• Trajectory Planning
  – Control Policies
  – Desired Trajectories
  – Optimization Methods
  – Pattern Generators

• Reading Assignment for Next Class
  See http://www-slab.usc.edu/courses/CS545
Learning Policies is the Goal of Learning Control

- Policy:

\[ u(t) = p(x(t), t, a) \]
Dynamic Programming & Reinforcement Learning

- Dynamic Programming
  - requires a model of the movement system
- Reinforcement Learning
  - can work without models of the movement system
- Essentials
  - both techniques require to learn a high-dimensional “value function” that assesses the quality of an action $u$ in a state $x$
  - learning the value function is a complex nonstationary, nonlinear learning process
  - both methods die the curse of dimensionality

$$V = \max_u \left[ r(x,u) + \tau \frac{\partial V(x)}{\partial x} f(x,u) \right]$$ (HJB-Eqn.)
Desired Trajectories

- **Essentials**
  - prescribe a desired trajectory
    \[
    (\theta, \dot{\theta})_{\text{desired}} = f(\xi_{\text{initial}}, \xi_{\text{target}}, t)
    \]
  - convert desired trajectory into a (time-dependent) control policy, e.g., by PD-controller
    \[
    u = p(x, t, a) = k_{q} (q(t)_{\text{desired}} - q) + k_{\dot{q}} (\dot{q}(t)_{\text{desired}} - \dot{q})
    \]
- **Problems**
  - Where do desired trajectories come from
  - How to accomplish reactive control
  - How to generalize to new tasks or new situations
Desired Trajectories (cont’d)

• There is a difference between PATH and TRAJECTORY planning
  – A trajectory involves geometry AND time
  – A path involves only geometry

• Planning can happen either in joint or operational space

\[ x_d = g(t, a) \]

\[ \text{or} \]

\[ q_d = f(t, a) \]

• There is usually an infinity of possible desired trajectories

• How is the desired trajectory represented?
  + Every point in time?
  + Only start & final point?
  + Via points?

• Movement Primitives
Joint Space Planning

• What could one plan?
  – Arbitrary trajectories from start to end
  – Trapezoidal (or any another kind of) velocity profiles
  – Polynomials:
    + 1.order: straight lines
    + 2.order: parabolas
    + 3.order: cubic splines
    + 5 order: quintic splines
  + Interesting:
    ◆ Analyze the shape of the trajectories in position, velocity, acceleration, and jerk space.
    ◆ How many constraints are needed to specify a trajectory
Example: Cubic Polynomial

- Cubic Polynomial:
  \[ q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \]
  \[ \dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \]
  \[ \ddot{q}(t) = 2a_2 + 6a_3 t \]

- Given: Start & Endpoint
  \[ q_s, q_f \]

- Plan a cubic polynomial through the start and endpoint
  - Two additional constraints are needed, for instance:
    \[ \dot{q}_s, \dot{q}_f \text{ or } \dot{q}_s, \ddot{q}_s \text{ or } \dot{q}_f, \ddot{q}_f \]
  - Determine the coefficients by using 4 boundary conditions, e.g.,
    \[ q_s = a_0 \]
    \[ \dot{q}_s = a_1 \]
    \[ q_f = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \]
    \[ \dot{q}_f = a_1 + 2a_2 t + 3a_3 t^2 \]
Planning Complex Paths

- Prescribe a set of via-points
- Plan simple trajectories between via-points
- Ensure smooth transitions between trajectory segments
  - E.g., the tangent of two adjacent trajectory segments should match
Optimization Approaches to Desired Trajectories

• Given:
  – “hard constraints”, e.g.,
    \[ q_s, q_f, t \]
  – “soft constraints”, i.e., an optimization criterion

\[ J = \int_{0}^{t} g(q, \dot{q}, \ldots) \, dt \]

• Goal:
  – Find the trajectory that fulfills the hard constraints while minimizing (or maximizing) the soft constraint

• Solution Methods:
  – Calculus of Variation
  – Dynamic Programming
Optimization Approaches

Examples

• Minimum kinetic energy

\[ J = \int_{0}^{\tau} q^2 \, dt \]

+ Results in a quadratic polynomial as solution

• Minimum Jerk

\[ J = \int_{0}^{\tau} \dddot{q}^2 \, dt \]

+ Results in a quintic polynomial as solution

• Minimum Torque Change

\[ J = \int_{0}^{\tau} \ddot{u}^2 \, dt \]

+ Results in something that does not have an analytical description
Operational Space Planning

• All joint space planning methods can also be used in operational space
• Inverse kinematics is needed to convert operational space trajectories into joint space
• The resulting joint space motion is usually quite complex
• Geometric problems can arise:
  – Intermediate points are unreachable
  – High joint space motion near singular postures
  – Start and goal reachable in different solutions
Examples of Geometric Problems
Pattern Generators for Desired Trajectories

- Use Pattern Generators to Create Kinematic Trajectory Plans
  - Use open parameters in pattern generator to generate different movement durations and target settings
Pattern Generators for Trajectory Planning

• What is a pattern generator?
  – A dynamical system (differential equation) with a particular behavior
    + E.g.: Reaching movement can be interpreted as a point attractive behavior:

\[
\dot{q}_d = a (q_f - q_d)
\]

\[
\text{Speed} \quad \text{Target}
\]

• What is the advantage of a pattern generator?
  – Independent of initial conditions
  – Online planning
  – Online modification through additional “coupling” terms. i.e., planning can react to sensory input

\[
\dot{q}_d = a (q_f - q_d) + b (q_d - q)
\]
Pattern Generators for Trajectory Planning

