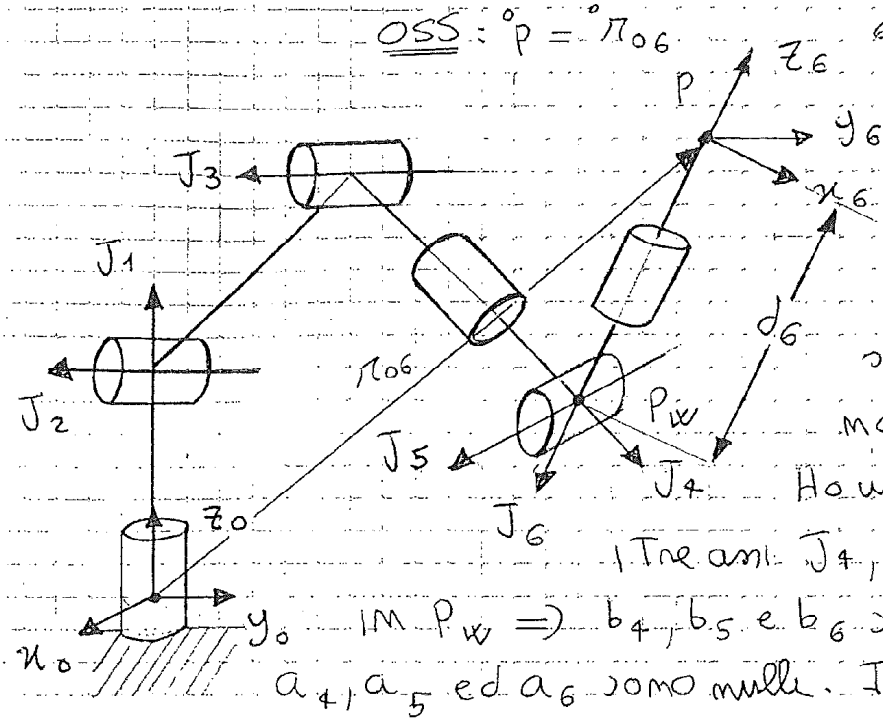
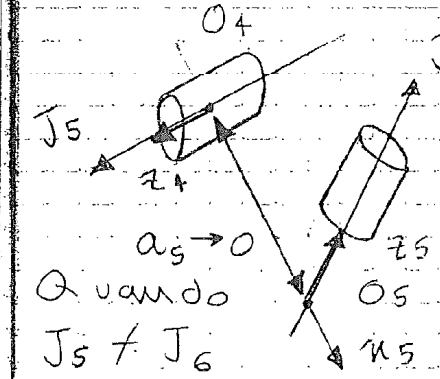
ROBOT ANTROPOMORFO (6 gradi di libertà)



OSS: Je KUKA è un robot antropomorfo madiffers = ce leggermente: infatto J_1 e J_2 mom si intersecano e J_4 mom intersecа J_3. Ho un polsosferico perchè i tre assi J_4, J_5 e J_6 si intersecano in P_W $\Rightarrow$ b_4, b_5 e b_6 sono nulli, o meglio a_4, a_5 ed a_6 sono nulli. Je braccio b_6 ha una lunghezza $\neq 0$ perchè si realizza mediante una Traslazione d_6 del riferimento ${}^6S$ lungo l'asse fittizio J_7 (parallelo a J_6). Infatti se facessimo un ragionamento al limite per J_6 che tende ad intersecare J_5, avremmo $a_5 \to 0$. Allo stesso modo formalmente $a_6 \to 0$, ma il braccio b_6 non è emq nullo perchè d_6 gioca il ruolo di lunghezza del braccio. Analogamente a quanto fatto precedentemente, dato il punto P e l'orientamento di ${}^6S$ (le specifiche sono posizione + orientamento del braccio Terminale), possiamo risalire alla posizione del polso P_W: $\boxed{{}^0P_W = {}^0P - d_6\,{}^0z_6}$



Quando $J_5 \neq J_6$

In coordinate omogenee:

${}^0\tilde{z}_6 = {}^0T_6\,{}^6\tilde{z}_6$, con ${}^6\tilde{z}_6 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ → Vettore non applicato preleva l'orientamento dell'ultima componente soltanto.

Je problema si risolve in questo modo e diventa solo il pro

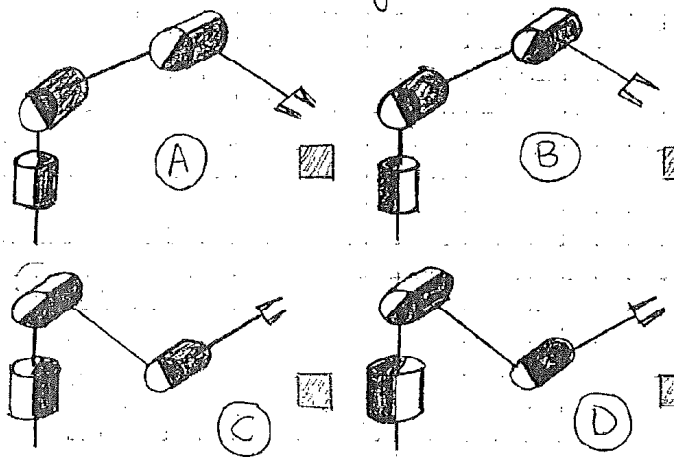
blema di determinare la posizione del centro del polso P_W

$$\begin{bmatrix} {}^0\pi_{03} \\ \hline 1 \end{bmatrix} = \begin{bmatrix} {}^0P_W \\ \hline 1 \end{bmatrix} = {}^0T_1(q_1)\,{}^1T_2(q_2)\,{}^2T_3(q_3)\,{}^3\tilde{\pi}_{33}$$

con ${}^3\tilde{\pi}_{33} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline 1 \end{bmatrix}$ Vettore applicato che indica la posizione del punto P_W nel riferimento ${}^3S$ in coordinate ${}^3S$ $\Rightarrow$

$$\Rightarrow \begin{bmatrix} {}^0\pi_{03} \\ \hline 1 \end{bmatrix} = {}^0T_3(q_1,q_2,q_3) \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline 1 \end{bmatrix} \Rightarrow {}^0P_W = \pi_{03}(q_1,q_2,q_3)$$

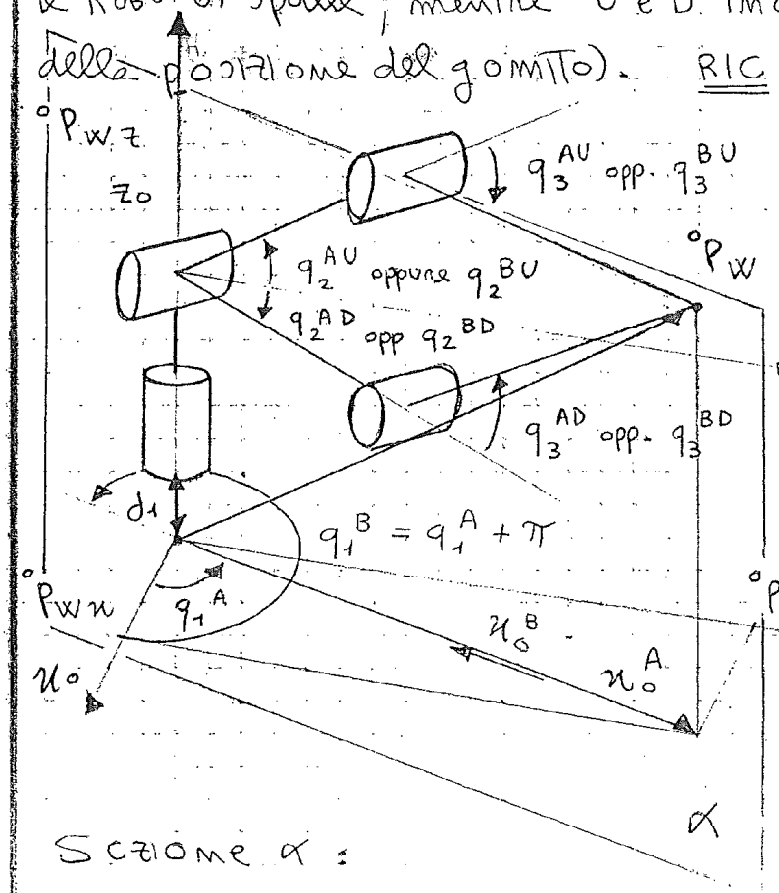
Nel pb di cinematica inversa ${}^0P_W$ è noto perchè calcolato a partire da P (una volta che si conosce l'orientamento della mano) $\Rightarrow$ bisogna calcolare q_1, q_2 e q_3 $\Rightarrow$ si deve risolvere un sistema di 3 eq. in 3 incognite. $\exists$ 4 modi diversi per posizionare il manipolatore nel punto P_W le quattro configurazioni compatibili con la posizione assegnata del centro polso sono disegnate a SX. La (A) e la (C) sono le soluzioni gomito su e gomito giù quando il robot è di fronte all'oggetto mentre (B) e (D) sono le soluzioni gomito su e gomito giù quando il robot è girato di spalle all'oggetto. A ciascuna di queste soluzioni avremo una certa ${}^0R_3$ cioè: ${}^0R_3^A$, ${}^0R_3^B$, ${}^0R_3^C$ ed ${}^0R_3^D$. Quindi dato un certo orientamento del riferimento sul braccio finale ${}^0R_6$ $\Rightarrow$



$$\underbrace{{}^0R_6}_{\text{orientamento imposto dalla specifica}} = \underbrace{{}^0R_3}_{\text{orientamento imposto dalla configurazione scelta}}\ \underbrace{{}^3R_6(q_4,q_5,q_6)}_{\text{orientamento dato dal polso}} \Rightarrow \boxed{{}^3R_6(q_4,q_5,q_6) = {}^0R_3^T\,{}^0R_6}$$

Visto che ho quattro configurazioni diverse per posizionare il polso, ciascuna con un diverso orientamento dello stesso ⇒ avremo 4 terne di soluzioni q_3, q_4, q_5 associate alle varie configurazioni. Questi sono algoritmi esatti con soluzioni in forma chiusa.

OSS: D'ora in poi chiameremo le soluzioni (A) e (C) rispettivamente q^{AU} e q^{AD}, mentre le (B) e (D) rispettivamente q^{BU} e q^{BD} (essendo A le soluzioni con robot di fronte e B quelle con il robot di spalle, mentre U e D indicano Up e Down a seconda della posizione del gomito).



RIC: Data una posizione del polso specifica da realizzare, gli angoli del giunto 1 che realizzano sono due a seconda che il robot si trovi di fronte o di spalle all'oggetto da afferrare:

$$q_1^A = \text{atan2}(^0P_{WY}, {}^0P_{WX})$$

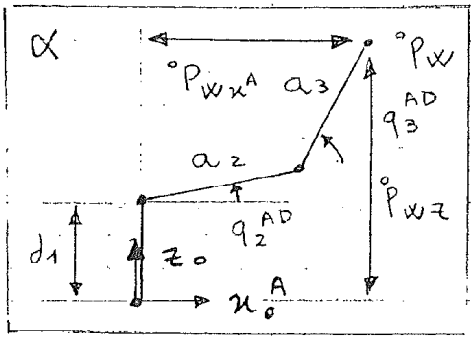
$$q_1^B = \text{atan2}(-{}^0P_{WY}, -{}^0P_{WX})$$

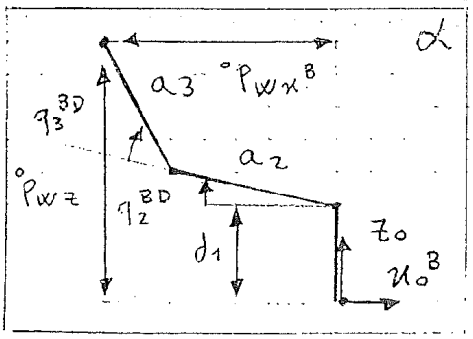
con $q_1^B = q_1^A + \pi$

e con $q_1^A, q_1^B \in [0, 2\pi)$.

Se ora consideriamo la posizione gomito giù con robot di fronte, la sezione α si presenta come mostrato a SX, mentre se il robot è di spalle, sempre con il gomito giù, allora la sezione α è quella indicata a DX.

Sezione α:





Possiamo quindi scrivere l'eq:

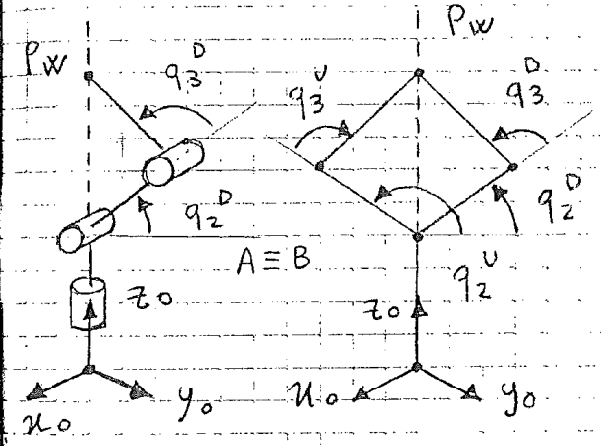
$$\beta\sqrt{{}^0p_{wx}^2 + {}^0p_{wy}^2} = a_2\cos q_2 + a_3\cos(q_2+q_3)$$

$${}^0p_{wz} - d_1 = a_2\sin q_2 + a_3\sin(q_2+q_3)$$

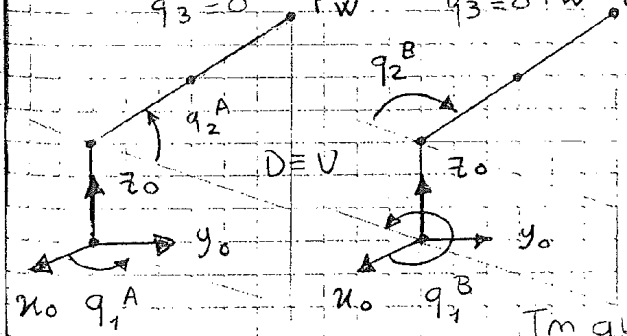
Dove β è un coefficiente che vale ± 1 a seconda di q_1:

$$\begin{cases} q_1^A \Rightarrow \beta = 1 \Rightarrow q_2^{AU} \text{ e } q_3^{AU} \text{ oppure } q_2^{AD} \text{ e } q_3^{AD} \\ q_1^B \Rightarrow \beta = -1 \Rightarrow q_2^{BU} \text{ e } q_3^{BU} \text{ oppure } q_2^{BD} \text{ e } q_3^{BD} \end{cases}$$

Quindi A, B distinguono le soluzioni con polsotorno dritto da quelle con polsotorno rovesciato mentre U, D distinguono quelle gomito su da quelle gomito giù. $\exists$ delle soluzioni singolari che sono quelle in cui $A \equiv B$, cioè il Terminale: $\beta\sqrt{{}^0p_{wx}^2 + {}^0p_{wy}^2} = 0$ ed il punto P_w giace sull'asse z_0 (cioè ${}^0P_w$ ha componenti ${}^0p_{wx}$ e ${}^0p_{wy}$ nulle).



A delle due configurazioni, gomito su e gomito giù, ho ∞ soluzioni perché il polsotorno è libero di ruotare mentre P_w mantiene la sua posizione. $\exists$ anche un'altra configurazione del robot che produce delle soluzioni singolari ed è quella a braccio completamente esteso ($q_3 = 0$)



$$q_1^A \rightarrow q_2^A,\ q_3^A = 0$$

$$q_1^B \rightarrow q_2^B,\ q_3^B = 0$$

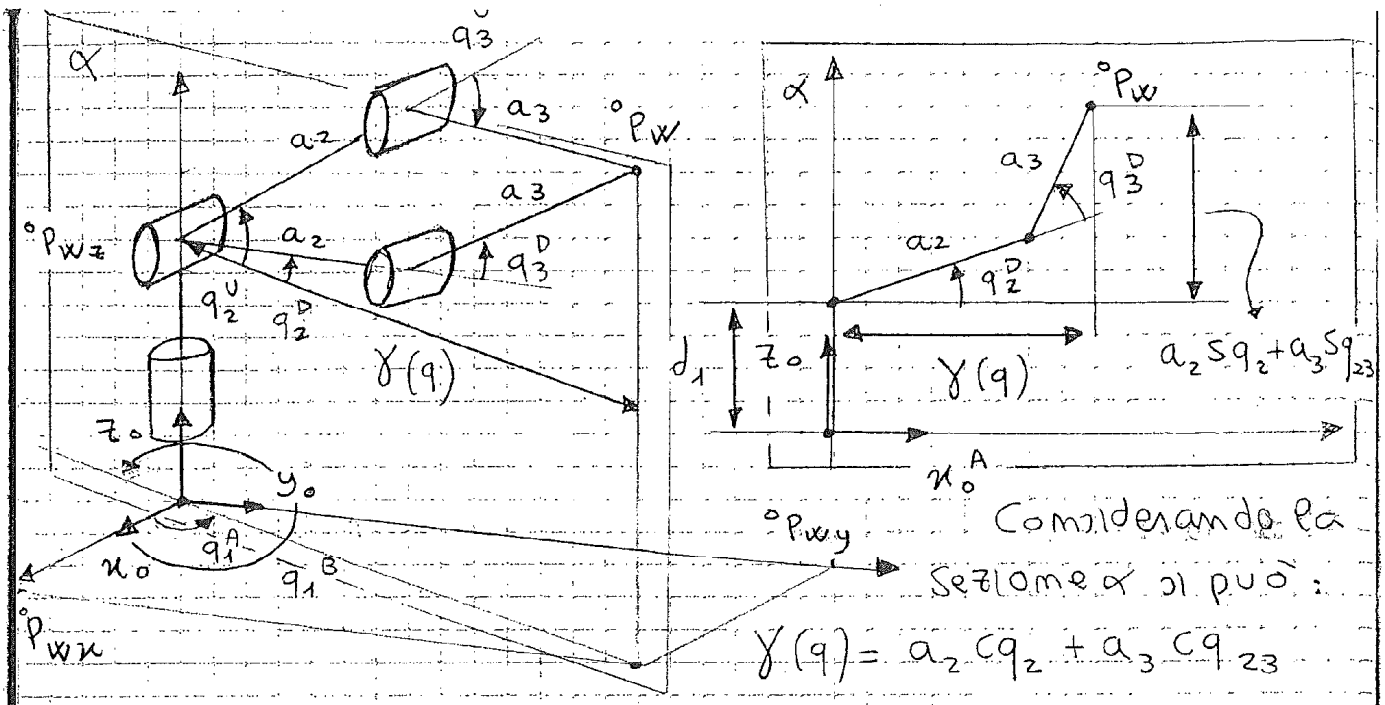
$$\text{con} \begin{cases} q_2^B = \pi - q_2^A \\ q_1^B = \pi + q_1^A \end{cases}$$

In questa configurazione del robot, sono le soluzioni gomito su e gomito giù quelle che coincidono (D≡

MATRICE JACOBIANA PER ROBOT A 3 GRADI DI LIBERTÀ

(valida solo per le Traslazioni): ${}^0\pi_{03} = K_p(q) \Rightarrow$ si può ottenere il settore velocità cartesiana derivando $K_p(q)$:

$${}^0\dot{\pi}_{03} = \frac{\partial K_p(q)}{\partial q}\dot{q} = J_p(q)\dot{q}$$ (per semplificare la notazione: ${}^0\pi_{03} = \pi$)



Considerando la sezione α si può:

$$\gamma(q) = a_2 c q_2 + a_3 c q_{23}$$

$$\left.\begin{aligned} {}^0\pi_x &= (a_2 c q_2 + a_3 c q_{23}) \cos q_1 = {}^0P_{Wx} \\ {}^0\pi_y &= (a_2 c q_2 + a_3 c q_{23}) \operatorname{sin} q_1 = {}^0P_{Wy} \\ {}^0\pi_z &= d_1 + a_2 s q_2 + a_3 s q_{23} = {}^0P_{Wz} \end{aligned}\right\} \Rightarrow$$

Sommando i quadrati $\Rightarrow$ ◔

OSS: $\gamma(q) > 0 \Rightarrow q_1^A = \operatorname{atan2}({}^0\pi_y, {}^0\pi_x)$

$\gamma(q) < 0 \Rightarrow q_1^B = \operatorname{atan2}(-{}^0\pi_y, -{}^0\pi_x) = q_1^A + \pi$

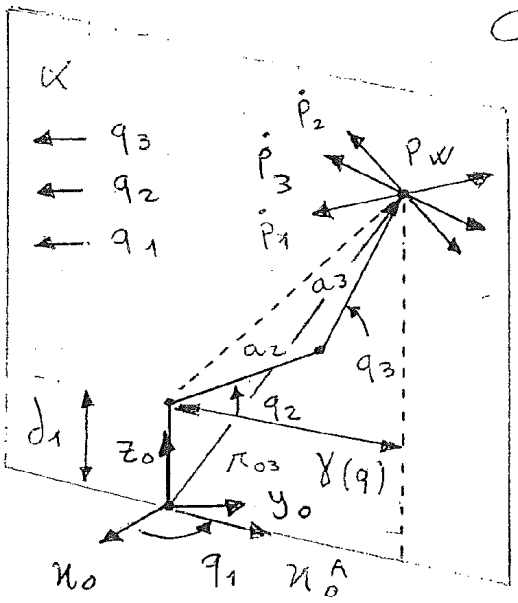
◔ $\Rightarrow$

$$\begin{aligned} \beta \sqrt{{}^0\pi_x^2 + {}^0\pi_y^2} &= \gamma(q) \quad \text{con } \beta = \pm 1 \\ {}^0\pi_z - d_1 &= a_2 s q_2 + a_3 s q_{23} \end{aligned}$$

Riottenuto le formule della cinematica inversa che abbiamo ricavato precedentemente. La matrice Jacobiana $J_p(q)$ è la matrice relativa al contributo delle $\dot{q}$ alla velocità lineare del centro polso P_W:

$$J_p(q) = \begin{bmatrix} \frac{\partial {}^0\pi_x}{\partial q_1} & \frac{\partial {}^0\pi_x}{\partial q_2} & \frac{\partial {}^0\pi_x}{\partial q_3} \\ \frac{\partial {}^0\pi_y}{\partial q_1} & \frac{\partial {}^0\pi_y}{\partial q_2} & \frac{\partial {}^0\pi_y}{\partial q_3} \\ \frac{\partial {}^0\pi_z}{\partial q_1} & \frac{\partial {}^0\pi_z}{\partial q_2} & \frac{\partial {}^0\pi_z}{\partial q_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial {}^0\pi}{\partial q_1} & \frac{\partial {}^0\pi}{\partial q_2} & \frac{\partial {}^0\pi}{\partial q_3} \end{bmatrix}$$

Poniamo interpretare, ognuno di questi tre settori colonna rappresenta il contributo di un determinato giunto (q_1 oppure q_2 oppure q_3) alla velocità cartesiana di P_W, o meglio alla sua velocità lineare.



