CHAPTER 9

Linear control of manipulators

9.1 INTRODUCTION

Armed with the previous material, we now have the means to calculate joint-position time histories that correspond to desired end-effector motions through space. In this chapter, we begin to discuss how to cause the manipulator actually to perform these desired motions.

The control methods that we will discuss fall into the class called **linear-control** systems. Strictly speaking, the use of linear-control techniques is valid only when the system being studied can be modeled mathematically by *linear* differential equations. For the case of manipulator control, such linear methods must essentially be viewed as approximate methods, for, as we have seen in Chapter 6, the dynamics of a manipulator are more properly represented by a *nonlinear* differential equation. Nonetheless, we will see that it is often reasonable to make such approximations, and it also is the case that these linear methods are the ones most often used in current industrial practice.

Finally, consideration of the linear approach will serve as a basis for the more complex treatment of nonlinear control systems in Chapter 10. Although we approach linear control as an approximate method for manipulator control, the justification for using linear controllers is not only empirical. In Chapter 10, we will prove that a certain linear controller leads to a reasonable control system even without resorting to a linear approximation of manipulator dynamics. Readers familiar with linear-control systems might wish to skip the first four sections of the current chapter.

9.2 FEEDBACK AND CLOSED-LOOP CONTROL

We will model a manipulator as a mechanism that is instrumented with sensors at each joint to measure the joint angle and that has an actuator at each joint to apply a torque on the neighboring (next higher) link.[1]. Although other physical arrangements of sensors are sometimes used, the vast majority of robots have a position sensor at each joint. Sometimes velocity sensors (tachometers) are also present at the joints. Various actuation and transmission schemes are prevalent in industrial robots, but many of these can be modeled by supposing that there is a single actuator at each joint.

We wish to cause the manipulator joints to follow prescribed position trajectories, but the actuators are commanded in terms of torque, so we must use some kind of **control system** to compute appropriate actuator commands that will realize this desired motion. Almost always, these torques are determined by using **feedback** from the joint sensors to compute the torque required.

Figure 9.1 shows the relationship between the trajectory generator and the physical robot. The robot accepts a vector of joint torques, τ, from the control system. The manipulator's sensors allow the controller to read the vectors of joint positions, Θ, and joint velocities, $\dot{\Theta}$. All signal lines in Fig. 9.1 carry $N \times 1$ vectors (where N is the number of joints in the manipulator).

Let's consider what algorithm might be implemented in the block labeled "control system" in Fig. 9.1. One possibility is to use the dynamic equation of the robot (as studied in Chapter 6) to calculate the torques required for a particular trajectory. We are given Θ_d, $\dot{\Theta}_d$, and $\ddot{\Theta}_d$ by the trajectory generator, so we could use (6.59) to compute

$$\tau = M(\Theta_d)\ddot{\Theta}_d + V(\Theta_d, \dot{\Theta}_d) + G(\Theta_d). \tag{9.1}$$

This computes the torques that our model dictates would be required to realize the desired trajectory. If our dynamic model were complete and accurate and no "noise" or other disturbances were present, continuous use of (9.1) along the desired trajectory would realize the desired trajectory. Unfortunately, imperfection in the dynamic model and the inevitable presence of disturbances make such a scheme impractical for use in real applications. Such a control technique is termed an **open-loop** scheme, because there is no use made of the feedback from the joint sensors

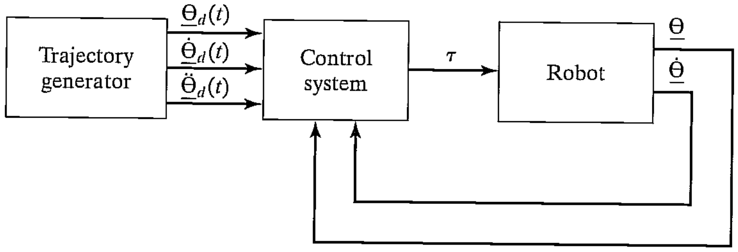

Trajectory generator
$\underline{\Theta}_d(t)$
$\underline{\dot{\Theta}}_d(t)$
$\underline{\ddot{\Theta}}_d(t)$
Control system
τ
Robot
$\underline{\Theta}$
$\underline{\dot{\Theta}}$

FIGURE 9.1: High-level block diagram of a robot-control system.

[1]Remember, all remarks made concerning rotational joints hold analogously for linear joints, and vice versa

(i.e., (9.1) is a function only of the desired trajectory, Θ_d, and its derivatives, and *not* a function of Θ, the actual trajectory).

Generally, the only way to build a high-performance control system is to make use of feedback from joint sensors, as indicated in Fig. 9.1. Typically, this feedback is used to compute any **servo error** by finding the difference between the desired and the actual position and that between the desired and the actual velocity:

$$\begin{aligned} E &= \Theta_d - \Theta, \\ \dot{E} &= \dot{\Theta}_d - \dot{\Theta}. \end{aligned} \tag{9.2}$$

The control system can then compute how much torque to require of the actuators as some function of the servo error. Obviously, the basic idea is to compute actuator torques that would tend to reduce servo errors. A control system that makes use of feedback is called a **closed-loop** system. The "loop" closed by such a control system around the manipulator is apparent in Fig. 9.1.

The central problem in designing a control system is to ensure that the resulting closed-loop system meets certain performance specifications. The most basic such criterion is that the system remain **stable**. For our purposes, we will define a system to be stable if the errors remain "small" when executing various desired trajectories even in the presence of some "moderate" disturbances. It should be noted that an improperly designed control system can sometimes result in **unstable** performance, in which servo errors are enlarged instead of reduced. Hence, the first task of a control engineer is to prove that his or her design yields a stable system; the second is to prove that the closed-loop performance of the system is satisfactory. In practice, such "proofs" range from mathematical proofs based on certain assumptions and models to more empirical results, such as those obtained through simulation or experimentation.

Figure 9.1, in which all signals lines represent $N \times 1$ vectors, summarizes the fact that the manipulator-control problem is a **multi-input, multi-output (MIMO)** control problem. In this chapter, we take a simple approach to constructing a control system by treating each joint as a separate system to be controlled. Hence, for an N-jointed manipulator, we will design N independent **single-input, single-output (SISO)** control systems. This is the design approach presently adopted by most industrial-robot suppliers. This **independent joint control** approach is an approximate method in that the equations of motion (developed in Chapter 6) are not independent, but rather are highly coupled. Later, this chapter will present justification for the linear approach, at least for the case of highly geared manipulators.

9.3 SECOND-ORDER LINEAR SYSTEMS

Before considering the manipulator control problem, let's step back and start by considering a simple mechanical system. Figure 9.2 shows a block of mass m attached to a spring of stiffness k and subject to friction of coefficient b. Figure 9.2 also indicates the zero position and positive sense of x, the block's position. Assuming a frictional force proportional to the block's velocity, a free-body diagram of the forces acting on the block leads directly to the equation of motion,

$$m\ddot{x} + b\dot{x} + kx = 0. \tag{9.3}$$

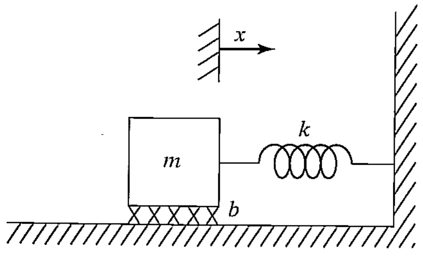


FIGURE 9.2: Spring–mass system with friction.

Hence, the open-loop dynamics of this one-degree-of-freedom system are described by a second-order linear constant-coefficient differential equation [1]. The solution to the differential equation (9.3) is a time function, $x(t)$, that specifies the motion of the block. This solution will depend on the block's **initial conditions**—that is, its initial position and velocity.

We will use this simple mechanical system as an example with which to review some basic control system concepts. Unfortunately, it is impossible to do justice to the field of control theory with only a brief introduction here. We will discuss the control problem, assuming no more than that the student is familiar with simple differential equations. Hence, we will not use many of the popular tools of the control-engineering trade. For example, **Laplace transforms** and other common techniques neither are a prerequisite nor are introduced here. A good reference for the field is [4].

Intuition suggests that the system of Fig. 9.2 might exhibit several different characteristic motions. For example, in the case of a very weak spring (i.e., k small) and very heavy friction (i.e., b large) one imagines that, if the block were perturbed, it would return to its resting position in a very slow, sluggish manner. However, with a very stiff spring and very low friction, the block might oscillate several times before coming to rest. These different possibilities arise because the character of the solution to (9.3) depends upon the values of the parameters m, b, and k.

From the study of differential equations [1], we know that the form of the solution to an equation of the form of (9.3) depends on the roots of its **characteristic equation**,

$$ms^2 + bs + k = 0. \tag{9.4}$$

This equation has the roots

$$\begin{aligned} s_1 &= -\frac{b}{2m} + \frac{\sqrt{b^2 - 4mk}}{2m}, \\ s_2 &= -\frac{b}{2m} - \frac{\sqrt{b^2 - 4mk}}{2m}. \end{aligned} \tag{9.5}$$

The location of s_1 and s_2 (sometimes called the **poles** of the system) in the real–imaginary plane dictate the nature of the motions of the system. If s_1 and s_2 are real, then the behavior of the system is sluggish and nonoscillatory. If s_1 and s_2 are complex (i.e., have an imaginary component) then the behavior of the system is oscillatory. If we include the special limiting case between these two behaviors, we have three classes of response to study:

1. **Real and Unequal Roots**. This is the case when $b^2 > 4\,mk$; that is, friction dominates, and sluggish behavior results. This response is called **overdamped**.
2. **Complex Roots**. This is the case when $b^2 < 4\,mk$; that is, stiffness dominates, and oscillatory behavior results. This response is called **underdamped**.
3. **Real and Equal Roots**. This is the special case when $b^2 = 4\,mk$; that is, friction and stiffness are "balanced," yielding the fastest possible nonoscillatory response. This response is called **critically damped**.

The third case (critical damping) is generally a desirable situation: the system nulls out nonzero initial conditions and returns to its nominal position as rapidly as possible, yet without oscillatory behavior.

Real and unequal roots

It can easily be shown (by direct substitution into (9.3)) that the solution, $x(t)$, giving the motion of the block in the case of real, unequal roots has the form

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}, \tag{9.6}$$

where s_1 and s_2 are given by (9.5). The coefficients c_1 and c_2 are constants that can be computed for any given set of initial conditions (i.e., initial position and velocity of the block).

Figure 9.3 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are real and unequal, the system exhibits sluggish or overdamped motion.

