Trajectory Generation with Acceleration Continuity and Velocity and Acceleration Constraints

Benjamin Navarro

September 2020

Polynomial interpolation and trajectory generation

The order of a polynomial used for interpolation or trajectory generation is given by the number of constraints to satisfy. In our case, we need to impose initial and final values as well as their first and second derivatives. This leads to six constraints that can be satisfied by a polynomial composed of six parameters, hence the use of fifth-order polynomials.

In this section we will recall how such functions are described and how their parameters can be computed to satisfy the given constraints. The equations of a fifth-order polynomial and its two first derivatives are given in (1)-(3):

$$\mathcal{P}(t, \mathbf{c}) = [t^5 \; t^4 \; t^3 \; t^2 \; t \; 1] \; \mathbf{c} \tag{1}$$
$$\dot{\mathcal{P}}(t, \mathbf{c}) = [5t^4 \; 4t^3 \; 3t^2 \; 2t \; 1 \; 0] \; \mathbf{c} \tag{2}$$
$$\ddot{\mathcal{P}}(t, \mathbf{c}) = [20t^3 \; 12t^2 \; 6t \; 2 \; 0 \; 0] \; \mathbf{c}, \tag{3}$$

where $t \in \mathbb{R}$ and $\mathbf{c} = [a \; b \; c \; d \; e \; f]^\top \in \mathbb{R}^6$ are the vector of the polynomial coefficients. Then, we define the initial and final constraints as:

$$\mathcal{P}(0) = \mathcal{P}_i \qquad \mathcal{P}(T) = \mathcal{P}_f \tag{4}$$
$$\dot{\mathcal{P}}(0) = \dot{\mathcal{P}}_i \qquad \dot{\mathcal{P}}(T) = \dot{\mathcal{P}}_f \tag{5}$$
$$\ddot{\mathcal{P}}(0) = \ddot{\mathcal{P}}_i \qquad \ddot{\mathcal{P}}(T) = \ddot{\mathcal{P}}_f, \tag{6}$$

with $T \in \mathbb{R}_{>0}$. Using $t \in [0, T]$ instead of the more general form $t \in [T_1, T_2]$ allows simpler expressions and solutions as well as faster computations. To compute the polynomial coefficients, we put the problem in matrix form:

$$\begin{bmatrix} \mathcal{P}_i \\ \dot{\mathcal{P}}_i \\ \ddot{\mathcal{P}}_i \\ \mathcal{P}_f \\ \dot{\mathcal{P}}_f \\ \ddot{\mathcal{P}}_f \end{bmatrix} = \mathbf{Ac} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}, \tag{7}$$

that can then be solved with:

$$\mathbf{c}(T, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f) = \mathbf{A}^{-1} \begin{bmatrix} \mathcal{P}_i \\ \dot{\mathcal{P}}_i \\ \ddot{\mathcal{P}}_i \\ \mathcal{P}_f \\ \dot{\mathcal{P}}_f \\ \ddot{\mathcal{P}}_f \end{bmatrix}. \tag{8}$$

It can be found that the determinant of $\mathbf{A} = -4T^9$, leading to the matrix always being invertible since T is strictly positive. Once the coefficients have been computed, the evaluation of the polynomial and its first and second derivatives can be obtained using (1)-(3). When generating multiple polynomials at the same time, the output vectors can be obtained using:

$$\boldsymbol{\mathcal{P}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} t^5 \\ t^4 \\ \vdots \\ 1 \end{bmatrix} \quad (9) \qquad \dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} 5t^4 \\ 4t^3 \\ \vdots \\ 0 \end{bmatrix} \quad (10) \qquad \ddot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}_{traj}) = \begin{bmatrix} \mathbf{c}_0^\top \\ \mathbf{c}_1^\top \\ \vdots \\ \mathbf{c}_p^\top \end{bmatrix} \begin{bmatrix} 20t^3 \\ 12t^2 \\ \vdots \\ 0 \end{bmatrix}, \quad (11)$$

with $p \in \mathbb{N}$ the number of polynomials.

