Chapter 9

Trajectory Generation

During robot motion, the robot controller is provided with a steady stream of goal positions and velocities to track. This specification of the robot position as a function of time is called a **trajectory**. In some cases, the trajectory is completely specified by the task – for example, the end-effector may be required to track a known moving object. In other cases, as when the task is simply to move from one position to another in a given time, we have freedom to design the trajectory to meet these constraints. This is the domain of **trajectory planning**. The trajectory should be a sufficiently smooth function of time, and it should respect any given limits on joint velocities, accelerations, or torques.

In this chapter we consider a trajectory as the combination of a **path**, a purely geometric description of the sequence of configurations achieved by the robot, and a **time scaling**, which specifies the times when those configurations are reached. We consider three cases: point-to-point straight-line trajectories in both joint space and task space; trajectories passing through a sequence of timed **via points**; and minimum-time trajectories along specified paths taking actuator limits into consideration. Finding paths that avoid obstacles is left to Chapter 10.

9.1 Definitions

A **path** $\theta(s)$ maps a scalar path parameter s, assumed to be 0 at the start of the path and 1 at the end, to a point in the robot's configuration space Θ, $\theta : [0,1] \to \Theta$. As s increases from 0 to 1, the robot moves along the path. Sometimes s is taken to be time and is allowed to vary from time $s = 0$ to the total motion time $s = T$, but it is often useful to separate the role of the geometric path parameter s from the time parameter t. A **time scaling** $s(t)$ assigns a value s to each time $t \in [0, T]$, $s : [0, T] \to [0, 1]$.

Together, a path and a time scaling define a **trajectory** $\theta(s(t))$, or $\theta(t)$ for short. Using the chain rule, the velocity and acceleration along the trajectory can be written as

$$\dot{\theta} = \frac{d\theta}{ds}\dot{s}, \tag{9.1}$$

$$\ddot{\theta} = \frac{d\theta}{ds}\ddot{s} + \frac{d^2\theta}{ds^2}\dot{s}^2. \tag{9.2}$$

To ensure that the robot's acceleration (and therefore dynamics) is well defined, each of $\theta(s)$ and $s(t)$ must be twice differentiable.

9.2 Point-to-Point Trajectories

The simplest type of motion is from rest at one configuration to rest at another. We call this a point-to-point motion. The simplest type of path for point-to-point motion is a straight line. Straight-line paths and their time scalings are discussed below.

9.2.1 Straight-Line Paths

A "straight line" from a start configuration θ_{start} to an end configuration θ_{end} could be defined in joint space or in task space. The advantage of a straight-line path from θ_{start} to θ_{end} in joint space is simplicity: since joint limits typically take the form $\theta_{i,\min} \leq \theta_i \leq \theta_{i,\max}$ for each joint i, the allowable joint configurations form a convex set Θ_{free} in joint space, so the straight line between any two endpoints in Θ_{free} also lies in Θ_{free}. The straight line can be written

$$\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}}), \qquad s \in [0, 1] \tag{9.3}$$

with derivatives

$$\frac{d\theta}{ds} = \theta_{\text{end}} - \theta_{\text{start}}, \tag{9.4}$$

$$\frac{d^2\theta}{ds^2} = 0. \tag{9.5}$$

Straight lines in joint space generally do not yield straight-line motion of the end-effector in task space. If task-space straight-line motions are desired, the start and end configurations can be specified by X_{start} and X_{end} in task space. If

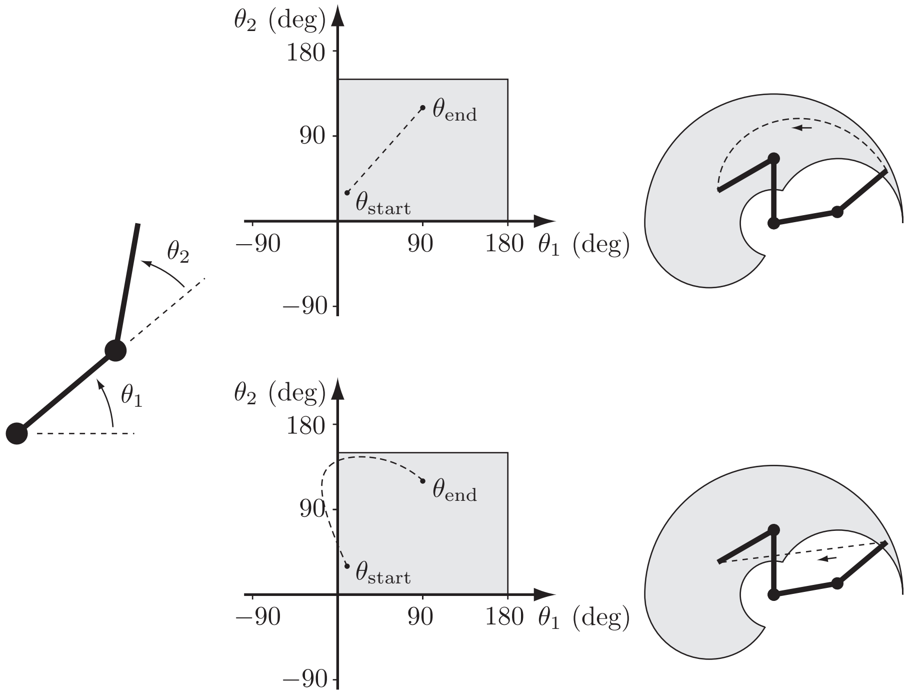

Figure 9.1: (Left) A 2R robot with joint limits $0^\circ \leq \theta_1 \leq 180^\circ$, $0^\circ \leq \theta_2 \leq 150^\circ$. (Top center) A straight-line path in joint space and (top right) the corresponding motion of the end-effector in task space (dashed line). The reachable endpoint configurations, subject to joint limits, are indicated in gray. (Bottom center) This curved line in joint space and (bottom right) the corresponding straight-line path in task space (dashed line) would violate the joint limits.

X_{start} and X_{end} are represented by a minimum set of coordinates then a straight line is defined as $X(s) = X_{\text{start}} + s(X_{\text{end}} - X_{\text{start}})$, $s \in [0, 1]$. Compared with the case when joint coordinates are used, the following issues must be addressed:

- If the path passes near a kinematic singularity, the joint velocities may become unreasonably large for almost all time scalings of the path.
- Since the robot's reachable task space may not be convex in X coordinates, some points on a straight line between two reachable endpoints may not be reachable (Figure 9.1).

In addition to the issues above, if X_{start} and X_{end} are represented as elements of $SE(3)$ instead of as a minimum set of coordinates, then there is the question of how to define a "straight" line in $SE(3)$. A configuration of the form $X_{\text{start}} + s(X_{\text{end}} - X_{\text{start}})$ does not generally lie in $SE(3)$.

One option is to use the screw motion (simultaneous rotation about and translation along a fixed screw axis) that moves the robot's end-effector from $X_{\text{start}} = X(0)$ to $X_{\text{end}} = X(1)$. To derive this $X(s)$, we can write the start and end configurations explicitly in the {s} frame as $X_{s,\text{start}}$ and $X_{s,\text{end}}$ and use our subscript cancellation rule to express the end configuration in the start frame:

$$X_{\text{start,end}} = X_{\text{start},s} X_{s,\text{end}} = X_{s,\text{start}}^{-1} X_{s,\text{end}}.$$

Then $\log(X_{s,\text{start}}^{-1} X_{s,\text{end}})$ is the matrix representation of the twist, expressed in the {start} frame, that takes X_{start} to X_{end} in unit time. The path can therefore be written as

$$X(s) = X_{\text{start}} \exp(\log(X_{\text{start}}^{-1} X_{\text{end}})s), \tag{9.6}$$

where X_{start} is post-multiplied by the matrix exponential since the twist is represented in the {start} frame, not the fixed world frame {s}.

This screw motion provides a "straight-line" motion in the sense that the screw axis is constant. The origin of the end-effector does not generally follow a straight line in Cartesian space, since it is following a screw motion. It may be preferable to decouple the rotational motion from the translational motion. Writing $X = (R, p)$, we can define the path

$$p(s) = p_{\text{start}} + s(p_{\text{end}} - p_{\text{start}}), \tag{9.7}$$

$$R(s) = R_{\text{start}} \exp(\log(R_{\text{start}}^{\mathrm{T}} R_{\text{end}})s) \tag{9.8}$$

where the frame origin follows a straight line while the axis of rotation is constant in the body frame. Figure 9.2 illustrates a screw path and a decoupled path for the same X_{start} and X_{end}.

9.2.2 Time Scaling a Straight-Line Path

A time scaling $s(t)$ of a path should ensure that the motion is appropriately smooth and that any constraints on robot velocity and acceleration are satisfied. For a straight-line path in joint space of the form (9.3), the time-scaled joint velocities and accelerations are $\dot{\theta} = \dot{s}(\theta_{\text{end}} - \theta_{\text{start}})$ and $\ddot{\theta} = \ddot{s}(\theta_{\text{end}} - \theta_{\text{start}})$, respectively. For a straight-line path in task space parametrized by a minimum set of coordinates $X \in \mathbb{R}^m$, simply replace θ, $\dot{\theta}$, and $\ddot{\theta}$ by X, $\dot{X}$, and $\ddot{X}$.