• Disadvantages of Pattern Generators
  – Analysis of behavior is non trivial
  – Need to integrate the equation of motion of the pattern generator at sufficiently high frequency
  – Exact shape of desired trajectories that are generated by the pattern generator are not easy to predict if external coupling is added
  – Modeling of with pattern generators usually requires the manipulation of nonlinear dynamical equations, which is non trivial again
Pattern Generators for Rhythmic and Discrete Movement

Discrete Movement

\[ \Delta v_1 = [t_1 - p_{1,r}]^+ \quad \Delta v_2 = [t_2 - p_{2,r}]^+ \quad \theta_r = p_{1,r} = -p_{2,r} \]

\[ \dot{x}_1 = -a_x x_1 + (v_1 - x_1)c_r + C_{1,r} \quad \dot{x}_2 = -a_x x_2 + (v_1 - x_2)c_r + C_{2,r} \]

\[ \dot{y}_1 = -a_y y_1 + (x_1 - y_1)c_r \quad \dot{y}_2 = -a_y y_2 + (x_1 - y_2)c_r \]

\[ \dot{r}_1 = a_r(-r_1 + (1-r_1)b v_1) \quad \dot{r}_2 = a_r(-r_2 + (1-r_2)b v_2) \]

\[ \dot{z}_1 = -a_z z_1 + (y_1 - z_1)(1-r_1)c_r \quad \dot{z}_2 = -a_z z_2 + (y_2 - z_2)(1-r_2)c_r \]

\[ \dot{p}_{1,r} = a_p c_r(z_1 - z_2) \quad \dot{p}_{2,r} = a_p c_r(z_2 - z_1) \]

Rhythmic Movement

\[ \Delta \omega_1 = [A - (p_1 - p_{1,r})]^+ \quad \Delta \omega_2 = [A - (p_2 - p_{2,r})]^+ \]

\[ \dot{\xi}_1 = a_\xi (-\xi_1 + \Delta \omega_1) \quad \dot{\xi}_2 = a_\xi (-\xi_2 + \Delta \omega_2) \]

\[ \dot{\psi}_1 = -a_\psi \psi_1 + (\xi_1 - \psi_1 - b \zeta_1 - w[\psi_2]^+ + C_{1,o})c_o \quad \dot{\psi}_2 = -a_\psi \psi_2 + (\xi_2 - \psi_2 - b \zeta_2 - w[\psi_1]^+ + C_{2,o})c_o \]

\[ \dot{\zeta}_1 = \frac{1}{5}(-a_\zeta \xi_1 + ([\psi_1]^+ - \zeta_1)c_o) \quad \dot{\zeta}_2 = \frac{1}{5}(-a_\zeta \xi_2 + ([\psi_2]^+ - \zeta_2)c_o) \]

\[ \dot{\theta} = \dot{\psi}_1 = -p_1 \quad \dot{\theta} = \dot{\psi}_2 = -p_2 \]

\[ \dot{\theta} = \dot{p}_1 = -\dot{p}_2 \]
Example from the Discrete Pattern Generator
Discrete Movements at Different Speeds
Example from the Rhythmic Pattern Generator
CHAPTER 5

TRAJECTORY PLANNING

The previous chapters focused on mathematical modeling of mechanical manipulators in terms of kinematics, differential kinematics and statics, and dynamics. Before studying the problem of controlling a manipulation structure, it is worth presenting the main features of motion planning algorithms for the execution of specific manipulator tasks. 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 a polynomial function interpolating 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 (path motion). 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 chapter ends with the presentation of a technique for dynamic scaling a trajectory which allows adapting trajectory planning to manipulator dynamic characteristics.

5.1 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 unmodeled 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, 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 time 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 joint (end-effector) trajectories in terms of a time sequence of the values attained by position, velocity, and acceleration. A path can be defined either in the joint space or in the operational space. Usually, the latter is preferred since it allows a natural description of the task the manipulator has to perform.

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 time 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 accounting for the presence of path constraints; 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 neighborhood of singular configurations and presence of redundant degrees of freedom, 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.

### 5.2 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 location, possible intermediate locations, and traveling 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 Chapter 9).

The planning algorithm generates a function $q(t)$ 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 be not very demanding from a computational viewpoint,
- joint positions and velocities be continuous functions of time (continuity of accelerations may be imposed, too),
- undesirable effects 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 motion); the results are then generalized to the case when also intermediate points along the path are specified (path motion). Without loss of generality, the single joint variable \( q(t) \) is considered.

5.2.1 Point-to-Point Motion

In the 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 to optimize 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. Assuming that rotation is executed through a torque \( \tau \) 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

\[
I \ddot{\omega} = \tau
\]

subject to the condition

\[
\int_0^{t_f} \omega(t) dt = q_f - q_i.
\]

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 \textit{cubic polynomial} can be chosen

\[
q(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0 ,
\]

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:

\[
\begin{align*}
a_0 &= q_i \\
a_1 &= \dot{q}_i \\
a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 &= q_f \\
3a_3 t_f^2 + 2a_2 t_f + a_1 &= \dot{q}_f
\end{align*}
\]

that allows computing the coefficients of the polynomial in \((5.1)\). Fig. 5.1 illustrates the time 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 \textit{fifth} order is needed. The motion time law for the generic joint is then given by

\[
q(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0,
\]

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 \((5.2)\), one obviously gives up minimizing the above performance index.

