3.2 Path Generation

Throughout the book we assume that there is given a prescribed path $q_d(t)$ the robot arm should follow. We design control schemes that make the manipulator follow this desired path or trajectory. *Trajectory planning* involves finding the prescribed path and is usually considered a separate design problem involving *collision avoidance,* concerns about *actuator saturation,* and so on. See [Lee et al. 1983].

We do not cover trajectory planning. However, we do cover two aspects of trajectory generation. First, we show how to convert a given prescribed path from Cartesian space to joint space. Then, given a table of desired points the end effector should pass through, we show how to reconstruct a continuous desired trajectory.

Converting Cartesian Trajectories to Joint Space

In robotic applications, a desired task is usually specified in the workspace or Cartesian space, as this is where the motion of the manipulator is easily described in relation to the external environment and workpiece. However, trajectory-following control is easily performed in the joint space, as this is where the arm dynamics are more easily formulated.

Therefore, it is important to be able to find the desired joint space trajectory $q_d(t)$ given the desired Cartesian trajectory. This is accomplished using the *inverse kinematics,* as shown in the next example. The example illustrates that the mapping of Cartesian to joint space trajectories may not be unique—that is, several joint space trajectories may yield the same Cartesian trajectory for the end-effector.

EXAMPLE 3.2-1: Mapping a Prescribed Cartesian Trajectory to Joint Space

In Example A.2-5 are derived the inverse kinematics for the two-link planar robot arm shown in Fig. 3.2-1. Let us use them to convert a path from Cartesian space to joint space.

Suppose that we want the two-link arm to follow a given workspace or Cartesian trajectory

$$p(t) = (x(t),y(t)) \tag{1}$$

in the (x,y) plane which is a function of time t. Since the arm is moved by actuators that control its angles θ_1, θ_2, it is convenient to convert the specified Cartesian trajectory $(x(t),y(t))$ into a *joint space trajectory* $(\theta_1(t),\theta_2(t))$ for control purposes.

This may be achieved by using the inverse kinematics transformations

$$r^2 = x^2 + y^2 \tag{2}$$

$$C = \cos\theta_2 = \frac{r^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{3}$$

$$D = \pm\sqrt{1 - \cos^2\theta_2} = \pm\sqrt{1 - C^2} \tag{4}$$

$$\theta_2 = \text{ATAN2}\,(D,C) \tag{5}$$

$$\theta_1 = \text{ATAN2}\,(y,x) - \text{ATAN2}\,(a_2\sin\theta_2,\ a_1 + a_2\cos\theta_2). \tag{6}$$

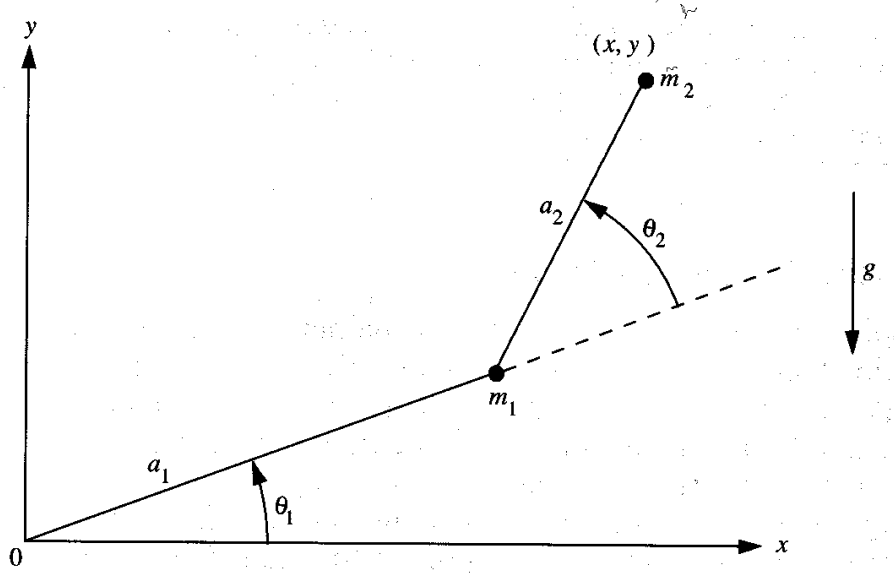


FIGURE 3.2-1 Two-link planar elbow arm.

