CS65K Robotics

Modelling, Planning and Control

Chapter 4: Trajectory Planning

LECTURE 9: TRAJECTORY PLANNING AND INVERSE JACOBIAN

DR. ERIC CHOU
IEEE SENIOR MEMBER

Objectives

- The difference between path and trajectory is explained
- Techniques for generation of point-to-point motion are presented
- Techniques for generation of motion through a sequence of points are presented
- A technique for automatic scaling of trajectories accounting for dynamic constraints is illustrated

Objectives

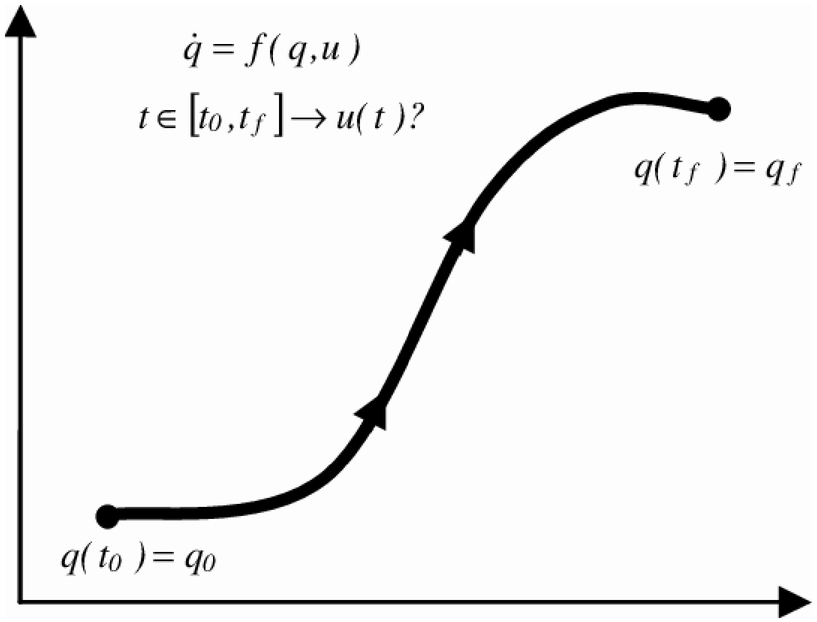
- The path primitive concept is introduced to plan position trajectories
- The angle/axis representation is adopted to plan orientation trajectories

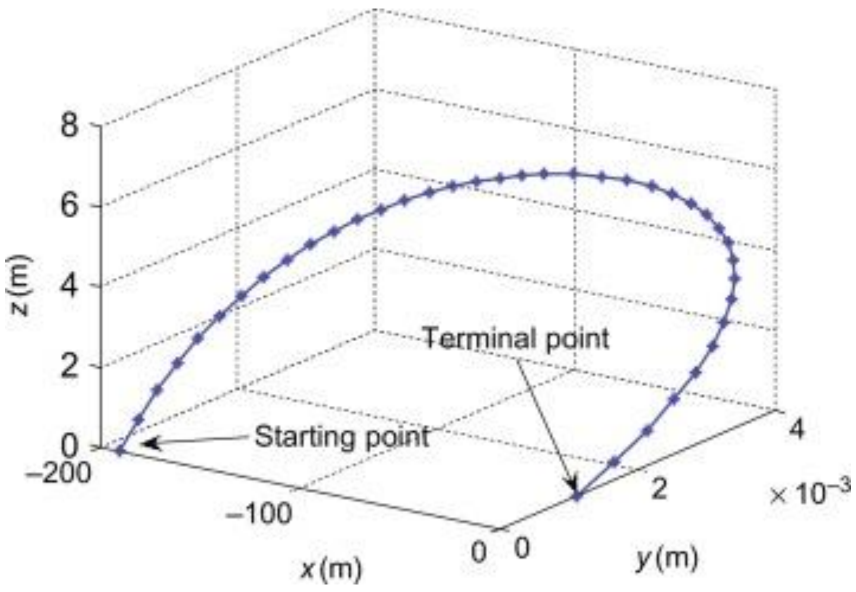
Trajectory Planning

SECTION 1

Trajectory Planning

- Trajectory generation: Figure out the velocity components of the end-effector motion along the path

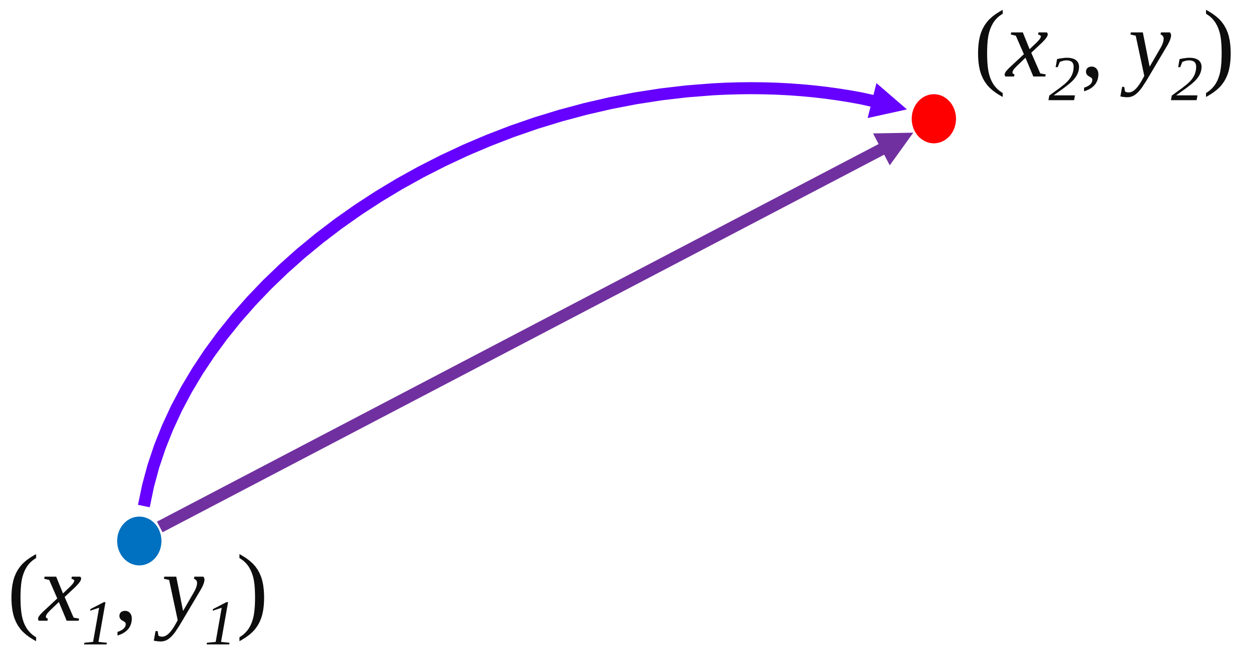




Parametric Equation

SECTION 2

Path Planning

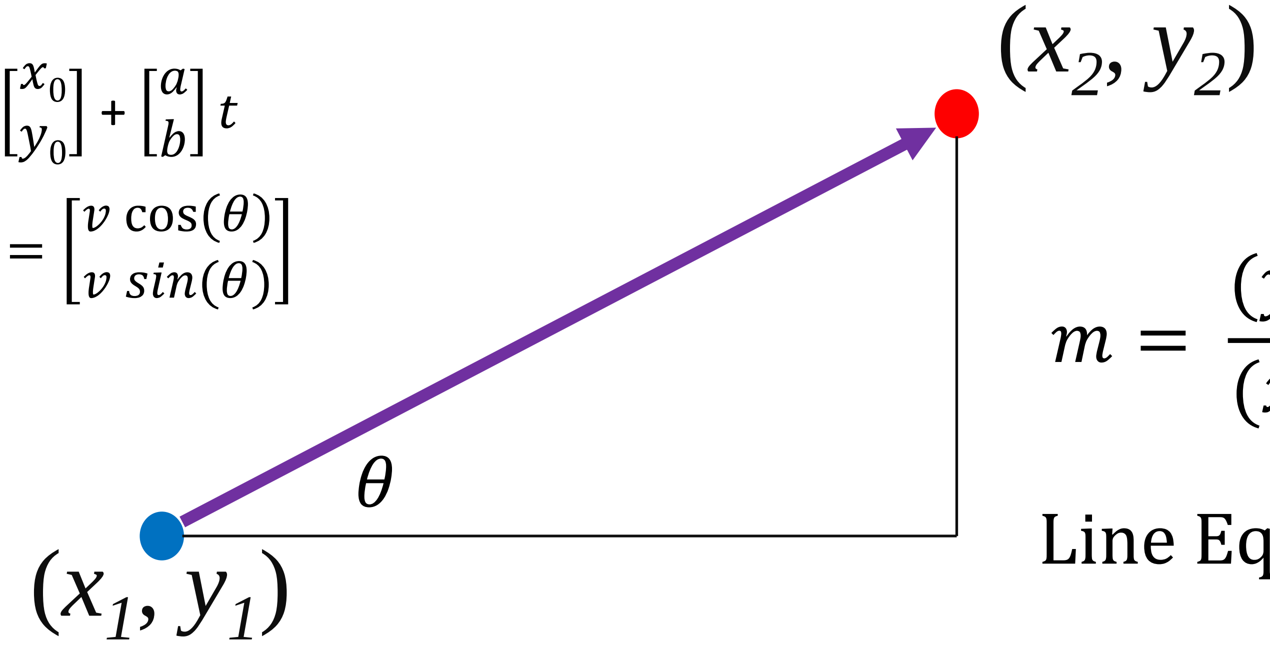


Parameter Equation

$$x(t) = x_0 + a\,t$$
$$y(t) = y_0 + b\,t$$

$$D(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix} t$$

$$\mathrm{v}(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\,sin(\theta) \end{bmatrix}$$