An alternative approach with time laws of blended polynomial type is frequently adopted in industrial practice, which allows directly verifying whether the resulting velocities and accelerations can be supported by the physical mechanical manipulator.

\(^1\) In fact, recall that the moment of inertia about the joint axis is a function of manipulator configuration.
FIGURE 5.1
Time history of position, velocity, and acceleration with a cubic polynomial time law.

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.

As can be seen from the velocity profiles in Fig. 5.2, 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 \( \dot{q}_c \) 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
FIGURE 5.2
Characterization of a time law with trapezoidal velocity profile in terms of position, velocity, and acceleration.

the (constant) velocity of the linear segment, i.e.,

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

(5.3)

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

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

(5.4)

Combining (5.3) with (5.4) gives

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

(5.5)

Usually, $\ddot{q}_c$ is specified and then, for given $t_f, q_i$ and $q_f$, the solution for $t_c$ is computed from (5.5) as ($t_c \leq t_f/2$)

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

(5.6)

Acceleration is then subject to the constraint

$$
|\dddot{q}_c| \geq \frac{4|q_f - q_i|}{t_f^2}.
$$

(5.7)
When the acceleration $\ddot{q}_c$ is chosen so as to satisfy (5.7) 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, Eq. (5.7) allows imposing a value of acceleration consistent with the trajectory. Then, $t_c$ is computed from (5.6), and the following sequence of polynomials is generated

$$ q(t) = \begin{cases} 
q_i + \frac{1}{2} \ddot{q}_c t^2 & 0 \leq t \leq t_c \\
q_i + \ddot{q}_c t_c (t - t_c / 2) & t_c < t \leq t_f - t_c \\
q_f - \frac{1}{2} \ddot{q}_c (t_f - t)^2 & t_f - t_c < t \leq t_f.
\end{cases} \quad (5.8)$$

Fig. 5.3 illustrates a representation of the motion time 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

$$ \frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \leq \frac{2|q_f - q_i|}{t_f}. \quad (5.9)$$

By recognizing that $\dot{q}_c = \ddot{q}_c t_c$, Eq. (5.5) allows computing $t_c$ as

$$ t_c = \frac{q_i - q_f + \ddot{q}_c t_f}{\ddot{q}_c}, \quad (5.10)$$

and thus the resulting acceleration is

$$ \ddot{q}_c = \frac{q_f^2}{q_i - q_f + \ddot{q}_c t_f}. \quad (5.11)$$

The computed values of $t_c$ and $\ddot{q}_c$ as in (5.10) and (5.11) allow generating the sequence of polynomials expressed by (5.8).

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.

### 5.2.2 Path Motion

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 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 locations.

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 behavior increases, and this may lead to trajectories which are not natural for the manipulator.
Figure 5.4
Characterization of a trajectory on a given path obtained through interpolating polynomials.

- 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 imposing 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 $P_k(t)$, 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. 5.4). 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 $\ddot{q}(t)$ shall 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 Velocity Constraints 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 \( k = 1, \ldots, N - 1 \):

\[
\begin{align*}
\Pi_k(t_k) &= q_k \\
\Pi_k(t_{k+1}) &= q_{k+1} \\
\dot{\Pi}_k(t_k) &= \dot{q}_k \\
\dot{\Pi}_k(t_{k+1}) &= \dot{q}_{k+1}.
\end{align*}
\]

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

\[
\ddot{\Pi}_k(t_{k+1}) = \ddot{\Pi}_k(t_{k+1})
\]

for \( k = 1, \ldots, N - 2 \).

Fig. 5.5 illustrates the time history of position, velocity, and acceleration obtained with the data: \( q_1 = 0, q_2 = 2\pi, q_3 = \pi/2, q_4 = \pi, t_1 = 0, t_2 = 2, t_3 = 3, t_4 = 5, \dot{q}_1 = 0, \dot{q}_2 = \pi, \dot{q}_3 = -\pi, \) and \( \dot{q}_4 = 0 \). 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:

\[
\begin{align*}
\dot{q}_1 &= 0 \\
\dot{q}_k &= \begin{cases} 
0 & \text{sgn}(v_k) \neq \text{sgn}(v_{k+1}) \\
\frac{1}{2}(v_k + v_{k+1}) & \text{sgn}(v_k) = \text{sgn}(v_{k+1})
\end{cases} \\
\dot{q}_N &= 0,
\end{align*}
\]

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.

Fig. 5.6 illustrates the time history of position, velocity, and acceleration obtained with the following data: \( q_1 = 0, q_2 = 2\pi, q_3 = \pi/2, q_4 = \pi, t_1 = 0, t_2 = 2, t_3 = 3, t_4 = 5, \dot{q}_1 = 0, \) and \( \dot{q}_4 = 0 \). It is easy to recognize that the imposed 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.
FIGURE 5.5
Time history of position, velocity, and acceleration with a time law of interpolating polynomials with velocity constraints at path points.

Given a sequence of $N$ path points, also the acceleration is 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:

\[
\begin{align*}
\Pi_{k-1}(t_k) &= q_k \\
\Pi_{k-1}(t_k) &= \Pi_k(t_k) \\
\dot{\Pi}_{k-1}(t_k) &= \dot{\Pi}_k(t_k) \\
\ddot{\Pi}_{k-1}(t_k) &= \ddot{\Pi}_k(t_k).
\end{align*}
\]

The resulting system for the $N$ path points, including the initial and final points, cannot
FIGURE 5.6
Time history of position, velocity, and acceleration with a time law of interpolating polynomials with computed velocities at path points.

