CHAPTER 10

Nonlinear control of manipulators

10.1 INTRODUCTION

In the previous chapter, we made several approximations to allow a linear analysis of the manipulator-control problem. Most important among these approximations was that each joint could be considered independent and that the inertia "seen" by each joint actuator was constant. In implementations of linear controllers as introduced in the previous chapter, this approximation results in nonuniform damping throughout the workspace and other undesirable effects. In this chapter, we will introduce a more advanced control technique for which this assumption will not have to be made.

In Chapter 9, we modeled the manipulator by n independent second-order differential equations and based our controller on that model. In this chapter, we will base our controller design directly on the $n \times 1$-nonlinear vector differential equation of motion, derived in Chapter 6 for a general manipulator.

The field of nonlinear control theory is large; we must therefore restrict our attention to one or two methods that seem well suited to mechanical manipulators. Consequently, the major focus of the chapter will be one particular method, apparently first proposed in [1] and named the **computed-torque method** in [2, 3]. We will also introduce one method of stability analysis of nonlinear systems, known as **Lyapunov's** method [4].

To begin our discussion of nonlinear techniques for controlling a manipulator, we return again to a very simple single-degree-of-freedom mass–spring friction system.

10.2 NONLINEAR AND TIME-VARYING SYSTEMS

In the preceding development, we dealt with a linear constant-coefficient differential equation. This mathematical form arose because the mass–spring friction system of Fig. 9.6 was modeled as a linear time-invariant system. For systems whose parameters vary in time or systems that by nature are nonlinear, solutions are more difficult.

When nonlinearities are not severe, **local linearization** can be used to derive linear models that are approximations of the nonlinear equations in the neighborhood of an **operating point**. Unfortunately, the manipulator-control problem is not well suited to this approach, because manipulators constantly move among regions of their workspaces so widely separated that no linearization valid for all regions can be found.

Another approach is to move the operating point with the manipulator as it moves, always linearizing about the desired position of the manipulator. The result of this sort of *moving linearization* is a linear, but time-varying, system. Although this quasi-static linearization of the original system is useful in some analysis and design techniques, we will not make use of it in our control-law synthesis procedure. Rather, we will deal with the nonlinear equations of motion directly and will not resort to linearizations in deriving a controller.

If the spring in Fig. 9.6 were not linear but instead contained a nonlinear element, we could consider the system quasi-statically and, at each instant, figure out where the poles of the system are located. We would find that the poles "move" around in the real–imaginary plane as a function of the position of the block. Hence, we could not select fixed gains that would keep the poles in a desirable location (for example, at critical damping). So we may be tempted to consider a more complicated control law, in which the gains are time-varying (actually, varying as a function of the block's position) in such a manner that the system is always critically damped. Essentially, this would be done by computing k_p such that the combination of the nonlinear effect of the spring would be exactly cancelled by a nonlinear term in the control law so that the overall stiffness would stay a constant at all times. Such a control scheme might be called a **linearizing** control law, because it uses a nonlinear control term to "cancel" a nonlinearity in the controlled system, so that the overall closed loop system is linear.

We will now return to our partitioned control law and see that it can perform this linearizing function. In our partitioned control-law scheme, the servo law remains the same as always, but the model-based portion now will contain a model of the nonlinearity. Thus, the model-based portion of the control performs a linearization function. This is best shown in an example.

EXAMPLE 10.1

Consider the nonlinear spring characteristic shown in Fig. 10.1. Rather than the usual linear spring relationship, $f = kx$, this spring is described by $f = qx^3$. If this spring is part of the physical system shown in Fig. 9.6, construct a control law to keep the system critically damped with a stiffness of k_{CL}.

The open-loop equation is

$$m\ddot{x} + b\dot{x} + qx^3 = f. \tag{10.1}$$

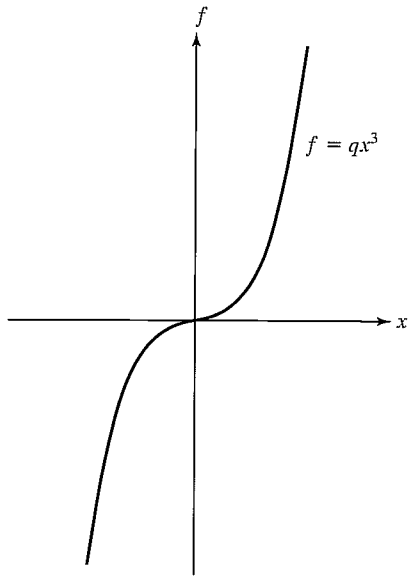

FIGURE 10.1: The force-vs.-distance characteristic of a nonlinear spring.

The model-based portion of the control is $f = \alpha f' + \beta$, where now we use

$$\alpha = m,$$
$$\beta = b\dot{x} + qx^3; \tag{10.2}$$

the servo portion is, as always

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e, \tag{10.3}$$

where the values of the gains are calculated from some desired performance specification. Figure 10.2 shows a block diagram of this control system. The resulting closed-loop system maintains poles in fixed locations.

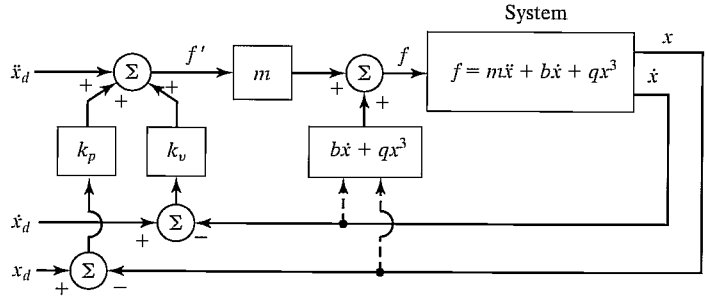

FIGURE 10.2: A nonlinear control system for a system with a nonlinear spring.

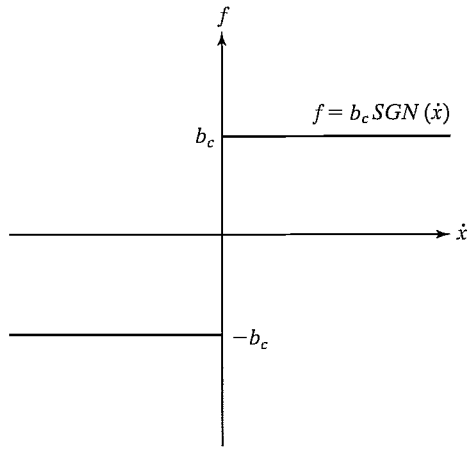

FIGURE 10.3: The force-vs.-velocity characteristic of Coulomb friction.

EXAMPLE 10.2

Consider the nonlinear friction characteristic shown in Fig. 10.3. Whereas linear friction is described by $f = b\dot{x}$, this **Coulomb friction** is described by $f = b_c sgn(\dot{x})$. For most of today's manipulators, the friction of the joint in its bearing (be it rotational or linear) is modeled more accurately by this nonlinear characteristic than by the simpler, linear model. If this type of friction is present in the system of Fig. 9.6, design a control system that uses a nonlinear model-based portion to damp the system critically at all times.

The open-loop equation is

$$m\ddot{x} + b_c sgn(\dot{x}) + kx = f. \tag{10.4}$$

The partitioned control law is $f = \alpha f' + \beta$, where

$$\begin{aligned} \alpha &= m, \\ \beta &= b_c sgn(\dot{x}) + kx, \\ f' &= \ddot{x}_d + k_v \dot{e} + k_p e, \end{aligned} \tag{10.5}$$

where the values of the gains are calculated from some desired performance specification.

EXAMPLE 10.3

Consider the single-link manipulator shown in Fig. 10.4. It has one rotational joint. The mass is considered to be located at a point at the distal end of the link, and so the moment of inertia is ml^2. There is Coulomb and viscous friction acting at the joint, and there is a load due to gravity.

