In some sense, the linearizing control law implements an *inverse model* of the system being controlled. The nonlinearities in the system cancel with 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 of partitioning the control into a model-based portion and a servo portion is still applicable, but 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 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:

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

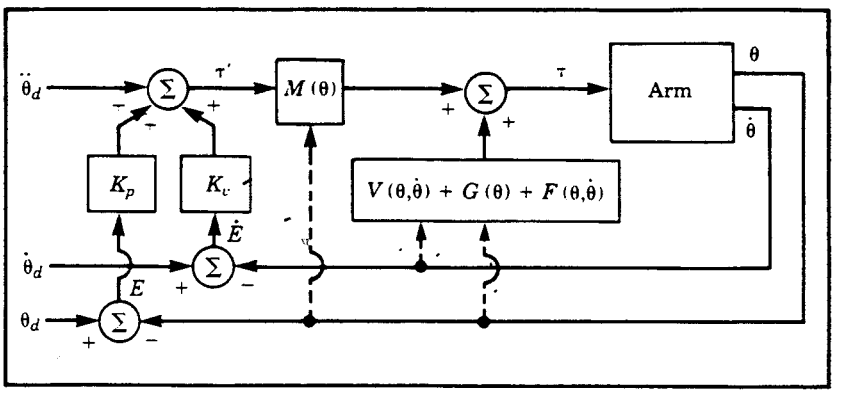

FIGURE 10.5 A model-based manipulator control system.

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 which 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 may incorporate a model of friction (or other nonrigid-body effects). Assuming that our model of friction is a function of joint positions and velocities, we add a 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

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

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 since 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 due to many reasons, the most important two being:

1. 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 which are rarely 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 may 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 and consequently, as 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 which 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 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. Based on the value 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. This is 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 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 which just consists 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* we will find that the error equation of this system is

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

Clearly, as 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 to consider designing a **robust controller**. That is, to find 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, and 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

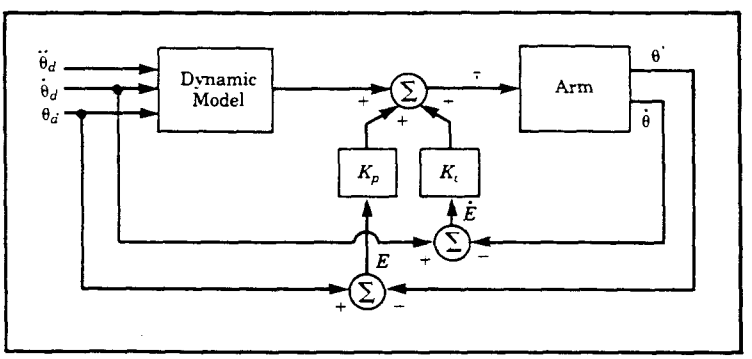

FIGURE 10.6 Control scheme with the model-based portion "outside" the servo loop.

* 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 $F(\Theta_d, \dot{\Theta}_d) \cong F(\Theta, \dot{\Theta})$.

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 appear as functions of manipulator position only. These functions might then be computed by a *background* process or by a second control computer [8], or 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 whereas the closed loop servo could run 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 which are not repeatable, because, for example, they change

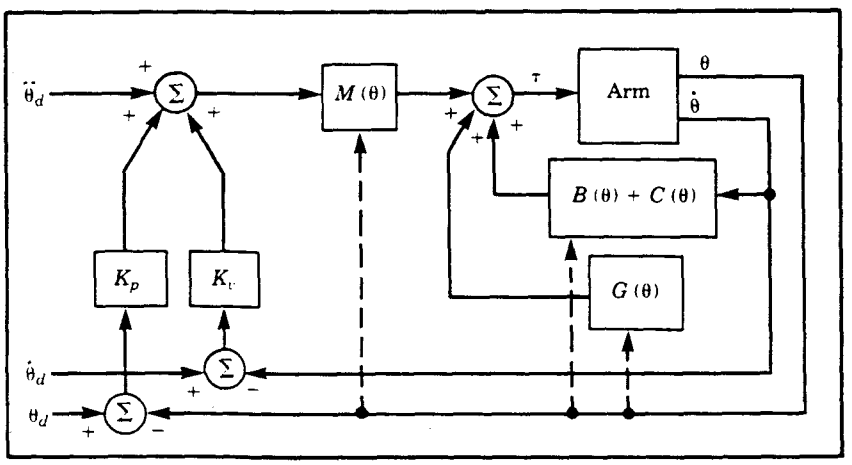

FIGURE 10.7 An implementation of the model-based manipulator control system.

