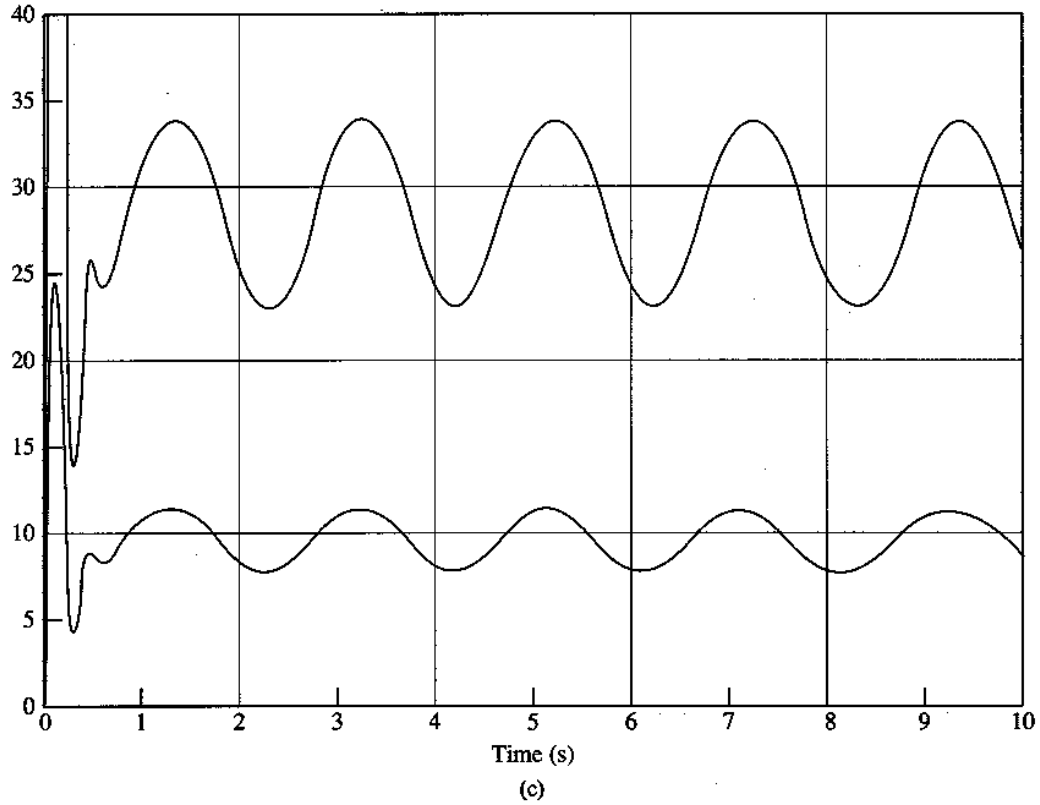


FIGURE 3.4-10 (*Cont.*) (c) $\omega_n = 50$ rad/s.

Classical Joint Control

A simple controller that often gives good results in practice is obtained by selecting in (3.4-43)

$$\hat{M} = I, \qquad \hat{N} = -\ddot{q}_d. \tag{3.4-50}$$

Then there results

$$\tau_c = -u. \tag{3.4-51}$$

By selecting u_i to depend only on joint variable i, this describes n decoupled individual joint controllers and is called *independent joint control.* Implementation on an actual robot arm is easy since the control input for joint i only depends on locally measured variables, not on the variables of the other joints. Moreover, the computations are easy and do not involve solving the complicated nonlinear robot inverse dynamics.

During the early days of robotics, independent joint control was popular [Paul 1981] since it allows a decoupled analysis of the closed-loop system using single-input/single-output (SISO) classical techniques. It is also called

classical joint control. There are strong arguments even today by several researchers that this control scheme is always suitable in practical implementations, and that modern nonlinear control schemes are too complicated for industrial robotic applications.

A traditional analysis of independent joint control control now follows. It provides a connection with classical control notions and is important for the robot controls designer to understand. See [Franklin et al. 1986] for a reference on classical control theory.

A simplified dynamical model of a robot arm with electric actuators may be written as (Section 2.6)

$$(J_m + r_i^2 m_{ii})\ddot{\theta}_i + (B_m + \frac{k_b k_m}{R})\dot{\theta}_i = \frac{k_m}{R} v_i - r_i d_i, \qquad i = 1, \ldots, n \quad (3.4\text{-}52)$$

with J_m the actuator motor inertia, B_m the rotor damping constant, k_m the torque constant, k_b the back emf constant, R the armature resistance, and r_i the gear ratio for joint i. The motor angle is denoted $\theta_i(t)$. The constant portions of the diagonal elements of $M(q)$ are denoted m_{ii}. The time-varying portions of these elements, as well as the off-diagonal elements of $M(q)$, the nonlinear terms $N(q,\dot{q})$, and any disturbances τ_d are all lumped into the disturbance $d_i(t)$. Thus $d_i(t)$ contains the effects on joint i of all the other joints. The control input is the motor armature voltage $v_i(t)$.

Note that predominantly motor parameters appear in this equation. In fact, if the gear ratio is small, even m_{ii} may be neglected. For this reason, if the gear ratio is small, the robot arm control problem virtually reduces to the problem of controlling the actuator motors.

Let us denote this simplified linear time-invariant model of manipulator joint i as

$$J\ddot{\theta} + B\dot{\theta} = u - rd. \quad (3.4\text{-}53)$$

where the constant k_m/R has been incorporated into the definitions of J and B. According to the properties in Table 2.3-1, the disturbance $d(t)$ is bounded, although not generally by a constant. The bound may be a function of $q(t)$, $\dot{q}(t)$, and even $\ddot{q}(t)$. This is generally ignored in a classical analysis. The effects of $d(t)$ are, however, somewhat ameliorated by multiplication by the gear ratio r. The effects of joint compliance (Chapter 6) are also ignored in classical joint control design. [For consistency with classical notation, there is a sign change in this section in the definition of $u(t)$ compared to our previous usage.]

