Trajectory representation as polynomials and splines

Robotics and Computer vision (RoVi)

Aljaz Kramberger

alk@mmmi.sdu.dk

SDU Robotics

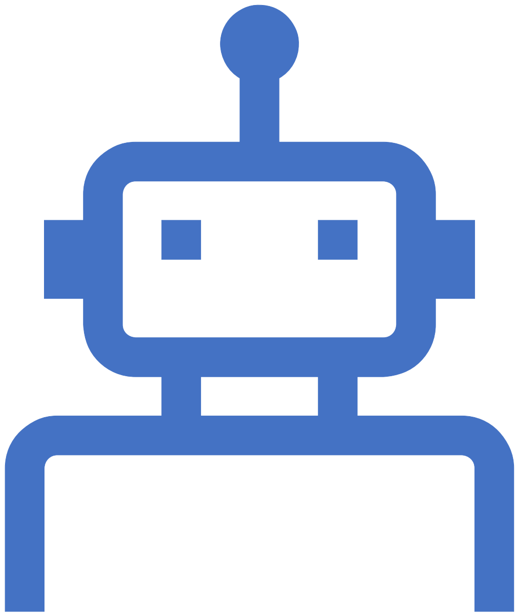
The Maersk Mc-Kinney Moller Institute

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, CRC Press, 2011



Agenda

- Limits on joint positions, velocities and accelerations,
- Cubic polynomial interpolation and splines
- Dynamic systems for trajectory planning

To start with…

and the score

Viewing Division 735: NIST ...

Results – Manufacturing Track

Team	Subtask	Threading Fasteners	Insertions	Wire Routing	Belt Drive	Time Bonus	Subtotal	Total
SDU Robotics	Disassembly	54	27	8	3	18	110	427
	Assembly	72	54	40	43	108	317	
CASIA&Wenjing College Yantai University	Disassembly	0	24	0	0	0	24	109
	Assembly	0	27	30	28	0	85	
New Dexterity Research Group	Disassembly	12	27	2	3	0	44	44
	Assembly	0	0	0	0	0	0	
Robotic Materials	Disassembly	0	15	0	0	0	15	27
	Assembly	0	12	0	0	0	12	
JAKS	Disassembly	0	12	0	0	0	12	18
	Assembly	0	6	0	0	0	6	
Munich School of Robotics and Machine	Disassembly	Did not compete						
	Assembly							

Handling limits on joint positions, velocities and accelerations

Limits on joint positions and velocities

- we consider the problem:

$$J(q)\Delta \mathrm{q} = \Delta \mathrm{u}$$

- The positional joint limits can be written as:

$$q_{min} \leq q + \Delta q \leq q_{max}$$

- Velocities can be treated similarly:

$$\dot{q}_{\mathrm{min}}\Delta t_i \leq \Delta \mathrm{q} \leq \dot{q}_{\mathrm{max}}\Delta t_i \qquad \Delta t_i = t_{i+1} - t_i$$

$$\dot{q}_{\mathrm{min}}\Delta t_i + q^i \leq q + \Delta q \leq \dot{q}_{\mathrm{max}}\Delta t_i + q^i$$

Limits on accelerations

$$G_{\min}\left(q^i, q^{i-1}, t_i\right) = \quad \max\left\{q_{\min}, \dot{q}_{\min}\Delta t_i + q^i, \frac{1}{2}\ddot{q}_{min}\Delta t_i[\Delta t_{i-1} + \Delta t_i] + q^i + \frac{\Delta t_i}{\Delta t_{i-1}}\left(q^i - q^{i-1}\right)\right\}$$

$$\mathrm{G}_{\max}\left(q^i, q^{i-1}, t_i\right) = \quad \min\left\{q_{\max}, \dot{q}_{\max}\Delta t_i + q^i, \frac{1}{2}\ddot{q}_{max}\Delta t_i[\Delta t_{i-1} + \Delta t_i] + q^i + \frac{\Delta t_i}{\Delta t_{i-1}}\left(q^i - q^{i-1}\right)\right.$$

- we can finally write the problem as:

$$min \parallel J(\mathrm{q})\Delta q - \Delta u \parallel$$

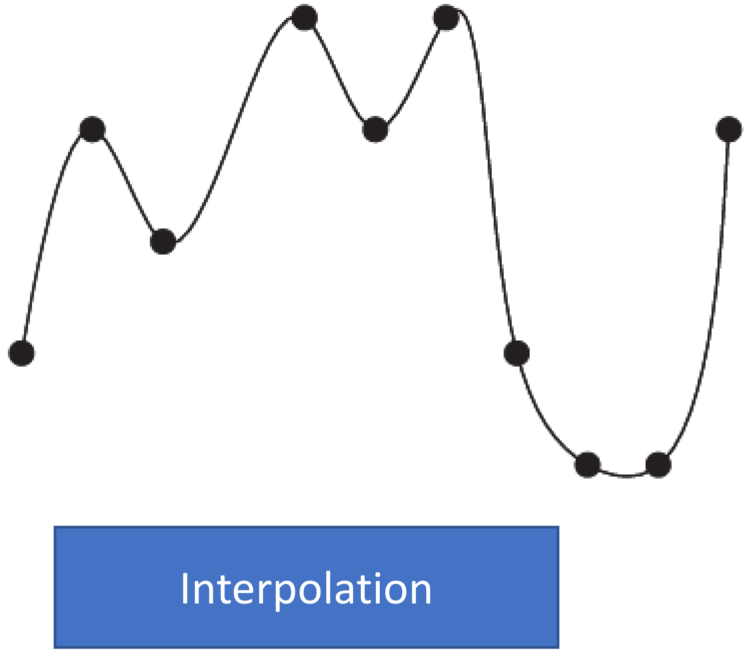
$$G_{min}\left(q^i, q^{i-1}, t_i\right) \leq \mathrm{q} + \Delta\mathrm{q} \leq G_{max}\left(q^i, q^{i-1}, t_i\right)$$

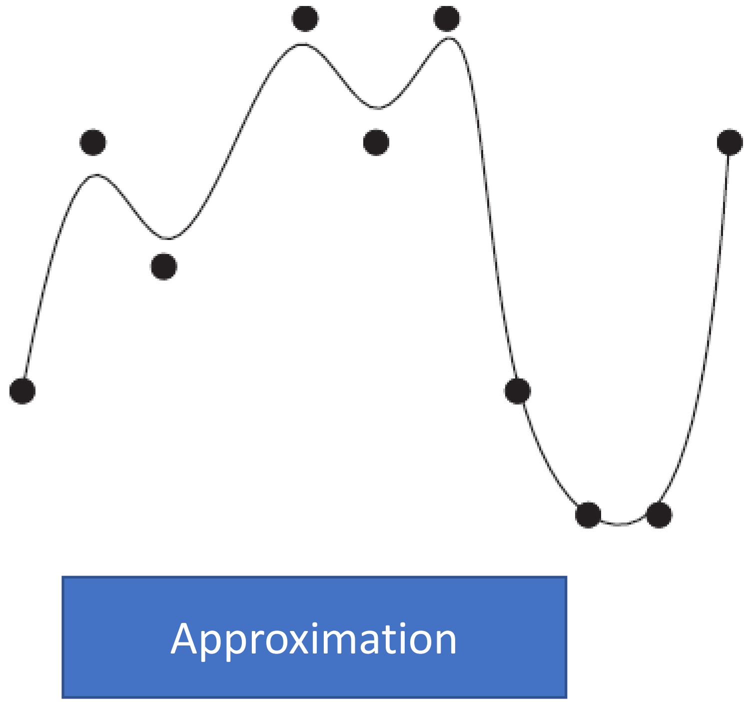
Pros and cons of this approach

- Advantages:
  - Suitable for time-critical trajectories (e.g. visual servoing),
  - good handling of positional limits,
  - always computes the feasible solution closest to the desired tool movement.
- Disadvantages:
  - a bit complicated to program (Quadratic Programming Problem - QPP),
  - tool path may deviate in space.

