ROBOTICS VISION AND CONTROL

Assignments

Francesco Dal Santo
Universitá degli studi di verona
2022-2023

Contents

1 Polynomials

1.1 Cubic polynomials given ti and tf

In this case we use the equation when t $\in [t_i,t_f]$:

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \tag{1}$$

Given the 4 unknowns a_3,a_2,a_1 and a_0, we need 4 constraints:

$$\begin{aligned}
q_i(t) &= a_3t_i^3 + a_2t_i^2 + a_1t_i + a_0 \\
q_f(t) &= a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 \\
\dot{q}_i(t) &= 3a_3t_i^2 + 2a_2t_i + a_1 \\
\dot{q}_f(t) &= 3a_3t_f^2 + 2a_2t_f + a_1
\end{aligned} \tag{2}$$

1.2 Cubic polynomials given ΔT

In this case we use the equation when t $\in [0,\Delta T]$:

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

Given the 4 unknowns a_3,a_2,a_1 and a_0, we need 4 constraints:

$$\begin{aligned}
q_i &= a_0 \\
\dot{q}_i &= a_1 \\
q_f &= a_0 + a_1\Delta T + a_2\Delta T^2 + a_3\Delta T^3 \\
\dot{q}_f &= a_1 + 2a_2\Delta T + 3a_3\Delta T^2
\end{aligned} \tag{4}$$

1.3 5^{th} order polynomials given ti and tf

In this case we use the equation when t $\in [t_i,t_f]$:

$$q(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{5}$$

Given the 6 unknowns a_5,a_4,a_3,a_2,a_1 and a_0, we need 6 constraints:

$$\begin{aligned}
q_i(t) &= a_5t_i^5 + a_4t_i^4 + a_3t_i^3 + a_2t_i^2 + a_1t_i + a_0 \\
\dot{q}_i(t) &= 5a_5t_i^4 + 4a_4t_i^3 + 3a_3t_i^2 + 2a_2t_i \\
\ddot{q}_i(t) &= 20a_5t_i^3 + 12a_4t_i^2 + 6a_3t_i + 2a_2 \\
q_f(t) &= a_5t_f^5 + a_4t_f^4 + a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 \\
\dot{q}_f(t) &= 5a_5t_f^4 + 4a_4t_f^3 + 3a_3t_f^2 + 2a_2t_f \\
\ddot{q}_f(t) &= 20a_5t_f^3 + 12a_4t_f^2 + 6a_3t_f + 2a_2
\end{aligned} \tag{6}$$

1.4 5^{th} order polynomials given ΔT

In this case we use the equation when t ∈ [0,ΔT]:

$$q(t) = a_5(t-ti)^5 + a_4(t-ti)^4 + a_3(t-ti)^3 + a_2(t-ti)^2 + a_1(t-ti) + a_0 \tag{7}$$

Given the 6 unknowns a_5,a_4,a_3,a_2,a_1 and a_0, we need 6 constraints:

$$\begin{aligned}
q_i &= a_0 \\
\dot{q}_i &= a_1 \\
\ddot{q}_i &= 2a_2 \\
q_f &= a_0 + a_1\Delta T + a_2\Delta T^2 + a_3\Delta T^3 + a_4\Delta T^4 + a_5\Delta T^5 \\
\dot{q}_f &= a_1 + 2a_2\Delta T + 3a_3\Delta T^2 + 4a_4\Delta T^3 + 5a_5\Delta T^4 \\
\ddot{q}_f &= 2a_2 + 6a_3\Delta T + 12a_4\Delta T^2 + 20a_5\Delta T^3
\end{aligned} \tag{8}$$

1.5 7^{th} order polynomials given ti and tf

In this case we use the equation when t ∈ [t_i,t_f]:

$$q(t) = a_7t^7 + a_6t^6 + a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{9}$$

Given the 8 unknowns a_7,a_6,a_5,a_4,a_3,a_2,a_1 and a_0, we need 8 constraints:

$$\begin{aligned}
q_i(t) &= a_7t_i^7 + a_6t_i^6 + +a_5t_i^5 + a_4t_i^4 + a_3t_i^3 + a_2t_i^2 + a_1t_i + a_0 \\
\dot{q}_i(t) &= 7a_7t_i^6 + 6a_6t_i^5 + 5a_5t_i^4 + 4a_4t_i^3 + 3a_3t_i^2 + 2a_2t_i + a_1 \\
\ddot{q}_i(t) &= 42a_5t_i^5 + 30a_6t_i^4 + 20a_5t_i^3 + 12a_4t_i^2 + 6a_3t_i + 2a_2 \\
\dddot{q}_i(t) &= 210a_5t_i^4 + 120a_6t_i^3 + 60a_5t_i^2 + 24a_4t_i + 6a_3 \\
q_f(t) &= a_7t_f^7 + a_6t_f^6 + a_5t_f^5 + a_4t_f^4 + a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 \\
\dot{q}_f(t) &= 7a_7t_f^6 + 6a_6t_f^5 + 5a_5t_f^4 + 4a_4t_f^3 + 3a_3t_f^2 + 2a_2t_f + a_1 \\
\ddot{q}_f(t) &= 42t_f^5 + 30a_6t_f^4 + 20a_5t_f^3 + 12a_4t_f^2 + 6a_3t_f + 2a_2 \\
\dddot{q}_f(t) &= 210a_5t_f^4 + 120a_6t_f^3 + 60a_5t_f^2 + 24a_4t_f + 6a_3
\end{aligned} \tag{10}$$

