Motion planning and robots control

Salih ABDELAZIZ

Maître de Conférences à l'UM2

LIRMM, Département robotique

Montpellier, France

abdelaziz@lirmm.fr

OUTLINE

- Motion planning
  - Joint space
  - Cartesian space

- Motion control
  - Pose control based only on kinematics
  - Pose control based on dynamics (computed torque)

- Force control
  - Implicit force control
  - Explicit force control

OUTLINE

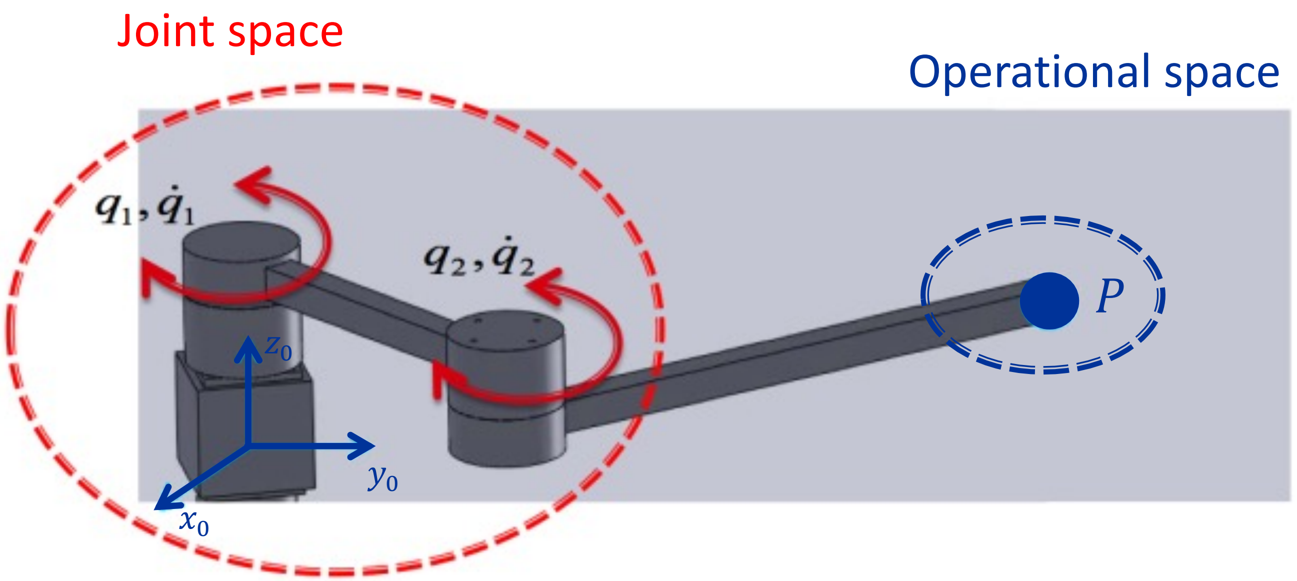
- Motion planning
  - Joint space
  - Cartesian space

- Motion control
  - Pose control based only on kinematics
  - Pose control based on dynamics (computed torque)

- Force control
  - Implicit force control
  - Explicit force control

DEFINITIONS

Let be a robot with m degrees of freedom and n joints



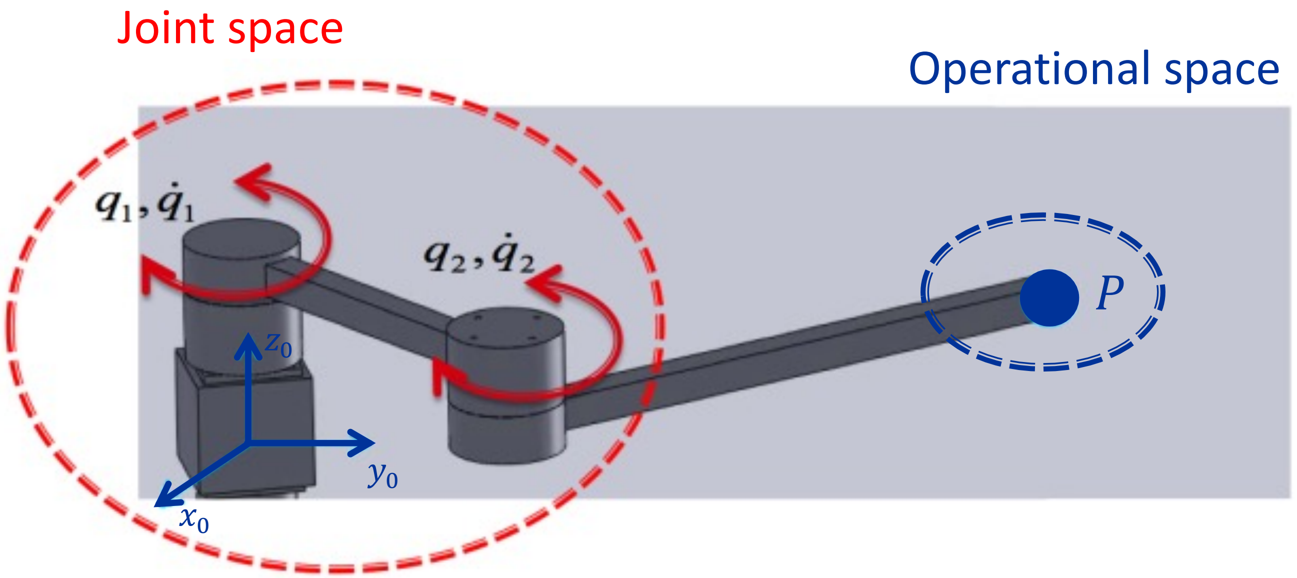
<u>Joint space :</u>

Vector of generalized coordinates: $\mathbf{q} = [q_1 \quad q_2 \quad \cdots \quad q_{\mathrm{n}}]^T$

Joint speed vector: $\dot{\mathbf{q}} = [\dot{q}_1 \quad \dot{q}_2 \quad \cdot \quad \dot{q}_{\mathrm{n}}]^T$

Vector of joint accelerations: $\ddot{\mathbf{q}} = [\ddot{q}_1 \quad \ddot{q}_2 \quad \dots \quad \ddot{q}_{\mathrm{n}}]^T$

DEFINITIONS



<u>Operational space :</u>

End effector pose: $\mathbf{x} \equiv$ position + orientation de P dans le repère $\mathcal{R}_0(x_0, y_0, z_0)$

Velocity of the end effector : $\dot{\mathbf{x}} = [\dot{x} \quad \dot{y} \quad \dot{z} \quad \omega_x \quad \omega_y \quad \omega_z]^T$

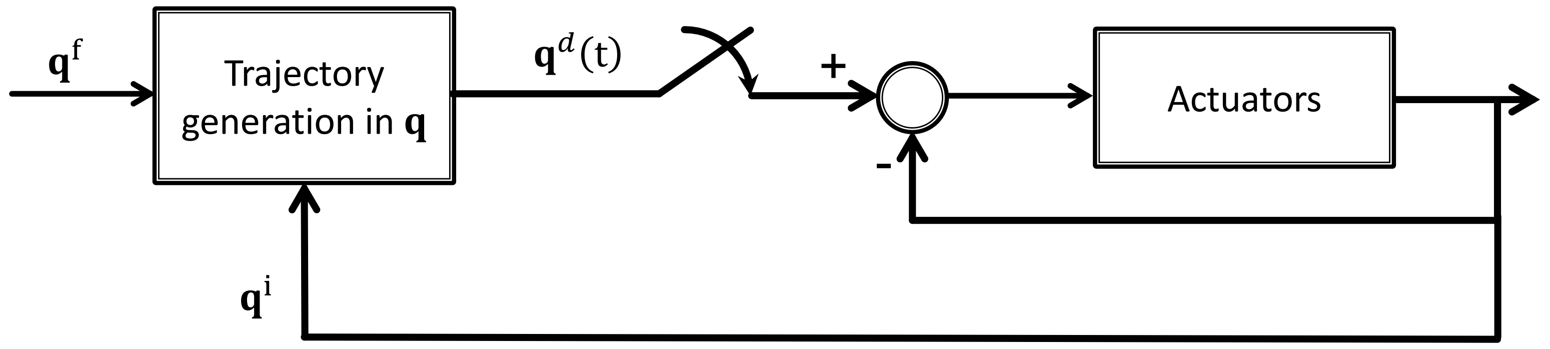
INTRODUCTION TO MOTION PLANNING

- Motion planning = function for calculating the robot's instructions in the form of successive positions of the end effector or the joints
-
- Motion class: movement between 2 points
  - via a free trajectory between points. *
  - With a constrained trajectory between the points (e.g. straight line)$^{\Delta}$

