Dynamical Simulation and Control of Articulated Limbs

Thesis by

Marcus Quintana Mitchell

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1997
(Submitted October 2, 1996)

Acknowledgements

Many people have given me help, guidance, and inspiration in my years in graduate school. To all of them I am very grateful.

I wish to thank Tom Annau, Carlos Brody, Sanjoy Mahajan, Erik Winfree, Sam Roweis, and past members of the Hopfield group for their stimulating conversation and insights.

I thank the members of my examining committee, Yaser Abu-Mostafa, Alan Barr, Joel Burdick, and John Hopfield, professors who lead by example.

I particularly thank my advisor, John Hopfield, for providing a fantastic working environment at Caltech.

I thank Laura Rodriguez for her unwavering support over the last few years.

I thank NASA and NSF for funding over most of my graduate career.

Special thanks go to my parents, who have consistently encouraged me, and to Courtney Lee, whose love and support has been invaluable.

Abstract

Many useful mechanisms can be modelled as *articulated systems*: collections of rigid bodies linked together with joints that constrain relative movement. The two parts of this thesis study the complementary problems of simulation and control for such systems. In the first part, we describe an implementation and extension of a physically based modelling framework known as "dynamic constraints" in which forces of constraint linking bodies in an articulated system are explicitly calculated. In addition to identifying some important robustness and stability issues for these calculations, we extend the framework to systems whose internal degrees of freedom can be directly parameterized. This permits significant efficiency gains for mechanisms which model limbs. The second part of the thesis centers on the adaptive control of limb configuration through simulated actuators. In this problem, the nonlinear structure and parametric details of a limb are assumed to be unknown. We present and illustrate the performance of an adaptive scheme which performs considerably better than conventional nonadaptive techniques, and which is competitive with adaptive methods which use more a priori knowledge of limb dynamics.

Contents

List of Figures

List of Tables

Chapter 1 Introduction and Overview

1.1 Themes

This thesis began as a study of a biologically inspired control problem, the control of the configuration of moving limbs like arms, legs, or fingers. A clear understanding of configuration control is a *first step* to designing mechanical systems to carry out complicated tasks in the presence of other objects, sensor and actuator limitations, and uncertainty about the environment. But a prerequisite for studying configuration control is a firm understanding of the behavior of a limb under the influence of the physical forces produced by actuators such as motors or muscles: the **dynamics** of the limb. The calculations needed for the dynamics of limb models are the subject of the first part of this thesis.

An **articulated limb** can be modelled as a collection of rigid bodies with constraints on their relative motion. The joints prevent and allow certain types of motion, while the actuators are able to induce motion by exerting forces on the links of the limb. Other components of the environment (e.g. gravity, friction, other objects) produce forces which induce and prevent movement as well; all these combine to determine the motion of the limb over time. Of course, the limb segments aren't perfectly rigid, joints can break, and actuators can't exert arbitrary forces, so a detailed account of all the potentially relevant constraints may be difficult or impossible to find. But the basic picture in figure 1.1 is a good starting point, and there are a number of interesting research issues associated with simulating the behavior of collections of constrained rigid bodies.

Distinct from simulation, control problems presume the availability of a number of **control inputs** to a mechanical system. These control inputs can be translated into

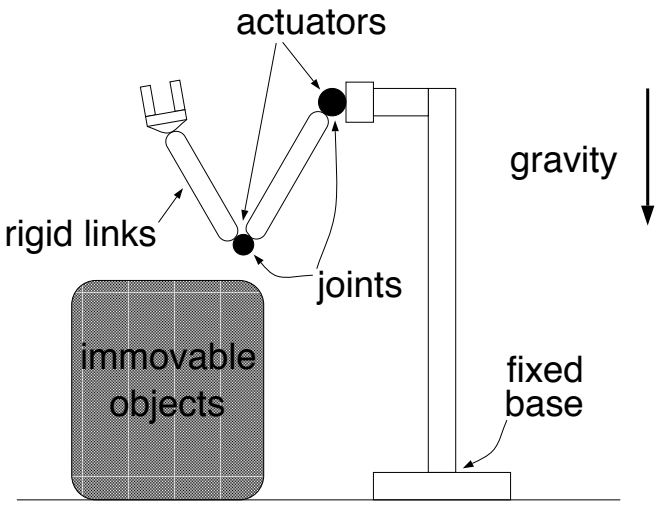


Figure 1.1: An arm in a cluttered environment

physical forces through actuator models, and dynamics calculations can be used to determine the resulting movement. The locations and orientations of the individual bodies in a limb can be parameterized to describe the **configuration** of the limb. Figure 1.1 shows a progression of limb models with the number of configurational degrees of freedom labelled. The most complicated limb shown is a reasonable model of the structure of a human arm. Many aspects of the dynamics for this complicated limb are reflected in the simpler arm models shown. Then the configuration control problem, defined more carefully in a later chapter, involves choosing control inputs which make the configuration parameters assume desired time-varying values.

One branch of the control problem is relevant to both purely virtual, simulated systems as well as to "real world" physical mechanisms. In this branch, control is attempted *without* perfect knowledge of the details of a dynamics model. The second part of this thesis concerns **adaptive** solutions for control under these circumstances.

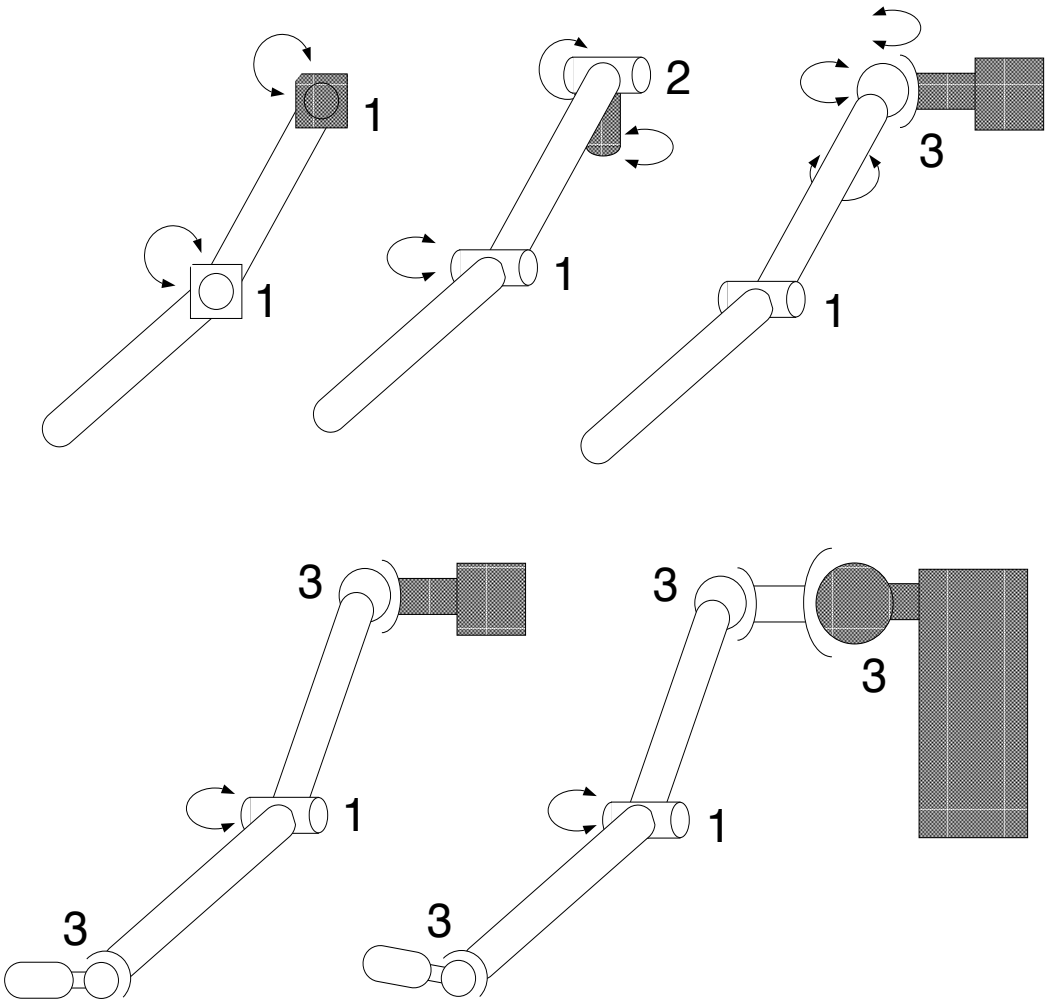


Figure 1.2: A progression of articulated limbs

1.1.1 Constraint-based dynamical simulation

This chapter describes a set of algorithms for the modelling and simulation of constrained rigid bodies. The system on which we build uses ideas from [Barzel and Barr, 1988] and [Barzel, 1992] and allows the simulation of individual rigid bodies moving under holonomic constraints enforced through the application of forces and torques. This "dynamic constraints" approach is powerful but can be quite inefficient when applied to highly constrained systems such as articulated limbs. Equations of motion for articulated limbs can be efficiently and directly generated, and we give formulas for incorporating such equations into the dynamic constraints framework. Further, a simple singular value decomposition-based method is given to enhance the robustness of the constraint force calculations.

1.1.2 Mixture models and adaptive configuration control of open-chain robots

A class of nonlinear compensators for the adaptive control of the configuration of open chain manipulators is presented in this chapter. The compensation scheme is based only on the linear structure of the equations of motion for rigid body dynamics. Adaptation rules are developed based on the passivity approach introduced by [Slotine and Li, 1991]. Through simulation experiments with a model two-link arm, we study a number of performance issues which are not always apparent in formal stability proofs.

1.2 Related work

The individual chapters contain references to the vast literature concerned with different aspects of these topics. But there are a few specific bodies of work that have been particularly useful in framing problems and looking for solutions.

Murray, Li, and Sastry [Murray et al., 1994] give an excellent account of the math-

ematical tools for modern robot mechanics. Their presentation clarifies the essential concepts and coordinate transformations needed to interpret constraints and formulate equations of motion.

[Barzel and Barr, 1988] set up a paradigm for constraint-based dynamical simulation that is the starting point for the work described in the second chapter. The conclusion of [Barzel and Barr, 1988] includes a number of ideas for future research, and the work described here represents a study of some of those topics. [Pfarner, 1996] has pursued similar extensions for efficient dynamics calculations for articulated systems in the dynamic constraints framework.

Slotine and Li Slotine and Li [1991] emphasize model-based adaptive control and the passivity framework. Chapter 3 is directly inspired by their work, which is the clearest presentation of the use of energy-based methods for nonlinear adaptive control in the literature.

The recent thesis research of Robert Sanner Sanner and Slotine [1995] and Brian Mirtich Mirtich [1996] studies the adaptive control and dynamic simulation problems respectively. Although there is a lot of overlap between their work and the work described here, differences in philosophy and motivation make the research complementary rather than redundant.

1.3 Preliminaries

The chapters are self-contained, but it's useful to review basics, fix some terminology, and emphasize the appearance of subtle representational issues from the very beginning. The information in this section can be found in many textbooks covering rigid body dynamics.

The configuration of a single rigid body is intuitively described by the position

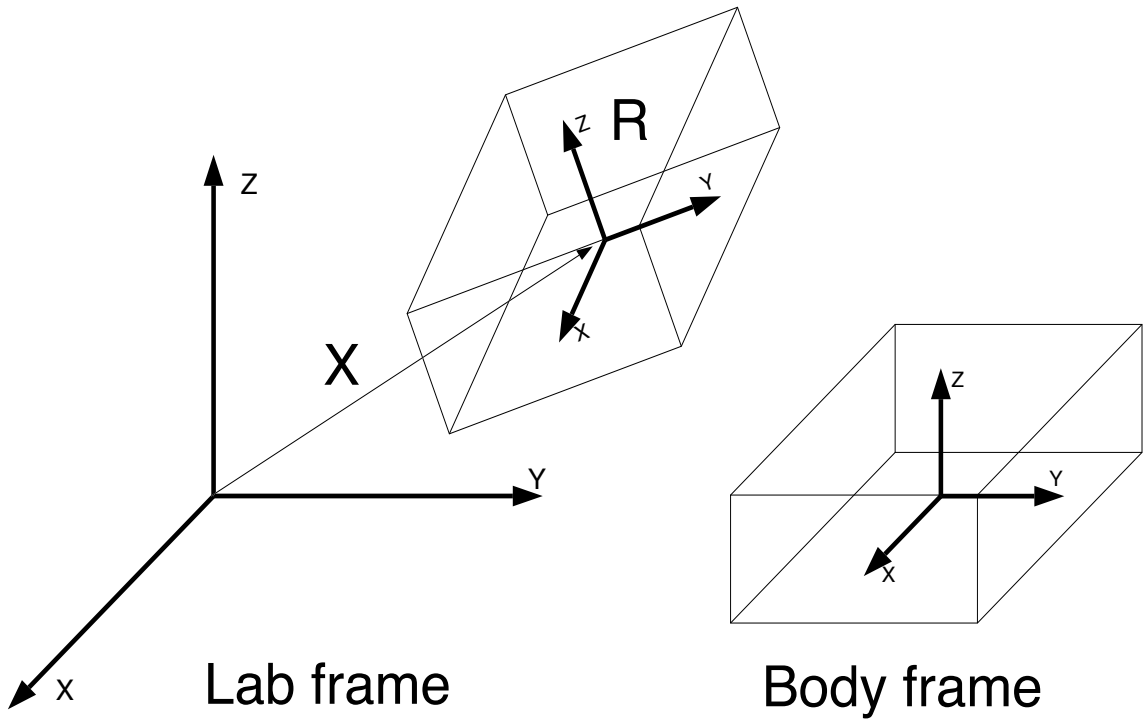


Figure 1.3: Coordinate frames

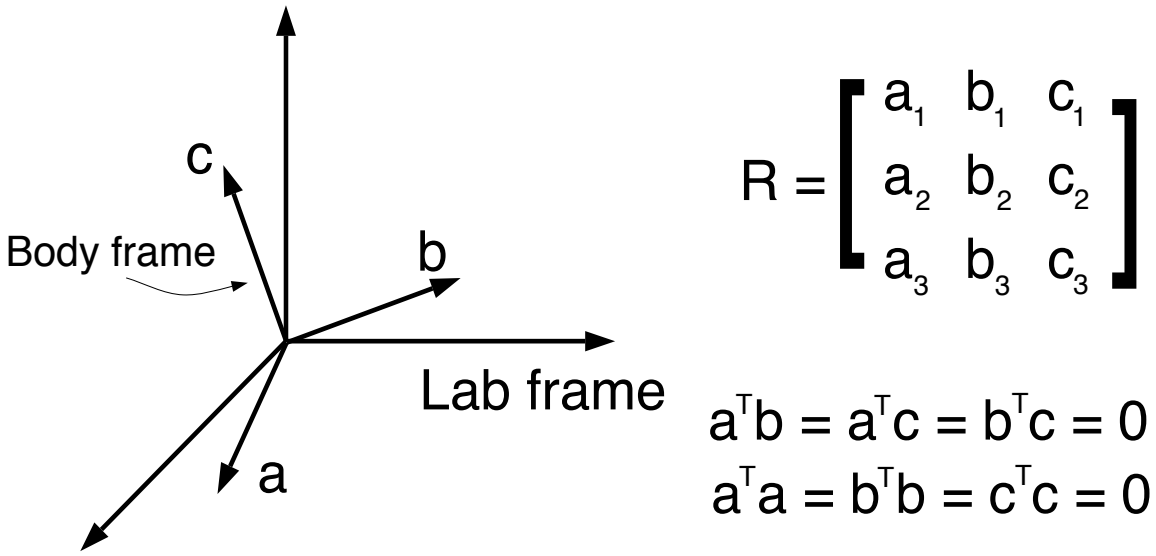


Figure 1.4: Representing orientation

and orientation of a coordinate frame attached to the body (figure 1.3). Position is represented by a 3-vector x, and orientation by a matrix R whose columns are the unit-length, right-handed, orthogonal axes of the body coordinate frame. Both x and R are measured with respect to a **laboratory frame**, or inertial frame, which is fixed. As the body translates and rotates in time, its configuration is described by the resulting trajectories $x(t)$ and $R(t)$.

Note that representing orientation with a rotation matrix (figure 1.3) requires 9 numbers for the entries, making the dimension of the configuration nominally $9 + 3 = 12$ dimensional. This is misleading, since the entries of R are not independently specifiable but are interrelated in a specific way: the columns are orthonormal as depicted in the figure, so in matrix form, $R^T R = RR^T = 1_{3x3}$, the 3x3 identity matrix.[1] So although *given* an object in some configuration we can find 12 numbers to describe that

[1] We use this notation for the identity matrix to avoid confusion with the moment of inertia matrix, described below.

configuration, it is not true that *any* 12 numbers can be unambiguously interpreted as an object's configuration. The rotation matrix is a kind of "redundant code" that makes studying rigid bodies analytically easier. "Valid" orientations are a subset of R^9 known as $SO(3)$, the special orthogonal group of $3x3$ matrices with determinant 1. $SO(3)$ has manifold structure, i.e. it can be identified with a curved three-dimensional surface in a 9-dimensional space, where the surface is locally Euclidean.

The pair (x, R) is in turn a representation of $SE(3)$, the special Euclidean group of rigid body transformations in 3-space. Elements of $SE(3)$ are denoted by $g \equiv (x, R)$. Analogous to $SO(3)$, $SE(3)$ can be thought of as a 6-dimensional curved submanifold of R^{12}. $SE(3)$ describes the configuration space of a rigid body, and the fact that it is 6-dimensional corresponds to the familiar fact that a rigid object free to translate and rotate has 6 "degrees of freedom." The degrees of freedom of a system are simply the variables in a local parameterization of its configuration space.

The evolution of x and R in time is determined by the influence of forces and torques on the velocity and angular velocity of the body, respectively (figure 1.3. The **Newton-Euler** equations state that $\frac{dp}{dt} = F$, and $\frac{dL}{dt} = T$, where p is the linear momentum $m\dot{x}$, and L is the angular momentum $I\omega$. Angular velocity ω is interpreted as the instantaneous right-handed spinning of a body about a unit axis $\omega/||\omega||$ at rate $||\omega||$ radians per second.[2] I is the $3x3$ matrix whose entries are the moments and products of inertia about the axes of the body frame R.[3] As the body orientation R with respect to the lab frame changes, so does I; it is given by $I = RI_bR^T$ where I_b is the (constant) inertia matrix with respect to the body-fixed coordinate frame. Note that I and I_b are always invertible.

So the rigid body is a dynamical system whose state – a collection of quantities summarizing the history of the system – is given by 18 numbers (x, R, p, L), the evolution

[2]Clearly, no axis is uniquely defined when $||\omega|| = 0$.

[3]For a rigid body of mass distribution $\rho(x)$, the inertia matrix is a 3x3 array.

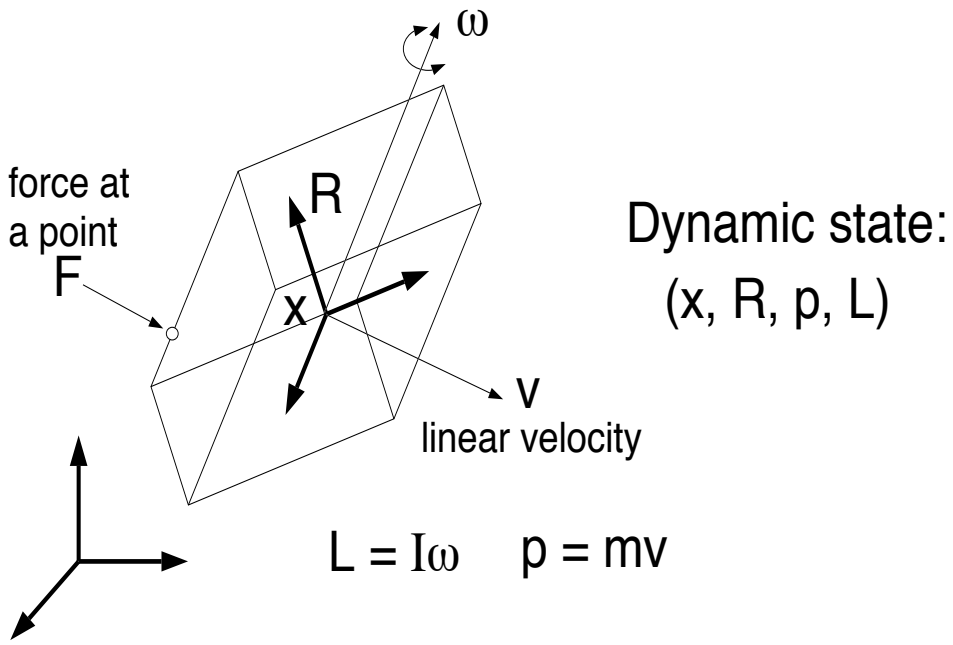


Figure 1.5: Dynamic state variables

of which are governed by the first-order equations

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dR}{dt} &= \omega \times R \\ \frac{dp}{dt} &= F(t) \\ \frac{dL}{dt} &= T(t) \end{aligned} \tag{1.1}$$

where $\times$ denotes the cross product, along with the static relations $\omega = I^{-1}L$, $I = RI_bR^T$, $v = \frac{1}{m}p$. Note that the only complicated parts concern R and ω, reflecting the previously mentioned coupling between the entries of R.

With multiple bodies linked together by joints, the situation becomes a little more complicated. A joint is a constraint on the relative configurations (and velocities and accelerations) of the linked bodies. The presence of joints adds *more* coupling between

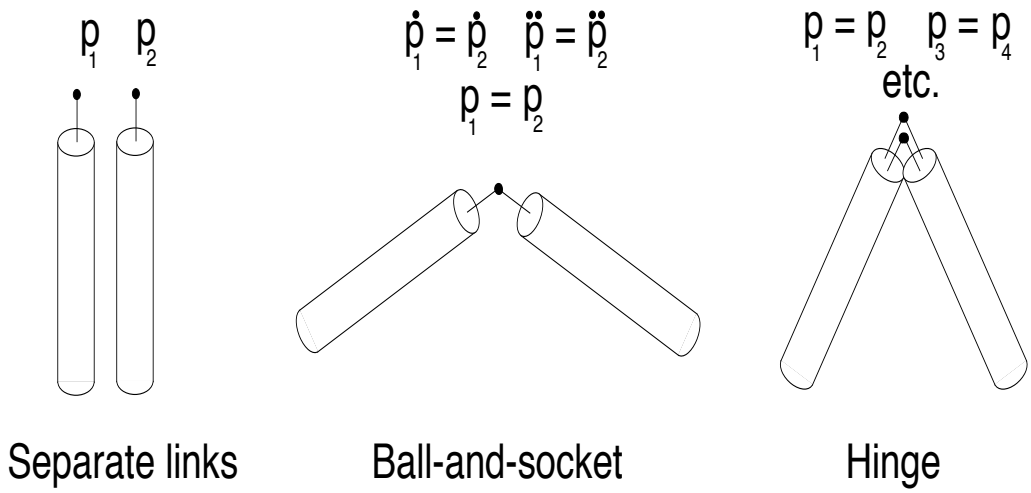


Figure 1.6: Constraining points on rigid bodies

the state variables for a collection of otherwise independent bodies.

