Robot trajectory generation

Václav Hlaváč

Czech Technical University in Prague

Center for Machine Perception *(bridging groups of the)*

Czech Institute of Informatics, Robotics and Cybernetics and

Faculty of Electrical Engineering, Department of Cybernetics

`http://people.ciirc.cvut.cz/hlavac, hlavac@ciirc.cvut.cz`

Courtesy: Alessandro De Luca, Universita di Roma, Sapienza; other presentations from the web.

Outline of the talk:

- Trajectory $\times$ path.
- Trajectory generation, problem formulation.
- Trajectory in operational and joint spaces.
- Curve approximation.
- Trajectory classification.
- Trajectory approximation by polynomials.

Practical requirements related to trajectories

- Provide the capability to move the manipulator arm and its end effector or the mobile robot from the initial posture to the final posture.
- Motion laws have to be considered in order not to:
  - violate saturation limits of joint drives,
  - excite the resonant modes of the actuator mechanical structure, being driven by electric/hydraulic/pneumatic or other more exotic drives.
- Explore path planning and trajectory generation methods providing smooth trajectories solving the practical robotics task.

Question: Why are smooth trajectories preferred?

Terminology: path vs. trajectory

- *Note: Terms path, trajectory are often confused. They are used as synonyms in informal discussions.*
- **Path** consists of ordered locii of points in the space (either the joint space or the operational space), which the robot should follow.
  - The path provides a pure geometric description of motion.
  - The path is usually planned globally taking into account obstacle avoidance, traversing a complicated maze, etc.
- **Trajectory** is a path plus velocities and accelerations in its each point.
  - The design of a trajectory does not need global information, which simplifies the task significantly.
  - The trajectory is specified and designed locally (divide and conquer methodology). Parts of the path are covered by individual trajectories.
  - It is often required that pieces of trajectories join smoothly, which induces that a single trajectory design takes into account only neighboring trajectories from the path.

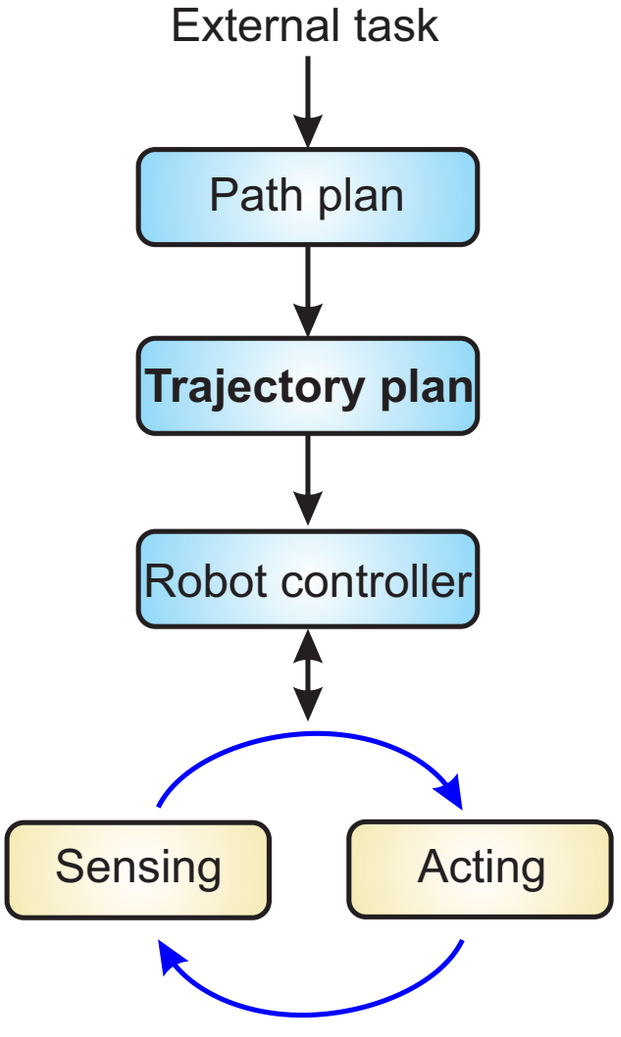
Robot motion planning, an overview

Path planning (global)

- The (geometric) path is a sequence of waypoints defining the trajectory coarsely.
- Issues solved at this level: obstacle avoidance, shortest path.

Trajectory generating (local)

- The path provided by path planning constitutes the input to the trajectory generator.
- Trajectory generator "approximates" the desired path waypoints by a class of polynomial functions and
- generates a time-based control sequence moving the manipulator/mobile platform from its initial configuration to the destination.



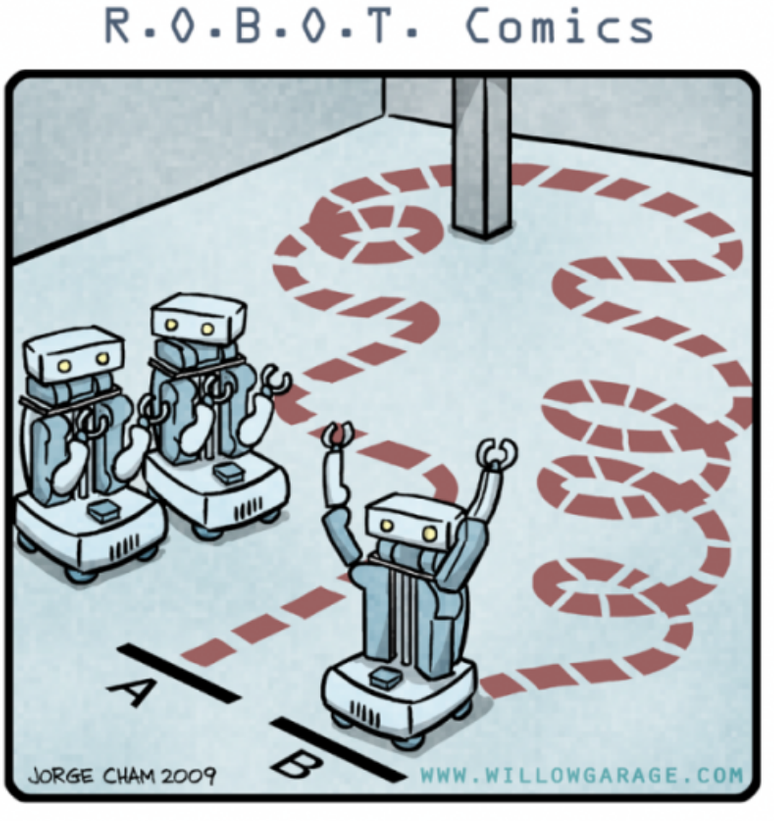
Path planning, the problem formulation

- The path planning task:

  Find a collision free path for the robot from one configuration to another configuration.

- Path planning is an algorithmically difficult search problem.

  - The involved task has an exponential complexity with respect to the degrees of freedom (controllable joints).
  - With industrial robots, the path planning is often replaced by a path/trajectory taught in by a human operator.