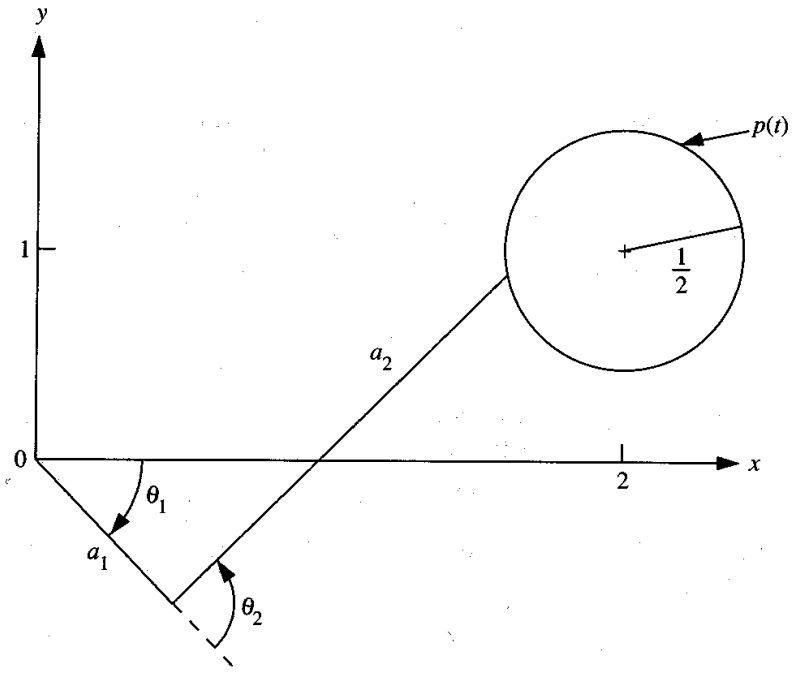


FIGURE 3.2-2 Desired Cartesian trajectory.

Suppose that the end of the arm should repeatedly trace out the circular workspace path $p(t)$ shown in Fig. 3.2-2, which is described by

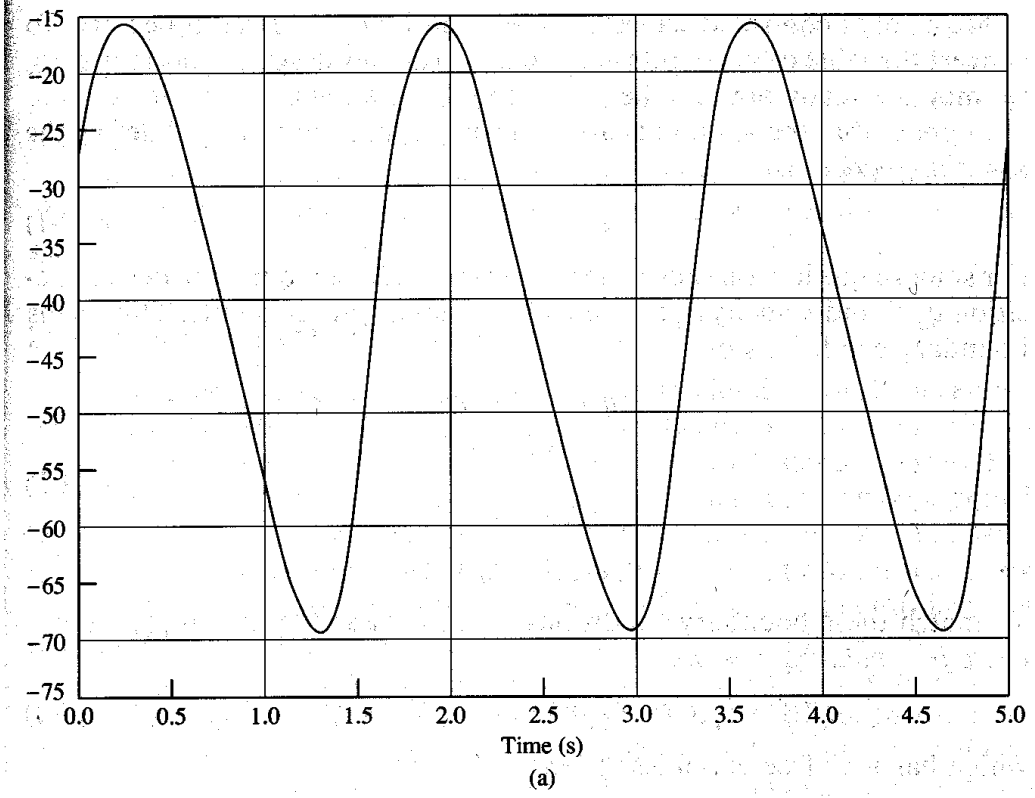
$$\begin{aligned} x(t) &= 2 + \tfrac{1}{2}\cos t \\ y(t) &= 1 + \tfrac{1}{2}\sin t. \end{aligned} \tag{7}$$