1.6 7^{th} order polynomials given ΔT

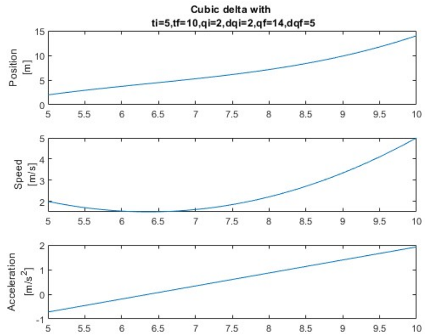
In this case we use the equation when t ∈ [0,ΔT]:

$$q(t) = a_7(t-ti)^7 + a_6(t-ti)^6 + a_5(t-ti)^5 + a_4(t-ti)^4 + a_3(t-ti)^3 + a_2(t-ti)^2 + a_1(t-ti) + a_0 \tag{11}$$

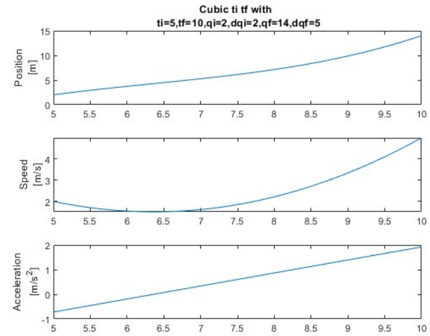
Given the 8 unknowns a_7,a_6,a_5,a_4,a_3,a_2,a_1 and a_0, we need 8 constraints:

$$\begin{aligned}
q_i &= a_0 \\
\dot{q}_i &= a_1 \\
\ddot{q}_i &= 2a_2 \\
\dddot{q}_i &= 3a_3 \\
q_f &= a_0 + a_1\Delta T + a_2\Delta T^2 + a_3\Delta T^3 + a_4\Delta T^4 + a_5\Delta T^5 + a_6\Delta T^6 + a_7\Delta T^7 \\
\dot{q}_f &= a_1 + 2a_2\Delta T + 3a_3\Delta T^2 + 4a_4\Delta T^3 + 5a_5\Delta T^4 + 6a_6\Delta T^5 + 7a_7\Delta T^6 \\
\ddot{q}_f &= 2a_2 + 6a_3\Delta T + 12a_4\Delta T^2 + 20a_5\Delta T^3 + 30a_6\Delta T^4 + 42a_7\Delta T^5 \\
\dddot{q}_f &= 6a_3 + 24a_4\Delta T + 60a_5\Delta T^2 + 120a_6\Delta T^3 + 210\Delta T^4
\end{aligned} \tag{12}$$

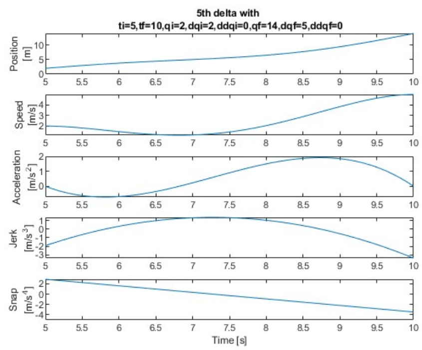
1.7 Results



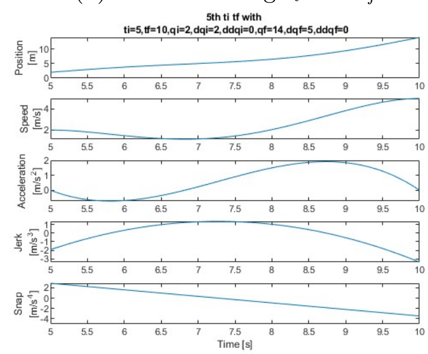
(a) 3^{th} order using ΔT



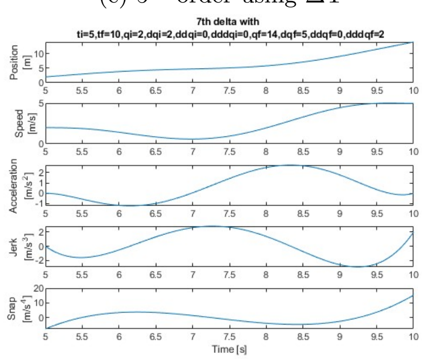
(b) 3^{th} order using t_i and t_f



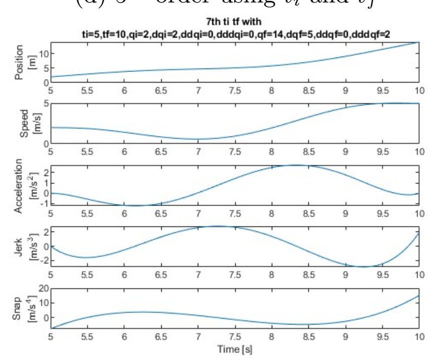
(c) 5^{th} order using ΔT



(d) 5^{th} order using t_i and t_f



(e) 7^{th} order using ΔT



(f) 7^{th} order using t_i and t_f

Figure 1: Results for polynomials trajectories

2 Trapeziodal

In this case we impose the following constraints:

- t_i,t_f
- q_i,q_f

using the following assumptions:

- $t_i = 0$
- $\dot{q_i} = 0$,$\dot{q_f} = 0$
- t_c = acceleration time = deceleration time

