Link estimation was implemented on the MIT Serial Link Direct Drive Arm (Figure 4.2), a three link serial manipulator with no transmission mechanism between the motors and the links. The ideal rigid body dynamics is a good model for this arm, uncomplicated by joint friction or backlash typical of other manipulators. Hence the fidelity of this manipulator's dynamic model suits estimation well. The coordinate system for this arm is defined in Figure 4.3. A set of inertial parameters is available for the arm (Table 4.1), determined by modeling with a CAD/CAM database (Lee, 1983). These values can serve as a point of comparison for our method, but they may not be accurate because of modeling errors.

The motors are rated at 660 Nm peak torque for joint 1 and 230 Nm for joints 2 and 3 (Asada and Youcef-Toumi, 1984). Joint 1 is presently capable of an angular acceleration of 1150 deg/sec^2, joints 2 and 3 in excess of 6000 deg/sec^2. In comparision, joint 1 of the PUMA 600 has a peak acceleration of 130 deg/sec^2 and joint 4 at the wrist 260 deg/sec^2. Joint position is measured by a resolver and joint velocity by a tachometer. The tachometer output is of high quality and leads to good acceleration estimates after differentiation. The accuracy of the acceleration estimates plus high angular accelerations greatly improves inertia estimation.

The joint torques are computed by measuring the currents of the 3 phase windings

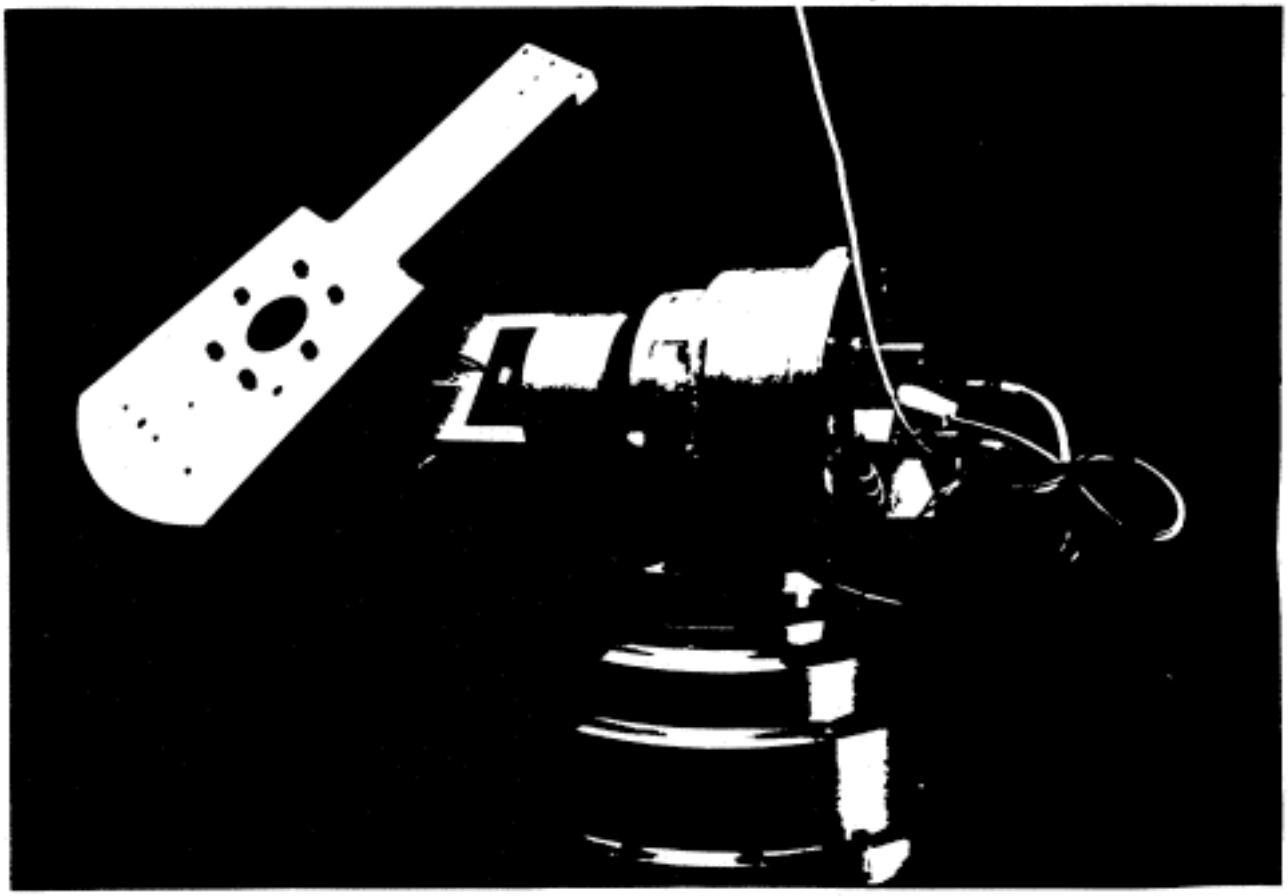


Figure 4.2: The MIT Serial Link Direct Drive Arm

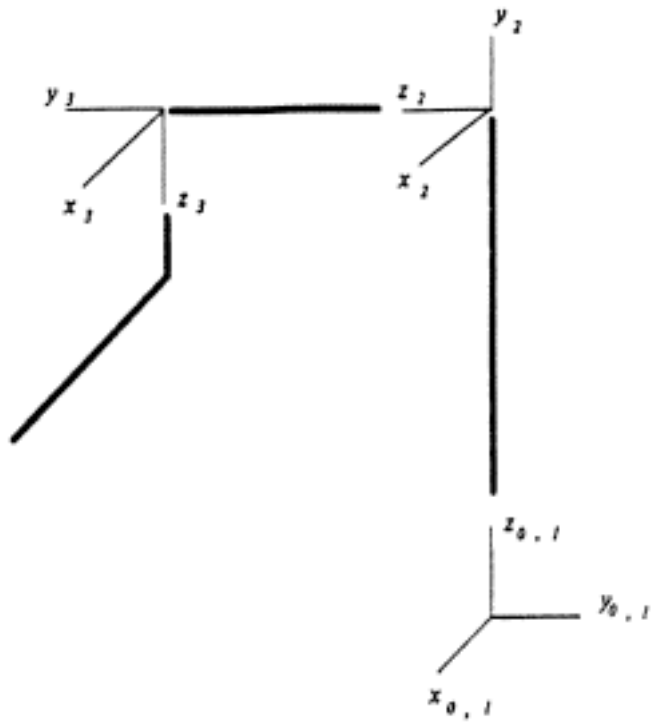


Figure 4.3: The link coordinate system.

Parameters	Link 1	Link 2	Link 3
$m(Kg)$	67.13	53.01	19.67
$mc_x(Kg \cdot m)$	0.0	0.0	0.3108
mc_y	2.432	3.4081	0.0
mc_z	35.8257	16.6505	0.3268
$I_{xx}(Kg \cdot m^2)$	23.1568	7.9088	0.1825
I_{xy}	0.0	0.0	0.0
I_{xz}	-0.3145	0.0	-0.0166
I_{yy}	20.4472	7.6766	0.4560
I_{yz}	-1.2948	-1.5036	0.0
I_{zz}	0.7418	0.6807	0.3900

Table 4.1: CAD-modeled inertial parameters.

of each motor (Asada, Youcef-Toumi, and Lim, 1984). For the three phase brushless permanent magnet motors of the direct drive arm, the output torque is related to the currents in the windings by:

$$\tau = K_T(I_a \sin\theta + I_b \sin(\theta + 120) + I_c \sin(\theta + 240)) \tag{4.10}$$

The torque constant K_T for each motor is calibrated statically by measuring the force produced by the motor torque at the end of a known lever arm. The force is measured using a BarryWright Company Astek FS6-10A-200 6-axis force/torque sensor. Asada, Youcef-Toumi, and Lim (1984) have found that for a motor similar to the motors of our manipulator, the torque versus current relationship was non-linear, especially for small magnitudes of torques, and also varied as a function of the rotor position. However, for the results presented in this paper, the nonlinear effects were ignored since substantial portions of the movements in the experiments required large magnitudes of torques. Since the least squares algorithm minimizes

the square of the error, torque errors for torques of small magnitudes do not affect the estimates very much.

For the estimation results presented, 600 data points were sampled while the manipulator was executing 3 sets of fifth order polynomial trajectories in joint space. The specifications of the trajectories were:

1. (330, 289.1, 230) to (80, 39.1, -10) degrees in 1.3s,
2. (330, 269.1, -30) to (80, 19.1, 220) degrees in 1.3s,
3. (80, 269.1, -30) to (330, 19.1, 220) degrees in 1.3s,

Since $\mathbf{K}^T\mathbf{K}$ in (4.9) is singular, estimates for the 30 unknowns are computed by adding a small number ($d = 10.0 << \lambda_{min}(\mathbf{K}^T\mathbf{K}) = 3395.0$) to the diagonal elements of $\mathbf{K}^T\mathbf{K}$.

Typical results, obtained using the ridge regression method, are shown in Table 4.2. Parameters that cannot be identified because of constrained motion near the base are denoted by 0.0*. The first nine parameters of the first link are not identifiable because this link has only one degree of freedom about its **z** axis. These nine parameters do not matter at all for the movement of the manipulator and thus can be arbitrarily set to 0.0.

Other parameters marked by (†) can only be identified in linear combinations, indicated explicitly in Table 4.3. The ridge regression automatically resolves the linear combinations in a least squares sense. It can be seen that the estimated sums roughly match the corresponding sums inferred from the CAD-modeled parameters, but the sizeable discrepancy indicates that one parameter set may be more accurate than the other.

To verify the accuracy of the estimated and the modeled parameters, the measured joint torques are compared to the torques computed from the above two sets of parameters using the measured joint kinematic data. As shown in Figure 4.4, the

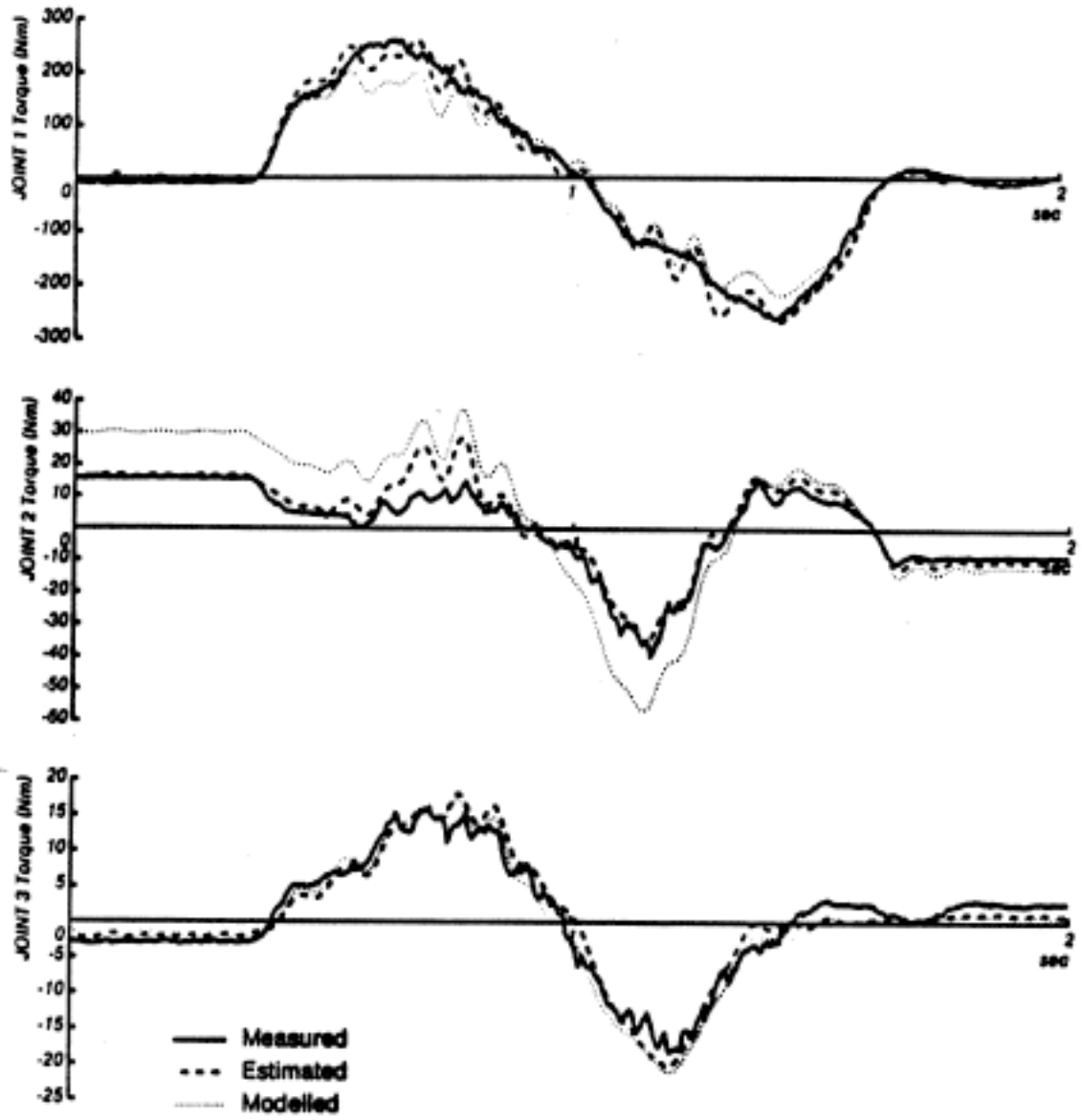


Figure 4.4: The measured, CAD-modelled, and the estimated joint torques

estimated torques match the measured torques very closely. The torques computed from the CAD/CAM modeled parameters do not match the measured torques as closely. This comparison verifies qualitatively that for control purposes the estimated parameters are in fact more accurate than the modeled parameters.

4.5 Discussion

Good estimates of the link inertial parameters were obtained, as determined from the match of predicted torques to measured torques. The potential advantage of this movement-based estimation procedure for increased accuracy as well as convenience was demonstrated by the less accurately predicted torques based on the CAD-modeled inertial parameters.

There are three groups of inertial parameters: fully identifiable, completely unidentifiable, and identifiable in linear combinations. Membership of a parameter in a group depends on the manipulator's particular geometry. Some link inertial parameters are unidentifiable because of restricted motion near the base and the lack of full force-torque sensing at each joint. For the first link, rotation is only possible about its z axis. Suppose full force-torque sensing is available at joint 1. It can be seen from (4.1) that I_{xz_1}, I_{yz_1}, and I_{yy_1} are unidentifiable because they have no effect on joint torque. Since the gravity vector is parallel to the z axis, c_{x_1} is also unidentifiable. If it is now supposed that only torque about the z axis can be sensed, then all inertial parameters for link 1 become unidentifiable except I_{zz_1}.

In a multi-link robot a new phenomenon arises. Some parameters can only be identified in linear combinations, because proximal joints must provide the torque sensing to identify fully the parameters of each link. Certain parameters from distal links are carried down to proximal links until a link appears with a rotation axis oriented appropriately for completing the identification. In between, these parameters appear in linear combinations with other parameters. This partial identifiability and

the difficulty of analysis become worse as the number of links are increased.

The ridge regression automatically resolves the linear combinations in a least squares sense, which make these inertial parameters appear superficially different from those derived by CAD modeling. An alternative method is generation and examination of the closed-form dynamics, which is a complex procedure for more than two degrees of freedom.

A second alternative is a numerical analysis via singular value decomposition of $\mathbf{K}$ in (4.8), yielding (Golub and Van Loan, 1983)

$$\mathbf{K} = \mathbf{U}\Sigma\mathbf{V}^T$$

where $\Sigma = \text{diag}\{\sigma_i\}$ and $\mathbf{U}$ and $\mathbf{V}^T$ are orthogonal matrices. For each column of $\mathbf{V}$ there corresponds a singular value σ_i which if not zero indicates that linear combination of parameters, $\mathbf{v}_i^T\boldsymbol{\psi}$, is identifiable. Since $\mathbf{K}$ is a function only of the geometry of the arm and the commanded movement, it can be generated exactly by simulation rather than by actually moving the real arm and recording data with inevitable noise.

The above two procedures isolate several sets of parameters whose linear combinations within each set are identifiable. Then we can arbitrarily set all but one of the parameters of each set to zero and apply the estimation algorithm for the reduced set of fully identifiable parameters. As shown in Table 4.2, for our 3 link manipulator, these two procedures result in grouping the 30 inertial parameters into the following categories:

1. fully identifiable: $m_3c_{x_3}$, $m_3c_{y_3}$, I_{xy_3}, I_{xx_3}, I_{yx_3}, I_{zz_3}, $m_2c_{x_2}$, I_{xy_2}, I_{xx_2}
2. completely unidentifiable: m_2, $m_2c_{x_2}$, m_1, $m_1c_{x_1}$, $m_1c_{y_1}$, $m_1c_{z_1}$, I_{xx_1}, I_{xy_1}, I_{xz_1}, I_{yy_1}, I_{yz_1}
3. identifiable in linear combinations: m_3, $m_3c_{z_3}$, I_{zz_3}, I_{yy_3}, $m_2c_{y_2}$, I_{xx_2}, I_{yy_2}, I_{yz_2}, I_{zz_2}, I_{zz_1}

Parameters	Link 1	Link 2	Link 3
$m(Kg)$	0.0*	0.0*	1.8920†
$mc_x(Kg \cdot m)$	0.0*	-0.1591	0.4676
mc_y	0.0*	0.6776†	0.0315
mc_z	0.0*	0.0*	-1.0087†
$I_{xx}(Kg \cdot m^2)$	0.0*	4.1562†	1.5276†
I_{xy}	0.0*	0.3894	-0.0256
I_{xz}	0.0*	0.0118	0.0143
I_{yy}	0.0*	5.2129†	1.8967†
I_{yz}	0.0*	-0.6050†	-0.0160
I_{zz}	9.33598†	-0.8194†	0.3568

Table 4.2: Estimated inertial parameters.

The completely unidentifiable parameters can be arbitrary set to zero. For the linear combination parameters, the combinations of these parameters that can be identified together are shown in Table 4.3. To obtain a particular solution for these parameters, one can set $m_3, m_3c_{z_3}, I_{xx_3}$, and I_{xz_3} to zero. Then, the remaining 6 parameters of this category can be treated as fully identifiable parameters along with the 9 parameters in the first category. Those columns of $\mathbf{K}$ which correspond to the 15 zeroed parameters are taken out, reducing $\mathbf{K}^T\mathbf{K}$ to a 15×15 full rank matrix. Now, simple least squares can be applied to estimate the 15 identifiable parameters. For our implementation experiments, the results of this method agreed with the results of ridge regression presented in the previous section.

Although not as simple as the ridge regression method, these methods are attractive since it allows us to reformulate the dynamics in terms of only the identifiable parameters. This then can increase the efficiency of the corresponding inverse dy-

Linear Combinations	Estimated	CAD-Modeled
$m_3 c_{x_3} l_2 + I_{yx_2}$	-1.0589	-1.3565
$I_{xx_3} - I_{yy_3}$	-0.3691	-0.2702
$I_{xx_2} + I_{xx_3}$	0.7082	0.8632
$I_{xx_1} + I_{xx_2} + I_{xx_3} + m_3 l_2^2$	15.7029	12.8169
$I_{xx_2} + I_{xx_3} - I_{yy_2}$	0.4709	0.4147
$m_3 c_{x_3} - m_2 c_{y_2}$	-1.6863	-3.0814

Table 4.3: Parameters in linear combinations ($l_2 = 0.45m$.)

namics computation for controller implementations (Hollerbach and Sahar, 1983). We are still investigating these issues with eventual goals of studying the effectiveness of different control algorithms using the estimated dynamic model and analyzing their robustness with modelling errors.

4.6 Open Questions

Statistical characterization of the errors, finding "rich" movements, quantifying desirable sensor properties and the importance of velocity sensing are important issues, just as in the rigid body load parameter estimation work.

The effect of deviations from rigid body dynamics caused by kinematic modelling errors, link flexibility, and unmodelled joint or actuator dynamics needs to be examined. For the direct drive arm we have verified experimentally the usefulness of the rigid body dynamics model, however. By identifying the inertial parameters directly rather than estimating them from a CAD/CAM model or the properties of the individual links we inadvertently compensate for some model structural error.

Chapter 5

Optimal Filtering For Parameter Estimation

5.1 Introduction

Often in parameter estimation there is a substantial amount of noise contaminating all of the sources of data. Furthermore, in many estimation situations there are variables in the parameter estimation equations that are not measured directly, but are known transformations of other measured data. These factors present problems for many parameter estimation approaches, such as least squares. This chapter presents a three step approach to these problems. The first step makes use of redundancy in the sensing to characterize the noise sources. The second step makes use of the noise characterization to design optimal filters for the different data sources. Finally, an eigenvalue/eigenvector based estimation procedure is used to produce the parameter estimates.

5.1.1 Problem Statement

The typical linear parameter estimation problem is of the following form:

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = 0 \tag{5.1}$$

or equivalently in vector form

$$\mathbf{x}^T \cdot \alpha = 0 \tag{5.2}$$

with x_i being measurements from various data sources and the α_i being the unknown parameters. Measurements from the data sources are each contaminated with statistically stationary zero mean random noise with variance σ_i.

