Chapter 13

Trajectory generation

13.1. Introduction

A robotic motion task is specified by defining a path along which the robot must move. A *path* is a sequence of *points* defined either in task coordinates (end-effector coordinates) or in joint coordinates. The issue of trajectory generation is to compute for the control system the desired reference joint or end-effector variables as functions of time such that the robot tracks the desired path. Thus, a *trajectory* refers to a path and a *time history* along the path.

The trajectories of a robot can be classified as follows:

- trajectory between two points with free path between them;
- trajectory between two points via a sequence of desired intermediate points, also called *via points*, with free paths between via points;
- trajectory between two points with constrained path between the points (straight line segment for instance);
- trajectory between two points via intermediate points with constrained paths between the via points.

In the first two classes, the trajectory is generally generated in the joint space. In the last two classes, it is better to generate the trajectory in the task space.

In the next sections, we present trajectory generation techniques related to this classification, but we first analyze the reasons that motivate the choice of either the joint space or the task space for the generation.

13.2. Trajectory generation and control loops

The two approaches to trajectory generation – in the joint space and in the task space – are illustrated in Figures 13.1 and 13.2 (superscripts i and f designate the initial and final values respectively).

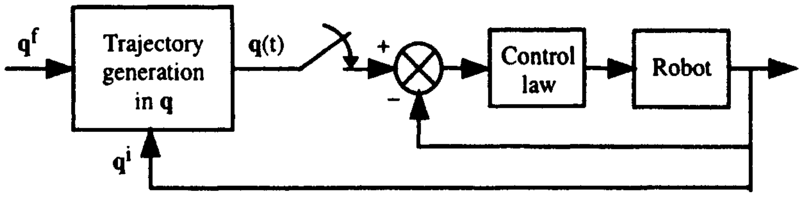

Figure 13.1. *Trajectory generation in the joint space*

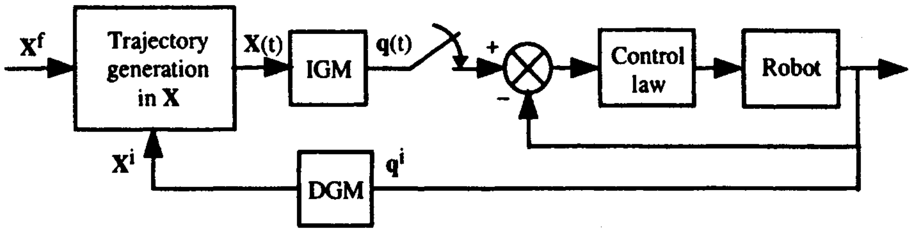

Figure 13.2. *Trajectory generation in the task space*

Trajectory generation in the joint space presents several advantages:

- it requires fewer on-line computations, since there is no need to compute the inverse geometric or kinematic models;
- the trajectory is not affected by crossing singular configurations;
- maximum velocities and torques are determined from actuator data sheets.

The drawback is that the corresponding end-effector path in the task space is not predictable, although it is repetitive, which increases risk of undesirable collisions when the robot works in a cluttered environment. In conclusion, the joint space scheme is appropriate to achieve fast motions in a free space.

Trajectory generation in the task space permits prediction of the geometry of the path. It has, however, a number of disadvantages:

- it may fail when the computed trajectory crosses a singular configuration;
- it fails when the generated points are out of the joint limits or when the robot is forced to change its current aspect (§ 5.7.4);

- velocities and accelerations of the task coordinates depend on the robot configuration. Lower bounds are generally used such that joint velocity and torque limits are satisfied. Consequently, the robot may work under its nominal performance.

The choice of a trajectory generation scheme depends on the application at hand. Each approach has its own limits, due to the fact that constraints are specified either in the joint space (velocities, torques, joint limits), or in the task space (accuracy, geometry of obstacles). Accounting for these remarks, the first two sections cover the problem of trajectory generation between two points in the joint space and the task space respectively. The last section extends the results to trajectory generation between several points.

13.3. Point-to-point trajectory in the joint space

We consider a robot with n degrees of freedom. Let $\mathbf{q}^i$ and $\mathbf{q}^f$ be the joint coordinate vectors corresponding to the initial and final configurations. Let $\mathbf{k}_v$ and $\mathbf{k}_a$ be the vectors of maximum joint velocities and maximum joint accelerations respectively. The value of k_{vj} can be exactly computed from the actuator specifications and transmission ratios, while the value of k_{aj} is approximated by the ratio of the maximum actuator torque to the maximum joint inertia (upper bound of the diagonal element A_{jj} of the inertia matrix defined in Chapter 9). Once the trajectory has been computed with these kinematic constraints, we can proceed to a time scaling in order to better match the maximum joint torques using the dynamic model [Hollerbach 84a].

The trajectory between $\mathbf{q}^i$ and $\mathbf{q}^f$ is determined by the following equation:

$$\mathbf{q}(t) = \mathbf{q}^i + r(t)\,\mathbf{D} \quad \text{for } 0 \leq t \leq t_f \qquad [13.1]$$

$$\dot{\mathbf{q}}(t) = \dot{r}(t)\,\mathbf{D} \qquad [13.2]$$

with $\mathbf{D} = \mathbf{q}^f - \mathbf{q}^i$.

The boundary conditions of the *interpolation function* r(t) are given by:

$$\begin{cases} r(0) = 0 \\ r(t_f) = 1 \end{cases}$$

Equation [13.1] can also be written as:

$$\mathbf{q}(t) = \mathbf{q}^f(t) - [1 - r(t)]\,\mathbf{D} \qquad [13.3]$$

which is more appropriate for tracking moving objects where $\mathbf{q}^f$ is time-varying [Taylor 79]. In this case, $\mathbf{D} = \mathbf{q}^f(0) - \mathbf{q}^i$.

Several interpolation functions can provide a trajectory such that $\mathbf{q}(0) = \mathbf{q}^i$ and $\mathbf{q}(t_f) = \mathbf{q}^f$. We will study successively the polynomial interpolation, the so-called *bang-bang acceleration profile*, and the bang-bang profile with a constant velocity phase termed *trapeze velocity profile*.

13.3.1. *Polynomial interpolation*

The most commonly used polynomials are the linear interpolation, the third degree polynomials (cubic) and the fifth degree polynomials (quintic).

13.3.1.1. *Linear interpolation*

The trajectory of each joint is described by a linear equation in time. The equation of the joint position is written as:

$$\mathbf{q}(t) = \mathbf{q}^i + \frac{t}{t_f}\mathbf{D} \qquad [13.4]$$

With this trajectory, the position is continuous but not the velocity. This induces undesirable vibrations on the robot and may cause early wear and tear of the mechanical parts.

13.3.1.2. *Cubic polynomial*

If the initial and final velocities are also set to zero, the minimum degree of the polynomial satisfying the constraints is at least three, and has the form:

$$\mathbf{q}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \mathbf{a}_3 t^3 \qquad [13.5]$$

The coefficients $\mathbf{a}_i$ are determined from the following boundary conditions:

$$\begin{cases} a_0 = q^i \\ a_1 = 0 \\ a_2 = \dfrac{3}{t_f^2} D \\ a_3 = -\dfrac{2}{t_f^3} D \end{cases} \qquad [13.6]$$

The expression [13.5] can also be written under the form [13.1] or [13.3] with the following interpolation function:

$$r(t) = 3 \left(\frac{t}{t_f}\right)^2 - 2 \left(\frac{t}{t_f}\right)^3 \qquad [13.7]$$

The cubic polynomial ensures the continuity of velocity but not of acceleration. Practically, the industrial robots are sufficiently rigid so that this discontinuity is filtered by the mechanical structure. Therefore, such a trajectory is generally satisfactory for most applications.

Figure 13.3 shows the position, velocity and acceleration profiles for joint j. The velocity is maximum at $t = t_f / 2$ and its magnitude is given by:

$$|\dot{q}_{jmax}| = \frac{3|D_j|}{2t_f} \quad \text{with } |D_j| = |q_j^f - q_j^i| \qquad [13.8]$$

The maximum acceleration occurs at $t = 0$ and $t = t_f$ with the magnitude:

$$|\ddot{q}_{jmax}| = \frac{6|D_j|}{t_f^2} \qquad [13.9]$$

13.3.1.3. *Quintic polynomial*

For high speed robots or when a robot is handling heavy or delicate loads, it is worth ensuring the continuity of accelerations as well, in order to avoid exciting resonances in the mechanics. The trajectory is said to be of class C^2. Since six constraints have to be satisfied, the interpolation requires a polynomial of at least fifth degree [Binford 77]. The additional two constraints are written as:

$$\begin{cases} \ddot{q}(0) = 0 \\ \ddot{q}(t_f) = 0 \end{cases} \qquad [13.10]$$