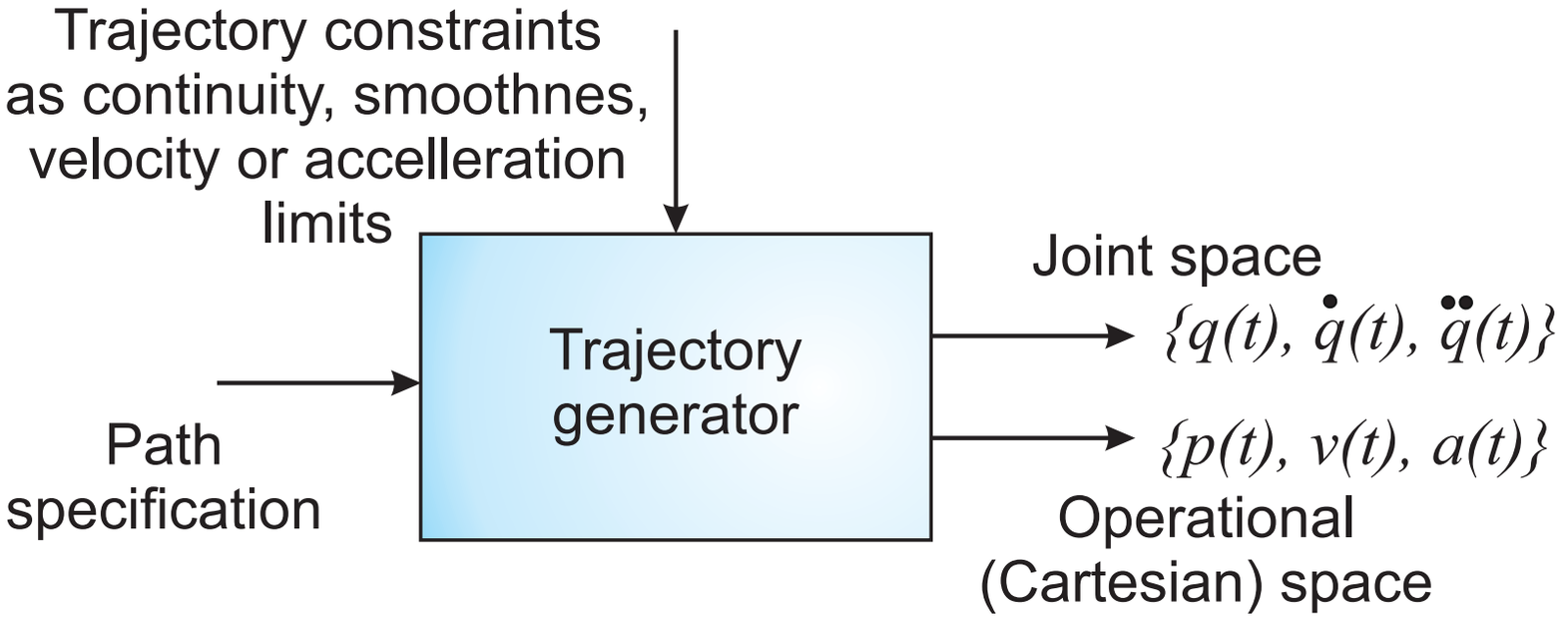
Note: A separate (next) lecture will be devoted to the path planning.

Trajectory generation, the problem formulation

- Trajectory generation aims at creating inputs to the motion control system which ensures that the planned trajectory is smoothly executed.
- The planned path is typically represented by via-points, which is the sequence of points (or end-effector poses) along the path.
- Trajectory generating = creating a trajectory connecting two or more via points.
  - In the industrial settings, a trajectory is taught-in by a human expert and later played back (by teach-and-playback).

    A more recent approach utilizes several tens of trajectories performed by human experts as the input. They vary statistically. Machine learning techniques are used to create the final trajectory.
  - In general, e.g. with mobile robots and more and more with industrial robots, path points are smoothly approximated using methods of function approximation from mathematics.

Trajectory generating, illustration



- Trajectory generating = finding the desired joint space trajectory $\mathbf{q}(t)$ given the desired operational (Cartesian) path inverse kinematics.
- $\mathbf{q}$ is a vector of joint parameters. Its dimension matches to the number of DOFs.
- $\mathbf{p}(t) = (x(t), y(t), z(t))$ is the position, $\mathbf{v}(t) = (x'(t), y'(t), z'(t)$ is the velocity, $\mathbf{a}(t) = (x''(t), y''(t), z''(t)$ is the acceleration.

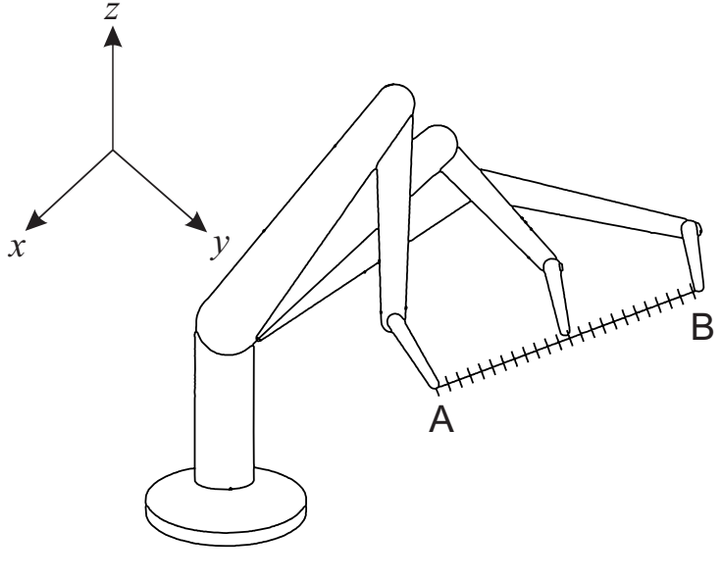
Joint space vs. operational space

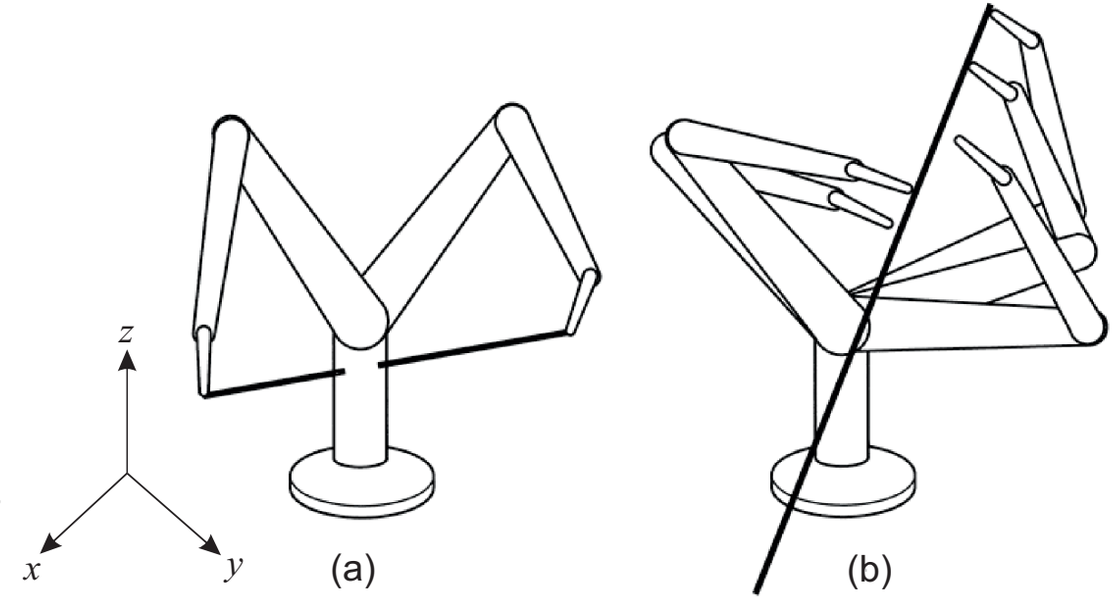
- Joint-space description:
  - The description of the motion to be made by the robot by its joint values.
  - The motion between the two points is unpredictable.
- Operational space description:
  - In many cases operational space = Cartesian space.
  - The motion between the two points is known at all times and controllable.
  - It is easy to visualize the trajectory, but it is difficult to ensure that singularity does not occur.