The parameter set, $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, is actually redundant, as it can be scaled without changing the problem, so typically either the length of the parameter vector is constrained or one of the parameters is set to some arbitrarily chosen value. We will require that the length of the parameter vector is 1, ie.,

$$\sum_{i=1}^{n} \alpha_i^2 = 1 \tag{5.3}$$

Many data points consisting of sets of observations $\{x_1, x_2, \ldots, x_n\}$ can be collected. The goal here is to find a set of parameters, $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, that obey any constraints such as a fixed parameter vector length, and also in some optimal sense provide a best fit to the data. We will discuss what we mean by optimal in what follows.

5.1.2 The Least Squares Approach

One approach that is widely used is known as least squares (Silvey, 1975). A crucial feature of this approach is that all but one of the data sources are assumed to be noise free, such that the problem is restated as follows: The constraining problem structure is now:

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_{n-1} x_{n-1} = x_n \tag{5.4}$$

where α_n is arbitrarily set to -1 and

$$x_n = \tilde{x}_n - \epsilon \tag{5.5}$$

$\tilde{x}_n$ is the true value of x_n and ϵ is the measurement error. The variance of the contamination noise, σ_i, is zero for all $i < n$. All of the noise is assumed to come from the variable ϵ, which is a zero mean random variable with variance σ_n. The equivalent vector form for Equation (5.4) is

$$\mathbf{x}^T \cdot \alpha + \epsilon = x_n \tag{5.6}$$

If all of the p data points collected are used to build a data matrix, $\mathbf{A}$, and an measurement vector, $\mathbf{y}$, the following matrix equation can be written:

$$\mathbf{A} \cdot \alpha + \epsilon = \mathbf{y} \tag{5.7}$$

with the data matrix $\mathbf{A}$ being the $p \times (n-1)$ matrix

$$\mathbf{A} = \begin{pmatrix} x_{1_1} & x_{2_1} & \cdots & x_{(n-1)_1} \\ x_{1_2} & x_{2_2} & \cdots & x_{(n-1)_2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1_p} & x_{2_p} & \cdots & x_{(n-1)_p} \end{pmatrix} \tag{5.8}$$

and the measurement vector $\mathbf{y}$ being a vector of length p.

$$\mathbf{y} = \begin{pmatrix} x_{n_1} \\ x_{n_2} \\ \vdots \\ x_{n_p} \end{pmatrix} \tag{5.9}$$

The criteria to be minimized by the chosen set of parameters, $\{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}\}$, is the length of the error vector, ϵ, ie.

$$\sum_{j=1}^{p} \epsilon_j^2 \tag{5.10}$$

The estimate that minimizes this criteria is given by the normal equations:

$$\hat{\alpha} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} \tag{5.11}$$

This approach works well on many estimation problems, even though in the real world no measurements are perfect. For some problems, however, the contaminating noise on the different data sources is substantial, and cannot be ignored. In this case, a different approach is necessary.

5.1.3 Motivation For A Different Approach: A Typical Estimation Problem

Recently, we have addressed the problem of estimating the inertial parameters of a rigid body load being manipulated by a robot equipped with a wrist force/torque sensor (Atkeson, An and Hollerbach, 1985a, 1985b). We were able to formulate the rigid body dynamics in the following form:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} \ddot{\mathbf{p}} - \mathbf{g} & [\dot{\omega}\times] + [\omega\times][\omega\times] & 0 \\ 0 & [(\mathbf{g} - \ddot{\mathbf{p}})\times] & [\bullet\dot{\omega}] + [\omega\times][\bullet\omega] \end{bmatrix} \begin{bmatrix} m \\ mc_x \\ mc_y \\ mc_z \\ I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \tag{5.12}$$

The derivation of this equation and a definition of the notation is provided in the Appendix. The important point is that this problem can be formulated as a linear parameter estimation problem, in that known parameters enter non-linearly, but

the unknown parameters only enter linearly in the determination of the forces and torques on the load. Equation (5.12) can be more compactly written as

$$\mathbf{w} = \mathbf{M}\boldsymbol{\phi} \tag{5.13}$$

where $\mathbf{w}$ is a wrench vector, a vector of three forces and three torques, $\mathbf{M}$ is a matrix describing the movement of the load, and $\boldsymbol{\phi}$ is the vector of unknown inertial parameters.

We noted that the unknown parameters appeared linearly in the problem, and therefore we thought that the least squares approach to estimating the parameters could be used. In order to apply the least squares approach to this problem we had to measure the robot position, velocity, and acceleration, and the forces and torques on the wrist sensor. Furthermore, the position, velocity, and acceleration must be measured perfectly. Unfortunately, this was not a reasonable model of reality. The robot was only equipped with position sensors, and we had to numerically differentiate the position data to find the velocity and acceleration. To do this we used a discrete differentiating filter convolved with a low pass filter (Hamming, 1977). A typical sample of the data is shown in Figure 5.1. The differentiation process amplified whatever noise was in the position record, leaving the derived acceleration record greatly contaminated with noise. The force and torque data was also clearly contaminated with noise (Figure 5.2).

Thus, this problem did not easily fit into the standard least squares framework, because all of the data sources have substantial contaminating noise.

One possible approach to handling the noise is to filter the data. However, whatever filter that is applied must preserve the relationship between the different data sources, and we need to know which filter gives us the best parameter estimates. The standard least squares parameter estimation approach gave us no guidance as to how to filter the data to achieve these goals.

Another approach to this problem is to symbolically integrate the load param-

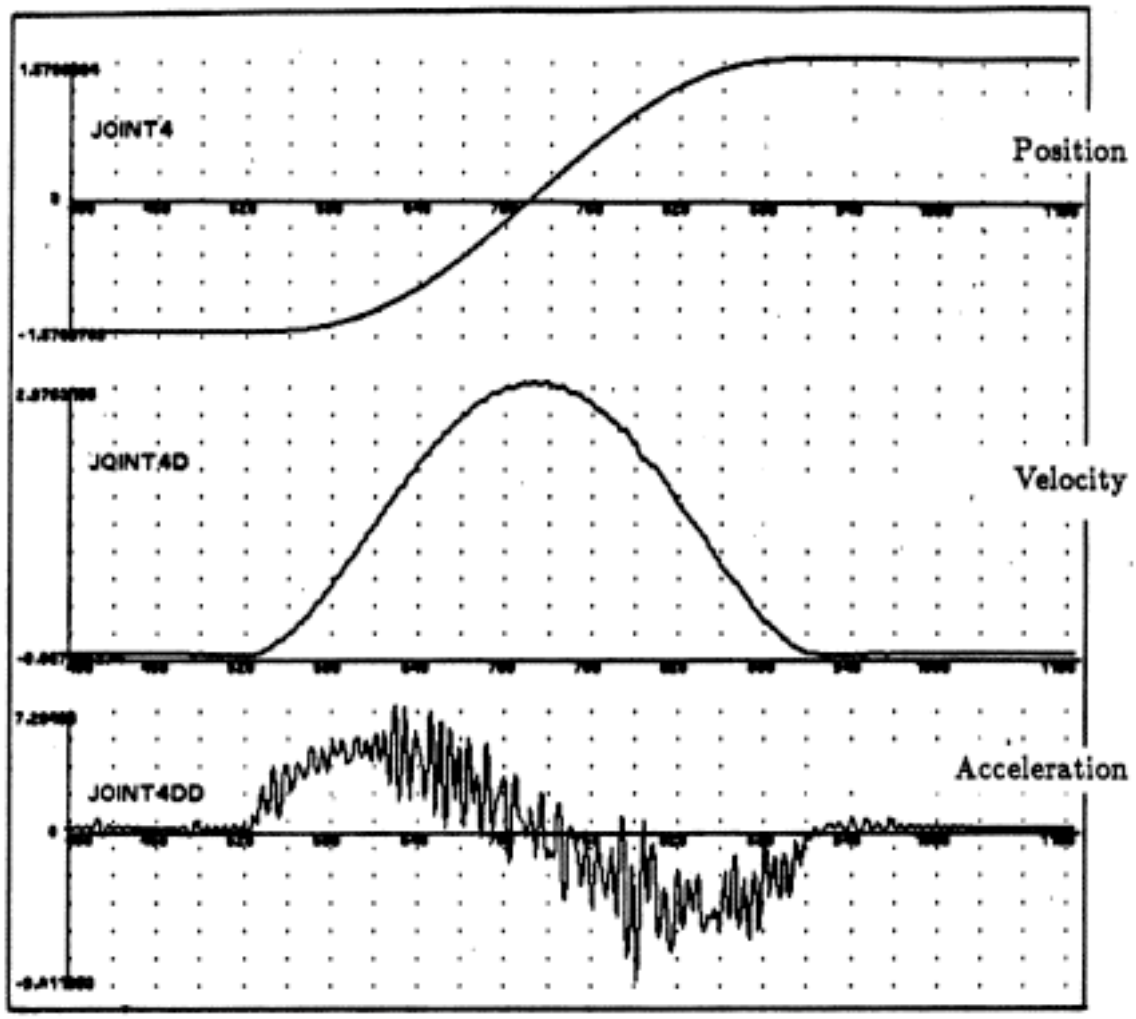


Figure 5.1: Example of measured joint angle, calculated joint angular velocity, and calculated joint angular acceleration.

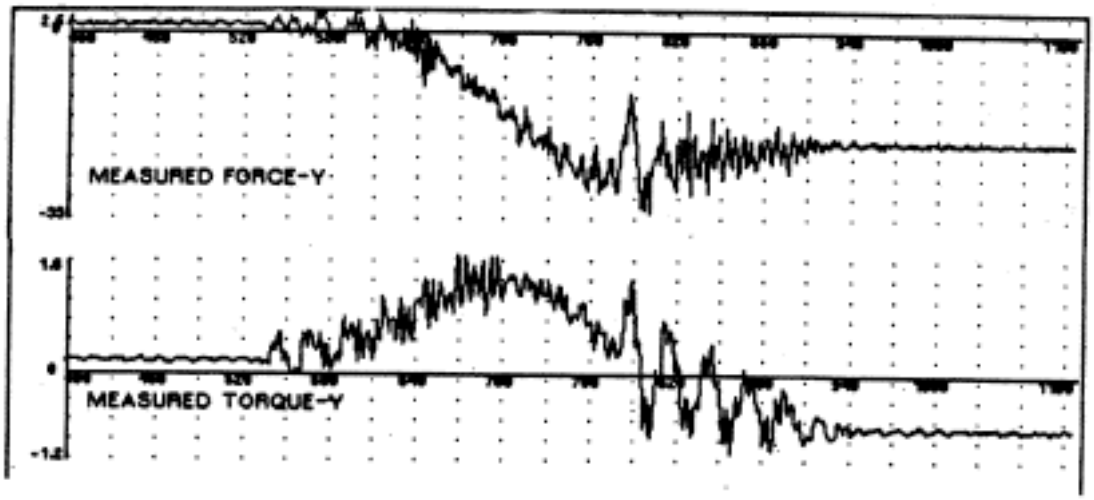


Figure 5.2: Example of measured forces and torques.

eter estimation equation, Equation (5.12), so as to avoid differentiating velocity to find accelerations. One approach to integrating the equations is presented in the appendix to this chapter, and we can express the resulting estimation equations in matrix form as:

$$\begin{bmatrix} \int_t^{t+T} \mathbf{f} \\ \int_t^{t+T} \mathbf{n} \end{bmatrix} = \left[\int_t^{t+T} \mathbf{M}\, d\tau \right] \begin{bmatrix} m \\ mc_x \\ mc_y \\ mc_z \\ I_{11} \\ I_{12} \\ I_{13} \\ I_{22} \\ I_{23} \\ I_{33} \end{bmatrix} \tag{5.14}$$

where the first row of $\left[\int_t^{t+T} \mathbf{M}\, d\tau\right]$ is

$$\left[\dot{\mathbf{p}}\Big|_t^{t+T} + \int_t^{t+T} \boldsymbol{\omega} \times \dot{\mathbf{p}}\, d\tau - \left(\int_t^{t+T} \mathbf{R}\, d\tau\right)^{\circ}\mathbf{g} \right.$$
$$\left. \left[\left(\boldsymbol{\omega}\Big|_t^{t+T}\right)\times\right] + \int_t^{t+T} [\boldsymbol{\omega}\times][\boldsymbol{\omega}\times]\, d\tau \qquad \mathbf{0}\right] \quad (15)$$

and the second row is

$$\left[\mathbf{0} \qquad \left[\left(-\dot{\mathbf{p}}\Big|_t^{t+T} - \int_t^{t+T} \boldsymbol{\omega} \times \dot{\mathbf{p}}\, d\tau + \left(\int_t^{t+T} \mathbf{R}\, d\tau\right)^{\circ}\mathbf{g}\right)\times\right] \right.$$
$$\left. \left[\bullet\left(\boldsymbol{\omega}\Big|_t^{t+T}\right)\right] + \int_t^{t+T} [\boldsymbol{\omega}\times][\bullet\boldsymbol{\omega}]\, d\tau\right] \quad (16)$$

It is not clear, however, that this is the right thing to do. If there is substantial low frequency noise or bias in the data the integration will amplify that noise relative to the signal frequencies of the data. What is needed is a problem formulation that gives us guidance as to how to design data processing filters for estimation and how to handle the need to differentiate or integrate some of the data to supply missing measurements.

5.1.4 Prototype Problem To Be Discussed As Example Of Procedure

The problem we are going to focus on in what follows is the load inertial parameter estimation problem stripped down to its bare essentials. We have a one-dimensional system and its dynamics are given by:

$$f = m \cdot a \quad (5.17)$$

which is simply *force* = *mass* · *acceleration*.

Our goal is to estimate the mass of the object as accurately as possible, given a fixed amount of data. Unfortunately, we only measure the force and the **velocity**

of the object. We wish to determine the data processing procedure to that produces the best estimates, in some sense. We have several choices:

- Differentiate velocity to estimate acceleration, and use the equation

$$f(t) = m\frac{d}{dt}v(t) \tag{5.18}$$

  to produce the estimates.

- Integrate the force, and use the equation

$$\int_{t_i}^{t_f} f(t)dt = m(v(t_f) - v(t_i)) \tag{5.19}$$

  to produce the estimates.

- Differentiate velocity to estimate acceleration, but then apply a filter of some sort to "clean up" the data. The estimates are produced from an equation of the form

$$l * f(t) = m(l * \frac{d}{dt}) * v(t) \tag{5.20}$$

  where l is some filter and $*$ is the convolution operator.

The goal of this chapter is to show which of these choices is the best choice to make, and show what factors affect the choice of data processing procedure. To do this, we will start by explaining how an estimation procedure that can handle noise in all the data works. We will first show how standard least squares is applied, and we want to look carefully at what criteria it optimizes and how it handles colored noise. Next we shall examine an eigenvalue/eigenvector estimation procedure, how it applies to noise on all data case, and how it handles colored noise. We will explain how the colored noise estimator can be partitioned into a white noise estimator preceded by appropriate data filters. Then we will ask what data processing produces the best estimates from this estimation procedure, and then we will show how to generate the information about the data and the noise that allows us to design the appropriate data processing procedures.

5.2 The Estimation Procedure

We will now discuss application of the estimation procedure to the prototype problem, estimating the mass of a one-dimensional load. First, we must describe notation for all the quantities we need to keep track of. $f(t)$, $a(t)$, and $v(t)$ are measured or estimated forces, accelerations, and velocities. $\tilde{f}(t)$, $\tilde{a}(t)$, and $\tilde{v}(t)$ are the underlying true values for these quantities. m is the true mass of the object, and $\hat{m}$ is the estimated mass. Note that

$$\tilde{f}(t) = m \cdot \tilde{a}(t) = m \cdot \frac{d}{dt}\tilde{v}(t) \tag{5.21}$$

holds for the true values of the data, but since

$$f(t) = \tilde{f}(t) + n_f(t) \tag{5.22}$$

$$a(t) = \tilde{a}(t) + n_a(t) \tag{5.23}$$

$$v(t) = \tilde{v}(t) + n_v(t) \tag{5.24}$$

where $n_f(t)$, $n_a(t)$, and $n_v(t)$ are the noise contaminating the data, the corresponding equalities for the measured data do not hold:

$$f(t) \neq m \cdot a(t) \neq m \cdot \frac{d}{dt}v(t) \tag{5.25}$$

5.2.1 Application Of Standard Least Squares To Prototype Problem

What is presented in this section is well known (Silvey, 1975, for example) and is presented so as to make what follows more comprehensible.

To show how to apply standard least squares to this problem, we must assume we have a perfect acceleration measurement, $a(t) = \tilde{a}(t)$, but a noise contaminated force measurement, $f(t) = m\tilde{a}(t) - n_f(t)$. We will model the noise, $n_f(t)$, as zero

mean, gaussian random noise. $n_f(t)$ is not necessarily "white" noise, however, but can have any power spectrum, denoted by $S_{n_f}(\omega)$.

We can estimate $\hat{m}$ using standard least squares in the following way. In the white noise case, $S_{n_f}(\omega) = 1$, the appropriate criteria to minimize is

$$\sum_t \epsilon^2(t) \tag{5.26}$$

where

$$\epsilon(t) = m \cdot a(t) - f(t) \tag{5.27}$$

We express the data in matrix/vector form $\mathbf{A}\boldsymbol{\phi} = \mathbf{f} + \boldsymbol{\epsilon}$:

$$\begin{pmatrix} a(1) \\ a(2) \\ \vdots \\ a(p) \end{pmatrix} \begin{pmatrix} m \end{pmatrix} = \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(p) \end{pmatrix} + \begin{pmatrix} \epsilon(1) \\ \epsilon(2) \\ \vdots \\ \epsilon(p) \end{pmatrix} \tag{5.28}$$

and the solution is given by the normal equations

$$\hat{\boldsymbol{\phi}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{f} \tag{5.29}$$

The standard least squares approach minimizes errors along only one coordinate direction (Figure 5.3A). This should be contrasted with the eigenvalue/eigenvector approach.

Effect Of Colored Noise

The effect of colored noise (the noise spectrum S_{n_f} is not constant) is to lead us to weight the errors in our criteria for an optimal estimate. To show this we must analyse how our parameter estimation schemes work in the frequency domain.

Since

$$\epsilon(t) = m \cdot a(t) - f(t) = n_f(t), \tag{5.30}$$

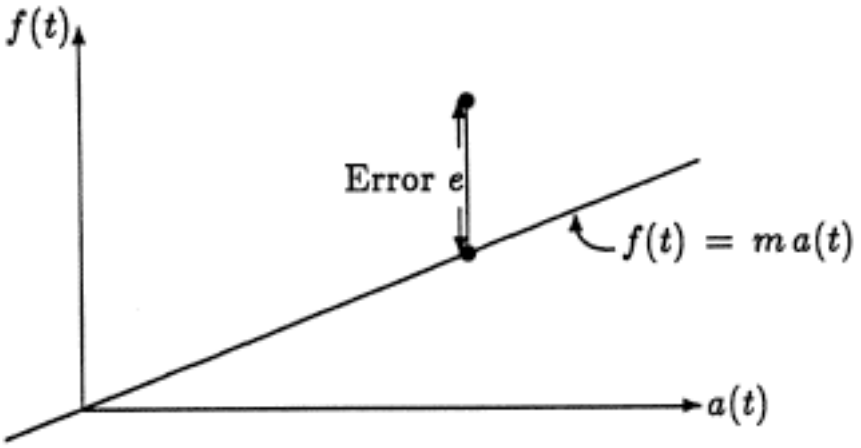


Figure 5.3: A

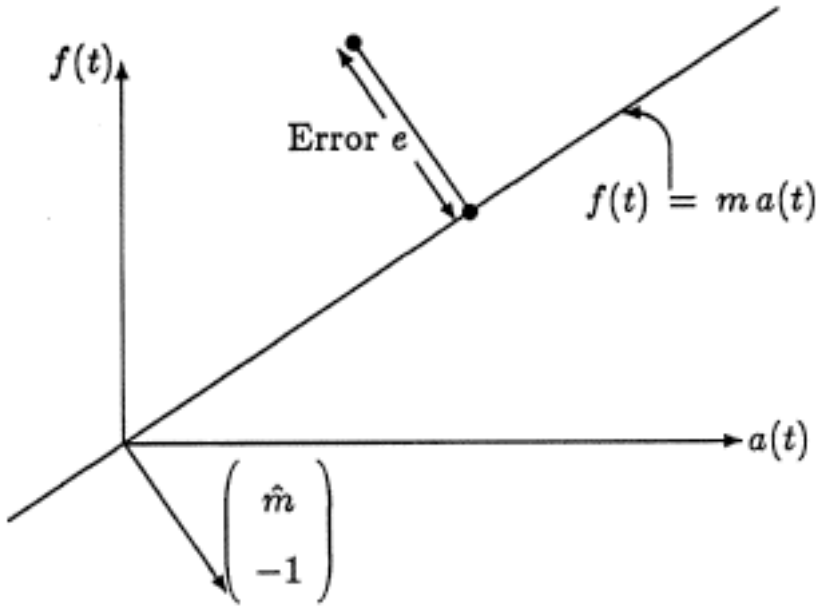


Figure 5.3: B

the power spectrum of the noise, S_ϵ, is simply the power spectrum of the noise contaminating the force measurements, S_{n_f}. In the white noise case (the noise spectrum S_{n_f} is constant) we wanted to minimize the criteria

$$\sum_t \epsilon^2(t) = \sum_t (m \cdot a(t) - f(t))^2 \tag{5.31}$$

and by Parseval's Theorem (Oppenheim and Wilsky, 1983)

$$\sum_t \epsilon^2(t) = \sum_\omega |E(\omega)|^2 \tag{5.32}$$

where $E(\omega)$ is the Fourier transform of $\epsilon(t)$. Since the expected spectrum of the noise was flat, it was reasonable to weight the error components at different frequencies equally.

