Trajectory construction

Robotics and Computer vision (RoVi)

Aljaz Kramberger

alk@mmmi.sdu.dk

SDU Robotics

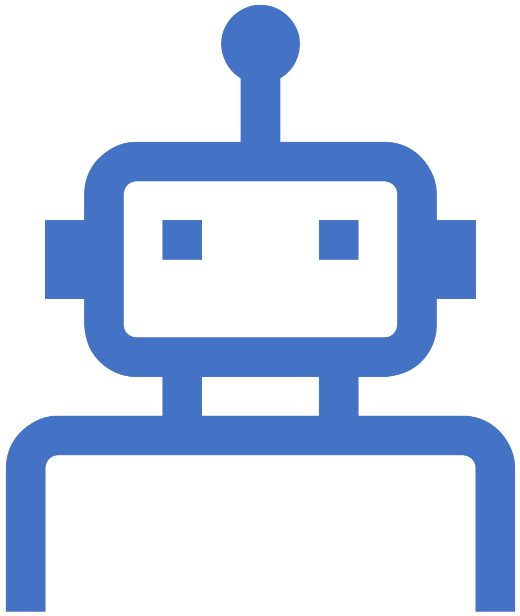
The Maersk Mc-Kinney Moller Institute

University of Southern Denmark

Based on M. Mihelj, T. Bajd. et al. Robtics – second edition, Springer, 2018.

Siciliano, Bruno, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. Robotics: modelling, planning and control. Springer Science & Business Media, 2010.

A Geometrical Introduction to Robotics and Manipulation, Richard Murray and Zexiang Li and Shankar S. Sastry 2011, CRC Press, Honkong



Agenda

- Introduction to trajectory’s
- general point to point movement,
  - linear interpolation between two frames,
  - quaternion interpolation,
  - parabolic blends,
  - example.

Trajectory's

General we distinguish…

Joint trajectories

Tool trajectories

Differences

- Joint trajectories are presented as:
  - $Trj_{joint} = \left\{q_i^1, q_i^2, \dots q_i^n, t_i\right\}_{i=1}^{T,n}$, t_i , $i = 1, \dots, T_i$ are the number of samples and n number of joints.
  - can directly be send to the robot.
  - precise deterministic execution – (better tracking),
  - the movement is not necessary linear although it was constructed such.

- Tool trajectories are presented as:
  - $Trj_{joint} = \{p_i, o_i, t_i\}_{i=1}^{T}$ where $p = (p_x, p_y, p_z)$ is the positional part and $o = \boldsymbol{R}$ orientational part of the trajectory.
  - Invers kinematics is needed in order to execute the trajectory.

Trajectory construction

- Trajectories can be constructed in different ways:
  - trajectories can be acquired with demonstration,
  - simple construction with point to point interpolation,
  - splines and polynomials,
  - using dynamic systems,
  - etc.

Point-to-point movement

Exercise

Exercise

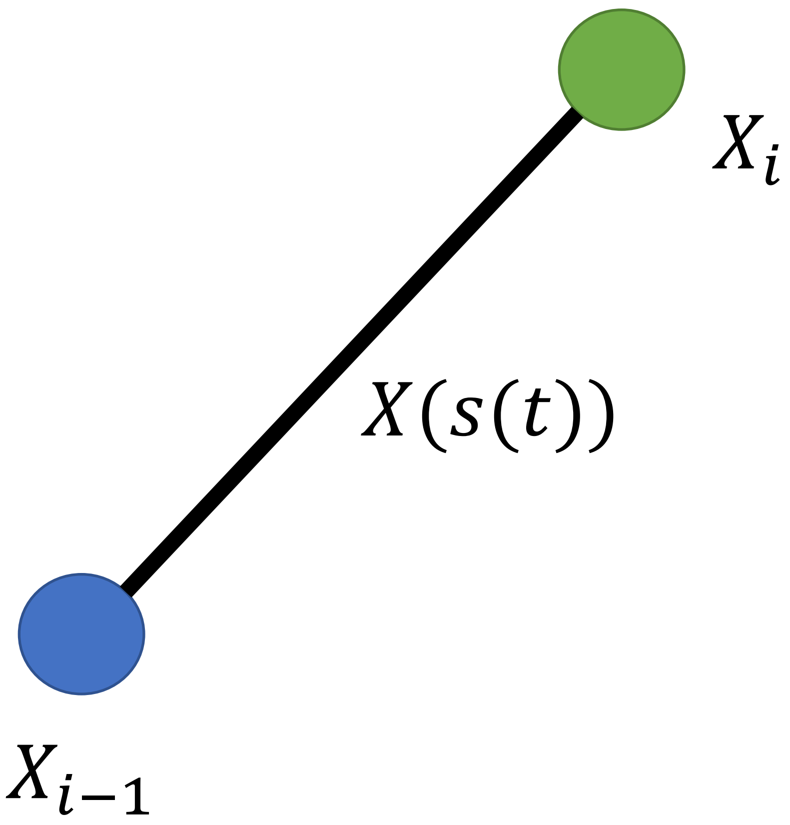
In groups discuss which are the necessary poses in 3D space, to satisfy a **pick and place** operation.

Linear interpolation

- Let's consider two vectors X_{i-1} and X_i. A linear interpolation between them is given as:

$$X(s(t)) = X_{i-1} + (X_i - X_{i-1})s(t) \quad 0 \leq s \leq 1$$

- where $s(t)$ denotes a non decreasing continuous time profile satisfying:
  - $s(t_{i-1}) = 0$ at start,
  - $s(t_i) = 1$ at the end of the movement.



Linear interpolation

- For velocity we will consider the following:

$$
\begin{aligned}
s(t) &= \frac{t - t_{i-1}}{t_i - t_{i-1}} \text{ (constant velocity)} \\
s(t) &= \left(\frac{t - t_{i-1}}{t_i - t_{i-1}} \right)^2 \text{ (linear ramp-up of velocity)} \\
s(t) &= 1 - \left(\frac{t_i - t}{t_i - t_{i-1}} \right)^2 \text{ (linear ramp-down of velocity)}
\end{aligned}
$$

Linear interpolation

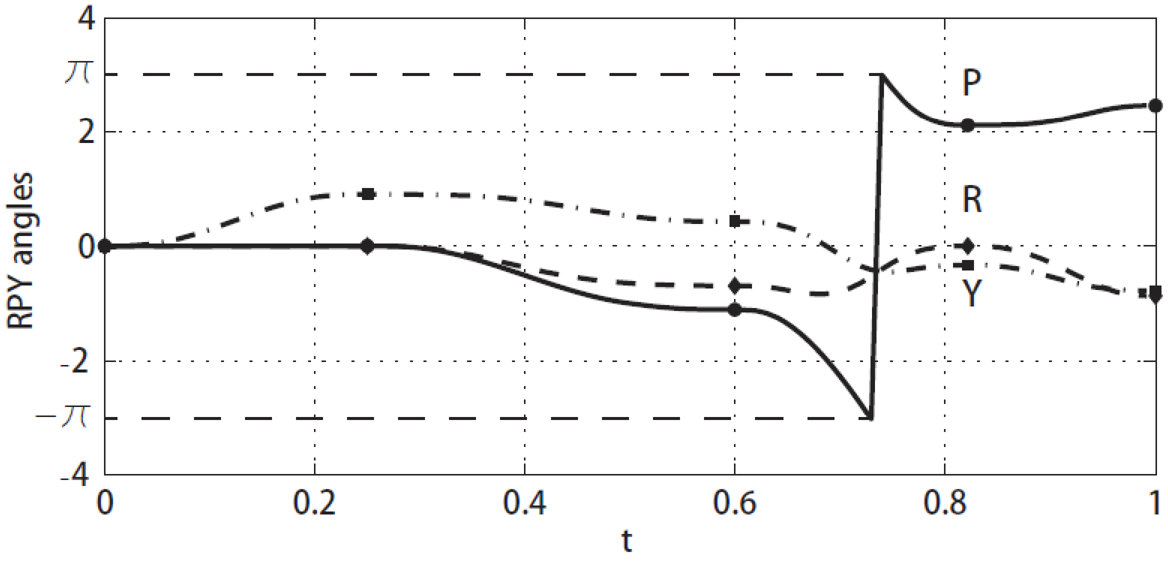
- In tool frame we can write the definition of the interpolated path with constant velocity as such:

$$\begin{aligned}
\mathbf{p}(t) &= \mathbf{p}^{i-1} + \frac{t - t_{i-1}}{t_i - t_{i-1}}(\mathbf{p}^i - \mathbf{p}^{i-1}) \\
\mathbf{R}(t) &= \mathbf{R}_{eaa}\left(\frac{t - t_{i-1}}{t_i - t_{i-1}}\mathbf{W}_{rot}(\mathbf{R}^i[\mathbf{R}^{i-1}]^T)\right)\mathbf{R}^{i-1}
\end{aligned}$$

- where $p(t)$ represents the position part of the tool trajectory and $\mathrm{R}(t)$ the orientation part of the trajectory.