We have a first constant acceleration in the start phase (t $\in [0,t_C]$)

$$\begin{cases} q(t) = a_0 + a_1 t + a_2 t^2 \\ \dot{q(t)} = a_1 + 2a_2 t \\ \ddot{q(t)} = 2a_2 \end{cases} \tag{13}$$

Then a constant cruise velocity (t $\in [t_c,t_f - t_c]$)

$$\begin{cases} q(t) = b_0 + b_1 t \\ \dot{q(t)} = b_1 \\ \ddot{q(t)} = 0 \end{cases} \tag{14}$$

Finally a constant deceleration (t $\in [t_f - t_c,t_f]$)

$$\begin{cases} q(t) = c_0 + c_1 t + c_2 t^2 \\ \dot{q(t)} = c_1 + 2c_2 t \\ \ddot{q(t)} = 2c_2 \end{cases} \tag{15}$$

Using the following constraints:

- $q_m = \frac{q_i + q_f}{2}$
- $t_m = \frac{t_f - t_i}{2}$
- $q_c = q(t_i + t_c)$
- $\ddot{q}_c t + c = \frac{q_m - q_c}{t_m - t_c}$
- $q_c = q_i + \frac{1}{2}\ddot{q}_c t_c^2$

We end up with the following equation:

$$\ddot{q}_c t_c^2 - \ddot{q}_c (t_f - t_i) t_c + (q_f - q_i) = 0 \tag{16}$$

$$q(t) = \begin{cases} q_i + \frac{\dot{q}_c}{2t_C}t^2 & 0 \le t \le t_c \\ q_i + \dot{q}_c(t - \frac{t_c}{2}) & t_c < t \le t_f - t_c \\ q_f - \frac{\dot{q}_c}{2t_c}(t_f - t)^2 & t_f - t_c < t \le t_f \end{cases}$$

2.1 Trapezoidal given t_c with $t_i = 0$ assumption

If we know t_c we can obtain $\ddot{q}_c$ and $\dot{q}_c$ with:

$$\begin{aligned} \ddot{q}_c &= \frac{q_f - q_i}{t_c t_f - t_c^2} \\ \dot{q}_c &= \ddot{q}_c t_c \end{aligned} \tag{17}$$

If $\alpha = \frac{t_c}{t_f}$ with $0 < \alpha \le \frac{1}{2}$:

$$\begin{aligned} \ddot{q}_c &= \frac{q_f - q_i}{\alpha(1-\alpha)t_f^2} \\ \dot{q}_c &= \frac{q_f - q_i}{(1-\alpha)t_f} \end{aligned} \tag{18}$$

2.2 Trapezoidal given $\ddot{q}_c$ with ti = 0 assumption

If we know $\ddot{q}_c$ and $t_i = 0$ we can obtain t_c as:

$$t_c = \frac{tf}{2} - \frac{1}{2}\sqrt{\frac{t_f^2\ddot{q}_c - 4(q_f - q_i)}{\ddot{q}_c}} \tag{19}$$

2.3 Trapezoidal given $\dot{q}_c$ with ti = 0 assumption

If we know $\dot{q}_c$ and $t_i = 0$ we can obtain t_c and $\ddot{q}_c$ as:

$$\begin{aligned} t_c &= \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c} \\ \ddot{q}_c &= \frac{\dot{q}_c{}^2}{q_i - q_f + \dot{q}_c t_f} \end{aligned} \tag{20}$$

2.4 Trapezoidal with preassigned $\ddot{q}_c$ and $\dot{q}_c$

Given the condition $t_i = 0$

$$\begin{aligned} t_c &= \frac{\dot{q}_c}{\ddot{q}_c} \\ t_f &= \frac{\dot{q}_c{}^2 + \ddot{q}_c(q_f - q_i)}{\dot{q}_c\ddot{q}_c} \end{aligned} \tag{21}$$

We can derive the coefficients of the polynomials as a function of (t_c and $\ddot{q}_c$)

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

If $q_f - q_i \geq \frac{\dot{q}_c^{\,2}}{\ddot{q}_c}$, we have a linear segment; otherwise the velocity profile has a triangular shape.

2.5 Trapezoidal with preassigned duration ΔT = t_f - t_i and maximum acceleration $\ddot{q}_c^{\,\mathbf{max}}$

First we should check the feasibility, i.e $q\ddot{(t)} \leq \ddot{q}_c^{\,\max}$

$$\ddot{q}_c^{\,\max}\Delta q > \frac{\left|\dot{q}_i^{\,2} - \dot{q}_f^{\,2}\right|}{2} \tag{22}$$

After the check we can compute the constant velocity:

$$\dot{q}_c = \frac{1}{2}(\dot{q}_i + \dot{q}_f + \ddot{q}_c^{\,\max}\Delta T - \sqrt{(\ddot{q}_c^{\,\max})^2\Delta T^2 - 4\ddot{q}_c^{\,\max}\Delta q + 2\ddot{q}_c^{\,\max}(\dot{q}_i + \dot{q}_f)\Delta T - (\dot{q}_i - \dot{q}_f)^2}) \tag{23}$$

where $\ddot{q}_c^{\,\max}$ must satisfy

$$(\ddot{q}_c^{\,\max})^2\Delta T^2 - 4\ddot{q}_c^{\,\max}\Delta q + 2\ddot{q}_c^{\,\max}(\dot{q}_i + \dot{q}_f)\Delta T - (\dot{q}_i - \dot{q}_f)^2 > 0 \tag{24}$$

