CS65K Robotics

Modelling, Planning and Control

Chapter 3: Differential Kinematics and Statics

Section 3.5-3.9

LECTURE 8: INVERSE DIFFERENTIAL KINEMATICS AND INVERSE JACOBIAN

DR. ERIC CHOU IEEE SENIOR MEMBER

Objectives

- The inverse kinematics problem is reformulated as the convergence of a suitable closed-loop scheme
- A Jacobian (pseudo-)inverse algorithm is introduced
- An alternative Jacobian transpose algorithm is derived

Objectives

- Different expressions for the orientation error are considered for the inverse kinematics algorithms
- The **angle** and **axis** representation leads to a compact formula for the orientation error
- The unit quaternion representation allows using the geometric Jacobian

Objectives

- The above first-order inverse kinematics algorithms are extended to the second order
- **Joint acceleration** solutions are computed
- Results of simulations and experiments for the various algorithms are compared

Jacobian Matrix

SECTION 1

Jacobian Matrix

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

$$I = A \; A^{-1}$$

$$I = A^{-1} A$$

$$I = J \; J^{-1}$$

Jacobian Matrix

$$J^{-1}J\begin{bmatrix}\dot{q}_1\\\dot{q}_2\\\vdots\\\dot{q}_n\end{bmatrix}=\begin{bmatrix}\dot{q}_1\\\dot{q}_2\\\vdots\\\dot{q}_n\end{bmatrix}=J^{-1}\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{z}\\\dot{\omega}_x\\\dot{\omega}_y\\\dot{\omega}_z\end{bmatrix}$$

$$I = A\ A^{-1}$$

$$I = A^{-1}A$$

$$I = J\ J^{-1}$$

How to find the Joint Variables

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix}$$

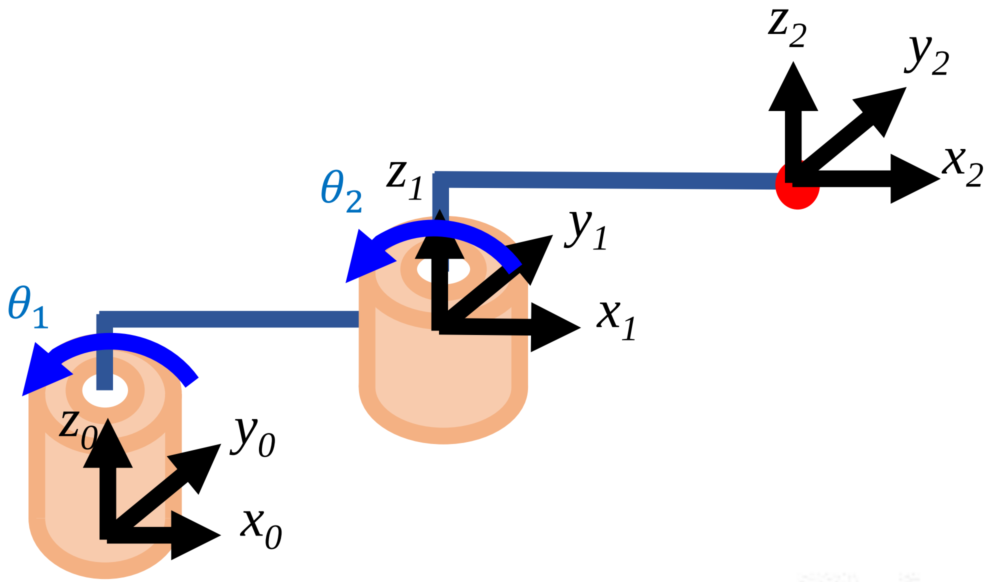
$$\Delta(\mathrm{q_i(t)\text{-}q_i(0)}) = \dot{q}_i t$$

$$\mathrm{q_i(t)} = \mathrm{q_i(0)} + \dot{q}_i t$$

Simplified SCARA Manipulator

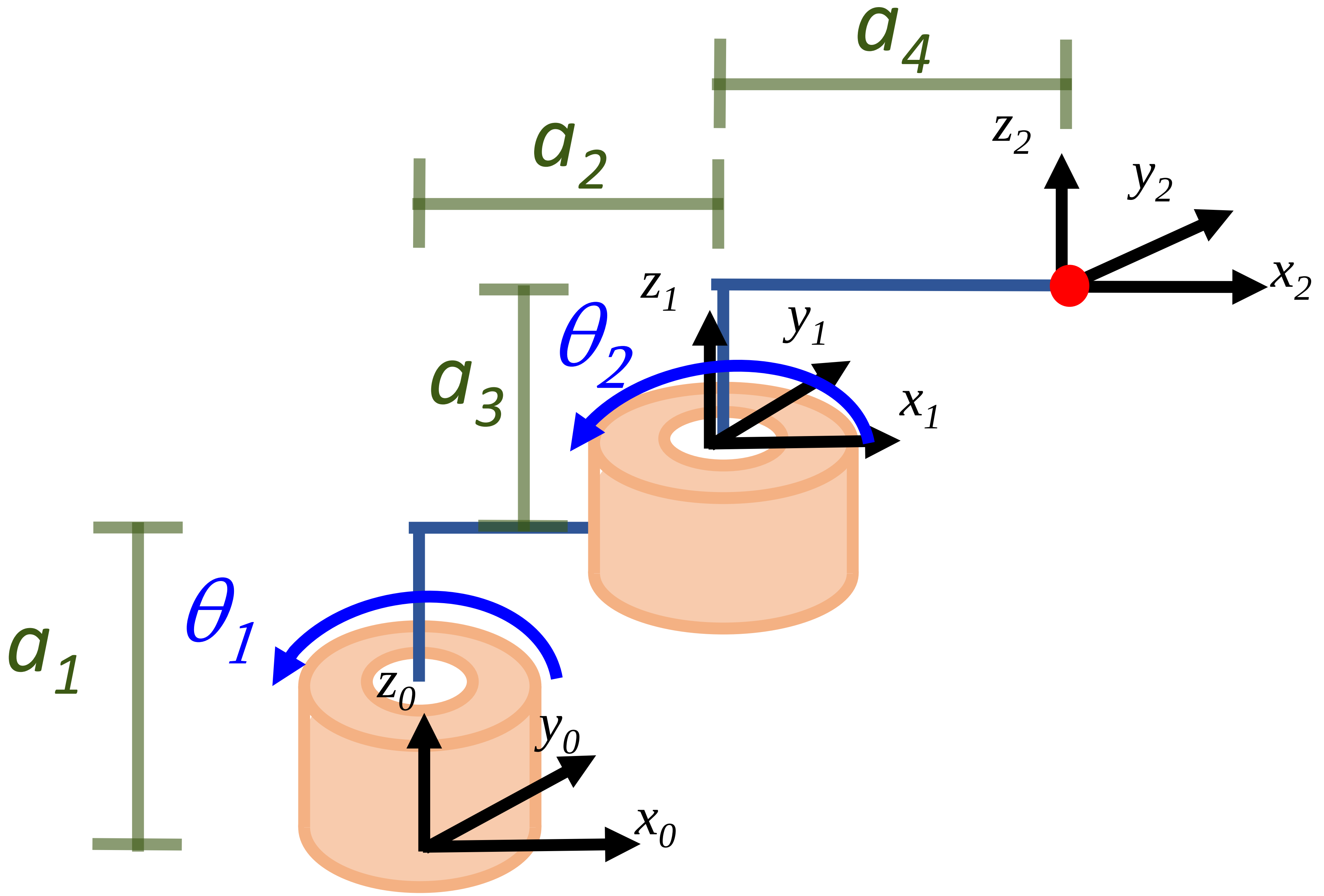
SECTION 2

Simplified Rotational Matrix and Jacobian Matrix



$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

If $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$



a_4
a_2
z_2
y_2
x_2
z_1
y_1
θ_2
a_3
x_1
θ_1
a_1
z_0
y_0
x_0

$$R = R_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$d_1^0 = \begin{bmatrix} a_2 \cos(\theta_1) \\ a_2 \sin(\theta_1) \\ a_1 \end{bmatrix}$$

$$H_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_2 \cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_2 \sin(\theta_1) \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = R_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$d_2^1 = \begin{bmatrix} a_4\cos(\theta_2) \\ a_4\sin(\theta_2) \\ a_3 \end{bmatrix}$$

$$H_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_4\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_4\sin(\theta_2) \\ 0 & 0 & 1 & a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^0 = H_1^0\, H_2^1$$