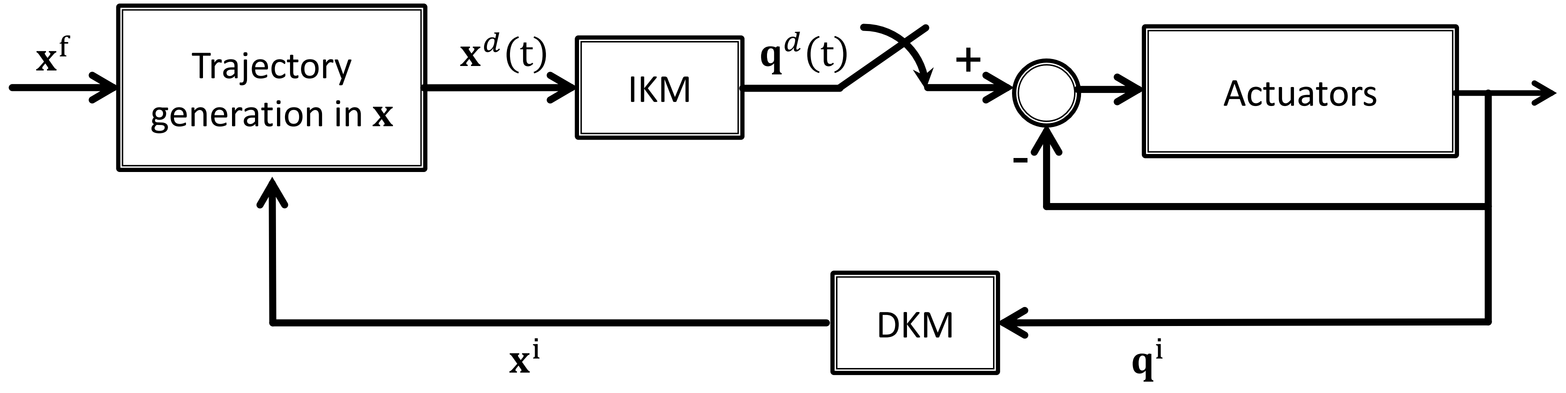
* can be performed in the joint space

$^{\Delta}$ can be done directly in the operational space

MOTION PLANNING : JOINT SPACE VS OPERATIONAL SPACE



Motion planning in the joint space



Motion planning in the operational space

MOVEMENT BETWEEN 2 POINTS IN THE JOINT SPACE

- Motion between $\mathbf{q}^{\mathrm{i}}$ and $\mathbf{q}^{\mathrm{f}}$ can be described as:

$$\mathbf{q}(\mathrm{t}) = \mathbf{q}^{\mathrm{i}} + \mathrm{r}(\mathrm{t})\mathbf{D} \quad 0 \le \mathrm{t} \le \mathrm{t_f}$$

with $\mathbf{D} = \mathbf{q}^{\mathrm{f}} - \mathbf{q}^{\mathrm{i}}$

$\mathbf{q}(\mathrm{t}) = \mathbf{q}^{\mathrm{d}}(\mathrm{t})$ for simplicity

$\mathbf{q}^{\mathrm{i}}$: vector of the generalized coordinates corresponding to an initial configuration

$\mathbf{q}^{\mathrm{f}}$: vector of the generalized coordinates corresponding to a final configuration

The values at the limits of the function $\mathrm{r}(\mathrm{t})$:

$$\begin{cases} \mathrm{r}(0) = 0 \\ \mathrm{r}(\mathrm{t_f}) = 1 \end{cases}$$

MOTION BETWEEN 2 POINTS IN THE JOINT SPACE

5th order polynomial equation:

- Adapted for high-speed robots or ones carrying large loads
- ➔ensure the continuity of acceleration in order to avoid exciting the mechanics
- Motion of class C^2

Six constraints to satisfy (2 in position, 2 in velocity and 2 for acceleration)

$$\mathbf{a}_0 = \mathbf{q}^i \quad ; \quad \mathbf{a}_1 = \mathbf{a}_2 = \mathbf{0} \quad ; \quad \mathbf{a}_3 = \frac{10}{\mathrm{t_f}^3}\mathbf{D} \quad ; \quad \mathbf{a}_4 = \frac{-15}{t_f{}^4}\mathbf{D} \quad ; \quad \mathbf{a}_5 = \frac{6}{t_f{}^5}\mathbf{D}$$

$$\mathrm{r(t)} = 10\left(\frac{\mathrm{t}}{t_f}\right)^3 - 15\left(\frac{\mathrm{t}}{\mathrm{t_f}}\right)^4 + 6\left(\frac{\mathrm{t}}{\mathrm{t_f}}\right)^5$$

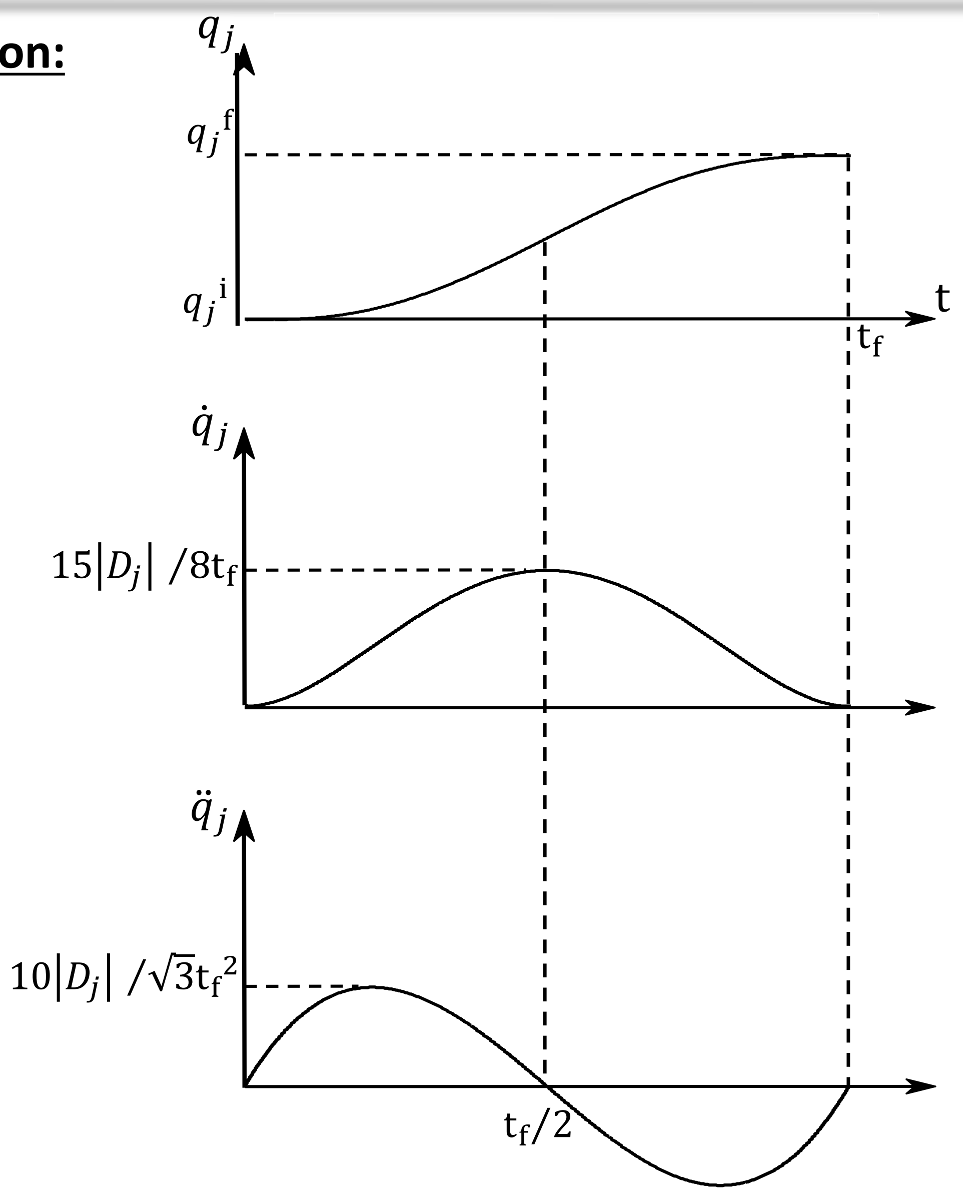
MOTION BETWEEN 2 POINTS IN THE JOINT SPACE

5th order polynomial equation:

Position

Velocity

Acceleration



MOTION BETWEEN 2 POINTS IN THE JOINT SPACE

The overall minimum time to be applied to all joints is constrained by the slowest joint:

$$\mathrm{t_f} = \max\left(\mathrm{t_{f_1}}, \mathrm{t_{f_2}}, \dots, \mathrm{t_{f_m}}\right)$$

Interpolation function	Minimum time
5th order polynomial equation	$\mathrm{t_{f_j}} = \max\left(\frac{15\lvert D_j\rvert}{8k_{v_j}}, \sqrt{\frac{10\lvert D_j\rvert}{\sqrt{3}k_{a_j}}}\right)$

MOTION BETWEEN 2 POINTS IN THE JOINT SPACE

- $\mathbf{k}_v = [k_{v_1} \quad \cdot \quad k_{v_\text{n}}]^T$: vector of maximum joints velocities (motors datasheet)
- $\mathbf{k}_a = [k_{a_1} \quad \cdot \quad k_{a_\text{n}}]^T$: vector of maximum joints accelerations (motors datasheet)

MOTION BETWEEN 2 POINTS IN THE JOINT SPACE

Benefits :

- Less online computations (no use of kinematic models)
- Movement not affected by the passage through the singular configurations
- Maximum speed and acceleration constraints directly deduced from the physical limits of the motors
-

Disadvantages :

- The geometry of the end effector trajectory in the operational space not predictable
- Risk of collisions if the robot operates in a cumbersome environment

MOVEMENT BETWEEN 2 POINTS IN OPERATIONAL SPACE

- We want to generate a trajectory between an initial pose (position and orientation) and a final pose of the end effector. For this, we define :