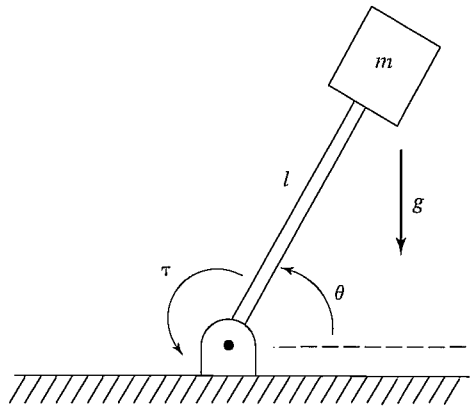

FIGURE 10.4: An inverted pendulum or a one-link manipulator.

The model of the manipulator is

$$\tau = ml^2\ddot{\theta} + v\dot{\theta} + csgn(\dot{\theta}) + mlg\cos(\theta). \tag{10.6}$$

As always, the control system has two parts, the linearizing model-based portion and the servo-law portion.

The model-based portion of the control is $f = \alpha f' + \beta$, where

$$\alpha = ml^2,$$
$$\beta = v\dot{\theta} + csgn(\dot{\theta}) + mlg\cos(\theta); \tag{10.7}$$

the servo portion is, as always,

$$f' = \ddot{\theta}_d + k_v\dot{e} + k_p e, \tag{10.8}$$

where the values of the gains are calculated from some desired performance specification.

We have seen that, in certain simple cases, it is not difficult to design a nonlinear controller. The general method used in the foregoing simple examples is the same method we will use for the problem of manipulator control:

1. Compute a nonlinear model-based control law that "cancels" the nonlinearities of the system to be controlled.
2. Reduce the system to a linear system that can be controlled with the simple linear servo law developed for the unit mass.

In some sense, the linearizing control law implements an *inverse model* of the system being controlled. The nonlinearities in the system cancel those in the inverse model; this, together with the servo law, results in a linear closed-loop system. Obviously, to do this cancelling, we must know the parameters and the structure of the nonlinear system. This is often a problem in practical application of this method.

10.3 MULTI-INPUT, MULTI-OUTPUT CONTROL SYSTEMS

Unlike the simple examples we have discussed in this chapter so far, the problem of controlling a manipulator is a multi-input, multi-output (MIMO) problem. That is, we have a *vector* of desired joint positions, velocities, and accelerations, and the control law must compute a *vector* of joint-actuator signals. Our basic scheme, partitioning the control law into a model-based portion and a servo portion, is still applicable, but it now appears in a matrix–vector form. The control law takes the form

$$F = \alpha F' + \beta, \tag{10.9}$$

where, for a system of n degrees of freedom, F, F', and β are $n \times 1$ vectors and α is an $n \times n$ matrix. Note that the matrix α is not necessarily diagonal, but rather is chosen to **decouple** the n equations of motion. If α and β are correctly chosen, then, from the F' input, the system appears to be n independent unit masses. For this reason, in the multidimensional case, the model-based portion of the control law is called a **linearizing and decoupling** control law. The servo law for a multidimensional system becomes

$$F' = \ddot{X}_d + K_v \dot{E} + K_p E, \tag{10.10}$$

where K_v and K_p are now $n \times n$ matrices, which are generally chosen to be diagonal with constant gains on the diagonal. E and $\dot{E}$ are $n \times 1$ vectors of the errors in position and velocity, respectively.

10.4 THE CONTROL PROBLEM FOR MANIPULATORS

In the case of manipulator control, we developed a model and the corresponding equations of motion in Chapter 6. As we saw, these equations are quite complicated. The rigid-body dynamics have the form

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta), \tag{10.11}$$

where $M(\Theta)$ is the $n \times n$ inertia matrix of the manipulator, $V(\Theta, \dot{\Theta})$ is an $n \times 1$ vector of centrifugal and Coriolis terms, and $G(\Theta)$ is an $n \times 1$ vector of gravity terms. Each element of $M(\Theta)$ and $G(\Theta)$ is a complicated function that depends on Θ, the position of all the joints of the manipulator. Each element of $V(\Theta, \dot{\Theta})$ is a complicated function of both Θ and $\dot{\Theta}$.

Additionally, we could incorporate a model of friction (or other non-rigid-body effects). Assuming that our model of friction is a function of joint positions and velocities, we add the term $F(\Theta, \dot{\Theta})$ to (10.11), to yield the model

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}). \tag{10.12}$$

The problem of controlling a complicated system like (10.12) can be handled by the partitioned controller scheme we have introduced in this chapter. In this case, we have

$$\tau = \alpha\tau' + \beta, \tag{10.13}$$

where τ is the $n \times 1$ vector of joint torques. We choose

$$\begin{aligned} \alpha &= M(\Theta), \\ \beta &= V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}), \end{aligned} \tag{10.14}$$

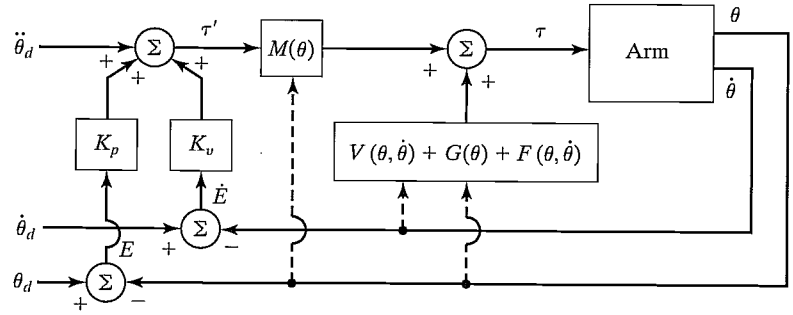

FIGURE 10.5: A model-based manipulator-control system.

with the servo law

$$\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_pE, \tag{10.15}$$

where

$$E = \Theta_d - \Theta. \tag{10.16}$$

The resulting control system is shown in Fig. 10.5.

Using (10.12) through (10.15), it is quite easy to show that the closed-loop system is characterized by the error equation

$$\ddot{E} + K_v\dot{E} + K_pE = 0. \tag{10.17}$$

Note that this vector equation is decoupled: The matrices K_v and K_p are diagonal, so that (10.17) could just as well be written on a joint-by-joint basis as

$$\ddot{e}_i + k_{vi}\dot{e} + k_{pi}e = 0. \tag{10.18}$$

The ideal performance represented by (10.17) is unattainable in practice, for many reasons, the most important two being

1. The discrete nature of a digital-computer implementation, as opposed to the ideal continuous-time control law implied by (10.14) and (10.15).
2. Inaccuracy in the manipulator model (needed to compute (10.14)).

In the next section, we will (at least partially) address these two issues.

10.5 PRACTICAL CONSIDERATIONS

In developing the decoupling and linearizing control in the last few sections, we have implicitly made a few assumptions that rarely are true in practice.

Time required to compute the model

In all our considerations of the partitioned-control-law strategy, we have implicitly assumed that the entire system was running in continuous time and that the computations in the control law require zero time for their computation. Given any amount of computation, with a large enough computer we can do the computations sufficiently

fast that this is a reasonable approximation; however, the expense of the computer could make the scheme economically unfeasible. In the manipulator-control case, the entire dynamic equation of the manipulator, (10.14), must be computed in the control law. These computations are quite involved; consequently, as was discussed in Chapter 6, there has been a great deal of interest in developing fast computational schemes to compute them in an efficient way. As computer power becomes more and more affordable, control laws that require a great deal of computation will become more practical. Several experimental implementations of nonlinear-model-based control laws have been reported [5–9], and partial implementations are beginning to appear in industrial controllers.

As was discussed in Chapter 9, almost all manipulator-control systems are now performed in digital circuitry and are run at a certain **sampling rate**. This means that the position (and possibly other) sensors are read at discrete points in time. From the values read, an actuator command is computed and sent to the actuator. Thus, reading sensors and sending actuator commands are not done continuously, but rather at a finite sampling rate. To analyze the effect of delay due to computation and finite sample rate, we must use tools from the field of **discrete-time control**. In discrete time, differential equations turn into difference equations, and a complete set of tools has been developed to answer questions about stability and pole placement for these systems. Discrete-time control theory is beyond the scope of this book, although, for researchers working in the area of manipulator control, many of the concepts from discrete-time systems are essential. (See [10].)

