INTRODUCTION ROBOTICS

LEZ.1 27/02 INTRODUCTION

Manipulators are composed by LINK connected by JOINTS → rotational / → prismatic

They have BASE, WRIST, END EFFECTOR.

DEGREE OF FREEDOM DoF: minimum number of variables to describe manipulator. In general **n° joints = n° DoF**

Each joint allow a DoF between 2 links

A KINEMATIC CHAIN can be closed or open (serial)

WORKSPACE: portion of the environment that the robot can reach, its shape depends on the type of the joints.

SENSORS → (computation process) → ACTUATION

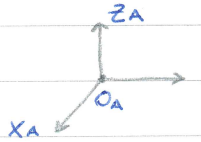
↳ Estimation, Perception, Control, Planning

LEZ.2 28/02 REPRESENTATION AND ORIENTATION

RIGID BODY ≜ 3D object where distance between any couple of points remain the same during a rotation.

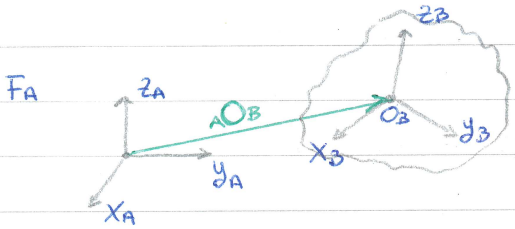
REFERENCE FRAME

TUPLE $(O_A, \hat{x}_A, \hat{y}_A, \hat{z}_A)$ $\hat{x}$: unit vector → $\|\hat{x}\| = 1$



Right-ended frame

Axis are orthogonal (⊥) to each other → $x_A \cdot y_A = x_A^T \cdot y_A = 0$



FA: reference word frame

B: body (fixed) frame

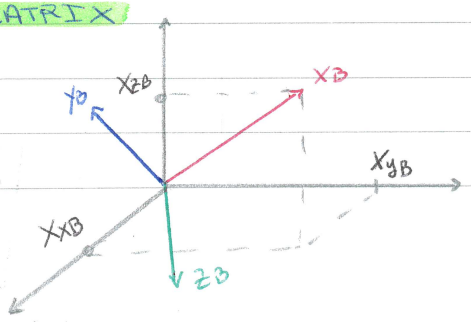
$^AO_B = [O_x, O_y, O_z]$

ROTATION Se ho le coordinate in B e le moltiplico per ^AR_B ottengo le coordinate in A

$^AR_B = [\,^AX_B \,;\, ^Ay_B \,;\, ^AZ_B\,]$ $\mathbb{R}^{3\times3}$ **ROTATION MATRIX**

↳ $^AX_B = [\,^AX_{xB}, {}^AX_{yB}, {}^AX_{zB}\,]$

The rotation matrix is obtained by projecting x_B, y_B, z_B into FA $^AR_B = [\,x_B \;\; y_B \;\; z_B\,]$



2

Se ${}^A R_B$ mi porta da B ad A, ${}^A R_B^T$ porta da A a B

Properties of Rotation matrix

${}^A R_B$ represents the orientation of frame B w.r.t frame A

1. 3 x 3 matrix → 9 entries
2. orthonormal = orthogonal + normal

	orthogonal: columns orthogonal to each other	normal: columns are unit vector
6 constraints	${}^A X_B^T {}^A Y_B = 0$	${}^A X_B^T {}^A X_B = 1$
9 entries − 6 cons. = 3 DOF	${}^A X_B^T {}^A Z_B = 0$	${}^A Y_B^T {}^A Y_B = 1$
	${}^A Y_B^T {}^A Z_B = 0$	${}^A Z_B^T {}^A Z_B = 1$

$R^{-1} = R^T \Rightarrow R^T R = R R^T = I$

3. $\det R = 1 \Rightarrow$ proper rotation, positive rotation is anti-clockwise (right-handed)

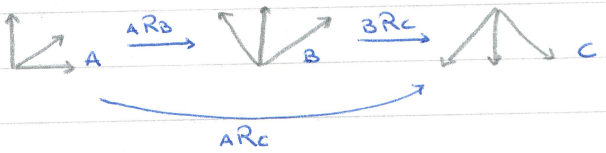
${}^A X_B$, ${}^A Y_B$, ${}^A Z_B$: Projections of frame B axes onto frame A

$${}^A R_B = \begin{bmatrix} X_A^T X_B & X_A^T Y_B & X_A^T Z_B \\ Y_A^T X_B & Y_A^T Y_B & Y_A^T Z_B \\ Z_A^T X_B & Z_A^T Y_B & Z_A^T Z_B \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \; y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \; z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

↳ canonical reference frame assigned to the world frame

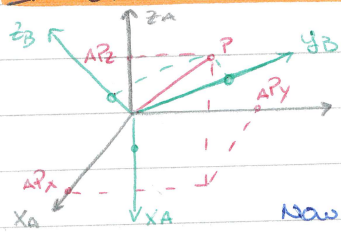
COMPOSITION OF ROTATIONS

${}^A R_B \, {}^B R_C = {}^A R_C$



NB : Product of rotation matrices is NOT commutative: ${}^A R_B {}^B R_C \neq {}^B R_C {}^A R_B$

INTERPRETATION: CHANGE OF COORDINATES OF VECTOR



${}^A R_B$ can be seen also as a linear operator that maps a vector from frame B to frame A

${}^A P = [{}^A P_x, {}^A P_y, {}^A P_z] = I_A P = {}^A \hat{X}_A \, {}^A P_x + {}^A \hat{Y}_A \, {}^A P_y + {}^A \hat{Z}_A \, {}^A P_z$

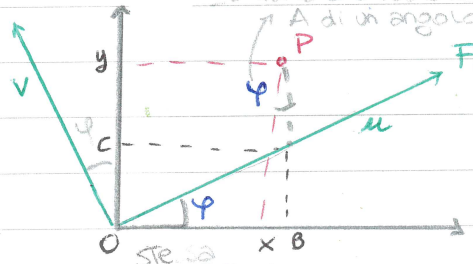
↳ P vista dal frame A

Now If we have available the coordinates of P w.r.t. frame B and we want To get ${}^A P$:

$${}^A P = \underbrace{[{}^A X_B : {}^A Y_B : {}^A Z_B]}_{{}^A R_B} \begin{bmatrix} {}^B P_x \\ {}^B P_y \\ {}^B P_z \end{bmatrix} = {}^A \hat{X}_B ({}^B P_x) + {}^A \hat{Y}_B ({}^B P_y) + {}^A \hat{Z}_B ({}^B P_z) \rightarrow {}^A P = {}^A R_B \, {}^B P$$

$${}^B P = {}^B R_A \, {}^A P$$

ELEMENTARY ROTATIONS



Il frame 3 è ruotato rispetto ad A di un angolo φ

- FA, FB have common origin
- ${}^{B}P = (u, v, w) \Rightarrow {}^{A}P = (x, y, z)$

$x = OB - XB = u\cos(\varphi) - v\sin(\varphi)$

$y = OC + CY = u\sin(\varphi) + v\cos(\varphi)$

$z = w \rightarrow$ does not change

↳ la rotazione avviene attorno a z

come trovare (x, y, z) conoscendo (u, v, w) e φ ?

Let's reorganize in matrix form: ← "forma matriciale"

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & v\cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

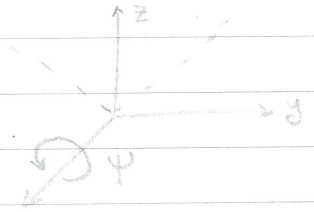
Elementary change of orientation about z axis

${}^{A}P = R_z(\varphi)\,{}^{B}P$

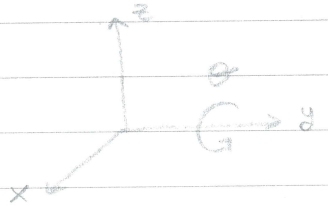
$R_z(-\varphi) = R_z(\varphi)^T \quad R_z^T(\varphi) =$ orient. of F_A w.r.t. F_B

If I rotate F_A by φ to get F_B then I need to rotate F_B of $-\varphi$ to get frame F_A

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}$$



$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$



Summary : **Rotation matrix Interpretation**

Same Rotation matrix ${}^{A}R_B$ can represents:

1. Orientation of a frame B w.r.t. a frame A
2. Change of coordinates of vector from rotated frame B to original frame A
3. Operator which rotates vectors → $v' = R(\psi, \theta, \varphi)\,v$

TIME DERIVATIVES R(t)

${}_A R_B \, {}_A R_B^T = I$ ↔ orthogonality

derivata di un prodotto ↓ d/dt calcolo la derivata di entrambi i lati dell'equazione

$\dot{{}_A R_B} \, {}_A R_B^T + {}_A R_B \, \dot{{}_A R_B}^T = \emptyset$ (matrice di zeri)

$\underbrace{\dot{{}_A R_B} \, {}_A R_B^T}_{S} + {}_A R_B \, \dot{{}_A R_B}^T = \emptyset$

$S + S^T = 0$ **SKEW-SYMMETRIC** (gli elementi sulla diagonale sono zero, gli altri sono opposti, $S_{ij} = -S_{ji}$)

$\dot{{}_A R_B} \, {}_A R_B^T = S \Rightarrow \boxed{\dot{{}_A R_B} = S \, {}_A R_B}$ *

HOW DOES S LOOK LIKE?

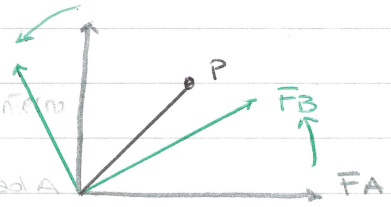
- consider constant vector P in a rotating frame B, its coord. in A are:

  ${}_A P = {}_A R_B(t) \, {}_B P$ (time varying) — Il frame B STA RUOTANDO attorno ad A

  ${}_A \dot{P} = \dot{{}_A R_B} \, {}_B P + {}_A R_B \, \dot{{}_B P} = \dot{{}_A R_B} \, {}_B P$

  $= 0$, P is a constant vector (wrt frame 2)

  ${}_A \dot{P} = S \, {}_A R_B \, {}_B P = S \, {}_A P$ (1)



- from mechanics: velocity of point in a frame rotating at ω is:

  ${}_A \dot{P} = {}_A\omega \times {}_A P$ (2)

Combining together equations (1) and (2) ⇒ $S = [{}_A\omega]_\times$

${}_A \dot{R}_B = [{}_A\omega]_\times \, {}_A R_B$ that is equivalent to: $\dot{{}_A R_B} = S \, {}_A R_B = {}_A\omega \times {}_A R_B$

Properties of skew-symmetric matrix

(I) square matrix

(II) $A = -A^T$ → 0 on-digonal, $A_{ij} = -A_{ji}$ off-diagonal

(III) 3×3 S can be used to represents cross product a×b as matrix multiplic.

$[a]_\times b \quad a = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad S(a) = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$ or $[a]_\times$

EULER ANGLES

Rotation matrices have some disadvantages as representation of orientation

(1) NON-MINIMAL REPRESENTATION : 9 entrie - 6 constraints = 3 ind. variables

(2) INTERPOLATION: position → $P(s) = P_i + (P_f - P_i)s$ $s \in [0,1]$

orientation → $R(s) = R_i + (R_f - R_i)s$? NO!

(3) DEFINITION OF ORIENTATION ERROR: $R_{error} \neq R^{des}(t) - R_{act}(t)$

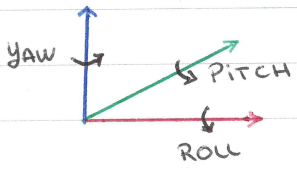
how can be able to treat orient / pos errors the same?

EULER ANGLES

(φ, θ, ψ) is NOT a vector but just a sequence of angles.

They define a sequence of rotations around independent axis.

There are 27 possible combinations but only 12 of these are valid (I need the ones with independent axis).



x: Roll ψ

y: Pitch θ

z: YAW φ

- We we'll consider only moving axes : frame w.r.t. each rotation occurs is about the current frame, not the original one (fixed axis)
- In robotics, the mostly used sequence is $\Phi = (z\,y'\,x'')$ → not preserve orthogonality

$R(\Phi) = R_z(\varphi) R_{y'}(\theta) R_{x''}(\psi)$ Rotazioni attorno moving axes

DIRECT PROBLEM $\Phi \rightarrow R$

$$\begin{bmatrix} c\varphi c\theta & c\theta s\theta s\psi - s\varphi c\psi & c\varphi s\theta c\psi + s\varphi s\psi \\ s\varphi c\theta & s\varphi s\theta s\psi + c\varphi c\psi & s\varphi s\theta c\psi - c\varphi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix}$$

solution for direct problem $\Phi(z\,y'\,x'')$

Elementary Rotations (seen before)

With successive (3 rotations) post-multiply the previous rotation with th foll. one

INVERSE PROBLEM $R \rightarrow \Phi$

$$\varphi = \arctan_2(r_{21}, r_{11})$$

$$\theta = \arctan_2\left(-r_{31} \pm \sqrt{r_{32}^2 + r_{33}^2}\right) \rightarrow \text{Give multiple solutions}$$

$$\psi = \arctan_2\left(\frac{r_{32}}{c\theta}, \frac{r_{33}}{c\theta}\right) \rightarrow c\theta = 0 \text{ gives singularity}$$

↳ Restrict angle domain

NOTE : The sequence xyz about fixed axes is equivalent to the sequence about moving axes $(z\,y'\,x'')$

$$\bar{R}_x(\psi)\bar{R}_y(\theta)\bar{R}_z(\varphi) = R_z(\varphi)R_{y'}(\theta)R_{x''}(\psi)$$

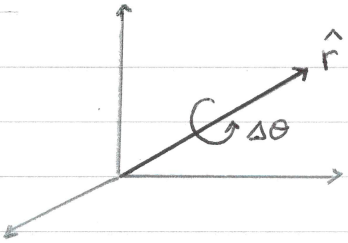
All Non-Elementary Matrix — All Elementary Matrix

LEZ. 3 06/03/2025 **DIRECT KINEMATICS OF A MANIPULATOR**

Kinematics: describes the motion without considering the forces that cause it

Direct kinematics: compute position/orientation of a frame as a function of joint variables q.

LEZ. 3 06/03/2025 **ANGLE-AXIS REPRESENTATION**



vettore unitario. Indica la direzione dell'asse attorno al quale avviene la rot.

$\hat{r}$: axis about which rotation is made

$\Delta\theta$: angle, magnitude of rotation

1 angle + $\hat{r}$ in 3 dimension } 4 PARAMETERS

constraints ci riportano a 3 parametri

$\hat{r}$ is a unit vector: $\|\hat{r}\| = 1$ constraint $\rightarrow r_x^2 + r_y^2 + r_z^2 = 1$ (they are not independent)

In order to map to rotation matrix we use **Rodriguez Formula**:

$$R(\Delta\theta, \hat{r}) = \hat{r}\hat{r}^T + (I - \hat{r}\hat{r}^T)\cos(\Delta\theta) + [\hat{r}]_\times \sin(\theta)$$

skew symmetric

DIRECT PROBLEM

Da Angle-Axis a rotation Matrix

NOTE: $(\Delta\theta, \hat{r})$ and $(-\Delta\theta, -\hat{r})$ give same result, so mapping is not injective.

Solving inverse problem gives 2 solutions.

ci sono 2 combinazioni Angle-Axis che portano alla stessa Matrice di rotazione

Properties of $R(\Delta\theta, \hat{r})$

- Axis is invariant to rotation: $R(\theta, \hat{r})\hat{r} = \hat{r}$

  $\hat{r}$ is an eigenvector associated to an eigenvalue $\lambda = 1$
- $\det(R) = \prod \lambda_i = 1$
- $\mathrm{Tr}(R) = \sum r_{ii} = 1 + 2\cos(\theta)$
- R is not injective map $R(\theta, \hat{r}) = R(-\theta, -\hat{r})$

INVERSE PROBLEM (non l'ha spiegato in classe, c'è nelle slide)

$$r = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \frac{1}{2\sin(\Delta\theta)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}$$

In $\Delta\theta = 0$, $\Delta\theta = \pi$ we have singularity

UNIT QUATERNION

Represent a point on unit sphere in $\mathbb{R}^4$ → 4 PARAMETERS (non minimal)

NO SINGULARITY

$$Q = \begin{bmatrix} \eta \\ \varepsilon \end{bmatrix}$$ scalar part in $\mathbb{R}$, vector part in $\mathbb{R}^3$

UNIT QUATERNION

To eliminate singular cases of axis/angles and euler angles, we can use UNIT QUATERNION representation Q.

