Robotics 1

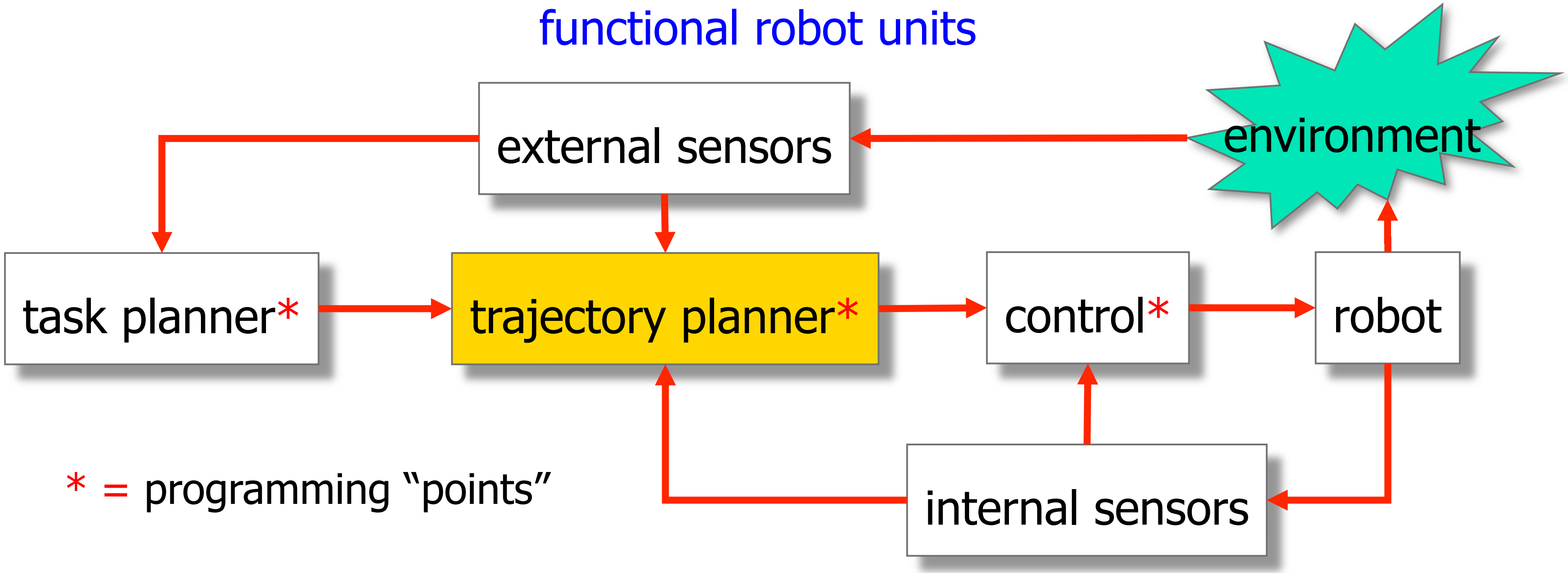
Trajectory planning

Prof. Alessandro De Luca

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

Trajectory planner interfaces



* = programming "points"

robot action described as a sequence of poses or configurations (with possible exchange of contact forces)

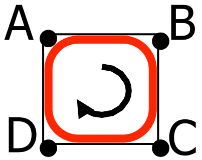
reference profile/values (continuous or discrete) for the robot controller

Trajectory definition
a standard procedure for industrial robots

1. define Cartesian pose points (position+orientation) using the teach-box
2. program an (average) velocity between these points, as a 0-100% of a maximum system value (different for Cartesian- and joint-space motion)
3. linear interpolation in the joint space between points sampled from the built trajectory

examples of additional features

a) over-fly



b) sensor-driven STOP

c) circular path through 3 points

main drawbacks

- semi-manual programming (as in "first generation" robot languages)
- limited visualization of motion

➡ a mathematical formalization of trajectories is useful/needed

From task to trajectory

TRAJECTORY $\begin{cases} \text{of motion } p_d(t) \\ \text{of interaction } F_d(t) \end{cases}$