$\mathbf{x}^{\mathrm{i}} = [x^{\mathrm{i}} \quad y^{\mathrm{i}} \quad z^{\mathrm{i}}]^T$: initial position of the end effector

$\mathbf{x}^{\mathrm{f}} = [x^{\mathrm{f}} \quad y^{\mathrm{f}} \quad z^{\mathrm{f}}]^T$: final position of the end effector

$\mathbf{R}^{\mathrm{i}}$: rotational matrix corresponding to the current (initial) orientation of the end effector with respect to the reference frame

$\mathbf{R}^{\mathrm{f}}$: rotational matrix corresponding to the desired (final) orientation of the end effector with respect to the reference frame

- let ${}^0\mathbf{T}_e{}^{\mathrm{i}}$ and ${}^0\mathbf{T}_e{}^{\mathrm{f}}$ be two matrices describing the initial and final poses of the end effector:

$${}^0\mathbf{T}_e{}^{\mathrm{i}} = \begin{pmatrix} & \mathbf{R}^{\mathrm{i}} & & \mathbf{x}^{\mathrm{i}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad {}^0\mathbf{T}_e{}^{\mathrm{f}} = \begin{pmatrix} & \mathbf{R}^{\mathrm{f}} & & \mathbf{x}^{\mathrm{f}} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

MOVEMENT BETWEEN 2 POINTS IN OPERATIONAL SPACE

- Motion of the end effector is decomposed into :
  - Translational motion: straight line
  - A rotational movement θ around an axis $\mathbf{u}$

- The distance D to be covered by the end effector (translational movement) is:

$$D = \left\|\mathbf{x}^{\mathbf{f}} - \mathbf{x}^{\mathbf{i}}\right\|_2 = \sqrt{(x^{\mathrm{f}} - x^{\mathrm{i}})^2 + (y^{\mathrm{f}} - y^{\mathrm{i}})^2 + (z^{\mathrm{f}} - z^{\mathrm{i}})^2}$$

- The calculation of the rotation is done from:

$$\mathbf{R}^{\mathrm{i}}\ \mathbf{rot}(\mathbf{r}, \theta) = \mathbf{R}^{\mathrm{f}}$$

We then try to determine $\mathbf{r}$ and θ

MOVEMENT BETWEEN 2 POINTS IN OPERATIONAL SPACE

- The solution is calculated by:

$$\mathbf{rot}(\mathbf{r},\vartheta) = \left(\mathbf{R}^{\mathrm{i}}\right)^{T}\mathbf{R}^{\mathrm{f}} = \begin{pmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{pmatrix}$$

$$C\vartheta = \frac{1}{2}\left(x_x + y_y + z_z - 1\right)$$

$$S\vartheta = \frac{1}{2}\sqrt{\left(y_z - z_y\right)^2 + (z_x - x_z)^2 + \left(x_y - y_x\right)^2}$$

- The vector u and the angle ϑ are computed as:

$$\vartheta = atan2(S\vartheta, C\vartheta)$$

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \frac{1}{2\,S\vartheta}\begin{pmatrix} y_z - z_y \\ z_x - x_z \\ x_y - y_x \end{pmatrix}$$

MOVEMENT BETWEEN 2 POINTS IN OPERATIONAL SPACE

- Finally, the motion of the end effector is described by:
-

$$ {}^{0}\mathbf{T}_{E}{}^{d}(\mathrm{t}) = \begin{pmatrix} & \mathbf{R}^{d}(\mathrm{t}) & & \mathbf{x}^{d}(\mathrm{t}) \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Such that:

$$ \mathbf{x}^{d}(\mathrm{t}) = \mathbf{x}^{\mathbf{i}} + \mathrm{r}(\mathrm{t})\left(\mathbf{x}^{\mathrm{f}} - \mathbf{x}^{\mathrm{i}}\right) $$
$$ \mathbf{R}^{d}(\mathrm{t}) = \mathbf{R}^{\mathrm{i}}\ \mathbf{rot}(\mathbf{u}, \mathrm{r}(\mathrm{t})\vartheta) $$

with

$$ \mathbf{rot}(\mathbf{u}, \mathrm{r}(\mathrm{t})\theta) = \begin{pmatrix} u_x{}^2(1-\cos(\mathrm{r}\vartheta)) + \cos(\mathrm{r}\vartheta) & u_x u_y(1-\cos(\mathrm{r}\vartheta)) - u_z \sin(\mathrm{r}\vartheta) & u_x u_z(1-\cos(\mathrm{r}\vartheta)) + u_y \sin(\mathrm{r}\vartheta) \\ u_x u_y(1-\cos(\mathrm{r}\vartheta)) + u_z \sin(\mathrm{r}\vartheta) & u_y{}^2(1-\cos(\vartheta)) + \cos(\mathrm{r}\vartheta) & u_y u_z(1-\cos(\mathrm{r}\vartheta)) - u_x \sin(\mathrm{r}\vartheta) \\ u_x u_z(1-\cos(\mathrm{r}\vartheta)) - u_y \sin(\mathrm{r}\vartheta) & u_y u_z(1-\cos(\mathrm{r}\vartheta)) + u_x \sin(\mathrm{r}\vartheta) & u_z{}^2(1-\cos(\mathrm{r}\vartheta)) + \cos(\mathrm{r}\vartheta) \end{pmatrix} $$

r has the form:

$$ \mathrm{r} = \mathrm{r}(\mathrm{t}) = 10\left(\frac{\mathrm{t}}{\mathrm{t_f}}\right)^3 - 15\left(\frac{\mathrm{t}}{\mathrm{t_f}}\right)^4 + 6\left(\frac{\mathrm{t}}{\mathrm{t_f}}\right)^5 $$

MOVEMENT BETWEEN 2 POINTS IN OPERATIONAL SPACE

Benefits:

- Allows to control the geometry of the trajectory

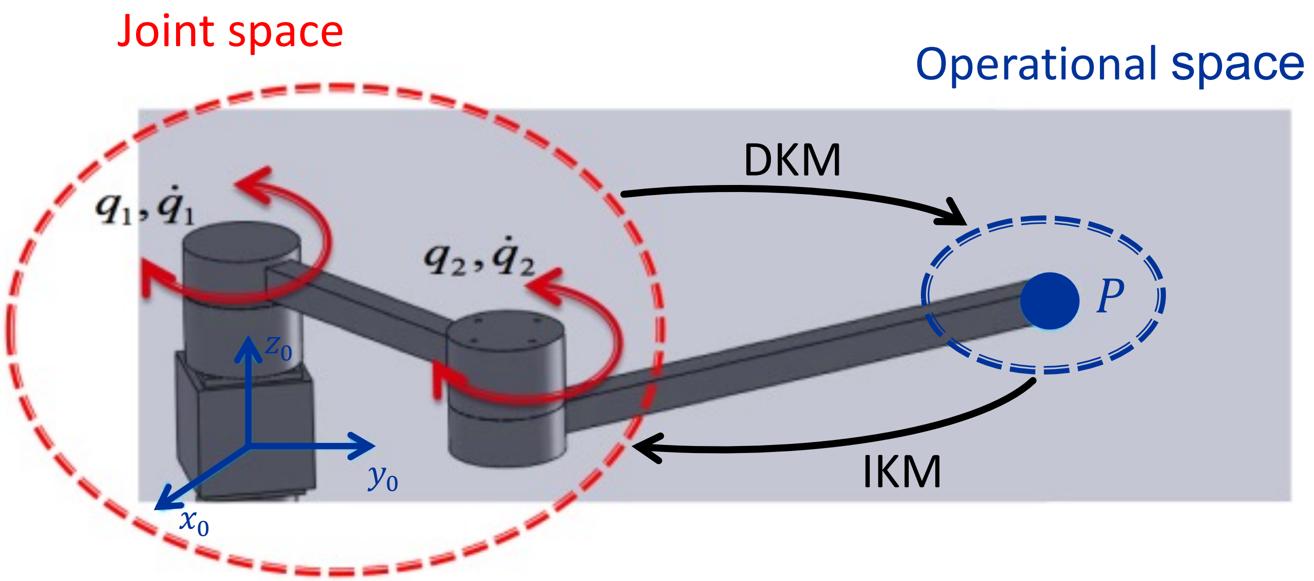
Disadvantage:

- Transformation into joint coordinates of each point in the trajectory
- Possibility of failure if passing through singular positions
- Possibility to fail if the trajectory points are not in the robot's workspace
- Speed and acceleration limits in the operational space vary depending on the robot configuration

OUTLINE

- Motion planning
  - Joint space
  - Cartesian space
- Motion control
  - Pose control based only on kinematics
  - Pose control based on dynamics (computed torque)
- Force control
  - Implicit force control
  - Explicit force control