From Points, via- points to trajectories

From Points, via- points to trajectories





- ***Definition: Interpolation*** Constructing new data points within the range of a discrete set of known data points (exact fitting).
- ***Definition: Approximation*** Inexact fitting of a discrete set of known data points.

Interpolation between points

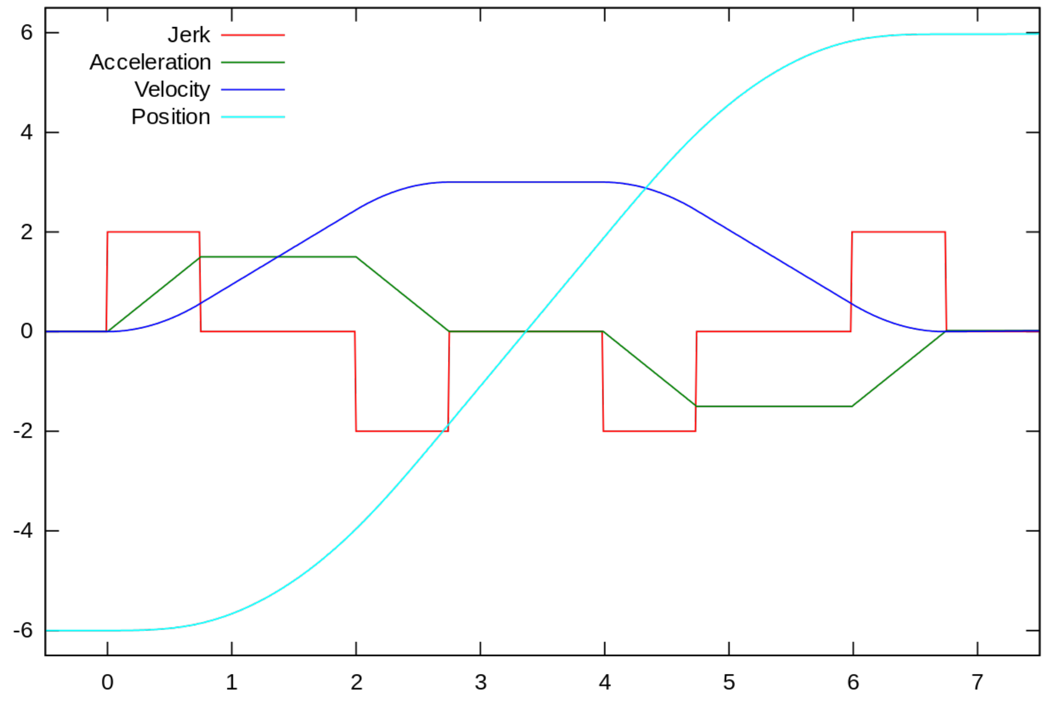
- In the last lecture we introduced:
  - linear interpolation between points,
  - linear interpolation with parabolic blends,
- In comparison, velocities and accelerations are better approximated with the parabolic blends.
- Still this method does not give a completely smooth trajectory!
- For this we must look at the derivative of the acceleration at blend points.

Definition of jerk for motion planning

- Jerk in motion control is the rate at which an object's acceleration changes with respect to time.
- It is a vector quantity (having both magnitude and direction). and expressed in m/s^3(SI units).
- Jerk can be calculated as:

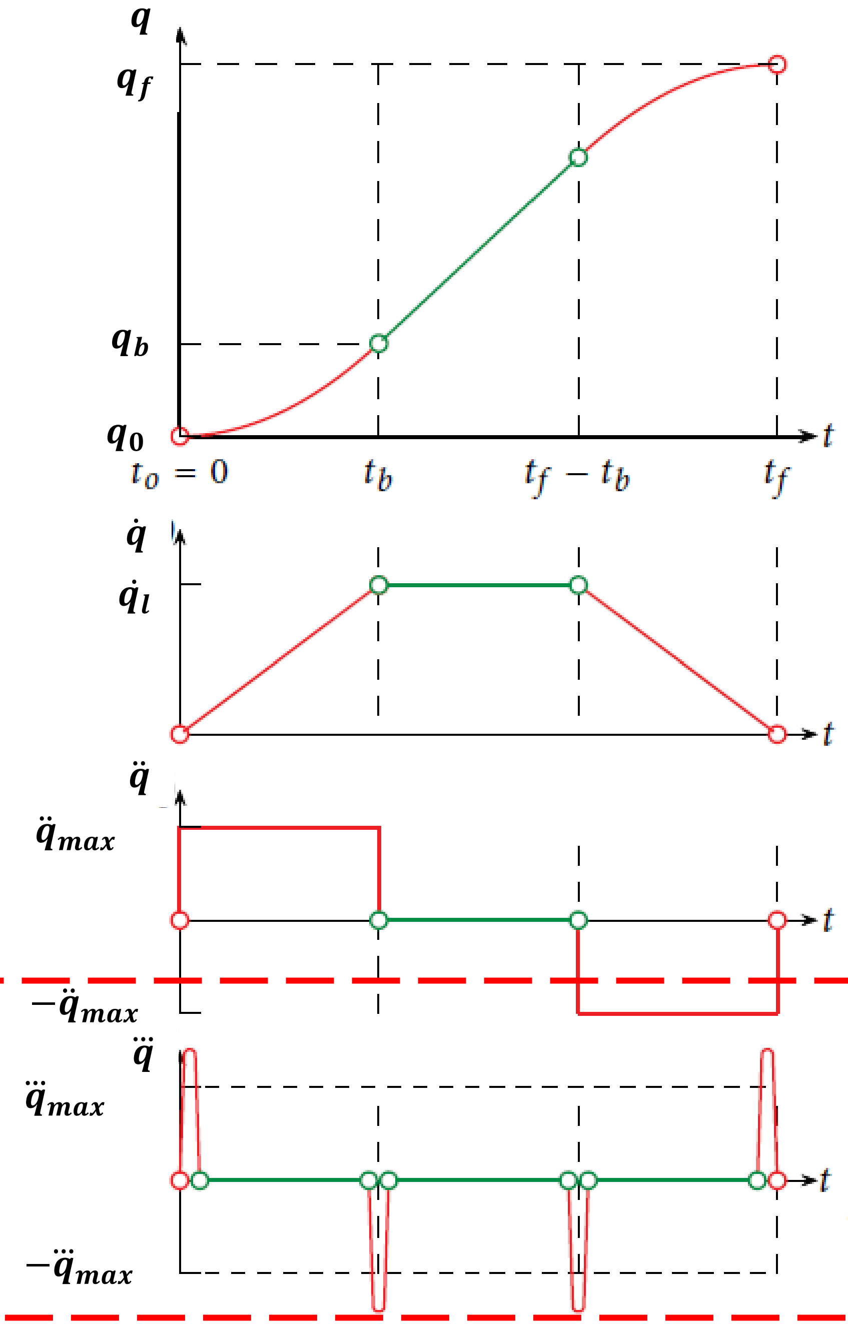
$$\vec{j} = \frac{da(t)}{dt} = \frac{d^2v(t)}{dt} = \frac{d^3q(t)}{dt}$$

- jerk can be calculated for joint and task space trajectories.