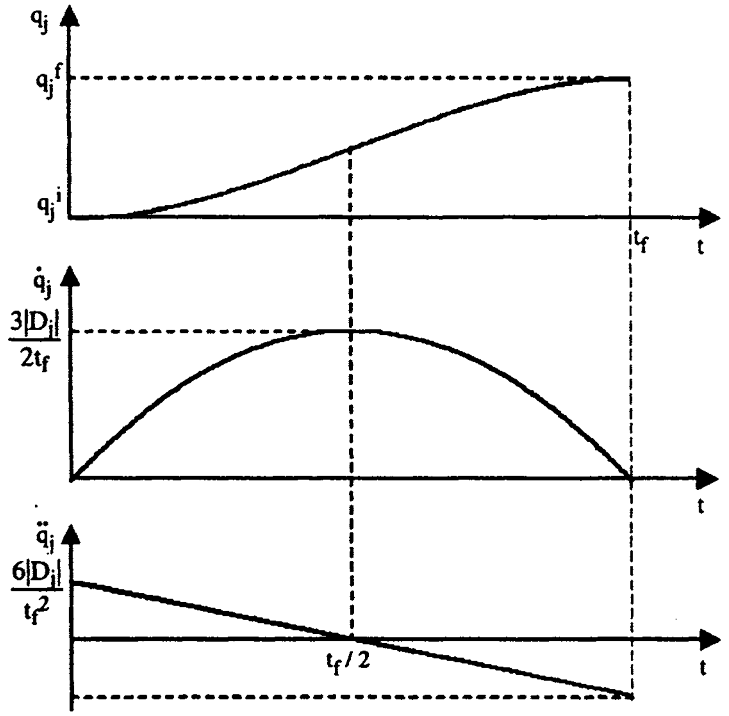

Figure 13.3. *Position, velocity and acceleration profiles for a cubic polynomial*

Solving for the six constraints yields the following interpolation function:

$$r(t) = 10\left(\frac{t}{t_f}\right)^3 - 15\left(\frac{t}{t_f}\right)^4 + 6\left(\frac{t}{t_f}\right)^5 \qquad [13.11]$$

The position, velocity and acceleration with respect to time for joint j are plotted in Figure 13.4. Maximum velocity and acceleration are given by:

$$|\dot{q}_{jmax}| = \frac{15\,|D_j|}{8\,t_f} \qquad [13.12]$$

$$|\ddot{q}_{jmax}| = \frac{10\,|D_j|}{\sqrt{3}\,t_f^2} \qquad [13.13]$$

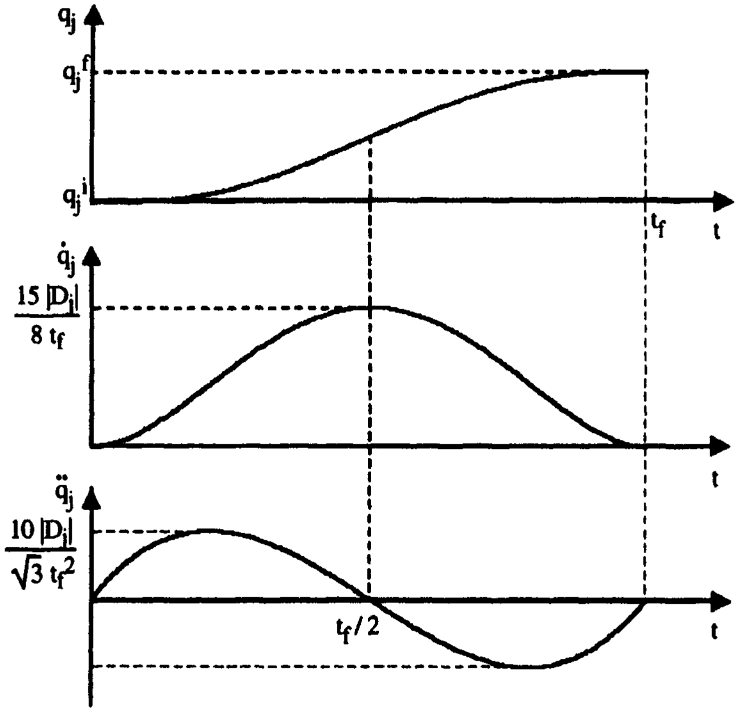

Figure 13.4. *Position, velocity and acceleration profiles for a quintic polynomial*

13.3.1.4. *Computation of the minimum traveling time*

Generally, the duration t_f of a trajectory is not specified. The goal is to minimize the time to travel from the initial configuration q^i to the final one q^f while satisfying velocity and acceleration constraints. The approach is to compute the minimum time separately for each joint, and then to synchronize all the joints at a common time.

The minimum traveling time t_{fj} for joint j occurs if either the velocity or the acceleration is saturated during the trajectory. This minimum time is computed from the maximum magnitudes of velocity and acceleration of the different polynomial interpolation functions (Table 13.1). The global minimum traveling time t_f is equal to the largest minimum time:

$$t_f = \max(t_{f1}, \ldots, t_{fn}) \qquad [13.14]$$

Table 13.1. *Minimum traveling time for joint j*

Interpolation function	Minimum time
Linear interpolation	$t_{fj} = \frac{\lvert D_j \rvert}{k_{vj}}$
Cubic polynomial	$t_{fj} = \max\left[\frac{3\,\lvert D_j \rvert}{2\,k_{vj}}, \sqrt{\frac{6\,\lvert D_j \rvert}{k_{aj}}}\right]$
Quintic polynomial	$t_{fj} = \max\left[\frac{15\,\lvert D_j \rvert}{8\,k_{vj}}, \sqrt{\frac{10\,\lvert D_j \rvert}{\sqrt{3}\,k_{aj}}}\right]$
Bang-bang profile (§ 13.3.2)	$t_{fj} = \max\left[\frac{2\,\lvert D_j \rvert}{k_{vj}}, 2\sqrt{\frac{\lvert D_j \rvert}{k_{aj}}}\right]$

13.3.2. *Bang-bang acceleration profile*

A bang-bang acceleration profile consists of a constant acceleration phase until $t_f/2$ followed by a constant deceleration phase (Figure 13.5). The initial and final velocities are zero. Thus, the trajectory is continuous in position and velocity, but discontinuous in acceleration.

The position is given by:

$$\begin{cases} \mathbf{q}(t) = \mathbf{q}^i + 2\left(\frac{t}{t_f}\right)^2 \mathbf{D} & \text{for } 0 \le t \le \frac{t_f}{2} \\ \mathbf{q}(t) = \mathbf{q}^i + \left[-1 + 4\left(\frac{t}{t_f}\right) - 2\left(\frac{t}{t_f}\right)^2\right] \mathbf{D} & \text{for } \frac{t_f}{2} \le t \le t_f \end{cases} \quad [13.15]$$

For joint j, the maximum velocity and acceleration are given by:

$$\lvert \dot{q}_{j\max} \rvert = \frac{2\,\lvert D_j \rvert}{t_f} \quad [13.16]$$

$$\lvert \ddot{q}_{j\max} \rvert = \frac{4\,\lvert D_j \rvert}{t_f^2} \quad [13.17]$$

The minimum traveling time is obtained as for a polynomial trajectory (Table 13.1).

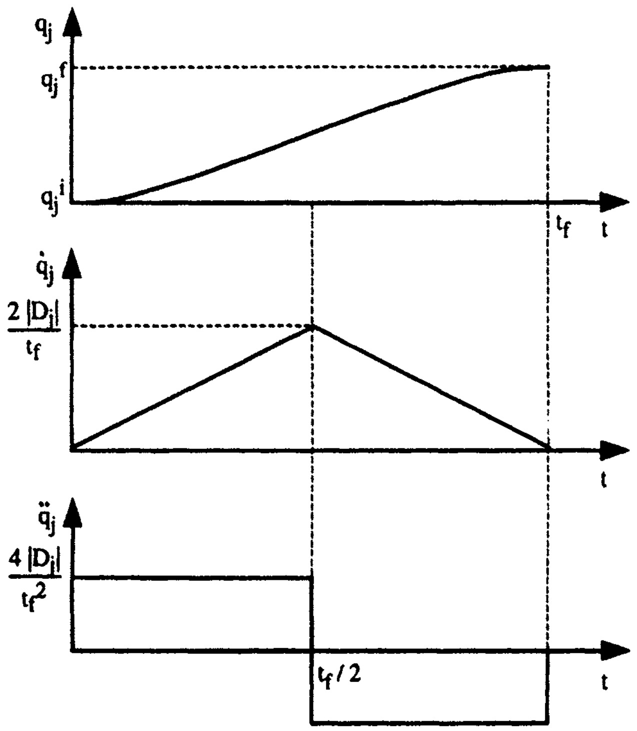

Figure 13.5. *Bang-bang velocity profile*

13.3.3. *Trapeze velocity profile*

With a bang-bang profile, when the velocity reaches its maximum, adding a constant velocity phase would allow us to saturate also the acceleration and to minimize the traveling time (Figure 13.6). According to equations [13.16] and [13.17], the condition to have a constant velocity phase on joint j is such that:

$$|D_j| > \frac{k_{vj}^2}{k_{aj}} \qquad [13.18]$$

The trapeze velocity trajectory results in the minimum traveling time among those providing the continuity of velocity. The joint j trajectory (Figure 13.7) is represented by the following equations:

$$\begin{cases} q_j(t) = q_j^i + \frac{1}{2} t^2 k_{aj} \operatorname{sign}(D_j) & \text{for } 0 \le t \le \tau_j \\ q_j(t) = q_j^i + (t - \frac{\tau_j}{2}) k_{vj} \operatorname{sign}(D_j) & \text{for } \tau_j \le t \le t_{fj} - \tau_j \\ q_j(t) = q_j^f - \frac{1}{2} (t_{fj} - t)^2 k_{aj} \operatorname{sign}(D_j) & \text{for } t_{fj} - \tau_j \le t \le t_{fj} \end{cases} \qquad [13.19]$$

with:

$$\tau_j = \frac{k_{vj}}{k_{aj}} \qquad [13.20]$$

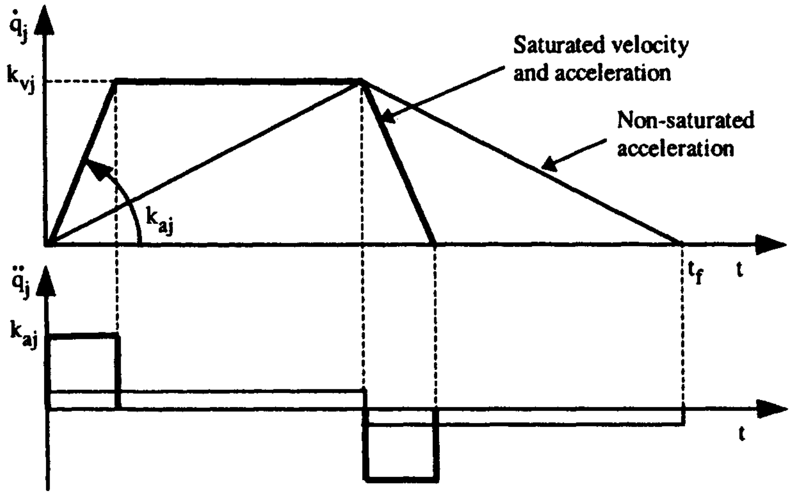

Figure 13.6. *Trapeze velocity profile versus bang-bang acceleration profile*

The area of the trapeze under the velocity curve is equal to the distance traveled in the interval $[0, t_{fj}]$, which can be written as:

$$|D_j| = |q_j^f - q_j^i| = 2\int_0^{\tau_j} k_{aj}\, t\, dt + \int_{\tau_j}^{t_{fj}-\tau_j} k_{vj} dt = k_{vj}\, t_{fj} - \frac{k_{vj}^2}{k_{aj}} \qquad [13.21]$$

Hence, the minimum time for joint j is given by:

$$t_{fj} = \frac{k_{vj}}{k_{aj}} + \frac{|D_j|}{k_{vj}} = \tau_j + \frac{|D_j|}{k_{vj}} \qquad [13.22]$$

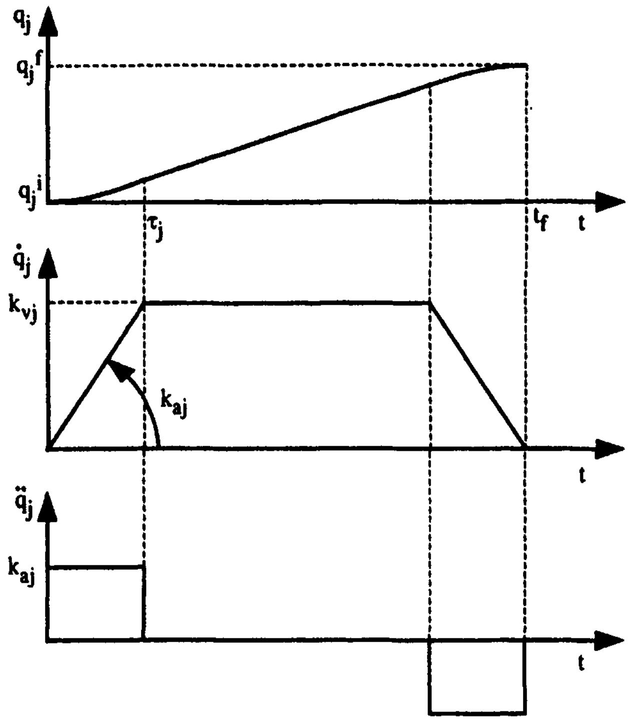

Figure 13.7. *Position, velocity and acceleration profiles for a trapeze trajectory*

In order to synchronize the joint trajectories, we present in the following a method giving homothetical trajectories with the same acceleration and deceleration duration for all joints. Such a method is the most common in use in industrial robot controllers. Let us designate by α_j the ratio between the velocity profile of joint j and an arbitrary joint k. We can write that (Figure 13.8):

$$\dot{q}_j(t) = \alpha_j \, \dot{q}_k(t) \quad \text{for } j = 1, \ldots, n \qquad [13.23]$$

Doing this, the duration τ of the acceleration phase of the synchronized trajectories is *a priori* not equal to any optimal τ_j computed for each joint separately (equation [13.20]).

Let $\lambda_j \, k_{vj}$ be the maximum velocity of the synchronized trajectory for joint j and let $\upsilon_j \, k_{aj}$ be the corresponding maximum acceleration. To calculate the optimal τ, resulting in a minimum time t_f, let us first solve the problem for two joints. According to equation [13.22], the minimum traveling time for each joint, if calculated separately, should be:

$$\begin{cases} t_{f1} = \tau_1 + \dfrac{|D_1|}{k_{v1}} \\ t_{f2} = \tau_2 + \dfrac{|D_2|}{k_{v2}} \end{cases} \qquad [13.24]$$

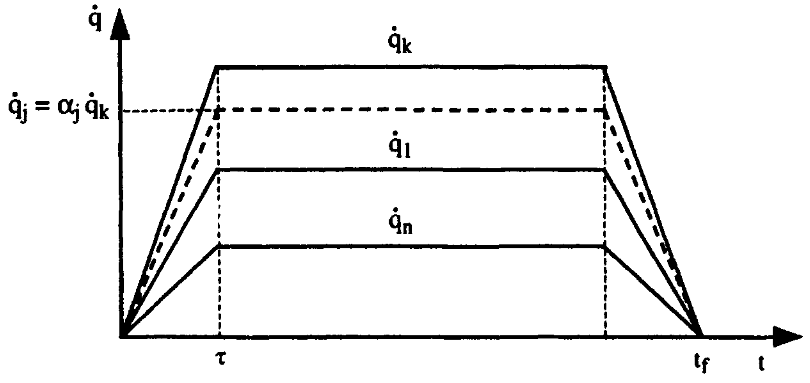

Figure 13.8. *Homothetical velocity profiles*

The synchronized trajectories should be such that:

$$t_f = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} + \frac{|D_1|}{\lambda_1 k_{v1}} = \frac{\lambda_2 k_{v2}}{\upsilon_2 k_{a2}} + \frac{|D_2|}{\lambda_2 k_{v2}} \qquad [13.25]$$

with $t_f \geq \max(t_{f1}, t_{f2})$.

From equation [13.25], it is straightforward to obtain:

$$\tau = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} = \frac{\lambda_2 k_{v2}}{\upsilon_2 k_{a2}} \qquad [13.26]$$

$$\lambda_2 = \lambda_1 \frac{k_{v1}|D_2|}{k_{v2}|D_1|} \qquad [13.27]$$

$$\upsilon_2 = \upsilon_1 \frac{k_{a1}|D_2|}{k_{a2}|D_1|} \qquad [13.28]$$

The velocity constraints imply that:

$$\begin{cases} 0 \le \lambda_1 \le 1 \\ 0 \le \lambda_2 \le 1 \end{cases} \qquad [13.29]$$

Combining the last inequality with equation [13.27] yields:

$$0 \le \lambda_1 \le \frac{k_{v2}|D_1|}{k_{v1}|D_2|} \qquad [13.30]$$

Likewise, from the acceleration constraints, we get:

$$\begin{cases} 0 \le \upsilon_1 \le 1 \\ 0 \le \upsilon_1 \le \dfrac{k_{a2}|D_1|}{k_{a1}|D_2|} \end{cases} \qquad [13.31]$$

The minimum time t_f is obtained when the parameters λ_1 and υ_1 are the largest and satisfy simultaneously the above constraints, which results in:

$$\begin{cases} \lambda_1 = \min\left[1, \dfrac{k_{v2}|D_1|}{k_{v1}|D_2|}\right] \\ \upsilon_1 = \min\left[1, \dfrac{k_{a2}|D_1|}{k_{a1}|D_2|}\right] \end{cases} \qquad [13.32]$$

and the corresponding duration of the acceleration phase is:

$$\tau = \frac{\lambda_1 k_{v1}}{\upsilon_1 k_{a1}} \qquad [13.33]$$

These equations are easily generalized for n joints:

$$\begin{cases} \lambda_1 = \min\left[1, \dfrac{k_{vj}|D_1|}{k_{v1}|D_j|}\right] \\ \upsilon_1 = \min\left[1, \dfrac{k_{aj}|D_1|}{k_{a1}|D_j|}\right] \end{cases} \quad \text{for } j = 2, \ldots, n \qquad [13.34]$$

assuming that $D_1 \neq 0$ and $D_j \neq 0$.