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 may 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 which 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 that 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

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

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, since $M(\Theta)$ is not diagonal in general.

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

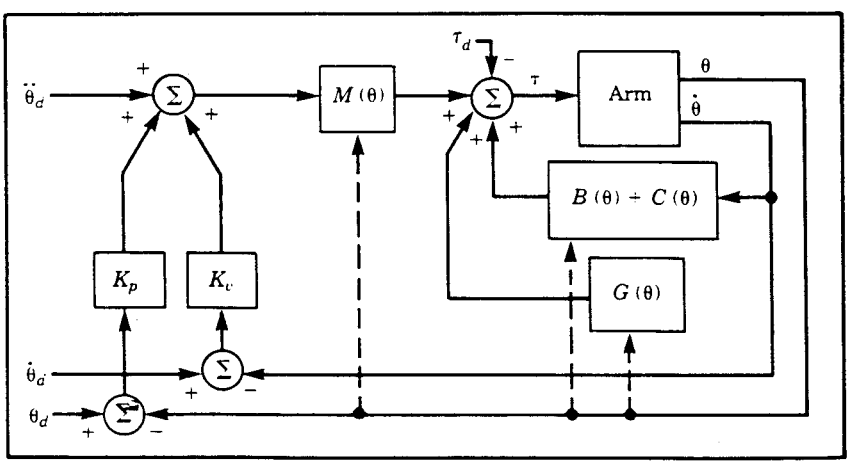

FIGURE 10.8 The model-based controller with an external disturbance acting.

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

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

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

$$\begin{aligned} &\ddot{E} + K_v \dot{E} + K_p E \\ &\quad = \hat{M}^{-1} \left[\left(M - \hat{M} \right) \ddot{\Theta} + \left(V - \hat{V} \right) + \left(G - \hat{G} \right) + \left(F - \hat{F} \right) \right], \end{aligned} \tag{10.25}$$

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 Present 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 may not be worthwhile, especially when lack of knowledge of parameters may 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 which are generally 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.

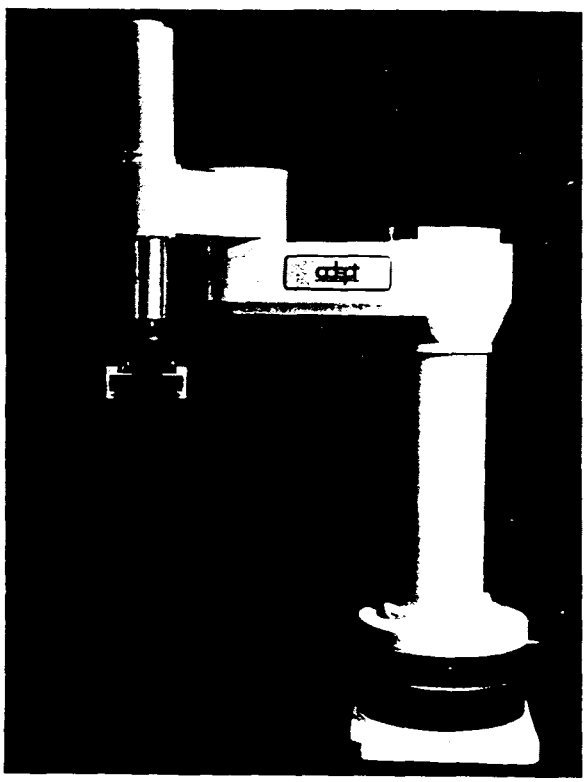

FIGURE 10.9 The Adept One, a direct drive robot by Adept Technology, Inc.

Individual joint PID control

Most present industrial robots 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

$$\tau' = \ddot{\Theta}_d - K_v \dot{E} + K_p E + K_i \int E dt. \tag{10.27}$$

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 discussed in Section 9.10.

The performance of a manipulator controlled in this way is not simple to describe. Since no decoupling is being done, 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 which 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 may be fairly small, and control is good. In such systems, it is important to keep the gains as high as possible so that these inevitable disturbances will be quickly suppressed.

Addition of gravity compensation

Since the gravity terms will tend to cause static positioning errors, 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. Since (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 since 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 which 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 determine 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 which 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 which is applicable to both linear and nonlinear systems.

Consider the simple mass-spring friction system originally considered 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).