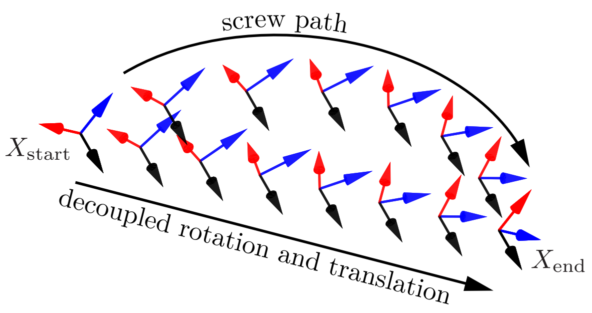

Figure 9.2: A path following a constant screw motion versus a decoupled path where the frame origin follows a straight line and the angular velocity is constant.

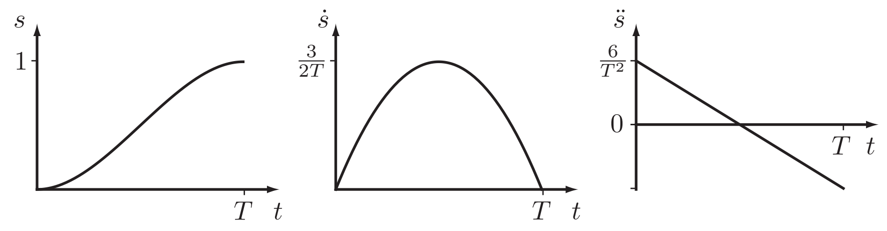

Figure 9.3: Plots of $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$ for a third-order polynomial time scaling.

9.2.2.1 Polynomial Time Scaling

Third-order Polynomials A convenient form for the time scaling $s(t)$ is a cubic polynomial of time,

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3. \tag{9.9}$$

A point-to-point motion in time T imposes the initial constraints $s(0) = \dot{s}(0) = 0$ and the terminal constraints $s(T) = 1$ and $\dot{s}(T) = 0$. Evaluating Equation (9.9) and its derivative

$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{9.10}$$

at $t = 0$ and $t = T$ and solving the four constraints for $a_0, \ldots, a_3$, we find

$$a_0 = 0, \quad a_1 = 0, \quad a_2 = \frac{3}{T^2}, \quad a_3 = -\frac{2}{T^3}.$$

Plots of $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$ are shown in Figure 9.3.

Substituting $s = a_2t^2 + a_3t^3$ into Equation (9.3) yields

$$\theta(t) = \theta_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}}), \tag{9.11}$$

$$\dot\theta(t) = \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}}), \tag{9.12}$$

$$\ddot\theta(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}}). \tag{9.13}$$

The maximum joint velocities are achieved at the halfway point of the motion, $t = T/2$:

$$\dot\theta_{\max} = \frac{3}{2T}(\theta_{\text{end}} - \theta_{\text{start}}).$$

The maximum joint accelerations and decelerations are achieved at $t = 0$ and $t = T$:

$$\ddot\theta_{\max} = \left|\frac{6}{T^2}(\theta_{\text{end}} - \theta_{\text{start}})\right|, \qquad \ddot\theta_{\min} = -\left|\frac{6}{T^2}(\theta_{\text{end}} - \theta_{\text{start}})\right|.$$

If there are known limits on the maximum joint velocities $|\dot\theta| \le \dot\theta_{\text{limit}}$ and maximum joint accelerations $|\ddot\theta| \le \ddot\theta_{\text{limit}}$, these bounds can be checked to see whether the requested motion time T is feasible. Alternatively, one could solve for T to find the minimum possible motion time that satisfies the most restrictive velocity or acceleration constraint.

Fifth-order Polynomials Because third-order time scaling does not constrain the endpoint path accelerations $\ddot s(0)$ and $\ddot s(T)$ to be zero, the robot is asked to achieve a discontinuous jump in acceleration at both $t = 0$ and $t = T$. This implies an infinite *jerk*, the derivative of acceleration, which may cause vibration of the robot.

One solution is to constrain the endpoint accelerations to $\ddot s(0) = \ddot s(T) = 0$. Adding these two constraints to the problem formulation requires the addition of two more design freedoms in the polynomial, yielding a quintic polynomial of time, $s(t) = a_0 + \cdots + a_5t^5$. We can use the six terminal position, velocity, and acceleration constraints to solve uniquely for $a_0, \ldots, a_5$ (Exercise 9.5), which yields a smoother motion with a higher maximum velocity than a cubic time scaling. A plot of the time scaling is shown in Figure 9.4.

9.2.2.2 Trapezoidal Motion Profiles

Trapezoidal time scalings are quite common in motor control, particularly for the motion of a single joint, and they get their name from their velocity profiles.

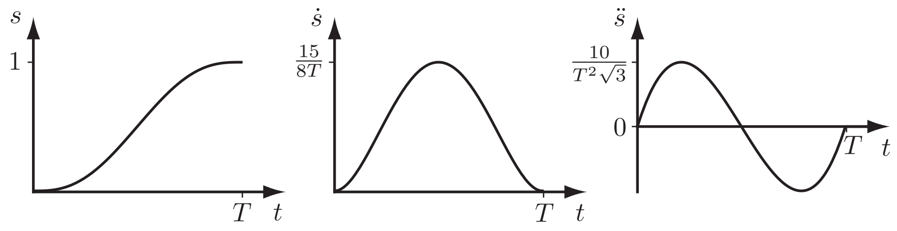

Figure 9.4: Plots of $s(t)$, $\dot{s}(t)$, and $\ddot{s}(t)$ for a fifth-order polynomial time scaling.

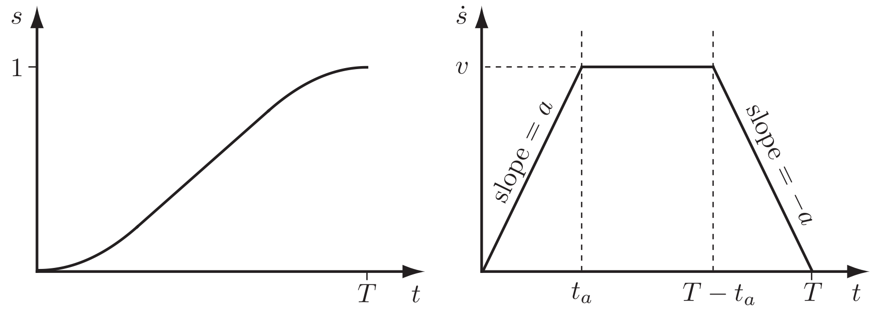

Figure 9.5: Plots of $s(t)$ and $\dot{s}(t)$ for a trapezoidal motion profile.

The point-to-point motion consists of a constant acceleration phase $\ddot{s} = a$ of time t_a, followed by a constant velocity phase $\dot{s} = v$ of time $t_v = T - 2t_a$, followed by a constant deceleration phase $\ddot{s} = -a$ of time t_a. The resulting $\dot{s}$ profile is a trapezoid and the s profile is the concatenation of a parabola, linear segment, and parabola as a function of time (Figure 9.5).

The trapezoidal time scaling is not as smooth as the cubic time scaling, but it has the advantage that if there are known constant limits on the joint velocities $\dot{\theta}_{\text{limit}} \in \mathbb{R}^n$ and on the joint accelerations $\ddot{\theta}_{\text{limit}} \in \mathbb{R}^n$ then the trapezoidal motion using the largest v and a satisfying

$$|(\theta_{\text{end}} - \theta_{\text{start}})v| \leq \dot{\theta}_{\text{limit}}, \tag{9.14}$$

$$|(\theta_{\text{end}} - \theta_{\text{start}})a| \leq \ddot{\theta}_{\text{limit}} \tag{9.15}$$

is the fastest straight-line motion possible. (See Exercise 9.8.)

If $v^2/a > 1$, the robot never reaches the velocity v during the motion (Exercise 9.10). The three-phase accelerate–coast–decelerate motion becomes a two-phase accelerate–decelerate "bang-bang" motion, and the trapezoidal profile $\dot{s}(t)$ in Figure 9.5 becomes a triangle.