Come si può vedere dal disegno a SX, il giunto q_1 contribuisce alla velocità lineare di P_W con una componente perpendicolare al piano α individuato dai versori z_0 ed x_0^A (mostrati in verde); mentre i giunti q_2 e q_3 contribuiscono con una componente giacente sul piano α (segnate in rosso ed in meno). Derivando il vettore π_{03} espresso nelle coordinate di ^0S (che abbiamo chiamato $^0\pi_{03} = {}^0\pi$ per brevità di notazione) otteniamo:

$$ {}^0\dot{\pi} = \frac{\partial {}^0\pi}{\partial q_1}\dot{q}_1 + \frac{\partial {}^0\pi}{\partial q_2}\dot{q}_2 + \frac{\partial {}^0\pi}{\partial q_3}\dot{q}_3 = \begin{bmatrix} \frac{\partial {}^0\pi}{\partial q_1} & \frac{\partial {}^0\pi}{\partial q_2} & \frac{\partial {}^0\pi}{\partial q_3} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \Rightarrow $$

$$ \Rightarrow {}^0\dot{\pi} = J_p(q)\,\dot{q} $$

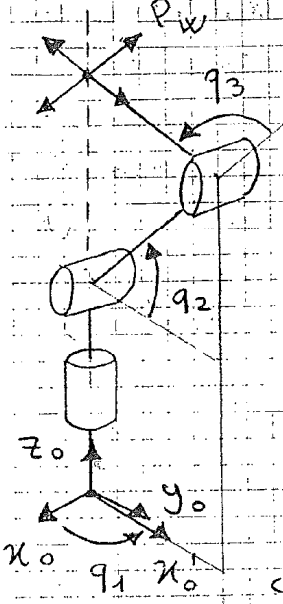
Per il calcolo di $^0\dot{\pi}$ si può procedere sia operando le derivate, sia ragionando intermini geometrici:

$$ {}^0\dot{\pi} = \begin{bmatrix} \boxed{\begin{matrix} -\gamma(q)\,s q_1 \\ \gamma(q)\,c q_1 \\ 0 \end{matrix}} & \boxed{\begin{matrix} -c q_1 (a_2 s q_1 + a_3 s q_{23}) \\ -s q_1 (a_2 s q_2 + a_3 s q_{23}) \\ a_2 c q_2 + a_2 c q_{23} \end{matrix}} & \boxed{\begin{matrix} -c q_1 (a_3 s q_{23}) \\ -s q_1 (a_3 s q_{23}) \\ a_3 c q_{23} \end{matrix}} \end{bmatrix} $$

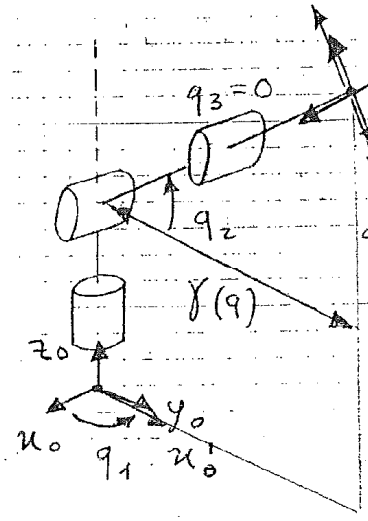
$\partial {}^0\pi / \partial q_1$ $\qquad$ $\partial {}^0\pi / \partial q_2$ $\qquad$ $\partial {}^0\pi / \partial q_3$

<u>OSS</u>: Il versore $x_0' = \begin{bmatrix} c q_1 \\ s q_1 \\ 0 \end{bmatrix}$ è parallelo al piano α. Visto che $\frac{\partial {}^0\pi}{\partial q_1} = \gamma(q)\begin{bmatrix} -s q_1 \\ c q_1 \\ 0 \end{bmatrix}$ è perpendicolare al versore x_0' (perchè ha prodotto scalare nullo $\Rightarrow$ $\partial \pi^0 / \partial q_1$ è una componente $\perp$ al piano α come si evince dal disegno. Il modulo di tale componente dipende da $\gamma(q)$ cioè da quanto il braccio è esteso $\Rightarrow$ il contributo alla velocità cartesiana dipende da $\gamma(q)$. Se $\gamma = 0$ (siamo nella configurazione con $P_W \in z_0$), q_1 non dà contributo alla velocità cartesiana di P_W. I contributi di $\partial {}^0\pi / \partial q_2$ e $\partial \pi / \partial q_3$ sono $\perp$ al contributo di $\partial \pi / \partial q_1$ ed appartengono al piano α. Tuttavia il contributo $\partial \pi / \partial q_2$ è proporzionale ai bracci a_2 ed a_3 mentre $\partial \pi / \partial q_3$ è proporzionale al solo braccio a_3. La matrice $^0\dot{\pi}$ è in genere una matrice a rango pieno pari a 3. Ci sono tuttavia delle particolari configurazioni del robot che provocano cadute di rango:

1) $\gamma = 0$ $\Rightarrow$ Quando il centro ass polso P_W appartiene all'asse z_0, il contributo di q_1 non produce una componente della velocità cartesiana (c'è contributo in $\mathbb{R}^6$ sull'orientamento, ma non in $\mathbb{R}^3$ per quanto riguarda la posizione). I movimenti di q_2 e di q_3 compensano spostamenti di P_W sul piano individuato dai versori z_0 ed x_0' $\Rightarrow$ ho una velocità nulla sul piano α $\Rightarrow$ ho una caduta di rango perché in quella posizione diminuiscono i gradi di libertà del robot in quanto non sono più liberi di muoversi in $\mathbb{R}^3$ ma posso muovermi in $\mathbb{R}^2$. Si dice che il robot che si toglie il cappello non può cantare.



2) $\gamma(q) = \cos q_2 (a_2 + a_3)$ $\Rightarrow$ Un'altra singolarità di $\mathring{\pi}$ si ha quando il braccio è completamente esteso. In tal caso $\gamma(q)$ è la proiezione di (a_2+a_3) sul piano $x_0 y_0$ $\Rightarrow$ dipende cioè dal $\cos q_2$. La $\mathring{\pi}$ diventa:



$$ {}^0\mathring{\pi} = \begin{bmatrix} -c q_2 (a_2+a_3) s q_1 & -s q_2 (a_2+a_3) c q_1 & -s q_2 a_3 c q_1 \\ c q_2 (a_2+a_3) c q_1 & -s q_2 (a_2+a_3) s q_1 & -s q_2 a_3 s q_1 \\ 0 & \underbrace{c q_2 (a_2+a_3)}_{\text{Le ultime due componenti della velocità sono parallele al settore: }} & a_3 c q_2 \end{bmatrix} $$

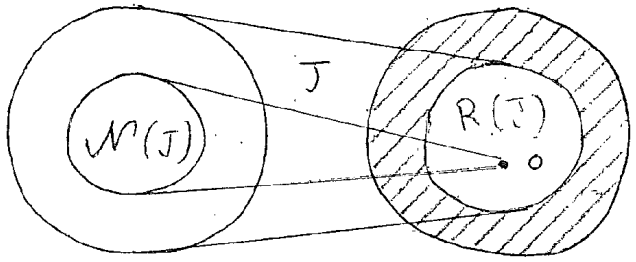
$$ {}^0v = \begin{bmatrix} -s q_2 c q_1 \\ -s q_2 s q_1 \\ c q_2 \end{bmatrix} $$

Infatti la seconda componente è pari ad $(a_2+a_3)\,{}^0v$ mentre la terza componente è pari ad $a_3\,{}^0v$. Ciò mi consente di muovermi cioè lungo una direzione $\Rightarrow$ perdo un altro grado di libertà perché ho ottenuto uno spazio generato da una retta. Questo tipo di singolarità è diverso dal precedente, dal quale si usciva muovendo q_2 oppure q_3. In questo caso infatti si esce dalla singolarità se si muove q_3 mentre movimenti di q_1 o q_2 non fanno uscire dalla condizione di singolarità. Inoltre muovendo di poco q_3 si esce matematicamente dalla singolarità ma praticamente il determinante dello Jacobiano è molto piccolo (quasi zero) $\Rightarrow$ è inversa sarà una matrice i cui elementi hanno valori molto elevati. Ricapitolando, in questo caso lo Jacobiano è:

$J_p(q) = \begin{bmatrix} * & \hat{z}(a_2+a_3) & \hat{z}a_3 \end{bmatrix}$ visto che ho una caduta di rango $\Rightarrow \det = 0 \Rightarrow$ ho un autosettore relativo ad un autovalore nullo. Se impongo al robot una velocità di giunto $\dot{q}$ proporzionale a tale autosettore:

$\dot{q} = \begin{bmatrix} 0 \\ -a_3 \\ a_2+a_3 \end{bmatrix} \Rightarrow$ La velocità cartesiana è nulla nell'istante in cui applico tale velocità ai giunti ed un dt di tempo dopo vale $q_3 = -a_3\, dt \Rightarrow$ si esce dalla singolarità.

OSS: Normalmente un robot non ridondante ha uno Jacobiano che non possiede un nullo (il nullo si viene a creare solo nel caso in cui il robot assume una configurazione che dà luogo a singolarità). Un braccio esteso ad esempio dà luogo a singolarità dello Jacobiano. La matrice Jacobiana $J_p(q)$ è un operatore di trasformazione che trasforma le velocità di giunto in velocità cartesiane o meno: $J_p(q) : \mathbb{R}^m \to \mathbb{R}^m$



L'area ombreggiata, rappresenta l'insieme delle velocità cartesiane non realizzabili nella configurazione q. L'area nera è l'insieme delle velocità di giunto che producono velocità cartesiana nulla nella configurazione q, mentre l'area blu è l'insieme delle velocità di giunto che danno luogo a tutte le velocità cartesiane realizzabili con la configurazione q. Nella programmazione di un robot è bene non evitare il passaggio vicino a configurazioni che danno luogo a delle singolarità.

OSS: Mediante la matrice Jacobiana $J_p(q)$ è possibile risolvere il Pb cinematico inverso differenziale: cioè data la configurazione q del robot, quali devono essere le velocità di giunto $\dot{q}$ affinché le velocità cartesiane siano quelle desiderate $\hat{x}_d$? $q \to \hat{x}(q) = \hat{x}(q)_{desiderata} = \hat{z}_d$, $\dot{q}$? La soluzione:

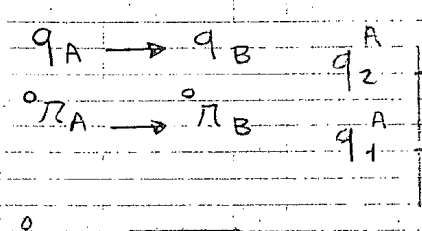
$\boxed{\dot{q}_d = J_p^{-1}(q)\, \hat{x}_d} \iff$ Se robot non è ridondante (cioè $J_p(q)$ è quadrata) e siamo lontani da una configurazione che produce singolarità (infatti se così non fosse $J^{-1} \to \infty \Rightarrow$ per avere una piccola velocità

cartesiana, occorrono delle velocità di giunto motrici esterne).
Il pb di cinematica differenziale diretta risolve banalmente

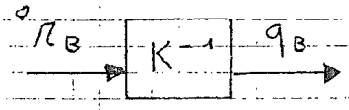
$$\dot{\pi} = J_p(q)\,\dot{q}$$

essendo $\dot{q}$ nota e si vuole conoscere qual'è la velocità cartesiana $\dot{\pi}$ corrispondente a tale velocità di giunto.

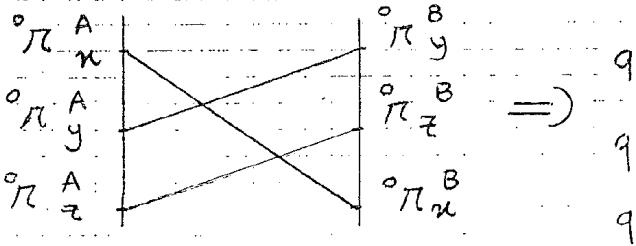
OSS: Se ad esempio la traiettoria cartesiana del punto P_W passa sull'asse z_0 oppure siamo a tale asse $\Rightarrow$ anche se la velocità cartesiana si mantiene costante, si ha una brusca accelerazione delle velocità di giunto nel passaggio attraverso z_0. Se specifico la traiettoria nello spazio dei giunti, posso fare in modo di non avere passaggi che producono delle singolarità mentre se specifico la traiettoria nello spazio cartesiano potrebbero verificarsi delle singolarità:



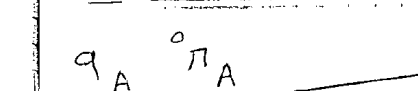
Si fa percorrere ad un giunto la traiettoria q_i una traiettoria rettilinea nello spazio dei giunti per passare dalla configurazione q_A alla configurazione q_B. Ciò corrisponde ad un passaggio dalla posizione $\dot{\pi}_A$ alla $\dot{\pi}_B$ nello spazio cartesiano con una traiettoria curvilinea non meglio nota.



Oppure possiamo pianificare una traiettoria rettilinea nello spazio cartesiano: in modo analogo a tali traiettorie rettilinee nello spazio cartesiano corrispondono delle traiettorie generiche nello spazio dei giunti. Un metodo per realizzare queste traiettorie è operare ee nelle stesse ed applicare l'algoritmo di cinematica inversa in ogni punto $\Rightarrow$ invertiamo la matrice Jacobiana discretizzando, cioè discretizzando la variazione continua di $J(q)$. Questa soluzione può dare pb di singolarità cinematica $\Rightarrow$ questo è il motivo per il quale le traiettorie rettilinee nello spazio cartesiano si



assegnano solo tra punti vicini fra loro. Un movimento più complesso è una composizione di traiettorie rettilinee tra punti vicini. Lo Jacobiano è utile anche per la soluzione del pb cinematico inverso:



${}^0\pi_A = K_p(q_A)$ ⟹ Se movimento da ${}^0\pi_A$ ad ${}^0\pi_B$ nel tempo Δ avviene a velocità: ${}^0v_d = \frac{{}^0\pi_B - {}^0\pi_A}{\Delta}$ ⟹ per avere tale velocità cartesiana che produce spostamento da ${}^0\pi_A$ ad ${}^0\pi_B$ basta imporre una velocità di giunto: $\dot{q} = J_p^{-1}(q)\,{}^0v_d$

Considerando una implementazione nel tempo discreto con tempo di campionamento fisso T_c:

${}^0\pi_A \doteq q(t_0) = q_A = K_p^{-1}({}^0\pi_A)$ ⟹ Integrando da t_0 a t_0+T_c

⟹ $q(t_0+T_c) = q_A + J_p^{-1}(q_A)\,{}^0v\,T_c$

↳ Se T_c è piccolo, e $J(q_A)$ non cambia molto passando da t_0 a t_0+T_c ⟹ $q(t+T_c) = q(t) + J_p^{-1}(q(t))\,{}^0v\,T_c$.

Questa traiettoria della variabile di giunto produrrebbe una traiettoria cartesiana rettilinea se è intervallo di campionamento $T_c \to 0$. Ma nella pratica $T_c \neq 0$ per cui sono introdotti degli errori e la traiettoria reale si discosta da quella ideale.

Per ovviare a questo inconveniente si possono introdurre dei meccanismi di correzione, tra i quali c'è il ricalcolo della della traiettoria dopo un numero prefissato N di passi:

${}^0\pi_N = K_p(q(NT_c))$ con ${}^0v = \frac{{}^0\pi_B - {}^0\pi_N}{\Delta - NT_c}$ dopodiché si itera l'applicazione dell'algoritmo di base.

ALGORITMI NUMERICI PER IL CALCOLO DELLA CINEMATICA INVERSA

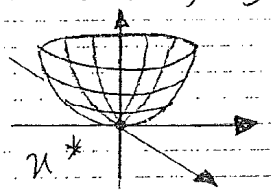
Pb di cinematica inversa: ${}^0p = K(q)$ con ${}^0p \in \mathbb{R}^m$ e $q \in \mathbb{R}^n$, dove $m = 6$ è il numero di gradi di libertà necessari per eseguire il compito (se è utensile è una punta o un verniciatore, il compito è descritto da $m = 5$ gradi di libertà, cioè un grado è ridondante) mentre n è il numero degli assi del robot cioè esprime la sua mobilità. Il Pb di cinematica inversa è, data una posizione desiderata ${}^0p_d$

$p_J = K(q)$ in funzione di q ⇒ sistema di m equazioni non lineari in m incognite. Prima di passare alla risoluzione di questo problema di cinematica inversa, si considera il pb: TEORIA DELL'OTTIMIZZAZIONE NON LINEARE. Data una funzione $V(x)$ scalare non lineare con $x \in \mathbb{R}^m$, cioè $V: \mathbb{R}^m \to \mathbb{R}^+$, si va alla ricerca di un minimo:

$\min_{x \in X} V(x)$ con $X \subseteq \mathbb{R}^m$, X: insieme di definizione di $V(x)$.

questo è un classico problema di minimizzazione. Ci sono molti metodi per calcolare questo minimo. Il legame che intercorre tra il problema di cinematica inversa e la Teoria della ottimizzazione non lineare, è duplice:

1) Se $V(x)$ è una funzione smooth (cioè "liscia", derivabile ∞ volte) ⇒ è ottimo si trova imponendo che sia nullo il gradiente della $V(x)$: questo avviene in corrispondenza del punto di minimo x^*: $\left.\frac{\partial V}{\partial x}\right|_{x^*} = 0$



Il valore di x^* si ricava risolvendo un sistema di m equazioni in m incognite (non lineari) ⇒ pb analogo a quello che bisogna risolvere nel pb di cinematica inversa.

2) Si può pensare di risolvere il problema di minimo della seguente funzione:

$V(q) = (^0p_J - K(q))^T (^0p_J - K(q)) = \|^0p_J - K(q)\|^2$ ⇒ in tal modo risolvo il pb di cinematica inversa ($^0p_J = K(q)$), calcolando il minimo della funzione $V(q)$.

Gli algoritmi per la risoluzione di questo problema sono molteplici. In matlab esistono le funzioni:

- f_{mins} ('calcV', x_0, ...) con calcV.m

  function V = calc(x)

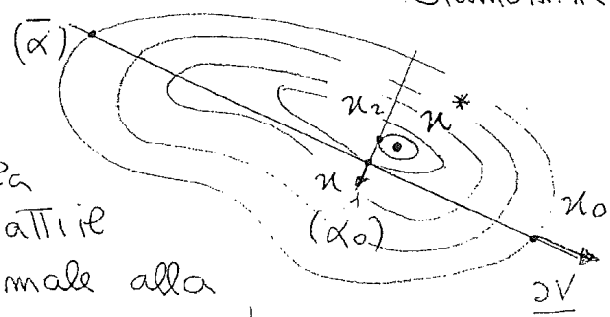
  → Punto iniziale che serve per innescare la sequenza che converge a x^* → punto che minimizza la funzione

f_{mins} calcola il minimo non vincolato della funzione V (il cui valore in x viene calcolato dalla funzione calcV definita in un file.m a parte). Viene imposto solo che la soluzione x^* appartenga all'insieme di definizione $X \subseteq \mathbb{R}^m$ della V. Un algoritmo come f_{mins}, di questo tipo, calcolano la direzione nella quale $V(x)$ diminuisce nel seguente modo:

si calcola $V(x_0)$ e si va poi alla ricerca di un x_1 tale che $V(x_1) < V(x_0)$ $\Rightarrow$ se ho solo $V(x)$ a disposizione, la ricerca può essere lunga perchè si cerca a "caso" la direzione una direzione nella quale la $V(x)$ sia decrescente.

Ci sono algoritmi che si muovono nominuma direzione casuale, ma lungo la direzione di massima discesa della funzione V $\Rightarrow$ lungo la direzione dell'antigradiente (infatti il gradiente di V pari a $\frac{\partial V}{\partial x}$ è normale alla curva di livello ed è diretto lungo la direzione di massima salita della V). Un algoritmo di Matlab che utilizza l'antigradiente $\left(-\frac{\partial V}{\partial x}\right)$ è quello della funzione f_{minunc} (dove unc sta per unconstrained → non vincolato $\Rightarrow$ ∃ anche funzioni che prevedono l'imposizione di vincoli). Tale algoritmo è facilitato nella scelta perchè dopo aver calcolato $V(x_0)$, calcola $-\frac{\partial V(x)}{\partial x}\Big|_{x_0}$ $\Rightarrow$ la direzione è identificata immediatamente.

Siamo in $\mathbb{R}^2$



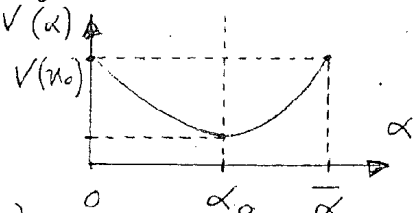
Nasce il Pb di determinare di quanto bisogna muoversi lungo tale direzione, cioè determinare il parametro $\alpha > 0$ tale che:

$$x_1 = x_0 - \alpha\left(\frac{\partial V}{\partial x}\Big|_{x_0}\right)$$

Infatti se α è troppo elevato potrebbe finire su curve di livello più alte di quella di partenza, mentre per α troppo piccolo il processo è troppo lungo. Posso pensare di risolvere un Pb di minimizzazione scalare della funzione calcolata lungo la direzione dell'antigradiente: si introduce allora la seguente funzione scalare:

$$V\left(x_0 - \alpha\frac{\partial V}{\partial x}\Big|_{x_0}\right) = \tilde{V}(\alpha)$$

Funzione scalare in α



Una delle tecniche per la scelta di α, è quella di considerare il valore di α relativo alla curva di livello tangente alla direzione dell'antigradiente, risolvendo l'eq: $\frac{\partial \tilde{V}}{\partial \alpha} = 0$ $\Rightarrow$ otteniamo un valore $\alpha = \alpha_0$ per cui $\frac{\partial \tilde{V}}{\partial \alpha}\Big|_{\alpha_0} = 0$ e si può individuare il punto x_1 che giace su quella particolare curva di livello

(OSS: Se prendiamo α maggiore di α_0, la $V(x)$ torna a crescere fino a raggiungere il valore $V(x_0)$ per $\alpha = \bar{\alpha}$). A questo punto dell'algoritmo si tiene quanto fatto finora, cioè si calcola il

gradiente in x_1, e si cerca nella direzione dell'antigradiente il valore di α relativo alla curva di livello tangente a tale direzione fino a trovare il punto x_2 ... fino a che non si arriva in $x = x^*$ cioè nel punto di minimo. I passi dell'algoritmo sono:

1) x_0, $K = 0$

2) $x(\alpha) = x_K - \alpha \left.\frac{\partial V}{\partial x}\right|_{x_K}$

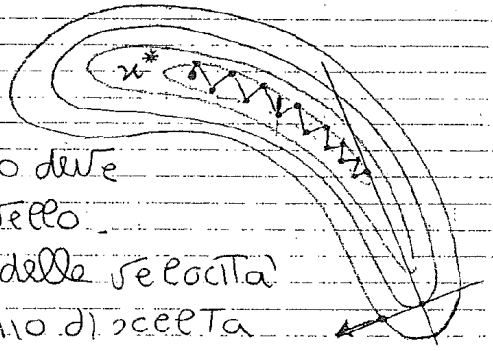
3) $\tilde{V}(\alpha) = V(x(\alpha))$

4) $\alpha_K : \left.\frac{\partial \tilde{V}}{\partial \alpha}\right|_{\alpha_K} = 0$ con $\frac{\partial \tilde{V}}{\partial \alpha} = \frac{\partial V}{\partial x}\frac{\partial x}{\partial \alpha}$

$$x_{K+1} = x_K - \alpha_K \left.\frac{\partial V}{\partial x}\right|_{x_K}^T$$

ALGORITMO DEL GRADIENTE (O DELL'ANTIGRADIENTE)

Non è uno degli algoritmi migliori perché è un po' lento: se le curve di livello presentano un andamento (una forma) a "banana" ⇒



⇒ i tempi di convergenza ad x^* possono essere molto lunghi perché l'algoritmo deve seguire tutta la forma delle curve di livello.

