```
      M12= m2*a2**2 + m2*a1*a2*cos(x(2))
      M22= m2*a2**2
      N1= -m2*a1*a2*(2*x(3)*x(4) + x(4)**2) * sin(x(2))
      N1= N1 + (m1+m2)*g*a1*cos(x(1)) + m2*g*a2*cos(x(1)+x(2))
      N2= m2*a1*a2*x(3)**2*sin(x(2))  + m2*g*a2*cos(x(1)+x(2))

C   COMPUTATION OF CONTROL TORQUES

      s1= qdpp(1) + kv*ep(1) + kp*e(1)
      s2= qdpp(2) + kv*ep(2) + kp*e(2)
      t1= M11*s1 + M12*s2 + N1
      t2= M12*s1 + M22*s2 + N2
C   The next lines are to plot the torque
      tp1= t1
      tp2= t2

      RETURN
      END
C
C
C ********************************************************
C   ROBOT ARM DYNAMICS SUBROUTINE
      SUBROUTINE ARM(x,xp)
      REAL x(*),xp(*),m1,m2,M11,M12,M22,MI11,MI12,MI22,N1,N2
      COMMON/CONTROL/t1, t2
C  The next line is to plot the Cartesian position
      COMMON/OUTPUT/ dum(6), y1, y2
      DATA m1,m2,a1,a2, g/1.,1.,1.,1., 9.8/

C   COMPUTATION OF M(q)
      M11= (m1+m2)*a1**2 + m2*a2**2 + 2*m2*a1*a2*cos(x(2))
      M12=  m2*a2**2 + m2*a1*a2*cos(x(2))
      M22=  m2*a2**2
C   INVERSION OF M(q) (For large n, use least-squares to find qpp)
      det= M11*M22 - M12**2
      MI11= M22/det
      MI12= -M12/det
      MI22= M11/det
C   NONLINEAR TERMS
      N1= -m2*a1*a2*(2*x(3)*x(4) + x(4)**2) * sin(x(2))
      N1= N1 + (m1+m2)*g*a1*cos(x(1)) + m2*g*a2*cos(x(1)+x(2))
      N2= m2*a1*a2*x(3)**2*sin(x(2))  + m2*g*a2*cos(x(1)+x(2))

C   STATE EQUATIONS
      xp(1)= x(3)
      xp(2)= x(4)
      xp(3)= MI11*(-N1 + t1) + MI12*(-N2 + t2)
      xp(4)= MI12*(-N1 + t1) + MI22*(-N2 + t2)

C  OUTPUT EQUATION - CARTESIAN POSITION

      y1= a1*cos(x(1)) + a2*cos(x(1)+x(2))
      y2= a1*sin(x(1)) + a2*sin(x(1)+x(2))

      RETURN
      END
```

FIGURE 3.4-2 Subroutines for simulation using TRESP.

Subroutine F(time,x,xp) is called by Runge–Kutta and contains the continuous dynamics. This includes both the controller and the arm dynamics. The state to be integrated is $x = [\theta_1 \quad \theta_2 \quad \theta_1 \quad \theta_2]^T$, and the subroutine should compute the state derivative $\dot{x}$ (i.e., xp, which signifies x-prime).

Subroutine CTL(x) contains the controller (3). Note the structure of this subroutine. First, the tracking error $e(t)$ and its derivative are computed. Then $M(q)$ and $N(q,\dot{q}) = V(q,\dot{q}) + G(q)$ are computed. Finally, (3) is manufactured.

Subroutine ARM(x,xp) contains the robot dynamics. First, $M(q)$ and $M^{-1}(q)$ are computed, and then $N(q,\dot{q})$. The state derivatives are then determined.

The results of the simulation are shown in the figures. Figure 3.4-3 shows the joint angles. Figure 3.4-4 shows the joint errors. The initial conditions result in a large initial error that vanishes within 0.6 s. Figure 3.4-5 shows the control torques; the larger torque corresponds to the inner motor, which must move two links.