Assuming that $v^2/a \leq 1$, the trapezoidal motion is fully specified by v, a, t_a, and T, but only two of these can be specified independently since they must satisfy $s(T) = 1$ and $v = at_a$. It is unlikely that we would specify t_a independently, so we can eliminate it from the equations of motion by the substitution $t_a = v/a$. The motion profile during the three stages (acceleration, coast, deceleration) can then be written in terms of v, a, and T as follows:

$$\text{for } 0 \leq t \leq \frac{v}{a}, \qquad \ddot{s}(t) = a, \tag{9.16}$$

$$\dot{s}(t) = at, \tag{9.17}$$

$$s(t) = \frac{1}{2}at^2; \tag{9.18}$$

$$\text{for } \frac{v}{a} < t \leq T - \frac{v}{a}, \qquad \ddot{s}(t) = 0, \tag{9.19}$$

$$\dot{s}(t) = v, \tag{9.20}$$

$$s(t) = vt - \frac{v^2}{2a}; \tag{9.21}$$

$$\text{for } T - \frac{v}{a} < t \leq T, \qquad \ddot{s}(t) = -a, \tag{9.22}$$

$$\dot{s}(t) = a(T - t), \tag{9.23}$$

$$s(t) = \frac{2avT - 2v^2 - a^2(t-T)^2}{2a}. \tag{9.24}$$

Since only two of v, a, and T can be chosen independently, we have three options:

- Choose v and a such that $v^2/a \leq 1$, ensuring a three-stage trapezoidal profile, and solve $s(T) = 1$ (using Equation (9.24)) for T:

  $$T = \frac{a + v^2}{va}.$$

  If v and a correspond to the highest possible joint velocities and accelerations, this is the minimum possible time for the motion.

- Choose v and T such that $2 \geq vT > 1$, ensuring a three-stage trapezoidal profile and that the top speed v is sufficient to reach $s = 1$ in time T, and solve $s(T) = 1$ for a:

  $$a = \frac{v^2}{vT - 1}.$$

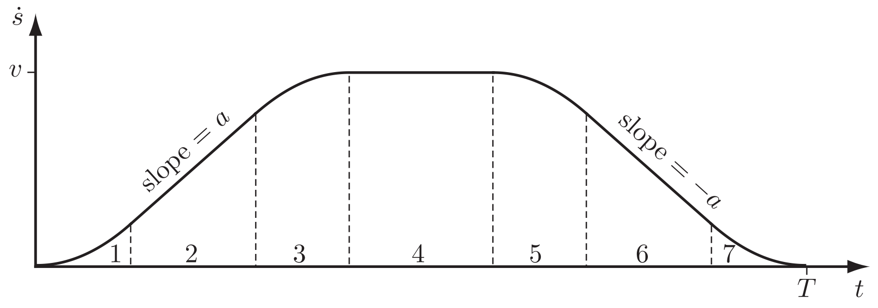

Figure 9.6: Plot of $\dot{s}(t)$ for an S-curve motion profile consisting of seven stages: (1) constant positive jerk, (2) constant acceleration, (3) constant negative jerk, (4) constant velocity, (5) constant negative jerk, (6) constant deceleration, and (7) constant positive jerk.

- Choose a and T such that $aT^2 \geq 4$, ensuring that the motion is completed in time, and solve $s(T) = 1$ for v:

$$v = \frac{1}{2}\left(aT - \sqrt{a}\sqrt{aT^2 - 4}\right).$$

9.2.2.3 S-Curve Time Scalings

Just as cubic polynomial time scalings lead to infinite jerk at the beginning and end of the motion, trapezoidal motions cause discontinuous jumps in acceleration at $t \in \{0, t_a, T - t_a, T\}$. A solution is a smoother **S-curve** time scaling, a popular motion profile in motor control because it avoids vibrations or oscillations induced by step changes in acceleration. An S-curve time scaling consists of seven stages: (1) constant jerk $d^3s/dt^3 = J$ until a desired acceleration $\ddot{s} = a$ is achieved; (2) constant acceleration until the desired $\dot{s} = v$ is being approached; (3) constant negative jerk $-J$ until $\ddot{s}$ equals zero exactly at the time $\dot{s}$ reaches v; (4) coasting at constant v; (5) constant negative jerk $-J$; (6) constant deceleration $-a$; and (7) constant positive jerk J until $\ddot{s}$ and $\dot{s}$ reach zero exactly at the time s reaches 1.

The $\dot{s}(t)$ profile for an S-curve is shown in Figure 9.6.

Given some subset of v, a, J, and the total motion time T, algebraic manipulation reveals the switching time between stages and conditions that ensure that all seven stages are actually achieved, similarly to the case of the trapezoidal motion profile.

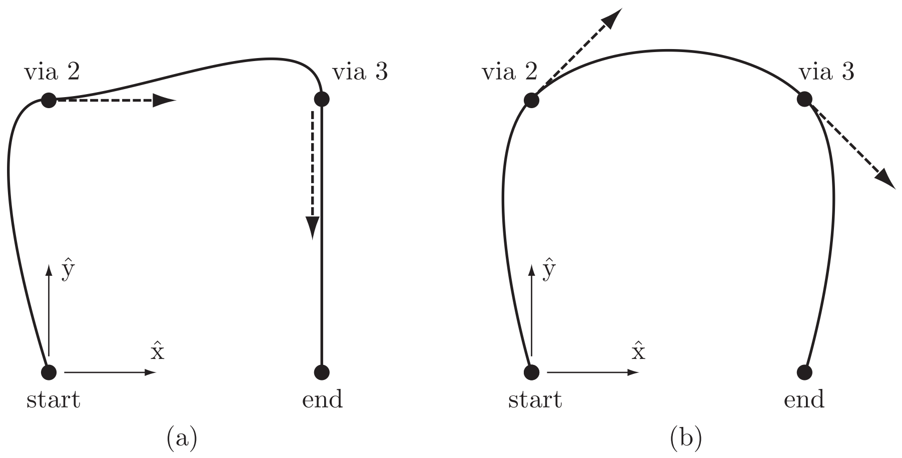

Figure 9.7: Two paths in an (x, y) space corresponding to piecewise-cubic trajectories interpolating four via points, including a start point and an end point. The velocities at the start and end are zero, and the velocities at vias 2 and 3 are indicated by the dashed tangent vectors. The shape of the path depends on the velocities specified at the via points.

9.3 Polynomial Via Point Trajectories

If the goal is to have the robot joints pass through a series of **via points** at specified times, without a strict specification on the shape of path between consecutive points, a simple solution is to use polynomial interpolation to find joint histories $\theta(t)$ directly without first specifying a path $\theta(s)$ and then a time scaling $s(t)$ (Figure 9.7).

Let the trajectory be specified by k via points, with the start point occurring at $T_1 = 0$ and the final point at $T_k = T$. Since each joint history is interpolated individually, we focus on a single joint variable and call it β to avoid a proliferation of subscripts. At each via point $i \in \{1, \ldots, k\}$, the user specifies the desired position $\beta(T_i) = \beta_i$ and velocity $\dot{\beta}(T_i) = \dot{\beta}_i$. The trajectory has $k-1$ segments, and the duration of segment $j \in \{1, \ldots, k-1\}$ is $\Delta T_j = T_{j+1} - T_j$. The joint trajectory during segment j is expressed as the third-order polynomial

$$\beta(T_j + \Delta t) = a_{j0} + a_{j1}\Delta t + a_{j2}\Delta t^2 + a_{j3}\Delta t^3 \tag{9.25}$$

in terms of the time Δt elapsed in segment j, where $0 \leq \Delta t \leq \Delta T_j$. Segment j

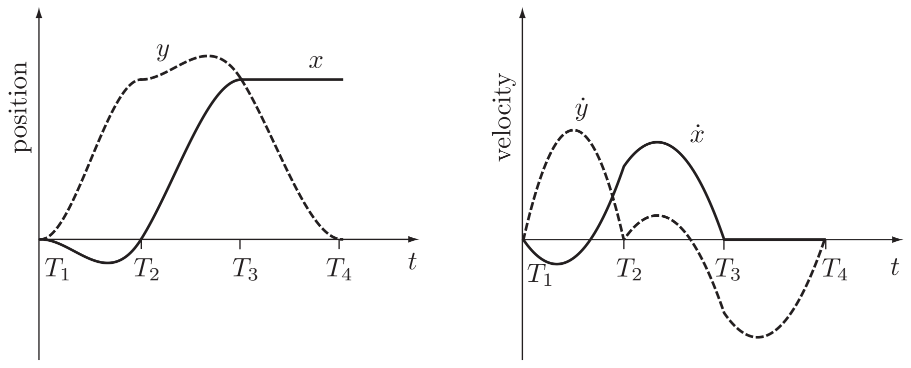

Figure 9.8: The coordinate time histories for the cubic via-point interpolation of Figure 9.7(a).

is subject to the four constraints

$$\beta(T_j) = \beta_j, \qquad \dot{\beta}(T_j) = \dot{\beta}_j,$$
$$\beta(T_j + \Delta T_j) = \beta_{j+1}, \qquad \dot{\beta}(T_j + \Delta T_j) = \dot{\beta}_{j+1}.$$

Solving these constraints for $a_{j0}, \ldots, a_{j3}$ yields

$$a_{j0} = \beta_j, \tag{9.26}$$

$$a_{j1} = \dot{\beta}_j, \tag{9.27}$$

$$a_{j2} = \frac{3\beta_{j+1} - 3\beta_j - 2\dot{\beta}_j \Delta T_j - \dot{\beta}_{j+1}\Delta T_j}{\Delta T_j^2}, \tag{9.28}$$

$$a_{j3} = \frac{2\beta_j + (\dot{\beta}_j + \dot{\beta}_{j+1})\Delta T_j - 2\beta_{j+1}}{\Delta T_j^3}. \tag{9.29}$$

Figure 9.8 shows the time histories for the interpolation of Figure 9.7(a). In this two-dimensional (x, y) coordinate space the via points $1, \ldots, 4$ occur at times $T_1 = 0$, $T_2 = 1$, $T_3 = 2$, and $T_4 = 3$. The via points are at $(0, 0)$, $(0, 1)$, $(1, 1)$, and $(1, 0)$ with velocities $(0, 0)$, $(1, 0)$, $(0, -1)$, and $(0, 0)$.