Now, let us consider a few selections for the control input $u(t)$.

PD Control. Selecting the PD control law

$$u = k_v\dot{e} + k_p e, \quad (3.4\text{-}54)$$

with $e(t) = \theta_{d_i}(t) - \theta_i(t)$ the tracking error for motor angle i, yields the closed-loop system shown in Fig. 3.4-11. Recall that the motor angle is related to

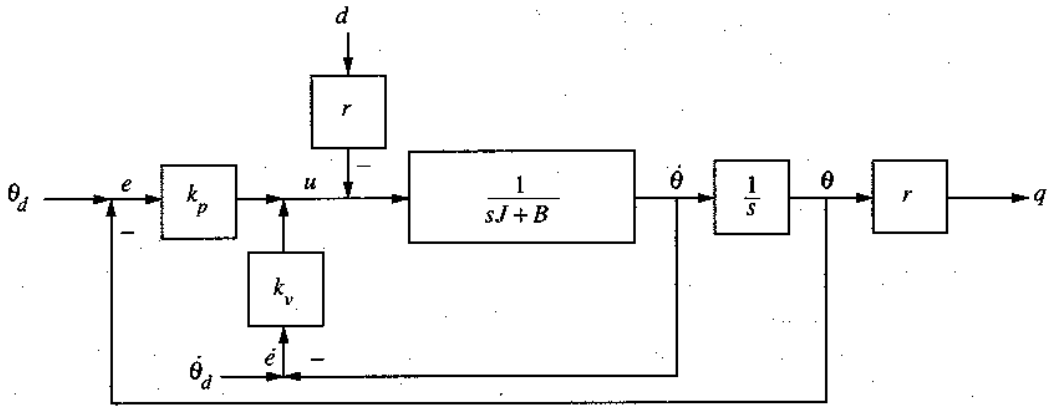


FIGURE 3.4-11 PD independent joint control.

the joint variable by $q_i = r_i\theta_i$. Thus $\theta_{d_i}(t)$ may be computed from $q_{d_i}(t)$. This PD controller is easily implemented on an actual arm with very little computing power.

Using Mason's theorem from classical control, the closed-loop transfer function for set-point tracking (e.g., with $\dot{\theta}_d = 0$) is found to be (see the problems)

$$\theta = \frac{k_p}{\Delta(s)}\,\theta_d - \frac{r}{\Delta(s)}\,d \tag{3.4-55}$$

with the closed-loop characteristic polynomial

$$\Delta(s) = s^2J + s(B + k_v) + k_p. \tag{3.4-56}$$

The PD gains can be selected to obtain a suitable natural frequency and damping ratio, as we have seen.

At steady state, the only nonzero contribution to the disturbance $d(t)$ is the neglected gravity $G(q)$. Table 2.3-1 shows that the gravity vector is bounded by a known value g_b for a given robot arm. Therefore, at steady state,

$$|d| < g_b.$$

Using the final value theorem, the steady-state tracking error for joint i using PD control is bounded by

$$e_{ss} < rg_b/k_p. \tag{3.4-57}$$

Therefore, for set-point tracking where a final value for q_d is specified and $\dot{q}_d = 0$, PD control with a large k_p might be suitable.

As a matter of fact, the next results show that PD control is often very suitable even for following a desired trajectory, not only for set-point control. It is proven in [Dawson 1990].

THEOREM 3.4-2: *If the PD control law* (3.4-54) *is applied to each joint and* $e(0) = 0, \dot{e}(0) = 0$, *the position and velocity tracking errors are bounded within a ball whose radius decreases approximately (for large* k_v*) as* $1/\sqrt{k_v}$. ■

This result gives credence to those who maintain that PD control is often good enough for practical applications.

Of course, the point is that k_v cannot be increased without limit without hitting the actuator torque limits. Other schemes to be discussed in the book allow good trajectory following without such large torques.

In [Paul 1981] are discussed several methods for modifying the PD control law to obtain better performance. These include gravity compensation [which yields exactly controller (3.4-49)], and acceleration feedforward, which amounts to using

$$u = k_v\dot{e} + k_p e + J\ddot{\theta}_{d_i} \tag{3.4-58}$$

for each joint. Also mentioned is joint-coupling control, which amounts to adding back into $u(t)$ some of the neglected terms in $M(q)$ and $N(q,\dot{q})$ that describe the interactions between the joints. Thus such corrections involve using better estimates for $\hat{M}$ and $\hat{N}$ in the approximate computed-torque control (3.4-43).

PID Control. It has been seen that PD independent joint control is often suitable for tracking control. However, at steady state there is a residual error (3.4-57) due to gravity. This can be removed using the *PID independent joint control law*

$$\begin{aligned} \dot{\varepsilon} &= e \\ u &= k_v\dot{e} + k_p e + k_i\varepsilon \end{aligned} \tag{3.4-59}$$

for each joint. See Fig. 3.4-12..