In cases where one of the poles has a much greater magnitude than the other, the pole of larger magnitude can be neglected, because the term corresponding to it will decay to zero rapidly in comparison to the other, **dominant pole**. This same notion of dominance extends to higher order systems—for example, often a

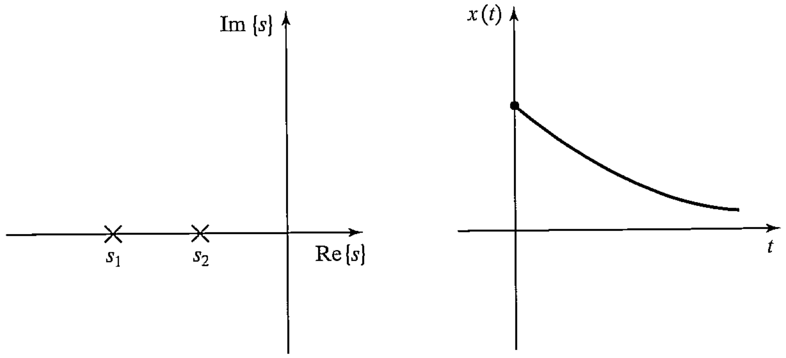


FIGURE 9.3: Root location and response to initial conditions for an overdamped system.

third-order system can be studied as a second-order system by considering only two dominant poles.

EXAMPLE 9.1

Determine the motion of the system in Fig. 9.2 if parameter values are $m = 1, b = 5$, and $k = 6$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + 5s + 6 = 0, \tag{9.7}$$

which has the roots $s_1 = -2$ and $s_2 = -3$. Hence, the response has the form

$$x(t) = c_1 e^{-2t} + c_2 e^{-3t}. \tag{9.8}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to compute c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 + c_2 = -1$$

and

$$-2c_1 - 3c_2 = 0, \tag{9.9}$$

which are satisfied by $c_1 = -3$ and $c_2 = 2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = -3e^{-2t} + 2e^{-3t}. \tag{9.10}$$

Complex roots

For the case where the characteristic equation has complex roots of the form

$$\begin{aligned} s_1 &= \lambda + \mu i, \\ s_2 &= \lambda - \mu i, \end{aligned} \tag{9.11}$$

it is still the case that the solution has the form

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t}. \tag{9.12}$$

However, equation (9.12) is difficult to use directly, because it involves imaginary numbers explicitly. It can be shown (see Exercise 9.1) that **Euler's formula**,

$$e^{ix} = \cos x + i \sin x, \tag{9.13}$$

allows the solution (9.12) to be manipulated into the form

$$x(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\lambda t} \sin(\mu t). \tag{9.14}$$

As before, the coefficients c_1 and c_2 are constants that can be computed for any given set of initial conditions (i.e., initial position and velocity of the block). If we write the constants c_1 and c_2 in the form

$$\begin{aligned} c_1 &= r \cos \delta, \\ c_2 &= r \sin \delta, \end{aligned} \tag{9.15}$$

then (9.14) can be written in the form

$$x(t) = re^{\lambda t}\cos(\mu t - \delta), \tag{9.16}$$

where

$$r = \sqrt{c_1^2 + c_2^2},$$
$$\delta = \text{Atan2}(c_2, c_1). \tag{9.17}$$

In this form, it is easier to see that the resulting motion is an oscillation whose amplitude is exponentially decreasing toward zero.

Another common way of describing oscillatory second-order systems is in terms of **damping ratio** and **natural frequency**. These terms are defined by the parameterization of the characteristic equation given by

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0, \tag{9.18}$$

where ζ is the damping ratio (a dimensionless number between 0 and 1) and ω_n is the natural frequency.[2] Relationships between the pole locations and these parameters are

$$\lambda = -\zeta\omega_n$$

and

$$\mu = \omega_n\sqrt{1 - \zeta^2}. \tag{9.19}$$

In this terminology, μ, the imaginary part of the poles, is sometimes called the **damped natural frequency**. For a damped spring–mass system such as the one in Fig. 9.2, the damping ratio and natural frequency are, respectively,

$$\zeta = \frac{b}{2\sqrt{km}},$$
$$\omega_n = \sqrt{k/m}. \tag{9.20}$$

When no damping is present ($b = 0$ in our example), the damping ratio becomes zero; for critical damping ($b^2 = 4km$), the damping ratio is 1.

Figure 9.4 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are complex, the system exhibits oscillatory or underdamped motion.

EXAMPLE 9.2

Find the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 1$, and $k = 1$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + s + 1 = 0, \tag{9.21}$$

[2]The terms *damping ratio* and *natural frequency* can also be applied to overdamped systems, in which case $\zeta > 1.0$.

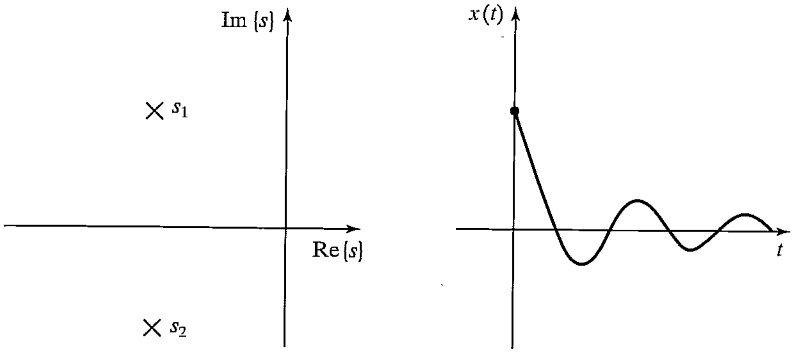


FIGURE 9.4: Root location and response to initial conditions for an underdamped system.

which has the roots $s_i = -\frac{1}{2} \pm \frac{\sqrt{3}}{2}i$. Hence, the response has the form

$$x(t) = e^{-\frac{t}{2}}\left(c_1 \cos\frac{\sqrt{3}}{2}t + c_2 \sin\frac{\sqrt{3}}{2}t\right). \tag{9.22}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to compute c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 = -1$$

and

$$-\frac{1}{2}c_1 - \frac{\sqrt{3}}{2}c_2 = 0, \tag{9.23}$$

which are satisfied by $c_1 = -1$ and $c_2 = \frac{\sqrt{3}}{3}$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = e^{-\frac{t}{2}}\left(-\cos\frac{\sqrt{3}}{2}t - \frac{\sqrt{3}}{3}\sin\frac{\sqrt{3}}{2}t\right). \tag{9.24}$$

This result can also be put in the form of (9.16), as

$$x(t) = \frac{2\sqrt{3}}{3}e^{-\frac{t}{2}}\cos\left(\frac{\sqrt{3}}{2}t + 120^\circ\right). \tag{9.25}$$

Real and equal roots

By substitution into (9.3), it can be shown that, in the case of real and equal roots (i.e., **repeated roots**), the solution has the form

$$x(t) = c_1 e^{s_1 t} + c_2 t e^{s_2 t}, \tag{9.26}$$

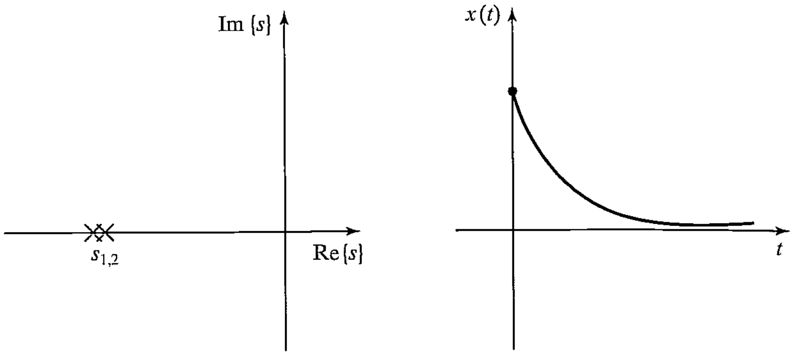


FIGURE 9.5: Root location and response to initial conditions for a critically damped system.

where, in this case, $s_1 = s_2 = -\frac{b}{2m}$, so (9.26) can be written

$$x(t) = (c_1 + c_2 t)e^{-\frac{b}{2m}t}. \tag{9.27}$$

In case it is not clear, a quick application of **l'Hôpital's rule** [2] shows that, for any c_1, c_2, and a,

$$\lim_{t\to\infty} (c_1 + c_2 t)e^{-at} = 0. \tag{9.28}$$

Figure 9.5 shows an example of pole locations and the corresponding time response to a nonzero initial condition. When the poles of a second-order system are real and equal, the system exhibits critically damped motion, the fastest possible nonoscillatory response.

EXAMPLE 9.3

Work out the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 4$, and $k = 4$ and the block (initially at rest) is released from the position $x = -1$.

The characteristic equation is

$$s^2 + 4s + 4 = 0, \tag{9.29}$$

which has the roots $s_1 = s_2 = -2$. Hence, the response has the form

$$x(t) = (c_1 + c_2 t)e^{-2t}. \tag{9.30}$$

We now use the given initial conditions, $x(0) = -1$ and $\dot{x}(0) = 0$, to calculate c_1 and c_2. To satisfy these conditions at $t = 0$, we must have

$$c_1 = -1$$

and

$$-2c_1 + c_2 = 0, \tag{9.31}$$

which are satisfied by $c_1 = -1$ and $c_2 = -2$. So, the motion of the system for $t \geq 0$ is given by

$$x(t) = (-1 - 2t)e^{-2t}. \tag{9.32}$$

In Examples 9.1 through 9.3, all the systems were stable. For any passive physical system like that of Fig. 9.2, this will be the case. Such mechanical systems always have the properties

$$\begin{aligned} m &> 0, \\ b &> 0, \\ k &> 0. \end{aligned} \tag{9.33}$$

In the next section, we will see that the action of a control system is, in effect, to change the value of one or more of these coefficients. It will then be necessary to consider whether the resulting system is stable.

9.4 CONTROL OF SECOND-ORDER SYSTEMS

Suppose that the natural response of our second-order mechanical system is not what we wish it to be. Perhaps it is underdamped and oscillatory, and we would like it to be critically damped; or perhaps the spring is missing altogether ($k = 0$), so the system never returns to $x = 0$ if disturbed. Through the use of sensors, an actuator, and a control system, we can modify the system's behavior as desired.

Figure 9.6 shows a damped spring–mass system with the addition of an actuator with which it is possible to apply a force f to the block. A free-body diagram leads to the equation of motion,

$$m\ddot{x} + b\dot{x} + kx = f. \tag{9.34}$$

Let's also assume that we have sensors capable of detecting the block's position and velocity. We now propose a **control law** which computes the force that should be applied by the actuator as a function of the sensed feedback:

$$f = -k_p x - k_v \dot{x}. \tag{9.35}$$

Figure 9.7 is a block diagram of the closed-loop system, where the portion to the left of the dashed line is the control system (usually implemented in a computer) and that to the right of the dashed line is the physical system. Implicit in the figure are interfaces between the control computer and the output actuator commands and the input sensor information.