$$H_2^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_2\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_2\sin(\theta_1) \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_4\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_4\sin(\theta_2) \\ 0 & 0 & 1 & a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & a_2c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & a_2s\theta_1 \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_4c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_4s\theta_2 \\ 0 & 0 & 1 & a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c\theta_1c\theta_2 - s\theta_1s\theta_2 & -c\theta_1s\theta_2 - s\theta_1c\theta_2 & 0 & a_4c\theta_1c\theta_2 - a_4s\theta_1s\theta_2 + a_2c\theta_1 \\ s\theta_1c\theta_2 + c\theta_1s\theta_2 & -s\theta_1s\theta_2 + c\theta_1c\theta_2 & 0 & a_4s\theta_1c\theta_2 + a_4c\theta_1s\theta_2 + a_2s\theta_1 \\ 0 & 0 & 1 & a_1 + a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & a_2\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & a_2\sin(\theta_1) \\ 0 & 0 & 1 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^0 = \begin{bmatrix} c\theta_1 c\theta_2 - s\theta_1 s\theta_2 & -c\theta_1 s\theta_2 - s\theta_1 c\theta_2 & 0 & a_4 c\theta_1 c\theta_2 - a_4 s\theta_1 s\theta_2 + a_2 c\theta_1 \\ s\theta_1 c\theta_2 + c\theta_1 s\theta_2 & -s\theta_1 s\theta_2 + c\theta_1 c\theta_2 & 0 & a_4 s\theta_1 c\theta_2 + a_4 c\theta_1 s\theta_2 + a_2 s\theta_1 \\ 0 & 0 & 1 & a_1 + a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \Omega_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \Omega_1 = \begin{bmatrix} a_2\, c\theta_1 \\ a_2\, s\theta_1 \\ a_1 \end{bmatrix} = d_1 = \begin{bmatrix} d_{11} \\ d_{12} \\ d_{13} \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \Omega_2 = \begin{bmatrix} a_4 c\theta_1 c\theta_2 - a_4 s\theta_1 s\theta_2 + a_2 c\theta_1 \\ a_4 s\theta_1 c\theta_2 + a_4 c\theta_1 s\theta_2 + a_2 s\theta_1 \\ a_1 + a_3 \end{bmatrix} = d_2 = \begin{bmatrix} d_{21} \\ d_{22} \\ d_{23} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} ? & ? \\ ? & ? \\ ? & ? \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

	Prismatic	Revolute
Linear	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_n - d_{i-1})$
Rotational	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$R_{i-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

$$Z_0 \times (d_2 - d_0) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} d_{21} \\ d_{22} \\ d_{23} \end{bmatrix} = \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ d_{21} & d_{22} & d_{23} \end{vmatrix} = -d_{22}i + d_{21}j + 0k = \begin{bmatrix} -d_{22} \\ d_{21} \\ 0 \end{bmatrix}$$

$$Z_1 \times (d_2 - d_1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times \begin{bmatrix} d_{21} - d_{11} \\ d_{22} - d_{12} \\ d_{23} - d_{13} \end{bmatrix} = \begin{vmatrix} i & j & k \\ 0 & 0 & 1 \\ d_{21} - d_{11} & d_{22} - d_{12} & d_{23} - d_{13} \end{vmatrix} = (d_{12} - d_{22})i + (d_{22} - d_{12})j + 0k$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\omega_z = \dot{\theta}_1 + \dot{\theta}_2$$

$$\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} -a_4 s\theta_1 c\theta_2 - a_4 c\theta_1 s\theta_2 - a_2 s\theta_1 & -a_4 s\theta_1 c\theta_2 - a_4 c\theta_1 s\theta_2 \\ a_4 c\theta_1 c\theta_2 - a_4 s\theta_1 s\theta_2 + a_2 c\theta_1 & a_4 c\theta_1 c\theta_2 - a_4 s\theta_1 s\theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

$$J^{-1} = \frac{\begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}}{\begin{vmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{vmatrix}}$$

$$\dot{\theta}_1 = {J^{-1}}_{11}\, \dot{x} + {J^{-1}}_{12}\, \dot{y}$$

$$\dot{\theta}_2 = {J^{-1}}_{21}\, \dot{x} + {J^{-1}}_{22}\, \dot{y}$$

Closed-loop Solutions

SECTION 3

Algorithmic Solutions

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + \boldsymbol{J}^{-1}(\boldsymbol{q}(t_k))\boldsymbol{v}_e(t_k)\Delta t$$

- Solution drift

Operational space error

$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x}_e$$

- Differentiating …

$$\begin{aligned}\dot{\boldsymbol{e}} &= \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}_e \\ &= \dot{\boldsymbol{x}}_d - \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}\end{aligned}$$

- Find

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}(\boldsymbol{e}) : \ \boldsymbol{e} \to \mathbf{0}$$

Jacobian (Pseudo-)Inverse

Error dynamics linearization

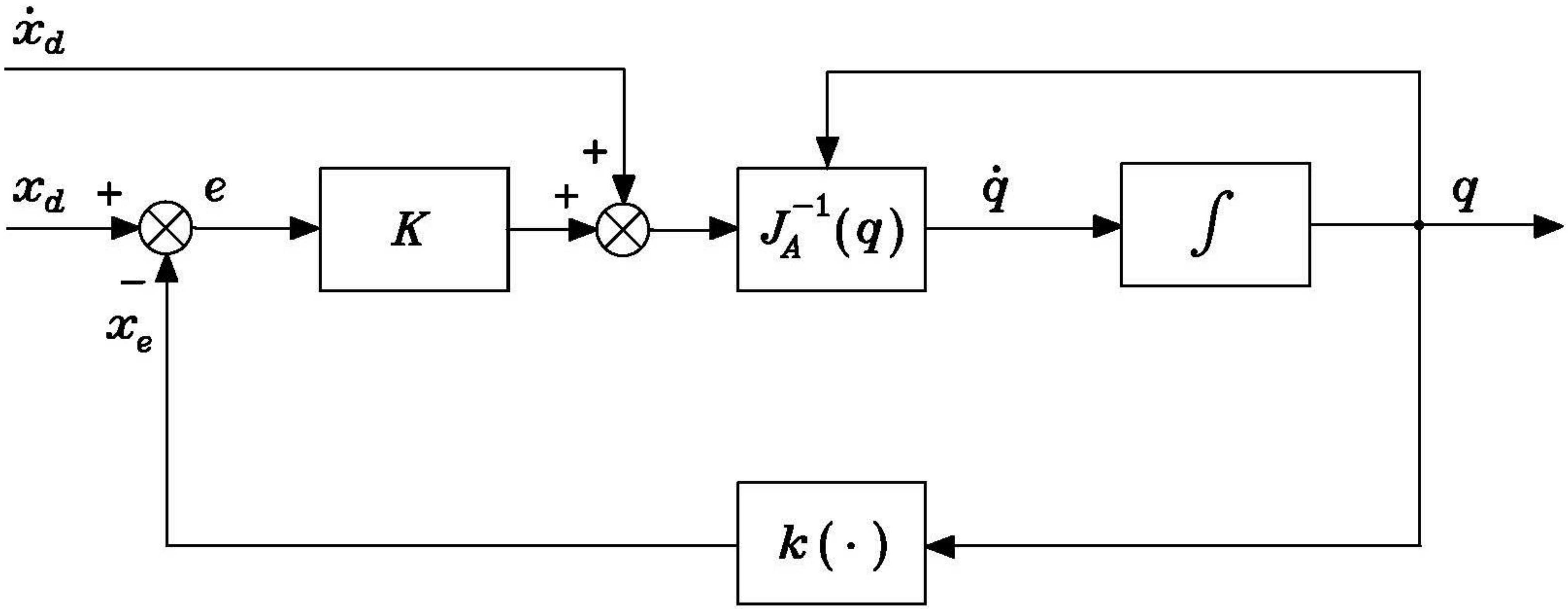
$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\,\boldsymbol{e})$$

$$\Downarrow$$

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\,\boldsymbol{e} = \boldsymbol{0} \qquad\qquad \boldsymbol{K} > \boldsymbol{O} \text{ (asymptotic stability)}$$

- For a *redundant* manipulator

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{\dagger}(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\,\boldsymbol{e}) + (\boldsymbol{I}_n - \boldsymbol{J}_A^{\dagger}\boldsymbol{J}_A)\dot{\boldsymbol{q}}_0$$



Block scheme of the inverse kinematics algorithm with Jacobian inverse

Jacobian Transpose

$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}(\boldsymbol{e})$ without linearizing error dynamics

- Lyapunov method

$$V(\boldsymbol{e}) = \frac{1}{2}\boldsymbol{e}^T \boldsymbol{K} \boldsymbol{e} \qquad V(\boldsymbol{e}) > 0 \quad \forall \boldsymbol{e} \neq \boldsymbol{0}$$