NOTE.– If, for a given joint j, the distance $|D_j|$ is such that the maximum velocity k_{vj} cannot be attained, we replace in the above formulas the term k_{vj} by the maximum achievable velocity. According to equation [13.18], this occurs when:

$$|D_j| < \frac{k_{vj}^2}{k_{aj}}$$

which implies that the maximum achievable velocity is:

$$k'_{vj} = \sqrt{|D_j|\,k_{aj}} \qquad [13.35]$$

13.3.4. *Continuous acceleration profile with constant velocity phase*

We can modify the previous method to have a continuous trajectory in acceleration by replacing the acceleration and deceleration phases either by a second degree polynomial (Figure 13.9a) or by a trapeze acceleration profile (Figure 13.9b) [Castain 84]. In the following, we detail the first approach, which is simpler to implement. Let τ' be the new duration of the acceleration and let $\lambda_j\, k_{vj}$ be the maximum velocity of the trapeze profile. The boundary conditions for joint j are defined as:

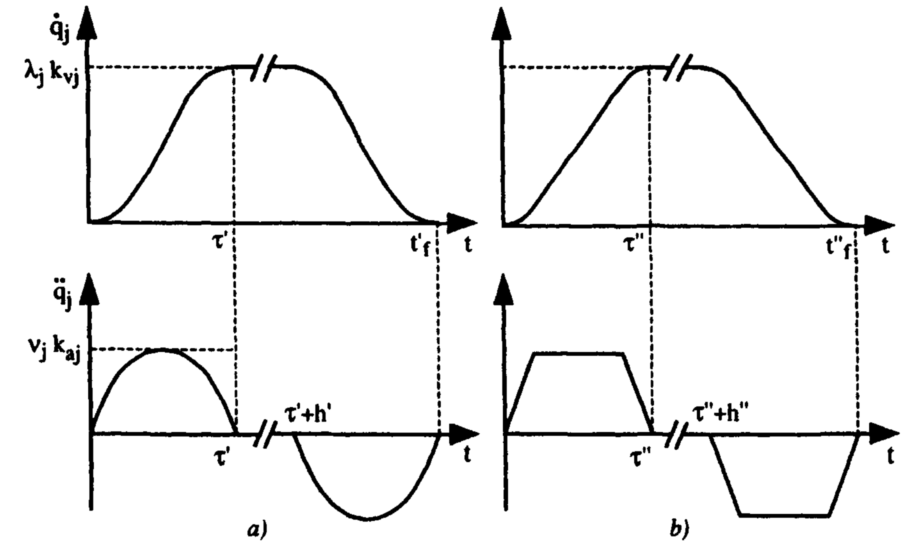

Figure 13.9. *Modification of the acceleration of the trapeze profile to ensure a continuous acceleration*

$$\begin{cases} q_j(0) = q_j^i \\ \dot{q}_j(0) = 0 \\ \dot{q}_j(\tau') = \lambda_j\, k_{vj}\, \text{sign}(D_j) \\ \ddot{q}_j(0) = 0 \\ \ddot{q}_j(\tau') = 0 \end{cases} \qquad [13.36]$$

From these constraints, we derive the equations of position, velocity and acceleration of joint j for $1 \le j \le n$ as $0 \le t \le \tau'$ as follows:

$$q_j(t) = q_j^i - \frac{1}{\tau'^3}\lambda_j\, k_{vj}\, \text{sign}(D_j)\left(\frac{1}{2}t - \tau'\right) t^3 \qquad [13.37]$$

$$\dot{q}_j(t) = -\frac{1}{\tau'^3}\lambda_j\, k_{vj}\, \text{sign}(D_j)\,(2t - 3\tau')\, t^2 \qquad [13.38]$$

$$\ddot{q}_j(t) = -\frac{6}{\tau'^3}\lambda_j\, k_{vj}\, \text{sign}(D_j)\,(t - \tau')\, t \qquad [13.39]$$

The acceleration is maximum at $t = \tau'/2$ and its magnitude is:

$$|\ddot{q}_{jmax}| = \frac{3}{2}\frac{\lambda_j\, k_{vj}}{\tau'} \qquad [13.40]$$

If we take for $|\ddot{q}_{jmax}|$ the value $\upsilon_j\, k_{aj}$ of the velocity trapeze profile, all the joints have the same synchronized duration of acceleration such that:

$$\tau' = \frac{3}{2}\frac{\lambda_j\, k_{vj}}{\upsilon_j\, k_{aj}} \qquad [13.41]$$

Hence, the duration of the acceleration phase is 1.5 times that with a constant acceleration. The joint position equation corresponding to the constant velocity phase, given a duration h', is as follows:

$$q_j(t) = q_j(\tau') + (t - \tau')\,\lambda_j\, k_{vj}\, \text{sign}(D_j) \quad \text{for } \tau' \le t \le \tau' + h' \qquad [13.42]$$

Assuming that the acceleration and deceleration phases are symmetrical ($t'_f = 2\tau' + h'$), the trajectory corresponding to the deceleration phase is defined in the interval $\tau' + h' \leq t \leq t'_f$ as:

$$\begin{cases} q_j(t) = q_j^f + \frac{1}{2}\left[\frac{1}{\tau'^3}(t-3\tau'-h')(t-\tau'-h')^3 + (2t-3\tau'-2h')\right] \lambda_j k_{vj} \text{sign}(D_j) \\ \dot{q}_j(t) = \left[\frac{1}{\tau'^3}(2t-5\tau'-2h')(t-\tau'-h')^2 + 1\right] \lambda_j k_{vj} \text{sign}(D_j) \\ \ddot{q}_j(t) = \frac{6}{\tau'^3}(t-2\tau'-h')(t-\tau'-h') \lambda_j k_{vj} \text{sign}(D_j) \end{cases} \quad [13.43]$$

According to equations [13.37] and [13.41], it should be noted that the distance traveled during the acceleration phase is equal to:

$$|q_j^i - q_j(\tau')| = \frac{3}{4}\frac{(\lambda_j k_{vj})^2}{\upsilon_j k_{aj}} \quad [13.44]$$

By computing the area under the velocity curve, we verify that:

$$t'_f = \tau' + \frac{|D_j|}{\lambda_j k_{vj}} \quad [13.45]$$

This expression is similar to equation [13.22] giving the traveling time for the trapeze profile, which suggests that the computation of λ_j and υ_j can be achieved with equations [13.32]. We note as well that to saturate the velocity and the acceleration of a joint trajectory, the distance to travel must be such that:

$$|D_j| > \frac{3}{2}\frac{k_{vj}^2}{k_{aj}} \quad [13.46]$$

If this condition is not verified, we replace k_{vj} in equations [13.34] and [13.36] by the maximum achievable velocity:

$$k'_{vj} = \sqrt{\frac{2}{3}|D_j|\, k_{aj}} \quad [13.47]$$

13.4. Point-to-point trajectory in the task space

Let ${}^0T_E^i$ and ${}^0T_E^f$ be the homogeneous transformations describing the initial and final desired locations respectively. For convenience, let us note:

$$ {}^0T_E^i = \begin{bmatrix} & A^i & & P^i \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } {}^0T_E^f = \begin{bmatrix} & A^f & & P^f \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The most common way to move from one location to the other is to split the motion into a linear translation between the origins of frames ${}^0T_E^i$ and ${}^0T_E^f$, and a rotation α around an axis of the end-effector ${}^E\mathbf{u}$ to align A^i and A^f. The translation and the rotation should be synchronized.

The distance to travel for the translation is obtained as:

$$ D = \|P^f - P^i\| = \sqrt{(P_x^f - P_x^i)^2 + (P_y^f - P_y^i)^2 + (P_z^f - P_z^i)^2} \qquad [13.48] $$

The terms $\mathbf{u}$ and α are computed from the equation:

$$ A^i \,\mathbf{rot}(\mathbf{u}, \alpha) = A^f \qquad [13.49] $$

where we recall that $\mathbf{rot}(\mathbf{u}, \alpha)$ is a (3x3) rotation matrix corresponding to a rotation of an angle α about a vector $\mathbf{u}$. Hence, we get:

$$ \mathbf{rot}(\mathbf{u}, \alpha) = [A^i]^T A^f = \begin{bmatrix} \mathbf{s}^{iT} \\ \mathbf{n}^{iT} \\ \mathbf{a}^{iT} \end{bmatrix} [\ \mathbf{s}^f \quad \mathbf{n}^f \quad \mathbf{a}^f\ ] = \begin{bmatrix} \mathbf{s}^i.\mathbf{s}^f & \mathbf{s}^i.\mathbf{n}^f & \mathbf{s}^i.\mathbf{a}^f \\ \mathbf{n}^i.\mathbf{s}^f & \mathbf{n}^i.\mathbf{n}^f & \mathbf{n}^i.\mathbf{a}^f \\ \mathbf{a}^i.\mathbf{s}^f & \mathbf{a}^i.\mathbf{n}^f & \mathbf{a}^i.\mathbf{a}^f \end{bmatrix} \qquad [13.50] $$

the symbol "." designating the dot product. Using equations [2.34] through [2.37] yields:

$$ \begin{cases} C\alpha = \frac{1}{2}[\mathbf{s}^i.\mathbf{s}^f + \mathbf{n}^i.\mathbf{n}^f + \mathbf{a}^i.\mathbf{a}^f - 1] \\ S\alpha = \frac{1}{2}\sqrt{(\mathbf{a}^i.\mathbf{n}^f - \mathbf{n}^i.\mathbf{a}^f)^2 + (\mathbf{s}^i.\mathbf{a}^f - \mathbf{a}^i.\mathbf{s}^f)^2 + (\mathbf{n}^i.\mathbf{s}^f - \mathbf{s}^i.\mathbf{n}^f)^2} \\ \alpha = \operatorname{atan2}(S\alpha, C\alpha) \\ \mathbf{u} = \frac{1}{2S\alpha}\begin{bmatrix} \mathbf{a}^i.\mathbf{n}^f - \mathbf{n}^i.\mathbf{a}^f \\ \mathbf{s}^i.\mathbf{a}^f - \mathbf{a}^i.\mathbf{s}^f \\ \mathbf{n}^i.\mathbf{s}^f - \mathbf{s}^i.\mathbf{n}^f \end{bmatrix} \end{cases} \qquad [13.51] $$

When $S\alpha$ is small, the vector $\mathbf{u}$ is computed as indicated in § 2.3.8.