Linear interpolation of rotations

- Interpolation using Euler angles, gives problems:
  - Parametrization singularity, in other words RPY angles defined on the interval $[-\pi, \pi]^3$ encounter an parametrization singularity.
  - **Derivatives** of the Euler angles have no physical meaning!



Linear interpolation of rotations

- A more meaningful approach:
  - Choose physically meaningful coordinates,
  - Add via- points to avoid parametrization singularity,
  - Generate trajectory and use inverse kinematics to obtain joint trajectory and then execute on the robot.
- For orientations specifically use a different representation:
  - equivalent axis angles,
  - rotation matrixes,
  - or quaternions.

Linear interpolation of rotations EEA

- With EEA we can interpolate relative changes in orientation.
- <u>Axis-angle interpolation how-to:</u>
  - Remember Euler's theorem:
    - All orientations **A**, **B** are related by a rotation **R** around an axis **v** by an angle γ: **B** = **AR**.
  - Compute **R** as $\mathbf{A}^{-1}\mathbf{B}$
  - Extract EEA representation $r = \boldsymbol{W}_{rot}(\boldsymbol{R})$ from **R** (see lecture 3)
  - Interpolate linearly between zero and **r** to yield **r**(t), i.e., **r**(t) = t**r**
  - Convert **r**(t) back to matrix **R**(t)
  - Get final orientation by computing **AR**(t)

Linear interpolation of rotations

- We cannot identify orientations/rotations with 3D points and have all of the nice features such as:
  - Nice interpolation
  - Smoothness (no gimbal lock)
  - Simple composition of rotations
  - No complicated multiple-valuedness
  - However we can, interpolate **relative** changes in orientation using the axis-angle representation.

- **A better approximation can be achieved with Quaternions**, a 4D construction that does exactly what we want, both absolutely and relatively.

Quaternion interpolation

Linear interpolation

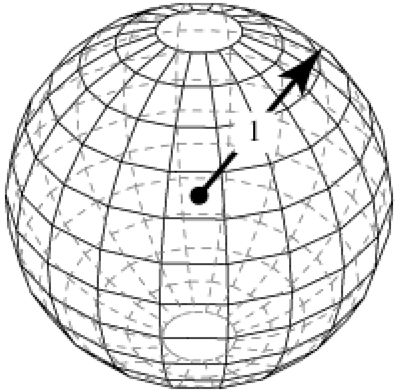
- Interpolation is used to determine intermediate points between two given points $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$
- Linear interpolation can be used to do interpolation in normal space
- The standard linear interpolation formula is

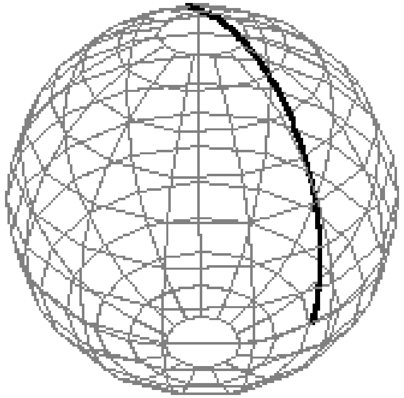
$$\boldsymbol{Q}^i = \boldsymbol{Q}_1 + t(\boldsymbol{Q_2} - \boldsymbol{Q_1})$$

- The general steps to apply this equation are:
  - Compute the difference between $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$
  - Take the fractional part of that difference
  - Adjust the original value by the fractional difference between the two points

Unit sphere

- A quaternion is a point on the 4-D unit sphere
- If we want to interpolate between two points on a sphere, we do not just want to Lerp between them
- Instead, we will travel across the surface of the sphere by following a great arc
- Move with constant angular velocity along the great circle between the two points
- There are two SLERP choices on a sphere – we pick the shortest



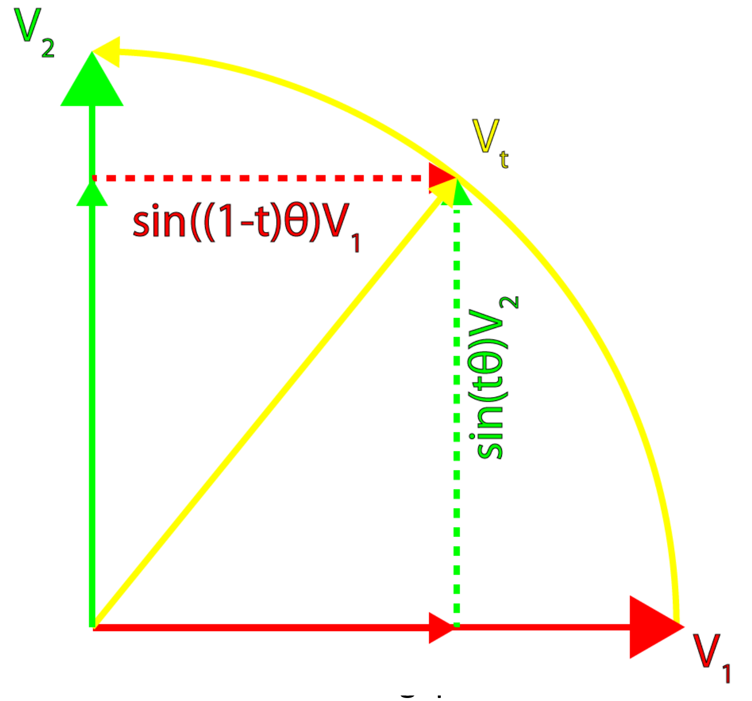


Source: Wolfram Research

Spherical interpolation of vectors

- We will apply a spherical interpolation of vectors to quaternions
- The general form of a spherical interpolation for vectors is defined as:

$$\boldsymbol{v}_t = \frac{\sin(1-t)\gamma}{\sin\gamma}\boldsymbol{v}_2 + \frac{\sin t\gamma}{\sin\gamma}\boldsymbol{v}_2$$



Spherical interpolation of quaternions

- The formula for spherical interpolation of vectors can be applied unmodified to quaternions:

$$Q_t = \frac{\sin(1-t)\gamma}{\sin\gamma} Q_2 + \frac{\sin t\gamma}{\sin\gamma} Q$$

- Angle γ need to be calculated

Spherical interpolation - angle θ

- We can obtain the angle γ by computing the dot-product between q_1 and q_2

$$\cos\gamma = \frac{Q_1 \cdot Q_2}{|Q_1||Q_2|} = \frac{w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2}{|Q_1||Q_2|}$$

$$\gamma = \cos^{-1}\left(\frac{w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2}{|Q_1||Q_2|}\right)$$

Spherical interpolation - considerations

- The quaternion dot-product can results in a negative value
  -> the resulting interpolation will travel the long-way around the 4D sphere
- If the result of the dot product is negative, then we can negate one of the orientations
- Negating the scalar and the vector part of the quaternion does not change the orientation that it represents

Spherical interpolation - considerations

- If the angular difference γ between Q_1 and Q_2 is very small then $\sin\gamma$ becomes 0 -> undefined result when we divide by $\sin\gamma$.
- In this case, we can fall-back to using linear interpolation between Q_1 and Q_2.
- As changes are small a straight line is close to the arc on the sphere

Quaternion difference

- quaternion logarithm which maps $S^3 \rightarrow \mathbb{R}^3$ is defined as:

$$\mathbf{r} = \log(\mathrm{Q}) = \log(v + \boldsymbol{u}) = \begin{cases} \arccos(v)\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, \boldsymbol{u} \neq 0 \\ [0,0,0]^{\mathrm{T}},\ \text{otherwise} \end{cases}$$

- which can be interpreted as a difference vector $\log(\boldsymbol{Q}_2 * \overline{\boldsymbol{Q}_1})$ between 2 unit quaternions.
- its inverse, the exponential map, which maps from $\mathbb{R}^3 \rightarrow S^3$ is defined:

$$\exp(\boldsymbol{r}) = \begin{cases} \cos(\| \boldsymbol{r} \|) + \sin(\| \boldsymbol{r} \|)\frac{\boldsymbol{r}}{\|\boldsymbol{r}\|}, \boldsymbol{r} \neq 0 \\ 0,\ \text{otherwise} \end{cases}$$

Exercise

Python example 1

Calculate a slerp (in python use the quaternion interp. function) curve in quaternion space between:

Q_1 =[0.7071, 0.7071, 0.0000, 0.0000]

Q_2 = [0.5000, -0.0000, 0.8660, -0.0000]

Interpolation coefficient = 100.

plot the results as interpolated curves on a sphere with the provided function.