Trajectories should often be described using multiple waypoints. To deal with this, we can split the trajectory into segments, each represented by a polynomial. This can be translated to:

$$\mathcal{P}(t, \mathbf{c}_s) = [t^5\ t^4\ t^3\ t^2\ t\ 1] \begin{cases} \mathbf{c}^0 & \text{if } 0 \leq t \leq T_0 \\ \mathbf{c}^1 & \text{if } T_0 < t \leq T_1 \\ \vdots & \\ \mathbf{c}^q & \text{if } T_{q-1} < t \leq T_q \end{cases}, \tag{12}$$

with $q \in \mathbb{N}$ the number of segments. To avoid discontinuities in the trajectory, we impose the following condition:

$$\dot{\mathcal{P}}_i^k = \dot{\mathcal{P}}_f^{k-1} \tag{13}$$

$$\ddot{\mathcal{P}}_i^k = \ddot{\mathcal{P}}_f^{k-1}, \tag{14}$$

$\forall k \in [1, q]$.

1 Trajectory generation

We will detail three complementary methods used in this work to generate task or joint space trajectories.

The first method, presented in Sect. 1.1, allows to generate polynomial-based trajectories under velocity and acceleration constraints with arbitrary initial and final position, velocity and acceleration. Similar solutions, such as the Reflexxes Motion Library [1], are already available, with the main differences being bang-bang acceleration profiles, no notion of waypoints to build complex trajectories and that they are only usable with rotations described by Euler angles. Bang-bang acceleration profiles will produce the shortest trajectories between two points, but are very demanding on the robot actuators and can even cause them some damage due to their discontinuous nature. Also, smooth trajectories are preferable during human-robot interaction, since they provide a more natural motion. Using fifth-order polynomials results in smooth and acceleration-continuous trajectories, at the cost of a longer completion time.

The second method allows multiple trajectories, possibly each composed of several segments, to be synchronized. It will be detailed in Sect. 1.2.

In Sect. 1.3, a method to generate trajectories in task space using unit quaternions while maintaining translational and rotational velocities and accelerations under a given limit will be presented.

1.1 Constrained trajectory generation

If a trajectory segment has to be completed in a given time, one can just use (8) to compute the coefficients of the polynomial. Instead, if the time is not constrained, but velocity and acceleration limits are given, T is a parameter to be determined.

Let us first consider the specific case of null initial and final velocities and accelerations:

$$\mathbf{c}(T, \mathcal{P}_i, 0, 0, \mathcal{P}_f, 0, 0) = \begin{bmatrix} \frac{6\Delta\mathcal{P}}{T^5} & -\frac{15\Delta\mathcal{P}}{T^4} & \frac{10\Delta\mathcal{P}}{T^3} & 0 & 0 & 0 \end{bmatrix}^\top, \tag{15}$$

with $\Delta\mathcal{P} = \mathcal{P}_f - \mathcal{P}_i$. Solving $\ddot{\mathcal{P}}(t_{v_{max}}, \mathbf{c}) = 0$ gives us the time at which the velocity is maximal, which is $t_{v_{max}} = \frac{T}{2}$. The maximum velocity is then determined by:

$$\max[\dot{\mathcal{P}}(\mathbf{c})] = \frac{30\Delta\mathcal{P}}{16T}, \tag{16}$$

which leads to the minimum time required to satisfy the velocity limit V_{max}:

$$T_{min,v} = \frac{30\Delta\mathcal{P}}{V_{max}}, \tag{17}$$

with $V_{max} \in \mathbb{R}_{>0}$. The same reasoning can be applied to the acceleration limit A_{max} resulting in:

$$T_{min,a} = \sqrt{\frac{10\sqrt{3}\Delta\mathcal{P}}{3A_{max}}}, \tag{18}$$

with $A_{max} \in \mathbb{R}_{>0}$. The minimum duration required for a segment to satisfy both constraints is:

$$T = \max(T_{min,v}, T_{min,a}). \tag{19}$$

When the initial and final velocities and accelerations are non zero, no trivial solution can be found for T. To solve this problem we can use Alg.1 that, given an initial value for T (set to 1 here, but could be any $T \in \mathbb{R}_{>0}$), will converge to the minimum time necessary to comply with both constraints. The first part will solve T for the velocity limit V_{max} and the second one for the acceleration constraint A_{max}. At the end, (19) is used to obtain the segment duration. In this algorithm, the minimum durations updates $(*)$ and $(**)$ for $T_{min,v}$ and $T_{min,a}$ are exact in the specific case described above and are assumed to be a good approximation in the general case. The maximum velocity $\max[\dot{\mathcal{P}}(\mathbf{c})]$ and acceleration $\max[\ddot{\mathcal{P}}(\mathbf{c})]$ for a given polynomial can be found using a zero search algorithm, such as [2].

```
$T_{min,v} = T_{min,a} = 1$
**repeat**
    $v_{max} = \left|\max[\dot{\mathcal{P}}(\mathbf{c}(T_{min,v}, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f))]\right|$
    $\Delta_v = v_{max} - V_{max}$
    $T_{min,v} = T_{min,v}\ \frac{v_{max}}{V_{max}}$ $(*)$
**until** $|\Delta_v| < \varepsilon_v$;
**repeat**
    $a_{max} = \left|\max[\ddot{\mathcal{P}}(\mathbf{c}(T_{min,a}, \mathcal{P}_i, \dot{\mathcal{P}}_i, \ddot{\mathcal{P}}_i, \mathcal{P}_f, \dot{\mathcal{P}}_f, \ddot{\mathcal{P}}_f))]\right|$
    $\Delta_a = a_{max} - A_{max}$
    $T_{min,a} = T_{min,a}\ \sqrt{\frac{a_{max}}{A_{max}}}$ $(**)$
**until** $\Delta_a < \varepsilon_a$;
$T = \max(T_{min,v}, T_{min,a})$.
```

Algorithm 1: Segment minimum time computation.

1.2 Synchronization

When multiple trajectories must be generated simultaneously, some synchronization mechanisms should be used. This is illustrated in Fig. 1, where two trajectories $\mathcal{P}_0$ and $\mathcal{P}_1$, each composed of two segments, are represented using no synchronization (top), waypoint synchronization (middle) and trajectory synchronization (bottom). For the the waypoint synchronization, the duration of

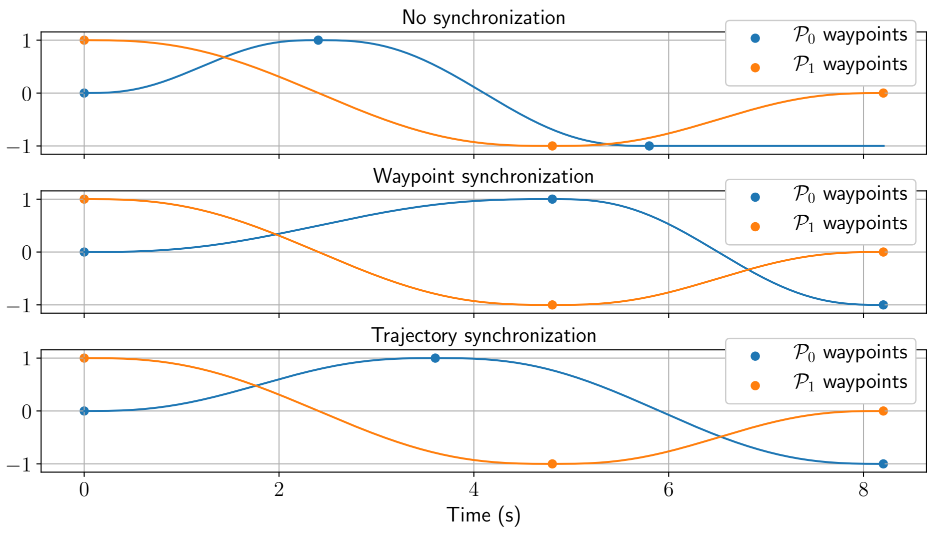

Figure 1: Comparison of the synchronization mechanisms.