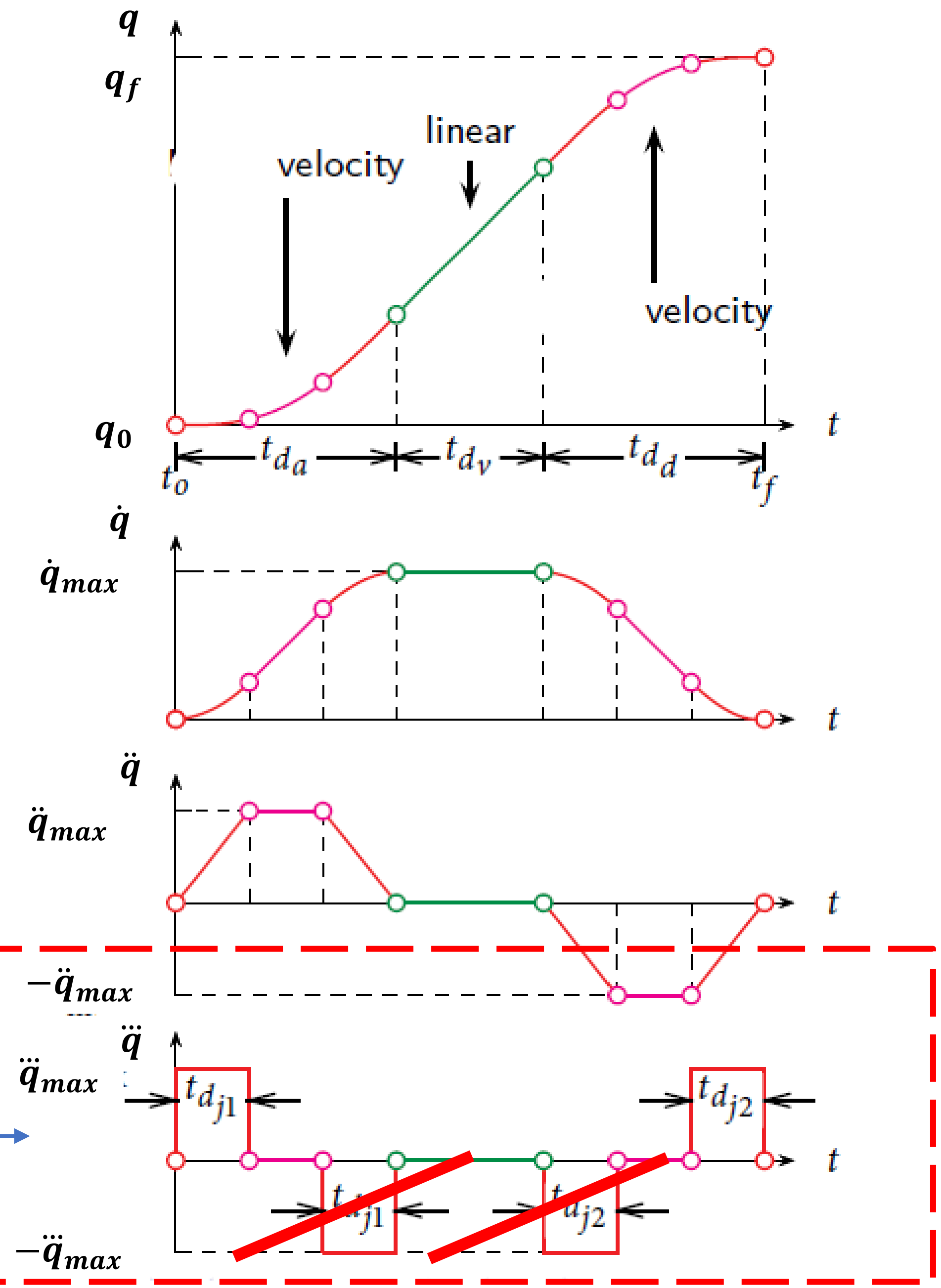
https://commons.wikimedia.org/w/index.php?curid=46643265

How does this show?



Introduction of trapezoidal acc. bled = S blends [L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer Verlag, 2008]

We are looking for = polynomial interpolation



Cubic Polynomial Interpolation

Cubic Polynomial Interpolation

- Cubic polynomial interpolation can be applied in task and joint space.
- Dense here means that we may assume that the displacements between adjacent frames are small.
- In such a case, linear interpolation is generally not suitable because the segments are two small (velocity is not continues between segments).
- Here, we recommend to use cubic interpolations or splines.

Cubic Polynomial Interpolation

- Cubic interpolation can be used in:
  - joint space, using n-tuple $X_i \equiv q^i$,
  - tool space, using 7-tuple $X_i \equiv (p^i, Q^i)$ for $i = 0,1,\ldots,N$.
- Orientations should be represented as quaternions (don't forget to normalize at the end!!).
- Between two frames ($X_{i-1}$and X_i) we obtain a curve, defined as:

$$X_{i-1,i}(t) = a_i t^3 + b_i t^2 c_i t + d_i$$

- where parameters a_i, b_i, c_i, d_i have to be determined.
- depending on the number of curves we have 4N parameters to determine.

Cubic Polynomial Interpolation

- The curves must satisfy the following constrains

$$X_{i-1,i}(t_{i-1}) = a_i t_{i-1}^3 + b_i t_{i-1}^2 + c_i t_{i-1} + d_i = X_{i-1} \; i = 1, \dots, N$$
$$X_i(t_i) = a_i t_i^3 + b_i t_i^2 + c_i t_i + d_i = X_i \; i = 1, \dots, N$$

