4

Trajectory Planning

For the execution of a specific robot task, it is worth considering the main features of motion planning algorithms. The goal of *trajectory planning* is to generate the reference inputs to the motion control system which ensures that the manipulator executes the planned trajectories. The user typically specifies a number of parameters to describe the desired trajectory. Planning consists of generating a time sequence of the values attained by an interpolating function (typically a polynomial) of the desired trajectory. This chapter presents some techniques for trajectory generation, both in the case when the initial and final point of the path are assigned (*point-to-point motion*), and in the case when a finite sequence of points are assigned along the path (*motion through a sequence of points*). First, the problem of trajectory planning in the *joint space* is considered, and then the basic concepts of trajectory planning in the *operational space* are illustrated. The treatment of the motion planning problem for mobile robots is deferred to Chap. 12.

4.1 Path and Trajectory

The minimal requirement for a manipulator is the capability to move from an initial posture to a final assigned posture. The transition should be characterized by motion laws requiring the actuators to exert joint generalized forces which do not violate the saturation limits and do not excite the typically modelled resonant modes of the structure. It is then necessary to devise planning algorithms that generate suitably smooth trajectories.

In order to avoid confusion between terms often used as synonyms, the difference between a path and a trajectory is to be explained. A *path* denotes the locus of points in the joint space, or in the operational space, which the manipulator has to follow in the execution of the assigned motion; a path is then a pure geometric description of motion. On the other hand, a *trajectory* is a path on which a timing law is specified, for instance in terms of velocities and/or accelerations at each point.

In principle, it can be conceived that the inputs to a *trajectory planning* algorithm are the path description, the path constraints, and the constraints imposed by manipulator dynamics, whereas the outputs are the end-effector trajectories in terms of a time sequence of the values attained by position, velocity and acceleration.

A geometric path cannot be fully specified by the user for obvious complexity reasons. Typically, a reduced number of parameters is specified such as extremal points, possible intermediate points, and geometric primitives interpolating the points. Also, the motion timing law is not typically specified at each point of the geometric path, but rather it regards the total trajectory time, the constraints on the maximum velocities and accelerations, and eventually the assignment of velocity and acceleration at points of particular interest. On the basis of the above information, the trajectory planning algorithm generates a time sequence of variables that describe end-effector position and orientation over time in respect of the imposed constraints. Since the control action on the manipulator is carried out in the joint space, a suitable inverse kinematics algorithm is to be used to reconstruct the time sequence of joint variables corresponding to the above sequence in the operational space.

Trajectory planning in the operational space naturally allows the presence of path constraints to be accounted; these are due to regions of workspace which are forbidden to the manipulator, e.g., due to the presence of obstacles. In fact, such constraints are typically better described in the operational space, since their corresponding points in the joint space are difficult to compute.

With regard to motion in the neighbourhood of singular configurations and presence of redundant DOFs, trajectory planning in the operational space may involve problems difficult to solve. In such cases, it may be advisable to specify the path in the joint space, still in terms of a reduced number of parameters. Hence, a time sequence of joint variables has to be generated which satisfy the constraints imposed on the trajectory.

For the sake of clarity, in the following, the case of joint space trajectory planning is treated first. The results will then be extended to the case of trajectories in the operational space.

4.2 Joint Space Trajectories

A manipulator motion is typically assigned in the operational space in terms of trajectory parameters such as the initial and final end-effector pose, possible intermediate poses, and travelling time along particular geometric paths. If it is desired to plan a trajectory in the *joint space*, the values of the joint variables have to be determined first from the end-effector position and orientation specified by the user. It is then necessary to resort to an inverse kinematics algorithm, if planning is done off-line, or to directly measure the above variables, if planning is done by the teaching-by-showing technique (see Chap. 6).

The planning algorithm generates a function $\boldsymbol{q}(t)$ interpolating the given vectors of joint variables at each point, in respect of the imposed constraints.

In general, a joint space trajectory planning algorithm is required to have the following features:

- the generated trajectories should be not very demanding from a computational viewpoint,
- the joint positions and velocities should be continuous functions of time (continuity of accelerations may be imposed, too),
- undesirable effects should be minimized, e.g., nonsmooth trajectories interpolating a sequence of points on a path.

At first, the case is examined when only the initial and final points on the path and the traveling time are specified (*point-to-point*); the results are then generalized to the case when also intermediate points along the path are specified (*motion through a sequence of points*). Without loss of generality, the single joint variable $q(t)$ is considered.

4.2.1 Point-to-Point Motion

In *point-to-point motion*, the manipulator has to move from an initial to a final joint configuration in a given time t_f. In this case, the actual end-effector path is of no concern. The algorithm should generate a trajectory which, in respect to the above general requirements, is also capable of optimizing some performance index when the joint is moved from one position to another.

A suggestion for choosing the motion primitive may stem from the analysis of an incremental motion problem. Let I be the moment of inertia of a rigid body about its rotation axis. It is required to take the angle q from an initial value q_i to a final value q_f in a time t_f. It is obvious that infinite solutions exist to this problem. Assumed that rotation is executed through a torque τ supplied by a motor, a solution can be found which minimizes the energy dissipated in the motor. This optimization problem can be formalized as follows. Having set $\dot{q} = \omega$, determine the solution to the differential equation

$$I\dot{\omega} = \tau$$

subject to the condition

$$\int_o^{t_f} \omega(t)dt = q_f - q_i$$

so as to minimize the performance index

$$\int_0^{t_f} \tau^2(t)dt.$$

It can be shown that the resulting solution is of the type

$$\omega(t) = at^2 + bt + c.$$

Even though the joint dynamics cannot be described in the above simple manner,[1] the choice of a third-order polynomial function to generate a joint trajectory represents a valid solution for the problem at issue.

Therefore, to determine a joint motion, the *cubic polynomial*

$$q(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{4.1}$$

can be chosen, resulting into a parabolic velocity profile

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

and a linear acceleration profile

$$\ddot{q}(t) = 6a_3 t + 2a_2.$$

Since four coefficients are available, it is possible to impose, besides the initial and final joint position values q_i and q_f, also the initial and final joint velocity values $\dot{q}_i$ and $\dot{q}_f$ which are usually set to zero. Determination of a specific trajectory is given by the solution to the following system of equations:

$$\begin{aligned} a_0 &= q_i \\ a_1 &= \dot{q}_i \\ a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 &= q_f \\ 3a_3 t_f^2 + 2a_2 t_f + a_1 &= \dot{q}_f, \end{aligned}$$

that allows the computation of the coefficients of the polynomial in (4.1).[2] Figure 4.1 illustrates the timing law obtained with the following data: $q_i = 0$, $q_f = \pi$, $t_f = 1$, and $\dot{q}_i = \dot{q}_f = 0$. As anticipated, velocity has a parabolic profile, while acceleration has a linear profile with initial and final discontinuity.

If it is desired to assign also the initial and final values of acceleration, six constraints have to be satisfied and then a polynomial of at least *fifth* order is needed. The motion timing law for the generic joint is then given by

$$q(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0, \tag{4.2}$$

whose coefficients can be computed, as for the previous case, by imposing the conditions for $t = 0$ and $t = t_f$ on the joint variable $q(t)$ and on its first two derivatives. With the choice (4.2), one obviously gives up minimizing the above performance index.

An alternative approach with timing laws of blended polynomial type is frequently adopted in industrial practice, which allows a direct verification

[1] In fact, recall that the moment of inertia about the joint axis is a function of manipulator configuration.

[2] Notice that it is possible to normalize the computation of the coefficients, so as to be independent both on the final time t_f and on the path length $|q_f - q_i|$.

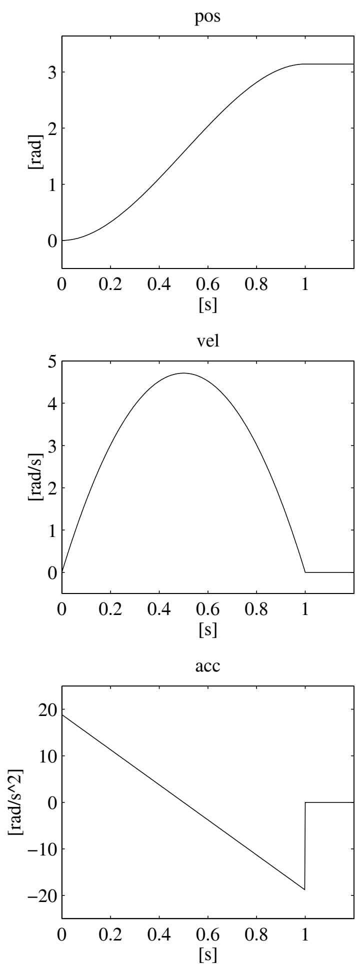

Fig. 4.1. Time history of position, velocity and acceleration with a cubic polynomial timing law

of whether the resulting velocities and accelerations can be supported by the physical mechanical manipulator.