Linear interpolation

Linear interpolation

- We can also define a linear interpolator in joint space:

$$q^i(t) = q_o^i + \frac{t - t_0}{t_f - t_0}\left(q_f^i - q_o^i\right), i = 1, \dots, \mathrm{numDOF}$$

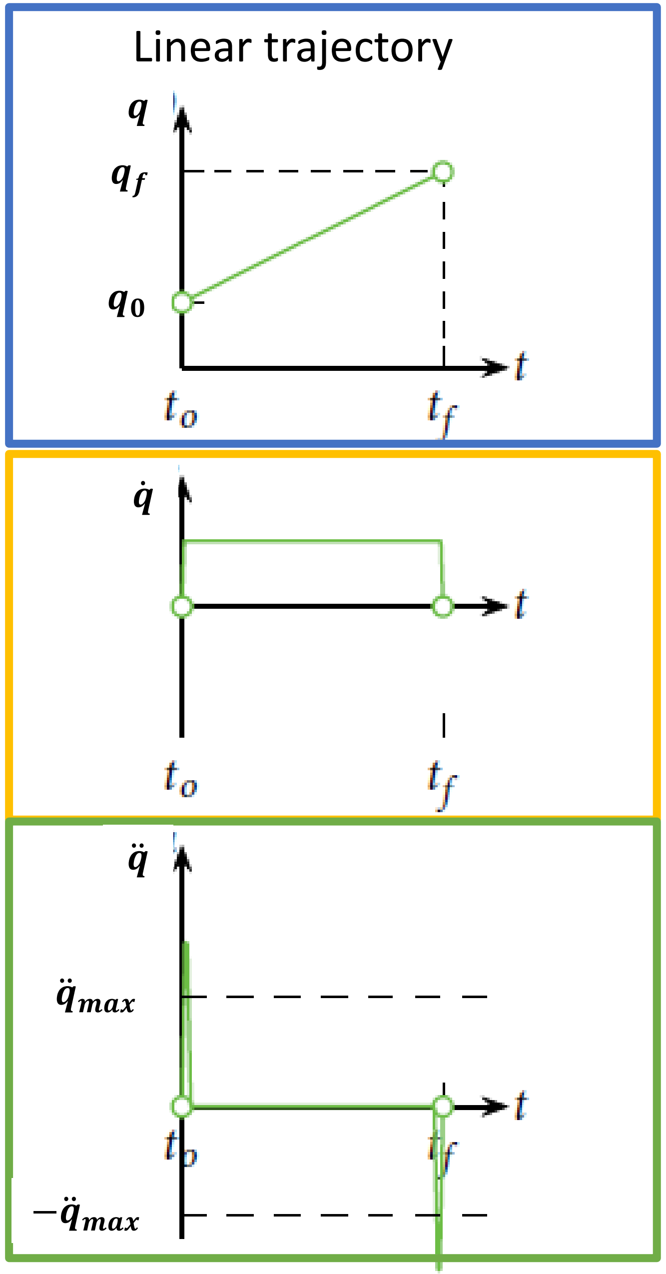
- It is given by a first order polynomial, for better results higher order polynomials work better:
  - the problem of velocity – discontinuity and unbounded acceleration in both ends.
- the outcome, compared with tool space interpolation is not equal!!!!!
- therefore, we have to carefully consider which type of interpolation is appropriate for the desired task/application.