It is interesting to note the ripples in $e(t)$ that appear in Fig. 3.4-4. These are artifacts of the integrator. The Runge–Kutta integration period was $T_R = 0.01$ s. When the simulation was repeated using $T_R = 0.001$ s, the tracking error was exactly zero after 0.6 s. It should be zero, since computed-torque, or inverse dynamics control, is a scheme for canceling the nonlinearities in the dynamics to yield a second-order linear error system. If all the arm parameters are exactly known, this cancellation is exact. It is a good exercise to repeat this simulation using various values for the PD gains (see the Problems).

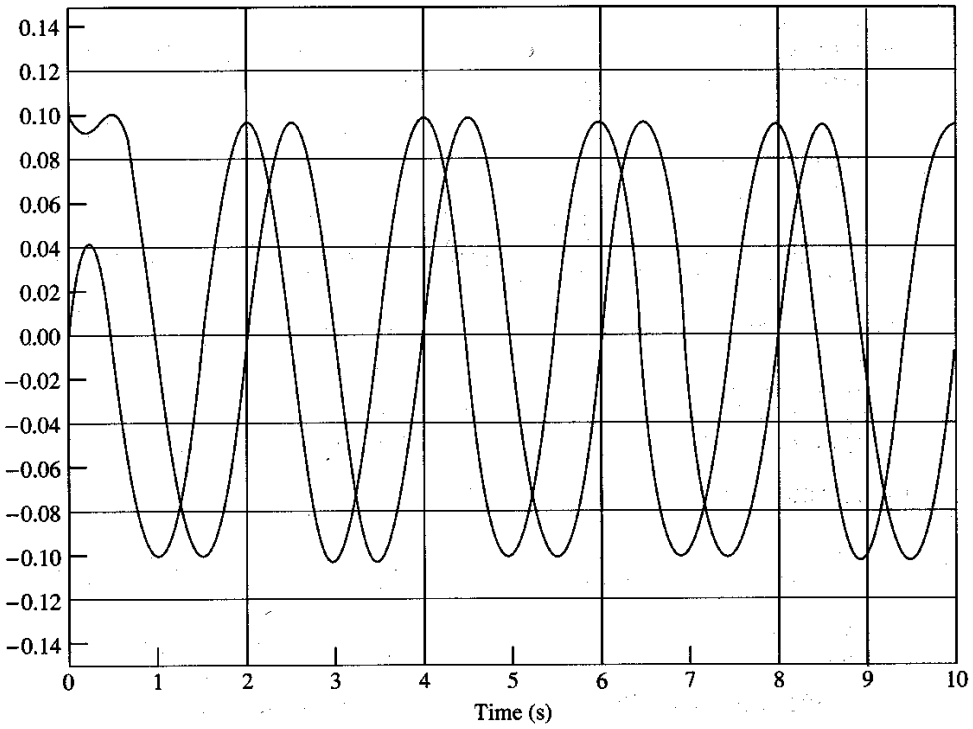


FIGURE 3.4-3 Joint angles $\theta_1(t)$ and $\theta_2(t)$ (rad).

