3.5 Digital Robot Control

Many robot control schemes are complicated and involve a great deal of computation for the evaluation of nonlinear terms. Therefore, they are implemented as digital control laws on digital signal processors (DSPs). Certain sorts of digital controllers for robot arms can be considered as members of the computer-torque-like class (see Table 3.4-1).

One approach to digital robot control is shown in Fig. 3.5-1. There, a sampler is placed on $q(t)$ and $\dot{q}(t)$ to define

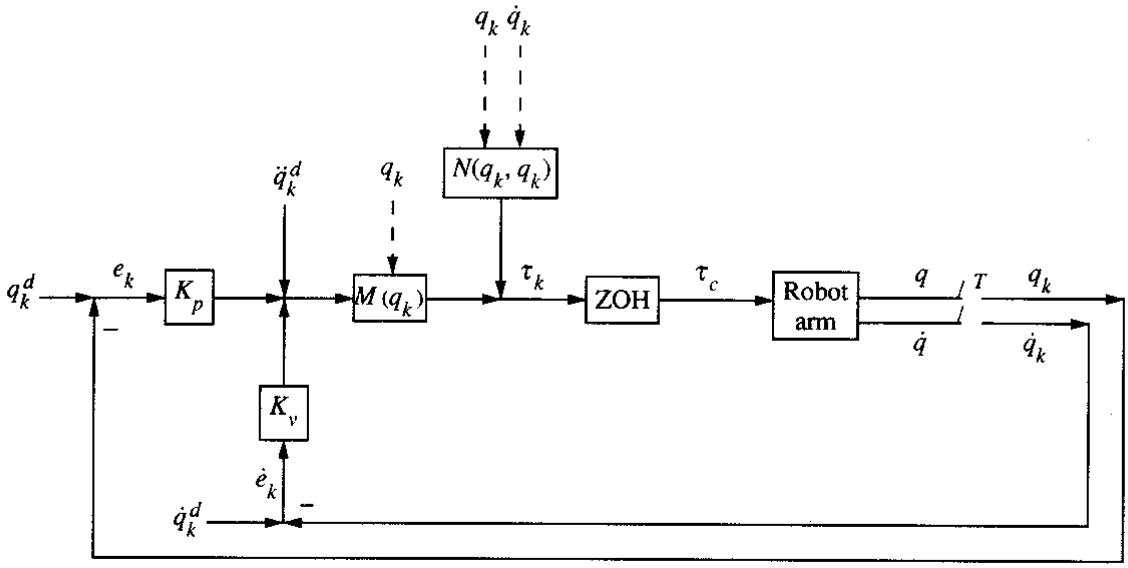


FIGURE 3.5-1 Digital robot control scheme.

$$\begin{aligned} q_k &\equiv q(kT) \in \mathbb{R}^n \\ \dot{q}_k &\equiv \dot{q}(kT) \in \mathbb{R}^n, \end{aligned} \tag{3.5-1}$$

with T the sample period. Sample periods in robotic applications vary from about 1 to 20 ms. A zero-order hold is used to reconstruct the continuous-time control input $\tau(t)$ needed for the actuators from the samples τ_k. In this section we use superscripts "d" for the desired trajectory for notational ease.

This digital control law amounts to selecting

$$\hat{M} = M(q(kT)), \qquad \hat{N} = N(q(kT),\dot{q}(kT)) \tag{3.5-2}$$

in the approximate computed-torque controller in Table 3.4-1 and a digital outer loop control signal u_k. Then, with PD outer-loop control, the arm control input is

$$\tau_k = M(q_k)\,(\ddot{q}_k^d + K_v\dot{e}_k + K_p e_k) + N(q_k,\dot{q}_k), \tag{3.5-3}$$

where the tracking error is $e(t) = q_d(t) - q(t)$. There are many variations of this control scheme. For instance, it is very reasonable to use multirate sampling and update $\hat{M}$ and $\hat{N}$ less frequently than the sampling frequency. That is, the *the inner nonlinear loop can be sampled more slowly than the outer linear feedback loop.* In view of the robustness properties of computed-torque control, this works quite well in practice.