Path planning or trajectory generation can be performed either in the joint space or operational space.

Example:
Troubles with the operational (Cartesian) space

Create a trajectory of the manipulator end effector to follow a straight line.





(a) The trajectory specified in Cartesian coordinates may force the robot to run into itself, and

(b) the trajectory may requires a sudden change in the joint angles due to singularities.

Trajectory in the operational space

- Calculate the path from the initial point to the final point.
- Assign a total time T_{path} to traverse the path.
- Discretize points in time and space.
- Blend a continuous time function between these points
- Solve inverse kinematics at each step.

Advantages

- Collision free path can be obtained.

Disadvantages

- Computationally expensive due to inverse kinematics.
- It is unknown how to set the total time T_{path}.

Trajectory in the joint space

- Calculate the inverse kinematics solution from the initial point to the final point.
- Assign the total time T_{path} using maximal velocities in joints.
- Discretize the individual joint trajectories in time.
- Blend a continuous function between these point.

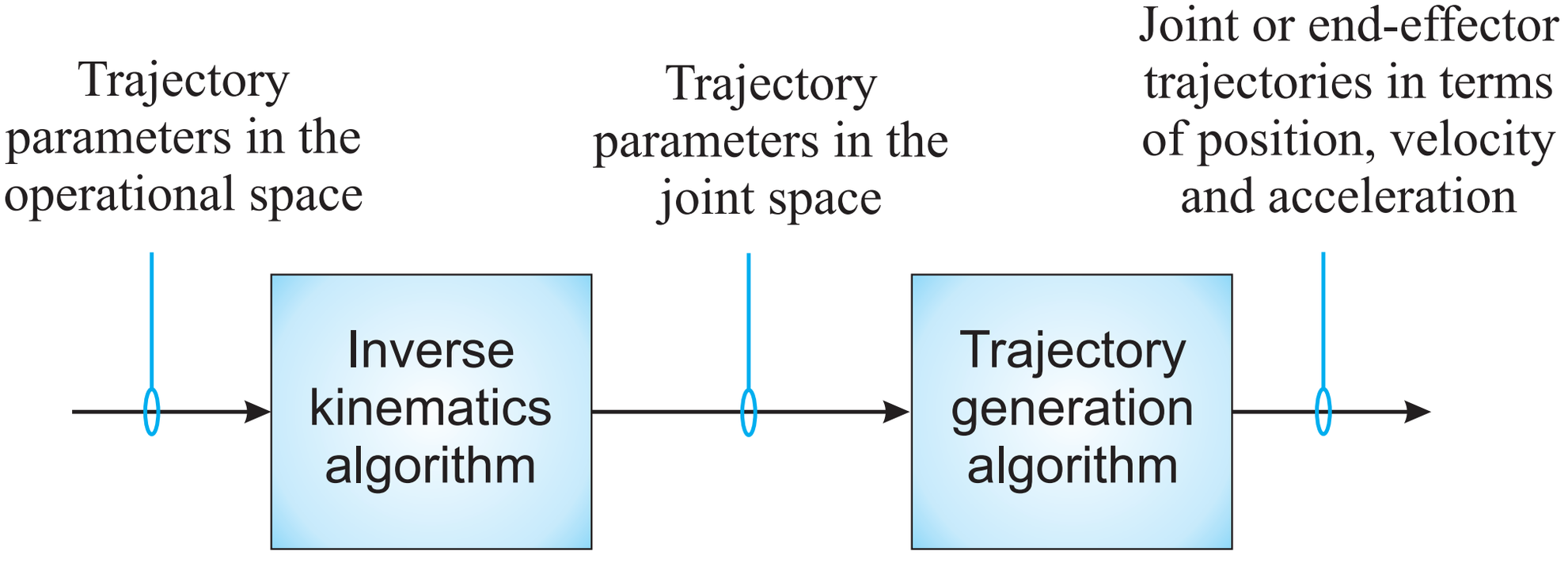
Advantages

- The inverse kinematics is computed only once.
- The approach can take into account joint angles limits and velocity constraints easily.

Disadvantages

- Cannot deal with obstacles represented in the operational space easily.

Trajectory from the operational space to the joints space



E.g., initial and final end-effector location, travelling time.

Types of the trajectory control

- Displacement control = control the end effector/mobile robot displacement, i.e. angles or positions in space, maybe including dynamics of motion.

  Examples:

  - Moving payloads.
  - Painting objects.

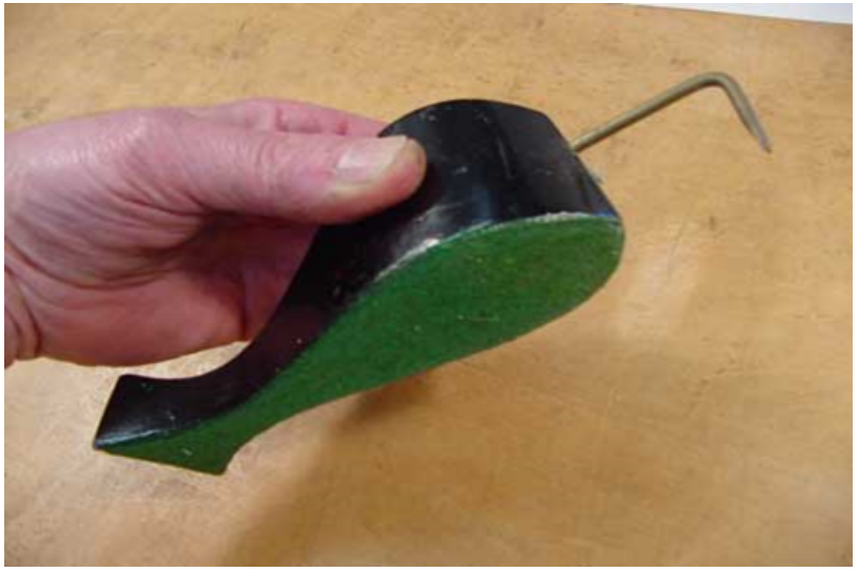
- Force control = control of both displacement (as above) and applied force.