In this case, a *trapezoidal velocity profile* is assigned, which imposes a constant acceleration in the start phase, a cruise velocity, and a constant deceleration in the arrival phase. The resulting trajectory is formed by a linear segment connected by two parabolic segments to the initial and final positions.

In the following, the problem is formulated by assuming that the final time of trajectory duration has been assigned. However, in industrial practice, the user is offered the option to specify the velocity percentage with respect to the maximum allowable velocity; this choice is aimed at avoiding occurrences when

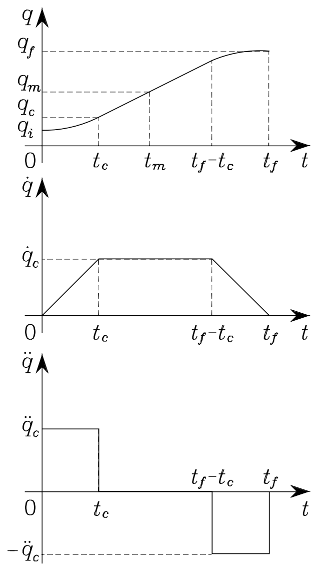

Fig. 4.2. Characterization of a timing law with trapezoidal velocity profile in terms of position, velocity and acceleration

the specification of a much too short motion duration would involve much too large values of velocities and/or accelerations, beyond those achievable by the manipulator.

As can be seen from the velocity profiles in Fig. 4.2, it is assumed that both initial and final velocities are null and the segments with constant accelerations have the same time duration; this implies an equal magnitude $\ddot{q}_c$ in the two segments. Notice also that the above choice leads to a symmetric trajectory with respect to the average point $q_m = (q_f + q_i)/2$ at $t_m = t_f/2$.

The trajectory has to satisfy some constraints to ensure the transition from q_i to q_f in a time t_f. The velocity at the end of the parabolic segment must be equal to the (constant) velocity of the linear segment, i.e.,

$$\ddot{q}_c t_c = \frac{q_m - q_c}{t_m - t_c} \tag{4.3}$$

where q_c is the value attained by the joint variable at the end of the parabolic segment at time t_c with constant acceleration $\ddot{q}_c$ (recall that $\dot{q}(0) = 0$). It is then

$$q_c = q_i + \frac{1}{2}\ddot{q}_c t_c^2. \tag{4.4}$$

Combining (4.3), (4.4) gives

$$\ddot{q}_c t_c^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0. \tag{4.5}$$

Usually, $\ddot{q}_c$ is specified with the constraint that $\operatorname{sgn}\ddot{q}_c = \operatorname{sgn}(q_f - q_i)$; hence, for given t_f, q_i and q_f, the solution for t_c is computed from (4.5) as ($t_c \le t_f/2$)

$$t_c = \frac{t_f}{2} - \frac{1}{2}\sqrt{\frac{t_f^2\ddot{q}_c - 4(q_f - q_i)}{\ddot{q}_c}}. \tag{4.6}$$

Acceleration is then subject to the constraint

$$|\ddot{q}_c| \ge \frac{4|q_f - q_i|}{t_f^2}. \tag{4.7}$$

When the acceleration $\ddot{q}_c$ is chosen so as to satisfy (4.7) with the equality sign, the resulting trajectory does not feature the constant velocity segment any more and has only the acceleration and deceleration segments (*triangular* profile).

Given q_i, q_f and t_f, and thus also an average transition velocity, the constraint in (4.7) allows the imposition of a value of acceleration consistent with the trajectory. Then, t_c is computed from (4.6), and the following sequence of polynomials is generated:

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t^2 & 0 \le t \le t_c \\ q_i + \ddot{q}_c t_c(t - t_c/2) & t_c < t \le t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c(t_f - t)^2 & t_f - t_c < t \le t_f. \end{cases} \tag{4.8}$$

Figure 4.3 illustrates a representation of the motion timing law obtained by imposing the data: $q_i = 0$, $q_f = \pi$, $t_f = 1$, and $|\ddot{q}_c| = 6\pi$.

Specifying acceleration in the parabolic segment is not the only way to determine trajectories with trapezoidal velocity profile. Besides q_i, q_f and t_f, one can specify also the cruise velocity $\dot{q}_c$ which is subject to the constraint

$$\frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \le \frac{2|q_f - q_i|}{t_f}. \tag{4.9}$$

By recognizing that $\dot{q}_c = \ddot{q}_c t_c$, (4.5) allows the computation of t_c as

$$t_c = \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c}, \tag{4.10}$$

and thus the resulting acceleration is

$$\ddot{q}_c = \frac{\dot{q}_c^2}{q_i - q_f + \dot{q}_c t_f}. \tag{4.11}$$

The computed values of t_c and $\ddot{q}_c$ as in (4.10), (4.11) allow the generation of the sequence of polynomials expressed by (4.8).

The adoption of a trapezoidal velocity profile results in a worse performance index compared to the cubic polynomial. The decrease is, however, limited; the term $\int_0^{t_f} \tau^2 dt$ increases by 12.5% with respect to the optimal case.

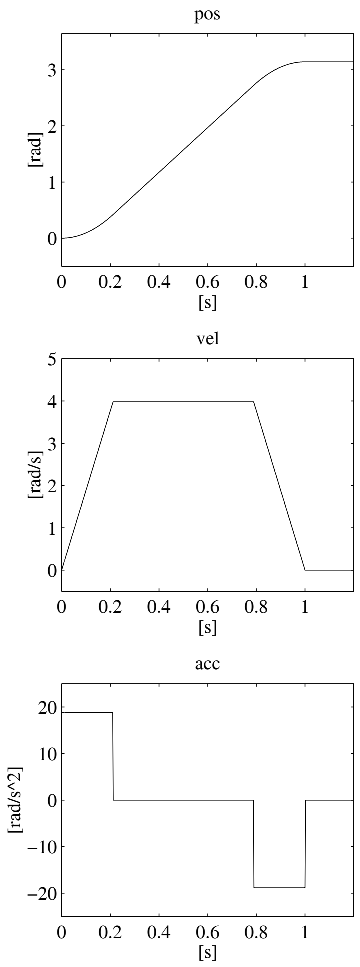

Fig. 4.3. Time history of position, velocity and acceleration with a trapezoidal velocity profile timing law

4.2.2 Motion Through a Sequence of Points

In several applications, the path is described in terms of a number of points greater than two. For instance, even for the simple point-to-point motion of a pick-and-place task, it may be worth assigning two intermediate points between the initial point and the final point; suitable positions can be set for lifting off and setting down the object, so that reduced velocities are obtained with respect to direct transfer of the object. For more complex applications, it may be convenient to assign a *sequence of points* so as to guarantee better monitoring on the executed trajectories; the points are to be specified more densely in those segments of the path where obstacles have to be avoided

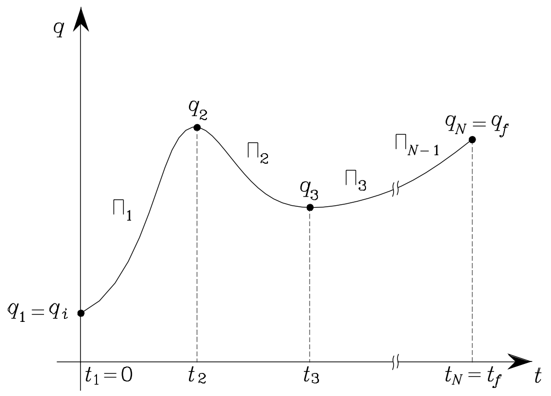

Fig. 4.4. Characterization of a trajectory on a given path obtained through interpolating polynomials

or a high path curvature is expected. It should not be forgotten that the corresponding joint variables have to be computed from the operational space poses.

Therefore, the problem is to generate a trajectory when N points, termed *path points*, are specified and have to be reached by the manipulator at certain instants of time. For each joint variable there are N constraints, and then one might want to use an $(N-1)$-order polynomial. This choice, however, has the following disadvantages:

- It is not possible to assign the initial and final velocities.
- As the order of a polynomial increases, its oscillatory behaviour increases, and this may lead to trajectories which are not natural for the manipulator.
- Numerical accuracy for computation of polynomial coefficients decreases as order increases.
- The resulting system of constraint equations is heavy to solve.
- Polynomial coefficients depend on all the assigned points; thus, if it is desired to change a point, all of them have to be recomputed.