Let k_{v1} and k_{a1} be the maximum velocity and acceleration for the translation motion, and let k_{v2} and k_{a2} be the maximum velocity and acceleration for the rotation motion. The methods described in § 13.3 can be used to generate a synchronized trajectory for the two variables D and α, resulting in the minimum time t_f. The trajectory of the end-effector frame is given by:

$$^0\mathbf{T}_E(t) = \begin{bmatrix} \multicolumn{3}{c}{\mathbf{A}(t)} & \mathbf{P}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad [13.52]$$

with:

$$\mathbf{P}(t) = \mathbf{P}^i + \frac{s(t)}{D}(\mathbf{P}^f - \mathbf{P}^i) = \mathbf{P}^i + r(t)\,(\mathbf{P}^f - \mathbf{P}^i) \qquad [13.53]$$

$$\mathbf{A}(t) = \mathbf{A}^i\,\mathbf{rot}(\mathbf{u}, r(t)\,\alpha) \qquad [13.54]$$

where s(t) = D r(t) is the curvilinear distance traveled at time t and r(t) is the interpolation function.

NOTES.–

– we can specify the rotation from $\mathbf{A}^i$ to $\mathbf{A}^f$ with the three Euler angles ϕ, θ and ψ. Let $(\phi^i, \theta^i, \psi^i)$ and $(\phi^f, \theta^f, \psi^f)$ designate the Euler angles corresponding to $\mathbf{A}^i$ and $\mathbf{A}^f$ respectively. Thus, equation [13.54] is replaced by:

$$\mathbf{A}(t) = \mathbf{A}^i\,\mathbf{rot}(\mathbf{z}, \phi^i + r(t)\,\phi)\,\mathbf{rot}(\mathbf{x}, \theta^i + r(t)\,\theta)\,\mathbf{rot}(\mathbf{z}, \psi^i + r(t)\,\psi) \qquad [13.55]$$

with $\phi = \phi^f - \phi^i$, $\theta = \theta^f - \theta^i$, $\psi = \psi^f - \psi^i$. The computation of ϕ, θ and ψ can be carried out as described in § 3.6.1. Thus, we have to deal with four variables: D, ϕ, θ and ψ;

– we can also choose to specify the rotation around an axis that is fixed with respect to frame R_0. In this case, $\mathbf{u}$ and α are calculated by solving:

$$\mathbf{rot}(\mathbf{u}, \alpha)\,\mathbf{A}^i = \mathbf{A}^f \qquad [13.56]$$

– the angular velocity $\boldsymbol{\omega}$ of the end-effector, with respect to the frame where $\mathbf{u}$ is defined, is such that:

$$\boldsymbol{\omega} = \mathbf{u}\,\dot{r}(t)\,\alpha = w\,\mathbf{u} \qquad [13.57]$$

13.5. Trajectory generation with via points

We now consider the problem of generating a trajectory when the path includes via points. These via points are inserted between the initial and final points in order to avoid collisions between the robot and its environment. Passing through the via points without stopping reduces the traveling time.

For each variable (joint or Cartesian), we can calculate a single polynomial passing through these points and satisfying the boundary conditions. However, the use of such a polynomial is difficult to exploit with increasing the number of points. Splitting the trajectory in low degree polynomials between the path points provides an elegant way of overcoming this problem and reduces the computational burden of trajectory generation.

In this section, we present three methods based on this principle. The first method consists of specifying linear interpolations with continuous acceleration blends; in the second method, the trajectory between two consecutive points is interpolated by a cubic spline providing continuity of velocity and acceleration; in the third method, the path generation is totally decoupled from the specification of the time history along the path, which gives the possibility of modifying at run-time the velocity of the robot while tracking the desired path.

13.5.1. *Linear interpolations with continuous acceleration blends*

This method can be used for both trajectory generation schemes in the joint space and in the task space. The via points are connected by straight line segments at constant velocity, and the segments are connected around each via point by continuous acceleration motions. This approach was initially described in [Taylor 79], [Paul 81]. The trajectory can be computed on-line, by only looking ahead at a single point. Experimental methods to identify this type of trajectories on a Puma robot have been proposed by [Blanchon 87], [Tondu 94], [Douss 96].

13.5.1.1. *Joint space scheme*

Let the path be represented by the configurations $\mathbf{q}^1$, $\mathbf{q}^2$, ..., $\mathbf{q}^{m-1}$, $\mathbf{q}^m$. First, according to the method presented in § 13.3.3, we compute the terms λ_j^k, υ_j^k and τ_k' for the segment k between points $\mathbf{q}^k$ and $\mathbf{q}^{k+1}$, for $k = 1, ..., m-1$, assuming zero velocity at the points.

The constant velocity on the segment k, denoted by $\dot{q}_j^k$, is such that:

$$\dot{q}_j^k = \lambda_j^k k_{vj} \operatorname{sign}(D_j^k) \quad \text{for } j = 1, ..., n \qquad [13.58]$$

with $D_j^k = q_j^{k+1} - q_j^k$.

This velocity allows us, if necessary, to stop at point k + 1 without overshooting. If we assume a constant velocity along the segment k (Figure 13.10), the common traveling time h_k on this segment is given by:

$$h_k = \frac{D_j^k}{\dot{q}_j^k} \quad \text{for } j = 1, \ldots, n \qquad [13.59]$$

To generate a smooth trajectory without stopping at the via points, we connect the trajectories at segments k – 1 and k by a blend (Figure 13.10). The duration of the blend region at point k is equal to $2T_k$. If a velocity continuity is only satisfactory, we can specify a constant acceleration along the blend. Otherwise, it is necessary to use a second degree function providing acceleration continuity. We describe here such a solution, which is a generalization to the one used for a point-to-point trajectory (equations [13.36] through [13.44]). The blend region is traveled at maximum acceleration for each joint in order that the obtained path is as close as possible to the via point. As will be verified further, the blend time is given by:

$$T_{k,j} = \frac{3}{4}\frac{|\dot{q}_j^k - \dot{q}_j^{k-1}|}{k_{aj}} \quad \text{for } k = 2, \ldots, m-1 \text{ and } j = 1, \ldots, n \qquad [13.60]$$

Thus, the blend times are not identical for all joints. The joints are only synchronized at the blends around the initial and final points where we can use equation [13.41] giving $T_{k,j} = \tau_k'/2$.

Let $q_{j,a}^k$ and $q_{j,b}^k$ be the positions of joint j at the beginning and at the end of the blend region around point k respectively (Figure 13.11):

$$\begin{cases} q_{j,a}^k = q_j^k - T_{k,j}\, \dot{q}_j^{k-1} \\ q_{j,b}^k = q_j^k + T_{k,j}\, \dot{q}_j^k \end{cases} \quad \text{for } k = 1, \ldots, m \qquad [13.61]$$

For convenience, $T_{k,j}$ will be denoted by T_k. The equation of the linear motion of joint j at segment k (Figure 13.10) is given by:

$$q_j(t) = (t - t_k - T_k)\,\dot{q}_j^k + q_j^k \quad \text{for } t_k + 2T_k \le t \le t_{k+1} \qquad [13.62]$$

with:

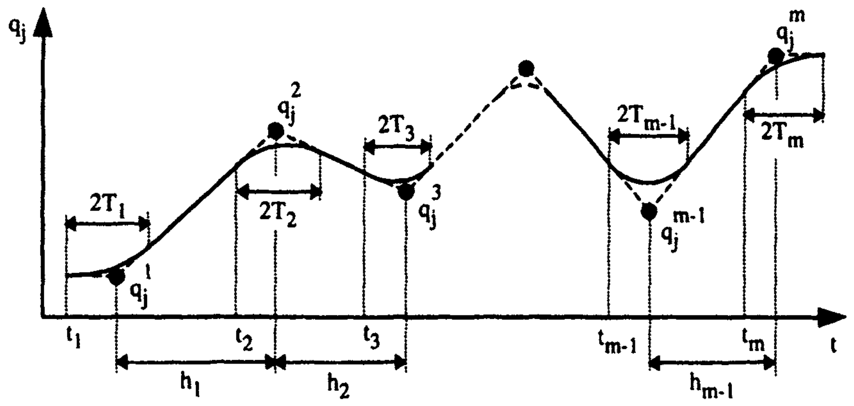

Figure 13.10. *Linear interpolations with continuous acceleration blends (joint space)*

$$\begin{cases} t_k = T_1 - T_k + \sum_{i=1}^{k-1} (h_i) \\ t_1 = 0 \end{cases} \quad \text{for } k = 2, \ldots, m \qquad [13.63]$$

We now consider the blend region around point k. Writing the boundary conditions gives:

$$\begin{cases} q_j(t_k) = q_{j,a}^k \\ q_j(t_k + 2T_k) = q_{j,b}^k \\ \dot{q}_j(t_k) = \dot{q}_j^{k-1} \\ \dot{q}_j(t_k + 2T_k) = \dot{q}_j^k \\ \ddot{q}_j(t_k) = 0 \\ \ddot{q}_j(t_k + 2T_k) = 0 \end{cases} \quad \text{for } k = 1, \ldots, m \qquad [13.64]$$

with $q_j(t_1) = q_j^1 = q_{j,a}^1$ and $q_j(t_m + 2T_m) = q_j^m = q_{j,b}^m$.