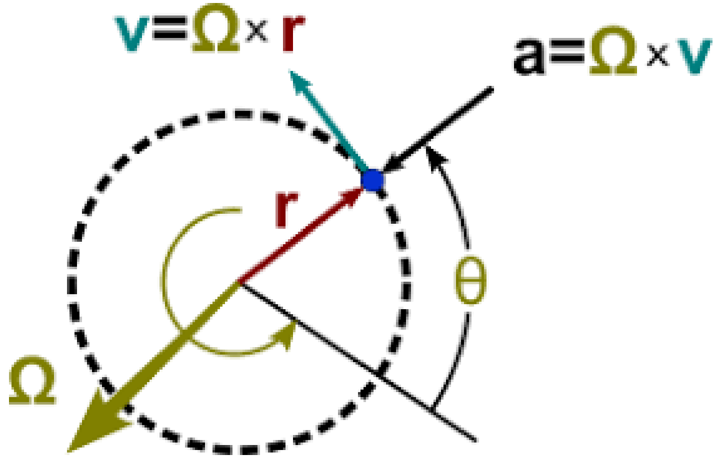


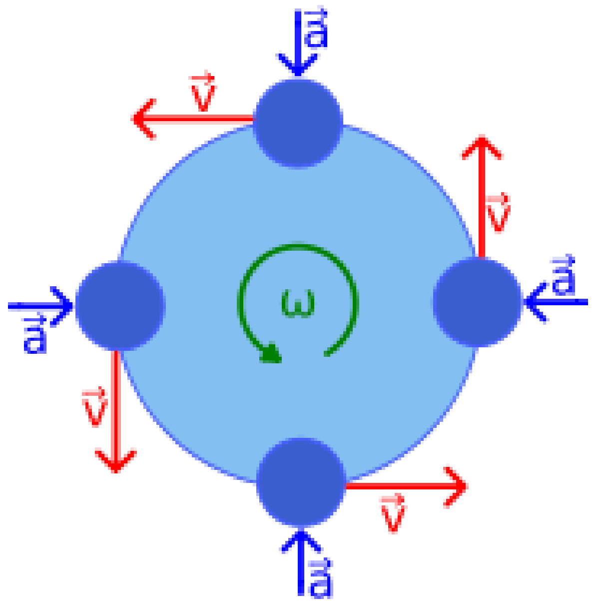
$$m = \frac{(y_2 - y_1)}{(x_2 - x_1)} = \tan(\theta)$$

Line Equation:

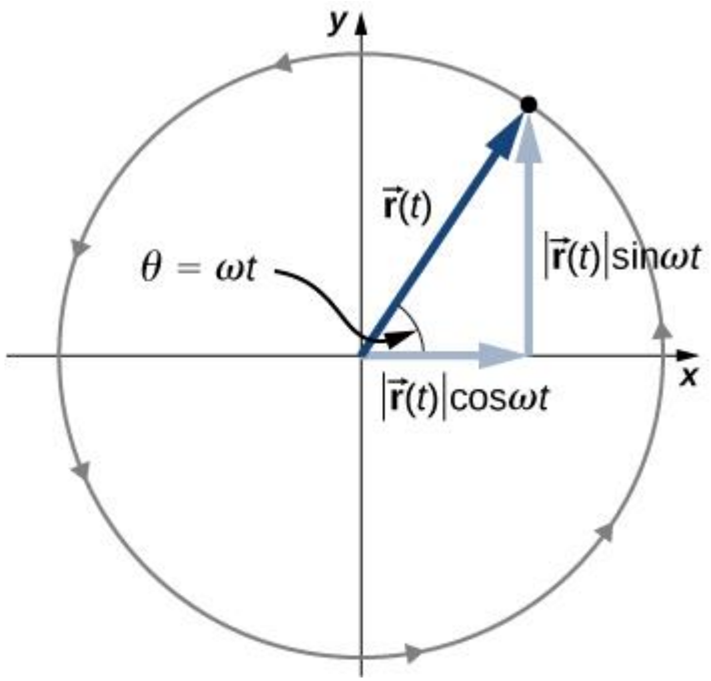
$$m = \frac{(y - y_0)}{(x - x_0)}$$

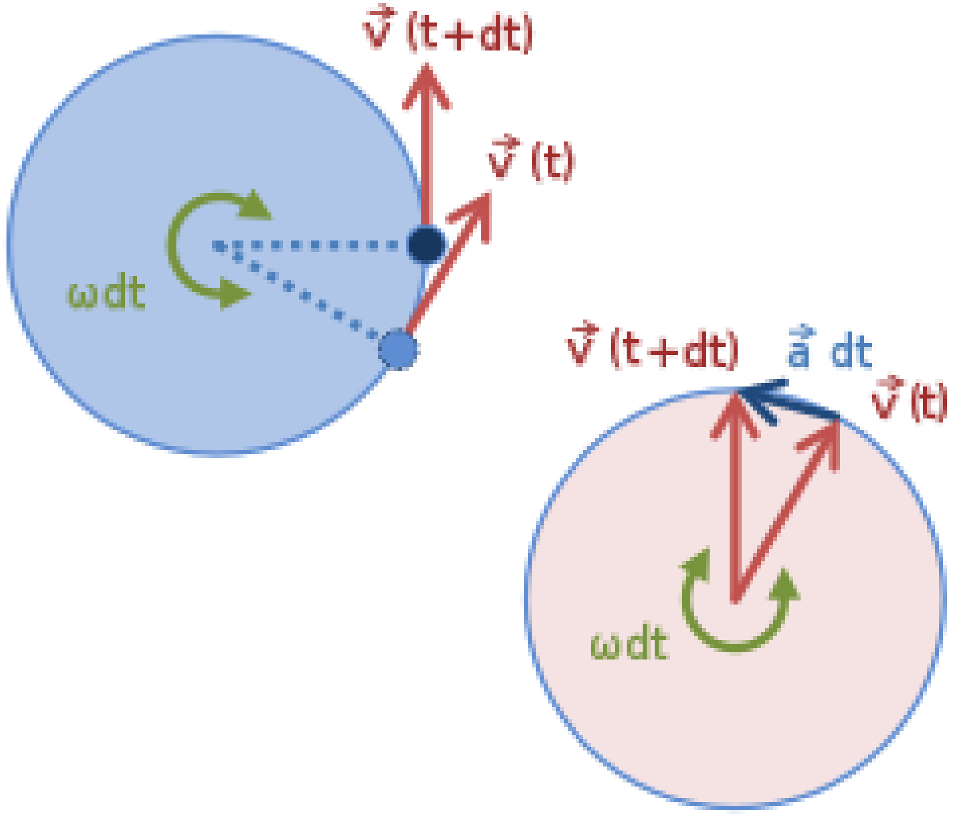
Unit Circle Motion



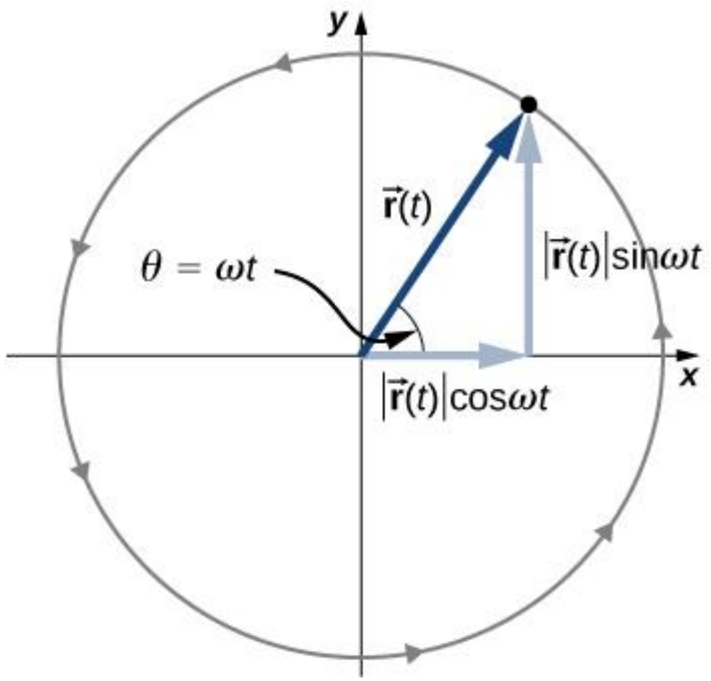


Unit Circle Motion





Unit Circle Motion



$$D(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} r\ cos(\omega t) \\ r\ sin(\omega t) \end{bmatrix}$$

$$v(t) = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \mathrm{r} \begin{bmatrix} -\omega\ sin(\omega t) \\ \omega\ cos(\omega t) \end{bmatrix}$$

Parametric Function

- Workspace variables can be linear or angular $q_i(t)$, for all i , q can be $\theta, or\ d$
- Describe the workspace variables as functions of joint variables and time)

$$X = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \varphi \end{bmatrix} = h(\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix})_{6\times1} = \begin{bmatrix} h_1(q_1, q_2, \cdots, q_6) \\ h_2(q_1, q_2, \cdots, q_6) \\ h_3(q_1, q_2, \cdots, q_6) \\ h_4(q_1, q_2, \cdots, q_6) \\ h_5(q_1, q_2, \cdots, q_6) \\ h_6(q_1, q_2, \cdots, q_6) \end{bmatrix}_{6\times1}$$

Jacobian Matrix

$$\begin{bmatrix} V \\ \Omega \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = J \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \xrightarrow{\dot{X} = J(q)\dot{q}} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} \xleftarrow[\dot{q} = J^{-1}(q)\dot{X}]{} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Joint Space

Task Space

Finding the position variable q (θ or d)