Although important, ideas and methods from discrete-time control theory are often difficult to apply to the case of nonlinear systems. Whereas we have managed to write a complicated differential equation of motion for the manipulator dynamic equation, a discrete-time equivalent is impossible to obtain in general because, for a general manipulator, the only way to solve for the motion of the manipulator for a given set of initial conditions, an input, and a finite interval is by numerical integration (as we saw in Chapter 6). Discrete-time models are possible if we are willing to use series solutions to the differential equations, or if we make approximations. However, if we need to make approximations to develop a discrete model, then it is not clear whether we have a better model than we have when just using the continuous model and making the continuous-time approximation. Suffice it to say that analysis of the discrete-time manipulator-control problem is difficult, and usually simulation is resorted to in order to judge the effect that a certain sample rate will have on performance.

We will generally assume that the computations can be performed quickly enough and often enough that the continuous-time approximation is valid.

Feedforward nonlinear control

The use of **feedforward control** has been proposed as a method of using a nonlinear dynamic model in a control law without the need for complex and time-consuming computations to be performed at servo rates [11]. In Fig. 10.5, the model-based control portion of the control law is "in the servo loop" in that signals "flow" through that black box with each tick of the servo clock. If we wish to select a sample

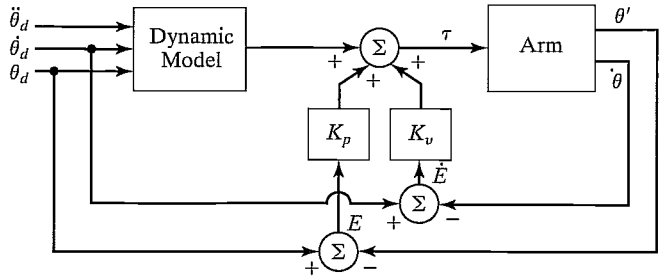

FIGURE 10.6: Control scheme with the model-based portion "outside" the servo loop.

rate of 200 Hz, then the dynamic model of the manipulator must be computed at this rate. Another possible control system is shown in Fig. 10.6. Here, the model-based control is "outside" the servo loop. Hence, it is possible to have a fast inner servo loop, consisting simply of multiplying errors by gains, with the model-based torques added at a slower rate.

Unfortunately, the feedforward scheme of Fig. 10.6 does not provide complete decoupling. If we write the system equations,[1] we will find that the error equation of this system is

$$\ddot{E} + M^{-1}(\Theta)K_v\dot{E} + M^{-1}(\Theta)K_pE = 0. \tag{10.19}$$

Clearly, as the configuration of the arm changes, the effective closed-loop gain changes, and the quasi-static poles move around in the real–imaginary plane. However, equation (10.19) could be used as a starting point for designing a **robust controller**—one that finds a good set of constant gains such that, despite the "motion" of the poles, they are guaranteed to remain in reasonably favorable locations. Alternatively, one might consider schemes in which variable gains are precomputed which change with configuration of the robot, so that the system's quasi-static poles remain in fixed positions.

Note that, in the system of Fig. 10.6, the dynamic model is computed as a function of the desired path only, so when the desired path is known in advance, values could be computed "off-line" before motion begins. At run time, the precomputed torque histories would then be read out of memory. Likewise, if time-varying gains are computed, they too could be computed beforehand and stored. Hence, such a scheme could be quite inexpensive computationally at run time and thus achieve a high servo rate.

Dual-rate computed-torque implementation

Figure 10.7 shows the block diagram of a possible practical implementation of the decoupling and linearizing position-control system. The dynamic model is expressed in its *configuration space* form so that the dynamic parameters of the manipulator will appear as functions of manipulator position only. These functions might then

[1] We have used the simplifying assumptions $M(\Theta_d) \cong M(\Theta)$, $V(\Theta_d, \dot{\Theta}_d) \cong (V(\Theta, \dot{\Theta})$, $G(\Theta_d) \cong G(\Theta)$, and $F(\Theta_d, \dot{\Theta}_d) \cong F(\Theta, \dot{\Theta})$.

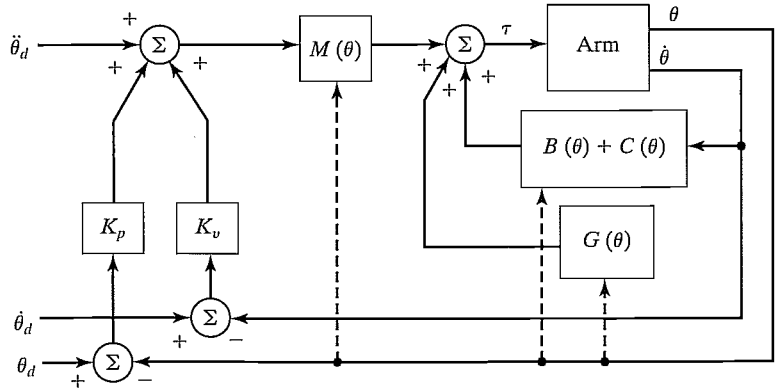

FIGURE 10.7: An implementation of the model-based manipulator-control system.

be computed by a *background* process or by a second control computer [8] or be looked up in a precomputed table [12]. In this architecture, the dynamic parameters can be updated at a rate slower than the rate of the closed-loop servo. For example, the background computation might proceed at 60 Hz while the closed-loop servo was running at 250 Hz.

Lack of knowledge of parameters

The second potential difficulty encountered in employing the computed-torque control algorithm is that the manipulator dynamic model is often not known accurately. This is particularly true of certain components of the dynamics, such as friction effects. In fact, it is usually extremely difficult to know the structure of the friction model, let alone the parameter values [13]. Finally, if the manipulator has some portion of its dynamics that is not repeatable—because, for example, it changes as the robot ages—it is difficult to have good parameter values in the model at all times.

By nature, most robots will be picking up various parts and tools. When a robot is holding a tool, the inertia and the weight of the tool change the dynamics of the manipulator. In an industrial situation, the mass properties of the tools might be known—in this case, they can be accounted for in the modeled portion of the control law. When a tool is grasped, the inertia matrix, total mass, and center of mass of the last link of the manipulator can be updated to new values that represent the combined effect of the last link plus tool. However, in many applications, the mass properties of objects that the manipulator picks up are not generally known, so maintenance of an accurate dynamic model is difficult.

The simplest possible nonideal situation is one in which we still assume a perfect model implemented in continuous time, but with external noise acting to disturb the system. In Fig. 10.8, we indicate a vector of disturbance torques acting at the joints. Writing the system error equation with inclusion of these unknown disturbances, we arrive at

$$\ddot{E} + K_v \dot{E} + K_p E = M^{-1}(\Theta)\tau_d, \tag{10.20}$$

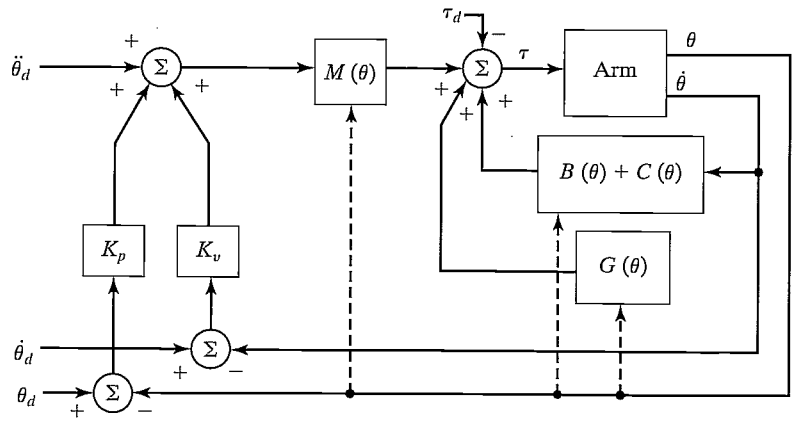

FIGURE 10.8: The model-based controller with an external disturbance acting.

where τ_d is the vector of disturbance torques at the joints. The left-hand side of (10.20) is uncoupled, but, from the right-hand side, we see that a disturbance on any particular joint will introduce errors at all the other joints, because $M(\Theta)$ is not, in general, diagonal.