OSS: Il parametro α è responsabile della velocità di convergenza dell'algoritmo. Il criterio di scelta del valore di α (ad ogni passo della procedura iterata) è indipendente dall'algoritmo del gradiente: ∃ infatti altri criteri in base ai quali scegliere α (ES: si può scegliere il primo valore di α per il quale c'è una discesa, cioè per il quale risulti $V(x_0) \geq V(x_1)$ ---).

Per dimostrare la convergenza dell'algoritmo del gradiente si può pensare alla ● come alla soluzione di una eq. differenziale discretizzata. Si consideri una eq. differenziale non lineare:

$$\dot{x}(t) = -\beta \left(\frac{\partial V}{\partial x}(x(t))\right)^T = f(x(t))$$

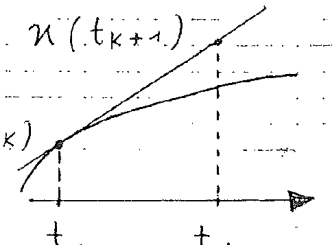
La soluzione con il metodo di Eulero al I° ordine è:

$$x(t_{K+1}) = x(t_K) + f(x(t_K))(t_{K+1} - t_K)$$

Adesso sostituendo $f(x(t)) = -\beta \left(\frac{\partial V(x)}{\partial x}\right)^T$

e ponendo $(t_{K+1} - t_K) = dt$

la soluzione diventa la seguente:



$$x_{k+1} = x_k - \beta \left.\frac{\partial V}{\partial x}\right|_{x_k}^T dt$$

Questa è la soluzione discretizzata della eq. differenziale

$$\begin{cases} x(0) \\ \dot{x}(t) = -\beta \dfrac{\partial V}{\partial x}^T (x(t)) \end{cases}$$

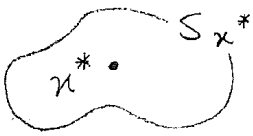
con $\beta > 0$ ed $x(t) \to x^*$

Per dimostrare che l'algoritmo dell'antigradiente converge, basta dimostrare che tale eq. differenziale ammette soluzione $x(t)$ ed $x(t)$ converge a x^*. Si fanno le seguenti Hp sulla $V(x)$:

Hp: 1) $V(x)$ è smooth (funzione "liscia").

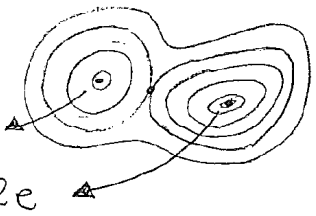
2) $\exists\, S_{x^*} : V^* = V(x^*) < V(x) \;\forall x \in S_{x^*} \setminus \{x^*\}$

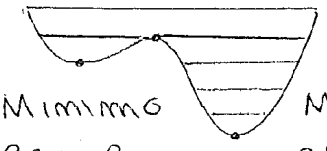
3) $\left.\dfrac{\partial V}{\partial x}\right|_{x^*} = 0$ e $\left.\dfrac{\partial V}{\partial x}\right|_{x \neq x^*} \neq 0$ con $x \in S_{x^*}$



OSS: Le ipotesi 1) e 2) non escludono la presenza o l'esistenza di minimi locali:

1) 2) $\Rightarrow$





Questo algoritmo (e molti altri di ricerca dei minimi) ha il difetto di non riuscire ad identificare con certezza il punto di minimo globale: se infatti x_0 si trova all'interno dell'area racchiusa dalla curva di livello verde $\Rightarrow$ l'algoritmo convergerebbe al minimo locale, se invece x_0 fosse all'interno di quella rossa $\Rightarrow$ l'algoritmo convergerebbe al minimo globale. L'Hp 3) esclude l'esistenza dei minimi locali (il punto stazionario è unico).

Th: $x(t) \to x^*(t)$ si vuole dimostrare che sotto tali Hp la soluzione della eq. differenziale ⊗ converge al punto di minimo

DIM: Si consideri una $\bar{V}(x) = V(x) - V(x^*) > 0$ funzione definita positiva all'interno di $S_{x^*} \to$ è una funzione di Lyapunov candidata, infatti dalla 1), 2) $\Rightarrow$ è liscia ed è definita positiva in S_{x^*}. Andiamo a verificare che $\dot{\bar{V}}(x) < 0$:

$$\dot{\bar{V}}(x) = \frac{\partial \bar{V}}{\partial x} = \frac{\partial V}{\partial x}\dot{x} = \frac{\partial V}{\partial x}\left(-\beta \left(\frac{\partial V}{\partial x}\right)^T\right) = -\beta \frac{\partial V}{\partial x}\left(\frac{\partial V}{\partial x}\right)^T \Rightarrow$$

$V(x^*) = \text{cost.}$ ⊗

OSS: $\dfrac{\partial V}{\partial x} = \left(\dfrac{\partial V}{\partial x_1} \;\cdots\; \dfrac{\partial V}{\partial x_m}\right)$

⊗ $\Rightarrow \dot{\bar{V}}(x) = -\beta \underbrace{\left\| \dfrac{\partial V}{\partial x} \right\|^2} < 0$

La quantità evidenziata in rosso è sempre positiva - si annulla solo

per $x = x^* \Rightarrow x^*$ è un punto di convergenza asintotica $\Rightarrow$ la soluzione $x(t)$ della ⊗ converge ad $x^*(t) \Rightarrow$ l'algoritmo del gradiente converge asintoticamente al punto di minimo (all'ottimo) scegliendo opportunamente α ⊠.

OSS: Non è una convergenza esponenziale, ma solo asintotica per cui potrebbe essere lento.

La soluzione al problema cinematico inverso $\dot{p}_d = K(q)$, può essere indicata più generalmente come soluzione al problema: $y = A(x)$ → Sistema di eq. non lineari che può essere risolto applicando l'algoritmo del gradiente ad una funzione V così definita: $V(x) = (y - A(x))^T (y - A(x))$, con: $A: \mathbb{R}^m \to \mathbb{R}^m$ ed $x, y \in \mathbb{R}^m \Rightarrow$ robot quadrati (non ridondanti)

$$\dot{x}(t) = -\beta \left(\frac{\partial V}{\partial x}\right)^T ; \quad \frac{\partial V}{\partial x} = 2(y - A(x))^T \left(-\frac{\partial A}{\partial x}\right) = -2(y - A(x))^T \frac{\partial A}{\partial x}$$

$$\Rightarrow \left(\frac{\partial V}{\partial x}\right)^T = -2\left(\frac{\partial A}{\partial x}\right)^T (y - A(x)) \Rightarrow \text{Sostituendo } \left(\frac{\partial V}{\partial x}\right)^T \text{ nella espressione di } \dot{x}(t) \Rightarrow$$

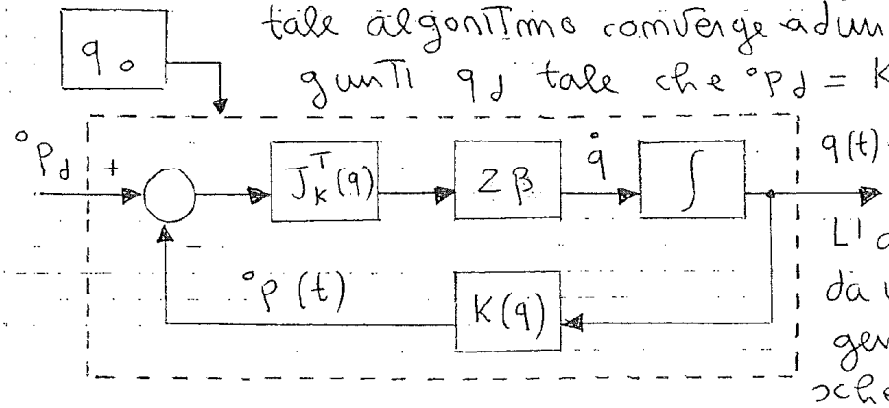
$$\Rightarrow \dot{x}(t) = -\beta \left(\frac{\partial V}{\partial x}\right)^T = 2\beta \left(\frac{\partial A}{\partial x}\right)^T (y - A(x(t))) \rightarrow \text{raggiungo il minimo quando } y - A(x) = 0 \Rightarrow y = A(x),$$

cioè quando il sistema è risolto mentre $x(0)$ rappresenta l'errore iniziale. Tornando allo specifico, in robotica ciò che si ottiene (ricordando che $y = A(x)$ rappresenta $\dot{p}_d = K(q)$ e ponendo $\frac{\partial K}{\partial q} = J_K(q) \rightarrow$ JACOBIANO ANALITICO) è l'algoritmo:

$$\boxed{\dot{q}(t) = 2\beta J_K^T(q(t)) \left(\dot{p}_d - K(q(t))\right)}$$

ALGORITMO DELLO JACOBIANO TRASPOSTO

Lo schema a blocchi reattivo a tale algoritmo è il seguente:



tale algoritmo converge ad una configurazione dei giunti q_d tale che $\dot{p}_d = K(q_d)$.

L'algoritmo è innescato da un configurazione q_0 generica iniziale. Questo schema d'inversione cinematica viene ricavato applicando l'algoritmo del gradiente alla $V(x) = (y - A(x))^T (y - A(x)) \rightarrow A: \mathbb{R}^m \to \mathbb{R}^m$ cioè per robot

quadrati, ma può essere applicato anche a robot ridondanti, ma in tal caso $\nexists$ più un'unica soluzione (cioè $y \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ con $m < n$) $\Rightarrow$ si converge ad una delle traiettorie q.d possibili, non necessariamente la migliore. Il difetto di questo algoritmo è che, se durante una evoluzione da q_0 a q_d si passa attraverso una configurazione singolare potrebbero esserci dei problemi. Cioè se è ennone $({}^0p_d - K(q(t))$, appartiene al nullo dello Jacobiano^T $({}^0p_d - K(q(t))) \in \mathcal{N}(J_K^T(q)) \Rightarrow$ è algoritmo si ferma senza raggiungere la posizione finale desiderata.

OSS: L'analisi della convergenza dell'algoritmo dello Jacobiano^T si può effettuare mediante il criterio di Lyapunov, ma solo per robot quadrato non ridondanti:

$$\dot V(x) = \frac{\partial V}{\partial x}\dot x = -2(y - A(x))^T \frac{\partial A}{\partial x}\dot x \Rightarrow$$

Sostituendo l'espressione di $\dot x$ trovata precedentemente:

$$\Rightarrow \dot V(x) = -2(y - A(x))^T \frac{\partial A}{\partial x} 2\beta \left(\frac{\partial A}{\partial x}\right)^T (y - A(x)) \Rightarrow$$

$$\Rightarrow \dot V(x) = -4\beta \underbrace{(y - A(x))^T \frac{\partial A}{\partial x}\left(\frac{\partial A}{\partial x}\right)^T (y - A(x))}_{\text{Simmetrica e semidefinita positiva}} \leq 0$$

La $\dot V(x)$ si annulla quando $x = x^*$ con $y = A(x^*)$

OSS: La $\dot V(x)$ si può annullare senza che si sia stato raggiunto il punto $x = x^*$ ($\Leftrightarrow y - A(x)$ è un vettore appartenente al nullo dello Jacobiano Trasposto $\Leftrightarrow (y - A(x)) \in \mathcal{N}\left(\left(\frac{\partial A}{\partial x}\right)^T\right)$

L'espressione di $\dot V$ nel caso specifico dei robot è:

$$\boxed{\dot V(q) = -4\beta ({}^0p_d - K(q))^T J_K(q) J_K^T(q) ({}^0p_d - K(q))}$$

Possono allora accadere tre situazioni:

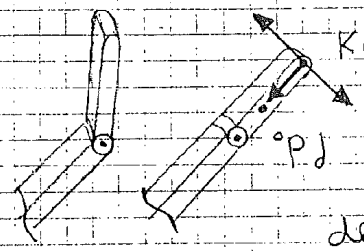
a) Non si passa mai per punti singolari durante l'evoluzione di q;

b) Durante l'evoluzione di q si passa per un punto singolare (cioè $J_K^T(q)$ cade di rango $\Rightarrow$ ha un nullo $\mathcal{N}(J^K(q)^T) \neq \{\emptyset\}$) ma il vettore d'ennone ${}^0p_d - K(q) \notin \mathcal{N}(J^K(q)^T) \Rightarrow$ non ci sono problemi;

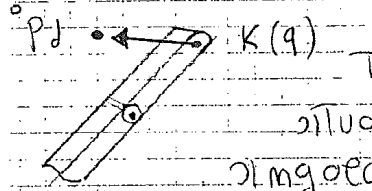
c) Durante l'evoluzione di q si passa per un punto singolare ed il vettore d'ennone appartiene al nullo dello Jacobiano Trasposto $\Rightarrow$ è algoritmo si ferma.

OSS: Se $T^K(q)^T$ è non singolare $\Rightarrow \mathcal{N}(T^K(q)^T) = \{\emptyset\}$

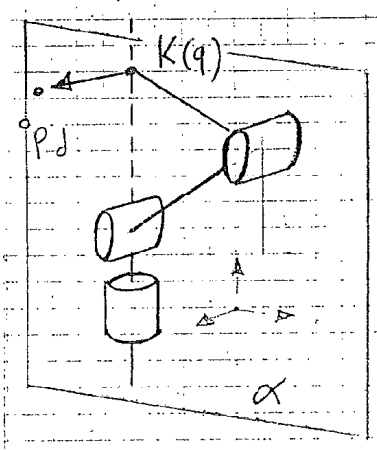
Graficamente la situazione è la seguente: come nel caso a) nel caso a SX, la differenza tra $K(q)$ attuale e la posizione desiderata ${}^0p_d$ è un vettore ortogonale alla direzione di moto possibile, indicata dalle colonne dello Jacobiano Trasposto: $({}^0p_d - K(q))^T [J_1 J_2] = 0$



La direzione del movimento è quindi proibita. In questo secondo caso, l'errore ${}^0p_d - K(q) \notin$ al null dello Jacobiano Trasposto per cui l'algoritmo converge a ${}^0p_d$. La situazione indicata è quella del caso b): infatti c'è singolarità perché il braccio è completamente esteso, ma la direzione di modo non è tale da produrre uno stallo.



Nel caso del robot antropomorfo, l'algoritmo dello Jacobiano Trasposto fallisce quando la configurazione è quella con la mano sull'asse verticale del robot e la direzione dell'errore ${}^0p_d - K(q)$ è perpendicolare al piano X indice To in figura a SX. OSS: Si noti come che è algoritmo è in stallo quando ha $\dot{V}(x) = 0$ ma è errore ${}^0p_d - K(x) \neq 0$. In tali situazioni si impone un movimento casuale per sbloccare l'algoritmo dallo stallo e fare in modo che converga a ${}^0p_d$.



ALGORITMO DI NEWTON APPLICATO ALLA CINEMATICA INVERSA.

Devo trovare una legge $\dot{x}(t) = f(x(t))$ tale che $x(t)$ evolva in $x^*(t)$ cioè $x(t) \to x^*(t)$, con x^* tale che $y = A(x^*)$.

Affinché ciò accada:

$V(x(t)) = (y - A(x))^T (y - A(x)) \to 0$ (in modo che $y = A(x)$)

Voglio che x^* sia il punto di equilibrio Asintoticamente stabile. CNS affinché ciò avvenga (per il Teorema di Lyapunov) è che: $\dot{V}(x) = -2(y - A(x))^T \frac{\partial A}{\partial x} \dot{x} = -2(y - A(x))^T \frac{\partial A}{\partial x} f(x) < 0$

Si può ottenere facilmente se ho $\left|\frac{\partial A}{\partial x}\right| \neq 0 \ \forall x$ e pongo:

$$\Rightarrow \boxed{f(x) = \beta \left(\frac{\partial A}{\partial x}\right)^{-1} (y - A(x))}$$; con questa scelta di $f(x)$ semplifico $\dot{V}$ ottenendo una forma:

$$\dot{V}(x) = -2\beta \underbrace{(y - A(x))^T (y - A(x))}_{V(x) > 0 \text{ : infatti } V(x)} = -2\beta V(x) < 0$$

c) è sempre definita positiva e si annulla solo $x = x^*$. Visto che x è una funzione del Tempo $\Rightarrow$ anche V dipende dal Tempo:

$$\dot{V}(t) = -2\beta V(t) \Rightarrow V(t) = e^{-2\beta t} V(0)$$

$$V(x) = (y - A(x))^T (y - A(x)) = \| y - A(x) \|^2$$

$\Rightarrow$ Sostituendo $V(x)$ in $V(t)$ e operando la radice quadrata

$$\Rightarrow \| y - A(x) \|^2 \leq e^{-2\beta t} \| y - A(x(0)) \|^2 \Rightarrow \text{operando la radice quadrata}$$

$$\Rightarrow \| y - A(x) \| \leq e^{-\beta t} \| y - A(x(0)) \|$$

C'è quindi una convergenza esponenziale della norma dell'errore (maggiorato da una funzione esponenziale decrescente). La $f(x)$ generale imposta: $f(x) = \beta \left(\frac{\partial A}{\partial x} \right)^{-1} (y - A(x))$ specificata in robotica diventa:

$$\boxed{\dot{q}(t) = \beta J_K^{-1}(q(t)) \left({}^{\circ}p_d - K(q(t)) \right)}$$

ALGORITMO DELLO JACOBIANO INVERSO

Lo schema a blocchi rappresentativo dell'algoritmo è lo stesso visto precedentemente (basta sostituire $J_K^T(q)$ con $J_K^{-1}(q)$, e 2β con β). OSS: Prima non si avevano indicazioni sulla convergenza (sulla velocità di convergenza), ora invece so che dell'coefficiente $\beta = \bar{\beta} \Rightarrow$ la velocità di convergenza dipende da $e^{-2\bar{\beta}t}$.

Tuttavia è algoritmo è più oneroso perché si deve calcolare l'inversa al posto della trasposta $\Rightarrow$ per cui vale solo per sistemi quadrati. Inoltre l'algoritmo dello Jacobiano inverso è meno robusto di quello dello Jacobiano Trasposto perché non può mai andare in singolarità (perché altrimenti fallisce) mentre il Trasposto può lavorare anche in singolarità (caso b)). Inoltre il risultato dell'algoritmo può essere o solo un punto finale x^* o anche Tutta una Traiettoria $x^*(t)$, che può essere migliorata o corretta imponendo dei vincoli (posizione finale di max manipolabilità...)

RIC: L'algoritmo dello Jacobiano inverso è basato sull'algoritmo di Newton e risolve pb del Tipo $y = A(x)$. Se parto da una soluzione d'innesco $x_0 \sim x^*$ cioè vicina alla soluzione reale (per la quale risulta $y = A(x^*)$ mentre in genere $y - A(x_0) \neq 0$) posso pensare di sviluppare $A(x^*)$ in serie (sviluppo in serie di un vettore di funzioni al primo ordine: $A(x) = \begin{pmatrix} a_1(x) \\ \vdots \\ a_m(x) \end{pmatrix}$):

$$A(x^*) \cong A(x_0) + \frac{\partial A}{\partial x}\Big|_{x_0} (x^* - x_0) \Rightarrow$$

$A(x^*)$ = y

$\Rightarrow$ Risolvo rispetto x^*: questa equazione

$\Rightarrow y - A(x_0) = \frac{\partial A}{\partial x}\Big|_{x_0}(x^* - x_0) \Rightarrow$ Questa è una eq. lineare nell'incognita x^*. È una situazione ideale, quella in cui si arriva all'ottimo in un solo passo (da x_0 a x^* direttamente). Visto che, generalmente, non miracoliamo $\Rightarrow$ devo iterare il procedimento $\Rightarrow$ algoritmo di Newton. Il primo passo dell'iterazione è:

$$\boxed{x^* = x_0 + \left(\frac{\partial A}{\partial x}\right)^{-1}\Big|_{x_0}\left(y - A(x_0)\right)}$$

$x_1 = x_0 + \left(\frac{\partial A}{\partial x}\right)^{-1}(y - A(x_0))$ mentre il passo generico k è:

$$\boxed{x_{k+1} = x_k + \left(\frac{\partial A}{\partial x}\right)^{-1}(y - A(x_k))}$$

che può essere pensato come soluzione discretizzata della eq. differenziale:

$$\dot{x}(t) = -\beta\left(\frac{\partial A}{\partial x}\right)^{-1}(y - A(x(t)))$$

con $\partial A/\partial x$ matrice Jacobiana della cinematica diretta.

Può inoltre essere utilizzato per cercare il minimo di $V(x): \mathbb{R}^n \to \mathbb{R}^+$ imponendo la condizione di stazionarietà:

$$0 = \left(\frac{\partial V}{\partial x}\right)^T\Big|_{x^*} = A(x) \Rightarrow \frac{\partial A}{\partial x} = \frac{\partial}{\partial x}\left[\left(\frac{\partial V}{\partial x}\right)^T\right] = \boxed{\frac{\partial^2 V}{\partial x^2}}$$

Se ho un algoritmo che risolve $y = A(x)$ (m eq. in n incognite), allora posso risolvere anche questo pb con $y = 0$ e $A(x) = \left(\frac{\partial V}{\partial x}\right)^T_{x^*}$

OSS: $\frac{\partial}{\partial x}\left[\left(\frac{\partial V}{\partial x}\right)^T\right]_{i,j} = \frac{\partial^2 V}{\partial x_i \partial x_j}$ è la matrice Hessiana, cioè la matrice delle derivate seconde (è una matrice simmetrica).

In tal caso l'algoritmo diventa il seguente:

$$\boxed{x_{k+1} = x_k - \left(\frac{\partial^2 V}{\partial x^2}\right)^{-1}\left(\frac{\partial V}{\partial x}\right)^T\Big|_{x_k}}$$ ⊕

Quindi l'algoritmo di Newton può essere utilizzato per la risoluzione di 2 pb:

1) Risolvere il problema $y = A(x) \Rightarrow$ pb di cinematica inversa
2) Risolvere il problema di minimo della $V(x)$.

OSS: L'algoritmo del gradiente per la minimizzazione di una funzione è: $x_{k+1} = x_k - \alpha_k\left(\frac{\partial V}{\partial x}\right)^T \Rightarrow$ È necessario scegliere un α_k che dica di quanto muoversi lungo la direzione di discesa. Nell'algoritmo ⊕ non devo fare questa scelta ma c'è il calcolo di una inversa. Inoltre affinché il pb di minimo ammetta una soluzione, l'Hessiana deve essere una matrice definita positiva (derivata seconda

positiva ⇒ concavità verso è alto ⇒ ∃ minimo della funzione), per cui il prodotto $\left(\frac{\partial^2 V}{\partial x^2}\right)^{-1}\left(\frac{\partial V}{\partial x}\right)^T_{|x^*} > 0$ ⇒ con il meno davanti è direzione è di discesa.