In the colored noise case weighting the errors equally at different frequencies no longer makes sense. Since the power spectrum of the noise is $S_{n_f}(\omega)$ and is not constant, we should weight the error components at different frequencies according to the expected amount of error at that frequency. Our optimal estimation criteria to minimize becomes (in the frequency domain):

$$\sum_\omega \frac{|E(\omega)|^2}{S_{n_f}(\omega)} = \sum_\omega \frac{|m \cdot A(\omega) - F(\omega)|^2}{S_{n_f}(\omega)} \tag{5.33}$$

This is equivalent to prefiltering the data with the frequency domain filter

$$\frac{1}{\sqrt{S_{n_f}(\omega)}} \tag{5.34}$$

or convolving the data with the inverse fourier transform of this filter.

5.2.2 Handling Noise On All Measurements

The eigenvalue/eigenvector parameter estimation procedure presented in this section was originally developed by (Koopmans, 1937), and (Levin, 1964) made use of it in pulse transfer function estimation. Many others have extended it in various ways:

(Smith, 1968, Bonivento and Guidorzi, 1971, Grosjean and Foulard, 1978, Kotta, 1979, Tian-Zing, et. al., 1982, Kotta, 1982)

Now let us assume there is noise on both the force, n_f, and the acceleration, n_a, measurement:

$$f(t) = \tilde{f}(t) - n_f(t) \tag{5.35}$$

$$a(t) = \tilde{a}(t) - n_a(t) \tag{5.36}$$

Both noise sources are zero mean, gaussian, and with power spectra S_{n_f} and S_{n_a} respectively. The noise sources are independent of each other, also. Initially, let us assume that variance of n_a and the variance of n_f are equal, and the noise sources are both white.

Now we need to reexamine what the error is that we are minimizing. We see in Figure 5.3B that the error we now want to minimize is in the direction of the vector

$$\begin{pmatrix} \hat{m} \\ -1 \end{pmatrix} \tag{5.37}$$

and for a data point $(a(t), f(t))$, the error is given by

$$\epsilon = \frac{\hat{m}a(t) - f(t)}{\left(\begin{pmatrix} m & -1 \end{pmatrix} \begin{pmatrix} m \\ -1 \end{pmatrix} \right)^{1/2}} \tag{5.38}$$

This is the component of the "true" error perpendicular to the relationship $f(t) = \hat{m} \cdot a(t)$.

We would like the estimate $\hat{m}$ to minimize

$$\sum_t \epsilon^2(t) = \sum_t \frac{(\hat{m} \cdot a(t) - f(t))^2}{\hat{m}^2 + 1} \tag{5.39}$$

This is a nonlinear minimization problem since the desired estimate, $\hat{m}$ appears in both the numerator and the denominator of the objective function, and is squared in the denominator. Thus, we cannot apply standard least squares techniques to this problem, or use the normal equations to find the estimate.

An Eigenvalue/Eigenvector Based Estimation Procedure

We can solve this nonlinear estimation problem if we think of it as a linear algebra problem (Koopmans, 1937). We express $\epsilon(t)$ as a vector:

$$\begin{pmatrix} a(1) & f(1) \\ a(2) & f(2) \\ \vdots & \vdots \\ a(p) & f(p) \end{pmatrix} \frac{\begin{pmatrix} \hat{m} \\ -1 \end{pmatrix}}{(\hat{m}^2+1)^{1/2}} = \begin{pmatrix} \epsilon(1) \\ \epsilon(2) \\ \vdots \\ \epsilon(p) \end{pmatrix} \tag{5.40}$$

and notice that the above forms the matrix/vector equation:

$$\mathbf{B}\mathbf{v} = \epsilon \tag{5.41}$$

Note that $\mathbf{v}$ has unit length.

Our estimation problem is now to find the unit length vector $\mathbf{v}$ that minimizes the length of $\mathbf{B}\mathbf{v}$. This will minimize

$$\sum_t \epsilon^2(t) \tag{5.42}$$

as we desire. Now that we have reformulated the estimation problem as a linear algebra problem, we can recognize that the desired $\hat{\mathbf{v}}$ is simply the right eigenvector of $\mathbf{B}$ corresponding to the smallest singular value of $\mathbf{B}$, or equivalently, $\hat{\mathbf{v}}$ is the right eigenvector corresponding to the smallest eigenvalue of $\mathbf{B}^T\mathbf{B}$ (Golub and Van Loan, 1983). To see this, we realize that minimizing the square of the errors is equivalent to minimizing

$$\sum_t \epsilon^2(t) = \hat{\mathbf{v}}^T\mathbf{B}^T\mathbf{B}\hat{\mathbf{v}} \tag{5.43}$$

We merely decompose a candidate $\hat{\mathbf{v}}$ into its projection along the orthogonal eigenvectors of $\mathbf{B}^T\mathbf{B}$, i.e.,

$$\hat{\mathbf{v}} = \gamma_1\mathbf{v}_1 + \gamma_2\mathbf{v}_2 + \cdots + \gamma_p\mathbf{v}_p \tag{5.44}$$

where $\mathbf{v}_1$ is the eigenvector corresponding to the smallest eigenvalue λ_1. Since $\mathbf{v}$ is of unit length, $\sum_{j=1}^{p} \gamma_j^2 = 1$. The function to minimize is now

$$\hat{\mathbf{v}}^T \mathbf{B}^T \mathbf{B} \hat{\mathbf{v}} = \lambda_1^2 \gamma_1^2 + \lambda_2^2 \gamma_2^2 + \cdots + \lambda_p^2 \gamma_p^2 \tag{5.45}$$

To minimize this sum we should choose $\hat{\mathbf{v}} = \mathbf{v}_1$, as previously stated. $\hat{m}$ can be easily computed from $\hat{\mathbf{v}}$, although it may be hard to find the error distribution of $\hat{m}$ given the error distribution in $\hat{\mathbf{v}}$.

Effect Of Colored Noise

The effect of colored noise is once again to lead us to weight the errors in our criteria for an optimal estimate. We once again look at the expected power spectrum of the noise:

$$\sum_{\omega} |E(\omega)|^2 = \sum_{\omega} \frac{|m \cdot A(\omega) - F(\omega)|^2}{m^2 + 1} \tag{5.46}$$

We note that the expected power spectrum of the noise is

$$S_e = \frac{m^2 S_{n_a} + S_{n_f}}{m^2 + 1} \tag{5.47}$$

and once again we should weight the errors. Since the denominator in the above expression for the power spectrum is a constant, it can be ignored in the weighting. The estimate $\hat{m}$ should minimize

$$\frac{\sum_{\omega} \frac{|m \cdot A(\omega) - F(\omega)|^2}{m^2 S_{n_a} + S_{n_f}}}{m^2 + 1} \tag{5.48}$$

This is equivalent to prefiltering the data with the frequency domain filter

$$\frac{1}{\sqrt{m^2 S_{n_a} + S_{n_f}}} \tag{5.49}$$

We note that our data filter we use to estimate $\hat{m}$ depends on the true value of the unknown parameter, m. We therefore are forced either to adopt an iterative estimation procedure, or to simply approximate our filters to the theoretically optimal

filters. We chose the second course, as it is not clear we will know the noise spectra exactly enough to know the true data filters to the accuracy where approximating m would make a difference.

Handling Missing Measurements

When there are data that appear in the parameter estimation equations but are not measured directly we must generate that missing data. This requires that we measure some other data that is directly related to the missing data by some known transformation. We apply that known transformation to the measured data to estimate the missing data, and we also need to transform the noise power spectrum of the measured data to generate the noise power spectrum of the missing data. In this example we reconstruct acceleration from velocity measurements.

Conclusions On Our Estimation Procedure

1. When there is substantial noise on more than one source of data it is appropriate to use an eigenvalue/eigenvector based parameter estimation algorithm.

2. The optimal filter (in the frequency domain) for the prototype problem is given by

$$\frac{1}{\sqrt{m^2 S_{n_a} + S_{n_f}}} \tag{5.50}$$

3. To design the optimal filter we need to know noise characteristics such as the power spectra S_{n_a} and S_{n_f}.

4. The true optimal filter depends on the true values of the parameters to be estimated. We are forced either to iterate or to approximate, and we choose approximation in this chapter.

5. This estimation approach can easily be generalized to apply to more complex problems with more than two measurements.

6. Missing data and its noise spectrum should be reconstructed from other measured data if possible.

5.3 A Filter Design Example

Figure 5.4 shows an example parameter estimation problem and the resulting optimal filters.

We present a simple example to clarify the concepts presented in the previous sections. Force and velocity are measured as in the prototypical problem, and the noise on each is independent zero mean white gaussian noise with variance 1. Therefore their power spectra are simply constants. We note that differentiating the velocity to estimate the acceleration will result in an acceleration noise spectrum equal to ω^2. The true value of the mass parameter is assumed to be 1 in this example, so the frequency domain data filter is

$$\frac{1}{\sqrt{m^2 S_{n_a} + S_{n_f}}} = \frac{1}{\sqrt{\omega^2 + 1}} \tag{5.51}$$

The frequency domain force filter is

$$\frac{1}{\sqrt{\omega^2 + 1}} \tag{5.52}$$

while the frequency domain velocity filter combines a differentiating filter with the data filter. The magnitude of the velocity filter is

$$\frac{\omega}{\sqrt{\omega^2 + 1}} \tag{5.53}$$

and the phase of the filter is a 90° phase advance at all frequencies, due to the differentiator.

Let us examine what happens to the filters as the frequency, ω, goes to zero and infinity. As the frequency goes to 0, the force filter becomes

$$\lim_{\omega \to 0} \frac{1}{\sqrt{\omega^2 + 1}} \to 1 \tag{5.54}$$

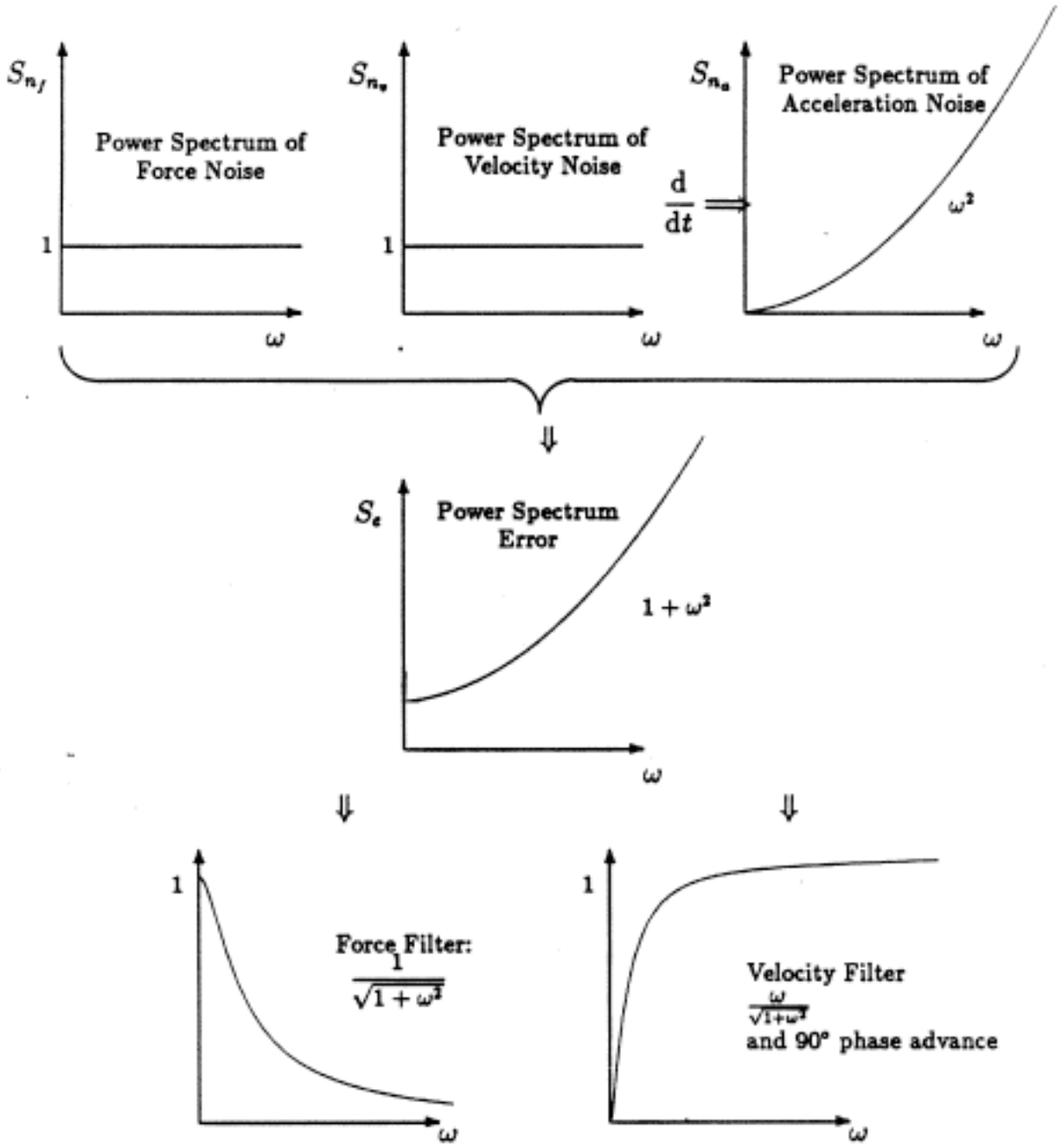


Figure 5.4: Example filters. (Mass = 1)

and the velocity filter becomes

$$\lim_{\omega \to 0} \frac{\omega}{\sqrt{\omega^2 + 1}} \to \omega \tag{5.55}$$

in the frequency domain. What is happening is at low frequencies the optimal filters essentially just differentiate velocity, and do nothing to the force signal. As the frequency goes to ∞, the force filter becomes

$$\lim_{\omega \to \infty} \frac{1}{\sqrt{\omega^2 + 1}} \to \frac{1}{\omega} \tag{5.56}$$

and the velocity filter becomes

$$\lim_{\omega \to \infty} \frac{\omega}{\sqrt{\omega^2 + 1}} \to 1 \tag{5.57}$$

in the frequency domain. What is happening here is at high frequencies the optimal filters essentially do nothing to the velocity signal, and integrate the force signal.

We see from the above limits and from the magnitude plots of the optimal filters in Figure 5.4 that the optimal filters avoid differentiating the velocity at high frequencies, where noise would be greatly amplified, but they also avoid integrating the force at low frequencies, where bias and drift would be greatly amplified (Figure 5.5).

The details of the filters depend on the noise spectra and the a priori estimates of the true values of the unknown parameters, but the filters make intuitive sense in what they are trying to do. We note that setting the acceleration noise to zero tells us that the data filter should be just as the standard least squares approach predicts.

5.4 Characterizing The Noise

We see that in order to design optimal filters for the data we need to know the power spectra of the noise. Given that in real life one never knows the true value of the

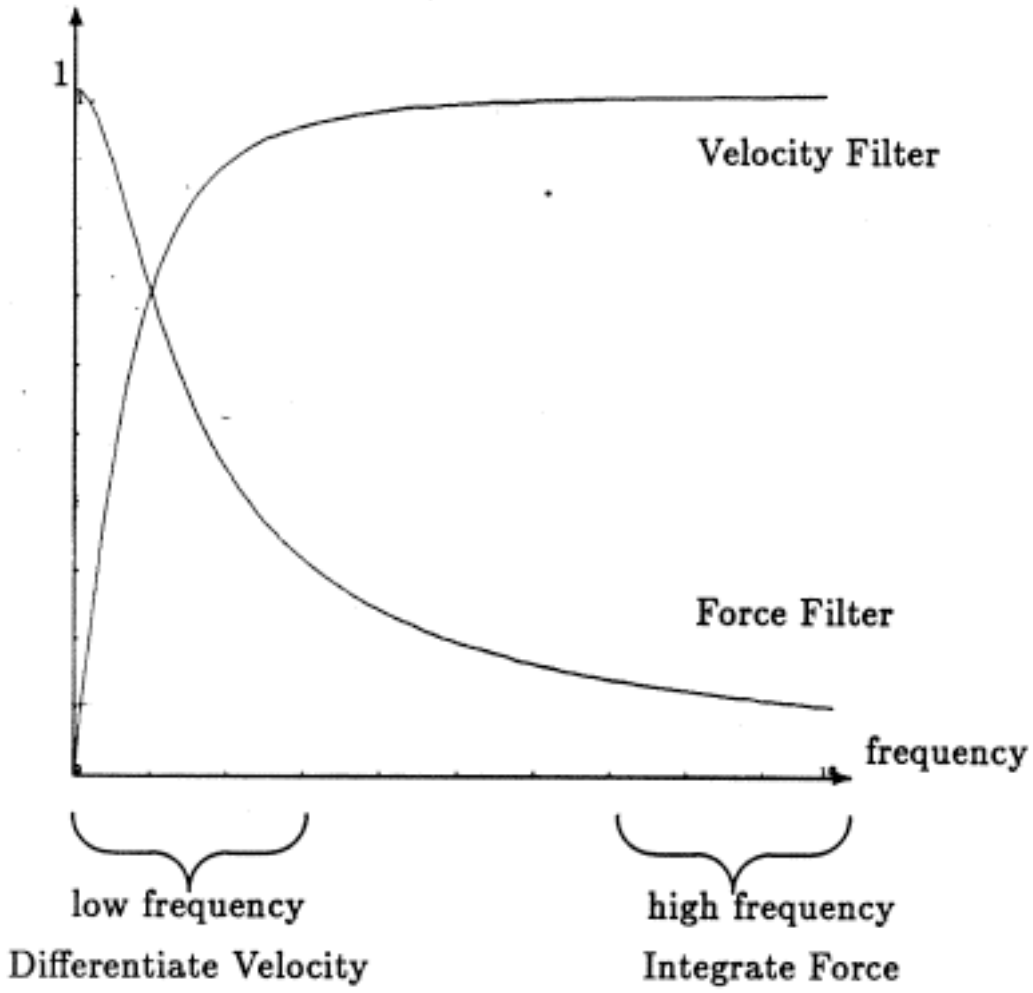


Figure 5.5: Example Filters Compared.

signal, but can only look at a noisy measurement, it seems unrealistic to expect to be able to find the noise power spectra. However, with no more sensing than the sensing we have already postulated, we can estimate the noise spectra of the data. We simply make use of the redundancy in the sensing.

Let us assume we measure force and velocity

$$f(t) = \tilde{f}(t) + n_f(t) \tag{5.58}$$

$$v(t) = \tilde{v}(t) + n_v(t) \tag{5.59}$$

We will let the true value of the mass of the object be 1 for this discussion, so that $\tilde{f}(t) = \frac{d}{dt}\tilde{v}(t)$, although it is quite easy to derive the same results for the general case. We assume that $n_f(t)$ and $n_v(t)$ are zero mean, stationary, and independent.

Let us examine what the power spectra of the observed data will be. First we note that the transfer function from velocity to force is

$$s \equiv H(s) \tag{5.60}$$

The power spectrum of the true force signal will be related to the power spectra of the true velocity signal by

$$S_{\tilde{f}}(\omega) = H(j\omega)H^*(j\omega)S_{\tilde{v}}(\omega) \tag{5.61}$$

and the true cross spectra will be

$$S_{\tilde{v}\tilde{f}}(\omega) = H(j\omega)S_{\tilde{v}}(\omega) = \frac{S_{\tilde{f}}(\omega)}{H^*(j\omega)} \tag{5.62}$$

The spectra and cross spectra of the actual data can now be easily found:

$$S_f(\omega) = S_{\tilde{f}}(\omega) + S_{n_f}(\omega) \tag{5.63}$$

$$S_v(\omega) = S_{\tilde{v}}(\omega) + S_{n_v}(\omega) \tag{5.64}$$

$$S_{vf}(\omega) = H(j\omega)S_{\tilde{v}}(\omega) = \frac{S_{\tilde{f}}(\omega)}{H^*(j\omega)} \tag{5.65}$$

This provides us with enough information to estimate the noise spectra:

$$\hat{S}_{n_f}(\omega) = S_f(\omega) - S_{vf}(\omega)H^*(j\omega) \tag{5.66}$$

$$\hat{S}_{n_v}(\omega) = S_v(\omega) - \frac{S_{vf}(\omega)}{H(j\omega)} \tag{5.67}$$

We have implemented this approach to noise characterization on simulated data and noise, and it seems to perform well. We have yet to test it on real data or actual noise.

Conclusions On Noise Characterization

1. We can use "redundant" sensing to characterize noise spectra.
2. We need to know the true underlying relationship between the redundant signals. In this case we needed to know:
   - The true mass m.
   - $\tilde{f}(t) = m \cdot \tilde{a}(t)$
   - $\tilde{a}(t) = \frac{d}{dt}\tilde{v}(t)$
3. This approach to noise characterization generalizes to many other redundant sensing situations.