Two issues are worth mentioning:

- The quality of the interpolated trajectories is improved by "reasonable" combinations of via-point times and via-point velocities. For example, if the user wants to specify a via-point location and time, but not the velocity, a heuristic could be used to choose a via velocity on the basis of the times and coordinate vectors to the via points before and after the via in question. As an example, the trajectory of Figure 9.7(b) is smoother than the trajectory of Figure 9.7(a).

- Cubic via-point interpolation ensures that velocities are continuous at via points, but not accelerations. The approach is easily generalized to the use of fifth-order polynomials and specification of the accelerations at the via points, at the cost of increased complexity of the solution.

If only two points are used (the start and end point), and the velocities at each are zero, the resulting trajectory is identical to the straight-line cubic polynomial time-scaled trajectory discussed in Section 9.2.2.1.

There are many other methods for interpolating a set of via points. For example, B-spline interpolation is popular. In B-spline interpolation, the path may not pass exactly through the via points, but the path is guaranteed to be confined to the convex hull of the via points, unlike the paths in Figure 9.7. This can be important to ensure that joint limits or workspace obstacles are respected.

9.4 Time-Optimal Time Scaling

In the case where the path $\theta(s)$ is fully specified by the task or an obstacle-avoiding path planner (e.g., Figure 9.9), the trajectory planning problem reduces to finding a time scaling $s(t)$. One could choose the time scaling to minimize the energy consumed while meeting a time constraint, or to prevent spilling a glass of water that the robot is carrying. One of the most useful time scalings minimizes the time of motion along the path, subject to the robot's actuator limits. Such time-optimal trajectories maximize the robot's productivity.

While the trapezoidal time scalings of Section 9.2.2.2 can yield time-optimal trajectories, this is only under the assumption of straight-line motions, constant maximum acceleration a, and constant maximum coasting velocity v. For most robots, because of state-dependent joint actuator limits and the state-dependent dynamics

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) = \tau, \tag{9.30}$$

the maximum available velocities and accelerations change along the path.

In this section we consider the problem of finding the fastest possible time scaling $s(t)$ that respects the robot's actuator limits. We write the limits on the ith actuator as

$$\tau_i^{\min}(\theta, \dot{\theta}) \leq \tau_i \leq \tau_i^{\max}(\theta, \dot{\theta}). \tag{9.31}$$

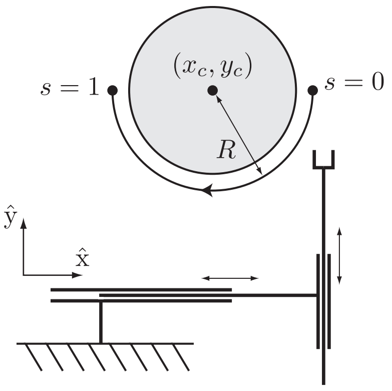

Figure 9.9: A path planner has returned a semicircular path of radius R around an obstacle in (x, y) space for a robot with two prismatic joints. The path can be represented in terms of a path parameter s, as $x(s) = x_c + R\cos s\pi$ and $y(s) = y_c - R\sin s\pi$ for $s \in [0, 1]$. For a 2R robot, inverse kinematics would be used to express the path as a function of s in joint coordinates.

The available actuator torque is typically a function of the current joint speed (see Section 8.9.1). For example, for a given maximum voltage of a DC motor, the maximum torque available from the motor drops linearly with the motor's speed.

Before proceeding we recall that the quadratic velocity terms $c(\theta, \dot{\theta})$ in Equation (9.30) can be written equivalently as

$$c(\theta, \dot{\theta}) = \dot{\theta}^{\mathrm{T}}\Gamma(\theta)\dot{\theta},$$

where $\Gamma(\theta)$ is the three-dimensional tensor of Christoffel symbols constructed from partial derivatives of components of the mass matrix $M(\theta)$ with respect to θ. This form shows more clearly the quadratic dependence on velocities. Now, beginning with Equation (9.30), replacing $\dot{\theta}$ by $(d\theta/ds)\dot{s}$ and $\ddot{\theta}$ by $(d\theta/ds)\ddot{s} + (d^2\theta/ds^2)\dot{s}^2$, and rearranging, we get

$$\underbrace{\left(M(\theta(s))\frac{d\theta}{ds}\right)}_{m(s)\in\mathbb{R}^n}\ddot{s} + \underbrace{\left(M(\theta(s))\frac{d^2\theta}{ds^2} + \left(\frac{d\theta}{ds}\right)^{\mathrm{T}}\Gamma(\theta(s))\frac{d\theta}{ds}\right)}_{c(s)\in\mathbb{R}^n}\dot{s}^2 + \underbrace{g(\theta(s))}_{g(s)\in\mathbb{R}^n} = \tau, \tag{9.32}$$

expressed more compactly as the vector equation

$$m(s)\ddot{s} + c(s)\dot{s}^2 + g(s) = \tau, \tag{9.33}$$

where $m(s)$ is the effective inertia of the robot when it is confined to the path $\theta(s)$, $c(s)\dot{s}^2$ comprises the quadratic velocity terms, and $g(s)$ is the gravitational torque.

Similarly, the actuation constraints (9.31) can be expressed as a function of s:

$$\tau_i^{\min}(s,\dot{s}) \le \tau_i \le \tau_i^{\max}(s,\dot{s}). \tag{9.34}$$

Substituting the ith component of Equation (9.33), we get

$$\tau_i^{\min}(s,\dot{s}) \le m_i(s)\ddot{s} + c_i(s)\dot{s}^2 + g_i(s) \le \tau_i^{\max}(s,\dot{s}). \tag{9.35}$$

Let $L_i(s,\dot{s})$ and $U_i(s,\dot{s})$ be the minimum and maximum accelerations $\ddot{s}$ satisfying the ith component of Equation (9.35). Depending on the sign of $m_i(s)$, we have three possibilities:

$$\left.\begin{array}{lll}
\text{if } m_i(s) > 0, & L_i(s,\dot{s}) = \dfrac{\tau_i^{\min}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}, & \\
 & U_i(s,\dot{s}) = \dfrac{\tau_i^{\max}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)} & \\
\text{if } m_i(s) < 0, & L_i(s,\dot{s}) = \dfrac{\tau_i^{\max}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)}, & \\
 & U_i(s,\dot{s}) = \dfrac{\tau_i^{\min}(s,\dot{s}) - c(s)\dot{s}^2 - g(s)}{m_i(s)} & \\
\text{if } m_i(s) = 0, & \text{we have a \textit{zero-inertia point}, discussed in Section 9.4.4.} &
\end{array}\right\} \tag{9.36}$$

Defining

$$L(s,\dot{s}) = \max_i L_i(s,\dot{s}) \qquad \text{and} \qquad U(s,\dot{s}) = \min_i U_i(s,\dot{s}),$$

the actuator limits (9.35) can be written as the state-dependent time-scaling constraints

$$L(s,\dot{s}) \le \ddot{s} \le U(s,\dot{s}). \tag{9.37}$$

The time-optimal time-scaling problem can now be stated:

Given a path $\theta(s), s \in [0,1]$, an initial state $(s_0,\dot{s}_0) = (0,0)$, and a final state $(s_f,\dot{s}_f) = (1,0)$, find a monotonically increasing twice-differentiable time scaling $s : [0,T] \to [0,1]$ that

(a) *satisfies $s(0) = \dot{s}(0) = \dot{s}(T) = 0$ and $s(T) = 1$, and*

(b) *minimizes the total travel time T along the path while respecting the actuator constraints (9.37).*

The problem formulation is easily generalized to the case of nonzero initial and final velocities along the path, $\dot{s}(0) > 0$ and $\dot{s}(T) > 0$.

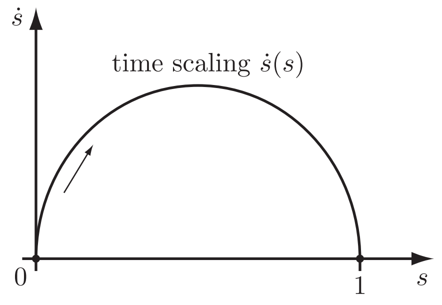

Figure 9.10: A time scaling in the $(s, \dot{s})$ phase plane is a curve, with $\dot{s} \geq 0$ at all times, connecting the initial path position and velocity $(0, 0)$ to the final position and velocity $(1, 0)$.