The control system we have proposed is a **position-regulation** system—it simply attempts to maintain the position of the block in one fixed place regardless

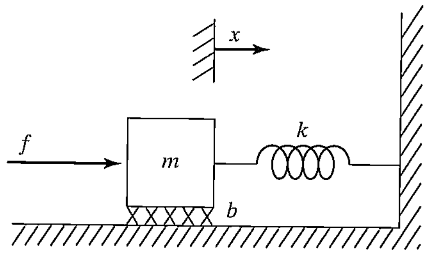


FIGURE 9.6: A damped spring–mass system with an actuator.

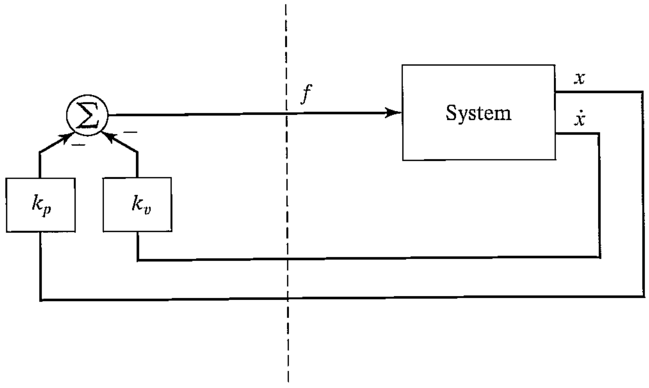


FIGURE 9.7: A closed-loop control system. The control computer (to the left of the dashed line) reads sensor input and writes actuator output commands.

of disturbance forces applied to the block. In a later section, we will construct a **trajectory-following** control system, which can cause the block to follow a desired position trajectory.

By equating the open-loop dynamics of (9.34) with the control law of (9.35), we can derive the closed-loop dynamics as

$$m\ddot{x} + b\dot{x} + kx = -k_p x - k_v \dot{x}, \tag{9.36}$$

or

$$m\ddot{x} + (b + k_v)\dot{x} + (k + k_p)x = 0, \tag{9.37}$$

or

$$m\ddot{x} + b'\dot{x} + k'x = 0, \tag{9.38}$$

where $b' = b + k_v$ and $k' = k + k_p$. From (9.37) and (9.38), it is clear that, by setting the **control gains**, k_v and k_p, we can cause the closed-loop system to appear to have any second system behavior that we wish. Often, gains would be chosen to obtain critical damping (i.e., $b' = 2\sqrt{mk'}$) and some desired **closed-loop stiffness** given directly by k'.

Note that k_v and k_p could be positive or negative, depending on the parameters of the original system. However, if b' or k' became negative, the result would be an unstable control system. This instability will be obvious if one writes down the solution of the second-order differential equation (in the form of (9.6), (9.14), or (9.26)). It also makes intuitive sense that, if b' or k' is negative, servo errors tend to get magnified rather than reduced.

EXAMPLE 9.4

If the parameters of the system in Fig. 9.6 are $m = 1$, $b = 1$, and $k = 1$, find gains k_p and k_v for a position-regulation control law that results in the system's being critically damped with a closed-loop stiffness of 16.0.

If we wish k' to be 16.0, then, for critical damping, we require that $b' = 2\sqrt{mk'} = 8.0$. Now, $k = 1$ and $b = 1$, so we need

$$\begin{aligned} k_p &= 15.0, \\ k_v &= 7.0. \end{aligned} \tag{9.39}$$

9.5 CONTROL-LAW PARTITIONING

In preparation for designing control laws for more complicated systems, let us consider a slightly different controller structure for the sample problem of Fig. 9.6. In this method, we will partition the controller into a **model-based portion** and a **servo portion**. The result is that the system's parameters (i.e., m, b, and k, in this case) appear only in the model-based portion and that the servo portion is independent of these parameters. At the moment, this distinction might not seem important, but it will become more obviously important as we consider nonlinear systems in Chapter 10. We will adopt this **control-law partitioning** approach throughout the book.

The open-loop equation of motion for the system is

$$m\ddot{x} + b\dot{x} + kx = f. \tag{9.40}$$

We wish to decompose the controller for this system into two parts. In this case, the model-based portion of the control law will make use of supposed knowledge of m, b, and k. This portion of the control law is set up such that it *reduces the system so that it appears to be a unit mass*. This will become clear when we do Example 9.5. The second part of the control law makes use of feedback to modify the behavior of the system. The model-based portion of the control law has the effect of making the system appear as a unit mass, so the design of the servo portion is very simple—gains are chosen to control a system composed of a single unit mass (i.e., no friction, no stiffness).

The model-based portion of the control appears in a control law of the form

$$f = \alpha f' + \beta, \tag{9.41}$$

where α and β are functions or constants and are chosen so that, if f' is taken as the *new input* to the system, *the system appears to be a unit mass*. With this structure of the control law, the system equation (the result of combining (9.40) and (9.41)) is

$$m\ddot{x} + b\dot{x} + kx = \alpha f' + \beta. \tag{9.42}$$

Clearly, in order to make the system appear as a unit mass from the f' input, for this particular system we should choose α and β as follows:

$$\begin{aligned} \alpha &= m, \\ \beta &= b\dot{x} + kx. \end{aligned} \tag{9.43}$$

Making these assignments and plugging them into (9.42), we have the system equation

$$\ddot{x} = f'. \tag{9.44}$$

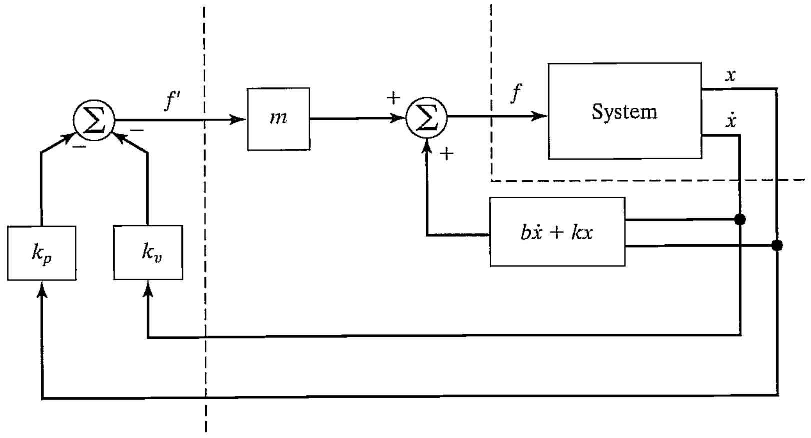


FIGURE 9.8: A closed-loop control system employing the partitioned control method.

This is the equation of motion for a unit mass. We now proceed as if (9.44) were the open-loop dynamics of a system to be controlled. We design a control law to compute f', just as we did before:

$$f' = -k_v\dot{x} - k_p x. \tag{9.45}$$

Combining this control law with (9.44) yields

$$\ddot{x} + k_v\dot{x} + k_p x = 0. \tag{9.46}$$

Under this methodology, the setting of the control gains is simple and is independent of the system parameters; that is,

$$k_v = 2\sqrt{k_p} \tag{9.47}$$

must hold for critical damping. Figure 9.8 shows a block diagram of the partitioned controller used to control the system of Fig. 9.6.

EXAMPLE 9.5

If the parameters of the system in Fig. 9.6 are $m = 1, b = 1$, and $k = 1$, find α, β, and the gains k_p and k_v for a position-regulation control law that results in the system's being critically damped with a closed-loop stiffness of 16.0.

We choose

$$\begin{aligned} \alpha &= 1, \\ \beta &= \dot{x} + x, \end{aligned} \tag{9.48}$$

so that the system appears as a unit mass from the fictitious f' input. We then set gain k_p to the desired closed-loop stiffness and set $k_v = 2\sqrt{k_p}$ for critical damping.

This gives

$$k_p = 16.0,$$
$$k_v = 8.0. \tag{9.49}$$

9.6 TRAJECTORY-FOLLOWING CONTROL

Rather than just maintaining the block at a desired location, let us enhance our controller so that the block can be made to follow a trajectory. The trajectory is given by a function of time, $x_d(t)$, that specifies the desired position of the block. We assume that the trajectory is smooth (i.e., the first two derivatives exist) and that our trajectory generator provides x_d, $\dot{x}_d$, and $\ddot{x}_d$ at all times t. We define the servo error between the desired and actual trajectory as $e = x_d - x$. A servo-control law that will cause trajectory following is

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e. \tag{9.50}$$

We see that (9.50) is a good choice if we combine it with the equation of motion of a unit mass (9.44), which leads to

$$\ddot{x} = \ddot{x}_d + k_v\dot{e} + k_p e, \tag{9.51}$$

or

$$\ddot{e} + k_v\dot{e} + k_p e = 0. \tag{9.52}$$

This is a second-order differential equation for which we can choose the coefficients, so we can design any response we wish. (Often, critical damping is the choice made.) Such an equation is sometimes said to be written in **error space**, because it describes the evolution of errors relative to the desired trajectory. Figure 9.9 shows a block diagram of our trajectory-following controller.

If our model is perfect (i.e., our knowledge of m, b, and k), and if there is no noise and no initial error, the block will follow the desired trajectory exactly. If there is an initial error, it will be suppressed according to (9.52), and thereafter the system will follow the trajectory exactly.

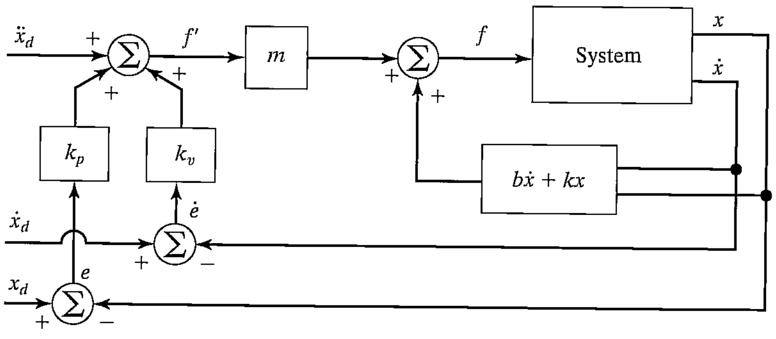


FIGURE 9.9: A trajectory-following controller for the system in Fig. 9.6.