One way to approach the equations of motion for collection of constrained bodies is to explicitly solve for the **forces of constraint** needed to hold the joints together. For example (figure 1.3), a **ball and socket** joint (e.g. a shoulder or a hip) requires that two points, one on each link, occupy the same location in space and have zero relative acceleration and velocity. Denoting the points on the two links that are to be joined as p_1 and p_2, it is possible to find the relative accelerations of the points in terms of the forces acting on the two links. If a constraint force f_{c1} acting on body 1 at point p_1, and another f_{c2} on body 2 at point p_2, it should be possible to find values for these forces that would hold the points together for all time. Chapter 2 details this kind of calculation; the upshot is that the forces of constraint can be found as the solution to a linear equation whose terms depend on the state variables of the constrained links.

So given an understanding of the dynamics of individual bodies, the dynamics for a linked system consist of the individual equations, and a linear constraint force calculation "on the side." Notice, though, that many more variables are used to describe the state of the system than should be strictly needed, since the joint constraints **couple** the degrees of freedom of the individual links. Direct parameterization of the available degrees of freedom addresses this issue, and the idea of generalized coordinates is central.

Collar and Simpson ([Collar and Simpson, 1987], p. 248) give an simple description of the idea of generalized coordinates:

> "A set of **generalized coordinates** is a set of variables which describe uniquely the dynamical configurations of the system, and which may be varied arbitrarily and independently without violating the constraints of the system. An infinite number of sets of generalized coordinates exists, but each set has the same number of members. For a system with holonomic constraints, this number is the number of degrees of freedom of the system."

Lagrange's equations come from the fact that work[4] is a change in kinetic energy, and hence does not depend on what coordinates one chooses to use to describe the system. By equating work done in a constrained coordinate system with work done in a (hypothesized) generalized coordinate system, the Euler Lagrange equations result:

$$\frac{d}{ddt}\frac{\partial E_k}{\partial \dot{\theta}} - \frac{\partial E_k}{\partial \theta} = \tau \tag{1.2}$$

where E_k is the kinetic energy, in this case the **Lagrangian** for the system[5]. The importance of this set of equations is that if the kinetic energy of a system can be written in terms of generalized coordinates, the equations of motion can be determined from

[4]The work done by a force acting at a point is the product of the magnitude of the force and the displacement of the point in the direction of the force.

[5]A full Lagrangian incorporates potential energy as well, but this is unnecessary for our purposes as potential forces can always be incorporated later.

(1.2). Kinetic energy is typically a quadratic form in some velocity parameterization, so that (1.2) results in the second-order dynamics familiar in mechanics, and discussed in more detail in the chapters to come.

Chapter 2 Constraint-based dynamical simulation

2.1 Introduction

The ability to simulate the behavior of mechanical models has a wide range of potential applications, and some of the most interesting concern the creation of virtual environments in which to study problems in movement control and manipulation of objects. Rigid bodies are a basic building block for mechanical simulators, and a variety of mechanisms can be modelled as a collection of constrained rigid bodies. This paper discusses one useful class of constraint – point constraints – and gives details of a simulation environment for articulated bodies constructed with and acting under the influence of these constraints.

We build directly on the work of [Barzel and Barr, 1988], who described a framework for the robust implementation of point constraints for individual rigid bodies. This framework, known as the **dynamic constraints** approach, provides a flexible means for constructing articulated bodies, but can be computationally demanding for highly constrained systems. Fortunately, dynamic constraints can be extended to allow the use of more efficient algorithms for determining the unconstrained motion of certain common types of articulated system. In fact, it is possible to identify a simple set of vector and matrix quantities required to calculate the forces associated with dynamic constraints, so that any object model (rigid body, articulated body, mass-spring lattice) for which these quantities can be computed can also be dynamically constrained. This extension of the dynamic constraints approach is the main topic of this paper.

Physically-based modelling is a highly active research area concerned with developing and implementing mathematical models of physical (typically mechanical) phenomena.

Some of the most relevant and pioneering recent work has emerged from projects at Cornell (Baraff [1992], Cremer [1992], Hopcroft [1986]), Caltech (Barzel [1992], Barzel and Barr [1988]), and Berkeley (Mirtich [1996], Lin and Canny [1995]), to name only a few. The work described here and in the companion paper [Mitchell, 1996] is an attempt to draw together a number of ideas from this literature to allow the simulation of multiple constrained, colliding, and contacting rigid bodies in the presence of active and passive forces such as gravity, spring forces, friction, and actuator forces.

In section 2.2, we review the important issues for simulating individual rigid bodies with point constraints. Section 2.3 describes examples and commentary on aspects of the performance of point constraints, and give a simple method for dealing with under and overdetermined constraint forces. Section 2.4 gives the extension to articulated bodies and comment on the general requirements for dynamic constraints. Section 2.5 illustrates the performance of dynamic constraints with a few case studies. Section 2.6 concludes by discussing a number of practical issues associated with building a robust simulation environment, and gives suggestions for future research.

2.2 Point constraints for simple bodies

2.2.1 An unconstrained rigid body

The equations of motion for a single rigid body acting under the influence of force F and torque T are

$$\begin{aligned} \frac{dx}{dt} &= v & \\ \frac{dR}{dt} = \omega \times R \qquad & \qquad \frac{dq}{dt} = \frac{1}{2}\bar{\omega} * q & (2.1) \\ \frac{dp}{dt} &= F & \\ \frac{dL}{dt} &= T & (2.2) \end{aligned}$$

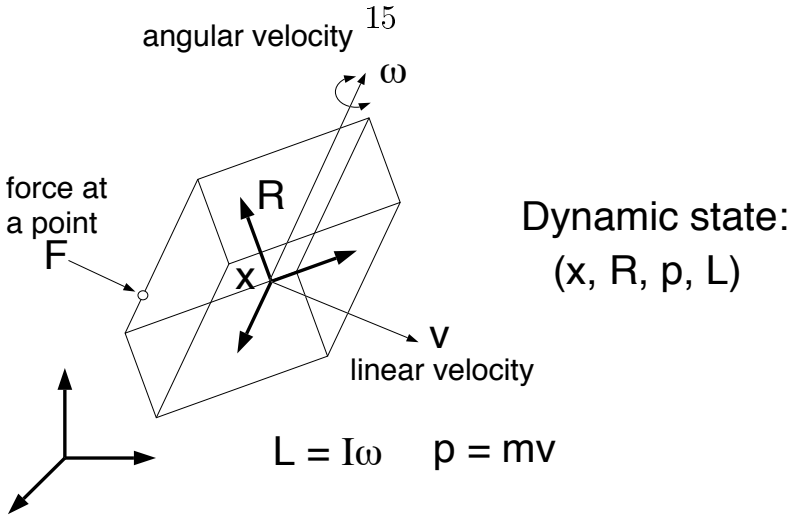
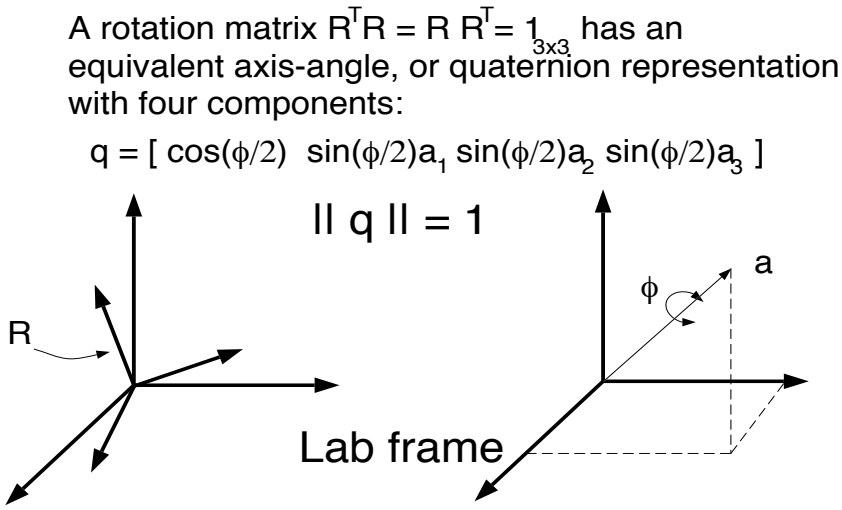


Figure 2.1: Frames and variables associated with a rigid body

where x is the position of the body's center of mass; R is an orthogonal matrix describing a frame attached to the object, originating at the center of mass; q is a unit **quaternion**, an alternate description of the body frame R; v is the velocity of the center of mass; ω is the angular velocity; p is the linear momentum; and L is the angular momentum. The symbol $\times$ denotes the cross product and $*$ denotes quaternion multiplication.[1] The **configuration** of the body is given by (x, R), which parameterizes $SE(3)$, the Euclidean group of rigid body transformations which acts as a **configuration space** for a rigid body. The **state** or dynamic state of the body is given by (x, R, p, L) or equivalently (x, R, v, ω). The configuration and state of a collection of bodies is defined in the obvious way. All quantities are measured with respect to a laboratory frame.

The differential equations (2.2) are a mathematical model of the behavior of a rigid body, and approximate solutions can be obtained by numerically integrating the equations. The position, momentum, and angular momentum are described by elements of R^3, and their respective differential equations are extremely simple. However, the parameterization of orientation is more complicated. A matrix R is a valid represen-

[1] See appendix for background on the cross products and quaternion multiplication appearing in (2.1).



tation of orientation only if it is orthonormal, and similarly, a 4-tuple q is valid if it is of unit length. Formally both differential equations have solutions with the property that if R (resp. q) is a valid orientation at time t_0, then solutions of 2.1 will be valid for $t > t_0$. In both cases,[2] the set of valid orientations is a smooth **submanifold** of R^9 or R^4, and the differential equations 2.1 define **vector fields** tangent to the manifold surface.

The inevitable presence of small numerical errors due to the discretization of 2.1 will cause the representations R and q to slowly depart from validity, as each non-infinitesimal step taken by a numerical integrator potentially leads farther and farther away from the manifold of valid orientations. As a result, maintaining the validity of the components of R and q in the strictest manner possible is essential to avoid the possible rapid onset of instability. The quaternion representation is quite convenient for this purpose since normalizing q achieves the desired projection onto the set of valid orientations. In contrast, the projection operation for R, involving normalization and orthogonalization for the rows, is a more complicated prospect, though approaches

[2]Quaternions have the additional issue that $q$and $-q$ describe the same orientation, but this is not a problem for numerical integration.

do exist [Barr, 1983]. In the sections below, the rotation matrix R is used for analysis involving body orientation, but for implementation, the quaternion is used in conjunction with explicit renormalization.

Forces and torques acting on a body can stem from many sources, such as gravity, springs and dampers, or simulated actuators. Work and energy are important quantities to keep track of since in addition to forces and torques, integration noise and numerical errors can inject kinetic energy into a system. The instantaneous work done by a force and torque at time t is given by

$$\Delta KE = F^T v + T^T \omega \tag{2.3}$$

and the kinetic energy is given by

$$KE = \frac{1}{2} m v^T v + \frac{1}{2} \omega^T I \omega \tag{2.4}$$

where m is the mass of the body and I is the orientation dependent inertia tensor given by $I = ...$ Note that $I = R I_b R^T$ where I_b is a constant inertia tensor expressed in the body frame.

A more compact notation, motivated by the form of the kinetic energy, will be useful in one of the sections below. The **body linear velocity** is defined as $v_b = R^T v$; it is simply the velocity of the center of mass as viewed from a coordinate system coincident with the body frame.[3] Similarly, the **body angular velocity** is $\omega_b = R^T \omega$. Then the kinetic energy becomes

$$KE = \frac{1}{2} m v_b^T R^T R v_b + \frac{1}{2} \omega_b^T R^T I R \omega_b \tag{2.5}$$

[3]This is not the same as "moving with the body frame," in which case the center of mass velocity is always exactly zero.

and since R is orthogonal and $I = RI_bR^T$,

$$KE = \frac{1}{2}mv_b^T v_b + \frac{1}{2}\omega_b^T I_b \omega_b = \frac{1}{2}V_b^T M_b V_b \tag{2.6}$$

where $V_b = \begin{bmatrix} \omega_b \\ v_b \end{bmatrix}$ is the (total) **body velocity** of the rigid body and M_b is the $6x6$ **generalized inertia matrix** expressed in the body frame and given by[4]

$$M_b = \begin{bmatrix} I_b & 0 \\ 0 & m1_{3x3} \end{bmatrix} \tag{2.7}$$

Just as linear velocity describes an element of the tangent space of R^3, the body velocity describes an element of the tangent space of $SE(3)$, the configuration space of a rigid body. Such elements are known as **twists** [Murray et al., 1994]; if $g_{ab} = (x_{ab}, R_{ab}) \in SE(3)$ transforms points in frame b to a new frame a, then twists (tangent vectors) in frame b transform according to

$$V_a = \begin{bmatrix} R_{ab} & 0 \\ \hat{x}_{ab}R_{ab} & R_{ab} \end{bmatrix} V_b = Ad_{ab}V_b \tag{2.8}$$

where $\hat{x}$ denotes the antisymmetric dual of a vector x (see appendix); $\hat{a}b = a \times b$. The dual and cross product notations are used interchangeably in the sections below. $Ad_{g_{ab}}$ is called the **adjoint**, an invertible $6x6$ matrix.

In addition to the body frame and lab frame, a third frame intermediate between the two is useful. The **spatial** frame has the same orientation as the lab frame, but is located at the body frame origin, the center of mass. The $g = (0, R)$ maps from the body frame to the spatial frame, and $g = (x, 1_{3x3})$ maps from the spatial frame to the

[4] 1_{nxn} denotes the n by n identity matrix.

lab frame. The **spatial velocity** is given by

$$V_s = Ad_{(0,R)} V_b = \begin{bmatrix} R\omega_b \\ Rv_b \end{bmatrix} = \begin{bmatrix} \omega \\ v \end{bmatrix} \tag{2.9}$$

and the **spatial inertia** of the body is obtained from the transformation invariance of the kinetic energy:

$$\begin{aligned} KE &= \frac{1}{2} V_b^T M_b V_b = \frac{1}{2} (Ad_{(0,R)}^{-1} V_s)^T M_b (Ad_{(0,R)}^{-1} V_s) && (2.10) \\ &= \frac{1}{2} V_s^T M_s V_s && (2.11) \end{aligned}$$

where

$$\begin{aligned} M_s &= Ad_{(0,R)}^{-T} M_b Ad_{(0,R)}^{-1} && (2.12) \\ &= \begin{bmatrix} RI_bR^T & 0 \\ 0 & m1_{3x3} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & m1_{3x3} \end{bmatrix} && (2.13) \end{aligned}$$

2.2.2 Point constraints and deviation functions

The equations above can be used to simulate individual bodies acting under the influence of known time-varying forces and torques. The next step is to design forces and torques to implement specific constraints, and this section reviews the methods in [Barzel and Barr, 1988] used to constraint points on bodies. The terminology in this section comes from [Barzel, 1992] and [Barzel and Barr, 1988]. The first essential idea is that of a deviation function: a differentiable function $D(t)$ of the state of a collection of bodies such that $D = 0$ implies satisfaction of some constraint.[5] For example, a useful way to constrain a body involves "nailing" a point on the body to a point in lab coordinates – the **point-to-nail** constraint. Therefore a natural deviation function is $D(t) = p(t) - n$, where p is the **body point** at which the constraint applies, and n is a **nail point**, both in lab coordinates. In order to maintain D at

[5]Deviation functions are typically vector valued, so an equation like $D = 0$ has the obvious interpretation.

zero through the application of forces, a deviation function should be twice differentiable, and its second derivative $\ddot{D}$ should be a linear function of forces present in the system. These requirements lead to a tractable solution for the forces[6] Any twice differentiable function of the **configuration** of the system of bodies might serve as a valid deviation function. In a slight abuse of language, a deviation function can be considered synonymous with the constraint it embodies.

The **dimension** of a constraint is the dimension of the associated deviation function; for example the point-to-nail constraint is 3 dimensional. This paper exclusively considers 3 dimensional constraints which are functions of the locations of points on bodies – point constraints – as these are the most important and useful type of equality constraint. As the name suggests, point constraints can be enforced with forces acting at points on the constrained bodies. Given a collection of deviation functions D_i for point constraints, the goal is to find **constraint forces** that will bring D_i to zero for all i, and maintain $D_i = 0$ thereafter. The dynamic constraints approach provides a robust way of calculating these forces.

The example of a single rigid body and a single point-to-nail constraint will illustrate the method. Let point p be considered rigidly attached to a body, so that its motion is determined by that of the body. Then $p = Rb+x$, where b gives the constant body-frame coordinates of the constrained point; the time variation of p is due solely to that of (x, R). For a point attached to a box, the point-to-nail deviation function $D = p - n$, where n is a point fixed in the lab frame. Let F_c be a constraint force acting on the box at p, and let F, T denote any other forces and torques acting about the center of mass. Then the acceleration of p, and hence D, due to all forces and

[6]Note that this is slightly different than Barzel and Barr [1988], who look at forces and torques. For reasons that will become clear later, we consider only forces of constraint acting at points, with no real loss of generality.

torques is

$$\begin{aligned} \ddot{D} &= \ddot{p} && (2.14)\\ &= \ddot{x} + \ddot{R}b && (2.15)\\ &= \frac{1}{m}(F_c + F) + \frac{d(\omega \times R)}{dt}b && (2.16)\\ &= \frac{1}{m}(F_c + F) - (Rb) \times \frac{d\omega}{dt} + \omega \times \omega \times Rb && (2.17) \end{aligned}$$

The angular acceleration $\frac{d\omega}{dt}$ will depend on the total torque about the center of mass. F_c will contribute a torque of $(Rb) \times F_c$, and a simple calculation shows that $\frac{d\omega}{dt} = I^{-1}\hat{L}\omega + I^{-1}(Rb) \times F_c + I^{-1}T$ where, as stated above, T represents any torques on the body other than those due to F_c. Incorporating this into equation (2.17) is not difficult, giving (see appendix or [Barzel and Barr, 1988] for details)

$$\ddot{D} = AF_c + \beta \qquad (2.18)$$

where

$$A = \frac{1}{m}1_{3x3} - (\widehat{Rb})I^{-1}(\widehat{Rb}) \qquad (2.19)$$

and

$$\beta = \frac{1}{m}F - (\widehat{Rb})I^{-1}T - (\widehat{Rb})I^{-1}\hat{L}\omega + \omega \times \omega \times (Rb) \qquad (2.20)$$

A fundamental property of accelerations for mechanical systems is the linear dependence on force seen in (2.18), and any deviation function which depends statically on configuration will have an acceleration which is linear in force. The terms making up β in (2.20) consist of accelerations due to nonconstraint forces and torques, and centripetal/Coriolis accelerations arising from the rotational motion of the body.

2.2.3 Assembling the constraint equation

Consider an initial state for the body in which $(D, \dot{D}) = (0, 0)$. Then provided a solution exists and is unique, $\ddot{D} = AF_c + \beta = 0$ can be solved at any given time for the unknown constraint force. Applying F_c then guarantees $\ddot{D} = 0$ and the constraint will be maintained indefinitely. However, solutions to $AF_c + \beta = 0$ will always contain some numerical error, yielding a small residual $AF_c + \beta = \eta$. The application of F_c then corresponds to $\ddot{D} = \eta$, with the result that $(D, \dot{D})$ will drift away from $(0, 0)$ in a manner determined by the properties of the disturbance $\eta(t)$. More importantly, it the initial state for the body does *not* satisfy $(D, \dot{D}) = 0$, then solving $\ddot{D} = 0$ is not adequate to enforce the constraint. The effects of numerical disturbances and nonzero initial $(D, \dot{D})$ can be addressed with stabilizing feedback, the main innovation of the dynamic constraints approach.

The equations of motion dynamics for the rigid body implicitly define a second order **deviation dynamics**, (2.18) above. Then $(D, \dot{D}) = 0$ defines a **constraint surface**, a possibly time-varying subset of the 6-dimensional "deviation state space." Convergence to this surface can be achieved by choosing $\ddot{D}$ to yield asymptotically stable behavior. For example, $\ddot{D} + 2\lambda\dot{D} + \lambda^2 D = 0$ gives an *exponentially* stable deviation dynamics when $\lambda > 0$. The choice of second order, critically damped dynamics leads to well behaved relaxation to the constraint surface in the ideal case in which the constraint equation (2.18) can be solved exactly:

$$F_c = A^{-1}(-\beta - 2\lambda\dot{D} - \lambda^2 D) = A^{-1}(-B). \tag{2.21}$$

Numerical inaccuracies lead to $\ddot{D} + 2\lambda\dot{D} + \lambda^2 D = \eta(t)$. If η is bounded, it's easy to see that $(D, \dot{D})$ will converge to a neighborhood of the origin, the size of which is governed by the bound for η.

Multiple point-to-nail constraints are easily handled by forming a block-structured **constraint equation** whose blocks take the form of the matrix and vector A and B

in (2.21). Denote the different constraints by D_i, each acts at a point (in body frame coordinates) b_i and has an associated constraint force F_i. For N simultaneous constraints, let $D = \begin{bmatrix} D_1^T & D_2^T & \dots & D_N^T \end{bmatrix}^T$ be a $3N$ dimensional concatenated deviation vector, and similarly let $F = \begin{bmatrix} f_1^T & f_2^T & \dots & f_N^T \end{bmatrix}$ be a $3N$ dimensional concatenated vector of constraint forces. Then

$$\ddot{D} == AF + B \tag{2.22}$$

where A is a block-structured matrix with N by N blocks, each of which is $3x3$, and B is a $3N$ vector. Let A_{ij} be the ijth block of A (*not* the ijth entry, but a $3x3$ matrix). Then A_{ij} is relates the force applied at a point b_j to the acceleration observed at a point p_i. This relationship requires information about both constraints and the mutual body which they act upon. Specifically, if constraint i acts at b_i, and constraint j acts at b_j, then

$$A_{ij} = \frac{1}{m} 1_{3x3} - \widehat{(Rb_i)} I^{-1} \widehat{(Rb_j)} \tag{2.23}$$

is the formula for block A_{ij}. The vector B is block structured as well; each of its N 3-dimensional vector blocks is given by (2.20) and the stabilized constraint expression (2.21), to give

$$B_j = \beta = \frac{1}{m} F - \widehat{(Rb_j)} I^{-1} T - \widehat{(Rb_j)} I^{-1} \hat{L}\omega + \omega \times \omega \times (Rb_j) + 2\lambda \dot{D}_j + \lambda^2 D_j \tag{2.24}$$

for the block corresponding to constraint j.