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 $\Pi_0(t_1) = q_1$ and $\Pi_N(t_f) = q_f$ have to be excluded since they are not defined. Also, $\ddot{\Pi}_0(t_1), \ddot{\Pi}_0(t_1), \ddot{\Pi}_N(t_f), \ddot{\Pi}_N(t_f)$ do not have to be counted as polynomials since they are just the imposed values of initial and final velocities and accelerations. In sum, one has $4N - 2$ equations in $4(N - 1)$ unknowns.

The system can be solved only if one eliminates the two equations which allow arbitrarily assigning 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:

$$
\Pi_{k-1}(t_k) = q_k \\
\Pi_k(t_k) = \Pi_k(t_k) \\
\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k) \\
\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k)
$$

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

$$
\Pi_1(t_1) = q_i \\
\dot{\Pi}_1(t_1) = \dot{q}_i \\
\ddot{\Pi}_1(t_1) = \ddot{q}_i \\
\Pi_{N+1}(t_{N+2}) = q_f \\
\dot{\Pi}_{N+1}(t_{N+2}) = \dot{q}_f \\
\ddot{\Pi}_{N+1}(t_{N+2}) = \ddot{q}_f
$$

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

$$
\Pi_{k-1}(t_k) = \Pi_k(t_k) \\
\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k) \\
\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k)
$$

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 $\Pi_k(t)$ is a cubic, its second derivative must be a linear function of time which then can be written as

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

where $\Delta t_k = t_{k+1} - t_k$ indicates the time interval to reach $q_{k+1}$ from $q_k$. By integrating (5.23) twice over time, the generic polynomial can be written as
\[ \Pi_k(t) = \frac{\dddot{\Pi}_k(t_k)}{6\Delta t_k} (t_{k+1} - t)^2 + \frac{\dddot{\Pi}_k(t_{k+1})}{6\Delta t_k} (t - t_{k+1})^2 \\
+ \left( \frac{\Pi_k(t_{k+1})}{\Delta t_k} - \frac{\Delta t_k \dddot{\Pi}_k(t_{k+1})}{6} \right) (t - t_k) \\
+ \left( \frac{\Pi_k(t_k)}{\Delta t_k} - \frac{\Delta t_k \dddot{\Pi}_k(t_k)}{6} \right) (t_{k+1} - t) \quad k = 1, \ldots, N + 1, \]

which depends on the 4 unknowns: \( \Pi_k(t_k), \Pi_k(t_{k+1}), \dddot{\Pi}_k(t_k) \) and \( \dddot{\Pi}_k(t_{k+1}) \).

Notice that the \( N \) variables \( q_k \) for \( k \neq 2, N + 1 \) are given via (5.13), while continuity is imposed for \( q_2 \) and \( q_{N+1} \) via (5.20). By using (5.14), (5.17), and (5.17'), the unknowns in the \( N + 1 \) equations (5.24) reduce to 2(\( N + 2 \)). By observing that Eqs. (5.18) and (5.18') depend on \( q_2 \) and \( q_{N+1} \), and that \( \dddot{q}_i \) and \( \dddot{q}_f \) are given, \( q_2 \) and \( q_{N+1} \) can be computed as a function of \( \dddot{\Pi}_1(t_1) \) and \( \dddot{\Pi}_{N+1}(t_{N+2}) \), respectively. Thus, a number of 2(\( N + 1 \)) unknowns are left.

By accounting for (5.16) and (5.22), and noticing that in (5.19) and (5.19') \( \dddot{q}_i \) and \( \dddot{q}_f \) are given, the unknowns reduce to \( N \).

At this point, Eqs. (5.15) and (5.21) can be utilized to write the system of \( N \) equations in \( N \) unknowns:

\[ \dddot{\Pi}_1(t_2) = \dddot{\Pi}_2(t_2) \\
\vdots \\
\dddot{\Pi}_{N}(t_{N+1}) = \dddot{\Pi}_{N+1}(t_{N+1}). \]

Time-differentiation of (5.24) gives both \( \dddot{\Pi}_k(t_{k+1}) \) and \( \dddot{\Pi}_{k+1}(t_{k+1}) \) for \( k = 1, \ldots, N \), and thus it is possible to write a system of linear equations of the kind

\[ A \begin{bmatrix} \dddot{\Pi}_2(t_2) & \cdots & \dddot{\Pi}_{N+1}(t_{N+1}) \end{bmatrix}^T = b \]

(5.25)

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

\[ A = \begin{bmatrix} a_{11} & a_{12} & \cdots & 0 & 0 \\
\vdots & a_{21} & a_{22} & \cdots & 0 & 0 \\
0 & 0 & \cdots & a_{N-1,N-1} & a_{N-1,N} \\
0 & 0 & \cdots & a_{N,N-1} & a_{NN} \end{bmatrix}, \]

which simplifies the solution to the system. This matrix is the same for all joints, since it depends only on the time intervals \( \Delta t_k \) 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, $\vec{H}_2(t_2)$ can be computed as a function of $\vec{H}_3(t_3)$ and then substituted in the second equation, which then becomes an equation in the unknowns $\vec{H}_3(t_3)$ and $\vec{H}_4(t_4)$. This is carried out forward by transforming all the equations in equations with two unknowns, except the last one which will have $\vec{H}_{N+1}(t_{N+1})$ 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.
Fig. 5.7 illustrates the time history of position, velocity, and acceleration obtained with the data: \( q_1 = 0, q_3 = 2\pi, q_4 = \pi/2, q_6 = \pi, t_1 = 0, t_3 = 2, t_4 = 3, t_6 = 5, \dot{q}_1 = 0, \) and \( \ddot{q}_6 = 0. \) 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 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 \( q(t) \) shall have a parabolic profile (blend) around \( t_k \); as a consequence, the entire trajectory is composed by a sequence of linear and quadratic polynomials, which in turn implies that a discontinuity on \( \ddot{q}(t) \) is tolerated.

