Master's degree in Computer Engineering for Robotics and Smart Industry

Robotics, Vision and Control

Report on the assignments given during the 2021/2022 a.y.

Author: Lorenzo Busellato, VR472249
email: lorenzo.busellato_02@studenti.univr.it

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Contents

1 Assignment 1 . . . **1**

1.1 Implement in MATLAB 3rd-, 5th-, 7th-order polynomials for $q_i > q_f$ and $q_i < q_f$ and for $t \in [t_i, t_f]$ and $t \in [0, \Delta T]$ . . . 1

2 Assignment 2 . . . **4**

2.1 Implement in MATLAB the trapezoidal trajectory taking into account the different constraints. . . . 4

3 Assignment 3 . . . **8**

3.1 Interpolating polynomials with computed velocities at path points and imposed velocity at initial/final points. . . . 8

3.2 Interpolating polynomials with continuous accelerations at path points and imposed velocity at initial/final points (+ Thomas algorithm) . . . 9

4 Assignment 4 . . . **10**

4.1 Compute cubic splines based on the accelerations with assigned initial and final velocities. . . . 10

4.2 Compute the smoothing cubic splines. . . . 11

5 Assignment 5 . . . **12**

5.1 Compute the 3D trajectory (position, velocity, acceleration and jerk) in the picture as a combination of linear and circular motion primitives and compare it with the trajectory obtained using one of the multi-point methods. . . . 12

6 Assignment 6 . . . **14**

6.1 Let p_1, p_2, p_3 be three points on a sphere of center P_0 and radius R. Design the trajectory such that (1) the EE will pass through the three points along the shortest path, and (2) the z axis of the EE is always orthogonal to the sphere. . . . 14

1 Assignment 1

1.1 Implement in MATLAB 3rd-, 5th-, 7th-order polynomials for $q_i > q_f$ and $q_i < q_f$ and for $t \in [t_i, t_f]$ and $t \in [0, \Delta T]$

All polynomial trajectories can be expressed as:

$$
\begin{aligned}
q(t) &= a_7(t-t_i)^7 + a_6(t-t_i)^6 + a_5(t-t_i)^5 + a_4(t-t_i)^4 + a_3(t-t_i)^3 + a_2(t-t_i)^2 + a_1(t-t_i) + a_0 \\
\dot{q}(t) &= 7a_7(t-t_i)^6 + 6a_6(t-t_i)^5 + 5a_5(t-t_i)^4 + 4a_4(t-t_i)^3 + 3a_3(t-t_i)^2 + 2a_2(t-t_i) + a_1 \\
\ddot{q}(t) &= 42a_7(t-t_i)^5 + 30a_6(t-t_i)^4 + 20a_5(t-t_i)^3 + 12a_4(t-t_i)^2 + 6a_3(t-t_i) + 2a_2 \\
\dddot{q}(t) &= 210a_7(t-t_i)^4 + 120a_6(t-t_i)^3 + 60a_5(t-t_i)^2 + 24a_4(t-t_i) + 6a_3 \\
\ddddot{q}(t) &= 840a_7(t-t_i)^3 + 360a_6(t-t_i)^2 + 120a_5(t-t_i) + 24a_4
\end{aligned}
$$

For the 3rd-order polynomial $a_7 = a_6 = a_5 = a_4 = 0$ while for the 5th-order polynomial $a_7 = a_6 = 0$.

The problem of determining the a_i coefficients of the polynomials is solved by setting up a system of equations using initial and final conditions on velocity (3rd-order), velocity and acceleration (5th-order), velocity, acceleration and jerk (7th-order).

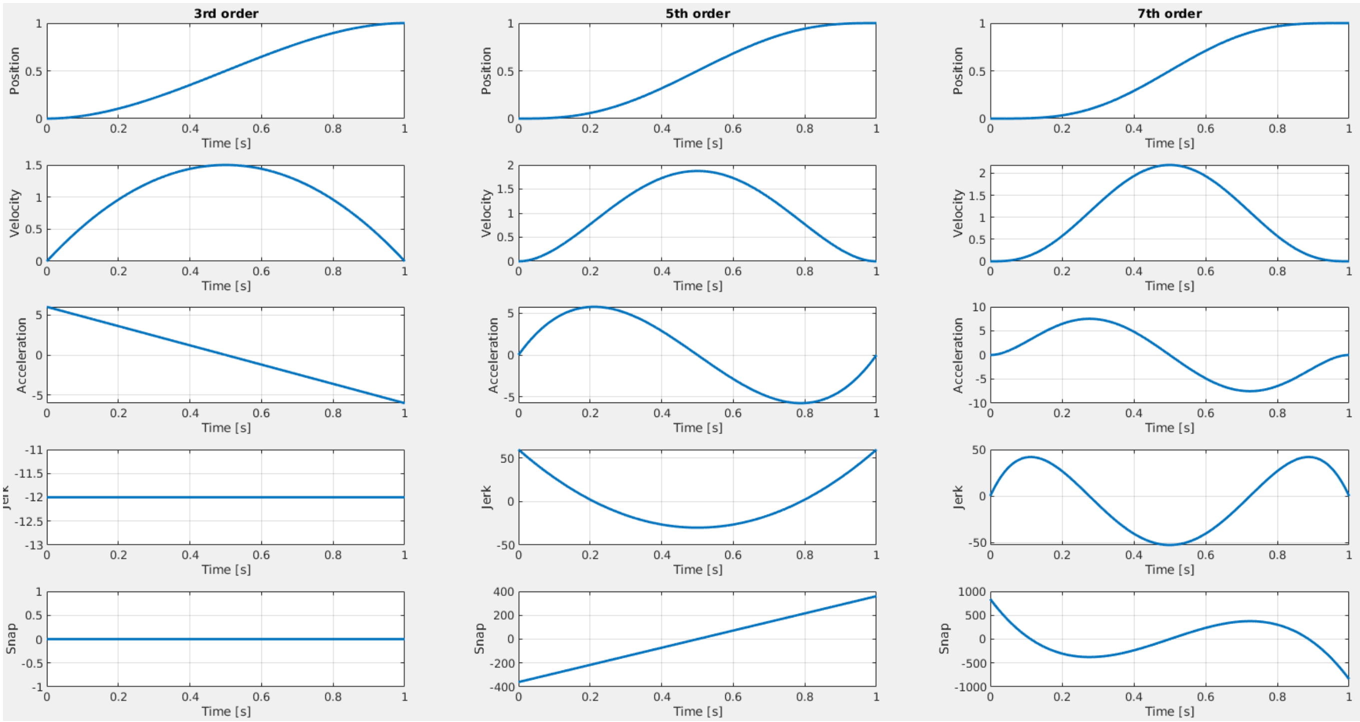

Figure 1: 3rd-, 5th-, 7th-order polynomial trajectories with $q_i < q_f$ and $t \in [0, \Delta T]$, $q_i = 0, q_f = 1, \Delta T = 1, v_i = v_f = a_i = a_f = j_i = j_f = 0$.

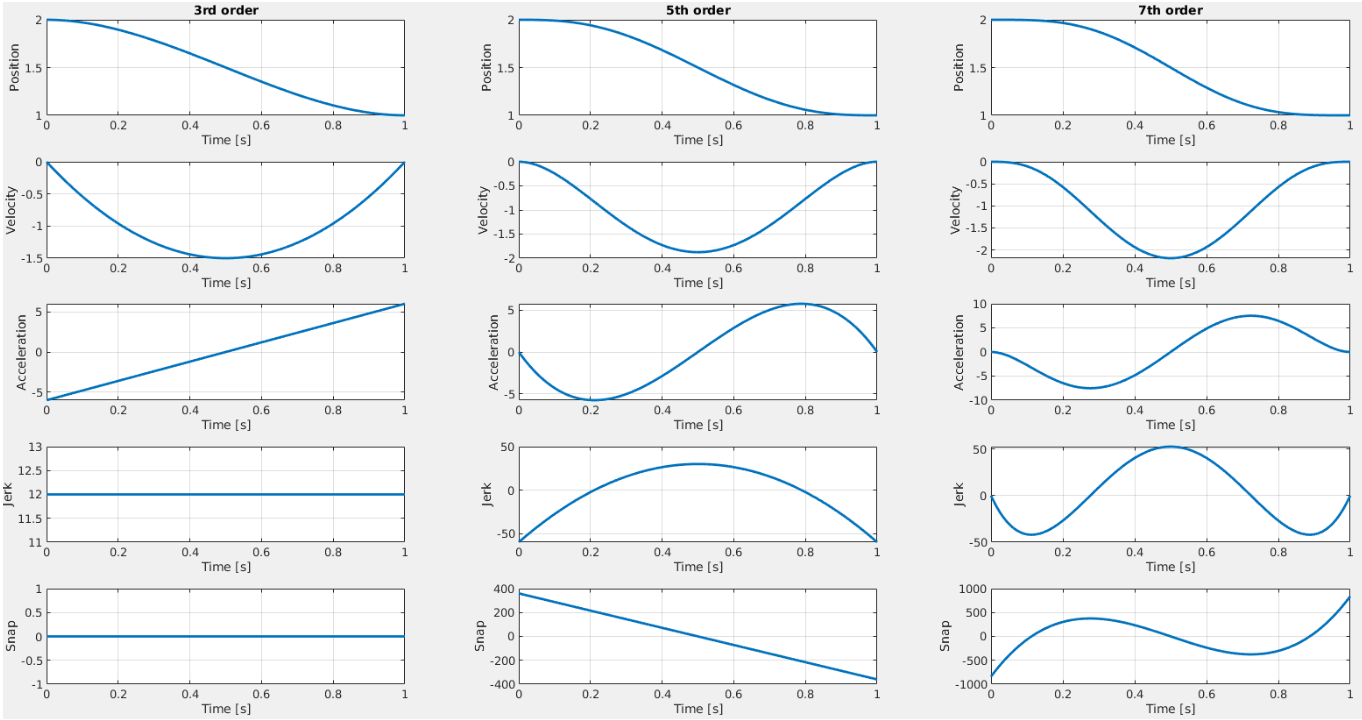

Figure 2: 3rd-, 5th-, 7th-order polynomial trajectories with $q_i > q_f$ and $t \in [0, \Delta T]$, $q_i = 2, q_f = 1, \Delta T = 1, v_i = v_f = a_i = a_f = j_i = j_f = 0$.