Inoltre è algoritmo di Newton applicato ad un paraboloide (cioè funzioni quadratiche) restituisce è ottimo in un sol passo: sia $V(x)$ una funzione quadratica $V(x) = x^T S x + c^T x + b$ con $S = S^T > 0$

DIM: Il minimo di $V(x)$ si può calcolare uguagliando a zero il gradiente oppure con un sol passo dell'algoritmo di Newton a partire da x_0 (perché è una funzione che assomiglia localmente ad un paraboloide):

ANNULLO GRADIENTE: $\frac{\partial V}{\partial x}_{|x^*} = 0 = 2x^{*T}S + c^T$ (Trasponendo (RIC: $S = S^T$)) $\Rightarrow 2Sx^* + c = 0 \Rightarrow$

$$\Rightarrow \boxed{x^* = -\frac{1}{2}S^{-1}c}$$

Se ora applichiamo è algoritmo di Newton partendo da è innesco x_0:

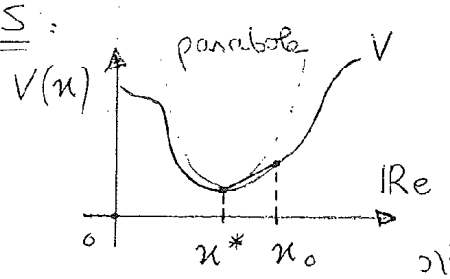
$$x_1 = x_0 - \frac{1}{2}S^{-1}(2Sx_0 + c) \Rightarrow \boxed{x_1 = -\frac{1}{2}S^{-1}c = x^*}$$ ▨

$$\frac{\partial}{\partial x}\left(\frac{\partial V}{\partial x}\right)^T = \frac{\partial}{\partial x}(2Sx + c) = 2S \Rightarrow \left(\frac{\partial^2 V}{\partial x^2}\right)^{-1} = \frac{1}{2}S^{-1}$$

$$\frac{\partial V}{\partial x}^T_{|x_0} = 2Sx_0 + c$$

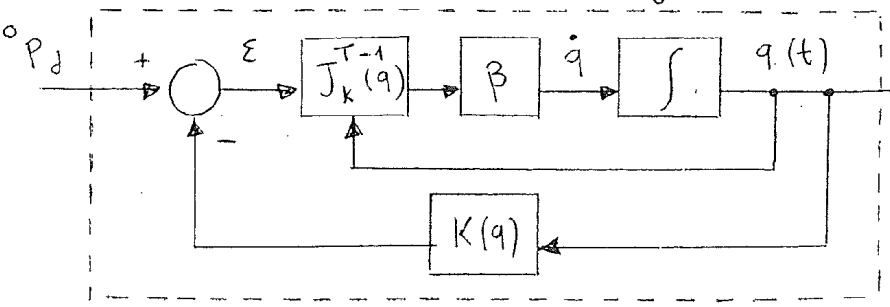
Quindi è algoritmo di Newton applicato ad una funzione quadratica porta alla soluzione con un sol passo. Nel caso della robotica tale algoritmo dà vita all'algoritmo dello Jacobiano inverso ⇒ viene usato per risolvere dei sistemi di eq. non lineari.

ES:



RIC: ALGORITMO GRADIENTE → ALGORITMO JACOBIANO TRASPOSTO
ALGORITMO DI NEWTON → ALGORITMO DELLO JACOBIANO INVERSO

Schema a blocchi relativo all'algoritmo dello Jacobiano { TRASPOSTO / INVERSO



$q(t) \to q_d$:

$^\circ p_d = K(q_d)$

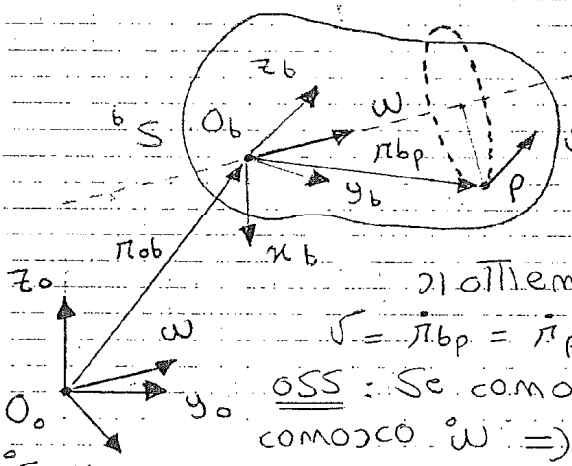
$\dot{q}(t) = \beta J_K^T(q)(^\circ p_d - K(q))$

$\dot{q}(t) = \beta J_K^{-1}(q)(^\circ p_d - K(q))$

OSS: $J_K(q) = \frac{\partial K(q)}{\partial q}$ con $K(q) = \begin{pmatrix} K_p(q) \\ K_o(q) \end{pmatrix}$ ed $\begin{cases} ^\circ\pi_{om} = K_p(q) \\ \widetilde{\varphi} = K_o(q) \end{cases}$

↳ Jacobiano analitico → sfrutta di derivate.

Lo Jacobiano $J_K(q)$ è detto analitico perché differisce da quello geometrico che introdurremo più avanti. Mediante lo schema a blocchi precedente, $\forall {}^0p_d$ si ricava q corrispondente. Dato che ${}^0T_n(q) = {}^0T_1(q_1)\,{}^1T_2(q_2)\cdots{}^{n-1}T_n(q_n)$ è facile calcolare in maniera algoritmica (numericamente) ea T_n dato q. Ma il termine Jacobiano analitico non deve trarre in inganno: infatti è illusorio avere ${}^0\pi_{on}$ e $\tilde{\varphi}$ in funzione di q in forma analitica (forma chiusa difficile da trattare simbolicamente). Lo stesso per ${}^0R_n(q) \to (\varphi, \vartheta, \psi)$. Questo è il motivo per cui si ricorre allo Jacobiano geometrico (più intuitivo fisicamente). Infatti se usiamo lo Jacobiano analitico, per rappresentare la velocità di un robot ci colleghiamo alle: $\begin{cases} {}^0\dot{\pi}_{on} = J_p(q)\,\dot{q} \\ \tilde{\omega} = J_o(q)\,\dot{q} \end{cases}$
In fisica si ragiona meglio come vettore velocità angolare (vettore che rappresenta uno stato di moto e consente di calcolare le velocità mediante prodotti vettoriali). Si consideri un corpo rigido in notazione convelocità ω allora adimane parente per il punto fino O_b (centro del sistema di riferimento ${}^bS$ solidale con il body). Il calcolo della velocità del punto P (sul corpo) si ottiene mediante il prodotto vettoriale:



$v = \dot{\pi}_{bp} = \dot{\pi}_p = \omega \times \pi_{bp}$ → rispetto ad O_b.

OSS: Se conosco le coordinate di ω in ${}^oS$, cioè se conosco ${}^o\omega \Rightarrow$ posso determinare la rotazione del corpo (ciò non sarebbe possibile se conoscessi gli angoli di Eulero φ, ϑ, ψ). Se conosco ${}^o\omega$ ed ${}^o\pi_{bp} \Rightarrow$

$\Rightarrow ({}^o\omega \times {}^o\pi_{bp}) = [{}^o\omega\times]\,{}^o\pi_{bp} \Rightarrow {}^o(\omega \times \pi_{bp}) = [{}^o\omega\times]\,{}^o\pi_{bp}$ in ${}^oS$

L'operatore lineare $[{}^o\omega\times]$ è una matrice antisimmetrica, infatti $\pi \cdot (\omega \times \pi) = 0 \ \forall \pi \Rightarrow {}^o\pi^T[{}^o\omega\times]\,{}^o\pi = 0 \Rightarrow [{}^o\omega\times]$ è una matrice antisimmetrica perché la forma quadratica sempre nulla $\forall {}^o\pi$.

OSS: Dimostriamo quanto detto sopra (cioè che se una matrice ha forma quadratica sempre nulla, allora è antisimmetrica).
HP: $x^T S x = 0 \ \forall x$; Th: $S^T = -S \Rightarrow$

DIM: $-0 = 0^T \Rightarrow -x^T S x = x^T S^T x \Rightarrow x^T (S + S^T) x = 0$.
Una generica matrice A è sempre esprimibile come somma di una parte simmetrica e di una antisimmetrica, cioè $\forall A$ ammette una decomposizione che è unica:

$$A = \frac{A^T + A}{2} + \frac{A - A^T}{2} \Rightarrow A = A_S + A_A$$

→ Solo la parte simm. $_S$ metrica della matrice (A_S) è responsabile della forma quadratica.

Infatti: $\begin{cases} A_A^T = -A_A \\ A_S^T = A_S \end{cases} \Rightarrow x^T A x = x^T A_S x + x^T A_A x = x^T A_S x$ ($x^T A_A x = 0$)

$$x^T A_A x = \frac{1}{2} x^T (A - A^T) x = \frac{1}{2} (x^T A x - x^T A^T x) = 0$$

$x^T A x = (x^T A x)^T = x^T A^T x$ scalare ⊠. Quando scrivo una forma quadratica devo sempre supporre che A sia una matrice simmetrica, in quanto il contributo della eventuale parte antisimmetrica della matrice alla forma quadratica è nullo.

OSS: $\frac{\partial}{\partial x}(x^T A x) = x^T A + x^T A^T = x^T (A + A^T) = 2 x^T A_S$ ($A \neq A^T$)

mentre se $S = S^T \Rightarrow \frac{\partial}{\partial x}(x^T S x) = x^T S + x^T S = 2 x^T S$

Quindi dato che $[{}^o\omega \times]$ dà contributo nullo alla forma quadratica $\forall \pi \Rightarrow$ è una matrice antisimmetrica.

$$[{}^o\omega \times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \Rightarrow {}^o(\omega \times \pi_{bp}) = [{}^o\omega \times]\, {}^o\pi_{bp} = [{}^o\omega \dot{\times}]\, {}^o\pi_{bp}$$

L'unico modo per avere che $A^T = -A \Rightarrow$ diagonale principale $= 0$.

Per conoscere la velocità angolare istantanea $\omega(t)$ associata allo stato di moto istantaneo ${}^oR_m(t)$ di un corpo rigido in un punto P del corpo stesso (${}^oR_m(t) \leftarrow \omega(t)$) → calcolo ${}^o\dot{\pi}_{bp}(t)$ in 2 modi

$${}^o\pi_{bp}(t) = {}^oR_b(t)\, {}^b\pi_{bp} \Rightarrow {}^o\dot{\pi}_{bp}(t) = {}^o\dot{R}_b(t)\, {}^b\pi_{bp} \Rightarrow$$

${}^o\pi_{bp}(t)$: Variamo nel riferimento ${}^oS$; ${}^oR_b(t)$: pura notazione; ${}^b\pi_{bp}$: costante in ${}^bS$

$$\boxed{{}^b\pi_{bp} = {}^oR_b^T(t)\, {}^o\pi_{bp}}$$

$\Rightarrow {}^o\dot{\pi}_{bp}(t) = {}^o\dot{R}_b(t)\, {}^oR_b^T(t)\, {}^o\pi_{bp}(t)$: velocità del punto P solidale al corpo rigido, nota se conosco la ${}^o\dot{R}_b(t)$. Un altro modo per calcolare ${}^o\dot{\pi}_{bp}(t)$ è la seguente:

${}^o\dot{\pi}_{bp}(t) = [{}^o\omega_b(t) \times]\, {}^o\pi_{bp}(t) \Rightarrow$ Eguagliando gli operatori ← $\boxed{[{}^o\omega_b \times] = {}^o\dot{R}_b\, {}^oR_b^T}$ ⊕

Si verifica facilmente che la matrice ${}^0\dot{R}_b\,{}^0R_b^T$ è antisimmetrica perché: ${}^0R_b\,{}^0R_b^T = I \xrightarrow{d/dt} \frac{d({}^0R_b\,{}^0R_b^T)}{dt} = 0 \Rightarrow$

$\Rightarrow \frac{d\,{}^0R_b\,{}^0R_b^T}{dt} = 0 \Rightarrow {}^0\dot{R}_b\,{}^0R_b^T + {}^0R_b\,{}^0\dot{R}_b^T = 0 \Rightarrow$

$\Rightarrow {}^0\dot{R}_b\,{}^0R_b^T = -{}^0R_b\,{}^0\dot{R}_b^T \Rightarrow {}^0\dot{R}_b\,{}^0R_b^T = -\left[{}^0\dot{R}_b\,{}^0R_b^T\right]^T \Rightarrow$

$\Rightarrow {}^0\dot{R}_b\,{}^0R_b^T$ è una matrice antisimmetrica ▨.

Se chiamo l'operatore $[{}^0\omega_b \times] = {}^0\dot{R}_b\,{}^0R_b^T = S \Rightarrow$ posso esprimere le componenti di ${}^0\omega_b$ in funzione delle componenti di S:

Si può fare il viceversa, cioè data la ${}^0\omega_b$ relativa ad una certa ${}^0R_b$ dalla ⊕ ricavo: ${}^0\dot{R}_b = [{}^0\omega_b \times]\,{}^0R_b$

$$\text{Si può fare il viceversa, cioè } {}^0\omega_b = \frac{1}{2}\begin{bmatrix} S_{32}-S_{23} \\ S_{13}-S_{31} \\ S_{21}-S_{12}\end{bmatrix} = \begin{bmatrix} {}^0\omega_x \\ {}^0\omega_y \\ {}^0\omega_z\end{bmatrix}$$

Questa equazione tornerà utile nella dinamica dei robot. Lo Jacobiano geometrico è quell'operatore che trasforma le velocità di giunto nelle velocità di traslazione e nel vettore velocità angolare dell'ultimo braccio (al posto delle derivate degli angoli di Eulero, come avveniva con Jacobiano analitico):

$$\boxed{J(q)\dot{q} = \begin{bmatrix} {}^0\dot{p}_{0n} \\ {}^0\omega_n \end{bmatrix}}$$

Lo Jacobiano geometrico $J(q)$ si indica senza pedici. Per poterlo esprimere bisogna mettere in relazione le velocità angolari del corpo rigido con le derivate degli angoli di Eulero: ${}^0\omega_b = G(\tilde{\varphi})\,\dot{\tilde{\varphi}}$ con $\dot{\tilde{\varphi}} = \begin{pmatrix}\dot{\varphi}\\ \dot{\vartheta}\\ \dot{\psi}\end{pmatrix}$ ⊕

OSS:

$$\begin{bmatrix} {}^0\dot{p}_{0n} \\ \dot{\tilde{\varphi}} \end{bmatrix} = \begin{bmatrix} J_P(q) \\ J_O(q) \end{bmatrix}\dot{q} = J_K(q)\dot{q} = \frac{\partial K}{\partial q}\dot{q} \rightarrow \text{JACOBIANO ANALITICO}$$

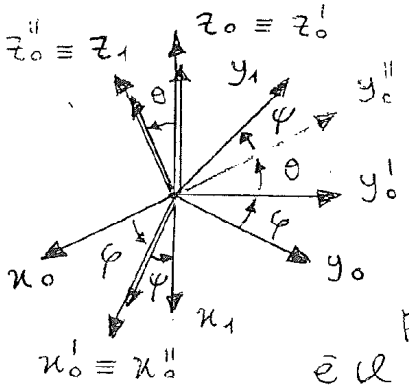
$$\begin{bmatrix} {}^0\dot{p}_{0n} \\ {}^0\omega_{0n} \end{bmatrix} = \begin{bmatrix} J_P(q) \\ J_\omega(q) \end{bmatrix}\dot{q} = J(q)\dot{q} \rightarrow \text{JACOBIANO GEOMETRICO}$$

Lo Jacobiano analitico ha una componente per la rappresentazione della posizione ed una per la rappresentazione dell'orientamento; quello geometrico è identico per quanto riguarda la componente posizionale, la differenza è nella descrizione dell'orientamento:

$^{\circ}\omega_{om} = G(\tilde{\varphi})\,\dot{\tilde{\varphi}} = G(\tilde{\varphi})\,J_o(q)\,\dot{q} = J_\omega(q)\,\dot{q} \Rightarrow$

$\Rightarrow \boxed{J_\omega(q) = G(\tilde{\varphi})\,J_o(q)}$ Cioè $J_\omega(q)$ si può fattorizzare. Dobbiamo trovare $G(\tilde{\varphi})$ per ottenere J_ω e quindi J, senza operare ed. derivate. In forma compatta si può scrivere: $\boxed{J(q) = \begin{bmatrix} I & 0 \\ 0 & G(\tilde{\varphi}) \end{bmatrix} J_K(q)}$

Calcoliamo la matrice $G(\tilde{\varphi})$ in un caso, quando applichiamo la convenzione ZXZ cioè gli angoli φ, ϑ, ψ di Eulero attorno ad assi solidali al riferimento ruotato. Visto che la rotazione avviene attorno ad assi correnti, la composizione si scrive con l'ultima rotazione fatta a DX e la prima a SX:



$^{\circ}R_1 = R(\varphi,\vartheta,\psi) = R_z(\varphi)\,R_x(\vartheta)\,R_z(\psi)$.

$\exists$ 2 modi per calcolare la $^{\circ}\dot{R}_1$, che ci servirà poi per calcolare $[^{\circ}\omega\times] = {}^{\circ}\dot{R}_1\,{}^{\circ}R_1^T$. Il primo è il metodo analitico (si consideri $^{\circ}R_1 = R$):

$$^{\circ}\dot{R}_1 = \frac{\partial R}{\partial\varphi}\dot{\varphi} + \frac{\partial R}{\partial\vartheta}\dot{\vartheta} + \frac{\partial R}{\partial\psi}\dot{\psi} = \dot{R} \Rightarrow \text{calcoliamo } [^{\circ}\omega\times]:$$

$$\Rightarrow [^{\circ}\omega\times] = \dot{R}R^T = \underbrace{\frac{\partial R}{\partial\varphi}R^T\dot{\varphi}}_{①} + \underbrace{\frac{\partial R}{\partial\vartheta}R^T\dot{\vartheta}}_{②} + \underbrace{\frac{\partial R}{\partial\psi}R^T\dot{\psi}}_{③};$$

Calcoliamo separatamente i tre contributi evidenziati per poi ricomporre il tutto:

$$① \Rightarrow \frac{\partial R}{\partial\varphi}R^T = \frac{\partial R_z(\varphi)}{\partial\varphi}R_z^T(\psi)\,R_x^T(\vartheta)\,R_z^T(\varphi) \xrightarrow{R^T = R^{-1}}$$

$$= \left[\frac{\partial R_z(\varphi)}{\partial\varphi}R_x(\vartheta)\,R_z(\psi)\right]R_z^T(\psi)\,R_x^T(\vartheta)\,R_z^T(\varphi) = \frac{\partial R_z(\varphi)}{\partial\varphi}R_z^T(\varphi)$$

Questo è il contributo di φ ad ω; ricordando che:

$$R_z^T(\varphi) = \begin{bmatrix} c\varphi & s\varphi & 0 \\ -s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad e \quad \frac{\partial R_z}{\partial\varphi} = \begin{bmatrix} -s\varphi & -c\varphi & 0 \\ c\varphi & -s\varphi & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow$$

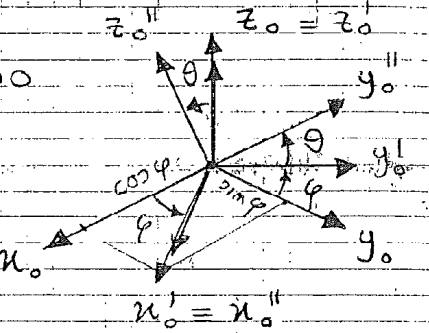
Il contributo al vettore velocità angolare della rotazione di un angolo φ attorno all'asse z ($R_z(\varphi(t))$) è il seguente:

$$\dot{R}_z(\varphi)\,R_z^T(\varphi)\dot{\varphi} = [^{\circ}\omega(t)\times] = \left[\frac{\partial R_z}{\partial\varphi}R_z^T(\varphi)\right]\dot{\varphi} \Rightarrow$$

Rotaz. di φ attorno a z: $^{\circ}\omega(t) = {}^{\circ}z_0\,\dot{\varphi} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dot{\varphi} = {}^{\circ}\omega_z$

$\Rightarrow \dot{R}_z(\varphi) R_z^T(\varphi) \dot{\varphi} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \varphi$ Matrice antisimmetrica nella quale è diversa da zero solo la componente ${}^0\omega_z$ lungo l'asse z ed ha modulo $\dot{\varphi}$ (confrontata con la definizione della matrice $[{}^0\omega\times]$. Il contributo di $\dot{\varphi}$ è esattamente lungo z_0. Consideriamo l'altro contributo $\dot{\vartheta}$ attorno all'asse x_0. Il contributo $\dot{\vartheta}$ attorno a x_0' dà una $\Rightarrow {}^0\omega = {}^0x_0' \dot{\vartheta} \Rightarrow$

$\Rightarrow {}^0\omega = \begin{pmatrix} c\varphi \\ s\varphi \\ 0 \end{pmatrix} \dot{\vartheta}$ Il contributo alla velocità angolare del corpo rigido prodotto da una rotazione di un angolo ϑ attorno ad x_0', si può dedurre facilmente in modo geometrico osservando che x_0' ha coordinate in ${}^0S$ pari a $\begin{pmatrix} c\varphi \\ s\varphi \\ 0 \end{pmatrix}$. Lo stesso risultato si ottiene se si considera il procedimento analitico:



② $\Rightarrow \frac{\partial R}{\partial \vartheta} R^T = R_z(\varphi) \frac{\partial R_x(\vartheta)}{\partial \vartheta} R_z(\varphi) \left(R_z^T(\varphi) R_x^T(\vartheta) R_z^T(\varphi) \right) \Rightarrow$

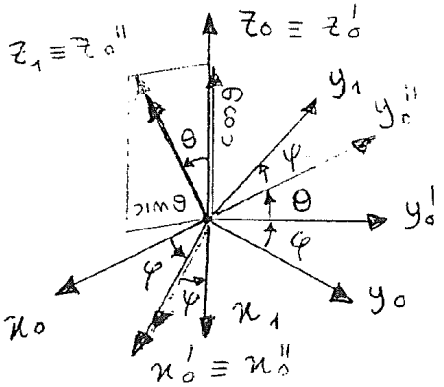
considerando che: $R_x^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\vartheta & s\vartheta \\ 0 & -s\vartheta & c\vartheta \end{bmatrix}$ e $\frac{\partial R_x}{\partial \vartheta} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -s\vartheta & -c\vartheta \\ 0 & c\vartheta & -s\vartheta \end{bmatrix}$

$$\frac{\partial R_x}{\partial \vartheta} R_x^T(\vartheta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\vartheta & s\vartheta \\ 0 & -s\vartheta & c\vartheta \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -s\vartheta & -c\vartheta \\ 0 & c\vartheta & -s\vartheta \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

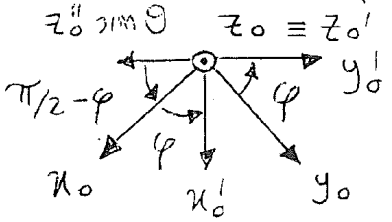
$$\Rightarrow \frac{\partial R}{\partial \vartheta} R^T = R_z(\varphi) \left[\frac{\partial R_x}{\partial \vartheta} R_x^T(\vartheta) \right] R_z^T(\varphi) = R_z(\varphi) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} R_z^T(\varphi) =$$

$$= \begin{bmatrix} c\varphi & -s\varphi & 0 \\ s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c\varphi & s\varphi & 0 \\ -s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & s\varphi \\ 0 & 0 & -c\varphi \\ -s\varphi & c\varphi & 0 \end{bmatrix} \Rightarrow$$

$\Rightarrow \dot{R}_x(\vartheta) R_x^T(\vartheta) \dot{\vartheta} = \begin{bmatrix} 0 & 0 & s\varphi \\ 0 & 0 & -c\varphi \\ -s\varphi & c\varphi & 0 \end{bmatrix} \dot{\vartheta}$ Il contributo di $\dot{\vartheta}$ alla velocità angolare è lungo l'asse x_0' che ha componente $c\varphi$ lungo x_0 e $s\varphi$ lungo y_0. Infatti la matrice: $[{}^0\omega\times] = \begin{bmatrix} 0 & -{}^0\omega_z & {}^0\omega_y \\ {}^0\omega_z & 0 & -{}^0\omega_x \\ -{}^0\omega_y & {}^0\omega_x & 0 \end{bmatrix}$ confrontandola con la matrice ● Trovata, si può notare come $\dot{\vartheta}$ interessi solo le componenti ${}^0\omega_x$ ed ${}^0\omega_y$ come elementi proporzionali rispettivamente a $c\varphi$ e $s\varphi$.