These drawbacks can be overcome if a suitable number of low-order *interpolating polynomials*, continuous at the path points, are considered in place of a single high-order polynomial.

According to the previous section, the interpolating polynomial of lowest order is the *cubic polynomial*, since it allows the imposition of continuity of velocities at the path points. With reference to the single joint variable, a function $q(t)$ is sought, formed by a sequence of $N-1$ cubic polynomials $\Pi_k(t)$, for $k = 1, \ldots, N-1$, continuous with continuous first derivatives. The function $q(t)$ attains the values q_k for $t = t_k$ $(k = 1, \ldots, N)$, and $q_1 = q_i$, $t_1 = 0$, $q_N = q_f$, $t_N = t_f$; the q_k's represent the path points describing the desired trajectory at $t = t_k$ (Fig. 4.4). The following situations can be considered:

- Arbitrary values of $\dot{q}(t)$ are imposed at the path points.
- The values of $\dot{q}(t)$ at the path points are assigned according to a certain criterion.
- The acceleration $\ddot{q}(t)$ has to be continuous at the path points.

To simplify the problem, it is also possible to find interpolating polynomials of order less than three which determine trajectories passing nearby the path points at the given instants of time.

Interpolating polynomials with imposed velocities at path points

This solution requires the user to be able to specify the desired velocity at each path point; the solution does not possess any novelty with respect to the above concepts.

The system of equations allowing computation of the coefficients of the $N-1$ cubic polynomials interpolating the N path points is obtained by imposing the following conditions on the generic polynomial $\Pi_k(t)$ interpolating q_k and q_{k+1}, for $k = 1, \ldots, N-1$:

$$\begin{aligned}\Pi_k(t_k) &= q_k\\ \Pi_k(t_{k+1}) &= q_{k+1}\\ \dot{\Pi}_k(t_k) &= \dot{q}_k\\ \dot{\Pi}_k(t_{k+1}) &= \dot{q}_{k+1}.\end{aligned}$$

The result is $N-1$ systems of four equations in the four unknown coefficients of the generic polynomial; these can be solved one independently of the other. The initial and final velocities of the trajectory are typically set to zero ($\dot{q}_1 = \dot{q}_N = 0$) and continuity of velocity at the path points is ensured by setting

$$\dot{\Pi}_k(t_{k+1}) = \dot{\Pi}_{k+1}(t_{k+1})$$

for $k = 1, \ldots, N-2$.

Figure 4.5 illustrates the time history of position, velocity and acceleration obtained with the data: $q_1 = 0$, $q_2 = 2\pi$, $q_3 = \pi/2$, $q_4 = \pi$, $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 5$, $\dot{q}_1 = 0$, $\dot{q}_2 = \pi$, $\dot{q}_3 = -\pi$, $\dot{q}_4 = 0$. Notice the resulting discontinuity on the acceleration, since only continuity of velocity is guaranteed.

Interpolating polynomials with computed velocities at path points

In this case, the joint velocity at a path point has to be computed according to a certain criterion. By interpolating the path points with linear segments, the relative velocities can be computed according to the following rules:

$$\dot{q}_1 = 0$$

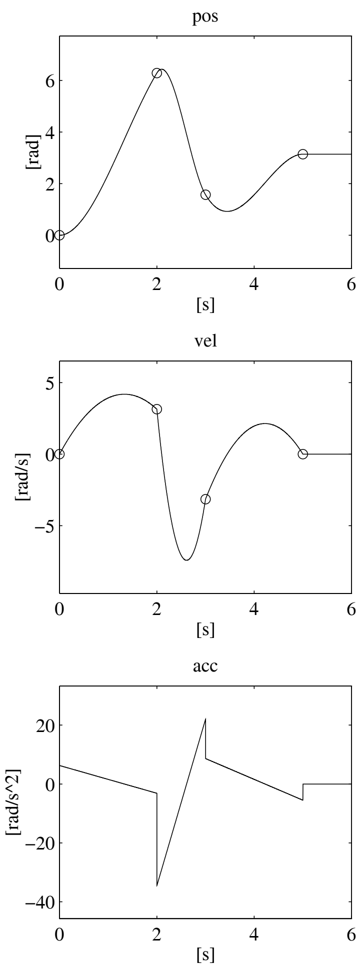

Fig. 4.5. Time history of position, velocity and acceleration with a timing law of interpolating polynomials with velocity constraints at path points

$$\dot{q}_k = \begin{cases} 0 & \operatorname{sgn}(v_k) \neq \operatorname{sgn}(v_{k+1}) \\ \frac{1}{2}(v_k + v_{k+1}) & \operatorname{sgn}(v_k) = \operatorname{sgn}(v_{k+1}) \end{cases} \tag{4.12}$$
$$\dot{q}_N = 0,$$

where $v_k = (q_k - q_{k-1})/(t_k - t_{k-1})$ gives the slope of the segment in the time interval $[t_{k-1}, t_k]$. With the above settings, the determination of the interpolating polynomials is reduced to the previous case.

Figure 4.6 illustrates the time history of position, velocity and acceleration obtained with the following data: $q_1 = 0$, $q_2 = 2\pi$, $q_3 = \pi/2$, $q_4 = \pi$, $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 5$, $\dot{q}_1 = 0$, $\dot{q}_4 = 0$. It is easy to recognize that the imposed

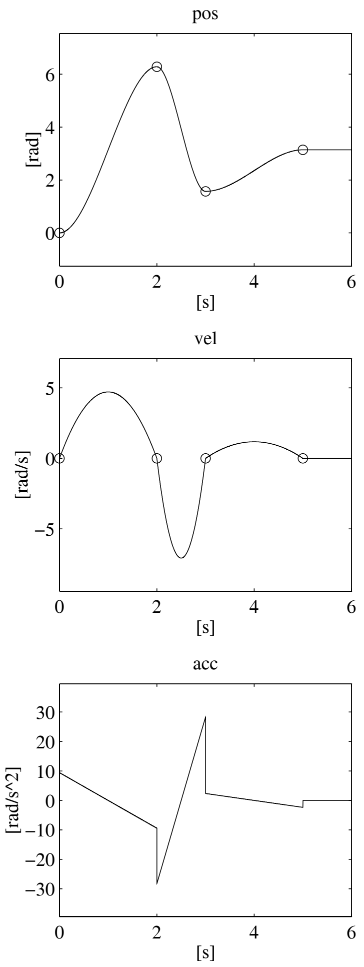

Fig. 4.6. Time history of position, velocity and acceleration with a timing law of interpolating polynomials with computed velocities at path points

sequence of path points leads to having zero velocity at the intermediate points.

Interpolating polynomials with continuous accelerations at path points (splines)

Both the above two solutions do not ensure continuity of accelerations at the path points. Given a sequence of N path points, the acceleration is also continuous at each t_k if four constraints are imposed, namely, two position constraints for each of the adjacent cubics and two constraints guaranteeing continuity of velocity and acceleration. The following equations have then to be satisfied:

$$\begin{aligned} \Pi_{k-1}(t_k) &= q_k \\ \Pi_{k-1}(t_k) &= \Pi_k(t_k) \\ \dot{\Pi}_{k-1}(t_k) &= \dot{\Pi}_k(t_k) \\ \ddot{\Pi}_{k-1}(t_k) &= \ddot{\Pi}_k(t_k). \end{aligned}$$

The resulting system for the N path points, including the initial and final points, cannot be solved. In fact, it is formed by $4(N-2)$ equations for the intermediate points and 6 equations for the extremal points; the position constraints for the polynomials $\Pi_0(t_1) = q_i$ and $\Pi_N(t_f) = q_f$ have to be excluded since they are not defined. Also, $\dot{\Pi}_0(t_1)$, $\ddot{\Pi}_0(t_1)$, $\dot{\Pi}_N(t_f)$, $\ddot{\Pi}_N(t_f)$ do not have to be counted as polynomials since they are just the imposed values of initial and final velocities and accelerations. In summary, one has $4N-2$ equations in $4(N-1)$ unknowns.

The system can be solved only if one eliminates the two equations which allow the arbitrary assignment of the initial and final acceleration values. Fourth-order polynomials should be used to include this possibility for the first and last segment.

On the other hand, if only third-order polynomials are to be used, the following deception can be operated. Two *virtual points* are introduced for which continuity constraints on position, velocity and acceleration can be imposed, without specifying the actual positions, though. It is worth remarking that the effective location of these points is irrelevant, since their position constraints regard continuity only. Hence, the introduction of two virtual points implies the determination of $N+1$ cubic polynomials.