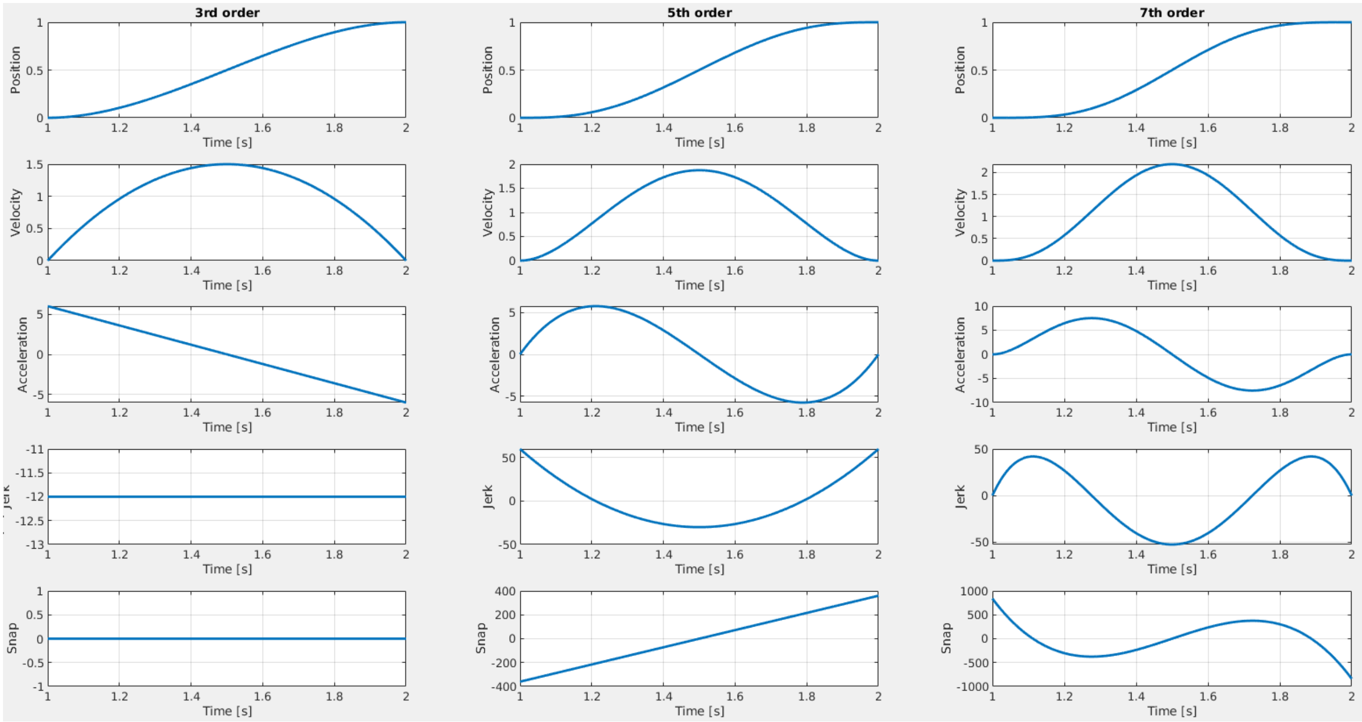

Figure 3: 3rd-, 5th-, 7th-order polynomial trajectories with $q_i < q_f$ and $t \in [t_i, t_f]$, $q_i = 0, q_f = 1, t_i = 1, t_f = 2, v_i = v_f = a_i = a_f = j_i = j_f = 0$.

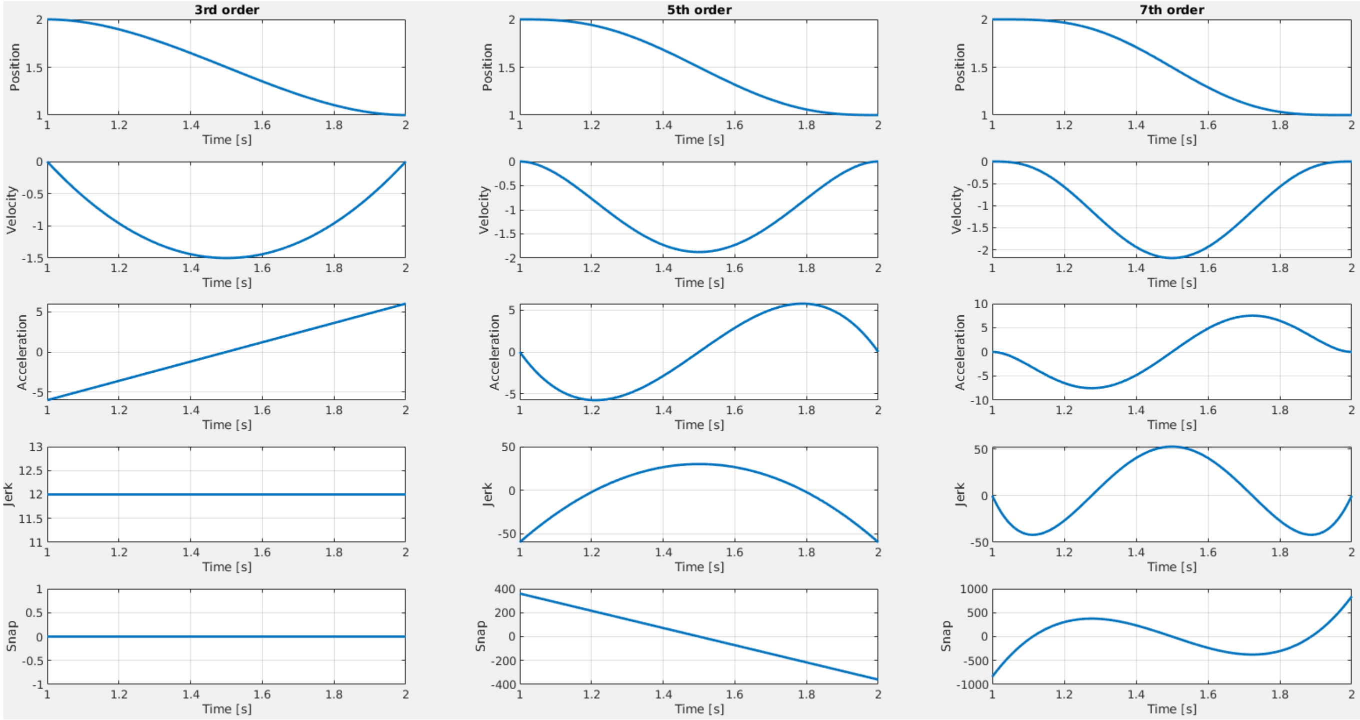

Figure 4: 3rd-, 5th-, 7th-order polynomial trajectories with $q_i > q_f$ and $t \in [t_i, t_f]$, $q_i = 2, q_f = 1, t_i = 1, t_f = 2, v_i = v_f = a_i = a_f = j_i = j_f = 0$.