The constraint force calculation boils down to the solution of a linear equation (2.22), which can be a computationally intensive task when many constraints are present. The matrix A may be sparse, in which case fairly efficient solution methods can be used. But it may also be singular, with both underdetermined and overdetermined cases as pointed out in [Barzel and Barr, 1988]. This fact is both a weakness and a strength of the dynamic constraints approach, and a later section will discuss it in the

context of articulated systems.

2.2.4 Multiple bodies, multiple constraints

The notation so far is specialized to the case of a single body acted on by a single type of point constraint, but the multiple-body multiple-constraint-type case follows directly. The form and dimensionality of (2.22) does not change; the blocks A_{ij} are still $3x3$ but they now depend on the configuration of *all* the bodies which both constraints i and j act upon. Typically point constraints act on one or two bodies; for example a point-to-point constraint acts on two bodies with a deviation function $D = p_a - p_b$ for points p_a on body a and p_b on body b respectively. Since $\ddot{D} = \ddot{p}_a - \ddot{p}_b$, the expressions (2.19) and (2.20) can again be used, with proper indexing and bookkeeping. Other versions of point constraints are possible; some are given in [Barzel and Barr, 1988] and others, such as point-to-line, -curve, and -surface,

Constraints on the velocities of points can be resolved with forces in a variant dynamic constraints approach as well. In this case, because of the first-order relationship between force and velocity, only $\dot{D}$, as opposed to $\ddot{D}$, is required to expose the linear force dependence of the deviation dynamics. For example, if $D = \dot{p} - v$ represents a deviation function for a point velocity constraint where v is constant, then $\dot{D} = \ddot{p} = AF + \beta$ as in the case of a point to nail constraint. Solving $AF_c + \beta = -\lambda D$ where $\lambda > 0$ then achieves a stabilized *first*-order deviation dynamics $\dot{D} + \lambda D = \eta(t)$, analogous to the stabilized dynamics for configuration constraints given above. Extensions to more general point velocity constraints are immediate: all that is required is that the deviation function be a differentiable function of the state (velocity and position) of the system.

From one perspective, the process of computing forces of constraint needed to achieve a desired acceleration can be thought as a type of acceleration constraint. The difference with point and velocity constraints, however, is that acceleration is not an

integrated quantity in a dynamic simulation. Rather than using constraint force calculations to *converge* to an acceleration value for which constraints are satisfied, setting a desired acceleration is simply a matter of solving the linear equation relating forces and accelerations as accurately as possible.

2.3 Examples

The formal ideas behind dynamic constraints are relatively straightforward to understand, but there are a number of observations that are not always obvious from the presentations in the computer graphics literature. This section underlines a few key points and concludes with a simple scheme for enhancing the robustness of dynamic constraints for articulated figures.

2.3.1 Standard behavior

Dynamic constraints force convergence to a constraint surface by cancelling any forces which affect the deviation function, and replacing the unconstrained dynamics with critically damped behavior. In overcoming other forces, dynamic constraints *inject energy* into the system: constraint forces do work as the constrained points move. Figure 3.4.1 illustrates this property for a point to nail constraint acting on the end a 1 k.g. cylinder. Because the constraint does not act through the center of mass, the resulting rotational movement generates centripetal and Coriolis forces as detailed in the formulas above. Figure 3.4.1a shows the constraint force magnitude as a function of time; note that after transients it approaches a constant value. Figure 3.4.1b shows the kinetic energy of the body, which increases as the deviation is reduced and also apparently approaches a constant value. Further 3.4.1c shows the magnitude of the deviation function and its derivative, displaying the expected critically damped stable behavior. Finally, 3.4.2 shows the residuals of the constraint equation solution – the disturbance $\eta(t)$ due to numerical error. All the simulations in this paper display the behavior of figure 3.4.1c and 3.4.2 unless otherwise noted.

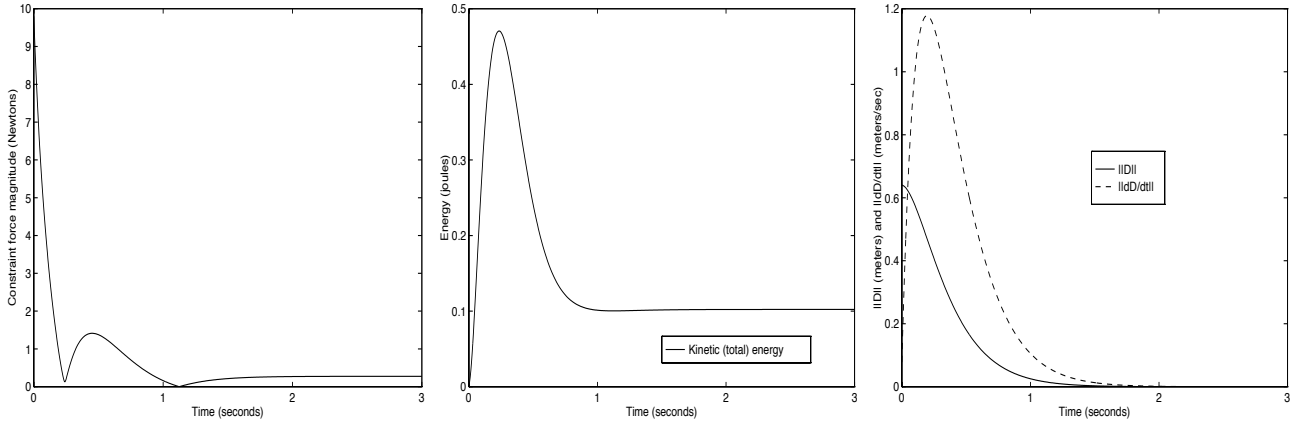


Figure 2.2: A point to nail constraint acting on a cylinder a. Constraint force magnitude b. Kinetic energy of the body. Note that in converging to the constraint surface, kinetic energy is "injected". c. Magnitude of the deviation and its derivative.

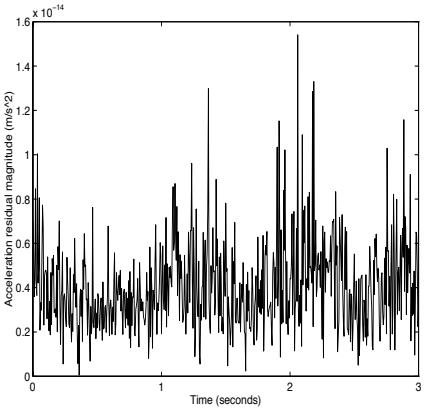


Figure 2.3: Residuals for solutions of the constraint equation

Figure 3.4.2 shows the same situation in the presence of gravity. For this case, the constraint force magnitude oscillates with the exchange of kinetic and gravitational potential energy; figure 3.4.2b shows the total energy approaching a constant. Finally, 3.4.2a and b show a similar situation when a spring (stiffness $10N/m$, damping $3\ N/m/s$) is attached to the cylinder. In each case, the initial conditions are identical, and the although the movement of the constrained point on the body is identically driven to the desired value, the complete motion still depends on the details of any other forces present.

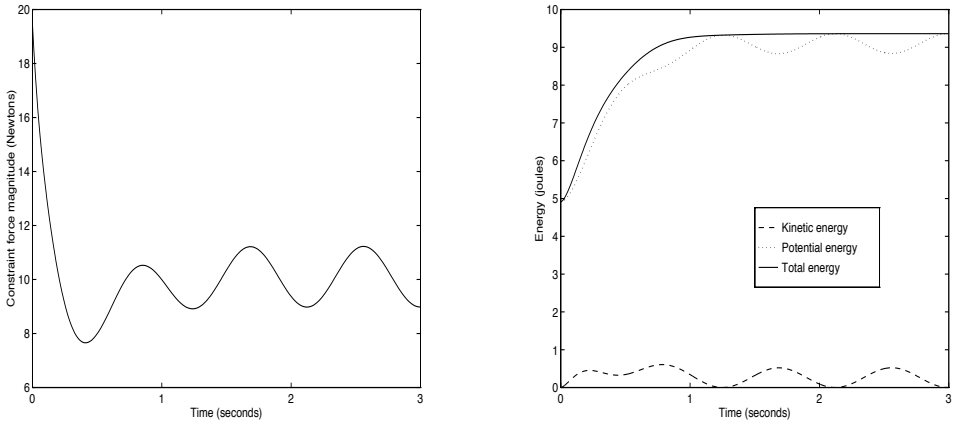


Figure 2.4: Point to nail constraint in the presence of gravity a. Constraint force magnitude b. Kinetic and potential energy.

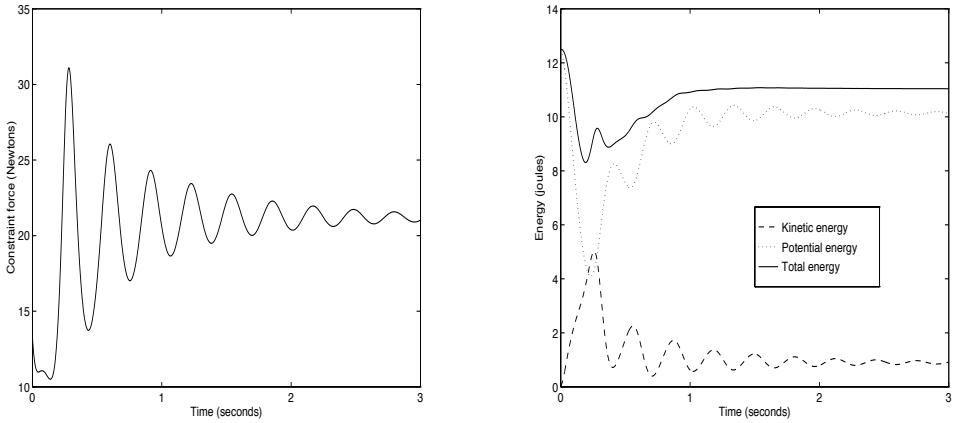


Figure 2.5: Point to nail constraint in with a spring attached to the cylinder a. Constraint force magnitude b. Kinetic and potential energy.

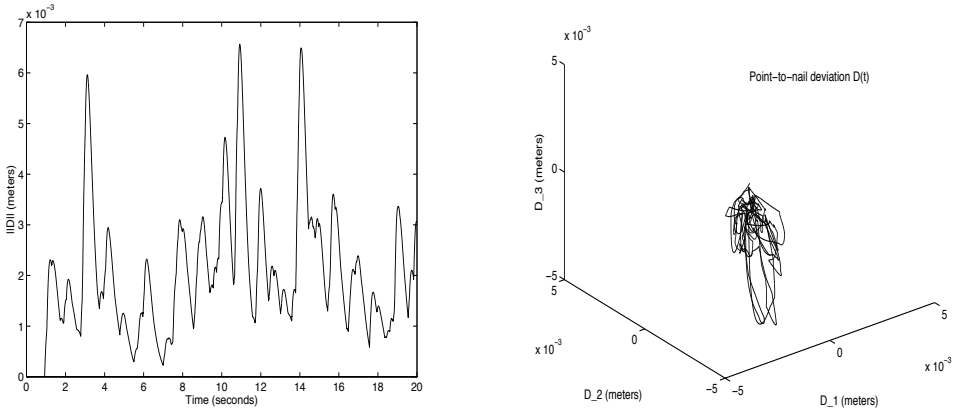


Figure 2.6: Behavior of a point constraint with $(D, \dot{D}) = 0$ initially, in the presence of gravity and no damping. a. $||D(t)||$ for a 20 second period b. Trajectory of $D(t)$ in 3-space for the same period as in a.

2.3.2 Constraints and work

When a simulation is initialized with $(D, \dot{D}) = 0$, in principle the constraint should do no work on the system at all. But since a point constraint cannot perfectly implement an abstract "nail" due to numerical errors, some movement of the constrained point is to be expected. One might expect (or hope) that the small size of the residuals would imply that the work done by a constraint which is initially satisfied would be "small" as well. Figure 3.4.2a shows the magnitude of the deviation function for a constrained cylinder under the influence of gravity *with zero damping* over a 20 second period. The maximum deviation over that time is certainly small, but is many orders of magnitude larger than the residuals in figure 3.4.2. Figure 3.4.2b shows a 3-D plot of the components of D as they vary in time; the deviation moves about in a small neighborhood of the origin.

The total energy of the system is shown in figure 3.4.2; there is a steady drift of energy due the action of the constraint force. Note that the increase in error is *not* easily attributable to numerical integration errors. It is true that standard numerical integration methods (such as the 4th-order Runge-Kutta method used here) are not guaranteed to conserve energy for mechanical systems. However, simulations involving rotational motion with no constraints, gravity or damping for the same object used

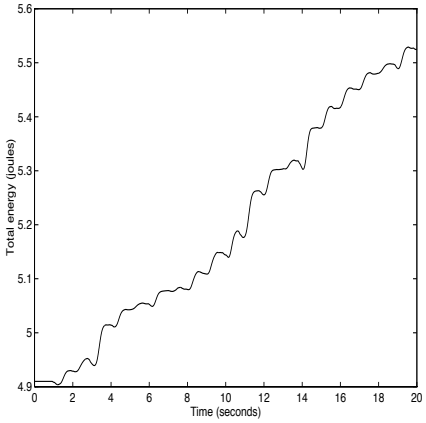


Figure 2.7: Total energy for the simulation of 3.4.2.

in figure 3.4.2 show that energy is in practice perfectly conserved. The energy gains of figure 3.4.2 represent a macroscopic injection of energy by the dynamic constraint, despite microscopic acceleration residuals at any given time.

As a result, some care must be taken in the use of a dynamic constraint as a model of a workless constraint; energy is clearly not conserved in frictionless, undamped settings. It is important to point out that undamped settings are rare, and that when there are forces such as friction, air and fluid resistance, etc., the energy contribution of a dynamic constraint may not be significant.

2.3.3 Penalty methods and control

Penalty methods ([Platt, 1989]) are a conceptually simpler way to model point constraints than the approach used here. These methods essentially insert a very stiff, perhaps nonlinear spring with rest length zero between points on bodies to be constrained. Any additional forces on the bodies may disturb the constrained points, and the spring will resist such disturbances. Adding damping is typically used to suppress ringing, and the methods are equivalent to *high gain* proportional-derivative (PD) "control" common in the robotics literature [Arimoto and Miyazaki, 1983]. Penalty methods are local, in the sense that the information needed to generate a constraint force depends only on the points associated directly with the constraint. In contrast,

dynamic constraints are global, requiring knowledge of *all* the forces and *all* the inertial information about a system. Penalty methods are much simpler to implement than dynamic constraints, and represent a common alternative; for example, the comprehensive simulation package *Newton* [Cremer, 1992] uses penalty methods to enforce constraints.

Although penalty methods and PD control are fundamentally useful, care must be taken when gains are very high to prevent instability. Like dynamic constraints, penalty methods inject energy into the system inadvertently, but the problem is much more severe. The discretization imposed by numerical integration makes this a difficult problem to overcome, especially if nonlinear gains are used. Further, the degree to which a point constraint is satisfied will vary depending on the additional forces present, causing potential problems with joint integrity. The dynamical constraints approach can be seen as addressing these issues by using global knowledge: simplicity is traded for accuracy. The dynamic constraints approach has a close relationship with control methods in the robotics literature known as feedback linearization [Murray et al., 1994] or computed torque [Spong and Vidyasagar, 1989], which also use complete and global knowledge of the structure and parameters of a system to impose a desired dynamics. Penalty methods and dynamic constraints are complementary simulation methods and they are dual to the range of control methods seen in the robotics literature. The next chapter discusses these control issues in more detail, using many of the concepts presented on the simulation side.

2.3.4 Singularity and pseudoinversion

For the simple examples presented so far, the solution of the linear constraint equation has been of no difficulty. For many systems with many constraints and linked bodies, efficient solutions of (2.22) can be found with standard methods, perhaps even exploiting sparsity or special structure of the constraint matrix. However, it is common to encounter situations where the constraint matrix has less than full rank, leading to

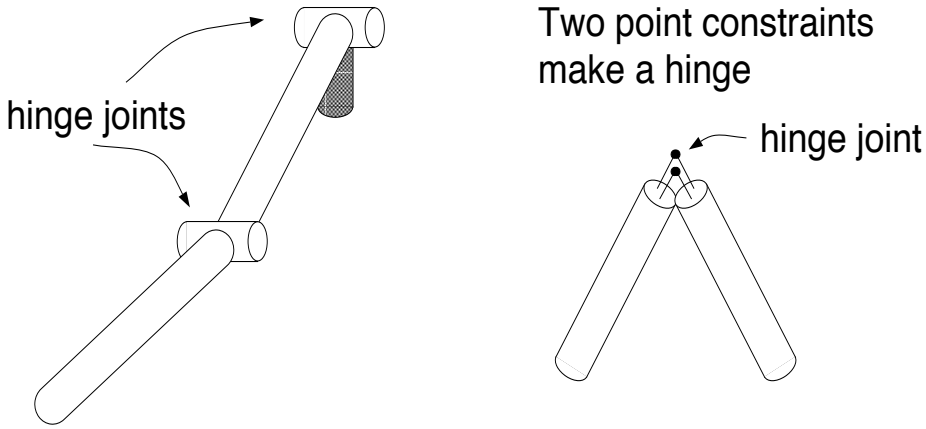


Figure 2.8: Hinge joints for articulated limbs

underdetermined or overdetermined constraint forces. One simple example of how this can occur comes from an application of point constraints to the construction of articulated limbs. Shoulder and hip joints can be easily modelled with point constraints, since these joints allow three rotational degrees of freedom and are mechanically similar to the idea of a point constraint. Elbow or knee joints, generally referred to as **hinge** joints, allow only one degree of freedom between bodies. The obvious way to implement a hinge joint in the dynamic constraints setting is to use two point constraints (figure 2.3.4).

Because a hinge joint removes 5 mechanical degrees of freedom, two point constraints used in this way will have underdetermined constraint forces. The singular value decomposition (SVD, [Press et al., 1990]) is a typical method for handling this situation, and is suggested in [Barzel and Barr, 1988] to give "reasonable" performance. By setting "small" singular values of the constraint matrix to zero before inverting to solve for the constraint forces, the SVD is intended to avoid excessive constraint forces in underdetermined directions. However, the definition of "small" can be problematic in a simulation environment where numerical values are intended to have physical meaning. Defining "zero" as a number close to machine tolerance may prevent problems during

the solution of the constraint forces, but the subsequent use of the solution as physical forces can wreak havoc with numerical integration. We have found the straightforward use of the SVD to be unacceptable, and instead use an iterative approach that can be interpreted as a (slightly) more complete model of a constraint between bodies.

Since point constraints are intended to model the actual connection between bodies, the constraint forces required are physical quantities. A more complete model, therefore, should associate a large but finite force magnitude with each point constraint, to represent the maximum force which can be exerted by a constraint. Since a real joint of an articulated limb has finite strength, so should a simulated joint. If constraint forces are calculated which exceed some maximum, an iterative procedure is invoked. The singular values of the constraint matrix are eliminated one by one, in ascending order, and the constraint force solution obtained at each step. If the constraints are underdetermined, and the joints are sufficiently strong, then at some point a feasible solution is obtained, giving small residuals. If the constraints are overdetermined, and/or the joints are not strong enough to exert force in the required directions, residuals will be "large." Each point constraint must also therefore include a bound on the maximum expected deviation. This bound will depend on the particular application, but it too has physical meaning and embodies the integrity of the joint. If a deviation bound is exceeded, then the constraint is considered "broken." Such a constraint can be removed from the constraint equation, which can then be solved again for the remaining constraint forces. Simulations in the next section will illustrate the usefulness of this simple scheme.

Note that another method for dealing with small singular values has found use in the robotics literature. The "damped least squares" method [Wampler and Liefer, 1988] simply replaces *all* inverse singular values $1/\mu$ with $\mu/(\mu^2 + \rho^2)$, where $\rho > 0$ is chosen in a task dependent way. The approach is elegant for control, in that solutions are accurate when singular values are "large," but become smoothly less so near singularities. The difficulty of clearly interpreting and choosing ρ, however, led us to

prefer a more functional criterion.

2.4 Articulated bodies

Point constraints give a tremendous amount of flexibility in constructing complicated figures, but they are limited by the relative inefficiency of having to solve a potentially ill-conditioned matrix each time the accelerations for the bodies in a system are required. For certain mechanisms, constraints remove most of the degrees of freedom of the component rigid bodies. For example, the planar arm shown in 2.3.4 can be constructed from 3 bodies for a total of 18 configurational degrees of freedom. The joints are modelled as hinges, each of which can be implemented with two point constraints. The final system has only 3 degrees of freedom, but its simulation requires integration of 18 nominal degrees of freedom, combined with solution of an 18 dimensional linear equation in order to determine constraint forces. Most of the effort for such a system goes towards the pseudoinversion of the constraint matrix, generating potentially large constraint forces that nonetheless do very little work. A more direct method for the simulation of such highly constrained systems would improve efficiency and numerical conditioning.

A more efficient approach entails the direct parameterization of the available degrees of freedom. For example, the angles between the links in figure 2.3.4 are a natural description of the configuration. As is typical in the robotics literature, the equations of motion for this articulated system can be written directly in terms of the joint angle parameterization. Rather than calculating the forces needed to keep the configuration of the system near a constraint surface defined by the hinge joints, this approach integrates *directly on* the surface. Forces transverse to the constraint surface are projected away rather than explicitly calculated and cancelled. As a result, no configurations for which the constraints are violated are ever encountered. The parameterization of the constraint surface typically defines a set of **generalized coordinates** in which unconstrained motion can occur.