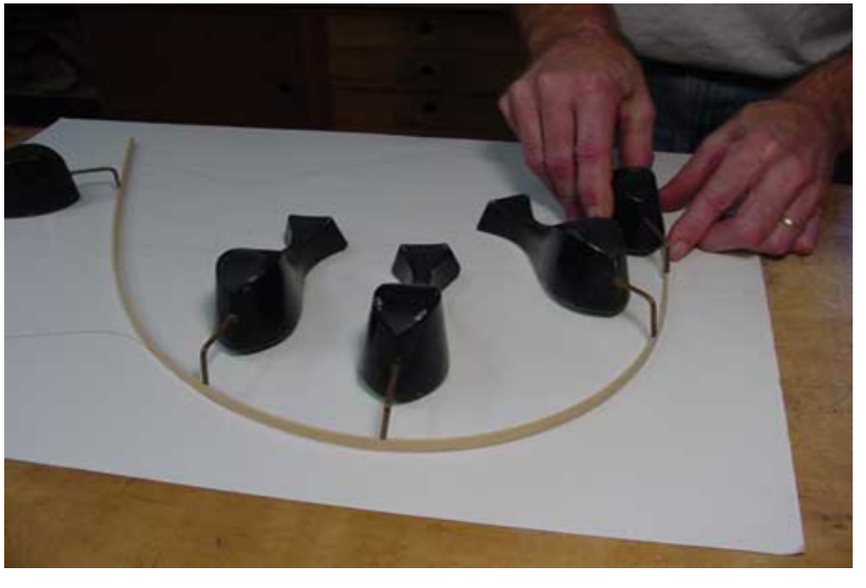
  Examples:

  - Machining.
  - Grinding.
  - Sanding.

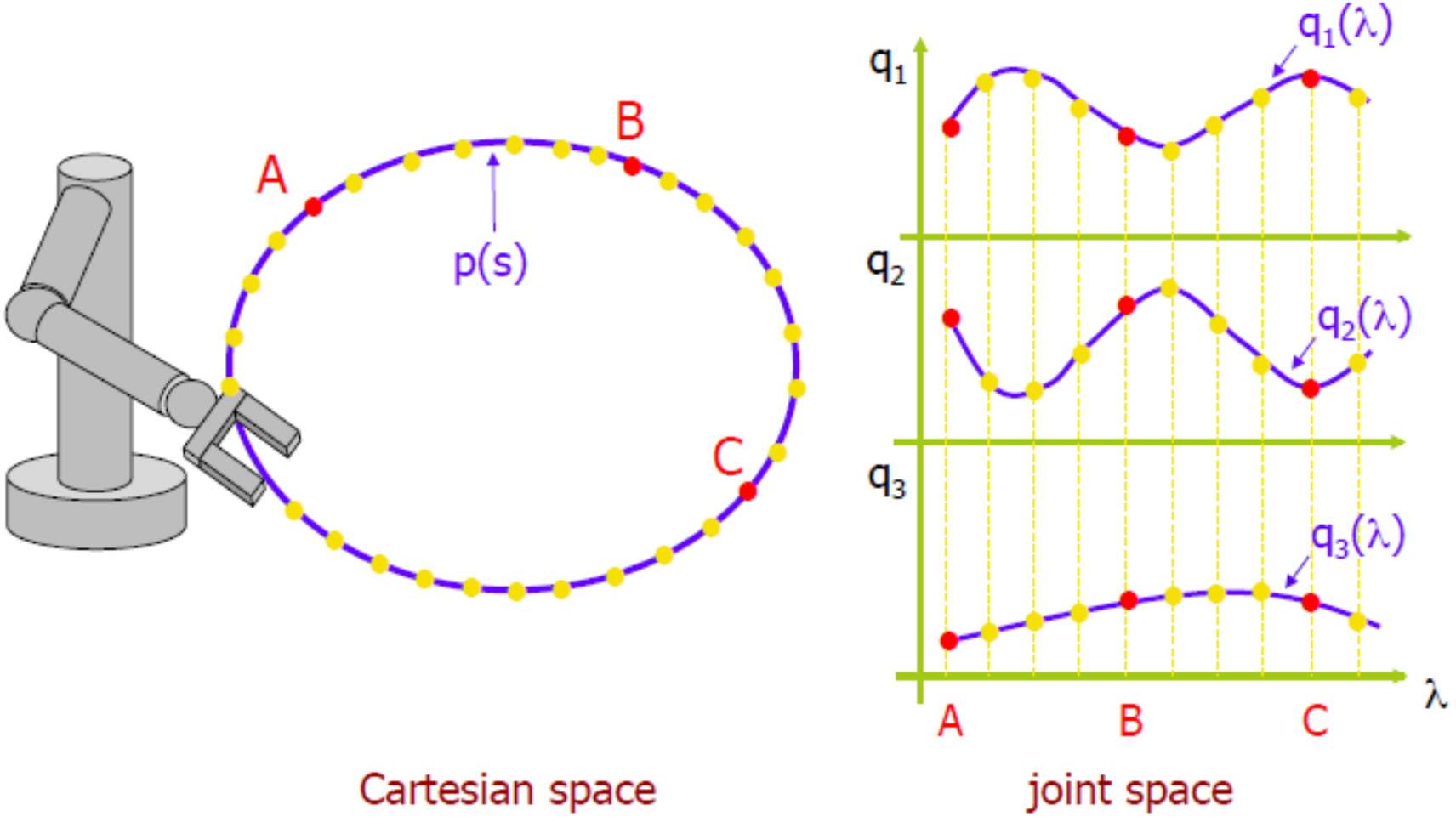
Curve approximation, the motivation

- A draftsman uses “ducks” and strips of wood (splines) to draw curves.
- “Wooden splines” provide the second-order continuity
- and pass through control points.





Motivating example of trajectories in operational and joint spaces



Cartesian space joint space

Courtesy for the image: Alessandro De Luca, Universita di Roma, Sapienza.

Trajectory = path + timing law

- Having the path (way-points), the trajectory is completed by a choice of a timing law
  - Operational (Cartesian) space: $\mathbf{p}(s) = (x(s), y(s), z(s)) \Rightarrow s = s(t)$.
  - Joint space: $\mathbf{q}(\lambda) = (q_1(\lambda), q_2(\lambda), \ldots, q_n(\lambda))$, where $n =$ DOFs $\Rightarrow \lambda = \lambda(t)$.

  If $s(t) = t$, the trajectory parametrization is the natural one given by the time.

- The timing law:
  - is chosen based on task specifications (stop in a point, move at a constant velocity, etc.);
  - may consider optimality criteria (min transfer time, min energy, etc.);
  - constraints are imposed by actuator capabilities (e.g. max torque, max velocity) and/or by the task (e.g., max acceleration on payload).

Note on the space-time decomposition on parametrized path

E.g., in the operational (Cartesian) space:

$$\dot{\mathbf{p}}(t) = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s}\,\dot{s}(t), \qquad \ddot{\mathbf{p}}(t) = \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}s}\,\ddot{s}(t) + \frac{\mathrm{d}^2\mathbf{p}}{\mathrm{d}s^2}\,\dot{s}(t)$$

Trajectory classification

- Space of the definition: the operational (Cartesian) space or the joint space.
- Task type: point-to-point (PTP), multiple points (knots), continuous, concatenated.
- Path geometry: rectilinear, polynomial, exponential, cycloid, ...
- Timing law: bang-bang in acceleration, trapezoidal in velocity, polynomial, ...
- Coordinated or independent:
  - Motion of all joints (or of all Cartesian components) start and ends at the same instants (say, $t = 0$ and $t = T$) $\Rightarrow$ the single timing law.
  - Motions are timed independently, e.g. according to requested displacement and robot capabilities. Such trajectory is performed in the joint space mostly.