Consider $N+2$ time instants t_k, where t_2 and t_{N+1} conventionally refer to the virtual points. The system of equations for determining the $N+1$ cubic polynomials can be found by taking the $4(N-2)$ equations:

$$\Pi_{k-1}(t_k) = q_k \tag{4.13}$$

$$\Pi_{k-1}(t_k) = \Pi_k(t_k) \tag{4.14}$$

$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k) \tag{4.15}$$

$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k) \tag{4.16}$$

for $k = 3, \ldots, N$, written for the $N-2$ intermediate path points, the 6 equations:

$$\Pi_1(t_1) = q_i \tag{4.17}$$

$$\dot{\Pi}_1(t_1) = \dot{q}_i \tag{4.18}$$

$$\ddot{\Pi}_1(t_1) = \ddot{q}_i, \tag{4.19}$$

$$\Pi_{N+1}(t_{N+2}) = q_f \tag{4.20}$$

$$\dot{\mathit{\Pi}}_{N+1}(t_{N+2}) = \dot{q}_f \tag{4.21}$$

$$\ddot{\mathit{\Pi}}_{N+1}(t_{N+2}) = \ddot{q}_f \tag{4.22}$$

written for the initial and final points, and the 6 equations:

$$\mathit{\Pi}_{k-1}(t_k) = \mathit{\Pi}_k(t_k) \tag{4.23}$$

$$\dot{\mathit{\Pi}}_{k-1}(t_k) = \dot{\mathit{\Pi}}_k(t_k) \tag{4.24}$$

$$\ddot{\mathit{\Pi}}_{k-1}(t_k) = \ddot{\mathit{\Pi}}_k(t_k) \tag{4.25}$$

for $k = 2, N+1$, written for the two virtual points. The resulting system has $4(N+1)$ equations in $4(N+1)$ unknowns, that are the coefficients of the $N+1$ cubic polynomials.

The solution to the system is computationally demanding, even for low values of N. Nonetheless, the problem can be cast in a suitable form so as to solve the resulting system of equations with a computationally efficient algorithm. Since the generic polynomial $\mathit{\Pi}_k(t)$ is a cubic, its second derivative must be a linear function of time which then can be written as

$$\ddot{\mathit{\Pi}}_k(t) = \frac{\ddot{\mathit{\Pi}}_k(t_k)}{\Delta t_k}(t_{k+1} - t) + \frac{\ddot{\mathit{\Pi}}_k(t_{k+1})}{\Delta t_k}(t - t_k) \qquad k = 1, \ldots, N+1, \tag{4.26}$$

where $\Delta t_k = t_{k+1} - t_k$ indicates the time interval to reach q_{k+1} from q_k. By integrating (4.26) twice over time, the generic polynomial can be written as

$$\begin{aligned} \mathit{\Pi}_k(t) = & \frac{\ddot{\mathit{\Pi}}_k(t_k)}{6\Delta t_k}(t_{k+1} - t)^3 + \frac{\ddot{\mathit{\Pi}}_k(t_{k+1})}{6\Delta t_k}(t - t_k)^3 \\ & + \left(\frac{\mathit{\Pi}_k(t_{k+1})}{\Delta t_k} - \frac{\Delta t_k \ddot{\mathit{\Pi}}_k(t_{k+1})}{6}\right)(t - t_k) \\ & + \left(\frac{\mathit{\Pi}_k(t_k)}{\Delta t_k} - \frac{\Delta t_k \ddot{\mathit{\Pi}}_k(t_k)}{6}\right)(t_{k+1} - t) \qquad k = 1, \ldots, N+1, \end{aligned} \tag{4.27}$$

which depends on the 4 unknowns: $\mathit{\Pi}_k(t_k)$, $\mathit{\Pi}_k(t_{k+1})$, $\ddot{\mathit{\Pi}}_k(t_k)$, $\ddot{\mathit{\Pi}}_k(t_{k+1})$.

Notice that the N variables q_k for $k \neq 2, N+1$ are given via (4.13), while continuity is imposed for q_2 and q_{N+1} via (4.23). By using (4.14), (4.17), (4.20), the unknowns in the $N+1$ equations in (4.27) reduce to $2(N+2)$. By observing that the equations in (4.18), (4.21) depend on q_2 and q_{N+1}, and that $\dot{q}_i$ and $\dot{q}_f$ are given, q_2 and q_{N+1} can be computed as a function of $\ddot{\mathit{\Pi}}_1(t_1)$ and $\ddot{\mathit{\Pi}}_{N+1}(t_{N+2})$, respectively. Thus, a number of $2(N+1)$ unknowns are left.

By accounting for (4.16), (4.25), and noticing that in ((4.19), (4.22) $\ddot{q}_i$ and $\ddot{q}_f$ are given, the unknowns reduce to N.

At this point, (4.15), (4.24) can be utilized to write the system of N equations in N unknowns:

$$\dot{\mathit{\Pi}}_1(t_2) = \dot{\mathit{\Pi}}_2(t_2)$$

$$\vdots$$
$$\dot{\Pi}_N(t_{N+1}) = \dot{\Pi}_{N+1}(t_{N+1}).$$

Time-differentiation of (4.27) gives both $\dot{\Pi}_k(t_{k+1})$ and $\dot{\Pi}_{k+1}(t_{k+1})$ for $k = 1, \ldots, N$, and thus it is possible to write a system of linear equations of the kind

$$\boldsymbol{A}\,[\,\ddot{\Pi}_2(t_2) \quad \ldots \quad \ddot{\Pi}_{N+1}(t_{N+1})\,]^T = \boldsymbol{b} \tag{4.28}$$

which presents a vector $\boldsymbol{b}$ of known terms and a nonsingular coefficient matrix $\boldsymbol{A}$; the solution to this system always exists and is unique. It can be shown that the matrix $\boldsymbol{A}$ has a tridiagonal band structure of the type

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & 0 & 0 \\ a_{21} & a_{22} & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & \ldots & a_{N,N-1} & a_{NN} \end{bmatrix},$$

which simplifies the solution to the system (see Problem 4.4). This matrix is the same for all joints, since it depends only on the time intervals Δt_k specified.

An efficient solution algorithm exists for the above system which is given by a *forward* computation followed by a *backward* computation. From the first equation, $\ddot{\Pi}_2(t_2)$ can be computed as a function of $\ddot{\Pi}_3(t_3)$ and then substituted in the second equation, which then becomes an equation in the unknowns $\ddot{\Pi}_3(t_3)$ and $\ddot{\Pi}_4(t_4)$. This is carried out forward by transforming all the equations in equations with two unknowns, except the last one which will have $\ddot{\Pi}_{N+1}(t_{N+1})$ only as unknown. At this point, all the unknowns can be determined step by step through a backward computation.

The above sequence of cubic polynomials is termed *spline* to indicate smooth functions that interpolate a sequence of given points ensuring continuity of the function and its derivatives.

Figure 4.7 illustrates the time history of position, velocity and acceleration obtained with the data: $q_1 = 0$, $q_3 = 2\pi$, $q_4 = \pi/2$, $q_6 = \pi$, $t_1 = 0$, $t_3 = 2$, $t_4 = 3$, $t_6 = 5$, $\dot{q}_1 = 0$, $\dot{q}_6 = 0$. Two different pairs of virtual points were considered at the time instants: $t_2 = 0.5$, $t_5 = 4.5$ (solid line in the figure), and $t_2 = 1.5$, $t_5 = 3.5$ (dashed line in the figure), respectively. Notice the parabolic velocity profile and the linear acceleration profile. Further, for the second pair, larger values of acceleration are obtained, since the relative time instants are closer to those of the two intermediate points.

Interpolating linear polynomials with parabolic blends

A simplification in trajectory planning can be achieved as follows. Consider the case when it is desired to interpolate N path points $q_1, \ldots, q_N$ at time

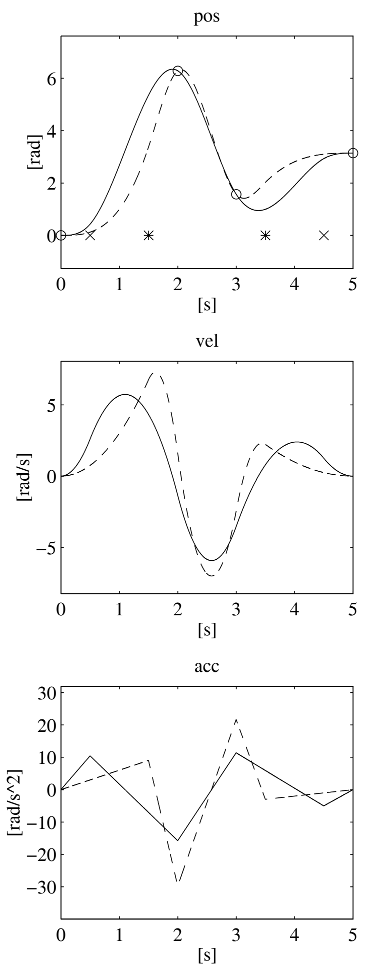

Fig. 4.7. Time history of position, velocity and acceleration with a timing law of cubic splines for two different pairs of virtual points

instants $t_1, \ldots, t_N$ with linear segments. To avoid discontinuity problems on the first derivative at the time instants t_k, the function $q(t)$ must have a parabolic profile (*blend*) around t_k; as a consequence, the entire trajectory is composed of a sequence of linear and quadratic polynomials, which in turn implies that a discontinuity on $\ddot{q}(t)$ is tolerated.