- Differentiating …

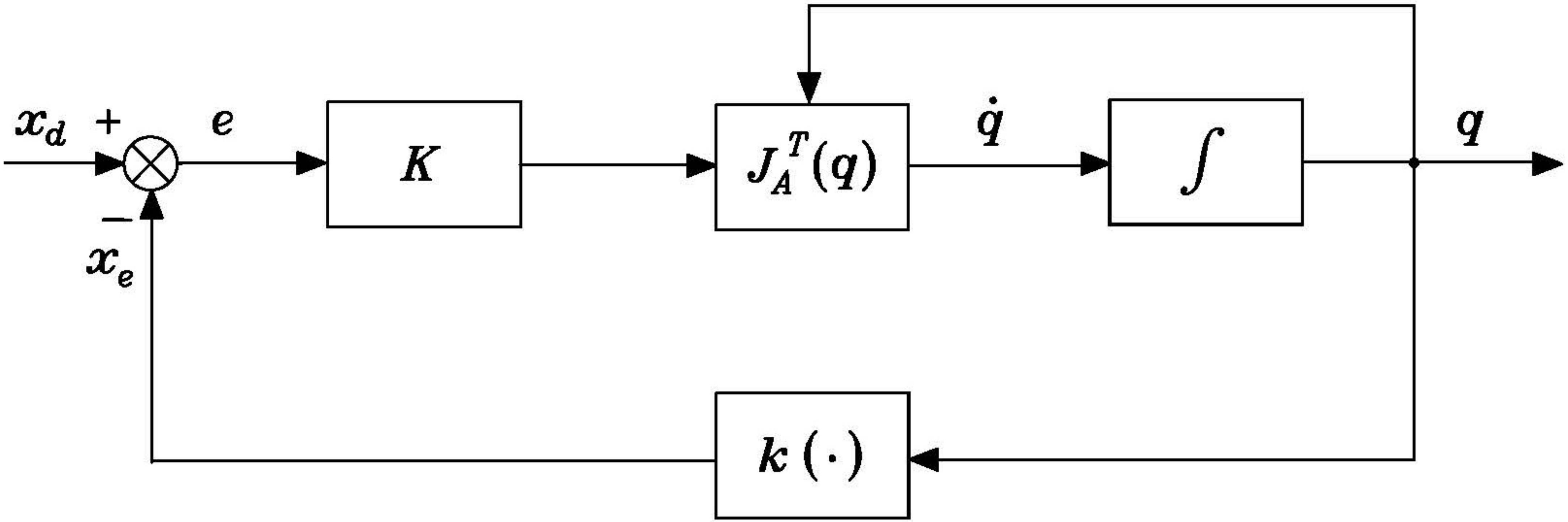
$$\dot{V} = \boldsymbol{e}^T \boldsymbol{K} \dot{\boldsymbol{x}}_d - \boldsymbol{e}^T \boldsymbol{K} \dot{\boldsymbol{x}}_e = \boldsymbol{e}^T \boldsymbol{K} \dot{\boldsymbol{x}}_d - \boldsymbol{e}^T \boldsymbol{K} \boldsymbol{J}_A(\boldsymbol{q}) \dot{\boldsymbol{q}}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^T(\boldsymbol{q}) \boldsymbol{K} \boldsymbol{e} \implies \dot{V} = \boldsymbol{e}^T \boldsymbol{K} \dot{\boldsymbol{x}}_d - \boldsymbol{e}^T \boldsymbol{K} \boldsymbol{J}_A(\boldsymbol{q}) \boldsymbol{J}_A^T(\boldsymbol{q}) \boldsymbol{K} \boldsymbol{e}$$

Jacobian Transpose

- If $\dot{\boldsymbol{x}}_d = \boldsymbol{0} \Longrightarrow$ $\dot{V} < 0$ with $V > 0$ (*asymptotic stability*)
- If $\mathcal{N}(\boldsymbol{J}_A^T) \neq \emptyset \Longrightarrow$ $\dot{V} = 0$ if $\boldsymbol{K}\boldsymbol{e} \in \mathcal{N}(\boldsymbol{J}_A^T)$ then $\dot{\boldsymbol{q}} = \boldsymbol{0}$ with $\boldsymbol{e} \neq \boldsymbol{0}$ (stuck?)

If $\dot{\boldsymbol{x}}_d \neq \boldsymbol{0} \Longrightarrow$ bounded $\boldsymbol{e}$ (increasing norm of $\boldsymbol{K}$) … $\boldsymbol{e}(\infty) \to \boldsymbol{0}$



Block scheme of the inverse kinematics algorithm with Jacobian transpose

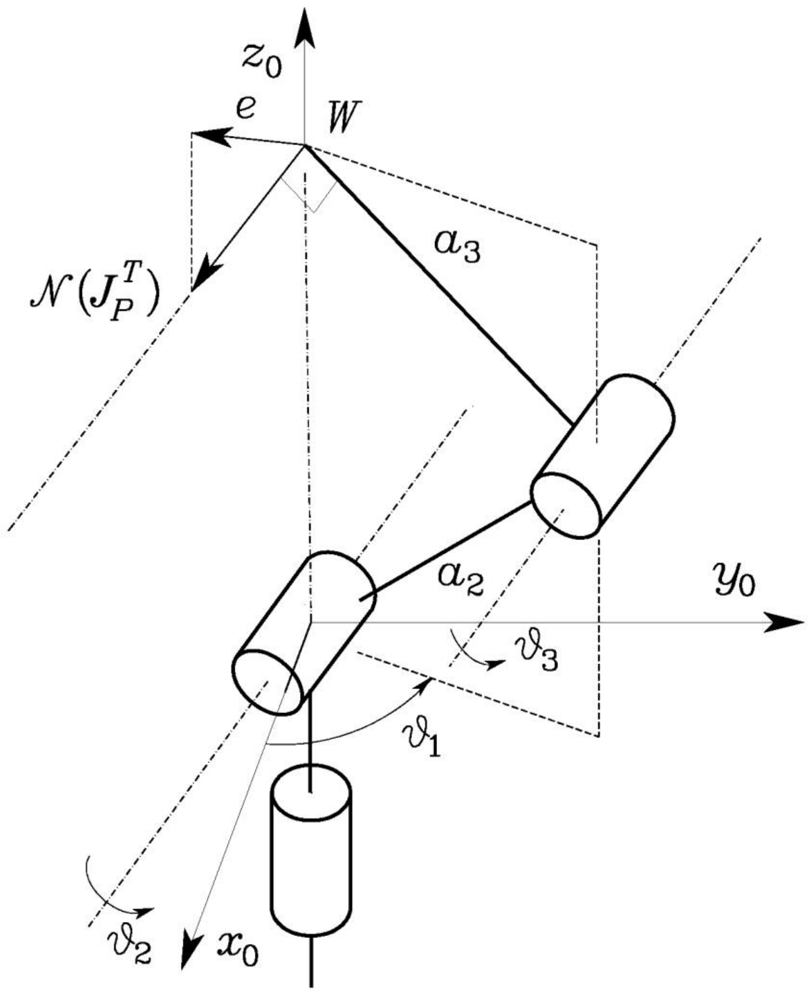
Anthropomorphic Arm

Null space (shoulder singularity)

$$\boldsymbol{J}_P^T = \begin{bmatrix} 0 & 0 & 0 \\ -c_1(a_2 s_2 + a_3 s_{23}) & -s_1(a_2 s_2 + a_3 s_{23}) & 0 \\ -a_3 c_1 s_{23} & -a_3 s_1 s_{23} & a_3 c_{23} \end{bmatrix}$$

- If desired path is along the line normal to the plane of the structure at the intersection with the wrist point

$$\frac{\nu_y}{\nu_x} = -\frac{1}{\tan\vartheta_1} \qquad \nu_z = 0$$



Characterization of the anthropomorphic arm at a shoulder singularity for the admissible solutions of the Jacobian transpose algorithm

Orientation Error

SECTION 4

Euler Angles

Position Error

$$\boldsymbol{e}_P = \boldsymbol{p}_d - \boldsymbol{p}_e(\boldsymbol{q})$$
$$\dot{\boldsymbol{e}}_P = \dot{\boldsymbol{p}}_d - \dot{\boldsymbol{p}}_e$$

Euler Angles

$$\boldsymbol{e}_O = \phi_d - \phi_e(\boldsymbol{q})$$
$$\dot{\boldsymbol{e}}_O = \dot{\phi}_d - \dot{\phi}_e$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q}) \begin{bmatrix} \dot{\boldsymbol{p}}_d + \boldsymbol{K}_P \boldsymbol{e}_P \\ \dot{\phi} + \boldsymbol{K}_O \boldsymbol{e}_O \end{bmatrix}$$

- Easy to specify $\phi_d(t)$
- Requires computation of ϕ_e with inverse formulae from $\boldsymbol{R}_e = \begin{bmatrix} \boldsymbol{n}_e & \boldsymbol{s}_e & \boldsymbol{a}_e \end{bmatrix}$

Manipulator with spherical wrist

- Compute $\boldsymbol{q}_P \implies \boldsymbol{R}_W$
- Compute $\boldsymbol{R}_W^T \boldsymbol{R}_d \implies \boldsymbol{q}_O$ (ZYZ Euler angles)

Angle and Axis

$$\boldsymbol{R}(\vartheta,\boldsymbol{r}) = \boldsymbol{R}_d\boldsymbol{R}_e^T(\boldsymbol{q})$$

Orientation Error

$$\boldsymbol{e}_O = \boldsymbol{r}\sin\vartheta \qquad -\pi/2 < \vartheta < \pi/2$$

$$= \frac{1}{2}(\boldsymbol{n}_e(\boldsymbol{q})\times\boldsymbol{n}_d + \boldsymbol{s}_e(\boldsymbol{q})\times\boldsymbol{s}_d + \boldsymbol{a}_e(\boldsymbol{q})\times\boldsymbol{a}_d) \qquad \boldsymbol{n}_e^T\boldsymbol{n}_d \geq 0,\, \boldsymbol{s}_e^T\boldsymbol{s}_d \geq 0,\, \boldsymbol{a}_e^T\boldsymbol{a}_d \geq 0$$