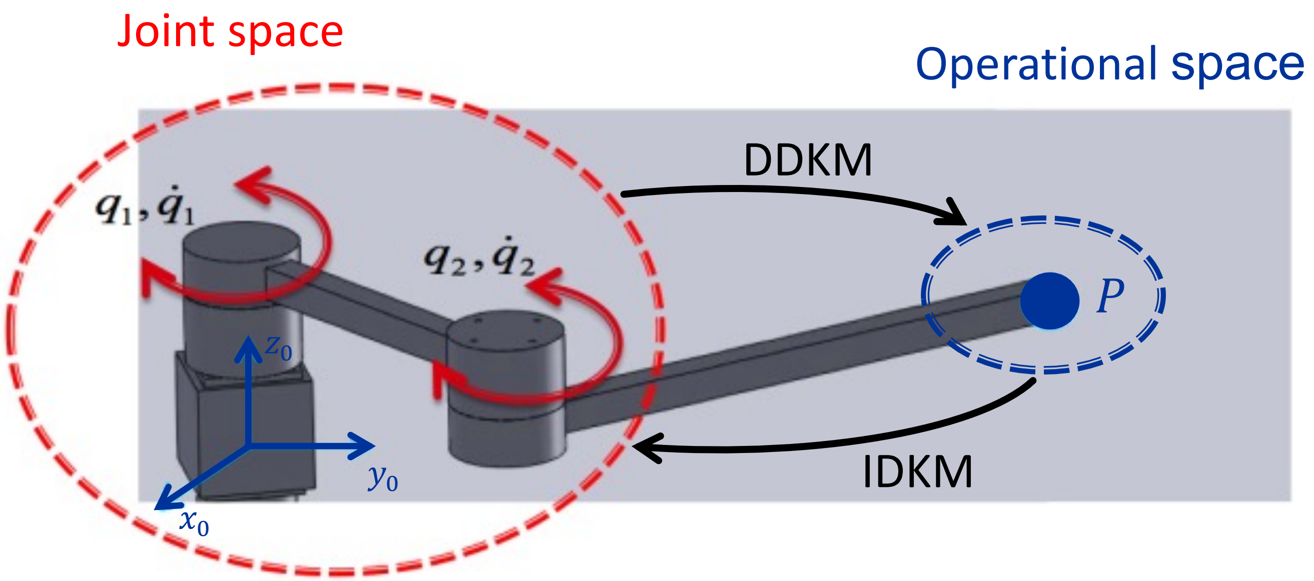
KINEMATIC MODELS (DIRECT AND INVERSE)



Direct kinematic model (DKM) : $\mathbb{x} = \boldsymbol{g}(\mathbf{q})$

Inverse kinematic model (IKM) : $\mathbf{q} = \boldsymbol{g}^{-1}(\mathbb{x})$

DIFFERENTIAL KINEMATIC MODELS (DIRECT AND INVERSE)



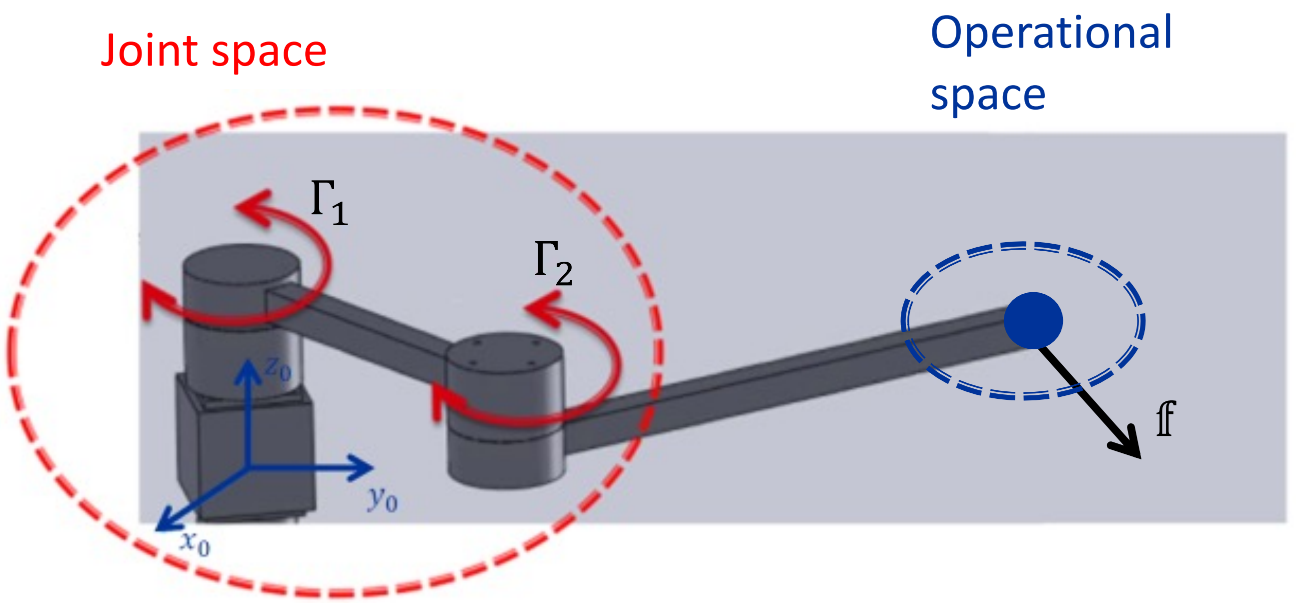
Direct differential kinematic model (DDKM) : $\dot{\mathbb{x}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}}$

$\mathbf{J}$: Jacobian matrix of the robot of dimension $\mathrm{m}\times\mathrm{n}$

Inverse differential kinematic model (IDKM) : $\dot{\mathbf{q}} = \mathbf{J}^{\#}(\mathbf{q})\,\dot{\mathbb{x}}$

if $\mathrm{m} = \mathrm{n} \Rightarrow \mathbf{J}^{\#} = \mathbf{J}^{-1}$ else $\mathbf{J}^{\#} = \mathbf{J}^{+}$ (Moore-Penrose inverse or pseudo-inverse)

STATIC MODEL



$\mathbf{\Gamma} = [\Gamma_1 \quad \Gamma_2 \quad . \quad \Gamma_\mathrm{n}]^T$: vector of joint torques (torques produced by actuators)

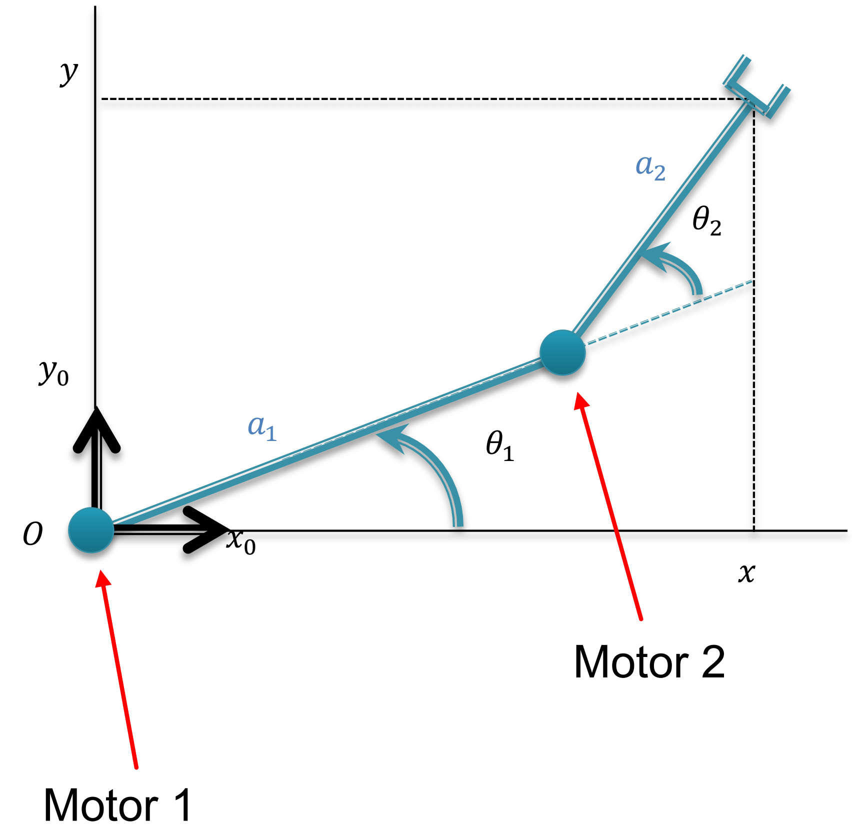
$\mathbb{f} = [f_x \quad f_y \quad f_z \quad m_x \quad m_y \quad m_z]^T$: force and moment applied on the end effector

$$\mathbf{\Gamma} = \mathbf{J}^T(\mathbf{q})\,\mathbb{f}$$

$\mathbf{J}^T$: Transpose of the Jacobian matrix of dimension $\mathrm{n}\times\mathrm{m}$