Guaranteed Performance on Sampling

It is usual in robot controls to design the controller in continuous time, providing rigorous proofs of stability and error boundedness. However, when the controller is implemented, a "small" sample period is selected and the stability is left to chance and verified by simulation studies. That this "wishful thinking" approach may not be so unreasonable is suggested by the next theorem. Define $\underline{e} = [\underline{e}^T \quad \dot{e}^T]^T$.

THEOREM 3.5-1: *Suppose that the PD digital control law* (3.5-3) *is applied to the robot arm. Then for every $L > 0$ there exists a T_M such that for all sampling periods T less than T_M, each error trajectory which at some time t_0 satisfies $\| \underline{e}(t_0) \| \leq L$ has $\| \underline{e}(t) \| \leq L + r$ for all $t \geq t_0$ and any $r > 0$.*

Proof:

Using the digital control law yields the error system (Table 3.4-1)

$$\ddot{e} = u - \Delta u + d, \tag{1}$$

where $d = M^{-1}\tau_d + \Delta\ddot{q}_d + \delta$ and

$$\Delta = M^{-1}(q)[M(q) - M(q_k)], \qquad \delta = M^{-1}(q)[N(q,\dot{q}) - N(q_k,\dot{q}_k)]. \tag{2}$$

Defining $\underline{e} = [e^T \quad \dot{e}^T]^T$ this may be written in state-space form as

$$\dot{\underline{e}} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \underline{e} + \begin{bmatrix} 0 \\ I \end{bmatrix} (u - \Delta u + d) \equiv A\underline{e} + B(u - \Delta u + d). \tag{3}$$

Applying now the outer-loop digital control

$$u = -K\underline{e}_k = -K\underline{e} + K(\underline{e} - \underline{e}_k) \tag{4}$$

yields the closed-loop system

$$\dot{\underline{e}} = (A - BK)\underline{e} + BK(\underline{e} - \underline{e}_k) + B(\Delta K\underline{e}_k + d). \tag{5}$$

The feedback K is selected so that (A − BK) is stable; hence

$$(A - BK)^T P + P(A - BK) = -Q \tag{6}$$

for some positive definite P and Q.

Selecting now the Lyapunov function

$$V(\underline{e}) = \tfrac{1}{2}\underline{e}^T P\underline{e}, \tag{7}$$

(5) and (6) reveal

$$\dot{V}(\underline{e}) = \tfrac{1}{2}\underline{e}^T Q\underline{e} + \underline{e}^T PB[K(\underline{e} - \underline{e}_k) + \mathbf{d} + \Delta K\underline{e}_k]. \tag{8}$$

Assuming for simplicity that the disturbance τ_d is zero, note that

$$\dot{V}(\underline{e}_k) = \tfrac{1}{2}\underline{e}_k^T Q\underline{e}_k \tag{9}$$

is negative definite at each sample time. [If τ_d is nonzero but bounded, then $V(e_k)$ is negative outside some ball, and the discussion may be modified.]

The remainder of the proof follows from a theorem of LaSalle and Lefschetz [1961] by showing that:

1. $\dot{V}(\underline{e}) \leq 0$ *when* $\| \underline{e} \| > L$.
2. $V(\underline{e}(t_1) < V(\underline{e}(t_2)$ *for all* $t_2 \geq t_1 \geq 0$ *when* $\|\underline{e}(t_1)\| \leq L$ *and* $\|\underline{e}(t_2)\| > L + r$.

At issue is the fact (9) and the continuity of $\dot{V}(\underline{e})$. ■

This result shows that good tracking is obtained for all sample periods less than some *maximum sample period* T_M which depends on the specific robot arm. A more refined result can show that the errors increase as the maximum desired acceleration $\ddot{q}_d$ increases, or equivalently, that smaller sample periods are required for larger desired accelerations.