Due to the symmetry, the trajectory of joint j along the blend segment k, when it exists ($T_k \neq 0$), is given by the following fourth degree polynomial:

$$q_j(t) = q_j^k - \frac{1}{16(T_k)^3}(t - t_k)^3 (t - t_k - 4T_k)(\dot{q}_j^k - \dot{q}_j^{k-1}) + (t - t_k - T_k)\, \dot{q}_j^{k-1} \qquad [13.65]$$

with $t_k \le t \le t_k + 2T_k$ and $1 \le k \le m$.

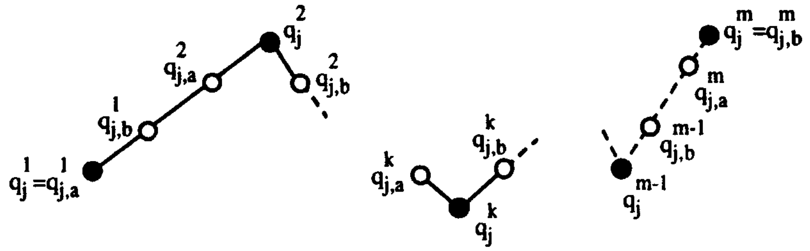

Figure 13.11. *Notations*

The corresponding velocity and acceleration equations are as follows:

$$\dot{q}_j(t) = \dot{q}_j^{k-1} - \frac{1}{4(T_k)^3}(t-t_k)^2(t-t_k-3T_k)(\dot{q}_j^k - \dot{q}_j^{k-1}) \qquad [13.66]$$

$$\ddot{q}_j(t) = -\frac{3}{4(T_k)^3}(t-t_k)(t-t_k-2T_k)(\dot{q}_j^k - \dot{q}_j^{k-1}) \qquad [13.67]$$

The acceleration is maximum at $t = t_k + T_k$ and has the magnitude:

$$|\ddot{q}_j(t)_{max}| = \frac{3}{4T_k}|\dot{q}_j^k - \dot{q}_j^{k-1}| \qquad [13.68]$$

This expression has been used to calculate T_k in terms of the maximum acceleration k_{aj} (equation [13.60]).

NOTES.–

– it is mandatory that $h_k \ge T_k + T_{k+1}$. If this condition does not hold, the velocity at segment k + 1 should be scaled down, or even set to zero at point k + 1;

– the maximum error around the via point k for joint j is given by:

$$E_j = |q_j^k(t=t_k+T_k) - q_j^k| = \frac{9}{64}\frac{(\dot{q}_j^k - \dot{q}_j^{k-1})^2}{k_{aj}} \qquad [13.69]$$

which justifies high acceleration value to minimize E_j;

– if it is possible to look ahead at several via points, the value of the constant velocity at each segment may be scaled up.

13.5.1.2. *Task space scheme*

The path in the task space is defined by a sequence of end-effector locations ${}^0T_E^1$, ..., ${}^0T_E^m$. The results obtained in the joint space may be extended in the task space after splitting the trajectory into:

- a translation between ${}^0P_E^k$ and ${}^0P_E^{k+1}$, represented by the distance to travel $D^k = \|{}^0P_E^{k+1} - {}^0P_E^k\|$;
- and a rotation represented by the three Euler angles to move from $\boldsymbol{\Theta}^k$ to $\boldsymbol{\Theta}^{k+1}$ (where $\boldsymbol{\Theta}^k = [\ \phi^k\ \ \theta^k\ \ \psi^k\ ]^T$ represents the Euler angles corresponding to ${}^0A_E^k$).

As in the joint space, we first calculate the trajectory parameters at each segment, assuming zero velocity at the via points according to the method presented in § 13.3.3. We thus obtain λ_j^k, υ_j^k and τ_k' for each segment, j = 1 designating the translation variable, j = 2, 3, 4 indicating the rotation variables. Then, the transition between the constant velocity segments is carried out by a second degree acceleration whose duration is $2T_k$ (Figure 13.12).

a) Translation motion

Let k_{v1} designate the maximum velocity of translation. The magnitude of the constant velocity of translation at segment k, denoted by v^k, is defined as:

$$v^k = \lambda_1^k\, k_{v1} \qquad [13.70]$$

The constant velocity at segment k is given by:

$$\mathbf{V}^k = v^k \frac{\mathbf{P}^{k+1} - \mathbf{P}^k}{\|\mathbf{P}^{k+1} - \mathbf{P}^k\|} \quad \text{for } k = 1, \ldots, m-1 \qquad [13.71]$$

and the traveling time h_k at segment k is equal to $\frac{|D^k|}{v^k}$.

The blend time around point k is such that:

$$T_{k,1} = \frac{3}{4}\frac{\|\mathbf{V}^k - \mathbf{V}^{k-1}\|}{k_{a1}} \quad \text{for } k = 2, \ldots, m-1 \text{ and } \mathbf{V}^0 = \mathbf{V}^m = \mathbf{0} \qquad [13.72]$$

where k_{a1} is the maximum acceleration for the translation motion. For the initial and final points, the blend time is equal to $T_{k,1} = \tau_k'/2$ where τ_k' is obtained by equation [13.41].

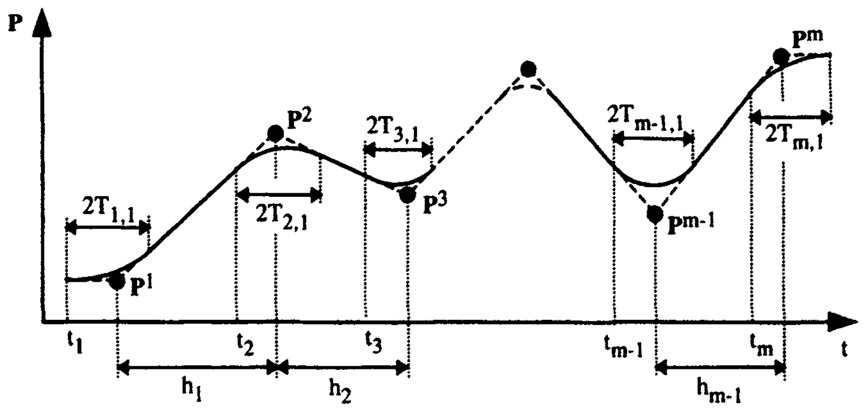

Figure 13.12. *Linear interpolations with continuous acceleration blends (translation* **P** *in the task space)*

The linear trajectory at segments k = 1, ..., m – 1 is given by:

$$\mathbf{P}(t) = (t - t_k - T_{k,1})\, \mathbf{V}^k + \mathbf{P}^k \quad \text{for } t_k + 2T_{k,1} \leq t \leq t_{k+1} \qquad [13.73]$$

where t_k is defined by equation [13.63].

The blend trajectory around point k for $t_k \leq t \leq t_k + 2T_{k,1}$ $(T_{k,1} \neq 0)$ and $1 \leq k \leq m$ is given by:

$$\mathbf{P}(t) = \mathbf{P}^k - \frac{1}{16(T_{k,1})^3}(t-t_k)^3(t-t_k-4T_{k,1})(\mathbf{V}^k - \mathbf{V}^{k-1}) + (t-t_k-T_{k,1})\,\mathbf{V}^{k-1} \qquad [13.74]$$

b) <u>Rotation motion</u>

Let β represent one of the Euler variables, and $\dot{\beta}^k$ be the velocity along segment k. The trajectory at constant velocity for $t_k + 2T_{k,j} \leq t \leq t_{k+1}$ and k = 1, ..., m – 1 is given by:

$$\beta(t) = (t - t_k - T_{k,j})\, \dot{\beta}^k + \beta^k \qquad [13.75]$$

where j = 2, 3, 4 designate ϕ, θ, ψ respectively, and the trajectory along the blend region for $t_k \leq t \leq t_k + 2T_{k,j}$ and $1 \leq k \leq m$ is given by:

$$\beta(t) = \beta^k - \frac{1}{16(T_{k,j})^3}(t-t_k)^3(t-t_k-4T_{k,j})(\dot{\beta}^k-\dot{\beta}^{k-1}) + (t-t_k-T_{k,j})\,\dot{\beta}^{k-1} \quad [13.76]$$

The blend time $T_{k,j}$ is deduced from equation [13.60] by replacing $\dot{q}_j$ with $\dot{\beta}$.

13.5.2. *Trajectory generation with cubic spline functions*

13.5.2.1. *Principle of the method*

As in § 13.5.1, we consider the path defined by a sequence of joint configurations $\mathbf{q}^1$, $\mathbf{q}^2$, ..., $\mathbf{q}^m$ such that $m \geq 4$. We assume that the corresponding traveling times $t_1, t_2, \ldots, t_m$ are known. On each segment k, i.e. between points k and k + 1, the trajectory is represented by a cubic function (Figure 13.13). This method is also termed *cubic spline function* [Edwall 82], [Lin 83].