- Unit norm : $\eta^2 + \varepsilon^T\varepsilon = 1 \rightarrow \|Q\| = 1$
- $(\theta, \hat{r})$ and $(-\theta, -\hat{r})$ give the same quaternion Q → duplication disappears

COMPUTATIONS: - NO rotation $Q = (1, 0^T)$

- Inverse $Q^{-1} = \bar{Q} = \begin{bmatrix} \eta \\ -\varepsilon \end{bmatrix}$

- Multiplication $Q_3 = Q_2 \otimes Q_1 = \begin{bmatrix} \eta_1\eta_2 - \varepsilon_1^T\varepsilon_2 \\ \eta_1\varepsilon_2 + \eta_2\varepsilon_1 + \varepsilon_1 \times \varepsilon_2 \end{bmatrix}$

Angle-Axis to Quaternion

$$Q = \begin{bmatrix} \eta = \cos\left(\frac{\Delta\theta}{2}\right) \\ \varepsilon = \hat{r}\sin\left(\frac{\Delta\theta}{2}\right) \end{bmatrix}$$

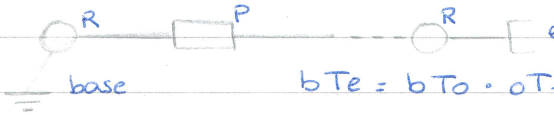
Quaternions to angle-axis

$$\begin{bmatrix} \Delta\theta = 2\arccos(\eta) \\ \hat{r} = \frac{\varepsilon}{\|\varepsilon\|} = \frac{\varepsilon}{\sqrt{1-\eta^2}} \end{bmatrix}$$

Rodrigues Formula for Quaternions $R = (\eta^2 - \varepsilon^T\varepsilon)I + 2\varepsilon\varepsilon^T - 2\eta[\varepsilon]_\times$

LEZ. 4 07/03/2025 DIRECT KINEMATICS of a MANIPULATOR

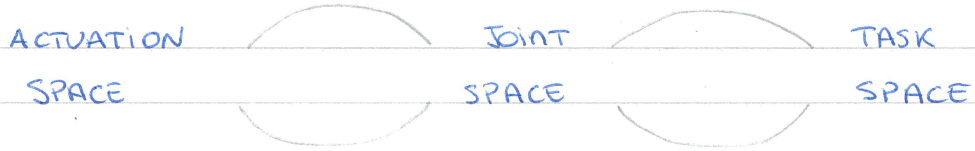
Kinematics: describes the motion without considering the forces that cause it.



^bT_e ← GOAL

$^bT_e = {}^bT_0 \cdot {}^0T_1 \cdot \dots {}^{n-1}T_n \cdot {}^nT_e$ Every transformation depends on 1 variable

KINEMATIC SPACES



To have a motion of the joints we need to actuate the actuators.
We assume 1 motor per joint, so dim(JOINT SPACE) = dim(ACTUATION SPACE).
Describe the joint space with generalized coordinates. $q = \begin{cases} \theta \text{ revolute} \\ d \text{ prismatic} \end{cases}$

KINEMATICS: PARAMETRIZATIONS

JOINT SPACE $q = (q_1, \dots, q_n)$ ⟷ (Direct: $r = f(q)$, Inverse: $q = f^{-1}(r)$) ⟷ TASK SPACE $r = (r_1, \dots, r_m)$

JOINT SPACE: space of the joints defined by the vector of joint variables $q = (q_1, \dots q_n)$

GENERALIZED COORDINATES: minimum and ambiguous number of coordinates that represents the motion taking into account kinematic constraints (joints) n = #Degrees of Freedom (DoF) = # robot joints

TASK SPACE

Space where a task is defined and trajectories are planned. Direct kinematics is defined for each task.

Typical Industrial Case $n = m = 6$

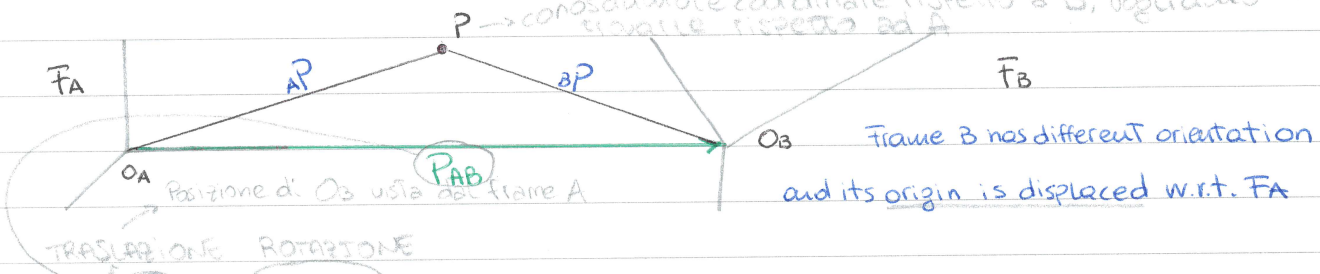
For a serial manipulators

$q \rightarrow DK \rightarrow r$ Easy, always have solution $r = fn(q)$

$r \rightarrow IK \rightarrow q$ Hard, NO or ∞ solutions

HOMOGENEOUS TRANSFORMS

Unique **ROTO-TRANSLATION** operator that combines the representation of position + orientation (POSE) of a rigid body w.r.t. another one.



Frame B has different orientation and its origin is displaced w.r.t. F_A

$^AP = {}^AP_{AB} + {}^AR_B\,{}^BP$ → if we rewrite in homogeneous coordinates:

$$^AP_R = \begin{bmatrix} ^AP \\ 1 \end{bmatrix} = \begin{bmatrix} ^AR_B & ^AP_{AB} \\ 0\,0\,0 & 1 \end{bmatrix} \begin{bmatrix} ^BP \\ 1 \end{bmatrix} = {}^AT_B\,{}^BP_R$$

Vector represented in hom. coord. with scale factor 1

Homogeneous transform

Linear Operator

^BP_R → Rappresenta una direzione

$\begin{bmatrix} ^BP_x \\ ^BP_y \\ ^BP_z \\ 1 \end{bmatrix}$ → scale factor

→ Per fare operazioni tra matrici e vettori

INTERPRETATIONS OF T: ① Describe (pose) of a frame wrt another

② Transform the representation of a geometric vector from a frame to another

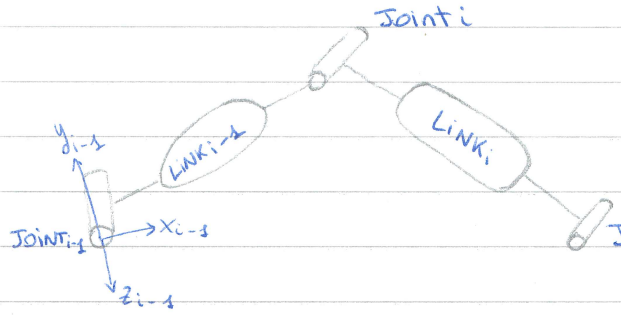
- ^AT_B transforms a geometric vector into another geometric vector
- ^AT_B is a linear relationship → I can compose several T

PURE ROTATION: $T(R,0) = \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix}$ **PURE TRASLATION:** $T(0,t) = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix}$

Denhavit - Hartenberg

Choose frames along the robot such that $^{i-1}T_i$ → q_i variable
→ 4 parameters
→ Each $^{i-1}T_i$ has the same functional form

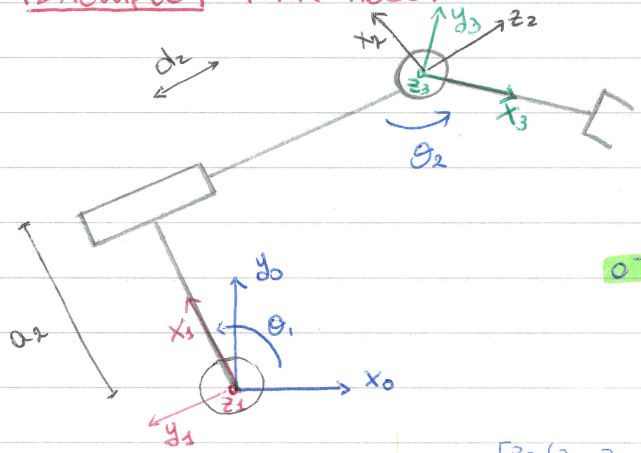
URDF CONVENTION



LINK i FRAME: • origin coincident with Joint i
• X axis along the main direction of Link$_i$

JOINT i FRAME: • z_i along rot. axis
• x_i same direction of x_{i-1}

Example RPR Robot

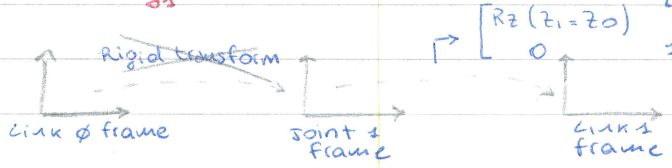


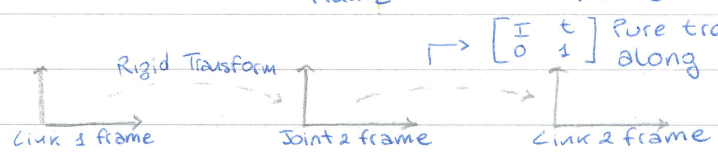
z: ↻ θ Angle between x_{i-1} and x_i
z: ↕ d Distance between x_{i-1} and x_i

Homegeneous Transforms (Between Links)

$$ {}^0T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

rotation around z_0
coincident origins





Merge together rigid transform and traslational transform due to prismatic joint

$$ {}^1T_2 = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 0 & -1 & -d_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Axes of frame 2 seen from 1 (expressed in 1)

$x_2 = x_1$
$y_2 = z_1$
$z_2 = -y_1$

Rotation

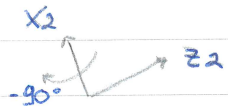
a_2 along x_1
0 along y_1
0 along z_1

Traslation

+ [-d_2 along y_1 ; pure joint translation]

Rigid transform

To compute from frame 2 to frame 3 first set $q_3 = 0$ (x_2, x_3 axes become coinci.)



(1) $-90°$ around x_2 (first apply the rigid transform):

$$R_x(-90°) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

(2) θ_3 around z_3

$$R_z(q_3) = \begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^2R_3 = R_x(-90°)\,R_z(q_3) = \begin{bmatrix} C_3 & -S_3 & 0 \\ 0 & 0 & 1 \\ -S_3 & -C_3 & 0 \end{bmatrix}$$

Traslation: no → $^2T_3 = \begin{bmatrix} C_3 & -S_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -S_3 & -C_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

$^0T_2 = {}^0T_1\,{}^1T_2$ → $^0T_3 = {}^0T_2\,{}^2T_3 = \begin{bmatrix} C_1C_3 - S_1S_3 & -C_1S_3 - S_1C_3 & 0 & a_2C_1 + d_2S_1 \\ S_1C_3 + C_1S_3 & -S_1S_3 + C_1C_3 & 0 & a_2S_1 - d_2C_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

NOTE: TRIGONOMETRIC RELATIONSHIPS

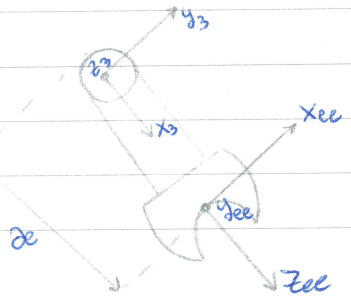
- $C_\alpha C_\beta - S_\alpha S_\beta = C(\alpha+\beta)$
- $S_\alpha C_\beta + C_\alpha S_\beta = S(\alpha+\beta)$

$$^0T_3 = \begin{bmatrix} C_{13} & -S_{13} & 0 & a_2C_1 + d_2S_1 \\ S_{13} & C_{13} & 0 & a_2S_1 - d_2C_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

frame 3 is obtained with a rotation around z_0

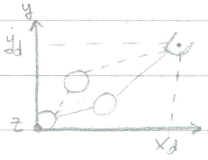
position of end-effector in F_0 (base frame)

To obtain the transform up to the end effector we need to define a frame for that and compute the rigid transform → $^3T_{ee} = \begin{bmatrix} 0 & 0 & 1 & d_e \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ → $^0T_{ee} = {}^0T_3\,{}^3T_{ee}$

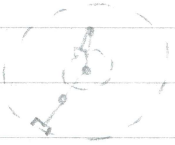


LEZ.5 13/03/2025 INVERSE KINEMATICS

PROBLEMS

(1) Multiple and finite solutions:  to reach the desired pose of end effector we have 2 configurations.

(2) Infinite solutions: redundant robots, where $n > m$. n is the number of joint m is task (variables)

(3) No solutions:  In this case, manipulator can't reach point outside bigger circle or inside smaller circle.

SOLVE IK → 2 APPROACHES

SYMBOLIC

- write down equations of FWD KIN
- try to solve them explicitally
  - Trovare una soluzione per q in termini della posizione desiderata P

NUMERICAL

- no need of analytical expression of FWD KIN
- Find q by evaluating FK several times → process iterative
- Formulated as an optimization problem
  - Minimizzare l'errore tra pos. attuale EE e posizione desiderata

ANALYTICAL WAY TO SOLVE IK

I can write solutions in closed form only in special cases, it's not easy to caiputo and to do this i need to compute DK.

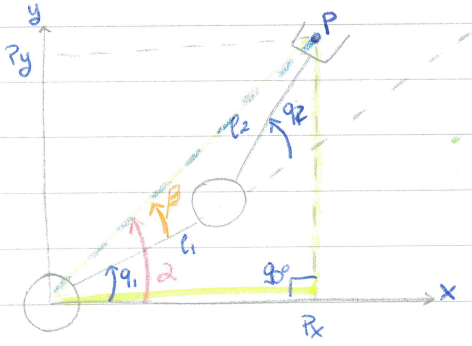
THEOREM (from Pfeifer): the IK admits a closed form solution if:

(1) robot has 6 DOF

(2) 3 consecutive axes intersect in 1 point

↳ this usually happen in a spherical wrist

EXAMPLE IK PLANAR RR



(1) WRITE DK $\begin{cases} P_x = l_1 C_1 + l_2 C_{12} \\ P_y = l_1 S_1 + l_2 S_{12} \end{cases}$

(2) COMPUTE IK

Intuition: distance between P and O only depends on q_2

$$P_x^2 + P_y^2 = l_1^2 C_1^2 + l_2^2 C_{12}^2 + l_1^2 S_1^2 + l_2 S_{12}^2 + 2 l_1 l_2 (C_1 C_{12} + S_1 S_{12})$$
$$\overset{!}{=} l_1^2 + l_2^2 + 2 l_1 l_2 C_2$$
$$\hookrightarrow C_2 = \frac{P_x^2 + P_y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$S_2 = \pm\sqrt{1 - C_2^2} \rightarrow \text{2 SOLUTIONS}$$
$$q_2 = \text{atan2}(S_2, C_2)$$

$q_1 = \alpha - \beta$ $\quad \alpha = \text{atan2}(P_y, P_x)$ $\quad \beta = \text{atan2}(l_2 S_2, l_1 + l_2 S_{12})$

$$\hookrightarrow q_1 = \text{atan2}(P_y, P_x) - \text{atan2}(l_2 S_2, l_1 + l_2 S_{12})$$

NUMERICAL IK

2 Methods $\begin{cases} \text{POSE LEVEL} \\ \text{VELOCITY LEVEL (differential kinematics)} \end{cases}$

SOLUTION METHODS

Analytical	Numerical
• closed form	• mandatory if $n > m$ (∞ solutions)
• requires intuitions (geometric)	• slower but easier to setup
• require to handle multiple solutions	• use analytic Jacobian of DK $J_R(q) = \frac{\partial f(q)}{\partial q}$
• only in specific cases	• Newton method / Gradient method
•	

LEZ. 6 14/03/2025 DIRECT DIFFERENTIAL KINEMATICS

Express the end-effector motion (linear/angular) velocity as a function of the joint velocities. This velocity mapping can be obtained in two ways:

(1) From the joint space to chosen task space

$\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix}$ velocity of n joints

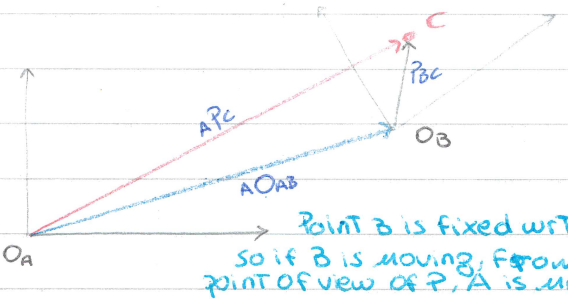
$$\dot{r} = \frac{\partial f(q)}{\partial q}\dot{q} = J_A(q)\dot{q}$$ Analytical Jacobian

(2) From joint space to cartesian workspace (linear and angular velocity).

$V = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$ velocità lineare $\quad \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$ velocità angolare

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J(q)\dot{q}$$ Geometric Jacobian

KINEMATIC ANALYSIS WITH MOVING FRAME



- F_A fixed, F_B moving
  - traslating: $^A\dot{O}_{AB}$
  - rotating: $^A\omega$
- Position of C attached to moving frame F_A is:

Point B is fixed w.r.t. B, $^AP_C = {}^AO_{AB} + {}^AP_{BC}$ (1)

so if B is moving, from the point of view of P, A is moving

The velocity of point C expressed in F_A is obtained by taking the time derivative of (1)

$$^AV_C = \frac{d}{dt}{}^AP_C = {}^A\dot{O}_{AB} + \frac{d}{dt}{}^AP_{BC}$$

The position of point C attached to moving frame F_B is:

$$^AP_C = {}^AO_{AB} + {}^AR_B\,{}^BP_{BC}$$ (2)

Taking the time derivative of (2)

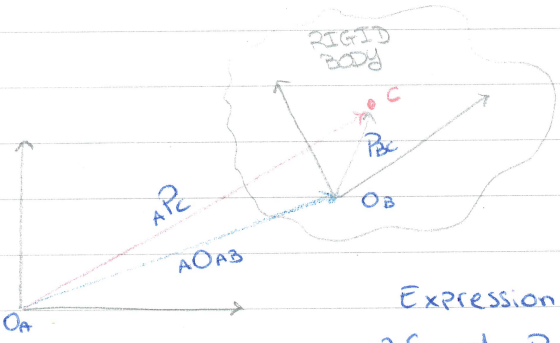
$$^AV_C = \frac{d}{dt}{}^AP_C = {}^A\dot{O}_{AB} + {}^AR_B\,{}^B\dot{P}_{BC} + {}^A\dot{R}_B\,{}^BP_{BC}$$

- ^AV_C: velocity of C seen from F_A
- $^A\dot{O}_{AB}$: velocity of F_B seen from F_A
- $^AR_B\,{}^B\dot{P}_{BC}$: variation due to motion of C w.r.t. F_B
- $^A\dot{R}_B\,{}^BP_{BC}$: variation due to rotation of F_B

* Now point C is moving also w.r.t frame B

* Now $^AP_{BC} = {}^AR_B\,{}^BP_{BC}$ because C is moving also w.r.t. B. ^AR_B converts C coord. expressed in B, in coord. expressed in A. This is necessary because if C is moving w.r.t B, his positional vector $^BP_{BC}$ changes.