Relevant characteristics of the trajectory

- Computational efficiency and memory space, e.g. store only coefficients of a polynomial function.
- Predictability vs. "wandering" out of the knots.
- Accuracy vs. the "overshot" on the final position.
- Flexibility allowing concatenation, over-fly, ...
- Continutity in space and in time.

  Continuity C^1 at least. Sometimes also up to C^2, i.e. up to jerk $= \frac{\mathrm{d}a}{\mathrm{d}t}$, where a is the acceleration.

Note: Continuity of the class C^k means that the time derivative up to the order k is smooth.

Trajectory planning in the joint space

- The curve parametrization in time $q = q(t)$ or in space $q = q(\lambda)$, $\lambda = \lambda(t)$
- It is sufficient to work component-wise, (q_i in vector q).
- An implicit definition of the trajectory is obtained jby solving a problem with specified boundary conditions in a given class of functions.
- Typical classes: polynomials (cubic, quintic, ...), (co)sinusoids, clothoids, ...
- Imposed conditions
  - Passage through points $\Rightarrow$ interpolation.
  - Initial, final, intermediate velocity or geometric tangent to the path.
  - Initial, final, intermediate velocity or geometric curvature.
  - Continuity up to C^k.

Many of the following methods and remarks can be applied directly also to Cartesian trajectory planning (and vice versa).

A cubic polynomial

- Four boundary constraints: $q(0) = q_{\text{ini}}$; $q(T) = q_{\text{fin}}$; $\dot{q}(0) = v_{\text{ini}}$; $\dot{q}(T) = v_{\text{fin}}$.
- $\Delta q = q_{\text{fin}} - q_{\text{ini}}$; the curve parametrization $\tau = t/T$, $\tau \in [0, 1]$.
- A cubic polynomial $q(\tau) = q_{\text{ini}} + \Delta q\,(a\tau^3 + b\tau^2 + c\tau + d)$.
- A "doubly normalized" polynomial $q_N(\tau)$ such that

  $q_N(0) = 0 \Leftrightarrow d = 0.$

  $q_N = 1 \Leftrightarrow a + b + c = 1.$

  $\dot{q}_N(0) = \left.\frac{\mathrm{d}q_N}{\mathrm{d}\tau}\right|_{\tau=0} = c = \frac{v_{\text{ini}}T}{\Delta q}.$

  $\dot{q}_N(1) = \left.\frac{\mathrm{d}q_N}{\mathrm{d}\tau}\right|_{\tau=1} = 3a + 2b + c = \frac{v_{\text{fin}}T}{\Delta q}.$

Cubic polynomial, a spacial case, rest-to-rest

- The boundary constraints and the parametrization remains as above.
- A cubic polynomial as well: $q(\tau) = q_{\text{ini}} + \Delta q\,(a\tau^3 + b\tau^2 + c\tau + d)$.
- $q_N(0) = 0 \Leftrightarrow d = 0.$

  $\dot{q}_N(0) = 0 \Leftrightarrow c = 0.$

  $q_N = 1 \Leftrightarrow a + b = 1.$

  $\dot{q}_N(1) = 0 \Leftrightarrow 3a + 2b = 0.$

  From previous two equations, $a = -2$, $b = 3$.

Quintic polynomial

- $q(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f$, i.e. 6 coefficients.
- satisfying constraints, e.g. in the normalized time τ: $q(0) = q_0$; $q(1) = q_1$; $\dot{q}(0) = v_0 T$; $\dot{q}(1)$; $\ddot{q}(0) = a_0 T^2$; $\ddot{q}(1) = a_1 T^2$;

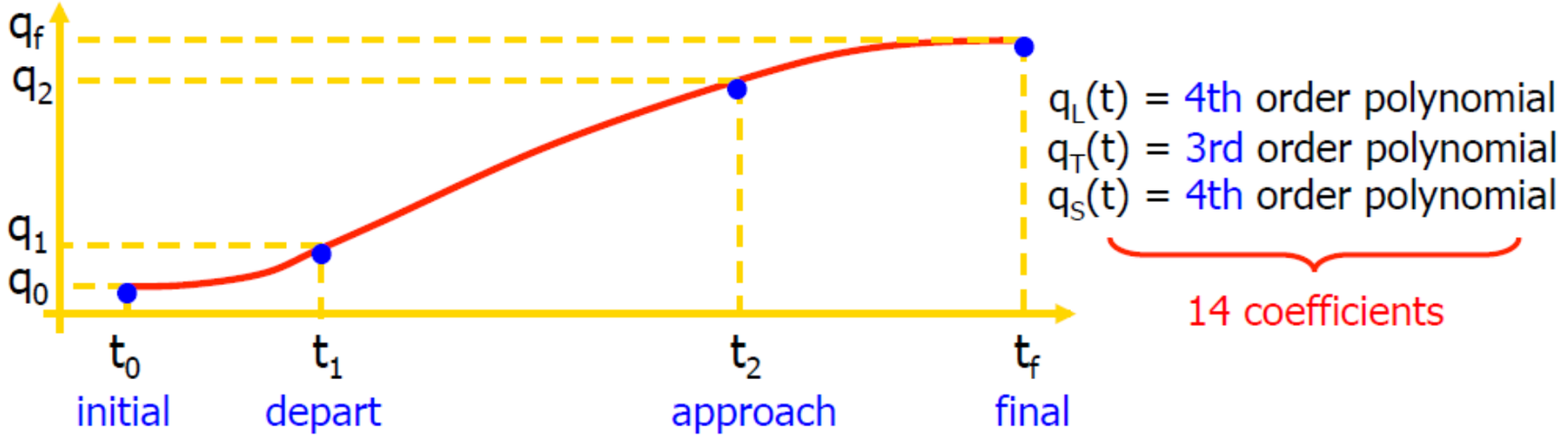
$$
\begin{aligned}
q(\tau) &= (1-\tau)^3 \left(q_0 + (3q_0 + v_0 T)\,\tau + (a_0 T^2 + 6v_0 T + 12q_0)\,\frac{\tau^2}{2} \right) \\
&= +\tau^3 \left(q_1 + (3q_1 - v_1 T)(1-\tau) + \frac{(a_1 T^2 - 6v_1 T + 12q_1)(1-\tau)^2}{2} \right)
\end{aligned}
$$

A special case, rest-to-rest:

- $v_0 = v_1 = a_0 = a_1 = 0$.
- $q(\tau) = q_0 + \Delta q(6\tau^5 - 15\tau^4 + 10\tau^3)$; $\Delta q = q_1 - q_0$.

4-3-4 polynomials

Three phases in pick-and-place operations: Lift off, Travel, Set down.