Some simple analyses might be performed on the basis of (10.20). For example, it is easy to compute the steady-state servo error due to a constant disturbance as

$$E = K_p^{-1} M^{-1}(\Theta) \tau_d. \tag{10.21}$$

When our model of the manipulator dynamics is not perfect, analysis of the resulting closed-loop system becomes more difficult. We define the following notation: $\hat{M}(\Theta)$ is our model of the manipulator inertia matrix, $M(\Theta)$. Likewise, $\hat{V}(\Theta, \dot{\Theta})$, $\hat{G}(\Theta)$, and $\hat{F}(\Theta, \dot{\Theta})$ are our models of the velocity terms, gravity terms, and friction terms of the actual mechanism. Perfect knowledge of the model would mean that

$$\begin{aligned} \hat{M}(\Theta) &= M(\Theta), \\ \hat{V}(\Theta, \dot{\Theta}) &= V(\Theta, \dot{\Theta}), \\ \hat{G}(\Theta) &= G(\Theta), \\ \hat{F}(\Theta, \dot{\Theta}) &= F(\Theta, \dot{\Theta}). \end{aligned} \tag{10.22}$$

Therefore, although the manipulator dynamics are given by

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + F(\Theta, \dot{\Theta}), \tag{10.23}$$

our control law computes

$$\begin{aligned} \tau &= \alpha\tau' + \beta, \\ \alpha &= \hat{M}(\Theta), \\ \beta &= \hat{V}(\Theta, \dot{\Theta}) + \hat{G}(\Theta) + \hat{F}(\Theta, \dot{\Theta}). \end{aligned} \tag{10.24}$$

Decoupling and linearizing will not, therefore, be perfectly accomplished when parameters are not known exactly. Writing the closed-loop equation for the system, we have

$$\ddot{E} + K_v\dot{E} + K_pE = \hat{M}^{-1}[(M - \hat{M})\ddot{\Theta} + (V - \hat{V}) + (G - \hat{G}) + (F - \hat{F})], \tag{10.25}$$

where the arguments of the dynamic functions are not shown for brevity. Note that, if the model were exact, so that (10.22) were true, then the right-hand side of (10.25) would be zero and the errors would disappear. When the parameters are not known exactly, the mismatch between actual and modeled parameters will cause servo errors to be excited (possibly even resulting in an unstable system [21]) according to the rather complicated equation (10.25).

Discussion of stability analysis of a nonlinear closed-loop system is deferred until Section 10.7.

10.6 CURRENT INDUSTRIAL-ROBOT CONTROL SYSTEMS

Because of the problems with having good knowledge of parameters, it is not clear whether it makes sense to go to the trouble of computing a complicated model-based control law for manipulator control. The expense of the computer power needed to compute the model of the manipulator at a sufficient rate might not be worthwhile, especially when lack of knowledge of parameters could nullify the benefits of such an approach. Manufacturers of industrial robots have decided, probably for economic reasons, that attempting to use a complete manipulator model in the controller is not worthwhile. Instead, present-day manipulators are controlled with very simple control laws that generally are completely error driven and are implemented in architectures such as those studied in Section 9.10. An industrial robot with a high-performance servo system is shown in Fig. 10.9.

Individual-joint PID control

Most industrial robots nowadays have a control scheme that, in our notation, would be described by

$$\alpha = I, \qquad \beta = 0, \tag{10.26}$$

where I is the $n \times n$ identity matrix. The servo portion is

$$\tau' = \ddot{\Theta}_d + K_v\dot{E} + K_pE + K_i\int E dt, \tag{10.27}$$

where K_v, K_p, and K_i are constant diagonal matrices. In many cases, $\ddot{\Theta}_d$ is not available, and this term is simply set to zero. That is, most simple robot controllers do not use a model-based component *at all* in their control law. This type of PID control scheme is simple because each joint is controlled as a separate control system. Often, one microprocessor per joint is used to implement (10.27), as was discussed in Section 9.10.

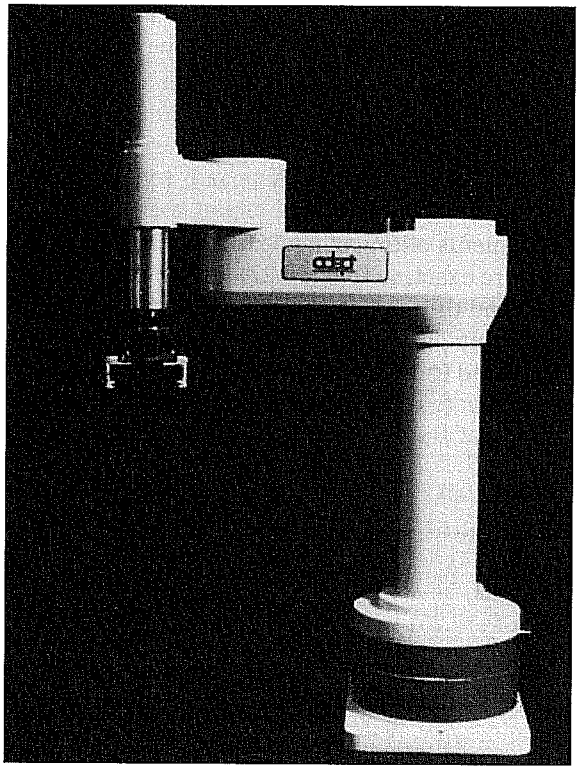

FIGURE 10.9: The Adept One, a direct-drive robot by Adept Technology, Inc.

The performance of a manipulator controlled in this way is not simple to describe. No decoupling is being done, so the motion of each joint affects the other joints. These interactions cause errors, which are suppressed by the error-driven control law. It is impossible to select fixed gains that will critically damp the response to disturbances for all configurations. Therefore, "average" gains are chosen, which approximate critical damping in the center of the robot's workspace. In various extreme configurations of the arm, the system becomes either underdamped or overdamped. Depending on the details of the mechanical design of the robot, these effects could be fairly small; then control would be good. In such systems, it is important to keep the gains as high as possible, so that the inevitable disturbances will be suppressed quickly.

Addition of gravity compensation

The gravity terms will tend to cause static positioning errors, so some robot manufacturers include a gravity model, $G(\theta)$, in the control law (that is, $\beta = \hat{G}(\Theta)$ in our notation). The complete control law takes the form

$$\tau' = \ddot{\Theta}_d + K_v \dot{E} + K_p E + K_i \int E dt + \hat{G}(\Theta). \tag{10.28}$$

Such a control law is perhaps the simplest example of a model-based controller. Because (10.28) can no longer be implemented on a strict joint-by-joint basis, the controller architecture must allow communication between the joint controllers or must make use of a central processor rather than individual-joint processors.

Various approximations of decoupling control

There are various ways to simplify the dynamic equations of a particular manipulator [3,14]. After the simplification, an approximate decoupling and linearizing law can be derived. A usual simplification might be to disregard components of torque due to the velocity terms—that is, to model only the inertial and gravity terms. Often, friction models are not included in the controller, because friction is so hard to model correctly. Sometimes, the inertia matrix is simplified so that it accounts for the major coupling between axes but not for minor cross-coupling effects. For example, [14] presents a simplified version of the PUMA 560's mass matrix that requires only about 10% of the calculations needed to compute the complete mass matrix, yet is accurate to within 1%.

10.7 LYAPUNOV STABILITY ANALYSIS

In Chapter 9, we examined linear control systems analytically to evaluate stability and also performance of the dynamic response in terms of damping and closed-loop bandwidth. The same analyses are valid for a nonlinear system that has been decoupled and linearized by means of a perfect model-based nonlinear controller, because the overall resulting system is again linear. However, when decoupling and linearizing are not performed by the controller, or are incomplete or inaccurate, the overall closed-loop system remains nonlinear. For nonlinear systems, stability and performance analysis is much more difficult. In this section, we introduce one method of stability analysis that is applicable to both linear and nonlinear systems.