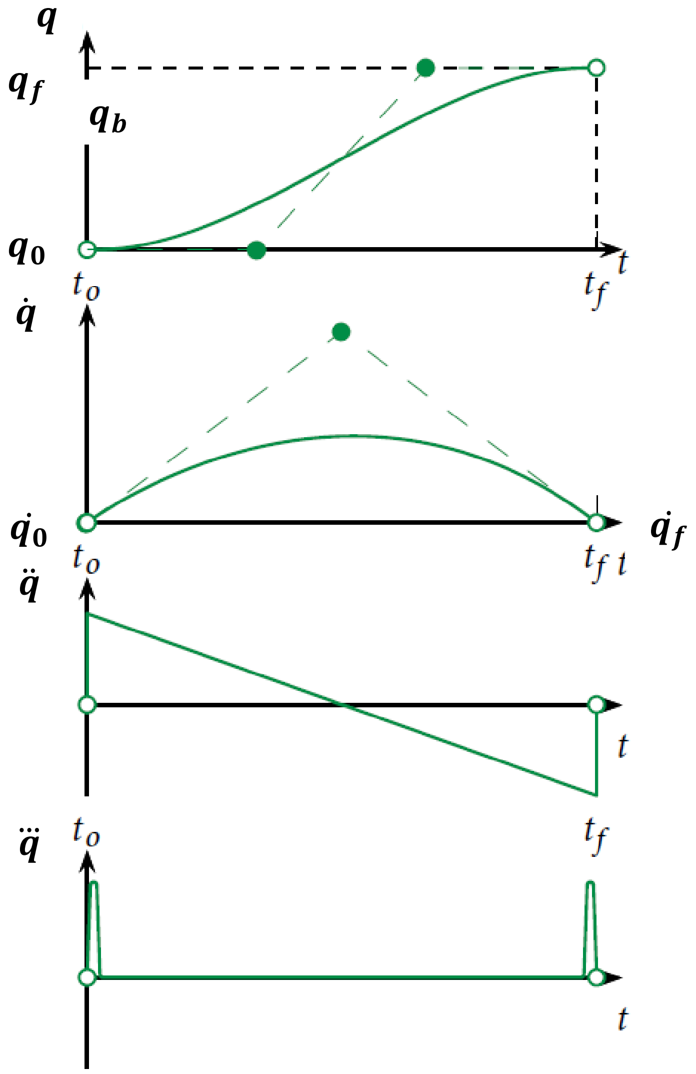
- which gives a 2N linear equations with unknowns,
- the remaining 2N equations can be given in several ways.
- for example if velocities at passage points are given we get

Cubic Polynomial Interpolation

$$\dot{X}_{i-1}(t_{i-1}) = 3a_i t_{i-1}^2 + 2b_i t_{i-1} + c_i = \dot{X}_{i-1} \quad i = 1, \dots, N$$
$$\dot{X}_i(t_i) = 3a_i t_i^2 + 2b_i t_i + c_i = \dot{X}_i \qquad i = 1, \dots, N$$

- which gives 4 linear equations with 4 unknowns for each sub-path.

Let’ take a closer look at this problem…



- Let’s consider a cub. interp. between two positions in joint space.
- where 4 parameters a0, a1 , a2 , a3 are to be determined by boundary conditions.
- Property: bounded acceleration, jerk impulse at both ends.

$$q(t) = a_0 + a_1(t-t_0) + a_2(t-t_0)^2 + a_3(t-t_0)^3$$
$$t \in [t_0, t_f], t_d \triangleq t_f - t_0$$

$$\begin{cases} q(t_0) = a_0 = q_0 \\ \dot{q}(t_0) = a_1 = \dot{q}_0 \\ q(t_f) = a_0 + a_1 t_d + a_2 t_d^2 + a_2 t_d^3 = q_f \\ \dot{q}(t_f) = a_1 + 2a_2 t_d + 3a_3 t_d^2 = \dot{q}_f \end{cases} \Rightarrow \begin{cases} a_0 = q_0 \\ a_1 = \dot{q}_0 \\ a_2 = \dfrac{3h - (2\dot{q}_0 + \dot{q}_f)t_d}{t_d^2} \\ a_3 = \dfrac{-2h + (\dot{q}_0 + \dot{q}_f)t_d}{t_d^3} \end{cases}$$

Multi point cubic interpolation

- Given $q_0, q_f, \dot{q_0}, \dot{q_f}$ at t_0, t_f and via points $[q_k]_1^m$ and $[t_k]_1^m$ where m is the number of sections between the points and k is the number of the specific point, we solve:

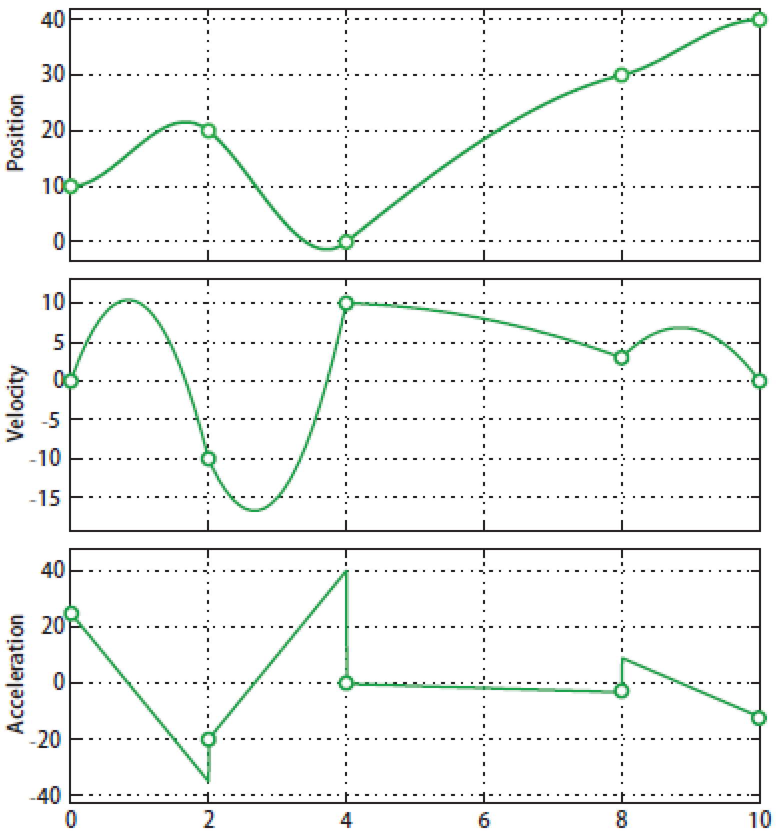
$$a_{0k} + a_{1k}(t - t_k) + a_{2k}(t - t_k)^2 + a_{3k}(t - t_k)^3$$

for all unknowns $\{a_{0k}, a_{1k}, a_{2k}, a_{3k}\}_0^m$.