Boundary constraints:

$$\begin{aligned} q(t_0) &= q_0,\ q(t_1^-) = q(t_1^+) = q_1, \quad q(t_2^-) = q(t_2^+) = q_2,\ q(t_f) = q_f \\ \dot{q}(t_0) &= \dot{q}(t_f) = 0, \quad \ddot{q}(t_0) = \ddot{q}_f(t_f) = 0 \\ \dot{q}(t_0^-) &= \dot{q}(t_0^+), \qquad \ddot{q}(t_i^-) = \ddot{q}(t_i^+), i = 1, 2 \end{aligned}$$

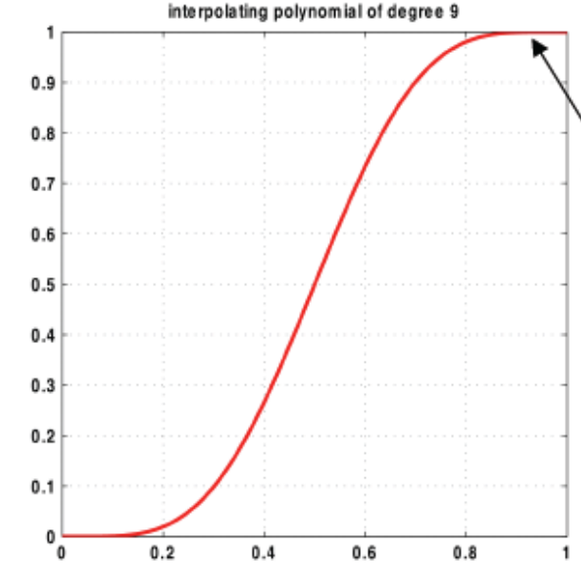
The first equation corresponds to six pasages; the second equation to 4 initial/final velocities/accelaretions; the third equation to 4 continuity constraints.

Higher-order polynomials

- provide a suitable solution class for satisfying symmetric boundary conditions in a point-to-point motion that imposes zero values on higher-order derivatives.
  - The interpolating polynomial is always of the odd degree.
  - The coefficients of such (doubly normalized) polynomial are always integers, alternate in sign, sum up to unity, and are zero for all term up to the power $= \frac{(\text{degree}-1)}{2}$.
- In all other cases (e.g., for interpolating a large number N of points), the use of higher-order polynomials is not recommended.
  - N-th order polynomials have $N-1$ maximum and minimum points.
  - Oscilations arise out of the interpolation points (wandering).

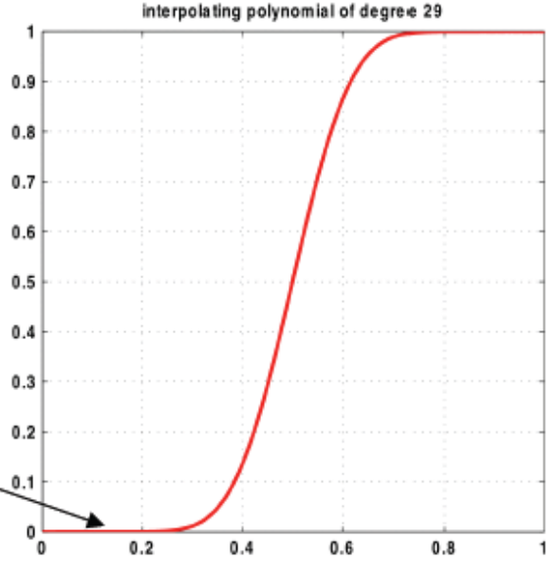
Numerical examples

9th degree



4 derivatives are zero

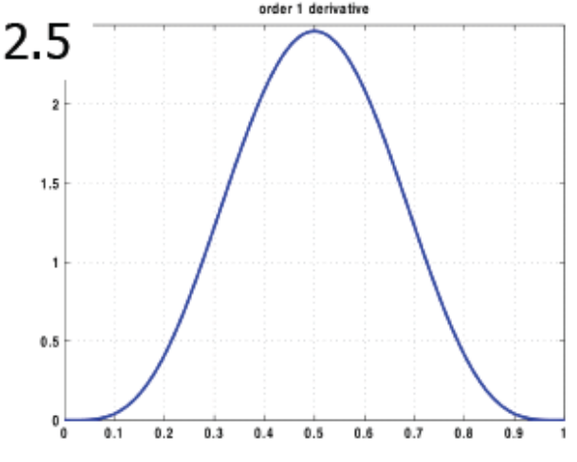
14 derivatives are zero!



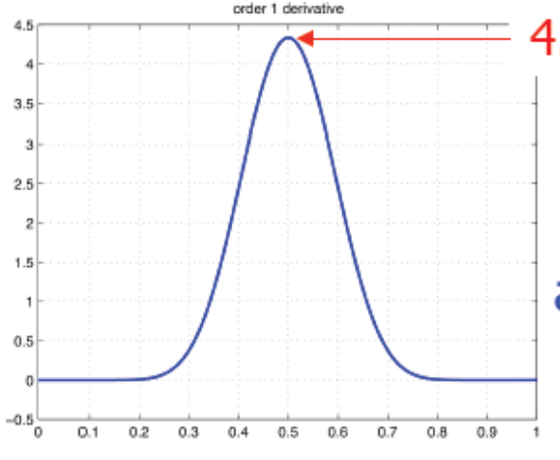
29th degree

no overshoot nor wandering

2.5



normalized velocity



4.5!!