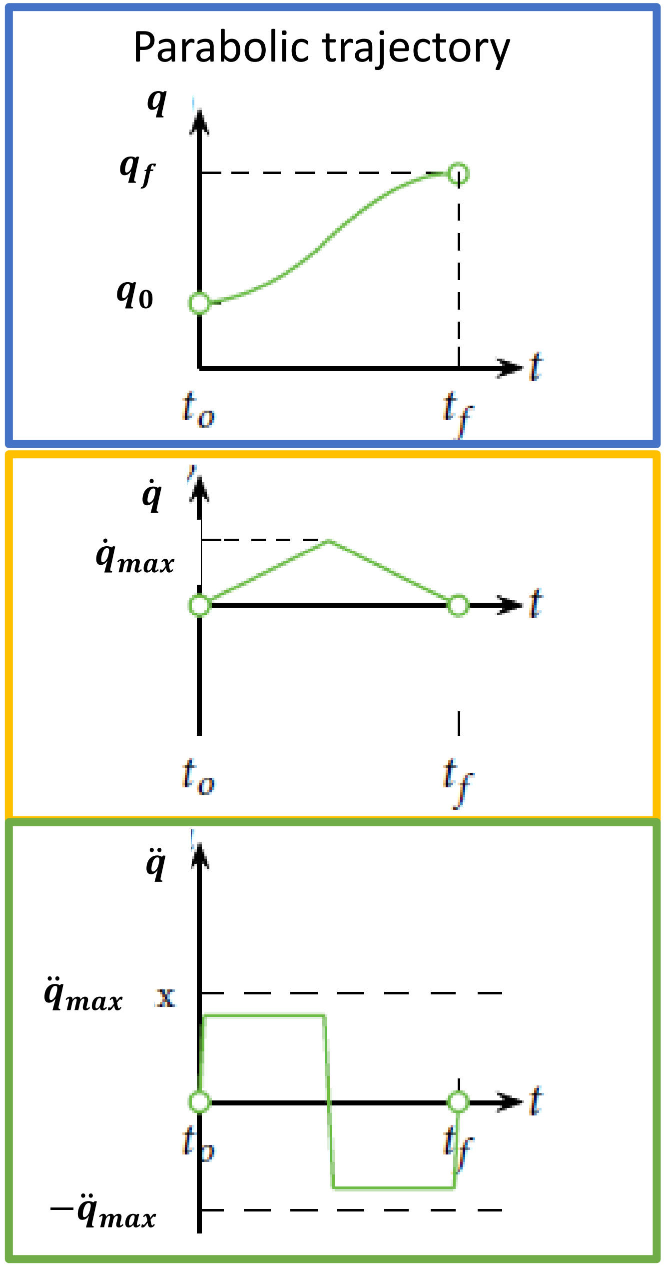
Trajectories

POSITION

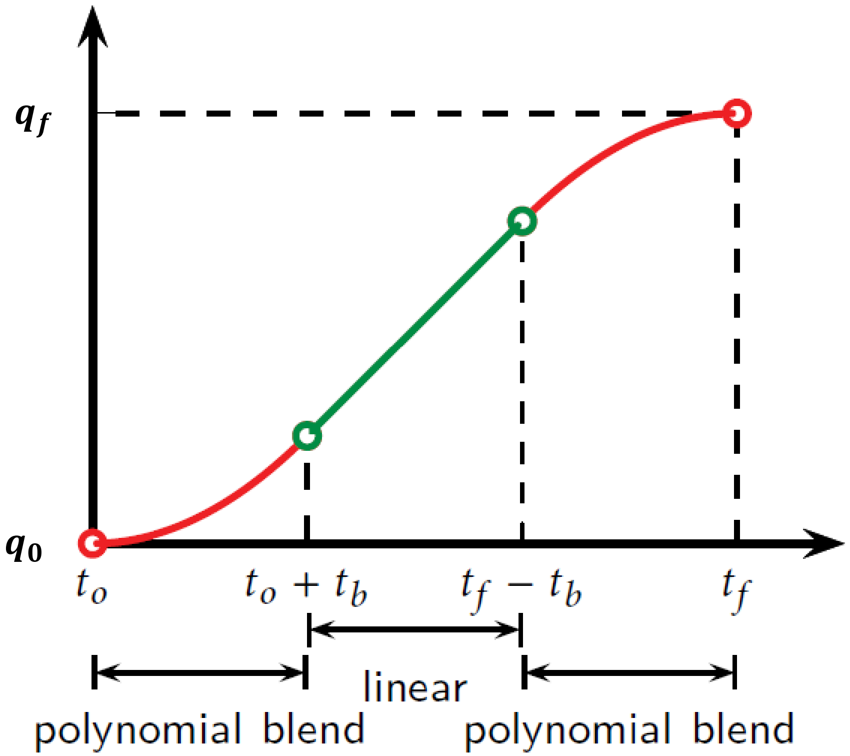
VELOCITY

ACCELERATION

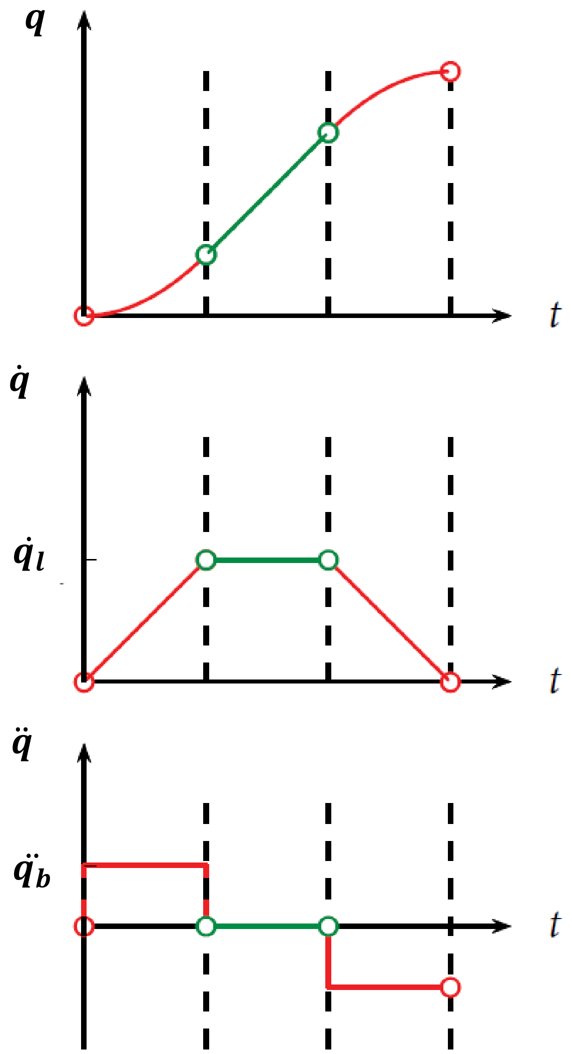




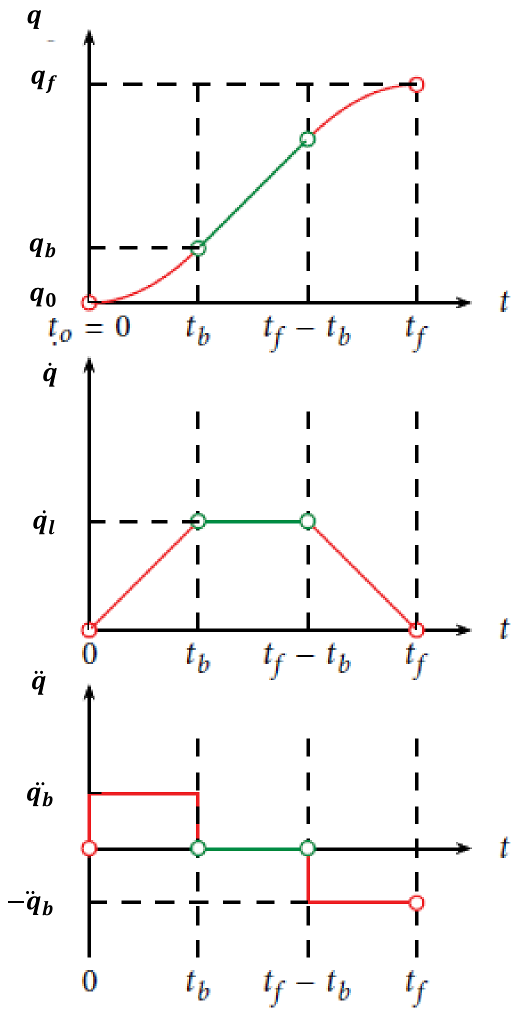
Let's first look at joint trajectories



- Parabolic blend
  - composite trajectory of a linear and a parabolic function.



Linear function with parabolic blend



- Input parameters:

$$\mathrm{q}(t_o) = q(0) = q_o$$
$$q(t_f) = q_f$$
$$t_d = t_f - t_o\text{: Duration of travel}$$

- Control parameters:

$\dot{q}_l(\leq \dot{q}_{\max})$: linear velocity

$q_b(\leq \ddot{q}_{\max})$: blend acceleration

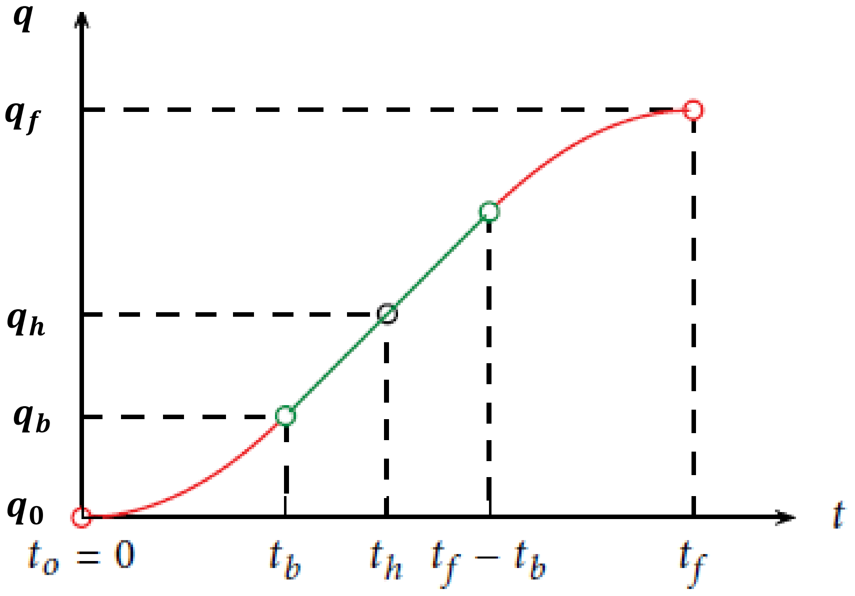
$t_b\left(0 < t_b \leq \dfrac{t_d}{2}\right)$: blend time

$t_d - 2t_b$: linear time

- Trajectory

$$\mathrm{q}(t) = \begin{cases} q_o + \frac{1}{2}\ddot{q}_b(t - t_o)^2 & t_o \leq t < t_o + t_b \\ q_o + \ddot{q}_b t_b\left(t - t_o - \frac{t_b}{2}\right) & t_o + t_b \leq t < t_f - t_b \\ q_f - \frac{1}{2}\ddot{q}_b\left(t_f - t\right)^2 & t_f - t_b \leq t \leq t_f \end{cases}$$

Derivation:



- Velocity match condition

$$\ddot{q}_b t_b = \frac{q_h - q_b}{t_h - t_b}, \left(t_h \triangleq \frac{t_d}{2}, q_h \triangleq q(t_h) \right)$$

- Blend region

$$q_b \triangleq q(t_b) = q_o + \frac{1}{2}\ddot{q}_b t_b^2$$

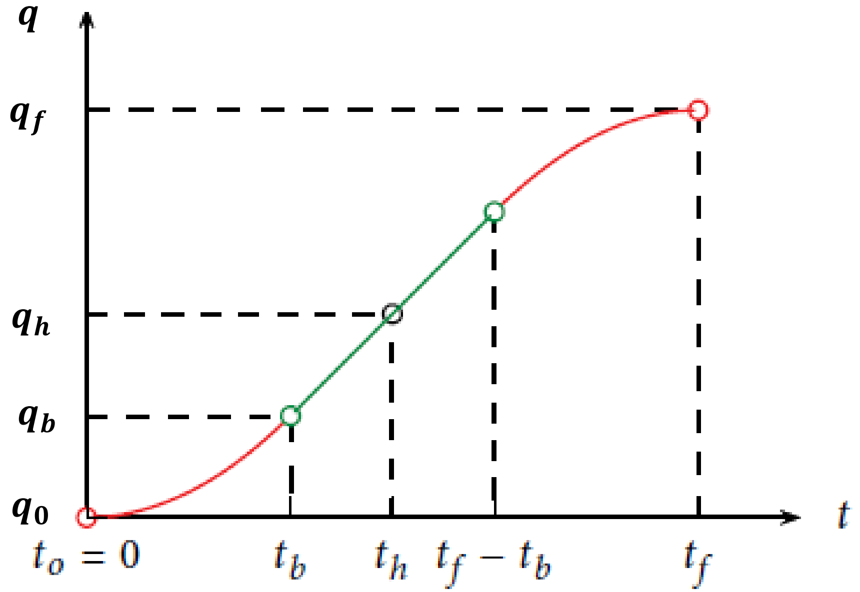
$$\Rightarrow \ddot{q}_b t_b^2 - \ddot{q}_b t_d t_b + \left(q_f - q_o\right) = 0$$

$$\Rightarrow t_b = \frac{t_d}{2} - \frac{\sqrt{\ddot{q}_b^2 t_d^2 - 4\ddot{q}_b\left(q_f - q_0\right)}}{2\ddot{q}_b}$$