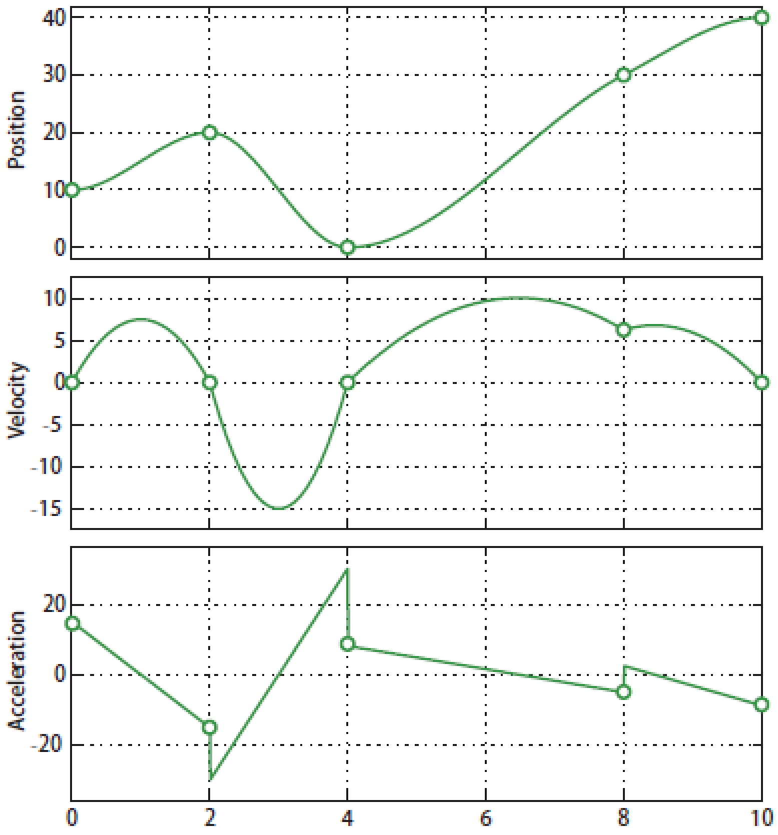
- accelerations at via points can be assigned by the user $[\dot{q_k}]_1^m$ and $t_d = t_f - t_0, h = q_f - q_0$, we can compute the coefficients.

$$\begin{cases} a_{0k} = q_0 & a_{1k} = \dot{q_0} \\ a_{2k} = \dfrac{3h - \left(2\dot{q_0} + \dot{q_f}\right)t_d}{t_d^2} & a_{3k} = \dfrac{-2h + \left(2\dot{q_0} + \dot{q_f}\right)t_d}{t_d^3}, k = 0,1, \dots, m \end{cases}$$

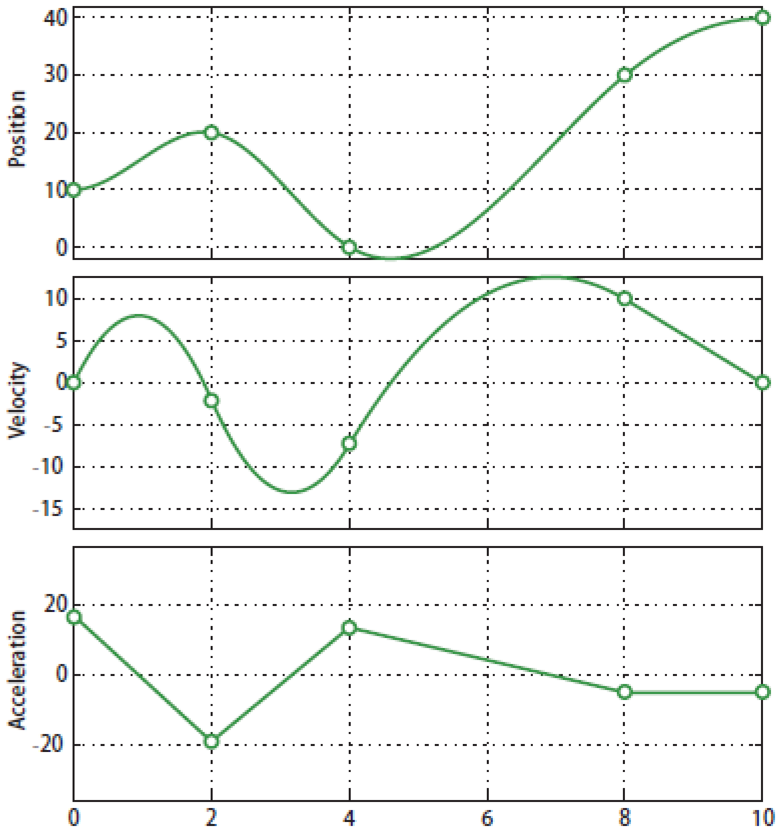
With via points Example:



Via points velocities are arbitrary assigned – large discontinuous acc.



Via points velocities assigned – large discontinuous acc.



Higher order polynomial – continuous acc.

Higher order polynomial – Quintic polynomial

$$\theta(t) = a_0 + a_1(t - t_0) + a_2(t - t_o)^2 + a_3(t - t_0)^3 + a_4(t - t_0)^4 + a_5(t - t_o)^5, t \in [t_0, t_f]$$

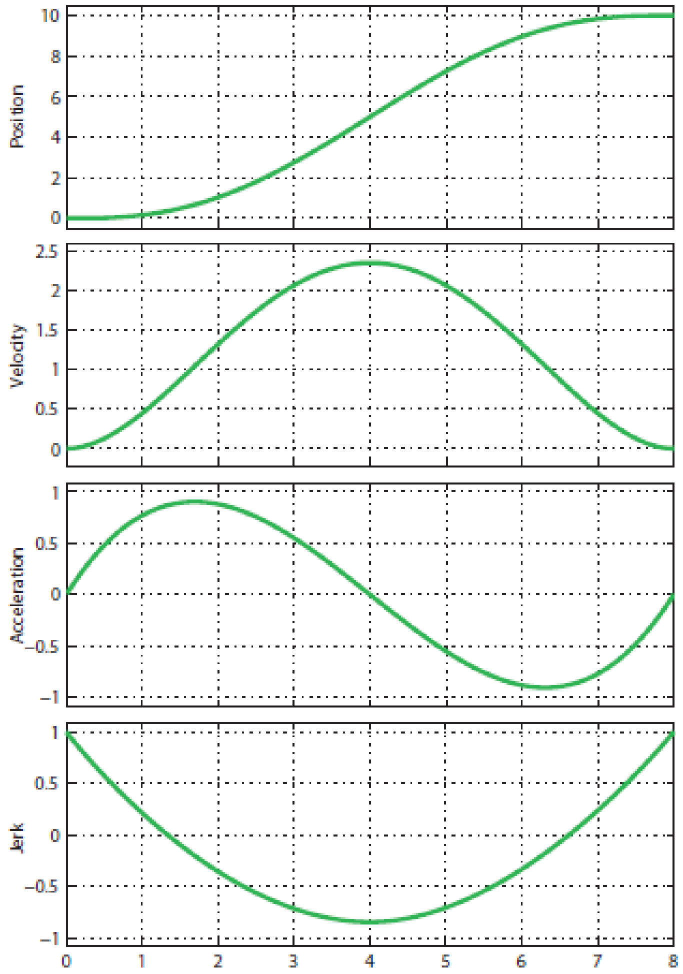
- With 6 unknown's coefficients $a_i, i = 0, \ldots, 5$.
- **Properties:**
- Smooth and bounded jerk
- Acc. continuity in composite curves.