9.7 DISTURBANCE REJECTION

One of the purposes of a control system is to provide **disturbance rejection**, that is, to maintain good performance (i.e., minimize errors) even in the presence of some external disturbances or **noise**. In Fig. 9.10, we show the trajectory-following controller with an additional input: a disturbance force f_{dist}. An analysis of our closed-loop system leads to the error equation

$$\ddot{e} + k_v \dot{e} + k_p e = f_{\text{dist}}. \tag{9.53}$$

Equation (9.53) is that of a differential equation driven by a forcing function. If it is known that f_{dist} is **bounded**—that is, that a constant a exists such that

$$\max_t f_{\text{dist}}(t) < a, \tag{9.54}$$

then the solution of the differential equation, $e(t)$, is also bounded. This result is due to a property of stable linear systems known as **bounded-input, bounded-output** or **BIBO** stability [3, 4]. This very basic result ensures that, for a large class of possible disturbances, we can at least be assured that the system remains stable.

Steady-state error

Let's consider the simplest kind of disturbance—namely, that f_{dist} is a constant. In this case, we can perform a **steady-state analysis** by analyzing the system at rest (i.e., the derivatives of all system variables are zero). Setting derivatives to zero in (9.53) yields the steady-state equation

$$k_p e = f_{\text{dist}}, \tag{9.55}$$

or

$$e = f_{\text{dist}} / k_p. \tag{9.56}$$

The value of e given by (9.56) represents a **steady-state error**. Thus, it is clear that the higher the position gain k_p, the smaller will be the steady-state error.

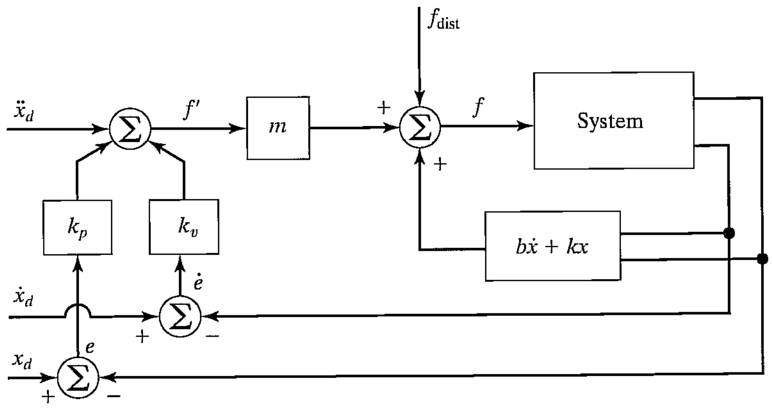


FIGURE 9.10: A trajectory-following control system with a disturbance acting.

Addition of an integral term

In order to eliminate steady-state error, a modified control law is sometimes used. The modification involves the addition of an *integral* term to the control law. The control law becomes

$$f' = \ddot{x}_d + k_v \ddot{e} + k_p e + k_i \int e dt, \tag{9.57}$$

which results in the error equation

$$\ddot{e} + k_v \dot{e} + k_p e + k_i \int e dt = f_{\text{dist}}. \tag{9.58}$$

The term is added so that the system will have no steady-state error in the presence of constant disturbances. If $e(t) = 0$ for $t < 0$, we can write (9.58) for $t > 0$ as

$$\dddot{e} + k_v \ddot{e} + k_p \dot{e} + k_i e = \dot{f}_{\text{dist}}, \tag{9.59}$$

which, in the steady state (for a constant disturbance), becomes

$$k_i e = 0, \tag{9.60}$$

so

$$e = 0. \tag{9.61}$$

With this control law, the system becomes a third-order system, and one can solve the corresponding third-order differential equation to work out the response of the system to initial conditions. Often, k_i is kept quite small so that the third-order system is "close" to the second-order system without this term (i.e., a dominant-pole analysis can be performed). The form of control law (9.57) is called a **PID control law**, or "proportional, integral, derivative" control law [4]. For simplicity, the displayed equations generally do not show an integral term in the control laws that we develop in this book.

9.8 CONTINUOUS VS. DISCRETE TIME CONTROL

In the control systems we have discussed, we implicitly assumed that the control computer performs the computation of the control law in zero time (i.e., infinitely fast), so that the value of the actuator force f is a continuous function of time. Of course, in reality, the computation requires some time, and the resulting commanded force is therefore a discrete "staircase" function. We shall employ this approximation of a very fast control computer throughout the book. This approximation is good if the rate at which new values of f are computed is much faster than the natural frequency of the system being controlled. In the field of **discrete time control** or **digital control**, one does not make this approximation but rather takes the **servo rate** of the control system into account when analyzing the system [3].

We will generally assume that the computations can be performed quickly enough that our continuous time assumption is valid. This raises a question: How quick is quick enough? There are several points that need to be considered in choosing a sufficiently fast servo (or sample) rate:

Tracking reference inputs: The frequency content of the desired or reference input places an absolute lower bound on the sample rate. The sample rate must be at least twice the bandwidth of reference inputs. This is usually not the limiting factor.

Disturbance rejection: In disturbance rejection, an upper bound on performance is given by a continuous-time system. If the sample period is longer than the correlation time of the disturbance effects (assuming a statistical model for random disturbances), then these disturbances will not be suppressed. Perhaps a good rule of thumb is that the sample period should be 10 times shorter than the correlation time of the noise [3].

Antialiasing: Any time an analog sensor is used in a digital control scheme, there will be a problem with aliasing unless the sensor's output is strictly band limited. In most cases, sensors do not have a band limited output, and so sample rate should be chosen such that the amount of energy that appears in the aliased signal is small.

Structural resonances: We have not included bending modes in our characterization of a manipulator's dynamics. All real mechanisms have finite stiffness and so will be subject to various kinds of vibrations. If it is important to suppress these vibrations (and it often is), we must choose a sample rate at least twice the natural frequency of these resonances. We will return to the topic of resonance later in this chapter.

9.9 MODELING AND CONTROL OF A SINGLE JOINT

In this section, we will develop a simplified model of a single rotary joint of a manipulator. A few assumptions will be made that will allow us to model the resulting system as a second-order linear system. For a more complete model of an actuated joint, see [5].

A common actuator found in many industrial robots is the direct current (DC) torque motor (as in Fig. 8.18). The nonturning part of the motor (the **stator**) consists of a housing, bearings, and either permanent magnets or electromagnets. These stator magnets establish a magnetic field across the turning part of the motor (the **rotor**). The rotor consists of a shaft and windings through which current moves to power the motor. The current is conducted to the windings via brushes, which make contact with the commutator. The commutator is wired to the various windings (also called the **armature**) in such a way that torque is always produced in the desired direction. The underlying physical phenomenon [6] that causes a motor to generate a torque when current passes through the windings can be expressed as

$$F = qV \times B, \tag{9.62}$$

where charge q, moving with velocity V through a magnetic field B, experiences a force F. The charges are those of electrons moving through the windings, and the magnetic field is that set up by the stator magnets. Generally, the torque-producing ability of a motor is stated by means of a single **motor torque constant**, which relates armature current to the output torque as

$$\tau_m = k_m i_a. \tag{9.63}$$

When a motor is rotating, it acts as a generator, and a voltage develops across the armature. A second motor constant, the **back emf constant**,[3] describes the voltage generated for a given rotational velocity:

$$v = k_e \dot{\theta}_m. \tag{9.64}$$

Generally, the fact that the commutator is switching the current through various sets of windings causes the torque produced to contain some **torque ripple**. Although sometimes important, this effect can usually be ignored. (In any case, it is quite hard to model—and quite hard to compensate for, even if it is modeled.)

Motor-armature inductance

Figure 9.11 shows the electric circuit of the armature. The major components are a voltage source, v_a, the inductance of the armature windings, l_a, the resistance of the armature windings, r_a, and the generated back emf, v. The circuit is described by a first-order differential equation:

$$l_a \dot{i}_a + r_a i_a = v_a - k_e \dot{\theta}_m. \tag{9.65}$$

It is generally desirable to control the torque generated by the motor (rather than the velocity) with electronic motor driver circuitry. These drive circuits sense the current through the armature and continuously adjust the voltage source v_a so that a desired current i_a flows through the armature. Such a circuit is called a **current amplifier** motor driver [7]. In these current-drive systems, the rate at which the armature current can be commanded to change is limited by the motor inductance l_a and by an upper limit on the voltage capability of the voltage source v_a. The net effect is that of a **low-pass filter** between the requested current and output torque.

Our first simplifying assumption is that the inductance of the motor can be neglected. This is a reasonable assumption when the natural frequency of the closed-loop control system is quite low compared to the cut-off frequency of the implicit low-pass filter in the current-drive circuitry due to the inductance. This assumption, along with the assumption that torque ripple is a negligible effect, means that we can essentially command torque directly. Although there might be a scale factor (such as k_m) to contend with, we will assume that the actuator acts as a pure torque source that we can command directly.

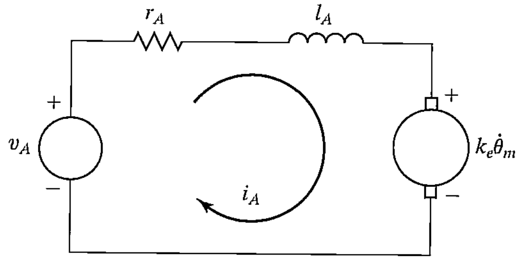


FIGURE 9.11: The armature circuit of a DC torque motor.

[3]"emf" stands for electromotive force.

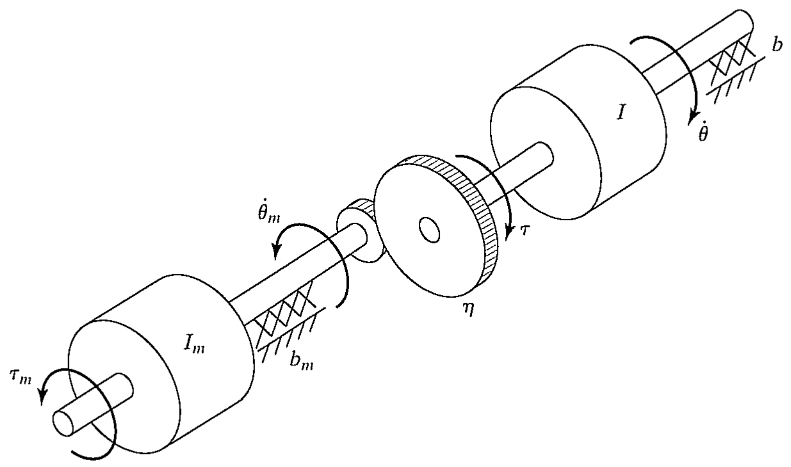


FIGURE 9.12: Mechanical model of a DC torque motor connected through gearing to an inertial load.