Velocity of a point on a rigid body (e.g. link)



C, F_B ∈ Rigid Body.

↳ there is no relative motion of C w.r.t. F_B due to rigidity constraints (i.e. C and F_B are rigidly attached)

Expression of velocity taken from (2):

$$^AV_C = \frac{d}{dt}\,{}^AP_C = {}^A\dot{O}_{AB} + {}^AR_B\,{}^B\dot{P}_{BC} + {}^A\dot{R}_B\,{}^BP_{BC}$$

$^AR_B\,{}^B\dot{P}_{BC}$ → $^BP_{BC}$ is constant

$^A\dot{R}_B\,{}^BP_{BC}$ → $^A\omega \times {}^AR_B$

$$= {}^A\dot{O}_{AB} + {}^A\omega \times {}^AR_B\,{}^BP_{BC}$$

$$= {}^A\dot{O}_{BB} + {}^A\omega \times {}^AP_{BC}$$

VELOCITY OF A RIGID BODY LINK OF A MANIPULATOR

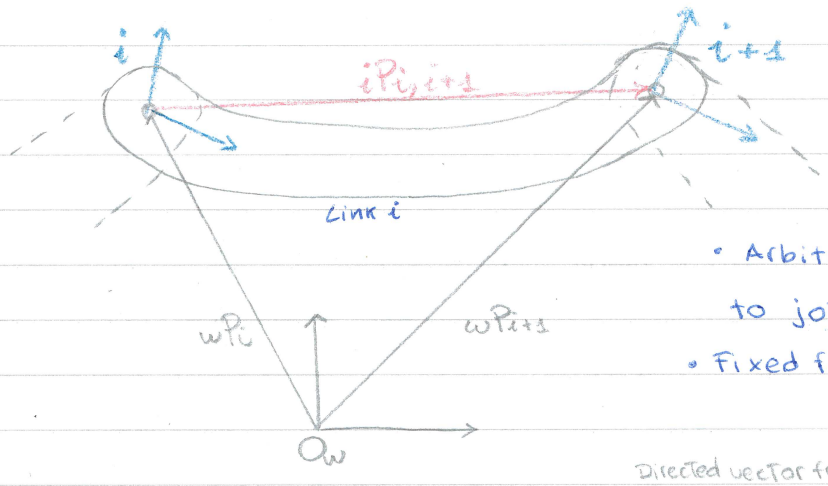
There are 2 sets of differential quantities:

(1) LINEAR v and ANGULAR ω velocity of end-effector
- these 2 quantities ∈ vector space
- they can be obtained by adding the contributions of each single linear (v_i) and angular (ω_i) joint velocity

(2) the ROTATIONAL RATE of the end-effector $\dot{\Phi}$
- Φ are the angles representing the sequence of rotations corresponding to a minimal representation → RPY roll-pitch-yaw
- $\dot{\Phi} \notin$ vector space, it cannot be obtained by adding contributions of each single joint.

IMPORTANT: In general $\dot{\Phi} \neq \omega$

After computing expression for linear (v_{i+1}) and angular (ω_{i+1}) velocities for a single rigid body link w.r.t. its predecessor (i), it's possible to compute the velocity of any link as long as the DK of the complete chain are known.



- Arbitrary link i attached to joints i and $i+1$
- Fixed frame w

Directed vector from frame i to $i+1$, express in frame i

Position of link $i+1$ in terms of link i in w: $^wP_{i+1} = {}^wP_i + {}^wR_i\,{}^iP_{i,i+1}$

↳ Differentiate (derivative of ^wR_i): $^w\dot{P}_{i+1} = {}^w\dot{P}_i + {}^wR_i\,{}^i\dot{P}_{i,i+1} + {}^w\omega_i \times {}^wR_i\,{}^iP_{i,i+1}$

$^w\dot{P}_{i+1} = {}^wv_{i+1}$, $^w\dot{P}_i = {}^wv_i$, ${}^wR_i\,{}^i\dot{P}_{i,i+1} = {}^wv_{i,i+1}$, ${}^wR_i\,{}^iP_{i,i+1} = {}^wP_{i,i+1}$

Linear velocity of link $i+1$ as a function of the linear and angular velocities of i:

$$^wv_{i+1} = {}^wv_i + {}^wv_{i,i+1} + {}^w\omega_i \times {}^wP_{i,i+1}$$

velocità di traslazione — velocità di P_{i+1} dovuta alla rotazione del link i

To obtain the expression for the angular velocity it is worth starting from the composition of rotation matrices: $^wR_{i+1} = {}^wR_i\,{}^iR_{i+1}$

↳ Time derivative: $^w\dot{R}_{i+1} = {}^w\dot{R}_i\,{}^iR_{i+1} + {}^wR_i\,{}^i\dot{R}_{i+1}$

$^w\dot{R}_i = {}^w\omega_i \times {}^wR_i$

Skew-Symmetric: $[{}^w\omega_i \times]\,{}^wR_i = S({}^w\omega_i)\,{}^wR_i$

$$S({}^w\omega_{i+1})\,{}^wR_{i+1} = S({}^w\omega_i)\,{}^wR_i\,{}^iR_{i+1} + {}^wR_i\,S({}^i\omega_{i,i+1})\,{}^iR_{i+1}$$
$$\overset{!}{=} S({}^w\omega_i)\,{}^wR_{i+1} + S({}^wR_i\,{}^i\omega_{i,i+1})\,{}^wR_{i+1}$$

Post-multiplying by $^{i+1}R_w$ ↳ $S({}^w\omega_{i+1}) = S({}^w\omega_i) + S({}^w\omega_{i,i+1})$

$$^w\omega_{i+1} = {}^w\omega_i + {}^w\omega_{i,i+1}$$

Summarizing (DK):

$$\begin{cases} {}^wv_{i+1} = {}^wv_i + {}^wv_{i,i+1} + {}^w\omega \times {}^wP_{i,i+1} \\ {}^w\omega_{i+1} = {}^w\omega_i + {}^w\omega_{i,i+1} \end{cases}$$

□ predecessor link ○ type of joint

These expressions are used to compute velocities from base link to end-effector.

Prismatic joint

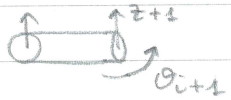
$^{w}\omega_{i,i+1} = 0$

$^{w}v_{i,i+1} = \dot{d}_{i+1} z_{i+1}$

$^{w}v_{i+1} = {}^{w}v_{i+1} + \dot{d}_{i+1} z_{i+1} + {}^{w}\omega_i \times {}^{w}P_{i,i+1}$

$^{w}\omega_{i+1} = {}^{w}\omega_i$ transmitting velocity to the following link

Revolute joint



$^{w}\omega_{i,i+1} = \dot{\theta}_{i+1} z_{i+1}$

$^{w}v_{i,i+1} = {}^{w}\omega_i \times {}^{w}P_{i,i+1}$

$^{w}v_{i+1} = {}^{w}v_i + {}^{w}\omega_i \times {}^{w}P_{i,i+1}$

$^{w}\omega_{i+1} = {}^{w}\omega_i + \dot{\theta}_{i+1} z_{i+1}$

VELOCITY OF THE END-EFFECTOR

The goal is to describe the end-effector linear velocity $^{w}v_e$ and angular velocity $^{w}\omega_e$ as a function of the joint positions $q(t)$ and joint vel. $\dot{q}(t)$

$$\begin{bmatrix} {}^{w}v_e \\ {}^{w}\omega_e \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} {}^{w}v_i \\ \sum_{i=1}^{n} {}^{w}\omega_i \end{bmatrix} = J(q)\dot{q} = \begin{bmatrix} J_P(q) \\ J_O(q) \end{bmatrix} \dot{q}$$

($J_P(q)$: linear contribution; $J_O(q)$: angular contribution)

$J_P(q)\dot{q} = \sum_{i=1}^{n} J_{Pi}\dot{q}$ $\quad J_O(q)\dot{q} = \sum_{i=1}^{n} J_{Oi}\dot{q}$

↳ The expression corresponding to their contribution change depending on type of joint

Linear velocity contribution we can write the velocity of the end effector by obtaining the derivative of its position using chain rule

↳ $^{w}v_e = {}^{w}\dot{P}_e = \sum_{i=1}^{n} \frac{\partial {}^{w}P_e}{\partial q_i} \dot{q}_i = \sum_{i=1}^{n} J_{Pi}\dot{q}_i$

We need to compute the contribution to the linear velocity of joint $i+1$ to the origin of the frame attached to the end-effector.

Prismatic Joint

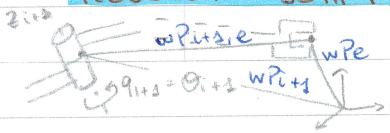
$^{w}v_e = {}^{w}v_{i,i+1}$

$^{w}v_{i,i+1} = \dot{d}_{i+1} z_{i+1}$ → $\dot{q}_{i+1}$

$\dot{q}_{i+1} J_{Pi+1} = \dot{d}_{i+1} z_{i+1}$

$J_{Pi+1} = z_{i+1}$

Revolute Joint



$^{w}v_{i+1,e} = {}^{w}\omega_{i,i+1} \times {}^{w}P_{i+1,e} = \dot{\theta}_{i+1} z_{i+1} \times ({}^{w}P_e - {}^{w}P_{i+1})$ ($\dot{\theta}_{i+1}$ → $\dot{q}_{i+1}$)

$\dot{q}_{i+1} J_{Pi+1} = \dot{\theta}_{i+1} z_{i+1} \times ({}^{w}P_e - {}^{w}P_{i+1})$

$J_{Pi+1} = z_{i+1} \times ({}^{w}P_e - {}^{w}P_{i+1})$

We have also 2 cases to compute angular velocity contribution

Prismatic joint

Link i: $^wv_{i+1} = {}^wv_i + {}^w\omega_{i,i+1}$

$^w v_e = {}^w v_n = \sum_{i=1}^{n} {}^w\omega_{i,i+1} = \sum_{i=1}^{n} J_{O_{i+1}} \dot{q}_{i+1}$

${}^w\omega_{i,i+1} = 0$ for prismatic joints

↳ $J_{O_{i+1}} = 0$

Revolute joint

Link i: $^wv_{i+1} = {}^wv_i + {}^w\omega_{i,i+1}$

$^w v_e = {}^w v_n = \sum_{i=1}^{n} {}^w\omega_{i,i+1} = \sum_{i=1}^{n} J_{O_{i+1}} \dot{q}_{i+1}$

${}^w\omega_{i,i+1} = \dot{\theta}_{i+1} z_{i+1}$ for revolute joints

$\dot{q}_{i+1} J_{O_{i+1}} = \dot{\theta}_{i+1} z_{i+1}$

↳ $J_{O_{i+1}} = z_{i+1}$

Summary $J = \begin{bmatrix} J_{P_1} & \cdots & J_{P_i} \\ J_{O_1} & \cdots & J_{O_i} \end{bmatrix} \rightarrow \begin{bmatrix} J_{P_i} \\ J_{O_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_i \\ 0 \end{bmatrix} & i \text{ is prismatic} \\ \begin{bmatrix} z_i \times ({}^wP_e - {}^wP_i) \\ z_i \end{bmatrix} & i \text{ is revolute} \end{cases}$

If we consider the following coordinate change:

$$\begin{bmatrix} {}^v v_e \\ {}^v\omega_e \end{bmatrix} = \begin{bmatrix} {}^vR_w & 0 \\ 0 & {}^vR_w \end{bmatrix} \begin{bmatrix} {}^w v_e \\ {}^w\omega_e \end{bmatrix} = \begin{bmatrix} {}^vR_w & 0 \\ 0 & {}^vR_w \end{bmatrix} J(q)\dot{q}$$

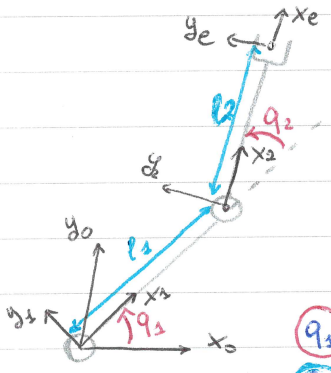
which leads to the following change of coordinates of the Geometric Jacobian ${}^vJ(q) = \begin{bmatrix} {}^vR_w & 0 \\ 0 & {}^vR_w \end{bmatrix} J(q)$

velocity in terms of linear velocity and end-effector orientation (v and Φ). $\dot{r}(p, \Phi) = \frac{\partial f(q)}{\partial q}\dot{q} = J_r(q)\dot{q}$

where $f(q)$ is a vector-valued function and its derivative is given by

$$\frac{\partial f}{\partial q}(q) = \begin{bmatrix} \frac{\partial f_1}{\partial q_1}(q) & \cdots & \frac{\partial f_1(q)}{\partial q_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial q_1}(q) & \cdots & \frac{\partial f_m}{\partial q_n}(q) \end{bmatrix}$$

Example: 2R PLANAR ARM



Frame 0 : base (fixed)
Frame 1: supports link 1
Frame 2: supports link 2
Frame e: End-effector

q_1 angle between F0 and F1 q_2: angle beteveen F1 and F2
l_1 Link 1 length l_2 Link 2 length

Direct Kinematics

From 0 to 1 (Pure rotation) $\quad {}^0T_1 = \begin{bmatrix} c_{q_1} & -s_{q_1} & 0 & 0 \\ s_{q_1} & c_{q_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

From 1 to 2 (roto-traslation) $\quad {}^1T_2 = \begin{bmatrix} c_{q_2} & -s_{q_2} & 0 & l_1 \\ s_{q_2} & c_{q_2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

From 2 to e (Rigid-Transform) $\quad {}^2T_e = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ (Identity matrix (I))

$${}^0T_2 = {}^0T_1\,{}^1T_2 = \begin{bmatrix} c(q_1+q_2) & -s(q_1+q_2) & 0 & l_1 c(q_1) \\ s(q_1+q_2) & c(q_1+q_2) & 0 & l_1 s(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow {}^0P_2$$

$${}^0T_e = {}^0T_2\,{}^2T_e = \begin{bmatrix} c(q_1+q_2) & -s(q_1+q_2) & 0 & l_2 c(q_1+q_2) + l_1 c(q_1) \\ s(q_1+q_2) & c(q_1+q_2) & 0 & l_2 s(q_1+q_2) + l_1 s(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow {}^0P_e$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J(q)\dot{q} \qquad \dim(J) = 6 \times 2 \qquad \begin{bmatrix} J_1 & J_2 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \qquad J_1 \in \mathbb{R}^6$$

For a revolute joint $J_i = \begin{bmatrix} z_i \times (w P_e - w P_i) \\ z_i \end{bmatrix}$ $\qquad J = \begin{bmatrix} z_1 \times ({}^0P_e - {}^0P_1) & z_2 \times ({}^0P_e - {}^0P_2) \\ z_1 & z_2 \end{bmatrix}$ ($ {}^0P_e - {}^0P_2 = {}^0P_{2e}$)

$${}^0P_e = \begin{bmatrix} l_1 C_1 + l_2 C_{12} \\ l_1 S_1 + l_2 S_{12} \\ 0 \end{bmatrix} \qquad {}^0P_{2e} = \begin{bmatrix} l_2 C_{12} \\ l_2 S_{12} \\ 0 \end{bmatrix} \rightarrow J(q) = \begin{bmatrix} -l_1 S_1 - l_2 S_{12} & -l_2 S_{12} \\ l_1 C_1 + l_2 C_{12} & l_2 C_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \in \mathbb{R}^{6\times 2}$$

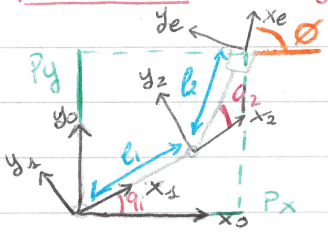
Why not just differentiating DK?

DK can be expressed in the following way: $r = \begin{bmatrix} wPe(q) \\ \Phi(q) \end{bmatrix} = f(q)$

Recall that Φ does not belong to a vector space and is a set of Euler angles. We can obtain the derivative of r with respect to time using chain rule: $\dot{r} = \frac{\partial f(q)}{\partial q} \frac{d}{dt} q = \underbrace{\frac{\partial f(q)}{\partial q}} \dot{q}$

↓ Analytical Jacobian $\dot{r} = J_r(q)\dot{q}$

Example Analytical Jacobian of 2R planar robot



DK: $P_x = \ell_1 C(q_1) + \ell_2 C(q_1+q_2)$

$P_y = \ell_1 S(q_1) + \ell_2 S(q_1+q_2)$

$\phi = q_1 + q_2$

Differentiating with respect to time:

$\dot{P}_x = -\ell_1 S(q_1)\dot{q}_1 - \ell_2 S(q_1+q_2)(\dot{q}_1+\dot{q}_2)$

$\dot{P}_y = \ell_1 C(q_1)\dot{q}_1 + \ell_2 C(q_1+q_2)(\dot{q}_1+\dot{q}_2)$

$\dot{\phi} = \dot{q}_1 + \dot{q}_2$

Rearranging: $\begin{bmatrix} \dot{P}_x \\ \dot{P}_y \\ \dot{\phi} \end{bmatrix} = \underbrace{\begin{bmatrix} -\ell_1 S(q_1) - \ell_2 S(q_1+q_2) & -\ell_2 S(q_1+q_2) \\ \ell_1 C(q_1) + \ell_2 C(q_1+q_2) & \ell_2 C(q_1+q_2) \\ 1 & 1 \end{bmatrix}}_{J_r(q)} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$

In general, the dimension of $J_r(q)$ is $J_r(q) \in \mathbb{R}^{m \times n}$

m: number of chosen task space coordinates to describe position and orientation of the end-effector

n: number of joint space coordinates. $J_r(q)$ performs the mapping from joint space to task space.