Inverse of Jacobian Matrix

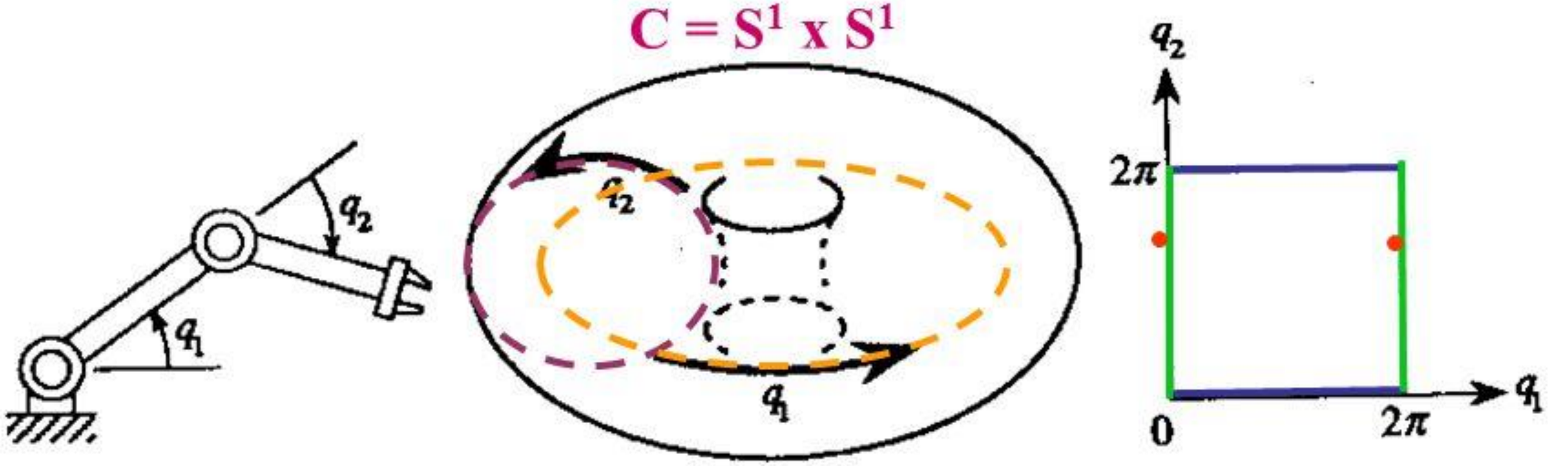
Finding the control functions described in time and joint variables. Given the positions (x, y, z) and angles (ϕ, θ, φ), to find the joint control function as a parametric function of time, or the velocity functions related to these joint functions.

Configuration Space

SECTION 3

Configuration Space of a Robot

- Space of all its possible configurations
- But the topology of this space is usually not that of a Cartesian space



Configuration **space**

- Robot **configuration** is described by a vector of **generalised** joint coordinates
- Each coordinate can be:
  - ➥ an angle, for a rotational (**revolute**) joint
  - ➥ a length, for a sliding (**prismatic**) joint

Joint configuration

Joint coordinate

Number of joints

Space of all possible configurations

$$\boldsymbol{q} = \{q_j, j \in [1 \cdots N]\} \in \mathcal{C}$$

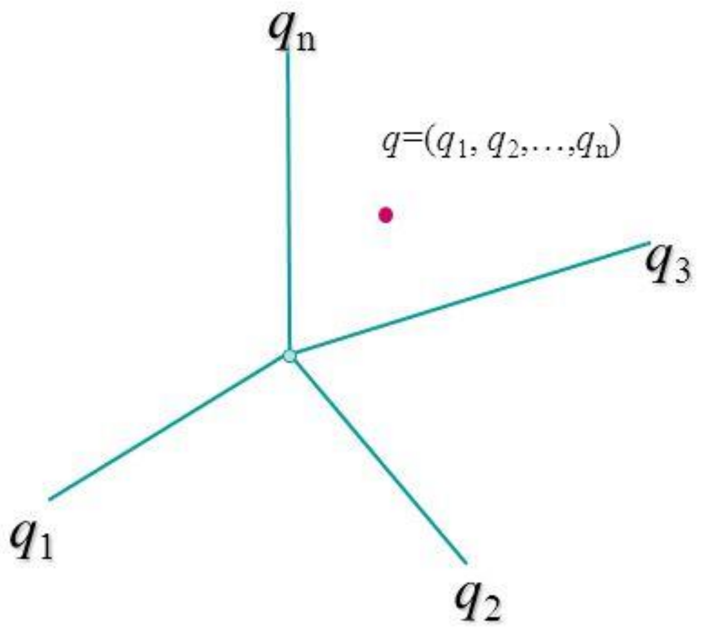
Configuration space $\mathcal{C} \subset \mathbb{R}^N$

Definitions

- **Configuration:**
  - Specification of all the variables that define the system completely
  - Example: Configuration of a $\boldsymbol{n}$ DOF robot is $q = (q_0, q_1, \dots q_{n-1})$
- **Configuration space (C-space):**
  - Set of all configurations
- **Free configuration:**
  - A configuration q that does not collide with obstacles
- **Free space (F) :**
  - Set of all free configurations
  - It is a subset of C

Configuration space

- The **configuration space** C is the set of all possible configurations.
  - A configuration is a point in C.
  - Similar to a
    - State space
    - Parameter space
- The **workspace** is all points reachable by the robot (or sometimes just the end effector)
- C can be very high dimensional while the workspace is just 2D or 3D



Path and Trajectory

SECTION 4

Path v/s Trajectory

- **Path:**
  - A sequence of robot configurations in a particular order without regard to the timing of these configurations.
- **Trajectory:**
  - It concerned about when each part of the path must be attained, thus specifying timing.

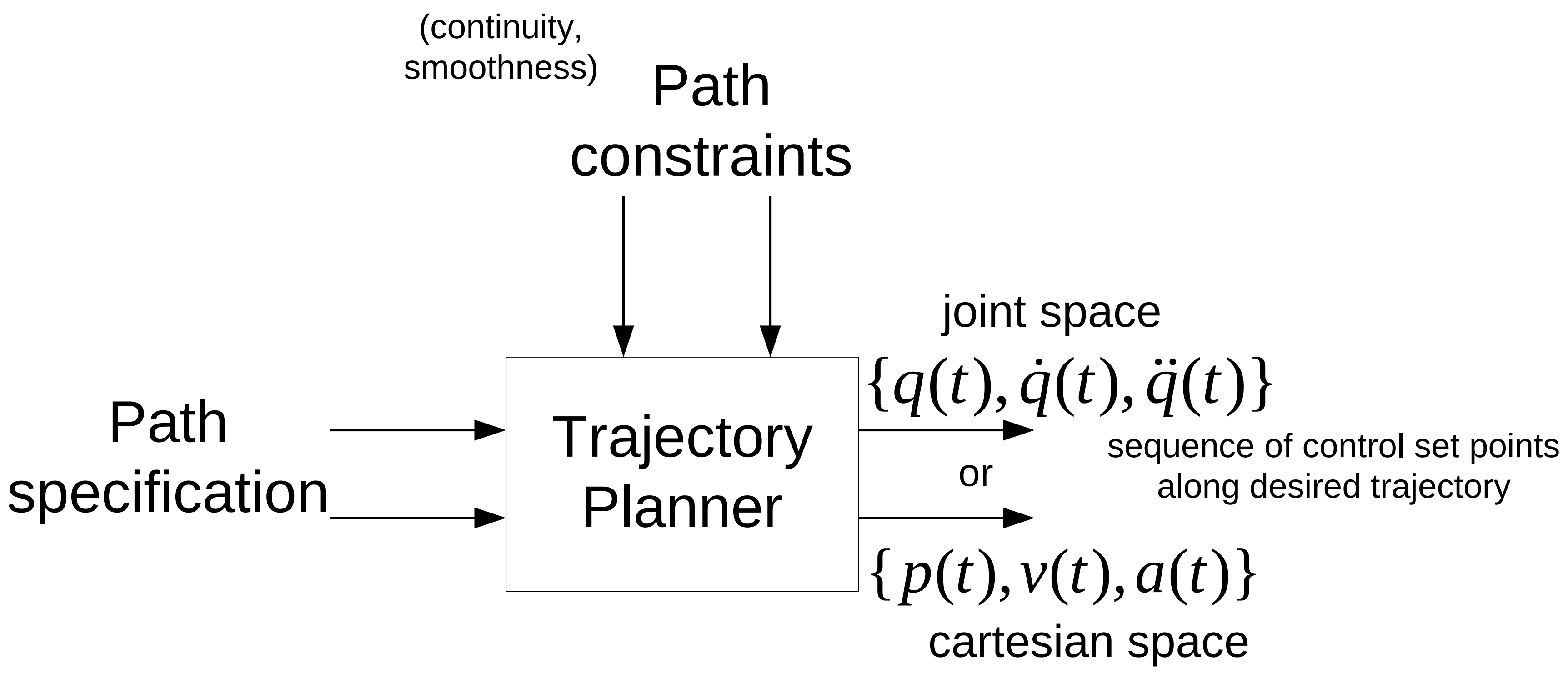
Path Planning

- **Problem statement:**
  - Compute a collision-free path for a rigid or articulated moving object among static obstacles.
- **Input**
  - Geometry of a moving object (a robot, a digital actor, or a molecule) and obstacles
  - How does the robot move?
  - Kinematics of the robot (degrees of freedom)
  - Initial and goal robot configurations (positions & orientations)
- **Output**
  - Continuous sequence of collision-free robot configurations connecting the initial and goal configurations

Trajectory Planning

- **Problem statement**
  - Turn a specified Cartesian-space trajectory of Pe into appropriate joint position reference values
- **Input**
  - Cartesian space path
  - Path constraints including velocity and acceleration limits and singularity analysis.
- **Output**
  - a series of joint position/velocity reference values to send to the controller