Then let $\Delta t_k = t_{k+1} - t_k$ be the time distance between q_k and q_{k+1}, and $\Delta t_{k,k+1}$ be the time interval during which the trajectory interpolating q_k and q_{k+1} is a linear function of time. Also let $\dot{q}_{k,k+1}$ be the constant velocity and $\ddot{q}_k$ be the acceleration in the parabolic blend whose duration is $\Delta t'_k$. The resulting trajectory is illustrated in Fig. 4.8. The values of q_k, Δt_k, and $\Delta t'_k$

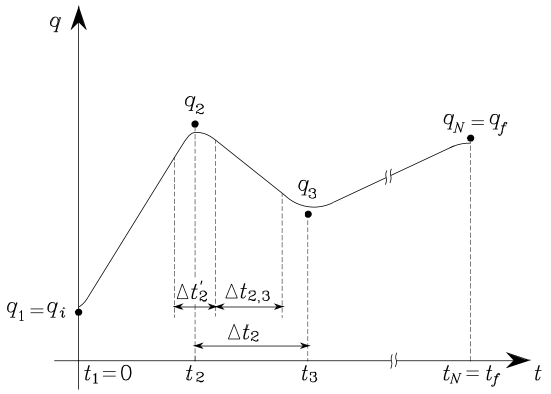

Fig. 4.8. Characterization of a trajectory with interpolating linear polynomials with parabolic blends

are assumed to be given. Velocity and acceleration for the intermediate points are computed as

$$\dot{q}_{k-1,k} = \frac{q_k - q_{k-1}}{\Delta t_{k-1}} \tag{4.29}$$

$$\ddot{q}_k = \frac{\dot{q}_{k,k+1} - \dot{q}_{k-1,k}}{\Delta t'_k}; \tag{4.30}$$

these equations are straightforward.

The first and last segments deserve special care. In fact, if it is desired to maintain the coincidence of the trajectory with the first and last segments, at least for a portion of time, the resulting trajectory has a longer duration given by $t_N - t_1 + (\Delta t'_1 + \Delta t'_N)/2$, where $\dot{q}_{0,1} = \dot{q}_{N,N+1} = 0$ has been imposed for computing initial and final accelerations.

Notice that $q(t)$ reaches none of the path points q_k but passes nearby (Fig. 4.8). In this situation, the path points are more appropriately termed *via points*; the larger the blending acceleration, the closer the passage to a via point.

On the basis of the given q_k, Δt_k and $\Delta t'_k$, the values of $\dot{q}_{k-1,k}$ and $\ddot{q}_k$ are computed via (4.29), (4.30) and a sequence of linear polynomials with parabolic blends is generated. Their expressions as a function of time are not derived here to avoid further loading of the analytical presentation.

Figure 4.9 illustrates the time history of position, velocity and acceleration obtained with the data: $q_1 = 0$, $q_2 = 2\pi$, $q_3 = \pi/2$, $q_4 = \pi$, $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 5$, $\dot{q}_1 = 0$, $\dot{q}_4 = 0$. Two different values for the blend times have been considered: $\Delta t'_k = 0.2$ (solid line in the figure) and $\Delta t'_k = 0.6$ (dashed line in the figure), for $k = 1, \ldots, 4$, respectively. Notice that in the first case the passage of $q(t)$ is closer to the via points, though at the expense of higher acceleration values.

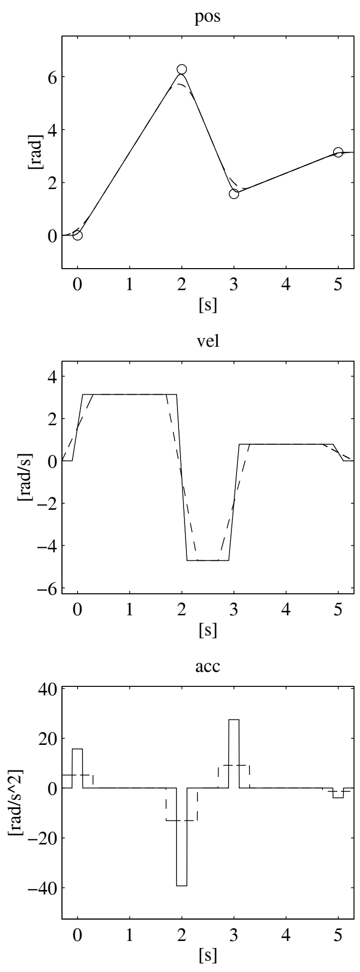

Fig. 4.9. Time history of position, velocity and acceleration with a timing law of interpolating linear polynomials with parabolic blends

The technique presented above turns out to be an application of the trapezoidal velocity profile law to the interpolation problem. If one gives up a trajectory passing near a via point at a prescribed instant of time, the use of trapezoidal velocity profiles allows the development of a trajectory planning algorithm which is attractive for its simplicity.

In particular, consider the case of one intermediate point only, and suppose that trapezoidal velocity profiles are considered as motion primitives with the possibility to specify the initial and final point and the duration of the motion only; it is assumed that $\dot{q}_i = \dot{q}_f = 0$. If two segments with trapezoidal velocity profiles were generated, the manipulator joint would certainly reach the intermediate point, but it would be forced to stop there, before continuing the motion towards the final point. A keen alternative is to start generating the second segment ahead of time with respect to the end of the first segment, using the sum of velocities (or positions) as a reference. In this way, the joint is guaranteed to reach the final position; crossing of the intermediate point at the specified instant of time is not guaranteed, though.

Figure 4.10 illustrates the time history of position, velocity and acceleration obtained with the data: $q_i = 0$, $q_f = 3\pi/2$, $t_i = 0$, $t_f = 2$. The intermediate point is located at $q = \pi$ with $t = 1$, the maximum acceleration values in the two segments are respectively $|\ddot{q}_c| = 6\pi$ and $|\ddot{q}_c| = 3\pi$, and the time anticipation is 0.18. As predicted, with time anticipation, the assigned intermediate position becomes a via point with the advantage of an overall shorter time duration. Notice, also, that velocity does not vanish at the intermediate point.

4.3 Operational Space Trajectories

A joint space trajectory planning algorithm generates a time sequence of values for the joint variables $\boldsymbol{q}(t)$ so that the manipulator is taken from the initial to the final configuration, eventually by moving through a sequence of intermediate configurations. The resulting end-effector motion is not easily predictable, in view of the nonlinear effects introduced by direct kinematics. Whenever it is desired that the end-effector motion follows a geometrically specified path in the *operational space*, it is necessary to plan trajectory execution directly in the same space. Planning can be done either by interpolating a sequence of prescribed path points or by generating the analytical motion primitive and the relative trajectory in a punctual way.

In both cases, the time sequence of the values attained by the operational space variables is utilized in real time to obtain the corresponding sequence of values of the joint space variables, via an inverse kinematics algorithm. In this regard, the computational complexity induced by trajectory generation in the operational space and related kinematic inversion sets an upper limit on the maximum sampling rate to generate the above sequences. Since these sequences constitute the reference inputs to the motion control system, a linear *microinterpolation* is typically carried out. In this way, the frequency at which reference inputs are updated is increased so as to enhance dynamic performance of the system.