9.4.1 The $(s, \dot{s})$ Phase Plane

The problem is easily visualized in the $(s, \dot{s})$ phase plane of the path-constrained robot, with s running from 0 to 1 on a horizontal axis and $\dot{s}$ on a vertical axis. Since $s(t)$ is monotonically increasing, $\dot{s}(t) \geq 0$ for all times t and for all $s \in [0, 1]$. A time scaling of the path is any curve in the phase plane that moves monotonically to the right from $(0, 0)$ to $(1, 0)$ (Figure 9.10). Not all such curves satisfy the acceleration constraints (9.37), however.

To see the effect of the acceleration constraints, at each $(s, \dot{s})$ in the phase plane, we can plot the limits $L(s, \dot{s}) \leq \ddot{s} \leq U(s, \dot{s})$ as a cone constructed from $\dot{s}$, L, and U, as illustrated in two dimensions in Figure 9.11(a). If $L(s, \dot{s}) \geq U(s, \dot{s})$, the cone disappears – there are no actuator commands that can keep the robot on the path at this state. These **inadmissible** states are indicated in gray in Figure 9.11(a). For any s, typically there is a single limit velocity $\dot{s}_{\text{lim}}(s)$ above which all velocities are inadmissible. The function $\dot{s}_{\text{lim}}(s)$ is called the **velocity limit curve**. On the velocity limit curve, $L(s, \dot{s}) = U(s, \dot{s})$, and the cone reduces to a single vector.

For a time scaling to satisfy the acceleration constraints, the tangent of the time-scaling curve must lie inside the feasible cone at all points on the curve. Figure 9.11(b) shows an example of an infeasible time scaling, which demands more deceleration than the actuators can provide at the state indicated.

For a minimum-time motion, the "speed" $\dot{s}$ must be as high as possible at every s while still satisfying the acceleration constraints and the endpoint constraints. To see this, write the total time of motion T as

$$T = \int_0^T 1 \, dt. \tag{9.38}$$

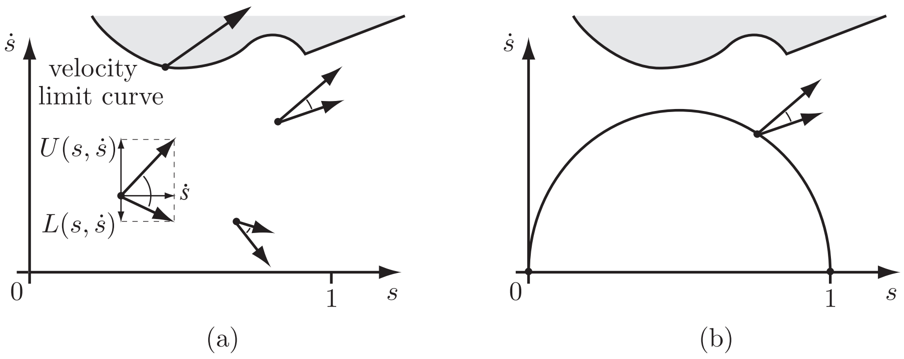

Figure 9.11: (a) Acceleration-limited motion cones at four different states. The upper ray of the cone is the sum of $U(s,\dot{s})$ plotted in the vertical direction (the change in velocity) and $\dot{s}$ plotted in the horizontal direction (the change in position). The lower ray of the cone is constructed from $L(s,\dot{s})$ and $\dot{s}$. The points in gray, bounded by the velocity limit curve, have $L(s,\dot{s}) \geq U(s,\dot{s})$: the state is inadmissible and there is no motion cone. On the velocity limit curve the cone is reduced to a single tangent vector. (b) The proposed time scaling is infeasible because the tangent to the curve is outside the motion cone at the state indicated.

Making the substitution $ds/ds = 1$, and changing the limits of integration from 0 to T (time) to 0 to 1 (s), we get

$$T = \int_0^T 1 \, dt = \int_0^T \frac{ds}{ds} \, dt = \int_0^1 \frac{dt}{ds} \, ds = \int_0^1 \dot{s}^{-1}(s) \, ds. \qquad (9.39)$$

Thus for time to be minimized, $\dot{s}^{-1}(s)$ should be as small as possible, and therefore $\dot{s}(s)$ must be as large as possible, at all s, while still satisfying the acceleration constraints (9.37) and the boundary constraints.

This implies that the time scaling must always operate either at the limit $U(s,\dot{s})$ or at the limit $L(s,\dot{s})$, and our only choice is when to switch between these limits. A common solution is a *bang-bang* trajectory: maximum acceleration $U(s,\dot{s})$ followed by a switch to maximum deceleration $L(s,\dot{s})$. (This is similar to the trapezoidal motion profile that never reaches the coasting velocity v in Section 9.2.2.2.) In this case the time scaling is calculated by numerically integrating $U(s,\dot{s})$ forward in s from $(0,0)$, integrating $L(s,\dot{s})$ backward in s from $(1,0)$, and finding the intersection of these curves (Figure 9.12(a)). The switch between maximum acceleration and maximum deceleration occurs at the intersection.

In some cases, the velocity limit curve prevents a single-switch solution (Fig-

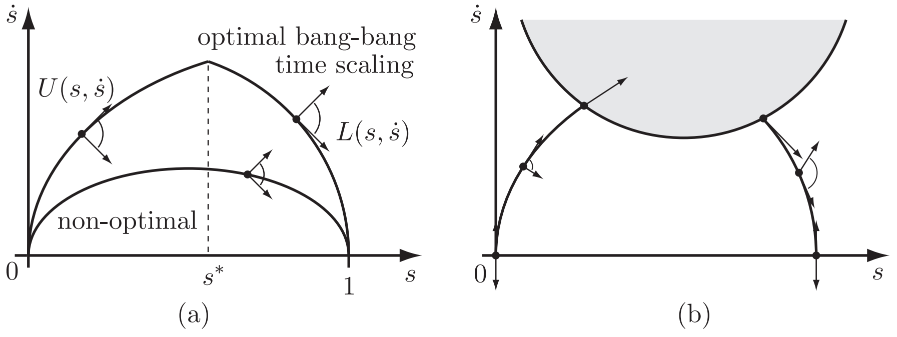

Figure 9.12: (a) A time-optimal bang-bang time scaling integrates $U(s, \dot{s})$ from $(0, 0)$ and switches to $L(s, \dot{s})$ at a switching point s^*. Also shown is a non-optimal time scaling with a tangent inside a motion cone. (b) Sometimes the velocity limit curve prevents a single-switch solution.

ure 9.12(b)). These cases require an algorithm to find multiple switching points.

9.4.2 The Time-Scaling Algorithm

Finding the optimal time scaling is reduced to finding the switches between maximum acceleration $U(s, \dot{s})$ and maximum deceleration $L(s, \dot{s})$, maximizing the "height" of the curve in the $(s, \dot{s})$ phase plane.

Time-scaling algorithm

1. Initialize an empty list of switches $\mathcal{S} = \{\}$ and a switch counter $i = 0$. Set $(s_i, \dot{s}_i) = (0, 0)$.
2. Integrate the equation $\ddot{s} = L(s, \dot{s})$ backward in time from $(1, 0)$ until $L(s, \dot{s}) > U(s, \dot{s})$ (the velocity limit curve is penetrated) or $s = 0$. Call this phase plane curve F.
3. Integrate the equation $\ddot{s} = U(s, \dot{s})$ forward in time from $(s_i, \dot{s}_i)$ until it crosses F or until $U(s, \dot{s}) < L(s, \dot{s})$ (the velocity limit curve is penetrated). Call this curve A_i. If A_i crosses F then increment i, set $(s_i, \dot{s}_i)$ to the $(s, \dot{s})$ value at which the crossing occurs, and append s_i to the list of switches $\mathcal{S}$. This is a switch from maximum acceleration to maximum deceleration. The problem is solved and $\mathcal{S}$ is the set of switches expressed in the path parameter. If instead the velocity limit curve is penetrated, let $(s_{\lim}, \dot{s}_{\lim})$ be the point of penetration and proceed to the next step.

4. Perform a binary search on the velocity in the range $[0, \dot{s}_{\text{lim}}]$ to find the velocity $\dot{s}'$ such that the curve integrating $\ddot{s} = L(s, \dot{s})$ forward from $(s_{\text{lim}}, \dot{s}')$ touches the velocity limit curve without penetrating it. The binary search is initiated with $\dot{s}_{\text{high}} = \dot{s}_{\text{lim}}$ and $\dot{s}_{\text{low}} = 0$.

   (a) Set the test velocity halfway between $\dot{s}_{\text{low}}$ and $\dot{s}_{\text{high}}$: $\dot{s}_{\text{test}} = (\dot{s}_{\text{high}} + \dot{s}_{\text{low}})/2$. The test point is $(s_{\text{lim}}, \dot{s}_{\text{test}})$.

   (b) If the curve from the test point penetrates the velocity limit curve, set $\dot{s}_{\text{high}}$ equal to $\dot{s}_{\text{test}}$. If instead the curve from the test point hits $\dot{s} = 0$, set $\dot{s}_{\text{low}}$ equal to $\dot{s}_{\text{test}}$. Return to step 4(a).