Trajectory Planning



Trajectory planning algorithm

- **Inputs**

Path description

Path constraints

Constraints imposed by manipulator dynamics

- **Output**

Joint (end-effector) trajectories in terms of a time sequence of the values attained by position, velocity and acceleration

Path and Trajectory

Reduced number of parameters

- Path
- Extremal points
- Possible intermediate points
- Geometric primitives interpolating the points
- Timing law
- Total trajectory time
- Velocity and/or acceleration at given points

Path and Trajectory

Trajectory planning in the operational space

- Natural task description
- Path constraints
- Singularities
- Redundancy

Trajectory planning in the joint space

- Inverse kinematics
- Control action

Joint Space Trajectories

SECTION 5

Joint Space Trajectories

Generation of a function $q(t)$ interpolating the given vectors of joint variables at each point, in respect of the imposed constraints

- Generated trajectories not very demanding from a computational viewpoint
- Joint positions and velocities (and accelerations) as continuous functions of time
- Undesirable effects minimized (non-smooth trajectories)

Point-to-point motion

- Extremal points and total time

Motion through a sequence of points

- Extremal points, intermediate points and transition times

Point-to-Point Motion

Generation of $q(t)$ to move from q_i to q_f in a time t_f

- Cubic polynomial

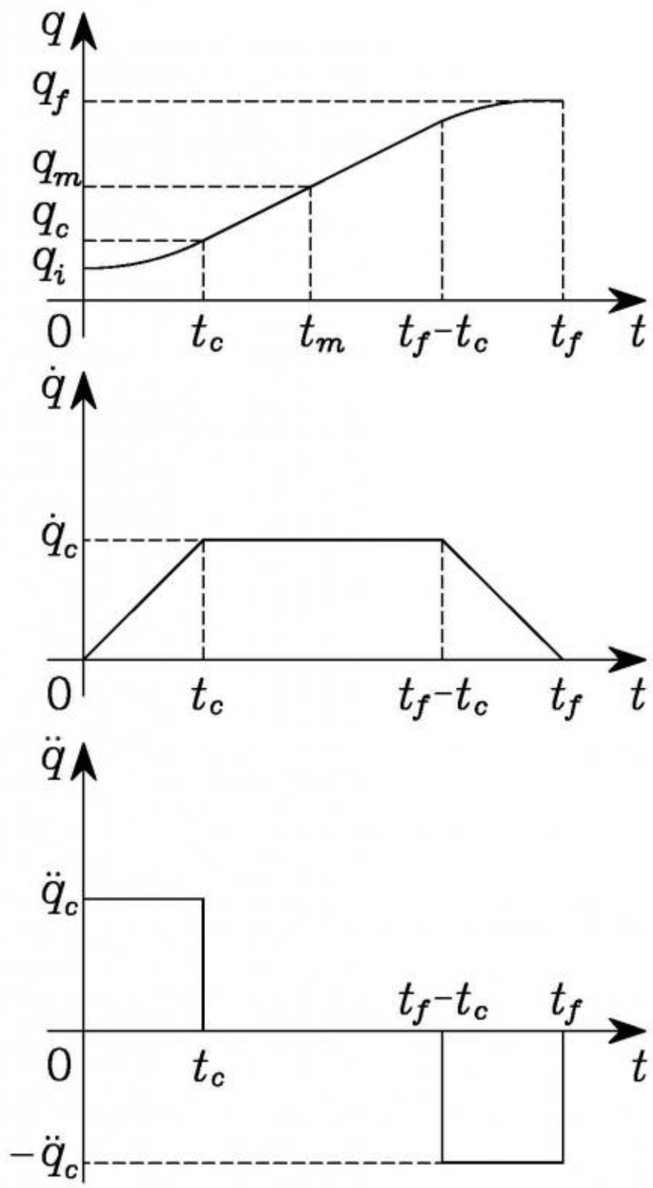
$$q(t) = a_3 t^3 + a_2 t^2 + a_1 t + a_0$$
$$\dot{q}(t) = 3a_3 t^2 + 2a_2 t + a_1$$
$$\ddot{q}(t) = 6a_3 t + 2a_2$$

Computation of coefficients

$$a_0 = q_i$$
$$a_1 = \dot{q}_i$$
$$a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0 = q_f$$
$$3a_3 t_f^2 + 2a_2 t_f + a_1 = \dot{q}_f$$

- Quintic polynomial

$$q(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$



Trapezoidal Velocity Profile

$$\ddot{q}_c t_c = \frac{q_m - q_c}{t_m - t_c}$$

$$q_c = q_i + \frac{1}{2}\ddot{q}_c t_c^2$$

$$\ddot{q}_c t_c^2 - \ddot{q}_c t_f t_c + q_f - q_i = 0$$

Trapezoidal Velocity Profile

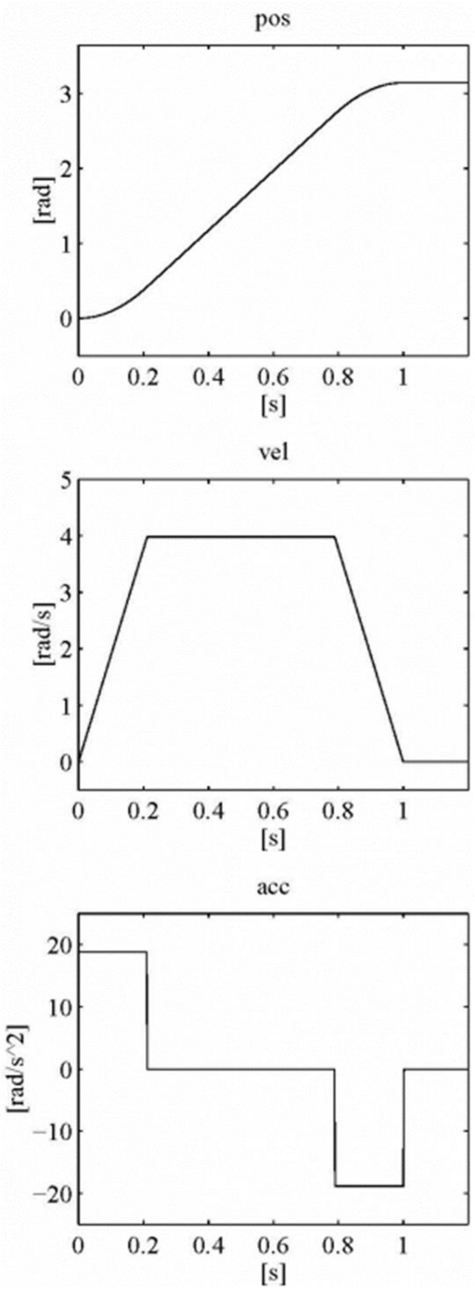
- $\ddot{q}_c$ specified ($\operatorname{sgn} \ddot{q}_c = \operatorname{sgn}(q_f - q_i)$)

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

$$|\ddot{q}_c| \geq \frac{4|q_f - q_i|}{t_f^2}$$

Trajectory

$$q(t) = \begin{cases} q_i + \frac{1}{2}\ddot{q}_c t^2 & 0 \leq t \leq t_c \\ q_i + \ddot{q}_c t_c (t - t_c/2) & t_c < t \leq t_f - t_c \\ q_f - \frac{1}{2}\ddot{q}_c (t_f - t)^2 & t_f - t_c < t \leq t_f \end{cases}$$



Time history of position, velocity and acceleration with a trapezoidal velocity profile timing law

Trapezoidal Velocity Profile III

- $\dot{q}_c$ specified

$$\frac{|q_f - q_i|}{t_f} < |\dot{q}_c| \leq \frac{2|q_f - q_i|}{t_f}$$

$$t_c = \frac{q_i - q_f + \dot{q}_c t_f}{\dot{q}_c}$$

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

Motion Through a Sequence of Points

SECTION 6

Motion Through a Sequence of Points

Opportunity to specify intermediate points (*sequence of points*)