Whenever the path is not to be followed exactly, its characterization can be performed through the assignment of N points specifying the values of the variables $\boldsymbol{x}_e$ chosen to describe the end-effector pose in the operational space at given time instants t_k, for $k = 1, \dots, N$. Similar to what was presented in the above sections, the trajectory is generated by determining a smooth interpolating vector function between the various path points. Such a function

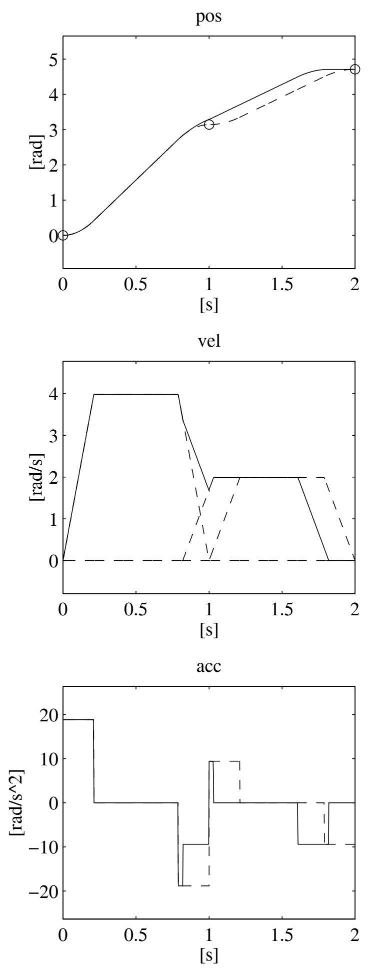

Fig. 4.10. Time history of position, velocity and acceleration with a timing law of interpolating linear polynomials with parabolic blends obtained by anticipating the generation of the second segment of trajectory

can be computed by applying to each component of $\boldsymbol{x}_e$ any of the interpolation techniques illustrated in Sect. 4.2.2 for the single joint variable.

Therefore, for given path (or via) points $\boldsymbol{x}_e(t_k)$, the corresponding components $x_{ei}(t_k)$, for $i = 1, \ldots r$ (where r is the dimension of the operational space of interest) can be interpolated with a sequence of cubic polynomials, a sequence of linear polynomials with parabolic blends, and so on.

On the other hand, if the end-effector motion has to follow a prescribed trajectory of motion, this must be expressed analytically. It is then necessary to refer to motion primitives defining the geometric features of the path and time primitives defining the timing law on the path itself.

4.3.1 Path Primitives

For the definition of *path primitives* it is convenient to refer to the parametric description of paths in space. Then let $\boldsymbol{p}$ be a (3×1) vector and $\boldsymbol{f}(\sigma)$ a continuous vector function defined in the interval $[\sigma_i, \sigma_f]$. Consider the equation

$$\boldsymbol{p} = \boldsymbol{f}(\sigma); \tag{4.31}$$

with reference to its geometric description, the sequence of values of $\boldsymbol{p}$ with σ varying in $[\sigma_i, \sigma_f]$ is termed *path* in space. The equation in (4.31) defines the *parametric representation* of the path Γ and the scalar σ is called parameter. As σ increases, the point $\boldsymbol{p}$ moves on the path in a given direction. This direction is said to be the direction induced on Γ by the parametric representation (4.31). A path is closed when $\boldsymbol{p}(\sigma_f) = \boldsymbol{p}(\sigma_i)$; otherwise it is open.

Let $\boldsymbol{p}_i$ be a point on the open path Γ on which a direction has been fixed. The *arc length* s of the generic point $\boldsymbol{p}$ is the length of the arc of Γ with extremes $\boldsymbol{p}$ and $\boldsymbol{p}_i$ if $\boldsymbol{p}$ follows $\boldsymbol{p}_i$, the opposite of this length if $\boldsymbol{p}$ precedes $\boldsymbol{p}_i$. The point $\boldsymbol{p}_i$ is said to be the origin of the arc length ($s = 0$).

From the above presentation it follows that to each value of s a well-determined path point corresponds, and then the arc length can be used as a parameter in a different parametric representation of the path Γ:

$$\boldsymbol{p} = \boldsymbol{f}(s); \tag{4.32}$$

the range of variation of the parameter s will be the sequence of arc lengths associated with the points of Γ.

Consider a path Γ represented by (4.32). Let $\boldsymbol{p}$ be a point corresponding to the arc length s. Except for special cases, $\boldsymbol{p}$ allows the definition of three unit vectors characterizing the path. The orientation of such vectors depends exclusively on the path geometry, while their direction depends also on the direction induced by (4.32) on the path.

The first of such unit vectors is the *tangent unit vector* denoted by $\boldsymbol{t}$. This vector is oriented along the direction induced on the path by s.

The second unit vector is the *normal unit vector* denoted by $\boldsymbol{n}$. This vector is oriented along the line intersecting $\boldsymbol{p}$ at a right angle with $\boldsymbol{t}$ and lies in the so-called *osculating plane* $\mathcal{O}$ (Fig. 4.11); such plane is the limit position of the plane containing the unit vector $\boldsymbol{t}$ and a point $\boldsymbol{p}' \in \Gamma$ when $\boldsymbol{p}'$ tends to $\boldsymbol{p}$ along the path. The direction of $\boldsymbol{n}$ is so that the path Γ, in the neighbourhood of $\boldsymbol{p}$ with respect to the plane containing $\boldsymbol{t}$ and normal to $\boldsymbol{n}$, lies on the same side of $\boldsymbol{n}$.

The third unit vector is the *binormal unit vector* denoted by $\boldsymbol{b}$. This vector is so that the frame $(\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b})$ is right-handed (Fig. 4.11). Notice that it is not always possible to define uniquely such a frame.

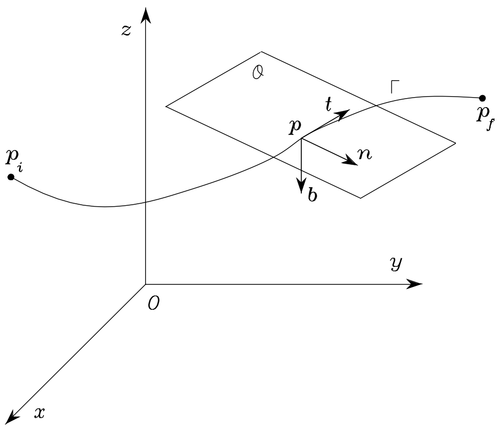

Fig. 4.11. Parametric representation of a path in space

It can be shown that the above three unit vectors are related by simple relations to the path representation Γ as a function of the arc length. In particular, it is

$$\begin{aligned} \boldsymbol{t} &= \frac{d\boldsymbol{p}}{ds} \\ \boldsymbol{n} &= \frac{1}{\left\| \dfrac{d^2\boldsymbol{p}}{ds^2} \right\|} \frac{d^2\boldsymbol{p}}{ds^2} \\ \boldsymbol{b} &= \boldsymbol{t} \times \boldsymbol{n}. \end{aligned} \tag{4.33}$$

Typical path parametric representations are reported below which are useful for trajectory generation in the operational space.

Rectilinear path

Consider the linear segment connecting point $\boldsymbol{p}_i$ to point $\boldsymbol{p}_f$. The parametric representation of this path is

$$\boldsymbol{p}(s) = \boldsymbol{p}_i + \frac{s}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i). \tag{4.34}$$

Notice that $\boldsymbol{p}(0) = \boldsymbol{p}_i$ and $\boldsymbol{p}(\|\boldsymbol{p}_f - \boldsymbol{p}_i\|) = \boldsymbol{p}_f$. Hence, the direction induced on Γ by the parametric representation (4.34) is that going from $\boldsymbol{p}_i$ to $\boldsymbol{p}_f$. Differentiating (4.34) with respect to s gives

$$\frac{d\boldsymbol{p}}{ds} = \frac{1}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) \tag{4.35}$$

$$\frac{d^2\boldsymbol{p}}{ds^2} = \boldsymbol{0}. \tag{4.36}$$

In this case it is not possible to define the frame $(\boldsymbol{t}, \boldsymbol{n}, \boldsymbol{b})$ uniquely.

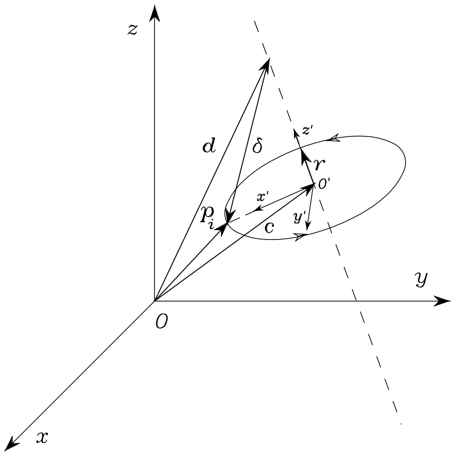

Fig. 4.12. Parametric representation of a circle in space

Circular path

Consider a circle Γ in space. Before deriving its parametric representation, it is necessary to introduce its significant parameters. Suppose that the circle is specified by assigning (Fig. 4.12):

- the unit vector of the circle axis $\boldsymbol{r}$,
- the position vector $\boldsymbol{d}$ of a point along the circle axis,
- the position vector $\boldsymbol{p}_i$ of a point on the circle.