- Constraints on $\ddot{q}_b$:

$$\ddot{q}_b \geq \frac{4\left(q_f - q_o\right)}{t_d^2}$$

Derivation:

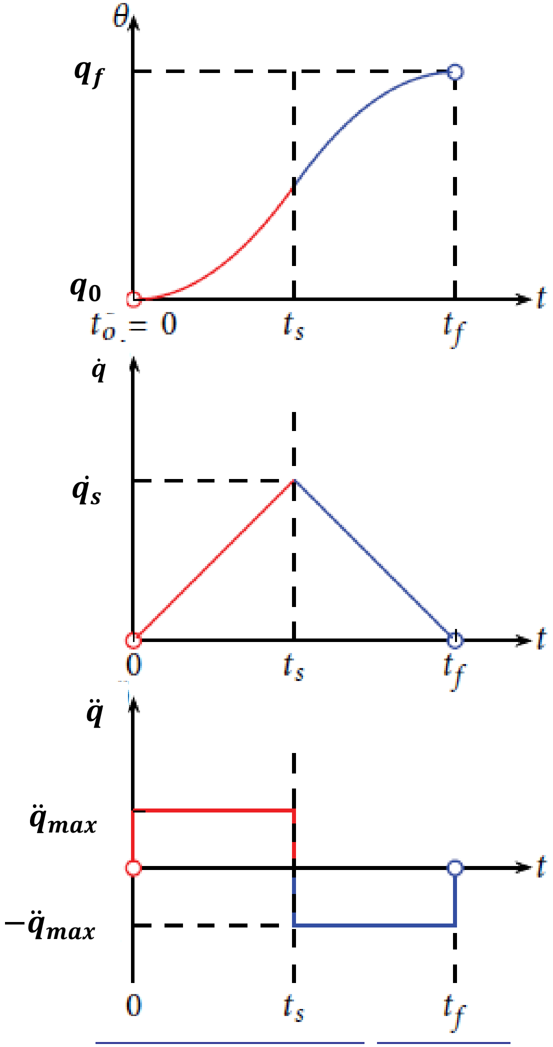


Observation:

$$\ddot{q}_b \geq \frac{4(q_f - q_o)}{t_d^2}$$

- As $\ddot{q}_b$ increases, t_b decreases and linear time $t_b \ - 2t_b$ increases, $\ddot{q_b} \rightarrow \infty$
- if the above equation expresses equality the, the linear part of the parabolic blend goes to zero.

Minimum time trajectory



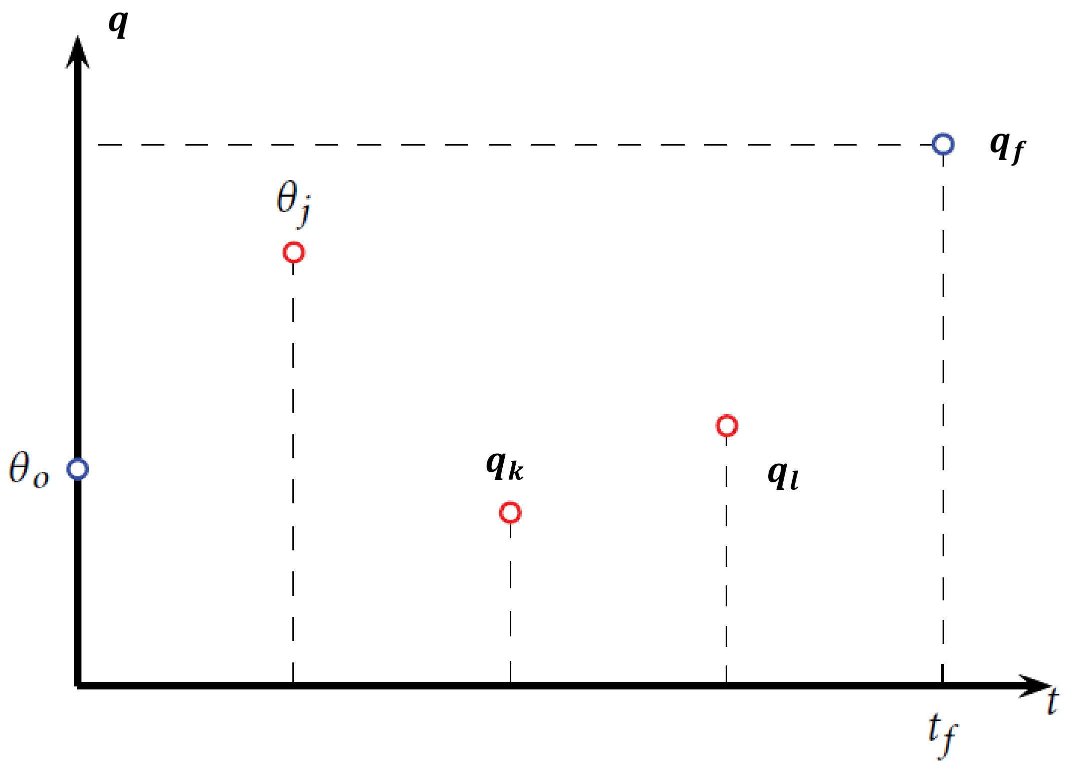
- Given: q_0, q_f and $\ddot{q}_{max}$
- objective to minimize: t_d
- The outcome is a solution without the linear part e.g. ***Bang-bang trajectory***

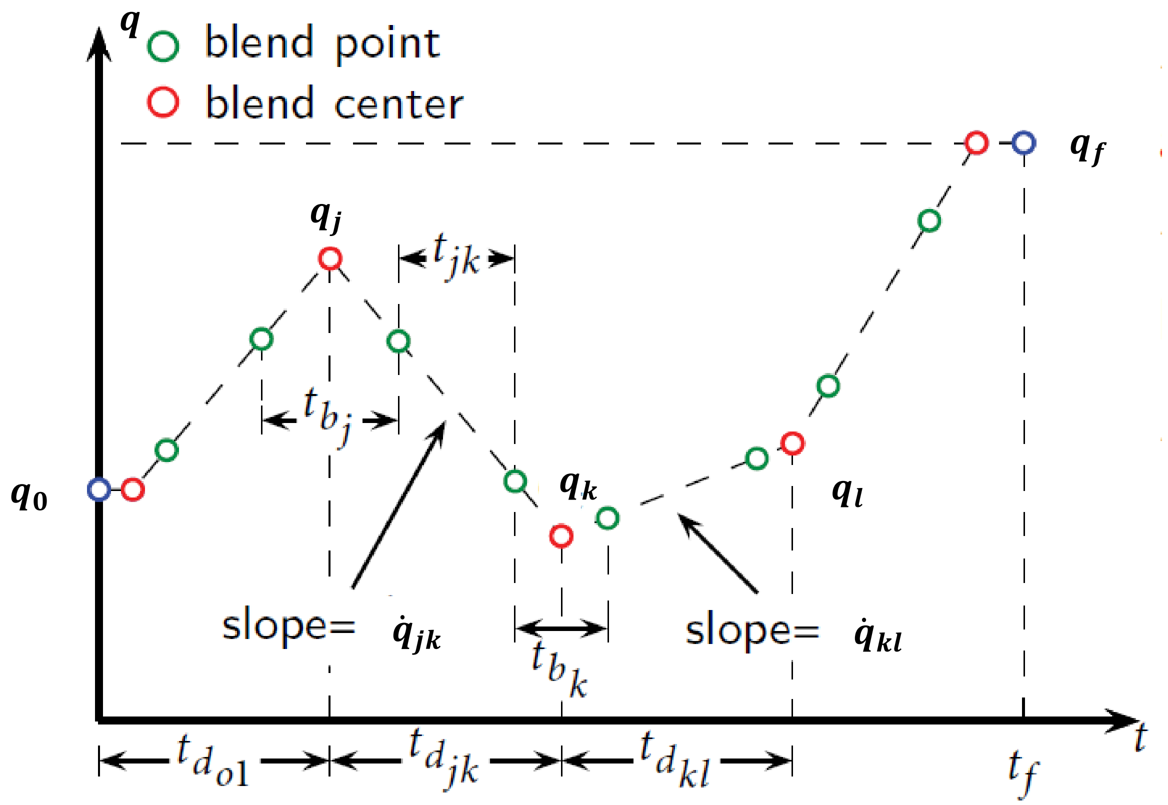
$$\ddot{q}(t) = \begin{cases} \ddot{q}_{max} & 0 \le t \le t_s \\ -\ddot{q}_{max} & t_s \le t \le t_d \end{cases}$$