5.5 Discussion

One may ask whether the additional complexity in data processing suggested by this chapter is worth the effort. We are in the process of applying this scheme to a real parameter estimation problem with data from actual hardware. This will enable us to more concretely assess the procedure's benefits.

We should point out, however, that one makes data filtering decisions all the time. If we numerically differentiate data we also low pass filter it, and must choose

the characteristics of the filter. If we integrate data, we must choose integration intervals, which set the filtering characteristics of the integrator. We should at least be aware of what good filtering decisions look like, even though we may not apply the optimal filters directly.

We can ask whether the filtering that is suggested here makes intuitive sense. Clearly, the noise matters in designing a filter. If there is almost no noise in a certain frequency range we should pay attention to those frequencies, while if the noise is much larger than any signal at some frequencies we should probably ignore those frequencies. This is exactly what the filter tells us to do. Also, the decision of when to integrate versus when to differentiate data seems reasonable, as explained in the filter design example section.

We are solving a nonlinear estimation problem, and one would have expected an iterative process somewhere. Note that the computation of eigenvalues and eigenvectors, or singular values and singular vectors require an iterative algorithm, and this is where the iteration occurs. Such eigenvalue/singular value computation procedures are extremely well developed and a lot of effort has been put in to making them numerically stable (Golub and Van Loan, 1983).

5.6 Conclusions

We have developed a parameter estimation procedure that shows us appropriate ways to filter data for parameter estimation. The procedure addresses two main problems:

1. Noise is typically present in all data.

2. Often we must reconstruct missing measurements.

5.7 Appendix

5.7.1 Deriving The Dynamics Equations

The equations for the forces and torques (expressed in the load coordinate system) necessary to move a load along a given trajectory are

$$\mathbf{f} = m\ddot{\mathbf{p}} - m\mathbf{g} + \dot{\boldsymbol{\omega}} \times m\mathbf{c} + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times m\mathbf{c}) \tag{5.68}$$

$$\mathbf{n} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) + m\mathbf{c} \times \ddot{\mathbf{p}} - m\mathbf{c} \times \mathbf{g} \tag{5.69}$$

where:

$\mathbf{f}$ = the net force on the load,
m = the mass of the load,
$\ddot{\mathbf{p}}$ = the acceleration of the load,
$\mathbf{g}$ = the gravity vector ($\mathbf{g} = [0, 0, -9.8\,\text{meters/sec}^2]$),
$\boldsymbol{\omega}$ = the angular velocity vector,
$\dot{\boldsymbol{\omega}}$ = the angular acceleration vector,
$\mathbf{c}$ = the unknown location of the center of mass of the load, relative to the force sensing coordinate system origin,
$\mathbf{n}$ = the net torque on the load, and
$\mathbf{I}$ = the moment of inertia tensor of the load about the force sensing coordinate system origin.

In order to formulate the above equations as a system of linear equations, the following notation is used:

$$\boldsymbol{\omega} \times \mathbf{c} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \triangleq [\boldsymbol{\omega}\times]\,\mathbf{c} \tag{5.70}$$

$$\mathbf{I}\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z & 0 & 0 & 0 \\ 0 & \omega_x & 0 & \omega_y & \omega_z & 0 \\ 0 & 0 & \omega_x & 0 & \omega_y & \omega_z \end{bmatrix} \begin{bmatrix} I_{xx} \\ I_{xy} \\ I_{xz} \\ I_{yy} \\ I_{yz} \\ I_{zz} \end{bmatrix} \triangleq [\bullet\omega] \begin{bmatrix} I_{xx} \\ I_{xy} \\ I_{xz} \\ I_{yy} \\ I_{yz} \\ I_{zz} \end{bmatrix} \tag{5.71}$$

where

$$\mathbf{I} = \mathbf{I}^T = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \tag{5.72}$$

Using these expressions, Equations (5.68) and (5.69) can be written as a single matrix equation expressed in load coordinate frame. This load dynamics formulation is presented in Equation (5.12).

5.7.2 Integrating The Forces And Torques In The Force Sensor Frame

Throughout this derivation we will use the superscript notation $^{\circ}$ and ^{p} to indicate what coordinate frame a vector is expressed in, if there is any confusion. If there is no such superscript, the vector is expressed in the load frame.

The term $\ddot{\mathbf{p}}$ is integrated using equation 7.22 of (Symon, 1971):

$$\mathbf{R}\frac{d}{dt}({}^{\circ}\dot{\mathbf{p}}) = \frac{d}{dt}(\mathbf{R}\cdot{}^{\circ}\dot{\mathbf{p}}) + \mathbf{R}({}^{\circ}\omega \times {}^{\circ}\dot{\mathbf{p}}) \tag{5.73}$$

were $\mathbf{R}$ is the rotation matrix representing the rotation from an inertial coordinate system to the force sensing coordinate system that continuously moves with the load. To get an integral form of this equation we simply integrate it:

$$\int_t^{t+T} \mathbf{R}\cdot{}^{\circ}\ddot{\mathbf{p}}\, d\tau = (\mathbf{R}\cdot{}^{\circ}\dot{\mathbf{p}})\Big|_t^{t+T} + \int_t^{t+T} \mathbf{R}({}^{\circ}\omega \times {}^{\circ}\dot{\mathbf{p}})\, d\tau \tag{5.74}$$

and performing the indicated rotations leaves us with

$$\int_t^{t+T} {}^p\ddot{\mathbf{p}}\, d\tau = {}^p\dot{\mathbf{p}}\Big|_t^{t+T} + \int_t^{t+T} {}^p\boldsymbol{\omega} \times {}^p\dot{\mathbf{p}}\, d\tau \tag{5.75}$$

Similarly

$$\int_t^{t+T} {}^p\dot{\boldsymbol{\omega}}\, d\tau = {}^p\boldsymbol{\omega}\Big|_t^{t+T} + \int_t^{t+T} {}^p\boldsymbol{\omega} \times {}^p\boldsymbol{\omega}\, d\tau = {}^p\boldsymbol{\omega}\Big|_t^{t+T} \tag{5.76}$$

since $\boldsymbol{\omega} \times \boldsymbol{\omega} = 0$. We also remember that

$$\int_t^{t+T} {}^p\mathbf{g}\, d\tau = \left(\int_t^{t+T} \mathbf{R}\, d\tau\right) {}^o\mathbf{g} \tag{5.77}$$

Each of the matrices $[{}^p\dot{\boldsymbol{\omega}}\times]$ and $[\bullet\dot{\boldsymbol{\omega}}]$ can be integrated element by element to show that

$$\int_t^{t+T} [{}^p\dot{\boldsymbol{\omega}}\times]\, d\tau = \left[\left(\int_t^{t+T} {}^p\dot{\boldsymbol{\omega}}\, d\tau\right)\times\right] = \left[\left({}^p\boldsymbol{\omega}\Big|_t^{t+T}\right)\times\right] \tag{5.78}$$

$$\int_t^{t+T} [\bullet {}^p\dot{\boldsymbol{\omega}}]\, d\tau = \left[\bullet\left(\int_t^{t+T} {}^p\dot{\boldsymbol{\omega}}\, d\tau\right)\right] = \left[\bullet\left({}^p\boldsymbol{\omega}\Big|_t^{t+T}\right)\right] \tag{5.79}$$

The matrix $[(\mathbf{g} - \ddot{\mathbf{p}})\times]$ can be integrated element by element in the same way. Each matrix element of the terms $[{}^p\boldsymbol{\omega}\times][{}^p\boldsymbol{\omega}\times]$ and $[{}^p\boldsymbol{\omega}\times][\bullet {}^p\boldsymbol{\omega}]$ are numerically integrated by adding values at each time step. The result of this integration is given in Equation (5.14).

Chapter 6

Single Trajectory Learning

6.1 Abstract

We present an algorithm enabling a robot that is repeating the same trajectory to reduce its trajectory error on repetition of the motion. The algorithm details how to use the trajectory following error (the time history of the differences between the desired trajectory and the actually executed trajectory) after any particular movement to update an actuator feedforward command that is used for the subsequent attempt. This approach to robot learning is based on explicit modeling of the robot; and uses an inverse of the robot model as the learning operator which processes the trajectory errors. Results are presented from a successful implementation of this procedure on the MIT Serial Link Direct Drive Arm. The major point of this chapter is that more accurate robot models improve trajectory learning performance, and learning algorithms do not reduce the need for good models in robot control.

[1]This chapter is a revised version of (Atkeson and McIntyre, 1986)

6.2 Introduction

Adaptive feedforward control for repetitive motions: In actual use robots tend to execute the same sequence of motions with the same loads repeatedly. We can take advantage of this pattern of usage by specializing the robot control system to store feedforward commands in a memory and play them back when necessary. This type of control system would repeat its errors on each movement, in contrast to the performance improvement with practice seen in human movement control. We propose an algorithm that uses practice to improve movement execution, by altering the stored feedforward commands on the basis of previous movement errors. This serves two purposes: 1) to improve robot trajectory following for repetitive movements, and 2) to increase our understanding of the role of practice in human motor control.

6.2.1 Previous Work

Recent work in a number of laboratories has focused on how to refine feedforward commands for repetitive movements on the basis of previous movement errors. The first paper on repeated trajectory learning seems to have been (Uchiyama, 1978). Subsequent work includes (Arimoto, 1985; Arimoto et. al., 1984a, 1984b, 1984c, 1985; Hara, Omata, and Nakano, 1985; Kawamura et. al., 1984, 1985; Casalino and Gambardella, 1986; Craig, 1984, 1983; Furuta and Yamakita, 1986; Harokopos, 1986a, 1986b; Mita and Kato, 1985; Morita, 1986; Togai and Yamano, 1985, 1986; Wang, 1984; Wang and Horowitz, 1985)

These papers discuss using linear learning operators and have emphasized stability of the proposed algorithms. There has been little work emphasizing performance, i.e. the convergence rate of the algorithm. Simulations of several of these algorithms have revealed very slow convergence and large sensitivity to disturbances and sensor and actuator noise.

6.2.2 Features Of This Work

What distinguishes this trajectory learning algorithm from previous robot trajectory learning schemes is the following combination:

Provides guidance for designing the learning operator: The central problem in making use of trajectory error measurements is transforming those errors into feedforward command corrections. Previous work has explored the requirements on the learning operator for convergence, but has given little guidance as to how one should choose a particular learning operator from the large set of learning operators that converge. The algorithm proposed here explicitly uses an inverse plant model as the learning operator. If there are no disturbances or sensor or actuator noise and we have perfect models of the robot, the algorithm will correct any errors in the feedforward command after one practice motion.

Handles nonlinear plants: The algorithm can make use of the nonlinear rigid body robot dynamics in the model used to generate feedforward command corrections. Previous work have used linearized or otherwise simplified robot models.

Makes explicit use of feedback control: The algorithm takes explicit advantage of the on-line trajectory improvement provided by the feedback controller in calculating the next feedforward command.

Successful implementation: The algorithm has been implemented on an actual robot, the MIT Serial Link Direct Drive Arm. Excellent trajectory learning performance requiring only a small number of practice trials has been achieved.

Generality: works with many feedback controllers: The algorithm applies to a wide variety of feedback controller structures, as only knowledge of the feedback controller output is required. It would be easy to combine this adaptive feedforward control algorithm with adaptive feedback controllers.

Generality: works with many plants: The algorithm does not require the plant dynamics to be of a particular type and applies to a wide range of robots

with (for example): actuator dynamics, joint compliance dynamics, and flexible link dynamics. This is an important feature of the algorithm, as it is becoming clear that a typical robot exhibits quite complex dynamics (Sweet and Good, 1985).

Analysis relevant to other schemes: The analysis of this algorithm makes clear why other trajectory improvement schemes that intuitively seem correct often perform badly or actually degrade performance in practice.

6.3 The Trajectory Learning Algorithm

6.3.1 The Control Problem

Our goal is to repetitively execute an unrestrained robot trajectory with as small a trajectory following error as possible. The desired trajectory has a finite time duration and the robot starts in a known initial steady state, $\mathbf{x}(0)$. We assume that modelling errors are much larger than any sensor and actuator noise or plant disturbances, and save the question of how to appropriately filter out non-repeatable noise and disturbances for a later paper.

There are three components of the trajectory learning algorithm: feedforward command initialization, movement execution, and feedforward command modification (Figure 6.1). After initializing the feedforward command, the movement execution and feedforward command modification steps are executed for each repeated movement attempt.

6.3.2 Feedforward Command Initialization

Given the desired state trajectory specification, $\mathbf{x}_d(t)$, the feedforward command memory is initialized using a model of the inverse of the robot dynamics (Figure 6.1A):

$$\mathbf{u}_{ff}^0(t) = \hat{R}^{-1}(\mathbf{x}_d(t), \dot{\mathbf{x}}_d(t)) \tag{6.1}$$

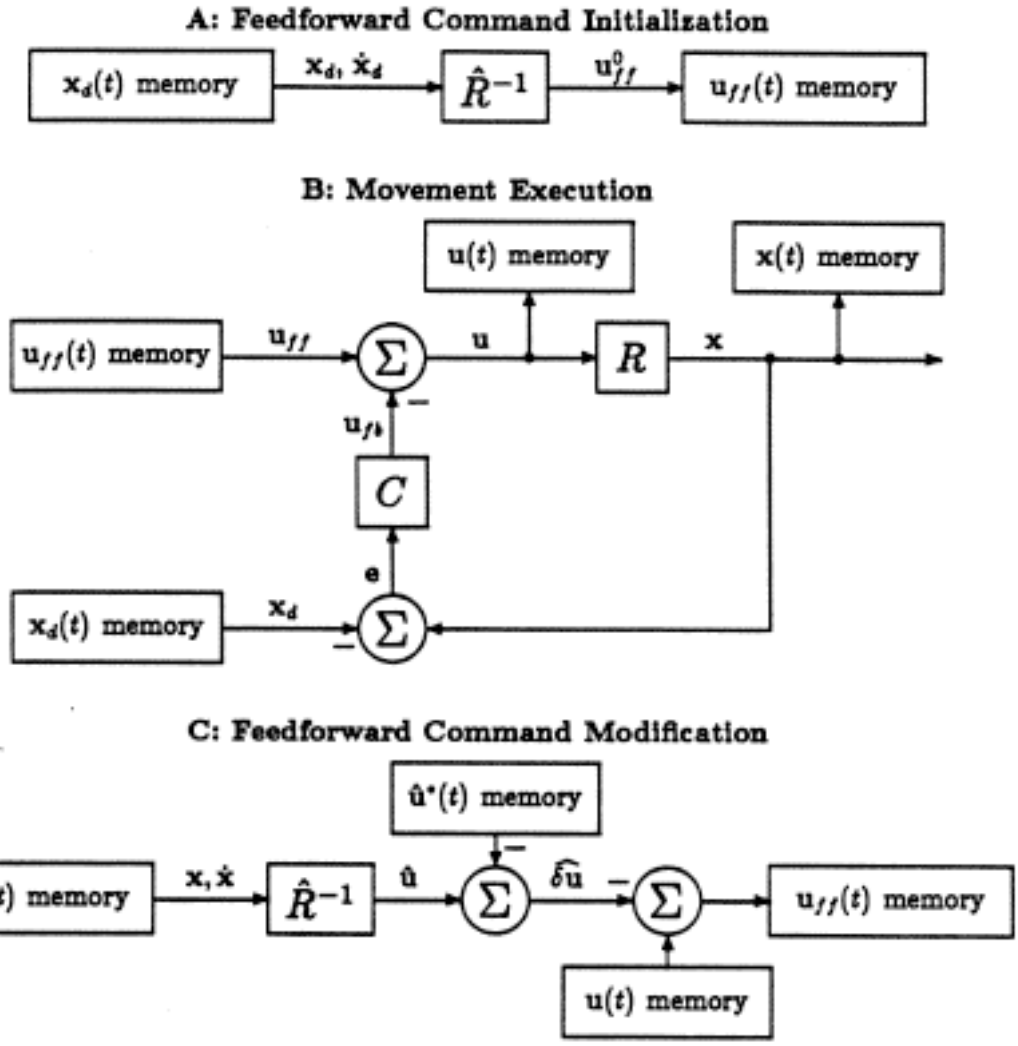


Figure 6.1: Block diagrams for feedforward command initialization, execution, and modification.

Using inverse models of robots to generate feedforward control signals: The robot itself transforms a vector of actuator commands, $\mathbf{u}$, into motion, $\mathbf{x}$

$$\dot{\mathbf{x}} = R(\mathbf{x}, \mathbf{u}) \tag{6.2}$$

$\mathbf{x}$ represents the state vector of the robot, which includes the position and velocity of each joint. $\dot{\mathbf{x}}$ is the derivative of $\mathbf{x}$. A model of the inverse of the robot dynamics transforms a trajectory specification, $\mathbf{x}(t)$, into actuator commands necessary to achieve the desired motion:

$$\mathbf{u} = \hat{R}^{-1}(\mathbf{x}, \dot{\mathbf{x}}) \tag{6.3}$$

For robot arms a rigid body dynamics model is often used to predict the forces and torques necessary to achieve a particular motion, and thus serves as a model of the inverted robot dynamics. The rigid body dynamics equations for a robot can be written as

$$\hat{R}^{-1}(\mathbf{x}, \dot{\mathbf{x}}) = Torques = \mathbf{I}(\boldsymbol{\theta}) \cdot \ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{\theta}} \cdot \mathbf{C}(\boldsymbol{\theta}) \cdot \dot{\boldsymbol{\theta}} + \mathbf{g}(\boldsymbol{\theta}) \tag{6.4}$$

where $\boldsymbol{\theta}(t)$ is the desired trajectory of the joint angles, $\mathbf{I}(\boldsymbol{\theta})$ is the inertia matrix of the arm, $\mathbf{C}(\boldsymbol{\theta})$ is the Coriolis and centripetal force tensor, and $\mathbf{g}(\boldsymbol{\theta})$ is the gravitational force vector (Hollerbach, 1984). For some robots it is argued that additional sources of dynamics are important (Goor, 1985a; Sweet and Good, 1985; Good, Sweet, and Stroebel, 1985). In these cases we can still model the robot dynamics and invert the model. For the purposes of this chapter we will use only rigid body robot models, as these types of models describe most of the dynamics seen in a direct drive robot arm.

Generating the robot model: In order to use a rigid body robot model the inertial parameters of the robot must be known. There are several sources of parameter estimates: 1) CAD/CAM models may be used to compute inertial parameter estimates, 2) the robot can be disassembled and the inertial parameters of the parts

measured, or 3) the inertial parameters can be estimated using movement and actuator data from actual use of the robot. In previous work we have taken the parameter estimation approach and shown that the dynamics are linear in the unknown inertial parameters of the links. We have identified the rigid body dynamics of the MIT Serial Link Direct Drive Robot Arm, and have used this dynamic model for control (An, Atkeson, and Hollerbach, 1985).

6.3.3 Movement Execution:

The ith attempt at the desired movement is executed using the current feedforward command, $\mathbf{u}^i_{ff}(t)$ (Figure 6.1B). This command includes an error, $\delta\mathbf{u}^i_{ff}(t)$, which is reduced by the actions of a suitably designed feedback controller, C, leaving an error, $\delta\mathbf{u}^i(t)$, in the actuator commands sent to the robot, $\mathbf{u}^i(t)$. For a linear plant and controller the amount of error reduction can be expressed using Laplace transforms:

$$\mathbf{u}^i(s) = \frac{1}{1 + C(s)R(s)}\mathbf{u}^i_{ff}(s) \tag{6.5}$$

The actuator commands and the actually executed movement trajectory, $\mathbf{x}^i(t)$, are stored for use by the learning module.

6.3.4 Feedforward Command Modification

The previous steps in this algorithm are standard components of many current robot controllers. One contribution of this work is to propose a particular approach to transforming trajectory errors into modifications of the feedforward command. We represent errors in our dynamic models of the robot as an input command disturbance, which we can correct for by modifying the feedforward command. We use our models of the inverted robot dynamics to estimate this actuator command error, $\widehat{\delta\mathbf{u}}^i(t)$, which is then subtracted from the previously applied actuator commands to form the feedforward command for the next movement attempt (Figure 6.1C).