2 Assignment 2

2.1 Implement in MATLAB the trapezoidal trajectory taking into account the different constraints.

The general expression for a trapezoidal velocity profile is:

$$q(t) = \begin{cases} q_i + \dot{q}_i(t-t_i) + \frac{\dot{q}_c - \dot{q}_i}{2t_a}(t-t_i)^2 & t_i \le t \le t_a + t_i \\ q_i + \dot{q}_i\frac{t_a}{2} + \dot{q}_c(t - t_i - \frac{t_a}{2})^2 & t_a + t_i \le t \le t_f - t_d \\ q_f - \dot{q}_f(t_f - t) - \frac{\dot{q}_c - \dot{q}_f}{2t_d}(t_f - t)^2 & t_f - t_d \le t \le t_f \end{cases}$$

If the initial and final velocities are null, then $t_a = t_d = t_c$.
If $\dot{q}_i = \dot{q}_f = 0$ then the possible constraints are:

- t_c:

$$\ddot{q}_c = \frac{q_f - q_i}{t_c t_f - t_c^2} \qquad \dot{q}_c = \ddot{q}_c t_c$$

- $\ddot{q}_c$:

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

- $\dot{q}_c$:

$$t_c = \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c} \qquad \ddot{q}_c = \frac{\dot{q}_c^2}{q_i - q_f + \dot{q}_c t_f}$$

- $\ddot{q}_c, \dot{q}_c$:

$$t_c = \frac{\dot{q}_c}{\ddot{q}_c} \qquad t_f = \frac{\dot{q}_c^2 + \ddot{q}_c(q_f - q_i)}{\dot{q}_c \ddot{q}_c}$$

In all cases the feasibility condition is that $2t_c \le t_f - t_i$.
If $\dot{q}_i, \dot{q}_f \neq 0$ the possible constraints are:

- $\ddot{q}_{c,max}$:

$$\dot{q}_c = \frac{1}{2}(\dot{q}_i + \dot{q}_f + \ddot{q}_{c,max}\Delta T + \sqrt{\ddot{q}_{c,max}^2\Delta T^2 - 4\ddot{q}_{c,max}\Delta q + 2\ddot{q}_{c,max}(\dot{q}_i + \dot{q}_f)\Delta T - (\dot{q}_i - \dot{q}_f)^2} \qquad t_a = \frac{\dot{q}_c - \dot{q}_i}{\ddot{q}_{c,max}} \; t_d = \frac{\dot{q}_c - \dot{q}_f}{\ddot{q}_{c,max}}$$

The trajectory is feasible when the argument of the square root is positive, and when the maximum acceleration satisfies:

$$\ddot{q}_{c,max}\Delta q > \frac{\left|\dot{q}_i^2 - \dot{q}_f^2\right|}{2} \qquad \ddot{q}_{c,max} \ge \ddot{q}_{c,lim} = \frac{2\Delta q - (\dot{q}_i - \dot{q}_f)\Delta T + \sqrt{4\Delta q^2 - 4\Delta q(\dot{q}_i + \dot{q}_f)\Delta T + 2(\dot{q}_i^2 + \dot{q}_f^2)\Delta T^2}}{\Delta T^2}$$

when $\ddot{q}_{c,max} = \ddot{q}_{c,lim}$ there is no constant velocity phase.

- $\ddot{q}_{max}, \dot{q}_{max}$: First we compute the condition:

$$\ddot{q}_{c,max}\Delta q \gtreqless \dot{q}_{c,max}^2 - \frac{\dot{q}_i^2 + \dot{q}_f^2}{2}$$

If the above is $>$ then:

$$\dot{q}_c = \dot{q}_{c,max} \quad t_a = \frac{\dot{q}_{c,max} - \dot{q}_i}{\ddot{q}_{c,max}} \quad t_d = \frac{\dot{q}_{c,max} - \dot{q}_f}{\ddot{q}_{c,max}} \quad \Delta T = \frac{\Delta q \ddot{q}_{c,max} + \dot{q}_{c,max}^2}{\ddot{q}_{c,max}\dot{q}_{c,max}}$$

If the above is $\le$ then:

$$\dot{q}_c = \dot{q}_{c,lim} = \sqrt{\ddot{q}_{c,max}\Delta q + \frac{\dot{q}_i^2 + \dot{q}_f^2}{2}} < \dot{q}_{c,max} \quad t_a = \frac{\dot{q}_{c,lim} - \dot{q}_i}{\ddot{q}_{c,max}} \quad t_d = \frac{\dot{q}_{c,lim} - \dot{q}_f}{\ddot{q}_{c,max}}$$