Position control

- 2 DOF planar robot

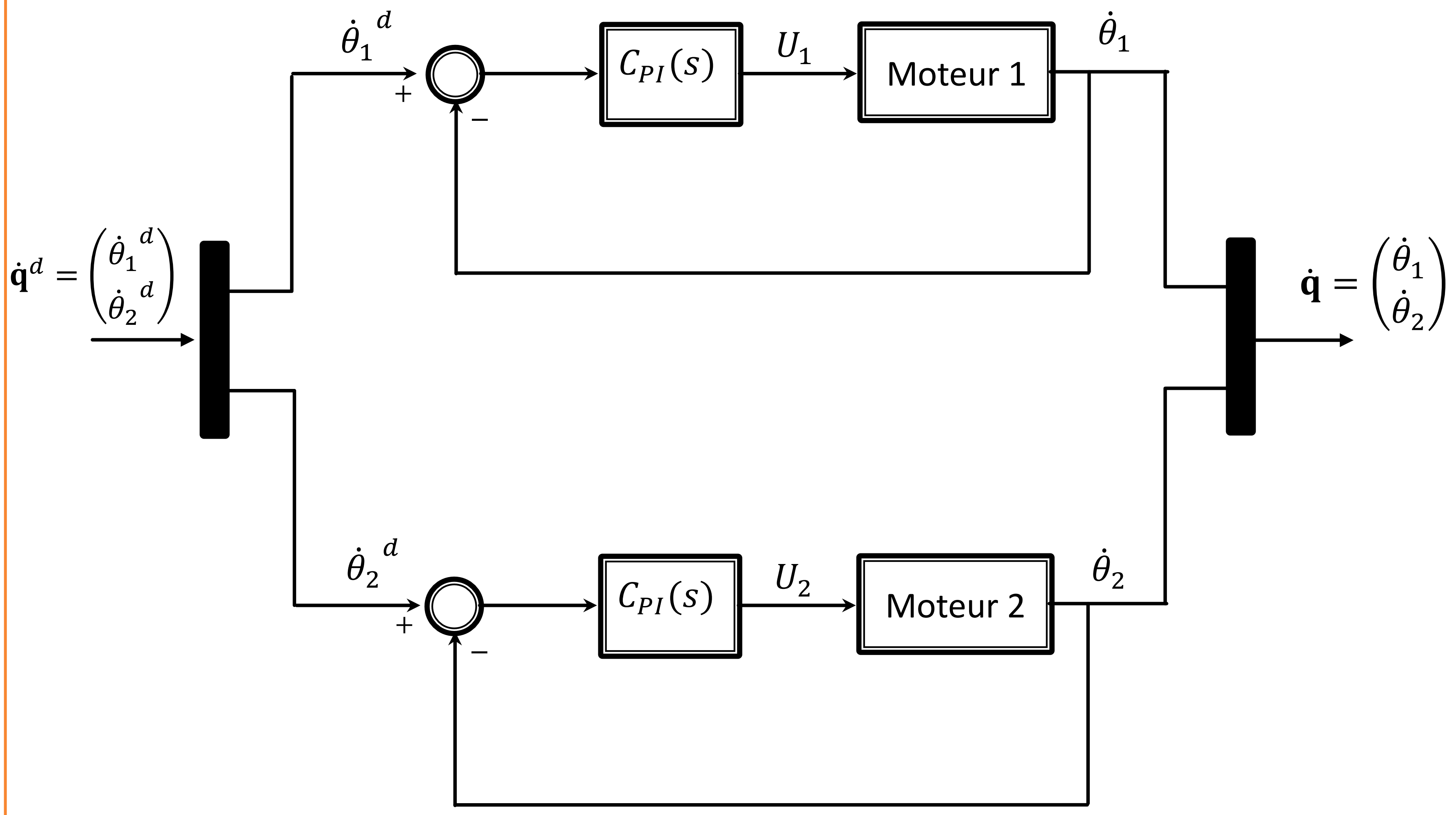


$$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$$

Position control

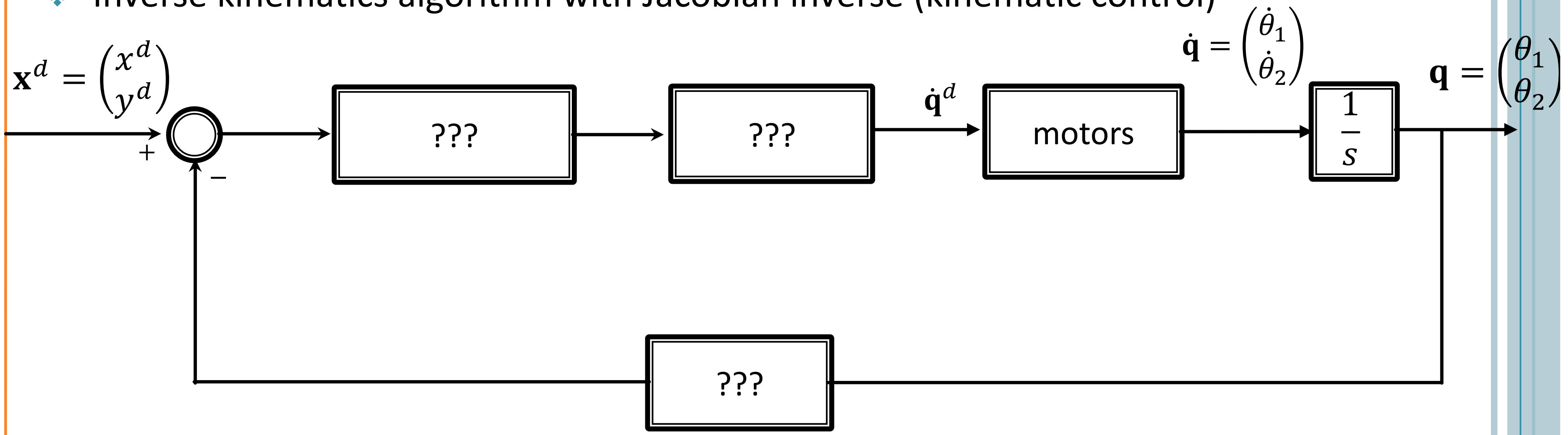
- Servomotors = motors controlled in velocity



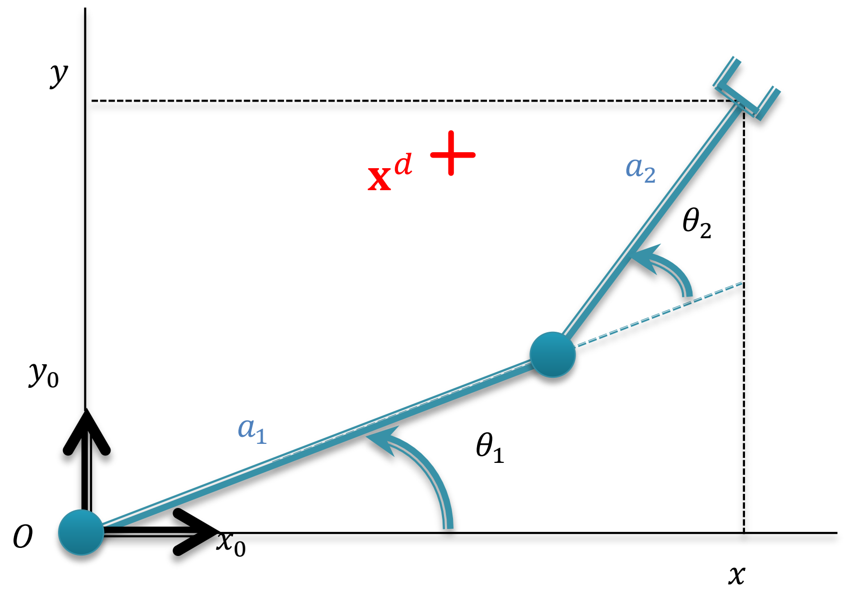
$\dot{\mathbf{q}}^d = \begin{pmatrix} \dot\theta_1{}^d \\ \dot\theta_2{}^d \end{pmatrix}$
$\dot\theta_1{}^d$
$+$
$-$
$C_{PI}(s)$
U_1
Moteur 1
$\dot\theta_1$
$\dot\theta_2{}^d$
$+$
$-$
$C_{PI}(s)$
U_2
Moteur 2
$\dot\theta_2$
$\dot{\mathbf{q}} = \begin{pmatrix} \dot\theta_1 \\ \dot\theta_2 \end{pmatrix}$

POSITION CONTROL

- Inverse kinematics algorithm with Jacobian inverse (kinematic control)

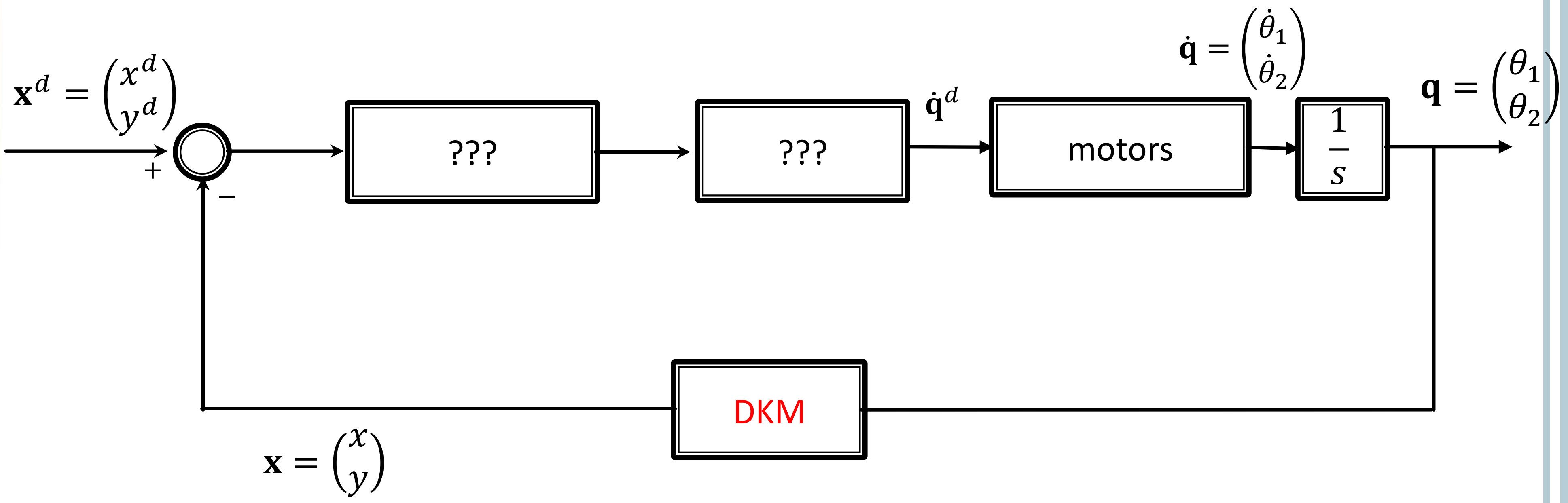


$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
+
−
???
???
$\dot{\mathbf{q}}^d$
motors
$\dot{\mathbf{q}} = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix}$
$\frac{1}{s}$
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
???