Estimating the actuator command error

We assume there exists a correct actuator command history, $\mathbf{u}^*(t)$, that drives the plant exactly along the desired trajectory in the absence of disturbances or noise, and define the current actuator command error as

$$\delta\mathbf{u}^i(t) = \mathbf{u}^i(t) - \mathbf{u}^*(t) = R^{-1}(\mathbf{x}^i(t), \dot{\mathbf{x}}^i(t)) - R^{-1}(\mathbf{x}_d^i(t), \dot{\mathbf{x}}_d^i(t)) \tag{6.6}$$

where $R^{-1}()$ is the true inverse robot dynamics. Our estimate of the error in the current actuator command is found by simply replacing the true inverse robot dynamics with our model:

$$\begin{aligned} \widehat{\delta\mathbf{u}}^i(t) &= \hat{R}^{-1}(\mathbf{x}^i(t), \dot{\mathbf{x}}^i(t)) - \hat{R}^{-1}(\mathbf{x}_d^i(t), \dot{\mathbf{x}}_d^i(t)) \\ &= \hat{R}^{-1}(\mathbf{x}^i(t), \dot{\mathbf{x}}^i(t)) - \hat{\mathbf{u}}^*(t) \end{aligned} \tag{6.7}$$

When the plant is linear we can apply the plant inverse directly to the trajectory error.

$$\hat{R}^{-1}(\mathbf{x}, \dot{\mathbf{x}}) - \hat{R}^{-1}(\mathbf{x}_d, \dot{\mathbf{x}}_d) = \hat{R}^{-1}(\mathbf{e}, \dot{\mathbf{e}}) \tag{6.8}$$

The command error estimate can be expressed in terms of the feedforward command error using Laplace transforms.

$$\mathbf{e}^i(s) = \frac{R(s)}{1 + C(s)R(s)}\delta\mathbf{u}_{ff}^i(s) \tag{6.9}$$

$$\widehat{\delta\mathbf{u}}^i(s) = \frac{1}{\hat{R}(s)}\mathbf{e}^i(s) = \frac{1}{\hat{R}(s)}\frac{R(s)}{1 + C(s)R(s)}\delta\mathbf{u}_{ff}^i(s) \tag{6.10}$$

Updating the feedforward command

The update for the feedforward command on the next movement is simply the modified actuator command.

$$\mathbf{u}_{ff}^{i+1}(t) = \mathbf{u}^i(t) - \widehat{\delta\mathbf{u}}^i(t) \tag{6.11}$$

By subtracting the correct command, $\mathbf{u}^*$, from both sides of Equation (6.11) and using Equations (6.5) and (6.10) we can express the Laplace transformed error propagation equation for the linear case:

$$\delta \mathbf{u}_{ff}^{i+1}(s) = \frac{\hat{R}(s) - R(s)}{\hat{R}(s)(1 + C(s)R(s))} \delta \mathbf{u}_{ff}^{i}(s) \tag{6.12}$$

With a perfect model, $\hat{R} = R$, the feedforward command errors will be zero after one movement.

6.4 An Implementation Of The Algorithm

We have implemented the trajectory learning algorithm on the MIT Serial Link Direct Drive Arm. This three joint arm is described in (An, Atkeson, and Hollerbach, 1985). To explore the effectiveness of our learning algorithm we will present results on learning a particular trajectory.

Discrete vs. Continuous Time: Since we are using a computer to generate the control signals, all signals and controlled systems were represented in discrete time rather than continuous time.

The Test Trajectory: All three joints of the Direct Drive Arm were commanded to follow a fifth order polynomial trajectory with zero initial and final velocities and accelerations and a 1.5 second duration. Figure 6.2 shows the shape of the trajectory for each joint, and Table 6.1 gives the initial and final joint positions, the peak joint velocities, and the peak joint accelerations.

The Feedback Controller: An independent digital feedback controller was implemented for each joint and was not modified during learning.

Initialization Of The Feedforward Command: The initial feedforward torques were generated from a rigid body dynamics model. The model and the estimation of its parameters are described in (An, Atkeson, and Hollerbach, 1985). The calculated feedforward torques are shown in Figure 6.3A.

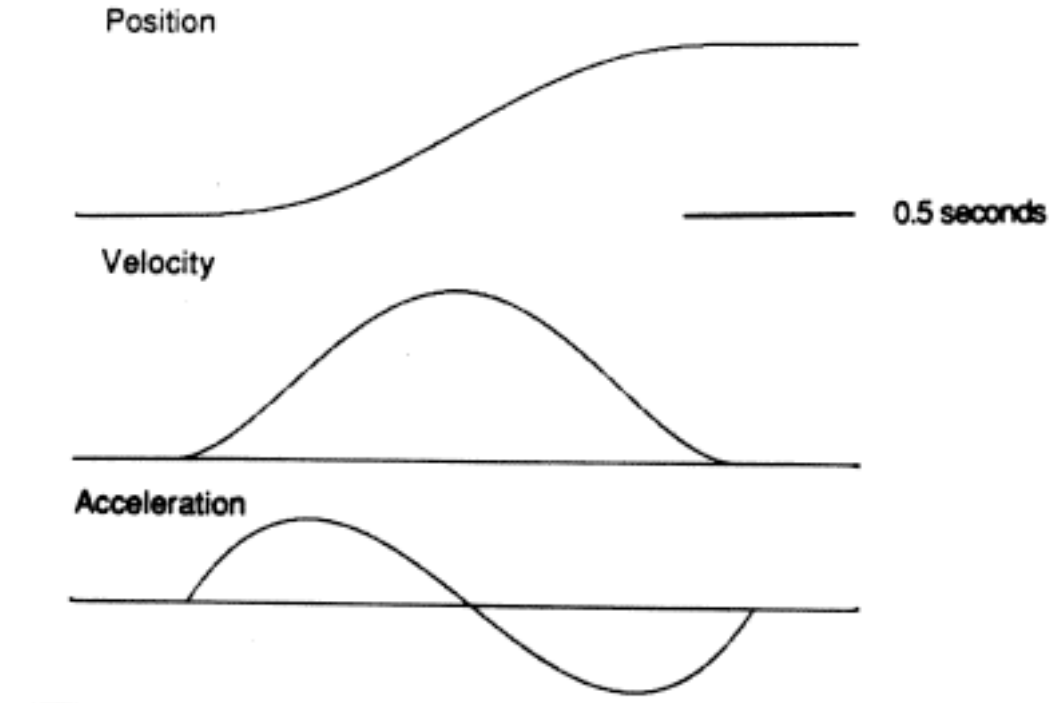


Figure 6.2: Desired trajectory.

Joint	Initial Position *radians*	Final Position *radians*	Peak Velocity *radians/s*	Peak Acceleration *radians/s²*
1	0.5	4.5	5.0	±10.3
2	5.0	1.0	-5.0	±10.3
3	4.0	-0.5	-5.6	±11.5

Table 6.1: Test trajectory parameters.

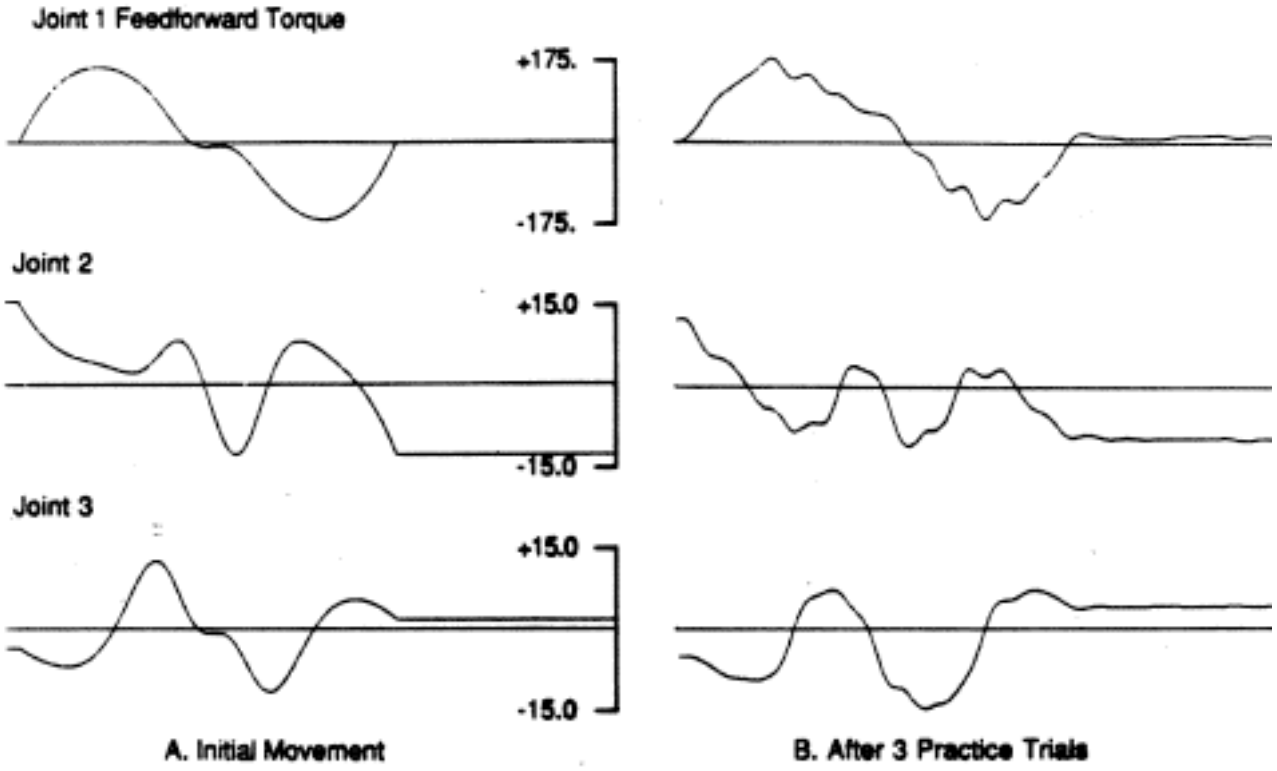


Figure 6.3: Feedforward Torques. (*newton – meters*)

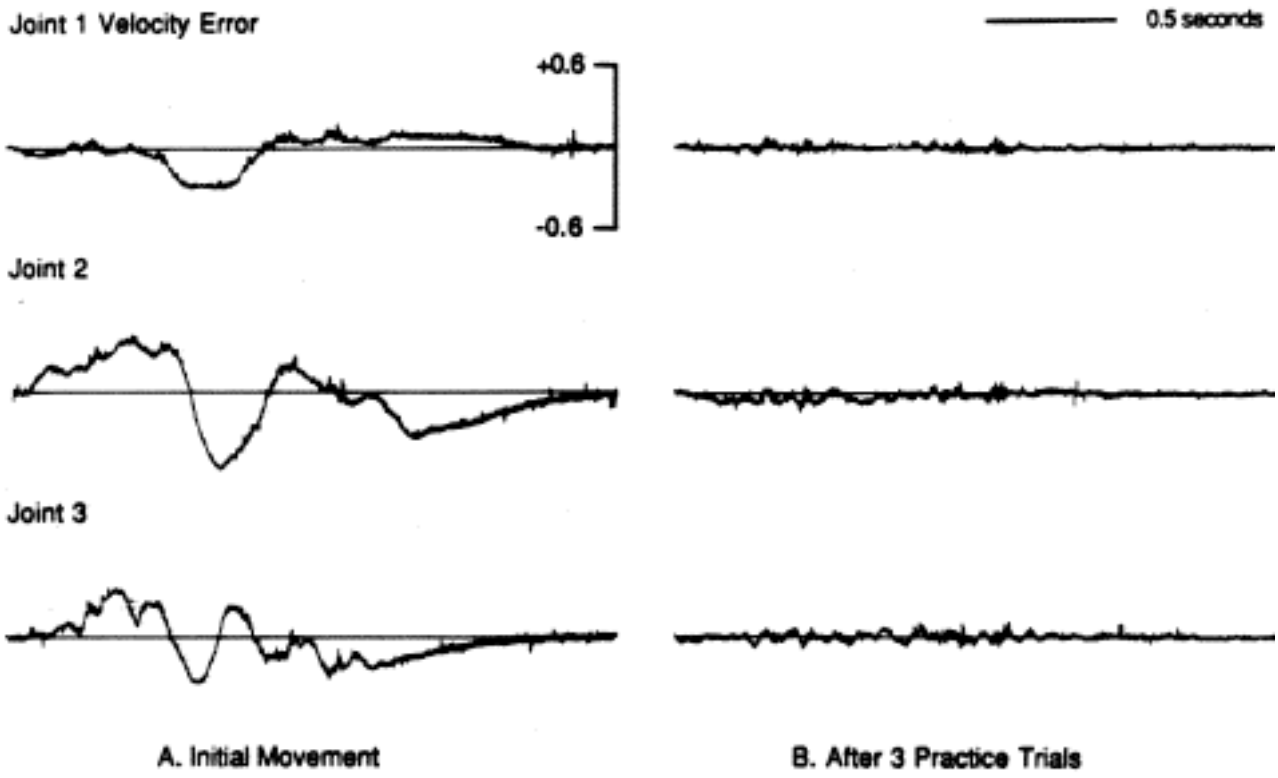


Figure 6.4: Velocity Errors. (*radians/second*)

Initial Trajectory Performance: As an index of trajectory following performance the velocity errors (the difference between the actual joint velocity and the desired joint velocity) for the first movement are shown in Figure 6.4A. We have plotted the raw velocity error data to give an idea of the relative size of the trajectory errors and sensor noise.

Calculating Acceleration and Filtering: In order to use the rigid body inverse dynamics model to compute joint torques it was necessary to compute the joint accelerations. Joint positions and velocities were measured directly. A digital differentiating filter combined with an 8Hz. low pass filter was applied to the velocity data to estimate accelerations.

To reject noise and non-repeatable disturbances it is necessary to filter the trajectory errors and controller output to improve the convergence of the learning process. In this implementation we applied low pass digital filters with an 8Hz. cutoff to the data used in the learning process. We filtered the references used by the learning operator with the same filter used on the data. It was necessary to correct for inconsistencies between the velocity sensors and the position measurements, which was done by calibrating the position reference to the feedback controller.

Final Trajectory Performance: The robot executed two additional training movements which are not shown, and its performance on the fourth attempt of the test trajectory was assessed. Figure 6.3B shows the modified feedforward commands used on the fourth movement, and should be compared with the predicted torques shown in Figure 6.3A. Figure 6.4B shows the velocity errors for the fourth movement, and should be compared with the initial movement velocity errors in Figure 6.4A. There has been a substantial reduction in trajectory following error after only three practice movements.

6.5 Using Simplified Models

It may seem unnecessary to use the full rigid body dynamics model of the robot in the learning algorithm. One might think that learning algorithms of this type allow one to avoid modeling the robot in full detail. This is not the case. Simplifying the robot model necessarily introduces additional modeling error. Without careful analysis such modeling errors may cause the learning algorithm to have poor performance, or even to degrade performance. As an illustration of the possible effects of modeling error due to the use of simplified models, we will now present a seemingly reasonable simplified model of a two joint robot arm that when used as the learning operator fails to learn. The robot arm is a planar two link mechanism with rotary joints and is described in the introduction to (Brady, et al., 1982.)

The simplified dynamic model that we have chosen for this example is that of two independent rotary joints with constant moments of inertia. That is, we ignore the centripetal and coriolis torques of the complete rigid body robot dynamics, and we assume constant moments of inertia around each joint. The moment of inertia of link 2 is a constant with respect to θ_2. The moment of inertia around joint 1 depends on θ_2. We approximate the moment around joint 1 as the average of the maximum and minimum moments around that joint, over all possible configurations of the robot. The equations of motion for such a simplified system are:

$$Torques = \mathbf{I} \cdot \ddot{\boldsymbol{\theta}} \tag{6.13}$$

where $\mathbf{I}$ is a constant diagonal 2×2 matrix. This gives a learning operation of:

$$\mathbf{u}_{ff}^{i+1} = \mathbf{u}^{i} - \mathbf{I}_{\mathbf{estimated}} \cdot (\ddot{\boldsymbol{\theta}} - \ddot{\boldsymbol{\theta}}_{d}) \tag{6.14}$$

The results of applying this learning algorithm to the simulated two joint robot movement are show in Figure 6.5. The movement is from point a to point b with zero initial and final velocity, acceleration, and jerk (seventh order polynomial). Feedback control is provided by independent PD controllers at each joint, each having

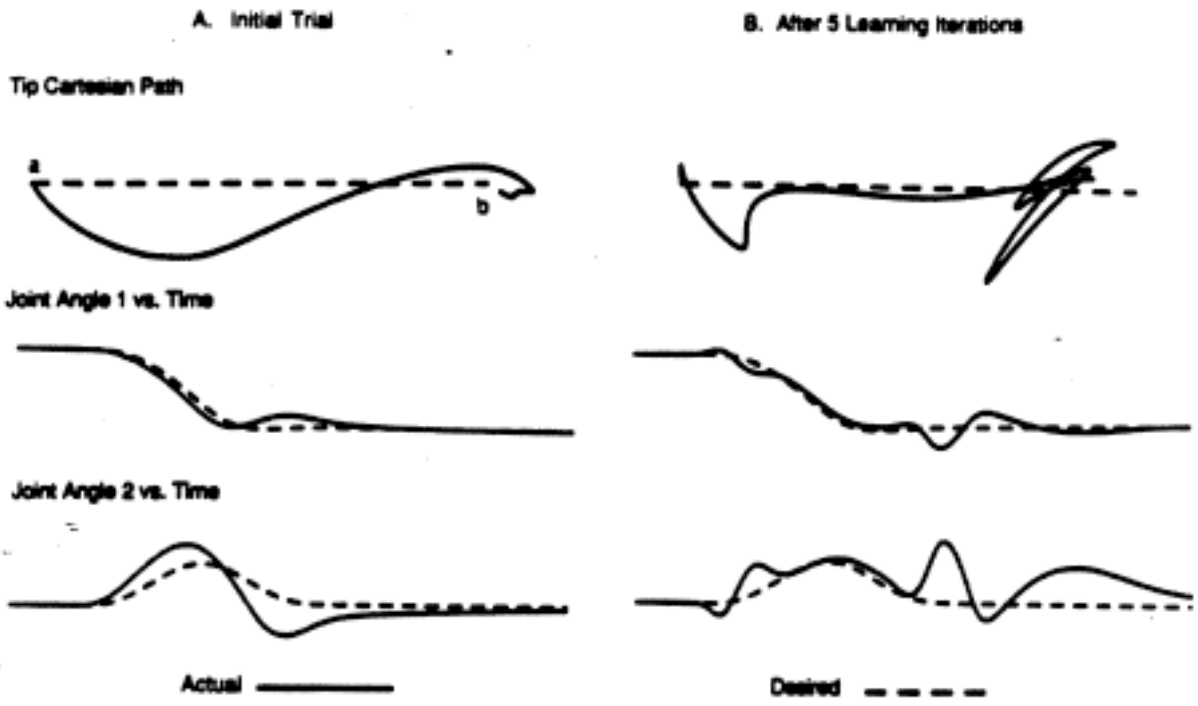


Figure 6.5: Simulated Two Joint Learning With Decoupled Model

a bandwidth of 1.0 hertz and damping coefficient of 0.707 (based on $\mathbf{I}_{\mathbf{estimated}}$). Figure 6.5A shows the performance of the system on the initial trial, with the initial feedforward torques, $\mathbf{u}_{ff}^{0}$, based on the simplified dynamics model. Figure 6.5B shows the performance of the system after five iterations of the learning algorithm. In this case, where an inaccurate inverse model of the robot has been used to update the feedforward torque command, no improvement in trajectory following performance is seen. Had the full inverse dynamics model been utilized (under these ideal conditions of no measurement noise, actuator noise, or external torque disturbances), our algorithm would have produced a perfect movement after one iteration.

Appropriate Robot Models: It has been argued that simplified models are appropriate for a robot with high gear ratios such as the PUMA. One must still model the other sources of dynamics prominent in these types of robots (Sweet and Good, 1985; Good, Sweet, and Stroebel, 1985). Higher order actuator dynamics may play an important role (see, for example, Goor (1985a,b)). Our point is not that the rigid body dynamics are the only appropriate model and must be used, but that we must be careful to include all significant dynamics in our models. Learning performance can be used to assess the quality of the models used to drive the learning.

6.6 Convergence

An important question that arises is how close our model of the inverted robot dynamics has to be to the true inverted robot dynamics for the proposed trajectory learning algorithm to converge to zero trajectory error. We will indicate how convergence can be tested in the general nonlinear case, but by specializing the convergence criteria to the linear case and presenting a numerical example we will show that a convergence proof does not guarantee acceptable performance.

6.6.1 Nonlinear Convergence Criteria

Analogies to solving simultaneous nonlinear equations: One way to view the proposed trajectory learning algorithm is as a procedure to solve the vector nonlinear equations:

$$\chi_d = \mathcal{R}(\mu_{ff}) \tag{6.15}$$

for μ_{ff}, where χ_d is the sampled version of the desired trajectory expressed as a vector $(\mathbf{x}_d[1], \mathbf{x}_d[2], \ldots, \mathbf{x}_d[T])^T$, μ_{ff} is a similar vector of the feedforward commands, T is the number of samples in the trajectory, and $\mathcal{R}$ is a nonlinear function representing the true robot and feedback controller dynamics. Since the initial state of the robot at the beginning of each movement is a known constant, $\mathcal{R}()$ does not require the robot initial state as an explicit argument.

Testing for convergence using fixed point theory: One approach to analyzing convergence of algorithms to solve nonlinear equations is to combine the original equations and the proposed algorithm into a single iteration function. Each cycle of movement repetition and feedforward command modification can be expressed as an iterative function,

$$\mu_{ff}^{i+1} = \mathcal{G}(\mu_{ff}^{i}) \tag{6.16}$$

and we can appeal to fixed point theory for convergence conditions for μ_{ff} (Isaacson and Keller, 1966). This is the approach suggested by (Wang, 1984; Wang and Horowitz, 1985).

6.6.2 Convergence Does Not Guarantee Good Performance.

Although conceptually elegant, the convergence conditions provided by fixed point theory are too weak to be useful. The reason for this is that the duration of the executed trajectory is finite and the controlled plant is causal. The trajectory duration

is referred to as the learning interval, as this is the time over which the feedforward command is modified. Convergence can be guaranteed by assuring that the feedforward command errors propagate forward in time on each cycle of movement execution and feedforward command modification until the command errors propagate out of the learning interval (Figure 6.6). Command and trajectory errors can grow exponentially as long as they are continually shifted forward in time, as we will show in the following example.