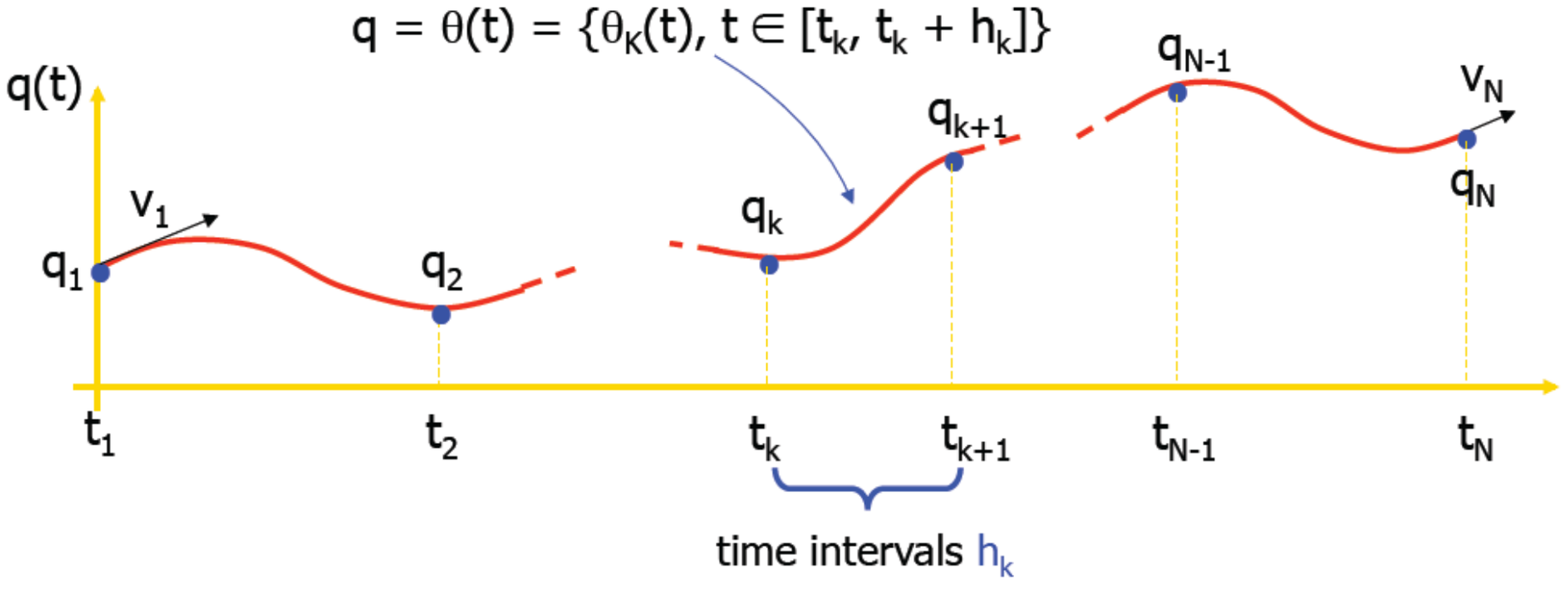
velocity peaking at midpoint

Interpolation using splines

- Task formulation: Interpolate N knots along the desired curve, keep C^2 continuity, i.e. continuity up to the second derivative.
- Solution: Approximate by splines.

  Splines are $N-1$ cubic polynomials concatenated to pass through knots (also control points) and being continuous in velocity and acceleration in the $N-2$ internal knots.
- $4\,(N-1)$ coefficients.
- $4\,(N-1)-2$ conditions, more specifically
  - $2\,(N-1)$ passages, pro each cubic in the two knots at its ends.
  - $N-2$ continuities for velocities at internal knots.
  - $N-2$ continuities for accelerations at internal knots.
- Two free parameters are still left over. These parameters can be used to, e.g., assign the initial and final velocities v_1, v_N.
- We present curves in therms of time t. It is similar for space λ.

Building a cubic spline



- The polynomial $\Theta_K(\tau) = a_{k0} + a_{k1}\tau + a_{k2}\tau^2 + a_{k3}\tau^3;\ \tau \in [0, h_k]$, $\tau = t - t_k,\ k = 1, \ldots, N-1.$
- Continuity conditions for velocity and acceleration:
  $\dot{\Theta}_k(h_k) = \dot{\Theta}_{k+1}(0);\ \ddot{\Theta}_k(h_k) = \ddot{\Theta}_{k+1}(0);$
  $k = 1, \ldots, N-2.$

An efficient algorithm

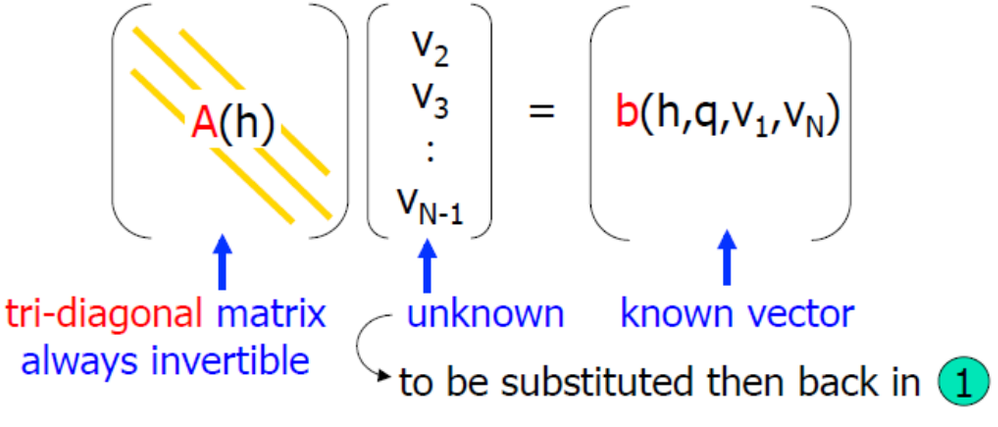
1. If all velocities v_k at internal knots were known then each cubic in the spline would be uniquely determined by

$$\begin{aligned} \Theta_k(0) &= q_k = a_{k0} \\ \dot{\Theta}(0) &= v_k = a_{k1} \end{aligned} \qquad \begin{bmatrix} h_{k2} & h_{k3} \\ 2\,h_k & 3\,h_{k2} \end{bmatrix} \begin{bmatrix} a_{k2} \\ a_{k3} \end{bmatrix} = \begin{bmatrix} q_{k+1} - q_k - v_k\,h_k \\ \\ v_{k+1} - v_k \end{bmatrix}$$

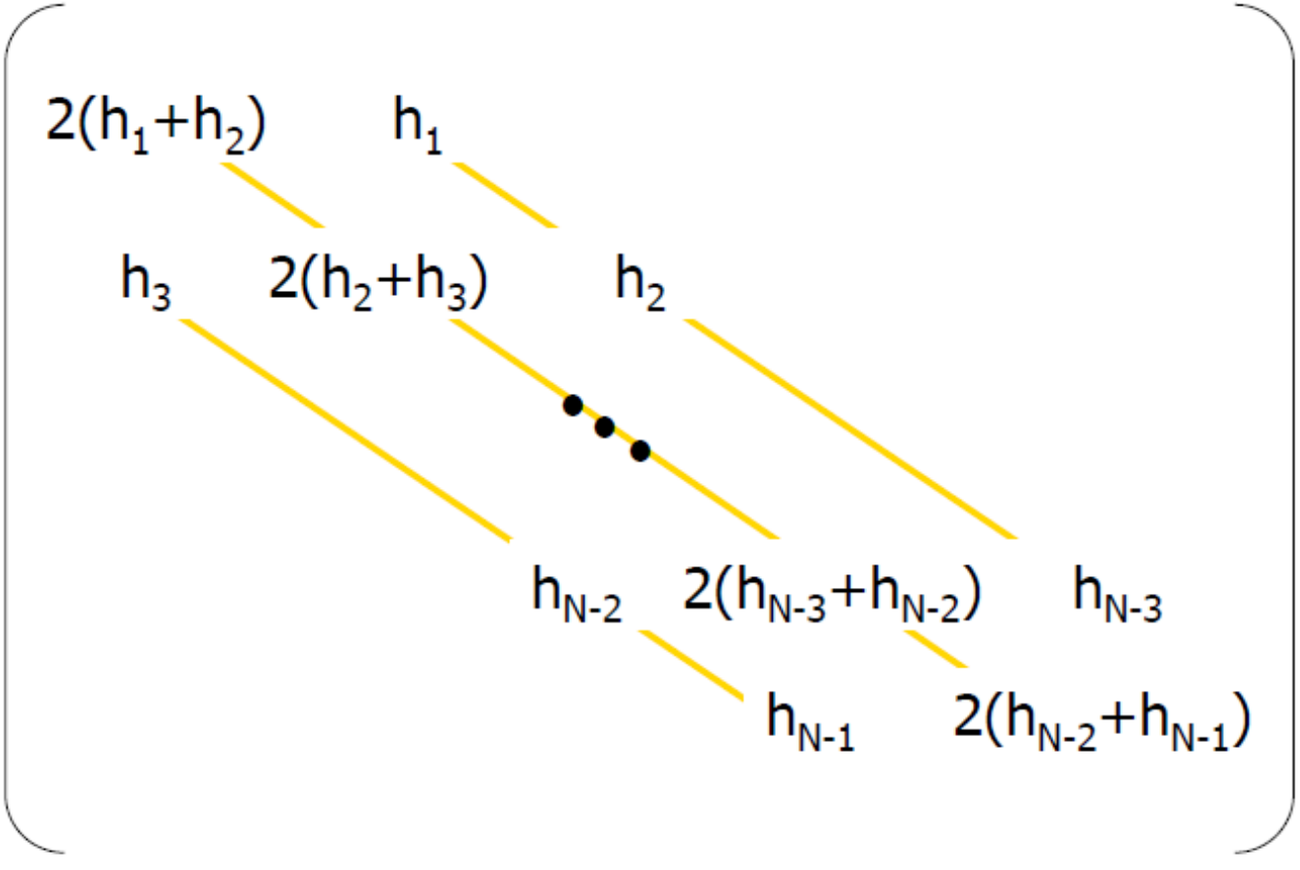
2. Impose $N-2$ acceleration continuity constraints