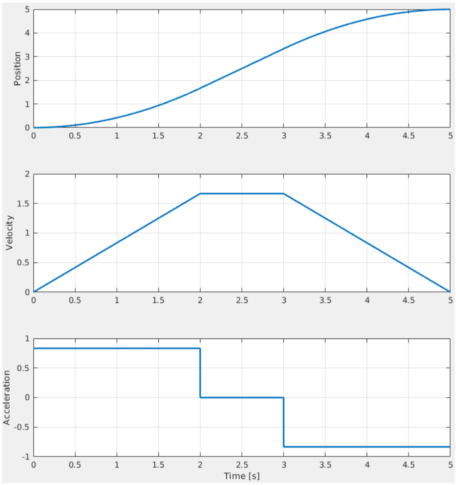

Figure 5: $t_c = 2s$

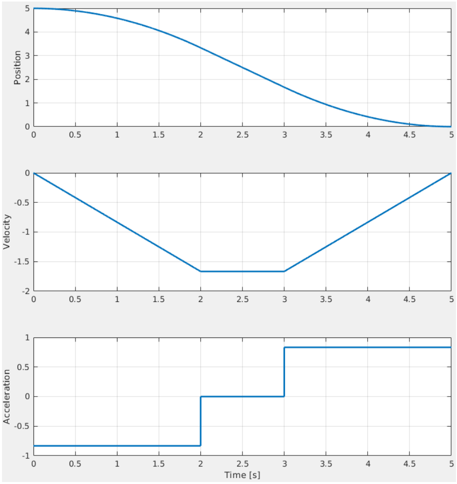

Figure 6: $t_c = 2s, q_f > q_i$

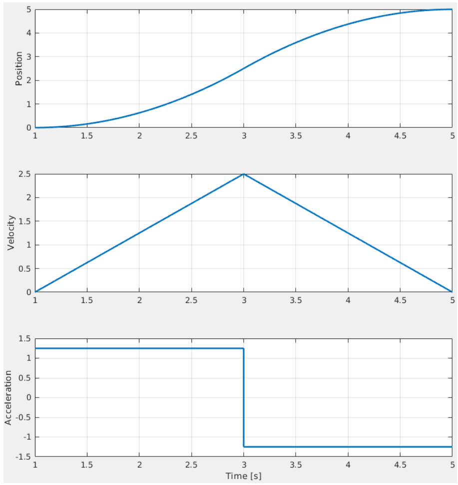

Figure 7: $t_c = 2s, t_i \neq 0$

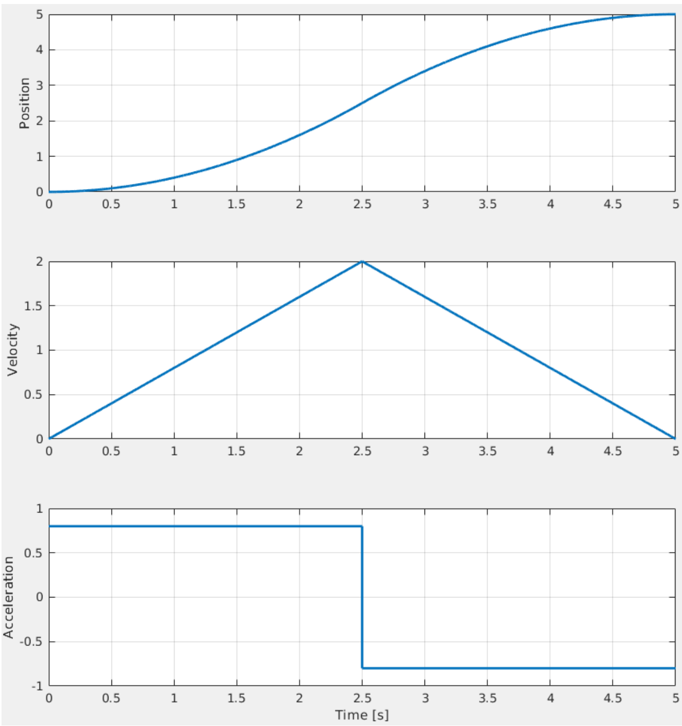

Figure 8: $v_c = v_{c,lim} = 2$

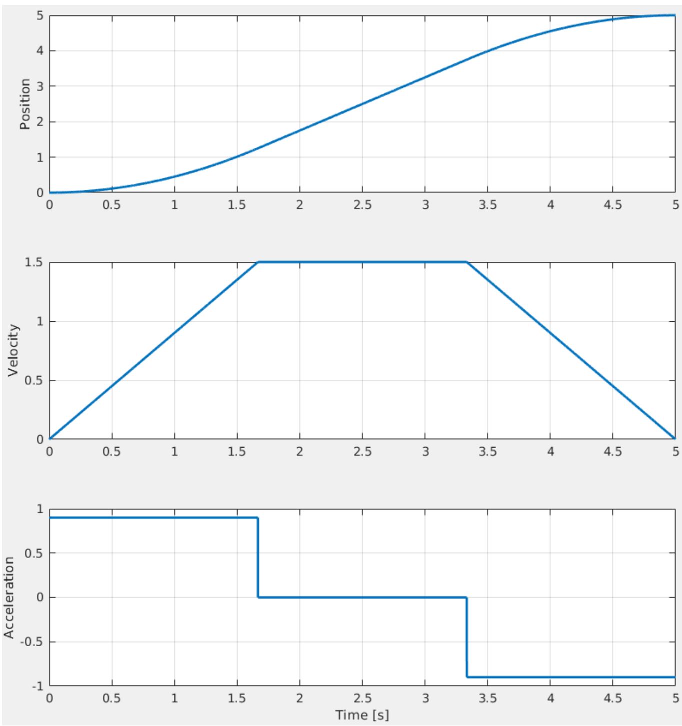

Figure 9: $v_c = 2$

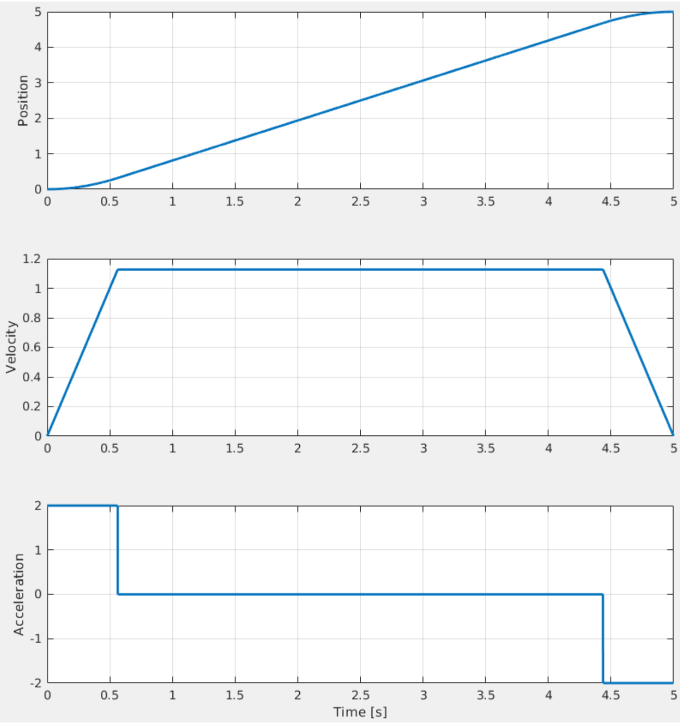

Figure 10: $a_c = 2$

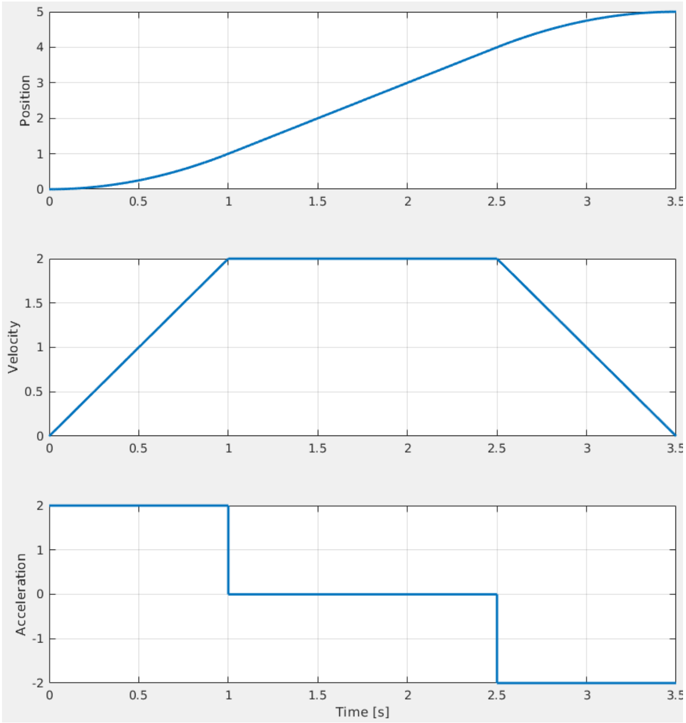

Figure 11: $a_c, v_c = 2$

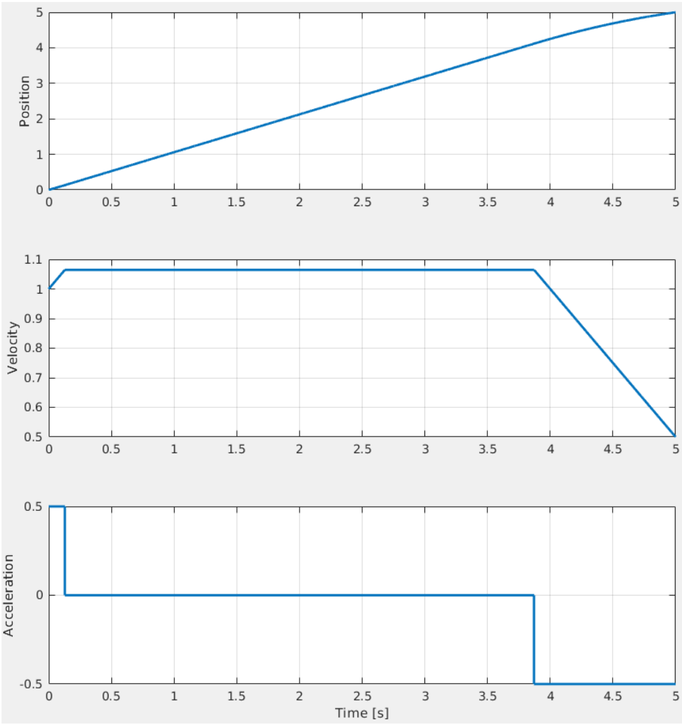

Figure 12: $\dot{q}_i = 1, \dot{q}_f = 0.5, \ddot{q}_{max} = 0.5$

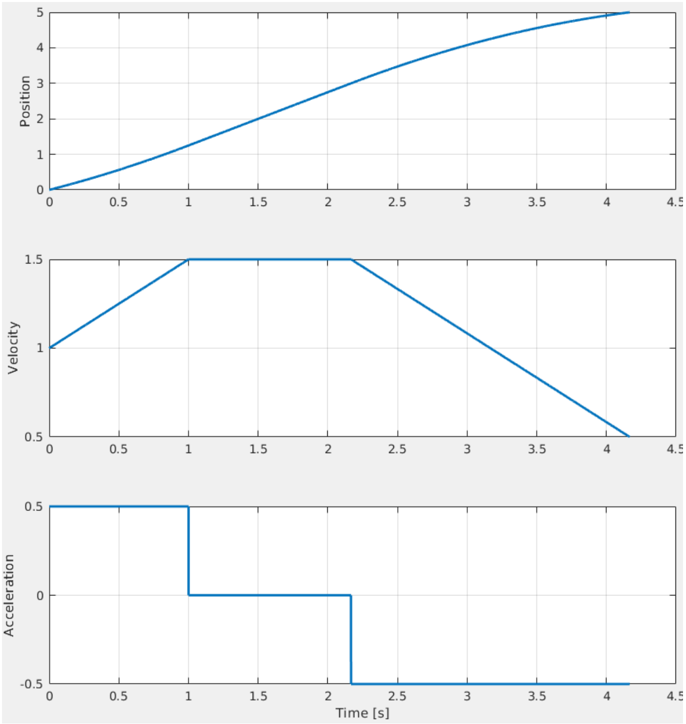

Figure 13: $\dot{q}_i = 1, \dot{q}_f = 0.5, \dot{q}_{max} = 1.5, \ddot{q}_{max} = 0.5$