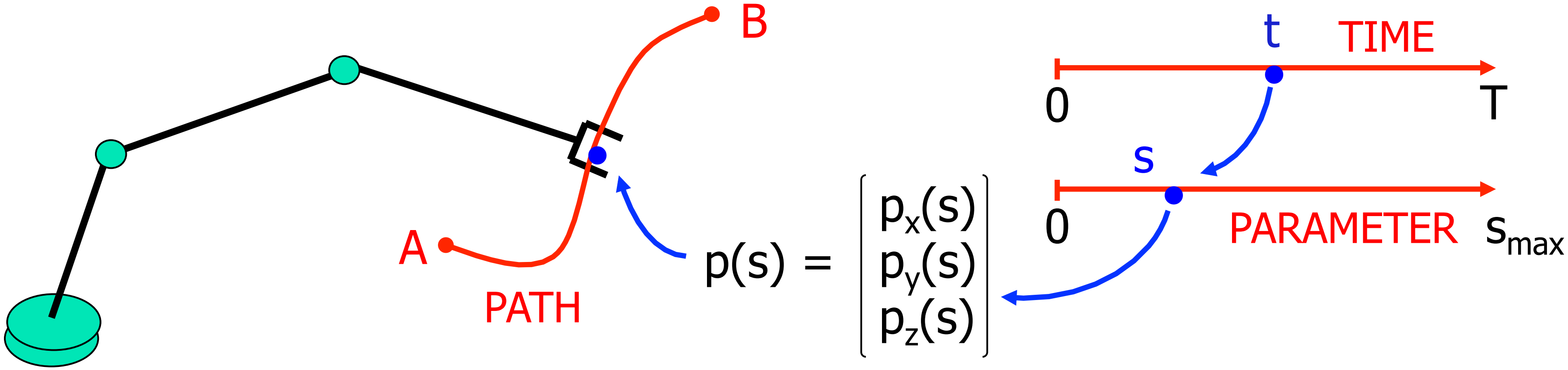
=

GEOMETRIC PATH parameterized by s: $p=p(s)$
(e.g., s is the arc length)