LEZ.7 NUMERICAL INVERSE KINEMATICS

IK: compute q such that $r = f(q)$ $\quad r \Rightarrow \boxed{IK} \Rightarrow q$

- At velocity level is easy: $\dot{q} = J^{-1} r$
- At position level numerical approaches are mandatory:
  (1) case of redundant robots
  (2) when a closed form solution does not exists or it is too hard to found

Numerical IK steps (input P_e^d, output q)

(1) sequence of q^i si parte da una configurazione casuale di joints q^i

(2) compute error $e_i = P_e^d - P_e(q^i)$ → uso DK per capire dove si trova EE con la configurazione di joint attuale

(3) Make $\|e_i\| \to 0$ as $i \to \infty$ Non c'è una soluzione "al primo colpo" ma è un processo iterativo

↳ It consists in the solution of an **OPTIMIZATION PROBLEM**

The problem is unconstrained

$$q^* = \arg\min_q \frac{1}{2} \| P(q) - p^d \|^2 = \frac{1}{2} \|e(q)\|^2 = C(q) \to \text{cost}$$

(sono dati dell'errore) (COST FUNCTION)

The norm is always differentiable

$\Delta P_e = J(q) \Delta q$

input: p^d = desired end-effector

output: q^* = joint positions

PROBLEM: $C(q)$ is non-linear and non-convex so it's hard to solve.

↳ We can find a local optimum

$\frac{\partial C(q)}{\partial q} = 0 \Rightarrow$ non-linear system of equations

Per trovare il minimo pongo la derivata uguale a zero

$\frac{\partial C}{\partial q}$ scalar / vector

↳ essendo un vettore, $\frac{\partial C}{\partial q}(q)$ esprime il gradiente

$$C(q) = \frac{1}{2} \|e(q)\|^2 = \frac{1}{2} e(q)^T e(q)$$

$e(q) = P(q) - p^d$

$$\frac{\partial C}{\partial q} = \underbrace{e(q)^T}_{\text{row vector}} \underbrace{\frac{\partial e(q)}{\partial q}}_{\text{Matrix}} = e(q)^T \left(\frac{\partial P(q)}{\partial q} - \frac{\partial p^d}{\partial q} \right) = e(q)^T J(q)$$

$\frac{\partial P(q)}{\partial q} = J(q)$ $\quad \frac{\partial p^d}{\partial q} = 0$ ↳ costante

se consideriamo di dover raggiungere sempre la stessa posizione, p^d è cost.

↳ $e(q)^T J(q) = 0 \iff J^T(q) e(q) = 0$

To find q^* such that $r(q) = 0$ I can use Gauss-Newton method.

q^* will be a local min/max for $C(q)$.

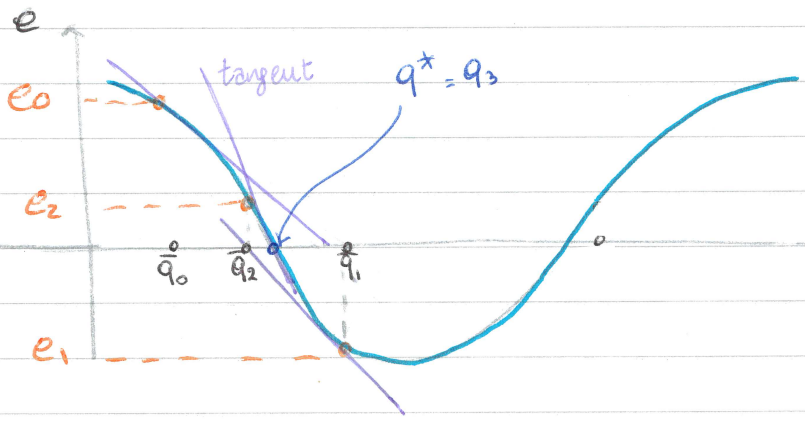
GAUSS-NEWTON APPROACH

Problema non lineare originale approssimato con una serie di problemi lineari

1. start with a guess $\bar{q}$
2. Compute a linear approximation of the problem
3. Solve it to find a new guess
4. Iterate till the solution of the linear problem will converge to the solution of the original (non linear) problem.

↳ cioè finché l'errore non è $e \to 0$

Example with 3 iterations



If we start from another guess we could converge to another solution

Linear Approximation of r(q)

$r(q) \approx r(\bar{q}) + \nabla_q r(\bar{q})^T \Delta q = \phi$ 1° order Taylor expansion $\bar{q}$ = stimated value

$\nabla_q r = \nabla_q (J^T e) = (\nabla_q J^T) e + J^T (\nabla_q e) \approx J^T J$

≈ 0 by Gauss-Newton approximation

$r(q) = r(\bar{q}) + J^T J \Delta q = 0$ must be equal to zero

↳ $J^T e + J^T J \Delta q = 0$

$J^T J \Delta q = -J^T e \Rightarrow \boxed{\Delta q = -(J^T J)^{-1} J^T e(\bar{q})}$ update Rule called Newton-STEP

Invertibility of $J^T J$

$J \in \mathbb{R}^{3 \times n}$ $J^T J \in \mathbb{R}^{n \times m}$ $\boxed{J^T}\boxed{J} = \boxed{J^T J}$

Tipically $n \geq 3$ $\text{rank}(J) \leq 3$ $J^T J$ is invertible if $\text{rank}(J^T J) = n$

PROPERTY: $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$

$\text{rank}(J^T J) = \text{rank}(J) \leq 3$

case 1: $n = 3 \Rightarrow J^T J$ is invertible

case 2: $n > 3 \Rightarrow J^T J$ is not invertible

* λI serve come termine di regolarizzazione rendendo J^TJ sempre invertibile e in più rendendo $(J^TJ)^{-1}$ sempre pos. def. si assicura che Δq sia un passo dell'algoritmo in una direzione che minimizza l'errore

REGULARIZATION

Nel caso in cui J^TJ non sia invertibile (singolarità)

$$\Delta q^* = -(J^TJ + \lambda I)^{-1} J^T e \qquad \lambda > 0 \text{ and small} \qquad \lambda I : \text{regularization term}$$

Always invertible and positive definite

$$= -J_\lambda^{\#} \lambda e \rightarrow \text{DAMPED PSEUDO-INVERSE}$$

Pseudo-Inversa smorzata

Δq is a descent direction for $C(q) \iff \nabla C^T \Delta q < 0 \rightarrow$ cost decreases if we move along Δq

$\nabla C = J^T e \qquad \Delta q = -(J^TJ)^{-1} J^T e$

$H = (J^TJ + \lambda I)$

$\nabla C^T \Delta q = -e^T J H^{-1} J^T e \rightarrow$ quadratic form $x^T A x$

$-e^T J H^{-1} J^T e < 0$

if $JH^{-1}J^T$ is positive definite $\Rightarrow \Delta q$ is a descent direction for $C(q)\ \forall q$

if H is positive definite → lo è sempre

Usiamo che è definita come $J^TJ + \lambda I$ e il termine λI le rende sempre positive definite *

If the cost is not decreasing but e is big

- zero gradient (vanishing gradient)
- Get stuck in a loop (Jumping around a local minimum)

↳ SOLUTION : Decrease learning rate

$\alpha \in [0,1]$ STEP-SIZE (learning rate)

$$q^{i+1} = q^i + \alpha \underbrace{\left[-(J^TJ + \lambda I)^{-1} J^T \underbrace{[p(q^i) - p^d]}_{e_i} \right]}_{\text{NEWTON STEP}} \qquad q^{i+1} = q^i + \alpha \Delta q$$

If $\alpha = 0$ we don't move

If $\alpha = 1$ we have original version of Gauss-Newton algorithm

HOW TO CHOOSE α? LINE SEARCH

- $\bar{q}_{TEST} = \bar{q}_i + \alpha \Delta q_i$
- reduction $= \|e(\bar{q}_i)\| - \|e(\bar{q}_{test})\|$ check if error is reduced
- if $![(\text{reduction} > 0) \text{ or } > \gamma \alpha \|e\|]$ (L'errore è diminuito?) (L'errore deve diminuire almeno di una certa frazione dell'errore di partenza.)

  $\alpha = \beta \alpha$ (riduzione learning rate) $\qquad \beta \in [0,1] \Longrightarrow$ reduction of α

  $\bar{q}_{TEST} = \bar{q}_i + \alpha \Delta q_i$ Si ricalcola con il nuovo learning rate ricalcolato

  reduction $= \|e(\bar{q}_i)\| - \|e(\bar{q}_{TEST})\|$ Ricalcolo riduzione
- RETURN $\bar{q}_{i+1} = \bar{q}_{TEST}$ Una volta che trovo un valore di α accettabile, l'algoritmo accetta il passo

Implementation issues and comments

1. Initial guess $\bar{q}_0$
   - only 1 solution is generated for each guess
   - multiple initializations $\Rightarrow$ obtain different solutions
2. Joint range limits are considered only at the end
3. STOPPING CRITERIA: $\| p^d - f(\bar{q}_i) \| \leq \varepsilon$ or $\| J^T e_i \| < \varepsilon$
   (deve essere sufficientemente piccolo)
   - separate criteria for position and orientation variables.

LEZ. 8 21/03/2025 **REDUNDANCY AND SINGULARITY**

REDUNDANCY ↙ ↘ **SINGULARITY**

You have more mobility (DoF) than you need

You have less mobility than you need

(1) dimension of task < DoFs

(2) Infinite solutions exists ∞^{n-m}

(3) Jacobian matrix rectangular (fat)

REDUNDANT MANIPULATORS

Redundant DoFs can be used to increase dexterity (e.g. for obstacle avoidance)

Among infinite solutions, we can find one that allow robot to not collide with obsacle

null space N(J): subspace of all vectors that are zeroed by matrix J.

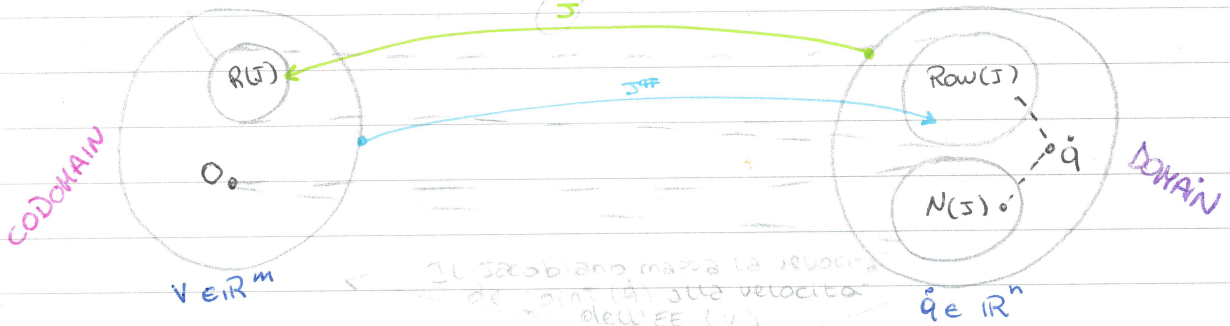
$$N(J) = \{ \dot{q}_0 \in \mathbb{R}^n : J\dot{q}_0 = 0 \in \mathbb{R}^m \} \quad \dim(N(J)) = n - \rho(J)$$

(rank)

Exists joint space velocity $\neq 0$ that doesn't effect e-e position

row space: subspace spanned by the raws of J OR equivalently by the columns of J^T

$$Row(J) = R(J^T) \quad \dim(Row(J)) = \rho(J)$$ same as R(J)



- $J^{\#}$ maps always onto the row space*
- J filters out whatever in the domain is in the null space and maps what is in the raw space into the range space
- The null space component is nullified by the action of J
  - * So using $\dot{q} = J^{\#}V_e$ to reach a certain velocity of end-effector, solution $\dot{q}$ is always the one with minimum norm (less energy) because, finding $\dot{q}$ only in Row(J), pseudo-inverse "filters" joint movement that doesn't affect end-effector.

RANK NULLITY THEOREM

(A) $\underbrace{\dim(\text{Row}(J))}_{\rho(J)} + \dim(N(J)) = n$ → Dimension of domain $\mathbb{R}^n = N(J) + R(J^\#)$

$\dot{q} = \dot{q}_{row} + \dot{q}_{null}$ — number of joints

(B) $\dim(R(J)) + \dim(N(J^\#)) = m$ → dimension of codomain $\mathbb{R}^m = N(J^\#) + R(J)$

from (A): any element $\dot{q} \in \mathbb{R}^n$ can be written as: $q = q_{ROW} + q_{NULL}$, $q_{ROW} \in R(J^T)$, $q_{NULL} \in N(J)$

from (B): any element $v \in \mathbb{R}^m$ can be written as $v = v_{RANGE} + v_{NULL}$, $v_{RANGE} \in R(J)$ $v_{NULL} \in N(J^\#)$

postural task

$m = 3$, $n > 3$ => ∞ solutions to the IK that depends on $\bar{q}_0$

GOAL: have solution independent of initial guess $\bar{q}_0$

IDEA: reformulate the cost to have the solution closer to a default configuration q^P (POSTURAL)

Redundancy and Null Space

- Null space in the domain can exists only when $n > \text{rank}(J)$.
- In case of redundant robot with $n > m$ and full rank ($\text{rank}(J) = m$) the dimension of the null-space is $n - m$
- To compute the Inv. Diff. Kin. we need to invert a rectangular matrix using a pseudo inverse:

$$\dot{q} = J^\# v + [I - J^\# J]\dot{q}_0$$ NULL-SPACE METHOD

$J^\# v \in \text{Row}(J)$ — pseudo-inverse helps avoid $N(J)$

{ Allow robot to do a secondary task without change EE velocity

Pseudo-Inverse Computation

- Necessary to invert a rectangular matrix
- Depending on the nature of the matrix A we can have right or left pseudo-inverses $A^\#$:

(A) ASSUME A is full rank:

Fat case ($m < n$) — codomain

$J^\# = J^T (J^T J)^{-1}$ — domain

Skinny case ($m > n$) — domain

$J^\# = (J^T J)^{-1} J^T$ — codomain

(B) Assume rank is not full:

- AA^T or A^TA are not invertible
- $A^\#$ still exists and can be computed numerically with SINGULAR VALUE DECOMPOSITION (SVD) of A.

Notes on Pseudoinverse:

- **FAT** ▭
  - more unknown than equations (ridondanza)
  - ∞ solutions → Trovare la velocità dei joint che producono la velocità desiderata dell'EE ma con il MINIMO SFORZO
  - Return the minimum norm solution to $Jq = V \rightarrow \dot{q} = J^{\#} V$ that min. $\|\dot{q}\|^2$
- **SKINNY** ▯
  - more equations than unknows
  - no exact solution exists → Sia che non esiste una soluzione esatta, restituisce la migliore approssimazione
  - returns the "best fit" (Least square) that minimizes $\|J\dot{q} - V\|^2$.

Least Square Program - SKINNY

$\dot{q} = J^{\#} V$ can be casted as an unconstraint optimization problem

↳ $Jq - V = 0$ → Trovare l'approssimazione migliore

$$\dot{q}^* = \arg\min_{\dot{q}} \frac{1}{2} \|J\dot{q} - V\| = \underbrace{\frac{1}{2} \dot{q}^T J^T J \dot{q} - V^T J + \frac{1}{2} V^T V}_{F(\dot{q})}$$

$$\nabla_{\dot{q}} f = J^T J \dot{q} - J^T V = 0 \Rightarrow J^T J \dot{q} = J^T V$$

$$\dot{q}^* = \underbrace{(J^T J)^{-1} J^T}_{J^{\#}_{SKINNY}} V$$

Norm Minimization - FAT

- If A is full-row rank ▭ } e

$\dot{q} = \arg\min_{\dot{q}} \|\dot{q}\|^2$ s.t. $J\dot{q} = V$ $\Rightarrow$ $\dot{q}^*$ minimizes $\|\dot{q}\|^2$ among the ∞ $\dot{q}$ that satisfy $J\dot{q} = V$

↳ Trovare la soluzione con norma minima

- If A is NOT full-row rank ▭ } e

$\dot{q} = \arg\min_{\dot{q} \in S} \|\dot{q}\|^2$ $\quad S = \{\dot{q} \in \mathbb{R}^n : \|J\dot{q} - v\| \text{ is minimum}\}$

↳ $\dot{q}^*$ minimizes $\|\dot{q}\|$ among the ∞ $\dot{q}$ that minimizes $\|J\dot{q} - v\|^2$

Durante l'algoritmo Gauss-Newton, nel calcolo dello step Δq:

- Se J_{SKINNY}: si calcola Δq come approssimazione migliore
- Se J_{FAT}: si calcola Δq cercando la migliore tra le ∞ soluzioni

Generalized Pseudo-Inverse

What we defined so far was a specific type of pseudo-inverse called MOORE-PENROSE pseudo-inverse, we indicate with the symbol A^+.