Note that a generalized coordinates can be complementary with an approach based on explicit constraint calculations. This is because in certain situation, e.g. self-assembly of structures, configuration control, or simulations in which constraints can "break," movement on *and* off the constraint surface must be represented. (also, there are constraints that cannot be globally parameterized, e.g. self-motion manifolds). Further, even if more direct methods for calculating the dynamics of an articulated body can be used, one might like to use the articulated body as a primitive to create still more structures with the dynamic constraints approach. This section details the calculations needed for the case of open kinematic chains constructed with hinges (such as the arm in 2.8), and comments on the extension to other types of nonrigid body.

2.4.1 Projecting onto the available degrees of freedom

To begin with, note that the fundamental tools for the derivation of articulated body equations of motion are **velocity transformations**, maps which convert time derivatives in one coordinate system to time derivatives in another. Such transformations allow alternate expressions for the kinetic energy of a collection of bodies, and most of the structure of rigid body dynamics comes directly from the properties of kinetic energy. An example of two bodies connected by a hinge will illustrate all the key concepts. The notation of [Rodriguez and Kreutz-Delgado, 1992] is used throughout.

The **cm spatial velocity** of a moving body was given in section 3.2 above as a 6-vector $\begin{bmatrix} \omega \\ v \end{bmatrix}$ consisting of the angular and linear velocities as viewed from a frame located at the center of mass, and oriented with the lab frame. The spatial velocity about a point p is the transformation of the cm spatial velocity to a frame translated by $b = p - x_{cm}$ (figure ab1). This transformation is given by

$$V_p = Ad_{(-b,1_{3x3})} V_{cm} = \begin{bmatrix} 1_{3x3} & 0 \\ -\hat{b} & 1_{3x3} \end{bmatrix} V_{cm} = \phi^T(b) V_{cm} \tag{2.25}$$

where $\phi(b) \equiv \begin{bmatrix} 1_{3x3} & \hat{b} \\ 0 & 1_{3x3} \end{bmatrix}$ is defined as a **spatial Jacobian**. The spatial Jacobian transforms velocities between frames that are translates of one another: $V_{p1} = \phi(p_1 - p_2)^T V_{p2}$. Now suppose the point p represents the location of a hinge between the body and a fixed base with hinge axis defined by a unit vector h_1 (figure ab2). Then the body is constrained to rotate around the hinge and its configuration is parameterized by a scalar angle θ_1. The spatial velocity of the body at point p is given by $\begin{bmatrix} h_1 \\ o \end{bmatrix} \dot{\theta}_1$ where $\dot{\theta}^1$ is the scalar velocity about the hinge. A **hinge Jacobian** $H_1^T \equiv \begin{bmatrix} h_1 \\ 0 \end{bmatrix}$ maps from hinge coordinate velocity to spatial velocity of the constrained body. If the hinge constrains the *relative* motion of two bodies (figure ab3), the $H_1\dot{\theta}_1$ gives the spatial velocity of body 1 *relative* to body 2. If V_2 is the cm spatial velocity of body 2 at a different point p_2 (figure ab4), then the total velocity of body 1 at p_1 is found with both the hinge and spatial Jacobians:

$$V_1 = H_1\dot{\theta}_1 + \phi(p_1 - p_2)^T V_2 \tag{2.26}$$

The kinetic energy of the combined system is $\frac{1}{2}V_{cm1}^T M_1 V_{cm1} + \frac{1}{2}V_{cm2}^T M_2 V_{cm2}$, with M_i given by formula (2.12) in section 3.2. The first term can be rewritten to account for the constraint:

$$\begin{aligned} KE_1 &= \frac{1}{2}(\phi(x_{cm1} - p_1)^T V_{p1})^T M_1 (\phi(x_{cm1} - p_1)^T V_{p1}) \\ &= \frac{1}{2}V_{p1}^T (\phi(x_{cm1} - p_1) M_1 \phi(x_{cm1} - p_1)^T) V_{p1} \\ &= \frac{1}{2}\dot{\theta}_1 H_1 \phi(x_{cm1} - p_1) M_1 \phi(x_{cm1} - p_1)^T H_1^T \dot{\theta}_1 \\ &\quad + \frac{1}{2}V_{p2}^T \phi(p_1 - p_2)\phi(x_{cm1} - p_1) M_1 \phi(x_{cm1} - p_1)^T \phi(p_1 - p_2)^T V_{p2} \quad (2.27) \\ & \quad (2.28) \end{aligned}$$

and since [Rodriguez and Kreutz-Delgado, 1992] the spatial Jacobian has the property $\phi(a-b)^T\phi(b-c)^T = \phi(a-c)^T$ (which follows directly from the definition), the

total kinetic energy becomes

$$KE = \frac{1}{2}\dot{\theta}_1[H_1\phi(x_{cm1}-p_1)M_1\phi(x_{cm1}-p_1)^T H_1^T]\dot{\theta}_1 + \frac{1}{2}V_{p2}^T[\phi(x_{cm1}-p_2)M_1\phi(x_{cm1}-p_2)^T + M_2]V_{p2} \tag{2.29}$$

where we've taken $V_{p2} = V_{cm2}$. The first bracketed term gives the inertia associated with the hinge joint, and is a scalar. The second bracketed term gives the inertia associated with the degrees of freedom of the second body. This has two parts: one due to the second body alone (M_2), and another due to the attachment of the first body ($\phi M_1 \phi^T$). Combining these allows the definition of a generalized inertia matrix for the two-body system:

$$KE = \frac{1}{2}\begin{bmatrix}\dot{\theta}_1 \\ V_2\end{bmatrix}^T \begin{bmatrix}H_1\phi_1 M_1 \phi_1^T H_1^T & 0 \\ 0 & \phi_2 M_1 \phi_2^T + M_2\end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ V_2\end{bmatrix} \tag{2.30}$$

$$= \frac{1}{2}\begin{bmatrix}\dot{\theta}_1 \\ V_2\end{bmatrix}^T \mathcal{M}\begin{bmatrix}\dot{\theta}_1 \\ V_2\end{bmatrix} \tag{2.31}$$

where $\phi_1 \equiv \phi(x_{cm1} - p_1)$, similar for ϕ_2. The matrix $\mathcal{M}$ is a $7x7$ symmetric positive definite matrix; with $\mathcal{M}$ in hand, equations of motion can be determined by classical means [Spong and Vidyasagar, 1989]. (Note , not by Euler Lagrange, since these aren't unconstrained coords.) Note that the forces of constraint which create the hinges are entirely implicit, since θ_1 is used directly to parameterize the single degree of freedom of the hinge. Note also that the Jacobians (spatial and hinge) feature prominently in the form of $\mathcal{M}$. The spatial notation introduced by [Rodriguez and Kreutz-Delgado, 1992] can be used to form the generalized inertia matrix for a variety of mechanisms, which in turn can be used to create equations of motion expressed directly in the available degrees of freedom. Mechanisms consisting of bodies connected by hinges to a fixed base give a particularly compact notation, and can be used to illustrate the extension of the dynamic constraints approach to articulated bodies.

2.4.2 Equations of motion in generalized coordinates

Following Rodriguez' notation, figure ab5 shows a sequence of bodies numbered from N to 1, attached by hinges with hinge axes h_k, and locations p_k $k = 1...N$. Each link has an individual cm spatial inertia M_{cmk}, which when expressed at joint location p_k is $M_k = \phi(x_{cmk} - p_k) M_{cmk} \phi(x_{cmk} - p_k)^T$ or

$$M_k = \begin{bmatrix} R_k I_{bk} R_k^T - m_k \hat{b}_k^2 & m_k \hat{b}_k \\ -m_k \hat{b}_k & m 1_{3x3} \end{bmatrix} \tag{2.32}$$

where $b_k = x_{cmk} - p_k$, and R_k is the lab orientation of link k. Each joint is parameterized by an angle θ_k. Analogous to the example with two bodies, the hinge Jacobians are defined as $H_k^T = \begin{bmatrix} H_k \\ 0 \end{bmatrix}$, and the spatial Jacobians are $\phi_{jk} \equiv \phi(p_k - p_j)$. Then if $\theta \equiv \begin{bmatrix} \theta_1 & \theta_2 & ...\theta_N \end{bmatrix}^T$, and if H^T is the $6N$ by N block diagonal matrix $H^T = diag(H_1^T, ..., H_N^T)$, M is the $6N$ by $6N$ block diagonal matrix $diag(M_1, ...M_k)$, and Φ is the $6N$ by $6N$ matrix defined by

$$\Phi \equiv \begin{bmatrix} 1_{6x6} & 0 & 0 & ... & 0 \\ \phi_{21} & 1_{6x6} & 0 & ... & 0 \\ \phi 31 & \phi_{32} & 1_{6x6} & ... & 0 \\ ... & ... & ... & ... & ... \\ \phi N1 & \phi_{N2} & \phi_{N3} & ... & 1_{6x6} \end{bmatrix} \tag{2.33}$$

then [Rodriguez and Kreutz-Delgado, 1992] the kinetic energy of the system of bodies is given by

$$KE = \frac{1}{2} \dot{\theta} H \Phi M \Phi^T H^T \dot{\theta} = \frac{1}{2} \dot{\theta} \mathcal{M} \dot{\theta} \tag{2.34}$$

Because the joint angles θ are generalized coordinates allowing unconstrained motion, equations of motion are obtained with the Euler-Lagrange equations [Spong and

Vidyasagar, 1989], giving

$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta})\dot{\theta} = \mathcal{H}\mathcal{F} = \tau \tag{2.35}$$

where $F = \begin{bmatrix} F_1^T & F_2^T & ... & F_N^T \end{bmatrix}^T$ is a $6N$ vector of spatial forces which act on the links, e.g. gravity, simulated actuators, dynamic constraints. The N-vector τ denotes **generalized forces**, the projection of external forces (via H) onto the directions of allowable motion. $C(\theta, \dot{\theta})$ is a matrix of Coriolis and centripetal terms stemming from the time derivative of $\mathcal{M}$. The N by N matrix $\mathcal{M}$ is symmetric and positive definite, from the properties of kinetic energy. The factorization in 2.34 is simply a compact notation for an iterative algorithm for forming $\mathcal{M}$. Alternate factorizations lead to efficient inversion techniques yielding the joint accelerations corresponding to a given state of the manipulator and particular external forces; theory and algorithms for such factorizations have been developed by Rodriguez and his colleagues [Rodriguez and Kreutz-Delgado, 1992].

The important point is that equation 2.35 allows the forward simulation of hinged mechanisms without calculation of constraint forces. The locations and orientations of the individual bodies are calculated from the hinge angles, hinge axes, and hinge locations. However, it is useful to be able to *further* constrain this kind of system with additional point constraints, so as to allow control and modelling of still more complicated structures.

2.4.3 Point constraints

The example of a point-to-nail constraint for systems obeying 2.35 will clarify the calculations needed. First consider the effect of a point force F applied at a location x, that is attached to link k. The instantaneous work done by such a force is $F^T\dot{x}$, where $\dot{x}$ is the velocity of the point of application. If the system is moving with joint velocity $\dot{\theta}$, then the spatial velocity of link k is $V_k = \sum_{i=k}^{N} \phi^T(i,k) H_i^T \dot{\theta}_i$ since the base

is fixed. For a compact form, define

$$B(x,k) = \begin{bmatrix} 0 \\ \cdots \\ \phi(x - p_k) \\ \cdot\cdot \\ 0 \end{bmatrix} \tag{2.36}$$

as a block structured $6N$ by 6 matrix with a single nonzero $6x6$ block in the kth position. Finally define $C = \begin{bmatrix} 0 & 1_{3x3} \end{bmatrix}$ and Φ, H as above. Then

$$\dot{x} = CB(x,k)^T \Phi^T H^T \dot{\theta} = J(x,k)\dot{\theta} \tag{2.37}$$

where $J(x,k) \equiv CB(x,k)^T \Phi^T H^T$ is a $3xN$ **point Jacobian**. The point Jacobian maps internal velocities to the linear velocity of a point attached to a link. The idea of virtual work allows the translation of a point force into an equivalent set of generalized forces: $\tau^T \dot{\theta} = F^T \dot{x} = (J^T F)^T \dot{\theta}$ so that $\tau = J^T F$ is the generalized force that does the same work as the point force F. This relationship gives the internal effect of a point force as might arise from a dynamic constraint. For a set of such constraint forces indexed by j, acting at points x_j on links k_j, the total generalized force is $\sum_j J(x_j, k_j)^T F_j = \sum_j J_j^T F_j$ for short.

For constraint i, the deviation function is given by $D_i = x_i - n_i$, so that $\dot{D}_i = J_i \dot{\theta}$ and $\ddot{D}_i = J_i \ddot{\theta} + \dot{J}_i \dot{\theta}$. Substituting the dynamics (2.35) gives

$$\ddot{D}_i = J_i \mathcal{M}^{-\infty}(\tau_{\lfloor} - \mathcal{C}\dot{\theta}) + \mathcal{J}_{\rangle} \mathcal{M}^{-\infty} \sum_{|} \mathcal{J}_{|}^{\mathcal{T}} \mathcal{F}_{|} + \dot{\mathcal{J}}_{\rangle} \dot{\theta} \tag{2.38}$$

where τ_{nc} are generalized forces due to nonconstraint sources such as gravity, simulated actuators, etc. Rearranging,

$$\ddot{D}_i = \sum_j \underbrace{[J_i \mathcal{M}^{-\infty} \mathcal{J}_{|}^{\mathcal{T}}]}_{\text{Block } ij} F_j + \underbrace{[J_i \mathcal{M}^{-\infty} (\tau_{\backslash\rfloor} - \mathcal{C}\dot{\theta} - \mathcal{G}) + \dot{\mathcal{J}}_{\rangle}\dot{\theta}]}_{\beta_j} \tag{2.39}$$

$$\tag{2.40}$$

which, as the bracketed terms suggest, can be rewritten in block structured form:

$$\ddot{D}_i = \left[J_i \mathcal{M}^{-\infty} \mathcal{J}_{\infty}^{\mathcal{T}} \quad J_i \mathcal{M}^{-\infty} \mathcal{J}_{\in}^{\mathcal{T}} \quad ... \quad J_i \mathcal{M}^{-\infty} \mathcal{J}_{\backslash}^{\mathcal{T}} \right] F_c + \beta_i \tag{2.41}$$

where $F_c = \left[F_{c1}^T \quad F_{c2}^T \quad ... \quad F_{cn}^T \right]^T$, is the concatenated vector of constraint forces.

The notation in (2.41) assumes that all constraints are point-to-nail and act on the same manipulator. But as with point constraints for individual bodies, a matrix block i, j will be zero if constraints i and j do not act on a common body. Note also that the blocks of the deviation equation (2.41) all include the inertia inverse $\mathcal{M}^{-\infty}$, which can be efficiently computed with iterative methods.

The deviation equation (2.41) indicates how a point constraint on an articulated body contributes a 3 by $3K$ row of the full constraint matrix for a system with K point constraints. The notation is specialized for point-to-nail constraints acting on a single manipulator, but the general case proceeds precisely as with point constraints for individual bodies. Note that $\mathcal{M}^{-\infty}$, which is the difficult part of the calculations needed to form (2.41), is the same for each block and should obviously be found only once for a given configuration. As mentioned above, Rodriguez gives $O(N)$ methods for determining this inverse. Because $\mathcal{M}$ can be defined for articulated bodies with a floating base (as in the two-body example of the previous section) or with a tree-structured topology, the form of (2.41) remains the same for these cases. Additional bookkeeping is needed however, to properly define the point Jacobians, but this bookkeeping is straightforward.

Comparing 2.41 to the corresponding equation for individual bodies lets us summarize the operations needed to solve for point constraint forces. This promotes modularity in the primitives of a dynamic constraints modelling system. If an object can present the following interface routines to the simulator, then it can be included in the dynamic constraints framework:

- $\mathsf{apply-point-force}$: Take a force acting at a point and determine its equivalent in internal forces.
- $\mathsf{force-to-accel}$: Compute a matrix operator mapping from a point force at one location to the acceleration induced at another location.
- $\mathsf{accel-from-force}$: Compute the vector acceleration at a location due to internal forces.
- $\mathsf{accel-from-velocity}$: Compute the vector acceleration at a location due to the velocities of internal coordinates. This is typically important when rotational motion is present.

By keeping the details of these calculations internal, they can often be made highly efficient using methods specialized to a particular internal parameterization. The recursive approaches of [Rodriguez and Kreutz-Delgado, 1992], as well as the symbolic approaches of [Armstrong et al., 1985] offer efficient alternatives with different advantages and disadvantages. Related work by [Pfarner, 1996] in particular studies methods applicable to structures with kinematic loops.

To summarize, by moving to more complicated primitives than a single rigid body, the dynamic constraints approach can take advantage of specialized methods for producing equations of motion. By identifying the operations needed to assemble the constraint equations for a collection of point constraints, and detailing a specific example for open-chain mechanisms, the sections above extend the modelling framework introduced in [Barzel and Barr, 1988].

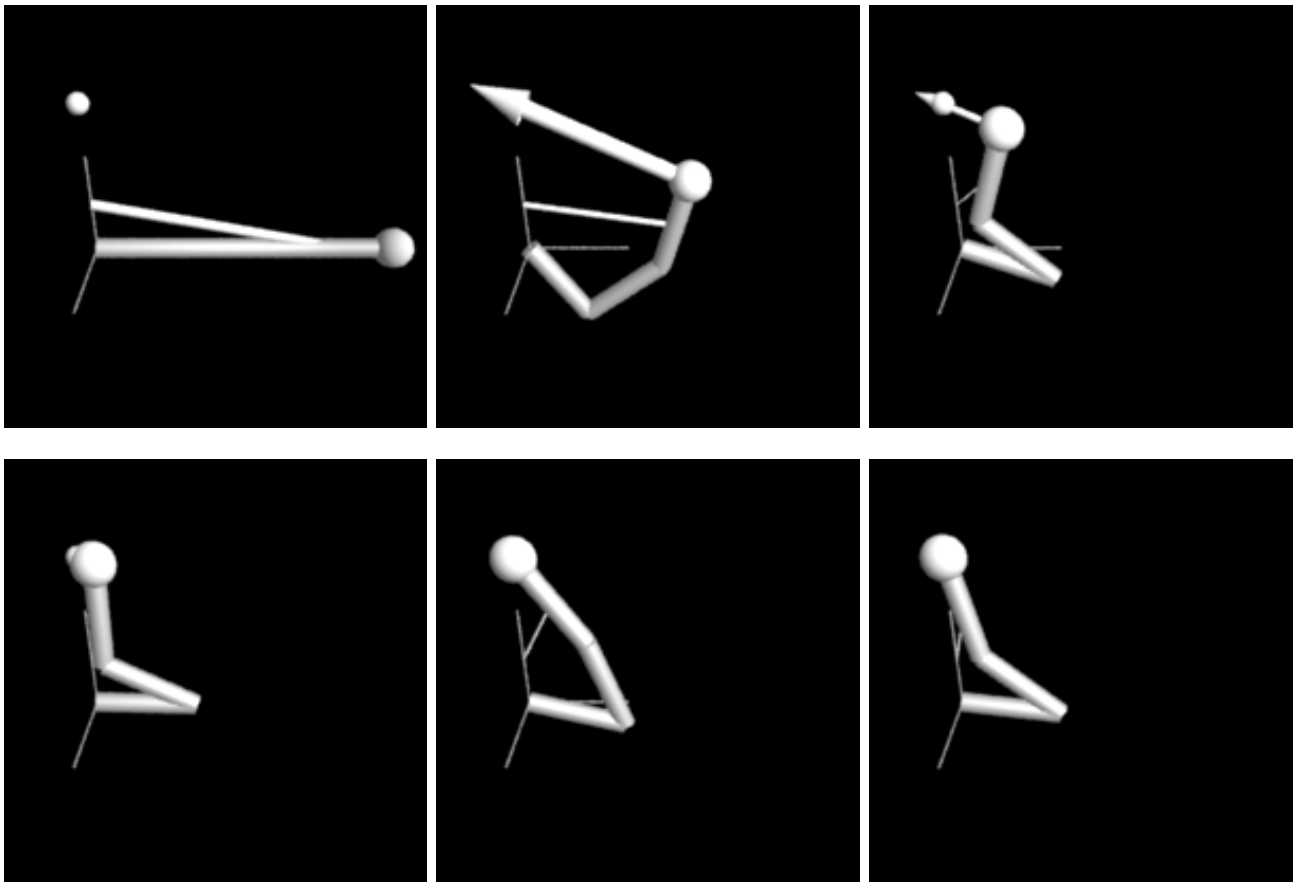


Figure 2.9: A point to nail constraint acting on a three-link arm. See text for details.

2.5 Examples

A simple example illustrates the combination of point constraints with an articulated limb. Figure2.9 shows a sequence of frames from a simulation of a three-link arm whose internal coordinates are the hinge angles. The frames are spaced by 1/3 second, and in addition to gravity the arm is disturbed by a spring attached to the distal link; the spring is denoted as a straight line joining the arm and the z-axis where the spring is anchored. A point-to-nail constraint acts at the center of the spherical tip, and the nail is denoted by the small sphere located on the z-axis. The velocity of the tip, which is the same as the derivative of the deviation function, is denoted by an arrow scaled proportionally to the magnitude of the velocity. The individual links of the arm are 1 meter long, 1 kg. cylinders.

Figure 2.5 shows the performance of the point constraint. The constraint force magnitudes are shown in 2.5a, along with the total kinetic energy and damped, exponentially stable behavior in 2.5b and c. This example shows an initial condition at a singularity of the arm; instantaneous motion in the direction demanded by the constraint is not possible. The constraint forces shown in 2.5a reflect this at the earliest times of the simulation with constraint forces that vary rapidly and extend far off the range of that plot. Figure 2.5a shows an expanded view of the earliest steps of the simulation. Note that the vertical axis of this plot is on a log scale, showing the tremendous increase in the constraint force magnitude near the singularity at full extension.

Figure 2.5b clarifies what is happening; this figure represents the residuals of the constraint equation. Initially, the constraint *cannot* be satisfied, and the singular direction is discarded with the SVD in the constraint force calculation. As the arm slowly moves away from the singularity, an abrupt transition occurs when the constraint *can* be satisfied with forces below the bounds allowed for this point constraint. The bounds for this example have been set to high values (200,000 N) for illustrative purposes. Note that naive use of the SVD would lead to constraint forces which destabilize the numerical integration for this example. Figure 2.11c shows the singular values for the constraint matrix as a function of time. Note that initially two of the values are near zero; the only allowable direction of motion at this time is tangential to the fully extended arm in the first frame of 2.5. Figure 2.5a and b show similar diagrams for a three-link elbow manipulator as in figure 2.3.4 above. In this case, the time variation of the manipulator is seen to bring the system into and out of singularity; this is reflected in the constraint force magnitude at left and the singular values shown at right.