Effective inertia

Figure 9.12 shows the mechanical model of the rotor of a DC torque motor connected through a gear reduction to an inertial load. The torque applied to the rotor, τ_m, is given by (9.63) as a function of the current i_a flowing in the armature circuit. The gear ratio (η) causes an increase in the torque seen at the load and a reduction in the speed of the load, given by

$$\tau = \eta\tau_m,$$
$$\dot{\theta} = (1/\eta)\dot{\theta}_m, \tag{9.66}$$

where $\eta > 1$. Writing a torque balance for this system in terms of torque at the rotor yields

$$\tau_m = I_m\ddot{\theta}_m + b_m\dot{\theta}_m + (1/\eta)\left(I\ddot{\theta} + b\dot{\theta}\right), \tag{9.67}$$

where I_m and I are the inertias of the motor rotor and of the load, respectively, and b_m and b are viscous friction coefficients for the rotor and load bearings, respectively. Using the relations (9.66), we can write (9.67) in terms of motor variables as

$$\tau_m = \left(I_m + \frac{I}{\eta^2}\right)\ddot{\theta}_m + \left(b_m + \frac{b}{\eta^2}\right)\dot{\theta}_m \tag{9.68}$$

or in terms of load variables as

$$\tau = (I + \eta^2 I_m)\ddot{\theta} + (b + \eta^2 b_m)\dot{\theta}. \tag{9.69}$$

The term $I + \eta^2 I_m$ is sometimes called the **effective inertia** "seen" at the output (link side) of the gearing. Likewise, the term $b + \eta^2 b_m$ can be called the **effective damping**. Note that, in a highly geared joint (i.e., $\eta \gg 1$), the inertia of the motor rotor can be a significant portion of the combined effective inertia. It is this effect that allows us to make the assumption that the effective inertia is a constant. We know from Chapter 6 that the inertia, I, of a joint of the mechanism actually varies with configuration and load. However, in highly geared robots, the variations represent a smaller percentage than they would in a **direct-drive** manipulator (i.e., $\eta = 1$). To ensure that the motion of the robot link is never underdamped, the value used for I should be the maximum of the range of values that I takes on; we'll call this value $I_{\max}$. This choice results in a system that is critically damped or overdamped in all situations. In Chapter 10, we will deal with varying inertia directly and will not have to make this assumption.

EXAMPLE 9.6

If the apparent link inertia, I, varies between 2 and 6 Kg-m^2, the rotor inertia is $I_m = 0.01$, and the gear ratio is $\eta = 30$, what are the minimum and maximum of the effective inertia?

The minimum effective inertia is

$$I_{\min} + \eta^2 I_m = 2.0 + (900)(0.01) = 11.0; \tag{9.70}$$

the maximum is

$$I_{\max} + \eta^2 I_m = 6.0 + (900)(0.01) = 15.0. \tag{9.71}$$

Hence, we see that, as a percentage of the total effective inertia, the variation of inertia is reduced by the gearing.

Unmodeled flexibility

The other major assumption we have made in our model is that the gearing, the shafts, the bearings, and the driven link are not flexible. In reality, all of these elements have finite stiffness, and their flexibility, if modeled, would increase the order of the system. The argument for ignoring flexibility effects is that, if the system is sufficiently stiff, the natural frequencies of these **unmodeled resonances** are very high and can be neglected compared to the influence of the dominant second-order poles that we have modeled.[4] The term "unmodeled" refers to the fact that, for purposes of control-system analysis and design, we neglect these effects and use a simpler dynamic model, such as (9.69).

Because we have chosen not to model structural flexibilities in the system, we must be careful not to excite these resonances. A rule of thumb [8] is that, if the lowest structural resonance is ω_{res}, then we must limit our closed-loop natural frequency according to

$$\omega_n \leq \frac{1}{2}\omega_{\text{res}}. \tag{9.72}$$

This provides some guidance on how to choose gains in our controller. We have seen that increasing gains leads to faster response and lower steady-state error, but we now see that unmodeled structural resonances limit the magnitude of gains. Typical industrial manipulators have structural resonances in the range from 5 Hz to 25 Hz [8]. Recent designs using direct-drive arrangements that do not contain flexibility

[4]This is basically the same argument we used to neglect the pole due to the motor inductance. Including it would also have raised the order of the overall system.

introduced by reduction and transmission systems have their lowest structural resonances as high as 70 Hz [9].

EXAMPLE 9.7

Consider the system of Fig. 9.7 with the parameter values $m = 1$, $b = 1$, and $k = 1$. Additionally, it is known that the lowest unmodeled resonance of the system is at 8 radians/second. Find α, β, and gains k_p and k_v for a position-control law so the system is critically damped, doesn't excite unmodeled dynamics, and has as high a closed-loop stiffness as possible.

We choose

$$\begin{aligned} \alpha &= 1, \\ \beta &= \dot{x} + x, \end{aligned} \tag{9.73}$$

so that the system appears as a unit mass from the fictitious f' input. Using our rule of thumb (9.72), we choose the closed-loop natural frequency to be $\omega_n = 4$ radians/second. From (9.18) and (9.46), we have $k_p = \omega_n^2$, so

$$\begin{aligned} k_p &= 16.0, \\ k_v &= 8.0. \end{aligned} \tag{9.74}$$

Estimating resonant frequency

The same sources of structural flexibility discussed in Chapter 8 give rise to resonances. In each case where a structural flexibility can be identified, an approximate analysis of the resulting vibration is possible if we can describe the effective mass or inertia of the flexible member. This is done by approximating the situation by a simple spring–mass system, which, as given in (9.20), exhibits the natural frequency

$$\omega_n = \sqrt{k/m}, \tag{9.75}$$

where k is the stiffness of the flexible member and m is the equivalent mass displaced in vibrations.

EXAMPLE 9.8

A shaft (assumed massless) with a stiffness of 400 Nt-m/radian drives a rotational inertia of 1 Kg-m^2. If the shaft stiffness was neglected in the modeling of the dynamics, what is the frequency of this unmodeled resonance?

Using (9.75), we have

$$\omega_{\text{res}} = \sqrt{400/1} = 20 \text{ rad/second} = 20/(2\pi)\text{Hz} \cong 3.2 \text{ Hz}. \tag{9.76}$$

For the purposes of a rough estimate of the lowest resonant frequency of beams and shafts, [10] suggests using a **lumped model** of the mass. We already

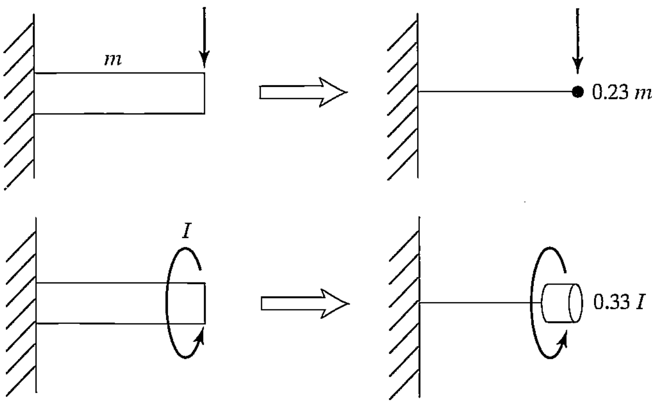


FIGURE 9.13: Lumped models of beams for estimation of lowest lateral and torsional resonance.

have formulas for estimating stiffness at the ends of beams and shafts; these lumped models provide the effective mass or inertia needed for our estimation of resonant frequency. Figure 9.13 shows the results of an energy analysis [10] which suggests that a beam of mass m be replaced by a point mass at the end of 0.23 m and, likewise, that a distributed inertia of I be replaced by a lumped 0.33 I at the end of the shaft.

EXAMPLE 9.9

A link of mass 4.347 Kg has an end-point lateral stiffness of 3600 Nt/m. Assuming the drive system is completely rigid, the resonance due to the flexibility of the link will limit control gains. What is ω_{res}?

The 4.347 Kg mass is distributed along the link. Using the method of Fig. 9.13, the effective mass is $(0.23)(4.347) \cong 1.0$ Kg. Hence, the vibration frequency is

$$\omega_{\text{res}} = \sqrt{3600/1.0} = 60 \text{ radians/second} = 60/(2\pi)\text{Hz} \cong 9.6 \text{ Hz}. \tag{9.77}$$

The inclusion of structural flexibilities in the model of the system used for control-law synthesis is required if we wish to achieve closed-loop bandwidths higher than that given by (9.75). The resulting system models are of high order, and the control techniques applicable to this situation become quite sophisticated. Such control schemes are currently beyond the state of the art of industrial practice but are an active area of research [11, 12].

Control of a single joint

In summary, we make the following three major assumptions:

1. The motor inductance l_a can be neglected.
2. Taking into account high gearing, we model the effective inertia as a constant equal to $I_{\max} + \eta^2 I_m$.
3. Structural flexibilities are neglected, except that the lowest structural resonance ω_{res} is used in setting the servo gains.

With these assumptions, a single joint of a manipulator can be controlled with the partitioned controller given by

$$\alpha = I_{\max} + \eta^2 I_m,$$

$$\beta = (b + \eta^2 b_m)\dot{\theta}, \tag{9.78}$$

$$\tau' = \ddot{\theta}_d + k_v \dot{e} + k_p e. \tag{9.79}$$

The resulting system closed-loop dynamics are

$$\ddot{e} + k_v \dot{e} + k_p e = \tau_{\text{dist}}, \tag{9.80}$$

where the gains are chosen as

$$k_p = \omega_n^2 = \frac{1}{4}\omega_{\text{res}}^2,$$

$$k_v = 2\sqrt{k_p} = \omega_{\text{res}}. \tag{9.81}$$

9.10 ARCHITECTURE OF AN INDUSTRIAL-ROBOT CONTROLLER

In this section, we briefly look at the architecture of the control system of the Unimation PUMA 560 industrial robot. As shown in Fig. 9.14, the hardware architecture is that of a two-level hierarchy, with a DEC LSI-11 computer serving as the top-level "master" control computer passing commands to six Rockwell 6503 microprocessors.[5] Each of these microprocessors controls an individual joint with a PID control law not unlike that presented in this chapter. Each joint of the PUMA 560 is instrumented with an incremental optical encoder. The encoders are interfaced to an up/down counter, which the microprocessor can read to obtain the current joint position. There are no tachometers in the PUMA 560; rather, joint positions are differenced on subsequent servo cycles to obtain an estimate of joint velocity. In order to command torques to the DC torque motors, the microprocessor

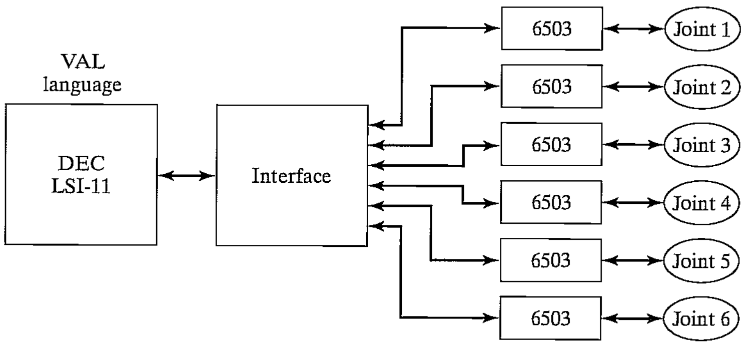


FIGURE 9.14: Hierarchical computer architecture of the PUMA 560 robot-control system.