The max acceleration must be larger than a limit value:

$$\ddot{q}_c^{\,\max} \geq \ddot{q}_c^{\,\lim} \quad = \frac{2\Delta q - (\dot{q}_i + \dot{q}_f)\Delta T + \sqrt{4\Delta q^2 - 4\Delta q(\dot{q}_i^{\,2} + \dot{q}_f^{\,2})\Delta T^2}}{\Delta T^2} \tag{25}$$

We can compute the acceleration and deceleration periods:

$$\begin{aligned} t_a &= \frac{\dot{q}_c - \dot{q}_i}{\ddot{q}_c^{\,\max}} \\ t_d &= \frac{\dot{q}_c - \dot{q}_f}{\ddot{q}_c^{\,\max}} \end{aligned} \tag{26}$$

2.6 Trapezoidal with preassigned maximum acceleration $\ddot{q}_c^{\mathbf{max}}$ and maximum velocity $\dot{q}_c^{\mathbf{max}}$

First we check for feasibility:

$$\ddot{q}_c^{\max}\Delta q > \frac{|\dot{q}_i^2 - \dot{q}_f^2|}{2} \tag{27}$$

If the trajectory exists,two cases are possible according to the fact tha the maximum velocity is reached (>) or not (<):

$$\ddot{q}_c^{\max}\Delta q \underset{<}{\geq} (\dot{q}_c^{\max})^2 - \frac{\dot{q}_i^2 + \dot{q}_f^2}{2} \tag{28}$$

If $\dot{q}_c^{\max}$ is reached:

$$\begin{aligned}
\dot{q}_c &= \dot{q}_c^{\max} \\
t_a &= \frac{\dot{q}_c^{\max} - \dot{q}_i}{\ddot{q}_c^{\max}} \\
t_d &= \frac{\dot{q}_c^{\max} - \dot{q}_f}{\ddot{q}_c^{\max}} \\
\Delta T &= \frac{\Delta q}{\dot{q}_c} + \frac{\dot{q}_c^{\max}}{2\ddot{q}_c^{\max}}(1 - \frac{\dot{q}_i}{\dot{q}_c^{\max}})^2 + \frac{\dot{q}_c^{\max}}{2\ddot{q}_c^{\max}}(1 - \frac{\dot{q}_f}{\dot{q}_c^{\max}})^2
\end{aligned} \tag{29}$$

If $\dot{q}_c^{\max}$ is not reached:

$$\begin{aligned}
\dot{q}_c = \dot{q}_c^{\lim} &= \sqrt{\ddot{q}_c^{\max}\Delta q + \frac{\dot{q}_i^2 + \dot{q}_f^2}{2}} < \dot{q}_c^{\max} \\
t_a &= \frac{\dot{q}_c^{\max} - \dot{q}_i}{\ddot{q}_c^{\max}} \\
t_d &= \frac{\dot{q}_c^{\max} - \dot{q}_f}{\ddot{q}_c^{\max}} \\
\Delta T &= t_a + t_d
\end{aligned} \tag{30}$$

2.7 Trajectory with $\dot{q}_i \neq 0$,$\dot{q}_f \neq 0$,$\ddot{q}_i = 0$,$\ddot{q}_f = 0$

Acceleration phase:

$$\begin{aligned} q(t) &= q_i + \dot{q}_i(t - t_i) + \frac{\dot{q}_c - \dot{q}_i}{2t_a}(t - t_i)^2 \\ \dot{q(t)} &= \dot{q}_i + \frac{\dot{q}_c - \dot{q}_i}{t_a}(t - t_i) \\ \ddot{q(t)} &= \frac{\dot{q}_c - \dot{q}_i}{t_a} =: \ddot{q}_c \end{aligned} \tag{31}$$

Constant velocity phase:

$$\begin{aligned} q(t) &= q_i + \dot{q}_i\frac{t_a}{2} + \dot{q}_c(t - t_i - \frac{t_a}{2}) \\ \dot{q(t)} &= \dot{q}_c \\ \ddot{q(t)} &= 0 \end{aligned} \tag{32}$$

Deceleration phase:

$$\begin{aligned} q(t) &= q_f - \dot{q}_f(t_f - t) - \frac{\dot{q}_c - \dot{q}_f}{2t_d}(t_f - t)^2 \\ \dot{q(t)} &= \dot{q}_f + \frac{\dot{q}_c - \dot{q}_f}{t_d}(t_f - t) \\ \ddot{q(t)} &= -\frac{\dot{q}_c - \dot{q}_f}{t_d} =: -\ddot{q}_c \end{aligned} \tag{33}$$

2.8 Multipoints with $q_c^{\dot{max}}$ and $q_c^{\ddot{max}}$

Here the goal is to design a trajectory through a sequence of points, the velocity at each point can be defined as:

$$\begin{aligned} \dot{q}(t_i) &= \dot{q}_i \\ \dot{q}(t_f) &= \dot{q}_f \end{aligned} \tag{34}$$

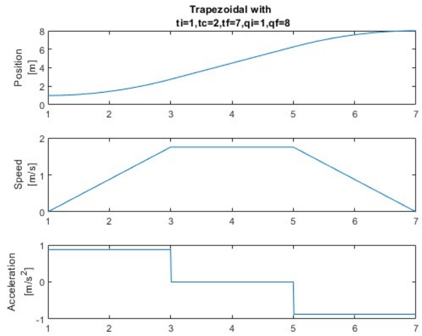
$$\dot{q}(t_k) = \begin{cases} 0 & sign(\Delta Q_k) \neq sign(\Delta Q_{\text{k+1}}) \\ sign(\Delta Q_k)\dot{q}^{\max} & sign(\Delta Q_k) = sign(\Delta Q_{\text{k+1}}) \end{cases}$$

and we use the **2.6** formulation to get the trajectory, however we pass through the points at time points different from the chosen ones.