- where the switching time t_s can be calculated by

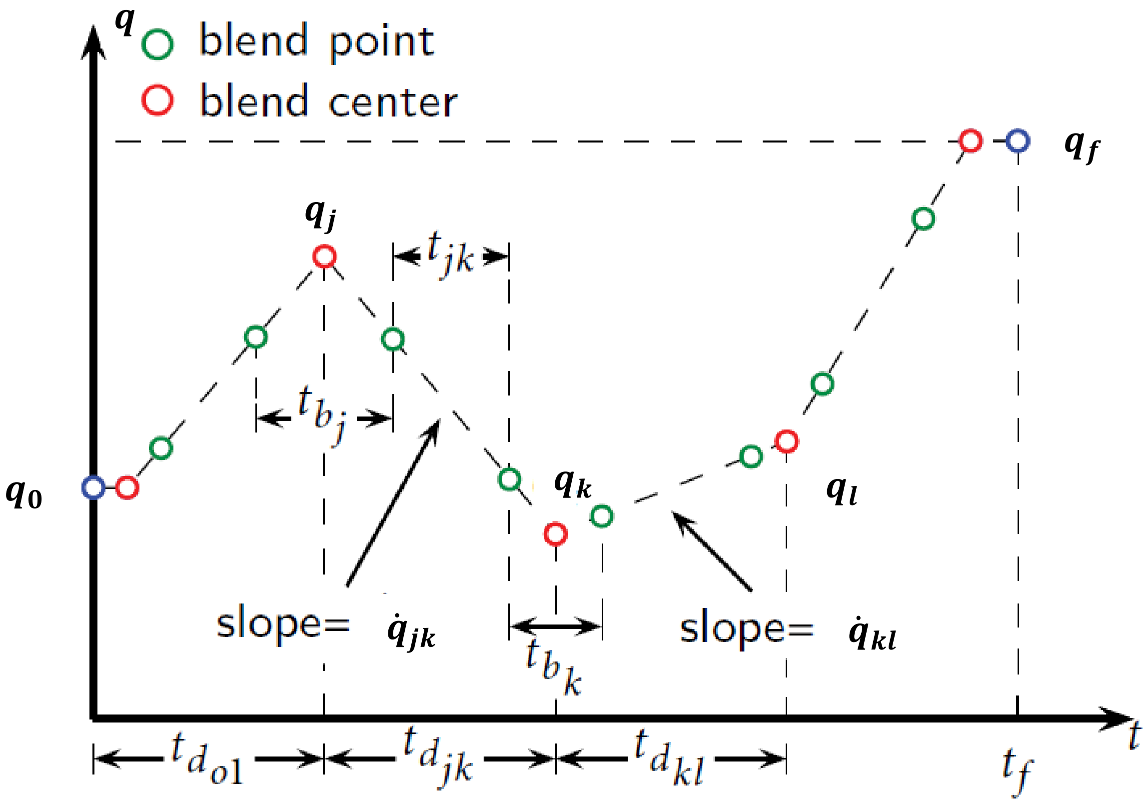
$$t_s = \frac{t_d}{2} = \sqrt{\frac{q_f - q_0}{\ddot{q}_{max}}}$$

Parabolic blend for a path with via points



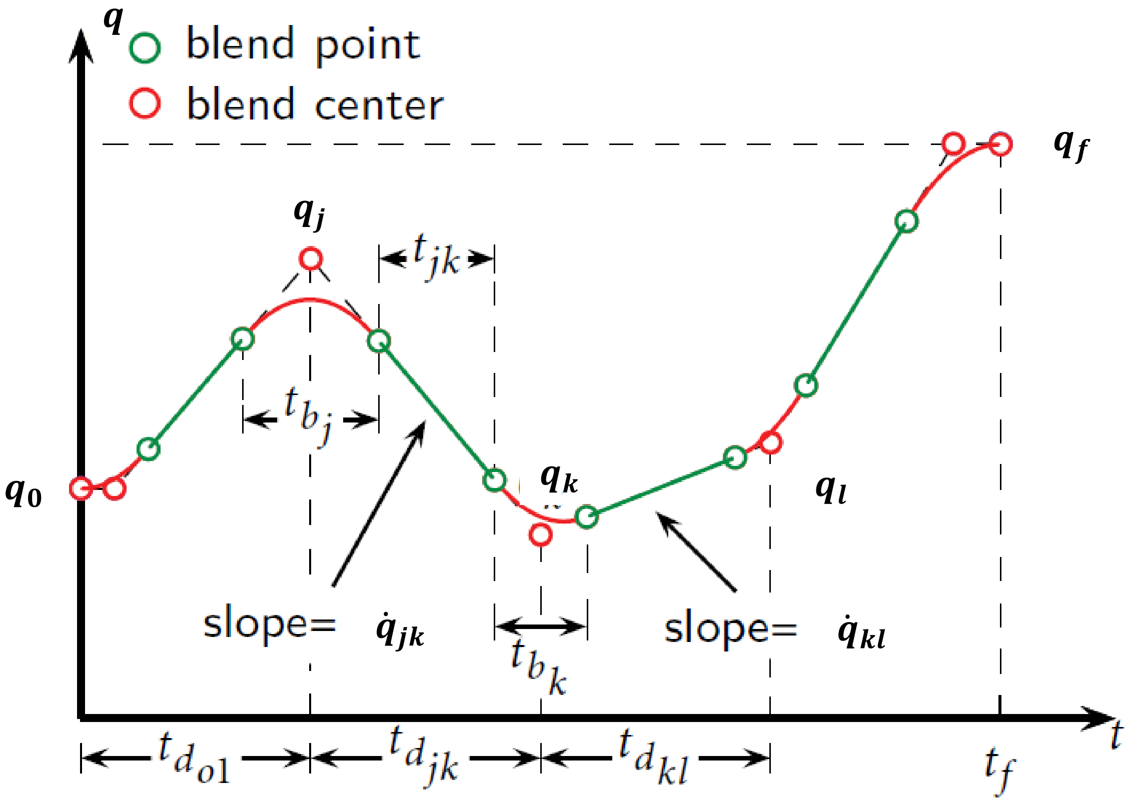


Parabolic blend for a path with via points



- *Given:* t_0 , q_o , t_f , q_f , $\ddot{q}_b$, via points $\{q_i\}_1^m$ at time $\{t_i\}_1^m$ (time duration $t_{d_{jk}} \triangleq t_k - t_j$).
- *Find:* $q(t)$ interpolating q_o , q_f and approximating $\{q_i\}_1^m$.
- *Solution: Linear Parabolic Blend* with via points

Parabolic blend for a path with via points



- <u>*Solution:*</u> LPB with via points

for via points $j, k, l = 1, \dots, m$:

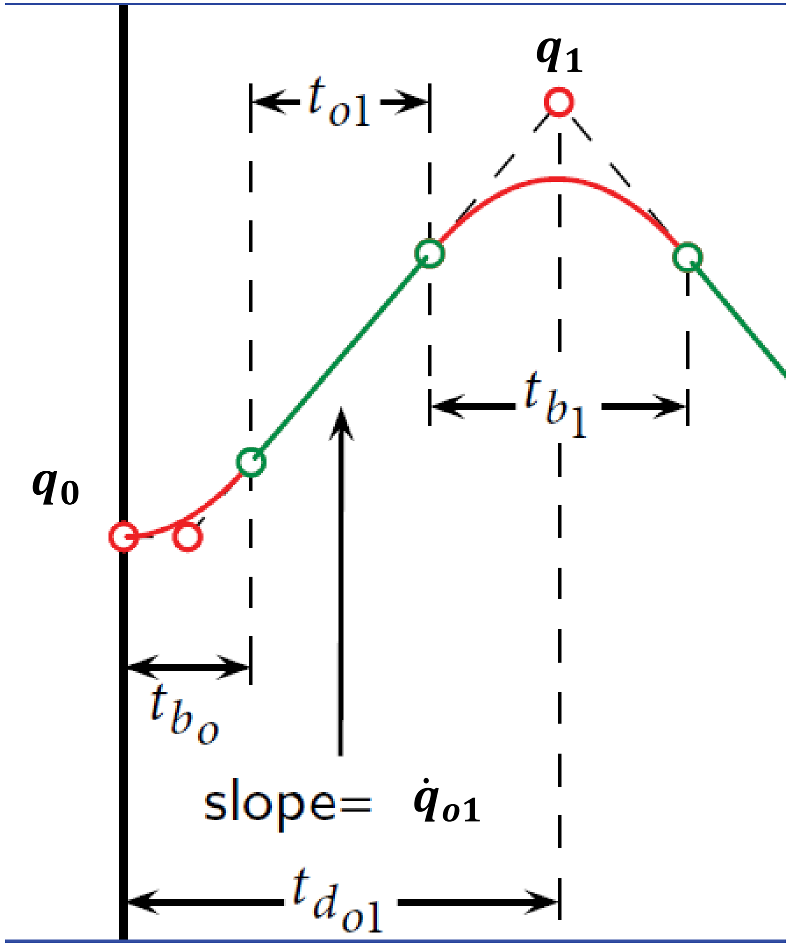
$\dot{q}_{jk} = \frac{q_k - q_j}{t_{d_{jk}}}$ (lin. vel),

$\ddot{q}_k = \mathrm{Sgn}\left(\dot{q}_{kl} - \dot{q}_{jk}\right)|\ddot{q}_b|$ (blend acc.),

$t_{b_k} = \frac{\dot{q}_{kl} - \dot{q}_{jk}}{\ddot{q}_k}$ (blend dur.)

$t_{jk} = t_{d_{jk}} - \frac{1}{2} t_{b_j} - \frac{1}{2} t_{b_k}$ (lin. dur.)

Let's divide it into two segments



- First segment:

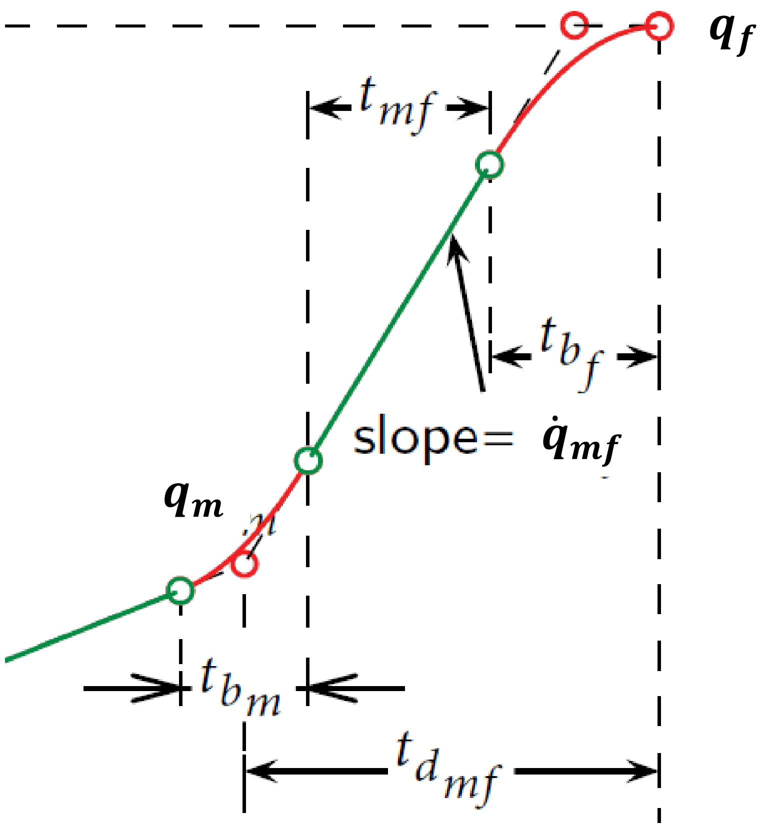
$\ddot{q}_o = \mathrm{Sgn}(q_1 - q_o)|\ddot{q}_b|$ (Blend acc.)

$t_{b_o} = t_{d_{o1}} - \sqrt{t_{d_{o1}}^2 - \frac{2(q_1 - q_o)}{\ddot{q}_o}}$ (Blend dur.)

$\dot{q}_{o1} = \frac{q_1 - q_o}{t_{d_{o1}} - \frac{1}{2} t_{b_o}}$ (lin. vel.)

$t_{o1} = t_{d_{o1}} - t_{b_o} - \frac{1}{2} t_{b_1}$ (lin. dur.)

Let's divide it into two segments



- Last segment:

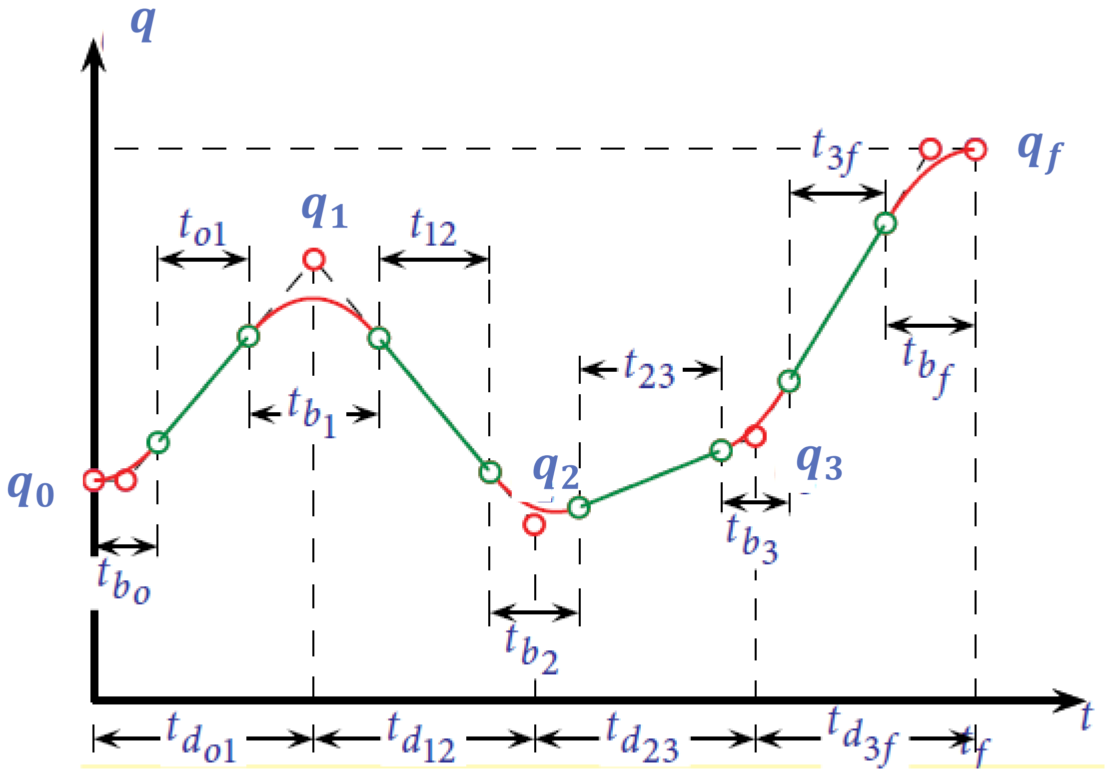
$\ddot{q}_f = \mathrm{Sgn}(q_\mathrm{m} - q_f)|\ddot{q}_b|$ (Blend acc.)

$t_{b_f} = t_{d_{mf}} - \sqrt{t_{d_{mf}}^2 - \frac{2(q_\mathrm{f} - q_m)}{\ddot{q}_f}}$ (Blend dur.)

$\dot{q}_{mf} = \frac{q_\mathrm{f} - q_m}{t_{d_{mf}} - \frac{1}{2} t_{b_f}}$ (lin. vel.)

$t_{mf} = t_{d_{mf}} - t_{b_f} - \frac{1}{2} t_{b_\mathrm{m}}$ (lin. dur.)