Dall'alto la situazione è la seguente per quanto riguarda la deduzione geometrica dell'ultima componente:



Le coordinate di z_0'' (e quindi di z_1) nel riferimento ^0S sono le seguenti:

$$^0z_1 = \begin{pmatrix} \cos(\pi/2-\varphi)\sin\theta \\ -\sin(\pi/2-\varphi)\sin\theta \\ \cos\theta \end{pmatrix} = \begin{pmatrix} \sin\varphi\sin\theta \\ -\cos\varphi\sin\theta \\ \cos\theta \end{pmatrix}$$

oppure si può ricorrere al metodo analitico visto precedentemente: il contributo di $\dot\varphi$ attorno all'asse $z_0'' \equiv z_1$ alla velocità di rotaz. è:

$$③ \Rightarrow \frac{\partial R}{\partial \varphi} R^T \dot\varphi = R_z(\varphi) R_x(\theta) \frac{\partial R_z(\varphi)}{\partial \varphi} R_z^T(\varphi) R_x^T(\theta) R_z^T(\varphi) \dot\varphi =$$

$$= \begin{bmatrix} c\varphi & -s\varphi & 0 \\ s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} -s\varphi & -c\varphi & 0 \\ c\varphi & -s\varphi & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} c\varphi & s\varphi & 0 \\ -s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} c\varphi & s\varphi & 0 \\ -s\varphi & c\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot\varphi$$

Svolgendo in modo analogo a quanto fatto prima il prodotto, si ottiene che $\dot\varphi$ interessa le componenti $^0\omega_x$, $^0\omega_y$ ed $^0\omega_z$ di $^0\omega$ con elementi proporzionali rispettivamente a $\sin\varphi\sin\theta$, $-\cos\varphi\sin\theta$ e $\cos\theta$. Più brevemente: il contributo di $\dot\varphi$ ad $^0\omega$:

$$^0z_1 \dot\varphi = {}^0R_1\, {}^1z_1 \dot\varphi = {}^0R_1 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \dot\varphi \Rightarrow$$ Il contributo di $\dot\varphi$ ad $^0\omega$ si può ottenere moltiplicando $\dot\varphi$ per l'ultima colonna della $^0R_1 = R_z(\varphi) R_x(\theta) R_z(\varphi)$. Riscrivendo tutti i risultati ottenuti in un'unica formula si ottiene la soluzione del Pb di 3 facciate precedenti:

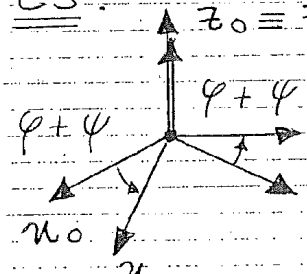
$$^0\omega_1(t) = {}^0z_0\dot\varphi + {}^0x_0'\dot\theta + {}^0z_1\dot\psi = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dot\varphi + \begin{bmatrix} c\varphi \\ s\varphi \\ 0 \end{bmatrix}\dot\theta + \begin{bmatrix} s\varphi s\theta \\ -c\varphi s\theta \\ c\theta \end{bmatrix}\dot\psi =$$

$$= \begin{bmatrix} 0 & c\varphi & s\varphi s\theta \\ 0 & s\varphi & -c\varphi s\theta \\ 1 & 0 & c\theta \end{bmatrix} \begin{bmatrix} \dot\varphi \\ \dot\theta \\ \dot\psi \end{bmatrix} = G(\tilde\varphi)\dot{\tilde\varphi} \Rightarrow \boxed{G(\tilde\varphi) = \begin{bmatrix} 0 & c\varphi & s\varphi s\theta \\ 0 & s\varphi & -c\varphi s\theta \\ 1 & 0 & c\theta \end{bmatrix}}$$

Abbiamo così ricavato la matrice $G(\tilde\varphi)$ che a partire dalle coordinate espresse in angoli di Eulero $\tilde\varphi$ (e dalle derivate di tali angoli), restituisce la velocità angolare ω. Questa rappresentazione, soffre di singolarità: infatti quando $\theta = 0$, si [illegible] la somma dei due rotazioni attorno agli assi z che

rimane sempre lo stesso ma non la natura di rotazione cioè conosco $\varphi+\psi$ ma non conosco φ e ψ separatamente. Infatti quando $\theta=0$, nella matrice di trasformazione $G(\tilde{\varphi})$ ho una caduta di rango: $\dot{\varphi}$ e $\dot{\psi}$ danno un contributo alla velocità angolare attorno allo stesso asse z_0 (che ha coordinate $\begin{pmatrix}0\\0\\1\end{pmatrix}$ in ^0S) $\Rightarrow$ singolarità. Ciò significa che dato un certo moto espresso in angoli di Eulero $\tilde{\varphi}$, $\exists$ sempre una $^0\omega$ tale che $^0\omega = G(\varphi)\dot{\tilde{\varphi}}$ cioè una velocità angolare che esprime tale stato di moto; mentre data una velocità angolare $^0\omega$, non sempre è possibile ottenere la descrizione di tale moto in termini di derivate degli angoli di Eulero cioè $\dot{\tilde{\varphi}} = G^{-1}(\tilde{\varphi})\,{}^0\omega$ in quanto $G(\tilde{\varphi})$ potrebbe non essere invertibile $\Rightarrow$ singolarità di rappresentazione, non è una singolarità strutturale, ma deriva dal metodo adottato per la rappresentazione dell'orientamento. Ciò accade quando voglio dare una rappresentazione in termini di angoli di Eulero e $z_0 \parallel z_n$.

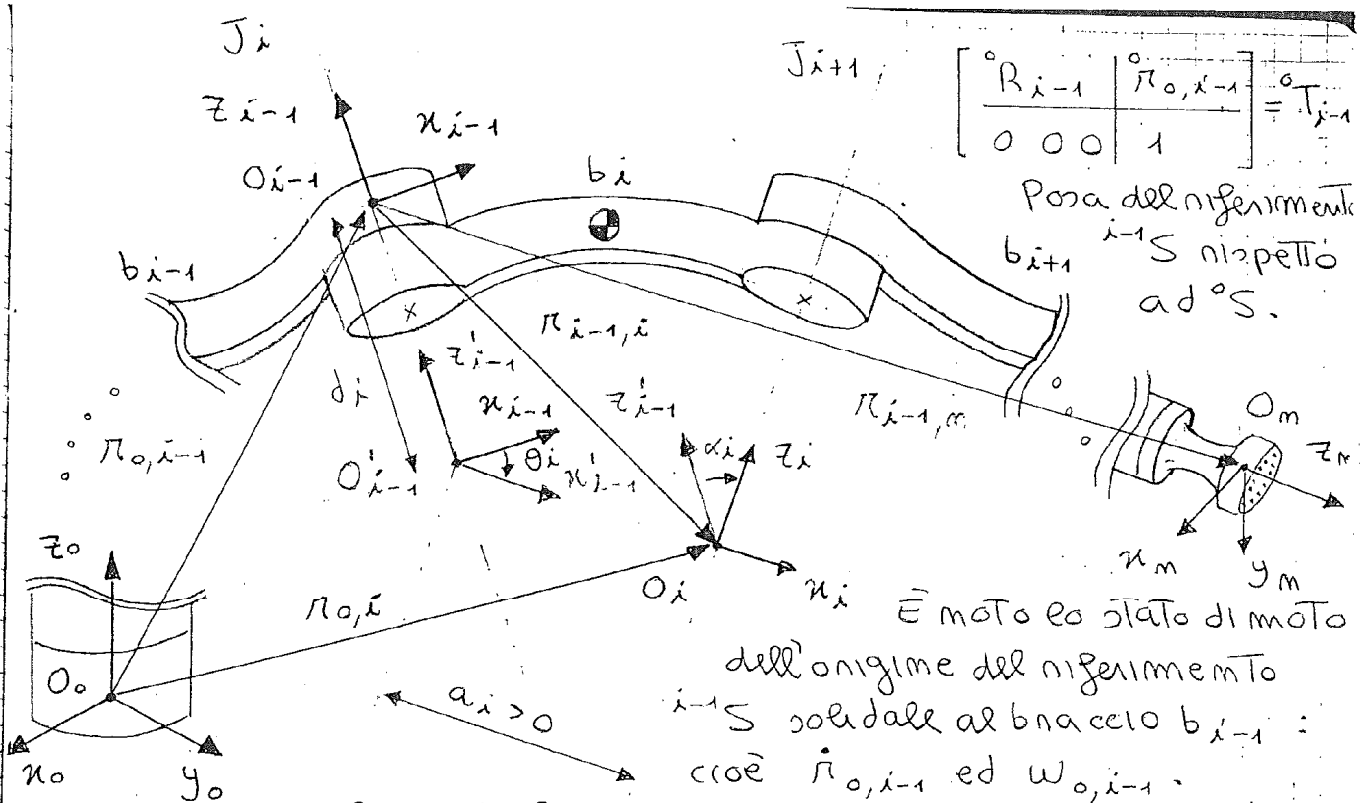
$$G(\tilde{\varphi}) = \begin{bmatrix} 0 & c\varphi & 0 \\ 0 & s\varphi & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



Es: Se voglio rappresentare una notazione attorno a z_0 come la velocità angolare, basta considerare una $\omega_z = \dot{\psi}+\dot{\varphi}$ $\Rightarrow$ avrei ∞ rappresentazioni in termini di derivate degli angoli di Eulero ed un unica rapp. corrispondente in termini di velocità angolare. Una notazione ω_x attorno ad x_1 si rappresenta con $\dot{\theta}$. Non posso rappresentare notazioni attorno ad assi fuori dal piano x_1-z_1. Questi sono i motivi per cui si cerca di evitare ogni qualvolta la rappresentazione $\tilde{\psi}$, e dalle $\dot{q}$ si cerca di ottenere direttamente le velocità angolari ω. Il calcolo dello Jacobiano geometrico si realizza senza effettuare delle derivate, ma ricorrendo agli algoritmi in modo da legare $\dot{q}$ in modo diretto alle velocità lineari ed angolari senza passare attraverso il calcolo esplicato della cinematica diretta:

$$\begin{bmatrix} {}^0\dot{r}_{0n} \\ {}^0\omega_{0n} \end{bmatrix} = J(q)\,\dot{q}$$

per capire il legame esistente tra gli stati di moto di due bracci adiacenti connessi da un giunto, si ricorre ad un disegno simile a quello visto nella convenzione di Denavit-Hartenberg:



$$\begin{bmatrix} {}^0R_{i-1} & {}^0\pi_{0,i-1} \\ 0\ 0\ 0 & 1 \end{bmatrix} = {}^0T_{i-1}$$

Posa del riferimento ${}^{i-1}S$ rispetto ad ${}^0S$.

È moto e stato di moto dell'origine del riferimento ${}^{i-1}S$ solidale al braccio b_{i-1}: cioè ${}^0\dot{\pi}_{0,i-1}$ ed $\omega_{0,i-1}$.

OSS: Per calcolare le energie cinetiche dei bracci, bisogna conoscere le coordinate del baricentro del braccio (che sono costanti nel sistema di riferimento solidale al braccio stesso) e la sua velocità angolare. Per poter calcolare lo stato di moto dell'organo terminale, devo conoscere oltre allo stato di moto del sistema ${}^{i-1}S$ rispetto ad ${}^0S$, anche lo stato di moto del riferimento ${}^iS$ rispetto ad ${}^{i-1}S$ cioè: ${}^{i-1}\dot{\pi}_{i-1,i}$ ed $\omega_{i-1,i}$. Im questo modo si può calcolare:

$$\begin{cases} {}^0\dot{\pi}_{0,i} = {}^0\dot{\pi}_{0,i-1} + {}^0\dot{\pi}_{i-1,i} \\ \omega_{0,i} = \omega_{0,i-1} + \omega_{i-1,i} \end{cases}$$

Questi contributi scritti dobbiamo essere sommati in un unico sistema di riferimento. (qui compaiono dei vettori di E^3, ma dobbiamo sommare coordinate in $\mathbb{R}^3$ espresse in uno stesso sistema di riferimento). Consideriamo il caso in cui il giunto i sia un giunto a cerniera:

$$\begin{cases} \omega_{i-1,i} = z_{i-1}\dot{\theta}_i \quad (\text{direzione } z_{i-1} \text{ e modulo } \dot{\theta}_i) \\ {}^{i-1}\dot{\pi}_{i-1,i} = \omega_{i-1,i} \times \pi_{i-1,i} = (z_{i-1} \times \pi_{i-1,i})\dot{\theta}_i \end{cases}$$

Velocità di rotazione e di traslazione di b_i rispetto b_{i-1}

Quindi gli incrementi che compaiono nella formula generale nel caso di giunto a cerniera sono quelli indicati sopra, mentre nel caso di giunto prismatico, sono quelli indicati di seguito:

$$\begin{cases} \omega_{i-1,i} = 0 \quad . \quad (\text{Non c'è rotazione tra } b_{i-1} \text{ e } b_i) \\ {}^{i-1}\dot{\pi}_{i-1,i} = z_{i-1}\dot{d}_i \quad \text{mentre c'è una traslazione lungo } z_{i-1}. \end{cases}$$

L'algoritmo ricorsivo che andremo a seguire ha inizio con:

$$\begin{cases} {}^{0}\dot{\pi}_{0,0} = 0 \\ {}^{0}\omega_{0,0} = 0 \end{cases}$$

Iterando e calcolando i vettori z_{i-1} ed $\pi_{i-1,i}$ ci muoviamo dal braccio zero fino al braccio m. Questi vettori possono essere precedati dalla matrice di rototraslazione ${}^{i-1}T_i(q_i)$; infatti risulta che z_{i-1} è l'ultima colonna della matrice di rotazione ${}^{i-1}R_i(q_i)$ mentre $\pi_{i-1,i}$ è l'ultima colonna (escluso l'1 finale) della matrice di rototraslazione ${}^{i-1}T_i(q_i)$. Oppure può essere convenientemente partire dal riferimento ${}^{m}S$ e calcolare ${}^{m-1}T_m(q_m) \Rightarrow {}^{m-1}z_{m-1} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ in modo da eseguire somme sempre nelle coordinate locali (in queste coordinate l'asse z si esprime facilmente). Quello che interessa calcolare è la matrice Jacobiana $J(q)$ nella forma completa cioè come composizione della velocità di traslazione e di rotazione:

$${}^{0}\dot{\pi}_{0m} = {}^{0}\dot{\pi}_{0,1} + {}^{0}\dot{\pi}_{1,2} + \dots + {}^{0}\dot{\pi}_{i-1,i} + \dots + {}^{0}\dot{\pi}_{m-1,m}$$

$${}^{0}\omega_{0,m} = {}^{0}\omega_{0,1} + {}^{0}\omega_{1,2} + \dots + {}^{0}\omega_{i-1,i} + \dots + {}^{0}\omega_{m-1,m}$$

Consideriamo un giunto a cerniera; il contributo del giunto i-mo alla velocità di traslazione (bloccando tutti gli altri giunti) dell'organo terminale:

$${}^{0}\dot{\pi}_{i-1,m} = \left([{}^{0}z_{i-1} \times] \, {}^{0}\pi_{i-1,m} \right) \dot{\vartheta}_i$$

Infatti per il giunto i-mo a cerniera si ha che:

$$\begin{cases} \dot{\pi}_{i-1,m} = \omega_{i-1,m} \times \pi_{i-1,m} \\ \omega_{i-1,m} = z_{i-1,m} \dot{\vartheta}_i \end{cases} \Rightarrow \dot{\pi}_{i-1,m} = (\dot{\vartheta}_i z_{i-1}) \times \pi_{i-1,m}$$

A valle del giunto i abbiamo un corpo rigido perché tutti gli altri giunti sono bloccati: la lunghezza di tale corpo è la distanza tra il centro O_{i-1} di ${}^{i-1}S$ ed il centro O_m di ${}^{m}S$. Quindi la struttura dello Jacobiano geometrico è la seguente:

$$J(q) = \begin{bmatrix} \text{TRASL.} \\ \text{ROTAZ.} \end{bmatrix}$$

colonna i-ma

La colonna i-ma del blocco relativo alle traslazioni della matrice Jacobiana geometrica è chiamata $J_{P_i}(q)$, con q_k tale che $k > i$ e $k \leq m$ (è funzione della porzione dei giunti a valle del giunto i), ed è un vettore tale che:

$J_{P_i}(q) \Rightarrow {}^0\dot{r}_{i-1,m} = J_{P_i}(q)\,\dot{q} = \left(\left[{}^0z_{i-1}\times\right]\,{}^0r_{i-1,m}\right)\dot{\theta}_i$

OSS: ${}^0\dot{r}_{0,m} = {}^0\dot{r}_{i-1,m}$ perchè non c'è movimento relativo tra O_0 ed O_{i-1} in quanto i giunti da J1 fino a J_{i-1} sono tutti bloccati e cioè J_i è libero di muoversi.

RIC: Per conoscere il contributo del giunto i-esimo alla velocità cartesiana dell'organo terminale si considera la seguente suddivisione della matrice Jacobiana geometrica:

$$J(q)\dot{q} = \begin{bmatrix} {}^0\dot{r}_{0m} \\ {}^0\omega_{0m} \end{bmatrix} = \begin{bmatrix} \square & \cdots & \begin{matrix} J_{P_i}(q) \\ J_{\omega_i}(q) \end{matrix} & \cdots & \square \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_i \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m} J_{Pk}(q)\,\dot{q}_k \\ \sum_{k=1}^{m} J_{\omega k}(q)\,\dot{q}_k \end{bmatrix}$$

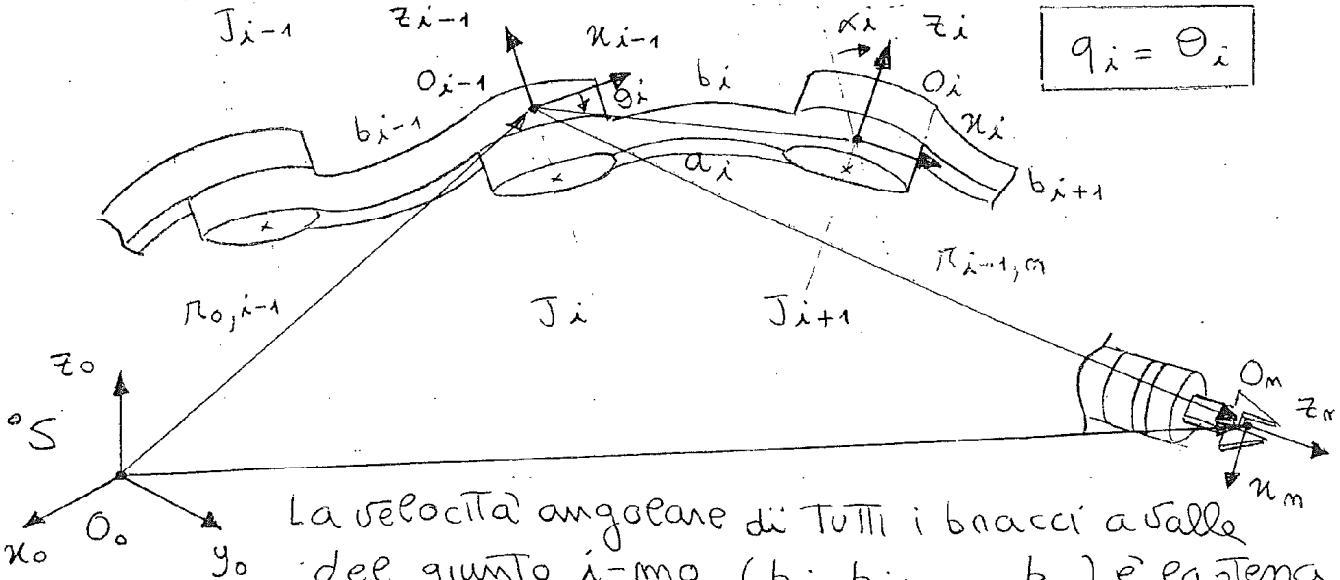
(i-ma colonna)

$J_{P_i}(q)$: contributo del giunto i-mo alla velocità di traslazione di ${}^mS$.

$J_{\omega_i}(q)$: contributo del giunto i-mo alla velocità di rotazione di ${}^mS$.

Nel caso in cui i sia un giunto a cerniera o rotoidale (bloccando tutti gli altri giunti, cioè isolando il contributo di i) si ha:

$$q_i = \theta_i$$



La velocità angolare di tutti i bracci a valle del giunto i-mo $(b_i, b_{i+1}, \ldots, b_m)$ è nulla in quanto tutti i giunti sono bloccati tranne J_i:

$$\omega_{i-1,m} = \omega_{0,m} = z_{i-1}\,\dot{\theta}_i = z_{i-1}\,\dot{q}_i \quad (\text{perchè } q_i = \theta_i)$$

La velocità angolare di ${}^mS$ rispetto ad ${}^{i-1}S$ e di ${}^mS$ rispetto ad ${}^0S$ è nulla perchè ${}^{i-1}S$ ed ${}^0S$ sono solidali in quanto i giunti a monte di J_i sono anch'essi bloccati.