- Differentiating …

$$\dot{\boldsymbol{e}}_O = \boldsymbol{L}^T\boldsymbol{\omega}_d - \boldsymbol{L}\boldsymbol{\omega}_e \qquad \boldsymbol{L} = -\frac{1}{2}\left(\boldsymbol{S}(\boldsymbol{n}_d)\boldsymbol{S}(\boldsymbol{n}_e) + \boldsymbol{S}(\boldsymbol{s}_d)\boldsymbol{S}(\boldsymbol{s}_e) + \boldsymbol{S}(\boldsymbol{a}_d)\boldsymbol{S}(\boldsymbol{a}_e)\right)$$

$$\dot{\boldsymbol{e}} = \begin{bmatrix}\dot{\boldsymbol{e}}_P\\ \dot{\boldsymbol{e}}_O\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{p}}_d - \boldsymbol{J}_P(\boldsymbol{q})\dot{\boldsymbol{q}}\\ \boldsymbol{L}^T\boldsymbol{\omega}_d - \boldsymbol{L}\boldsymbol{J}_O(\boldsymbol{q})\dot{\boldsymbol{q}}\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{p}}_d\\ \boldsymbol{L}^T\boldsymbol{\omega}_d\end{bmatrix} - \begin{bmatrix}\boldsymbol{I} & \boldsymbol{O}\\ \boldsymbol{O} & \boldsymbol{L}\end{bmatrix}\boldsymbol{J}\dot{\boldsymbol{q}}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\begin{bmatrix}\dot{\boldsymbol{p}}_d + \boldsymbol{K}_P\boldsymbol{e}_P\\ \boldsymbol{L}^{-1}\left(\boldsymbol{L}^T\boldsymbol{\omega}_d + \boldsymbol{K}_O\boldsymbol{e}_O\right)\end{bmatrix}$$

Unit Quaternion

$$\Delta Q = Q_d * Q_e^{-1}$$

Orientation Error

$$\boldsymbol{e}_O = \Delta\epsilon = \eta_e(\boldsymbol{q})\epsilon_d - \eta_d\epsilon_e(\boldsymbol{q}) - \boldsymbol{S}(\epsilon_d)\epsilon_e(\boldsymbol{q})$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\begin{bmatrix}\dot{\boldsymbol{p}}_d + \boldsymbol{K}_P\boldsymbol{e}_P \\ \omega_d + \boldsymbol{K}_O\boldsymbol{e}_O\end{bmatrix} \implies \omega_d - \omega_e + \boldsymbol{K}_O\boldsymbol{e}_O = \boldsymbol{0} \text{ (nonlinear error dynamics)}$$

- Quaternion propagation

$$\dot{\eta}_e = -\frac{1}{2}\epsilon_e^T\omega_e \qquad \dot{\epsilon}_e = \frac{1}{2}\left(\eta_e\boldsymbol{I}_3 - \boldsymbol{S}(\epsilon_e)\right)\omega_e$$

Stability analysis

$$V = (\eta_d - \eta_e)^2 + (\epsilon_d - \epsilon_e)^T(\epsilon_d - \epsilon_e) \qquad \dot{V} = -\boldsymbol{e}_O^T\boldsymbol{K}_O\boldsymbol{e}_O$$

Second-order Inverse Kinematics Algorithm

SECTION 5

Second-order Inverse Kinematics Algorithm

- Differentiating …

$$\dot{\boldsymbol{x}}_e = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}$$

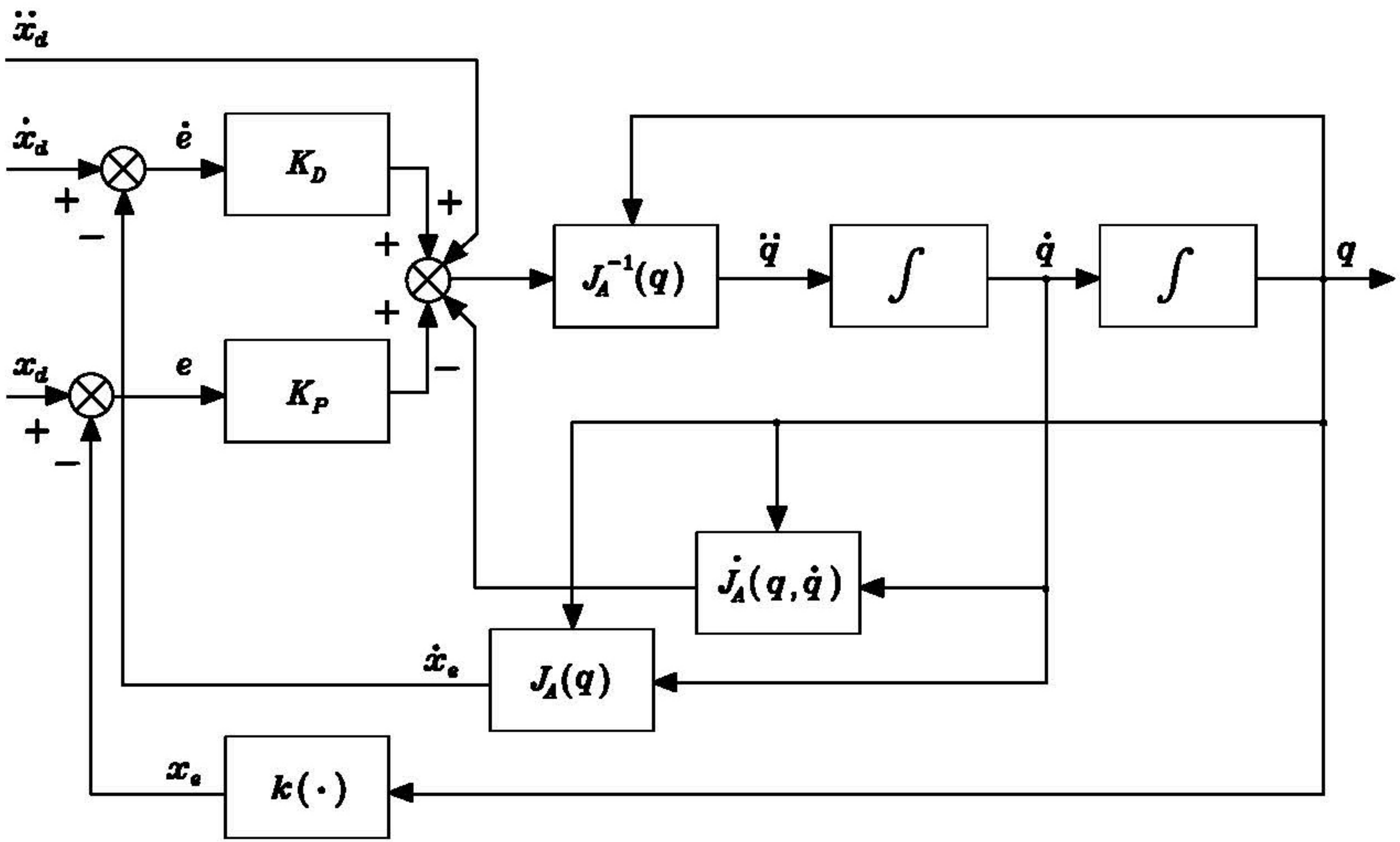
Error dynamics

$$\ddot{\boldsymbol{x}}_e = \boldsymbol{J}_A(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

$$\ddot{\boldsymbol{e}} = \ddot{\boldsymbol{x}}_d - \boldsymbol{J}_A(\boldsymbol{q})\ddot{\boldsymbol{q}} - \dot{\boldsymbol{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$$

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})\left(\ddot{\boldsymbol{x}}_d + \boldsymbol{K}_D\dot{\boldsymbol{e}} + \boldsymbol{K}_P\boldsymbol{e} - \dot{\boldsymbol{J}}_A(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\right) \implies \ddot{\boldsymbol{e}} + \boldsymbol{K}_D\dot{\boldsymbol{e}} + \boldsymbol{K}_P\boldsymbol{e} = \boldsymbol{0}$$

$$\boldsymbol{K}_P, \boldsymbol{K}_D > \boldsymbol{O} \text{ (asympotic stability)}$$



Block scheme of the second-order inverse kinematics algorithm with Jacobian inverse

Comparison Among Inverse Kinematics Algorithms

Three-link Planar Arm

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q})$$

$$\begin{bmatrix} p_x \\ p_y \\ \phi \end{bmatrix} = \begin{bmatrix} a_1c_1 + a_2c_{12} + a_3c_{123} \\ a_1s_1 + a_2s_{12} + a_3s_{123} \\ \vartheta_1 + \vartheta_2 + \vartheta_3 \end{bmatrix}$$

- $a_1 = a_2 = a_3 = 0.5\,\mathrm{m}$

$$\boldsymbol{J}_A = \begin{bmatrix} -a_1s_1 - a_2s_{12} - a_3s_{123} & -a_2s_{12} - a_3s_{123} & -a_3s_{123} \\ a_1c_1 + a_2c_{12} + a_3c_{123} & a_2c_{12} + a_3c_{123} & a_3c_{123} \\ 1 & 1 & 1 \end{bmatrix}$$

Comparison Among Inverse Kinematics Algorithms

Desired trajectory

$$\boldsymbol{q}_i = \begin{bmatrix} \pi & -\pi/2 & -\pi/2 \end{bmatrix}^T \text{ rad} \quad \Longrightarrow \quad \boldsymbol{p}_{di} = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^T \text{ m}, \phi_{di} = 0 \text{ rad}$$

$$\boldsymbol{p}_d(t) = \begin{bmatrix} 0.25(1 - \cos \pi t) \\ 0.25(2 + \sin \pi t) \end{bmatrix} \quad \phi_d(t) = \sin \frac{\pi}{24} t \qquad 0 \le t \le 4$$