The example plant

Consider a rotary single degree of freedom robot. The dynamics of this robot in continuous time are

$$\tau = I\ddot{\theta} \tag{6.17}$$

where τ is the torque at the joint, I is the moment of inertia of the robot, and $\ddot{\theta}$ is the angular acceleration. We can express the dynamics of this robot in discrete time as

$$\theta_k - 2\theta_{k-1} + \theta_{k-2} = \frac{h^2}{2I}(\tau_k + \tau_{k-1}) \tag{6.18}$$

where h is the sampling period (Astrom and Wittenmark, 1984). Note that for later notational convenience in inverting the plant the sampled torques have been renumbered to remove the plant delay. A sampling frequency of 1kHz is used in the numerical simulations to follow.

We use a feedback controller

$$\tau_{fb_{k+1}} = -k(\theta_k - \theta_{d_k}) - b(\dot{\theta}_k - \dot{\theta}_{d_k}) \tag{6.19}$$

to give the second order plant a natural frequency of 10Hz and a damping ratio of 0.707. The angle and the angular velocity of the single joint are both measured directly.

Deriving the convergence criteria

Since the plant is linear the inverse plant model used for feedforward command modification will be linear, and it can therefore operate directly on the trajectory error, $e[t]$. The example plant is also minimum phase and the plant inverse will be causal. In order to put convergence bounds on the learning operator we will refer to a general learning operator L instead of requiring the learning operator to be the inverse of the robot dynamics, $\hat{R}^{-1}$. L will be restricted to be a causal linear operator.

The propagation of feedforward command errors from one movement to the next is given by subtracting the correct command, $\mathbf{u}^*$, from both sides of Equation (6.11):

$$\begin{aligned} \delta\mathbf{u}_{ff}^{i+1}[t] &= \delta\mathbf{u}^i[t] - \widehat{\delta\mathbf{u}}^i[t] \\ &= \delta\mathbf{u}^i[t] - l[t] * e[t] \end{aligned} \tag{6.20}$$

where $l[t]$ is the impulse response of the learning operator L and $*$ is the convolution operator. Since the learning interval (the duration over which we modify a particular feedforward command) is finite and includes only T samples, we can express Equation (6.20) as a matrix equation by replacing convolutions with matrix operations.

The sequence of sampled command errors and feedforward command errors are expressed as vectors, and the transformation between them can be expressed as a matrix equation $\delta\mathbf{u}^i = \mathbf{A}\delta\mathbf{u}_{ff}^i$:

$$\begin{pmatrix} \delta\mathbf{u}^i[0] \\ \delta\mathbf{u}^i[1] \\ \vdots \\ \delta\mathbf{u}^i[T] \end{pmatrix} = \begin{pmatrix} a[0] & 0 & \cdots & 0 \\ a[1] & a[0] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a[T] & a[T-1] & \cdots & a[0] \end{pmatrix} \begin{pmatrix} \delta\mathbf{u}_{ff}^i[0] \\ \delta\mathbf{u}_{ff}^i[1] \\ \vdots \\ \delta\mathbf{u}_{ff}^i[T] \end{pmatrix} \tag{6.21}$$

where $\mathbf{A}$ is a lower triangular Toeplitz matrix. The elements $a[t]$ are the impulse response of the transfer function relating feedforward command errors to actuator

command errors, $1/(1 + C[z]R[z])$, and we note that $a[0]$ must always be 1 due to a minimum loop delay of one sample before the feedback command can compensate for feedforward command errors.

The trajectory errors are generated by feedforward command errors according to the matrix equation $\mathbf{e}^i = \mathbf{B}\delta\mathbf{u}^i_{ff}$:

$$\begin{pmatrix} \mathbf{e}^i[0] \\ \mathbf{e}^i[1] \\ \vdots \\ \mathbf{e}^i[T] \end{pmatrix} = \begin{pmatrix} b[0] & 0 & \cdots & 0 \\ b[1] & b[0] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b[T] & b[T-1] & \cdots & b[0] \end{pmatrix} \begin{pmatrix} \delta\mathbf{u}^i_{ff}[0] \\ \delta\mathbf{u}^i_{ff}[1] \\ \vdots \\ \delta\mathbf{u}^i_{ff}[T] \end{pmatrix} \tag{6.22}$$

where $b[t]$ is the impulse response corresponding to the transfer function $R[z]/(1 + C[z]R[z])$.

The effect of the learning operator can be expressed as a lower triangular Toeplitz matrix, $\mathbf{L}$, using the matrix equation $\widehat{\delta\mathbf{u}}^i = \mathbf{L}\mathbf{e}^i$:

$$\begin{pmatrix} \widehat{\delta\mathbf{u}}^i[0] \\ \widehat{\delta\mathbf{u}}^i[1] \\ \vdots \\ \widehat{\delta\mathbf{u}}^i[T] \end{pmatrix} = \begin{pmatrix} l[0] & 0 & \cdots & 0 \\ l[1] & l[0] & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l[T] & l[T-1] & \cdots & l[0] \end{pmatrix} \begin{pmatrix} \mathbf{e}^i[0] \\ \mathbf{e}^i[1] \\ \vdots \\ \mathbf{e}^i[T] \end{pmatrix} \tag{6.23}$$

Combining the previous matrix equations, Equation 6.20 can be written as the matrix equation:

$$\delta\mathbf{u}^{i+1}_{ff} = (\mathbf{A} - \mathbf{LB})\delta\mathbf{u}^i_{ff} \tag{6.24}$$

Testing for convergence of the feedforward command error to zero reduces to guaranteeing that the absolute values of all the eigenvalues of the matrix $(\mathbf{A} - \mathbf{LB})$ are less than 1. Since $\mathbf{A}$, $\mathbf{L}$, and $\mathbf{B}$ are all lower triangular Toeplitz matrices, the eigenvalues are all equal to the diagonal element of $(\mathbf{A}-\mathbf{LB})$, $a[0]-l[0]b[0]$. We recall that $a[0] = 1$, and note that the test for convergence in the finite learning interval case reduces to a test involving only the first element of the learning operator impulse

response and the controlled plant impulse response:

$$|1 - l[0]b[0]| < 1 \tag{6.25}$$

This convergence test does not require L to be a very accurate inverse plant model.

A numerical example

To demonstrate that the above convergence condition does not guarantee acceptable performance let us example a particular learning operator which satisfies the convergence test but generates bad performance. Let us choose a learning operator that satisfies the finite learning interval convergence test exactly such that

$$1 - l[0]b[0] = 0 < 1 \tag{6.26}$$

Such a learning operator is given by

$$l[t] = \begin{cases} \frac{1}{b[0]} & \text{if } t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.27}$$

With this learning operator the eigenvalues of $(\mathbf{A} - \mathbf{LB})$ are all zero, and we are guaranteed exact convergence in at most T movements, where T is the number of samples in the learning interval. This learning operator corresponds to correcting the feedforward command with a scaled version of the position error history.

To examine the transient response of this learning operator we resort to numerical simulation. We initialize the feedforward command to have a single error on the first sample. The resulting position error is shown on the first graph of Figure 6.6. Several of the following movements are also shown, and by the 10th movement the position error has increased by a factor of 10^{26}. It is difficult to see that with each movement the first non-zero position error of the previous movement is exactly cancelled, as the position errors later during the movement are blowing up exponentially. This exponentially growing wave of errors does shift forward in time,

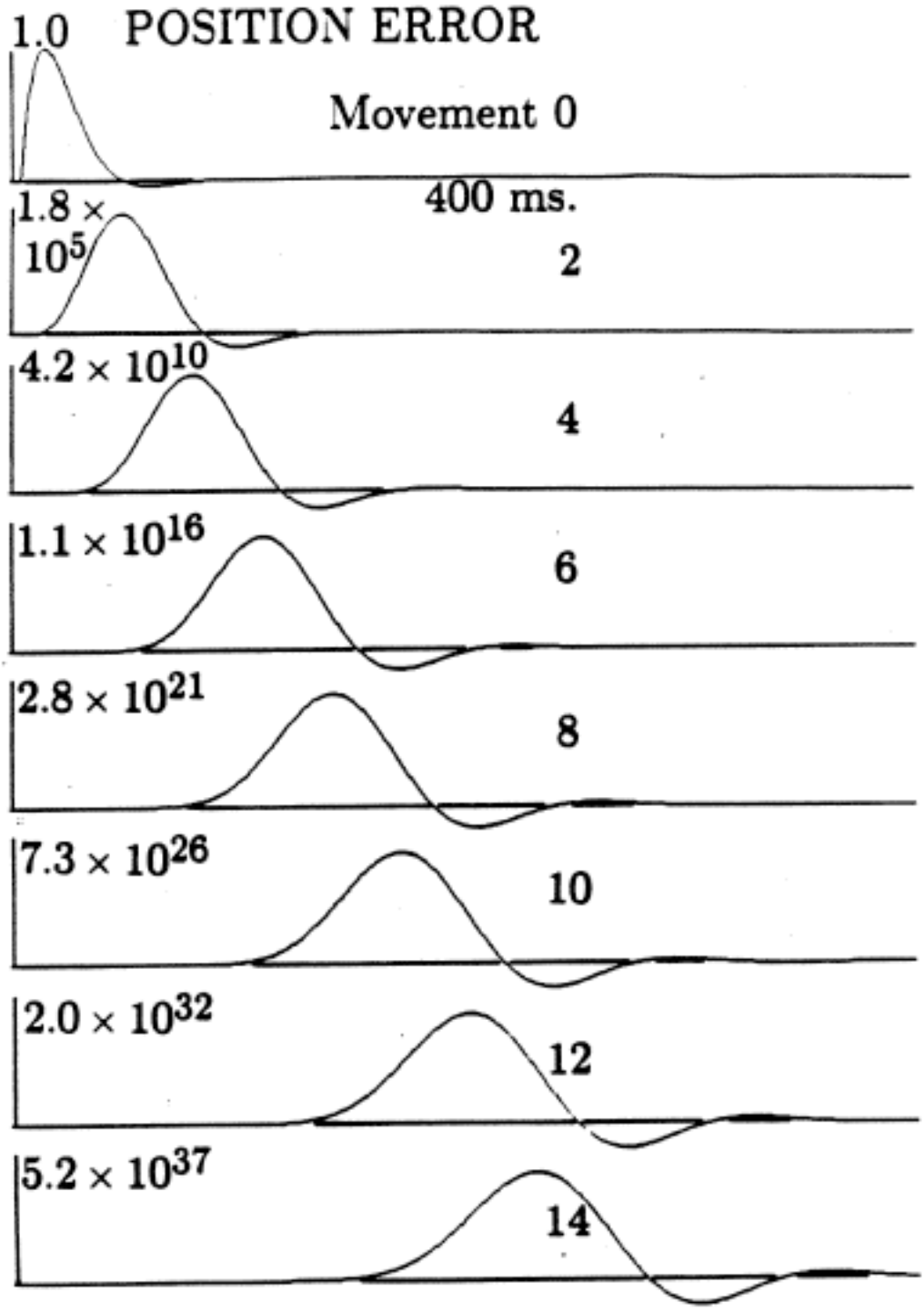


Figure 6.6: Simulated learning performance for a bad plant inverse model

and after T movements will have shifted completely outside the learning interval. Had the correct inverse dynamics model been used as the learning operator (under these ideal conditions of no measurement noise, actuator noise, or disturbances) the feedforward command error would have been entirely cancelled after one movement.

This type of finite time interval convergence relies on the absence of any introduction of new errors. In the presence of sensor and actuator noise and plant disturbances the convergence promised by the finite time interval analysis will probably not occur. In addition, as the sampling interval length changes in the discretization of a continuous time plant, the convergence properties of this type of learning algorithm change. Convergence is improved when the plant is sampled at low frequencies.

A performance test

The previous example demonstrates the need to test performance in addition to showing convergence of a learning algorithm. In the nonlinear case performance must usually be checked by simulation and by actual implementation, but with linear plants and feedback controllers we can take advantage of superposition and the ability to transform to the frequency domain to check how errors at all frequencies are affected by the learning algorithm. From Equation 6.12 or 6.20 we see that the errors are propagated according to

$$\begin{aligned} \delta \mathbf{u}_{ff}^{i+1}[z] &= \frac{1-L[z]R[z]}{1+C[z]R[z]}\delta \mathbf{u}_{ff}^{i}[z] \\ &= \Lambda[z]\delta \mathbf{u}_{ff}^{i}[z] \end{aligned} \tag{6.28}$$

where the argument z indicates that we have shifted to the frequency domain using the Z transform. With a perfect inverse model, $(L = R^{-1})$, $\Lambda[z]$ is zero, implying convergence of the feedforward command error to zero after one practice movement. The learning algorithm will perform badly if at any frequency the gain of $\Lambda[z]$ is greater than 1. The gain of $\Lambda[z]$ as a function of frequency can be used to evaluate the efficacy of proposed learning operators. In the numerical example showing con-

vergence with poor performance, the poor performance was indicated by a maximum gain of $\Lambda[z]$ equal to 530. This is approximately the factor by which the maximum position error was growing on each movement repetition in the simulation.

6.7 Conclusions

The contributions of this work are:

- The derivation of a nonlinear trajectory learning algorithm based on explicit modelling of the plant.
- The demonstration of good performance of the learning algorithm by implementation on the MIT Serial Link Direct Drive Arm.
- The demonstration by a simple example that proofs of convergence do not guarantee acceptable performance and of the need to verify performance.

We conclude that more accurate robot models improve trajectory learning performance, and learning algorithms do not reduce the need for good models in robot control.

Chapter 7

Future Research

There are many components of motor learning, and the work in this thesis is only a small step in a research program in motor learning. Some of the future research which is very closely related to the research presented in the thesis is discussed in the previous chapters. Other topics are mentioned here.

7.1 Local models

The single trajectory learning work can be extended to learning a group of similar trajectories. This involves building a representation of the dynamics in a small region of the augmented state space (the state variables and their derivatives), which I refer to as a local model. The local model could be a representation of the linearized dynamics about one of the desired trajectories in the group, or it could be a tabular representation. This local modelling would be a correction model to the previously identified global model. Key issues here are storage of all the information necessary to represent the fine details of the local dynamics, and deciding whether to pool information from different tasks or store experience according to the type of task.

7.2 Task level learning

This thesis addressed only isolated learning modules. It is important to examine learning a complete task in addition to focussing on modules, as there will be many learning modules acting simultaneously in any realistic motor problem, and exploring their interaction is important. It is important to focus on other kinds of tasks where the effector system must respond to external demands rather than follow a given plan.

Often task performance can be represented by a small number of parameters, and the execution of an arbitrary trajectory is not required. Reducing the degrees of freedom of possible motor performance greatly simplifies the learning problem, and reduces the amount of knowledge necessary to learn from practice.

7.3 Plan optimization

The work in this thesis has focussed on improving execution of a given plan. The perfect execution of a flawed plan will not lead to optimal performance. Plans can be improved on the basis of models derived from experience.

7.4 Object perception

Rigid body load identification is a first step towards general object perception on the basis of force/torque information. There are many types of loads that are not well modelled by the rigid body dynamics framework, and alternative approaches to identification and control of these loads should be explored.

7.5 Human motor learning psychophysics

This work suggests a program of research on the quantitative psychophysics of motor learning. My previous research, not discussed in this thesis, on biological movement kinematics, dynamics, and control provides a strong base for planned studies on biological motor learning. Findings relevant to motor learning include an invariance of the tangential velocity profile across many different movement conditions, separability and scaling of load dynamics, and scaling of control gains with movement speed. In the immediate future my biological research will focus on two areas:

A Competence model: My own and other theoretical work have demonstrated possible roles of internal dynamic models in control and learning, and provided mathematical bounds on the required accuracy of these models. The goal of this research is to estimate, using simulations, how good human internal models actually are. However, in order for these estimates to be meaningful we need to have accurate estimates of the mechanical properties of the arm and the effective feedback gains during movement. I have begun this research by examining elbow movement, and discovered both modulation of effective feedback gains during movement and scaling of effective feedback gains for movements made at different speeds. These measurements need to be extended to multi-joint movement, and then useful simulations can be undertaken.

Motor learning psychophysics: Theoretical analysis and implementation on robots of different trajectory learning algorithms has revealed distinctive patterns of trajectory modification for alternative algorithms. In order to test for the possible use of any of these algorithms by humans we need to quantitatively measure patterns of trajectory modification during learning of particular trajectories.

7.6 Concluding note

This thesis provides a particular view on what intelligence is: identifying models of both the environment and the organism itself, and using the identified models to predict, plan, and control action. Attempting to build intelligent machines that achieve or surpass human levels of performance tests our understanding of these ideas, and how complete this view of intelligence really is.

Appendix A: Using The Identified Arm Model For Control

Experimental Evaluation of Feedforward and Computed Torque Control

C.H. An, C.G. Atkeson, J.D. Griffiths, and J.M. Hollerbach

MIT Artifical Intelligence Laboratory
Cambridge, Mass. 02139 USA

Abstract. Trajectory tracking errors resulting from the application of various controllers have been experimentally determined on the MIT Serial Link Direct Drive Arm. The controllers range from simple analog PD control applied independently at each joint to feedforward and computed torque methods incorporating full dynamics. It was found that trajectory tracking errors decreased as more dynamic compensation terms were incorporated. There was no significant difference in trajectory tracking performace between the feedforward controller using independent digital servos and the full computed torque controller.

1 Introduction

Despite voluminous publications on the control of robot arms, there have been few experimental evaluations of the performance of proposed controllers. A major reason for this is the lack of a suitable manipulator for testing that fits these publications' modeling assumptions. Commercial robots are characterized by high gear ratios, substantial joint friction, and slow movement. As a result, their dynamics are approximated well by single-joint dynamics (Goor, 1985; Good, Sweet, and Strobel, 1985). Moreover, hardly any commercial robots allow the control of joint torque, which is required in many of the proposed controllers.

Direct drive arms are increasingly overcoming some of the performance limitations of highly geared robots (Asada and Youcef-Toumi, 1984; Curran and Mayer, 1985; Kuwahara et al., 1985). The manipulator dynamics are made more ideal by the reduction of joint friction and backlash effects, and the control of joint torques becomes more feasible. Hence direct drive arms have the potential for serving as good experimental devices for testing advanced arm control strategies. When gearing is eliminated, however, the full nonlinear dynamic interactions between moving links are manifested.

This paper reports on two sets of experiments with the MIT Serial Link Direct Drive Arm (SLDDA) involving a subset of proposed control strategies. The first set of experiments is based on a hybrid control system. There is an independent analog servo for each joint with the position and velocity references, and feedforward commands generated by a microprocessor. Since most commercial arms are controlled by simple independent PID controller for each joint, an independent PD controller was tested on this arm to provide a baseline for comparison. The PD controller was augmented by feeding forward first gravity compensation and then the complete rigid body dynamics to ascertain any trajectory following improvements attained by taking the nonlinear dynamics more fully into account. The second set of experiments shows the preliminary results of digital servo implementation, using one Motorola 68000 based microproces-

sor to control all the joints of the SLDDA. The on-line computed torque approach is compared to the PD and to the feedforward approaches using the digital servo.

The accuracy of the manipulator dynamic model impinges on the performance of feedforward and computed torque control. Since friction is negligible for direct drive arms, and presuming that one has good control of joint torques, the issue of accuracy reduces to how well the inertial parameters of the rigid links are known. In our previous work, we developed an algorithm for estimating these inertial parameters for any multi-link robot as a result of movement, and applied it to the SLDDA (An, Atkeson, and Hollerbach, 1985). The present paper presents results of utilizing the estimated model to control the robot by both off-line (feedforward) and on-line (computed torque) computation of the joint torques.

1.1 Control Algorithms

The full dynamics of an n degree-of-freedom manipulator are described by

$$\mathbf{n} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \mathbf{F}(\mathbf{q}, \dot{\mathbf{q}}) \tag{1}$$

where $\mathbf{n}$ is the vector of joint torques (for rotational joints), $\mathbf{q}$ is the vector of joint angles, $\mathbf{J}$ is the inertia matrix, $\mathbf{V}$ is the vector of coriolis and centripetal terms, $\mathbf{G}$ is the gravity vector, and $\mathbf{F}$ is a vector of friction terms. The simplest and most common form of robot control is independent joint PD control, described by

$$\mathbf{n} = \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}) \tag{2}$$

where $\dot{\mathbf{q}}_d$ and $\mathbf{q}_d$ are the desired joint velocities and positions, and $\mathbf{K}_p$ and $\mathbf{K}_v$ are $n \times n$ diagonal matrices of position and velocity gains.

Feedforward control augments the basic PD controller by providing a set of nominal torques $\mathbf{n}_{ff}$:

$$\mathbf{n}_{ff}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) = \hat{\mathbf{J}}(\mathbf{q}_d)\ddot{\mathbf{q}}_d + \hat{\mathbf{V}}(\mathbf{q}_d, \dot{\mathbf{q}}_d) + \hat{\mathbf{G}}(\mathbf{q}_d) + \hat{\mathbf{F}}(\mathbf{q}_d, \dot{\mathbf{q}}_d) \tag{3}$$

where the hat ($\hat{}$) refers to the modelled values. When this equation is combined with (2), the feedforward controller results:

$$\mathbf{n} = \mathbf{n}_{ff}(\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d) + \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}) \tag{4}$$

The feedforward term $\mathbf{n}_{ff}$ can be thought of as a set of nominal torques which linearize the dynamics (1) about the operating points $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$, and $\ddot{\mathbf{q}}_d$. Therefore, it is reasonable to add the linear feedback terms $\mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$ as the control for the linearized system. These feedforward terms can be computed off-line, since they are function of the parameters of the desired trajectory only.