The principle is to globally calculate the joint accelerations at the via points to satisfy the velocity and acceleration continuity. The acceleration of the cubic function for joint j (for convenience, we will omit the subscript j) is written as a linear function of time for $t_k \leq t \leq t_{k+1}$ and $k = 1, \ldots, m-1$:

$$\ddot{F}_k(t) = \frac{(t_{k+1}-t)}{h_k}\ddot{F}_k(t_k) + \frac{(t-t_k)}{h_k}\ddot{F}_k(t_{k+1}) \quad \text{with } h_k = t_{k+1}-t_k \quad [13.77]$$

Integrating equation [13.77] twice yields the velocity and position equations:

$$\dot{F}_k(t) = -\frac{(t_{k+1}-t)^2}{2h_k}\ddot{F}_k(t_k) + \frac{(t-t_k)^2}{2h_k}\ddot{F}_k(t_{k+1}) + \left[\frac{q^{k+1}}{h_k} - \frac{h_k\ddot{F}_k(t_{k+1})}{6}\right] - \left[\frac{q^k}{h_k} - \frac{h_k\ddot{F}_k(t_k)}{6}\right] \quad [13.78]$$

$$F_k(t) = \frac{(t_{k+1}-t)^3}{6h_k}\ddot{F}_k(t_k) + \frac{(t-t_k)^3}{6h_k}\ddot{F}_k(t_{k+1}) + (t-t_k)\left[\frac{q^{k+1}}{h_k} - \frac{h_k\ddot{F}_k(t_{k+1})}{6}\right] + (t_{k+1}-t)\left[\frac{q^k}{h_k} - \frac{h_k\ddot{F}_k(t_k)}{6}\right] \quad [13.79]$$

where $F_k(t_k) = q^k$ and $F_k(t_{k+1}) = q^{k+1}$.

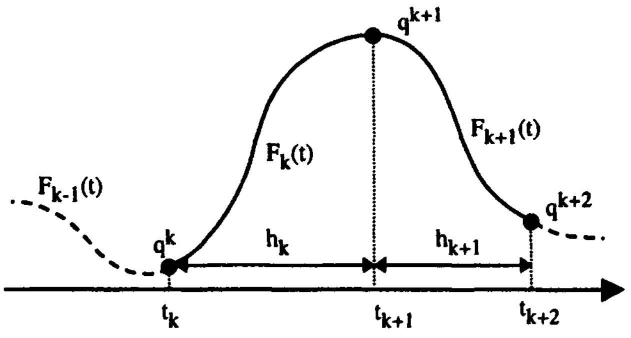

Figure 13.13. *Notations used for the cubic spline functions*

Thus, knowing the accelerations $\ddot{F}_k(t_k)$ allows us to calculate $F_k(t)$ for $k = 1, \ldots, m-1$. Continuous velocity constraint implies that:

$$\dot{F}_k(t_k) = \dot{F}_{k-1}(t_k) \quad \text{for } k = 2, \ldots, m-1 \tag{13.80}$$

which, after substitution, yields:

$$h_{k-1}\,\ddot{q}^{k-1} + 2\,(h_{k-1} + h_k)\,\ddot{q}^k + h_k\,\ddot{q}^{k+1} = 6\left(\frac{D^k}{h_k} - \frac{D^{k-1}}{h_{k-1}}\right) \tag{13.81}$$

$$\text{with } \begin{cases} \ddot{F}_k(t_{k+1}) = \ddot{F}_{k+1}(t_{k+1}) = \ddot{q}^{k+1} \\ D^k = q^{k+1} - q^k \end{cases}$$

For convenience, we rewrite equation [13.81] in matrix form:

$$[h_{k-1} \quad 2(h_{k-1}+h_k) \quad h_k] \begin{bmatrix} \ddot{q}^{k-1} \\ \ddot{q}^k \\ \ddot{q}^{k+1} \end{bmatrix} = 6\left(\frac{D^k}{h_k} - \frac{D^{k-1}}{h_{k-1}}\right) \tag{13.82}$$

Let us assume that $m \geq 4$ and that $\ddot{q}_j^1$ and $\ddot{q}_j^m$ are known[1] (for example, zero). By calculating equation [13.82] for $k = 2, \ldots, m-1$, and combining all the equations together, we obtain for joint j a system of equations that can be written in the following matrix form:

[1] One could assume that $\dot{q}_j^1$ and $\dot{q}_j^m$ are known instead.

$$\mathbf{M}\,\ddot{\mathbf{q}}_j = \mathbf{N}_j \qquad [13.83]$$

with $\ddot{\mathbf{q}}_j = [\ddot{q}_j^2 \;\; \dots \;\; \ddot{q}_j^{m-1}]^T$.

Thus, we can calculate the accelerations at points k = 2, ..., m−1, and consequently the interpolation functions F_k for k = 1, ..., m−1. The matrix **M** is identical for all the joints but the vector $\mathbf{N}_j$ is different. **M** is tridiagonal and regular. Efficient methods to inverse such matrices can be implemented [de Boor 78]. It is worth noting that the initial and final joint velocities are obtained from these equations. To specify desired velocities at the initial and final points, we can either use fourth degree polynomials or two cubic spline functions for the first and the last segments [Edwall 82]. In the following, we develop the second solution. It requires specification of two additional points: one after the initial point and the other before the final point. For convenience, we consider that the total number of points is still denoted by m.

Let us assume that velocities and accelerations on the boundary points are given by $\dot{q}(t_1)$, $\ddot{q}(t_1)$, $\dot{q}(t_m)$, $\ddot{q}(t_m)$. To satisfy the constraints of continuity, Lin [Lin 83] has shown that the new second point should be defined as:

$$q^2 = q^1 + h_1\,\dot{q}(t_1) + \frac{h_1{}^2}{3}\,\ddot{q}(t_1) + \frac{h_1{}^2}{6}\,\ddot{q}(t_2) \qquad [13.84]$$

and the (m−1)th one as:

$$q^{m-1} = q^m - h_{m-1}\,\dot{q}(t_m) + \frac{h_{m-1}{}^2}{3}\,\ddot{q}(t_m) + \frac{h_{m-1}{}^2}{6}\,\ddot{q}(t_{m-1}) \qquad [13.85]$$

Thus, the first two and the last two equations of system [13.82] must be modified and the matrix **M** becomes:

$$\begin{bmatrix}
3h_1 + 2h_2 + \frac{h_1^2}{h_2} & h_2 & 0 & \dots & \dots & \dots & 0 \\
h_2 - \frac{h_1^2}{h_2} & 2(h_2+h_3) & h_3 & 0 & \dots & \dots & 0 \\
0 & h_3 & 2(h_3+h_4) & h_4 & 0 & \dots & 0 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots \\
0 & 0 & \dots & 0 & h_{m-3} & 2(h_{m-3}+h_{m-2}) & h_{m-2} - \frac{h_{m-1}^2}{h_{m-2}} \\
0 & 0 & \dots & 0 & 0 & h_{m-2} & 3h_{m-1} + 2h_{m-2} + \frac{h_{m-1}^2}{h_{m-2}}
\end{bmatrix} \qquad [13.86]$$

while the vector $\mathbf{N}_j$ becomes:

$$\mathbf{N}_j = \begin{bmatrix} 6(\frac{q^3}{h_2}+\frac{q^1}{h_1})-6(\frac{1}{h_1}+\frac{1}{h_2})(q^1+h_1 v_1+\frac{h_1^2}{3}a_1)-h_1 a_1 \\ \frac{6}{h_2}(q^1+h_1 v_1+\frac{h_1^2}{3}a_1)+\frac{6q^4}{h_3}-6(\frac{1}{h_2}+\frac{1}{h_3})q^3 \\ 6[\frac{(q^5-q^4)}{h_4}-\frac{(q^4-q^3)}{h_3}] \\ \cdots \\ \frac{6}{h_{m-2}}(q^m-h_{m-1}v_m+\frac{h_{m-1}^2}{3}a_m)+\frac{6q^{m-3}}{h_{m-3}}-6(\frac{1}{h_{m-2}}+\frac{1}{h_{m-3}})q^{m-2} \\ 6(\frac{q^m}{h_{m-1}}+\frac{q^{m-2}}{h_{m-2}})-6(\frac{1}{h_{m-1}}+\frac{1}{h_{m-2}})(q^m-h_{m-1}v_m+\frac{h_{m-1}^2}{3}a_m)-h_{m-1}a_m \end{bmatrix} \qquad [13.87]$$

with $v_1 = \dot{q}_j(t_1)$, $a_1 = \ddot{q}_j(t_1)$, $v_m = \dot{q}_j(t_m)$ and $a_m = \ddot{q}_j(t_m)$.

13.5.2.2. *Calculation of the minimum traveling time on each segment*

If the traveling times ($h_1, \ldots, h_{m-1}$) are not specified, their calculation in order to obtain a minimum global time is not as simple as in the case of a point-to-point trajectory. Optimization techniques must then be implemented [Lin 83]. Nowadays, this problem is facilitated by the existence of efficient optimization softwares.

If [$h_1, h_2, \ldots, h_{m-1}$] is the vector of variables to be optimized and T the total traveling time, the problem is formulated as follows:

Minimize the function $T = \sum_{k=1}^{m-1} h_k$ under the constraint that velocities, accelerations and eventually jerks (rate of change of the acceleration) in the joint space remain within their bounds all over the trajectory.

Since cubic spline functions are linear in acceleration, the corresponding inequality constraints are expressed by:

$$|\ddot{q}_j^k| \leq k_{aj} \quad \text{for } j = 1, \ldots, n \text{ and } k = 2, \ldots, m-1 \qquad [13.88]$$

The magnitude of the jerk is bounded such that:

$$\frac{|\ddot{q}_j^{k+1} - \ddot{q}_j^k|}{h_k} \leq k_{sj} \quad \text{for } j = 1, \ldots, n \text{ and } k = 2, \ldots, m-1 \qquad [13.89]$$

Maximum velocities occur at the time when $\ddot{q}_j(t) = 0$. If for a given function $F_k(t)$, the value of this time is not between t_k and $t_k + h_k$, then the maximum velocity for this function will be $|\dot{q}_j^k|$ or $|\dot{q}_j^{k+1}|$; otherwise it is necessary to calculate the velocity corresponding to this sampling time.