   Continue the binary search until a specified tolerance. Let $(s_{\text{tan}}, \dot{s}_{\text{tan}})$ be the point where the resulting curve just touches the velocity limit curve tangentially (or comes closest to the curve without hitting it). The motion cone at this point is reduced to a single vector $(L(s, \dot{s}) = U(s, \dot{s}))$, tangent to the velocity limit curve.

5. Integrate $\ddot{s} = L(s, \dot{s})$ backwards from $(s_{\text{tan}}, \dot{s}_{\text{tan}})$ until it intersects A_i. Increment i, set $(s_i, \dot{s}_i)$ to the $(s, \dot{s})$ value at the intersection, and label as A_i the curve segment from $(s_i, \dot{s}_i)$ to $(s_{\text{tan}}, \dot{s}_{\text{tan}})$. Append s_i to the list of switches $\mathcal{S}$. This is a switch from maximum acceleration to maximum deceleration.

6. Increment i and set $(s_i, \dot{s}_i)$ to $(s_{\text{tan}}, \dot{s}_{\text{tan}})$. Append s_i to the list of switches $\mathcal{S}$. This is a switch from maximum deceleration to maximum acceleration. Go to step 3.

Figure 9.13 shows steps 2–6 of the time-scaling algorithm. (Step 2) Integration of $\ddot{s} = L(s, \dot{s})$ backward from $(1, 0)$ until the velocity limit curve is reached. (Step 3) Integration of $\ddot{s} = U(s, \dot{s})$ forward from $(0, 0)$ to the intersection $(s_{\text{lim}}, \dot{s}_{\text{lim}})$ with the velocity limit curve. (Step 4) Binary search to find $(s_{\text{lim}}, \dot{s}')$ from which $\ddot{s} = L(s, \dot{s})$, integrated forward from $(s_{\text{lim}}, \dot{s}')$, touches the velocity limit curve tangentially. (Step 5) Integration backward along $L(s, \dot{s})$ from $(s_{\text{tan}}, \dot{s}_{\text{tan}})$ to find the first switch from acceleration to deceleration. (Step 6) The second switch, from deceleration to acceleration, is at $(s_2, \dot{s}_2) = (s_{\text{tan}}, \dot{s}_{\text{tan}})$. (Step 3) Integration forward along $U(s, \dot{s})$ from $(s_2, \dot{s}_2)$ results in intersection with F at $(s_3, \dot{s}_3)$, where a switch occurs from acceleration to deceleration. The optimal time scaling consists of switches at $\mathcal{S} = \{s_1, s_2, s_3\}$.

9.4.3 A Variation on the Time-Scaling Algorithm

Remember that each point $(s, \dot{s})$ below the velocity limit curve has a cone of feasible motions, while each point on the velocity limit curve has a single feasible

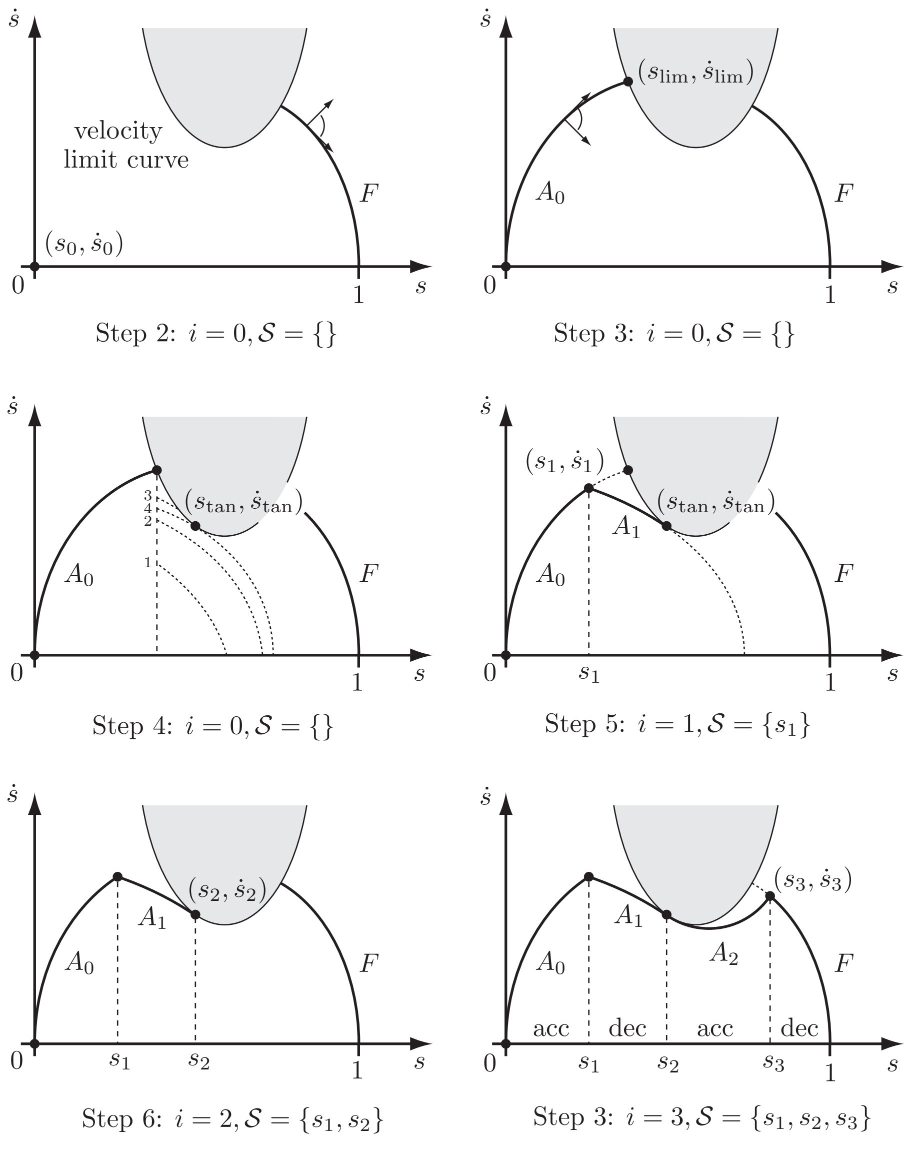
$\dot{s}$
velocity limit curve
F
$(s_0, \dot{s}_0)$
0
1
s
Step 2: $i = 0, \mathcal{S} = \{\}$
$(s_{\text{lim}}, \dot{s}_{\text{lim}})$
A_0
Step 3: $i = 0, \mathcal{S} = \{\}$
$(s_{\text{tan}}, \dot{s}_{\text{tan}})$
3
4
2
1
Step 4: $i = 0, \mathcal{S} = \{\}$
$(s_1, \dot{s}_1)$
A_1
s_1
Step 5: $i = 1, \mathcal{S} = \{s_1\}$
$(s_2, \dot{s}_2)$
s_2
Step 6: $i = 2, \mathcal{S} = \{s_1, s_2\}$
$(s_3, \dot{s}_3)$
A_2
acc
dec
acc
dec
s_3
Step 3: $i = 3, \mathcal{S} = \{s_1, s_2, s_3\}$

Figure 9.13: The time-scaling algorithm.

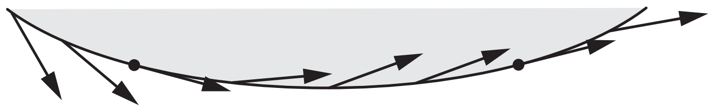

Figure 9.14: A point on the velocity limit curve can only be a part of a time-optimal time scaling if the feasible motion vector at that point is tangential to the curve. A search along the velocity limit curve illustrated reveals that there are only two points on this particular curve (marked by dots) that can belong to a time-optimal time scaling.

vector. The only points on the velocity limit curve that can be part of an optimal solution are those where the feasible motion vector is tangent to the velocity limit curve; these are the points $(s_{\text{tan}}, \dot{s}_{\text{tan}})$ referred to above. Recognizing this, the binary search in step 4 of the time-scaling algorithm, which is essentially searching for a point $(s_{\text{tan}}, \dot{s}_{\text{tan}})$, can be replaced by explicit construction of the velocity limit curve and a search for points on this curve satisfying the tangency condition. See Figure 9.14.

9.4.4 Assumptions and Caveats

The description above covers the major points of the optimal time-scaling algorithm. A few assumptions were glossed over; they are made explicit now.

- **Static posture maintenance.** The algorithm, as described, assumes that the robot can maintain its configuration against gravity at any state $(s, \dot{s} = 0)$. This ensures the existence of valid time scalings, namely, time scalings that move the robot along the path arbitrarily slowly. For some robots and paths this assumption may be violated owing to weakness of the actuators. For example, some paths may require some momentum to carry the motion through configurations that the robot cannot maintain statically. The algorithm can be modified to handle such cases.

- **Inadmissible states.** The algorithm assumes that at every s there is a unique velocity limit $\dot{s}_{\text{lim}}(s) > 0$ such that all velocities $\dot{s} \leq \dot{s}_{\text{lim}}(s)$ are admissible and all velocities $\dot{s} > \dot{s}_{\text{lim}}(s)$ are inadmissible. For some models of actuator dynamics or friction this assumption may be violated – there may be isolated "islands" of inadmissible states. The algorithm can be modified to handle this case.