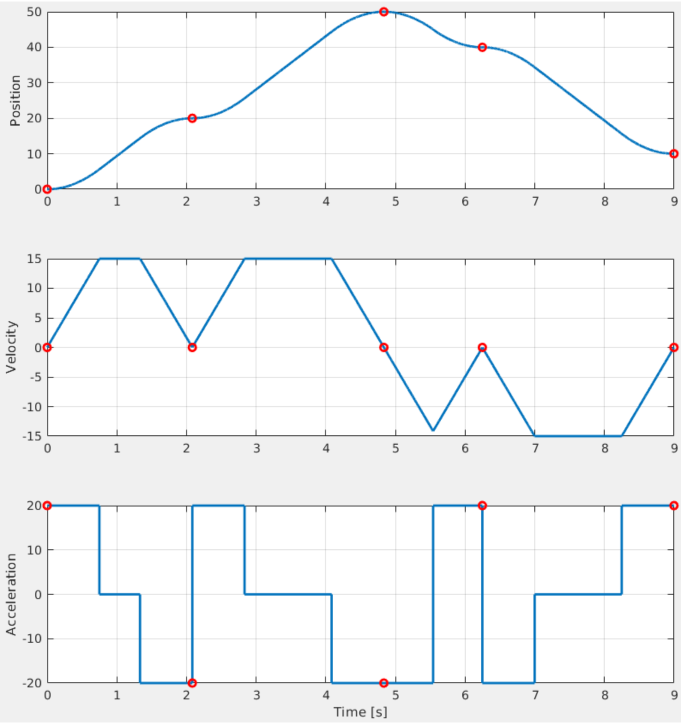

Figure 14: Multipoint trajectory without velocity heuristic

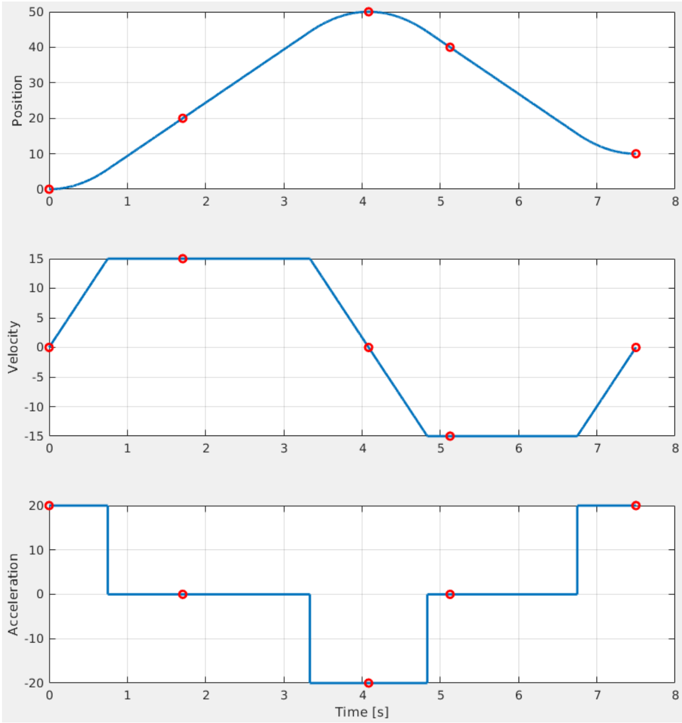

Figure 15: Multipoint trajectory with velocity heuristic

To generate a multipoint trajectory we simply repeat the computation for each consecutive point pair. To improve the resulting velocity and acceleration profiles we can apply an heuristic to assign velocities to each waypoint, so that a less jagged profile is obtained:

$$\dot{q}(t_i) = \dot{q}_i$$
$$\dot{q}(t_k) = \begin{cases} 0 & \text{if } sign(\Delta Q_k) \neq sign(\Delta Q_{k+1}) \\ sign(\Delta Q_k)\dot{q}_{max} & \text{if } sign(\Delta Q_k) = sign(\Delta Q_{k+1}) \end{cases}$$
$$\dot{q}(t_f) = \dot{q}_f$$

3 Assignment 3

3.1 Interpolating polynomials with computed velocities at path points and imposed velocity at initial/final points.

To implement multipoint trajectories we concatenate cubic splines. The interpolating trajectory is then:

$$q(t) \coloneqq \{\Pi_k(t), t \in [t_k, t_{k+1}], k = 0, \dots, n-1\}$$

where:

$$\Pi(t) = a_3^k(t-t_k)^3 + a_2^k(t-t_k)^2 + a_1^k(t-t_k) + a_0^k$$

Fixing velocities on all path points yields:

$$\begin{cases} a_0^k &= q_k \\ a_1^k &= \dot{q}_k \\ a_2^k &= \frac{1}{T_k}\left(\frac{3(q_{k+1}-q_k)}{T_k} - 2\dot{q}_k - \dot{q}_{k+1}\right) \\ a_3^k &= \frac{1}{T_k^2}\left(\frac{2(q_k-q_{k+1})}{T_k} + \dot{q}_k + \dot{q}_{k+1}\right) \end{cases}$$

where $T_k = t_{k+1} - t_k$.

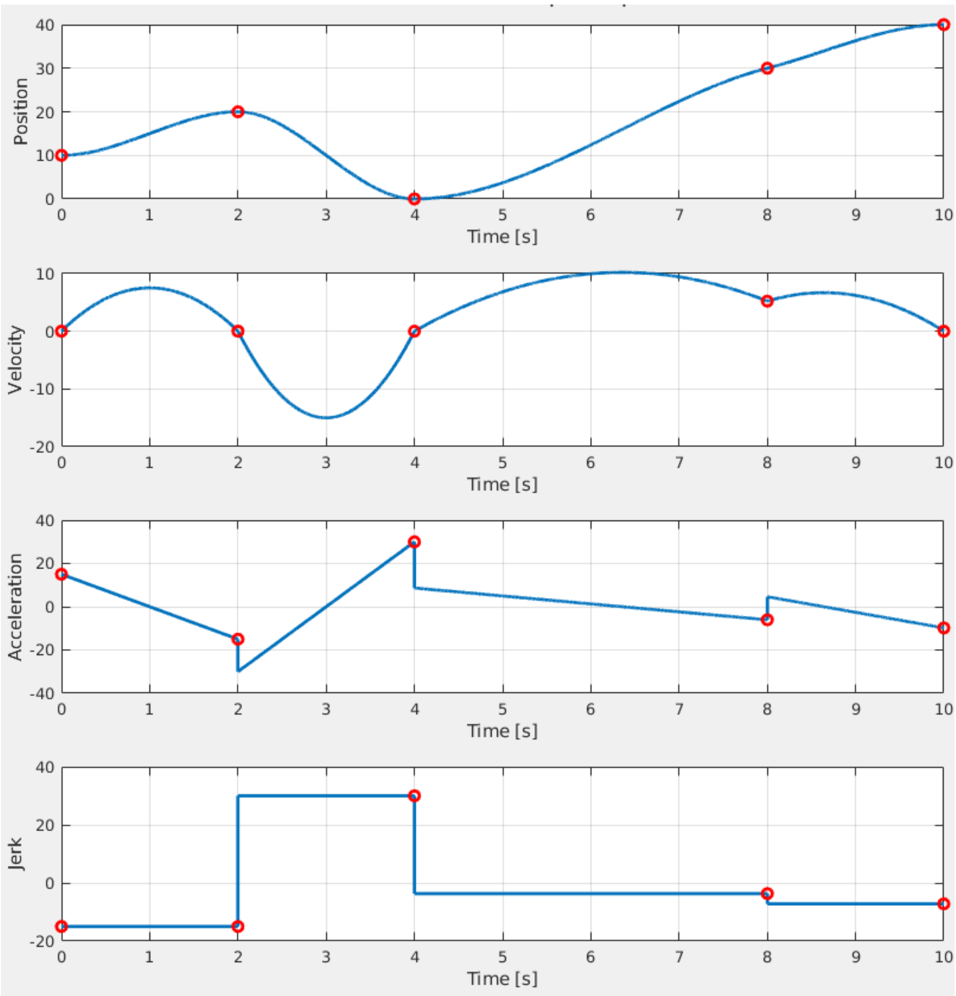

Figure 16: Trajectory through $q_k = \begin{bmatrix}10 & 20 & 0 & 30 & 40\end{bmatrix}$ at times $t_k = \begin{bmatrix}0 & 2 & 4 & 8 & 10\end{bmatrix}$ with velocities $\dot{q}_k = \begin{bmatrix}0 & 0 & 0 & 5.2 & 0\end{bmatrix}$

3.2 Interpolating polynomials with continuous accelerations at path points and imposed velocity at initial/final points (+ Thomas algorithm)

Path point velocities are not generally known. We can estimate them with Euler's approximation:

$$v_k = \frac{q_k - q_{k-1}}{t_k - t_{k-1}} \implies \begin{cases} \dot{q}(t_0) & = \dot{q}_0 \\ \dot{q}(t_k) & = \begin{cases} 0 & \text{if } sign(\Delta Q_k) \neq sign(v_{k+1}) \\ \frac{v_k + v_{k+1}}{2} & \text{if } sign(v_k) = sign(v_{k+1}) \end{cases} \\ \dot{q}(t_n) & = \dot{q}_n \end{cases}$$

Imposing acceleration continuity at path points results in the definition of a linear system $A\dot{q} = c$, where A is a tridiagonal matrix. Thanks to this property the system can be solved for $\dot{q}$ efficiently using Thomas' algorithm. Given:

$$\begin{bmatrix} b_1 & c_1 & 0 & \dots & & & & \\ a_2 & b_2 & c_2 & 0 & \dots & & & \\ 0 & \ddots & \ddots & \ddots & 0 & \dots & & \\ & 0 & a_k & b_k & c_k & 0 & \dots & \\ & & 0 & \ddots & \ddots & \ddots & 0 & \\ & & & 0 & a_{n-1} & b_{n-1} & c_{n-1} & \\ & & & & 0 & a_n & b_n & \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_k \\ \vdots \\ \dot{q}_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_k \\ \vdots \\ d_n \end{bmatrix}$$

Thomas' algorithm is as follows:

Forward elimination:

for $k = 2 : 1 : n$ **do**
  $m \leftarrow \frac{a_k}{b_{k-1}}$
  $b_k \leftarrow b_k - mc_{k-1}$
  $d_k \leftarrow d_k - md_{k-1}$
end for

Backward substitution:

$x_n \leftarrow \frac{d_n}{b_n}$
for $k = 2 : 1 : n$ **do**
  $x_k \leftarrow \frac{d_k - c_k x_{k+1}}{b_k}$
end for

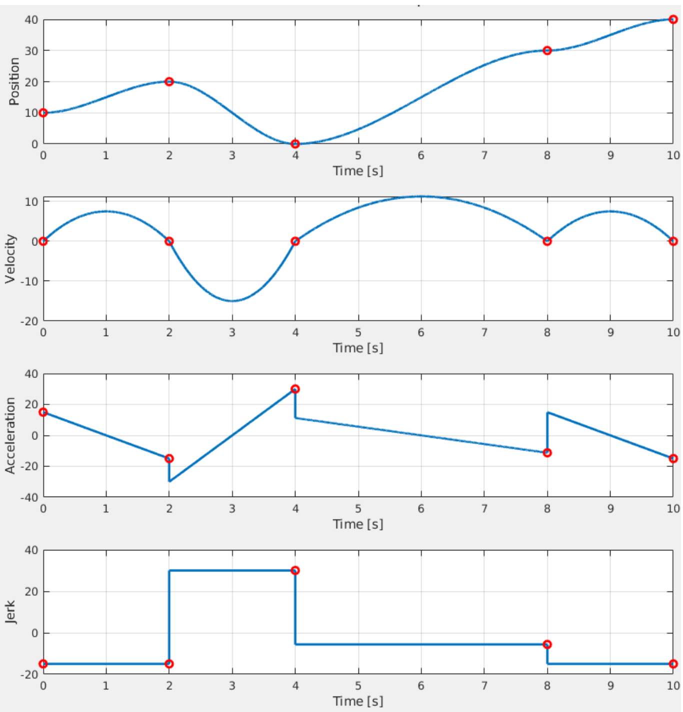

Figure 17: Trajectory interpolation with Euler's approximation.