Consider the simple mass–spring friction system originally introduced in Chapter 9, whose equation of motion is

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{10.29}$$

The total energy of the system is given by

$$v = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2, \tag{10.30}$$

where the first term gives the kinetic energy of the mass and the second term gives the potential energy stored in the spring. Note that the value, v, of the system energy is always nonnegative (i.e., it is positive or zero). Let's find out the *rate* of change of the total energy by differentiating (10.30) with respect to time, to obtain

$$\dot{v} = m\dot{x}\ddot{x} + kx\dot{x}. \tag{10.31}$$

Substituting (10.29) for $m\ddot{x}$ in (10.31) yields

$$\dot{v} = -b\dot{x}^2, \tag{10.32}$$

which we note is always nonpositive (because $b > 0$). Thus, energy is always leaving the system, unless $\dot{x} = 0$. This implies that, however initially perturbed, the system

will lose energy until it comes to rest. Investigating possible resting positions by means of a steady-state analysis of (10.29) yields

$$kx = 0, \tag{10.33}$$

or

$$x = 0. \tag{10.34}$$

Hence, by means of an energy analysis, we have shown that the system of (10.29) with any initial conditions (i.e., any initial energy) will eventually come to rest at the equilibrium point. This stability proof by means of an energy analysis is a simple example of a more general technique called **Lyapunov stability analysis** or **Lyapunov's second** (or **direct**) **method**, after a Russian mathematician of the 19th century [15].

An interesting feature of this method of stability analysis is that we can conclude stability without solving for the solution of the differential equation governing the system. However, while Lyapunov's method is useful for examining *stability*, it generally does not provide any information about the transient response or *performance* of the system. Note that our energy analysis yielded no information on whether the system was overdamped or underdamped or on how long it would take the system to suppress a disturbance. It is important to distinguish between stability and performance: A stable system might nonetheless exhibit control performance unsatisfactory for its intended use.

Lyapunov's method is somewhat more general than our example indicated. It is one of the few techniques that can be applied directly to nonlinear systems to investigate their stability. As a means of quickly getting an idea of Lyapunov's method (in sufficient detail for our needs), we will look at an extremely brief introduction to the theory and then proceed directly to several examples. A more complete treatment of Lyapunov theory can be found in [16, 17].

Lyapunov's method is concerned with determining the stability of a differential equation

$$\dot{X} = f(X), \tag{10.35}$$

where X is $m \times 1$ and $f(\cdot)$ could be nonlinear. Note that higher order differential equations can always be written as a set of first-order equations in the form (10.35). To prove a system stable by Lyapunov's method, one is required to propose a generalized energy function $v(X)$ that has the following properties:

1. $v(X)$ has continuous first partial derivatives, and $v(X) > 0$ for all X except $v(0) = 0$.
2. $\dot{v}(X) \leq 0$. Here, $\dot{v}(X)$ means the change in $v(X)$ along all system trajectories.

These properties might hold only in a certain region, or they might be global, with correspondingly weaker or stronger stability results. The intuitive idea is that a positive definite "energy-like" function of state is shown to always decrease or remain constant—hence, the system is stable in the sense that the size of the state vector is bounded.

When $\dot{v}(X)$ is strictly less than zero, asymptotic convergence of the state to the zero vector can be concluded. Lyapunov's original work was extended in an

important way by LaSalle and Lefschetz [4], who showed that, in certain situations, even when $\dot{v}(X) \leq 0$ (note equality included), asymptotic stability can be shown. For our purposes, we can deal with the case $\dot{v}(X) = 0$ by performing a steady-state analysis in order to learn whether the stability is asymptotic or the system under study can "get stuck" somewhere other than $v(X) = 0$.

A system described by (10.35) is said to be **autonomous** because the function $f(\cdot)$ is not an explicit function of time. Lyapunov's method also extends to **nonautonomous** systems, in which time is an argument of the nonlinear function. See [4, 17] for details.

EXAMPLE 10.4

Consider the linear system

$$\dot{X} = -AX, \tag{10.36}$$

where A is $m \times m$ and positive definite. Propose the **candidate Lyapunov function**

$$v(X) = \frac{1}{2}X^T X, \tag{10.37}$$

which is continuous and everywhere nonnegative. Differentiating yields

$$\begin{aligned} \dot{v}(X) &= X^T \dot{X} \\ &= X^T(-AX) \\ &= -X^T AX, \end{aligned} \tag{10.38}$$

which is everywhere nonpositive because A is a positive definite matrix. Hence, (10.37) is indeed a Lyapunov function for the system of (10.36). The system is asymptotically stable because $\dot{v}(X)$ can be zero only at $X = 0$; everywhere else, X must decrease.

EXAMPLE 10.5

Consider a mechanical spring–damper system in which both the spring and damper are nonlinear:

$$\ddot{x} + b(\dot{x}) + k(x) = 0. \tag{10.39}$$

The functions $b(\cdot)$ and $k(\cdot)$ are first- and third-quadrant continuous functions such that

$$\begin{aligned} \dot{x}b(\dot{x}) &> 0 \; for \; x \neq 0, \\ xk(x) &> 0 \; for \; x \neq 0. \end{aligned} \tag{10.40}$$

Once having proposed the Lyapunov function

$$v(x, \dot{x}) = \frac{1}{2}\dot{x}^2 + \int_0^x k(\lambda)d\lambda, \tag{10.41}$$

we are led to

$$\begin{aligned}\dot{v}(x,\dot{x}) &= \dot{x}\ddot{x} + k(x)\dot{x},\\ &= -\dot{x}b(\dot{x}) - k(x)\dot{x} + k(x)\dot{x},\\ &= -\dot{x}b(\dot{x}).\end{aligned} \tag{10.42}$$

Hence, $\dot{v}(\cdot)$ is nonpositive but is only semidefinite, because it is not a function of x but only of $\dot{x}$. In order to conclude asymptotic stability, we have to ensure that it is not possible for the system to "get stuck" with nonzero x. To study all trajectories for which $\dot{x} = 0$, we must consider

$$\ddot{x} = -k(x), \tag{10.43}$$

for which $x = 0$ is the only solution. Hence, the system will come to rest only if $x = \dot{x} = \ddot{x} = 0$.

EXAMPLE 10.6

Consider a manipulator with dynamics given by

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) \tag{10.44}$$

and controlled with the control law

$$\tau = K_p E - K_d\dot{\Theta} + G(\Theta), \tag{10.45}$$

where K_p and K_d are diagonal gain matrices. Note that this controller does not force the manipulator to follow a trajectory, but moves the manipulator to a goal point along a path specified by the manipulator dynamics and then regulates the position there. The resulting closed-loop system obtained by equating (10.44) and (10.45) is

$$M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + K_d\dot{\Theta} + K_p\Theta = K_p\Theta_d; \tag{10.46}$$

it can be proven globally asymptotically stable by Lyapunov's method [18, 19].

Consider the candidate Lyapunov function

$$v = \frac{1}{2}\dot{\Theta}^T M(\Theta)\dot{\Theta} + \frac{1}{2}E^T K_p E. \tag{10.47}$$

The function (10.47) is always positive or zero, because the manipulator mass matrix, $M(\Theta)$, and the position gain matrix, K_p, are positive definite matrices. Differentiating (10.47) yields

$$\begin{aligned}\dot{v} &= \frac{1}{2}\dot{\Theta}^T\dot{M}(\Theta)\dot{\theta} + \dot{\theta}^T M(\theta)\ddot{\Theta} - E^T K_p\dot{\Theta}\\ &= \frac{1}{2}\dot{\Theta}^T\dot{M}(\Theta)\dot{\Theta} - \dot{\Theta}^T K_d\dot{\Theta} - \dot{\Theta}^T V(\Theta, \dot{\Theta})\\ &= -\dot{\Theta}^T K_d\dot{\Theta},\end{aligned} \tag{10.48}$$

which is nonpositive as long as K_d is positive definite. In taking the last step in (10.48), we have made use of the interesting identity

$$\frac{1}{2}\dot{\Theta}^T \dot{M}(\Theta)\dot{\Theta} = \dot{\Theta}^T V(\Theta, \dot{\Theta}), \tag{10.49}$$

which can be shown by investigation of the structure of Lagrange's equations of motion [18–20]. (See also Exercise 6.17.)

Next, we investigate whether the system can get "stuck" with nonzero error. Because $\dot{v}$ can remain zero only along trajectories that have $\dot{\Theta} = 0$ and $\ddot{\Theta} = 0$, we see from (10.46) that, in this case,

$$K_p E = 0, \tag{10.50}$$

and because K_p is nonsingular, we have

$$E = 0. \tag{10.51}$$

Hence, control law (10.45) applied to the system (10.44) achieves global asymptotic stability.