the shortest segments (the ones of $\mathcal{P}_0$ in the example) is increased to match with the longest one ($\mathcal{P}_1$ segments here). For the trajectory synchronization, each segment duration of the shortest trajectories ($\mathcal{P}_0$) is increased so that their total duration matches the longest one ($\mathcal{P}_1$). This can be translated to:

$$T_j^k = \begin{cases} T_{min,j}^k & \text{if no synchronization} \\ \max(T^k) & \text{for waypoint synchronization} \\ T_{min,j}^k + \frac{\max(T_j) - T_{min,j}}{N} & \text{for trajectory synchronization,} \end{cases} \tag{20}$$

where T_j^k denotes the duration of the k-th segment of the j-th trajectory, $T_{min,j}^k$ is the minimum desired duration for the segment, $T_{min,j} = \sum_{k=1}^{N} T_{min,j}^k$ the minimum trajectory duration, $\max(T^k)$ the longest of the k-th segments, $\max(T_j)$ the longest of the trajectories and $N \in \mathbb{N}$ is the number of waypoints. When $T_{min,j}^k$ is computed to respect velocity and acceleration constraints (i.e., output by Alg.1), increasing it for synchronization purposes does not violate the initial limits since the segment maximum velocity and acceleration can only be lowered.

1.3 The case of orientations

Using unit quaternions to describe orientation in three dimensional space has several advantages over both Euler angles (simpler composition, no gimbal lock) and rotation matrices (more compact and numerically stable). Nevertheless, using unit quaternions over Euler angles has the disadvantage that their interpolation under velocity and acceleration (or higher derivative) constraints is not trivial. To overcome this, we propose a method to convert the quaternion interpolation problem in a form that allows the trajectory generator described above to be used. We define the orientation quaternion as:

$$\boldsymbol{q} = \mathcal{Q}\left(\mathbf{q}_v, q_w\right) \tag{21}$$

with $\mathbf{q}_v$ being the vector part and q_w the scalar part. The target orientation is denoted by $\boldsymbol{q}_r$. Then, we can compute an angular error vector $\Delta\boldsymbol{\theta}$ between $\boldsymbol{q}_r$ and $\boldsymbol{q}$ vector using:

$$\Delta\boldsymbol{q}(t) = \boldsymbol{q}_r(t)\overline{\boldsymbol{q}}(0) \tag{22}$$

$$\phi = \begin{cases} 2\cos^{-1}(\Delta q_w) & \text{if } \Delta q_w \geq 0 \\ 2\cos^{-1}(\Delta q_w) - 2\pi & \text{otherwise} \end{cases} \tag{23}$$

$$\Delta\boldsymbol{\theta}(t) = \phi \frac{\Delta\boldsymbol{q}_v(t)}{\|\Delta\boldsymbol{q}_v(t)\|}. \tag{24}$$

Using (23) gives $\phi \in \left]-\pi, \pi\right]$, allowing the rotation to be kept at its minimum (e.g., a rotation of $-\pi$ instead of $\frac{3\pi}{4}$). The trajectory generator described earlier can then be configured to output a trajectory going from $\mathbf{0}^3 = [0\ 0\ 0]^\top$ to $\Delta\boldsymbol{\theta}(t)$ under the desired velocity and acceleration constraints $\boldsymbol{\omega}_{max}$ and $\dot{\boldsymbol{\omega}}_{max}$:

$$\Delta\boldsymbol{\theta}^*(t) = \boldsymbol{\mathcal{P}}(t, \mathbf{c}') \tag{25}$$

$$\boldsymbol{\omega}^*(t) = \dot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}') \tag{26}$$

$$\dot{\boldsymbol{\omega}}^*(t) = \ddot{\boldsymbol{\mathcal{P}}}(t, \mathbf{c}'). \tag{27}$$

For pose tracking, the orientation quaternion $\boldsymbol{q}^*$ to be tracked can be computed after each interpolation as follow:

$$\boldsymbol{q}_\Delta(t) = \mathcal{Q}\left(\frac{\Delta\boldsymbol{\theta}^*(t)}{2}, 0\right) \tag{28}$$

$$\boldsymbol{q}^*(t) = e^{\boldsymbol{q}_\Delta(t)}\boldsymbol{q}(0) \tag{29}$$

References

[1] Torsten Kroger. Opening the door to new sensor-based robot applications - the reflexxes motion libraries. In *IEEE International Conference on Robotics and Automation*. IEEE, may 2011.

[2] M. A. Jenkins and J. F. Traub. A three-stage algorithm for real polynomials using quadratic iteration. *SIAM Journal on Numerical Analysis*, 7(4):545–566, 1970.