We now discuss some issues in discretizing the inner nonlinear loop and then the outer linear loop.

Discretization of Inner Nonlinear Loop

There is no convenient exact way to discretize nonlinear dynamics. Given a nonlinear state-space system

$$\dot{x} = f(x,u), \tag{3.5-4}$$

Euler's approximation yields

$$x_{k+1} = x_k + Tf(x_k,u_k). \tag{3.5-5}$$

There do exist "exact" techniques for deriving discrete nonlinear robot

dynamics. They rely on discretizing the robot arm dynamics in such a way that energy and momentum are conserved at each sampling instant [Neuman and Tourassis 1985]. See also [Elliott 1990]. Unfortunately, these schemes result in extremely complicated discrete dynamical equations, even for simple robot arms. It is very difficult to derive guaranteed digital control laws for them.

In this section we simply take the discretized inner nonlinear loop as given by the approximation (3.5-2).

Joint Velocity Estimates from Position Measurements

Throughout this chapter in the examples we have simulated continuous-time robot controllers assuming that both the joint positions and velocities are measured exactly. In point of fact, it is usual to measure the joint velocities using optical encoders, and then estimate the joint velocities from these position measurements. Simply computing the joint velocities using the Euler approximation

$$\dot{q}_k = (q_k - q_{k-1})/T \tag{3.5-6}$$

is virtually doomed to failure, since this high-pass filter amplifies the encoder measurement noise.

Denote the joint velocity estimates by v_k. Then a filtered derivative can be used to compute v_k from q_k using

$$v_k = \nu v_{k-1} + (q_k - q_{k-1})/T, \tag{3.5-7}$$

where ν is a design parameter. If ν is small, it corresponds to a fast pole near $z = 0$, which provides some high-pass filtering to reject unwanted sensor noise. Example 3.5-1 will illustrate the use of this joint velocity estimation filter.

The velocity estimation filter design can be optimized for the given encoder noise statistics by using an alpha-beta tracker to reconstruct v_k [Lewis 1986a,b, Lowe and Lewis 1991]. This is a specialized form of Kalman filter. It should be noted that the velocity estimates are not only used in the outer linear loop for computing $\dot{e}_k$; they must be used to compute the inner nonlinear terms in (3.5-3) as well.

Discretization of Outer PD/PID Control Loop

We have seen that a useful computed-torque outer feedback loop is the PID controller. Given a continuous-time PID controller, a digital PID controller for the outer loop may be designed as follows [Lewis 1992].

A continuous PID controller that only uses joint position measurements $q(t)$ is given by

$$\begin{aligned} u &= -K^c(s)e \\ K^c(s) &= k\left[1 + \frac{1}{T_I s} + \frac{T_D s}{1 + T_D s/N}\right], \end{aligned} \tag{3.5-8}$$

where k is the proportional gain, T_I is the integration time constant or "reset" time, T_D is the derivative time constant. Rather than use pure differentiation, a "filtered derivative" is used which has a pole far left in the s-plane at $s = -N/T_D$. The value for N is often in the range 3 to 10; it is usually fixed by the manufacturer of the controller [Åström and Wittenmark 1984]. A special case of the PID controller, of course, is the PD controller, which is therefore also covered by this discussion.

A common approximate discretization technique for converting continuous-time controllers $K^c(s)$ to digital controllers $K(z)$ is the bilinear transform (BLT), where

$$K(z) = K^c(s') \tag{3.5-9}$$

$$s' = \frac{2}{T}\frac{z-1}{z+1}. \tag{3.5-10}$$

This corresponds to approximating integration by the trapezoidal rule. Under this mapping, stable continuous systems with poles at s are mapped into stable discrete systems with poles at

$$z = \frac{1 + sT/2}{1 - sT/2}. \tag{3.5-11}$$

The finite zeros also map according to this transformation. However, the zeros at infinity in the s-plane map into zeros at $z = -1$.