2.6 Conclusions

We have presented a dynamical simulation environment with the ability to simulate individual rigid bodies, robustly enforce point constraints via explicit calculation of

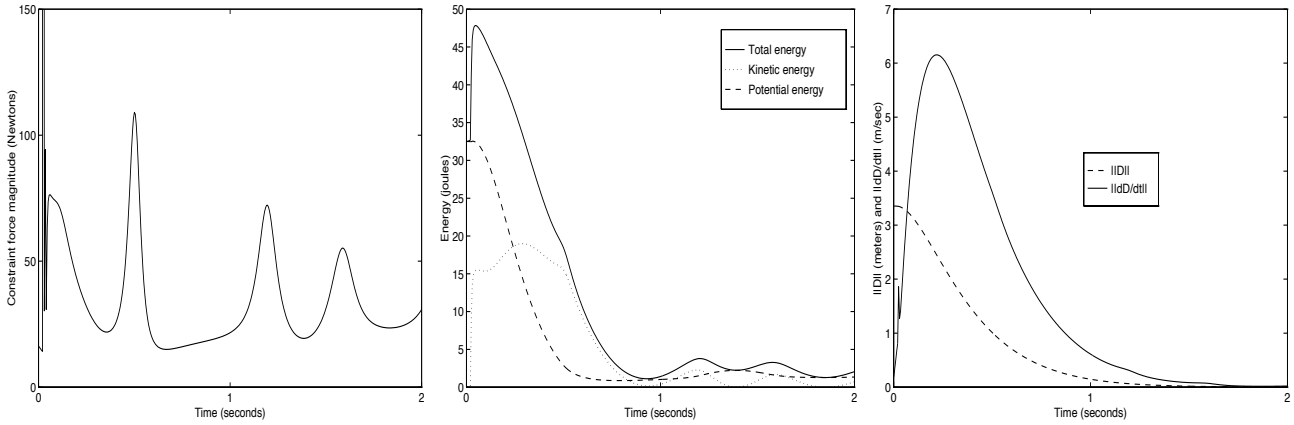


Figure 2.10: A point to nail constraint acting on a three-link arm. a. Constraint force magnitudes, clipped at 150 Newtons. See figure 2.5 for complete view. b. Kinetic and potential energy c. Deviation and its derivative.

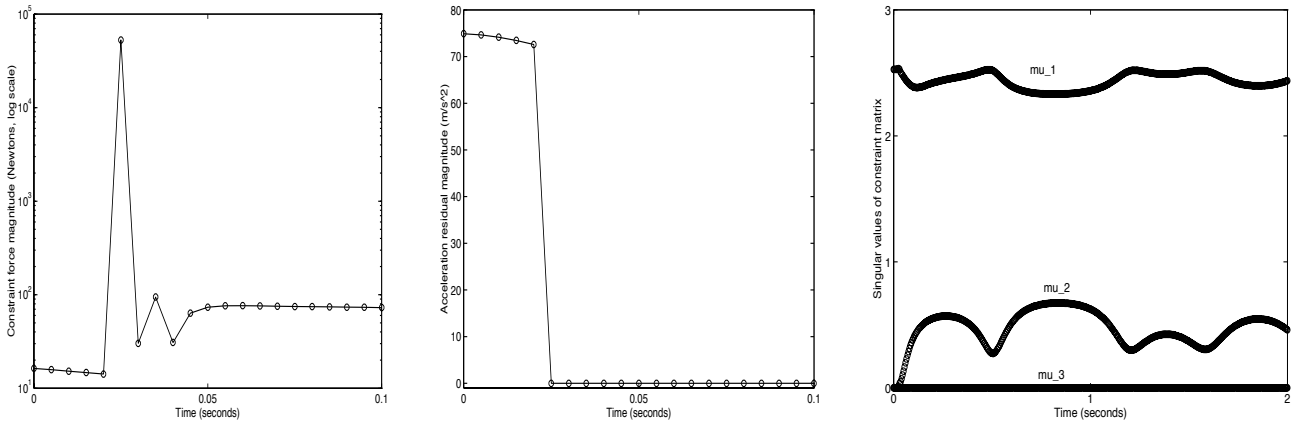


Figure 2.11: a. Complete view of constraint forces, semilog scale b. Acceleration residuals. a. Singular values of the constraint matrix for the arm in figure 2.5

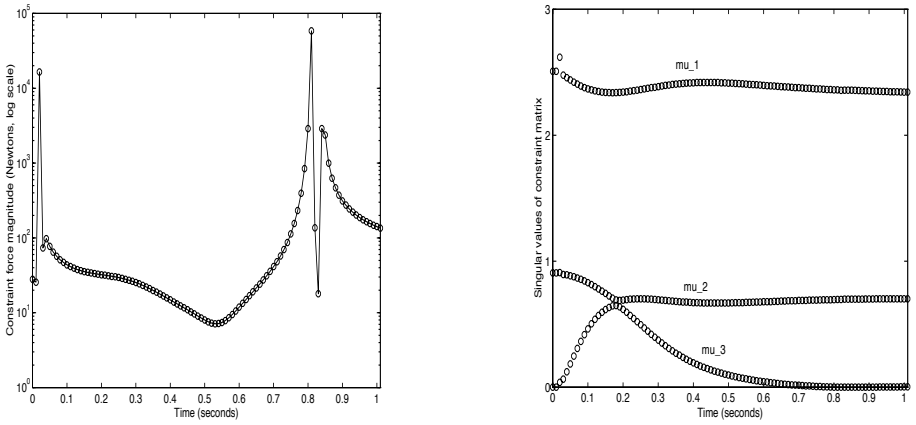


Figure 2.12: a. Constraint forces for another example of a constrained arm (not shown). b. Singular values of the constraint matrix.

constraint forces, and include articulated bodies which can be constrained as well.

A number of important lessons have been learned in the course of implementing the system described in the sections above. A large scale dynamical simulation is "merely" a model of a small set of physical phenomena. It's important to remember that every single concept employed in such a model is an abstraction, starting with the very idea of the configuration of a rigid body and continuing to the ideas of static and dynamic friction. Unfortunately, the criteria such abstractions in the context of simulation are still evolving, and the differences between current research projects reflect differences of opinion as to what is "essential." For the approach described here, we can list two of the priorities that guided decisions:

- **Extended simulations and long term robustness**. Our ultimate goal for dynamic simulation is the ability to test active, adaptive sensing and control strategies. Most dynamic simulations are measured in (simulated) seconds; to eventually simulate for minutes, hours, or days will require an absolute faith in the stability of the simulated world over such time scales. Therefore the dynamic constraints paradigm was the ideal foundation on which to build.

- **Efficiency for important special cases**. The flexibility of assembling articulated figures with point constraints is useful, but the method cannot compete with the speed and efficiency of carefully derived equations of motion which avoid the process of allowing, then constraining away degrees of freedom. Therefore we decided that the two methods should be able to easily coexist.

Here are some lessons learned:

- Numerical integration will introduce disturbance terms into **any** calculation you attempt. It's essential to identify calculations which are unstable or marginally stable, since such calculations can cause a simulation to explode unexpectedly.

- Assume in advance that *any* matrix has the potential to be singular. Formal guarantees of nonsingularity can be meaningless in the (inevitable) presence of noise. The singular value decomposition, for example, is an essential tool. $O(n^3)$

calculations are expensive, so gratuitous use of the SVD must be avoided. But we judge stability to be more important than any other consideration.

- As many parameters should be adaptive as possible. The heterogeneity of a dynamic simulation environment means that you cannot always anticipate proper values for scalings, tolerances, and step sizes.

- Energy is a unifying concept: keep track of all possible energy sources in any simulation. It's important to be able to account for *all* changes in energy that occur.

2.7 Appendix

This appendix contains some background information needed for the discussion above.

Cross products. The cross product, or vector product is an operation on two vectors that returns a third. The cross product is typically defined as (hairy definition here). The operation is linear, and therefore has a matrix representation that is useful for manipulation. For two vectors a and b, $a \times b = S(a)b$, where $S(a)$ is a $3x3$ matrix given by

$$\begin{pmatrix} 0 & -a_1 & a_2 \\ a_1 & 0 & -a_3 \\ -a_2 & a_3 & 0 \end{pmatrix} \tag{2.42}$$

$S(a)$ is skew-symmetric, i.e. $S = -S^T$. This property implies that the real parts of the eigenvalues of S are zero, and the imaginary parts come in complex-conjugate pairs (which share an eigenvector?), so that $x^T S x = 0$ for any x. Thus the standard properties of the cross product, e.g. $a \times b = -b \times a$ can be quickly recalled through the skew-symmetry property of S. We also write $\hat{a}$ as a synonym for $S(a)$.

Quaternion multiplication. A quaternion is represented by 4 numbers, a scalar

part, and a (3-dimensional) vector part, so $q = (s, v)$. The product of two quaternions is given by the multiplication operator

$$q_1 * q_2 = (s_1 s_2 - v_1^T v_2, s_1 v_2 + s_2 v_1 - v_1 \times v_2) \tag{2.43}$$

For a unit quaternion, the scalar part and vector part are interpretable as a rotation through an angle θ about a unit axis a where $s = cos(\theta/2)$ and $v = sin(\theta/2)a$. The infinitesimal rotation represented by angular velocity is represented by a quaternion $\bar{\omega} = (0, \omega)$. The differential equation for the evolution of $q(t)$ for angular velocity $\omega(t)$ given in 2.1 is interpreted as

$$\frac{dq}{dt} = \frac{1}{2}\bar{\omega} * q = \frac{1}{2}\begin{pmatrix} -\omega^T v \\ s\omega + \omega \times v \end{pmatrix} \tag{2.44}$$

where $q = (s, v)$.

Integrating orientation Euler integration of an object spinning with constant angular velocity is in fact unstable. The discrete iteration corresponds to $R(t + \Delta) = R(t) + \Delta\omega \times R(t) = R + \Delta S R = (1_{3x3} + \Delta S)R = AR$ where $S(\omega)$ is a **skew-symmetric** matrix, the **dual** of ω, and the matrix A can have maximum eigenvalue magnitude slightly greater than 1, implying instability. The problem is less severe, but nonetheless present for quaternions.

Torque dependence of $\ddot{p}$. We left off in equation (2.17) at

$$\ddot{D} = \frac{1}{m}(F_c + F) - (Rb) \times \frac{d\omega}{dt} + \omega \times \omega \times Rb \tag{2.45}$$

If b_i are the points of application *in body coordinates* of the constraint forces, then

$$\begin{aligned}
\frac{d\omega}{dt} &= \frac{dI^{-1}L}{dt} && (2.46)\\
&= I^{-1}\frac{dL}{dt} + \frac{dI^{-1}}{dt}L && (2.47)\\
&= I^{-1}T_{total} + (\frac{dR}{dt}I_b^{-1}R^T + RI_b^{-1}\frac{dR}{dt}^T)L && (2.48)\\
&= I^{-1}T_{total} + ((\hat{\omega}R)I_b^{-1}R^T + RI_b^{-1}(\hat{\omega}R)^T)L && (2.49)\\
&= I^{-1}T_{total} + \hat{\omega}I^{-1}L + I^{-1}\hat{\omega}^T L && (2.50)\\
&= I^{-1}T_{total} + \hat{\omega}\omega + I^{-1}\hat{L}\omega && (2.51)\\
&= I^{-1}T_{total} + I^{-1}\hat{L}\omega && (2.52)\\
& && (2.53)
\end{aligned}$$

where $T_{total} = T_{ext} + T_c = T_{ext} + \sum_i (Rb_i) \times F_i$

Virtual work. The principle of **virtual work** (Spong and Vidyasagar [1989]) is an expression of the fact that energy does not depend on the coordinates in which it is expressed. The work done by a force F acting at x is $F^T\dot{x}$; alternately, the work done by a generalized force is $\tau^T\dot{\theta}$. Setting these equal, and noting that $\dot{x} = J(\theta)\dot{\theta}$ gives $(J\dot{\theta})^T F = \dot{\theta}^T\tau_F$, or $\dot{\theta}^T(J^T F) = \dot{\theta}^T\tau_F$. Since $\dot{\theta}$ is arbitrary, this implies $J^T F = \tau_F$.

2.8 Programming constructs

This section is an appendix giving some of the specific routines and conventions needed for programming.

Rigid bodies and articulated bodies share a common component, the **link**. Rigid bodies have one link, and articulated bodies have more than one. Each link is composed of a number of segments; these are geometric primitives which define the geometric, material, and inertial properties of that link. A segment is represented by the class shape. There is no specific class for links, we use a pointerlist to store the shapes

defining each segment in a link. One such list is needed for each link.

Let $g = (R, x)$ be a rigid body configuration (or transformation). Each segment keeps a configuration g_{seg} with respect to the link it is attached to. This configuration is constant, and for multi-link bodies, is defined with respect to the lab frame translated to the link origin, the *link frame*. For single-link bodies, it is with respect to the principle frame located at the center of mass. In addition, each segment keeps a time-varying configuration $g_{lab}(t)$ with respect to the lab. Hence points can be transformed between the segment frame and the lab frame with this information, e.g. $p_s = g_{lab}(t)b_s = Rb_s + x$. Bodies have a method update_segments(void) which properly sets the lab configurations g_{lab} of the segments which make up the body. This simplifies collision detection and scene drawing, both of which can just consider a scene as a collection of individual primitives. (This isn't true of collision response, though).

For multi-link bodies, for $g_{seg} = (R, x)$ on the k-th link, and a point $\bar{p}_k$ in segment coordinates, a point in link coordinates is just $p_k = R\bar{p}_k + x$ and to translate to the base we have the recursion $p_{k+1} = e^{h_k\theta_k}p_k + L(k+1, k)$. Since $L(n+1, n)$ is the location of the base, p_{n+1} gives the lab coordinates of the point $\bar{p}_k$.

In addition, bodies can perform such transformations through the routines like body − pt, which takes a point in the frame of a segment, and returns the corresponding lab coordinates. Note that direct reference to the link associated with the segment is not made.

The relevant frames associated with a link are the cm frame, which originates at the center of mass of the link and is typically aligned with the principle axes; the joint frame, which is attached to the proximal joint of a link and is by convention a translate of the lab frame at the zero position of the entire body; and the individual segment frames. Link information is loaded at run-time by giving, for each link, information about joint axis, joint-to-joint vector, and joint-relative center of mass location as well

as segment information.

2.8.1 Forward kinematics

The function $\mathsf{body-pt(bpt, segnum)}$ takes a point in coordinates of the frame attached to segment *segnum* and returns the corresponding lab coordinates. Each segment frame is related to a link frame by a constant transformation. The link frames for an articulated body are obtained via Denavit-Hartenberg convention, so that the argument b must be first transformed to the appropriate link frame, then to lab coordinates. To facilitate these conversions, a body keeps methods which generate configuration-dependent homogeneous transformation matrices, e.g. g_{01} is the $4x4$ matrix which takes points from link frame 1 to link frame 0, the base frame.

2.8.2 Force to acceleration

Rigid bodies

For a collection of constraint forces F_j and external forces F_k^{ext}, both in lab coordinates and applied at points in body coordinates b_j and b_k^{ext} respectively, the acceleration of a point $p_A = x + Rb_A$ in lab coordinates is

$$\ddot{p}_A = \frac{1}{m_A}\sum_j F_j - \widehat{(Rb_A)} I_A^{-1} \sum_j \widehat{(Rb_j)} F_j \tag{2.54}$$

$$+\frac{1}{m_A}\sum_k F_k^{ext} - \widehat{(Rb_A)} I_A^{-1} \sum_k \widehat{(Rb_k^{ext})} F_k^{ext} + \beta \tag{2.55}$$

Two routines must be supplied by a body (rigid or articulated) which is to be subjected to point constraints. First, $\mathsf{acc_{OF}}()$ represents the *non-constraint* related terms in the acceleration of a point b_A, which is the body-coordinates version of p_A. Second, $\mathsf{force2accel}()$ represents the matrix coefficient of the constraint force F_j in the acceleration expression. So

$$\mathsf{acc\_0F(b_A)} = \frac{1}{m_A}\sum_j F_j - \widehat{(Rb_A)} I_A^{-1} \sum_j \widehat{(Rb_j)} F_j + \beta \tag{2.56}$$

where

$$\beta = -\widehat{(Rb_A)} I_A^{-1} \hat{L}\omega + \omega \times \omega \times (Rb_A) \tag{2.57}$$

and terms like $\widehat{(Rb)}F$ are of course torques in lab coordinates.

For the second routine, we have

$$\mathsf{force2accel}(\mathsf{b_A}, \mathsf{b_j}) = M = \frac{1}{m_A} 1_{3\times 3} - \widehat{(Rb_A)} I_A^{-1} \widehat{(Rb_j)} \tag{2.58}$$

This is a little different from Barzel and Barr [1988], in which the block was decomposed into 4 pieces, $\Gamma G + \Lambda H$.

Articulated bodies

The acceleration of a point attached to a rigid body due to a collection of forces F_j applied at points b_j was given above by

$$\begin{aligned} \ddot{p}_A &= J_A \ddot{\theta} + \dot{J}_A \dot{\theta} && (2.59) \\ &= \sum_j J_A M_A^{-1} J_j^T F_j + J_A M_A^{-1} (\tau_{nc} - C\dot{\theta}) + \dot{J}_A \dot{\theta} && (2.60) \\ & && (2.61) \end{aligned}$$

and τ_{nc} represents *non-constraint* forces such as gravity, joint limits, and internal friction. $J_A(b, s)$ is the Jacobian associated with a point b attached to the frame of link s (b is written w.r.t. frame s). Therefore the routine for the non-constraint-related terms in the acceleration of a point is just

$$\mathsf{acc\_0}(\mathsf{b_A}) = J_A M_A^{-1} (\tau_{nc} - C\dot{\theta}) + \dot{J}_A \dot{\theta} \tag{2.62}$$

$$\tag{2.63}$$

and the matrix coefficient for the effect of a constraint force on point acceleration is

$$\mathsf{force2accel}(\mathsf{b_A}, \mathsf{b_j}) = J_A M_A^{-1} J_j^T \tag{2.64}$$

The term $\dot{J}\dot{\theta}$ calls for special attention. Spatial accelerations are propagated outward from the base according to Rodriguez and Kreutz-Delgado [1992] equation 2.18:

$$\alpha(k) = \phi^T(k+1,k)\alpha(k+1) + H^T(k)\ddot{\theta}(k) + a(k) \tag{2.65}$$

where $a(k)$ describes the velocity dependent "bias accelerations." Focusing on the terms independent of $\ddot{\theta}$, the velocity-dependent acceleration of a point p in lab coordinates that is attached to link k is

$$\alpha(p) = \phi^T(p - O_k)\alpha(k) + a(p) \tag{2.66}$$

where

$$a(p) = \begin{bmatrix} 0 \\ \omega(k) \times \omega(k) \times (p - O_k) \end{bmatrix} \tag{2.67}$$

and $\dot{J}\dot{\theta} = lin(\alpha(p))$ where $lin()$ extracts the linear part of a spatial vector. See equations 2.19, 2.27 in [Rodriguez and Kreutz-Delgado, 1992].

2.8.3 Inertial properties

Murray et al. [1994] give a very clean account of the inertial properties of a rigid body and the relevant coordinate transform issues. Here is a piece of their development, modified slightly to agree with the notation of Rodriguez (which is more useful for implementation).

The kinetic energy of a body can be written using the velocity of a mass element at point p in lab coordinates. Since $\dot{p} = \dot{x} + \dot{R}b$ where x is the center of mass location,

and R is the orientation of the body frame, the kinetic energy is

$$\begin{aligned} KE &= \frac{1}{2}\int_V \rho(b)||\dot{x}+\dot{R}b||^2 dV && (2.68)\\ &= \frac{1}{2}m||\dot{x}||^2 + \frac{1}{2}\omega_b^T I_b \omega_b && (2.69)\\ &= \frac{1}{2}V_b^T M_b V_b && (2.70)\end{aligned}$$

where $V_b = \begin{bmatrix}\omega_b \\ v_b\end{bmatrix}$ is the *body* velocity of the rigid body, $g^{-1}\dot{g}$, which consists of a linear and an angular component. The **body linear velocity** is $v_b = R^T\dot{x}$ where $\dot{x}$ is the center-of-mass velocity in lab coordinates (note that $||v_b||^2 = \dot{x}^T R R^T \dot{x} = ||\dot{x}||^2$). The **body angular velocity** is $\omega_b = R^T\omega_s$ where ω_s is angular velocity in lab coordinates. M_b is the **generalized inertia matrix** at the center of mass, defined as

$$M_b = \begin{bmatrix} I_b & 0 \\ 0 & m1_{3x3}\end{bmatrix} \qquad (2.71)$$

and it has a block diagonal form because x is located at the center of mass. It is fully diagonal if, in addition, R is aligned with the principle axes of the body, so that I_b is diagonal. We refer to (x, R) so chosen as the **base frame** of the body, and call (2.71) the generalized inertia of a body at the base frame. (All that's really needed is the origin located at the cm; nondiagonal I_b isn't a big deal).

Suppose $g_{ab} = (x_{ab}, R_{ab})$ transforms points in the base frame b to a new frame a. Note in passing that the reverse transformation is $g_{ba} = (x_{ba}, R_{ba}) = (-R_{ab}^T x_{ab}, R_{ab}^T)$. The inertia matrix M_b can be transformed to frame a by transforming the velocities (twists) V_b that appear in the quadratic form (2.70):

$$\begin{aligned} V_a &= \begin{bmatrix} R_{ab} & 0 \\ \hat{x}_{ab}R_{ab} & R_{ab}\end{bmatrix} V_b && (2.72)\\ &= Ad_{ab}V_b && (2.73)\end{aligned}$$

with the **adjoint** transformation Ad_{ab} defined by the equation. Ad_{ab} transforms velocities in frame b to velocities in frame a. The inverse transformation is

$$(Ad_{ab})^{-1} = \begin{bmatrix} R_{ab}^T & 0 \\ -R_{ab}^T \hat{x}_{ab} & R_{ab}^T \end{bmatrix} \tag{2.74}$$

and $V_b = Ad_{ab}^{-1} V_a$. It should be possible to show that $Ad_{ab}^{-1} = Ad_{ba}$, shouldn't it? At any rate, kinetic energy (invariant under coordinate transformations) becomes

$$KE = \frac{1}{2} V_b^T M_b V_b = \frac{1}{2} (Ad_{ab}^{-1} V_a)^T M_b (Ad_{ab}^{-1} V_a) \tag{2.75}$$

$$= \frac{1}{2} V_a^T M_a V_a \tag{2.76}$$

where

$$M_a = Ad_{ab}^{-T} M_b Ad_{ab}^{-1} \tag{2.77}$$

$$= \begin{bmatrix} R_{ab} I_b R_{ab}^T - m\hat{x}^2 & m\hat{x} \\ -m\hat{x} & m1_{3x3} \end{bmatrix} \tag{2.78}$$

Going in the other direction, an inertia matrix with respect to frame a can be transformed to frame b by $M_b = Ad_{ab}^T M_a Ad_{ab}$. These formulas are useful for computing the inertial properties of rigid bodies with respect to a link frame when the bodies are composed of a number of segments, each of which is defined by a constant (x_{ab}, R_{ab}) which takes points in the segment frame to the link frame. Segment frames are by convention chosen at the center of mass of a shape primitive. Note that (2.77) is the general formula, while (2.78) is the formula specialized to the case when M_b is block diagonal, i.e. when frame b is located at the center of mass.