+

TIMING LAW describes the time evolution of $s=s(t)$

$\Big\}\ p(s(t))$

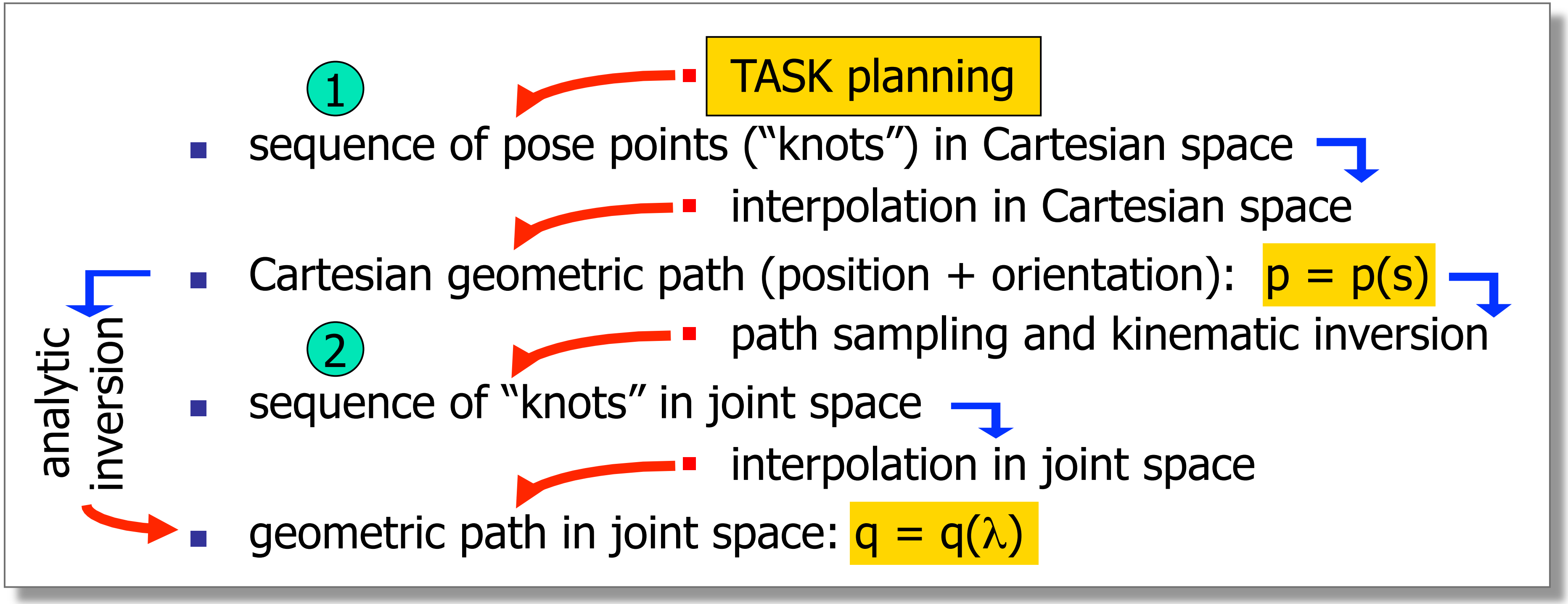


$$p(s) = \begin{bmatrix} p_x(s) \\ p_y(s) \\ p_z(s) \end{bmatrix}$$

example: TASK planner provides A, B
TRAJECTORY planner generates $p(t)$

Trajectory planning

operative sequence



TASK planning

①

- sequence of pose points ("knots") in Cartesian space

  - interpolation in Cartesian space

- Cartesian geometric path (position + orientation): $p = p(s)$

②

  - path sampling and kinematic inversion

- sequence of "knots" in joint space

  - interpolation in joint space

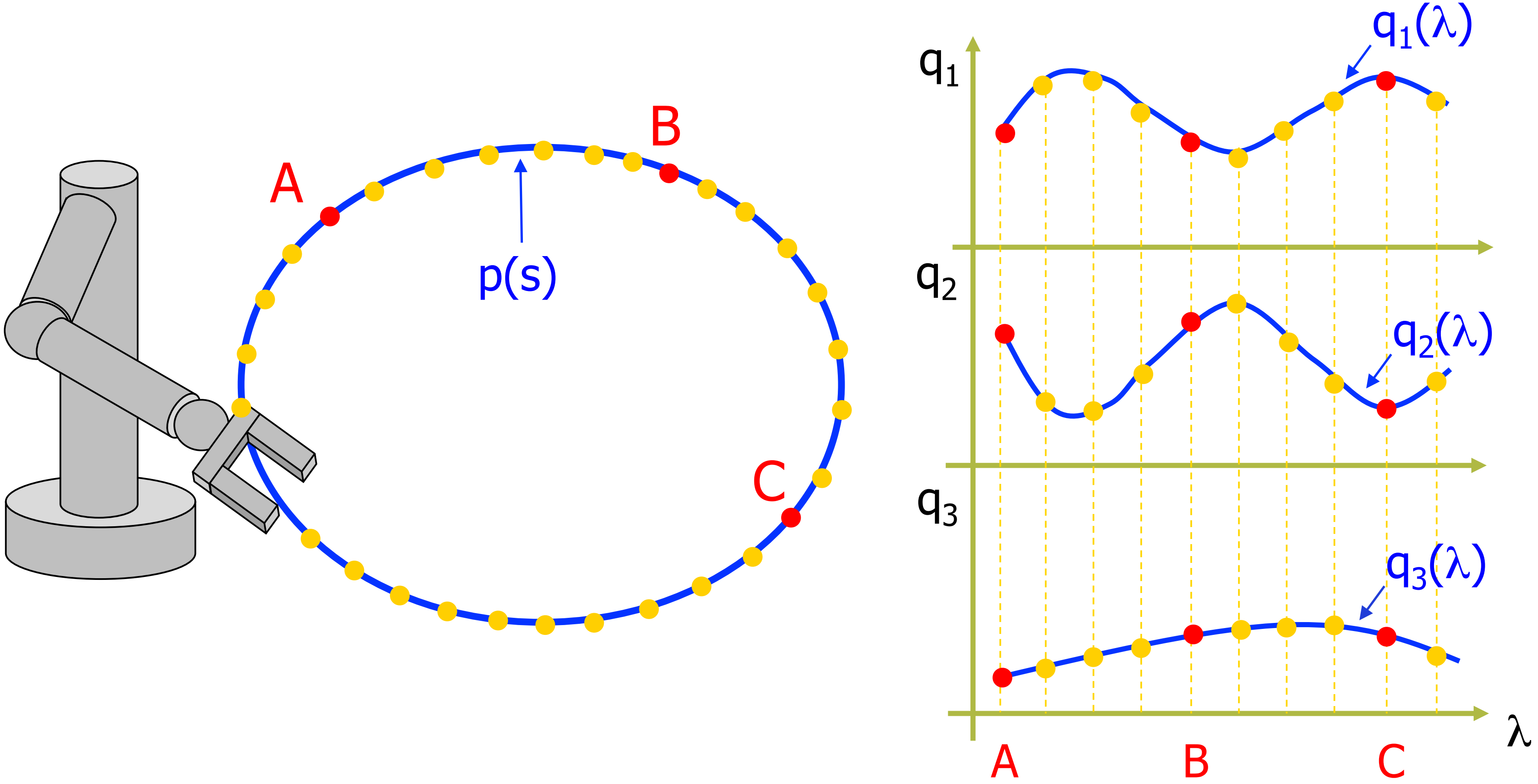
- geometric path in joint space: $q = q(\lambda)$

analytic inversion

additional issues to be considered in the planning process

- obstacle avoidance
- on-line/off-line computational load
- sequence ② is more "dense" than ①

Example



Cartesian space

joint space

Cartesian vs. joint trajectory planning

- planning in Cartesian space
  - allows a more direct visualization of the generated path
  - obstacle avoidance, lack of "wandering"
- planning in joint space
  - does not need on-line kinematic inversion
- issues in kinematic inversion
  - $\dot{q}$ e $\ddot{q}$ (or higher-order derivatives) may also be needed
    - Cartesian task specifications involve the geometric path, but also bounds on the associated timing law
  - for redundant robots, choice among ∞^{n-m} inverse solutions, based on optimality criteria or additional auxiliary tasks
  - off-line planning in advance is not always feasible
    - e.g., when interaction with the environment occurs or sensor-based motion is needed