By using these expressions for each time t in the inverse kinematics equations, we obtain the required joint-space trajectories $q(t) = (\theta_1(t),\theta_2(t))$ given in Fig. 3.2-3 that yield the circular Cartesian motion of the end effector (using $a_1 = 2$, $a_2 = 2$).

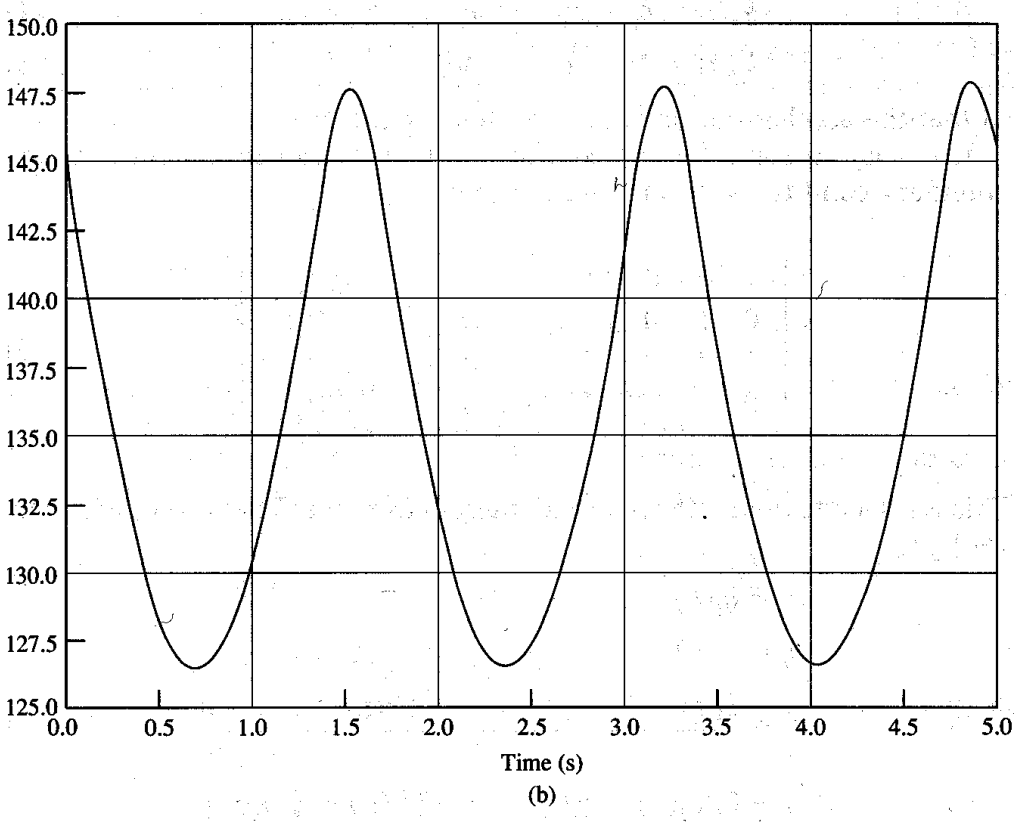
We have computed the joint variables for the "elbow down" configuration. Selecting the opposite sign in (4) gives the "elbow up" joint space trajectory yielding the same Cartesian trajectory.