Given a collection of disjoint segments, with segment frames given relative to a link frame, we can transform all the individual segment inertias (given for frames located at the centers of mass) to the link frame, with equation (2.77), and add them up (from the additivity of kinetic energy) to get M_L, the constant inertia relative to the link frame

at the zero position. In order to get the spatial inertia matrix[7], we use $g_{SL} = (0, R_{SL})$ which maps points in the link frame to points in the spatial frame. Then the spatial inertia is given by $M_S = Ad_{SL}^{-T} M_L Ad_{SL}^{-1}$, which takes each $3x3$ matrix block of M_L and replaces it with $R_{SL} P R_{SL}^T$. Given the configuration dependent spatial inertia matrices for each link, the full generalized inertia matrix for an open-chain manipulator can be efficiently formed using the algorithm in Rodriguez and Kreutz-Delgado [1992].

Note in passing that equation (2.74) with $R = 1_{3x3}$ gives the twist transformation for a frame that is translated, but not rotated, i.e. $Ad_{(x,1_{3x3})}^{-1} = \phi^T(x)$ in Rodriguez' notation Rodriguez and Kreutz-Delgado [1992]. So transforming from the spatial inertia to lab inertia is just $M_{Lab} = \phi(x) M_S \phi^T(x)$ in Rodriguez' notation. This lab inertia for a single link is only part of the full generalized inertia calculation, however.

[7]The spatial inertia matrix is *not* the transformation of the link inertia matrix to the lab. It's the transformation of the link inertia to a frame oriented with the lab but located at the link origin. We should carefully define this as the spatial frame, as distinct from the lab frame.

Chapter 3 A class of mixture models for adaptive configuration control of open-chain robots

3.1 Introduction

Motivated by the need for control algorithms for anthropomorphic robots whose structural details are incompletely known, this paper presents a computational study of adaptive schemes for trajectory tracking in open chain manipulators. We present a class of *nonparametric* controllers – controllers whose detailed structure is not based on rigid body nonlinear dynamics – and compare performance with nonadaptive and model-based adaptive methods in a variety of circumstances. Such comparisons are relatively rare, with emphasis typically placed on formal proofs of stability. The main contribution of this paper is to give some insight, through simple examples and simulation study, regarding performance issues that are not always apparent in formal proofs.

The most basic and widely used technique for configuration control employs proportional-derivative (PD) feedback, a non-adaptive technique in which actuator torques are designed to emulate a force field whose equilibrium occurs at in the neighborhood of a desired set point. When used for maintaining constant configurations, the approach is provably globally asymptotically stable for open chain manipulators in the absence of external forces such as gravity[Arimoto and Miyazaki, 1983]. Implementation of PD control requires relatively little knowledge about robot structure, though some knowledge is implicit in the choice of reasonable feedback gains. A common complaint, however, is that PD control is not adequate for more demanding time-varying configuration tracking problems.

On the other end of the spectrum lie control methods which use detailed knowledge

of the kinematic and inertial structure of the robot. Computed torque [Murray et al., 1994, =] is a non-adaptive method in which the rigid body dynamics and external forces acting on a manipulator are perfectly canceled, resulting in a second order system whose dynamics may in principle be arbitrarily designed. As the name suggests, computed torque centers on the *inverse dynamics* of a manipulator, the nonlinear equations which give the joint torques associated with particular values of joint configuration, velocity and acceleration. Perfect knowledge of the inertial parameters appearing in the inverse dynamics is a stringent requirement, and accurate measurement of these parameters is a challenging task [Armstrong et al., 1985]. However, a unique property of the inverse dynamics for open chain manipulators has motivated several direct adaptive approaches: the unknown inertial parameters appear linearly in the inverse dynamics equations. A flurry of activity in the robotics literature produced a number of *model-based* adaptive controllers exploiting this property, controllers with provable stability for tracking control [Slotine and Li, 1991], [Wen and Bayard, 1988], [Whitcomb et al., 1993]. The algorithms all use nonlinear compensation terms in addition to PD feedback, and the fundamental analysis tool is the mechanical kinetic energy of the manipulator.

An important common denominator of the model-based approaches is the assumption that "nonidealities," be they due to unmodelled dynamics, external forces, etc., are nonexistent or bounded and "small." However, two points should be carefully considered. First, deviations due to externally imposed force fields, joint limits, actuator torque limits, unexpected obstacles, etc., are an integral part of control in "unengineered" and uncertain environments. Second, it's not always necessary to have an accurate dynamical model in order to provide accurate control; the success of PD feedback rests on this fact. The first point has motivated the use of *nonparametric* methods for adaptive control[1] which apply compensation terms intended to *approximate* the inverse dynamics. In these studies, great effort goes into guaranteeing that

[1]This use of the term *nonparametric* is standard in the regression and neural network literature. The methods described always have free parameters, but they do not necessarily correspond to physically meaningful quantities such as masses and moments of inertia.

a highly accurate inverse dynamics model can be constructed, and in some cases, perfect modelling ability is assumed. But the second point mentioned above applies equally well to nonparametric methods, and perfect modelling may be necessary.

We begin from the premise that the attractive feature of nonparametric methods is the promise of control without precise implementation of global nonlinear inverse dynamics. In the sections below we present and study a heuristically motivated compensation scheme – motivated by Lyapunov analysis but *not* capable of provable perfect tracking. Of course, *perfect* tracking is an unattainable goal anyway, so our purpose is to show through simple examples that the method presented is competitive with model-based schemes. We carefully consider a number of factors influencing performance, since many ingredients contribute to the success or failure of an adaptive scheme.

In section (3.2), we review the relevant properties of open-chain rigid body dynamics, PD control, and Lyapunov synthesis. Section (3.3) introduces the approximate compensation scheme that is the main topic of this paper, and also describes the model-based scheme of [Slotine and Li, 1991] used for comparison. Section (3.4) describes simulation results and discusses a number of issues relevant to assessing the performance of adaptive controllers.

3.2 Preliminaries

A Lagrangian analysis [Murray et al., 1994], [Spong and Vidyasagar, 1989] leads to a compact set of equations governing the generalized coordinates $\theta, \dot{\theta}$ of an open-chain manipulator[2]:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} = \tau_{ext}(\theta, \dot{\theta}) + \tau_a \tag{3.1}$$

[2]An open chain manipulator is snake-like in structure and contains no closed kinematic loops.

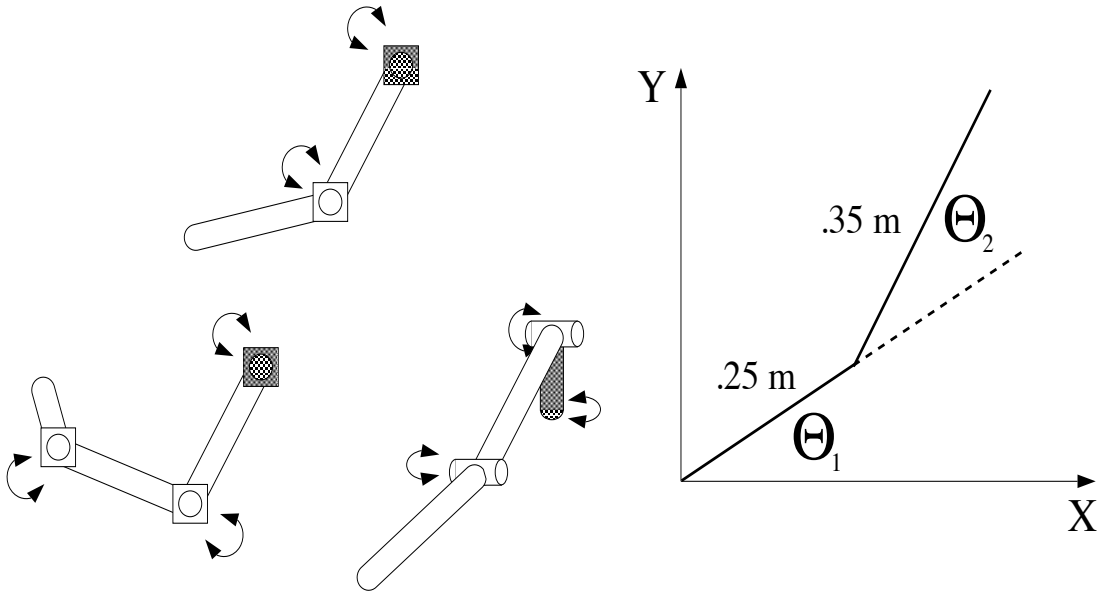


Figure 3.1: Arm-like manipulator models

$M(\theta)$ is the generalized inertia matrix, and it is symmetric and positive definite for any θ. $C(\theta, \dot{\theta})$ is a matrix describing Coriolis and centripetal effects; it stems from the configuration dependence of M, is itself linearly dependent on $\dot{\theta}$, and satisfies $\dot{M} = C + C^T$, so that $\dot{M} - 2C$ is skew-symmetric independent of θ and $\dot{\theta}$. On the right-hand side of (3.1), τ_{ext} represents external forces such as gravity and friction, and τ_a are actuator torques under our control.

The equations (3.1) apply to any open-chain manipulator, but of primary interest are anthropomorphic *arm-like* manipulators which have a shoulder, elbow, and possibly a wrist. Figure 1a shows a few common mechanisms, and figure 1b shows a planar two-link arm in more detail. The simulations presented below use the two-link arm model because elbow-shoulder interactions are arguably the dominant rigid-body issue for arm-like manipulators, and because a two-link arm is a common system for experimental (e.g. [Whitcomb et al., 1993]) evaluation of control methods.

A number of problems in configuration control are associated with the dynami-

cal system (3.1). These problems call for the generation of actuator torques to satisfy a performance objective for the generalized coordinates $\theta(t)$. For the *stabilization* problem, $\tau_a(t)$ is chosen to guarantee the asymptotic stability of a desired set point θ_d. For *tracking*, the actuator torques should cause $\theta(t)$ to converge asymptotically to a desired trajectory $\theta_d(t)$ for which $||\dot{\theta}_d(t)||$ and $||\ddot{\theta}_d(t)||$ are bounded. Practically speaking, convergence in some small finite time is desired.

Because (3.1) describes a mechanical system which obeys conservation of energy (in fact, that is how the equations of motion are derived), an elegant solution to the stabilization problem [Arimoto and Miyazaki, 1983] is to create an artificial potential field whose unique energy minimum occurs at the desired configuration. The addition of damping terms to dissipate kinetic energy guarantees that the closed loop system will approach the minimum energy configuration. The kinetic energy of a manipulator which obeys (3.1) is given by the quadratic form $V_{ke} = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta}$. The time derivative of V_{ke} gives a clear idea of how kinetic energy is "injected" into the system; it is the work done by external and actuator torques (see appendix for details):

$$\dot{V}_{ke} = \dot{\theta}^T \tau_{ext} + \dot{\theta}^T \tau_a \tag{3.2}$$

In the absence of external forces ($\tau_{ext} = 0$), PD control for the stabilization problem creates a linear spring force field with linear damping:

$$\tau_a = -K_p(\theta - \theta_d) - K_d\dot{\theta} \tag{3.3}$$

where K_p and K_d are positive definite gain matrices, and θ_d is the desired constant set point. A Lyapunov function consisting of the sum of the kinetic energy V_{ke} and the "virtual" spring energy $\frac{1}{2}(\theta - \theta_d)^T K_p(\theta - \theta_d)$ can be used to prove global asymptotic stability of the set point.

When conservative and/or dissipative forces such as gravity and friction are present,

the PD control (3.3) still implies global asymptotic stability; however, the equilibrium point to which the system converges is not given by θ_d, but is given implicitly by $\theta_* + K_p^{-1}\tau_{ext}(\theta_*, 0) = \theta_d$. This equation can have multiple solutions, and local stability depends on the eigenvalues of $[1_{nxn} + K_p^{-1}\frac{\partial \tau_{ext}}{\partial \theta}]^{-1}$, where 1_{nxn} is the n by n identity matrix. If the K_p gains are sufficiently large[3], local stability for any set point is preserved. Formally, a stable equilibrium θ_* can be positioned arbitrarily close to the desired set point θ_d with choice of sufficiently large K_p. However, the resulting high arm stiffness is not always acceptable, especially in environments with the possibility of interaction with other objects. For example, biomechanical studies (e.g. [Bennett et al., 1992]) indicate that in natural postures, the human arm is *not* stiff enough to nullify the influence of gravitational forces, suggesting a role for feedforward compensation.

Moving on to the tracking problem, define error signals $e = \theta - \theta_d$, $\dot{e} = \dot{\theta} - \dot{\theta}_d$ for the time varying desired configuration $\theta_d(t)$. Then tracking requires a control law which asymptotically stabilizes the error dynamics for $(e, \dot{e})$ about $(0, 0)$. The starting point for all solutions is a time-varying PD control of the form

$$\tau_a = -K_p(\theta - \theta_d(t)) - K_d(\dot{\theta} - \dot{\theta}_d(t)) \tag{3.4}$$

Unlike in the constant set point case, $(e, \dot{e}) = 0$ is not asymptotically stable under PD control, even in the absence of external forces. However, the resulting closed loop system is "totally stable" in that $(e, \dot{e})$ will remain bounded for bounded desired trajectories with sufficiently small velocities and accelerations. The proof follows easily from the definition of total stability in [Slotine and Li, 1991]. Note that this does not ensure small tracking errors, but it hints at the generally good behavior of PD control; in practice for "slow" desired trajectories and "high" feedback gains, PD control performs well [Arimoto, 1984]. But of course there are situations which call for fast movements and/or low feedback gains, and in such cases a more elaborate solution is

[3]By which we mean "if the minimum eigenvalue of K_p is sufficiently large."

warranted.

Since the initial appearance of Lyapunov-based proofs for PD solutions to the stabilization problem, several methods have been developed to address the more stringent demands of the tracking problem ([Wen and Bayard, 1988], [Whitcomb et al., 1993], [Slotine and Li, 1991], [Sadegh and Horowitz, 1987]). These methods have the common feature of requiring structural details of the manipulator model in order to form nonlinear compensation torques. Perfect tracking is obtained by using perfect knowledge of the configuration-based nonlinearity and inertial constants in (3.1) so as to formally invert the dynamics and replace them with exponentially stable error dynamics. If the inertial parameters are known and the kinematics carefully modelled, the information needed for these **model-based** schemes can be computed relatively efficiently.[4] Such modelling and measurement can be quite challenging [Armstrong et al., 1985], a fact which has driven the development of *provably stable* model-based adaptive control schemes [Bayard and Wen, 1988], [Whitcomb et al., 1993],[Slotine and Li, 1991], [Sadegh and Horowitz, 1987].

We will use the same tools developed for model-based adaptive schemes to motivate the nonparametric method given in the next section. The method of [Slotine and Li, 1991] is the starting point; the key idea of their approach, as descended from the "sliding surface" literature [Slotine and Li, 1983], is to consider $\dot{\theta}$, rather than θ, as the controlled quantity. If $\dot{\theta}$ could be controlled directly, setting $\dot{\theta} = \dot{\theta}_d - \lambda(\theta - \theta_d)$ for $\lambda > 0$ would solve the tracking problem. Defining the *reference velocity* $\dot{\theta}_r(t) = \dot{\theta}_d - \lambda(\theta - \theta_d)$, the tracking problem can be converted to the problem of making the reference velocity error $s \equiv \dot{\theta} - \dot{\theta}_r = \dot{e} + \lambda e$ converge to zero. $\dot{\theta}_r$ describes a *sliding surface* [Slotine and Li, 1991], a time-varying vector field, onto which we would like $\dot{\theta}(t)$ to converge. Other approaches for addressing time variation due to $\theta_d(t)$ exist, but Lyapunov-based stability analysis is greatly simplified by the notions of a reference velocity and a sliding

[4] Computational expense was once a compelling motivation for non-model-based approaches. However advances in algorithms and computing power have made the calculation of terms in (3.1) possible at real-time speeds, if the necessary parameters are known.

surface.

Defining a kind of pseudo kinetic energy, the s-energy, $V_s = \frac{1}{2}s^T M s$ by analogy to V_{ke}, differentiating gives (see appendix for details):

$$\dot{V}_s = s^T(\tau_a - M\ddot{\theta}_r - C\dot{\theta}_r + \tau_{ext}) = s^T\tau_a - s^T\tau_r \quad (3.5)$$

$$\tau_r \equiv M\ddot{\theta}_r + C\dot{\theta}_r + \tau_{ext} \quad (3.6)$$

where the first term, $s^T\tau_a$ consists of energy injected due to control action, and the second term, $s^T\tau_r$ consists of energy stemming from the nonlinear dynamics, external forces, and details of the desired trajectory. τ_r defines a time-varying *reference torque* closely related to the inverse dynamics (3.1). PD feedback for time-varying θ_d can be rewritten

$$\tau_{pd} = -K_p(\theta - \theta_d) - K_d(\dot{\theta} - \dot{\theta}_d) = -K_d(\dot{\theta} - (\dot{\theta}_d - K_d^{-1}K_p(\theta - \theta_d))) \quad (3.7)$$

$$= -K_d s \quad (3.8)$$

and provided that $K_d^{-1}K_p$ is positive definite, we can set $\lambda \equiv K_d^{-1}K_p$. Then if the control torques consist of a PD term and a compensation term τ_c, i.e. $\tau_a = \tau_{pd} + \tau_c$, (3.5) becomes

$$\dot{V}_s = -s^T K_d s + s^T(\tau_c - \tau_r) \quad (3.9)$$

and we can define the *reference torque error* as $e_\tau = \tau_c - \tau_r$.

The model-based approaches of Slotine and Li (and the closely related approaches of [Wen and Bayard, 1988], [Whitcomb et al., 1993]) are based on viewing τ_c as a static function of state measurements and of the desired trajectory. The reference torque $\tau_r = M(\theta)\ddot{\theta}_r + C(\theta,\dot{\theta})\dot{\theta}_r - \tau_{ext}(\theta,\dot{\theta})$ is a function of θ, $\dot{\theta}$, $\dot{\theta}_r$, and $\ddot{\theta}_r$. If a parameterized form $\tau_c(p,\theta,\dot{\theta},\dot{\theta}_r,\ddot{\theta}_r)$ is chosen such that for some choice of constant

parameters p_0, $\tau_c(p_0) = \tau_r$, then the compensation torques exactly cancel the reference torques. For this parameter choice, $\dot{V}_s = -s^T K_d s \leq 0$ and $s \to 0 \implies e \to 0$ so that asymptotic perfect tracking results.

Nonparametric methods ([Jordan and Rumelhart, 1992], [Kawato et al., 1987], [Sanner and Slotine, 1995]) involve choosing a more generic, non-model based form for τ_c, and determining adaptation rules for the unknown parameters. In analyzing such methods, great emphasis is placed on the static function approximation capabilities of a nonparametric form for τ_c. Typically by assumption or by design, the form for τ_c has the ability to globally approximate τ_r so that the worst case *reference torque error*

$$sup_{\theta,\dot{\theta},\dot{\theta}_r,\ddot{\theta}_r} ||\tau_c - \tau_r|| \tag{3.10}$$

is "small" for some choice of constant parameters. The tools of approximation theory can be employed to this end, but computational resources required for designs based on global approximation theory can be quite severe (e.g. [Sanner and Slotine, 1995]). Given the inevitable presence of unmodelled dynamics, it is important to understand what can happen when τ_c *cannot* act as a good static global approximation to τ_r.

3.3 Compensation schemes for dynamic forces

In this section we motivate an architecture and learning rules for a class of compensation torques based on *mixture models*. Our scheme is like all adaptive and nonadaptive schemes in having a PD component, so a few points about the use of PD control alone for tracking are worth underlining. First, although formal stability analysis often requires only that K_p, K_d be positive definite, the behavior of the closed loop depends on the interaction of these gains with inertial properties. Picking reasonable gains requires at least gross knowledge of the scale of the inertia matrix $M(\theta)$; hence *no* adaptive controller can claim *no* knowledge of the inertial parameters. Second, highly accurate tracking control can be achieved with high gain PD control [Arimoto, 1984].

However, as gains are increased, instability due to the interaction of high gains with unmodelled aspects such as delays or high-frequency resonances is a serious practical issue. Further, for a system intended for interaction with the environment, high gains are often undesirable; configuration control should be decoupled (to the degree possible) from mechanical impedance of the manipulator. Third, it's possible to design gains for a PD tracking controller based on a linearization of the nonlinear dynamics (3.1) about some configuration ([Spong and Vidyasagar, 1989], [Murray et al., 1994]); The resulting pole placement produces gain matrices which take account of the coupling between joints. Of course, that approach requires knowledge that adaptive methods assume is unavailable. At any rate, PD control serves as a baseline solution for tracking control, and is a starting point for more complicated schemes requiring additional design and computational effort.