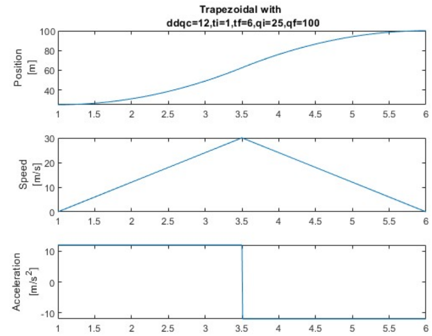
2.9 Multipoints with respected t_k

We calculate the velocity at t_k as before and then apply the **2.5**, however in this case we don't respect the $q_c^{\dot{max}}$ condition but we pass through the points at the correct time points.

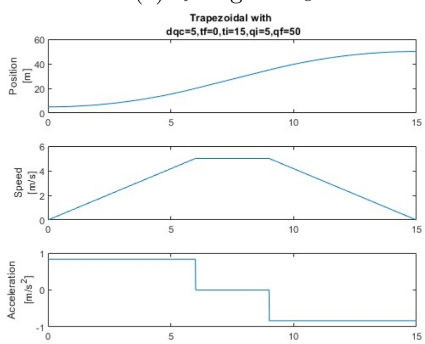
2.10 Results



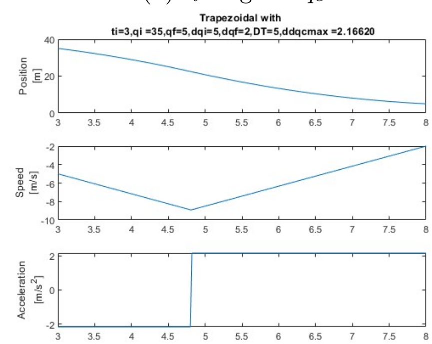
(a) t_i=0 given t_c



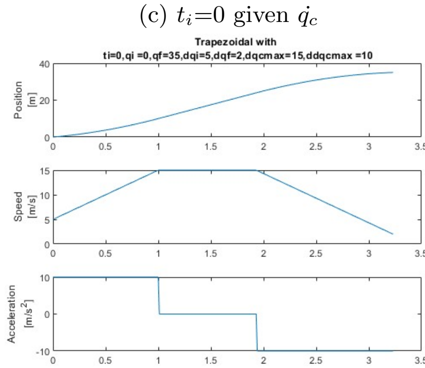
(b) t_i=0 given $\ddot{q_c}$



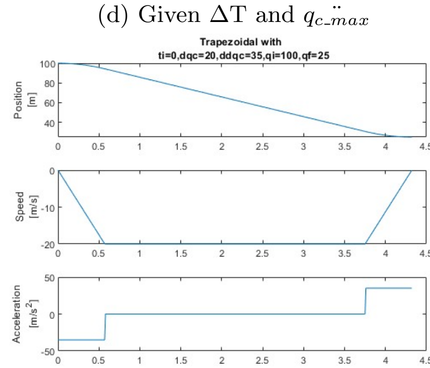
(c) t_i=0 given $\dot{q_c}$



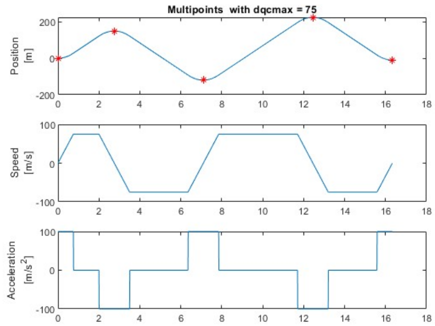
(d) Given ΔT and $q_{c\_\ddot{m}ax}$



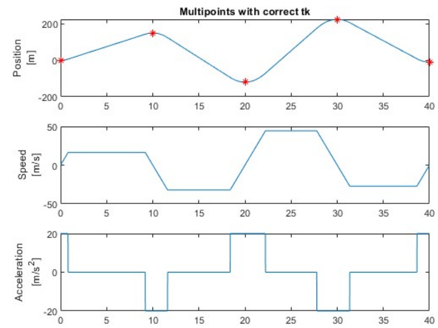
(e) Given $q_{c\_\ddot{m}ax}$ and $q_{c\_\dot{m}ax}$



(f) t_i=0 given $\ddot{q_c}$ and $\dot{q_c}$



(a) Multipoints with $q_c^{max} = 75$



(b) Multipoint with correct tk (0,10,20,30,40)

3 Joint Space Trajectories - Sequence of Points

Given n+1 (q_k, t_k), we want to design a trajectory such that the end-effector passes by each point q_k at a specific instant of time t_k.