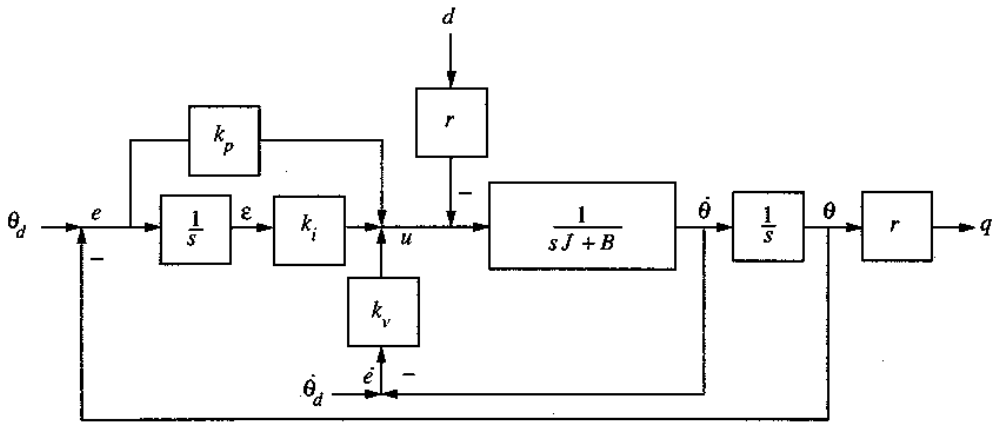


FIGURE 3.4-12 PID independent joint control.

TABLE 3.4-1 Computed-Torque-Like Robot Controllers

Robot Dynamics:

$$M(q)\ddot{q} + V(q,\dot{q}) + F_v\dot{q} + F_d(\dot{q}) + G(q) + \tau_d = \tau$$

or

$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau$$

where

$$N(q,\dot{q}) \equiv V(q,\dot{q}) + F_v\dot{q} + F_d(\dot{q}) + G(q)$$

Tracking Error:

$$e(t) = q_d(t) - q(t)$$

PD Computed-Torque:

$$\tau = M(q)\,(\ddot{q}_d + K_v\dot{e} + K_p e) + N(q,\dot{q})$$

PID Computed-Torque:

$$\dot{\varepsilon} = e$$

$$\tau = M(q)\,(\ddot{q}_d + K_v\dot{e} + K_p e + K_i\varepsilon) + N(q,\dot{q})$$

Approximate Computed-Torque Control:

$$\tau_c = \hat{M}(\ddot{q}_d - u) + \hat{N}$$

Error System:

$$\ddot{e} = (I - \Delta)u + d$$

where $\Delta = I - M^{-1}\hat{M}$, $\delta = M^{-1}(N - \hat{N})$

$$d(t) = M^{-1}\tau_d + \Delta\ddot{q}_d(t) + \delta(t)$$

PD-Gravity Control:

$$\tau_c = K_v\dot{e} + K_p e + G(q)$$

Classical PD Control:

$$\tau_c = K_v\dot{e} + K_p e$$

Classical PID Control:

$$\dot{\varepsilon} = e$$

$$\tau_c = K_v\dot{e} + K_p e + K_i\varepsilon$$

Digital Control (see Section 3.5):

$$\tau_k = M(q_k)\,(\ddot{q}_k^d + K_v\dot{e}_k + K_p e_k) + N(q_k,\dot{q}_k)$$

Now the transfer function for set-point tracking ($\dot{\theta}_d = 0$) is

$$\theta = \frac{k_p s + k_i}{\Delta_1(s)}\theta_d - \frac{rs}{\Delta_1(s)}d, \tag{3.4-60}$$

with closed-loop characteristic polynomial

$$\Delta_1(s) = s^3 J + s^2(B + k_v) + sk_p + k_i. \tag{3.4-61}$$

Now the final value theorem shows that the steady-state error for set-point control is zero. The Routh test shows that for stability it is required that

$$k_i < (B + k_v)k_p/J. \tag{3.4-62}$$

In using an integral term in the control law, it is important to be aware of the possibility of *integrator windup* due to actuator saturation limits, which was discussed earlier in the section "PID Outer-Loop Design."

EXAMPLE 3.4-4: Classical Joint Control and Torque Saturation Limits

In Example 3.4-1 we simulated the exact computed-torque control law for a two-link planar elbow arm. Here we want to show the results of using PD and PID classical joint control on the same arm (with the same desired trajectory). We are also interested in demonstrating the effects of torque saturation limits. To highlight the effects without extraneous details, we shall use only the arm dynamics and no actuator dynamics or gear ratio. These may easily be included by making some slight modifications to the subroutine arm(x,xp) in Fig. 3.4-2.