From the s-energy derivative (3.9), a criterion for good compensator τ_c is that it follow the time-varying signal $\tau_r(t)$ as closely as possible. Neglecting external forces (to be considered later), τ_r is a linear function of the known reference velocity and acceleration $\dot{\theta}_r$ and $\ddot{\theta}_r$. A natural compensator form emphasizing this linearity is

$$\tau_c = \hat{A}\ddot{\theta}_r + \hat{B}\dot{\theta}_r \tag{3.11}$$

Because the dominant nonlinearity of M and C is configuration dependent, we propose forming the matrices $\hat{A}$ and $\hat{B}$ as linear mixtures, i.e.

$$\hat{A} = \sum_i A_i g_i(\theta) \;\; \hat{B} = \sum_i B_i g_i(\theta) \tag{3.12}$$

where the $g_i(\theta)$ are scalar nonlinear configuration dependent *gating functions*. By choosing gating functions to be normalized gaussians defined by

$$g_i(\theta) = \frac{1}{Z} e^{-||\theta - c_i||^2/2\sigma^2} \tag{3.13}$$

$$Z = \sum_i e^{-||\theta - c_i||^2/2\sigma^2} \tag{3.14}$$

$$\tag{3.15}$$

there is a simple interpretation of equation 3.12 as a collection of linear "experts" [Jordan and Jacobs, 1994]. The gating functions form a partition of unity over the configuration space ($\sum_i g_i(\theta) = 1$); each partition has an associated "expert" used to produce a torque $\tau_{ci} = A_i\ddot{\theta}_r + B_i\dot{\theta}_r$ and the individual τ_{ci} are blended to produce the final compensation torque τ_c. The variance of the gaussians in (3.13) controls the sharpness of the partition boundaries; low variances correspond to hard boundaries while high variances lead to significant smoothing between the different linear maps.

Note that there are *many* possible forms for nonlinear regressors[5], and (3.12) is just one example. The neural network literature has intensively studied the problem of function approximation through combinations of nonlinear *basis functions*; (3.12) is a variant in which the g_i play the role of basis functions. It's beyond the scope of this paper to discuss the details of this literature (see [Girosi et al., 1995] for review), but there are three points that are relevant here. First, there is no fundamental difference in approximation power between different nonlinear basis functions. A recent result establishes that essentially any collection of nonlinear functions produces linear combinations which are dense in the space of continuous functions on bounded domains. Second, an important qualification to the previous statement occurs when basis functions reflecting known nonlinear structure are employed. In such a case, the number of basis functions needed to accurately approximate a target function can potentially be much smaller than the number required if a more "generic" basis set such as normalized gaussians were used. Third, new issues arise when the parameters

[5] nonlinear as a function of θ

defining the basis functions themselves may be altered. Although this case, which includes the "classic" multilayer perceptron algorithms, offers interesting computational possibilities [Barron, 1993], the resulting nonlinearly parameterized models are difficult to analyze. In this paper we assume that the gating functions are fixed *a priori*.

The compensator (3.11) mimics the linear structure of the reference torque (3.6) to create a class of compensators with configuration dependent nonlinearity. Note however that there is *no* set of parameters for which a compensator of the form (3.11) can exactly approximate (3.6), regardless of the number of gating functions used. The Coriolis/centripetal torques in (3.6) are given by

$$[C\dot{\theta}_r]_i = \sum_j \sum_k \Gamma_{ijk}\dot{\theta}_j\dot{\theta}_{rk} \tag{3.16}$$

where the *Christoffel* symbols [Armstrong et al., 1985] are configuration dependent. The product velocity terms in this expression cannot be matched by (3.11), and part of the purpose of this study is to consider the potential effects of this kind of structural mismatch in approximation ability.

The "ultimate" compensation model is one which exploits *complete* knowledge of the rigid body dynamics structure, and is useful for comparison purposes. As mentioned above, the model-based approach of [Slotine and Li, 1991] exploits the fact that 3.6 admits an *exact* linearly parameterized form for any open-chain robot. Their algorithm employs compensation torques of the form

$$\tau_c = Y(\theta, \dot{\theta}, \dot{\theta}_r, \ddot{\theta}_r)\hat{a} \tag{3.17}$$

where the parameters $\hat{a}$ correspond to sums and products of inertial constants. The "regressor matrix" Y plays a role like the bass or gating functions, in that it encapsulates the fixed and nonlinear parts of the compensator. The elements of Y can be obtained from a derivation (and symbolic manipulation) of the equations of motion,

whose details depend on the kinematic structure. Therefore the elements of Y represent the "right" basis functions for rigid body dynamics, but obviously require strong knowledge about the manipulator structure.

3.3.1 Learning rules and dissipative maps

Adaptation rules for the mixture compensators above can be developed through a simple interpretation of the Lyapunov synthesis methods used in model-based control. *If* we knew the reference torque error $e_\tau = \tau_c - \tau_r$, any free parameters in a compensator τ_c could be adjusted to reduce $\frac{1}{2}e_\tau^T e_\tau$, giving a gradient rule of the form $\dot{w} = -\eta \frac{\partial \tau_c}{\partial w}^T e_\tau$ for parameter w. However, e_τ cannot be directly obtained, since it depends on τ_r, which in turn depends on the unknown rigid body dynamics details. The s-energy derivative in (3.9) implies a *dissipative* mapping between e_τ and s, the reference velocity error which can be measured. The general properties of dissipative and passive mappings are well known in linear and model based adaptive control [Astrom and Wittenmark, 1995], [Narendra and Annaswamy, 1989]. Here we emphasize that signals related by a dissipative map are in effect *positively correlated*. From the positive definiteness of the inertia matrix $M(\theta)$, the s-energy $V(t) = s^T M s$ is lower bounded by zero. From (3.9),

$$V_s(t) = -\int_0^t s^T K_d s + \int_0^t s^T e_\tau + V_s(0) \geq 0 \tag{3.18}$$

For initial conditions of $\theta(0) = \theta_d(0)$, $\dot{\theta}(0) = \dot{\theta}_d(0)$, the $V_s(0)$ term vanishes. Note that if for any period of time, say $[0, \bar{t}]$, the reference velocity error s is nonzero, then $\int_0^{\bar{t}} s^T K_d s$ will be strictly greater than zero since K_d is positive definite. Then (3.18) implies

$$\int_0^{\bar{t}} s^T e_\tau \geq \int_0^{\bar{t}} s^T K_d s > 0 \tag{3.19}$$

and for all $t \geq \bar{t}$, $\int_0^t s^T e_\tau > 0$. Intuitively, this result indicates that the vector output of a dissipative system has, on average, a positive projection on the input (e_τ) if both

are nonzero. In this case, the dissipative relationship suggests that s can serve as an approximation for e_τ; adjustments based on s instead of e_τ will point *on average* in the correct direction. Then a gradient rule for the mixture compensator (3.11) is simply

$$\begin{aligned} \dot{A}_{ijk} &= -\eta g_i(\theta) s_j \ddot{\theta}_{rk} \\ \dot{B}_{ijk} &= -\eta g_i(\theta) s_j \dot{\theta}_{rk} \end{aligned} \tag{3.20}$$

where we've substituted s for e_τ in a gradient formula. This adaptation rule is very similar in form to the model-based adaptive rule given in [Slotine and Li, 1991]:

$$\dot{a}_i = -\eta \frac{\partial \tau_c}{\partial a_i}^T s = -\eta Y_{ji} s_j \tag{3.21}$$

In all cases, η is a positive learning rate or *adaptation gain*; its role will be discussed in the results section.

A more sophisticated adaptation rule with adaptive learning rates can be derived by applying ideas from recursive estimation. The compensator torque can be written[6]

$$\tau_c = \sum_i [A_i \; B_i] \begin{bmatrix} g_i(\theta)\ddot{\theta}_r \\ g_i(\theta)\dot{\theta}_r \end{bmatrix} \tag{3.22}$$

$$= W\Phi(t) \tag{3.23}$$

where W is the n by $2nN$ matrix $W \equiv [A_1 B_1 A_2 B_2 ... A_N B_N]$ of adjustable parameters, and $\Phi(t)$ is the $2nN$ dimensional *regressor* vector

$$\Phi^T = [g_1\phi^T g_2\phi^T ... g_N\phi^T] \tag{3.24}$$

$$\text{where } \phi = \begin{bmatrix} \ddot{\theta}_r \\ \dot{\theta}_r \end{bmatrix} \tag{3.25}$$

[6]If P and Q are m by n matrices, then $[P\; Q]$ denotes a m by 2n matrix, and $\begin{bmatrix} P \\ Q \end{bmatrix}$ denotes a 2m by n matrix.

analogous to Y from the model based approach. If τ_c is intended to take be an estimate of τ_r, we can write $\tau_r = W_0\Phi + d(t)$ where $d(t)$ is a model mismatch term which is zero in the event that τ_r can be perfectly approximated. Then $e_\tau = (W - W_0)\Phi - d(t)$, and the gradient rule takes the form

$$\dot{W} = -\eta e_\tau \Phi^T \approx -\eta s \Phi^T \tag{3.26}$$

Since the problem is one of linear regression (the nonlinearity being hidden inside Φ), the ability to fit the parameters in W depends on the degree to which Φ spans its $2nN$ dimensional space as it evolves in time. As is standard in linear adaptive control, we can use the time-varying excitation matrix $\Phi\Phi^T$ to determine directions corresponding to well determined parameters. An adaptation rule which varies learning rates based on these directions is

$$\dot{W} = -P(t)e_\tau \Phi^T \tag{3.27}$$

$$\dot{P} = -P\Phi\Phi^T P \tag{3.28}$$

where $P(t)$ is a $2nN$ by $2nN$ matrix of adapting learning rates, whose evolution depends on the rank of the excitation matrix $\Phi\Phi^T$. As with the gradient rule (3.20), s can serve as an approximation to e_τ in the adaptation (3.27). The initial conditions $P(0)$ can be chosen as $\eta 1_{2nNx2nN}$ where η is a positive constant. If the regressor Φ is sufficiently exciting, $P(t)$ will decay to zero. In practice, $P(t)$ decays according to the subspace of directions in parameter space that are determined by the "data" given in a particular trajectory of $\Phi(t)$. To recap, the schemes for tracking include:

1. PD control: $\tau = \tau_{pd} = -K_p(\theta - \theta_d) - K_d(\dot{\theta} - \dot{\theta}_d)$. This corresponds to zero additional compensation, and represents a lower baseline for tracking performance.

2. Linear mixture $\tau = \tau_{pd} + \tau_c = \tau_{pd} + \hat{A}\ddot{\theta}_r + \hat{B}\dot{\theta}_r$, mixtures and adaptation schemes defined above.

3. Model-based regressor $\tau = \tau_{pd} + \tau_c = \tau_{pd} + Y(\theta, \dot{\theta}, \dot{\theta}_r, \ddot{\theta}_r)a$, $\dot{a} = -\eta Y^T s$. This corresponds to complete knowledge of the structure of the nonlinear compensation, and represents an upper baseline for performance.

The next section details some simulation results illustrating the performance of these schemes.

3.4 Simulations

A number of simulation experiments with a two-link arm model (figure 1) were performed in order to compare the performance of the compensation schemes described above. Inertial and kinematic parameters corresponding to two different arms were tested. The first parameter set, due to [Uno et al., 1989] approximates an average human arm and has been used in a number of biomechanical studies modelling human arm movements. The second, due to [Slotine and Li, 1991] corresponds to an arm that is larger, has significantly greater inertia, and has a mass distribution that approximates the firm grasp of a load. Parameter details are given in the appendix. Unless otherwise noted, all simulations are performed in the absence of external forces.

3.4.1 Desired trajectories and error measures

Following the experimental comparison of model-based adaptive controllers in [Whitcomb et al., 1993], we adopted a number of conventions for comparing tracking performance. Desired trajectories were varied over different domains in joint space; figure 2a shows the example domains used. Also, different types of desired trajectory were used; sinusoidal and smooth point-to-point movements (figure 2b,2c). Finally, for a given joint space domain and trajectory type, different movement speeds and accelerations were tested. For example, we looked at sinusoidal trajectories with different frequencies and point-to-point movements with different peak speeds (not shown). In the sections below, *movement bandwidth* informally refers to the peak speed and acceleration of a trajectory. A *tracking task* is defined by a particular choice of trajectory

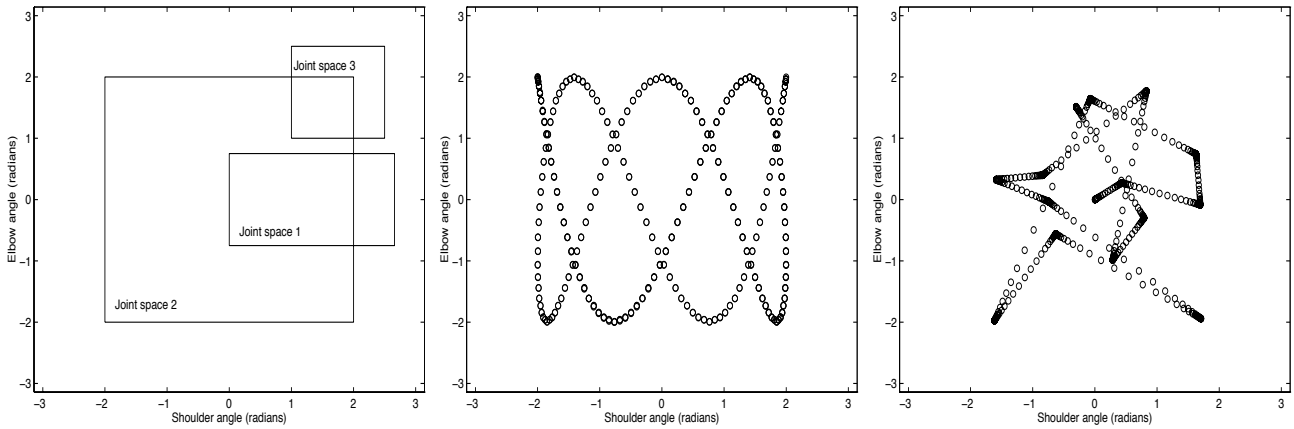


Figure 3.2: Desired trajectories for tracking. a) Examples of three different joint space areas covered by different $\theta_d(t)$. b) Sinusoidal $\theta_d(t)$, shoulder and elbow frequency $.75\pi$ and 2π respectively. c) Point-to-point $\theta_d(t)$, bell-shaped velocity profile. In b) and c) circles are spaced 20 msec. apart.

type, joint space area, and movement bandwidth. A wide variety of tracking tasks was studied; for complete details, see the appendix. The purpose of simulating a set of tracking tasks instead of a single illustrative example (cf. [Sanner and Slotine, 1995], [Kawato et al., 1987]) was to isolate the different factors contributing to success or failure of the proposed tracking schemes.

All the controllers were simulated with identical feedback gains for the PD part of the control torque. As noted above, higher gains lead to more accurate tracking, but there are many contexts in which high gains cannot be employed. For this reason, we experimented with gains well below any stability limits, loosely based on experimentally-determined stiffness and viscosity ranges for the human arm [Bennett et al., 1992].

To make sensible comparisons under varied conditions, a quantitative performance metric must be established. This is a subtle problem for nonlinear systems, since the tools of frequency domain performance analysis are not always applicable. A physically meaningful measure, used effectively in the experimental comparison of

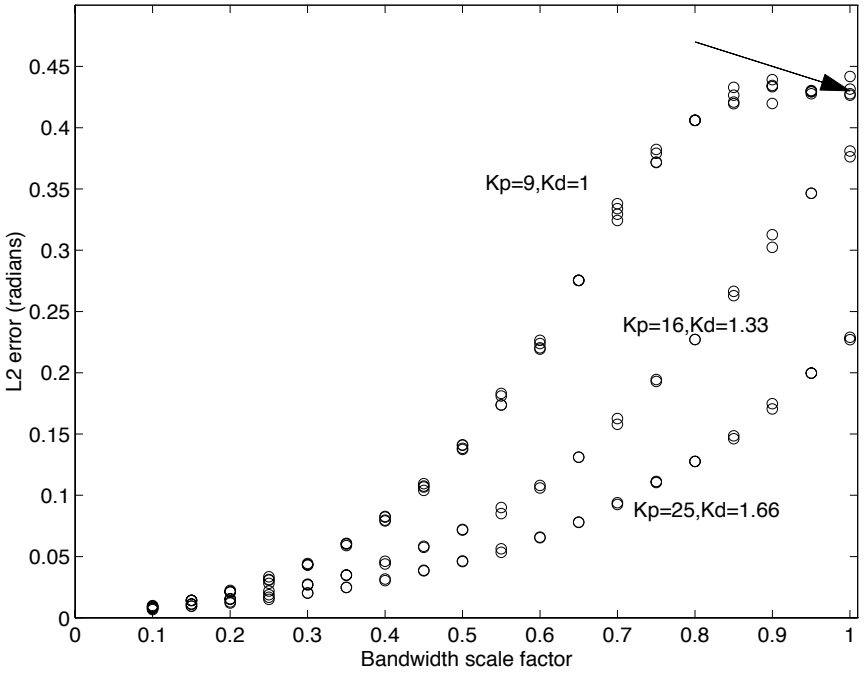


Figure 3.3: L_2 error for PD controllers at 3 different gain values for a range of movement bandwidths of a sinusoidal trajectory $\theta_d(t) = (1.33(1-cos(2\pi kt), .75cos(2\pi kt))$. The bandwidth scale factor k varies from 0 to 1 along the horizontal axis.

[Whitcomb et al., 1993], is the L_2 error norm given by

$$(\frac{1}{T_1 - T_0}\int_{T_0}^{T_1} ||e(t)||^2 dt)^{\frac{1}{2}} \tag{3.29}$$

where $e(t) = \theta(t) - \theta_d(t)$. This measure has units of radians ($1rad. \approx 57^o$) and is a form of the commonly used *root mean squared error* from signal processing.

In addition to an absolute error measure, it's useful to look at the *relative* performance of different control schemes. In a number of the figures below, the L_2 error of an adaptive method on a particular tracking task is presented relative to the performance of a PD controller (with the same gains) on the same task. The resulting *normalized* performance measure permits the combination of data from *different* tasks.

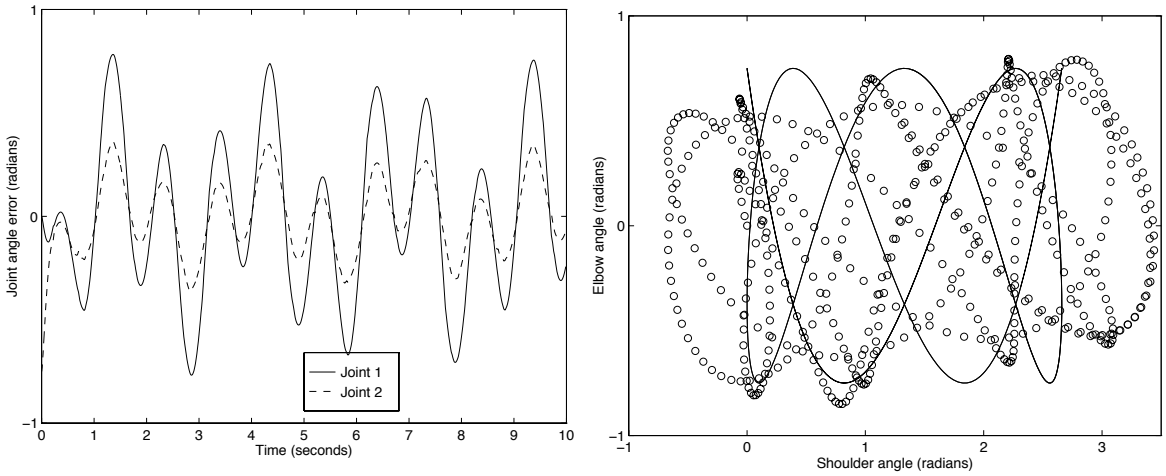


Figure 3.4: Typical low-gain PD performance on a fast sinusoidal trajectory. a) Joint angle errors asa function of time. b) Desired (solid) and actual (circles) trajectories superimposed in joint space. Circles are spaced 20 msec. apart.

3.4.2 Results

Figure 3 shows the L_2 error for PD control for different bandwidth trajectories and different choices of PD gains. As expected, slower trajectories and/or higher gains lead to better performance. The arrow in the figure indicates an example of the performance of a "low" gain controller on a fast trajectory. An expanded view of this trial is shown in figure 4, which shows time-domain and joint space views of the performance. Figure 4a shows the (substantial) joint angle error as a function of time, and 4b shows the actual and desired trajectories superimposed in joint space. For this choice of gains ($K_p = 9, K_d = 1$, in the range of stiffness and viscosity for a comparably sized human arm), PD control does a rather poor job of tracking the desired trajectory.

Figure 5 summarizes the main results for the adaptive schemes. In this figure and the figures below, IDSL refers to the model-based inverse dynamics method of Slotine and Li. M1, M2, ... M8 refer to mixture compensators (3.11) with $N = 1, 2, ..., 8$ partitions respectively. Adaptation gain is the same for al the adaptive controllers ($\eta = 0.01$), as are the PD gains ($K_p = 9, K_d = 1$). Each data point represents the L_2 error measured over a 10 second period, after adaptation for 30 seconds. Although there are important qualifications to be discussed below, the basic trends in the figure

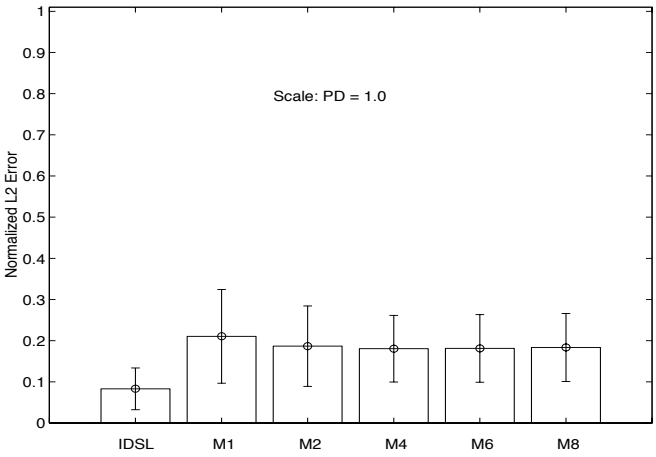


Figure 3.5: Normalized L2 errors for the Slotine-Li arm, sinusoidal $\theta_d(t)$. Bars denote mean performance, error tics denote 1 standard deviation. Plot combines results from different trajectory types, movement bandwidths, and joint space areas.

are consistently observed:

- All adaptive controllers can exhibit performance substantially better than PD control in the absence of external forces.
- The mixture models tend to enhance performance despite an inability to perfectly approximate the reference torque.
- The model-based controller, which uses additional information about the kinematic structure, performs consistently better than the approximate mixture models when there are no external forces.

Figure 6 shows a particular example of the performance of the adaptive scheme $M4$ on the same tracking task shown in figure 4. There is certainly a dramatic improvement in performance over the PD controller. The next few sections describe additional findings concerning the behavior of the compensation schemes.