Boundary contitions:

$$\theta(t_o) = \theta_o, \quad \theta(t_f) = \theta_f$$

$$\dot{\theta}(t_o) = \dot{\theta}_o, \quad \dot{\theta}(t_f) = \dot{\theta}_f$$

$$\ddot{\theta}(t_o) = \ddot{\theta}_o, \quad \ddot{\theta}(t_f) = \ddot{\theta}_f$$

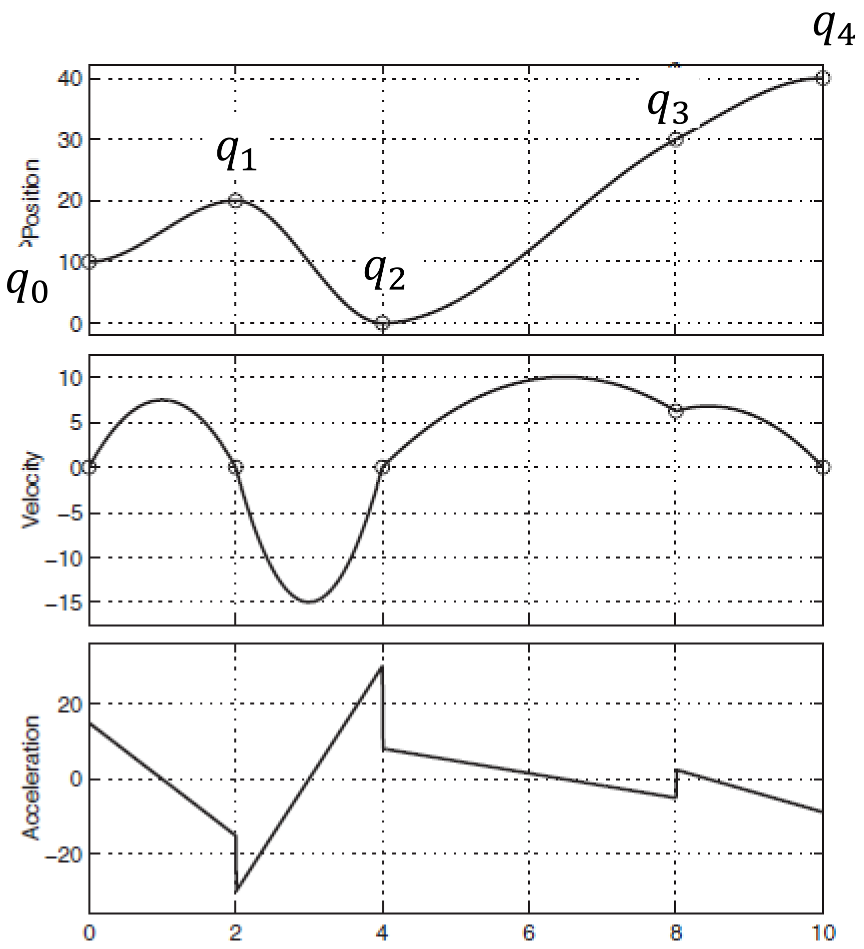


We calculate the coefficients:

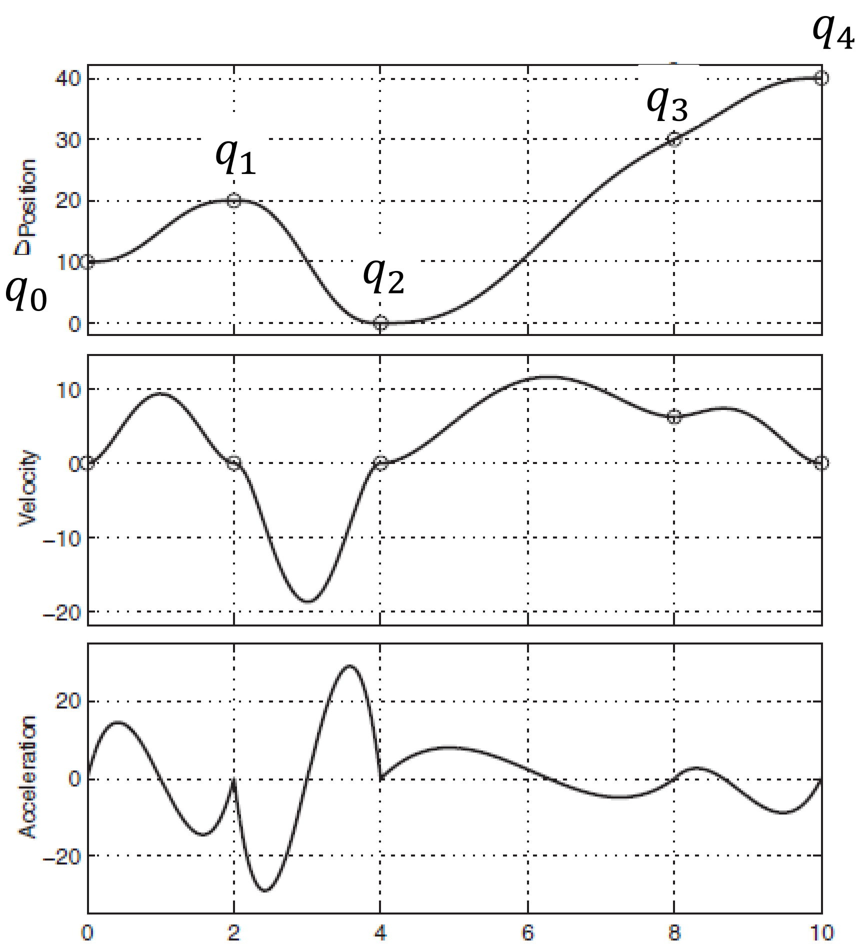
We define $t_d = t_f - t_0, h = q_f - q_0$, then

$$a_0 = q_0$$
$$a_1 = \dot{q}_0$$
$$a_2 = \frac{1}{2}\ddot{q}_0$$
$$a_3 = \frac{1}{2t_d^3}\left[20h - \left(8\dot{q}_f + 12\dot{q}_0\right)t_d - \left(3\ddot{q}_0 - \ddot{q}_f\right)t_d^2\right]$$
$$a_4 = \frac{1}{2t_d^4}\left[-30h - \left(14\dot{q}_f + 16\dot{q}_0\right)t_d - \left(3\ddot{q}_0 - 2\ddot{q}_f\right)t_d^2\right]$$
$$a_5 = \frac{1}{2t_d^2}\left[12h - 6\left(\dot{q}_f + \dot{q}_0\right)t_d - \left(\ddot{q}_f - \ddot{q}_0\right)t_d^2\right]$$

Comparison



Cubic polynomials: acceleration discontinuity.



Quintic polynomials: continuity in acceleration.