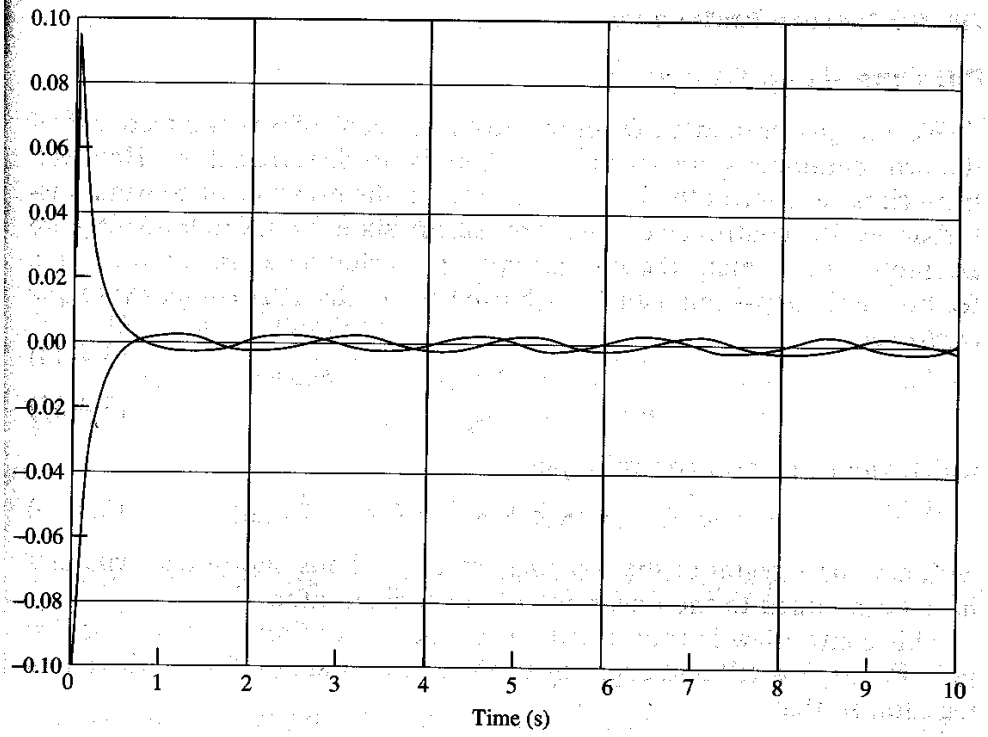


FIGURE 3.4-4 Tracking error $e_1(t)$, $e_2(t)$ (rad).

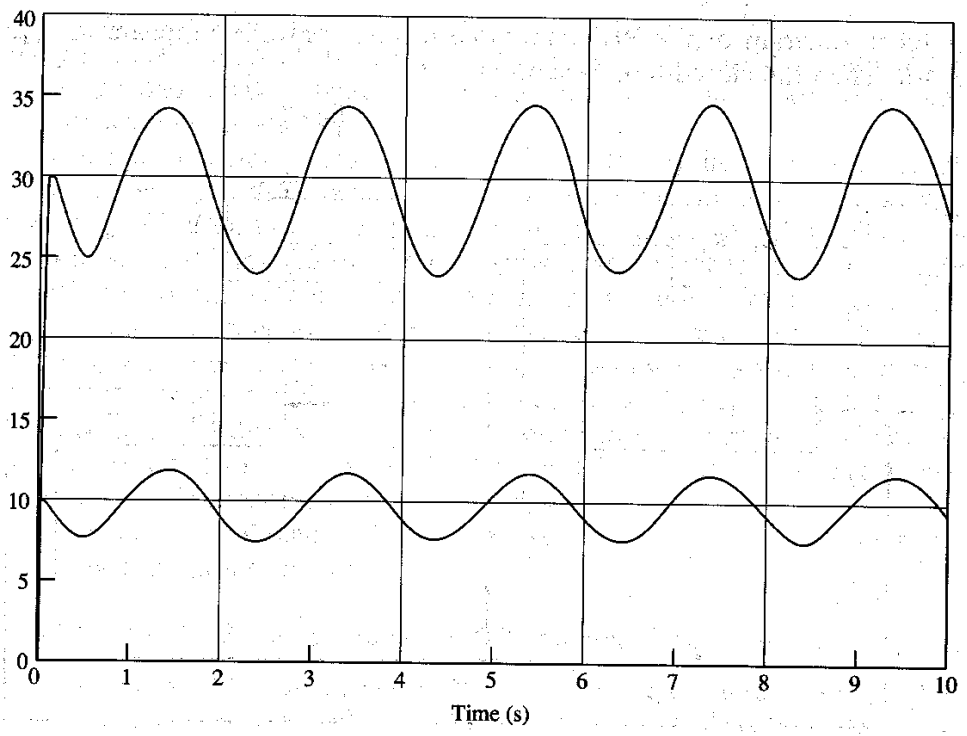


FIGURE 3.4-5 Torque inputs (N-m).