MATLAB simulation with Euler numerical integration

$$\boldsymbol{q}(t_{k+1}) = \boldsymbol{q}(t_k) + \dot{\boldsymbol{q}}(t_k)\Delta t \quad \Delta t = 1 \text{ ms}$$

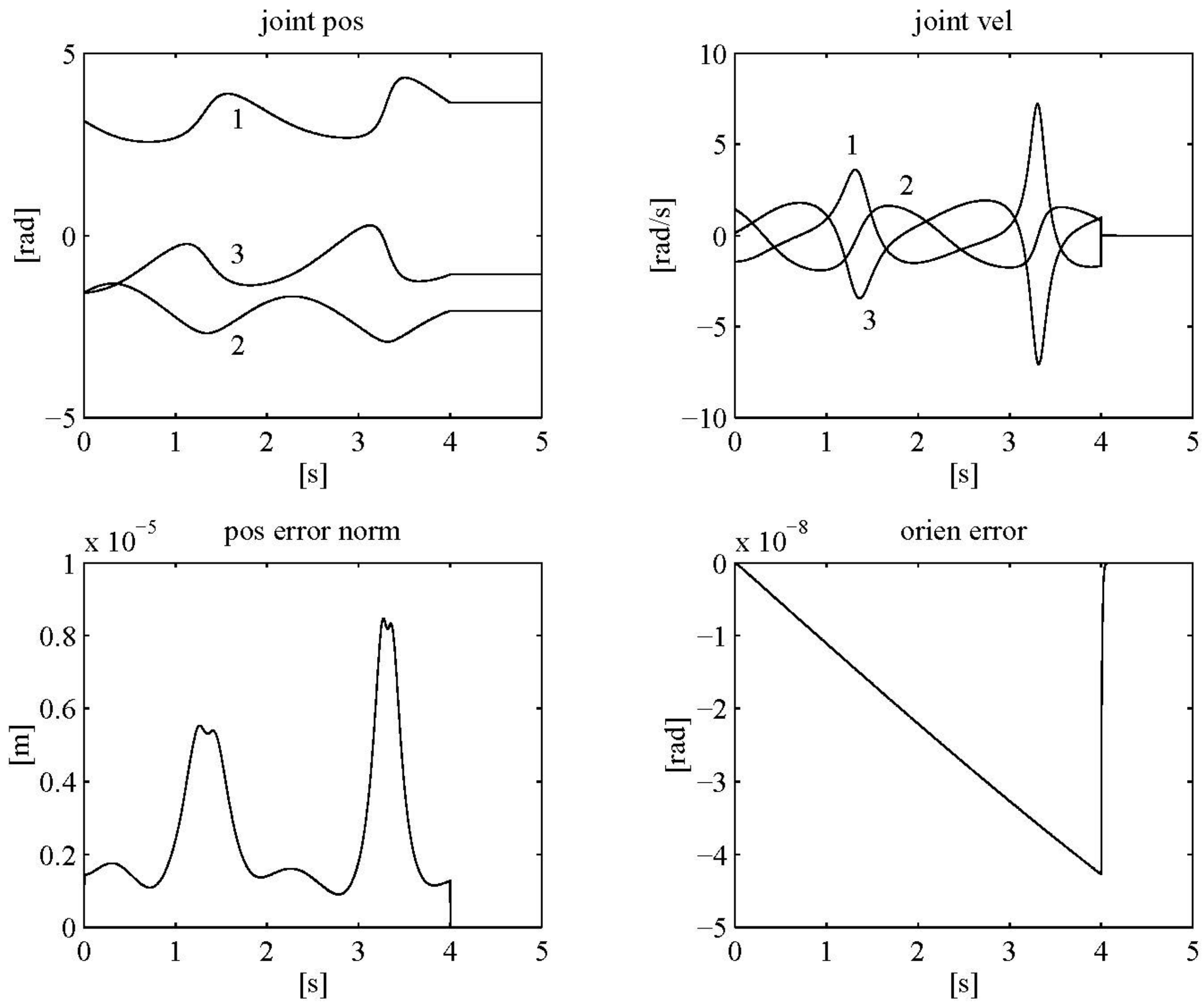
Open-loop Vs. Closed-loop Inverse Kinematics

Open-loop Inverse Jacobian Algorithm

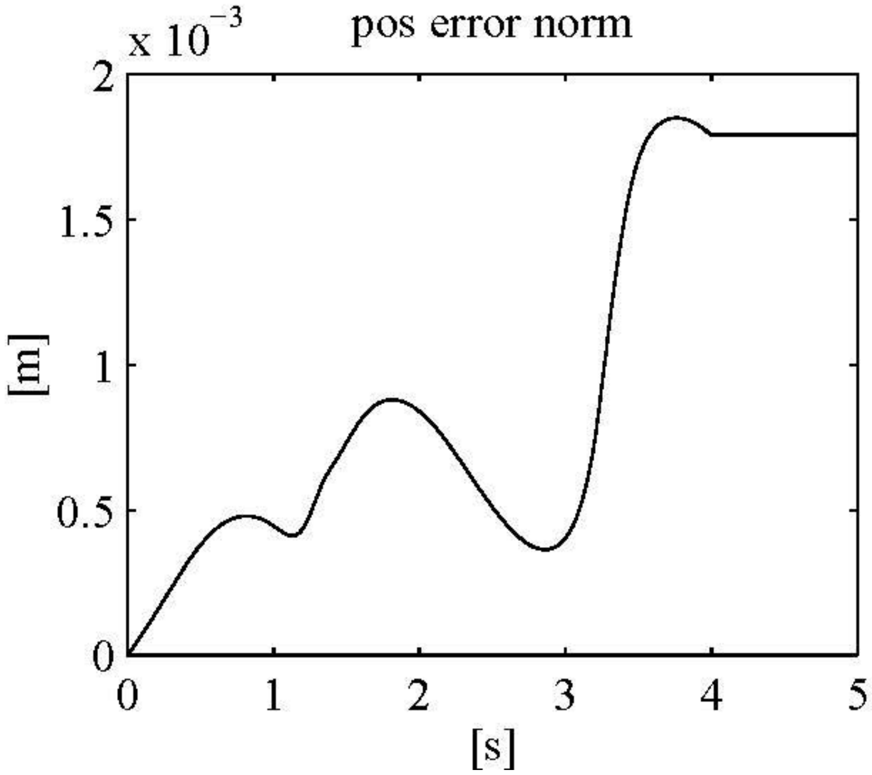
$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})\dot{\boldsymbol{x}}_d$$

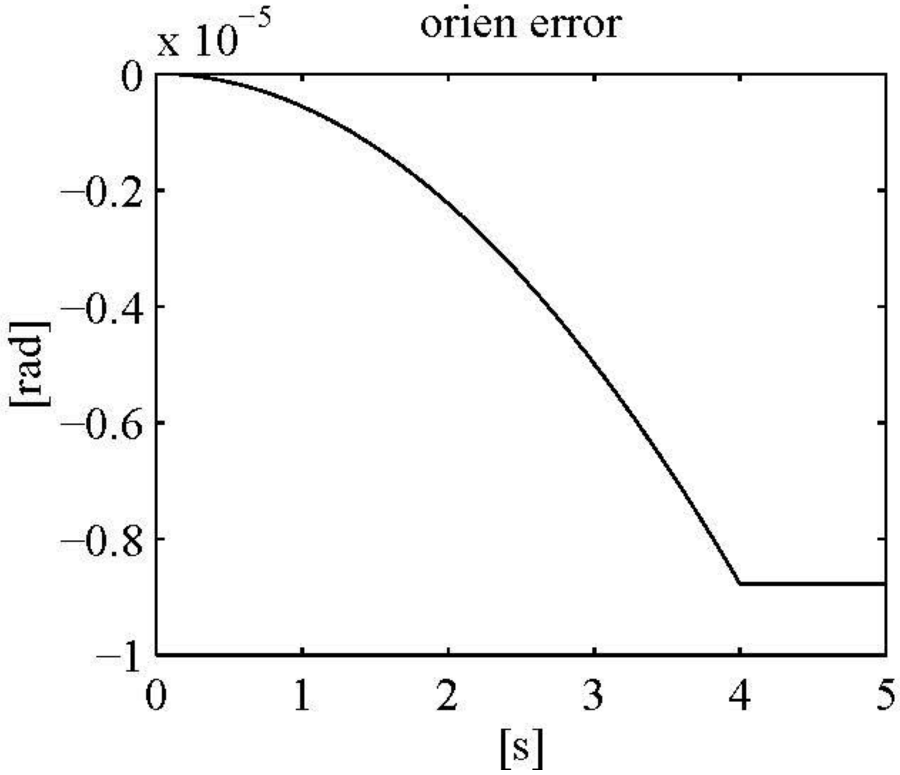
Closed-loop Inverse Jacobian Algorithm

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_A^{-1}(\boldsymbol{q})\left(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}\right) \qquad \boldsymbol{K} = \mathrm{diag}\{500, 500, 100\}$$



Time history of the joint positions and velocities, and of the norm of end-effector position error and orientation error with the closed-loop inverse Jacobian algorithm