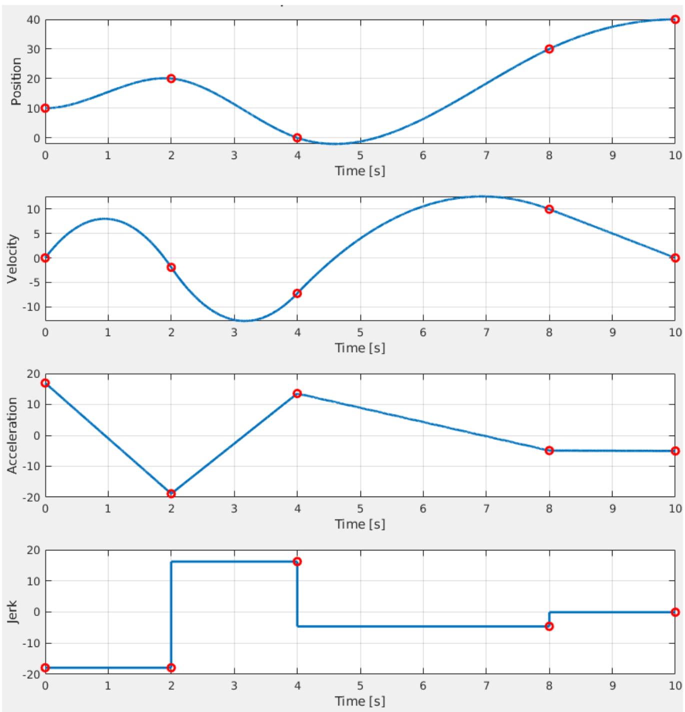

Figure 18: Trajectory interpolation with continuous accelerations.

4 Assignment 4

4.1 Compute cubic splines based on the accelerations with assigned initial and final velocities.

The interpolation based on accelerations is centred around the solution of the system of linear equations:

$$A\ddot{q} = c$$

$$\begin{bmatrix} 2T_0 & T_0 & 0 & & & & \\ T_0 & 2(T_0+T_1) & T_0 & 0 & & & \\ & \ddots & \ddots & \ddots & & & \\ & & T_k & 2(T_k+T_{k+1}) & T_{k+1} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & T_{n-2} & 2(T_{n-2}+T_{n-1}) & T_{n-1} \\ & & & & & T_{n-1} & 2T_{n-1} \end{bmatrix} \begin{bmatrix} \ddot{q}_0 \\ \vdots \\ \ddot{q}_k \\ \vdots \\ \ddot{q}_n \end{bmatrix} = \begin{bmatrix} 6\left(\frac{q_1-q_0}{T_0} - \dot{q}_0\right) \\ \vdots \\ 6\left(\frac{q_k-q_{k-1}}{T_{k-1}} - \frac{q_{k-1}-q_{k-2}}{T_{k-2}}\right) \\ \vdots \\ 6\left(\dot{q}_n - \frac{q_n-q_{n-1}}{T_{n-1}}\right) \end{bmatrix}$$

Thanks to the tridiagonal form of matrix A, the system can be solved with Thomas' algorithm.

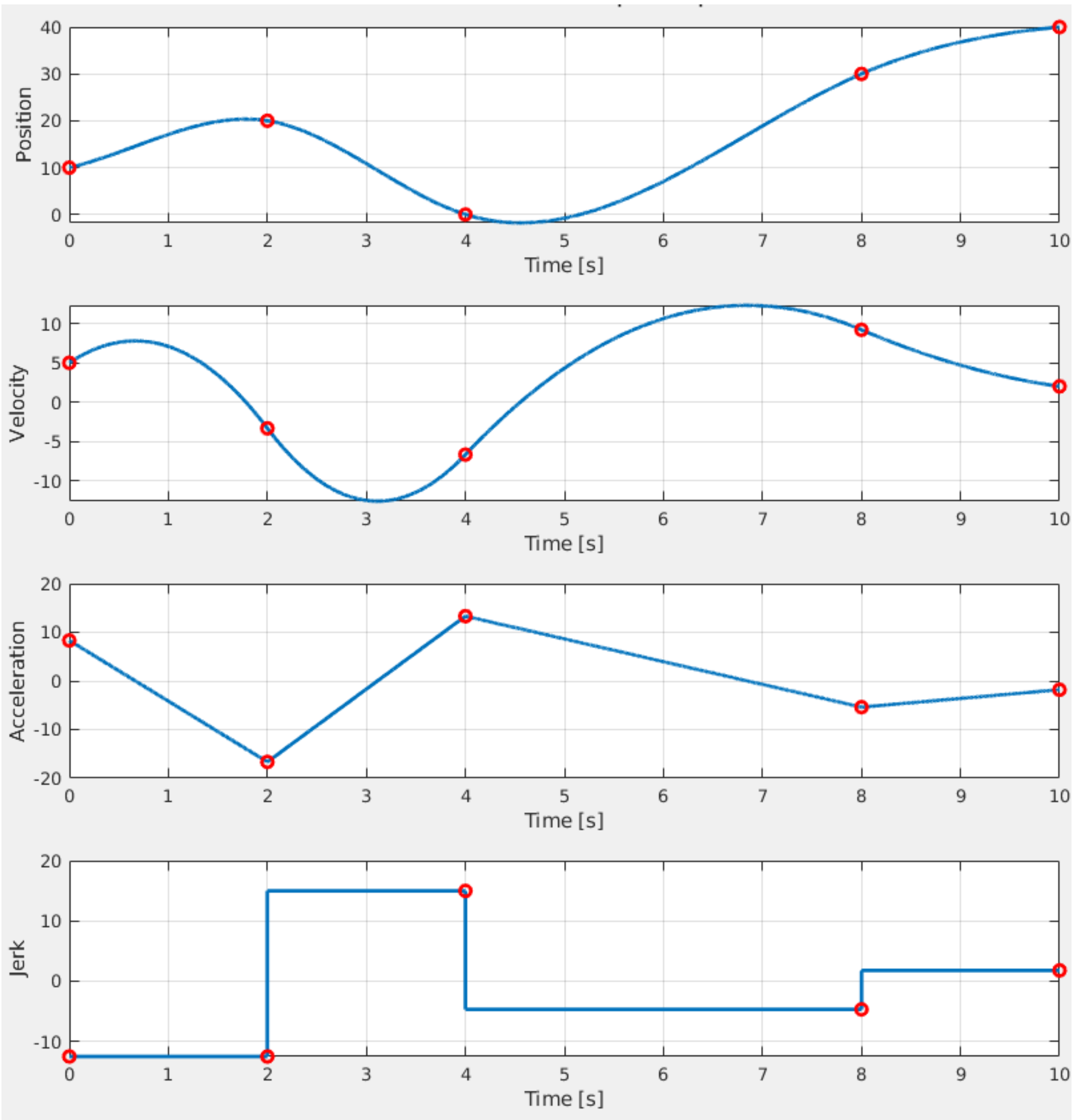

Figure 19: Trajectory based on accelerations through $q_k = \begin{bmatrix}10 & 20 & 0 & 30 & 40\end{bmatrix}$ at times $q_k = \begin{bmatrix}0 & 2 & 4 & 8 & 10\end{bmatrix}$ with $\dot{q}_i = 5$ and $\dot{q}_f = 2$.

4.2 Compute the smoothing cubic splines.

The smoothing cubic splines express a trade-off between the fitting of the given points and the minimization of the curvature and acceleration of the trajectory, through a parameter μ. The closeness of the approximation is expressed by a weight vector w, which weighs every point in the trajectory. The closer w_k is to zero, the closer the resulting trajectory is to the interpolation of the point.