GENERALIZED PSEUDO-INVERSE (fat matrix) $A^{\#}_W = WA^T(AWA^T)^{-1}$

↳ ∞ pseudo-inverse exist setting a weight W

CHECK: $AA^{\#}_W = AWA^T(AWA^T)^{-1} = I_{m\times m} \Rightarrow$ OK

The solution of $\dot{q} = J^{\#}_W V$ in this case minimizes the weighted norm $\|\dot{q}\|^2 = \dot{q}^T W \dot{q}$ W is pos. definite

Singular Values

- Extension of eigenvalues to rectangular matrix
- let $A \in \mathbb{R}^{m\times n}$
- The matrix $A^TA \in \mathbb{R}^{n\times n}$:
  - (1) is symmetric → n real eigenvalues λ_i
  - (2) $\lambda_i \geq 0 \quad i \in [1, \dots, n]$
  - (3) the singular values of A are linked to the eigenvalues of A^TA

$$\sigma_i = \sqrt{\lambda_i} \qquad \sigma_1 > \sigma_2 > \dots > \sigma_n$$

- If $m < n$ then A has at most rank m and therefore at most m non-zero σ_i
- The number of non-zero singular values of A equals the rank $\rho(A)$ and tell us the dimension of the range space $R(A)$ $\sigma_1 > \sigma_2 > \dots > 0$
- The number of zero singular values tell us the dimension of the null-space $N(A)$ $\sigma_{\rho+1} = \dots = \sigma_n = 0$

KINEMATIC SINGULARITIES AT VELOCITY LEVEL

In a singular configuration $\bar{q}_s$:

(1) End-effector mobility loss (cannot move in some cartesian directions of the task space)

(2) Possibility of ∞ solutions to the IK problems

(3) The Jacobian loses rank $\rho(J) < m$

$$J = \begin{matrix} \delta x \\ \delta y \\ \delta z \end{matrix} \begin{bmatrix} \delta q_1 & \delta q_2 & \delta q_3 \\ & & \\ 0 & 0 & 0 \end{bmatrix}$$

It becomes a redundant manipulator in sub space orthogonal to singular direction

No joint motion can create movement along z.

↳ Only if we have a robot aligned with z axis, we can have a row of zeros. Otherwise we get a row DEPENDANT from others.

If the Jacobian degenerates at singularity the dimension of range space decreases while the dimension of null-space increases

$$\rho(J) + \dim(N(J)) = n$$

(4) Close to singularity you can have big $\dot{q}$ to achieve small end-effector velocities

Compute Singular Configurations

(A) NON-REDUNDANT ROBOT ($m = n$): solve the non-linear scalar equation for $\bar{q}_s$: $\det(J(\bar{q}_s)) = 0 \Rightarrow \bar{q}_s$

(B) REDUNDANT ROBOT ($m < n$):

(1) Check symbolically where $\det(J(\bar{q}_s))\, J(\bar{q}_s)^T = 0$

(2) Find $\bar{q}_s$ such that all $m \times m$ minors (square matrix) that you can extract from J have $\det = 0$

INVERSION OF DIFFERENTIAL KINEMATICS

Task Velocity v ⟶ Joint Velocity $\dot{q} = J^{-1}(q)\, v$

REQUIRES: J squared non-singular

PROBLEM:
- Singularity → J is not invertible
- Redundant Robots → J is rectangular (FAT) so inverse is not defined.

Use $J^{\#}$. If there's a loss of rank in J then $J^{\#} v$ finds the solution $\dot{q}$ closest to v.

DAMPED LEAST SQUARE METHOD

Try to find a trade-off between realizying the velocity task at the cartesian level and have small joint velocity. => we can cast this as an optimization problem with 2 terms in the cost:

$$\min_{\dot{q}} \underbrace{\frac{1}{2}\|J\dot{q} - v\|}_{A} + \underbrace{\frac{\lambda}{2}\|\dot{q}\|^2}_{B} \qquad \lambda > 0 \text{ Damping Parameter}$$

A: Least Square B: Damped

The solution can be obtained in closed form (put gradient of cost = 0)

$$\dot{q}^* = (J^T J + \lambda I_m)^{-1} J^T v$$

$$\dot{q}^* = J^T (J J^T + \lambda I_m)^{-1} v \quad \rightarrow \text{Equivalent for redundant robot (with } J^T A^T)$$

Pros can be used both for recovering from singularity of a square Jacobian or if you have a rectangular Jacoban to invert

cons with big λ you penalize velocity but you get bigger tracking error

HIGHER-ORDER DIFFERENTIAL INVERSION

First order differential kinematics $\dot{r} = J(q)\dot{q}$

$$\ddot{r} = \begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = J(q)\ddot{q} + \dot{J}(\dot{q})\dot{q} \quad \text{second order diff. kinematics}$$

$$\Downarrow \text{INVERTE}$$

$$\ddot{q} = J(q)^{-1}(\ddot{r} - \dot{J}(q)\dot{q})$$

IK SUMMARY

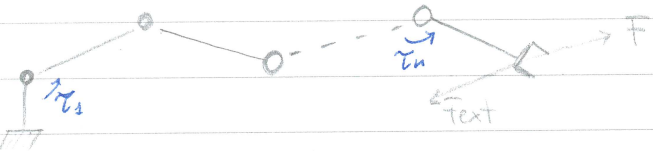
$$q = IK(r)$$
$$\dot{q} = J^{-1}\dot{r}$$
$$\ddot{q} = J^{-1}(\ddot{r} - \dot{J}\dot{q})$$

r: generic task function

LEZ. 9 04/04/2025 DYNAMICS OF A RIGID BODY - Statics

STATICS VS **DYNAMICS**

studied the cause of the motion (forces are induced) — contrario — Case where we have forces but no motion (equilibrium)



$\tau \in \mathbb{R}^n$ torques applied by motors to a joint

F: equivalent force exerted by ~~motors at joints~~ the robot at the end-effector

Fext: forces by environment at end-effector $F_{ext} = -F$

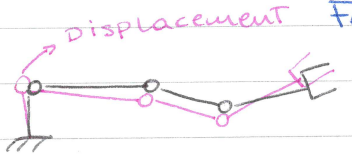
↳ environment reaction is equal and opposite to robot action

WORK

motors τ ⇝ Robot system ⇜ environment forces

CONVENTION: POSITIVE WORK ~~FORCES~~ is done when forces are applied on the robot

QUESTION OF STATIC: what are the joint torques that will balance an Fext exerted by the environment? $\left(\frac{dP}{d\Phi}\right) = J dq$



At equilibrium we can define:

- dq_i: infinitesimal displacement
- δq_i: virtual displacement (sat. constraints)

↳ no kin. energy variation and zero velocity

PRINCIPLE OF VIRTUAL WORK

Sum of virtual works done by all torques/forces acting on the system is zero at equilibrium.

$$\tau^T dq - F^T \left(\frac{dP}{d\Phi}\right) = \tau^T \delta q - F^T J dq = 0$$

$v = J\dot{q}$, $\dot{q}$

work of Joints — work of end effector

$$(\tau^T - F^T J)\, dq = 0 \quad \forall dq \Rightarrow \tau^T - F^T J = 0 = \boxed{\tau = J^T F}$$

J^T agisce come moltiplicatore tra le forze che agiscono sull'EE (F) e le forze risultanti sui joint (τ)

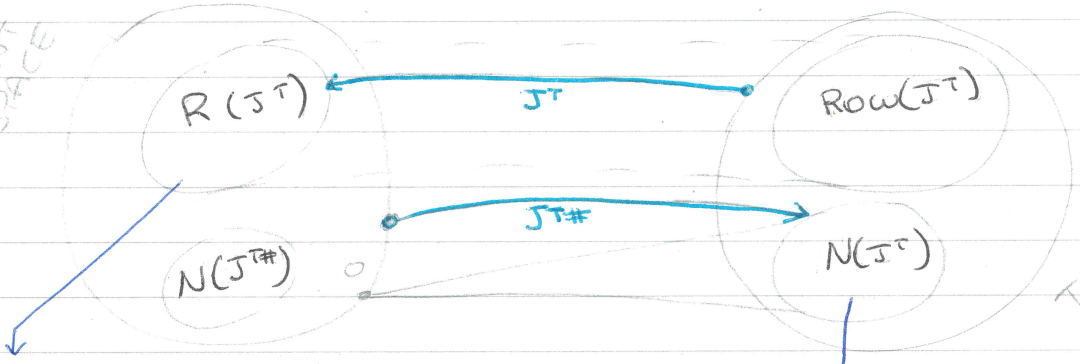
KINETO-STATIC DUALITY

JOINT SPACE — TASK SPACE

Velocity $\dot{q}$ —$J(q)$→ Velocity in cartesian space $\begin{bmatrix} v \\ w \end{bmatrix}$

Forces at joint τ ←$J^T(q)$— Forces at end-effector F

$\text{rank}(J) = \text{rank}(J^T)$ ⇒ singular conf. are the same, same analysis holds



Joint torques that can balance end-effector forces

$R(J^T) = \{\tau \in \mathbb{R}^n : \exists F \in \mathbb{R}^m, J^T F = \tau\}$

Forces that don't need to be balanced (they're balanced by reaction forces of the structure)

$N(J^T) = \{F \in \mathbb{R}^m : J^T F = 0\}$

MAPPING END-EFFECTOR FORCES

① NON-REDUNDANT MANIPULATOR $m = p$

(1.1) $\text{rank}(J/J^T) = m$ (Full Rank) $\det(J) \neq 0$ No singularity

$\nexists N(J^T)$, $\nexists N(J)$

(1.2) $\text{rank}(J/J^T) < m$ Robot is at singularity

$\exists N(J): \exists \dot{q} \neq 0 : J\dot{q} = 0$

$\exists N(J^T): \exists F \neq 0 : J^T F = 0$

② REDUNDANT MANIPULATOR $(m < n)$ J^T rectangular and SKINNY (J FAT)

(2.1) $\text{rank}(J/J^T) = m$ (Full)

$\exists N(J) \Rightarrow \exists \dot{q} \neq 0 : J\dot{q} = 0$ $\quad \nexists N(J^T)$

(2.2) $\text{rank}(J/J^T) < m$

$\exists N(J) \Rightarrow \exists \dot{q} \neq 0 : J\dot{q} = 0$ $\quad \exists N(J^T) \Rightarrow \exists F \neq 0 : J^T F = 0$

EXAMPLE CASE (1.2) $m=n=2$ $\rho(J)<m=n$

SINGULARITY $\Rightarrow q_2 = 0 \Rightarrow \rho(J) = 1$ (braccio completamente esteso)

$$J|_{q_2=0} = \begin{bmatrix} (l_1+l_2)s_1 & -l_2 s_1 \\ (l_1+l_2)c_1 & l_2 c_1 \end{bmatrix}$$

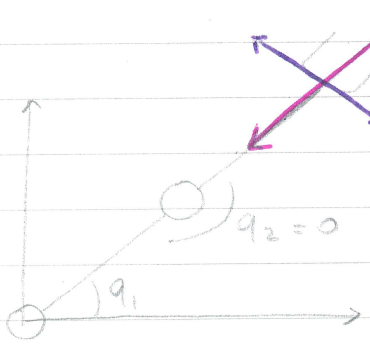
$\{ R(J) = (1) \quad R(J^T) = \}$

↳ le colonne sono linearmente dipendenti (la seconda è proporzionale alla prima)

$R(J)$: rappresenta le velocità dell'EE che può ottenere in questa conf. $= \alpha \begin{bmatrix} -s_1 \\ c_1 \end{bmatrix}$ set of points

$N(J^T)$: find a vector v such that $J^T v = 0 \Rightarrow J^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

↳ $N(J^T) = \alpha \begin{bmatrix} c_1 \\ s_1 \end{bmatrix}$ direction



$$N(J) = \bar{\alpha} \begin{bmatrix} l_2 \\ -(l_1+l_2) \end{bmatrix}$$

Recap on Spaces

$R(J^{T\#}) \cong R(J)$ → feasible motions

forces from torques

$N(J^{T\#}) \cong N(J^T)$

forbidden motion — force from structure

$N(J)$: self motion

VELOCITY AND FORCE TRANSFORMATION

- We can apply virtual work principle to map forces / moments at different places of a rigid body.

velocity

$$ {}^{A}V_A = {}^{A}V_B + {}^{A}W_B \times {}^{A}P_{AB} $$

$$ \hookrightarrow \begin{bmatrix} {}^{A}R_B & -{}^{A}R_B [{}^{B}P_{BA}]_\times \\ 0 & {}^{A}R_B \end{bmatrix} \begin{bmatrix} {}^{B}V_B \\ {}^{B}W_B \end{bmatrix} $$

$$ w = w_A = w_B \Rightarrow {}^{A}W = {}^{A}R_B \, {}^{B}W $$

$$ \begin{bmatrix} {}^{A}V_A \\ {}^{A}W \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{A}R_B & -{}^{A}R_B [{}^{B}P_{BA}]_\times \\ 0 & {}^{A}R_B \end{bmatrix}}_{{}^{A}A_B} \begin{bmatrix} {}^{B}V_B \\ {}^{B}W \end{bmatrix} $$

force

$$ \begin{bmatrix} {}^{B}f \\ {}^{B}m \end{bmatrix} = {}^{A}A_B^T \begin{bmatrix} {}^{A}f \\ {}^{A}m \end{bmatrix} $$

trasformazione di forza da un frame A ad un frame B

$$ \hookrightarrow \begin{bmatrix} {}^{B}R_A & 0 \\ [{}^{B}P_{BA}]_\times \, {}^{B}R_A & {}^{B}R_A \end{bmatrix} $$

DIRECT: Date le forze calcola il movimento risultante
INVERSE: Dato un movimento (trajectory) calcola le forze necessarie per eseguire quel movimento

LEZ. 10 11/04/2025 ROBOT DYNAMICS Newton-Euler Approach

DYNAMIC MODEL of ROBOTS CALCOLANDO COME IL ROBOT SI MUOVERÀ NEL TEMPO

Dynamic model: provides a description of the relationship between the sources of motion and the resulting motion.

PROBLEM: forces applied in different point of the robot need to be transformed into equivalent ones that will make work on the generalized coord. q

generalized forces associated with generalized coord

$$u(t) \longrightarrow \boxed{\Phi} \longrightarrow q, \dot{q}, \ddot{q}$$

DIRECT DYNAMICS $u(t) = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_N \end{bmatrix}$ coppie di giunti dei motori applicate

input for $t \in [0, \tau]$ $\Rightarrow$ Robot $\Rightarrow$ $q(t) = \begin{bmatrix} q_1 \\ \vdots \\ q_N \end{bmatrix}$ $\forall t$ — Resulting Motion

NUMERICAL SOLUTION: integrate numerically the diff. equations $\Phi = u$ of the model → By simulation, NO closed form

PURPOSE: predict robot behaviour for controller design and mechanical design.

INVERSE DYNAMICS: $q^d(t)$, $\dot{q}^d(t)$, $\ddot{q}^d(t)$

input: desired motion for $t \in [0, \tau]$ $\Rightarrow$ Robot $\Rightarrow$ $u^d(t) = \begin{bmatrix} \tau_1^d \\ \vdots \\ \tau_N^d \end{bmatrix}$ for $t \in [0, \tau]$

↓

Torques needed to realize the motion.

- EXPERIMENTAL SOLUTION: repeated trials with iterative learning
- ANALYTIC SOLUTION: use dynamic model to compute algebraically $u^d(t)$ at every time instant t.

PURPOSE: model-based control and actuator sizing.

APPROACHES TO DYNAMIC MODELING

EULER-LAGRANGE	NEWTON-EULER
• energy-based	• Balance of forces/moments extended to multi bodies
• symbolic closed form	• dynamic equations in numeric form
⊕ analysis of dynamic properties	⊕ efficient recursive implementation
⊕ analysis of control scheme	⊕ best for implementing ID in real time ($O(N)$, N = #DoFs)
⊖ explodes for many DoFs	

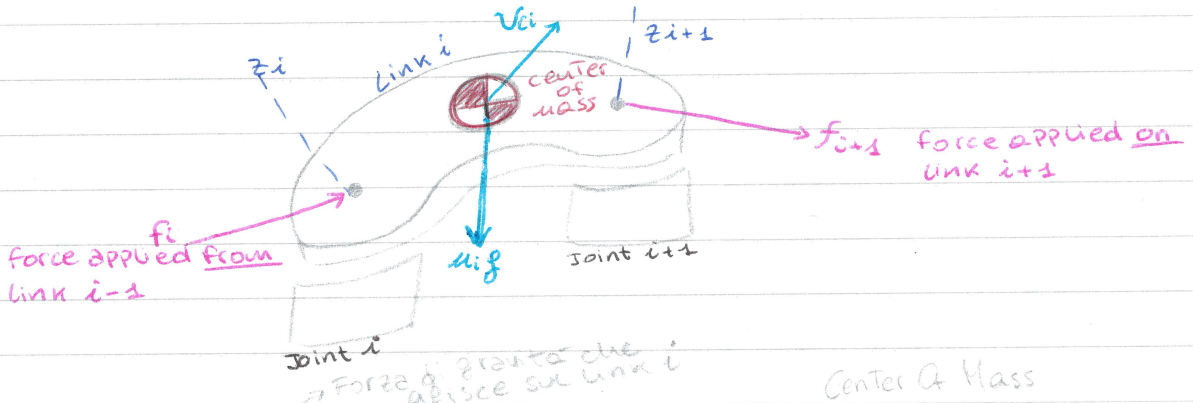
NEWTON-EULER METHOD

Is based on classical mechanics equations. Based on balance of forces / moments on a single link, considerating interactions from nearby links in the kinematic chain. Same model output as Lagrangian.