Polynomial Path Interpolation

Suppose that a desired trajectory for the manipulator motion has been determined, either in Cartesian space or, using the inverse kinematics, in joint space. For convenience, we use the joint space variable $q(t)$ for notation. It is not possible to store the entire trajectory in computer memory, and few practically useful trajectories have a simple closed-form expression. Therefore, it is usual to store in computer memory a sequence of points $q_i(t_k)$ for each joint variable i that represent the desired values of that variable at the discrete times t_k. Thus $q(t_k)$ is a point in $\mathbf{R}^n$ that the joint variables should pass through at time t_k. We call these *via points.*



(a)



(b)

FIGURE 3.2-3 Required joint-space trajectories: (a) θ_1 (deg); (b) θ_2 (deg).

Most robot control schemes require a continuous desired trajectory. To convert the table of via points $q_i(t_k)$ to a continuous desired trajectory $q_d(t)$, we may use many options. Let us discuss here *polynomial interpolation.*

Suppose that the via points are uniformly spaced in time and define the *sampling period* as

$$T = t_{k+1} - t_k. \tag{3.2-1}$$

For smooth motion, on each time interval $[t_k, t_{k+1}]$ we require the desired position $q_d(t)$ and velocity $\dot{q}_d(t)$ to match the tabulated via points. This yields boundary conditions of

$$\begin{aligned} q_{d_i}(t_k) &= q_i(t_k) \\ \dot{q}_{d_i}(t_k) &= \dot{q}_i(t_k) \\ q_{d_i}(t_{k+1}) &= q_i(t_{k+1}) \\ \dot{q}_{d_i}(t_{k+1}) &= \dot{q}_i(t_{k+1}). \end{aligned} \tag{3.2-2}$$

To match these boundary conditions, it is necessary to use on $[t_k, t_{k+1}]$ the *cubic interpolating polynomial*

$$q_{d_i}(t) = a_i + (t - t_k)b_i + (t - t_k)^2 c_i + (t - t_k)^3 d_i, \tag{3.2-3}$$

which has four free variables. Then

$$\dot{q}_{d_i}(t) = b_i + 2(t - t_k)c_i + 3(t - t_k)^2 d_i \tag{3.2-4}$$

$$\ddot{q}_{d_i}(t) = 2c_i + 6(t - t_k)d_i \tag{3.2-5}$$

so that the acceleration is *linear* on each sample period.

It is easy to solve for the coefficients that guarantee matching of the boundary conditions. In fact, we see that

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} q_i(t_k) \\ \dot{q}_i(t_k) \\ q_i(t_{k+1}) \\ \dot{q}_i(t_{k+1}) \end{bmatrix}. \tag{3.2-6}$$

This is solved to obtain the required interpolating coefficients on each interval $[t_k, t_{k+1}]$

$$\begin{aligned} a_i &= q_i(t_k) \\ b_i &= \dot{q}_i(t_k) \\ c_i &= \frac{3[q_i(t_{k+1}) - q_i(t_k)] - T[2\dot{q}_i(t_k) + \dot{q}_i(t_{k+1})]}{T^2} \\ d_i &= \frac{2[(q_i(t_k) - q_i(t_{k+1})] + T[\dot{q}_i(t_k) + \dot{q}_i(t_{k+1})]}{T^3}. \end{aligned} \tag{3.2-7}$$

Note that this technique requires storing the desired position *and velocity* at each sampling point in tabular form. A variant uses a higher-order polynomial to ensure continuous position, velocity, *and acceleration* at each sample time t_k.

Although we have used the joint variable notation $q(t)$, it should be emphasized that trajectory interpolation can also be performed in Cartesian space.

Linear Function with Parabolic Blends

Using cubic interpolating polynomials, the acceleration on each sample period is linear. However, in many practical applications there are good reasons for insisting on *constant* accelerations within each sample period. For instance, any real robot has upper limits on the torques that can be supplied by its actuators. For linear systems (think of Newton's law) this translates into constant accelerations. Therefore, constant accelerations are less likely to saturate the actuators. Besides that, most industrial robot controllers are programmed to use constant accelerations on each sample period.