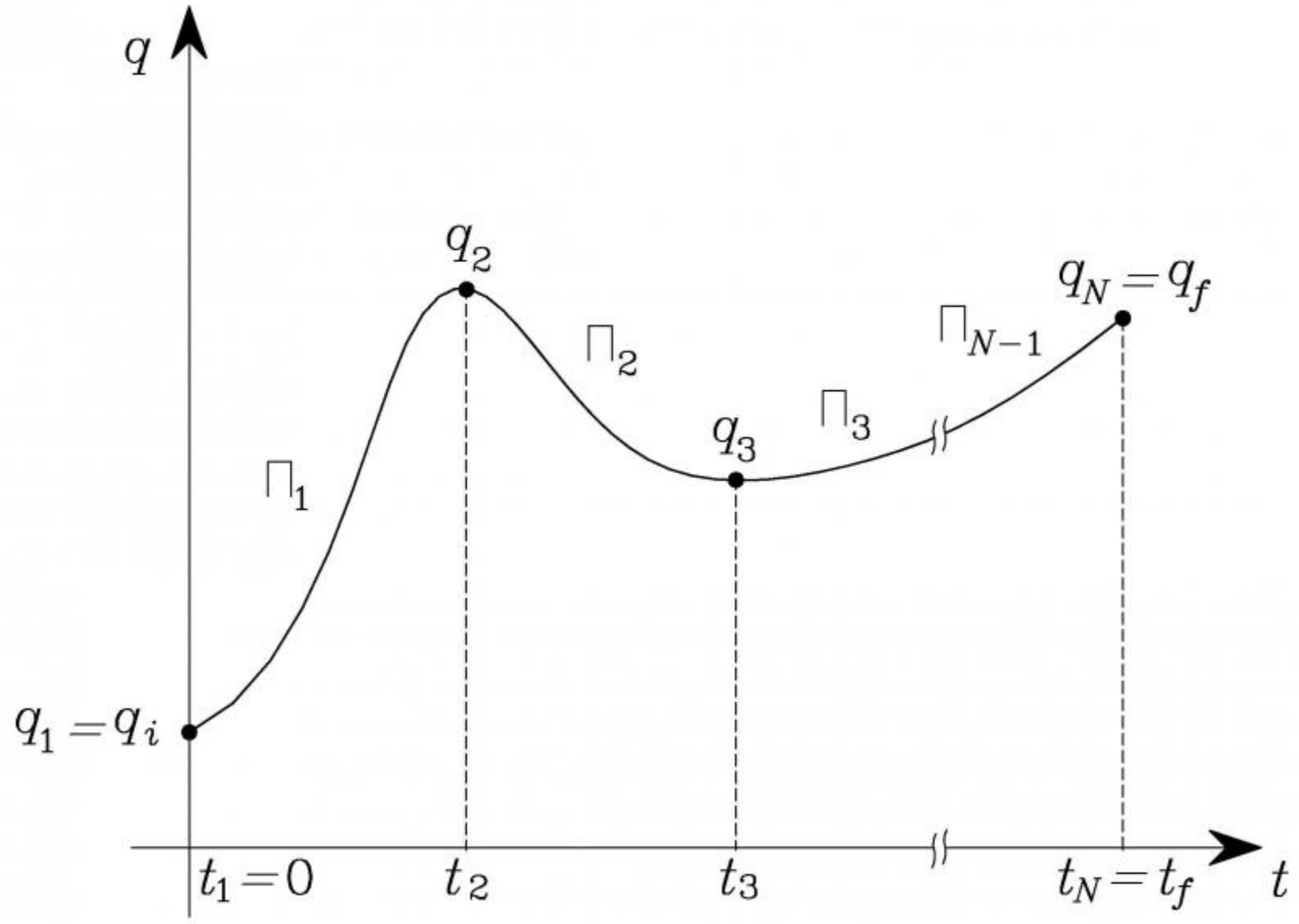
Given N path points, find an interpolating function across these points

- $(N\text{-}1)$-order polynomial
  - It is not possible to assign the initial and final velocities
  - As the order of a polynomial increases its oscillatory behavior increases (non-smooth trajectories)
  - Numerical accuracy for computation of polynomial coefficients decreases as order increases
  - The resulting system of constraint equations is heavy to solve
  - Polynomial coefficients depend on all the assigned points $\Rightarrow$ if it is desired to change a point, all of them have to be recomputed

Motion Through a Sequence of Points II

Sequence of low-order interpolating polynomials continuous at path points
- Arbitrary values of $\dot{q}(t)$ are imposed at path points
- The values of $\dot{q}(t)$ at path points are assigned according to a certain criterion
- The acceleration $\ddot{q}(t)$ has to be continuous at path points

Sequence of interpolating polynomials of order less than three which determine trajectories passing nearby path points at given instants of time



Characterization of a trajectory on a given path obtained through interpolating polynomials

Interpolating Polynomials with Imposed Velocities at Path Points

SECTION 7

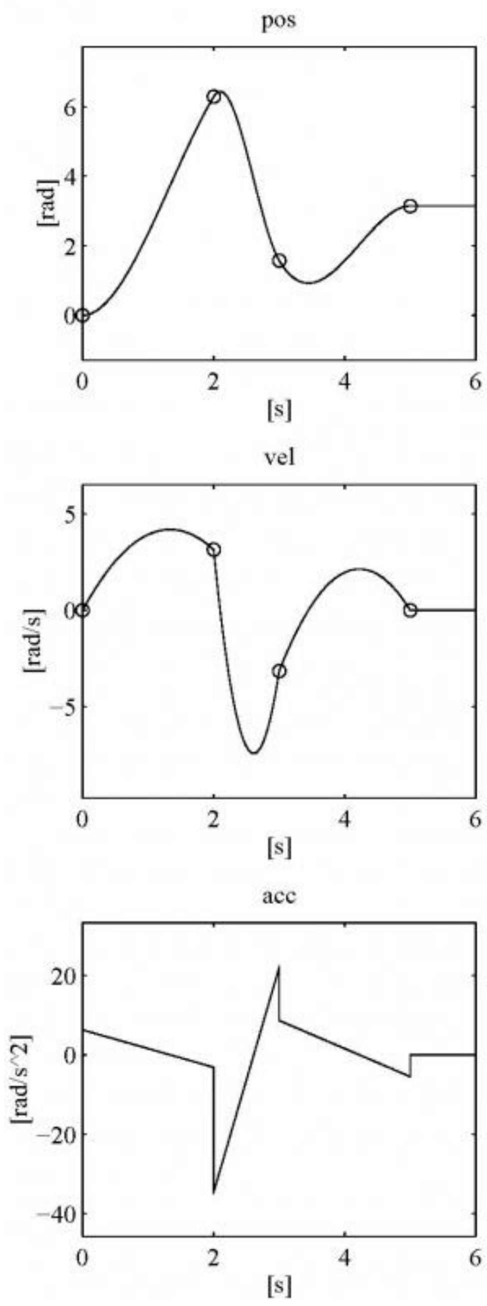
Interpolating Polynomials with Imposed Velocities at Path Points

$$\begin{aligned} \Pi_k(t_k) &= q_k \\ \Pi_k(t_{k+1}) &= q_{k+1} \\ \dot{\Pi}_k(t_k) &= \dot{q}_k \\ \dot{\Pi}_k(t_{k+1}) &= \dot{q}_{k+1} \end{aligned}$$

- Continuity of velocity at path points

$$\dot{\Pi}_k(t_{k+1}) = \dot{\Pi}_{k+1}(t_{k+1})$$

Time history of position, velocity and acceleration with a timing law of interpolating polynomials with velocity constraints at path points



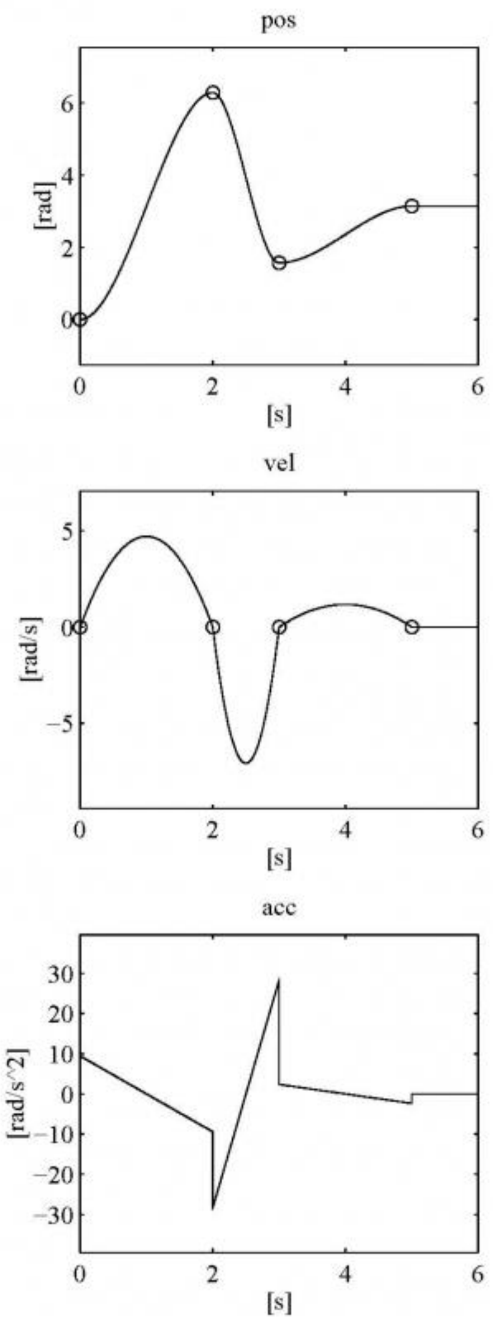
Interpolating Polynomials with Computed Velocities at Path Points

$$\dot{q}_1 = 0$$

$$\dot{q}_k = \begin{cases} 0 & \operatorname{sgn}(v_k) \neq \operatorname{sgn}(v_{k+1}) \\ \frac{1}{2}(v_k + v_{k+1}) & \operatorname{sgn}(v_k) = \operatorname{sgn}(v_{k+1}) \end{cases}$$

$$\dot{q}_N = 0$$

$$v_k = (q_k - q_{k-1})/(t_k - t_{k-1})$$



Time history of position, velocity and acceleration with a timing law of interpolating polynomials with computed velocities at path points

Interpolating Polynomials with Continuous Accelerations at Path Points (Splines)

$$\Pi_{k-1}(t_k) = q_k$$
$$\Pi_{k-1}(t_k) = \Pi_k(t_k)$$
$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k)$$
$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k)$$

- $4N-2$ equations in $4(N\text{-}1)$ unknowns (fourth-order polynomials for first and last segment?)
- *2 virtual points* (continuity on position, velocity and acceleration) $\Longrightarrow N+1$ cubic polynomials

Splines

SECTION 8

Splines II

- $4\,(N\text{-}2)$ equations for $N\text{-}2$ intermediate points

$$\Pi_{k-1}(t_k) = q_k$$
$$\Pi_{k-1}(t_k) = \Pi_k(t_k)$$
$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k)$$
$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k)$$

Splines II

- 6 equations for the initial and final points

$$\Pi_1(t_1) = q_i$$
$$\dot{\Pi}_1(t_1) = \dot{q}_i$$
$$\ddot{\Pi}_1(t_1) = \ddot{q}_i$$

$$\Pi_{N+1}(t_{N+2}) = q_f$$
$$\dot{\Pi}_{N+1}(t_{N+2}) = \dot{q}_f$$
$$\ddot{\Pi}_{N+1}(t_{N+2}) = \ddot{q}_f$$