NEWTON EQUATION: $\Sigma f_i = \frac{d}{dt}(m v_{ci}) = m \dot{v}_{ci} = m a_{ci}$ vectors expr. in world frame

la seconda legge di Newton

↳ variation of linear momentum



Forza che agisce sul link i

Center of Mass

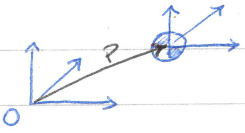
③ $f_i - f_{i+1} + m_i g = m_i a_{ci}$ Linear Acceleration of COM of link i

SLIDE 342-343 Examples of moments of inertia

Huygens-Steiner's Theorem

Descrive la distribuzione della massa di un corpo rispetto a una rotazione

compute the tensor of inertia about another frame given I_{COM} (Inertia tensor of the Center Of Mass).



$$I_O = I_{COM} + m(P^T P I_{3x3} - P P^T) = I_{COM} - m[P]_x[P]_x$$

Identity

skew-symmetric

Oltre alla traslazione, è necessario considerare anche la rotazione del sist. rotazionale

Euler Equation

equation for rotational motion: (Bilancio dei momenti)

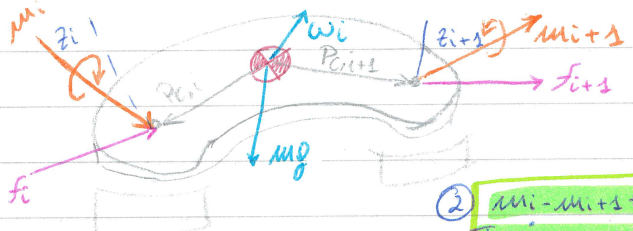
(1) we choose as pole the COM

(2) we express the vectors in world frame

$$\Sigma m_i = I_{COM}\dot{\omega} + \omega \times I_{COM}\omega \quad \frac{d}{dt}(I_{COM}\omega) \rightarrow \text{Momento Angolare}$$

somma dei momenti esterni che agiscono sul link

Inertia of link i wrt its COM expressed in frame w



② $m_i - m_{i+1} + P_{c,i} \times f_i + P_{c,i+1} \times (-f_{i+1}) = I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i)$

angular acceleration ↑ ↳ gyroscopic effect

Tensore di inerzia del link i rispetto al suo centro di massa

RECURSION IDEA

(A) Forward Pass: propagate velocities and accelerations from base to end-effector through FORWARD KINEMATICS (Per calcolare veloc. e accelerazioni)

(B) Backward Pass: compute forces/moments for each link going from end-effector to base using (1), (2) this will provide f_i, μ_i that should be projected along the axis of the supporting joint. (Calcolare le forze e i momenti di reazione dai joint)

ACCELERATION RECURSION (Revolute Joint)

- $\omega_i = \omega_{i-1} + \dot{q}_i z_i$ (vettore unitario dell'asse di rotazione)
- $\dot{\omega}_i = \dot{\omega}_{i-1} + \ddot{q} z_i + \dot{q}_i \omega_{i-1} \times z_i$ (Acc. ang.; Termine centripeto / Coriolis)
- $v_i = v_{i-1} + \omega_{i-1} \times P_{i-1,i}$
- $a_i = a_{i-1} + \dot{\omega}_{i-1} \times P_{i-1,i} + \omega_{i-1} \times (\omega_{i-1} \times P_{i-1,i})$ (Acc angolare; Acc centripeta)
- a_{ci} from a_i same way

BACKWARD RECURSION (F/TR)

$f_i = f_{i+1} + m_i (a_{ci} - g_i)$ (Accelerazione centro di massa del link i)

$\mu_i = \mu_{i+1} - P_{c,i} \times f_i + P_{c,i+1} \times f_{i+1} + I_i \dot{\omega}_i + \omega_i \times I_i \omega_i$ (Eq. Eulero)

$$\tau_i = \begin{cases} z_i^T f_i & \text{for prismatic joints} \Rightarrow \text{is a } \underline{\text{scalar}} \\ z_i^T \mu_i & \text{for revolute joints} \end{cases}$$

(Proiezione della forza nella sull'asse di traslazione; Proiezione del momento lungo l'asse di rotazione)

RECOURSIVE NEWTON-EULER ALGORITHM (efficient) FOR INVERSE DYNAMIC

INPUT: base_link $\omega_0, \ddot{P}_0 - g_0, \dot{\omega}_0$

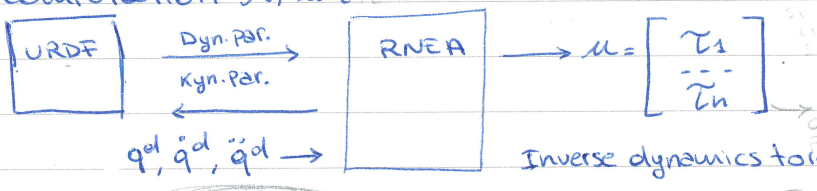
Joints $q, \dot{q}, \ddot{q}$

Last_link f_{n+1} μ_{n+1} (terminal condition) (Forze e momenti ESTERNI applicati all'END EFFECTOR)

FWD RECURSION: computation $\omega_i, \dot{\omega}_i, \ddot{P}_{ci}, \ddot{P}_{ci}$ (Partendo dalla base, si calcolano ricorsivamente per ogni link; Pos e acc centro di massa)

BWD RECURSION: computation f_i, μ_i (Partendo dall'EE (dove sono noti momenti e forze esterne) si calcolano ricors. momenti e forze sui link)

OUTPUT: τ_i



- strongly reccomended for real-time especial when n !!!
- works for any kind of robot.

Recursion makes Newton-Euler algorithm COMPUTATIONALLY EFFICIENT

Joint Space Dynamic Model: $H(q)\ddot{q} + c(q,\dot{q}) + g(q) = u$ (forze/coppie inerziali; forze Coriolis centrifughe; forze applicate ai giunti)

- the model we found are on the form $\Phi(q,\dot{q},\ddot{q}) = u$ and the dynamic equations are valid for any type of robot.
- it can be shown that each element of vector $c(q,\dot{q})$ can be computed with: $c_k(q,\dot{q}) = \dot{q}^T C_k(q)\dot{q}$

where $C_k(q) = \frac{1}{2}\left[\frac{\partial}{\partial q}H_k + \left(\frac{\partial}{\partial q}H_k\right)^T - \frac{\partial}{\partial q_k}H\right]$

↳ CHRISTOFFEL SYMBOL

- Alternatively is possible the derivation in the scalar version:

$$\underbrace{\sum_{j=1}^{n} m_{kj}(q)\ddot{q}_j}_{\text{Inertial Terms}} + \underbrace{\sum_{i,j}^{n} c_{kij}(q)\dot{q}_i\dot{q}_j}_{\text{Centrifugal } (i=j)} + \underbrace{\frac{\partial U}{\partial q_k}}_{\text{Gravity terms } g_k(q)} = u_k \qquad k \in [1,n]$$

(potential energy)

↳ m_{kj}: inertia seen at joint k when joint j accelerates

↳ c_{kij}: coridis force felt at joint k when $\dot{q}_i \neq 0$ $\dot{q}_j \neq 0$

Possible uses of RNEA (Recursive Newton Euler Algorithm)

- compute inverse dynamics $u = RNEA(g, q, \dot{q}, \ddot{q}^d) = H(q)\ddot{q}^d + c(q,\dot{q}) + g(q)$
- Gravity terms $u = RNEA(g, q, 0, 0) = g(q)$
- Centrifugal / Coridis $u = RNEA(0, q, \dot{q}, 0) = c(q,\dot{q})$ (inertial/centrifugal)
- i-th column of the inertia matrix $u = RNEA(0, q, 0, e_i) = H_i(q)$
- Generalized Momentum $u = RNEA(0, q, 0, \dot{q}) = H(q)\dot{q}$

USE of RNEA for DIRECT DYNAMICS (simulation)

(A) compute bias terms $h(q,\dot{q})$ $h = RNEA(0g, q, \dot{q}, 0) \rightarrow O(n)$ compl comput.
↳ Acc a zero x isolare i termini di bias

(B) compute inertia matrix $M_i = RNEA(0, q, 0, e_i) \rightarrow O(n^2)$
↳ solo l'iesimo elemento è 1

(C) compute accelerations $\ddot{q} = M(q)^{-1}(u-h) \rightarrow O(n^3)$

(D) Integrate $\ddot{q}_k \Rightarrow q_{k+1}, \dot{q}_{k+1}$

(E) Iterate $u = \begin{bmatrix} \tau_1 \\ \tau_n \end{bmatrix} \rightarrow \boxed{\begin{array}{l} SIM \\ M\ddot{q} + h = u \end{array}} \rightarrow q, \dot{q}$

FORWARD / EXPLICIT EULER INTEGRATION METHOD

GOAL : integrate accelerations into velocities / positions

TAYLOR APPROXIMATION $f(t+dt) = f(t) + dt\, f'(t) + \frac{1}{2!} dt^2 f''(t) + \dots$

↳ forward Euler is a 1° order method $f(t+dt) \approx f(t) + dt\, f'(t)$

ASSUMPTION : constant acceleration during time step

$$\left.\begin{aligned} \dot{q}(t+dt) &= \dot{q}(t) + dt\, \ddot{q}(t) \\ q(t+dt) &= q(t) + dt\, \dot{q}(t) \end{aligned}\right] \text{ FORWARD EULER}$$

We can directly integrate position from acceleration:

$$q = \iint \ddot{q}\, dt = \int \ddot{q} t + \dot{q}(0)\, dt = \frac{t^2}{2} \ddot{q} + \dot{q}(0)\, t + q(0)$$

↳
$$\left.\begin{aligned} \dot{q}(t+dt) &= \dot{q}(t) + dt\, \ddot{q}(t) \\ q(t+dt) &= q(t) + \dot{q}(t) dt + \frac{t^2}{2} \ddot{q}(t) \end{aligned}\right] \text{ IMPROVED FORWARD / EXPLICIT EULER}$$

LEZ. 12 09/05/2025 - ROBOT CONTROL, PID for manipulators

Control of Robots is the analysis and synthesis of the robot's **control laws**. → finds the time behaviour of the forces and torques to be delivered by the joint actuators to ensure the execution of a desired task.

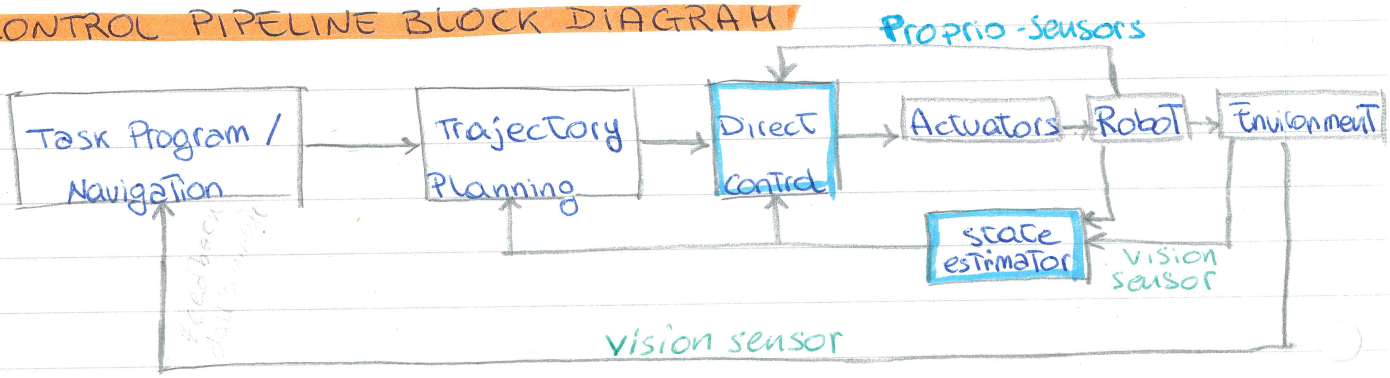
Robot Control Task: "high level" commands. Each high-level description of the control task can be expressed as lower-level sub-tasks.

Pure planning (open loop) is not enough to face of disturbances (from the environment). Another problem is MODEL UNCERTAINTIES, helped by feedback loop.

There are different types of control schemes, we'll see the

classic feedback control: tolerates mild disturbances and small parameter variation.

CONTROL PIPELINE BLOCK DIAGRAM



- real-time requirements

SLIDE 589 Panoramic View

EQUILIBRIUM STATE OF A ROBOT

$H(q)\ddot{q} + h(q,\dot{q}) = u$ (BIAS) u: control input (Joint Torque) n 2° order differential equ.

[An equilibrium point is a point where this equation is equal to zero.]

If we organize the dynamics equation in a state-space form:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Rightarrow \dot{x} = \begin{bmatrix} x_2 \\ -H^{-1}(x_1)\left[h(x_1,x_2)\right] \end{bmatrix} + \begin{bmatrix} 0 \\ H^{-1}(x_1) \end{bmatrix} u$$

$\dot{q} = x_2$

$\ddot{q} = H(q)^{-1}(u - h(q,\dot{q}))$

$\dot{x} = f_x(x_1,x_2) + f_u(x_1)u$ 2n non-linear 1° order differential equations (ODE)

Drift term — Forcing Term

(A) $\bar{x}$ unforced equilibrium $\rightarrow$ $f_u(\bar{x})u = 0$ we don't command robot

(no command $u = 0$)

$$\begin{cases} \bar{x}_2 = 0 \\ -H^{-1}(\bar{x}_1)^{-1} h(\bar{x}_1, 0) = 0 \rightarrow g(\bar{x}_1) = 0 \end{cases}$$

Il robot è in grado di sollevare il proprio peso senza bisogno di azioni esterne dai motori

Gravity is balanced by structure

(B) $\bar{x}$ forced equilibrium $\rightarrow$ $u = u(x)$ we say to robot to go in a point and stay there

$$\begin{cases} \bar{x}_2 = 0 \text{ la velocità dei joint è zero} \\ -H^{-1}h(\bar{x}_1,0) + H^{-1}u(\bar{x}) = 0 \rightarrow u(\bar{x}) = g(\bar{x}_1) \end{cases}$$

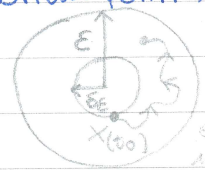
le coppie applicate ai joint devono bilanciare la gravità

Joint torques balance gravity

Stability

An equilibrium point $\bar{x}$ is stable if $\forall \varepsilon > 0 \; \exists \delta_\varepsilon > 0 : \|x(t_0) - \bar{x}\| < \delta_\varepsilon \Rightarrow$

$$\Rightarrow \|x(t) - \bar{x}\| < \varepsilon \quad \forall t > t_0$$



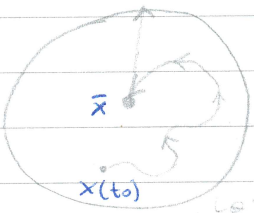
Se il sistema viene perturbato leggermente ($\varepsilon > 0$) dal suo punto di equilibrio, la sua traiettoria non si "allontanerà" definitamente, ma rimarrà confinata in un intorno del punto di equilibrio

Asymptotic Stability

$\bar{x}$ is asimptotically stable if:

$\exists \delta > 0 : \|x(t_0) - \bar{x}\| < \delta \Rightarrow \|x(t) - \bar{x}\| \rightarrow 0 \quad t \rightarrow \infty$

$\forall \delta > 0$ GLOBAL asymptotic stability



GoesTo equilibrium point

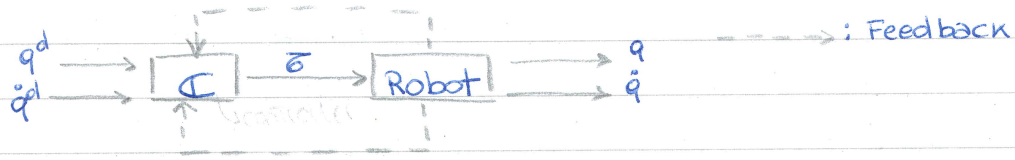
oltre ad essere stabile, la traiettoria converge al punto di equilibrio per $t \rightarrow \infty$

Stability of linear systems:

- $\dot{x} = Ax$
- equilibrium point is UNIQUE $Ax = 0 \rightarrow x = 0$
- If eigenvalues of A ($\lambda(A)$) have $\mathrm{Re}(\lambda) < 0$ the equilibrium is globally asimptotically stable.