A constant acceleration profile is shown in Fig. 3.2-4a. The associated velocity and position profiles are shown in Fig. 3.2-4b and c. The position trajectory has three parts: a quadratic or parabolic initial portion, a linear midsection, and a parabolic final portion. Therefore, let us discuss interpolation of via points using *linear functions with parabolic blends* (LFPB).

The time at which the position trajectory switches from parabolic to linear is known as the *blend time* t_b. A position $q_{d_i}(t)$ *should be specified for each joint variable i.* The trajectory in Fig. 3.2-4c can be written for joint i as

$$q_{d_i}(t) = \begin{cases} a_i + (t - t_k)b_i + (t - t_k)^2 c_i, & t_k \le t < t_k + t_b \\ d_i + v_i t, & t_k + t_b \le t < t_{k+1} - t_b \\ e_i + (t - t_{k+1})f_i + (t - t_{k+1})^2 g_i, & t_{k+1} - t_b \le t \le t_{k+1}. \end{cases} \tag{3.2-8}$$

The coefficient v_i may be interpreted as the maximum velocity allowed for joint variable i. The design parameters are v_i and t_b.

It is straightforward to solve for the coefficients on each time interval $[t_k, t_{k+1}]$ that ensure satisfaction of the boundary conditions (3.2-2). The result is

$$\begin{aligned} a_i &= q_i(t_k), \qquad b_i = \dot{q}_i(t_k), \qquad c_i = \frac{v_i - \dot{q}_i(t_k)}{2t_b} \\ d_i &= \frac{q_i(t_k) + q_i(t_{k+1}) - v_i t_{k+1}}{2} \\ e_i &= q_i(t_{k+1}), \qquad f_i = \dot{q}_i(t_{k+1}) \\ g_i &= \frac{v_i t_{k+1} + q_i(t_k) - q_i(t_{k+1}) + 2t_b[(\dot{q}_i(t_{k+1}) - v_i]}{2t_b^2}. \end{aligned} \tag{3.2-9}$$