a. PD Independent Joint Control

For the two-link arm, PD independent joint control is simply

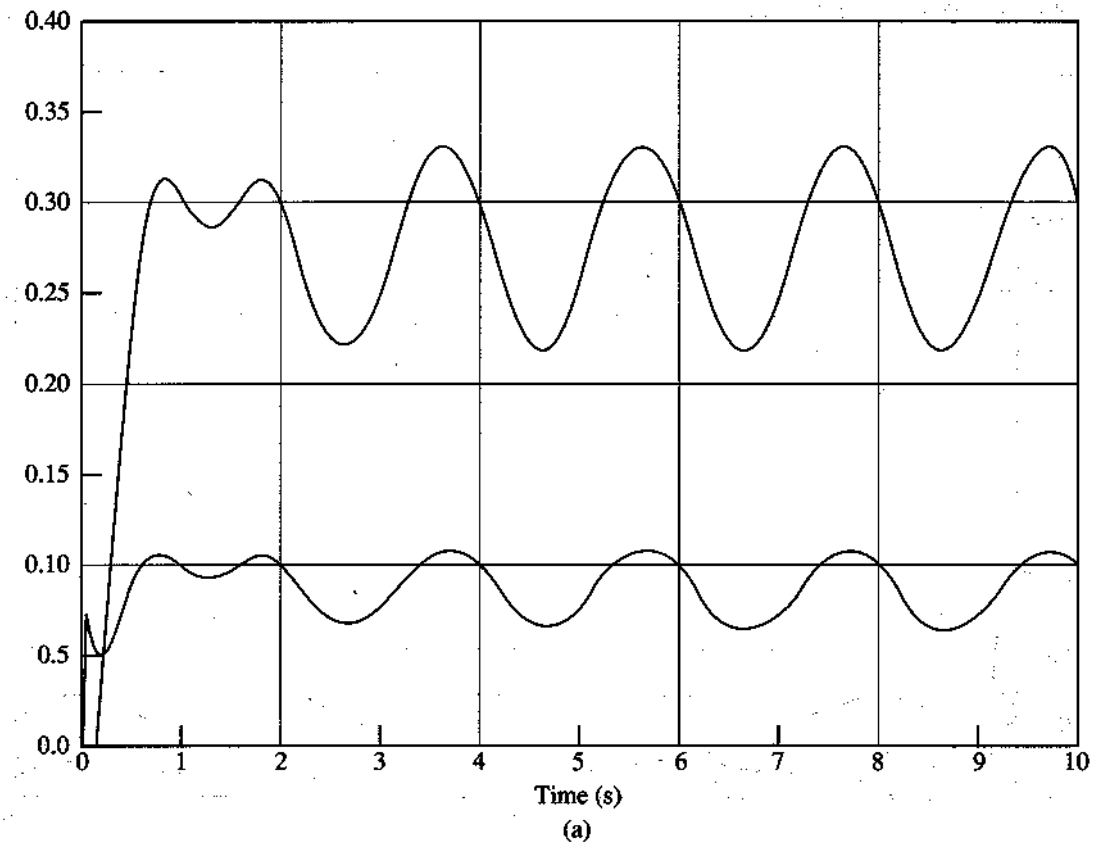
$$\tau_j = k_v\dot{e}_j + k_p e_j, \qquad j = 1,2, \tag{1}$$

with the tracking error $e(t) = q_d(t) - q(t)$. This control law is very simple and direct to implement on an actual arm without even using a DSP. It is much simpler than the control in Example 3.4-1.

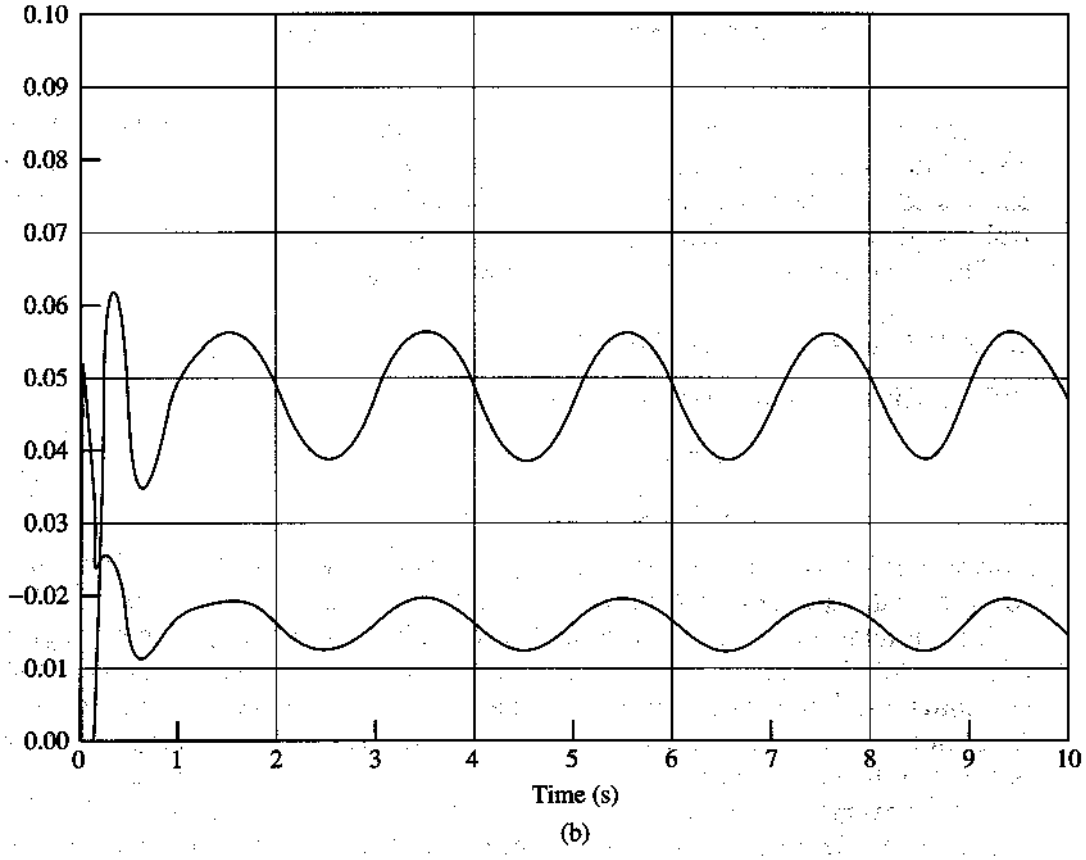
The results of using the PD controller on the two-link arm are shown in Fig. 3.4-13 for several values of a closed-loop natural frequency ω_n. Recall that the PD gains for critical damping are

$$k_p = \omega_n^2, \qquad k_v = 2\omega_n. \tag{2}$$

The low-gain results ($k_p = 100$, $k_v = 20$) in part (a) of the figure are very bad (note the scale). However, the high-gain results in part (c) are much better. Even higher gains would cause a further tracking error decrease.