PID Outer-Loop Design

We have just seen that PD computed-torque control is very effective if all the arm parameters are known and there is no disturbance τ_d. However, from classical control theory we know that in the presence of constant disturbances, PD control gives a nonzero steady-state error. Consequently, we are motivated to make the system type I by including an integrator in the feedforward loop—this can be achieved using the *PID computed-torque controller*

$$\dot{\varepsilon} = e \tag{3.4-34}$$

$$u = -K_v\dot{e} - K_p e - K_i\varepsilon, \tag{3.4-35}$$

which yields the arm control input

$$\tau = M(q)\,(\ddot{q}_d + K_v\dot{e} + K_p e + K_i\varepsilon) + N(q,\dot{q}), \tag{3.4-36}$$

with $\varepsilon(t)$ the integral of the tracking error $e(t)$. Thus additional dynamics have been added to the linear outer-loop compensator.

This control law is conveniently described by defining the state as $x = [\varepsilon^T \quad e^T \quad \dot{e}^T]^T \in \mathbb{R}^{3n}$ and augmenting the error dynamics (3.4-8) with an integrator, so that

$$\frac{d}{dt}\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix} = \begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ I\end{bmatrix}u + \begin{bmatrix}0\\ 0\\ I\end{bmatrix}w. \tag{3.4-37}$$

A block diagram of the PID computed-torque controller appears in Fig. 3.4-6. Then the closed-loop system is

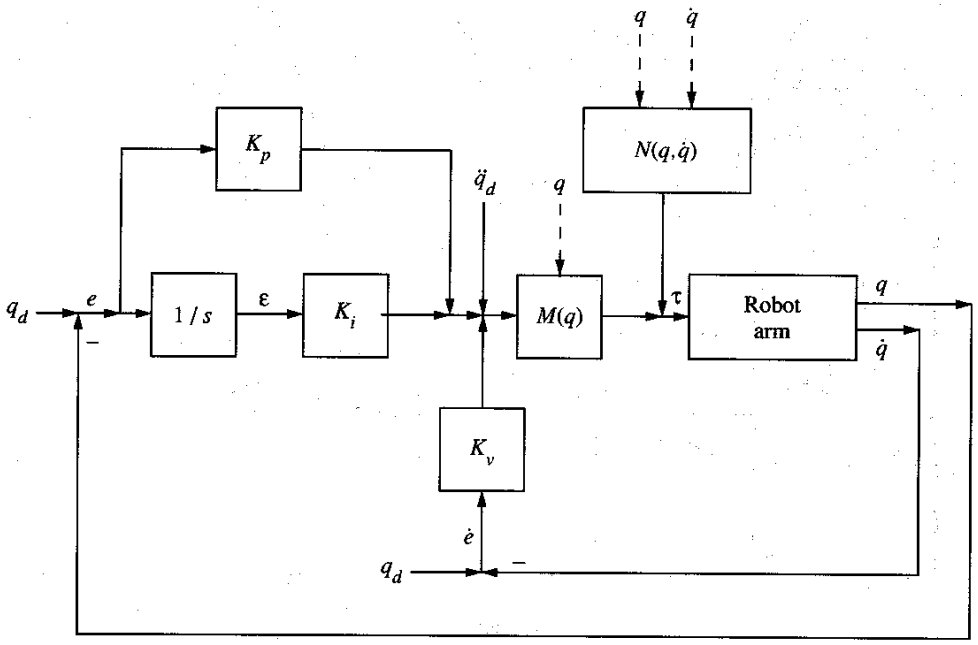


FIGURE 3.4-6 PID computed-torque controller.

$$\frac{d}{dt}\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix}=\begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ -K_i & -K_p & -K_v\end{bmatrix}\begin{bmatrix}\varepsilon\\ e\\ \dot{e}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ I\end{bmatrix}w. \tag{3.4-38}$$

The closed-loop characteristic polynomial is

$$\Delta c(s) = |s^3 I + K_v s^2 + K_p s + K_i|. \tag{3.4-39}$$

Selecting diagonal control gains

$$K_v = \text{diag}\{k_{v_i}\}, \qquad K_p = \text{diag}\{k_{p_i}\}, \qquad K_i = \text{diag}\{k_{i_i}\} \tag{3.4-40}$$

gives

$$\Delta_c(s) = \prod_{i=1}^{n} (s^3 + k_{v_i} s^2 + k_{p_i} s + k_{i_i}). \tag{3.4-41}$$

By using the Routh test it can be found that for closed-loop stability we require that

$$k_{i_i} < k_{v_i} k_{p_i}; \tag{3.4-42}$$

that is, the integral gain should not be too large.