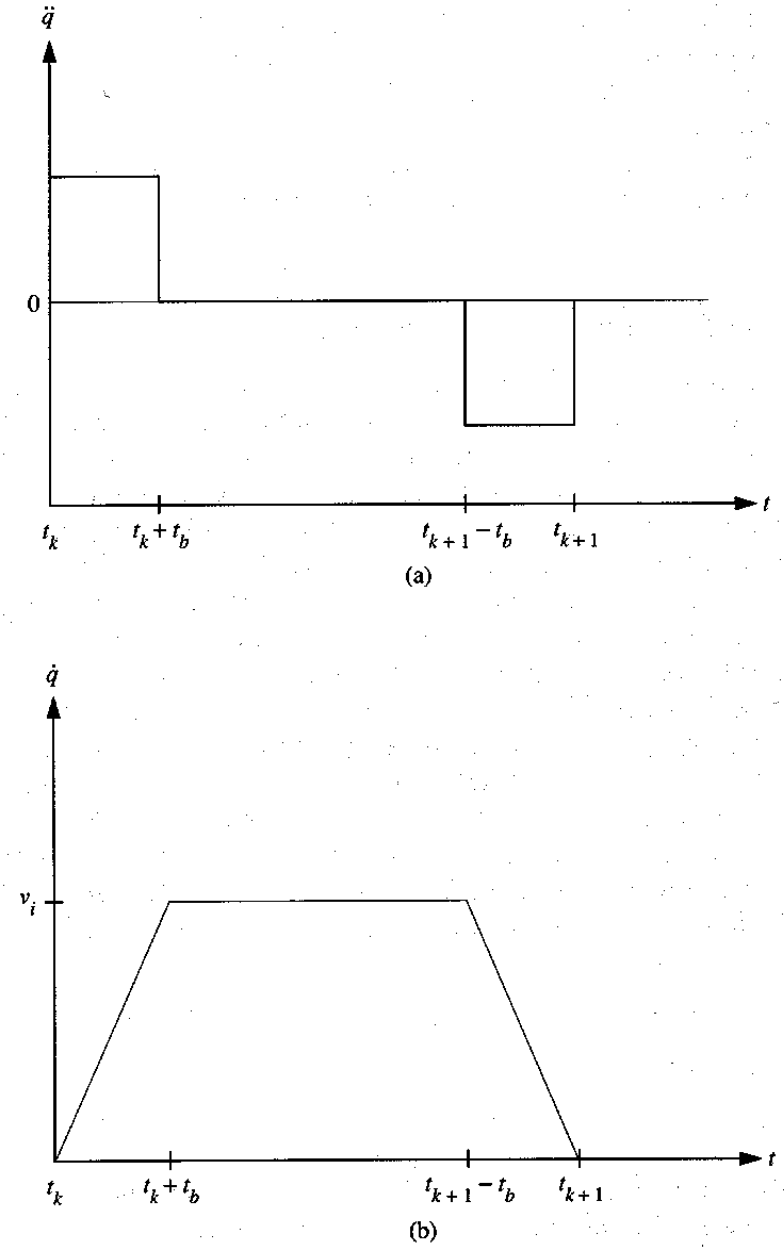
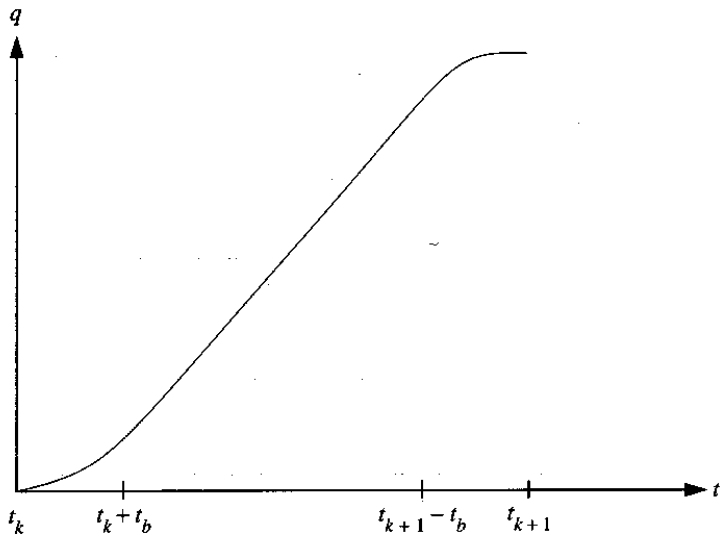


FIGURE 3.2-4 LFPB trajectory: (a) acceleration; (b) velocity.

Minimum-Time Trajectories

There is an important special class of LFPB trajectories. Suppose the acceleration is limited by a maximum value of a_M and it is desired for the robot arm to get from one position to another in *minimum time.* For simplicity assume that the initial and final velocities are equal to zero. The general case is covered in [Lewis 1986] (see the Problems).

A minimum-time trajectory is shown in Fig. 3.2-5. To drive joint variable i from a rest position of $q_0 \equiv q_i(t_0)$ to a desired final rest position of $q_f \equiv$



(c)

FIGURE 3.2-4 (*Cont.*) (c) position.

$q_i(t_f)$ in a minimum time t_f, the maximum acceleration a_M should be applied until the *switching time* t_s, after which time the maximum deceleration $-a_M$ should be applied until t_f. Note that both t_s and t_f depend on q_0 and q_f. We may write

$$q_i(t_s) = q_0 + \tfrac{1}{2}a_M(t_s - t_0)^2$$
$$\dot{q}_i(t_s) = a_M(t_s - t_0)$$
$$q_i(t_f) = q_i(t_s) + \dot{q}_i(t_s)\,(t_f - t_s) - \tfrac{1}{2}a_M(t_f - t_s)^2$$
$$\dot{q}_i(t_f) = \dot{q}_i(t_s) - a_M(t_f - t_s).$$