This proof is important in that it explains, to some extent, why today's industrial robots work. Most industrial robots use a simple error-driven servo, occasionally with gravity models, and so are quite similar to (10.45).

See Exercises 10.11 through 10.16 for more examples of nonlinear manipulator-control laws that can be proven stable by Lyapunov's method. Recently, Lyapunov theory has become increasingly prevalent in robotics research publications [18–25].

10.8 CARTESIAN-BASED CONTROL SYSTEMS

In this section, we introduce the notion of **Cartesian-based control**. Although such approaches are not currently used in industrial robots, there is activity at several research institutions on such schemes.

Comparison with joint-based schemes

In all the control schemes for manipulators we have discussed so far, we assumed that the desired trajectory was available in terms of time histories of joint position, velocity, and acceleration. Given that these desired inputs were available, we designed **joint-based control** schemes, that is, schemes in which we develop trajectory errors by finding the difference between desired and actual quantities expressed in joint space. Very often, we wish the manipulator end-effector to follow straight lines or other path shapes described in Cartesian coordinates. As we saw in Chapter 7, it is possible to compute the time histories of the joint-space trajectory that correspond to Cartesian straight-line paths. Figure 10.10 shows this approach to manipulator-trajectory control. A basic feature of the approach is the **trajectory-conversion** process, which is used to compute the joint trajectories. This is then followed by some kind of joint-based servo scheme such as we have been studying.

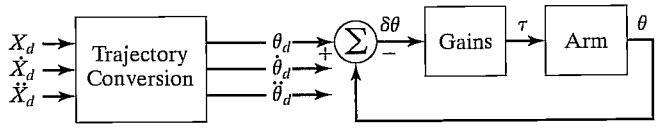

FIGURE 10.10: A joint-based control scheme with Cartesian-path input.

The trajectory-conversion process is quite difficult (in terms of computational expense) if it is to be done analytically. The computations that would be required are

$$\begin{aligned} \Theta_d &= INVKIN(\chi_d), \\ \dot{\Theta}_d &= J^{-1}(\Theta)\dot{\chi}_d, \\ \ddot{\Theta}_d &= \dot{J}^{-1}(\Theta)\dot{\chi}_d + J^{-1}(\Theta)\ddot{\chi}_d. \end{aligned} \tag{10.52}$$

To the extent that such a computation is done at all in present-day systems, usually just the solution for Θ_d is performed, by using the inverse kinematics, and then the joint velocities and accelerations are computed numerically by first and second differences. However, such numerical differentiation tends to amplify noise and introduces a lag unless it can be done with a noncausal filter.[2] Therefore, we are interested in either finding a less computationally expensive way of computing (10.52) or suggesting a control scheme in which this information is not needed.

An alternative approach is shown in Fig. 10.11. Here, the sensed position of the manipulator is immediately transformed by means of the kinematic equations into a Cartesian description of position. This Cartesian description is then compared to the desired Cartesian position in order to form errors in Cartesian space. Control schemes based on forming errors in Cartesian space are called **Cartesian-based control** schemes. For simplicity, velocity feedback is not shown in Fig. 10.11, but it would be present in any implementation.

The trajectory-conversion process is replaced by some kind of coordinate conversion inside the servo loop. Note that Cartesian-based controllers must perform many computations in the loop; the kinematics and other transformations are now "inside the loop." This can be a drawback of the Cartesian-based methods; the resulting system could run at a lower sampling frequency compared to joint-based

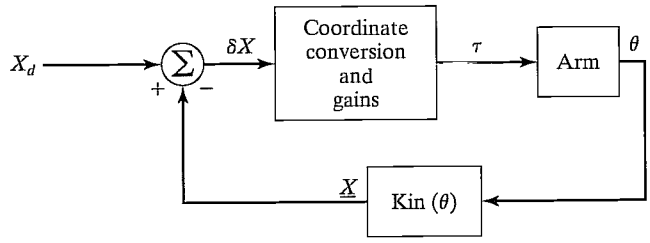

FIGURE 10.11: The concept of a Cartesian-based control scheme.

[2]Numerical differentiation introduces a lag unless it can be based on past, present, and future values. When the entire path is preplanned, this kind of noncausal numerical differentiation can be done.

systems (given the same size of computer). This would, in general, degrade the stability and disturbance-rejection capabilities of the system.

Intuitive schemes of Cartesian control

One possible control scheme that comes to mind rather intuitively is shown in Fig. 10.12. Here, Cartesian position is compared to the desired position to form an error, δX, in Cartesian space. This error, which may be presumed small if the control system is doing its job, may be mapped into a small displacement in joint space by means of the inverse Jacobian. The resulting errors in joint space, $\delta\theta$, are then multiplied by gains to compute torques that will tend to reduce these errors. Note that Fig. 10.12 shows a simplified controller in which, for clarity, the velocity feedback has not been shown. It could be added in a straightforward manner. We will call this scheme the **inverse-Jacobian controller**.

Another scheme which could come to mind is shown in Fig. 10.13. Here, the Cartesian error vector is multiplied by a gain to compute a Cartesian force vector. This can be thought of as a Cartesian force which, if applied to the end-effector of the robot, would push the end-effector in a direction that would tend to reduce the Cartesian error. This Cartesian force vector (actually a force–moment vector) is then mapped through the Jacobian transpose in order to compute the equivalent joint torques that would tend to reduce the observed errors. We will call this scheme the **transpose-Jacobian controller**.

The inverse-Jacobian controller and the transpose-Jacobian controller have both been arrived at intuitively. We cannot be sure that such arrangements would be stable, let alone perform well. It is also curious that the schemes are extremely similar, except that the one contains the Jacobian's inverse, the other its transpose. Remember, the inverse is not equal to the transpose in general (only in the case of a strictly Cartesian manipulator does $J^T = J^{-1}$). The exact dynamic performance

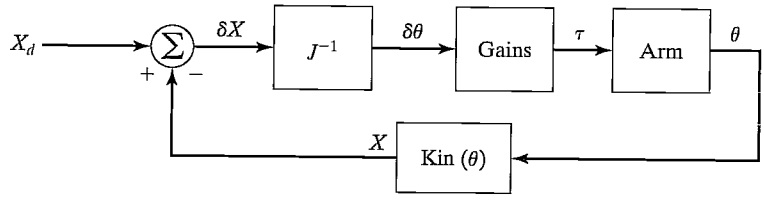

FIGURE 10.12: The inverse-Jacobian Cartesian-control scheme.

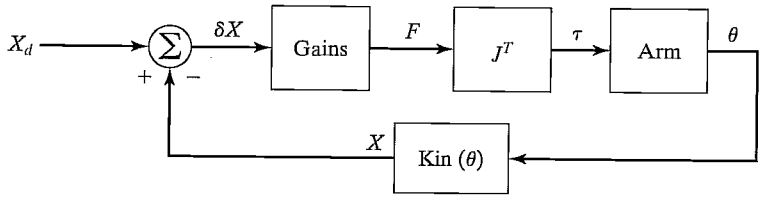

FIGURE 10.13: The transpose-Jacobian Cartesian-control scheme.

of such systems (if expressed in a second-order error-space equation, for example) is very complicated. It turns out that both schemes will work (i.e., can be made stable), but not well (i.e., performance is not good over the entire workspace). Both can be made stable by appropriate gain selection, including some form of velocity feedback (which was not shown in Figs. 10.12 and 10.13). While both will work, neither is *correct*, in the sense that we cannot choose fixed gains that will result in fixed closed-loop poles. The dynamic response of such controllers will vary with arm configuration.

Cartesian decoupling scheme

For Cartesian-based controllers, like joint-based controllers, good performance would be characterized by constant error dynamics over all configurations of the manipulator. Errors are expressed in Cartesian space in Cartesian-based schemes, so this means that we would like to design a system which, over all possible configurations, would suppress Cartesian errors in a critically damped fashion.