Let then \( \Delta t_k = t_{k+1} - t_k \) 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. Let also \( \dot{q}_{k,k+1} \) 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. 5.8. The values of \( q_k, \Delta t_k, \) and \( \Delta t'_k \) are assumed to be given. Velocity and acceleration for the intermediate points are computed as

\[
\begin{align*}
\dot{q}_{k-1,k} &= \frac{q_k - q_{k-1}}{\Delta t_{k-1}} \\
\ddot{q}_k &= \frac{\dot{q}_{k,k+1} - \dot{q}_{k-1,k}}{\Delta t'_k},
\end{align*}
\]  

(5.26)

these equations are straightforward.
FIGURE 5.9
Time history of position, velocity, and acceleration with a time law of interpolating linear polynomials with parabolic blends.

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 \( q_{0,1} = q_{N,N+1} = 0 \) 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. 5.8). 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 \( q_k, \Delta t_k, \) and \( \Delta t'_k \), the values of \( \dot{q}_{k-1,k} \) and \( \ddot{q}_k \) are computed via (5.26) and a sequence of linear polynomials with parabolic blends is
FIGURE 5.10
Time history of position, velocity, and acceleration with a time law of interpolating linear polynomials with parabolic blends obtained by anticipating the generation of the second segment of trajectory.

generated. Their expressions as a function of time are not derived here to avoid further loading of the analytic presentation.

Fig. 5.9 illustrates the time history of position, velocity, and acceleration obtained with the data: \( q_1 = 0, q_2 = 2\pi, q_3 = \pi/2, q_4 = \pi, t_1 = 0, t_2 = 2, t_3 = 3, t_4 = 5, \dot{q}_1 = 0, \) and \( \ddot{q}_4 = 0. \) 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.

The above presented technique 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 developing 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.

Fig. 5.10 illustrates the time history of position, velocity, and acceleration obtained with the data: \( q_i = 0, q_f = 3\pi/2, t_i = 0, \) and \( t_f = 2 \). 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.

### 5.3 OPERATIONAL SPACE TRAJECTORIES

A joint space trajectory planning algorithm generates a time sequence of values for the joint variables \( q(t) \) 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 \( x \) chosen to describe the end-effector location in the operational space at given time instants \( t_k \), for \( k = 1, \ldots, N \). Similarly 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 can be computed by applying to each component of \( x \) any of the interpolation techniques illustrated in Section 5.2.2 for the single joint variable.

Therefore, for given path (or via) points \( x(t_k) \), the corresponding components \( x_i(t_k) \), 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 time law on the path itself.

### 5.3.1 Path Primitives

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

\[
p = f(\sigma);
\]

with reference to its geometric description, the sequence of values of \( p \) with \( \sigma \) varying in \([\sigma_i, \sigma_f]\) is termed *path* in space. Eq. (5.27) defines the parametric representation of the path \( \Gamma \) and the scalar \( \sigma \) is called *parameter*. As \( \sigma \) increases, the point \( p \) moves on the path in a given direction. This direction is said to be the direction induced on \( \Gamma \) by the parametric representation (5.27). A path is closed when \( p(\sigma_f) = p(\sigma_i) \); otherwise it is open.

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

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

\[
p = f(s);
\]

the range of variation of the parameter \( s \) will be the sequence of path coordinates associated with the points of \( \Gamma \).

Consider a path \( \Gamma \) represented by (5.28). Let \( p \) be a point corresponding to the path coordinate \( s \). Except for special cases, \( p \) 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 (5.28) on the path.

The first of such unit vectors is the *tangent unit vector* denoted by \( t \). This vector is oriented along the direction induced on the path by \( s \).
FIGURE 5.11
Parametric representation of a path in space.

The second unit vector is the normal unit vector denoted by \( n \). This vector is oriented along the line intersecting \( p \) at a right angle with \( t \) and lies in the so-called osculating plane \( O \) (Fig. 5.11); such plane is the limit position of the plane containing the unit vector \( t \) and a point \( p' \in \Gamma \) when \( p' \) tends to \( p \) along the path. The direction of \( n \) is so that the path \( \Gamma \), in the neighborhood of \( p \) with respect to the plane containing \( t \) and normal to \( n \), lies on the same side of \( n \).

The third unit vector is the binormal unit vector denoted by \( b \). This vector is so that the frame \( (t, n, b) \) is right-handed (Fig. 5.11). Notice that it is not always possible to uniquely define such frame.