Then the velocity equations yield

$$\dot{q}_i(t_f) = a_M(t_s - t_0) - a_M(t_f - t_s) = 0$$

or

$$t_s = (t_f + t_0)/2. \tag{3.2-10}$$

That is, the switching from maximum acceleration to maximum deceleration occurs at the *half-time point*. Now simple manipulations on the position equations yield

$$q_i(t_f) = q_0 + \tfrac{1}{2}a_M(t_s - t_0)^2 + a_M(t_s - t_0)\,(t_f - t_s) - \tfrac{1}{2}a_M(t_f - t_s)^2 = q_f$$

$$\frac{q_f - q_0}{a_M} = \tfrac{1}{2}(t_s - t_0)^2 + (t_s - t_0)\,(t_f - t_s) - \tfrac{1}{2}(t_f - t_s)^2$$

whence (3.2-10) yields

$$t_s = t_0 + \sqrt{(q_f - q_0)/a_M}. \tag{3.2-11}$$

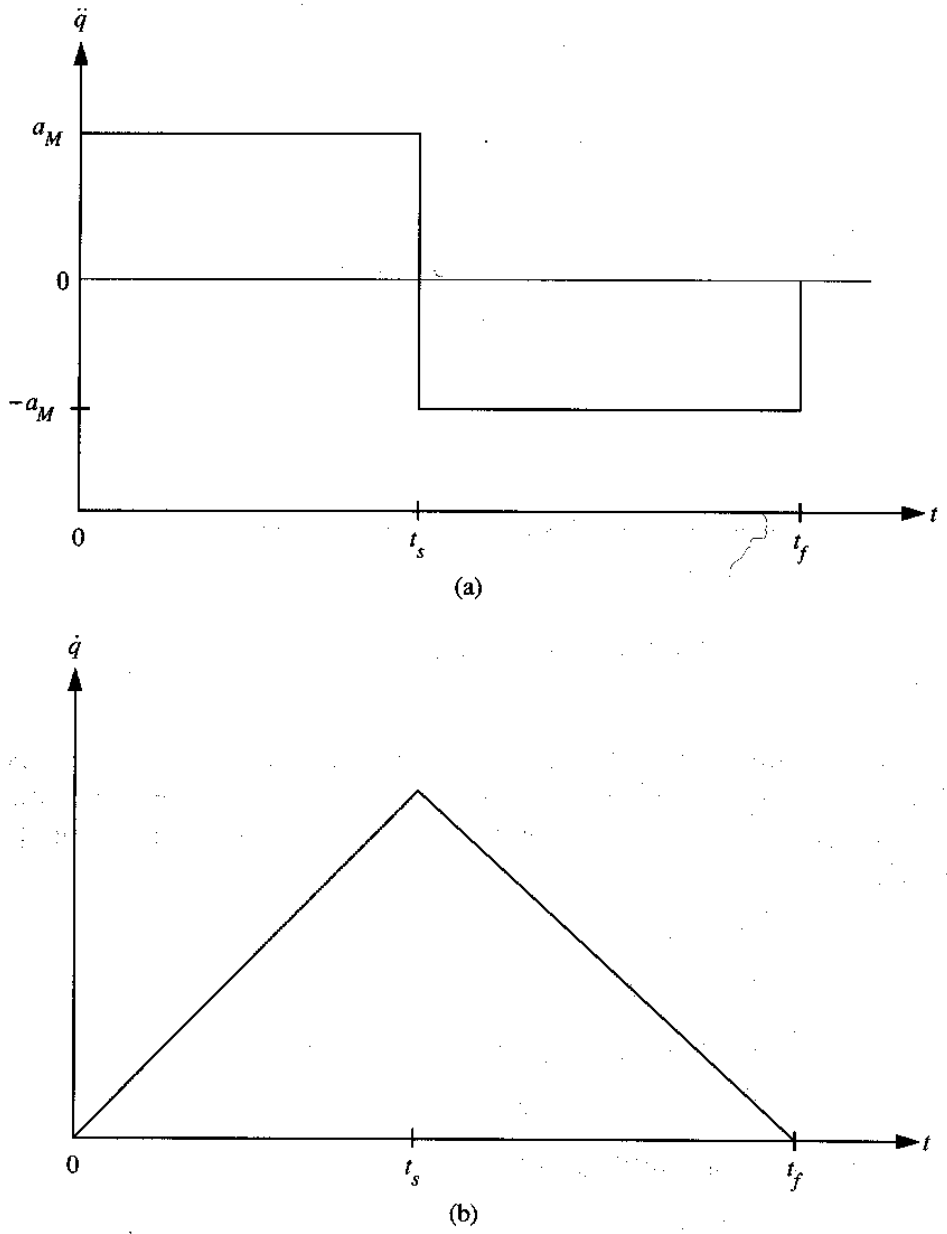


FIGURE 3.2-5 Minimum-time trajectory: (a) acceleration; (b) velocity.