Position control

- Inverse kinematics algorithm with Jacobian inverse (kinematic control)



$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
+
−
???
???
$\dot{\mathbf{q}}^d$
motors
$\dot{\mathbf{q}} = \begin{pmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{pmatrix}$
$\frac{1}{s}$
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
DKM
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$

POSITION CONTROL

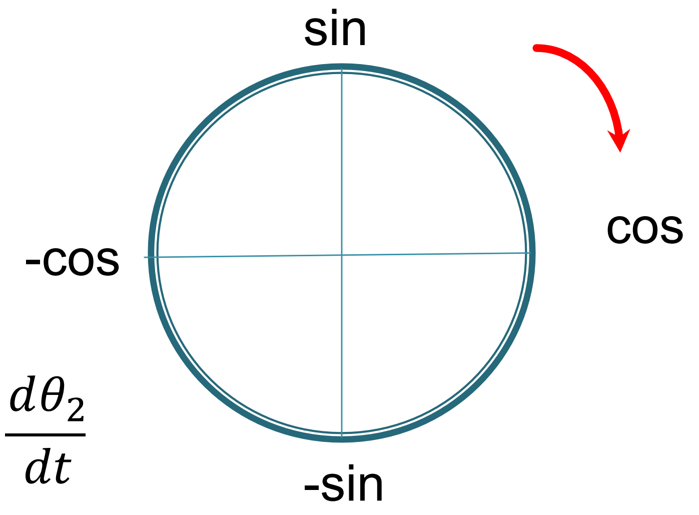
To establish the analytical Jacobian of the 2 DOF robot, it would be necessary to derive the direct kinematic model:

$$\text{DKM: } \begin{cases} x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2) \end{cases}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = ??$$

Position control

$$\begin{cases} \dfrac{d\sin\theta}{d\theta} = \cos\theta \\ \dfrac{d\cos\theta}{d\theta} = -\sin\theta \end{cases}$$



$$\frac{dx}{dt} = \frac{\partial x}{\partial\theta_1}\frac{d\theta_1}{dt} + \frac{\partial x}{\partial\theta_2}\frac{d\theta_2}{dt}$$
$$= \frac{\partial x}{\partial\theta_1}\dot{\theta}_1 + \frac{\partial x}{\partial\theta_2}\dot{\theta}_2$$

For the record:

$$x = a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2)$$

$$\boxed{\begin{aligned} \frac{\partial x}{\partial\theta_1} &= -a_1\sin\theta_1 - a_2\sin(\theta_1+\theta_2) \\ \frac{\partial x}{\partial\theta_2} &= -a_2\sin(\theta_1+\theta_2) \end{aligned}}$$

POSITION CONTROL

$$\begin{cases} \dfrac{d\sin\theta}{d\theta} = \cos\theta \\ \dfrac{d\cos\theta}{d\theta} = -\sin\theta \end{cases}$$

$$\begin{aligned}\frac{dy}{dt} &= \frac{\partial y}{\partial\theta_1}\frac{d\theta_1}{dt} + \frac{\partial y}{\partial\theta_2}\frac{d\theta_2}{dt} \\ &= \frac{\partial y}{\partial\theta_1}\dot{\theta}_1 + \frac{\partial y}{\partial\theta_2}\dot{\theta}_2\end{aligned}$$

For the record:

$$y = a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2)$$

$$\boxed{\begin{aligned}\frac{\partial y}{\partial\theta_1} &= a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) \\ \frac{\partial y}{\partial\theta_2} &= a_2\cos(\theta_1+\theta_2)\end{aligned}}$$

POSITION CONTROL

$$\frac{dx}{dt} = \{-a_1 \sin\theta_1 - a_2 \sin(\theta_1+\theta_2)\}\dot{\theta}_1 + \{-a_2 \sin(\theta_1+\theta_2)\}\dot{\theta}_2$$

$$\frac{dy}{dt} = \{a_1 \cos\theta_1 + a_2 \cos(\theta_1+\theta_2)\}\dot{\theta}_1 + \{a_2 \cos(\theta_1+\theta_2)\}\dot{\theta}_2$$

$$\dot{\mathbf{x}} = \begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}\frac{dx}{dt}\\ \frac{dy}{dt}\end{pmatrix} = \begin{pmatrix}-a_1 \sin\theta_1 - a_2 \sin(\theta_1+\theta_2) & -a_2 \sin(\theta_1+\theta_2)\\ a_1 \cos\theta_1 + a_2 \cos(\theta_1+\theta_2) & a_2 \cos(\theta_1+\theta_2)\end{pmatrix}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix}$$

$$= \mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}}$$

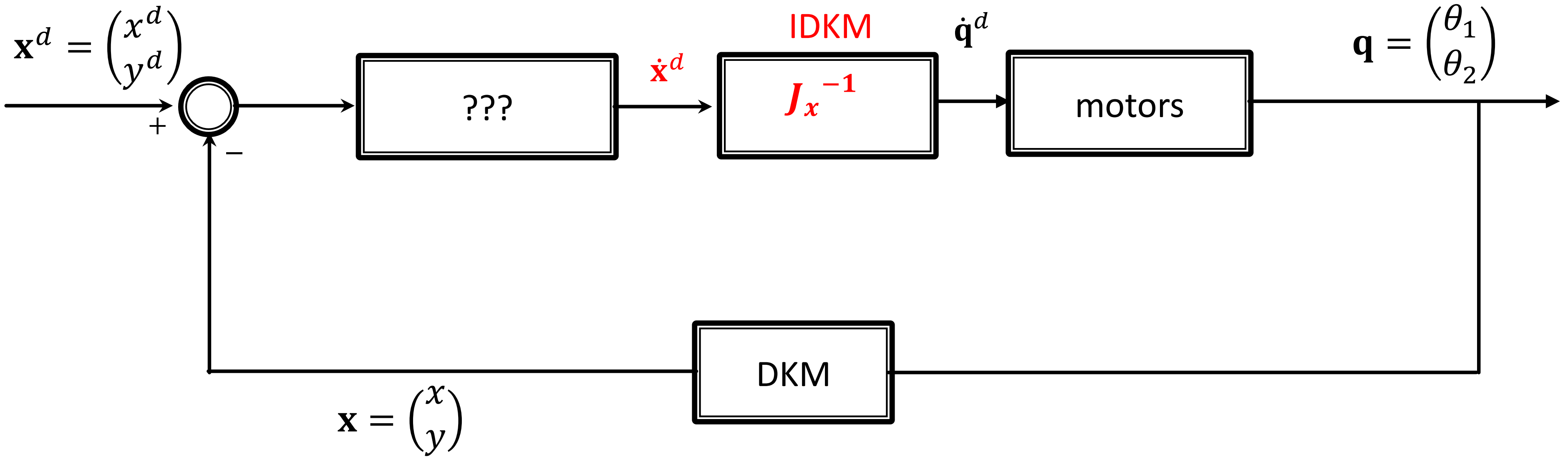
with:

$$\mathbf{J}_A(\mathbf{q}) = \mathbf{J}_x(\mathbf{q}) = \begin{pmatrix}-a_1 \sin\theta_1 - a_2 \sin(\theta_1+\theta_2) & -a_2 \sin(\theta_1+\theta_2)\\ a_1 \cos\theta_1 + a_2 \cos(\theta_1+\theta_2) & a_2 \cos(\theta_1+\theta_2)\end{pmatrix}$$

POSITION CONTROL

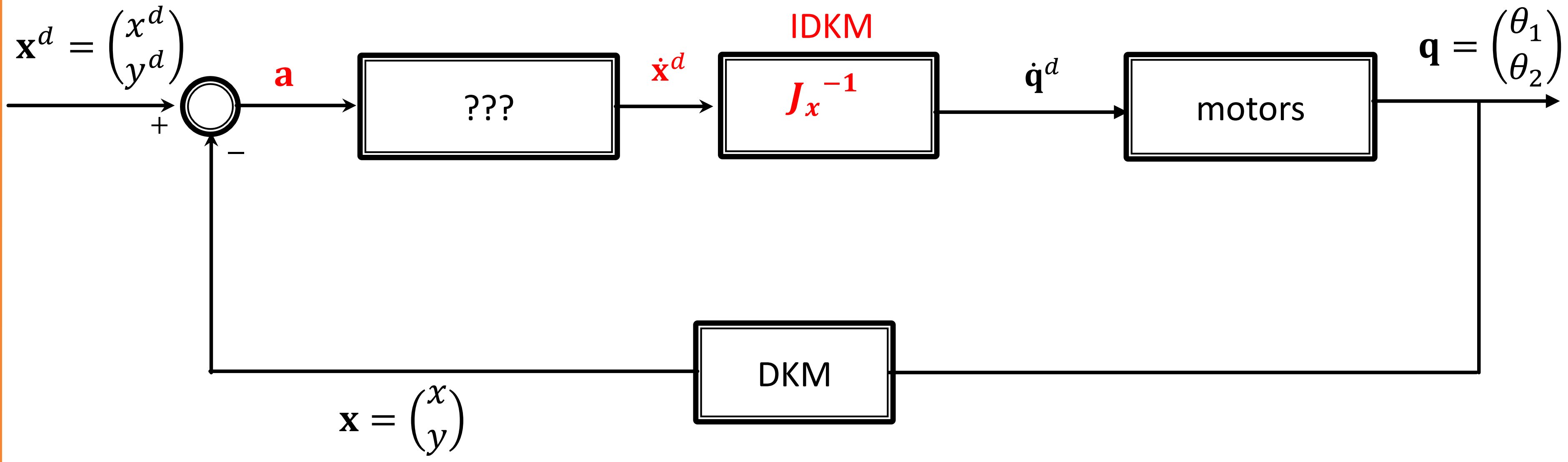
$$\dot{\mathbf{q}} = \mathbf{J}_x(\boldsymbol{q})^{-1}\dot{\mathbf{x}}$$

$\mathbf{J}_A(\boldsymbol{q})^{-1}$ is the inverse of the analytical Jacobian of the robot.