Visto che il contributo elementare alla ${}^0\omega_{0m}$ è $J_{\omega_i}(q)\,\dot{q}_i$ ed in questo caso è pari a ${}^0z_{i-1}\,\dot{q}_i \Rightarrow$

$\Rightarrow \boxed{J_{\omega_i}(q) = {}^0z_{i-1}}$ Lo stesso discorso si può fare per le velocità lineari: il braccio terminale si muove alla stessa velocità sia rispetto al riferimento ^0S sia rispetto ad $i-1S$ perché tali riferimenti sono solidali in quanto i giunti a monte di J_i sono tutti bloccati:

$$\dot{\pi}_{0,m} = \dot{\pi}_{i-1,m} = \omega_{i-1,m} \times \pi_{i-1,m} = \omega_{0,m} \times \pi_{i-1,m} \Rightarrow$$

($\omega_{i-1,m} = \omega_{0,m}$)

Esprimiamo questa identità vettoriale in coordinate ^0S

$$\Rightarrow {}^0\dot{\pi}_{0,m} = \left[{}^0\omega_{0,m} \times\right] {}^0\pi_{i-1,m} = \left[{}^0z_{i-1} \times\right] {}^0\pi_{i-1,m}\, \dot{q}_i$$

$${}^0\omega_{0,m} = {}^0z_{i-1}\dot{\vartheta}_i = {}^0z_{i-1}\dot{q}_i$$

Visto che il contributo elementare alla ${}^0\dot{\pi}_{0m}$ è $J_{P_i}(q)\dot{q}_i$ ed in questo caso è pari a $\left[{}^0z_{i-1} \times\right] {}^0\pi_{i-1,m}\dot{q}_i \Rightarrow$

$$\Rightarrow \boxed{J_{P_i}(q) = \left[{}^0z_{i-1} \times\right] {}^0\pi_{i-1,m}}$$ Quindi, nel caso in cui il giunto J_i sia di tipo rotoidale (o a cerniera), la colonna i-ma della matrice Jacobiana geometrica è formata dai seguenti due sottoblocchi:

$$\begin{cases} J_{P_i}(q) = \left[{}^0z_{i-1} \times\right] {}^0\pi_{i-1,m} \\ J_{\omega_i}(q) = {}^0z_{i-1} \end{cases} \quad \text{con} \quad J(q) = \left[\begin{array}{c|c|c} \cdots & \begin{matrix} J_{P_i}(q) \\ \hline J_{\omega_i}(q) \end{matrix} & \cdots \end{array}\right]$$

(i-ma colonna)

Nel caso in cui $q_i = d_i$ (giunto prismatico → è l'asse z_{i-1} è l'asse di traslazione del giunto J_i) ⇒ il contributo alla velocità angolare di tale giunto è nullo mentre il contributo alla traslazione è diretto secondo l'asse ${}^0z_{i-1}$:

$${}^0\dot{\pi}_{0m} = {}^0z_{i-1}\dot{d}_i = {}^0z_{i-1}\dot{q}_i = J_{P_i}(q_i)\dot{q}_i \Rightarrow J_{P_i}(q) = {}^0z_{i-1}$$

Quindi nel caso in cui il giunto J_i sia prismatico, la colonne i-ma della matrice Jacobiana geometrica è formata dai seguenti sottoblocchi:

$$\begin{cases} J_{P_i}(q) = {}^0z_{i-1} \\ J_{\omega_i}(q) = 0 \end{cases} \quad \text{con} \quad J(q) = \left[\begin{array}{c|c|c} \cdots & \begin{matrix} J_{P_i}(q) \\ \hline J_{\omega_i}(q) \end{matrix} & \cdots \end{array}\right]$$

(i-ma colonna)

Bisogna quindi calcolare le coordinate dei versori z_{i-1} ed $\pi_{i-1,m}$ espresse nel riferimento ^0S cioè $^0z_{i-1}$ ed $^0\pi_{i-1,m}$. Le coordinate $^0z_{i-1}$ sono facili da calcolare a partire da:

$$^0T_{i-1}(q) = \left[\begin{array}{c|c} ^0R_{i-1}(q) & ^0\pi_{0,i-1} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \quad \text{con } ^0R_{i-1}(q) = \left[^0x_{i-1}\ ^0y_{i-1}\ ^0z_{i-1}\right]$$

cioè si può ottenere prendendo l'ultima componente (colonna) della matrice di rotazione contenuta all'interno della matrice di rototraslazione. OSS: $J_{w_i}(q)$ è una funzione $J_{w_i}(q_1, q_2, \dots, q_{i-1})$ cioè sono solo i giunti a monte di J_i a concorrere per la determinazione dell'orientamento dell'asse z_{i-1}. Questo non è vero per $J_{P_i}(q)$ in quanto la velocità lineare dell'organo terminale varia a seconda dell'estensione del corpo rigido a valle del giunto J_i: cioè il vettore $\pi_{i-1,m}$ è più o meno lungo a seconda della configurazione di $q_{i+1}, \dots, q_m$. Il termine $^0\pi_{i-1,m}$ stabilisce nelle coordinate di ^0S la posizione dell'organo terminale rispetto ad $O_{i-1} \Rightarrow$ la velocità $^0\dot{\pi}_{i-1,m}$ è diversa a seconda che il braccio sia esteso o rannicchiato $\Rightarrow J_{P_i}(q)$ dipende dalla configurazione di tutti i giunti $J_1, \dots, J_m \Rightarrow$ $\Rightarrow q_1, \dots, q_m$.

$$^{i-1}T_m(q) = \left[\begin{array}{c|c} ^{i-1}R_m(q) & ^{i-1}\pi_{i-1,m} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \longrightarrow {}^{i-1}\pi_{i-1,m}(q)$$

nelle coordinate ^{i-1}S.

Per avere $\pi_{i-1,m}$ nelle coordinate ^0S devo moltiplicare per la matrice di rotazione:

$$^0\pi_{i-1,m}(q) = {}^0R_{i-1}(q)\ {}^{i-1}\pi_{i-1,m}(q)$$

Utilizziamo ancora le coordinate ^{i-1}S:

$$^{i-1}\dot{\pi}_{i-1,m} = {}^{i-1}\omega_{i-1,m} \times {}^{i-1}\pi_{i-1,m} = \left(\begin{pmatrix}0\\0\\1\end{pmatrix} \times {}^{i-1}\pi_{i-1,m}\right)\dot{q}_i$$

$$^{i-1}\omega_{i-1,m} = {}^{i-1}z_{i-1}\,\dot{\theta}_i = {}^{i-1}z_{i-1}\,\dot{q}_i$$

Asse z_{i-1} nelle coordinate ^{i-1}S

Si lavora ancora in ^{i-1}S perché come si può vedere è facile fare i prodotti vettoriali in quanto il vettore z_{i-1} nelle coordinate ^{i-1}S è espresso facilmente: $\begin{pmatrix}0\\0\\1\end{pmatrix} \times \begin{pmatrix}x\\y\\z\end{pmatrix} = \begin{pmatrix}y\\-x\\0\end{pmatrix}$

Una volta calcolato $^{i-1}\dot{\pi}_{i-1,m}$ si torna facilmente in ^0S: $^0\dot{\pi}_{i-1,m} = {}^0R_{i-1}(q)\ {}^{i-1}\dot{\pi}_{i-1,m}(q)$

Un altro possibile modo per calcolare il vettore $\pi_{i-1,n}$ è di considerare le due matrici di rototraslazione (o meglio le ultime colonne di tali matrici):

$$\left.\begin{matrix} {}^0T_n(q) \rightarrow {}^0\pi_{0,n} \\ {}^0T_{i-1}(q) \rightarrow {}^0\pi_{0,i-1} \end{matrix}\right\| \Rightarrow {}^0\pi_{i-1,n} = {}^0\pi_{0,n} - {}^0\pi_{0,i-1}$$

Quindi, le informazioni contenute nelle matrici cinematiche di rototraslazione sono sufficienti per costruire la matrice Jacobiana geometrica che esprime un legame lineare tra le velocità di giunto e quelle cartesiane. Lo Jacobiano algebrico è invece più difficile da calcolare. Il legame che intercorre tra Jacobiano algebrico (o analitico) e quello geometrico è:

$$J(q) = \begin{bmatrix} I & 0 \\ 0 & G(\tilde{\varphi}) \end{bmatrix} J_K(q)$$

Lo Jacobiano geometrico $\exists$ sempre, perché a partire da una certa configurazione q è sempre possibile calcolare $J_K(q)$ e quindi $J(q)$ mentre non sempre è possibile fare il ragionamento inverso. Apparentemente $J_K(q)$ sembrerebbe $\nexists$ per configurazioni di singolarità di rappresentazione derivanti dall'inversione della matrice $G(\varphi) \Rightarrow$ ci sono delle velocità di giunto che pongono dar luogo a delle velocità cartesiane che non sono rappresentabili mediante derivate degli angoli di Eulero. Ciò che avviene quando ci si avvicina ai punti di singolarità di rappresentazione è che $|G^{-1}| \rightarrow \infty$, ciò significa che che $|J_K| \rightarrow \infty$ e $|G| \rightarrow 0 \Rightarrow$ il loro prodotto $(\infty \cdot 0)$ continua ad essere finito $\Rightarrow J(q)$ è limitato e non ci sono pb nella rappresentazione dell'orientamento e della velocità di rotazione mediante velocità angolari.

$$G(\tilde{\varphi}) = \begin{bmatrix} 0 & c\varphi & s\varphi s\vartheta \\ 0 & s\varphi & -c\varphi s\vartheta \\ 1 & 0 & c\vartheta \end{bmatrix}$$

Nonostante delle colonne tendano ad allinearsi in prossimità dei punti di singolarità, $J(q)$ è sempre limitato perché la singolarità non è geometrica (derivante da limitazioni strutturali del robot), ma è una singolarità di rappresentazione $\Rightarrow J(q)$ non cade

non di rango perché il prodotto $0 \cdot \infty$ è cmq limitato. Si può capire meglio cosa sia una singolarità di rappresentazione, cambiando la matrice $G(\tilde{\varphi})$ con una matrice $S(\Theta)$ che lega la velocità angolare alle derivate degli angoli di Roll-Pitch-Yaw che esprimono la velocità di rotazione del corpo:

$$J(q) = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & S(\Theta) \end{array}\right] J_{K\pi}(q)$$

In questo caso infatti si avranno delle singolarità di rappresentazione in altri punti rispetto a quelle che si avevano precedentemente. Per avere delle singolarità vere, cioè strutturali $\Rightarrow$ si devono imporre velocità proibite dalla geometria del robot che provocano una caduta del rango delle matrice $J(q)$. Queste singolarità strutturali si sono $\perp$ dalla scelta del tipo di rappresentazione.

OSS: Se il fatto che per $|G| \to 0$ lo $J(q)$ rimanga cmq limitato $\Rightarrow |J_K(q)| \to \infty \Rightarrow$ lo Jacobiano algebrico "può esplodere" cioè avere termini $\to \infty$ in corrispondenza dei punti di singolarità di rappresentazione.

Considerando ora gli schemi visti per la risoluzione della cinematica inversa con lo Jacobiano trasposto oppure inverso (J_K^T oppure J_K^{-1}), potranno essere modificati in modo da ricavarsi tali Jacobiani direttamente dallo Jacobiano geometrico (invece di doverli derivare dalle eq. cinematiche):

$$J_K(q) = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & G^{-1}(\tilde{\varphi}) \end{array}\right] J(q) \quad \text{con} \quad \begin{bmatrix} {}^0r_{om} \\ \hline \tilde{\varphi} \end{bmatrix} = \begin{bmatrix} K_p(q) \\ \hline K_o(q) \end{bmatrix}$$

Lo schema a blocchi è lo stesso (non cambia nulla). In alternativa si può realizzare lo schema inserendo direttamente $J(q)$ cioè lo Jacobiano geometrico.

OSS: Una terza interpretazione dell'algoritmo dello Jacobiano inverso $J_K^{-1}(q)$ è quella di soluzione al pb del minimo di una funzione: $V(q) = ({}^0p_d - K(q))^T({}^0p_d - K(q)) \Rightarrow$ si cerca una legge di aggiornamento $\dot{q}$ tale da risolvere il pb del minimo con $\dot{q} = f(q, {}^0p_d)$. Derivando la $V(q)$ otteniamo la seguente espressione per $\dot{V}(q)$:

$$\dot{V}(q) = -2({}^0p_d - K(q))^T \frac{\partial K}{\partial q} \dot{q} \Rightarrow$$

Questa espressione ci ha suggerito 2 leggi di aggion

mamento possibile: $\begin{cases} \dot{q} = \beta \left(\frac{\partial K}{\partial q}\right)^{-1} ({}^{\circ}p_J - K(q)) \\ \dot{q} = \beta \left(\frac{\partial K}{\partial q}\right)^{-1} ({}^{\circ}p_J - K(q)) \end{cases}$ Le coefficiente $\beta > 0$ risolve pb di scalature della soluzione

Entrambe queste 2 leggi danno luogo ad una $\dot{V}(q) < 0$ definita negativa (o al massimo semidefinita positiva se ci sono cadute di rango) $\Rightarrow$ fanno convergere l'algoritmo all'ottimo. Il β che si aggiunge serve per scalare la velocità di movimento lungo la direzione trovata. Spesso al posto dello scalare $\beta > 0$, si mette una matrice simmetrica e definita positiva: $K = K^T > 0$: $\dot{q}(t) = K J_K^{-1}(q) ({}^{\circ}p_J - K(q))$

In tal caso la $\dot{V}(q)$ continua ad essere (semi) definita positiva:

$\boxed{J_K^T(q)} \longrightarrow \dot{V}(q) = -2 ({}^{\circ}p_J - K(q))^T \boxed{J_K(q) K J_K^T(q)} ({}^{\circ}p_J - K(q))$

Infatti se $K = K^T > 0$ e $J_K(q)$ è una matrice a rango pieno $\Rightarrow$ $\Rightarrow$ la matrice evidenziata è ancora simmetrica e (semi) definita positiva $\Rightarrow$ $\dot{V}(q)$ continua ad essere ≤ 0.

Una possibilità più generale per scrivere la $V(q)$ è la seguente

$V(q) = ({}^{\circ}p_J - K(q))^T K ({}^{\circ}p_J - K(q)) \Rightarrow$

$\Rightarrow \dot{V}(q) = -2 ({}^{\circ}p_J - K(q))^T K \left(\frac{\partial K}{\partial q}\right) \dot{q}$ che ci suggerisce 2 leggi di aggiornamento

$\begin{cases} \dot{q} = \left(\frac{\partial K}{\partial q}\right)^T K ({}^{\circ}p_J - K(q)) \\ \dot{q} = \left(\frac{\partial K}{\partial q}\right)^{-1} K ({}^{\circ}p_J - K(q)) \end{cases}$ corrispondenti rispettivamente all'uso dello Jacobiano Trasposto e di quello inverso.

$\boxed{J_K^T(q)} \longrightarrow \dot{V}(q) = -2 ({}^{\circ}p_J - K(q))^T \boxed{K J_K(q) J_K^T(q) K} ({}^{\circ}p_J - K(q))$

Ancora una volta la matrice è (semi) definita $> 0 \Rightarrow$ $\Rightarrow \dot{V}(q) \leq 0$. Inoltre questa scelta della $V(q)$ ci permette di avere una $\dot{V}(q) \leq 0$ anche nel caso in cui scegliamo la legge di aggiornamento che fa uso dello Jacobiano inverso:

$\boxed{J_K^{-1}(q)} \longrightarrow \dot{V}(q) = -2 ({}^{\circ}p_J - K(q))^T \boxed{K^2} ({}^{\circ}p_J - K(q))$

Anche nella $\dot{V}(q)$ relativa all'algoritmo dello Jacobiano inverso la matrice K^2 è (semi) definita positiva $\Rightarrow$ $\dot{V}(q) \leq 0$.

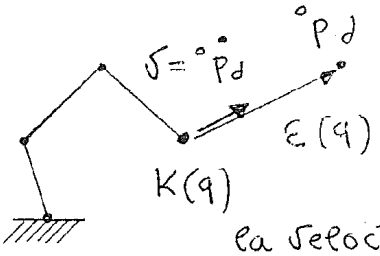
Quindi la terza interpretazione dell'algoritmo dello Jacobiano inverso ($J_K^{-1}(q)$) di cui si parla nella 55 nella

gacciata precedente è la seguente: data la legge di aggiornamento basata sullo Jacobiano algebrico inverso:

$$\dot{q} = J_K^{-1}(q)\, K\, ({}^{\circ}p_d - K(q))$$

se ${}^{\circ}p_d$ sta ad indicare la posizione desiderata espressa in coordinate cartesiane del sistema di riferimento ${}^{\circ}S$, mentre $K(q)$ è la posizione attuale (Trasformazione Cinematica delle variabili di giunto q) $\Rightarrow$ la differenza ${}^{\circ}p_d - K(q)$ è interpretabile come l'errore in una certa configurazione $\varepsilon(q)$:

$$\boxed{\varepsilon(q) = {}^{\circ}p_d - K(q)}$$



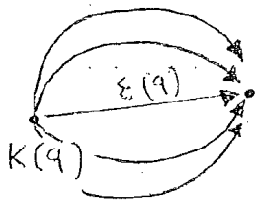
L'obbiettivo è quello di portare l'errore $\varepsilon(q)$ a zero imponendo una velocità dei giunti q che dia luogo ad una velocità cartesiana desiderata $v = {}^{\circ}\dot{p}_d$ combinazione dell'errore $\varepsilon(q)$, cioè tale che: ${}^{\circ}\dot{p}_d = \beta\varepsilon = \beta({}^{\circ}p_d - K(q))$. Se $J_K^{-1}(q)\,\exists$, la velocità di giunto che realizza tale velocità cartesiana è

$$\dot{q} = J_K^{-1}(q)\,{}^{\circ}\dot{p}_d = J_K^{-1}(q)\,\beta\,\varepsilon = \beta\, J_K^{-1}(q)\,({}^{\circ}p_d - K(q))$$

oppure con la K:

$$\dot{q} = J_K^{-1}(q)\, K\, ({}^{\circ}p_d - K(q))$$

OSS: Il fatto che al posto di $\beta > 0$ scalare ci sia una matrice K simmetrica e definita positiva non garantisce che la velocità cartesiana abbia la direzione dell'errore $\varepsilon(q)$, ma fa comunque in modo che la traiettoria sia prossima a quella rettilinea (Traiettorie in nero nel disegno a DX). Cioè per realizzare ${}^{\circ}p_d = K(q_d)$ non è strettamente necessario $\beta > 0$ scalare ma va bene $K = K^T > 0$. Se chiamo l'errore $\varepsilon(q) = {}^{\circ}p_d - K(q) \Rightarrow$



$$\Rightarrow \dot{\varepsilon}(q) = \cancel{{}^{\circ}\dot{p}_d}^{\,0} - \frac{\partial K}{\partial q}\dot{q} = -J_K(q)\,\dot{q} = -\cancel{J_K(q)}\,\cancel{J_K^{-1}(q)}\,K\varepsilon \Rightarrow$$

${}^{\circ}p_d = \text{cost.}$ perché la posizione desiderata è fissa (non una traiettoria)

$$\begin{cases} \dot{q}(t) = J_K^{-1}(q)\, K\varepsilon \\ \varepsilon(q) = {}^{\circ}p_d - K(q) \end{cases}$$

$$\Rightarrow \boxed{\dot{\varepsilon}(q) = -K\varepsilon(q)}$$

L'errore $\varepsilon(q)$ evolve come lo stato di un sistema lineare stazionario con costante di Tempo definita dagli autovalori della $K > 0 \Rightarrow -K < 0 \Rightarrow$ ha autovalori negativi $\Rightarrow$ la dinamica è asintoticamente stabile:

$$\boxed{\varepsilon(t) = e^{-Kt}\,\varepsilon(0)}$$

evoluzione asintoticamente stabile. Nonostante ci a dimostrare la stessa dinamica per $J_K^T(q) =$

$\varepsilon = {}^{\circ}p_J - K(q) \Rightarrow \dot{\varepsilon}(q) = -J_K(q)\dot{q}$ con $\dot{q} = J_K^{\dagger} K\varepsilon \Rightarrow$

$\Rightarrow \boxed{\dot{\varepsilon}(t) = -J_K(q) J_K^{\dagger}(q) K\varepsilon(q)}$ con $q = q(t)$.

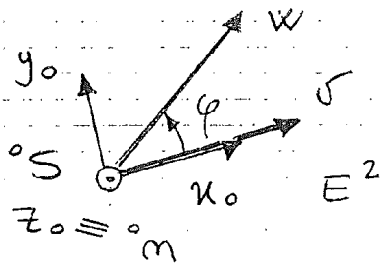
Sistema tempo variante

$\Rightarrow$ è analisi della convergenza non è semplice come quella fatta precedentemente in quanto la dinamica non lineare non viene cancellata $\Rightarrow$ è necessario effettuare l'analisi come criterio di Lyapunov.

OSS: La limitazione dell'algoritmo che sfrutta $J_K^{-1}(q)$ è che è ristretto ai robot non ridondanti (quadrati) e che pertanto è stato applicato solo allo Jacobiano analitico e non a quello geometrico.

OSS: I percorsi più svantaggiosi non sempre sono quelli più brevi si preferisce dare priorità a percorsi che privilegiano il movimento dei bracci più leggeri e tengono fermi il più possibile quelli più pesanti. Nella realizzazione di un percorso non si pensa mai ad una rappresentazione minima. Si utilizza lo Jacobiano geometrico per realizzare l'algoritmo dello Jacobiano inverso per la soluzione del Pb cinematico inverso. Per poter fare questo, bisogna tener presente la seguente:

OSS: Una matrice di rotazione ${}^{\circ}R_1$ è rappresentativa di una rotazione che per il Teorema di Eulero si può effettuare in un solo passo ruotando il corpo di un angolo di Eulero φ attorno all'asse di Eulero ${}^{\circ}m$ (espresso nelle coordinate di ${}^{\circ}S$):



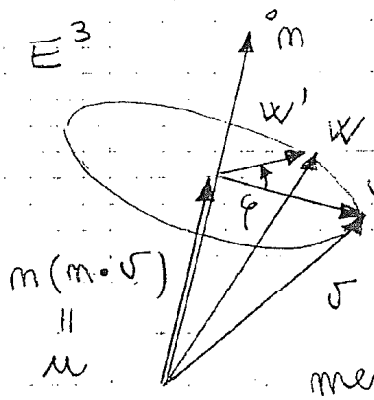
cioè risulta: ${}^{\circ}R_1 = R({}^{\circ}m, \varphi)$

asse di Eulero in ${}^{\circ}S$ — nel piano

angolo di Eulero

La rotazione di v si può esprimere in E^2

$w = R_z(\varphi)\, v = v\cos(\varphi) + (z_0^{m} \times v)\sin\varphi$



Andiamo a ricavare ora l'operatore che opera una rotazione attorno all'asse di Eulero m del vettore v di un angolo φ (angolo di Eulero). La componente lungo m non viene ruotata:

$u = m(m \cdot v)$ e $w = R(m, \varphi)\, v$

mentre la componente $\perp$ ad m viene ruotata:

$v' = v - u = v - n(n\cdot v)$ mentre il vettore ruotato w:

$w = u + w' = n(n\cdot v) + v'\cos\varphi + (n\times v')\sin(\varphi) \Rightarrow$

Infatti si può dimostrare:  $= (n\times v)$

$(n\times v') = n\times(v - n(n\cdot v)) = n\times v - \cancel{(n\times n)}^{0}(n\cdot v) = n\times v$

$\Rightarrow w = n(n\cdot v) + (v - n(n\cdot v))\cos\varphi + (n\times v)\sin\varphi = R(n,\varphi)v$

Quella scritta è una relazione in $\mathbb{E}^3$ tra vettori: se esprimiamo tutto nelle coordinate di $^{\circ}S \Rightarrow$ lavoreremo in $\mathbb{R}^3$:

$^{\circ}w = \underbrace{^{\circ}n\,^{\circ}n^{T}}\,^{\circ}v + I\cos\varphi\,^{\circ}v - \cos\varphi\,\underbrace{^{\circ}n\,^{\circ}n^{T}}\,^{\circ}v + \sin\varphi\,\underbrace{[^{\circ}n\times]}\,^{\circ}v \Rightarrow$

vettore prodotto scalare — prodotto vettoriale

$\Rightarrow {}^{\circ}w = (\underbrace{^{\circ}n\,^{\circ}n^{T}} + (I - {}^{\circ}n\,{}^{\circ}n^{T})\cos\varphi + [{}^{\circ}n\times]\sin\varphi)\,{}^{\circ}v \Rightarrow$

↳ matrice: colonna x riga → rango unitario.