Path and timing law

- after choosing a path, the trajectory definition is completed by the choice of a timing law

$$p = p(s) \quad \Rightarrow s = s(t) \qquad \text{(Cartesian space)}$$

$$q = q(\lambda) \quad \Rightarrow \lambda = \lambda(t) \qquad \text{(joint space)}$$

  - if s(t) = t, path parameterization is the natural one given by time
- the timing law
  - is chosen based on task specifications (stop in a point, move at constant velocity, and so on)
  - may consider optimality criteria (min transfer time, min energy,…)
  - constraints are imposed by actuator capabilities (max torque, max velocity,…) and/or by the task (e.g., max acceleration on payload)

note: on parameterized paths, a space-time decomposition takes place

e.g., in Cartesian space

$$\dot{p}(t) = \frac{dp}{ds}\dot{s} \qquad \ddot{p}(t) = \frac{dp}{ds}\ddot{s} + \frac{d^2p}{ds^2}\dot{s}^2$$

Trajectory classification

- space of definition
  - Cartesian, joint
- 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) = single timing law

  or
  - motions are timed independently (according to the requested displacement and robot capabilities) – mostly only in joint space

Relevant characteristics

- computational efficiency and memory space
    - e.g., store only the coefficients of a polynomial function
- predictability (vs. “wandering” out of the knots) and accuracy (vs. “overshoot” on final position)
- flexibility (allowing concatenation, over-fly, …)
- continuity (in space and in time)

  (at least **C^1**, but also up to jerk = $\frac{da}{dt}$)

Trajectory planning in joint space

- q = q(t) or q = q(λ), λ = λ(t)
- it is sufficient to work component-wise (q_i in vector q)
- an implicit definition of the trajectory, by 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 = interpolation
  - initial, final, intermediate velocity (or geometric tangent for paths)
  - initial, final acceleration (or geometric curvature)
  - continuity up to the k-th order time (or space) derivative: class $\mathbf{C}^k$

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

Cubic polynomial

$\boxed{q(0) = q_{in}}$ $\boxed{q(T) = q_{fin}}$ $\boxed{\dot{q}(0) = v_{in}}$ $\boxed{\dot{q}(T) = v_{fin}}$ ← 4 conditions

$$\Delta q = q_{fin} - q_{in}$$

$$\tau = t/T, \; \tau \in [0, 1]$$

$$q(\tau) = q_{in} + \Delta q \, \underbrace{[a\tau^3 + b\tau^2 + c\tau + d]}_{\text{"doubly normalized" polynomial } q_N(\tau)}$$

4 coefficients → "doubly normalized" polynomial $q_N(\tau)$

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

$q_N(1) = 1 \;\Leftrightarrow\; a + b + c = 1$

$q_N'(0) = dq_N/d\tau|_{\tau=0} = c = v_{in}T/\Delta q$

$q_N'(1) = dq_N/d\tau|_{\tau=1} = 3a + 2b + c = v_{fin}T/\Delta q$

special case: $v_{in} = v_{fin} = 0$ (rest-to-rest)

$$q_N'(0) = 0 \;\Leftrightarrow\; c = 0$$

$$\left.\begin{array}{l} q_N(1) = 1 \;\Leftrightarrow\; a + b = 1 \\ q_N'(1) = 0 \;\Leftrightarrow\; 3a + 2b = 0 \end{array}\right\} \Leftrightarrow \begin{array}{l} a = -2 \\ b = 3 \end{array}$$

Quintic polynomial

$q(\tau) = a\tau^5 + b\tau^4 + c\tau^3 + d\tau^2 + e\tau + f$ 6 coefficients

$\tau = t/T,\ \tau \in [0, 1]$

allows to satisfy 6 conditions, for example (in normalized time τ)

$\boxed{q(0) = q_0}$ $\boxed{q(1) = q_1}$ $\boxed{q'(0) = v_0 T}$ $\boxed{q'(1) = v_1 T}$ $\boxed{q''(0) = a_0 T^2}$ $\boxed{q''(1) = a_1 T^2}$