On the other hand, the computed torque controller computes the dynamics on-line, using the sampled joint position and velocity data. The control equation is:

$$\mathbf{n}_{ct}(\mathbf{q}_d, \mathbf{q}, \dot{\mathbf{q}}_d, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_d) = \hat{\mathbf{J}}(\mathbf{q})\ddot{\mathbf{q}}^* + \hat{\mathbf{V}}(\mathbf{q}, \dot{\mathbf{q}}) + \hat{\mathbf{G}}(\mathbf{q}) + \hat{\mathbf{F}}(\mathbf{q}, \dot{\mathbf{q}}) \tag{5}$$

where $\ddot{\mathbf{q}}^*$ is given by,

$$\ddot{\mathbf{q}}^* = \ddot{\mathbf{q}} + \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}). \quad (6)$$

If the robot model is exact, then each link of the robot is decoupled, and the trajectory error goes to zero. Gilbert and Ha (1984) have shown that the computed torque control method is robust to small modelling errors.

Previously, Liegeois, Fournier, and Aldon (1980) suggested feedforward control as an alternative to the on-line computation requirements of computed torque control, although they did not present any experimental results. Golla, Garg, and Hughes (1981) discussed different linear state-feedback controller using a linearized model of a manipulator. Asada, Kanade, and Takeyama (1983) presented some results of applying a feedforward control to the early version of a direct drive arm at the Robotics Institute of CMU, though for quite slow movements and for inertial parameters derived by CAD modeling. The computed torque method has been considered by several other investigators (Paul, 1972; Markiewicz, 1973; Bejczy, 1974; Luh, Walker, and Paul, 1980). Although simulation results have been presented, there has been no published report on the actual implementation of this controller, mainly due to the lack of an appropriate manipulator or on-line computational facility.

In this paper, we first use the feedforward controller to evaluate the accuracy of our estimates of the link inertial parameters, and to compare its performance against several other simpler control methods for high speed movements. Then, we present some preliminary results on the implementation of a computed torque controller, again using the estimated inertial parameters of the links.

1.2 Estimates of inertial parameters

The inertial parameters in the feedforward computation for the SLDDA were estimated by an algorithm developed previously (An, Atkeson, and Hollerbach, 1985). It was shown that the unknown inertial parameters of each link (mass, center of mass, and moments of inertia) appear linearly in the rigid body dynamics of a manipulator. Then a least squares algorithm was used to compute the estimates of these parameters.

The accuracy of the inertial parameters was verified initially by comparing the measured joint torques to the torques computed from the estimated parameters. This comparison, together with the torques computed from parameters derived by CAD modelling, is shown in Figure 1 for a $1.3s$ trajectory of all the joints moving 250 degrees. The results were actually superior for the dynamically estimated parameters than for the CAD-modelled parameters. A more practical verification of the estimated parameters is in generating feedforward torques as part of a control algorithm. The results of such experiments are presented in the next section.

2 Robot Controller Experiments

In this section, performances of several different controllers for full motion of the SLDDA are evaluated using the hybrid controller. The reference positions and velocities, and the

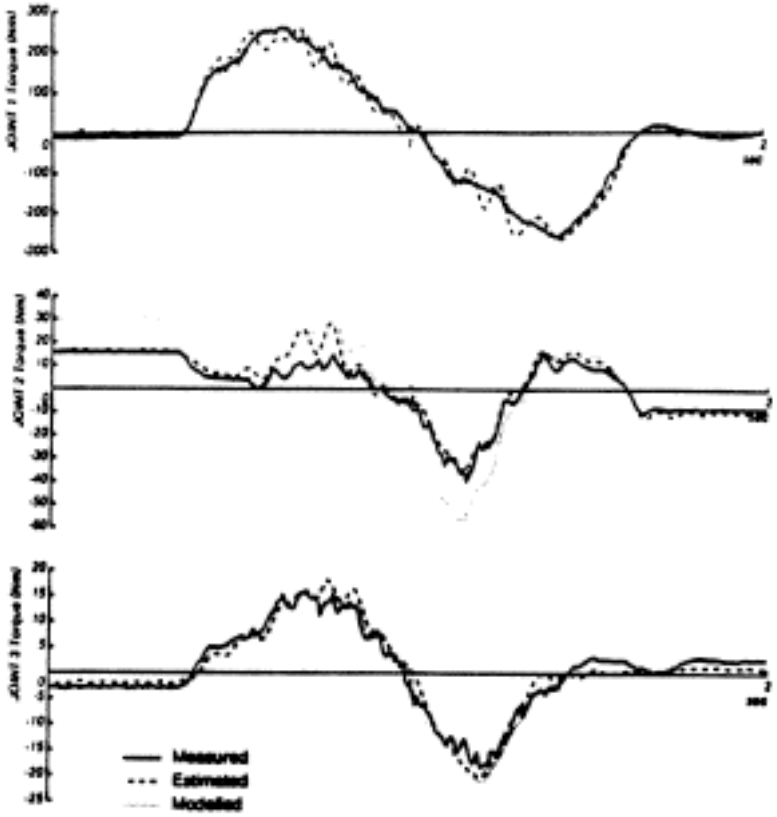


Figure 1: The measured, CAD-modelled, and estimated joint torques.

feedforward torques are generated by a microprocessor and input to three independent joint analog servos. We present evaluations of the following five control methods used for high speed movements of all three joints of the manipulator:

1. PD controller with position reference only:

$$\mathbf{n} = -\mathbf{K}_v\dot{\mathbf{q}} + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

2. PD controller with position reference and feedforward of gravity torques:

$$\mathbf{n} = \hat{\mathbf{G}}(\mathbf{q}_d) - \mathbf{K}_v\dot{\mathbf{q}} + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

3. PD controller with position and velocity references:

$$\mathbf{n} = \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

4. PD controller with position and velocity references plus feedforward of gravity torques:

$$\mathbf{n} = \hat{\mathbf{G}}(\mathbf{q}_d) + \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

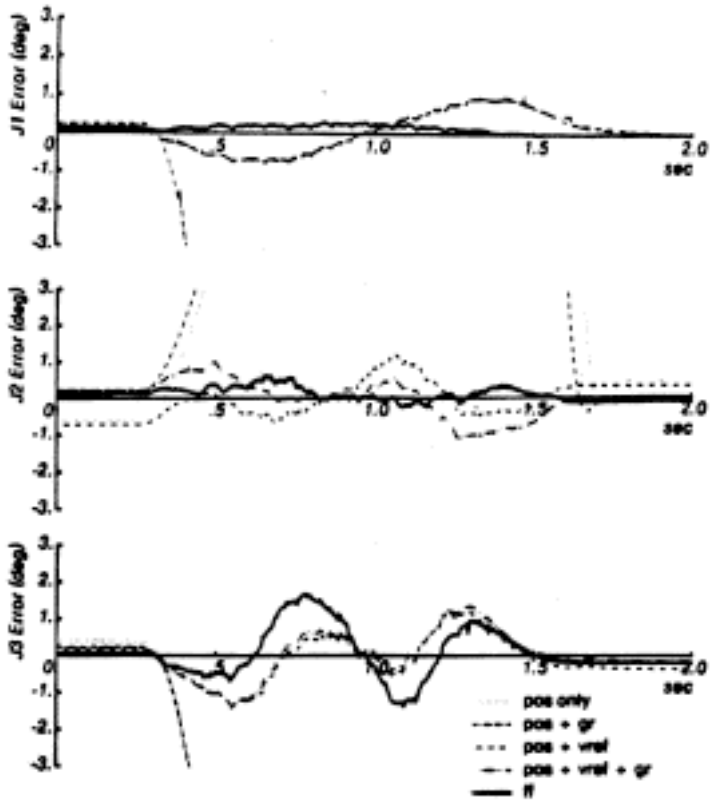


Figure 2: Trajectory errors of the 5 controllers for full 1.3s motion.

5. PD controller with position and velocity references plus feedforward of full dynamics:

$$\mathbf{n} = \hat{\mathbf{J}}(\mathbf{q}_d)\ddot{\mathbf{q}}_d + \hat{\mathbf{V}}(\mathbf{q}_d, \dot{\mathbf{q}}_d) + \hat{\mathbf{G}}(\mathbf{q}_d) + \mathbf{K}_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) + \mathbf{K}_p(\mathbf{q}_d - \mathbf{q})$$

In these experiments, friction was neglected. The nominal position and velocity gains were adjusted experimentally to achieve high stiffness and overdamped charateristics without the feedforward terms.

A fifth order polynomial in joint space was used to generate the reference trajectory. The joints moved from $(80°, 269.1°, -30°)$ to $(330°, 19.1°, 220°)$ in 1.3s, with peak velocities of 360 deg/sec and the peak accelerations of 854 deg/sec^2 for each joint. For control methods (2), (4), and (5), the estimates of the link inertial parameters given by An, Atkeson, and Hollerbach (1985) were used to compute the feedforward torques.

The trajectory errors for the above 5 controllers are shown on Figure 2. The errors for the first controller are very large and are out of the graph range. Adding a gravity feedforward term does not help very much, and the trajectory errors for Controller 2 are also very large. This was expected since gravity feedforward is a static correction to Controller 1, and the dynamic effects dominate the response for high speed movements. Modifying the first controller by adding a velocity reference signal improved the response greatly. As with Controller 2, adding a gravity feedforward term did not reduce the trajectory errors very much, and influenced mainly the steady state errors for joints 2 and 3.

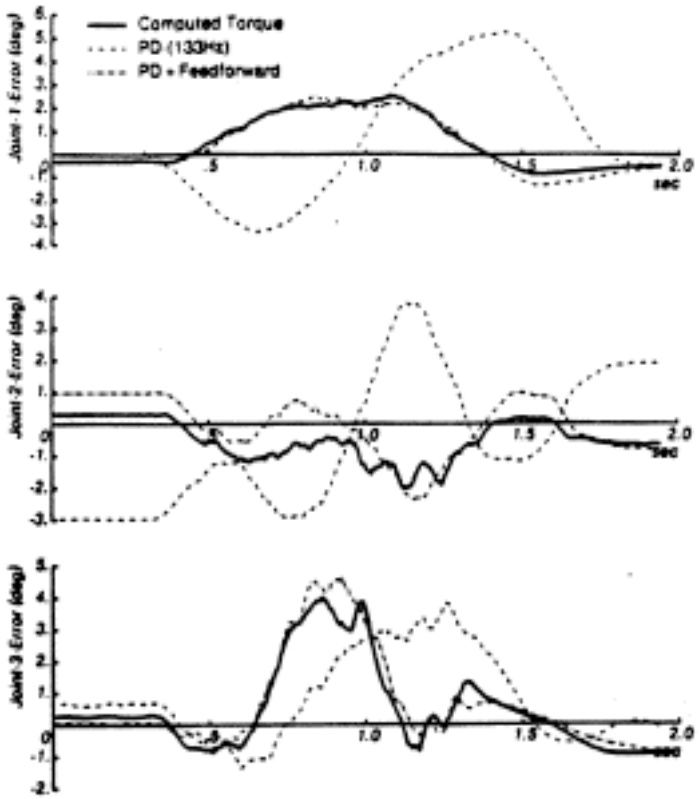


Figure 3: Trajectory errors of the three digital controllers for full 1.3s motion.

The full feedforward controller reduced the trajectory errors significantly for joints 1 and 2, with peak errors of only 0.33° and 0.64°, respectively. For joint 3, the feedforward torques did not help because of the light inertia and the dominance of unmodelled dynamics in the motor and in bearing friction. The high feedback gains make this joint somewhat robust to these unmodelled dynamics; yet, the trajectory errors could not be reduced below 1.4° with the feedforward torques based on the ideal rigid body model of the link.

2.1 Computed Torque Controller Experiment

In this section, some preliminary results are presented for the computed torque method implemented on the SLDDA. In this implementation, the analog servos are disabled, and the feedback computation is done digitally by one Motorola 68000 based microprocessor using scaled fixed-point arithmetic. Written in the C language, the controller, including the full computation of the robot dyamics, runs at a 133 Hz sampling frequency. Although further improvements in computation time are possible, this speed was adequate in demonstrating the efficacy of dynamic compensation. The details of this implementation are discussed in (Griffiths, 1986).

A similar fifth order polynomial trajectory as in the previous section was used for this experiment. Figure 3 shows the trajectory errors for three controllers: the digital PD controller, the feedforward controller using a digital servo, and the computed torque controller. The computed torque and the feedforward controllers both show a significant

reduction in tracking errors for joints 1 and 2 compared with the PD control, with no clear distinction between feedforward and computed torque. The tracking errors for joint 1 range from 4.4° to 2.2° and for joint 2 go from 3.5° to 2.0° with the addition of dynamic component. As before, the trajectory errors for joint 3 were not reduced by the computed torque or the feedforward controller. Again, this seems to indicate that our model for the third link may not be very good.

The trajectory tracking performance of the computed torque controller is not as good as that of the analog feedforward controller of the previous section. The main reason for this is the slow sampling frequency (133 Hz) of the digital controller, as compared to the 1 KHz sampling frequency at which the reference inputs were given to the analog servos. Improvements in the computation time should also improve the tracking performance of the computed torque controller.

3 Conclusions

We have presented experimental results of using an estimated dynamic model of the manipulator for dynamic compensation via feedforward and computed torque control methods. The results indicate that dynamic compensation can improve trajectory accuracy significantly and that the estimated rigid body model of the manipulator is quite accurate and adequate for control purposes for joints 1 and 2. The unmodelled dynamics of the light third link, including the motor dynamics and friction, are dominant and yield larger trajectory errors than at the other two joints. Therefore, for joint 3, it may be necessary to use a more complete model to improve trajectory following.

The results of the digital implementation of the feedforward and computed torque controllers were not as good as the hybrid feedforward controller. This indicates that if a robot was being used solely for free space movements without significant variation of its loads, then a hybrid controller using an independent analog servo for each joint may be quite adequate. A hybrid controller, however, is not flexible, and cannot handle varying loads or interactions with the environment. Future experiments with the MIT Serial Link Direct Drive Arm will concentrate on improving the computation time for the digital control system and on issues of force control.

Acknowledgments

This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Defense Advanced Research Projects Agency under Office of Naval Research contracts N00014-80-C-050 and N00014-82-K-0334 and by the Systems Development Foundation. Partial support for CGA was provided by a Whitaker Fund Graduate Fellowship and for JMH by an NSF Presidential Young Investigator Award.

References

An, C.H., Atkeson, C.G, and Hollerbach, J.M., 1985, "Estimation of inertial parameters of rigid body links of manipulators," *Proc. 24th Conf. Decision and Control*, Fort Lauderdale, Florida, Dec. 11-13.

Asada, H., Kanade, K., and Takeyama, I., 1983, "Control of a direct-drive arm," *Trans. of ASME*, 105, pp. 136-142.

Asada, H., Youcef-Toumi, K., and Lim, S.K., 1984, "Joint torque measurement of a direct-drive arm," *Proc. 23rd Conf. Decision and Control*, Las Vegas, Dec. 12-14, pp. 1332-1337.

Bejczy, A.K., Tarn, T.J., and Chen, Y.L., 1985, "Robot arm dynamic control by computer," *Proc. IEEE Conf. Robotics and Automation*, St. Louis, Mar. 25-28, pp. 960-970.

Curran, R., and Mayer, G., 1985, "The architecture of the AdeptOne direct-drive robot," *Proc. American Control Conf.*, Boston, June 19-21, pp. 716-721.

Gilbert, E.G., and Ha, I.J., 1984, "An Approach to Nonlinear Feedback Control with Applications to Robotics," *IEEE Trans. Systems, Man, Cybern.*, SMC-14, pp. 879-884.

Golla, D.F., Garg, S.C., and Hughes, P.C., 1981, "Linear-state feedback control of manipulators," *Mech. Machine Theory*, 16, pp. 93-103.

Good, M.C., Sweet, L.M., and Strobel, K.L., 1985, "Dynamic models for control system design of integrated robot and drive systems," *ASME J. Dynamic Systems, Meas., Control*, 107, pp. 53-59.

Goor, R.M., 1985, "A new approach to robot control," *Proc. American Control Conf*, Boston, June 19-21, pp. 385-389.

Griffiths, John D., 1986, Experimental Evaluation of Computed Torque Control, B.S. Thesis, MIT, Mechanical Eng..

Kuwahara, H., Ono, Y., Nikaido, M., and Matsumoto, T., 1985, "A precision direct-drive robot arm," *Proc. American Control Conf.*, Boston, June 19-21, pp. 722-727.

Liégeois, A., Fournier, A. and Aldon, M., 1980, "Model Reference Control of High-Velocity Industrial Robots," *Proc. Joint Automatic Control Conf.*, San Francisco, CA, August 13-15.

Luh, J.Y.S., Walker, M.W., and Paul, R.P.C., 1980, "Resolved-Acceleration Control of Mechanical Manipulators," *IEEE Trans. Auto. Contr.*, AC-25(3), pp. 468-474.

Markiewicz, B., 1973, "Analysis of the Computed Torque Drive Method and Comparison With Conventional Position Servo for a Computer-Controlled Manipulator," Jet Propulsion Laboratory, Pasadena, CA, March 15.

Paul, R. C., Sept. 1972, "Modeling,trajectory calculation and servoing of a computer controlled arm," AIM-177, Stanford University Artificial Intelligence Laboratory.

References

Albus, J. S. (1975a) "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)." *J. Dynamic Systems, Measurement, and Control* **97**: 220-227

Albus, J. S. (1975b) "Data storage in the cerebellar model articulation controller (CMAC)." *J. Dynamic Systems, Measurement, and Control* **97**: 228-233

An, C. H., Atkeson, C. G. and Hollerbach, J. M. (1985) December 11-13 "Estimation of Inertial Parameters of Rigid Body Links of Manipulators." *Proc 24th Conf. Decision and Control*, Fort Lauderdale, Florida

Anand, D. K. (1984) Introduction to Control Systems. 2nd Edition New York, Pergamon Press

Anderson, B. D. O. and R. M. Johnstone (1981) "Convergence Results for Widrow's Adaptive Controller." *IFAC Conf. on System Identification*

Arimoto, S. (1985) May "Mathematical Theory of Learning With Applications to Robot Control." *Proc. of 4th Yale Workshop on Applications of Adaptive Systems Theory*, Center for Systems Science, Yale University, pp 215-220

Arimoto, S., S. Kawamura, and F. Miyazaki (1984a) "Bettering Operation of Robots by Learning." *J. of Robotic Systems* **1**: (2)123-140

Arimoto, S., S. Kawamura, and F. Miyazaki (1984b) August "Can Mechanical Robots Learn by Themselves?." *Proc. of 2nd Inter. Symp. Robotics Research*, Kyoto, Japan

Arimoto, S., S. Kawamura, and F. Miyazaki (1984c) Dec "Bettering Operation of Dynamic Systems By Learning: A New Control Theory For Servomechanisms of Mechatronics Systems." *Proc. 23rd IEEE CDC*, Las Vegas, Nevada

Arimoto, S., S. Kawamura, F. Miyazaki, and S. Tamaki (1985) Dec. 11-13 "Learning Control Theory for Dynamical Systems." *Proc.*

24th Conf. on Decision and Control, Fort Lauderdale, Florida

Asada, H., Kanade, K., and Takeyama, I. (1983) "Control of a Direct-Drive Arm." *Trans. of ASME* **105**: 136-142

Asada, H., and Youcef-Toumi, K. (1984) "Analysis and design of a direct-drive arm with a five-bar-link parallel drive mechanism." *ASME J. Dynamic Systems, Meas., Control* **106**: 225-230

Asada, H., Youcef-Toumi, K., and Lim, S.K. (1984) Dec. 12-14 "Joint torque measurement of a direct-drive arm." *Proc. 23rd Conf. Decision and Control*, Las Vegas, pp 1332-1337

Åström, Karl J. (1983) "Theory and Applications of Adaptive Control — A Survey." *Automatica* **19**: (5)471–486

Åström, Karl J. and Wittenmark, Björn (1984) Computer Controlled Systems: Theory and Design. Englewood Cliffs, N.J., Prentice-Hall, Inc.

Atkeson, C.G., An, C.H., and Hollerbach, J.M. (1985a) Dec. 11-13 "Rigid body load identification for manipulators." *Proc. 24th Conf. Decision and Control*, Fort Lauderdale, Florida

Atkeson, C. G., An, C. H., and Hollerbach, J. M. (1985b) Oct. "Estimation of Inertial Parameters of Manipulator Loads and Links." *Proc. 3rd Int. Symp. Robotics Research*, Gouvieux (Chantilly), France, pp 32-39

C. Atkeson and J. McIntyre (1986) July 1-3 "Applications of Adaptive Feedforward Control in Robotics." *Proceedings, 2nd IFAC Workshop on Adaptive Systems in Control and Signal Processing*, Lund, Sweden

Bejczy, A. K. (1974) "Robot Arm Dynamics and Control." Technical Memorandum 33-669, *Jet Propulsion Laboratory*, Pasadena, CA.