With these parameters, the position vector $\boldsymbol{c}$ of the centre of the circle can be found. Let $\boldsymbol{\delta} = \boldsymbol{p}_i - \boldsymbol{d}$; for $\boldsymbol{p}_i$ not to be on the axis, i.e., for the circle not to degenerate into a point, it must be

$$|\boldsymbol{\delta}^T \boldsymbol{r}| < \|\boldsymbol{\delta}\|;$$

in this case it is

$$\boldsymbol{c} = \boldsymbol{d} + (\boldsymbol{\delta}^T \boldsymbol{r})\boldsymbol{r}. \tag{4.37}$$

It is now desired to find a parametric representation of the circle as a function of the arc length. Notice that this representation is very simple for a suitable choice of the reference frame. To see this, consider the frame O'–$x'y'z'$, where O' coincides with the centre of the circle, axis x' is oriented along the direction of the vector $\boldsymbol{p}_i - \boldsymbol{c}$, axis z' is oriented along $\boldsymbol{r}$ and axis y' is chosen so as to complete a right-handed frame. When expressed in this reference frame, the parametric representation of the circle is

$$\boldsymbol{p}'(s) = \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix}, \tag{4.38}$$

where $\rho = \|\boldsymbol{p}_i - \boldsymbol{c}\|$ is the radius of the circle and the point $\boldsymbol{p}_i$ has been assumed as the origin of the arc length. For a different reference frame, the path representation becomes

$$\boldsymbol{p}(s) = \boldsymbol{c} + \boldsymbol{R}\boldsymbol{p}'(s), \tag{4.39}$$

where $\boldsymbol{c}$ is expressed in the frame $O\text{–}xyz$ and $\boldsymbol{R}$ is the rotation matrix of frame $O'\text{–}\,x'y'z'$ with respect to frame $O\text{–}xyz$ which, in view of (2.3), can be written as

$$\boldsymbol{R} = [\,\boldsymbol{x}' \quad \boldsymbol{y}' \quad \boldsymbol{z}'\,];$$

$\boldsymbol{x}'$, $\boldsymbol{y}'$, $\boldsymbol{z}'$ indicate the unit vectors of the frame expressed in the frame $O\text{–}xyz$. Differentiating (4.39) with respect to s gives

$$\frac{d\boldsymbol{p}}{ds} = \boldsymbol{R}\begin{bmatrix} -\sin(s/\rho) \\ \cos(s/\rho) \\ 0 \end{bmatrix} \tag{4.40}$$

$$\frac{d^2\boldsymbol{p}}{ds^2} = \boldsymbol{R}\begin{bmatrix} -\cos(s/\rho)/\rho \\ -\sin(s/\rho)/\rho \\ 0 \end{bmatrix}. \tag{4.41}$$

4.3.2 Position

Let $\boldsymbol{x}_e$ be the vector of operational space variables expressing the *pose* of the manipulator's end-effector as in (2.80). Generating a trajectory in the operational space means to determine a function $\boldsymbol{x}_e(t)$ taking the end-effector frame from the initial to the final pose in a time t_f along a given path with a specific motion timing law. First, consider end-effector position. Orientation will follow.

Let $\boldsymbol{p}_e = \boldsymbol{f}(s)$ be the (3×1) vector of the parametric representation of the path Γ as a function of the arc length s; the origin of the end-effector frame moves from $\boldsymbol{p}_i$ to $\boldsymbol{p}_f$ in a time t_f. For simplicity, suppose that the origin of the arc length is at $\boldsymbol{p}_i$ and the direction induced on Γ is that going from $\boldsymbol{p}_i$ to $\boldsymbol{p}_f$. The arc length then goes from the value $s = 0$ at $t = 0$ to the value $s = s_f$ (path length) at $t = t_f$. The timing law along the path is described by the function $s(t)$.

In order to find an analytical expression for $s(t)$, any of the above techniques for joint trajectory generation can be employed. In particular, either a cubic polynomial or a sequence of linear segments with parabolic blends can be chosen for $s(t)$.

It is worth making some remarks on the time evolution of $\boldsymbol{p}_e$ on Γ, for a given timing law $s(t)$. The velocity of point $\boldsymbol{p}_e$ is given by the time derivative of $\boldsymbol{p}_e$

$$\dot{\boldsymbol{p}}_e = \dot{s}\frac{d\boldsymbol{p}_e}{ds} = \dot{s}\boldsymbol{t},$$

where $\boldsymbol{t}$ is the tangent vector to the path at point $\boldsymbol{p}$ in (4.33). Then, $\dot{s}$ represents the magnitude of the velocity vector relative to point $\boldsymbol{p}$, taken with the positive or negative sign depending on the direction of $\dot{\boldsymbol{p}}$ along $\boldsymbol{t}$. The magnitude of $\dot{\boldsymbol{p}}$ starts from zero at $t = 0$, then it varies with a parabolic or trapezoidal profile as per either of the above choices for $s(t)$, and finally it returns to zero at $t = t_f$.

As a first example, consider the segment connecting point $\boldsymbol{p}_i$ with point $\boldsymbol{p}_f$. The parametric representation of this path is given by (4.34). Velocity and acceleration of $\boldsymbol{p}_e$ can be easily computed by recalling the rule of differentiation of compound functions, i.e.,

$$\dot{\boldsymbol{p}}_e = \frac{\dot{s}}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) = \dot{s}\boldsymbol{t} \tag{4.42}$$

$$\ddot{\boldsymbol{p}}_e = \frac{\ddot{s}}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) = \ddot{s}\boldsymbol{t}. \tag{4.43}$$

As a further example, consider a circle Γ in space. From the parametric representation derived above, in view of (4.40), (4.41), velocity and acceleration of point $\boldsymbol{p}_e$ on the circle are

$$\dot{\boldsymbol{p}}_e = \boldsymbol{R}\begin{bmatrix} -\dot{s}\sin(s/\rho) \\ \dot{s}\cos(s/\rho) \\ 0 \end{bmatrix} \tag{4.44}$$

$$\ddot{\boldsymbol{p}}_e = \boldsymbol{R}\begin{bmatrix} -\dot{s}^2\cos(s/\rho)/\rho - \ddot{s}\sin(s/\rho) \\ -\dot{s}^2\sin(s/\rho)/\rho + \ddot{s}\cos(s/\rho) \\ 0 \end{bmatrix}. \tag{4.45}$$

Notice that the velocity vector is aligned with $\boldsymbol{t}$, and the acceleration vector is given by two contributions: the first is aligned with $\boldsymbol{n}$ and represents the centripetal acceleration, while the second is aligned with $\boldsymbol{t}$ and represents the tangential acceleration.

Finally, consider the path consisting of a sequence of $N + 1$ points, $\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N$, connected by N segments. A feasible parametric representation of the overall path is the following:

$$\boldsymbol{p}_e = \boldsymbol{p}_0 + \sum_{j=1}^{N} \frac{s_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}), \tag{4.46}$$

In (4.46) s_j is the arc length associated with the j-th segment of the path, connecting point $\boldsymbol{p}_{j-1}$ to point $\boldsymbol{p}_j$, defined as

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} \\ s'_j(t) & t_{j-1} < t < t_j \\ \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\| & t_j \le t \le t_f, \end{cases} \tag{4.47}$$

where $t_0 = 0$ and $t_N = t_f$ are respectively the initial and final time instants of the trajectory, t_j is the time instant corresponding to point $\boldsymbol{p}_j$ and $s'_j(t)$ can be an analytical function of cubic polynomial type, linear type with parabolic blends, and so forth, which varies continuously from the value $s'_j = 0$ at $t = t_{j-1}$ to the value $s'_j = \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|$ at $t = t_j$.

The velocity and acceleration of $\boldsymbol{p}_e$ can be easily found by differentiating (4.46):

$$\dot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\dot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \dot{s}_j \boldsymbol{t}_j \tag{4.48}$$

$$\ddot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\ddot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \ddot{s}_j \boldsymbol{t}_j, \tag{4.49}$$

where $\boldsymbol{t}_j$ is the tangent unit vector of the j-th segment.

Because of the discontinuity of the first derivative at the path points between two non-aligned segments, the manipulator will have to stop and then go along the direction of the following segment. Assumed a relaxation of the constraint to pass through the path points, it is possible to avoid a manipulator stop by connecting the segments near the above points, which will then be named *operational space via points* so as to guarantee, at least, continuity of the first derivative.