Splines II

- 6 equations for the virtual points

$$\Pi_{k-1}(t_k) = \Pi_k(t_k)$$
$$\dot{\Pi}_{k-1}(t_k) = \dot{\Pi}_k(t_k)$$
$$\ddot{\Pi}_{k-1}(t_k) = \ddot{\Pi}_k(t_k)$$

$$\Downarrow$$

- System of $4(N+1)$ equations in $4(N+1)$ unknowns (coefficients of the $N+1$ cubic polynomials

Splines III

- Computationally efficient algorithm

$$\ddot{\Pi}_k(t) = \frac{\ddot{\Pi}_k(t_k)}{\Delta t_k}(t_{k+1} - t) + \frac{\ddot{\Pi}_k(t_{k+1})}{\Delta t_k}(t - t_k) \qquad k = 1, \ldots, N+1$$

$$\begin{aligned}\Pi_k(t) &= \frac{\ddot{\Pi}_k(t_k)}{6\Delta t_k}(t_{k+1} - t)^3 + \frac{\ddot{\Pi}_k(t_{k+1})}{6\Delta t_k}(t - t_k)^3 \\ &\quad + \left(\frac{\Pi_k(t_{k+1})}{\Delta t_k} - \frac{\Delta t_k \ddot{\Pi}_k(t_{k+1})}{6}\right)(t - t_k) \\ &\quad + \left(\frac{\Pi_k(t_k)}{\Delta t_k} - \frac{\Delta t_k \ddot{\Pi}_k(t_k)}{6}\right)(t_{k+1} - t) \qquad k = 1, \ldots, N+1,\end{aligned}$$

- 4 unknowns: $\Pi_k(t_k),\ \Pi_k(t_{k+1}),\ \ddot{\Pi}_k(t_k),\ \ddot{\Pi}_k(t_{k+1})$

Splines IV

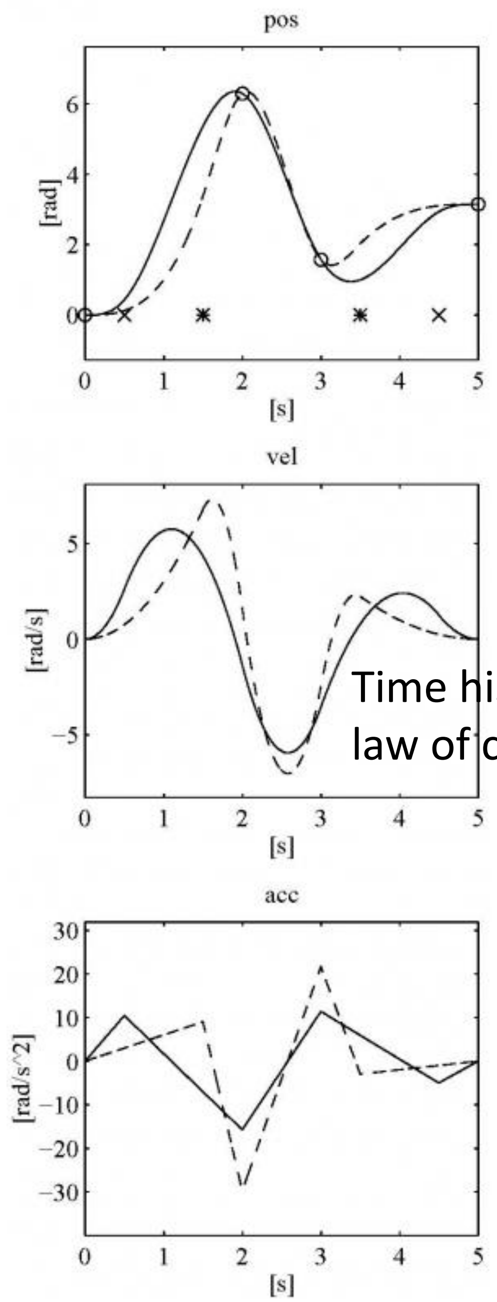
- N variables q_k for $k \neq 2, N+1$ given
- Continuity on q_2 and q_N+1
- Continuity on q_k for $k = 3, \cdots, N$
- $\dot{q}_i$ and $\dot{q}_f$ given
- Continuity on $\ddot{q}_k$ for $k = 2, \cdots, N+1$
- $\ddot{q}_i$ and $\ddot{q}_f$ given $\Longrightarrow$

$$\begin{aligned} \dot{\Pi}_1(t_2) &= \dot{\Pi}_2(t_2) \\ &\vdots \\ \dot{\Pi}_N(t_{N+1}) &= \dot{\Pi}_{N+1}(t_{N+1}) \end{aligned}$$

- System of linear equations

$$\boldsymbol{A} \begin{bmatrix} \ddot{\Pi}_2(t_2) & \dots & \ddot{\Pi}_{N+1}(t_{N+1}) \end{bmatrix}^T = \boldsymbol{b}$$

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & 0 & 0 \\ a_{21} & a_{22} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & \dots & a_{N,N-1} & a_{NN} \end{bmatrix}$$



Time history of position, velocity and acceleration with a timing law of cubic splines for two different pairs of virtual points

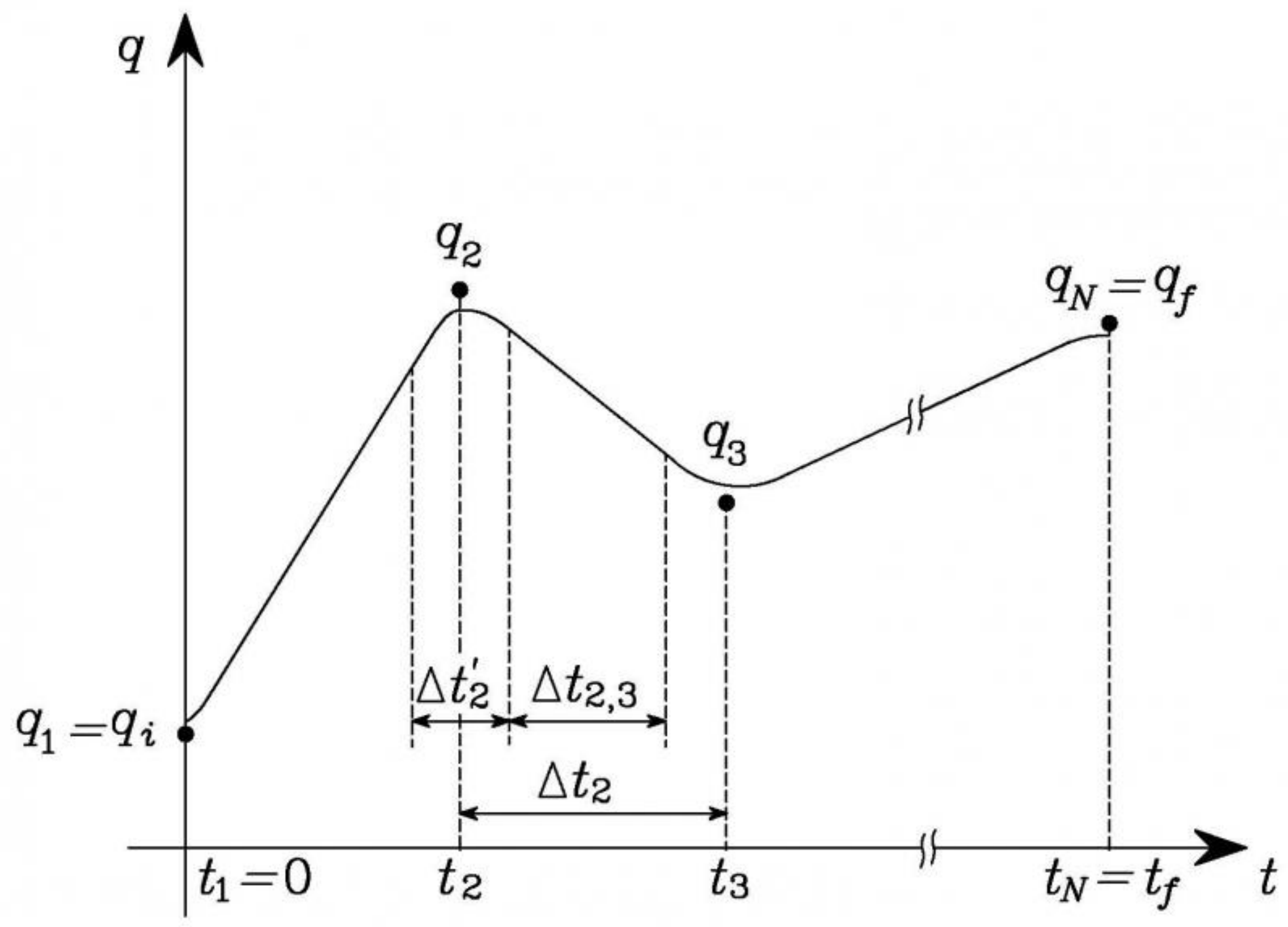
Interpolating Linear Polynomials With Parabolic Blends