[5]These simple 8-bit computers are already old technology. It is common these days for robot controllers to be based on 32-bit microprocessors.

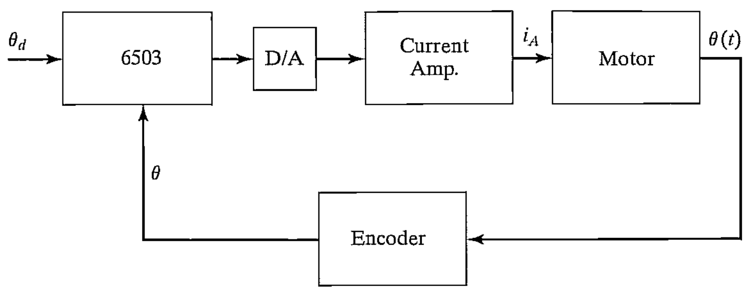


FIGURE 9.15: Functional blocks of the joint-control system of the PUMA 560.

is interfaced to a digital-to-analog converter (DAC) so that motor currents can be commanded to the current-driver circuits. The current flowing through the motor is controlled in analog circuitry by adjusting the voltage across the armature as needed to maintain the desired armature current. A block diagram is shown in Fig. 9.15.

Each 28 milliseconds, the LSI-11 computer sends a new position command (**set-point**) to the joint microprocessors. The joint microprocessors are running on a 0.875 millisecond cycle. In this time, they interpolate the desired position set-point, compute the servo error, compute the PID control law, and command a new value of torque to the motors.

The LSI-11 computer carries out all the "high-level" operations of the overall control system. First of all, it takes care of interpreting the VAL (Unimation's robot programming language) program commands one by one. When a motion command is interpreted, the LSI-11 must perform any needed inverse kinematic computations, plan a desired trajectory, and begin generating trajectory via points every 28 milliseconds for the joint controllers.

The LSI-11 is also interfaced to such standard peripherals as the terminal and a floppy disk drive. In addition, it is interfaced to a **teach pendant**. A teach pendant is a handheld button box that allows the operator to move the robot around in a variety of modes. For example, the PUMA 560 system allows the user to move the robot incrementally in joint coordinates or in Cartesian coordinates from the teach pendant. In this mode, teach-pendant buttons cause a trajectory to be computed "on the fly" and passed down to the joint-control microprocessors.

BIBLIOGRAPHY

[1] W. Boyce and R. DiPrima, *Elementary Differential Equations*, 3rd edition, John Wiley and Sons, New York, 1977.

[2] E. Purcell, *Calculus with Analytic Geometry*, Meredith Corporation, New York, 1972.

[3] G. Franklin and J.D. Powell, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1980.

[4] G. Franklin, J.D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1986.

[5] J. Luh, "Conventional Controller Design for Industrial Robots—a Tutorial," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, No. 3, June 1983.

[6] D. Halliday and R. Resnik, *Fundamentals of Physics*, Wiley, New York 1970.

[7] Y. Koren and A. Ulsoy, "Control of DC Servo-Motor Driven Robots," *Proceedings of Robots 6 Conference*, SME, Detroit, March 1982.

[8] R.P. Paul, *Robot Manipulators*, MIT Press, Cambridge, MA, 1981.

[9] H. Asada and K. Youcef-Toumi, *Direct-Drive Robots—Theory and Practice*, MIT Press, Cambridge, MA, 1987.

[10] J. Shigley, *Mechanical Engineering Design*, 3rd edition, McGraw-Hill, New York, 1977.

[11] W. Book, "Recursive Lagrangian Dynamics of Flexible Manipulator Arms," *The International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[12] R. Cannon and E. Schmitz, "Initial Experiments on the End-Point Control of a Flexible One Link Robot," *The International Journal of Robotics Research*, Vol. 3, No. 3, 1984.

[13] R.J. Nyzen, "*Analysis and Control of an Eight-Degree-of-Freedom Manipulator*," Ohio University Master's Thesis, Mechanical Engineering, Dr. Robert L. Williams II, Advisor, August 1999.

[14] R.L. Williams II, "*Local Performance Optimization for a Class of Redundant Eight-Degree-of-Freedom Manipulators*," **NASA Technical Paper 3417**, NASA Langley Research Center, Hampton, VA, March 1994.

EXERCISES

9.1 [20] For a second-order differential equation with complex roots

$$s_1 = \lambda + \mu i,$$
$$s_2 = \lambda - \mu i,$$

show that the general solution

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t},$$

can be written

$$x(t) = c_1 e^{\lambda t} \cos(\mu t) + c_2 e^{\lambda t} \sin(\mu t).$$

9.2 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 2$, $b = 6$, and $k = 4$ and the block (initially at rest) is released from the position $x = 1$.

9.3 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 2$, and $k = 1$ and the block (initially at rest) is released from the position $x = 4$.

9.4 [13] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 4$, and $k = 5$ and the block (initially at rest) is released from the position $x = 2$.

9.5 [15] Compute the motion of the system in Fig. 9.2 if parameter values are $m = 1$, $b = 7$, and $k = 10$ and the block is released from the position $x = 1$ with an initial velocity of $x = 2$.

9.6 [15] Use the (1, 1) element of (6.60) to compute the variation (as a percentage of the maximum) of the inertia "seen" by joint 1 of this robot as it changes configuration. Use the numerical values

$$l_1 = l_2 = 0.5 \text{ m},$$
$$m_1 = 4.0 \text{ Kg},$$
$$m_2 = 2.0 \text{ Kg}.$$

Consider that the robot is direct drive and that the rotor inertia is negligible.

9.7 [17] Repeat Exercise 9.6 for the case of a geared robot (use $\eta = 20$) and a rotor inertia of $I_m = 0.01$ Kg m^2.

9.8 [18] Consider the system of Fig. 9.6 with the parameter values $m = 1$, $b = 4$, and $k = 5$. The system is also known to possess an unmodeled resonance at $\omega_{\text{res}} = 6.0$ radians/second. Determine the gains k_v and k_p that will critically damp the system with as high a stiffness as is reasonable.

9.9 [25] In a system like that of Fig. 9.12, the inertial load, I, varies between 4 and 5 Kg-m^2. The rotor inertia is $I_m = 0.01$ Kg-m^2, and the gear ratio is $\eta = 10$. The system possesses unmodeled resonances at 8.0, 12.0, and 20.0 radians/second. Design α and β of the partitioned controller and give the values of k_p and k_v such that the system is never underdamped and never excites resonances, but is as stiff as possible.

9.10 [18] A designer of a direct-drive robot suspects that the resonance due to beam flexibility of the link itself will be the cause of the lowest unmodeled resonance. If the link is approximately a square-cross-section beam of dimensions $5 \times 5 \times 50$ cm with a 1-cm wall thickness and a total mass of 5 Kg, estimate ω_{res}.

9.11 [15] A direct-drive robot link is driven through a shaft of stiffness 1000 Nt-m/radian. The link inertia is 1 Kg-m^2. Assuming the shaft is massless, what is ω_{res}?

9.12 [18] A shaft of stiffness 500 Nt-m/radian drives the input of a rigid gear pair with $\eta = 8$. The output of the gears drives a rigid link of inertia 1 Kg-m^2. What is the ω_{res} caused by flexibility of the shaft?

9.13 [25] A shaft of stiffness 500 Nt-m/radian drives the input of a rigid gear pair with $\eta = 8$. The shaft has an inertia of 0.1 Kg-m^2. The output of the gears drives a rigid link of inertia 1 Kg-m^2. What is the ω_{res} caused by flexibility of the shaft?

9.14 [28] In a system like that of Fig. 9.12, the inertial load, I, varies between 4 and 5 Kg-m^2. The rotor inertia is $I_m = 0.01$ Kg-m^2, and the gear ratio is $\eta = 10$. The system possesses an unmodeled resonance due to an end-point stiffness of the link of 2400 Nt-m/radian. Design α and β of the partitioned controller, and give the values of k_p and k_v such that the system is never underdamped and never excites resonances, but is as stiff as possible.

9.15 [25] A steel shaft of length 30 cm and diameter 0.2 cm drives the input gear of a reduction of $\eta = 8$. The rigid output gear drives a steel shaft of length 30 cm and diameter 0.3 cm. What is the range of resonant frequencies observed if the load inertia varies between 1 and 4 Kg-m^2?

PROGRAMMING EXERCISE (PART 9)

We wish to simulate a simple trajectory-following control system for the three-link planar arm. This control system will be implemented as an independent-joint PD (proportional plus derivative) control law. Set the servo gains to achieve closed-loop stiffnesses of 175.0, 110.0, and 20.0 for joints 1 through 3 respectively. Try to achieve approximate critical damping.

Use the simulation routine **UPDATE** to simulate a discrete-time servo running at 100 Hz—that is, calculate the control law at 100 Hz, not at the frequency of the numerical integration process. Test the control scheme on the following tests:

1. Start the arm at $\Theta = (60, -110, 20)$ and command it to stay there until $time = 3.0$, when the set-points should instantly change to $\Theta = (60, -50, 20)$. That is, give a step input of 60 degrees to joint 2. Record the error–time history for each joint.
2. Control the arm to follow the cubic-spline trajectory from Programming Exercise Part 7. Record the error–time history for each joint.