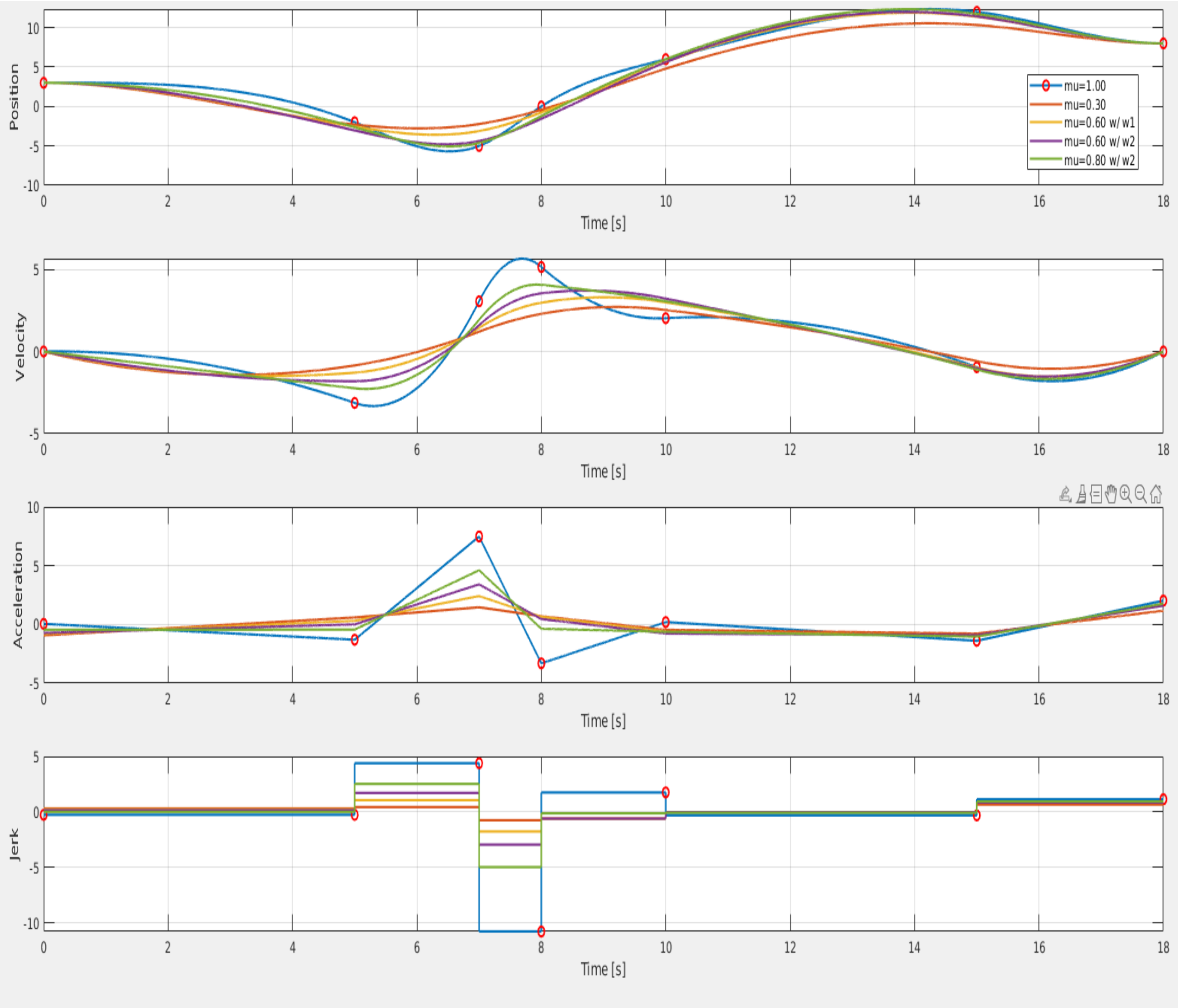

Figure 20: Comparison of smoothing cubic splines. $w1 = \begin{bmatrix} \infty & 1 & 1 & 1 & 1 & 1 & \infty \end{bmatrix}$,$w2 = \begin{bmatrix} \infty & 1 & 5 & 1 & 1 & 1 & \infty \end{bmatrix}$. Note that the actual used value is $1/w_k$.

5 Assignment 5

5.1 Compute the 3D trajectory (position, velocity, acceleration and jerk) in the picture as a combination of linear and circular motion primitives and compare it with the trajectory obtained using one of the multi-point methods.

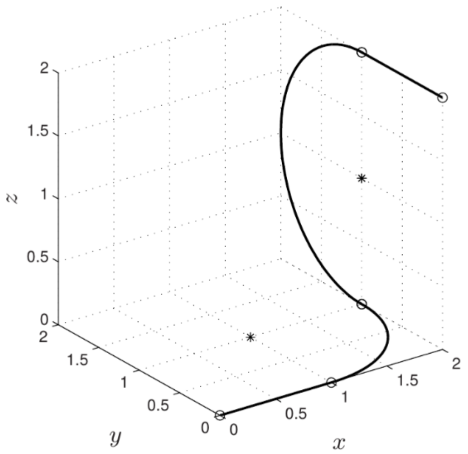

Figure 21: 3D trajectory - reference.

Operational space trajectories can be computed by composing multiple motion primitives. The motion primitives used here are:

- Rectilinear path:

$$p(u) = p_i + u(p_f - p_i) \qquad u \in [0, 1]$$

- Circular path:

$$p(u) = c + Rp'(u) = c + R \begin{bmatrix} \rho\cos(u) \\ \rho\sin(u) \\ 0 \end{bmatrix} = c + \begin{bmatrix} (P-c)' & e_3' \times (P-c)' & e_3' \end{bmatrix} \begin{bmatrix} \rho\cos(u) \\ \rho\sin(u) \\ 0 \end{bmatrix} \qquad u \in [0, \theta]$$

  where c is the position of the center of the circular path, P is the position of the starting point on the circle and $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$.

For comparison, a 3D trajectory is computed by computing the smoothing cubic splines along the three directions. The smoothing spline trajectory is also computed with added waypoints to improve the tracking of the position.

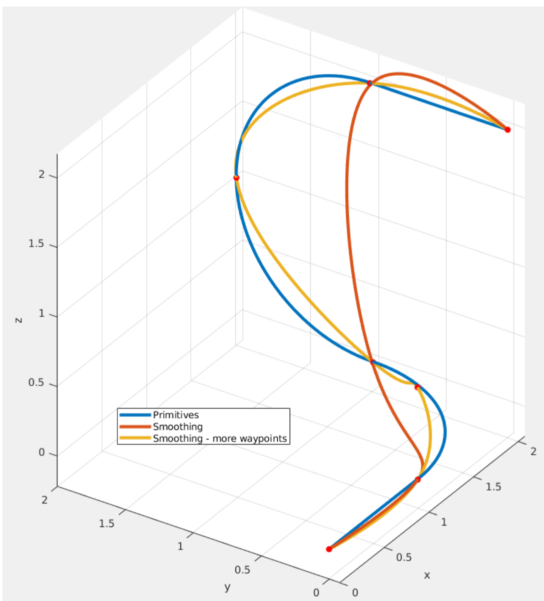

Figure 22: 3D trajectory - motion primitives vs smoothing splines ($w_k = \infty \forall k$, $\mu = 1$).

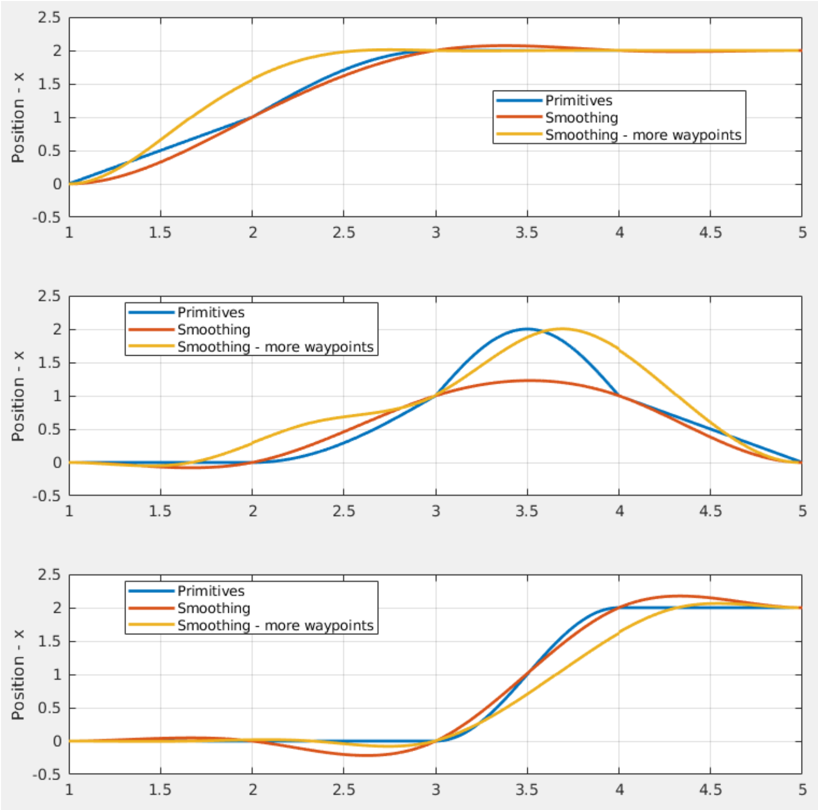

Figure 23: 3D trajectory - Position comparison.