To initialize the optimization procedure, we calculate a lower bound h'_k of the traveling time on each segment k using the equation:

$$h'_k = \max_j \{\frac{|D_j^k|}{k_{vj}}\} \qquad \text{for } j = 1, \ldots, n \text{ and } k = 1, \ldots, m-1 \qquad [13.90]$$

For the first and last two segments, the traveling times are initialized as follows:

$$\begin{cases} h'_1 = h'_2 = \max\{\dfrac{|q_j^3 - q_j^1|}{2\,k_{vj}}\} \\ h'_{m-2} = h'_{m-1} = \max\{\dfrac{|q_j^m - q_j^{m-2}|}{2\,k_{vj}}\} \end{cases} \qquad \text{for } j = 1, \ldots, n \qquad [13.91]$$

Then, in order to derive an acceptable solution satisfying the constraints, we scale up the time by a factor λ, which modifies the velocity, acceleration and jerk by $1/\lambda$, $1/\lambda^2$ and $1/\lambda^3$ respectively. The time h'_k is thus replaced by h_k:

$$h_k = \lambda h'_k \qquad [13.92]$$

The scale factor λ is selected to saturate the velocity, the acceleration, or the jerk:

$$\lambda = \max\left[\frac{|\dot{q}_{jmax}|}{k_{vj}}, (\frac{|\ddot{q}_{jmax}|}{k_{aj}})^{1/2}, (\frac{|\dddot{q}_{jmax}|}{k_{sj}})^{1/3}\right] \qquad \text{for } j = 1, \ldots, n \qquad [13.93]$$

where $\dot{q}_{jmax}$, $\ddot{q}_{jmax}$ and $\dddot{q}_{jmax}$ denote the maximum velocity, acceleration and jerk of joint j.

NOTES.–

- the minimum traveling time problem is a nonlinear programming problem that can be solved with the "constr" function (Optimization ToolBox) of Matlab (quasi-Newton algorithm);
- instead of calculating the global trajectory for all the points, Chand and Doty [Chand 85] showed that the trajectory could be computed on-line by iteratively considering only a limited number of points at each time.

13.5.3. *Trajectory generation on a continuous path in the task space*

In the previous sections, we showed how to generate a trajectory of the robot endpoint passing through, or close to, a sequence of points. Another procedure, commonly used in continuous processes such as machining or arc welding, consists of first generating a continuous path from the sequence of points at hand, then determining a time history along the path. Processing separately the path generation and the trajectory generation allows the robot to follow the specified spatial path whatever the velocity, which can be modified on-line by an external action (operator or sensor).

Generally, the geometry of the path is described in terms of a parameter u over the interval $0 \leq u \leq 1$. For convenience, let us consider only the position path (the following results are extendable to orientation):

$$\mathbf{P}(u) = [\ P_x(u)\ \ P_y(u)\ \ P_z(u)\ ]^T \qquad [13.94]$$

To specify a continuous trajectory in acceleration, the path **P**(u) should be of class C^2 in u, which means a continuity of curvature. For example, cubic splines, cubic B-splines or Bézier curves can be used to represent **P**(u) as a polynomial function in u, as is done in CAD/CAM systems [Bartels 88], [Léon 91].

Then, we determine the trajectory by choosing for u a suitable function of time. The simplest function is $u = \lambda\, t$, but it is more interesting to specify the velocity of the curvilinear abscissa of the tool along the path. This requires computation of a one-to-one mapping between the curvilinear abscissa, denoted by s, and the parameter u, using the fact that:

$$\frac{ds(u)}{du} = \sqrt{[\frac{dPx(u)}{du}]^2 + [\frac{dPy(u)}{du}]^2 + [\frac{dPz(u)}{du}]^2} = \|\frac{d\mathbf{P}(u)}{du}\| \qquad [13.95]$$

The curvilinear abscissa s is given as a function of u by integration:

$$s = \int_{u_1}^{u_2} \frac{ds(u)}{du}\, du = \int_{u_1}^{u_2} \|\frac{d\mathbf{P}(u)}{du}\|\, du \qquad [13.96]$$

Thus, the trajectory generation consists of specifying the time history of the curvilinear abscissa s. Cartesian velocities and accelerations are given by:

$$\dot{\mathbf{P}}(s) = \frac{d\mathbf{P}(s)}{dt} = \frac{d\mathbf{P}(s)}{ds}\frac{ds}{dt} = \dot{s}\,\frac{d\mathbf{P}(s)}{ds} \qquad [13.97]$$

$$\ddot{\mathbf{P}}(s) = \ddot{s}\frac{d\mathbf{P}(s)}{ds} + \dot{s}^2\frac{d^2\mathbf{P}(s)}{ds^2} \qquad [13.98]$$

Considering equation [13.95], we obtain:

$$\frac{d\mathbf{P}(s)}{ds} = \frac{d\mathbf{P}(u)}{du}\frac{du}{ds} = \frac{d\mathbf{P}(u)}{du}\frac{1}{\left\|\frac{d\mathbf{P}(u)}{du}\right\|} \qquad [13.99]$$

A closed-form solution for **P**(s) exists in the case of straight line and circular paths. For a straight line path for instance, let $\mathbf{P}^i$ and $\mathbf{P}^f$ be the initial and final points, and D be the Cartesian distance to travel. We can write that:

$$\mathbf{P}(s) = \mathbf{P}^i + \frac{s}{D}(\mathbf{P}^f - \mathbf{P}^i) \qquad [13.100]$$

For a circular path in the (x_c, y_c) plane of the circle, the equation is given by:

$$\mathbf{P}(s) = \mathbf{P}^c + \begin{bmatrix} R\cos(\frac{s}{R} + \phi_0) \\ R\sin(\frac{s}{R} + \phi_0) \end{bmatrix} \qquad [13.101]$$

where $\mathbf{P}^c$ is the vector of the x and y coordinates of the circle center, R is the circle radius, ϕ_0 is the angle between the vector $\mathbf{P}^c\mathbf{P}^i$ and the axis x_c, and $\mathbf{P}^i$ is the initial point such that:

$$\begin{bmatrix} \cos(\phi_0) \\ \sin(\phi_0) \end{bmatrix} = \frac{1}{R}(\mathbf{P}^c - \mathbf{P}^i) \qquad [13.102]$$

In the general case, we numerically determine a polynomial giving s as a function of u. Indeed, integrating the equation ds/du yields curvilinear abscissa s(u) at regular intervals of u. Then, to evaluate s(u), it is sufficient to interpolate the resulting points with a polynomial function in u, whose coefficients c_i are estimated by a least square procedure. A fourth degree polynomial should provide sufficient accuracy [Froissart 91]:

$$s(u) = \sum_{i=0}^{4} c_i u^i \qquad [13.103]$$

Then, the algorithm can be as follows:

- compute a path $\mathbf{P}(u)$;
- compute the curvilinear abscissa $s(u)$ along the path;
- compute the time history $s(t)$;
- compute $u(t)$;
- compute the trajectory $\mathbf{P}(t)$.

The time history can be computed as indicated previously for the curvilinear abscissa s. Another method has been proposed in [Sgarbi 92]: it consists of accelerating until the desired velocity is reached (or velocity and acceleration bounds are attained), maintaining this value, and finally decelerating to finish at zero velocity at the end of the path. Thus, the velocity tracks a trapeze profile. Let $\dot{s}$ be the current curvilinear velocity, $\dot{s}^d$ be the desired one, T_e be the sampling period, and $L(i)$ be the distance between the current point and the final point. The algorithm is as follows:

```
if $\{\text{abs}\,\frac{1}{T_e}[\dot{s}(i-1) - \dot{s}^d(i)]\} > \ddot{s}_{max}$, then:
        $\dot{s}(i) = \dot{s}(i-1) + \ddot{s}_{max}\, T_e\, \text{sign}[\dot{s}^d(i) - \dot{s}(i-1)]$
                                        else $\dot{s}(i) = \dot{s}^d(i)$
if $L(i) < \frac{[\dot{s}(i)]^2}{2\ddot{s}_{max}}$, then begin deceleration phase.
```

An immediate extension of this algorithm would be to generate a continuous curvilinear acceleration by implementing a trapeze acceleration profile.

13.6. Conclusion

In this chapter, we have presented several methods of trajectory generation that are commonly used in robotics. We have first dealt with point-to-point trajectories: different interpolation functions have been studied, namely the trapeze velocity profile, which is implemented in most of the industrial controllers. For each function, we computed the minimum traveling time, from which it is possible to synchronize the joints so that they reach the final point simultaneously. We have also presented three methods of trajectory generation with via points. In the first method, straight line segments joining the via points are blended together by continuous acceleration phases. In the second method, the path passes through the via points, the trajectory being described by a sequence of cubic spline functions. In the third method, the trajectory is computed on a predefined continuous path.

These methods apply for both joint space and task space. The choice of a space depends on the trajectory specification and on the task description.

The interested reader will find in [Shin 85], [Fourquet 90], [Shiller 94], other techniques using the dynamic model which allows replacement of the constraints of acceleration by those more realistic of actuator torques. Likewise, in [Pledel 96], an approach using an exact model of actuators is considered. However, instead of implementing these techniques, an *a posteriori* verification of the constraint validity and scaling the traveling time may be satisfactory [Hollerbach 84a].