(a)



(b)

FIGURE 3.4-13 PD classical joint control tracking error $e_1(t)$, $e_2(t)$ (rad): (a) $\omega_n = 10$ rad/s; (b) $\omega_n = 25$ rad/s.

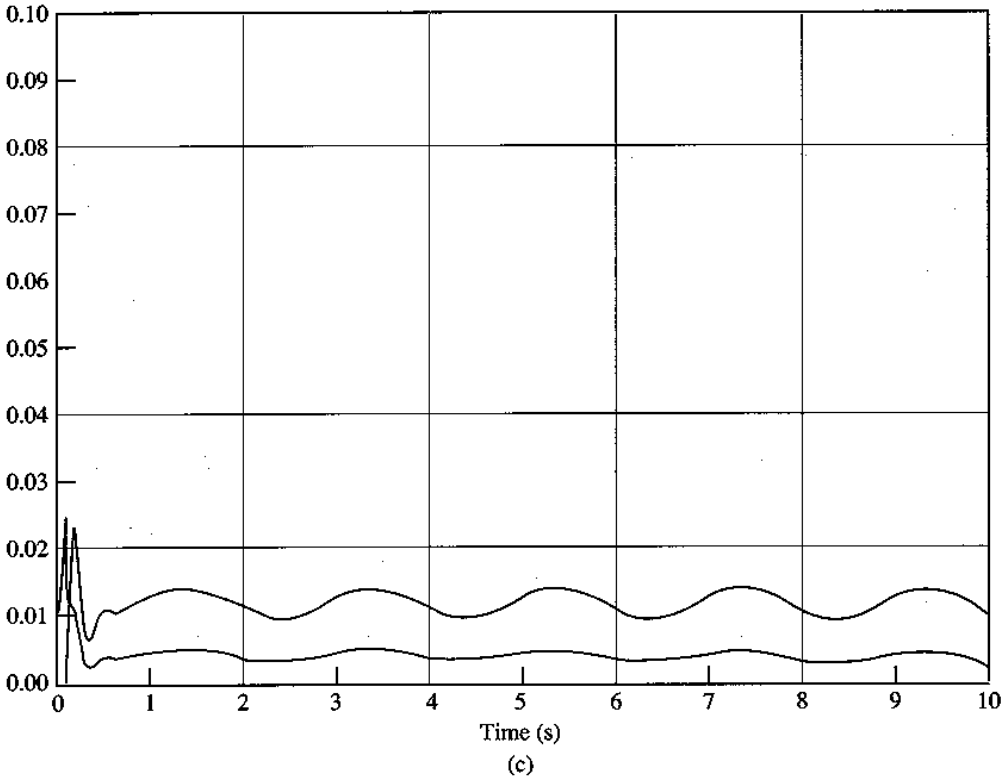


FIGURE 3.4-13 (*Cont.*) (c) $\omega_n = 50$ rad/s.

It is important to notice that there is in all cases a dc component of the tracking error. This is due to the fact that the gravity terms are neglected and should be contrasted with the results of Example 3.4-3. Adding the gravity terms in the control law, therefore, significantly increases the tracking performance.

The associated control torques are shown in Fig. 3.4-14.

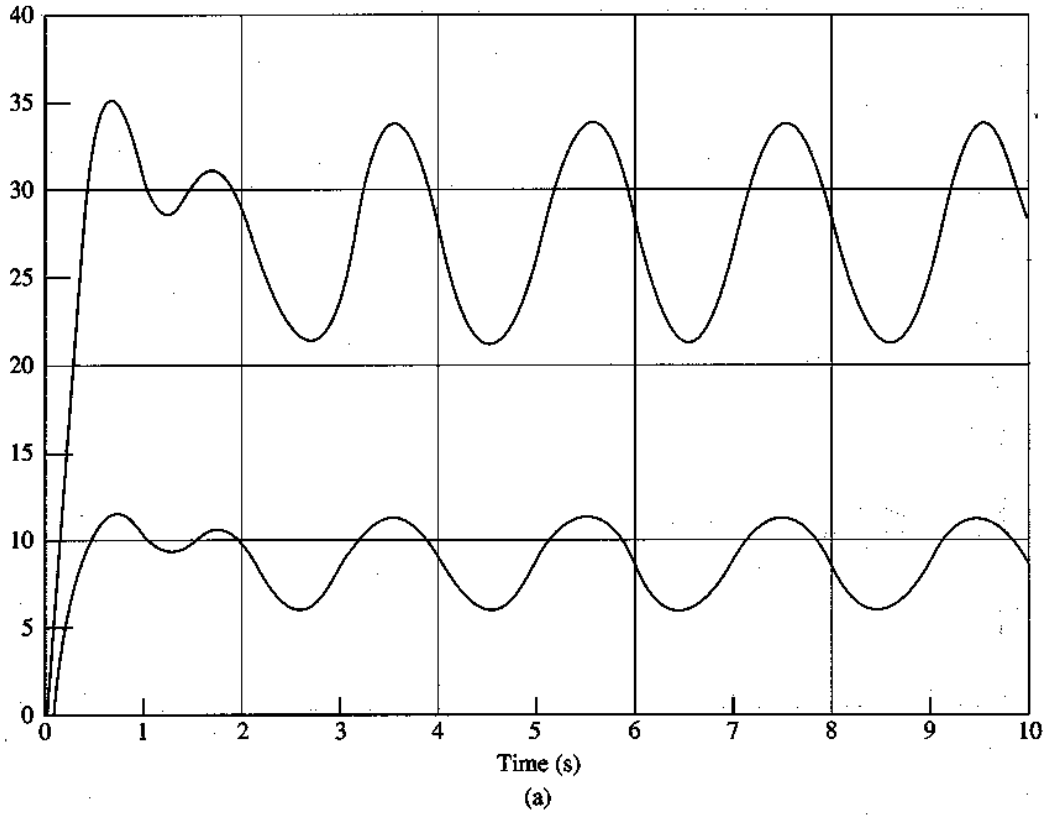
b. PID Independent Joint Control

PID independent joint control for the two-link arm is

$$\begin{aligned} \dot{\varepsilon}_j &= e_j, \\ \tau_j &= k_v \dot{e}_j + k_p e_j + k_i \varepsilon_j, \qquad j = 1,2. \end{aligned} \tag{3}$$

This simple law is easily implemented by adding two states, x(5) and x(6), to subroutine arm(x,xp) in Fig. 3.4-2 to account for the integrators (see Example 3.4-2).