Example

Generate trajectories

- Download the Mujoco_example_UR5e_Poly_.py script. Generate joint trajectories using the following functions from the library:
  - mstraj(), jtraj() and mtraj() – make use of the provided template.
  - Use the defined parameters in the template and test what the effect of changing them would be.
  - Check how the functions work and implement them such that they can be used with via points.
  - Calculate the velocity, acceleration and jerk for the entire generated trajectory.
  - Plot the results in subplots
  - Compare the results

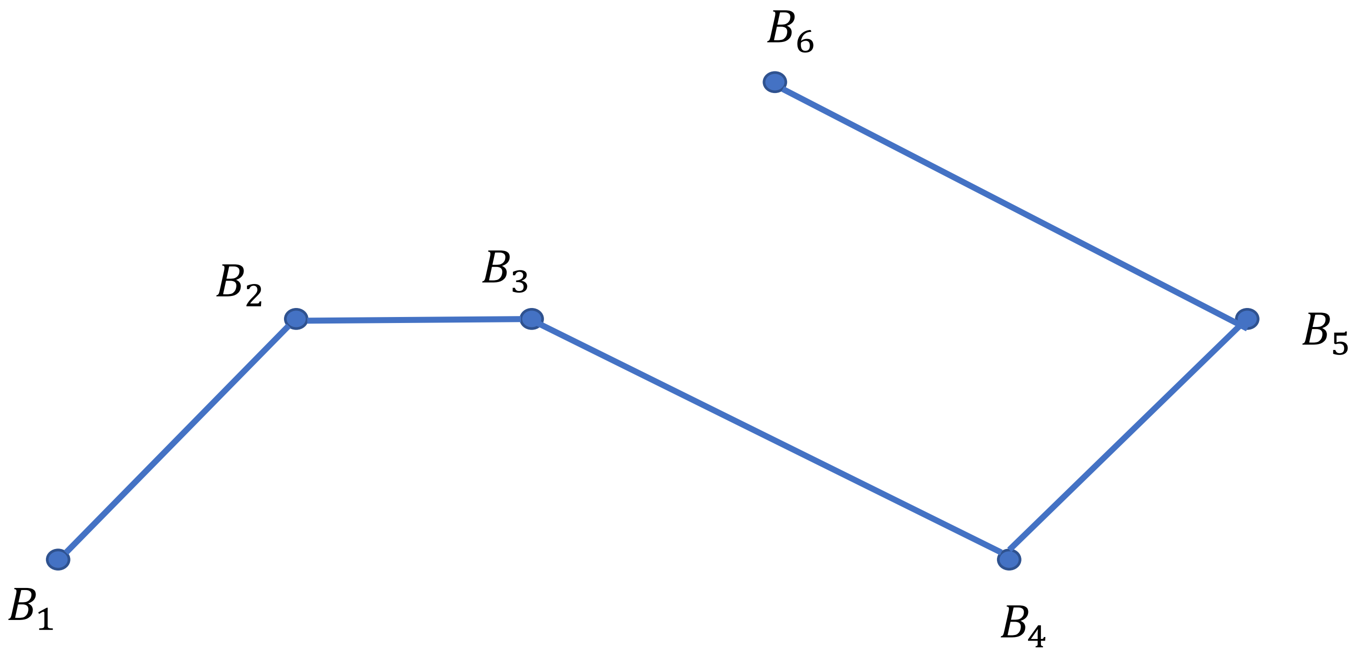
Splines

Splines

- The word “spline” refers to thin strip of wood or metal. At one time, curves were designed for ships or planes by mounting actual strips so that they went through a desired points but were free to move otherwise.
- Definition:
  - A cubic spline curve is a piecewise cubic curve with continuous second derivative.
- Definition (a special case):
  - A cubic spline curve is relaxed if its second derivative is zero at each endpoint.
- An easy way of making a controlled-design curve with many control points is to use B-spline curves.

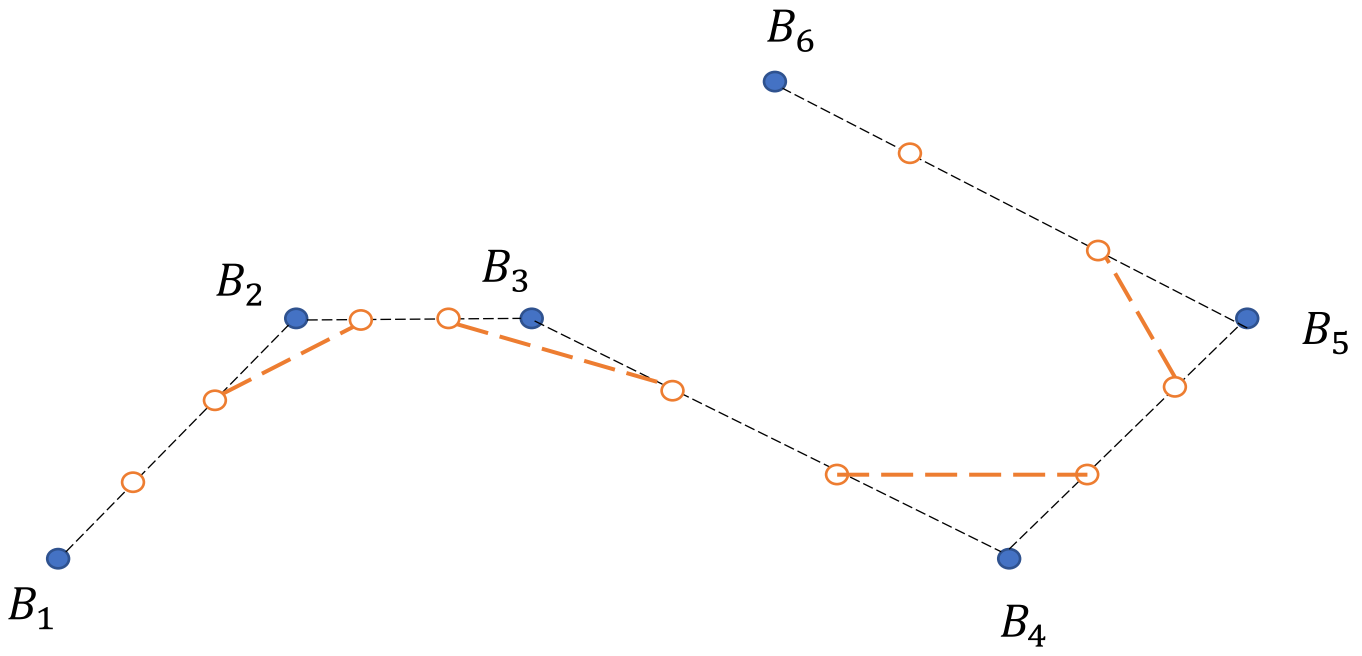
Splines - by hand

- Lets consider a couple of points.
- we can connect them with a line (do linear interpolation).
- maybe not optimal.