Actuator Saturation and Integrator Windup. It is important to be aware of an effect in implementing PID control on any actual robot manipulator that can cause serious problems if not accounted for. Any real robot arm will have limits on the voltages and torques of its actuators. These limits may or may not cause a problem with PD control, but are virtually guaranteed to cause problems with integral control due to a phenomenon known as *integrator windup* [Lewis 1992].

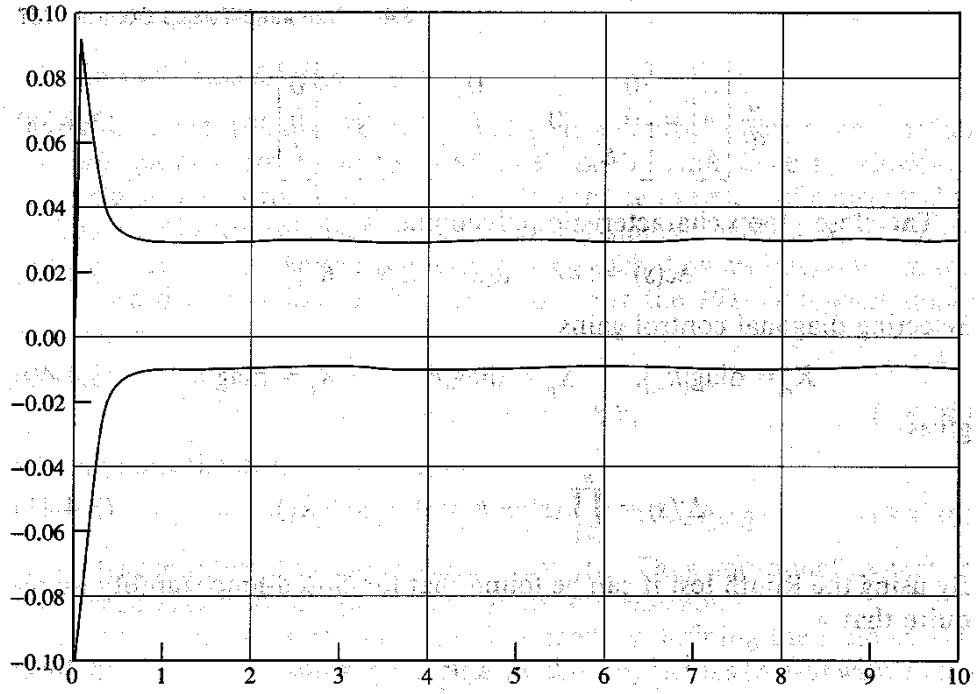
Consider the simple case where $\tau = k_i \varepsilon$, with $\varepsilon(t)$ the integrator output. The torque input $\tau(t)$ is limited by its maximum and minimum values τ_{max} and τ_{min}. If $k_i \varepsilon(t)$ hits τ_{max}, there may or not may not be a problem. The problem arises if the integrator input remains positive, for then the integrator continues to integrate upwards and $k_i \varepsilon(t)$ may increase well beyond τ_{max}. Then, when the integrator input becomes negative, it may take considerable time for $k_i \varepsilon(t)$ to decrease below τ_{max}. In the meantime τ is held at τ_{max}, giving an incorrect control input to the plant.

Integrator windup is easy to correct using *antiwindup protection* in a digital controller. This is discussed in Section 3.5. The effects of uncorrected windup are demonstrated in Example 3.4-4.