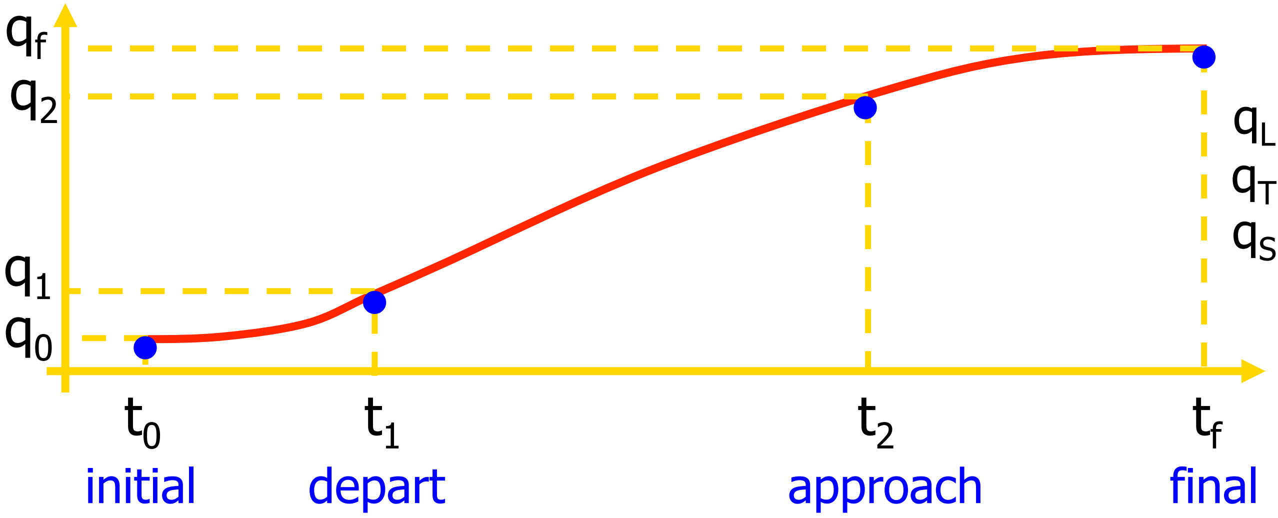
$$\begin{aligned} q(\tau) = {} & (1 - \tau)^3[q_0 + (3q_0+v_0T)\tau + (a_0T^2+6v_0T+12q_0)\tau^2/2] \\ & +\tau^3 [q_1 + (3q_1 -v_1T)(1-\tau) + (a_1T^2 -6v_1T+12q_1) (1 - \tau)^2/2] \end{aligned}$$

special case: $v_0 = v_1 = a_0 = a_1 = 0$

$q(\tau) = q_0 + \Delta q\,[6\tau^5 - 15\tau^4 + 10\tau^3]$ $\Delta q = q_1 - q_0$

4-3-4 polynomials

three phases (Lift off, Travel, Set down) in pick-and-place operations



$$\underbrace{\begin{array}{l} q_L(t) = \text{4th order polynomial} \\ q_T(t) = \text{3rd order polynomial} \\ q_S(t) = \text{4th order polynomial} \end{array}}_{\text{14 coefficients}}$$

boundary conditions

$$q(t_0) = q_0 \quad q(t_1^-) = q(t_1^+) = q_1 \quad q(t_2^-) = q(t_2^+) = q_2 \quad q(t_f) = q_f \;\Big\}\; \text{6 passages}$$

$$\dot{q}(t_0) = \dot{q}(t_f) = 0 \qquad \ddot{q}(t_0) = \ddot{q}(t_f) = 0 \;\Big\}\; \text{4 initial/final velocity/acceleration}$$

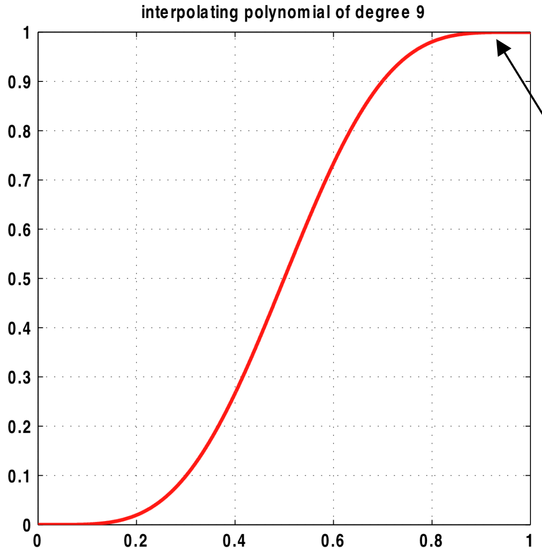
$$\dot{q}(t_i^-) = \dot{q}(t_i^+) \qquad \ddot{q}(t_i^-) = \ddot{q}(t_i^+) \quad i = 1,2 \;\Big\}\; \text{4 continuity}$$

Higher-order polynomials

- a suitable solution class for satisfying symmetric boundary conditions (in a PTP motion) that impose zero values on higher-order derivatives
  - the interpolating polynomial is always of odd degree
  - the coefficients of such (doubly normalized) polynomial are always integers, alternate in sign, sum up to unity, and are zero for all terms up to the power = (degree-1)/2
- in all other cases (e.g., for interpolating a large number N of points), their use is not recommended
  - N-th order polynomials have N-1 maximum and minimum points
  - oscillations arise out of the interpolation points (wandering)