Time history of the norm of end-effector position error and orientation error with the open-loop inverse Jacobian algorithm

Jacobian Pseudo-inverse Vs. Jacobian Transpose

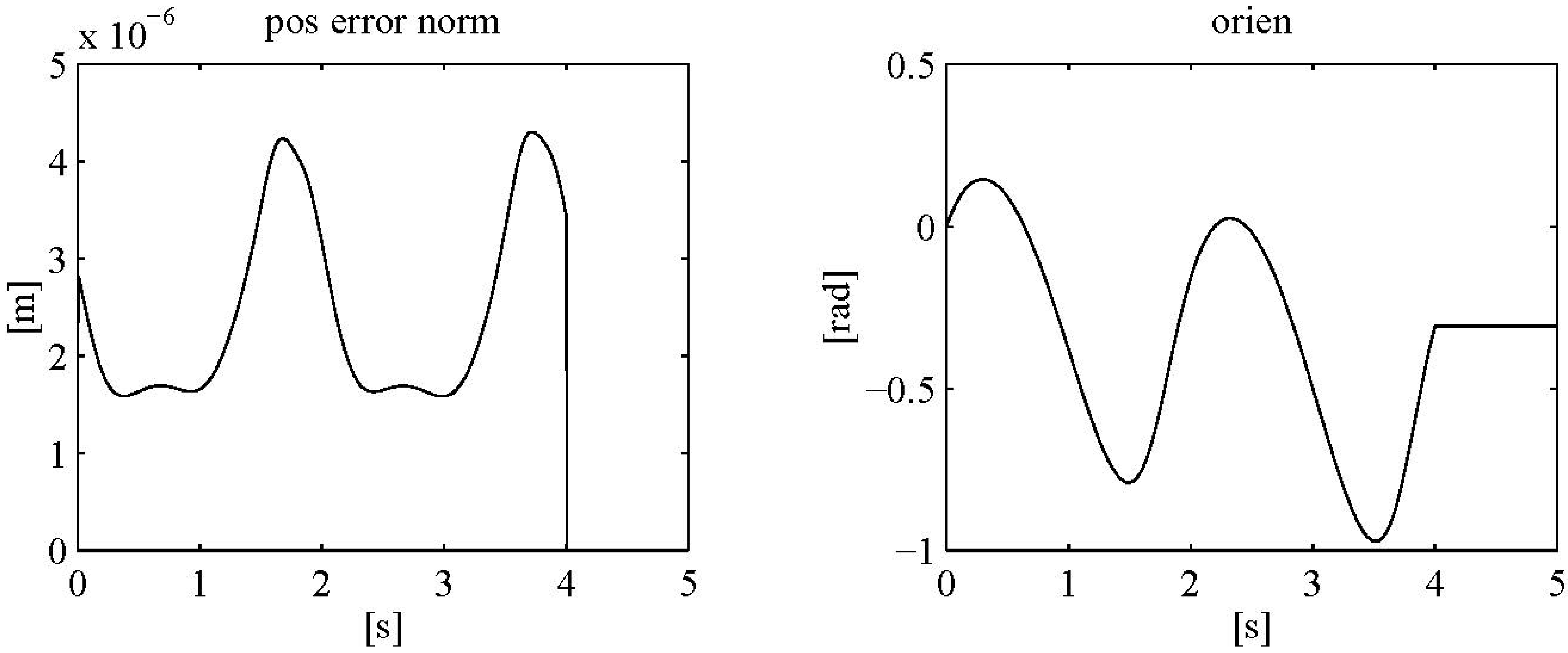
$\phi \ (r = 2, n = 3)$

Free orientation

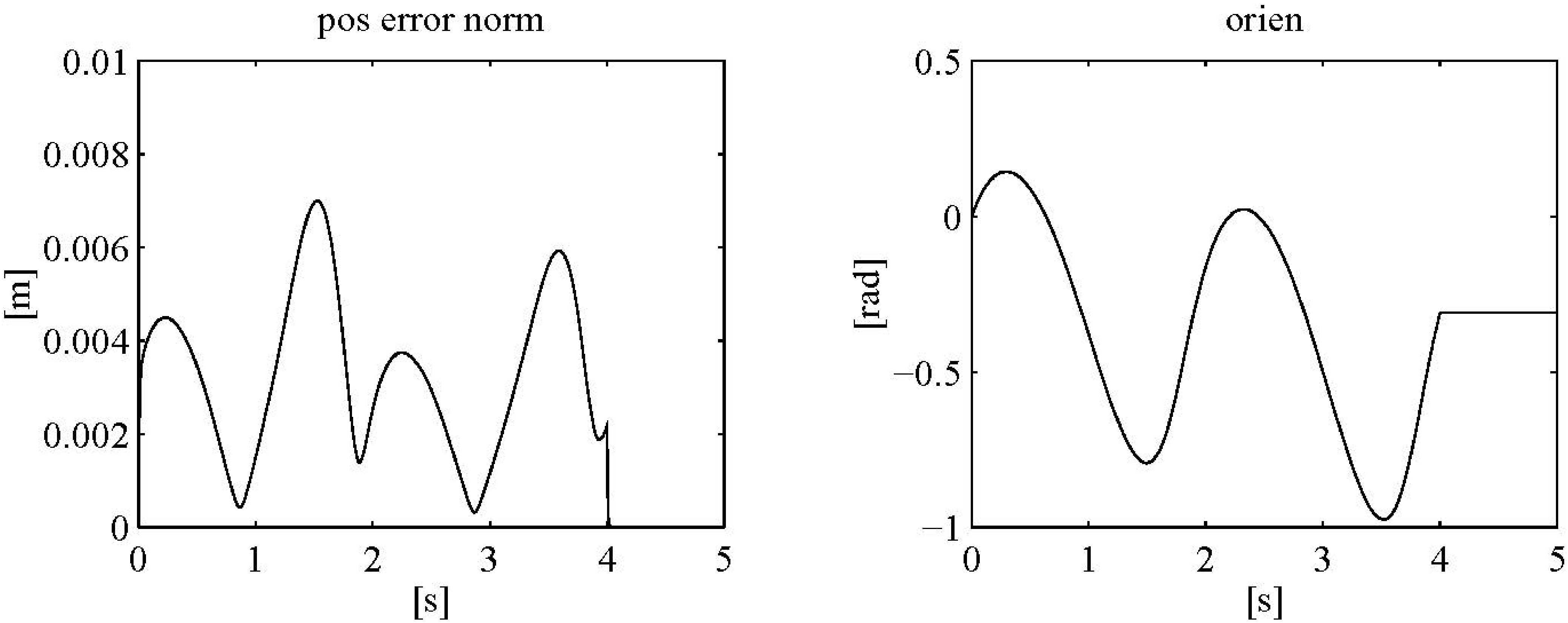
Jacobian Pseudo-inverse Algorithm

Jacobian Transpose Algorithm

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_P^T \boldsymbol{K}_P \boldsymbol{e}_P \qquad \boldsymbol{K}_p = \mathrm{diag}\{500, 500\}$$



Time history of the norm of end-effector position error and orientation with the Jacobian pseudo-inverse algorithm



Time history of the norm of end-effector position error and orientation with the Jacobian transpose algorithm

Use of Redundancy

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_P^{\dagger}(\boldsymbol{q})\left(\dot{\boldsymbol{p}}_d + \boldsymbol{K}_P \boldsymbol{e}_P\right) + \left(\boldsymbol{I} - \boldsymbol{J}_P^{\dagger}\boldsymbol{J}_P\right)\dot{\boldsymbol{q}}_0 \qquad \boldsymbol{K}_p = \mathrm{diag}\{500, 500\}$$

Manipulability Measure:

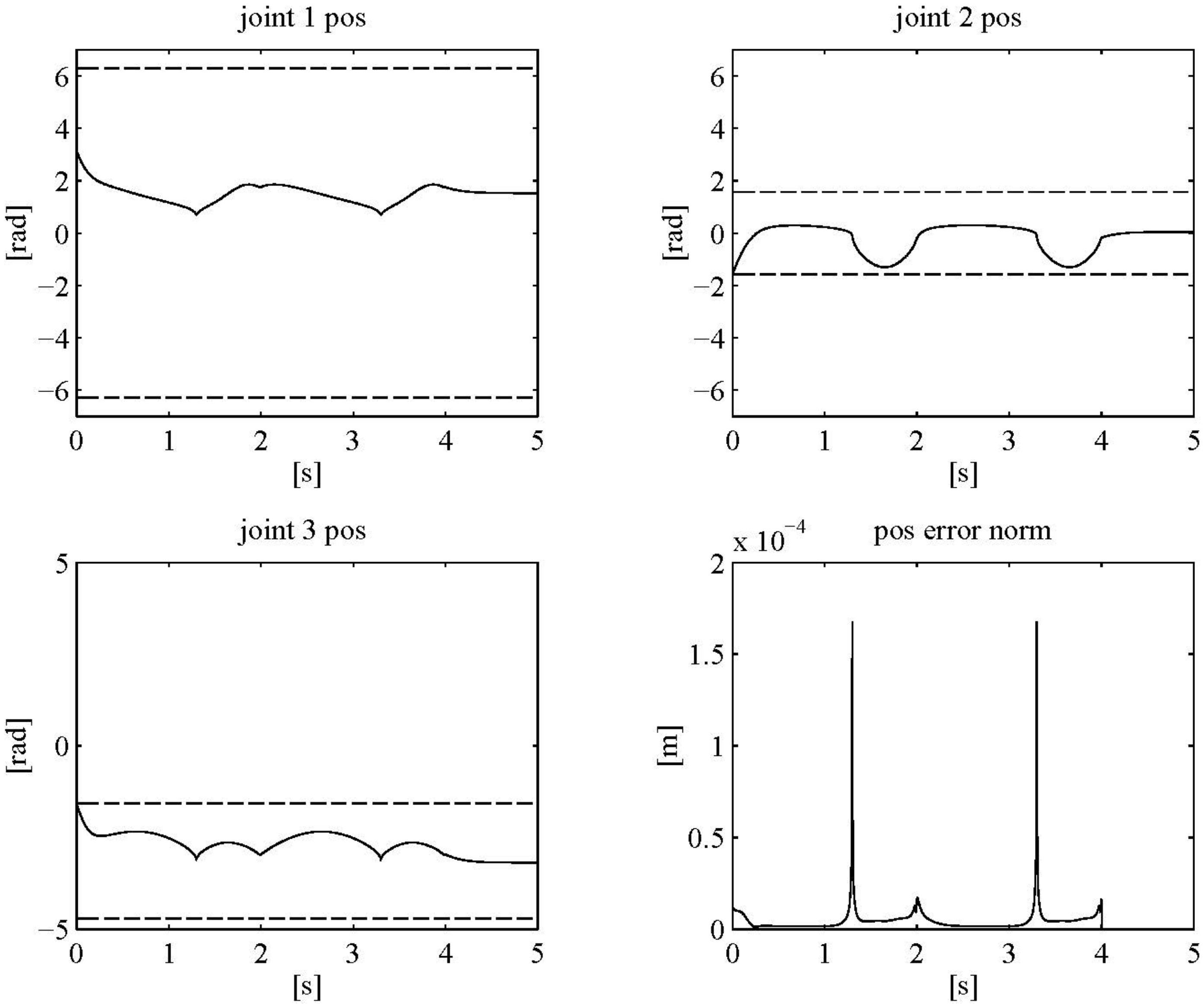
$$w(\vartheta_2, \vartheta_3) = \frac{1}{2}(s_2^2 + s_3^2)$$

$$\dot{\boldsymbol{q}}_0 = k_0 \left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T \qquad k_0 = 50$$

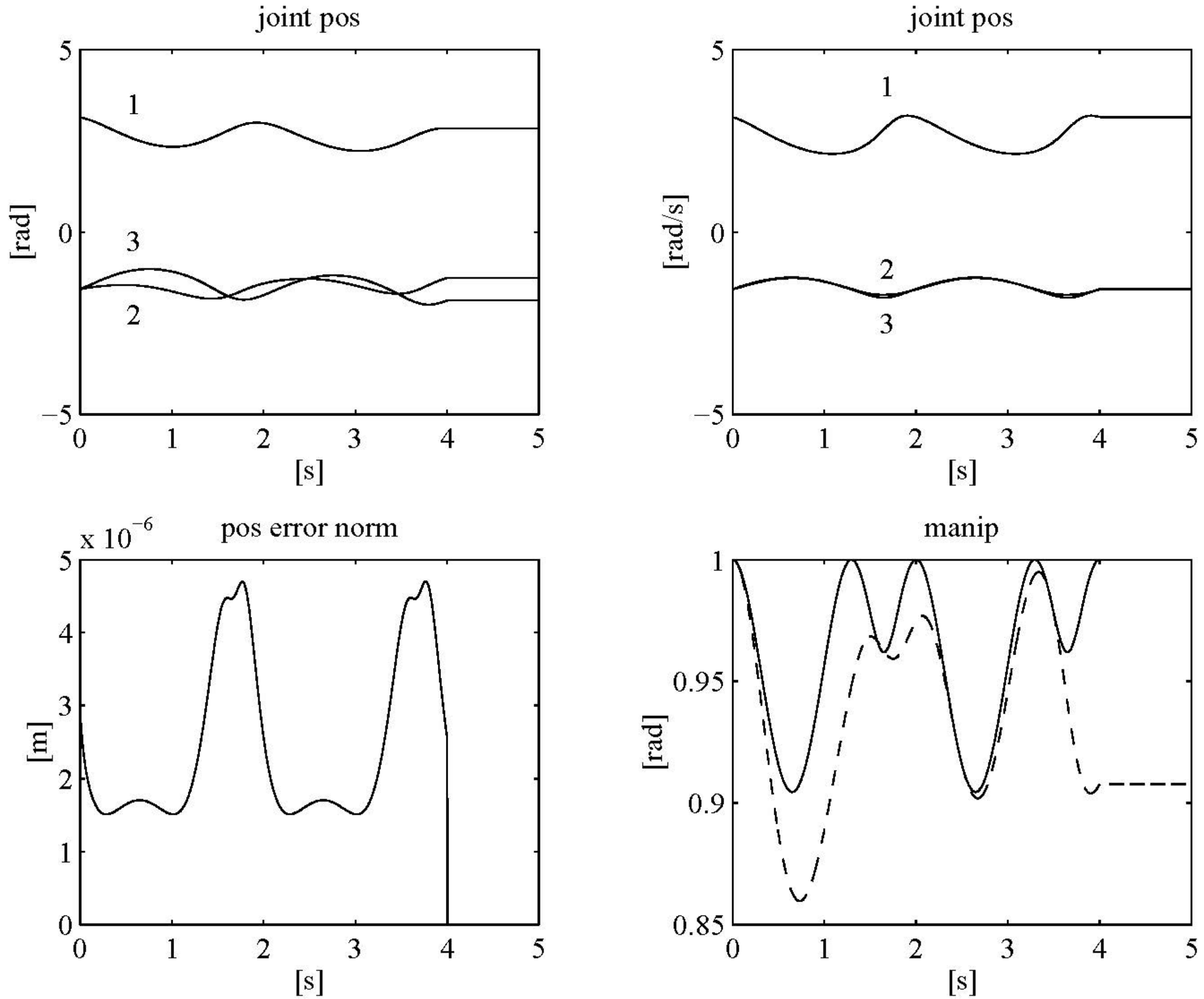
Distance from Mechanical Joint Limits:

$$w(\boldsymbol{q}) = -\frac{1}{6}\sum_{i=1}^{3}\left(\frac{q_i - \bar{q}_i}{q_{iM} - q_{im}}\right)^2 \qquad \begin{aligned} q_{1m} &= -2\pi, \; q_{1M} = 2\pi \\ q_{2m} &= -\pi/2, \; q_{2M} = \pi/2 \\ q_{3m} &= -3\pi/2, \; q_{3M} = -\pi/2 \end{aligned}$$

$$\dot{\boldsymbol{q}}_0 = k_0 \left(\frac{\partial w(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T \qquad k_0 = 250$$



Time history of the joint positions and the norm of end-effector position error with the Jacobian pseudo-inverse algorithm and joint limit constraint (joint limits are denoted by dashed lines)



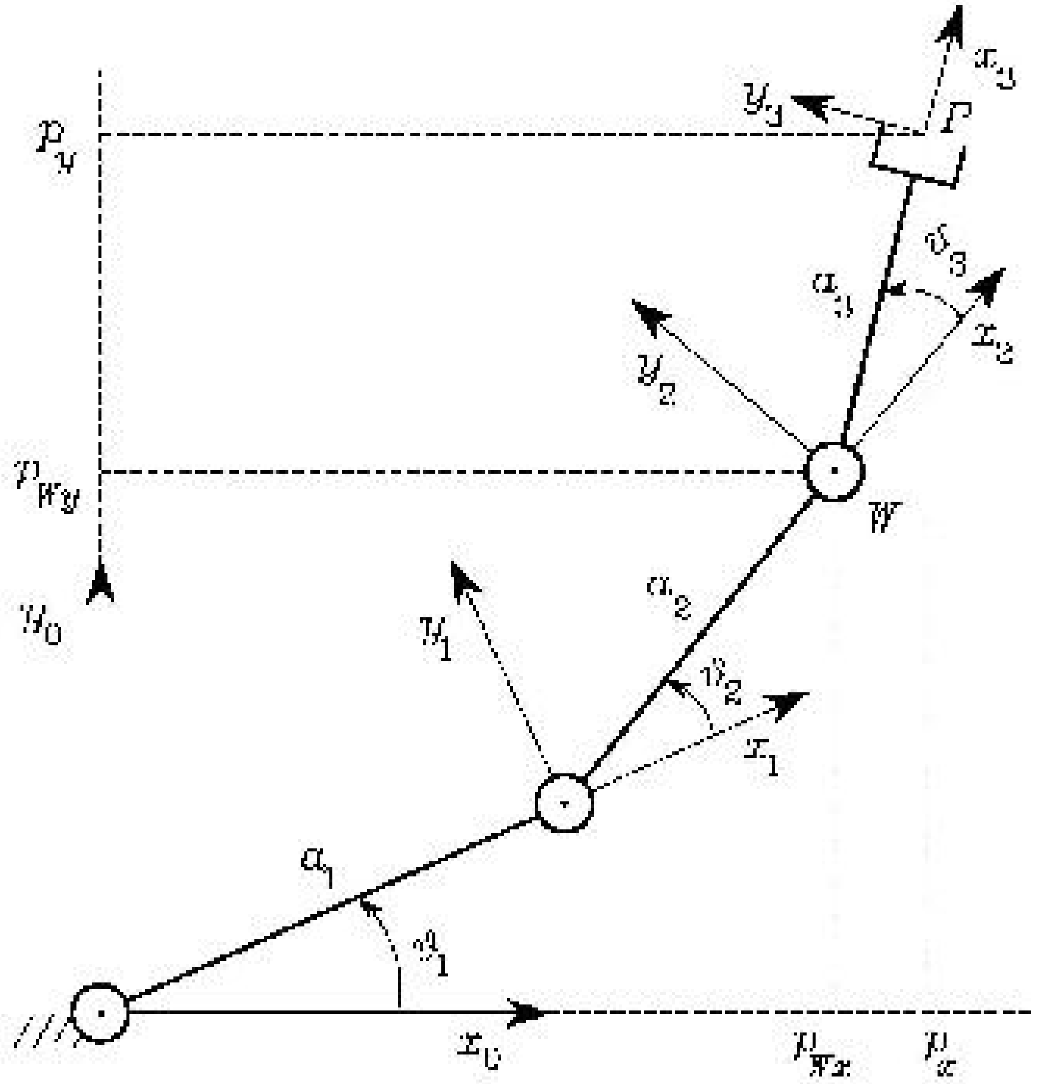
Time history of the joint positions, the norm of end-effector position error, and the manipulability measure with the Jacobian pseudo-inverse algorithm and manipulability constraint; upper left: with the unconstrained solution, upper right: with the constrained solution