3.1 Interpolating polynomials with continuous velocity

For k = 1,...,n-1 we compute:

$$vk := \frac{q_k - q_{k-1}}{t_k - t_{k-1}} \tag{35}$$

The velocity in each path point is computed as:

$$\dot{q_k} = \begin{cases} 0 & if sgn(v_k) \neq sgn(v_{k+1}) \\ \frac{v_k + v_{k+1}}{2} & if sgn(v_k) = sgn(v_{k+1}) \end{cases}$$

We can the use the solution of the system given the conditions on positions and velocity:

$$\begin{aligned} a_0^k &= q_k \\ a_1^k &= \dot{q_k} \\ a_2^k &= \frac{1}{T_k}[\frac{3(q_{k+1} - q_k)}{T_k} - 2\dot{q_k} - \dot{q_{k+1}}] \\ a_3^k &= \frac{1}{T_k^2}[\frac{2(q_k - q_{k+1})}{T_k} + \dot{q_k} + \dot{q_{k+1}}] \end{aligned} \tag{36}$$

Finally we can compute the trajectories:

$$\begin{aligned} q_{k+1} &= a_3^k T_k^3 + a_2^k T_k^2 + a_1^k T_k + a_0^k \\ \dot{q_{k+1}} &= 3a_3^k T_k^2 + 2a_2^k T_k + a_1^k \\ \ddot{q_{k+1}} &= 6a_3^k T_k + 2a_2^k \end{aligned} \tag{37}$$

3.2 Interpolating polynomials with continuous acceleration

We start by initializing the matrix c:

$$c(k) = 3\frac{T_{k+1}}{T_k}(q_{k+1} - q_k) + 3\frac{T_k}{T_{k+1}}(q_{k+2} - q_{k+1}) \tag{38}$$

We then set the matrix A:

$$\begin{aligned}
A(0,0) &= 2(T_0 + T_1) \\
A(0,1) &= T(0) \\
A(k,k-1) &= T_{k+1} \\
A(k,k) &= 2(T_k + T_{k+1}) \\
A(k,k+1) &= T_k \\
A(end,end) &= 2(T_{n-2} + T_{n-1}) \\
A(end,end-1) &= T_{n-1}
\end{aligned} \tag{39}$$

We apply the Forward elimination part of the Thomas algorithm:
For every path point:

$$\begin{aligned}
m &= \frac{A(k,k-1)}{A(k-1,k-1)} \\
A(k,k) &= A(k,k) - m * A(k-1,k) \\
c(k) &= c(k) - m * c(k-1)
\end{aligned} \tag{40}$$

then we know $dqk(end) = \frac{c(end)}{A(end,end)}$ whilst for the other we apply the backward substitution:
Going back from last element:

$$dqk(k) = \frac{c(k) - A(k,k+1) * \dot{q_k}(k+1)}{A(k,k)} \tag{41}$$

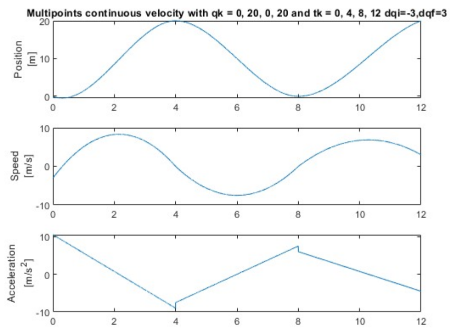
Finally we can the use the solution of the system given the conditions on positions and velocity:

$$\begin{aligned}
a_0^k &= q_k \\
a_1^k &= \dot{q_k} \\
a_2^k &= \frac{1}{T_k}[\frac{3(q_{k+1} - q_k)}{T_k} - 2\dot{q_k} - \dot{q_{k+1}}] \\
a_3^k &= \frac{1}{T_k^2}[\frac{2(q_k - q_{k+1})}{T_k} + \dot{q_k} + \dot{q_{k+1}}]
\end{aligned} \tag{42}$$

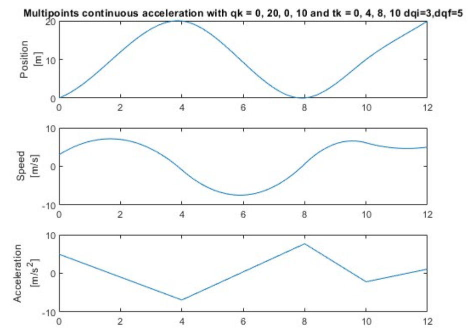
and calculate the trajectories:

$$\begin{aligned}
q_{k+1} &= a_3^k T_k^3 + a_2^k T_k^2 + a_1^k T_k + a_0^k \\
\dot{q_{k+1}} &= 3a_3^k T_k^2 + 2a_2^k T_k + a_1^k \\
\ddot{q_{k+1}} &= 6a_3^k T_k + 2a_2^k
\end{aligned} \tag{43}$$

3.3 Results



(a) Multipoints with continuous velocity



(b) Multipoints with continuous acceleration

4 Cubical Splines

4.1 Cubic splines based on the accelerations with assigned initial and final velocities

We start by defining the initial and final conditions of the matrices c and A:

$$\begin{aligned}
c(0) &= 6(\frac{q_1 - q_0}{T_0} - \dot{q_0}) \\
c(end) &= 6(\dot{q_f} - \frac{q_n - q_{n-1}}{T_{n-1}}) \\
A(0,0) &= 2T_0 \\
A(0,1) &= T_0 \\
A(end, end-1) &= T_{end-1} \\
A(end, end) &= 2T_{end-1}
\end{aligned} \tag{44}$$

and the other points:

$$\begin{aligned}
A(k, k-1) &= T_{k-1} \\
A(k, k) &= 2T_{k-1} + 2T_i \\
A(k, k+1) &= T_k \\
c(n) &= 6(\frac{q_n - q_{n-1}}{T_{n-1}} - \frac{q_{n-1} - q_{n-2}}{T_{n-2}})
\end{aligned} \tag{45}$$