The simulation in Fig. 3.4-13c was repeated, but now adding an integral gain of $k_i = 1000$. The result is shown in Fig. 3.4-15. After several iterations of the sinusoidal motion, the dc value of the tracking errors goes to zero, so that the performance is much improved. That is, adding an integral term can compensate for neglecting the gravity terms in the control law. However, it takes awhile for the error to converge to a zero dc value, and the results are still not as good as the PD-gravity controller in Example 3.4-3 for the same PD gains. On the other hand, the integral term can reject other terms besides gravity (e.g., friction of some sorts). The torque control input was virtually the same in form as Fig. 3.4-14c.



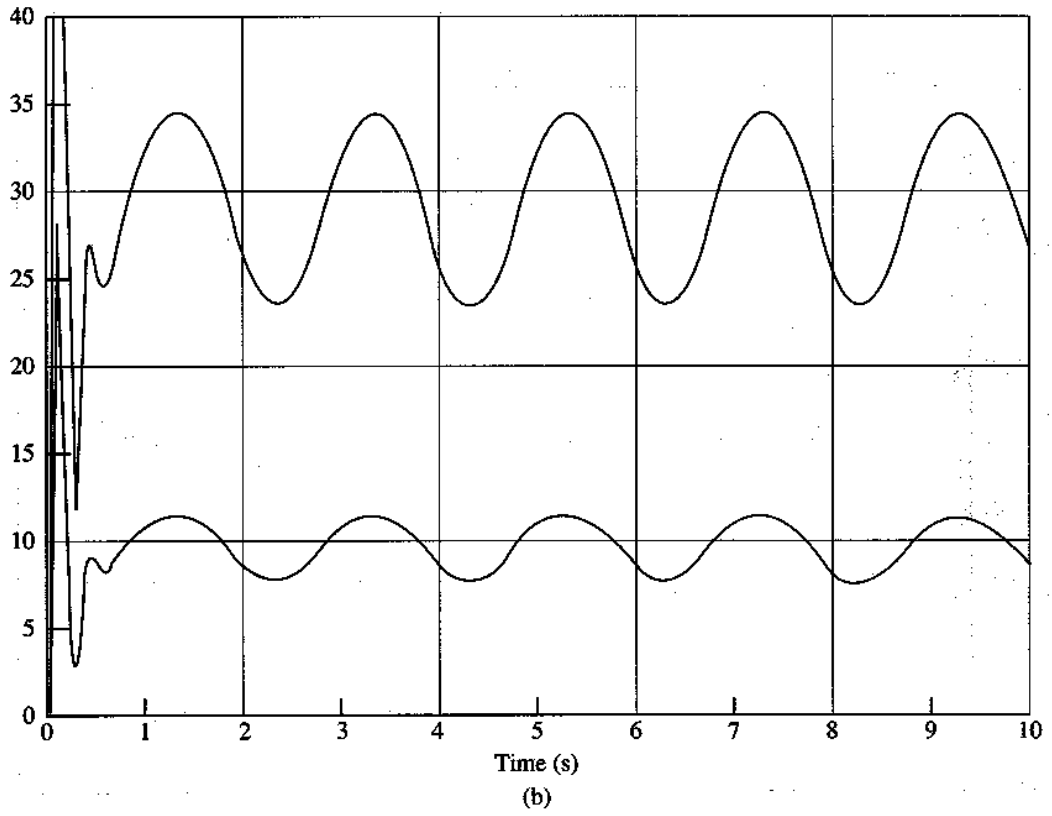
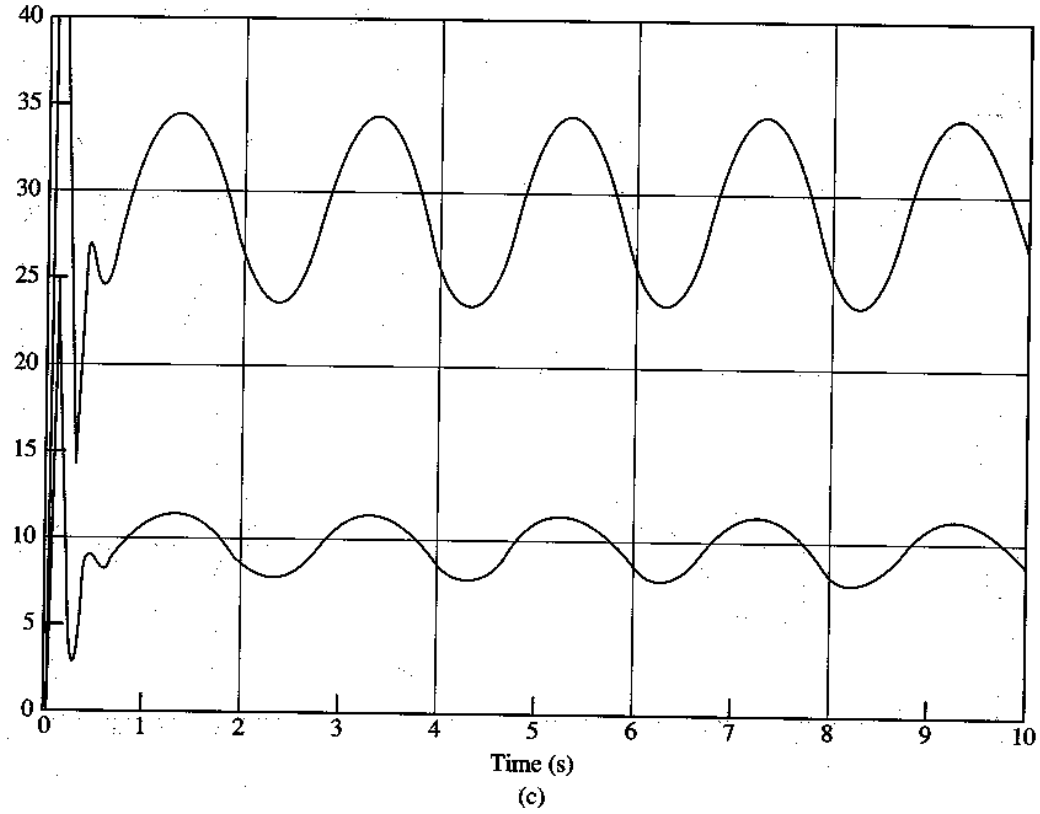