Using the BLT to discretize (3.5-8) yields

$$K(z) = k\left[1 + \frac{T}{T_{Id}}\frac{z+1}{z-1} + \frac{T_{Dd}}{T}\frac{z-1}{z-\nu}\right] \tag{3.5-12}$$

with the discrete integral and derivative time constants

$$T_{Id} = 2T_I \tag{3.5-13}$$

$$T_{Dd} = \frac{NT}{1 + NT/2T_D} \tag{3.5-14}$$

and the derivative-filtering pole at

$$\nu = \frac{1 - NT/2T_D}{1 + NT/2T_D}. \tag{3.5-15}$$

It is easy to implement this digital outer-loop filter in terms of difference equations on a DSP. First, write $K(z)$ in terms of z^{-1}, which is the unit delay in the time domain (i.e., a delay of T seconds), as

$$K(z^{-1}) = k\left[1 + \frac{T}{T_{Id}}\frac{1+z^{-1}}{1-z^{-1}} + \frac{T_{Dd}}{T}\frac{1-z^{-1}}{1-\nu z^{-1}}\right]. \tag{3.5-16}$$

[*Note:* There is some abuse in notation in denoting (3.5-16) as $K(z^{-1})$; this we shall accept.]

Now suppose that the control input u_k is related to the tracking error as

$$u_k = -K(z^{-1})e_k. \tag{3.5-17}$$

Then u_k may be computed from past and present values of e_k using auxiliary variables as follows:

$$v_k^I = v_{k-1}^I + (T/T_{Id})\,(e_k + e_{k-1}) \tag{3.5-18}$$

$$v_k^D = \nu v_{k-1}^D + (T_{Dd}/T)\,(e_k - e_{k-1}) \tag{3.5-19}$$

$$-u_k = k(e_k + v_k^I + v_k^D). \tag{3.5-20}$$

The variables v_k^I and v_k^D represent the integral and derivative portions of the digital PID controller, respectively. These difference equations are easily implemented in software.

Actuator Saturation and Integrator Antiwindup Compensation

Actuator saturation leading to *integrator windup* is a problem that can occur in the outer PID loop of a robot controller. In Example 3.4-4 we saw the deleterious effects of integrator windup. It is easy to implement *antiwindup protection* digitally [Åström and Wittenmark 1984, Lewis 1992]. The antiwindup protection circuit would be placed into the outer linear feedback loop of a robot control system, where the integrator (controller memory) is located.

Suppose that the controller is given in transfer function form

$$R(z^{-1})v_k = T(z^{-1})r_k - S(z^{-1})w_k, \tag{3.5-21}$$

where r_k is the reference command and w_k is the controller input, composed generally of the tracking error and the plant measured output. Note that z^{-1} is interpreted in the time domain as a unit delay of T seconds. The controller output v_k is passed through a hold device to generate the continuous plant control input $u(t)$.

We have assumed thus far that the desired plant control input $v_k \in \mathbb{R}^m$ computed by the controller can actually be applied to the plant. However, in practical systems the plant inputs (such as motor voltages, etc.) are limited by *maximum* and *minimum* allowable values. Thus the relation between the *desired plant input* v_k and the *actual plant input* u_k is given by the sort of behavior shown in Fig. 3.5-2, where u_H and u_L represent, respectively, the maximum and minimum control effort allowed by the mechanical actuator. If there are no control limits, we may set $u_k = v_k$.