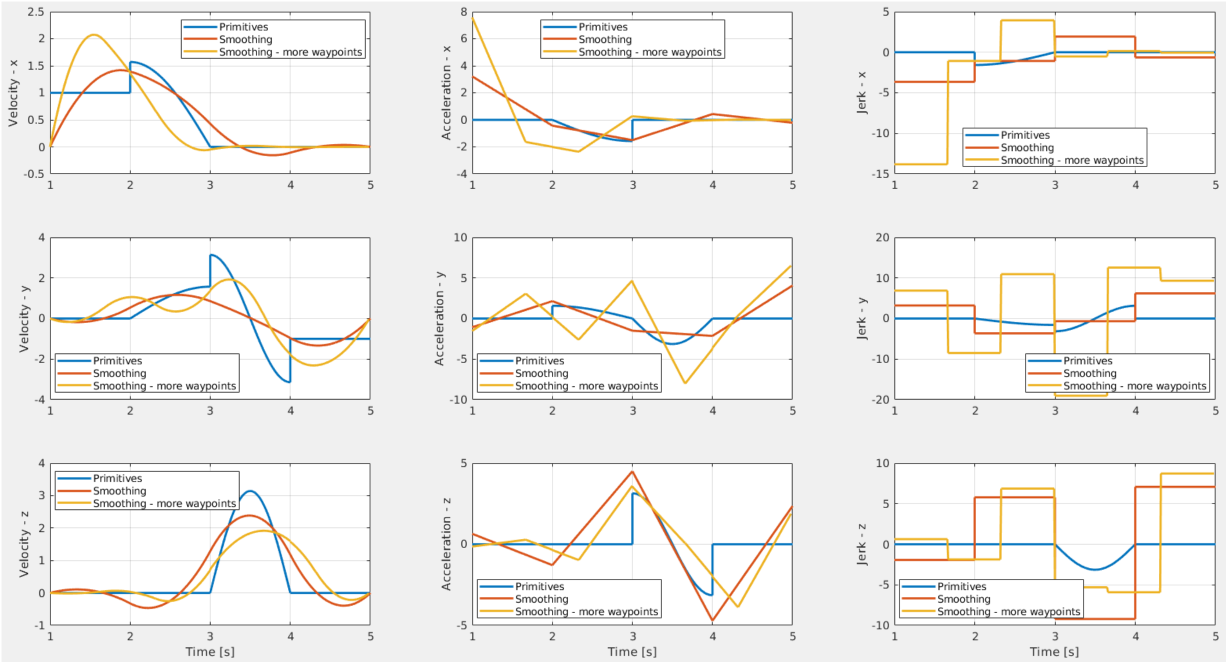

Figure 24: 3D trajectory - Velocity, acceleration, jerk comparison.

6 Assignment 6

6.1 Let p_1, p_2, p_3 be three points on a sphere of center P_0 and radius R. Design the trajectory such that (1) the EE will pass through the three points along the shortest path, and (2) the z axis of the EE is always orthogonal to the sphere.

The shortest path γ on a spherical surface lies on a great circle, which is the biggest circle that belongs to the spherical surface. Such a path is called geodesic.

First of all, the radius vectors of the starting and ending points, p_1 and p_2, of the geodesic are used to compute its direction:

$$\begin{matrix} r_1 = \frac{(p_1 - p_0)}{R} \\ r_2 = \frac{(p_2 - p_0)}{R} \end{matrix} \implies r_g = r_1 \times r_2$$

The rotation matrix that rotates the great circle with $z = 0$ to align it with the geodesic is then:

$$R_\gamma = \begin{bmatrix} r_1 & r_g \times r_1 & r_g \end{bmatrix}$$

where the cross product between r_g and r_1 was arbitrarily chosen (r_2 would have worked as well) to obtain a third orthogonal direction.

Using the rotation matrix we can rotate the parametric form of the great circle as follows:

$$\gamma = R_\gamma \begin{bmatrix} R\cos(u) \\ R\sin(u) \\ 0 \end{bmatrix} + p_0$$

From the parametrization, we obtain the axes of the Frenet frame associated to each point:

$$t = \frac{d\gamma}{du} = R_\gamma \begin{bmatrix} -R\sin(u) \\ R\cos(u) \\ 0 \end{bmatrix} \qquad n = \frac{d^2\gamma}{du^2} = R_\gamma \begin{bmatrix} -R\cos(u) \\ -R\sin(u) \\ 0 \end{bmatrix} \qquad b = t \times n$$

The computed normal vector n is always perpendicular to the sphere surface. In fact it lays on the same direction of the corresponding radius vector of the parametric geodesic:

$$r_\gamma = \frac{\gamma - p_0}{R} = R_\gamma \begin{bmatrix} \cos(u) \\ \sin(u) \\ 0 \end{bmatrix} \implies r_\gamma \times n = R_\gamma \left(\begin{bmatrix} \cos(u) \\ \sin(u) \\ 0 \end{bmatrix} \times \begin{bmatrix} -\cos(u) \\ -\sin(u) \\ 0 \end{bmatrix} \right)$$

The cross product in the parentheses is null for any value of u, therefore n is always parallel to the corresponding radius vector. Since the radius vector's direction is perpendicular to the sphere by definition and the normal vector n starts from a point belonging to the sphere's surface, the vector n is always perpendicular to the sphere's surface.

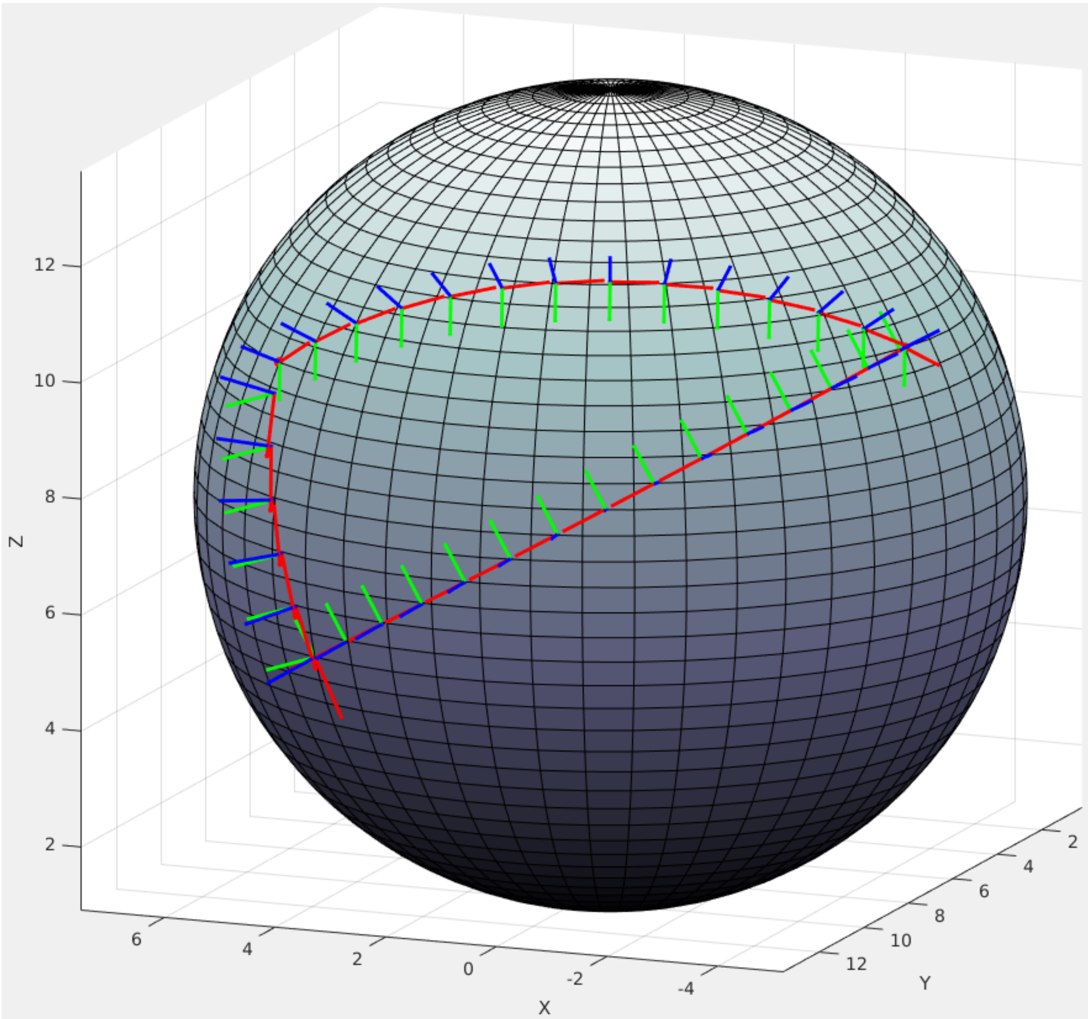

Figure 25: Geodesics between three random sphere points with the associated Frenet frames (n in blue, t in green, b in red).