Further Insights

Case 1:

Consider the 3-link planar arm in the figure, whose link lengths are respectively 0.5 m, 0.3 m, 0.3 m. Perform a computer implementation of the inverse kinematics algorithm using the Jacobian pseudo-inverse along the operational space path given by a straight line connecting the points of coordinates (0.8,0.2) m and (0.8,-0.2) m. Add a constraint aimed at avoiding link collision with a circular object located at (0.3,0) m. The initial arm configuration is chosen so that it can be modeled in 2D .

The final time is 2 s. Use sinusoidal motion timing laws. Adopt the Euler numerical integration scheme with an integration time of 1 ms.



SCARA Manipulator

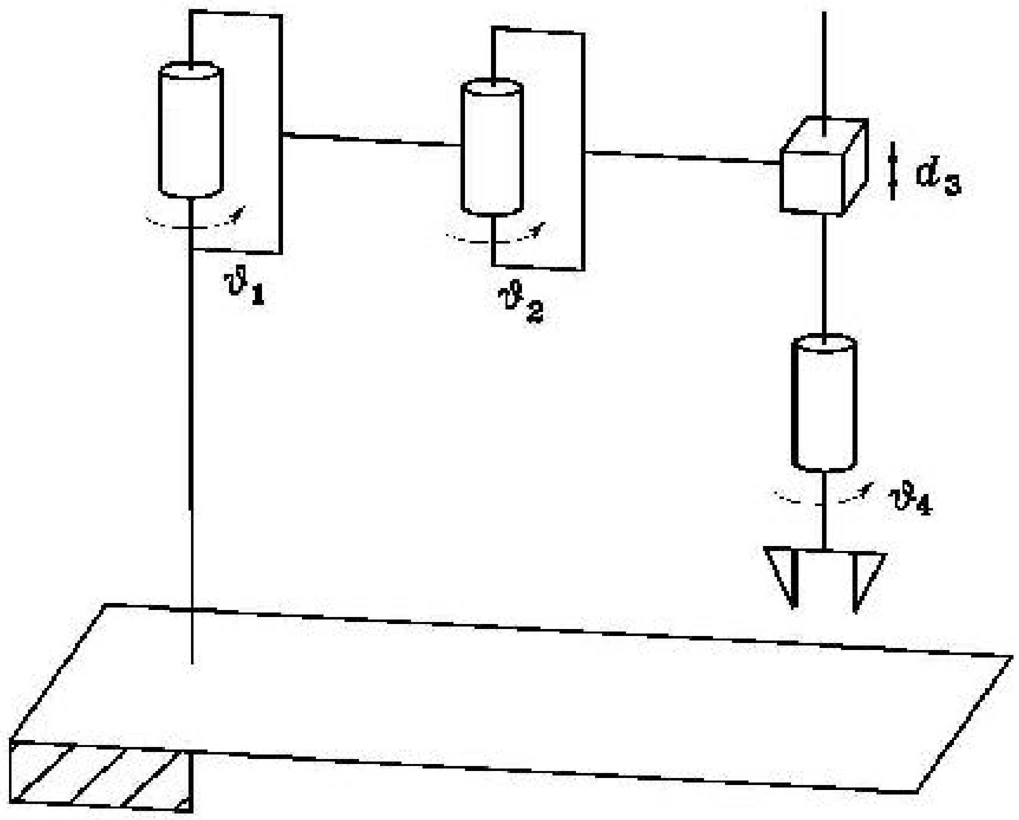
Further Insights II

Case 2:

Consider the SCARA manipulator in the figure, whose links both have a length of 0.5 m and are located at a height of 1 m from the supporting plane.

Perform a computer implementation of the inverse kinematics algorithms with both Jacobian inverse and Jacobian transpose along the operational space path whose position is given by a straight line connecting the points of coordinates (0.7,0,0) m and (0,0.8,0.5) m, and whose orientation is given by a rotation from 0 rad to 2π rad. The initial arm configuration is chosen so that the arm can be shown on a 2-D plane. The final time is 2 s. Use sinusoidal motion timing laws.

Adopt the Euler numerical integration scheme with an integration time of 1 ms.

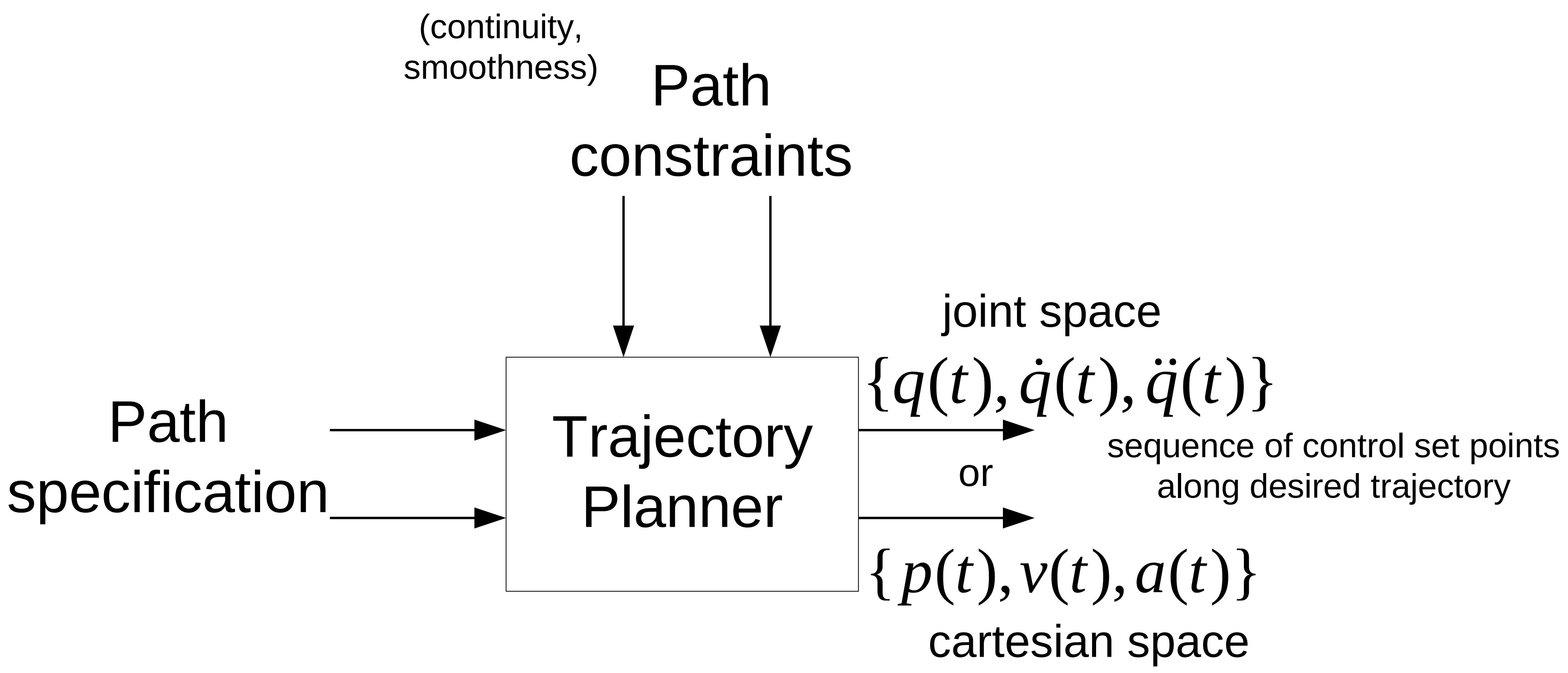


SCARA manipulator

Python Program

SECTION 6

Trajectory Planning



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