Just as we achieved good control with a joint-based controller that was based on a linearizing and decoupling model of the arm, we can do the same for the Cartesian case. However, we must now write the dynamic equations of motion of the manipulator in terms of Cartesian variables. This can be done, as was discussed in Chapter 6. The resulting form of the equations of motion is quite analogous to the joint-space version. The rigid-body dynamics can be written as

$$\mathcal{F} = M_x(\Theta)\ddot{\chi} + V_x(\Theta, \dot{\Theta}) + G_x(\Theta), \tag{10.53}$$

where $\mathcal{F}$ is a fictitious force–moment vector acting on the end-effector of the robot and χ is an appropriate Cartesian vector representing position and orientation of the end-effector [8]. Analogous to the joint-space quantities, $M_x(\Theta)$ is the mass matrix in Cartesian space, $V_x(\Theta, \dot{\Theta})$ is a vector of velocity terms in Cartesian space, and $G_x(\Theta)$ is a vector of gravity terms in Cartesian space.

Just as we did in the joint-based case, we can use the dynamic equations in a decoupling and linearizing controller. Because (10.53) computes $\mathcal{F}$, a fictitious Cartesian force vector which should be applied to the hand, we will also need to use the transpose of the Jacobian in order to implement the control—that is, after $\mathcal{F}$ is calculated by (10.53), we cannot actually cause a Cartesian force to be applied to the end-effector; we instead compute the joint torques needed to effectively balance the system if we were to apply this force:

$$\tau = J^T(\Theta)\mathcal{F}. \tag{10.54}$$

Figure 10.14 shows a Cartesian arm-control system using complete dynamic decoupling. Note that the arm is preceded by the Jacobian transpose. Notice that the controller of Fig. 10.14 allows Cartesian paths to be described directly, with no need for trajectory conversion.

As in the joint-space case, a practical implementation might best be achieved through use of a dual-rate control system. Figure 10.15 shows a block diagram of a Cartesian-based decoupling and linearizing controller in which the dynamic parameters are written as functions of manipulator position only. These dynamic parameters are updated at a rate slower than the servo rate by a background

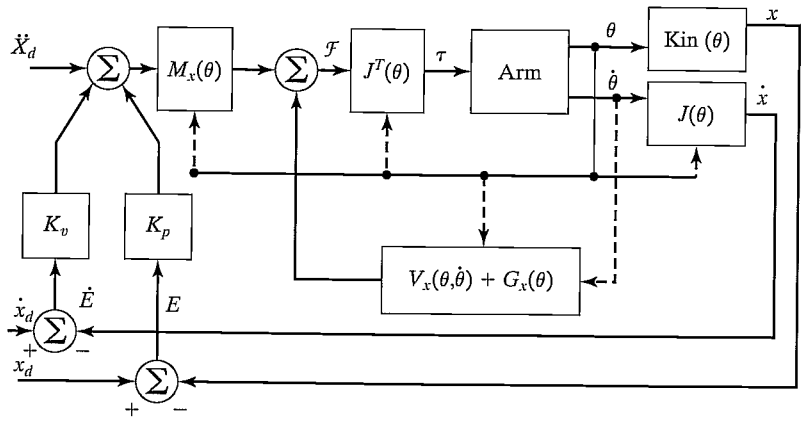

FIGURE 10.14: The Cartesian model-based control scheme.

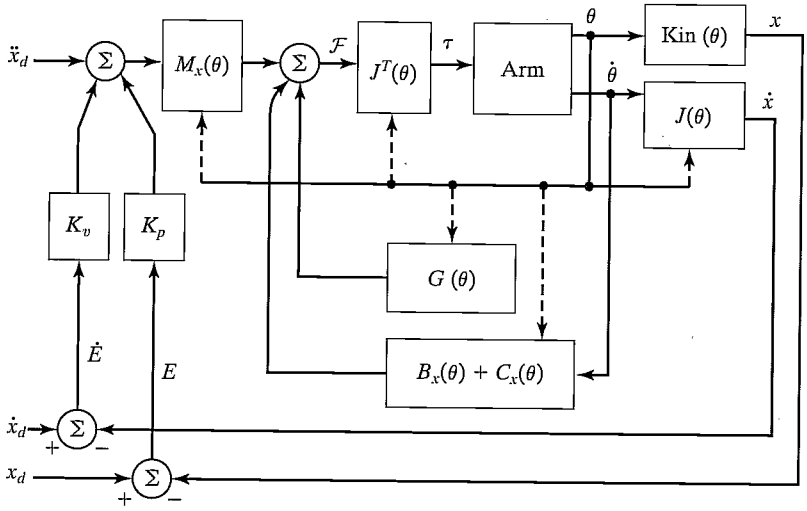

FIGURE 10.15: An implementation of the Cartesian model-based control scheme.

process or a second control computer. This is appropriate, because we desire a fast servo (perhaps running at 500 Hz or even higher) to maximize disturbance rejection and stability. The dynamic parameters are functions of manipulator position only, so they need be updated at a rate related only to how fast the manipulator is changing configuration. The parameter-update rate probably need not be higher than 100 Hz [8].

10.9 ADAPTIVE CONTROL

In the discussion of model-based control, it was noted that, often, parameters of the manipulator are not known exactly. When the parameters in the model do not

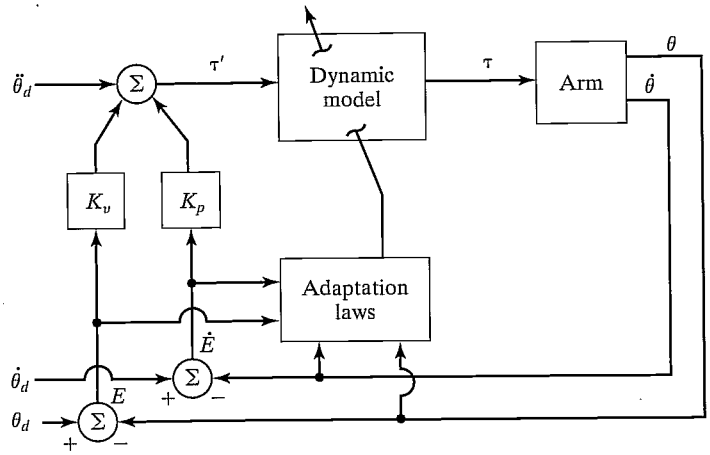

FIGURE 10.16: The concept of an adaptive manipulator controller.

match the parameters of the real device, servo errors will result, as is made explicit in (10.25). These servo errors could be used to drive some adaptation scheme that attempts to update the values of the model parameters until the errors disappear. Several such adaptive schemes have been proposed.

An ideal adaptive scheme might be like the one in Fig. 10.16. Here, we are using a model-based control law as developed in this chapter. There is an adaptation process that, given observations of manipulator state and servo errors, readjusts the parameters in the nonlinear model until the errors disappear. Such a system would *learn* its own dynamic properties. The design and analysis of adaptive schemes are beyond the scope of this book. A method that possesses exactly the structure shown in Fig. 10.16 and has been proven globally stable is presented in [20, 21]. A related technique is that of [22].

BIBLIOGRAPHY

[1] R.P. Paul, "Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm," Technical Report AIM-177, Stanford University Artificial Intelligence Laboratory, 1972.

[2] B. Markiewicz, "Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator," Jet Propulsion Laboratory Technical Memo 33–601, March 1973.

[3] A. Bejczy, "Robot Arm Dynamics and Control," Jet Propulsion Laboratory Technical Memo 33–669, February 1974.

[4] J. LaSalle and S. Lefschetz, *Stability by Liapunov's Direct Method with Applications*, Academic Press, New York, 1961.

[5] P.K. Khosla, "Some Experimental Results on Model-Based Control Schemes," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[6] M. Leahy, K. Valavanis, and G. Saridis, "The Effects of Dynamic Models on Robot Control," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[7] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd Edition, Springer-Verlag, London, 2000.

[8] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 1, 1987.

[9] C. An, C. Atkeson, and J. Hollerbach, "Model-Based Control of a Direct Drive Arm, Part II: Control," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[10] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*, 2nd edition, Addison-Wesley, Reading, MA, 1989.

[11] A. Liegeois, A. Fournier, and M. Aldon, "Model Reference Control of High Velocity Industrial Robots," *Proceedings of the Joint Automatic Control Conference*, San Francisco, 1980.