It can be shown that the above three unit vectors are related by simple relations to the path representation \( \Gamma \) as a function of the path coordinate. In particular, it is:

\[
\begin{align*}
t &= \frac{dp}{ds} \\
n &= \frac{1}{\left\| \frac{d^2p}{ds^2} \right\|} \frac{d^2p}{ds^2} \\
b &= t \times n.
\end{align*}
\]  

(5.29)

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

**Segment in space.** Consider the linear segment connecting point \( p_i \) to point \( p_f \). The parametric representation of this path is

\[
p(s) = p_i + \frac{s}{\left\| p_f - p_i \right\|}(p_f - p_i).
\]  

(5.30)
Notice that \( p(0) = p_i \) and \( p(\|p_f - p_i\|) = p_f \). Hence, the direction induced on \( \Gamma \) by the parametric representation (5.30) is that going from \( p_i \) to \( p_f \). Differentiating (5.30) with respect to \( s \) gives

\[
\frac{dp}{ds} = \frac{1}{\|p_f - p_i\|}(p_f - p_i),
\]

\[
\frac{d^2p}{ds^2} = 0.
\]

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

**Circle in space.** Consider a circle \( \Gamma \) in space. Before deriving its parametric representation, it is necessary to introduce its significant parameters. Suppose that the circle is specified by assigning (Fig. 5.12):

- the unit vector of the circle axis \( r \),
- the position vector \( d \) of a point along the circle axis,
- the position vector \( p_i \) of a point on the circle.

With these parameters, the position vector \( c \) of the center of the circle can be found. Let \( \delta = p_i - d \); for \( p_i \) not to be on the axis, i.e., for the circle not to degenerate into a point, it must be

\[
|\delta^Tr| < \|\delta\|;
\]

in this case it is

\[
c = d + (\delta^Tr)r.
\]

(5.32)
It is now desired to find a parametric representation of the circle as a function of the path coordinate. 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 center of the circle, axis $x'$ is oriented along the direction of the vector $p_i - c$, axis $z'$ is oriented along $r$ 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

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

(5.33)

where $\rho = ||p_i - c||$ is the radius of the circle and the point $p_i$ has been assumed as the origin of the path coordinate. For a different reference frame, the path representation becomes

$$p(s) = c + Rp'(s),$$

(5.34)

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

$$R = \begin{bmatrix}
x' & y' & z'
\end{bmatrix};$$

$x'$, $y'$, $z'$ indicate the unit vectors of the frame expressed in the frame $O-xyz$. Differentiating (5.34) with respect to $s$ gives

$$\frac{dp}{ds} = R \begin{bmatrix}
-s\sin(s/\rho) \\
\cos(s/\rho) \\
0
\end{bmatrix},$$

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

(5.35)

### 5.3.2 Position and Orientation Trajectories

Let $x$ be the vector of operational space variables expressing position and orientation of the manipulator's end effector as in (2.49). Generating a trajectory in the operational space means to determine a function $x(t)$ taking the end-effector frame from the initial to the final location in a time $t_f$ along a given path with a specific motion time law. First, consider end-effector frame position. Orientation will follow.

**Position.** Let $p = f(s)$ be the $(3 \times 1)$ vector of the parametric representation of the path $\Gamma$ as a function of the path coordinate $s$; the origin of the end-effector frame moves from $p_i$ to $p_f$ in a time $t_f$. For simplicity, suppose that the origin of the path coordinate is at $p_i$ and the direction induced on $\Gamma$ is that going from $p_i$ to $p_f$. The path coordinate then goes from the value $s = 0$ at $t = 0$ to the value $s = s_f$ (path length) at $t = t_f$. The time law along the path is described by the function $s(t)$. 
192 MODELING AND CONTROL OF ROBOT MANIPULATORS

In order to find an analytic 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 \( p \) on \( \Gamma \), for a given time law \( s(t) \). The velocity of point \( p \) is given by the time derivative of \( p \)

\[
p = \dot{s} \frac{dp}{ds} = \dot{s} t,
\]

where \( t \) is the tangent vector to the path at point \( p \) in (5.29). Then, \( \dot{s} \) represents the magnitude of the velocity vector relative to point \( p \), taken with the positive or negative sign depending on the direction of \( \dot{p} \) along \( t \). The magnitude of \( \dot{p} \) 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 \( p_i \) with point \( p_f \). The parametric representation of this path is given by (5.30). Velocity and acceleration of \( p \) can be easily computed by recalling the rule of differentiation of compound functions, i.e.,

\[
\dot{p} = \frac{\dot{s}}{\|p_f - p_i\|} (p_f - p_i) = \dot{s} t
\]

(5.36)

\[
\ddot{p} = \frac{\ddot{s}}{\|p_f - p_i\|} (p_f - p_i) = \ddot{s} t.
\]

As a further example, consider a circle \( \Gamma \) in space. From the parametric representation derived above, in view of (5.35), velocity and acceleration of point \( p \) on the circle are:

\[
\dot{p} = R \begin{bmatrix}
-\dot{s} \sin (s/\rho) \\
\dot{s} \cos (s/\rho) \\
0
\end{bmatrix}
\]

(5.37)

\[
\ddot{p} = R \begin{bmatrix}
-\ddot{s} \cos (s/\rho)/\rho - \ddot{s} \sin (s/\rho) \\
-\dddot{s} \sin (s/\rho)/\rho + \dddot{s} \cos (s/\rho) \\
0
\end{bmatrix}
\]

Notice that the velocity vector is aligned with \( t \), and the acceleration vector is given by two contributions: the first one is aligned with \( n \) and represents the centripetal acceleration, while the second one is aligned with \( t \) and represents the tangential acceleration.

**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 well known, the three columns of the rotation matrix represent the three unit vectors of the end-effector frame \( (n, s, a) \) expressed in the base frame. To generate a trajectory, however, it is not convenient to refer to the rotation matrix to characterize orientation. For instance, a linear interpolation on the unit vectors \( n, s, a \) describing the initial and final orientation does not guarantee orthonormality of the above vectors at each instant of time.
In view of the above difficulty, for trajectory generation purposes, orientation is often described in terms of the angle triplet \( \phi = (\varphi, \vartheta, \psi) \); e.g., Euler angles ZYZ or RPY angles, for which a time law can be specified. Usually, \( \phi \) moves along the segment connecting its initial value \( \phi_i \) to its final value \( \phi_f \). Also in this case, it is convenient to choose a cubic polynomial or a linear segment with parabolic blends time law. In this way, the angular velocity \( \omega \) of the time-varying frame, which is related to \( \phi \) by the linear relationship (3.39), will have continuous magnitude.