SECTION 9

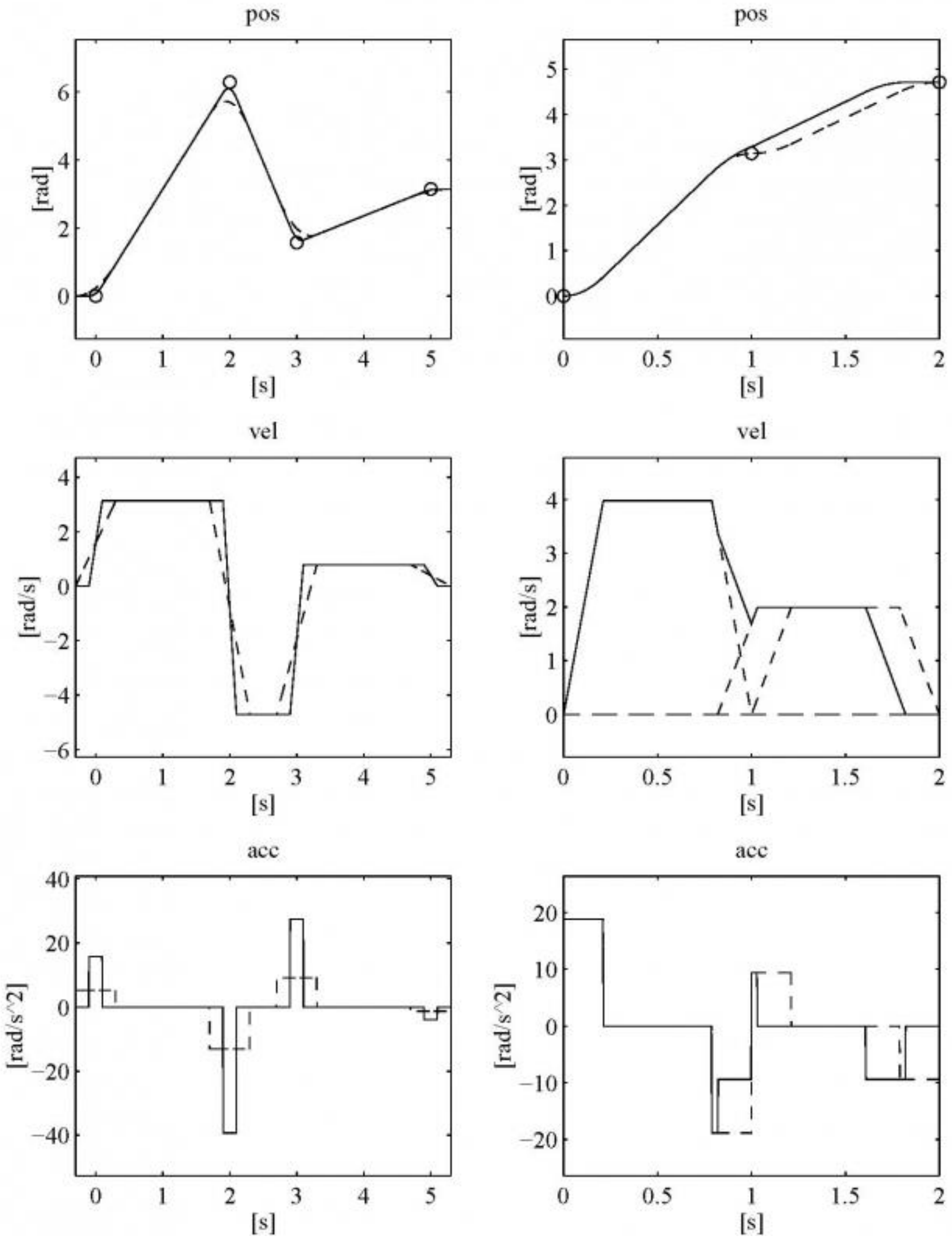
Interpolating Linear Polynomials With Parabolic Blends

$$\dot{q}_{k-1,k} = \frac{q_k - q_{k-1}}{\Delta t_{k-1}}$$

$$\ddot{q}_k = \frac{\dot{q}_{k,k+1} - \dot{q}_{k-1,k}}{\Delta t'_k}$$



Characterization of a trajectory with interpolating linear polynomials with parabolic blends



Left: Time history of position, velocity and acceleration with a timing law of interpolating linear polynomials with parabolic blends. Right: Time history of position, velocity and acceleration with a timing law of interpolating linear polynomials with parabolic blends obtained by anticipating the generation of the second segment of trajectory

Dynamic Scaling of Trajectories

SECTION 10

Dynamic Scaling of Trajectories

- A technique for trajectory dynamic scaling is introduced, which adapts trajectory planning to the dynamic characteristics of the manipulator

$$\begin{aligned}\boldsymbol{\tau}(t) &= \boldsymbol{B}(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \boldsymbol{C}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t))\dot{\boldsymbol{q}}(t) + \boldsymbol{g}(\boldsymbol{q}(t)) \\ &= \boldsymbol{B}(\boldsymbol{q}(t))\ddot{\boldsymbol{q}}(t) + \boldsymbol{\Gamma}(\boldsymbol{q}(t))[\dot{\boldsymbol{q}}(t)\dot{\boldsymbol{q}}(t)] + \boldsymbol{g}(\boldsymbol{q}(t)) \\ &= \boldsymbol{\tau}_s(t) + \boldsymbol{g}(\boldsymbol{q}(t))\end{aligned}$$

$$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} = \boldsymbol{\Gamma}(\boldsymbol{q})[\dot{\boldsymbol{q}}\dot{\boldsymbol{q}}]$$

$$[\dot{\boldsymbol{q}}\dot{\boldsymbol{q}}] = \begin{bmatrix} \dot{q}_1^2 & \dot{q}_1\dot{q}_2 & \dots & \dot{q}_{n-1}\dot{q}_n & \dot{q}_n^2 \end{bmatrix}^T$$

Dynamic Scaling of Trajectories II

- Time scaling $r(t): \quad r(0) = 0 \quad r(t_f) = \bar{t}_f$

$$\boldsymbol{q}(t) = \bar{\boldsymbol{q}}(t)$$

$$\dot{\boldsymbol{q}} = \dot{r}\bar{\boldsymbol{q}}'(r)$$

$$\ddot{\boldsymbol{q}} = \dot{r}^2\bar{\boldsymbol{q}}''(r) + \ddot{r}\bar{\boldsymbol{q}}'(r)$$

$$\Downarrow$$

$$\boldsymbol{\tau} = \dot{r}^2\Big(B(\bar{\boldsymbol{q}}(r))\bar{\boldsymbol{q}}''(r) + \boldsymbol{\Gamma}(\bar{\boldsymbol{q}}(r))[\bar{\boldsymbol{q}}'(r)\bar{\boldsymbol{q}}'(r)]\Big) + \ddot{r}\boldsymbol{B}(\bar{\boldsymbol{q}}(r))\bar{\boldsymbol{q}}'(r) + \boldsymbol{g}(\bar{\boldsymbol{q}}(r))$$

$$= \boldsymbol{\tau}_s(t) + \boldsymbol{g}(\bar{\boldsymbol{q}}(r))$$

$$\bar{\boldsymbol{\tau}}_s(r) = \boldsymbol{B}(\bar{\boldsymbol{q}}(r))\bar{\boldsymbol{q}}''(r) + \Gamma(\bar{\boldsymbol{q}}(r))[\bar{\boldsymbol{q}}'(r)\bar{\boldsymbol{q}}'(r)]$$

$$\Downarrow$$

$$\boldsymbol{\tau}_s(t) = \dot{r}^2\Big(\boldsymbol{B}(\bar{\boldsymbol{q}}(r))\bar{\boldsymbol{q}}''(r) + \boldsymbol{\Gamma}(\bar{\boldsymbol{q}}(r))[\bar{\boldsymbol{q}}'(r)\bar{\boldsymbol{q}}'(r)]\Big) + \boldsymbol{B}(\bar{\boldsymbol{q}}(r))\bar{\boldsymbol{q}}'(r)$$

Dynamic Scaling of Trajectories II

- Simple choice $r(t) = ct$

$$\boldsymbol{\tau}_s(t) = c^2 \bar{\boldsymbol{\tau}}_s(ct)$$

Joint q_i corresponding to the largest violation:

$$\frac{|\tau_s|}{|\bar{\tau}_i - g(q_i)|} = c^2$$

Path Primitives

SECTION 11

Path Primitives

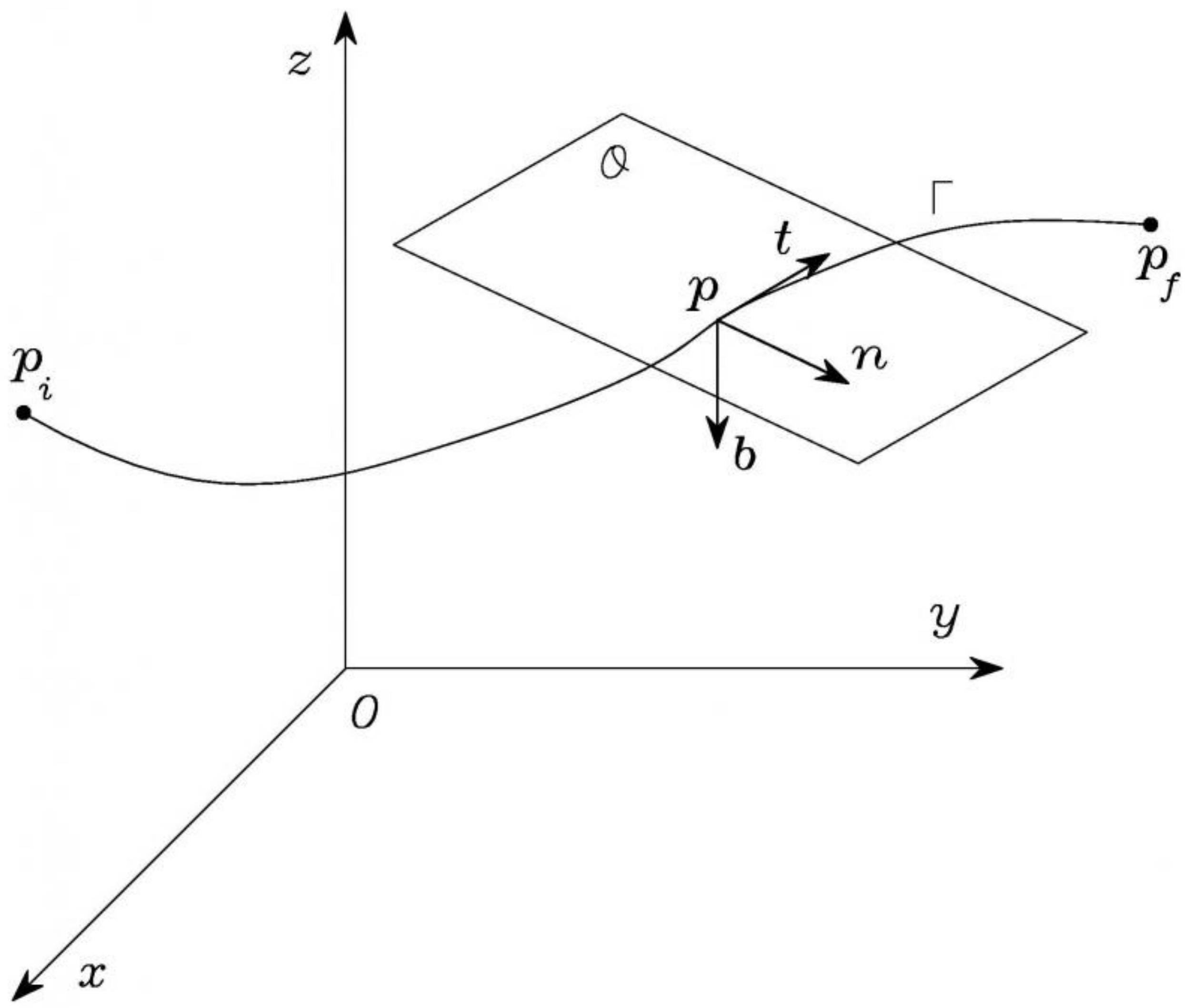
- Parametric description of path in space