A closed-loop formulation of minimum-time control appears in [Lewis 1986].

Unfortunately, minimum-time trajectories computed using a constant maximum acceleration are not directly relevant in robotics. This is due to the fact that an actual manipulator has a *torque limit* of τ_M. Since the robot equation (see Table 2.3-1) is nonlinear, this does not correlate to a constant upper bound on the acceleration. For instance, a robot arm has different maximum accelerations in its fully extended and fully retracted positions. For more discussion see [Chen 1989, Geering et al. 1986, Gourdeau and Schwartz 1989, Jayasuriya and Suh 1985, Kahn and Roth 1971, Kim and Shin 1985, Shin and McKay 1985.]

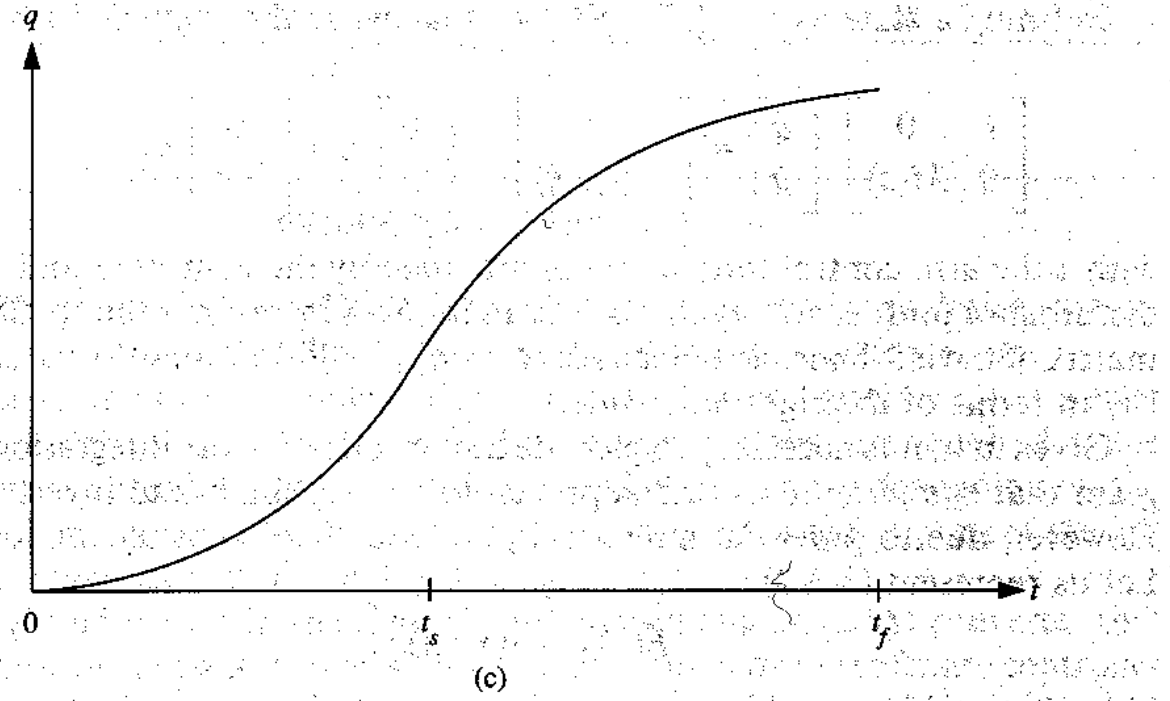


FIGURE 3.2-5 (*Cont.*) (c) position.