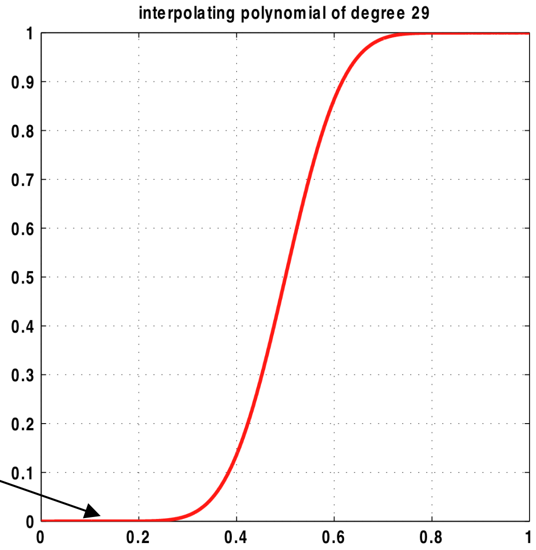
Numerical examples

9th
degree



4 derivatives
are zero

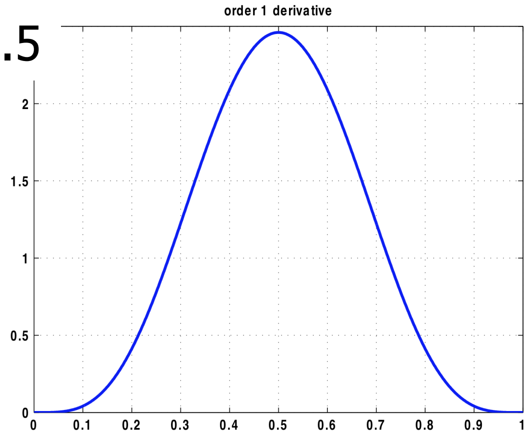
14 derivatives
are zero!



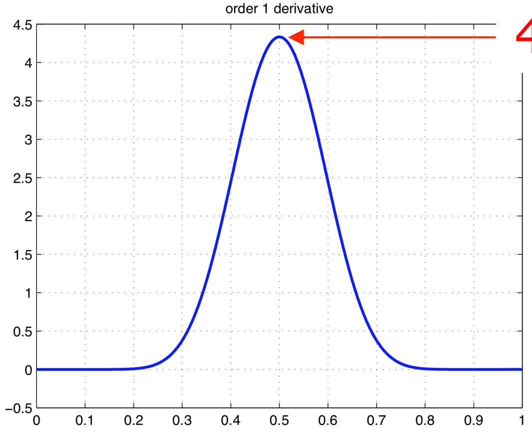
29th
degree

no
overshoot
nor
wandering

2.5



normalized
velocity



4.5!!