Adaptation gain

The adaptation gain η has a significant qualitative impact on performance, in that high values can induce instability. When η is low enough to ensure stability, differences

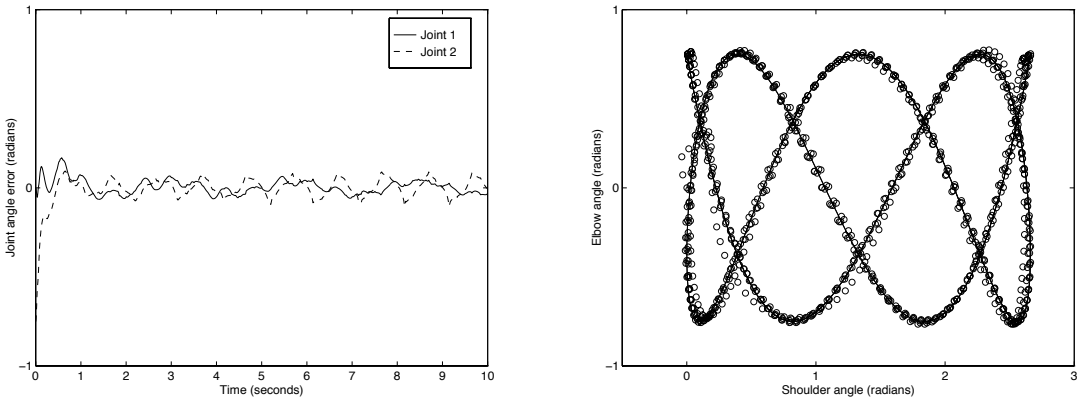


Figure 3.6: Mixture-compensator performance on a fast sinusoidal trajectory (compare with figure 4). a) Joint angle errors as a function of time. b) Desired (solid) and actual (circles) trajectories superimposed in joint space. Circles are spaced 20 msec. apart.

between trials using different values for η are marginal. Figure 7 shows performance for three different values of adaptation gain. However it is essential to note that the data for $\eta = 0.1$ excludes a number of unstable trials; the arrows indicate compensators for which instability was observed. For adaptation gains higher than $\eta = 0.1$, instability was typical, and *all* the adaptive schemes could exhibit unstable behavior. This is an important observation because there are *always* unmodelled dynamic effects. In this case, the unmodelled dynamics are not due to external physical forces (since this simulation is contrived to exclude them). Rather, the stability problems arise because we are actually simulating a discrete time approximation to a continuous set of differential equations, an issue for real and simulated robots. The typical adaptive control analysis requires only that adaptation gains be positive for formal stability. In practice, as is the case with PD feedback gains, the adaptation gain must be sufficiently small to avoid problems due to discretization.

Unfortunately, there is usually no simple way to determine the appropriate adaptation gain for a particular situation. Recursive estimation learning rules like the one described in section3.3.1 partially address this problem, but those rules still require initial conditions for learning rates. As is the case with gradient learning rules, if the

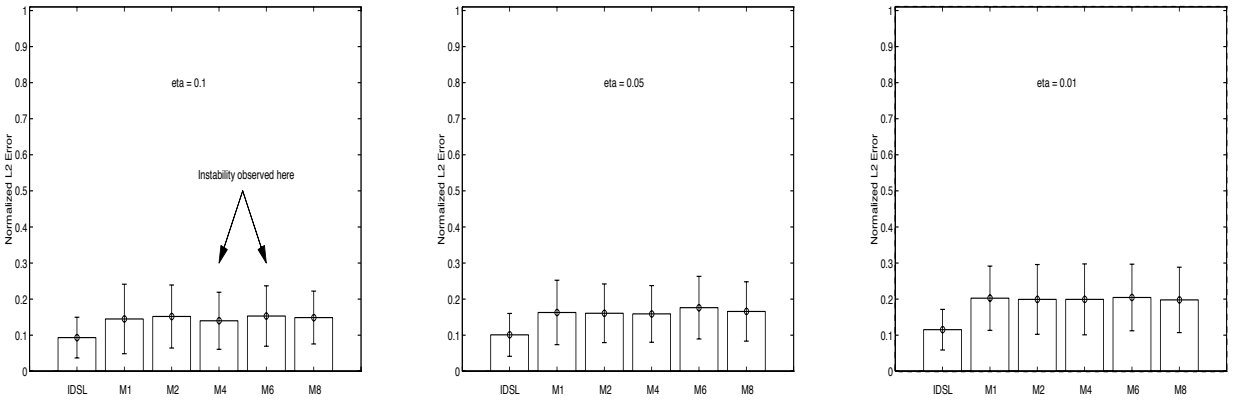


Figure 3.7: Normalized L2 errors for the Uno arm for different values of adaptation gain η. a) $\eta = 0.1$, b) $\eta = 0.05$, c) $\eta = 0.01$.

initial conditions for learning rates in a recursive estimation scheme are "too large," instability can result. The qualitative sensitivity to adaptation gain is a major stumbling block for all adaptive control schemes, and rigorous methods are lacking in all but the simplest situations.

External forces

Significant external forces that are not explicitly accounted for in the structure of the adaptive compensators degrade performance, but improvements over PD control are still typical. Further, tasks with unmodelled external forces tend to reduce the differences between the model-based and mixture compensators. Figure 8 shows an example of performance in the presence of gravity as an unmodelled external force. In 8a, the absolute L_2 errors are shown with and without gravity (Note that this figure combines data from different tasks, but the std. deviation bars are omitted for clarity). In 8b, normalized errors are shown. Note that the adaptive methods were able to perform well on many trajectories despite the fact that there was no constant parameter choice for any of the methods which would allow global approximation of the static gravitational force. Figure 9 shows another example in which nonlinear viscosity forces are added as external forces. Again, absolute performance is degraded (not shown), relative performance continues to favor the adaptive methods, and dif-

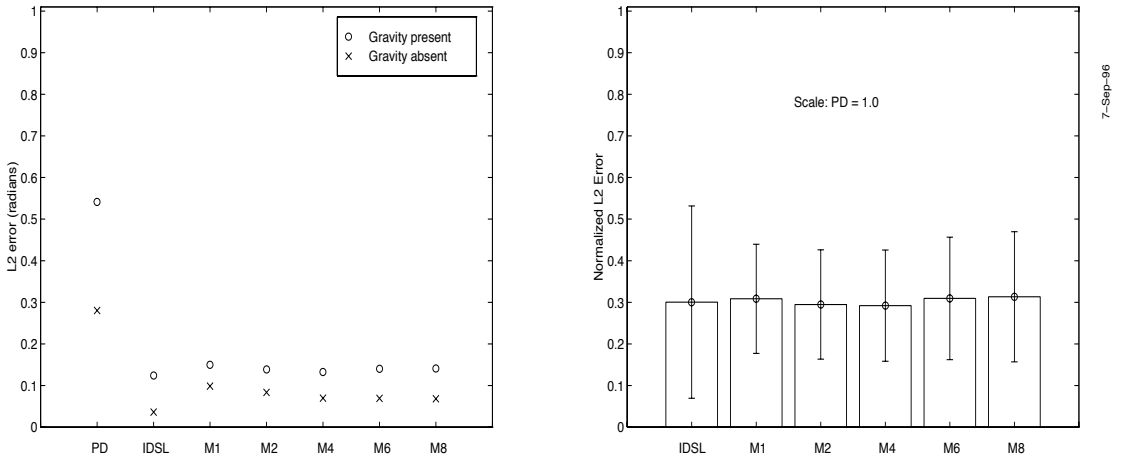


Figure 3.8: External forces: gravity. a) Absolute L_2 errors with and without gravity; std. deviation bars omitted for clarity. b) Normalized errors with gravity present.

ferences between the adaptive methods are reduced.

In both examples, the adaptation feedback loop allows τ_c to track τ_r. There is a limit to this ability in the case of gravity as an external force, however. If the desired trajectory is slow or constant, the adaptive methods (which copntain no static terms) will not be able to hold the slowly varying postures due to the dominance of the gravitational forces. Of course, static forces can be modelled and the compensator for modified accordingly, with adaptation rules easily obtainable as with the dynamic compensator above. The point is that the tracking problem need not always be equated with the problem of perfect modelling of inverse dynamics, and that adaptation feedback is essential.

Parameter convergence

For gradient learning, the parameter derivatives will be nonzero when s is nonzero. As a result, the parameters do not converge to constant values, but rather fluctuate in order to keep the compensation torques close to the reference torques. Figure 10 shows an example of the time varying behavior of some of the adaptive parameters for an M4 compensator; the traces are typical and representative of parameter fluctuations

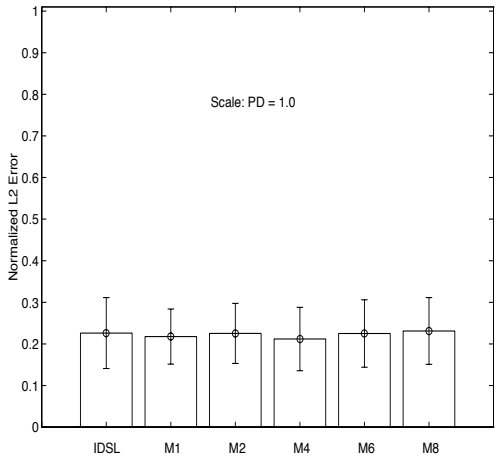


Figure 3.9: External forces: nonlinear viscosity, normalized errors.

for all the adaptive methods (including model-based). These fluctuations allow the compensation torque applied by a particular scheme to follow the reference torques despite the fact that constant parameters would lead to poorer inverse dynamics approximation. This is an essential feature of on-line adaptive methods, and has been neglected in the literature. When the tracking problem is cast as anexercise in static approximation, parameter fluctuation is viewed as something to be avoided. But our experience with simulations suggests that parameter fluctuation typically enhances the performance of an adaptive method by allowing feedback from the reference velocity error s to make up for structural definiences in the adaptive compensator.

Different mixture models

For most of the examples encountered, there is no consistent difference between the different types of mixture model. In fact, careful modelling of the rigid body nonlinearity may be uneccessary. This is not the same as saying the nonlinearity is negligible. Rather, if the nonlinear terms are treated as disturbancees, the adaptive feedback loop may still be able to track the desired reference torque and achieve "good" performance. This is because the disturbances due to nonlinearity are *not* random noise; they are trigonometric and quadratic functions of the state, and tend to vary smoothly as the state does. The passive relation between s and τ_c and τ_r provides a good source of

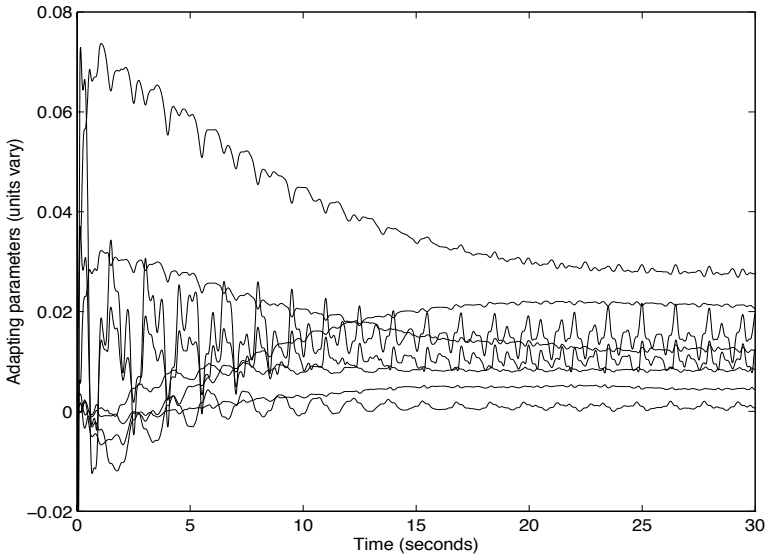


Figure 3.10: Typical behavior of adapting parameters in a mixture compensator as a function of time for a sinusoidal trajectory.

information about these "disturbances." Preliminary simulations with more elaborate arms suggest that linear or weakly nonlinear adaptive methods continue to substantially better performance than PD control.

One consequence of this is that it is unclear whether substantial effort intended to guarantee a certain level of static nonlinear approximation is warranted. For example, [Sanner and Slotine, 1995] develop an approximation theoretic approach to the design of a compensator for the two-link arm model used here. The radial basis function networks needed to model the inertia and Coriolis/Centripetal matrices have over 5000 adjustable parameters; the resulting controller does far better than PD control, but there is no indication that it performs better than a computationally simpler adaptive scheme. Our simulations suggest marginal and diminishing returns from increasing the number of basis functions *for the typically encountered desired trajectories.* Design schemes for nonparametric models of static functions tend to produce implausibly conservative resource requirements [Barron, 1993].

3.5 Conclusions

Online adaptive control is an important building block for designing systems, real or virtual, intended to operate in the presence of parametric or structural uncertainty. The adaptive method presented here is a simple scheme based on a selective view of the structure of the equations of motion for open chain manipulators. The dissipative relationship between a performance measure, s, and an unknown control error, e_τ, is exploited as in model-based control to derive gradient and recursive-estimation adaptation rules. The simulation experiments indicate the usefulness of the approach, with important qualifications concerning the choice of adaptation gains. Compensation based on mixture models therefore offers a promising middle ground between PD control and more knowledge-intensive model-based schemes.

3.6 Appendix

3.6.1 Lyapunov arguments

The kinetic energy of an open-chain manipulator described by (3.1) is

$$V_{ke}(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^T M(\theta)\dot{\theta} \tag{3.30}$$

so that

$$\dot{V}_{ke} = \dot{\theta}^T M(\theta)\ddot{\theta} + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} \tag{3.31}$$

and substituting the closed loop dynamics (3.1) for the first term gives

$$\begin{aligned} \dot{V}_{ke} &= \dot{\theta}^T(\tau_{ext} + \tau_a - C(\theta, \dot{\theta})\dot{\theta}) + \frac{1}{2}\dot{\theta}^T \dot{M}(\theta)\dot{\theta} && (3.32)\\ &= \frac{1}{2}\dot{\theta}^T(\dot{M} - 2C)\dot{\theta} + \dot{\theta}^T(\tau_{ext} + \tau_a) && (3.33)\\ &= \dot{\theta}^T\tau_{ext} + \dot{\theta}^T\tau_a && (3.34) \end{aligned}$$

where the standard definition of the Coriolis/centripetal matrix C gives the skew-symmetry of $\dot{M} - 2C$ ([Murray et al., 1994], [Slotine and Li, 1991]).

For s-energy $V_s = \frac{1}{2} s^T M s$, differentiating gives

$$\begin{aligned} \dot{V}_s &= s^T M(\theta)\dot{s} + \frac{1}{2} s^T \dot{M}(\theta) s & (3.35) \\ &= s^T M\ddot{\theta} - s^T M\ddot{\theta}_r + \frac{1}{2} s^T \dot{M}(\theta) s & (3.36) \\ &= s^T(-M\ddot{\theta}_r - C\dot{\theta}_r + \tau_{ext}) + s^T(\tau_{pd} + \tau_{comp}) & (3.37) \end{aligned}$$

were the last line is obtained by adding and subtracting $C\dot{\theta}_r$ from the previous one.

3.6.2 Simulation details

The sinusoidal desired trajectories used were of the form

$$\begin{aligned} \theta_{d1}(t) &= \alpha_1 + \beta_1(1 - cos(\omega t)) & (3.38) \\ \theta_{d2}(t) &= \alpha_2 + \beta_2 cos(\omega t) & (3.39) \end{aligned}$$

where α_i and β_i are chosen to translate and scale the trajectory into one of the joint space regions shown in figure 2. Choice of $\alpha_i = 0$, $\beta_1 = 1.33$, $\beta_2 = 0.75$, $\omega_1 = 0.75\pi$, $\omega_2 = 2\pi$ produces the desired trajectory used in [Sanner and Slotine, 1995]. For other choices, the trajectories are of the general form used in [Whitcomb et al., 1993].

The trajectories for straight-line movements with bell-shaped velocity profiles were created by interpolating between start and end configurations with a sigmoidal differential equation:

$$\dot{\sigma} = \lambda\sigma(1.0 - \sigma) \qquad (3.40)$$

With initial conditions $1 >> \sigma(0) > 0$, this equation produces solutions in the sigmoid family $\sigma(t) = 1/(1+e^{-t})$, which have the properties of smooth, symmetric acceleration and decceleration, bell-shaped velocity profiles, and peak velocity easily controlled by λ. For start and end points θ_d^{start} and θ_d^{end}, desired configuration trajectories can then be produced with

$$\theta_d(t) = \theta_d^{start} + \sigma(t)(\theta_d^{end} - \theta_d^{start}) \tag{3.41}$$

$$\dot{\theta}_d(t) = \lambda\sigma(1.0 - \sigma)(\theta_d^{end} - \theta_d^{start}) \tag{3.42}$$

$$\ddot{\theta}_d(t) = \lambda^2\sigma(1.0 - \sigma)(1 - 2\sigma)(\theta_d^{end} - \theta_d^{start}) \tag{3.43}$$

A complete sequence can be prduced by concatenating segments produced with these equations. There are obviously many ways of producing straight line movements (e.g. critically damped second-order dynamics), we used this scheme because it roughly matches the observed metrics of human point-to-point movements.

The differential equations associated with the equations of motion (3.1) and the adaptive compensation schemes (refs) were numerically integrated with a 5th order Cash-Karp Runge-Kutta adaptive time step method [Press et al., 1990]. The continuous time error norm (3.29) was evaluated discretely in the obvious way:

$$(\frac{1}{t_N - t_0}\sum_{i=0}^{N-1} ||e(t_i)||^2(t_{i+1} - t_i))^{\frac{1}{2}} \tag{3.44}$$

where t_i is the ascending sequence of sample times produced by the numerical integration routine.

Bibliography

Arimoto. Control and dynamics. In Michael Brady, editor, *Robotics Science*. Plenum Press, New York, New York, 1984.

S. Arimoto and F. Miyazaki. Stability and robustness of PID feedback control for robot manipulators of sensory capability. In *Proceedings International Symposium of Robotics Research*, pages 1–1, Cambridge, Massachusetts, 1983. MIT Press.

Brian Armstrong, Oussama Khatib, and Joel Burdick. The explicit dynamical model and inertial parameters of the puma 560 arm. In *IEEE International Conference on Robotics and Automation Proceedings*, pages 510–518, 1985.

Karl Astrom and Bjorn Wittenmark. *Adaptive control*. Addison-Wesley, Reading, Massachusetts, 1995.

David Baraff. *Dynamic simulation of non-penetrating rigid bodies*. Ph.D. thesis, Cornell University, 1992.

Alan Barr. *Geometric modelling and fluid dynamic analysis of swimming spermatozoa*. Ph.D. thesis, Renselaer Polytechnic Institute, 1983.

Andrew Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39:930–945, 1993.

Ronen Barzel. *Physically based modelling for computer graphics: a structured approach*. Ph.D. thesis, California Institute of Technology, 1992.

Ronen Barzel and Al Barr. A modelling system based on dynamic constraints. *Computer Graphics*, 22(4):179–188, 1988.

David S. Bayard and John T. Wen. New class of control laws for robotic manipulators, Part 2: Adaptive case. *International Journal of Control*, 47(5):1387–1406, 1988.

D. J. Bennett, J. M. Hollerbach, Y. Xu, and I.W. Hunter. Time-varying stiffness of human elbow joint during cyclic voluntary movement. *Experimental Brain Research*, 88:433–442, 1992.

A. R. Collar and A. Simpson. *Matrices and engineering dynamics*. Ellis Horwood Limited, Chichester, West Sussex, England, 1987.

James Cremer. The Isaac project. *Cornell University Technical Report*, 1992.

F. Girosi, M. Jones, and Tomaso Poggio. Regularization theory and neural network architectures. *Neural Computation*, 7:219–269, 1995.

James Hopcroft. The Newton project. *Cornell University Technical Report*, 1986.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

Michael I. Jordan and David E. Rumelhart. Forward models: supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.

M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169–185, 1987.

Ming Lin and John Canny. Closest features for collision detection. *Berkeley Technical Report*, 1995.

Brian Mirtich. Impulse based simulation. Technical report, University of California at Berkeley, 1996.

Marcus Mitchell. Impulses and constraints for dynamic simulation, in preparation. Technical report, California Institute of Technology, 1996.

Richard Murray, Zexiang Li, and Shankar Sastry. *A mathematical introduction to robot manipulation*. CRC Press, Boca Raton, Florida, 1994.

Kumpati Narendra and Anna Annaswamy. *Stable adaptive systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

Preston Pfarner. In preparation. Master's thesis, California Institute of Technology, 1996.

John Platt. *Constraint methods for neural networks and computer graphics*. Ph.D. thesis, California Institute of Technology, 1989.

William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, Cambridge, 1990.

Guillermo Rodriguez and Kenneth Kreutz-Delgado. Spatial operator factorization and inversion of the manipulator mass matrix. *IEEE Transactions on Robotics and Automation*, 8(1):65–75, 1992.

N. Sadegh and R. Horowitz. Stability analysis of an adaptive controller for robotic manipulators. In *Proceedings IEEE Conference on Robotics and Automation*, pages 1223–1229, 1987.

Robert Sanner and Jean-Jacques Slotine. Stable adaptive control of robot manipulators using "neural" networks. *Neural Computation*, 7(4):753–790, 1995.

Jean-Jacques Slotine and Weiping Li. Tracking control of nonlinear systems using sliding surfaces, with application to robot manipulators. *International Journal of Control*, 38:465–492, 1983.

Jean-Jacques Slotine and Weiping Li. *Applied nonlinear control*. Prentice Hall, Englewood Cliffs, N.J., 1991.

Mark Spong and K. Vidyasagar. *Robot control and analysis*. John Wiley, 1989.

Mark W. Spong and M. Vidyasagar. *Robot dynamics and control*. John Wiley and Sons, 1989.

Y. Uno, M. Kawato, and R. Suzuki. Formation and control of optimal trajectory in human multijoint arm movement - minimum torque-change model. *Biological Cybernetics*, 61:89–101, 1989.

C. W. Wampler and L. J. Liefer. Applications of damped least-squares methods to resolved rate and resolved-acceleration control of manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 110:31–38, 1988.

John T. Wen and David S. Bayard. New class of control laws for robotic manipulators, Part 1: Non-adaptive case. *International Journal of Control*, 47(5):1361–1385, 1988.

Louis L. Whitcomb, Alfred A. Rizzi, and Daniel E. Koditschek. Comparative experiments with a new adaptive controller for robot arms. *IEEE Transactions on Robotics and Automation*, 9(1):59–69, 1993.