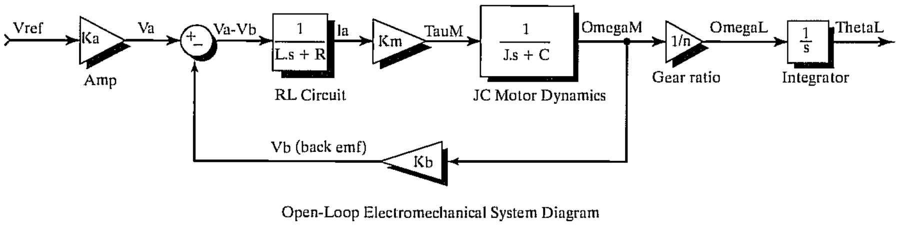
MATLAB EXERCISE 9

This exercise focuses on linearized independent joint-control simulation for the shoulder joint (joint 2) of the NASA eight-axis AAI ARMII (Advanced Research Manipulator II) manipulator arm—see [14]. Familiarity with linear classical feedback-control systems, including block diagrams and Laplace transforms, is assumed. We will use Simulink, the graphical user interface of MATLAB.

Figure 9.16 shows a linearized open-loop system-dynamics model for the ARMII electromechanical shoulder joint/link, actuated by an armature-controller DC servomotor. The open-loop input is reference voltage V_{ref} (boosted to armature voltage via an amplifier), and the output of interest is the load shaft angle ThetaL. The figure also shows the feedback-control diagram, where the load-shaft angle is sensed via an optical encoder and provided as feedback to the PID controller. The table describes all system parameters and variables.

If we reflect the load shaft inertia and damping to the motor shaft, the effective polar inertia and damping coefficient are $J = J_M + J_L(t)/n^2$ and $C = C_M + C_L/n^2$. By virtue of the large gear ratio n, these effective values are not much different from the motor-shaft values. Thus, the gear ratio allows us to ignore variations in the configuration-dependent load-shaft inertia $J_L(t)$ and just set a reasonable average value.

The ARMII shoulder joint constant parameters are given in the accompanying table [13]. Note that we can use the English units directly, because their effect cancels out inside the control diagram. Also, we can directly use deg units for the angle. Develop a Simulink model to simulate the single-joint control model from the model and feedback-control diagram shown; use the specific parameters from the table. For the nominal case, determine the PID gains by trial and error for "good" performance (reasonable percent overshoot, rise time, peak time, and settling time). Simulate the resulting motion for moving this shoulder joint for a step input of 0 to 60 deg. Plot the simulated load-angle value over time, plus the load-shaft angular velocity over time. In addition, plot the



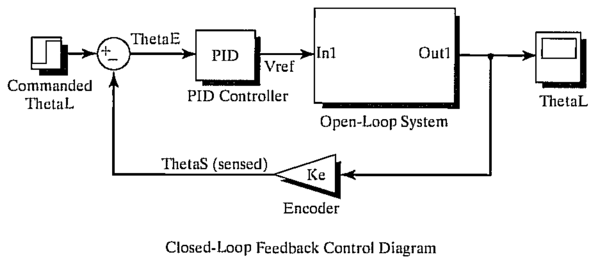


FIGURE 9.16: Linearized open-loop system-dynamics model for the ARMII electromechanical shoulder joint/link, actuated by an armature-controller DC servomotor.

TABLE 9.1: ARMII shoulder joint constant parameters.

$V_a(t)$	armature voltage	$\tau_M(t)$	generated motor torque	$\tau_L(t)$	load torque
$L = 0.0006H$	armature induc- tance	$\theta_M(t)$	motor shaft angle	$\theta_L(t)$	load shaft angle
$R = 1.40\Omega$	armature resis- tance	$\omega_M(t)$	motor shaft velocity	$\omega_L(t)$	load shaft velocity
$i_a(t)$	armature current	$J_M = 0.00844$ lb$_f$-in-s^2	lumped motor polar inertia	$J_L(t) = 1$ lb$_f$-in-s^2	lumped load polar inertia
$V_b(t)$	back emf voltage	$C_M = 0.00013$ lb$_f$-in/deg/s	motor shaft viscous damping coefficient	$C_L = 0.5$ lb$_f$-in/deg/s	load shaft viscous damping coefficient
$K_a = 12$	amplifier gain	$n = 200$	gear ratio	$g = 0$ in/s^2	gravity (ignore gravity at first)
$K_b = 0.00867$ V/deg/s	back emf constant	$K_M = 4.375$ lb$_f$-in/A	torque con- stant	$K_e = 1$	encoder transfer function

control effort—that is, the armature voltage V_a over time. (On the same graph, also give the back emf V_b.)

Now, try some changes—Simulink is so easy and enjoyable to change:

1) The step input is frustrating for controller design, so try a ramped step input instead: Ramp from 0 to 60 deg in 1.5 sec, then hold the 60-deg command for all time greater than 1.5 sec. Redesign PID gains and restimulate.
2) Investigate whether the inductor L is significant in this system. (The electrical system rises much faster than the mechanical system—this effect can be represented by time constants.)
3) We don't have a good estimate for the load inertia and damping (J_L and C_L). With your best PID gains from before, investigate how big these values can grow (scale the nominal parameters up equally) before they affect the system.
4) Now, include the effect of gravity as a disturbance to the motor torque T_M. Assume that the moving robot mass is 200 lb and the moving length beyond joint 2 is 6.4 feet. Test for the nominal "good" PID gains you found; redesign if necessary. The shoulder load angle θ_2 zero configuration is straight up.

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

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

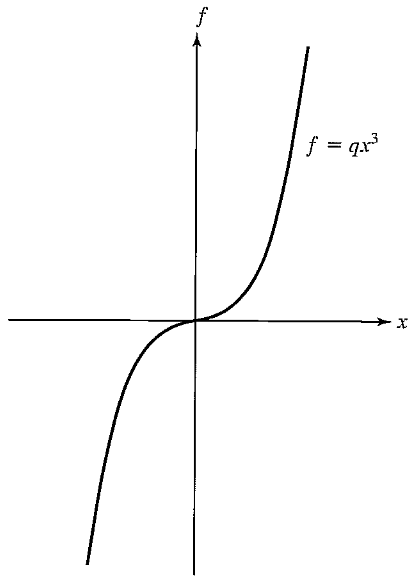


FIGURE 10.1: The force-vs.-distance characteristic of a nonlinear spring.

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.

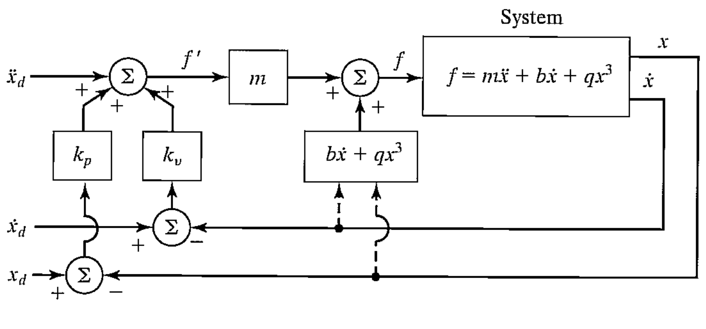


FIGURE 10.2: A nonlinear control system for a system with a nonlinear spring.

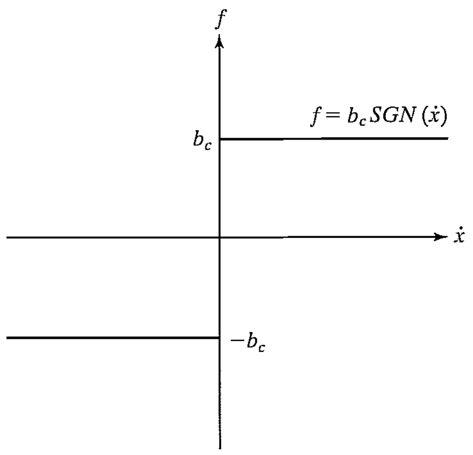


FIGURE 10.3: The force-vs.-velocity characteristic of Coulomb friction.

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.

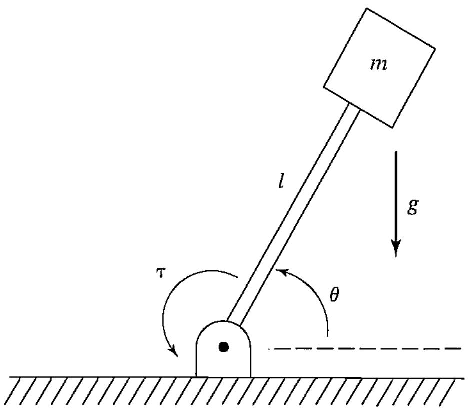


FIGURE 10.4: An inverted pendulum or a one-link manipulator.

The model of the manipulator is

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

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

$$\begin{aligned} \alpha &= ml^2, \\ \beta &= v\dot{\theta} + c\,sgn(\dot{\theta}) + mlg\cos(\theta); \end{aligned} \tag{10.7}$$

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

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

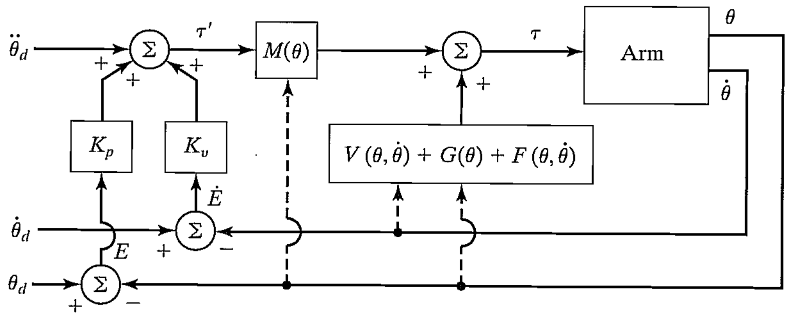


FIGURE 10.5: A model-based manipulator-control system.

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

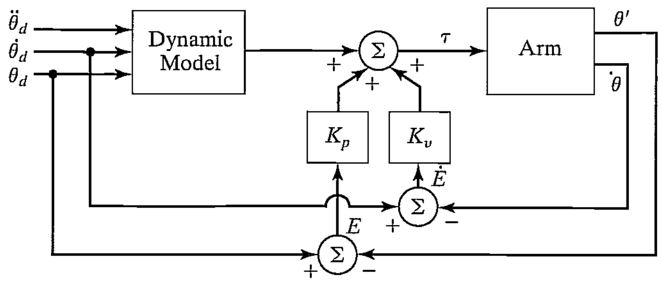


FIGURE 10.6: Control scheme with the model-based portion "outside" the servo loop.