$$\boldsymbol{p} = \boldsymbol{f}(\sigma)$$

$$\boldsymbol{t} = \frac{d\boldsymbol{p}}{ds}$$

$$\boldsymbol{n} = \frac{1}{\left\| \frac{d^2\boldsymbol{p}}{ds^2} \right\|} \frac{d^2\boldsymbol{p}}{ds^2}$$

$$\boldsymbol{b} = \boldsymbol{t} \times \boldsymbol{n}$$



Parametric representation of a path in space

Position Trajectories

- Rectilinear path

$$\boldsymbol{p}_e(s) = p_i + \frac{s}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i)$$

$$\dot{\boldsymbol{p}}_e = \frac{\dot{s}}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) = \dot{s}\boldsymbol{t}$$

$$\ddot{\boldsymbol{p}}_e = \frac{\ddot{s}}{\|\boldsymbol{p}_f - \boldsymbol{p}_i\|}(\boldsymbol{p}_f - \boldsymbol{p}_i) = \ddot{s}\boldsymbol{t}$$

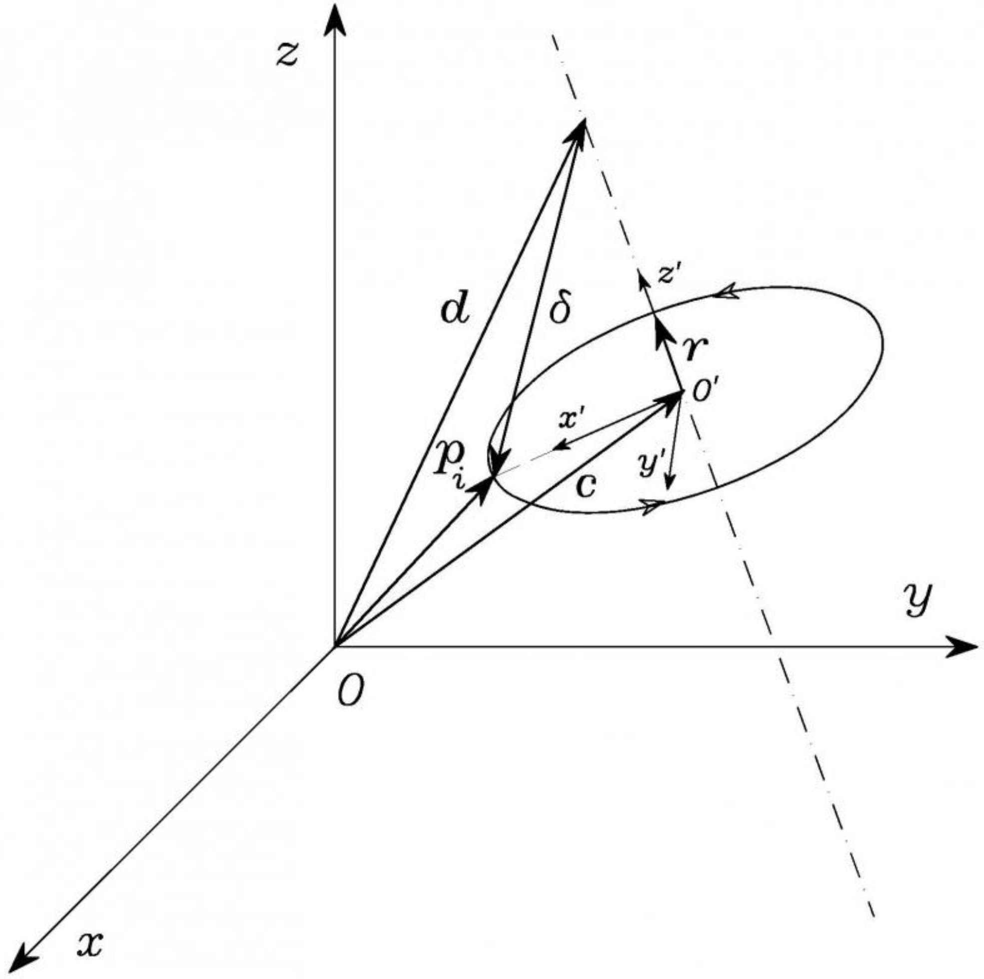
Position Trajectories II

- Circular path

$$\boldsymbol{p}_e(s) = \boldsymbol{c} + \boldsymbol{R}\begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix}$$

$$\dot{\boldsymbol{p}}_e = \boldsymbol{R}\begin{bmatrix} -\dot{s}\sin(s/\rho) \\ \dot{s}\cos(s/\rho) \\ 0 \end{bmatrix}$$

$$\ddot{\boldsymbol{p}}_e = \boldsymbol{R}\begin{bmatrix} -\dot{s}^2\cos(s/\rho)/\rho - \ddot{s}\sin(s/\rho) \\ -\dot{s}^2\sin(s/\rho)/\rho + \ddot{s}\cos(s/\rho) \\ 0 \end{bmatrix}$$



Parametric representation of a circle in space

Sequence of Points

- Sequence of $N+1$ points $\boldsymbol{p}_0, \boldsymbol{p}_1, \dots, \boldsymbol{p}_N$ connected by N segments

$$\boldsymbol{p}_e = \boldsymbol{p}_0 + \sum_{j=1}^{N} \frac{s_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) \qquad j = 1, \dots, N$$

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} \\ s_j'(t) & t_{j-1} < t < t_j \\ \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\| & t_j \le t \le t_f \end{cases}$$

$$\dot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\dot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \dot{s}_j \boldsymbol{t}_j$$

$$\ddot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\ddot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \ddot{s}_j \boldsymbol{t}_j$$

Sequence of Points

- Via points

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} - \Delta t_j \\ s'_j(t + \Delta t_j) & t_{j-1} - \Delta t_j < t < t_j - \Delta t_j \\ \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\| & t_j - \Delta t_j \le t \le t_f - \Delta t_N \end{cases}$$

$$\Delta t_j = \Delta t_{j-1} + \delta t_j \qquad j = 1, \ldots, N \quad \Delta t_0 = 0$$

Orientation Trajectories

SECTION 12

Orientation Trajectories

- Interpolation on the unit vectors $\boldsymbol{n}_e, \boldsymbol{s}_e, \boldsymbol{a}_e$ (?)
- Interpolation on Euler angles

$$\boldsymbol{\phi}_e = \boldsymbol{\phi}_i + \frac{s}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i)$$

$$\dot{\boldsymbol{\phi}}_e = \frac{\dot{s}}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i)$$

$$\ddot{\boldsymbol{\phi}}_e = \frac{\ddot{s}}{\|\boldsymbol{\phi}_f - \boldsymbol{\phi}_i\|}(\boldsymbol{\phi}_f - \boldsymbol{\phi}_i)$$

Orientation Trajectories II

- Adoption of angle and axis representation ($\boldsymbol{R}_f = \boldsymbol{R}_i \boldsymbol{R}_f^i$)

$$\boldsymbol{R}_f^i = \boldsymbol{R}_i^T \boldsymbol{R}_f = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\vartheta_f = \cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\boldsymbol{r}^i = \frac{1}{2 \sin \vartheta_f} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

Orientation Trajectories II

$$\boldsymbol{R}^i(t): \quad \boldsymbol{R}^i(0) = \boldsymbol{I} \quad \boldsymbol{R}^i(t_f) = \boldsymbol{R}^i_f \qquad\qquad \vartheta(0) = 0 \quad \vartheta(t_f) = \vartheta_f$$

$$\boldsymbol{\omega}^i = \dot{\vartheta}\,\boldsymbol{r}^i$$
$$\dot{\boldsymbol{\omega}}^i = \ddot{\vartheta}\,\boldsymbol{r}^i$$

$$\Downarrow$$

$$\boldsymbol{R}_e(t) = \boldsymbol{R}_i \boldsymbol{R}^i(t)$$
$$\boldsymbol{\omega}_e(t) = \boldsymbol{R}_i \boldsymbol{\omega}^i(t)$$
$$\dot{\boldsymbol{\omega}}_e(t) = \boldsymbol{R}_i \dot{\boldsymbol{\omega}}^i(t)$$