velocity
peaking
at midpoint

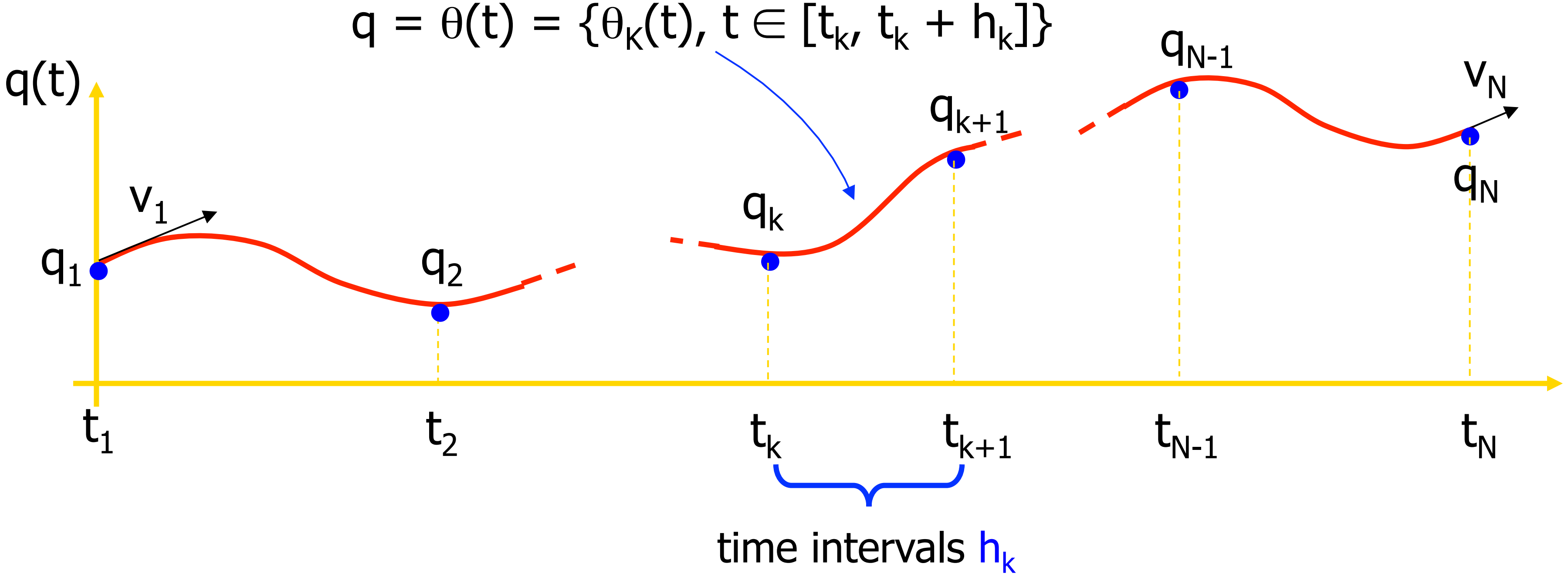
Interpolation using splines

- problem

  interpolate N knots, with continuity up to the second derivative
- solution

  spline: N-1 cubic polynomials, concatenated so as to pass through N knots and being continuous in velocity and acceleration in the N-2 internal knots
- 4(N-1) coefficients
- 4(N-1)-2 conditions, or
  - 2(N-1) of passage (for each cubic, in the two knots at its ends)
  - N-2 of continuity for velocity (at the internal knots)
  - N-2 of continuity for acceleration (at the internal knots)
- 2 free parameters are still left over
  - can be used, e.g., to assign initial and final velocities, v_1 and v_N
- presented next in terms of time t, but similar in terms of space λ

Building a cubic spline



$$\boxed{\theta_K(\tau) = a_{k0} + a_{k1}\,\tau + a_{k2}\,\tau^2 + a_{k3}\,\tau^3} \qquad \tau \in [0, h_k],\ \tau = t - t_k \quad (k = 1, \ldots, N-1)$$

continuity conditions
for velocity and acceleration

$$\boxed{\begin{array}{ll} \dot{\theta}_K(h_k) = \dot{\theta}_{K+1}(0) & \\ & k = 1, \ldots, N-2 \\ \ddot{\theta}_K(h_k) = \ddot{\theta}_{K+1}(0) & \end{array}}$$

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

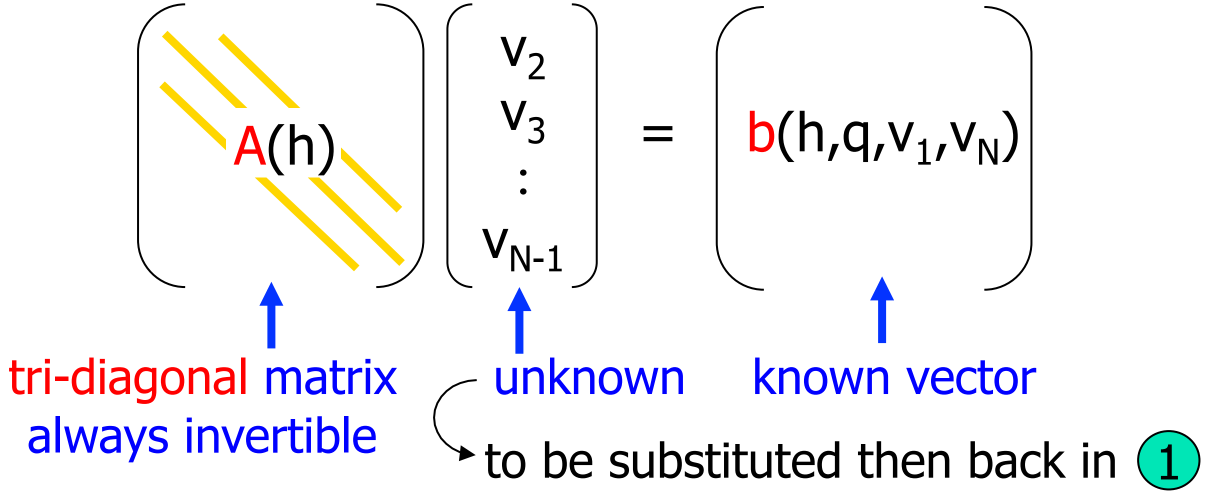
$$\theta_K(0) = q_K = a_{K0} \qquad \begin{pmatrix} h_K^2 & h_K^3 \\ 2h_K & 3h_K^2 \end{pmatrix} \begin{pmatrix} a_{K2} \\ a_{K3} \end{pmatrix} = \begin{pmatrix} q_{K+1} - q_K - v_K h_K \\ v_{K+1} - v_K \end{pmatrix} \quad ①$$

$$\dot{\theta}_K(0) = v_K = a_{K1}$$

2. impose the continuity for accelerations (N-2 conditions)

$$\ddot{\theta}_K(h_k) = 2\, a_{K2} + 6\, a_{K3}\, h_K = \ddot{\theta}_{K+1}(0) = 2\, a_{K+1,2}$$