- **Zero-inertia points.** The algorithm assumes that there are no zero-inertia points (Equation (9.36)). If $m_i(s) = 0$ in (9.36) then the torque provided by actuator i has no dependence on the acceleration $\ddot{s}$, and the ith actuator constraint in (9.35) directly defines a *velocity* constraint on $\dot{s}$. At a point s with one or more zero components in $m(s)$, the velocity limit curve is defined by the minimum of (a) the velocity constraints defined by the zero-inertia components and (b) the $\dot{s}$ values satisfying $L_i(s, \dot{s}) = U_i(s, \dot{s})$ for the other components. For the algorithm as described, singular arcs of zero-inertia points on the velocity limit curve may lead to rapid switching between $\ddot{s} = U(s, \dot{s})$ and $\ddot{s} = L(s, \dot{s})$. In such cases, choosing an acceleration tangent to the velocity limit curve and lying between $U(s, \dot{s})$ and $L(s, \dot{s})$, preserves time optimality without causing chattering of the controls.

It is worth noting that the time-scaling algorithm generates trajectories with discontinuous acceleration, which could lead to vibrations. Beyond this, inaccuracies in models of robot inertial properties and friction make direct application of the time-scaling algorithm impractical. Finally, since a minimum-time time scaling always saturates at least one actuator, if the robot moves off the planned trajectory, there may be no torque left for corrective action by a feedback controller.

Despite these drawbacks, the time-scaling algorithm provides a deep understanding of the true maximum capabilities of a robot following a path.

9.5 Summary

- A trajectory $\theta(t)$, $\theta : [0, T] \to \Theta$, can be written as $\theta(s(t))$, i.e., as the composition of a path $\theta(s)$, $\theta : [0, 1] \to \Theta$, and a time scaling $s(t)$, $s : [0, T] \to [0, 1]$.

- A straight-line path in joint space can be written $\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}})$, $s \in [0, 1]$. A similar form holds for straight-line paths in a minimum set of task-space coordinates. A "straight-line" path in $SE(3)$, where $X = (R, p)$, can be decoupled to a Cartesian path and a rotation path:

$$p(s) = p_{\text{start}} + s(p_{\text{end}} - p_{\text{start}}), \tag{9.40}$$
$$R(s) = R_{\text{start}} \exp(\log(R_{\text{start}}^{\text{T}} R_{\text{end}})s). \tag{9.41}$$

- A cubic polynomial $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ can be used to time scale a point-to-point motion with zero initial and final velocities. The acceleration undergoes a step change (an infinite jerk) at $t = 0$ and $t = T$. Such an impulse in jerk can cause vibration of the robot.

- A quintic polynomial $s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$ can be used to time-scale a point-to-point motion with zero initial and final velocities and accelerations. The jerk is finite at all times.

- The trapezoidal motion profile is a popular time scaling in point-to-point control, particularly the control of a single motor. The motion consists of three phases: constant acceleration, constant velocity, and constant deceleration, resulting in a trapezoid in $\dot{s}(t)$. Trapezoidal motion involves step changes in acceleration.

- The S-curve motion profile is also popular in point-to-point control of a motor. It consists of seven phases: (1) constant positive jerk, (2) constant acceleration, (3) constant negative jerk, (4) constant velocity, (5) constant negative jerk, (6) constant deceleration, and (7) constant positive jerk.

- Given a set of via points including a start state, a goal state, and other via states through which the robot's motion must pass, as well as the times T_i at which these states should be reached, a series of cubic-polynomial time scalings can be used to generate a trajectory $\theta(t)$ interpolating the via points. To prevent step changes in acceleration at the via points, a series of quintic polynomials can be used instead.

- Given a robot path $\theta(s)$, the dynamics of the robot, and limits on the actuator torques, the actuator constraints can be expressed in terms of $(s, \dot{s})$ as the vector inequalities

$$L(s, \dot{s}) \leq \ddot{s} \leq U(s, \dot{s}).$$

  The time-optimal time scaling $s(t)$ is such that the "height" of the curve in the $(s, \dot{s})$ phase plane is maximized while satisfying $s(0) = \dot{s}(0) = \dot{s}(T) = 0$, $s(T) = 1$, and the actuator constraints. The optimal solution always operates at maximum acceleration $U(s, \dot{s})$ or maximum deceleration $L(s, \dot{s})$.

9.6 Software

Software functions associated with this chapter are listed below.

`s = CubicTimeScaling(Tf,t)`
Computes $s(t)$ for a cubic time scaling, given t and the total time of motion T_f.

`s = QuinticTimeScaling(Tf,t)`

Computes $s(t)$ for a quintic time scaling, given t and the total time of motion T_f.

`traj = JointTrajectory(thetastart,thetaend,Tf,N,method)`
Computes a straight-line trajectory in joint space as an $N \times n$ matrix, where each of the N rows is an n-vector of the joint variables at an instant in time. The first row is θ_{start} and the Nth row is θ_{end}. The elapsed time between each row is $T_f/(N-1)$. The parameter `method` equals either 3 for a cubic time scaling or 5 for a quintic time scaling.

`traj = ScrewTrajectory(Xstart,Xend,Tf,N,method)`
Computes a trajectory as a list of N $SE(3)$ matrices, where each matrix represents the configuration of the end-effector at an instant in time. The first matrix is X_{start}, the Nth matrix is X_{end}, and the motion is along a constant screw axis. The elapsed time between each matrix is $T_f/(N-1)$. The parameter `method` equals either 3 for a cubic time scaling or 5 for a quintic time scaling.

`traj = CartesianTrajectory(Xstart,Xend,Tf,N,method)`
Computes a trajectory as a list of N $SE(3)$ matrices, where each matrix represents the configuration of the end-effector at an instant in time. The first matrix is X_{start}, the Nth matrix is X_{end}, and the origin of the end-effector frame follows a straight line, decoupled from the rotation. The elapsed time between each matrix is $T_f/(N-1)$. The parameter `method` equals either 3 for a cubic time scaling or 5 for a quintic time scaling.

9.7 Notes and References

In 1985, Bobrow et al. [15] and Shin and McKay [168] published papers nearly simultaneously that independently derived the essence of the time-optimal time-scaling algorithm outlined in Section 9.4. A year earlier, Hollerbach addressed the restricted problem of finding dynamically feasible time-scaled trajectories for uniform time scalings where the time variable t is replaced by ct for $c > 0$ [59].

The original papers of Bobrow et al. and Shin and McKay were followed by a number of papers refining the methods by addressing zero-inertia points, singularities, algorithm efficiency, and even the presence of constraints and obstacles [138, 173, 163, 164, 165, 166, 167, 139, 140]. In particular, a computationally efficient method for finding the points $(s_{\text{tan}}, \dot{s}_{\text{tan}})$, where the optimal time scaling touches the velocity limit curve, is described in [138, 173]. This algorithm is used to improve the computational efficiency of the time-scaling algorithm; see, for example, the description and supporting open-source code in [139, 140]. In this chapter the binary search approach in step 4 of the time-scaling algorithm

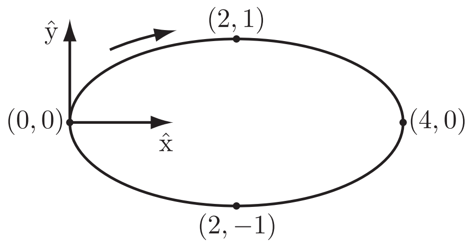

Figure 9.15: An elliptical path.

follows the presentation in [15] because of its conceptual simplicity.

Other research has focused on numerical methods such as dynamic programming or nonlinear optimization to minimize cost functions such as actuator energy. One early example of work in this area is by Vukobratović and Kirćanski [191].

9.8 Exercises

Exercise 9.1 Consider an elliptical path in the (x, y)-plane. The path starts at $(0, 0)$ and proceeds clockwise to $(2, 1)$, $(4, 0)$, $(2, -1)$, and back to $(0, 0)$ (Figure 9.15). Write the path as a function of $s \in [0, 1]$.

Exercise 9.2 A cylindrical path in $X = (x, y, z)$ is given by $x = \cos 2\pi s$, $y = \sin 2\pi s$, $z = 2s$, $s \in [0, 1]$, and its time scaling is $s(t) = \frac{1}{4}t + \frac{1}{8}t^2, t \in [0, 2]$. Write down $\dot{X}$ and $\ddot{X}$.

Exercise 9.3 Consider a path from $X(0) = X_{\text{start}} \in SE(3)$ to $X(1) = X_{\text{end}} \in SE(3)$ consisting of motion along a constant screw axis. The path is time scaled by some $s(t)$. Write down the twist $\mathcal{V}$ and acceleration $\dot{\mathcal{V}}$ at any point on the path given $\dot{s}$ and $\ddot{s}$.