Thus, to describe the actual case in a practical control system with actuator saturation, we are forced to include *nonlinear saturation functions* in the control channels as shown in Fig. 3.5-3. Consider the simple case where the controller is an integrator with input w_k and output v_k. Then all is well as long as v_k is between u_L and u_H, for in this region the plant input u_k equals v_k. However, if v_k exceeds u_H, then u_k is limited to its maximum value u_H. This in itself may not be a problem. The problem arises if w_k remains positive, for

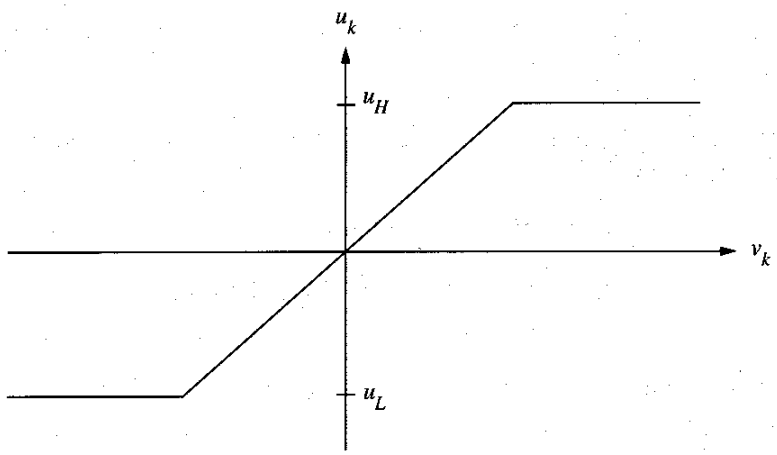


FIGURE 3.5-2 Actuator saturation function.

then the integrator continues to integrate and v_k may increase well beyond u_H. Then, when w_k becomes negative, it may take considerable time for v_k to decrease below u_H. In the meantime, u_k is held at u_H, giving an incorrect control input to the plant. This effect of integrator saturation is called *windup*. It arises because the controllers we design are generally dynamical in nature, which means that they store information or energy. Windup occurs when $R(z)$ is not a stable polynomial.

To correct integrator windup, it is necessary to *limit the state of the controller* so that it is consistent with the saturation effects being experienced by the plant input u_k. This may be accomplished as follows. Select a desired stable observer polynomial $A_0(z)$ and add $A_0(z^{-1})u_k$ to both sides to obtain

$$A_0 u_k = Tr_k - Sw_k + (A_0 - R)u_k. \tag{3.5-22}$$

A regulator with antiwindup compensation is then given by

$$A_0 v_k = Tr_k - Sw_k + (A_0 - R)u_k. \tag{3.5-23}$$

$$u_k = \text{sat}(v_k). \tag{3.5-24}$$

A special case occurs when $A_0(z)$ has all n of its poles at the origin. Then the regulator displays *deadbeat behavior;* after n time steps its state remains limited to an easily computed value dependent on the values of w_k and u_k (see the Problems).

The next example demonstrates some aspects of digital control in robotics.

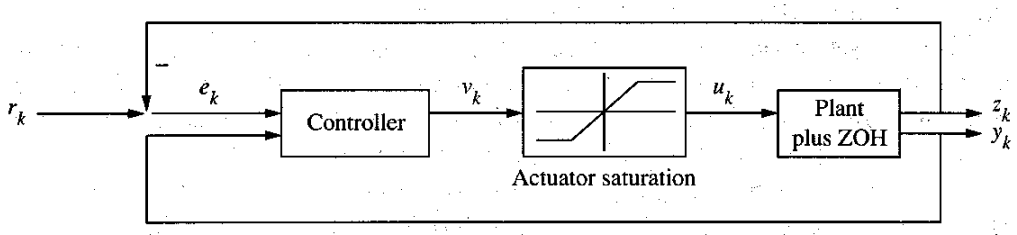


FIGURE 3.5-3 Control system including actuator saturation.