We then apply the Thomas algorithm to get $\ddot{q_k}$ and we use that in the solved system:

$$\begin{aligned}
a_0^k &= q_k \\
a_1^k &= \frac{q_{k+1} - q_k}{T_k} - \frac{\ddot{q_{k+1}} + 2\ddot{q_k}}{6} T_k \\
a_2^k &= \frac{\ddot{q_k}}{2} \\
a_3^2 &= \frac{\ddot{q_{k+1}} - \ddot{q_k}}{6T_k}
\end{aligned} \tag{46}$$

Finally we can solve for the trajectories:

$$\begin{aligned}
q_k &= a_0^k \\
\ddot{q_k} &= 2a_2^k \\
q_{k+1} &= a_3^k T_k^3 + a_2^k T_k^2 + a_1^k T_k + a_0^k \\
\dot{q_{k+1}} &= 3a_3^k T_k^2 + 2a_2^k T_k + a_1^k \\
\ddot{q_{k+1}} &= 6a_3^k T_k + 2a_2^k
\end{aligned} \tag{47}$$

4.2 Smoothing cubic splines

Smoothing cubic splines are designed to approximate and not interpolate a set of given data points, we set:

- w_k: parameters which can be arbitrarily chosen in order to modify the weight of the k-th quadratic error on the global optimization problem
- μ: weights the trade-off between the two conflicting goals (fitting the points and smoothing the trajectory)
- $\lambda = \frac{1-\mu}{6\mu}$

We set the matrices A and c:

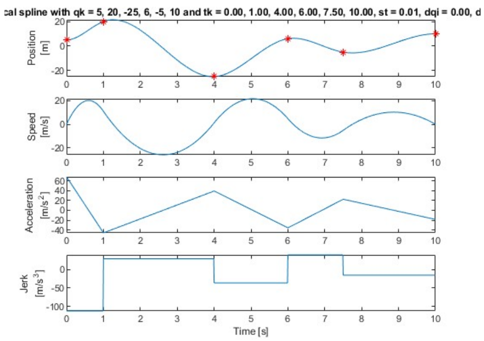
$$
\begin{aligned}
A(0,0) &= 2T_0 \\
A(0,1) &= T_0 \\
C(0,0) &= -\frac{6}{T_0} \\
C(0,1) &= \frac{6}{T_0} \\
A(k,k-1) &= T_{k-1} \\
A(k,k) &= 2T_{k-1} + 2T_k \\
A(k,k+1) &= T_k \\
C(k,k-1) &= \frac{6}{T_k - 1} \\
C(k,k) &= -(\frac{6}{T_{k-1}} + \frac{6}{T_k}) \\
C(k,k+1) &= \frac{6}{T_k} \\
A(end,end-1) &= T_{end-1} \\
A(end,end) &= 2T_{end-1} \\
C(end,end-1) &= -\frac{6}{T_{k-1}}
\end{aligned}
\tag{48}
$$

We calculate:

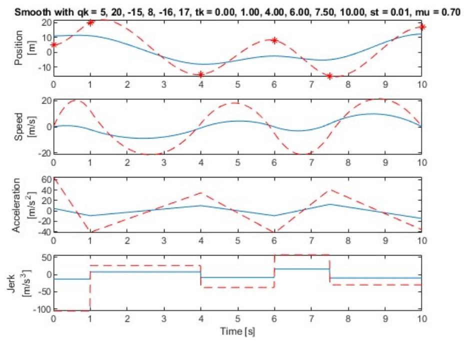
$$
\begin{aligned}
\ddot{s} &= Cq * (A + \lambda C W^{-1} C^T)^{-1} \\
s &= q - \lambda W^{-1} C^T \ddot{s}
\end{aligned}
\tag{49}
$$

Then we compute the trajectories as before using s as the new path points.

4.3 Results



(a) Cubical spline



(b) Smoothed version

5 3D Trajectory using linear and circular motion primitives

The trajectory composed by a set of linear segments is continuous but it is characterized by discontinuous derivatives at the intermediate points.

For the rectilinear path we define the arc length and the velocity as:

$$\begin{aligned} p(s) &= p_i + s\frac{p_f - p_i}{||p_f - p_i||} \\ \frac{dp}{ds} &= \frac{p_f - p_i}{||p_f - p_i||} \end{aligned} \tag{50}$$

For the circular path we first get the z axis of the circumference cut by the point1-point2 plane by:

$$\begin{aligned} \mathbf{z}' &= \mathbf{p1\_c} \times \mathbf{p2\_c} \\ angle &= arccos(\mathbf{v}_1 \cdot \mathbf{v}_2) \end{aligned} \tag{51}$$

where v_1 and v_2 are the versors of pi_c and p2_c; if the result is equal to 0, we force the direction.