Exercise 9.4 Consider a straight-line path $\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}}), s \in [0, 1]$ from $\theta_{\text{start}} = (0, 0)$ to $\theta_{\text{end}} = (\pi, \pi/3)$. The motion starts and ends at rest. The feasible joint velocities are $|\dot{\theta}_1|, |\dot{\theta}_2| \leq 2$ rad/s and the feasible joint accelerations are $|\ddot{\theta}_1|, |\ddot{\theta}_2| \leq 0.5$ rad/s^2. Find the fastest motion time T using a cubic time scaling that satisfies the joint velocity and acceleration limits.

Exercise 9.5 Find the fifth-order polynomial time scaling that satisfies $s(T) = 1$ and $s(0) = \dot{s}(0) = \ddot{s}(0) = \dot{s}(T) = \ddot{s}(T) = 0$.

Exercise 9.6 As a function of the total time of motion T, find the times at which the acceleration $\ddot{s}$ of the fifth-order polynomial point-to-point time scaling is a maximum or a minimum.

Exercise 9.7 If you want to use a polynomial time scaling for point-to-point motion with zero initial and final velocities, accelerations, and jerks, what would be the minimum order of the polynomial?

Exercise 9.8 Prove that the trapezoidal time scaling, using the maximum allowable acceleration a and velocity v, minimizes the time of motion T.

Exercise 9.9 Plot by hand the acceleration profile $\ddot{s}(t)$ for a trapezoidal time scaling.

Exercise 9.10 If v and a are specified for a trapezoidal time scaling of a robot, prove that $v^2/a \leq 1$ is a necessary condition for the robot to reach the maximum velocity v during the path.

Exercise 9.11 If v and T are specified for a trapezoidal time scaling, prove that $vT > 1$ is a necessary condition for the motion to be able to complete in time T. Prove that $vT \leq 2$ is a necessary condition for a three-stage trapezoidal motion.

Exercise 9.12 If a and T are specified for a trapezoidal time scaling, prove that $aT^2 \geq 4$ is a necessary condition to ensure that the motion completes in time.

Exercise 9.13 Consider the case where the maximum velocity v is never reached in a trapezoidal time scaling. The motion becomes a bang-bang motion: constant acceleration a for time $T/2$ followed by constant deceleration $-a$ for time $T/2$. Write down the position $s(t)$, velocity $\dot{s}(t)$, and acceleration $\ddot{s}(t)$ for both phases, in analogy to Equations (9.16)–(9.24).

Exercise 9.14 Plot by hand the acceleration profile $\ddot{s}(t)$ for an S-curve time scaling.

Exercise 9.15 A seven-stage S-curve is fully specified by the time t_J (the duration of a constant positive or negative jerk), the time t_a (the duration of constant positive or negative acceleration), the time t_v (the duration of constant velocity), the total time T, the jerk J, the acceleration a, and the velocity v. Of these seven quantities, how many can be specified independently?

Exercise 9.16 A nominal S-curve has seven stages, but it can have fewer if certain inequality constraints are not satisfied. Indicate which cases are possible with fewer than seven stages. Sketch by hand the $\dot{s}(t)$ velocity profiles for these cases.

Exercise 9.17 If the S-curve achieves all seven stages and uses a jerk J, an acceleration a, and a velocity v, what is the constant-velocity coasting time t_v in terms of v, a, J, and the total motion time T?

Exercise 9.18 Write your own via-point cubic-polynomial interpolation trajectory generator program for a two-dof robot. A new position and velocity specification is required for each joint at 1000 Hz. The user specifies a sequence of via-point positions, velocities, and times, and the program generates an array consisting of the joint angles and velocities at every millisecond from time $t = 0$ to time $t = T$, the total duration of the movement. For a test case with at least three via points (one at the start and and one at the end, both with zero velocity, and at least one more via point), plot
(a) the path in the joint angle space (similar to Figure 9.7), and
(b) the position and velocity of each joint as a function of time (these plots should look similar to Figure 9.8).

Exercise 9.19 Via points with specified positions, velocities, and accelerations can be interpolated using fifth-order polynomials of time. For a fifth-order polynomial segment between via points j and $j+1$, of duration ΔT_j, with β_j, β_{j+1}, $\dot{\beta}_j$, $\dot{\beta}_{j+1}$, $\ddot{\beta}_j$, and $\ddot{\beta}_{j+1}$ specified, solve for the coefficients of the fifth-order polynomial (which is similar to Equations (9.26)–(9.29)). A symbolic math

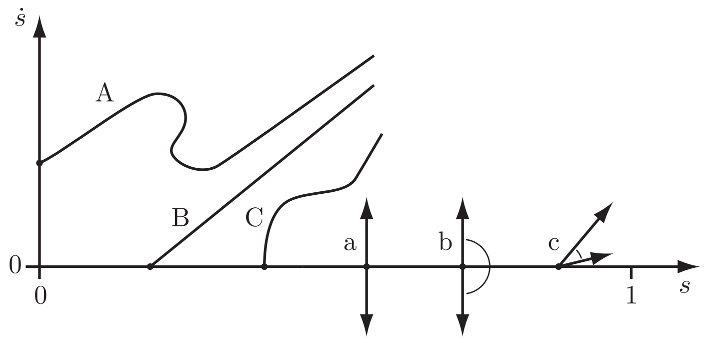

Figure 9.16: A, B, and C are candidate integral curves, originating from the dots indicated, while a, b, and c are candidate motion cones at $\dot{s} = 0$. Two of the integral curves and two of the motion cones are incorrect.

solver will simplify the problem.

Exercise 9.20 By hand or by computer, plot a trapezoidal motion profile in the $(s, \dot{s})$-plane.

Exercise 9.21 Figure 9.16 shows three candidate motion curves in the $(s, \dot{s})$-plane (A, B, and C) and three candidate motion cones at $\dot{s} = 0$ (a, b, and c). Two of the three curves and two of the three motion cones cannot be correct for any robot dynamics. Indicate which are incorrect and explain your reasoning. Explain why the remaining curve and motion cone are possibilities.

Exercise 9.22 Under the assumptions of Section 9.4.4, explain why the time-scaling algorithm of Section 9.4.2 (see Figure 9.13) is correct. In particular,

(a) explain why, in the binary search of step 4, the curve integrated forward from $(s_{\text{lim}}, \dot{s}_{\text{test}})$ must either hit (or run tangent to) the velocity limit curve or hit the $\dot{s} = 0$ axis (and does not hit the curve F, for example);

(b) explain why the final time scaling can only touch the velocity limit curve tangentially; and

(c) explain why the acceleration switches from minimum to maximum at points where the time scaling touches the velocity limit curve.

Exercise 9.23 Explain how the time-scaling algorithm should be modified to handle the case where the initial and final velocities, at $s = 0$ and $s = 1$, are nonzero.

Exercise 9.24 Explain how the time-scaling algorithm should be modified if the robot's actuators are too weak to hold it statically at some configurations of the path (the static-posture-maintenance assumption is violated), but the assumptions on inadmissible states and zero-inertia points are satisfied. Valid time scalings may no longer exist. Under what condition(s) should the algorithm terminate and indicate that no valid time scaling exists? (Under the assumptions of Section 9.4.4 the original algorithm always finds a solution and therefore does not check for failure cases.) What do the motion cones look like at states $(s, \dot{s} = 0)$ where the robot cannot hold itself statically?

Exercise 9.25 Create a computer program that plots the motion cones in the $(s, \dot{s})$-plane for a 2R robot in a horizontal plane. The path is a straight line in joint space from $(\theta_1, \theta_2) = (0, 0)$ to $(\pi/2, \pi/2)$. Use the dynamics from Equation (8.9) (with $g = 0$), then rewrite the dynamics in terms of $s, \dot{s}, \ddot{s}$ instead of $\theta, \dot{\theta}, \ddot{\theta}$. The actuators can provide torques in the range $-\tau_{i,\text{limit}} - b\dot{\theta}_i \leq \tau_i \leq \tau_{i,\text{limit}} - b\dot{\theta}_i$, where $b > 0$ indicates the velocity dependence of the torque. The cones should be drawn at a grid of points in $(s, \dot{s})$. To keep the figure manageable, normalize each cone ray to the same length.

Exercise 9.26 We have been assuming forward motion on a path, $\dot{s} > 0$. What if we allowed backward motion on a path, $\dot{s} < 0$? This exercise involves drawing motion cones and an integral curve in the $(s, \dot{s})$-plane, including both positive and negative values of $\dot{s}$. Assume that the maximum acceleration is $U(s, \dot{s}) = U > 0$ (constant over the $(s, \dot{s})$-plane) and the maximum deceleration is $L(s, \dot{s}) = L = -U$. You can assume, for example, that $U = 1$ and $L = -1$.

(a) For any constant s, draw the motion cones at the five points where $\dot{s}$ takes the values $\{-2, -1, 0, 1, 2\}$.

(b) Assume the motion starts at $(s, \dot{s}) = (0, 0)$ and follows the maximum acceleration U for time t. Then it follows the maximum deceleration L for time $2t$. Then it follows U for time t. Sketch by hand the integral curve. (The exact shape does not matter, but the curve should have the correct features.)