EXAMPLE 3.5-1: Simulation of Digital Robot Computed-Torque Controller

In Example 3.4-1 we simulated a continuous-time PD computed-torque (CT) controller for the two-link planar elbow arm. In this example some issues in

```
C   DIGITAL COMPUTED-TORQUE CONTROLLER FOR 2-LINK PLANAR ARM
C   SUBROUTINES FOR USE WITH TRESP

C   DIGITAL COMPUTED-TORQUE CONTROLLER SUBROUTINE
      SUBROUTINE DIG(IK,TD,x)
      REAL x(*),m1,m2,M11,M12,M22,N1,N2,kp,kv
      COMMON/CONTROL/t1, t2
C   The next line is to plot the errors and torques
      COMMON/OUTPUT/ e(2), ep(2), tp1, tp2
      COMMON/TRAJ/qd(6), qdp(6), qdpp(6)
      DATA m1,m2,a1,a2, g/1.,1.,1.,1., 9.8/
      DATA kp, kv/ 100,20/

C   COMPUTE TRACKING ERRORS
      e(1) = qd(1)  - x(1)
      e(2) = qd(2)  - x(2)
      ep(1)= qdp(1) - x(3)
      ep(2)= qdp(2) - x(4)

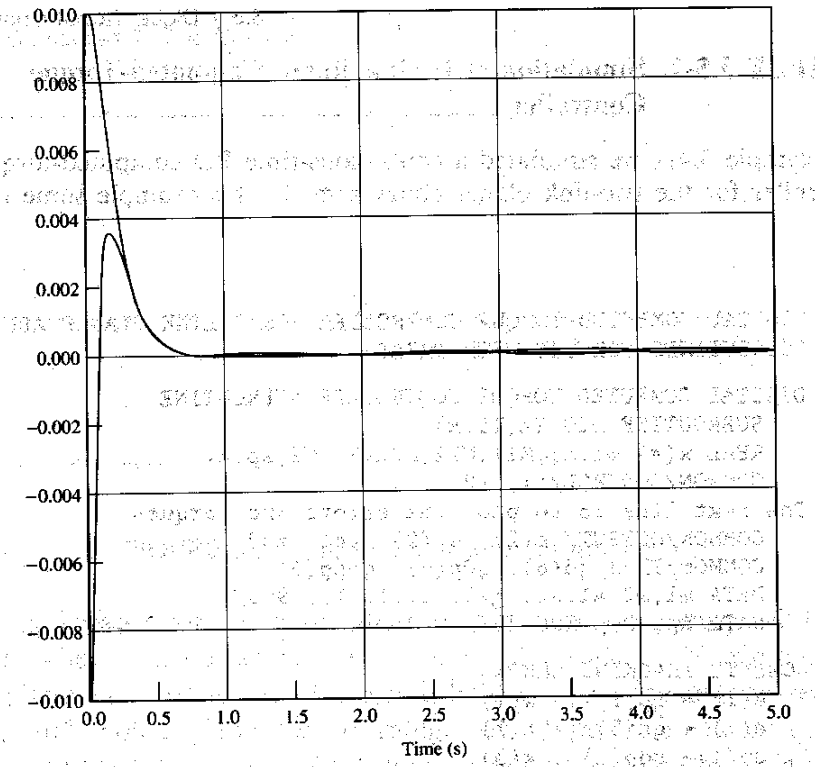
C   COMPUTATION OF M(q), N(q,qp)
      M11= (m1+m2)*a1**2 + m2*a2**2 + 2*m2*a1*a2*cos(x(2))
      M12=  m2*a2**2 + m2*a1*a2*cos(x(2))
      M22=  m2*a2**2
      N1= -m2*a1*a2*(2*x(3)*x(4) + x(4)**2) * sin(x(2))
      N1= N1 + (m1+m2)*g*a1*cos(x(1)) + m2*g*a2*cos(x(1)+x(2))
      N2= m2*a1*a2*x(3)**2*sin(x(2))  + 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   CONTINUOUS SUBROUTINE CALLED BY RUNGE-KUTTA INTEGRATOR
      SUBROUTINE F(t,x,xp)
      REAL x(*), xp(*)
C
C   ROBOT ARM DYNAMICS
      CALL ARM(x,xp)
C
      RETURN
      END
```

FIGURE 3.5-4 Subroutines for use with TRESP for digital CT control.