[12] M. Raibert, "Mechanical Arm Control Using a State Space Memory," SME paper MS77-750, 1977.

[13] B. Armstrong, "Friction: Experimental Determination, Modeling and Compensation," IEEE Conference on Robotics and Automation, Philadelphia, April 1988.

[14] B. Armstrong, O. Khatib, and J. Burdick, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[15] A.M. Lyapunov, "On the General Problem of Stability of Motion," (in Russian), Kharkov Mathematical Society, Soviet Union, 1892.

[16] C. Desoer and M. Vidyasagar, *Feedback Systems: Input–Output Properties*, Academic Press, New York, 1975.

[17] M. Vidyasagar, *Nonlinear Systems Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[18] S. Arimoto and F. Miyazaki, "Stability and Robustness of PID Feedback Control for Robot Manipulators of Sensory Capability," Third International Symposium of Robotics Research, Gouvieux, France, July 1985.

[19] D. Koditschek, "Adaptive Strategies for the Control of Natural Motion," *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, December 1985.

[20] J. Craig, P. Hsu, and S. Sastry, "Adaptive Control of Mechanical Manipulators," IEEE Conference on Robotics and Automation, San Francisco, April 1986.

[21] J. Craig, *Adaptive Control of Mechanical Manipulators*, Addison-Wesley, Reading, MA, 1988.

[22] J.J. Slotine and W. Li, "On the Adaptive Control of Mechanical Manipulators," *The International Journal of Robotics Research*, Vol. 6, No. 3, 1987.

[23] R. Kelly and R. Ortega, "Adaptive Control of Robot Manipulators: An Input–Output Approach," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

[24] H. Das, J.J. Slotine, and T. Sheridan, "Inverse Kinematic Algorithms for Redundant Systems," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

[25] T. Yabuta, A. Chona, and G. Beni, "On the Asymptotic Stability of the Hybrid Position/Force Control Scheme for Robot Manipulators," IEEE Conference on Robotics and Automation, Philadelphia, 1988.

EXERCISES

10.1 [15] Give the nonlinear control equations for an α,β-partitioned controller for the system

$$\tau = (2\sqrt{\theta} + 1)\ddot{\theta} + 3\dot{\theta}^2 - \sin(\theta).$$

Choose gains so that this system is always critically damped with $k_{CL} = 10$.

10.2 [15] Give the nonlinear control equations for an α,β-partitioned controller for the system

$$\tau = 5\theta\dot{\theta} + 2\ddot{\theta} - 13\dot{\theta}^3 + 5.$$

Choose gains so that this system is always critically damped with $k_{CL} = 10$.

10.3 [19] Draw a block diagram showing a joint-space controller for the two-link arm from Section 6.7, such that the arm is critically damped over its entire workspace. Show the equations inside the blocks of a block diagram.

10.4 [20] Draw a block diagram showing a Cartesian-space controller for the two-link arm from Section 6.7, such that the arm is critically damped over its entire workspace. (See Example 6.6.) Show the equations inside the blocks of a block diagram.

10.5 [18] Design a trajectory-following control system for the system whose dynamics are given by

$$\tau_1 = m_1 l_1^2 \ddot{\theta}_1 + m_1 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2,$$

$$\tau_2 = m_2 l_2^2 (\ddot{\theta}_1 + \ddot{\theta}_2) + v_2 \dot{\theta}_2.$$

Do you think these equations could represent a real system?

10.6 [17] For the control system designed for the one-link manipulator in Example 10.3, give an expression for the steady-state position error as a function of error in the mass parameter. Let $\psi_m = m - \hat{m}$. The result should be a function of $l, g, \theta, \psi_m, \hat{m}$, and k_p. For what position of the manipulator is this at a maximum?

10.7 [26] For the two-degree-of-freedom mechanical system of Fig. 10.17, design a controller that can cause x_1 and x_2 to follow trajectories and suppress disturbances in a critically damped fashion.

10.8 [30] Consider the dynamic equations of the two-link manipulator from Section 6.7 in configuration-space form. Derive expressions for the sensitivity of the computed

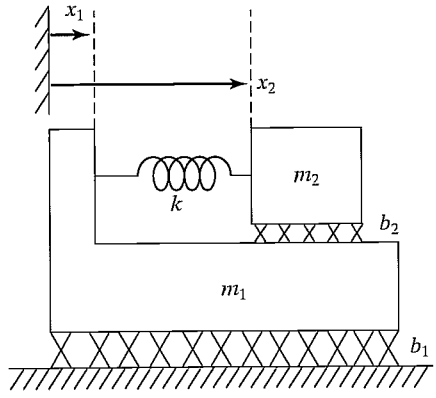

FIGURE 10.17: Mechanical system with two degrees of freedom.

torque value versus small deviations in Θ. Can you say something about how often the dynamics should be recomputed in a controller like that of Fig. 10.7 as a function of average joint velocities expected during normal operations?

10.9 [32] Consider the dynamic equations of the two-link manipulator from Example 6.6 in Cartesian configuration-space form. Derive expressions for the sensitivity of the computed torque value versus small deviations in Θ. Can you say something about how often the dynamics should be recomputed in a controller like that of Fig. 10.15 as a function of average joint velocities expected during normal operations?

10.10 [15] Design a control system for the system

$$f = 5x\dot{x} + 2\ddot{x} - 12.$$

Choose gains so that this system is always critically damped with a closed-loop stiffness of 20.

10.11 [20] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - M(\Theta)K_v\dot{\Theta} + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. *Hint*: This is similar to example 10.6.

10.12 [20] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - \hat{M}(\Theta)K_v\dot{\Theta} + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. The matrix $\hat{M}(\Theta)$ is a positive definite estimate of the manipulator mass matrix. *Hint*: This is similar to example 10.6.

10.13 [25] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -M(\Theta)[K_p\Theta + K_v\dot{\Theta}] + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$ where k_v is a scalar and I_n is the $n \times n$ identity matrix. *Hint*: This is similar to example 10.6.

10.14 [25] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -\hat{M}(\Theta)[K_p\Theta + K_v\dot{\Theta}] + G(\Theta)$$

yields an asymptotically stable nonlinear system. You may take K_v to be of the form $K_v = k_v I_n$, where k_v is a scalar and I_n is the $n \times n$ identity matrix. The matrix $\hat{M}(\Theta)$ is a positive definite estimate of the manipulator mass matrix. *Hint*: This is similar to example 10.6.

10.15 [28] Consider a position-regulation system that (without loss of generality) attempts to maintain $\Theta_d = 0$. Prove that the control law

$$\tau = -K_p\Theta - K_v\dot{\Theta}$$

yields a stable nonlinear system. Show that stability is not asymptotic and give an expression for the steady-state error. *Hint*: This is similar to Example 10.6.

10.16 [30] Prove the global stability of the Jacobian-transpose Cartesian controller introduced in Section 10.8. Use an appropriate form of velocity feedback to stabilize the system. *Hint*: See [18].

10.17 [15] Design a trajectory-following controller for a system with dynamics given by

$$f = ax^2\dot{x}\ddot{x} + b\dot{x}^2 + c\sin(x),$$

such that errors are suppressed in a critically damped fashion over all configurations.

10.18 [15] A system with open-loop dynamics given by

$$\tau = m\ddot{\theta} + b\dot{\theta}^2 + c\dot{\theta}$$

is controlled with the control law

$$\tau = m[\ddot{\theta}_d + k_v\dot{e} + k_p e] + \sin(\theta).$$

Give the differential equation that characterizes the closed-loop action of the system.

PROGRAMMING EXERCISE (PART 10)

Repeat Programming Exercise Part 9, and use the same tests, but with a new controller that uses a complete dynamic model of the 3-link to decouple and linearize the system. For this case, use

$$K_p = \begin{bmatrix} 100.0 & 0.0 & 0.0 \\ 0.0 & 100.0 & 0.0 \\ 0.0 & 0.0 & 100.0 \end{bmatrix}.$$

Choose a diagonal K_v that guarantees critical damping over all configurations of the arm. Compare the results with those obtained with the simpler controller used in Programming Exercise Part 9.