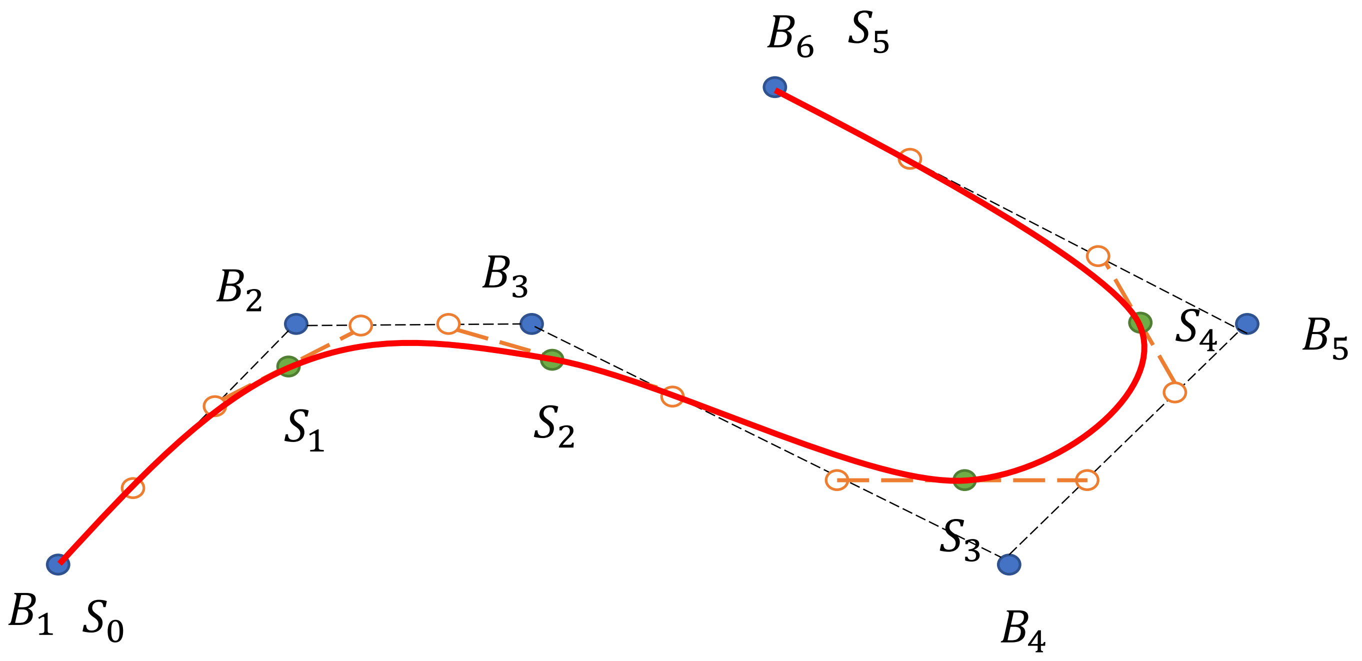
Splines – by hand

- We can divide the space between the points into thirds,
- and connect them between each other.



Splines – by hand

- Split the connecting line into half,
- which gives us the anchor points of the spline,
- A cubic spline is defined via fitting a Bezier Curve to the anchor points.



Cubic splines

- If the velocities are prescribed only at the first and last frame then the following set of equations are used:

$$\dot{X}_{0,1}(t_0) = 3a_1 t_0^2 + 2b_1 t_0 + c_1 = \dot{X}_0$$

$$\dot{X}_{N-1,N}(t_N) = 3a_N t_N^2 + 2b_N t_N + c_N = \dot{X}_N$$

$$3a_i t_i^2 + 2b_i t_i + c_i = 3a_{i+1} t_i^2 + 2b_{i+1} t_i + c_{i+1} \quad i = 1, \ldots, N$$

$$6a_i t_i + 2b_i = 6a_{i+1} t_i + 2b_{i+1} \quad i = 1, \ldots, N$$

- thus we obtain a set of 4N coupled linear equations that can be solved by standard numerical methods.

Dynamic Movement Primitives

Dynamic movement primitives (DMPs)

- **DMPs** are defined at the acceleration level:

$$\tau \dot{z} = K(g - y_0) - Dz + f(x)$$
$$\tau \dot{y} = z$$

- Guarantees continuity and smoothness for robot control,
- preserves the physical meaning of $f(x)$ as a force,
- **g** represents the goal of the desired movement,
- y_0 represents the start of the desired movement,
- K,D positive gains, configured in such a way that the dynamics has a stable point at the defined g.

Benefits

- Not explicit time dependent,
- demonstrated/ recorded movement can be reproduced,
- the movement can be altered online,
- the movement can be defined with a couple of simple parameters,
- the movement can be stopped and resumed again,
- the system is stable, and can handle disturbances,
- can be implemented on various systems.

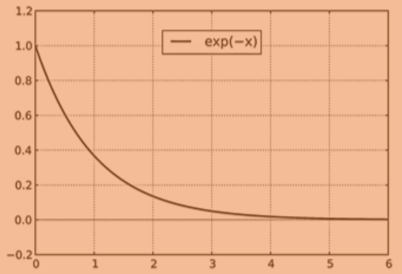
Recap of Dynamic Movement Primitives

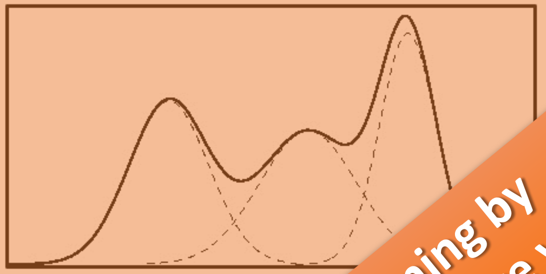
Trajectory Dynamics
Transformation System

$$\tau\,\dot{v} = K(g - y) - Dv + f(x)$$
$$\tau\,\dot{x} = v$$

Temporal Evolution
Phase System

$$\dot{x} = -\alpha x$$



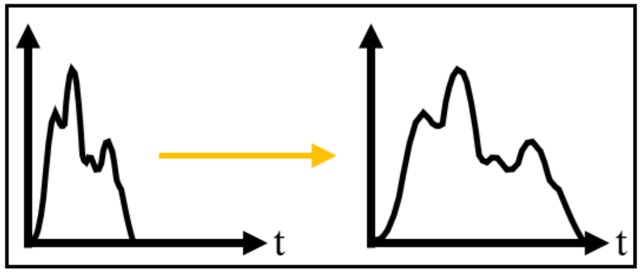


Shape Appro
Forcin
(Learned fr

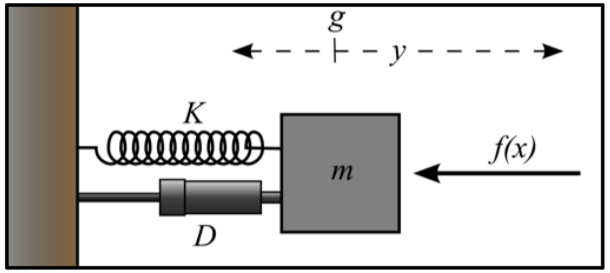
$f(x)$ ... $(g - y_0)$

$e^{-\frac{(x-c_i)^2}{\sigma^2}}$

Applied in robot learning by demonstration – a lecture will be given next semester.



Time Constant
Controls Duration of Movement



2nd Order Dynamical System
(Spring-damper system)

DMP: Cartesian-space Orientation

Position
- Three independent regular DMPs.
- Expressed here in vector form.

$$\tau\dot{z} = \alpha_z\big(\beta_z(g_p - p) - z\big)$$
$$\tau\dot{p} = z$$

Orientation
- Represented as unit quaternions.
- Singularity free representation.
- Similar properties as positional DMPs.

$$\tau\dot{\eta} = \alpha_z\big(\beta_z(2log(g_q * \bar{q})) - \eta\big)$$
$$\tau\dot{q} = \frac{1}{2}\eta * q$$

Shared Phase
- Synchronizes position and orientation

$$\tau\dot{x} = -\alpha_x x$$

Take home message

Position and velocity limits can be directly incorporated in the inverse kinematics'.

DMP’s can be utilized for constructing point to point movements – properties.

With a higher order polynomial, better approximation and acceleration , jerk conditions are obtained.

Splines, provide continuous acceleration and velocities, with the consequence of precision inaccurate.