PD CONTROL

Proportional + Derivative action



INPUTS:

- Joint reference trajectories: $q^d(t)$, $\dot{q}^d(t)$
  - ↳ regulation problem: $q^d = const$, $\dot{q}^d = \emptyset$
- Joint positions/velocities (sensor measurements)

CONTROL LAW:

$$u(t) = K_P (q^d(t) - q(t)) + K_d (\dot{q}^d(t) - \dot{q}(t))$$

(position error; prop. gain matrix; derivative gain matrix; velocity error)

Closed-Loop Behaviour

If we replace in dynamics $H(q)\ddot{q} + C(q,\dot{q}) + g(q) = u = K_P(q^d - q) + K_d(\dot{q}^d - \dot{q})$

at equilibrium $(\dot{q} = 0, \ddot{q} = 0)$ with constant reference $(\dot{q}^d = const, \ddot{q}^d = 0)$

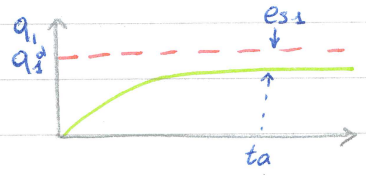
$x_2 = \dot{q} = 0$

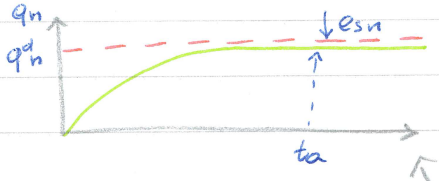
$\dot{x}_2 = \ddot{q} = 0$

$$H\ddot{q} + C(q,\dot{q}) + g(q) = K_P(q^d - q) - K_d(\dot{q}^d - \dot{q})$$

Bigger is K_P, smaller is e_s

Steady state error at equilibrium $e_s = q^d - q = K_P^{-1} g(q)$





- Distant joint has smaller error because it carries less weight
- If we choose K_P, K_d positive definite, equilibrium is asimptotically stable
- Tipically $K_P = diag(K_{Pi})$ $K_d = diag(K_{di})$ → Each joint's position error only affects its own torque
  - ↳ Decentralized linear control (local to each joint)

NOTE: It's hard to define optimal values of K_P, K_d for the whole workspace

K_P, K_d corresponds to STIFNESS OF VIRTUAL SPRINGS and viscosity of VIRTUAL DAMPERS

PD + GRAVITY COMPENSATION

Idea: add a non-linear cancellation of gravity in the control law

$$u = K_p(q^d - q) + K_d(\dot{q}^d - \dot{q}) + g(q) \qquad \text{at equilibrium } \dot{q}, \ddot{q} = 0$$

error — derivative of error — ↳ gravity compensation

$$M\ddot{q} + C(q,\dot{q}) + g(q) = K_p(q^d - q) + K_d(\dot{q}^d - \dot{q}) + g(q)$$

$M\ddot{q}$, $C(q,\dot{q})$, $K_d(\dot{q}^d - \dot{q})$ si semplificano per la condizione di equilibrio $\dot{q}, \ddot{q} = 0$

$$K_p e_s = 0 \Rightarrow e_s = 0$$ stato stazionario

- ⊕ At equilibrium $q \to q^d$, $\dot{q} \to 0$ no steady-state error
- ⊕ Global asymptotic stability (ensured if $K_p > 0$, $K_d > 0$)
- ⊖ We need to identify $g(q)$
  - ⊖ If $g(q)$ is cancelled approximately the steady state error is not zero and $q \to q^* \neq q^d$ ⇒ if you increase K_p, $q \to q^d$
    ⇒ In real system $K_p \uparrow \to$ instabilities

PID CONTROL

We can see gravity as a constant disturbance acting on the system.

$$u(t) = K_p(e(t)) + K_d(\dot{e}(t)) + K_i \int_0^t (e(t))\, d\tau \qquad e(t) = q^d(t) - q(t)$$

↳ CONTROL LAW — output = u_I — Errore di posizione nel tempo

The integral component grows magnifying the error till the disturbance is not compensated

$e(t) > 0$ $u_I \uparrow$ ⇒ u_I ∝ mean value of e

$e(t) < 0$ $u_I \downarrow$ ↳ Il termine integrale continua a sommare l'errore finché non è bilanciato

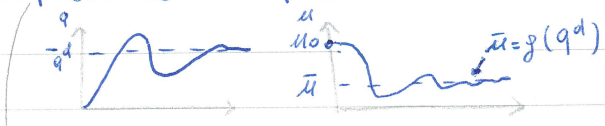
at equilibrium $\dot{q}, \ddot{q} = 0$ we'll have $K_I \int_0^t (q^d - q)\, d\tau = g(q)$ and $e_s = 0$

↳ Integral action compensates gravity — Per $q^d = q$ l'integrale converge ad un valore che bilancia esattam. $g(q^d)$

PID is independent from any robot model.

REGULATORS

$\bar{q}^d(t) = \text{const}$ $\dot{q}^d = 0$



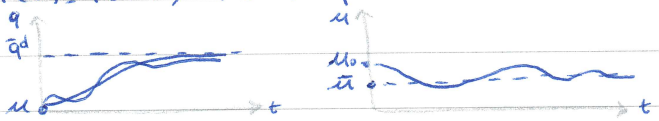
control effort can be high at $T = 0$

↳ In assenza di regolazione, l'obiettivo è portare il robot ad una posizione desiderata e lasciarlo lì

TRAJECTORY PLANNING

With a planner that generates a time varying reference trajectory that interpolates actual to desired position

$q^d(0) = q(0)$, $q^d(T) = \bar{q}^d$



Position error is ϕ at $T = 0$

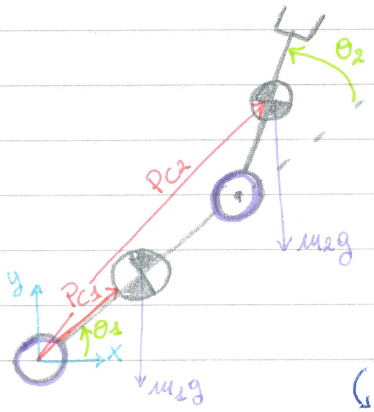
Motion stays in the vicinity of ref. trajectory.

↳ Non si limita a definire un set di obiettivi, ma anche al "come arrivarci"

LEZ. 13 15/05/2025 ROBOT CONTROL – INVERSE DYNAMICS

DECENTRALIZED CONTROL

- Independent controllers for each joint
- Non model based
- Control system is composed of n SISO (single IN / single OUT) loops, ignoring the dynamic COUPLING EFFECTS that are dealt as DISTURBANCES.



Equation of motion of link 1:

$$I_{11}\ddot{q}_1 + \boxed{I_{12}\ddot{q}_2 + I_{112}\dot{q}_1\dot{q}_2 + I_{112}\dot{q}_2^2}^{A} + g_1 = \tau_1$$

Equation of motion of link 2:

$$I_{22}\ddot{q}_2 + \boxed{I_{21}\ddot{q}_1 - \frac{I_{112}}{2}\ddot{q}_1}^{B} + g_2 = \tau_2$$

A: Influence of joint 2 on joint 1

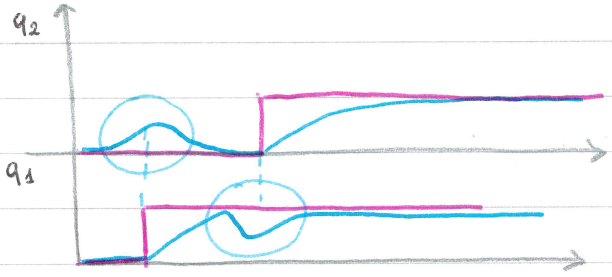
B: Influence of joint 1 on joint 2

↳ This is the coupling effect, ignored in centralized control (or in decentralized?)

Decentralized control Drawbacks:

For a multi-DOF articulated robot, the dynamics of each link is subject also to forces/torques due to: - static loads (gravity)
- motion coupling with other links (inertial, centrifugal)
- its own motion simultaneous with that of other links

These "dynamic couplings" are completely neglected by a decentralized approach.



Irregularities caused by the movement of the other joint

Link 1 and Link 2 influence each other

The effects of these nonlinear couplings and loads can be partially "masked" in the dynamic behavior of a joint axis/motor load if transmissions with high reduction ratios are used.

To solve it we have to consider joint interactions and compensate for them.

CENTRALIZED CONTROL

The controller when computing input torque for a joint takes into account the behaviour of the other joints.

IDEA: compensate the non-linearity to cancel the coupled dynamics and achieve a linear system with low gains (FEEDBACK LINEARIZATION IDEA).

If, to compute the compensation, we employ:

- DESIRED VARIABLES → inverse dynamics
- ACTUAL STATE VARIABLES → computed torque

FEED-BACK LINEARIZATION

- FL is a methodology to control non-linear systems. $\dot{x} = a(x)u + \beta(x)$

  defining a control action as: $u = a(x)^{-1}(v - \beta(x))$

I obtain a linearized system in the new control input v.

$\dot{x} = v$ SINGLE INTEGRATOR $v \rightarrow \boxed{\frac{1}{s}} \rightarrow x$

- If we have a model of the system we can compensate for non linearities.
- The system becomes LINEAR and DECOUPLED wrt any input of the system. (e.g. v_i influences only x_i)

⊕ we can set any stabylizing controller in v

⊖ Perfect cancellation is not possible due to model inaccuracies.

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ H(x_1)^{-1} \end{bmatrix} u + \begin{bmatrix} x_2 \\ -h(x_1, x_2) \end{bmatrix}$$

$$\dot{x} = a(x)u + \beta(x)$$

INVERSE DYNAMIC COMPENSATION

Reference trajectory is used to compute the torques that compensate coupling effects. $q^d(t) \rightarrow$ / $\dot{q}^d(t) \rightarrow$ [MODEL] $\rightarrow \tau$ (torque that compensate coupling effects)

Given a twice differentiable trajectory for $t \in [0, T]$ $q^d(t), \dot{q}^d(t), \ddot{q}^d(t)$ assuming PERFECT KNOWLEDGE of the dynamics applying the FEED-FORWARD action: $u^d = H(q^d)\ddot{q}^d + h(q^d, \dot{q}^d)$ if $q^d(0) = q(0)$, $\dot{q}^d(0) = \dot{q}(0)$ leads to the exact reproduction of the desired motion but:

- initial state not matched
- disturbances
- unmodelled dynamics

} Divergence from desired trajectory

The feed-back term to make the control scheme more robust.

$$u = u^d + K_P(q^d - q) + K_P(\dot{q}^d - \dot{q})$$

open-loop — closed-loop

Feed forward compensates non-linearities

$H(q)\ddot{q} + h(q, \dot{q}) = H(q^d)\ddot{q}^d + h(q^d, \dot{q}^d) + PD$ $\quad PD = K_P e + K_D \dot{e} = \Delta Model$

The compensation is computed on $q^d, \dot{q}^d$ not on $q, \dot{q}$ $\Rightarrow$ Any tracking error causes inexact cancellation

COMPUTED TORQUE CONTROL

The idea is to compute the linearizing controller using the measurements.

We have to do 2 steps:

(1) Evaluate H, h at the current state $q, \dot{q}$

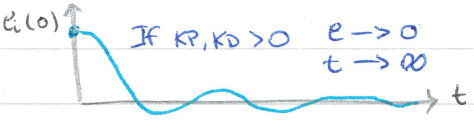
(2) Add an additional feedback.

Final control law can be written as: ① $u = \hat{H}(q)v + \hat{h}(q, \dot{q})$ Non-linear state feedback

replacing in the non-linear dynamics and assuming $\hat{H} = H$, $\hat{h} = h$: $H(q)\ddot{q} + h = Hv + h$

leads to: ② $\ddot{q} = v$ * $\ddot{q} = \begin{bmatrix} \ddot{q}_1 \\ \vdots \\ \ddot{q}_n \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$ n decoupled equations

OUR GOAL IS: $\ddot{e} + K_d\dot{e} + K_P e = 0$ * $\Rightarrow$ tracking error is governed by a 2 order dynamics that can be arbitrarily assigned by suitably selecting gains of K_P, K_D



↳ GLOBAL ASYMPTOTIC STABILITY for any $K_{Pi} > 0$, $K_{Di} > 0$

* L'accelerazione di ogni giunto è direttamente controllata dall'input v

* obtained by LINEAR CONTROL LAW to stabilize the tracking error to zero:

$$v = \ddot{q}^d + K_P(q^d - q) + K_d(\dot{q}^d - \dot{q})$$

The time evolution is governed by the eigenvalues that are the roots of the polynomial:

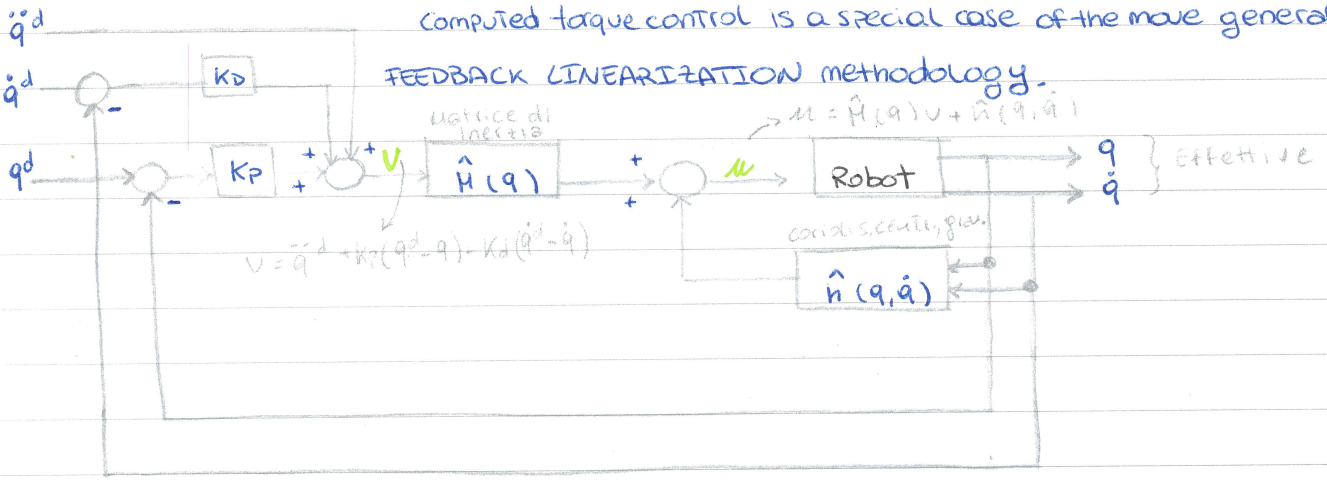
$\ddot{e} + K_D \dot{e} + K_P e = 0 \xrightarrow{\text{Laplace transform}} s^2 E(s) + K_D s E(s) + K_P E(s) = 0 \quad s_{1,2} = \frac{-K_D \pm \sqrt{K_D^2 - 4K_P}}{2}$

If we have a PID: $\left(s^2 + K_D s + K_i \frac{1}{s}\right) e(s) = 0 \rightarrow \underbrace{(s^3 + K_D s^2 + K_P s + K_i)}_{P} e(s) = 0$

↳ Asymptotically stable if the roots of characteristic polynomial P have $Re < 0$

From root criterion: $K_{Di} > 0 \quad K_{Ii} > 0 \quad K_{Pi} > \frac{K_{Ii}}{K_{Di}}$

BLOCK DIAGRAM



Computed torque control is a special case of the more general FEEDBACK LINEARIZATION methodology.

comments on computed torque

- Compensation terms $\hat{M}(q), \hat{h}(q,\dot{q})$ must be computed ONLINE with a small sampling time
- We can use RNEA to compute $\hat{M}(q), \hat{h}(q,\dot{q})$ $\quad M(q) = RNEA(q, 0, 0, e_i)$ $\quad h(q,\dot{q}) = RNEA(q, \dot{q}, 0)$
- A perfect cancellation ($\hat{M} = M$, $\hat{h} = h$) is difficult in practice.
- K_P and K_D can be tuned considering the system decentralized
- If we start from a cartesian trajectory. $p^d(t), \dot{p}^d(t), \ddot{p}^d(t)$, we can map it to joint space and still have a joint space controller:

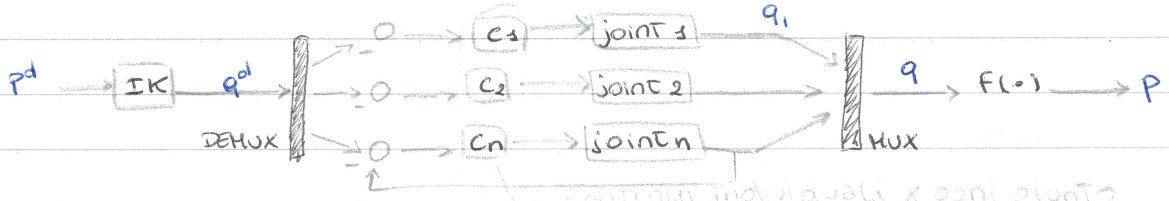
$$q^d(t) = f^{-1}(p^d(t))$$
$$\dot{q}^d(t) = J^{-1}(q^d)\, p^d(t)$$
$$\ddot{q}^d(t) = J^{-1}(q^d)\left[\ddot{p}^d(t) - J(q^d)\,\dot{q}^d\right]$$

LEZ. 14 23/05/2025 TASK SPACE CONTROL - Cartesian Space

Given a reference trajectory for the end-effector $p^d(t)$, compute $u(t)$ such that $p(t) \approx p^d(t)$ where $p \in \mathbb{R}^3$ is the position of the end-effector.

(A) IDEA 1: Inverse Kinematics for end-effector control

compute $q^d(t)$ such that $x^d(t) = f(q^d(t))$ where $f(\cdot)$ is the forward kinematics function. Then apply joint motion control.



if $q \to q^d$ then $p \to p^d$

- ⊕ simple
- ⊕ joint feedback rejects perturbations that don't affect end-effector
- ⊖ Constant joint stiffness results in time-varying stiffness at EE in config changes
- ⊖ Need IK (may be slow in numeric case)

(B) Direct Cartesian Space Control

- The error driving the feedback action is defined at the EE level on the cartesian space coordinates.
- Desired trajectory is defined DIRECTLY in the cartesian space (No IK)
- Measurements in the cartesian space are obtained by forward kinematics computations of joint variables.