$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
+
−
???
$\dot{\mathbf{x}}^d$
IDKM
J_x^{-1}
$\dot{\mathbf{q}}^d$
motors
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
DKM
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$

Position control



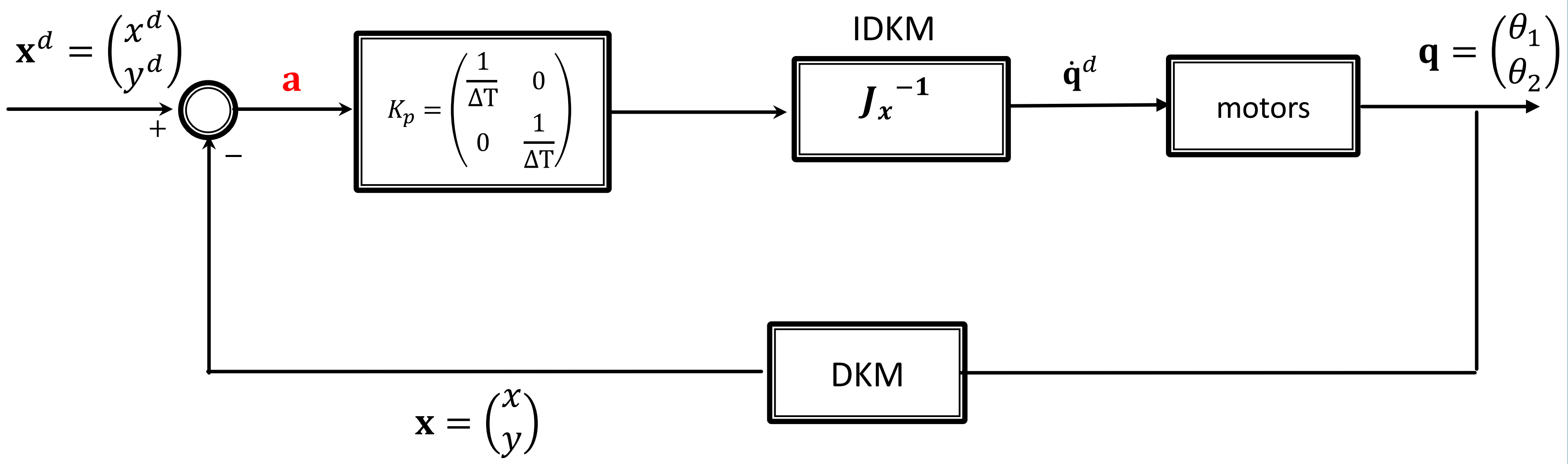
$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
+
−
a
???
$\dot{\mathbf{x}}^d$
IDKM
J_x^{-1}
$\dot{\mathbf{q}}^d$
motors
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
DKM
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$

$$\mathbf{a} = \mathbf{x^d} - \mathbf{x}$$

$$= \begin{pmatrix} x^d \\ y^d \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x^d - x \\ y^d - y \end{pmatrix}$$

$$\dot{\mathbf{x}}^d = \begin{pmatrix} \dot{x}^d \\ \dot{y}^d \end{pmatrix}$$

Position control



$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
+
−
a
$K_p = \begin{pmatrix} \frac{1}{\Delta T} & 0 \\ 0 & \frac{1}{\Delta T} \end{pmatrix}$
IDKM
J_x^{-1}
$\dot{\mathbf{q}}^d$
motors
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
DKM
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$

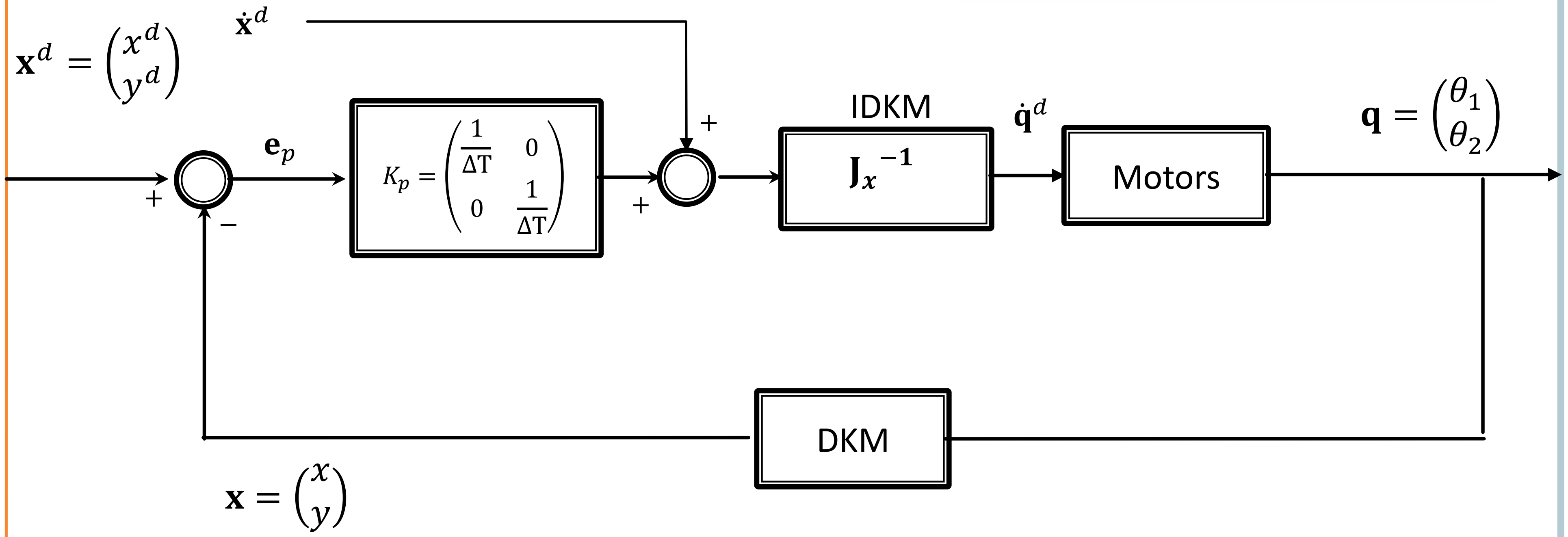
$$\mathbf{a} = \begin{pmatrix} x^d - x \\ y^d - y \end{pmatrix}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix}$$

Link between **a** and $\dot{\mathbf{x}}$ **??**

$$\dot{\mathbf{x}} = \begin{pmatrix} \frac{1}{\Delta T} & 0 \\ 0 & \frac{1}{\Delta T} \end{pmatrix} \begin{pmatrix} x^d - x \\ y^d - y \end{pmatrix} = \begin{pmatrix} \frac{x^d - x}{\Delta T} \\ \frac{y^d - y}{\Delta T} \end{pmatrix}$$

POSITION CONTROL



$\mathbf{x}^d = \begin{pmatrix} x^d \\ y^d \end{pmatrix}$
$\dot{\mathbf{x}}^d$
$\mathbf{e}_p$
$K_p = \begin{pmatrix} \frac{1}{\Delta T} & 0 \\ 0 & \frac{1}{\Delta T} \end{pmatrix}$
+
−
+
+
IDKM
$\mathbf{J}_x^{-1}$
$\dot{\mathbf{q}}^d$
Motors
$\mathbf{q} = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$
DKM
$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$

kinematic control

POSITION CONTROL

Control law:

$$\dot{\mathbf{q}}^d = \boldsymbol{J_A}^{-1}\left(\dot{\mathbf{x}}^d + \boldsymbol{K}_p \mathbf{e}_p\right)$$

Robot model:

$$\dot{\mathbf{q}} = \boldsymbol{J_A}^{-1}\dot{\mathbf{x}}$$

If the actuators are correctly controlled:

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}^d$$

This leads to:

$$\dot{\mathbf{x}} = \dot{\mathbf{x}}^d + \boldsymbol{K}_p \mathbf{e}_{\boldsymbol{p}}$$

$$\dot{\mathbf{e}}_p + \boldsymbol{K}_p \mathbf{e}_p = \mathbf{0}$$

If $\boldsymbol{K_p}$ is positive definite (usually diagonal) matrix, the system is *asymptotically stable*

POSITION CONTROL

- In the case of redundant manipulator, the control law $\dot{\mathbf{q}}^d = \boldsymbol{J_A}^{-\mathbf{1}}(\dot{\mathbf{x}}^d + \boldsymbol{K_p}\mathbf{e}_p)$ can be generalized into:

$$\dot{\mathbf{q}}^d = \boldsymbol{J_A}^{+}(\dot{\mathbf{x}}^d + \boldsymbol{K_p}\mathbf{e}_p) + (\boldsymbol{I_n} - \boldsymbol{J_A}^{+}\boldsymbol{J_A})\dot{\mathbf{q}}_\mathbf{0}$$

- The matrix $(\boldsymbol{I_n} - \boldsymbol{J_A}^{+}\boldsymbol{J_A})$ allows to generate internal motion described by $(\boldsymbol{I_n} - \boldsymbol{J_A}^{+}\boldsymbol{J_A})\dot{\mathbf{q}}_\mathbf{0}$ without changing the end effector position and orientation.