$\Rightarrow {}^{\circ}w = {}^{\circ}R_1\,{}^{\circ}v \Rightarrow$ possiamo così ricavare la matrice rappresentativa dell'operatore di rotazione (o del cambio di coordinate):

$$\boxed{{}^{\circ}R_1({}^{\circ}n,\varphi) = \left({}^{\circ}n\,{}^{\circ}n^{T} + (I - {}^{\circ}n\,{}^{\circ}n^{T})\cos\varphi + [{}^{\circ}n\times]\sin\varphi\right)}$$

Se $\varphi = 0 \Rightarrow {}^{\circ}R_1({}^{\circ}n,\varphi) = I \Rightarrow$ non c'è trasformazione.

RIC:

$$^{\circ}n = \begin{bmatrix} ^{\circ}n_x \\ ^{\circ}n_y \\ ^{\circ}n_z \end{bmatrix};\quad {}^{\circ}n\,{}^{\circ}n^{T} = \begin{bmatrix} ^{\circ}n_x^2 & ^{\circ}n_x\,^{\circ}n_y & ^{\circ}n_x\,^{\circ}n_z \\ ^{\circ}n_x\,^{\circ}n_y & ^{\circ}n_y^2 & ^{\circ}n_y\,^{\circ}n_z \\ ^{\circ}n_x\,^{\circ}n_z & ^{\circ}n_y\,^{\circ}n_z & ^{\circ}n_z^2 \end{bmatrix};\quad [^{\circ}n\times] = \begin{bmatrix} 0 & -^{\circ}n_z & ^{\circ}n_y \\ ^{\circ}n_z & 0 & -^{\circ}n_x \\ -^{\circ}n_y & ^{\circ}n_x & 0 \end{bmatrix}$$

$$^{\circ}R_1 = \begin{bmatrix} c\varphi + n_x^2(1-c\varphi) & ^{\circ}n_x\,^{\circ}n_y(1-c\varphi) - ^{\circ}n_z s\varphi & ^{\circ}n_x\,^{\circ}n_z(1-c\varphi) + ^{\circ}n_y s\varphi \\ ^{\circ}n_x\,^{\circ}n_y(1-c\varphi) + ^{\circ}n_z s\varphi & c\varphi + ^{\circ}n_y^2(1-c\varphi) & ^{\circ}n_y\,^{\circ}n_z(1-c\varphi) - ^{\circ}n_x \\ ^{\circ}n_x\,^{\circ}n_z(1-c\varphi) - ^{\circ}n_y s\varphi & ^{\circ}n_y\,^{\circ}n_z(1-c\varphi) + ^{\circ}n_x s\varphi & c\varphi + ^{\circ}n_z^2(1-c\varphi) \end{bmatrix}$$

La cosa interessante è risolvere il pb inverso, cioè data la matrice rappresentativa della rotazione, trovare l'asse e l'angolo di Eulero. ∃ un metodo che prescinde da questa rappresentazione. Data infatti una ${}^{\circ}R_1$, grazie alle proprietà di ortonormalità di cui gode risulta avere:

$\lambda_1 = 1,\ \lambda_2 = \lambda_3^{*} = e^{-j\varphi}$ con $e^{j\varphi} = \cos\varphi + j\sin\varphi$

$|{}^{\circ}R_1| = \lambda_1\lambda_2\lambda_3 = 1 = 1\cdot e^{j\varphi}e^{-j\varphi}$ con $\varphi < \pi$

L'asse di Eulero è l'autovettore relativo all'autovalore $\lambda_1 = 1$

di modulo unitario uguale a quello di cui ruota e quello unico vettore che una volta trasformato rimane invariato:

${}^0\hat{n} = {}^0R_1\,{}^0\hat{n}$ (cioè quello per cui $A\mu = \lambda\mu$ con $\lambda = 1$ ed μ autovettore relativo all'autovalore $\lambda = 1$ della matrice A).

OSS: La stessa notazione può essere rappresentata da due angoli con verso opposto attorno ad uno stesso asse, ascendo dall'orientamento dell'asse stesso, cioè una notazione di un angolo φ attorno ad un asse ${}^0\hat{n}$ è equivalente ad una notazione di un angolo $-\varphi$ attorno all'asse $-{}^0\hat{n}$:

$\boxed{{}^0\hat{n}, \varphi \rightarrow -{}^0\hat{n}, -\varphi}$ questo è il motivo per cui possiamo sempre considerare un angolo $\varphi \in [0, \pi)$ che può essere calcolato a partire dagli autovalori λ_2 e λ_3. Oppure si può considerare come ${}^0\hat{n}$ la parte antisimmetrica di ${}^0R_1$ cioè:

$$S = {}^0R_1 - {}^0R_1^T = \left([{}^0\hat{n}\times] - [{}^0\hat{n}\times]^T\right)\sin\varphi =$$

$$= 2[{}^0\hat{n}\times]\sin\varphi \Rightarrow$$

→ Applicando la relazione della facciata precedente

$$\Rightarrow [{}^0\hat{n}\times] = \frac{({}^0R_1 - {}^0R_1^T)}{2\sin\varphi}$$

Ma ho comunque bisogno di conoscere il valore di $\sin\varphi$. Tenendo conto:

OSS: Data una matrice A come polinomio caratteristico:

$\det(\lambda I - A) = \lambda^n + a_{n-1}\lambda^{n-1} + \dots + a_1\lambda + a_0 = (\lambda-\lambda_1)(\lambda-\lambda_2)\dots(\lambda-\lambda_n) \Rightarrow$ allora risulta che:

$$\mathrm{Tr}(A) = \lambda_1 + \lambda_2 + \dots + \lambda_n = a_{11} + a_{22} + \dots + a_{nn} = -a_{n-1}$$

$$\det(A) = a_0$$

Quindi data la matrice ${}^0R_1 \rightarrow \mathrm{Tr}({}^0R_1) = r_{11} + r_{22} + r_{33}$

Sappiamo inoltre che:

$$\mathrm{Tr}({}^0R_1) = 1 + e^{j\varphi} + e^{-j\varphi} = 1 + \cos\varphi + j\sin\varphi + \cos\varphi - j\sin\varphi$$

$$= 1 + 2\cos\varphi \Rightarrow \cos(\varphi) = \frac{\mathrm{Tr}({}^0R_1) - 1}{2} \quad \text{con } \varphi \in (0, \pi)$$

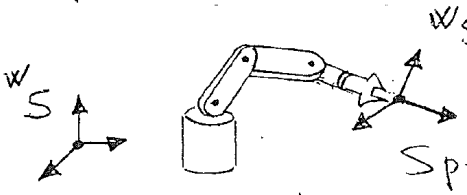
$$\Rightarrow \sin(\varphi) = +\sqrt{1 - \cos^2(\varphi)}$$

→ Prendo la soluzione positiva perché consideriamo φ tra 0 e π.

Si ricordi inoltre che: ${}^0\hat{n}_x^2 + {}^0\hat{n}_y^2 + {}^0\hat{n}_z^2 = 1$.

Completiamo la cinematica inversa costruendo la Jacobiano geometrico, angoli ed assi di Eulero

a partire dalla conoscenza dei parametri geometrici del robot. Infatti conoscendo i parametri di Denavit-Hartenberg, si può calcolare la matrice di rototraslazione:



$$^{W}T_{m}(q) = \left[\begin{array}{c|c} ^{W}R_{m} & ^{W}p_{Om} \\ \hline 0\;0\;0 & 1 \end{array}\right]$$

Spesso la matrice di rototraslazione completa comprende anche i riferimenti ^{W}S (world) ed ^{T}S (tool):

$$^{W}T_{T}(q) = {}^{W}T_{0}\,{}^{0}T_{m}(q)\,{}^{m}T_{T} = \left[\begin{array}{c|c} ^{W}R_{T}(q) & ^{W}p_{W,T} \\ \hline 0\;0\;0 & 1 \end{array}\right]$$

Con ^{W}S solidale ad ^{0}S ed ^{T}S solidale ad ^{m}S $\Rightarrow$ le matrici di rototraslazione $^{W}T_{0}$ e $^{m}T_{T}$ sono costanti. La matrice Jacobiana geometrica che esprime il legame tra $\dot{q}$ ed il vettore della velocità generalizzata che finora è stato scritto:

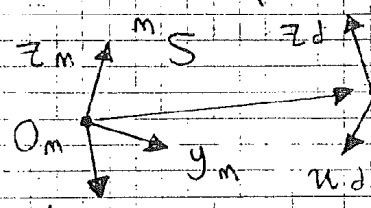
$$\underline{\underline{^{0}v_{0,m}}} = \left[\begin{array}{c} ^{0}\dot{p}_{Om} \\ \hline ^{0}\omega_{Om} \end{array}\right] = J(q)\dot{q} \quad (\dot{q} \rightarrow {}^{0}v_{Om})$$

nel caso in cui consideriamo anche i sistemi di riferimento ^{W}S ed ^{T}S dovrebbe essere riscritto considerando la nuova velocità generalizzata. Tuttavia noi continueremo a considerare la $^{0}T_{m}(q)$ perché considerando anche ^{W}S ed ^{T}S non aggiungiamo niente di nuovo alla trattazione. Inizialmente abbiamo visto l'algoritmo per la risoluzione del pb cinematico inverso che fa uso dello Jacobiano algebrico:

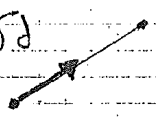
$$\left[\begin{array}{c} ^{0}\dot{p}_{Om} \\ \hline \dot{\varphi} \end{array}\right] = \left[\begin{array}{c} K_{p}(q) \\ \hline K_{o}(q) \end{array}\right] = \frac{\partial J_{K}}{\partial q}$$

Bisogna notare però che lo Jacobiano geometrico $J(q)$ è più semplice da calcolare perché non si devono calcolare esplicitamente le funzioni, ma si ricorre ad un algoritmo numerico (senza dover operare delle derivate). Studiamo allora il modo di realizzare un algoritmo per la cinematica inversa che faccia uso dello Jacobiano geometrico (direttamente). Sia n = # assi del robot (DOF: degrees of freedom, gradi di libertà, dimensione dello spazio operativo) e sia m il numero di gradi di libertà relativi al compito da svolgere (se devo descrivere un compito in E^3 con tutti gli orientamenti possibili $\rightarrow m = 6$). Si consideri prima il caso di robot NON RIDONDANTI cioè $m = n$ $\Rightarrow$ In tal caso se voglio ottenere una certa velocità generalizzata desiderata $^{0}v_{d}$, le velocità di giunto che

dobbiamo imporre sono: $\dot{q}_d = J_K^{-1}(q)\, v_d$



Per portare e a tenna solidale con ea mano del robot (mS) a coincidere con quella desiderata ho ∞ modi possibili. La tecnica che fa uso dello Jacobiano algebrico inverso $J_K^{-1}(q)$ può essere interpretata nel seguente modo: se è errore di posizionamento impone una velocità cartesiana proporzionale alla direzione dell'errore stesso, cioè:



$$^\circ v_d = \beta\,(^\circ p_d - K(q))$$

Errore di posizione nelle coordinate cartesiane (traiettoria minima)

Per una velocità di giunto da imporre è: $\dot{q} = \beta\, J_K^{-1}(q)\,(^\circ p_d - K(q))$

Consideriamo in questo modo una traslazione lungo una traiettoria lineare (minima) da $K(q)$ a $^\circ p_d$. In modo analogo, ricordando il teorema di Eulero, si può procedere in questo modo anche per gli orientamenti cioè si passa dall'orientamento corrente a quello desiderato mediante una rotazione dell'angolo di Eulero attorno all'asse di Eulero $\Rightarrow$ per poter fare questo è necessario applicare l'algoritmo visto precedentemente che riassumiamo di seguito:

data una matrice di rotazione $^\circ R_1$, ora rappresentata in termini di angolo ed asse di Eulero: $^\circ R_1 = R = R(m, \varphi)$. La traccia della matrice R è:

$$T_r(R) = 1 + e^{J\varphi} + e^{-J\varphi} = 1 + 2\cos\varphi \Rightarrow \boxed{\cos(\varphi) = \frac{T_r(R) - 1}{2}}$$

$$e^{J\varphi} + e^{-J\varphi} = \cos\varphi + J\sin\varphi + \cos\varphi - J\sin\varphi = 2\cos\varphi$$

OSS: La traccia di una matrice R è invariante rispetto a trasformazioni di coordinate cioè: $T_r(R) = T_r(TRT^{-1})$

Inoltre è stato visto che:

$$\text{Se } R = {}^\circ R = R(^\circ m, \varphi) \Rightarrow [^\circ m\times]\sin\varphi = \frac{1}{2}\,(^\circ R_1 - {}^\circ R_1^T)$$

Parte antisimmetrica della matrice di rotazione $^\circ R_1$

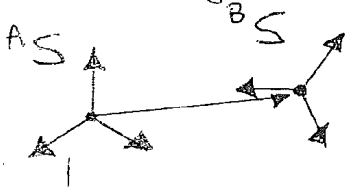
Il $\sin\varphi$ si calcola a partire dal $\cos\varphi$ che è stato già calcolato precedentemente, prendendo per convenzione

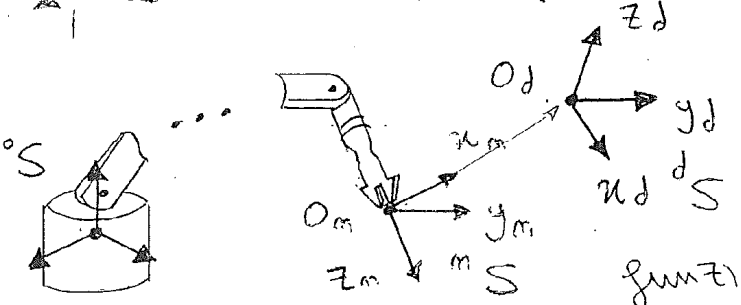
la soluzione positiva:

$$\cos\varphi = \frac{Tr(R)-1}{2} \rightarrow \sin\varphi = \sqrt{1-\cos^2\varphi} \rightarrow [\hat{m}\times] = \frac{1}{2}\frac{{}^0R_1 - {}^0R_1^T}{\sin\varphi}$$

OSS: Se $\varphi = 0 \Rightarrow R = I \Rightarrow \nexists$ notazione

OSS: La convenzione di prendere sempre l'angolo $\varphi > 0$, deriva dal fatto che $\nexists$ ambiguità tra angoli positivo e negativo perchè $R(\hat{m}, \varphi) = R(-\hat{m}, \varphi)$. Se conosco le coordinate dell'asse di Eulero $\hat{m}$ e l'angolo di Eulero φ, posso imporre una velocità angolare ω diretta come $\hat{m}$ e con verso pari a quello di φ per portare ${}^mS$ su ${}^TS$. Se ho due terni di riferimento ${}^AS$ ed ${}^BS$, l'algoritmo visto ci fornisce allora la rotazione minima, per portare ${}^AS$ a coincidere con ${}^BS$, nelle coordinate di ${}^AS$.





Traslando il discorso sulla mano del robot, i dati che ci hanno a disposizione sono l'orientamento della mano in funzione di q nelle coordinate di ${}^0S$ cioè la matrice di rotazione ${}^0R_m(q)$ e l'orientamento desiderato ${}^0R_d$ nelle coordinate di ${}^0S$. Il legame tra queste due matrici è la composizione:

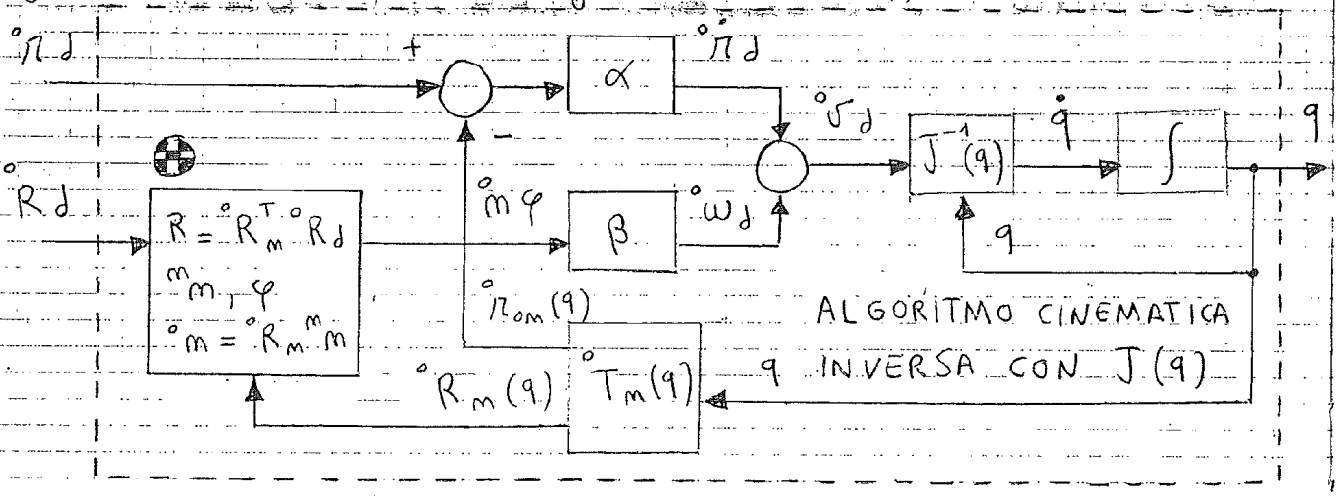
$$\boxed{{}^0R_d = {}^0R_m(q)\,{}^mR_d}$$

dove è messa in evidenza la semplificazione formale tra apice m e pedice m. La matrice ${}^mR_d$ corrisponde alla matrice ${}^AR_B$ dell'esempio precedente e serve ad esprimere le coordinate di ${}^BS$ (${}^dS$) nel riferimento ${}^TS$ (${}^mS$). Dalla relazione precedente si ricava:

$$\boxed{{}^mR_d = {}^0R_m^T(q)\,{}^0R_d}$$

Applicando l'algoritmo, riusciamo quindi a calcolare l'asse di Eulero espresso nelle coordinate ${}^mS \Rightarrow$ cioè il versore ${}^m\hat{m}$. Se voglio conoscerlo nelle coordinate ${}^0S \Rightarrow {}^0\hat{m} = {}^0R_m\,{}^m\hat{m}$. Mediante la formula ${}^mR_d = {}^0R_m^T(q)\,{}^0R_d$ effettuiamo la differenza tra due orientamenti: se lavorassimo con gli angoli di Eulero per rappresentare l'orientamento di differenza, avremmo dovuto effettuare la differenza tra gli angoli. Lavorando con le matrici di rotazione, l'orientamento di differenza si rappresenta mediante opportune composizioni

Quindi d'algoritmo allemiamo "in, φ emi nambe espressi nelle coordinate locali ${}^{m}S$ (cioè ${}^{A}S$). Nella rappresentazione della traslazione di gerenza, si fa la differenza tra due vettori rappresentativi di determinate posizioni. Il vettore velocità desiderata ${}^{\circ}\dot{\pi}_d$ che realizza il moto verso la posizione desiderata, ha direzione ${}^{\circ}\pi_d - {}^{\circ}\pi_{om}$ pari a quella dell'errore di posizionamento. Possiamo adesso introdurre l'algoritmo di cinematica inversa che fa uso dello Jacobiano geometrico $J(q)$ direttamente:



OSS: ${}^{\circ}v_d = \begin{pmatrix} {}^{\circ}\dot{\pi}_d \\ {}^{\circ}\omega_d \end{pmatrix}$ L'errore di orientamento ${}^{\circ}m\varphi$, spesso viene indicato anche con ${}^{\circ}m$ o $m\varphi$ e si ricava dall'algoritmo che nello schema, a blocchi è contrassegnato da ⊕. L'errore di posizione invece è pari a ${}^{\circ}\pi_d - {}^{\circ}\pi_{om}(q)$ e moltiplicato per lo scalare α dà la velocità lineare desiderata ${}^{\circ}\dot{\pi}_d$. Possiamo quindi dire che la velocità generalizzata ${}^{\circ}v_d$ desiderata è proporzionale alla direzione tale di ridurre l'errore (ci si muove nella direzione in cui l'errore si riduce). L'algoritmo si blocca quando l'errore è zero cioè ${}^{\circ}\dot{\pi}_d = 0$ ed ${}^{\circ}R_m^T(q)\,{}^{\circ}R_d = I \Rightarrow$
$\Rightarrow {}^{m}R_d = I \Rightarrow \varphi = 0$. La convergenza è esponenziale e dipende da α, β (se usiamo lo Jacobiano inverso). Se utilizziamo invece lo Jacobiano Trasposto ($J^T(q)$) la convergenza non è più esponenziale. Questo è uno dei modi di pensare nello spazio degli orientamenti. Ci sono altri possibili modi di rappresentare gli orientamenti, ad esempio con i quaternioni, per i quali ci sono delle leggi di composizione

analoghe per il calcolo dell'errore di orientamento. Se adesso consideriamo un robot RIDONDANTE (cioè $m > m$ ed il numero di assi del robot è maggiore del numero di gradi di libertà necessari per specificare il compito che il robot deve eseguire), posso ancora applicare l'algoritmo visto a SX?

OSS 1: Se ho $Ax = b$ con A quadrata e invertibile, per calcolare x non conviene invertire A e calcolare poi x con $x = A^{-1}b$ perché è altamente inefficiente. Ha senso operare in questo modo se devo risolvere un gran numero di questi problemi (tutti del Tipo $Ax = b$) cioè se devo risolvere il pb $x_K = A_K^{-1} b_K$ con $K = 1, 2, \dots, m$ ed m grande. Se però devo risolverne uno solo non è efficiente calcolare A^{-1} ed è preferibile ricorrere ad esempio alla diagonalizzazione.

OSS 2: Se il robot è NON ridondante, data una certa configurazione q, $\exists$ una soluzione unica al pb cinematico inverso e l'algoritmo convergerà a tale soluzione certamente.

Quindi il pb $\dot{v}_d = J(q)\dot{q}$ con $J(q)$ quadrato, ammette un unica soluzione $\dot{q}$.

Se il robot è ridondante $\Rightarrow m > m$ e $J(q)$ è una matrice rettangolare bassa (ha più colonne che righe) $\Rightarrow$ il pb da risolvere è allora $\dot{v}_d = [\dots\dots]\dot{q} \Rightarrow$ è un sistema di m equazioni in m incognite (ho più incognite che equazioni) $\Rightarrow$ ammette ∞^{m-m} soluzioni possibili. Una possibile soluzione è quella dell'equazione omogenea associata che può essere sommata alla soluzione della eq. non omogenea. Chiamando $\tilde{q}$ la soluz. dell'eq. omogenea e $\dot{q}^*$ quella dell'eq. non omogenea risulta che:

$$\dot{q}^* \rightarrow \dot{v}_d = J(q)\dot{q}^*$$
$$\dot{q}^* + \tilde{\dot{q}} \rightarrow \dot{v}_d = J(q)(\dot{q}^* + \tilde{\dot{q}})$$

con $\tilde{\dot{q}} \in \mathcal{N}(J(q))$ cioè $\boxed{J(q)\tilde{\dot{q}} = 0}$

con $\tilde{\dot{q}}$ indichiamo allora gli automovimenti del robot, cioè tutte quelle velocità di giunto che lasciano inalterata la posa della mano in quanto producono una velocità generalizzata nulla ($\dot{v}_d = 0$). Risulta inoltre che:

$$\dim(\mathcal{N}(J(q))) = n - \rho(J(q)) \geq n - m$$

OSS: L'uguale vale in configurazioni non singolari del

robot. Nel caso in cui si è in una configurazione singolare, il rango di $J(q)$ non è più m (si ha una caduta di rango ed $m - \rho(J) \leq m - m$). Nel caso di robot ridondanti sorge il pb di scegliere una tra le tante traiettorie possibili (possibilmente la più comveniente). La soluzione che troveremo risolverà il pb di cinematica inversa differenziale $\dot{q} = {}^{\circ}v_J \rightarrow \dot{q}$ per il robot non ridondante e dovrà essere inserita nell'algoritmo per la soluzione della cinematica inversa $({}^{\circ}r_J, {}^{\circ}R_J \rightarrow q)$ rappresentato nello schema a blocchi di 2 facciate prima, al posto della $J^{-1}(q)$. Per trovare la soluzione più comveniente tra le ∞ possibili, si può pensare di minimizzare un indice che dipende dalla velocità (per una data configurazione q fissa): $\gamma_q(\dot{q}) \geq 0$. Il pb diventa allora di trovare la soluzione della ${}^{\circ}v_J = J(q)\dot{q}$ tale da minimizzare l'indice definito $\gamma_q(\dot{q})$ cioè:

$$\dot{q}^* = \min_{{}^{\circ}v_J = J(q)\dot{q}} \gamma_q(\dot{q})$$

pb di minimizzazione o di minimo vincolato → ci servirà per risolvere il pb di cinematica differenziale inversa nel caso di robot ridondanti, data la configurazione q.