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

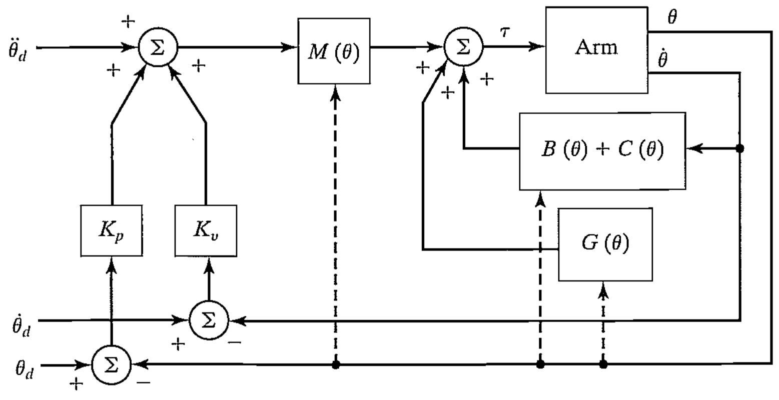


FIGURE 10.7: An implementation of the model-based manipulator-control system.

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

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

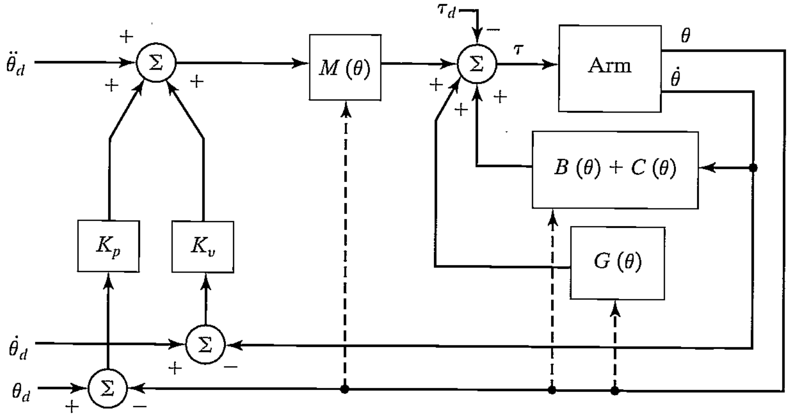


FIGURE 10.8: The model-based controller with an external disturbance acting.

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

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

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 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

$$\begin{aligned} \alpha &= I, \\ \beta &= 0, \end{aligned} \tag{10.26}$$

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 was discussed in Section 9.10.

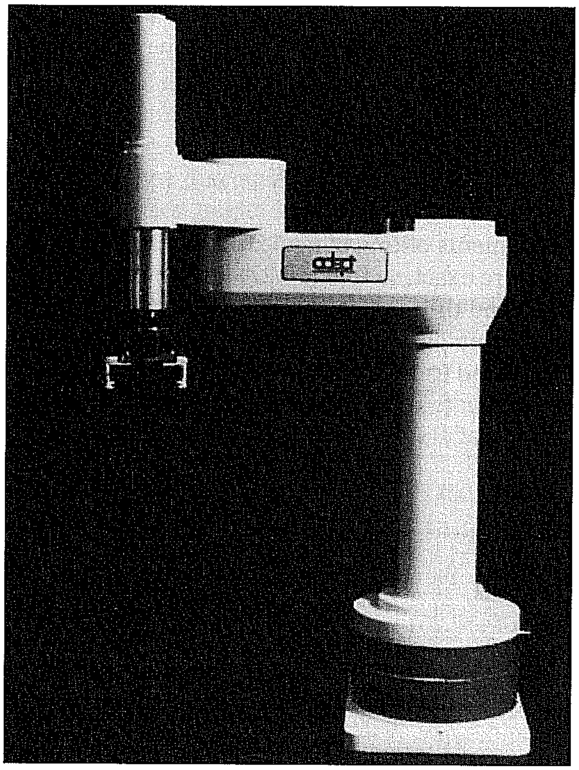


FIGURE 10.9: The Adept One, a direct-drive robot by Adept Technology, Inc.

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

$$m\ddot{x} + b\dot{x} + kx = 0. \qquad (10.29)$$

The total energy of the system is given by

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

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

$$\dot{v} = m\dot{x}\ddot{x} + kx\dot{x}. \qquad (10.31)$$

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

$$\dot{v} = -b\dot{x}^2, \qquad (10.32)$$

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 $\upsilon(X)$ that has the following properties:

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

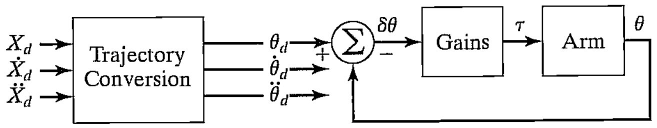


FIGURE 10.10: A joint-based control scheme with Cartesian-path input.

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

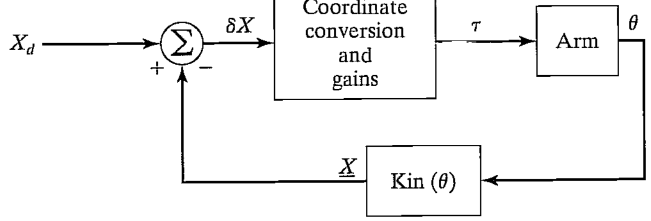


FIGURE 10.11: The concept of a Cartesian-based control scheme.

[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

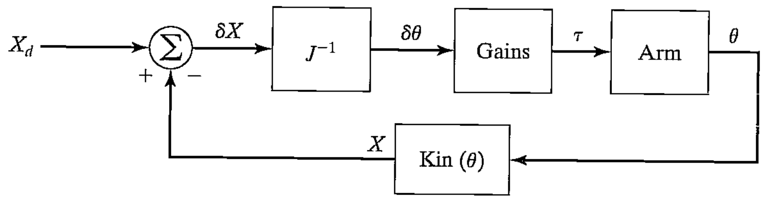


FIGURE 10.12: The inverse-Jacobian Cartesian-control scheme.

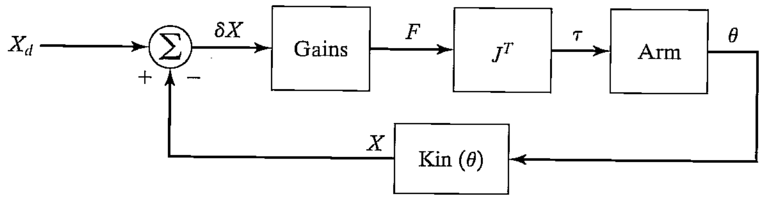


FIGURE 10.13: The transpose-Jacobian Cartesian-control scheme.

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

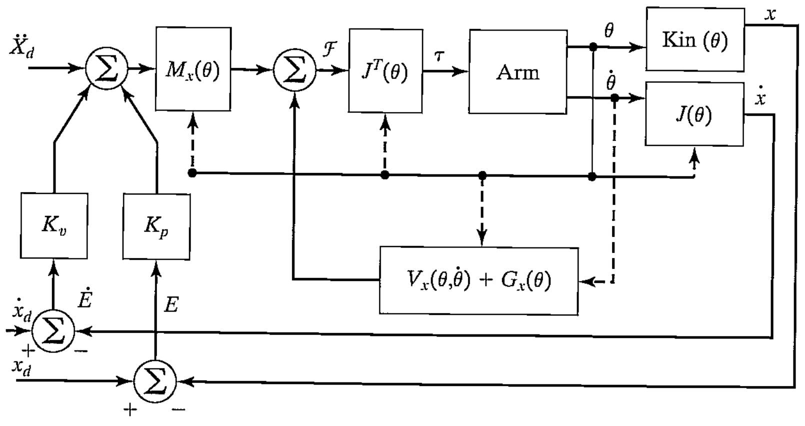


FIGURE 10.14: The Cartesian model-based control scheme.

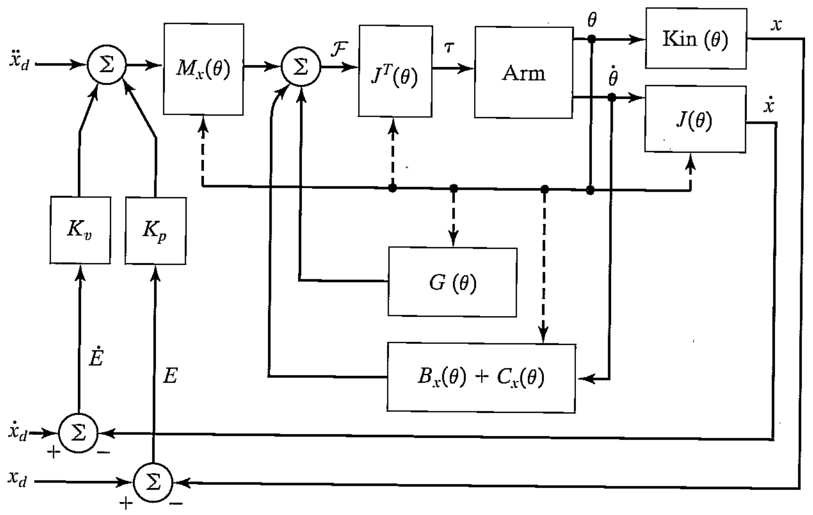


FIGURE 10.15: An implementation of the Cartesian model-based control scheme.

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

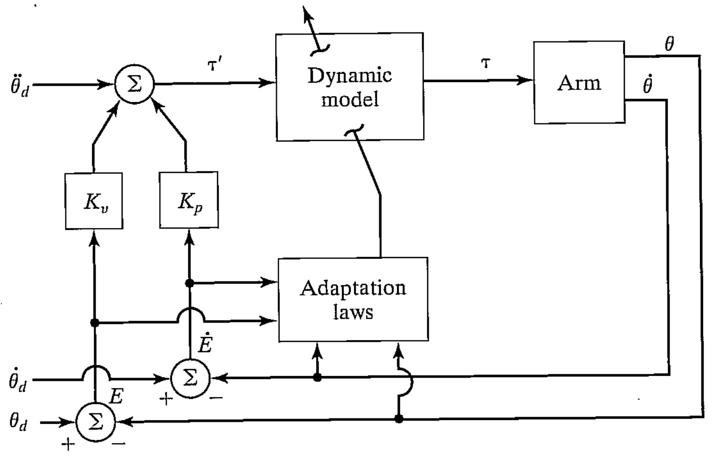


FIGURE 10.16: The concept of an adaptive manipulator controller.

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

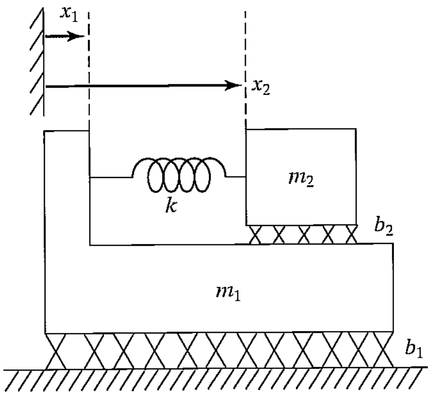


FIGURE 10.17: Mechanical system with two degrees of freedom.

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.