We now get the unit vectors:

$$\begin{aligned} \mathbf{x}' &= \frac{\textbf{p1-centre}}{||\textbf{p1-centre}||} \\ \mathbf{y}' &= \mathbf{x}' \times \mathbf{z}' \\ \mathbf{z}' &= \frac{\mathbf{z}'}{||\mathbf{z}'||} \end{aligned} \tag{52}$$

To move along a circular arc we can relay on the parametric representation:

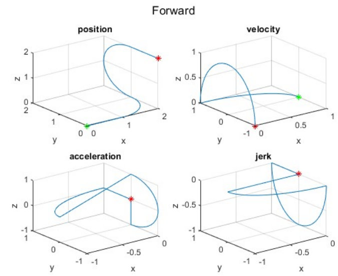
$$p'(u) = \begin{pmatrix} \rho \cos(u) \\ \rho \sin(u) \\ 0 \end{pmatrix} \tag{53}$$

and bring it in Σ:

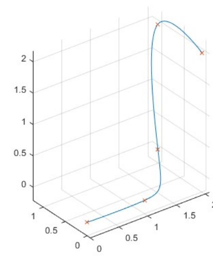
$$\begin{aligned} p(u) &= \mathbf{c} + R\mathbf{p}'(u) \\ R &= [\mathbf{x'}\ \mathbf{y'}\ \mathbf{z'}] \end{aligned} \tag{54}$$

For the comparison with the spline method we apply the algorithm on x,y and z coordinates separately and then put them together to get the 3d path.

5.1 Results

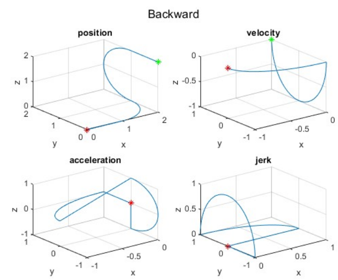


(a) Linear and circular primitives

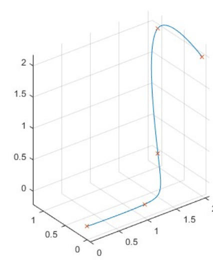


(b) Using splines

Figure 6: Forward trajectory



(a) Linear and circular primitives



(b) Using splines

Figure 7: Backward trajectory

6 Trajectory perpendicular to a sphere

The code here is equal to the previous assignment, except for the addition of the 3 Frenet frames T,N,B:

$$\begin{aligned}
T &= \frac{d\mathbf{p}}{ds} = R \begin{pmatrix} -\rho\, \sin(\mathrm{s}) \\ \rho\, \cos(\mathrm{s}) \\ 0 \end{pmatrix} \\
N &= \frac{d^2\mathbf{p}}{d^2 s} = R \begin{pmatrix} -\rho\, \cos(\mathrm{s}) \\ -\rho\, \sin(\mathrm{s}) \\ 0 \end{pmatrix} \\
B &= \mathbf{T} \times \mathbf{N}
\end{aligned} \tag{55}$$

where ρ is the radius, R is the matrix described in the previous assignment and **T,N,B** are then reduced to unit vectors.

6.1 Results

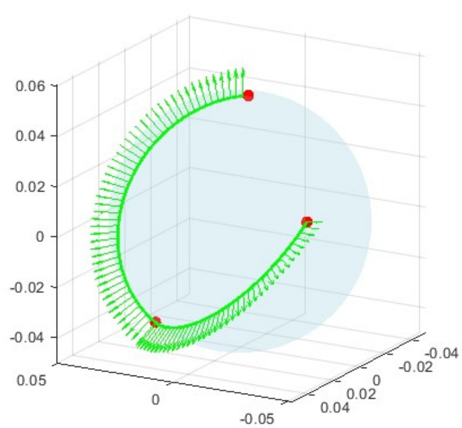


Figure 8: Trajectory perpendicular to a sphere

7 Pick and Place in Unity environment

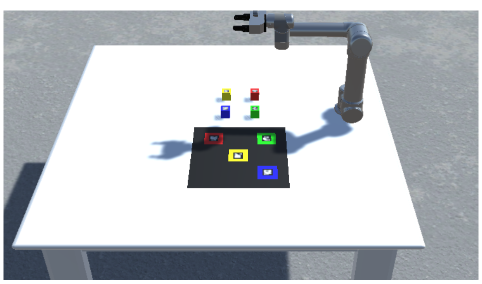
In this assignment we are requested to perform a pick and place task.
Every trajectory in this assignment was calculated with the linear primitive for simplicity and to avoid collisions with the environment.

To complete this task we used a finite state machine approach, where the first phase involves getting the poses with the camera attached to the ur5 robot.
To do this the robot does a rectangular-like trajectory to scan the environment, saving the initial and final poses for each cube.

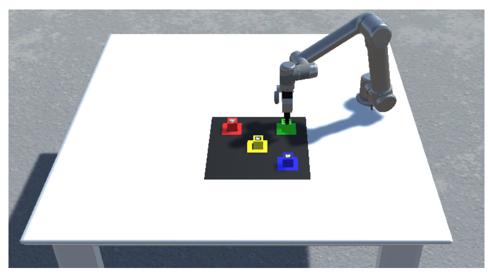
Then we follow a predecided order assigning a trajectory that goes from the starting pose to the final pose. To avoid collisions we added a first way point which brings the cube up from its starting pose, then goes horizontally over the destination and then down again; same for the trajectory that is calculated after placing a cube to reach the new cube to pick.

A video of the final result can be found in the GitHub folder.

7.1 Results



(a) Start of simulation



(b) End of simulation