3. expressing the coefficients a_{k2}, a_{k3}, $a_{k+1,2}$ in terms of the still unknown knot velocities (see step 1.) yields a linear system of equations that is always (easily) solvable



$$\begin{pmatrix} A(h) \end{pmatrix} \begin{pmatrix} v_2 \\ v_3 \\ \vdots \\ v_{N-1} \end{pmatrix} = \begin{pmatrix} b(h,q,v_1,v_N) \end{pmatrix}$$

tri-diagonal matrix always invertible

unknown

known vector

to be substituted then back in ①

Structure of A(h)

$$\begin{pmatrix}
2(h_1+h_2) & h_1 & & & \\
h_3 & 2(h_2+h_3) & h_2 & & \\
& & \ddots & & \\
& & h_{N-2} & 2(h_{N-3}+h_{N-2}) & h_{N-3} \\
& & & h_{N-1} & 2(h_{N-2}+h_{N-1})
\end{pmatrix}$$

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

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

$$
\begin{pmatrix}
\frac{3}{h_1 h_2}[h_1^2(q_3 - q_2) + h_2^2(q_2 - q_1)] - h_2 v_1 \\
\frac{3}{h_2 h_3}[h_2^2(q_4 - q_3) + h_3^2(q_3 - q_2)] \\
\vdots \\
\frac{3}{h_{N-3} h_{N-2}}[h_{N-3}^2(q_{N-1} - q_{N-2}) + h_{N-2}^2(q_{N-2} - q_{N-3})] \\
\frac{3}{h_{N-2} h_{N-1}}[h_{N-2}^2(q_N - q_{N-1}) + h_{N-1}^2(q_{N-1} - q_{N-2})] - h_{N-2} v_N
\end{pmatrix}
$$

Properties of splines

- the spline is the solution with minimum curvature among all interpolating functions having continuous second derivative
- a spline is uniquely determined from the set of data $q_1,\dots,q_N$, $h_1,\dots,h_{N-1}$, v_1, v_N
- the total transfer time is $T = \Sigma h_k = t_N - t_1$
- the time intervals h_k can be chosen so as to minimize T (linear objective function) under (nonlinear) bounds on velocity and acceleration in [0,T]
- for cyclic tasks ($q_1=q_N$), it is preferable to simply impose continuity of velocity and acceleration at $t_1=t_N$ as the "squaring" conditions
    - in fact, even choosing $v_1=v_N$ doesn't guarantee acceleration continuity
    - 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

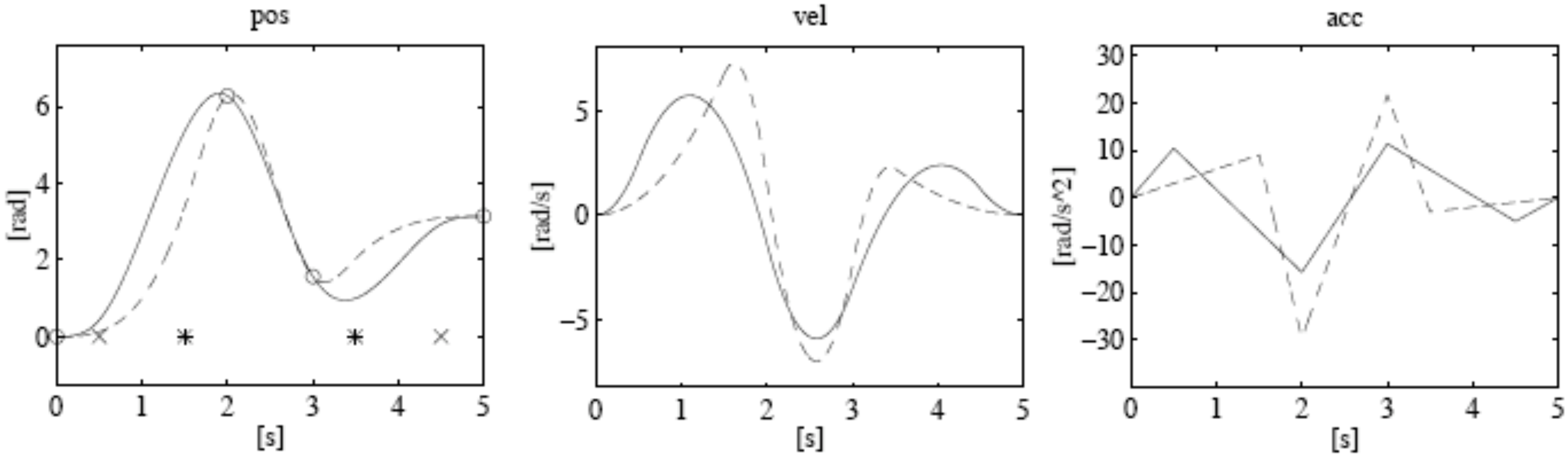
handling assigned initial and final accelerations

- two more parameters are needed in order to impose also the initial acceleration α_1 and final acceleration α_N
- two "fictitious 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

  ⇒ two free parameters are left over (one in the first cubic and the other in the last cubic), which are used to satisfy the boundary conditions 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″