FIGURE 3.4-14 PD classical joint control torque inputs (N-m): (a) $\omega_n = 10$ rad/s; (b) $\omega_n = 25$ rad/s.



(c)

FIGURE 3.4-14 (*Cont.*) (c) $\omega_n = 50$ rad/s.

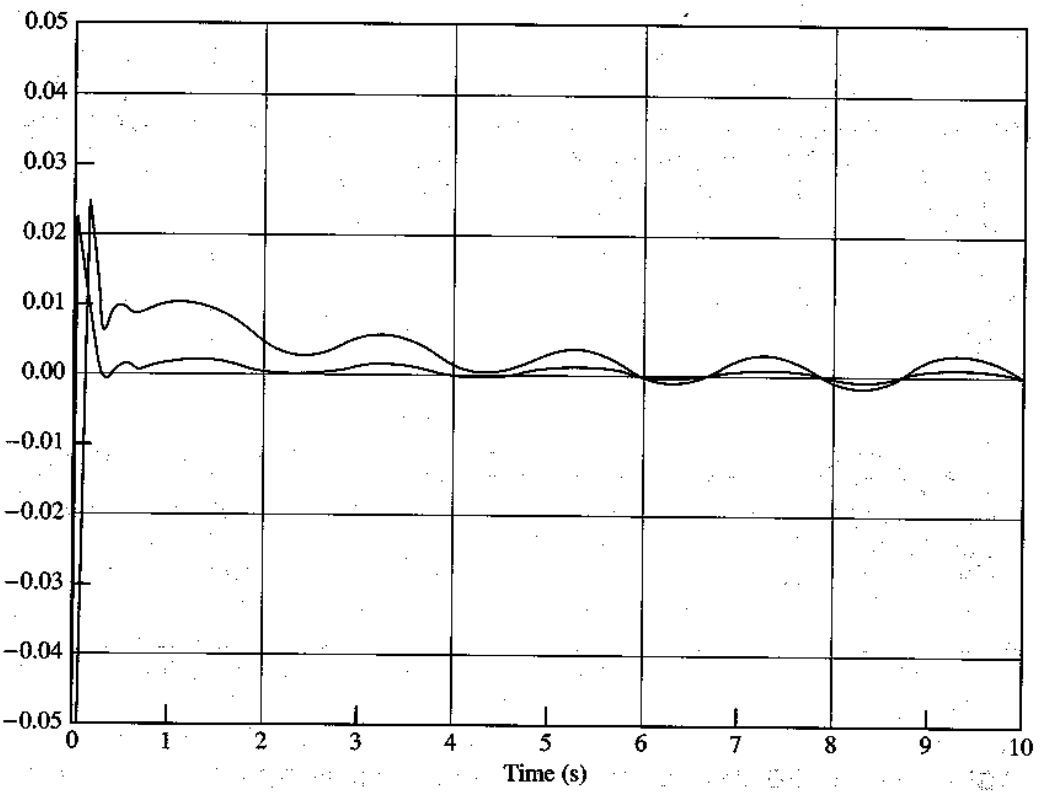


FIGURE 3.4-15 PID classical joint control: $k_v = 100$, $k_p = 2500$, $k_i = 1000$.