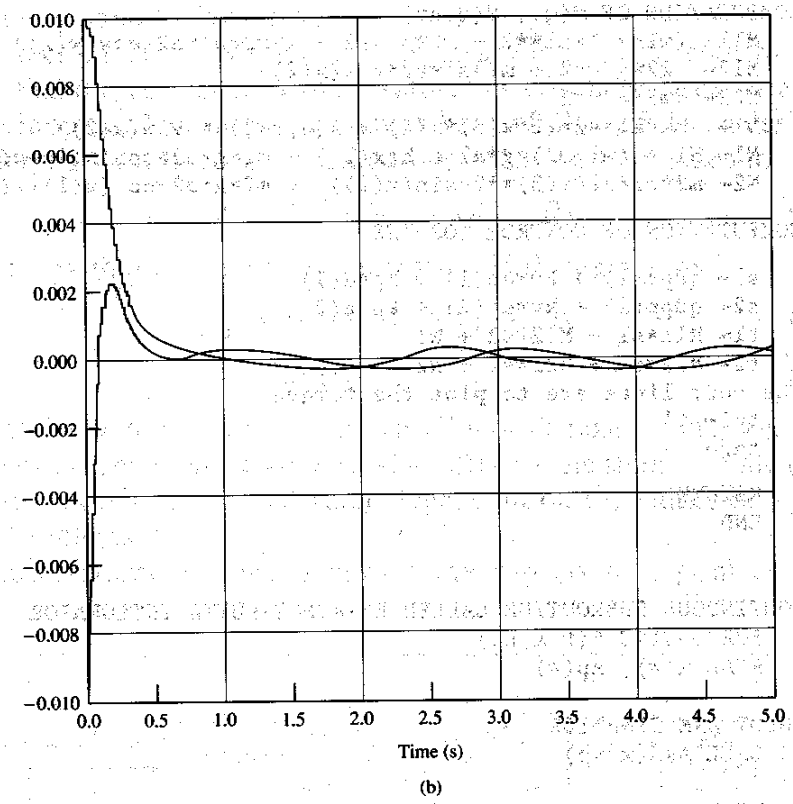
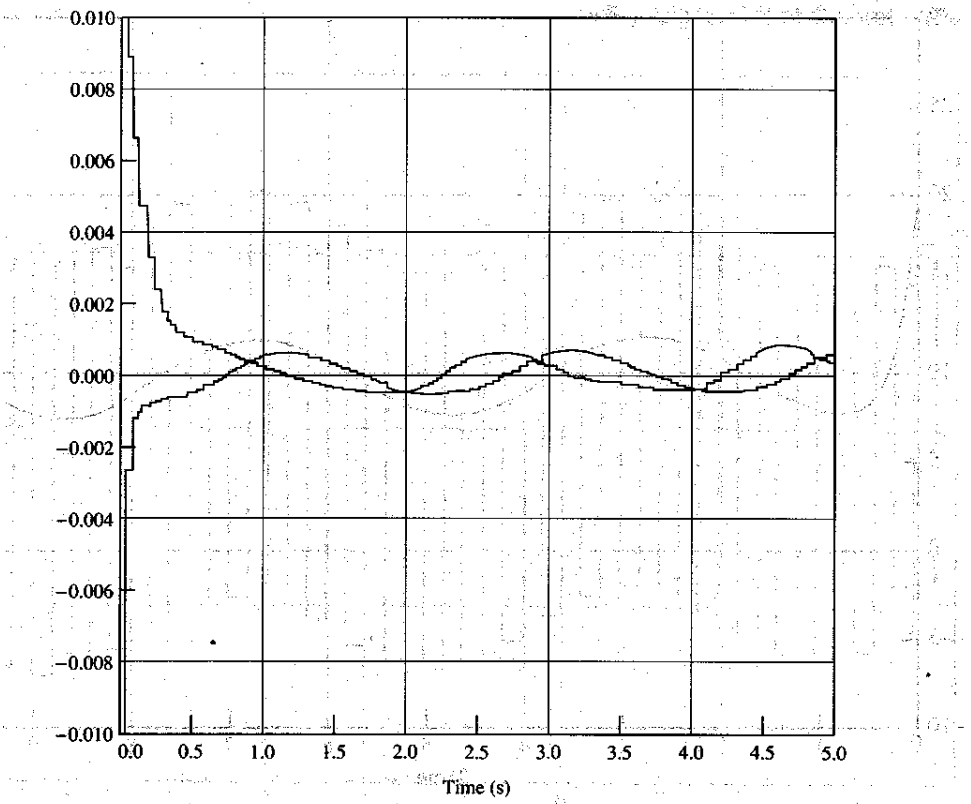
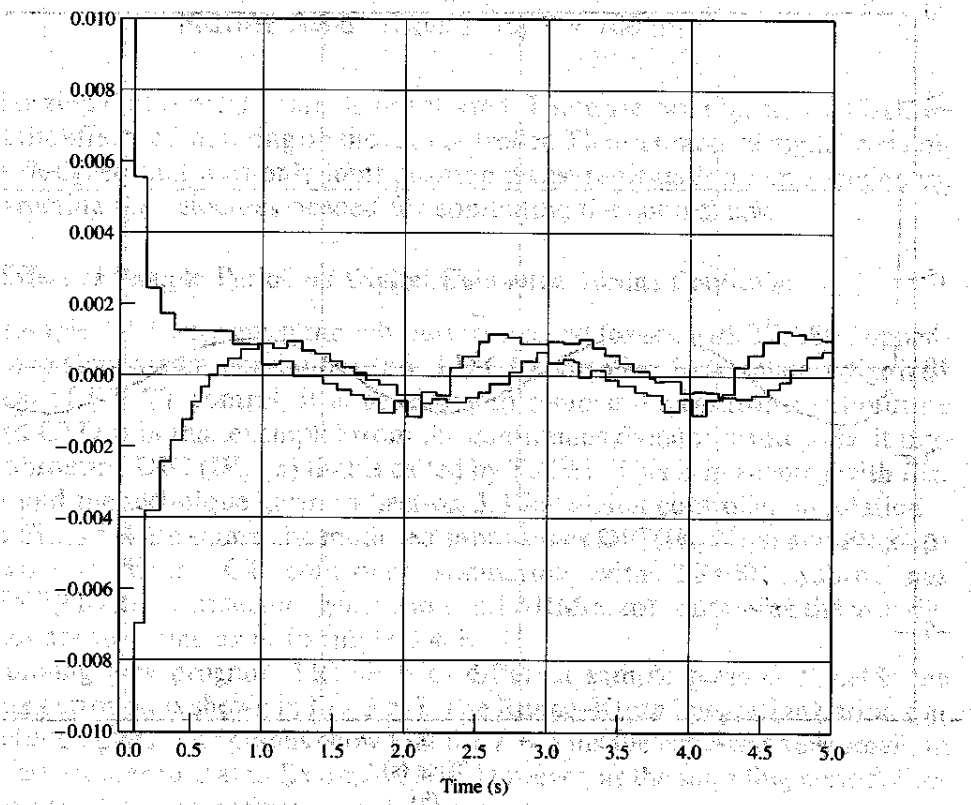


FIGURE 3.5-5 Joint tracking error on sampling the CT controller: (a) $T = 5$ ms; (b) $T = 20$ ms.



(c)



(d)

FIGURE 3.5-5 *(Cont.)* (c) $T = 50$ ms; (d) $T = 100$ ms.