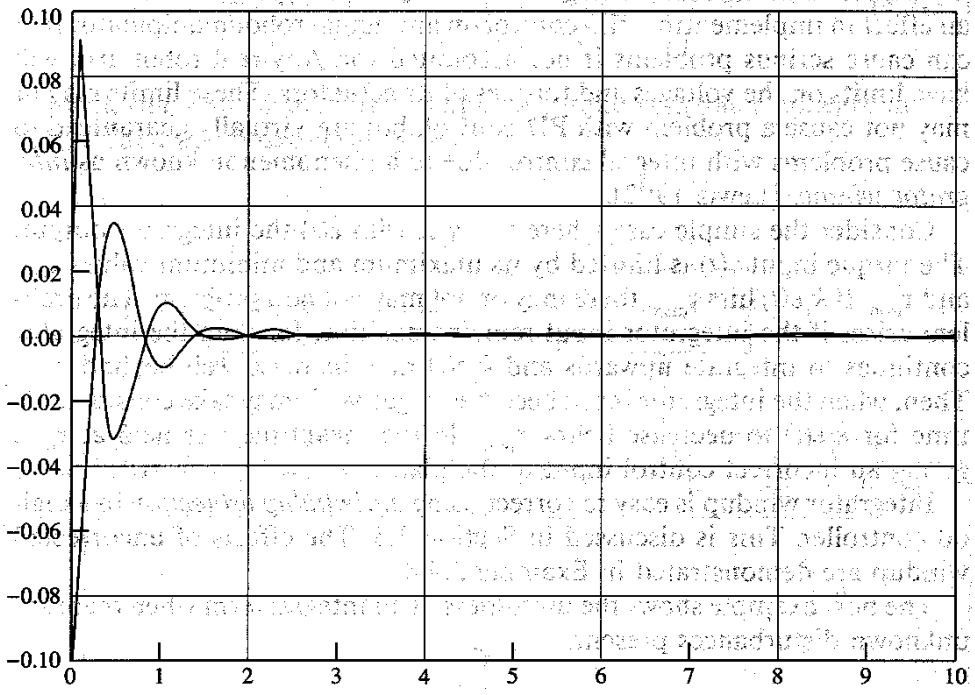
The next example shows the usefulness of an integral term when there are unknown disturbances present.

EXAMPLE 3.4-2: Simulation of PID Computed-Torque Control

In Example 3.4-1 we simulated the PD computed-torque controller for a two-link planar arm. In this example we add a constant unknown disturbance to the arm dynamics and compare PD to PID computed torque.

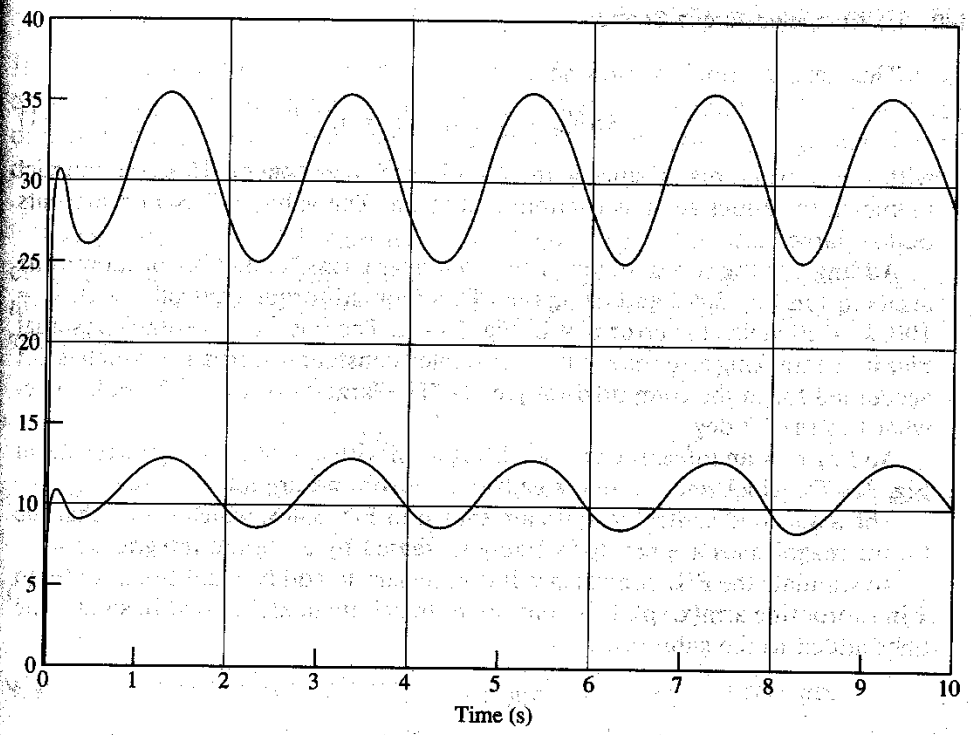


(a)