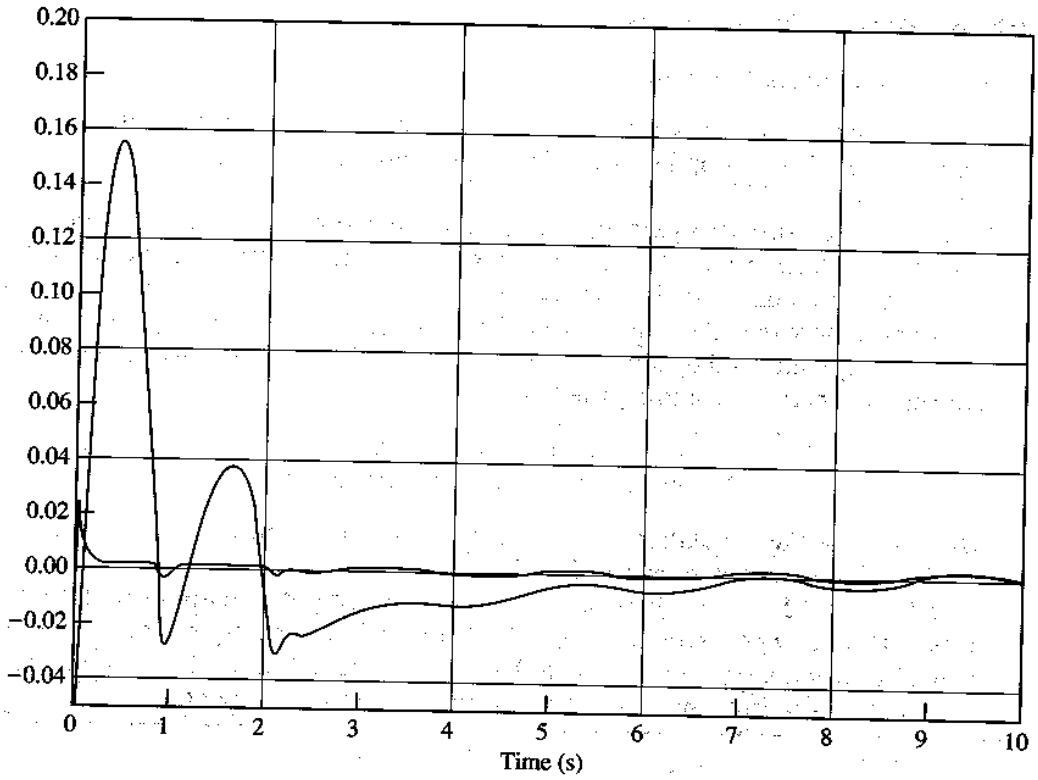


FIGURE 3.4-16 PID classical joint control with torque limits of ±35 N-m. Tracking error in rads.

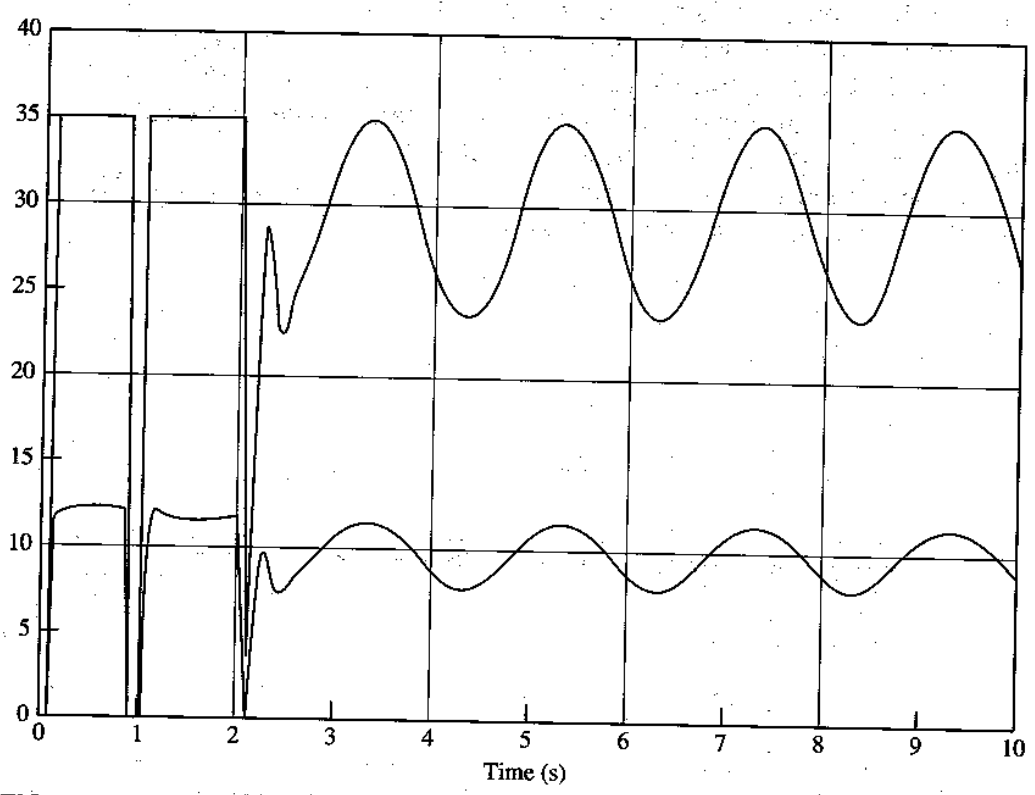


FIGURE 3.4-17 PID classical joint control with torque limits of ±35 N-m. Torque inputs in N-m.

c. Actuator Saturation Limits

We discussed integrator windup due to actuator saturation limits earlier in the section "PID Outer-Loop Design." Here we should like to demonstrate its deleterious effects.

The control torque in part (b) of this example is similar to the torque in Fig. 3.4-14c; it is fairly well behaved, except for some frantic action near time zero, where the maximum positive excursion is 78 N-m.

Suppose now that there are torque limits of $t_{max} = 35$ N-m, $t_{min} = -35$N-m. This is easily simulated by modifying subroutine CTL(x) in Fig. 3.4-2 to include a saturation function by adding the lines

```
IF(abs(t1) .gt.tmax) t1 = sign(tmax,t1)
IF(abs(t2) .gt.tmax) t2 = sign(tmax,t2)          (4)
```

The results of the simulation with these limits are shown in Fig. 3.4-16 and are terrible. The torque is shown in Fig. 3.4-17.

In Section 3.5 we show how to ameliorate the effects of actuator saturation by using *antiwindup protection* in a digital controller. The simulation results for PD control with actuator limits are not as bad as the ones just shown for PID control, since saturation limits have less effect if no integrator is present.