IDEA: instead of defining a motion for the joints, directly control the forces that act on the EE

Kinematics: $p = f(q)$ $\dot{p} = J(q)\dot{q}$

Cartesian PD + Gravity Compensation

Create a virtual elastic/viscous attractor at the EE and convert their force/movement into joint torques. Assumption: **No redundancy** ($m = n$)

$$u = J^T(q)\left[K_p(p^d - p) + K_d(\dot{p}^d - \dot{p})\right] + g(q)$$

J^T maps force/movements from EE to joint space

↳ because $n = m \Rightarrow J^T$ square

Redundant Manipulators

visto che con i DoF in eccesso posso eseguire sub-tasks

Employ a null-space method to add a POSTURAL TASK that tries to keep the robot in a DEFAULT CONFIGURATION.

$$u = J^T f^d + [I - J^T J^{T\#}]\, \tau_0$$

compito primario — Null space projector of $J^{T\#}$. Filtra le coppie in modo che non influenzino il compito primario dell'EE, permettendo di usare i DoF ridondanti.

where:

$$f^d = K_p(p^d - p) + K_d(\dot{p}^d - \dot{p}) \quad \text{Cartesian task}$$

$$\tau_0 = K_q(q_0 - q) - K_{\dot{q}}(\dot{q}) \quad \text{Postural Task}$$

compito secondario che punta a riportare il robot verso una joint conf. desiderata

(c) Computed torque in Cartesian space

Joint space dynamics: $M\ddot{q} + h = u$ (1)

[assumption NO redundancy]

To achieve the inversion we need first to consider the dynamic behaviour of the manipulator at the EE (cartesian dynamics).

$$\dot{p} = J\dot{q} \Rightarrow \ddot{p} = J\ddot{q} + \dot{J}\dot{q} \Rightarrow \ddot{q} = J^{-1}(\ddot{p} - \dot{J}\dot{q}) \quad (2)$$

Replacing (2) into (1): $MJ^{-1}(\ddot{p} - \dot{J}\dot{q}) + h = u$

If we left-multiply by J^{-T}: $\underbrace{(J^{-T}MJ^{-1})}_{\Lambda}\ddot{p} + \underbrace{J^{-T}h - (J^{-T}MJ^{-1})\dot{J}\dot{q}}_{\mu} = \underbrace{J^{-T}u}_{F}$

Dynamics Reflected at the EE: $\Lambda(q)\ddot{p} + \mu(q,\dot{q}) = F$ (1)

$\Lambda(q)$: joint space inertia "seen" by EE

$\mu(q,\dot{q})$: cartesian bias term

F: cartesian force

We can apply feedback linearization to linearize (1):

$F = \hat{\Lambda}(q)v + \hat{\mu}(q,\dot{q})$ assuming perfect knowledge of Λ and μ:

$\Lambda\ddot{p} = \Lambda v \rightarrow I_{m\times m}\ddot{p} = v$ (2) unit mass, DECOUPLED SYSTEM

Now we can design a stabilizing controller for the tracking error in the m double integrators of (2):

$v = \ddot{p}^d + K_p(p^d - p) + K_d(\dot{p}^d - \dot{p})$ accelerazione che vogliamo dare a EE

Resulting in the closed-loop dynamics: $I_{m\times m}\underbrace{(\ddot{p}^d - \ddot{p})}_{\dot{e}_x} + K_p\underbrace{(p^d - p)}_{e_x} + K_d\underbrace{(\dot{p}^d - \dot{p})}_{\dot{e}_x} = 0$

Final control law: $u = J^T[\hat{\Lambda}(q)(\ddot{p}^d + K_p e_x + K_d \dot{e}_x) + \hat{\mu}(q,\dot{q})]$

termine di correzione dell'errore dal controllore PD

Physical Interpretation: Inertia depends on q and is variable in different cartesian directions.

When you apply a force F the EE accelerates in a different direction.

Redundant Manipulators

$m < n \Rightarrow J$ is FAT $\Rightarrow \nexists J^{-1}$

We can obtain only I-O decoupling/Linearization (but not the unde STATE dynamics) by replacing J^{-T} with $J^{T\#}$

Alternative Derivation $H\ddot{q} + h = u$ left-multiply by JM^{-1}

$$J\ddot{q} + JM^{-1}h = JM^{-1}u$$

$$\ddot{p} - \dot{J}\dot{q} + JM^{-1}h = JM^{-1}u \quad \text{left-multiply by } \Lambda$$

$$\Lambda\ddot{p} - \Lambda\dot{J}\dot{q} + \Lambda JM^{-1}h = \Lambda JM^{-1}u$$

$$\Lambda\ddot{p} = \underbrace{\Lambda(JM^{-1}h - \dot{J}\dot{q})}_{\mu} = \underbrace{\Lambda JM^{-1}}_{J^{T\#}}u$$

For a redundant robot: $F = J^{T\#}u \Rightarrow \boxed{\Lambda\ddot{p} + \mu = F}$ cartesian Dynamics

Redundancy will remain an internal dynamic that should be stabilized by a postural task in the null-space of $J^{T\#}$.

Final control law: $\boxed{u = J^T[\hat{\Lambda}v + \hat{h}] + [I - J^T J^{T\#}]u_p}$

u_p is the postural task to stabilize dynamics

$u_p = K_q(q^d - q) - K_{\dot{q}}\dot{q}$

(D) Inverse Dynamics + Joint Computed Torque

Solve IK at the acceleration level to get $\ddot{q}^d$ then use joint space inverse dynamics to get u.

[Assume Redundancy (GENERAL CASE)]

$\ddot{p} = J\ddot{q} + \dot{J}\dot{q} \Rightarrow \ddot{q}^d = J^{\#}(v - \dot{J}\dot{q}) + (I - J^{\#}J)q_0$

($\ddot{q}$) IK Joint Space Inverse Dynamics

plug it into the JSID to obtain joint torques:

$u = H\ddot{q}^d + h = H[J^{\#}(v - \dot{J}\dot{q}) + (I - J^{\#}J)q_0] + h$

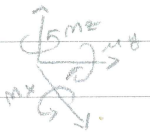
($\ddot{q}$)

↳ FAST TO IMPLEMENT (NO need to COMPUTE Λ)

LEZ. 15 29/05/2025 **ORIENTATION CONTROL IN ROBOT MANIPULATORS**

Until now we focused on cartesian control. Now we want to ~~introduce~~ integrate tracking of orientation of a frame.

$m = [m_x, m_y, m_z] \in \mathbb{R}^3 \rightarrow J_o^T \rightarrow \tau \in \mathbb{R}^n$



PD FOR ORIENTATION CONTROL

$$m = J_o^T (K_o e_o + D_o (\omega^d - J_o \dot{q}))$$

velocità angolare attuale

orientation error

$\dot{e}$

riportarlo all'orientamento desiderato

stabilizzatore

This is equivalent to have torsional springs and dampers in the 3 directions. Momentum is proportional to error $m = K_o e_o$

To control the orientation we need to define an error variable that indicates the deviation between the actual and the desired orientation.

The position error e_p is defined as $e_p = p^d - p$. For the orientation error e_o difficulty arise due to the fact that we need to consider the algebra of the rotation group.

Its expression depends on the particular representation that we choose for the orientation (rot. matrix, angle-axis, euler angles, quaternions)

Orientation error with ROTATION MATRIX

Desired: ^wR_d actual EE orientation: ^wR_e (computed with DK)

$^eR_d = {}^wR_e^T \, {}^wR_d$ orientation of frame d wrt frame e

The orientation error can be defined in 2 ways according to which coordinate frame is used as reference.

Using directly ^eR_d for orientation control is limited, due to ~~it~~ the difficulty of handling the 9 elements of the matrix => use angle-axis

Orientation error using ANGLE-AXIS

vettore la cui direzione punta lungo l'asse di rotazione necessario per correggere l'errore

Define orientation error $e_{e_o} = \hat{r} \sin(\Delta\theta)$ → we can link e_o to the elements of $^eR_d = R$ $\quad e_{e_o} = \frac{1}{2} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$ from R to angle-axis

e_o is expressed in the frame e and therefore it should be mapped in the same frame of J before using for the computation of the restoring movement m: $^we_o = {}^wR_e \, e_{e_o}$

Orientation error with EULER ANGLES

We can define orientation error in the same way of position error:

$e_o = \Phi_d - \Phi_e(q)$ $\quad \Phi_d, \Phi_e$ can be obtained by wR_d and eR_d

If we compute the moment $u_\phi = K_o e_o$ will be defined in a frame with non-orthogonal axes. Let us use analytical Jacobian instead of geometric one to map to torques:

$\dot{q} \rightarrow [J_A] \rightarrow \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix}$ $\quad \dot{q} \rightarrow [J_{A,o}] \rightarrow \dot{\Phi}$ $\quad u \rightarrow [J_{A,o}^T] \rightarrow \tau$

PD control law: $u = J_{A,o}^T [K_o e_o + D_o(\dot{\phi}_d - \dot{\phi})]$ $\quad J_{A,o} = T_w(\phi) J_o$

There are representation singularities in J_A due to $T_w(\phi)$.

Need to derive ϕ_e from $wR_e(q)$ with atan2 -> complex because we have no closed form expression from q to Φ_e.

Orientation error with QUATERNIONS $\quad Q = [\eta \; \varepsilon]$

(1) Compute quaternions Q_e and Q_d associated with wR_e and wR_d

$$Q = \frac{1}{2} \begin{bmatrix} A \\ (R_{23} - R_{32})/A \\ (R_{31} - R_{13})/A \\ (R_{12} - R_{21})/A \end{bmatrix}, \quad A = (1 + R_{11} + R_{22} + R_{33})^{1/2}$$

(2) Composition of quaternions is analogous to composition of rotation matrix and is used the quaternion multiplication $\otimes$ formula

$R_3 = R_1 \cdot R_2$

$\Downarrow$

$$Q_3 = Q_1 \otimes Q_2 = \begin{bmatrix} \eta_1\eta_2 - \varepsilon_1^T \varepsilon_2 \\ \eta_1\varepsilon_2 + \eta_2\varepsilon_1 + \varepsilon_1 \times \varepsilon_2 \end{bmatrix}$$

In our case the quaternion that represents the relative orientation of frame d with respect to frame e is: $eR_d = wR_e^T \, wR_d$

$$\Delta Q = \bar{Q}_e \otimes Q_d = \begin{bmatrix} \eta_e\eta_d + \varepsilon_e^T \varepsilon_d \\ \eta_e\varepsilon_d - \eta_d\varepsilon_e - \varepsilon_e \times \varepsilon_d \end{bmatrix}$$

$\Downarrow$

$e\varepsilon_o$ vector part of ΔQ

e_o is expressed in e frame and need to be mapped into w frame.

No singularity.

Parte usata come errore per il controllo

LEZ. 16 CONTROL OF INTERACTION IN ROBOT MANIPULATORS

The robot has to interact with the environment, so we need to know how the object is behaving. The objective is **regulate contact forces**.

Interaction task of interest/objectives: the interaction task with the environment of interest, usually require

- accurate following/reproduction by the robot end-effector of desired traj. defined on the surface of objects;
- control of forces/torques applied at the contact with environments having low (soft) or high (rigid) stiffness.

Contour following the use of a purely positional control strategy may lead to problems due to positioning errors and uncertainties in task planning related to an incomplete knowledge of the environment.

Passive vs Active methods

Passive physical springs are introduced between robot and environment, to reduce interaction force.

⊕ higher control stability, infinite ~~control~~ force bandwidth

⊖ low flexibility

Active use feedback control technique.

(A) **Direct force control** employs explicit force feedback

(B) **Indirect force control** force and position at the same time

(C) **Hybrid position and force control**: control force in some direction and position in others

Passive control of compliance with RCC

Endw the manipulator with devices that facilitates the execution of the task in a passive way. It's tipically placed between robot's wrist and the gripper.

Direct force control

GOAL: we have a reference force f^d that we want to track.

IDEA: (1) measure contact force f

(2) if $f < f^d$ => apply more force

(3) if $f > f^d$ => apply less force

$$f^* = f^d + K_f \underbrace{(f^d - f)}_{ef} + \text{integral}$$

Termine integrale nel controllo (PID) per cancellare errori sistematici

f^*: force to apply

$$\tau = -J^T f^* + h$$

external force

DYNAMICS: $M\ddot{q} + h = \tau + J^T f$

- at steady state: $\cancel{g} - J^T f = -J^T f^* + \cancel{g} \Rightarrow f = f^d + K_f e_f$
  - ↳ PROBLEM: the choice of K can cause instabilities

Active methods: Impedance control (Indirect force control)

POSITION CONTROL: no matter the force applied

FORCE CONTROL: no matter which position achieve

} Impedance control is in the middle

IDEA: indirectly regulate contact forces by generating a motion that satisfies a dynamic relationship between force and position.

mechanical impedance: dynamic relation between force and velocity (or position)

$$M_x \ddot{P} + D_x \dot{P} + K_x P = F$$

mass, damping, stiffness

IMPEDANCE IN CARTESIAN SPACE

A mechanical impedance gives an idea on how a point of a system moves if you apply a force on it.

No need of environment model describing how reaction forces are generated by the environment as a consequence of a deformation.

Since error loop is missing, we can't control forces but we just try to keep them small.

CARTESIAN SPACE IMPEDANCE CONTROL

Typically the desired impedance is defined wrt a reference trajectory

$$M_x^d(\ddot{p}-\ddot{p}^d)+D_x^d(\dot{p}-\dot{p}^d)+K_x^d(p-p^d)=F_{ext} \quad (1)$$

Desired apparent inertia — Desired Damping — Desired stiffness

Real dynamics (in contact): $M(q)\ddot{q}+h(q,\dot{q})=u+J^T F_{ext}$ (2)

Knowing that $\ddot{q}=J^{-1}(\ddot{p}-\dot{J}\dot{q})$ we can plug it into (2) to get the Torque commands doing feedback linearization in cartesian space (with force measure).

$$\ddot{p}=\ddot{p}^d+M_x^{d-1}\left[D_x^d(\dot{p}^d-\dot{p})+K_x^d(p^d-p)+F_{ext}\right] \quad (3)$$

$$u=MJ^{-1}\left\{\ddot{p}^d-\dot{J}\dot{q}+M_x^{d-1}\left[K_x^d e_x+D_x^d \dot{e}_x\right]\right\}+h+\left[MJ^{-1}M_x^{d-1}-J^T\right]F_{ext}$$

1) ① Prendere la dinamica desiderata (es: ② Trasformarla in accelerazione dei giunti; ③ Aggiungere termini di massa e Coriolis (M, h); ④ compensare F_{ext})

PROBLEM: • requires measure of F_{ext}

• an exact cancellation is hard

INERTIA SHAPING FEATURE

Mask the true inertia of the manipulator and impose a desired one at the end-effector.

If you choose the desired inertia M_x^d equal to the natural cartesian one $\Lambda(q)$

$$u=MJ^{-1}\left\{\ddot{p}^d-\dot{J}\dot{q}\right\}+J^T\left[D_x^d(\dot{p}^d-\dot{p})+K_x^d(p^d-p)\right]+h$$

con comp. di gravità

↳ ≈ PD + FFwD in cartesian space. No contact force feedback needed (F_{ext})

In case of $p^d=const$, $\dot{p}^d=0$, $\ddot{p}^d=0$ this simplifies to a cartesian PD + gravity compensation: $u=J^T\left[K_x^d(p^d-p)-D_x^d\dot{p}\right]+g(q)$

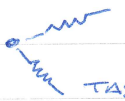
NOTE: for a small displacements a diagonal cartesian stiffness corresponds to a variable joint stiffness.

$$u=\underbrace{J^T K_x J}_{K_q}(q^d-q)$$

$$K_q(q)=J(q)^T K_x J(q)$$

SUPERIMPOSITION OF IMPEDANCES

Because the desired impedances are linear we can superimpose their effects



- Each impedance represents one TASK
- The behaviour will be a compromise between the tasks (if they are conflicting)

Selection of impedance parameters

The impedance parameters are chosen according to the admissible contact force and the desired transient:

1. Select K_x (stiffness) according to how much force we expect for a certain deflexion $K \uparrow\uparrow \Rightarrow$ HIGH FORCE

Thumb Rule: adapt to dynamic characteristics of the environment:

- be stiff with compliant environment
- be compliant if environment is stiff

Desired motion $p^d(t)$ should be planned slightly inside the environment.

2. Select M_x (inertia) according to how "light" we want to "feel" the robot at the end-effector

- Directions where we expect contact: $M^d_{x,i} \uparrow$ reduced acc.
  $K^d_{x,i} \downarrow$ Low force
- Directions where we expect free motion: $M^d_{x,i} \downarrow$ fast acc.
  $K^d_{x,i} \uparrow$ good tracking

3. Select D_x (damping) to regulate the transient. K_x, D_x could also be non linear.

Rotational Impedance

$M_\theta \Delta\ddot{\Phi} + D_\theta \Delta\dot{\Phi} + K_\theta \Delta\Phi = T(\phi)^T \mu_{EXT}$ TORSIONAL IMPEDANCE

↳ due to the presence of T this relation is subject to representation singularities.

ADMITTANCE CONTROL (Indirect force control)

For robots that are only position controlled. Maps contact force into displacements wrt the reference position. Need a force sensor to measure the contact force. Admittance control modifies the setpoints of position control.

CARTESIAN SPACE: $\Delta q \approx J^{-1}(q)\, K_x^{-1}\, F_{ext}$

JOINT SPACE: $\Delta q \approx K_q^{-1}\, J^T F_{ext}$