$$\ddot{\Theta}_k(h_k) = 2a_{k2} + 6a_{k3}h_k = \ddot{\Theta}_{k+1}(0) = 2\,a_{k+1,2}$$

3. Expressing the coefficients a_{k2}, a_{k3}, $a_{k+1,2}$ in terms of still unknown knot velocities, see step 1, yields a linear system of equations, which are always solvable



Structure of $A(h)$



diagonally dominant matrix (for $h_k > 0$)
[the same matrix for all joints]

Structure of $b(h, q, v_1, v_N)$

$$\frac{3}{h_1 h_2}\left[h_1^2(q_3 - q_2) + h_2^2(q_2 - q_1)\right] - h_2 v_1$$

$$\frac{3}{h_2 h_3}\left[h_2^2(q_4 - q_3) + h_3^2(q_3 - q_2)\right]$$

$$\vdots$$

$$\frac{3}{h_{N-3} h_{N-2}}\left[h_{N-3}^2(q_{N-1} - q_{N-2}) + h_{N-2}^2(q_{N-2} - q_{N-3})\right]$$

$$\frac{3}{h_{N-2} h_{N-1}}\left[h_{N-2}^2(q_N - q_{N-1}) + h_{N-1}^2(q_{N-1} - q_{N-2})\right] - h_{N-2} v_N$$

Properties of splines

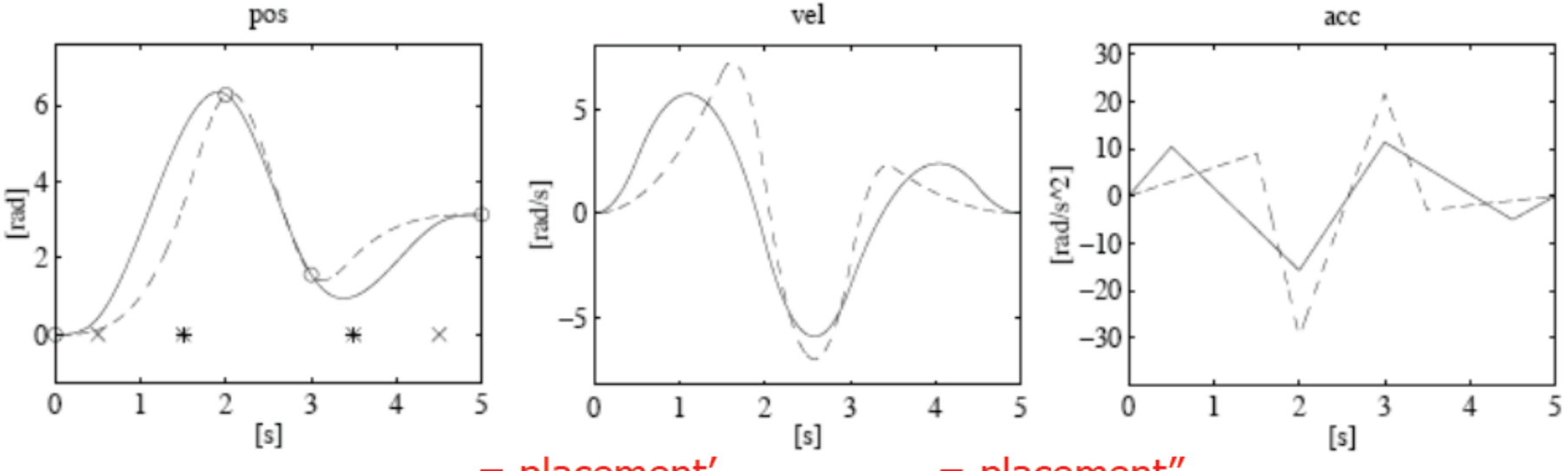
- The spline is the solution with the minimum curvature among all interpolating functions having continuous derivatives up to the second one.
- A spline is uniquely determined from the data $q_1, \ldots, q_n$, $h_1, \ldots, h_{N-1}$, $v_1, \ldots, v_n$.
- The total transfer time is $T = \sum_{k=1}^{K} h_k = t_N - t_1$.
- The time intervals h_k can be chosen to minimize T (linear objective function) under (nonlinear) bounds on velocity and acceleration in $[0, T]$.
- For cyclic taks ($q_1 = q_N$), it is preferable to impose simply the continuity of velocity and acceleration at $t_1 = t_N$ as the "squaring" conditions
  - in fact, even choosing $v_1 = v_N$ does not guarantee the acceleration continuities
  - in this way, the first = last knot will be handled as all other internal knots
- When initial and final accelerations are also assigned, the spline construction can be suitably modified.

A modification handling assigned initial and final accelerations

- Two more parameters are needed in order to impose also the initial acceleration α_1 and final acceleration α_N.
- Two "fictious knots" are inserted in the first and last original intervals, increasing the number of cubic polynomials from $N-1$ to $N+1$.
- In these two knots only continuity conditions on position, velocity and acceleration are imposed $\Rightarrow$ two free parameters are left over (one in the first cubic and one in the last cubic), which are used to satisfy the boundary constraints on acceleration.
- Depending on the (time) placement of the two additional knots, the resulting spline changes.

A numerical example

- N = 4 knots (3 cubic polynomials)
  - joint values $q_1 = 0$, $q_2 = 2\pi$, $q_3 = \pi/2$, $q_4 = \pi$
  - at $t_1 = 0$, $t_2 = 2$, $t_3 = 3$, $t_4 = 5$ (thus, $h_1 = 2$, $h_2 = 1$, $h_3 = 2$)
  - boundary velocities $v_1 = v_4 = 0$
- 2 added knots to impose accelerations at both ends (5 cubic polynomials)
  - boundary accelerations $\alpha_1 = \alpha_4 = 0$
  - two placements: at $t_1' = 0.5$ and $t_4' = 4.5$ (×), or $t_1'' = 1.5$ and $t_4'' = 3.5$ (*)



——— = placement′ ---------- = placement″