Therefore, for given \( \phi_i \) and \( \phi_f \), the position, velocity and acceleration profiles are:

\[
\begin{align*}
\phi &= \phi_i + \frac{s}{\|\phi_f - \phi_i\|} (\phi_f - \phi_i) \\
\dot{\phi} &= \frac{\dot{s}}{\|\phi_f - \phi_i\|} (\phi_f - \phi_i) \\
\ddot{\phi} &= \frac{\ddot{s}}{\|\phi_f - \phi_i\|} (\phi_f - \phi_i),
\end{align*}
\]

(5.38)

where the time 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 (2.23), the end-effector frame angular velocity as in (3.39), and the angular acceleration by differentiating (3.39) itself.

An alternative way to generate a trajectory for orientation of clearer interpretation in the Cartesian space can be derived by resorting to the method of the equivalent axis of rotation presented in Section 2.4. Given two coordinate frames in the Cartesian space with the same origin and different orientation, it is always possible to determine a unit vector \( \mathbf{r} \) so that the second frame can be obtained from the first frame by a rotation of magnitude \( \theta_f \) about the axis of such unit vector.

Let \( R_i \) and \( R_f \) 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 \( R_f = R_i R_i^T \), Eq. (2.5) allows writing

\[
R_i^T R_f = \begin{bmatrix}
    r_{11} & r_{12} & r_{13} \\
    r_{21} & r_{22} & r_{23} \\
    r_{31} & r_{32} & r_{33}
\end{bmatrix}.
\]

If the matrix \( R^i(t) \) is defined to describe the transition from \( R_i \) to \( R_f \), it must be \( R^i(0) = I \) and \( R^i(t_f) = R_f \). Hence, the matrix \( R^i_f \) can be expressed as the rotation matrix about a fixed axis in space; the unit vector \( \mathbf{r} \) of the axis and the angle of rotation \( \theta_f \) can be computed by using (2.21):

\[
\begin{align*}
\theta_f &= \cos^{-1} \left( \frac{r_{11} + r_{22} + r_{33} - 1}{2} \right) \\
\mathbf{r} &= \frac{1}{2 \sin \theta_f} \begin{bmatrix}
    r_{32} - r_{23} \\
    r_{13} - r_{31} \\
    r_{21} - r_{12}
\end{bmatrix}
\end{align*}
\]

(5.39)
for $\sin \vartheta_f \neq 0$.

The matrix $\mathbf{R}_i^e(t)$ can be interpreted as a matrix $\mathbf{R}_i^e(\vartheta(t))$ and computed via (2.19); it is then sufficient to assign a time law to $\vartheta$, of the type of those presented for the single joint with $\vartheta(0) = 0$ and $\dot{\vartheta}(t_f) = \vartheta_f$, and compute the components $r_x, r_y, r_z$ of the unit vector of the fixed axis from (5.39). Therefore,

$$\mathbf{R}(t) = \mathbf{R}_i \mathbf{R}_i^e(\vartheta(t))$$

represents the rotation matrix describing the end-effector orientation as a function of time, with respect to the base frame.

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

### 5.4 DYNAMIC SCALING OF TRAJECTORIES

The existence of dynamic constraints to be taken into account for trajectory generation has been mentioned in Section 5.1. In practice, with reference to the given trajectory time or path shape (segments with high curvature), the trajectories that can be obtained with any of the previously illustrated methods may impose too severe dynamic performance for the manipulator. A typical case is that when the required torques to generate the motion are larger than the maximum torques the actuators can supply. In this case, an infeasible trajectory has to be suitably time-scaled.

Suppose a trajectory has been generated for all the manipulator joints as $q(t)$, for $t \in [0, t_f]$. Computing inverse dynamics allows evaluating the time history of the torques $\tau(t)$ required for the execution of the given motion. By comparing the obtained torques with the torque limits available at the actuators, it is easy to check whether or not the trajectory is actually executable. The problem is then to seek an automatic trajectory dynamic scaling technique—avoiding inverse dynamics recomputation—so that the manipulator can execute the motion on the specified path with a proper time law without exceeding the torque limits.

Consider the manipulator dynamic model as given in (4.41) with $\mathbf{F}_v = \mathbf{O}$, $\mathbf{F}_s = \mathbf{O}$ and $h = 0$, for simplicity. The term $\mathbf{C}(q, \dot{q})$ accounting for centrifugal and Coriolis forces has a quadratic dependence on joint velocities, and thus it can be formally rewritten as

$$\mathbf{C}(q, \dot{q})\ddot{q} = \mathbf{\Gamma}(q)[\dot{q}\ddot{q}], \quad (5.40)$$

where $[\dot{q}\ddot{q}]$ indicates the symbolic notation of the $(n(n + 1)/2 \times 1)$ vector

$$[\dot{q}\ddot{q}] = \begin{bmatrix} \dot{q}_1^2 & \dot{q}_1\dot{q}_2 & \cdots & \dot{q}_{n-1}\dot{q}_n & \dot{q}_n^2 \end{bmatrix}^T;$$

$\mathbf{\Gamma}(q)$ is a proper $(n \times n(n + 1)/2)$ matrix that satisfies (5.40). In view of such position, the manipulator dynamic model can be expressed as

$$\mathbf{B}(q(t))\ddot{q}(t) + \mathbf{\Gamma}(q(t))[\dot{q}(t)\ddot{q}(t)] + \mathbf{g}(q(t)) = \tau(t), \quad (5.41)$$
where the explicit dependence on time $t$ has been shown.