POSE CONTROL IN OPERATIONAL SPACE - POSITION ERROR

- The kinematic control scheme above presented uses the analytical Jacobian.
- For the position error, the expression is:

$$\mathbf{e}_p = \mathbf{x}^d - \mathbf{x}$$

$\mathbf{x}^d$ is the desired position of the end effector

$\mathbf{x}$ is the computed end effector position

- The first time derivative of the position error is:

$$\dot{\mathbf{e}}_p = \dot{\mathbf{x}}^d - \dot{\mathbf{x}}$$

- Orientation error depends on the particular representation of the end effector orientation, namely Euler angles, angle and axis, and unit quaternion

Pose control in operational space - orientation error using Euler angles

- Let’s define:

$\boldsymbol{\phi} = \begin{pmatrix} \varphi \\ \theta \\ \psi \end{pmatrix}$ as the orientation of the end-effector

- The orientation error is:

$$\mathbf{e}_O = \boldsymbol{\phi}^d - \boldsymbol{\phi}(\mathbf{q})$$

$\boldsymbol{\phi}^d$ represents the desired orientation

$\boldsymbol{\phi}(\mathbf{q})$ is the computed set of Euler angles (orientation of the end effector)

- The first time derivative of the orientation error is:

$$\dot{\mathbf{e}}_O = \dot{\boldsymbol{\phi}}^d - \dot{\boldsymbol{\phi}}$$

POSE CONTROL IN OPERATIONAL SPACE - ORIENTATION ERROR USING EULER ANGLES

- Assuming that neither kinematic nor orientation representation singularities occur, the Jacobian inverse solution for nonredundant manipulator is:

$$\dot{\mathbf{q}} = \boldsymbol{J_A}^{-\mathbf{1}} \begin{bmatrix} \dot{\mathbf{x}}^{\boldsymbol{d}} + \mathbf{K}_p \boldsymbol{e}_p \\ \dot{\boldsymbol{\phi}}^{\boldsymbol{d}} + \mathbf{K}_O \boldsymbol{e}_O \end{bmatrix}$$

$\mathbf{K}_p$ and $\mathbf{K}_O$ are positive definite matrices

Difficulties:

- Computation of the orientation variables $\boldsymbol{\phi}$ from the joint variables $\mathbf{q}$ is not easy, except for simple cases

- The computation of the orientation variables $\boldsymbol{\phi}$ requires the computation of the rotation matrix ${}^{0}\mathbf{R}_n$.
- In fact, only the dependence of ${}^{0}\mathbf{R}_n$ on $\mathbf{q}$ is always known, but not that of $\boldsymbol{\phi}$ on $\mathbf{q}$
- The use of inverse functions atan2 involves a non-negligible complexity in the analytical Jacobian, and occurrence of representation singularities present another drawbacks

Pose control in operational space - Orientation error using angle and axis

- From trajectory planning :

$$\mathbf{R}^d(\mathrm{t}) = \mathbf{R}^{\mathrm{i}}\ \mathbf{rot}(\mathbf{u}, \mathrm{r}(\mathrm{t})\vartheta)$$

- Let’s define the vectors $\mathbf{n}^d$, $\mathbf{s}^d$ and $\mathbf{a}^d$ as the column of the matrix $\mathbf{R}^d(\mathrm{t})$:

$$\mathbf{R}^d = (\mathbf{n}^d \quad \mathbf{s}^d \quad \mathbf{a}^d)$$

- Let’s define also define the vectors $\mathbf{n}^e$, $\mathbf{s}^e$ and $\mathbf{a}^e$ as the column of the matrix $\mathbf{R}^e$:

$$\mathbf{R}^e = (\mathbf{n}^e \quad \mathbf{s}^e \quad \mathbf{a}^e)$$

$\mathbf{R}^e$ represents the current orientation of the robot.

$$\mathbf{e}_O = \frac{1}{2}\left(\mathbf{n}^e \wedge \mathbf{n}^d + \mathbf{s}^e \wedge \mathbf{s}^d + \mathbf{a}^e \wedge \mathbf{a}^d\right)$$

POSE CONTROL IN OPERATIONAL SPACE - ORIENTATION ERROR USING ANGLE AND AXIS

- Differentiating $\mathbf{e}_O$ with respect to time gives:

$$\dot{\mathbf{e}}_O = \mathbf{L}^T \boldsymbol{\omega}_d - \mathbf{L}\boldsymbol{\omega}_e$$

With:

$$\mathbf{L} = \frac{-1}{2}\left(\mathbf{S}(\mathbf{n}^d)\mathbf{S}(\mathbf{n}^e) + \mathbf{S}(\mathbf{s}^d)\mathbf{S}(\mathbf{s}^e) + \mathbf{S}(\mathbf{a}^d)\mathbf{S}(\mathbf{a}^e)\right)$$

The matrix $\mathbf{S}$ is defined for a vector $\mathbf{n} = (n_x \quad n_y \quad n_z)^T$ so that :

$$\mathbf{S}(\mathbf{n}) = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix}$$

The control law can be defined as:

$$\dot{\mathbf{q}}^d = \mathbf{J}^{-1}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{x}}^d + \mathbf{K}_p \boldsymbol{e}_p \\ \mathbf{L}^{-1}(\mathbf{L}^T \boldsymbol{\omega}_d + \mathbf{K}_O \boldsymbol{e}_O) \end{bmatrix}$$

With: $\boldsymbol{\omega}_d = \mathbf{R}^{\mathrm{i}}\ \dot{\mathrm{r}}\ (t)\ \vartheta\ \mathbf{u}$

It is worth remarking that this control law works better than the one using Euler angles since it uses the geometric Jacobian instead of the analytical Jacobian, thus avoiding the occurrence of representation singularities

KINEMATIC SINGULARITIES

- The computation of the inverse or pseudo-inverse of the Jacobian is possible if and only if the Jacobian has full rank.
- In the case of a singularity (rank deficient), the system $\mathbb{V}= \mathbf{J}\,\dot{\mathbf{q}}$ contains linear dependent equations
- Solution (damped least square inverse):

$$\mathbf{J}^{*} = \mathbf{J}^{T}\left(\mathbf{J}\,\mathbf{J}^{\mathrm{T}} + k^{2}\mathbf{I}\right)^{-1}$$

Where k is a damping factor that renders the inversion better conditioned from a numerical viewpoint.

KINEMATIC SINGULARITIES

- To determine the singularities of the robot, it is enough to solve the equation $det(\mathbf{J}) = 0$ => rank deficient of the matrix $\mathbf{J}$

Example: robot 2 DOF

$$\mathbf{J} = \mathbf{J}_A = \begin{pmatrix} -a_1 \sin\theta_1 - a_2 \sin(\theta_1 + \theta_2) & -a_2 \sin(\theta_1 + \theta_2) \\ a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) & a_2 \cos(\theta_1 + \theta_2) \end{pmatrix}$$

Let's define:

- $S_1 = \sin\theta_1$
- $C_1 = \cos\theta_1$
- $S_{12} = \sin(\theta_1 + \theta_2)$
- $C_{12} = \cos(\theta_1 + \theta_2)$

$$\mathbf{J} = \begin{pmatrix} -a_1 S_1 - a_2 S_{12} & -a_2 S_{12} \\ a_1 C_1 + a_2 C_{12} & a_2 C_{12} \end{pmatrix}$$

KINEMATIC SINGULARITIES

$$det(\mathbf{J}) = a_2C_{12}(-a_1S_1 - a_2S_{12}) + a_2S_{12}(a_1C_1 + a_2C_{12}) = 0$$

$$\Rightarrow \; det(\mathbf{J}) = -a_1a_2(S_1C_{12} - C_1S_{12}) = 0$$

We have:

$$\sin(\alpha - \beta) = \sin\alpha \cos\beta - \cos\alpha \sin\beta$$

$$S_1C_{12} - C_1S_{12} = \sin\theta_1 \cos(\theta_1 + \theta_2) - \cos\theta_1 \sin(\theta_1 + \theta_2) = \sin(\theta_1 - (\theta_1 + \theta_2))$$

$$= \sin\big(-(\theta_2)\big) = -\sin(\theta_2) = -S_2$$

$$\Rightarrow det(\mathbf{J}) = a_1a_2S_2 = 0$$
$$\Rightarrow S_2 = 0$$
$$\Rightarrow \; \theta_2 = 0 \text{ or } \theta_2 = \pi$$

- The robot is in a singular configuration if $\theta_2 = 0$ or $\theta_2 = \pi$