As already illustrated for planning of interpolating linear polynomials with parabolic blends passing by the via points in the joint space, the use of trapezoidal velocity profiles for the arc lengths allows the development of a rather simple planning algorithm

In detail, it will be sufficient to properly anticipate the generation of the single segments, before the preceding segment has been completed. This leads to modifying (4.47) as follows:

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} - \Delta t_j \\ s'_j(t + \Delta t_j) & t_{j-1} - \Delta t_j < t < t_j - \Delta t_j \\ \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\| & t_j - \Delta t_j \le t \le t_f - \Delta t_N, \end{cases} \tag{4.50}$$

where Δt_j is the time advance at which the j-th segment is generated, which can be recursively evaluated as

$$\Delta t_j = \Delta t_{j-1} + \delta t_j,$$

with $j = 1, \ldots, N$ e $\Delta t_0 = 0$. Notice that this time advance is given by the sum of two contributions: the former, Δt_{j-1}, accounts for the sum of the time advances at which the preceding segments have been generated, while the latter, δt_j, is the time advance at which the generation of the current segment starts.

4.3.3 Orientation

Consider now end-effector orientation. Typically, this is specified in terms of the rotation matrix of the (time-varying) end-effector frame with respect to the base frame. As is well known, the three columns of the rotation matrix represent the three unit vectors of the end-effector frame with respect to the base frame. To generate a trajectory, however, a linear interpolation on the unit vectors $\boldsymbol{n}_e$, $\boldsymbol{s}_e$, $\boldsymbol{a}_e$ describing the initial and final orientation does not guarantee orthonormality of the above vectors at each instant of time.

Euler angles

In view of the above difficulty, for trajectory generation purposes, orientation is often described in terms of the Euler angles triplet $\boldsymbol{\phi}_e = (\varphi, \vartheta, \psi)$ for which a timing law can be specified. Usually, $\boldsymbol{\phi}_e$ moves along the segment connecting its initial value $\boldsymbol{\phi}_i$ to its final value $\boldsymbol{\phi}_f$. Also in this case, it is convenient to choose a cubic polynomial or a linear segment with parabolic blends timing law. In this way, in fact, the angular velocity $\boldsymbol{\omega}_e$ of the time-varying frame, which is related to $\dot{\boldsymbol{\phi}}_e$ by the linear relationship (3.64), will have continuous magnitude.

Therefore, for given $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_f$ and timing law, the position, velocity and acceleration profiles are

$$\begin{aligned} \boldsymbol{\phi}_e &= \boldsymbol{\phi}_i + \frac{s}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i) \\ \dot{\boldsymbol{\phi}}_e &= \frac{\dot{s}}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i) \\ \ddot{\boldsymbol{\phi}}_e &= \frac{\ddot{s}}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i); \end{aligned} \tag{4.51}$$

where the timing law for $s(t)$ has to be specified. The three unit vectors of the end-effector frame can be computed — with reference to Euler angles ZYZ — as in (2.18), the end-effector frame angular velocity as in (3.64), and the angular acceleration by differentiating (3.64) itself.

Angle and axis

An alternative way to generate a trajectory for orientation of clearer interpretation in the Cartesian space can be derived by resorting to the the angle and axis description presented in Sect. 2.5. Given two coordinate frames in the Cartesian space with the same origin and different orientation, it is always possible to determine a unit vector so that the second frame can be obtained from the first frame by a rotation of a proper angle about the axis of such unit vector.

Let $\boldsymbol{R}_i$ and $\boldsymbol{R}_f$ denote respectively the rotation matrices of the initial frame O_i–$x_iy_iz_i$ and the final frame O_f–$x_fy_fz_f$, both with respect to the base frame. The rotation matrix between the two frames can be computed by recalling that $\boldsymbol{R}_f = \boldsymbol{R}_i\boldsymbol{R}^i_f$; the expression in (2.5) leads to

$$\boldsymbol{R}^i_f = \boldsymbol{R}_i^T\boldsymbol{R}_f = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.$$

If the matrix $\boldsymbol{R}^i(t)$ is defined to describe the transition from $\boldsymbol{R}_i$ to $\boldsymbol{R}_f$, it must be $\boldsymbol{R}^i(0) = \boldsymbol{I}$ and $\boldsymbol{R}^i(t_f) = \boldsymbol{R}^i_f$. Hence, the matrix $\boldsymbol{R}^i_f$ can be expressed as the rotation matrix about a fixed axis in space; the unit vector $\boldsymbol{r}^i$ of the axis and the angle of rotation ϑ_f can be computed by using (2.27):

$$\vartheta_f = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \tag{4.52}$$

$$\boldsymbol{r} = \frac{1}{2\sin\vartheta_f}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{4.53}$$

for $\sin\vartheta_f \neq 0$.

The matrix $\boldsymbol{R}^i(t)$ can be interpreted as a matrix $\boldsymbol{R}^i(\vartheta(t), \boldsymbol{r}^i)$ and computed via (2.25); it is then sufficient to assign a timing law to ϑ, of the type of those presented for the single joint with $\vartheta(0) = 0$ and $\vartheta(t_f) = \vartheta_f$, and compute the components of $\boldsymbol{r}^i$ from (4.52). Since $\boldsymbol{r}^i$ is constant, the resulting velocity and acceleration are respectively

$$\boldsymbol{\omega}^i = \dot{\vartheta}\,\boldsymbol{r}^i \tag{4.54}$$

$$\dot{\boldsymbol{\omega}}^i = \ddot{\vartheta}\,\boldsymbol{r}^i. \tag{4.55}$$

Finally, in order to characterize the end-effector orientation trajectory with respect to the base frame, the following transformations are needed:

$$\begin{aligned} \boldsymbol{R}_e(t) &= \boldsymbol{R}_i\boldsymbol{R}^i(t) \\ \boldsymbol{\omega}_e(t) &= \boldsymbol{R}_i\boldsymbol{\omega}^i(t) \\ \dot{\boldsymbol{\omega}}_e(t) &= \boldsymbol{R}_i\dot{\boldsymbol{\omega}}^i(t). \end{aligned}$$

Once a path and a trajectory have been specified in the operational space in terms of $\boldsymbol{p}_e(t)$ and $\boldsymbol{\phi}_e(t)$ or $\boldsymbol{R}_e(t)$, inverse kinematics techniques can be used to find the corresponding trajectories in the joint space $\boldsymbol{q}(t)$.

Bibliography

Trajectory planning for robot manipulators has been addressed since the first works in the field of robotics [178]. The formulation of the interpolation problem of the path points by means of different classes of functions has been suggested in [26].

The generation of motion trajectories through sequences of points in the joint space using splines is due to [131]. Alternative formulations for this problem are found in [56]. For a complete treatment of splines, including geometric properties and computational aspects, see [54]. In [155] a survey on the functions employed for trajectory planning of a single motion axis is given, which accounts for performance indices and effects of unmodelled flexible dynamics.

Cartesian space trajectory planning and the associated motion control problem have been originally treated in [179]. The systematic management of the motion by the via points using interpolating linear polynomials with parabolic blends has been proposed in [229]. A detailed presentation of the general aspects of the geometric primitives that can be utilized in robotics to define Cartesian space paths can be found in the computer graphics text [73].

Problems

4.1. Compute the joint trajectory from $q(0) = 1$ to $q(2) = 4$ with null initial and final velocities and accelerations.

4.2. Compute the timing law $q(t)$ for a joint trajectory with velocity profile of the type $\dot{q}(t) = k(1 - \cos(at))$ from $q(0) = 0$ to $q(2) = 3$.

4.3. Given the values for the joint variable: $q(0) = 0$, $q(2) = 2$, and $q(4) = 3$, compute the two fifth-order interpolating polynomials with continuous velocities and accelerations.

4.4. Show that the matrix $\boldsymbol{A}$ in (4.28) has a tridiagonal band structure.

4.5. Given the values for the joint variable: $q(0) = 0$, $q(2) = 2$, and $q(4) = 3$, compute the cubic interpolating spline with null initial and final velocities and accelerations.

4.6. Given the values for the joint variable: $q(0) = 0$, $q(2) = 2$, and $q(4) = 3$, find the interpolating polynomial with linear segments and parabolic blends with null initial and final velocities.

4.7. Find the timing law $\boldsymbol{p}(t)$ for a Cartesian space rectilinear path with trapezoidal velocity profile from $\boldsymbol{p}(0) = [\,0 \quad 0.5 \quad 0\,]^T$ to $\boldsymbol{p}(2) = [\,1 \quad -0.5 \quad 0\,]^T$.

4.8. Find the timing law $\boldsymbol{p}(t)$ for a Cartesian space circular path with trapezoidal velocity profile from $\boldsymbol{p}(0) = [\,0 \quad 0.5 \quad 1\,]^T$ to $\boldsymbol{p}(2) = [\,0 \quad -0.5 \quad 1\,]^T$; the circle is located in the plane $x = 0$ with centre at $\boldsymbol{c} = [\,0 \quad 0 \quad 1\,]^T$ and radius $\rho = 0.5$, and is executed clockwise for an observer aligned with x.