Bernard, J. W. and I. Lefkowitz (1962) "An Approach to Optimizing Control Based on a Generalized Dynamic Model." *Joint Automatic Control Conference*, pp 3-2, 1–9

Bonivento, C, and R. Guidorzi (1971) Aug. 11-13 "Parametric Identification of Linear Multivariable Systems." *Proc. 12th Joint Automatic Control Conference*, St. Louis, Missouri, pp 342-349

Brady, Michael, Hollerbach, John M., Johnson, Timothy L., Lozano-Pérez, Tomás, and Mason, Matthew T. (1982) Robot Motion: Planning and Control. Cambridge, MA., The MIT Press

Bristol, E. H. (1967) "A Simple Adaptive System For Industrial Control."

Instrumentation Technology **14**: (6)

Casalino, G. and L. Gambardella (1986a) April 7-10 "Learning of Movements in Robotic Manipulators." *Proc. 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp 572-578

Coiffet, P. (1983) Robot Technology: Interaction With The Environment. Englewood Cliffs, NJ, Prentice-Hall, Inc.

Craig, J. J. (1984) June 6-8 "Adaptive Control of Manipulators Through Repeated Trials." *Proc. American Control Conference*, San Diego, pp 1566-1574

Craig, J. J. (1983) *Adaptive Control of Manipulators*, PhD. Thesis Proposal, Computer Science Department, Stanford University, April..

D'Azzo, John J. and Houpis, Constantine H. (1966) Feedback Control System Analysis and Synthesis. New York, N.Y., McGraw-Hill Book Company

Davies, W. D. T. (1970) System Identification for Self-Adaptive Control. New York, Wiley-Interscience

Doebelin, Ernest O. (1962) Dynamics Analysis and Feedback Control. New York, Mcgraw-Hill Book Company

Eliçabe, G. E. and G. R. Meira (1983) "Adaptive Control with Feedforward Compensation and Reduced Reference Model." In: Landau, I. D., M. Tomizuka, and D. M. Auslander, Eds.(eds) Adaptive Systems in Control and Signal Processing 1983: Proceedings of the IFAC Workshop. New York, Pergamon Press, pp 79–80

Eveleigh, Virgil W. (1972) Introduction to Control Systems Design. New York, Mcgraw-Hill Book Company

Franklin, Gene F. and Powell, J. David (1980) Digital Control of Dynamic Systems. Reading, MA., Addison-Wesley Publishing Company

Furuta, K. and M. Yamakita (1986a) April 7-10 "Iterative Generation of Optimal Input of a Manipulator." *Proc. 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp 579-584

Gelb, A., ed. (1974) Applied Optimal Estimation. Cambridge, MA, The MIT Press

Gerstenberger, Michael Duane (1985) May " Manipulator Control Using Recursive Dynamics Computation." S. M. and E. E. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer

Gilbert, E.G., and Ha, I.J. (1984) "An approach to nonlinear feedback control with applications to robotics." *IEEE Trans. Systems, Man, Cybern.* **SMC-14**: 879-884

Golla, D.F., Garg, S.C., and Hughes, P.C. (1981) "Linear-state feedback control of manipulators." *Mech. Machine Theory* **16**: 93-103

Golub, G.H., and Van Loan, C.F. (1983) Matrix computations., John Hopkins University Press

Good, M.C., Sweet, L.M., and Strobel, K.L. (1985) "Dynamic models for control system design of integrated robot and drive systems." *ASME J. Dynamic Systems, Meas., Control,* **107**: 53-59

Goodwin, Graham C. and Kwai Sang Sin (1984) Adaptive Filtering, Prediction, and Control. Englewood Cliffs, NJ., Prentice-Hall, Inc.

Goor, R.M. (1985a) June 19-21 "A new approach to robot control." *Proc. American Control Conf,* Boston, pp 385-389

Goor, R. M. (1985b) Nov. 17–22 "A New Approach to Minimum Time Robot Control." *Winter Annual Meeting of ASME,* Miami Beach, FL., pp 1–11

Grabbe, Eugene M., Simon Ramo, and Dean E. Wooldridge, Eds. (1961) Handbook of Automation, Computation, and Control, Volume 3: Systems and Components. Los Angeles, CA., Thompson Ramo Wooldridge, Inc.

Graham, Robert E. (1946) "Linear Servo Theory." *The Bell System Technical Journal* **25**: (4)616–651

Grosjean, A., and C. Foulard (1976) 21-27 September, 1976 (published in 1978, by North Holland Publ., New York) "Extensions of Levin's Method (or Eigenvector Method) For the Identification of Discrete, Linear, Multivariable, Stochastic, Time Invariant, Dynamic Systems." *Identification and System Parameter Estimation, Proc. of 4th IFAC Symp.*, pp 2003-2010

Hamming, R. W. (1977) Digital Filters. Englewood Cliffs, NJ, Prentice-Hall, Inc.

Hara, S., T. Omata, and M. Nakano (1985) Dec. 11-13 "Synthesis of Repetitive Control Systems and its Application." *Proc. 24th Conf. on Decision and Control,* Fort Lauderdale, Florida

Harokopos, E. G. (1986a) April 7-10 "Optimal Learning Control of Mechanical Manipulators in Repetitive Motions." *Proc. 1986 IEEE International*

Conference on Robotics and Automation, San Francisco, CA, pp 396-401

Harokopos, E. G. (1986b) April 20-24 "Learning and Optimal Control of Industrial Robots in Repetitive Motions." *Proc. Robots 10*, Chicago, Illinois, pp 4-27 - 4-46

Harris, C. J. and S. A. Billings, Eds. (1981) Self-Tuning and Adaptive Control: Theory and Applications. New York, Peter Peregrinus, Ltd.

Hildreth, Ellen C and John M. Hollerbach (1985) "The Computational Approach to Vision and Motor Control." AIM 846, Artificial Intelligence Laboratory, *Massachusetts Institute of Technology*, Cambridge, MA.

Hirzinger, G. and K. Landzettel (1985) March 25-28 "Sensory Feedback Structures for Robots With Supervised Learning." *Proc. of 1985 IEEE Int. Conf. on Robotics and Automation*, St. Louis, Missouri

Hogan, Neville (1984) "Impedance Control of Industrial Robots." *Robotics and Computer-Integrated Manufacturing* **1**: (1)97–113

Hogan, N. (1985a) "Impedance control: An approach to manipulation: Part I - Theory." *ASME J. of Dynamic Systems, Measurement, and Control* **107**: 1-7

Hogan, N. (1985b) "Impedance control: An approach to manipulation: Part II - Implementation." *ASME J. of Dynamic Systems, Measurement, and Control* **107**: 8-16

Hogan, N. (1985c) "Impedance control: An approach to manipulation: Part III - Applications." *ASME J. of Dynamic Systems, Measurement, and Control* **107**: 17-24

Hollerbach, J. M. (1984) "Dynamic Scaling of Manipulator Trajectories." *Journal of Dynamics Systems, Measurement, and Control* **106**: 102-106

Hollerbach, J. M., and Sahar, G. (1983) "Wrist-partitioned inverse kinematic accelerations and manipulator dynamics." *Int. J. Robotics Research* **2**: (4)61-76

Hollars, Michael G. and Robert H. Cannon, Jr. (1985) Nov. 17–22 "Initial Experiments on the End-Point Control of a Two-Link Manipulator with Flexible Tendons." *ASME Winter Annual Meeting*, Miami Beach

Horowitz, Isaac M. (1963) Synthesis of Feedback Systems. New York, Academic Press

Isaacson, Eugene and Herbert Bishop Keller (1966) Analysis of Numerical Methods. New York, John Wiley and Sons

Isermann, Rolf (1981) Digital Control Systems. New York, Springer Verlag

Isermann, Rolf (1982) "Parameter Adaptive Control Algorithms — A Tutorial." *Automatica* **18**: (5)513–528

Kanade, T., P. K. Khosla, and N. Tanaka (1984) December "Real-Time Control of CMU Direct-Drive Arm II Using Customized Inverse Dynamics." *Proc. of 23rd Conf. on Decision and Control*, Las Vegas, Nevada

Kawamura, S., F. Miyazaki, and S. Arimoto (1985) Dec. 11-13 "Applications of Learning Method for Dynamic Control of Robot Manipulators." *Proc. 24th Conf. on Decision and Control*, Fort Lauderdale, Florida

Kawamura, S., F. Miyazaki, and S. Arimoto (1984) Oct "Iterative Learning Control For Robotic Systems." *Proc. of IECON 84*, Tokyo, Japan

Kazerooni, Homayoon (1985) February " A Robust Design Method For Impedance Control of Constrained Dynamic Systems." Ph. D. Thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering

Khosla, Pradeep K., and Takeo Kanade (1985) December "Parameter Identification of Robot Dynamics." *Proceedings of 24th Conference on Decision and Control*, Ft. Lauderdale, FL, pp 1754–1760

Koopmans, T. (1937) Linear Regression Analysis of Economic Time Series. Haarlem, The Netherlands, DeErven F. Bohm

Kotta, U., (1982) "On-line Eigenvector Algorithms For The Identification of Dynamic Systems." *IFAC Conference on Identification and System Parameter Estimation*, Washington, DC, USA, pp 1265-1269

Kotta, U. (1979) "Structure and Parameter Estimation of Multivariable Systems Using Eigenvector Method." *IFAC Conference on Identification and System Parameter Estimation*, pp 453-458

Landau, Yoan D. (1979) Adaptive Control: The Model Reference Approach. New York, Marcel Dekker

Landau, I. D., M. Tomizuka, and D. M. Auslander, Eds. (1983) Adaptive Systems in Control and Signal Processing 1983: Proceedings of the IFAC Workshop. New York, Pergamon Press

Lee, K. (1983) December " Shape Optimization of Assemblies Using Geometric Properties." Ph.D. Thesis, MIT, Department of Mechanical Engineering Department

Leigh, J. R. (1985) Applied Digital Control: Theory, Design, and Implementation. Englewood Cliffs, NJ, Prentice Hall International

Levin, M. J., (1964) "Estimation of a System Pulse Transfer Function In the Presence of Noise." *IEEE Trans. on Auto. Contr.* **AC-9**: 229-235

Liégeois, Alain (1985) Robot Technology: Performance and Computer-Aided Design. Englewood Cliffs, N.J., Prentice-Hall, Inc.

Liegeois, A., Fournier, A., and Aldon, M.J. (1980) August 13-15 "Model reference control of high-velocity industrial robots." *Proc. Joint Automatic Control Conf*, San Francisco

Ljung, L. and Soderstrom, T. (1983) Theory and Practice of Recursive Idenfication., MIT Press

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

Luh, J.Y.S., Walker, M., and Paul, R.P. (1980b) "On-line computational scheme for mechanical manipulators." *J. Dynamic Systems, Meas., Control* **102**: 69-76

Macmillan, R. H. (1951) An Introduction to the Theory of Control in Mechanical Engineering. Cambridge, England, Cambridge University Press

Markiewicz, B. R. (1973) "Analysis of the computed Torque Drive Method and Comparison with the Conventional Position Servo for a Computer-Controlled Manipulator." Technical Memorandum 33-601, *Jet Propulsion Laboratory*, Pasadena, CA.

Marquardt, D.W., and Snee, R.D. (1975) "Ridge regression in practice." *Amer. Statistician* **29**: 3-20

Marr, David (1977) "Artificial Intelligence – A Personal View." *Artificial Intelligence* **9**: 37-48

Marr, David (1982) Vision. San Francisco, W. H. Freeman and Co.

Marshall, S. A. (1978) Introduction to Control Theory. London, The Macmillan Press, Ltd.

Mayeda, H., Osuka, K., and Kangawa, A. (1984) July 2-6 "A new identification method for serial manipulator arms." *Preprints IFAC 9th World Congress*, Budapest, pp 74-79

Mita, T., and E. Kato (1985) Dec. 11-13 "Iterative Control and its Application to Motion Control of Robot Arm – A Direct Approach to Servo-Problems." *Proc. 24th Conf. on Decision and Control*, Fort Lauderdale, Florida

Moore, J. R. (1951) "Combination Open-Cycle Closed-Cycle Systems." *Proceedings of the I.R.E.* **39**: 1421-1432

Morita, Atsushi (February 27) " A Study of Learning Controllers For Robot Manipulators With Sparse Data." M.S. Thesis, Massachusetts Institute of Technology, Department of Mechanical Engineering

Mukerjee, A. (1984) November "Adaptation in biological sensory-motor systems: A model for robotic control." *Proc., SPIE Conf. on Intelligent Robots and Computer Vision, SPIE Vol. 521*, Cambridge

Mukerjee, A., and Ballard, D.H. (1985) Mar. 25-28 "Self-calibration in robot manipulators." *Proc. IEEE Conf. Robotics and Automation*, St. Louis, pp 1050-1057

Neuman, C.P., and Khosla, P.K. (1985) May 29-31 "Identification of robot dynamics: an application of recursive estimation." *Proc. 4th Yale Workshop on Applications of Adaptive Systems Theory*, New Haven, pp 42-49

Ogata, Katsuhiko (1970) Modern Control Engineering. Englewood Cliffs, N.J., Prentice-Hall, Inc.

Olsen, H.B., and Bekey, G.A. (1985) Mar. 25-28 "Identification of parameters in models of robots with rotary joints." *Proc. IEEE Conf. Robotics and Automation*, St. Louis, pp 1045-1050

Oppenheim, A. V., and A. S. Willsky (1983) Signals and Systems. Englewood Cliffs, NJ, Prentice-Hall, Inc.

Orin, D. E., R. B. McGhee, M. Vukobratovic, and G. Hartoch (1979) "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods." *Mathematical Biosciences* **43**: 107-130

Owens, D. H. (1978) Feedback and Multivariable Systems. New York, Peter Peregrinus

Paul, R. C. (1972) "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm." A. I. Memo 177, Stanford Artificial Intelligence Laboratory, *Stanford University*,

Paul, R.P. (1981) Robot Manipulators: Mathematics, Programming, and Control. Cambridge, Mass., MIT Press

Peschon, John (1965) Disciplines and Techniques of Systems Control. New York, Blaisdell Publishing Company

Preminger, J. and J. Rootenberg (1964) "Some Considerations Relating to Control Systems Employing the Invariance Principle." *IEEE Transactions on Automatic Control* **AC-9**: 209-215

Raibert, M. H. (1977) December "Analytical equations vs. table look-up for manipulation: a unifying concept." *Proc. IEEE Conf. Decision and Control*, New Orleans, La., pp 576-579

Raibert, M. H. (1978) "A model for sensorimotor control and learning." *Biological Cybernetics* **29**: 29-36

Raibert, M. H. and B. K. P. Horn (1978) "Manipulator control using the Configuration Space Method." *Industrial Robot* **5**: (2)69-73

Raven, Francis H. (1978) Automatic Control Engineering. 3rd Edition New York, Mcgraw-Hill Book Company

Samson, C. (1983) Dec. 14-16 "Robust nonlinear control of robotic manipulators." *Proc. 22nd IEEE Conf. Decision and Control*, San Antonio

Saridis, George N. (1977) Self-Organizing Control of Stochastic Systems. New York, Marcel Dekker, Inc.

Savas, Emanuel S. (1965) Computer Control of Industrial Processes. New York, Mcgraw-Hill Book Company

Schumann, R. and H. Christ (1979) June 17–21 "Adaptive Feedforward Controllers for Measurable Disturbances." *Joint Automatic Control Conference*, Denver, CO

Schwarzenbach, J. and K. F. Gill (1984) System Modelling and Control. 2nd Edition Baltimore, MD, Edward Arnold

Shinskey, F. G. (1963) "Feedforward Control Applied." *ISA Journal* **10**: (11)61–65

Shinskey, F. G. (1979) Process-Control Systems: Application, Design, Adjustment. 2nd Edition New York, McGraw-Hill Book Company

Slotine, J.-J. E. (1985) "The robust control of robot manipulators." *Int. J. Robotics Research* **4**: (2)49-64

Silvey, S. D. (1975) Statistical Inference. London, Chapman and Hall

Smith, F. W. (1968) "System Laplace-Transform Estimation from

Sampled Data." *IEEE Trans. on Auto. Contr.* **AC-13**: 37-44

Spong, M.W., Thorp, J.S., and Kleinwaks, J.M. (1984) Dec. 12-14 "The control of robot manipulators with bounded input. Part II: robustness and disturbance rejection." *Proc. 23rd IEEE Conf. Decision and Control*, Las Vegas, pp 1047-1052

Sweet, L.M., and Good, M.C. (1984) Dec. 12-14 "Re-definition of the robot motion control problem: effects of plant dynamics, drive system constraints, and user requirements." *Proc. 23rd Conf. Decision and Control*, Las Vegas, pp 724-732

Sweet, L. M., and M. C. Good (1985) "Re-definition of the robot motion control problem: effects of plant dynamics, drive system constraints, and user requirements." *IEEE Control Systems Magazine* 5: (3)18-25

Symon, K.R. (1971) Mechanics. Reading, Mass., Addison-Wesley

Szentágothai, John and Michael A. Arbib (1975) Conceptual Models of Neural Organization. Cambridge, MA, The MIT Press

Takahashi, Yasundo and Rabins, Michael J. and Auslander, David M. (1970) Control and Dynamic Systems. Reading, MA., Addison-Wesley Publishing Company

Tian-Xing, Y., L. Duan, Z. Zhong-Jun (1982) "Re-definition of the robot motion control problem: effects of plant dynamics, drive system constraints, and user requirements." *Proc. of the American Control Conference*, pp 165-169

Togai, Masaki and Osamu Yamano (1986) April 7-10 "Learning Control and Its Optimality: Analysis and Is Application to Controlling Industrial Robots." *Proc. 1986 IEEE International Conference on Robotics and Automation*, San Francisco, CA, pp 248-253

Togai, M., and O. Yamano (1985) Dec. 11-13 "Analysis and Design of an Optimal Learning Control Scheme for Industrial Robots: A Discrete Time Approach." *Proc. 24th Conf. on Decision and Control*, Fort Lauderdale, Florida

Tou, Julius T. (1959) Digital and Sampled-data Control Systems. New York, Mcgraw-Hill Book Company, Inc.

Tsypkin, Ya. Z. (1971) Adaptation and Learning in Automatic Systems. New York, Academic Press

Uchiyama, M. (1978) "Formation of High-Speed Motion Pattern of a Mechanical Arm by Trial." *Trans. of Society of Instrument and Control Engineers*

(Japan) **19**: (5)706–712

Voronov, A. A. and V. Yu. Rutkovsky (1984) "State-of-the-art and Prospects of Adaptive Systems." *Automatica* **20**: (5)547–557

Vukobratović, M. and V. Potkonjak (1982) Dynamics of Manipulation Robots: Theory and Application. New York, Spring-Verlag

Vukobratović, M. and D. Stokić (1982) Control of Manipulation Robots: Theory and Application. New York, Spring-Verlag

Wang, S. H. (1984) October "Computed Reference Error Adjustment Technique (CREATE) For The Control of Robot Manipulators." *22nd Annual Allerton Conf. on Communication, Control, and Computing*

Wang, S. H., and I. Horowitz (1985) March "CREATE - A New Adaptive Technique." *Proc. of the Nineteenth Annual Conf. on Information Sciences and Systems*

Webb, C. R. (1964) Automatic Control: An Introduction. New York, McGraw-Hill Book Company

Weyrick, Robert C. (1975) Fundamentals of Automatic Control. New York, McGraw-Hill Book Company

Whitney, D.E., Lozinski, C.A., and Rourke, J.M. (1984) "Industrial Robot Calibration Method and Results." CSDL-P-1879, *Charles Stark Draper Laboratory*, Cambridge, Mass..

Widrow, Bernard, Eugene Walach, and Shmuel Shaffer (1983) Oct. 31 – Nov. 2 "Adaptive Signal Processing for Adaptive Control." *Proceedings of the Seventeenth Asilomar Conference on Circuits, Systems, & Computers*, Santa Clara, CA, pp 265–270

Widrow, B. and E. Walach (1983) June 20–22 "Adaptive Signal Processing for Adaptive Control." *Proceedings of the IFAC Workshop: Adaptive Systems in Control and Signal Processing, 1983*, San Francisco, pp 7–12

Widrow, B. and others (1981) November "On Adaptive Inverse Control." *Proceedings of the Fifteenth Asilomar Conference on Circuits, Systems, & Computers*

Widrow, B., John M. McCool, and Barry P. Medoff (1978) Nov. 6–8 "Adaptive Control by Inverse Modeling." *Proceedings of Twelfth Asilomar Conference on Circuits, Systems, & Computers*, Pacific Grove, CA, pp 90–94

Winston, Patrick Henry (1984) Artificial Intelligence (2nd edition). Reading, MA, Addison Wesley

Wittenmark B. (1975) "Stochastic Adaptive Control Methods: A Survey." *Int. J. Control* **21**: (5)705-730

Wittenmark, B (1973) "A Self-tuning Regulator." 7311, Dept. of Automatic Control, *Lund Institute of Technology*, Lund, Sweden.

Woolverton, P. F. and P. W. Murrill (1967) "An Evaluation of Four Ideas in Feedforward Control." *Instrumentation Technology* **14**: (1)

Yale (1985) "Proceedings of The Fourth Yale Workshop on Applications of Adaptive Systems Theory." Center for Systems Science, Becton Center, *Yale University*, New Haven, Conn..