Consider the new variable $\bar{q}(r(t))$ satisfying the equation

$$q(t) = \bar{q}(r(t)), \quad (5.42)$$

where $r(t)$ is a strictly increasing scalar function of time with $r(0) = 0$ and $r(t_f) = \bar{r}_f$. Differentiating (5.42) twice with respect to time provides the following relations

$$\dot{q} = \dot{\bar{r}} \dot{q}(r)$$
$$\ddot{q} = \dot{\bar{r}}^2 \ddot{q}(r) + \ddot{\bar{r}} \dot{q}(r), \quad (5.43)$$

where the prime denotes the derivative with respect to $r$. Substituting (5.43) into (5.41) yields

$$\dot{\bar{r}}^2 \left( B(\bar{q}(r)) \dddot{q}(r) + \Gamma(\bar{q}(r)) [\dot{q}(r) \ddot{q}(r)] \right) + \dot{\bar{r}} B(\bar{q}(r)) \dot{q}(r) + g(\bar{q}(r)) = \tau. \quad (5.44)$$

In (5.41) it is possible to identify the term

$$\tau_s(t) = B(q(t)) \ddot{q}(t) + \Gamma(q(t)) [\dot{q}(t) \ddot{q}(t)], \quad (5.45)$$

representing the torque contribution that depends on velocities and accelerations. Correspondingly, in (5.44) one can set

$$\tau_s(t) = \dot{\bar{r}}^2 \left( B(\bar{q}(r)) \dddot{q}(r) + \Gamma(\bar{q}(r)) [\dot{q}(r) \ddot{q}(r)] \right) + \dot{\bar{r}} B(\bar{q}(r)) \dot{q}(r). \quad (5.46)$$

By analogy with (5.45), it can be written

$$\bar{\tau}_s(r) = B(\bar{q}(r)) \dddot{q}(r) + \Gamma(\bar{q}(r)) [\dot{q}(r) \ddot{q}(r)], \quad (5.47)$$

and then Eq. (5.46) becomes

$$\tau_s(t) = \dot{\bar{r}}^2 \bar{\tau}_s(r) + \dot{\bar{r}} B(\bar{q}(r)) \dot{q}(r). \quad (5.48)$$

Eq. (5.48) gives the relationship between the torque contributions depending on velocities and accelerations required by the manipulator when this is subject to motions having the same path but different time laws, obtained through a time scaling of joint variables as in (5.42).

Gravitational torques have not been considered, since they are a function of the joint positions only, and thus their contribution is not influenced by trajectory time scaling.

The simplest choice for the scaling function $r(t)$ is certainly the linear function

$$r(t) = ct,$$

with $c$ a positive constant. In this case Eq. (5.48) becomes

$$\tau_s(t) = c^2 \bar{\tau}_s(ct),$$
which reveals that a linear time scaling by $c$ causes a scaling of the magnitude of the torques by the coefficient $c^2$. Let $c > 1$: Eq. (5.42) shows that the trajectory described by $\dot{q}(r(t))$, assuming $r = ct$ as the independent variable, has a duration $\bar{t}_f > t_f$ to cover the entire path specified by $q$. Correspondingly, the torque contributions $\tau_c(ct)$ computed as in (5.47) are scaled by the factor $c^2$ with respect to the torque contributions $\tau_c(t)$ required to execute the original trajectory $q(t)$.

With the use of a recursive algorithm for inverse dynamics computation, it is possible to check whether the torques exceed the allowed limits during trajectory execution; obviously, limit violation shall not be caused by the sole gravity torques. It is necessary to find the joint for which the torque has exceeded the limit more than the others, and to compute the torque contribution subject to scaling, which in turn determines the factor $c^2$. It is then possible to compute the time-scaled trajectory as a function of the new time variable $r = ct$ which no longer exceeds torque limits. It should be pointed out, however, that with this kind of linear scaling the entire trajectory may be penalized, even when a torque limit on a single joint is exceeded only for a short interval of time.

PROBLEMS
5.1. Compute the joint trajectory from $q(0) = 1$ to $q(2) = 4$ with null initial and final velocities and accelerations.
5.2. Compute the time 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 $q(2) = 3$.
5.3. 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.
5.4. Show that the matrix $A$ in (5.25) has a tridiagonal band structure.
5.5. 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.
5.6. 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.
5.7. Find the motion time law $p(t)$ for a Cartesian space straight path with trapezoidal velocity profile from $p(0) = [0 0.5 0]T$ to $p(2) = [1 -0.5 0]T$.
5.8. Find the motion time law $p(t)$ for a Cartesian space circular path from $p(0) = [0 0.5 1]T$ to $p(2) = [0 -0.5 1]^T$; the circle is located in the plane $x = 0$ with center at $c = [0 0 1]^T$ and radius $\rho = 0.5$, and is executed clockwise for an observer aligned with $x$.
5.9. For the two-link planar arm of Example 4.2, perform a computer implementation of dynamic linear time scaling along the trajectory of Fig. 4.6, on the assumption of symmetric torque limits of 3000 N-m. Adopt a sampling time of 1 ms.

BIBLIOGRAPHY