Example with 3 via points



- Given:

$q_o = 1 \; q_f = 2.5 \; \ddot{q}_b = 4 \; q_1 = 2 \; q_2 = 0.8 \; q_3 = 1.2$

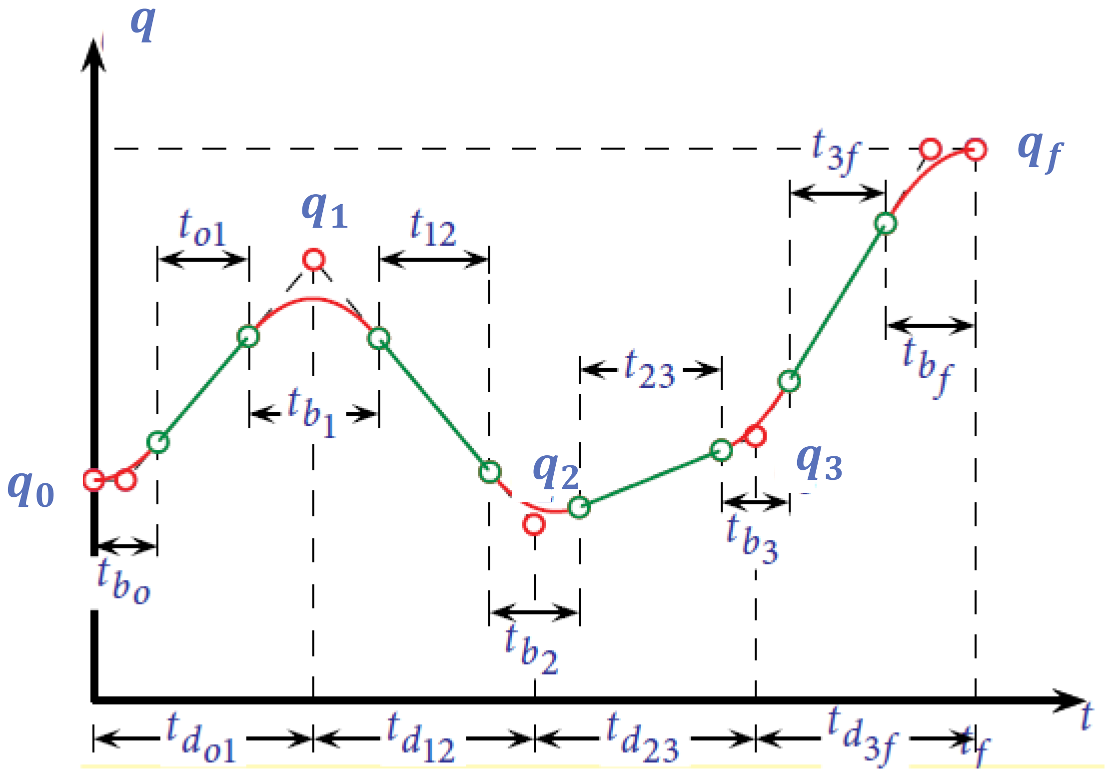
we calculate using the equations from the previous slide:

$$\ddot{q}_0 = 4,$$

$$t_{b_o} = 1 - \sqrt{1^2 - \frac{2(2-1)}{4}} = 0.29$$

$$\dot{q}_{o1} = \frac{2-1}{1 - \frac{1}{2}0.29} = 1.17$$

Example with 3 via points



we calculate:

$\dot{q}_{12} = \frac{0.8-2}{1} = -1.2,$

$\ddot{q}_1 = -4,$

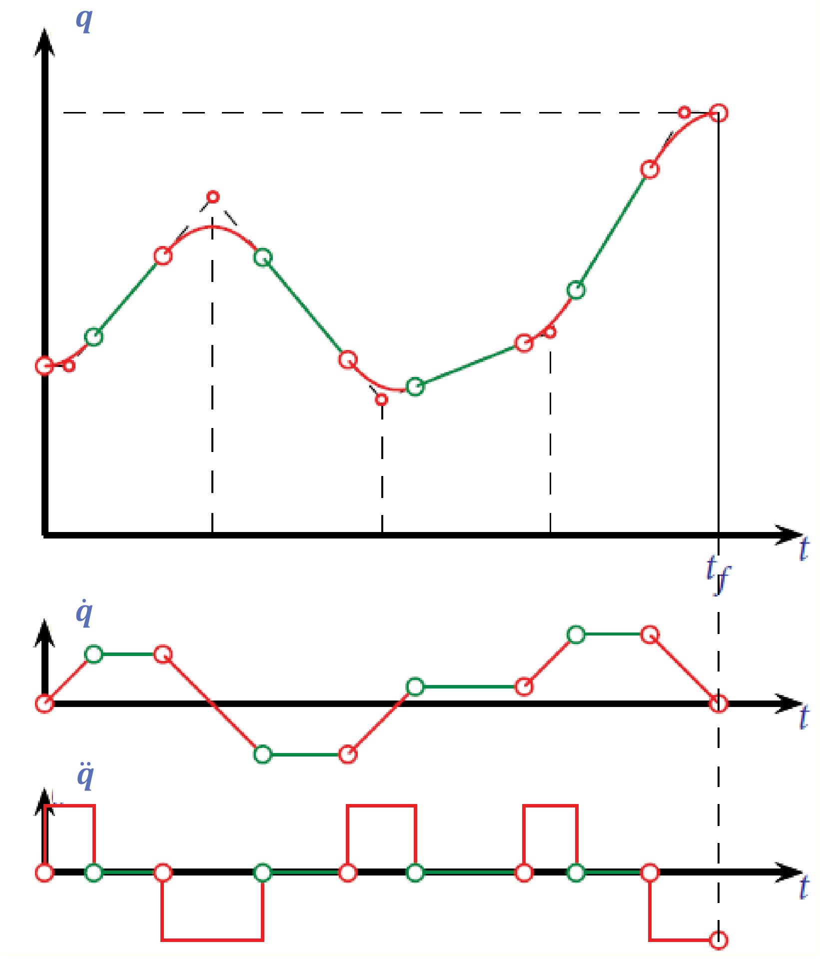
$t_{b_1} = \frac{-1.2-1.17}{-4} = 0.59, t_{01} = 1 - 0.29 - \frac{1}{2}0.59 = 0.41$

$\dot{q}_{23} = \frac{1.2-0.8}{1} = 0.4,$

$\ddot{q}_2 = 4, t_{b_2} = \frac{0.4+1.2}{4} = 0.4,$

$t_{12} = 1 - \frac{1}{2}0.59 - \frac{1}{2}0.4 = 0.51$

Example with 3 via points



we calculate:

$\ddot{q}_f = -4,$

$t_{b_f} = 1 - \sqrt{1^2 + \frac{2(2.5-1.2)}{-4}} = 0.41$

$\ddot{q}_3 = 4,$

$t_{b_3} = \frac{1.63-0.4}{4} = 0.31$

$t_{3f} = 1 - \frac{1}{2}0.31 - 0.41 = 0.44$

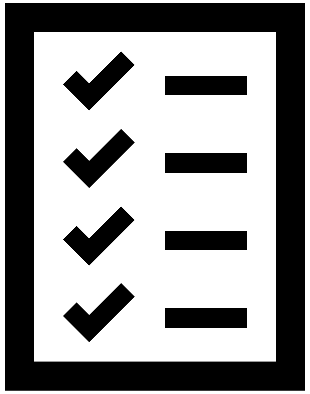
$t_{23} = 1 - \frac{1}{2}0.4 - \frac{1}{2}0.31 = 0.65$

Parabolic blend in Task space

- Parabolic blend in task space, can for the positional part be calculated in the same way as for joint space parabolic blends.
- Orientation can be computed with piece vise computing slerp trajectories in quaternion space (more info *Dantam N, Stilman M. Spherical parabolic blends for robot workspace trajectories. In2014 IEEE/RSJ International Conference on Intelligent Robots and Systems 2014 Sep 14 (pp. 3624-3629)*).
- Followed by solving the inverse kinematics for the generated task space path.

Example

Python example



Construct a linear path using python functions (**<u>ctraj</u>**) between the initial joint configuration of the UR robot (self.q0), and an offset tool space pose given as:

- sm.SE3.Trans(0.2, 0.2, 0.35)
- with 100 samples
- extend the generated trajectory with another linear interpolation starting at the current pose and ending at:
- sm.SE3.Trans(0,0,-0.4)
- with 50 samples.
- Calculate a joint interpolated trajectory (**jtraj**) starting at the current configuration and ending at a user defined configuration (using sliders)
- with 20 samples.

1. Plot the position part and the orientation part of the trajectoire.
2. Plot the velocities and accelerations.