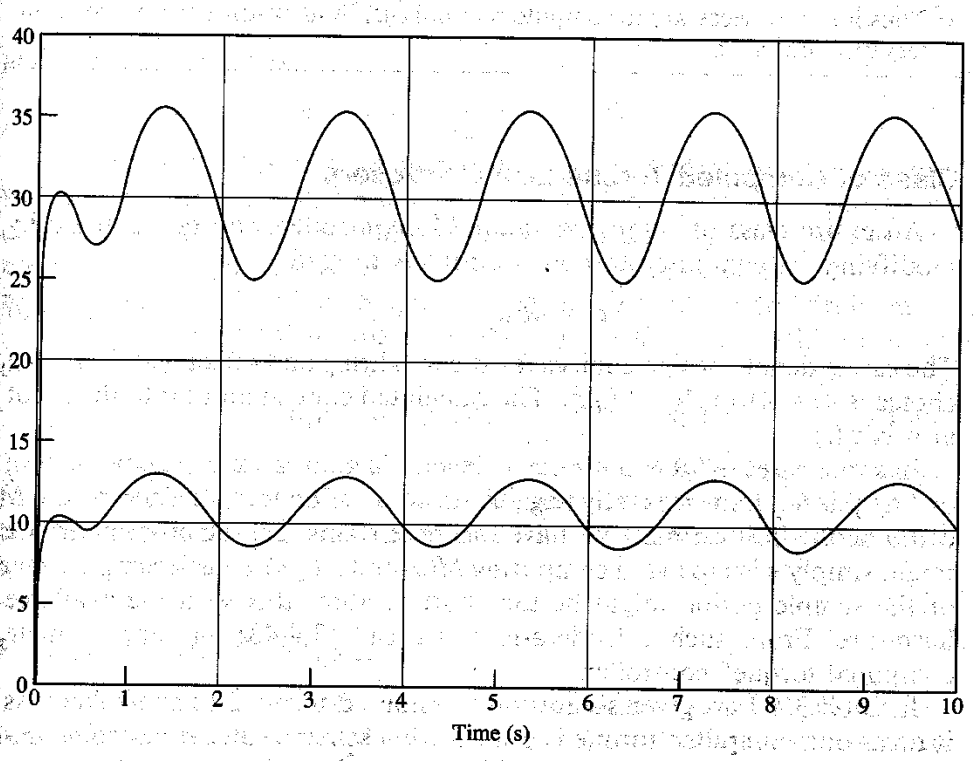


(b)

FIGURE 3.4-7 Computed-torque controller tracking errors $e_1(t)$, $e_2(t)$ (rad): (a) PD control; (b) PID control.



(a)



(b)

FIGURE 3.4-8 Computed-torque controller torque inputs (N-m): (a) PD control; (b) PID control.

Thus let the arm dynamics be

$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau, \tag{1}$$

with τ_d a constant disturbance with 1N-m in each component. This could model unknown dynamics such as friction, and so on. The value of 1 N-m represents quite a large bias.

Adding 1 to the computation of the nonlinear terms N1 and N2 in subroutine arm(x,xp) in Fig. 3.4-2 and using the PD computed-torque controller with $k_p = 100$, $k_v = 20$ yields the error plot in Fig. 3.4-7a. There is a unacceptable residual bias in the tracking error due to the unmodeled constant disturbance, which is not accounted for in the computed-torque law. The largest error is 0.033 rad, somewhat less than 2 deg.

Adding now an integral-error weighting term with $k_i = 500$ yields the results in Fig. 3.4-7b, which show a zero steady-state error and are quite good.

The associated control torques are shown in Fig. 3.4-8, which shows that the torque magnitudes are not appreciably increased by using the integral term.

To simulate the PID control law, it is necessary to add two additional states to x in subroutine arm(x,xp). It is convenient to call them x(5) and x(6), so that the lines added to the subroutine are

```
xp(5) = e(1).
xp(6) = e(2).
```

(2)

Thus it is now necessary to compute e(1) and e(2) in subroutine arm(x,xp) for integration purposes.