OSS: $\gamma_q(\dot{q}) = \begin{cases} \|\dot{q}\|^2 \rightarrow \text{Min. norma della velocità} \\ \|\dot{q} - \dot{q}_0\|^2 \rightarrow \dot{q}_0 \text{ è una velocità di giunto che mi piacerebbe il robot avesse.} \end{cases}$

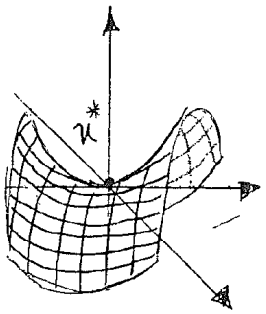
Questo è utile quando si vuole massimizzare un indice di manipolabilità $\beta(q) = \det(J(q)J^T(q)) \rightarrow$ infatti se il determinante di $J(q)J^T(q)$ è elevato sono tranquillo perché vuol dire che siamo lontani dai punti di singolarità, dove il determinante è nullo. È bene portare il robot nelle condizioni di massima manipolabilità, muovendomi lungo la direzione di massima salita della funzione obiettivo: $\dot{q}_0 = \alpha \frac{\partial \beta(q)}{\partial q} \rightarrow$ velocità obiettivo

Quindi adottando una $\gamma_q(q) = \|\dot{q} - \dot{q}_0\|$ ho come obiettivo primario la realizzazione del vincolo: ${}^{\circ}v_J = J(q)\dot{q}$ e come obiettivo secondario quello di massimizzare la manipolabilità minimizzando $\gamma_q(q) = \|\dot{q} - \dot{q}_0\|$ in modo che $\dot{q} \rightarrow \dot{q}_0$.

Nel caso di minimo non vincolato, si trova il punto stazionario x^* risolvendo, in forma chiuso o mediante un algoritmo numerico, il seguente sistema di equazioni:

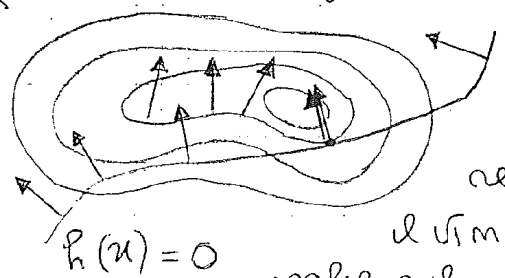
$$\left.\frac{\partial V}{\partial x}\right|_{x^*}^{T} = 0 \rightarrow x^*$$

Affinché il punto stazionario calcolato sia un punto di minimo, deve soddisfare la condizione $\left.\frac{\partial^2 V}{\partial x^2}\right|_{x^*} > 0$ $\Rightarrow$ cioè concavità dell'Hessiana.



Se così non fosse il punto stazionario x^* potrebbe anche essere un punto di flesso come indicato a SX. (Stazionarietà + concavità Hessiana $\Rightarrow$ punto di minimo). Nel caso in cui si voglia risolvere un pb di minimo vincolato la situazione in $\mathbb{R}^2$ è la seguente: se $x \in \mathbb{R}^2$, non

$$\begin{cases} \min V(x) \\ h(x) = 0 \end{cases}$$

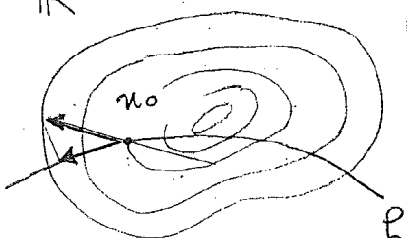


tutti i punti di $\mathbb{R}^2$ sono ammissibili: sol quelli appartenenti al luogo geometrico determinato da $h(x) = 0$ (una retta, una circonferenza ...) soddisfano il vincolo ed è tra questi punti che bisogna scegliere il minimo. Graficamente si intuisce che bisogna considerare la curva di livello tangente al vincolo $\Rightarrow$ la più bassa (sotto Hp di vincoli e curve di livello "lisce" cioè che ammettono derivate). Si vede così che il gradiente del vincolo $\partial h / \partial x$ (considerando la sola curva di livello a livello del mare cioè $h(x) = 0$) deve avere la stessa direzione del gradiente di $V(x)$ cioè di $\partial V(x)/\partial x$ $\Rightarrow$ si ha il minimo per:

$$\frac{\partial h}{\partial x} \;/\!/\; \frac{\partial V}{\partial x}$$

CONDIZIONE DI MINIMO VINCOLATO IN $\mathbb{R}^2$

OSS: In assenza di vincoli si avrebbe $\frac{\partial V}{\partial x}^T = 0$



- GRADIENTE
- PROIEZIONE DEL GRADIENTE LUNGO $h(x) = 0$

Algoritmicamente a partire da un punto x_0 si controlla se $\partial h/\partial x \;/\!/\; \partial V/\partial x$, ed in caso la condizione non è verificata ci si sposta lungo la direzione dell'antigradiente della curva di livello, proiettato nella direzione del vincolo cioè lungo la direzione della curva di livello a livello zero di $h(x)$.

In questo modo si passa su curve di livello più piccole, rimanendo comunque sul simplesso. Se voglio proiettare un vettore v lungo la direzione di un vettore m:

$\Pi(v|m) = (v \cdot m)\, m$, se m non è un versore $\Pi(v|m) = \dfrac{(v \cdot m)\, m}{(m \cdot m)}$

L'algoritmo di ricerca del minimo si ferma quando il gradiente $\partial V/\partial x$ è ortogonale al simplesso, in tal caso risulta nulla la proiezione (ed è verificata la condizione di minimo in quanto anche $\partial h/\partial x$ è perpendicolare al simplesso per cui $\Rightarrow \partial V/\partial x \parallel \partial h/\partial x$). Nel caso di $\mathbb{R}^m$ non si può ricorrere al grafico ma si può generalizzare il risultato ottenuto in $\mathbb{R}^2$ ricorrendo alla condizione:

$$\left(\frac{\partial V}{\partial x}\right)^T_{x^*} \in R\left(\left(\frac{\partial h}{\partial x}\right)^T_{x^*}\right)$$

è appartenenza al range di $\partial h/\partial x$ da parte di $\partial V/\partial x$ è una generalizzazione della condizione di parallelismo vista in $\mathbb{R}^2$ che matematicamente si esplicita dicendo che deve $\exists$ un vettore di moltiplicatori $\lambda \in \mathbb{R}^s$ tale che:

$$\left(\frac{\partial V}{\partial x}\right)^T_{x^*} = \left(\frac{\partial h}{\partial x}\right)^T_{x^*} \lambda \Rightarrow \begin{pmatrix} \frac{\partial V}{\partial x_1} \\ \vdots \\ \frac{\partial V}{\partial x_m} \end{pmatrix} = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_s}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_1}{\partial x_m} & \cdots & \frac{\partial h_s}{\partial x_m} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_s \end{pmatrix}$$

con $h(x) = \begin{pmatrix} h_1(x) \\ \vdots \\ h_s(x) \end{pmatrix}$

Infatti $h(x) = 0 \rightarrow$ simplesso vettoriale $\Rightarrow$ un insieme di s simplessi scalari

La funzione Lagrangiana nell'ambito della Teoria dell'ottimizzazione si scrive nel seguente modo:

$$\mathcal{L}(x, \lambda) = V(x) + \lambda^T h(x) = V(x) + h^T(x)\lambda \quad \text{con } x \in \mathbb{R}^m \rightarrow$$

$\rightarrow$ variabile del Pb originario e $\lambda \in \mathbb{R}^s$. La Teoria ci dice che il minimo vincolato si ha in corrispondenza della coppia (x^*, λ^*) tale da essere un punto stazionario per $\mathcal{L}$:

$$x^*, \lambda^* : \quad \frac{\partial \mathcal{L}}{\partial x}\Big|_{x^*,\lambda^*} = 0 \qquad \frac{\partial \mathcal{L}}{\partial \lambda}\Big|_{x^*,\lambda^*} = 0$$

CONDIZIONI DI STAZIONARIETÀ PER LA LAGRANGIANA.

Queste due condizioni possono essere interpretate così:

ndo dalla seconda: $\left.\frac{\partial \mathcal{L}}{\partial \lambda}\right|_{x^*} = h^T(x^*) = 0 \Rightarrow$ Aeè ottimo il vincolo deve essere soddisfatto

mentre dalla prima:

$$\left.\frac{\partial \mathcal{L}}{\partial x}\right|_{x^*,\lambda^*} = \left(\frac{\partial V}{\partial x} + \lambda^T \frac{\partial h}{\partial x}\right)\Big|_{x^*,\lambda^*} = 0 \Rightarrow \left.\frac{\partial V}{\partial x}\right|_{x^*}^T = -\left(\frac{\partial h}{\partial x}\right)\Big|_{x^*}^T \lambda^*$$

$$\Rightarrow \left(\frac{\partial V}{\partial x}\right)\Big|_{x^*}^T \in R\left(\left(\frac{\partial h}{\partial x}\right)\Big|_{x^*}^T\right) \Rightarrow$$ All'ottimo deve essere soddisf. la condizione di parallelismo generalizzata in $\mathbb{R}^m$ cioè: $\Pi\left(\left(\frac{\partial V}{\partial x}\right)^T \Big| \left(R\left(\frac{\partial h}{\partial x}\right)^T\right)^{\perp}_{x^*}\right)$

Applichiamo questa teoria di ottimizzazione alla robotica per risolvere il problema di minimizzare un indice $\gamma_q(\dot{q})$ che dipende da $\dot{q}$ per una data configurazione q (fissa) sotto il vincolo della velocità desiderata che bisogna realizzare $^{\circ}v_d = J(q)\dot{q} \Rightarrow$

$$\Rightarrow \boxed{\begin{matrix} \min \\ {}^{\circ}v_d = J(q)\dot{q} \end{matrix} \quad \gamma_q(\dot{q})}$$ con $^{\circ}v_d$ velocità generalizzata desiderata: $^{\circ}v_d = \begin{pmatrix} {}^{\circ}\dot{r} \\ {}^{\circ}\omega \end{pmatrix}$

È un pb di minimizzazione vincolata rispetto a $\dot{q}$. La lagrangiana in queste condizioni risulta nel seguente modo:

$$\mathcal{L}_q(\dot{q},\lambda) = \gamma_q(\dot{q}) + \lambda^T\left({}^{\circ}v_d - J(q)\dot{q}\right)$$

OSS: $\min_{h(x)=0} V(x) \rightarrow L(x,\lambda) = V(x) + \lambda^T h(x)$ in questo caso $h_q(\dot{q}) = {}^{\circ}v_d - J(q)\dot{q} = 0$

La condizione di stazionarietà è:

$$x^*,\lambda^*: \quad \left(\frac{\partial L}{\partial x}\right)_{x^*,\lambda^*} = 0$$

$$\left(\frac{\partial L}{\partial \lambda}\right)_{x^*,\lambda^*} = 0 \Rightarrow h(x) = 0$$

In questo caso le condizioni diventano le seguenti:

$$\left.\frac{\partial \mathcal{L}}{\partial \dot{q}}\right|_{q^*,\lambda^*} = \left.\frac{\partial \gamma_q(\dot{q})}{\partial \dot{q}}\right|_{q^*} - \lambda^{*T} J(q^*) = 0$$

$$\left(\frac{\partial \mathcal{L}}{\partial \lambda}\right)^T_{q^*,\lambda^*} = {}^{\circ}v_d - J(q)\dot{q}^* = h_q(\dot{q}^*) = 0$$

Questo pb per γ generica non ammette soluzione in generale in forma esplicita

Se consideriamo però il caso di γ funzione quadratica del tipo $\gamma_q(\dot{q}) = \dot{q}^T W \dot{q}$ con $W = W^T > 0 \rightarrow \exists$ soluzione in forma chiusa in fatto $\forall$ sia A: $\dot{q}^T A \dot{q} = \dot{q}^T(A_s + A_A)\dot{q} = \dot{q}^T A_s \dot{q}$ si può sempre considerare W simmetrica senza perdita di generalità perchè la parte antisimmetrica di una matrice dà un

combinabile nulla alla forma quadratica. Inoltre $W > 0$ deve essere definita positiva perché altrimenti non sarebbe garantita l'unicità del minimo. Si studia allora il seguente pb:

$$\begin{cases} \min \ \dot{q}^T W \dot{q} \\ \dot{x}_d = J(q)\dot{q} \end{cases}$$

cioè si vuole realizzare una velocità desiderata impiegando delle velocità di giunto le più piccole possibile. Se la matrice di peso $W = I \Rightarrow$ si minimizza la somma dei quadrati delle velocità di giunto $= \min \dot{q}_1^2 + \dot{q}_2^2 + \dots + \dot{q}_n^2 \Rightarrow$ si assegna lo stesso peso a tutti i giunti. Se voglio invece che i giunti più sforzati (cioè quelli che portano il peso di tutta la struttura del robot) abbiano un peso maggiore (debbano essere più minimizzati) degli altri giunti meno sforzati:

$$W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{bmatrix} \quad \text{con } w_1 \geq w_2 \geq \dots \geq w_n$$

OSS: Se do peso nullo a $\dot{q}_n \Rightarrow$ posso avere anche $\dot{q}_n \to \infty$. Infatti:

$\dot{q}^T W \dot{q} = w_1 \dot{q}_1^2 + w_2 \dot{q}_2^2 + \dots + w_n \dot{q}_n^2 \Rightarrow$ Se $w_n = 0 \Rightarrow \dot{q}_n$ non contribuisce nell'indice da minimizzare per cui può assumere valori anche molto elevati. Al contrario se w_n è molto elevato $\Rightarrow \dot{q}_n$ non potrà assumere valori elevati perché il contributo all'indice è rilevante.

OSS: Si può prendere W in modo che la forma quadratica risultante sia l'energia cinetica del robot. Infatti l'energia cinetica del robot è funzione di q e $\dot{q}$:

$$T(q,\dot{q}) = \frac{1}{2}\dot{q}^T B(q)\dot{q}$$

→ Ad un certo stato dinato del robot è associata una certa energia cinetica a seconda della velocità e della configurazione che il robot ha in quel momento. $B(q)$ è la matrice di inerzia del robot. Potrei in questo modo minimizzare l'energia associata alla realizzazione della velocità $\dot{x}_d$:

$$\begin{cases} \min \ \frac{1}{2}\dot{q}^T B(q)\dot{q} \\ \dot{x}_d = J(q)\dot{q} \end{cases}$$

La matrice $B(q)$ è non simmetrica ed ha pesi maggiori in alto a SX (sui giunti che sono più sforzati) ed è un indice oggettivo mentre W è cmq un indice soggettivo perché è sempre chi progetta a decidere cosa è meglio. In generale risolviamo il pb di minimo

$W > 0$ Simcolato con simcolo: $\gamma_q(\dot{q}) = \dot{q}^T W \dot{q}$ dove $W_q = W(q)$:

$$\boxed{\begin{matrix} \min \\ {}^{\circ}v_d = J(q)\dot{q} \end{matrix} \quad \dot{q}^T W_q \dot{q}} \rightarrow$$ ammette soluzioni in forma chiusa, perché $\gamma_q(\dot{q})$ è una funzione quadratica.

La lagrangiana è la seguente:

$$L_q(\dot{q}, \lambda) = \dot{q}^T W_q \dot{q} + \lambda^T ({}^{\circ}v_d - J(q)\dot{q})$$

Le condizioni di stazionarietà sono:

$$\begin{cases} (1) \ \dfrac{\partial L}{\partial \dot{q}}\Big|_{\dot{q}^*, \lambda^*} = 2\dot{q}^{*T} W_q - \lambda^{*T} J(q) = 0 \\ (2) \ {}^{\circ}v_d = J(q)\dot{q}^* \end{cases}$$

Nelle Hp di $\rho(J(q)) = m$ (rango pieno visto che stiamo considerando robot ridondanti con $n > m$ cioè Jacobiano nell'angolare basso del tipo: $\rightarrow \boxed{\begin{matrix} J(q) \\ n \end{matrix}}\ m$)

Trasponendo la (1) e considerando la (2) otteniamo la soluzione in forma chiusa:

$$(1) \xrightarrow{T} 2 W_q \dot{q}^* = J^T(q)\lambda^* \Rightarrow \dot{q}^* = \frac{1}{2} W_q^{-1} J^T(q)\lambda^* \xrightarrow{(2)} \Rightarrow$$

$$\Rightarrow {}^{\circ}v_d = \frac{1}{2} J(q) W_q^{-1} J^T(q)\lambda^* \Rightarrow \lambda^* = 2\left(J(q) W_q^{-1} J^T(q)\right)^{-1} {}^{\circ}v_d$$

L'Hp di rango pieno di $J(q)$ $\Rightarrow$ la matrice evidenziata inverte è simmetrica e definita positiva infatti $x^T J(q) W_q^{-1} J^T(q) x > 0$ perché $\mathcal{N}(J^T(q)) = \{0\}$.

DIM: $z = J^T(q)x \Rightarrow$ dalla $J(q) W_q^{-1} J^T(q)$ che ha forma quadratica $x^T J(q) W_q^{-1} J^T(q) x = z^T W_q^{-1} z > 0$ perché $W_q^{-1} = W_q^{-T} > 0$ $\Rightarrow$ anche la matrice evidenziata inverte è simmetrica e definita positiva $\Rightarrow$ è invertibile. ⊠

Poniamo allora calcolare la $\dot{q}^*$ cioè la velocità di giunto che realizza la velocità cartesiana desiderata ${}^{\circ}v_d$ e minimizza la norma pesata $\|\dot{q}\|_W = \dot{q}^T W \dot{q} = W_1 \dot{q}_1^2 + \dots + W_n \dot{q}_n^2$ è:

$$\boxed{\dot{q}^* = W_q^{-1} J^T(q)\left(J(q) W_q^{-1} J^T(q)\right)^{-1} {}^{\circ}v_d}$$

Esprimiamo la stessa soluzione in termini più generici: sia A la matrice dei coefficienti e y il vettore dei termini noti $Ax = y$.

$$\begin{matrix} \min \\ (Ax = y) \end{matrix} \quad x^T W x \rightarrow$$

Sistema di m equazioni in n incognite con $\rho(A) = m < n \Rightarrow$ ammette ∞^{n-m} soluzioni

La soluzione che minimizza è indice $x^T W x$ (a minima norma pesata):

$$\boxed{x^* = W^{-1} A^T (A W^{-1} A^T)^{-1} y} \Rightarrow \boxed{x^* = A_W^{\dagger} y}$$

La matrice $A_W^\dagger = W^{-1}A^T(AW^{-1}A^T)^{-1}$ è detta pseudo inversa destra della matrice A. Infatti se A è quadrata ed è invertibile risulta che $A_W^\dagger \equiv A^{-1}$:

→ Teorema di Binet $(AB)^{-1} = B^{-1}A^{-1}$

DIM: $A_W^\dagger = W^{-1}A^T(AW^{-1}A^T)^{-1} = W^{-1}A^T(A^T)^{-1}W A^{-1} = A^{-1}$ ▨.

È chiamata pseudo inversa destra perché se la moltiplica a DX della A (rettangolare) si ottiene l'identità:

DIM: $AA_W^\dagger = AW^{-1}A^T(AW^{-1}A^T)^{-1} = I_m$ ▨

Moltiplicando la A a Sx di A ricordando non si ottiene I_m, infatti:

$$\rho(A_W^\dagger A) = \rho\left(\boxed{A_W^\dagger}\ \boxed{A}\right) = \rho\left(\left[\begin{matrix}\cdots\\ \cdots\end{matrix}\right]\right) = m < n$$

$(m\times m)(m\times n)$ $\qquad$ $(m\times n)$

Perché il prodotto tra due matrici di rango al massimo pari ad m è ancora una matrice di rango al massimo pari ad $m \Rightarrow$ la matrice di dimensioni $m\times n$ non può essere I_n perché la matrice identità è una matrice a rango pieno.

OSS: La matrice pseudo inversa destra è utile anche per calcolare la proiezione di un vettore generico nel nucleo della matrice A. Data $Ax = y \rightarrow \mathcal{N}(A) = \{x : Ax = 0\}$ Pb omogeneo con $\dim(\mathcal{N}(A)) = n - \rho(A) = n - m$; in un robot più è grande la dimensione del null dello Jacobiano tanto più sono gli automovimenti, cioè quei movimenti dei giunti che danno un contributo nullo alla velocità cartesiana ($\dot q$ tali che $\dot v_d = 0$).

Sia $z \in \mathbb{R}^n$ (stessa dimensione di x). In generale $Az \neq 0$.

A partire da z posso generare un vettore $\tilde z$ appartenente al nucleo di A (cioè la proiezione di z nel nucleo di A) nel seguente modo:

$$\boxed{\tilde z = (I_n - A_W^\dagger A)z \in \mathcal{N}(A)} \quad \forall z \in \mathbb{R}^n$$

DIM: $A\tilde z = A(I_n - A_W^\dagger A)z = (A - A)z = 0 \in \mathcal{N}(A)$ ▨.

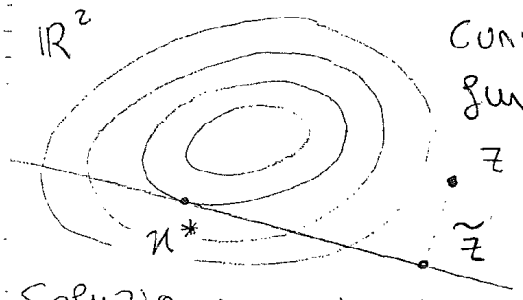
Quindi la matrice $(I_n - A_W^\dagger A)$ è il proiettore nel nucleo di A che permette di generare tutte le soluzioni possibili del Pb di minimizzazione a partire da $x^* = A_W^\dagger y$ nel modo seguente:

$$\boxed{x^s = A_W^\dagger y + (I_n - A_W^\dagger A)z}$$

questa è la soluzione completa al Pb: $y = Ax^s$, $\forall z \in \mathbb{R}^n$.

Tale soluzione rappresenta tutte le ∞^{n-m} soluzioni

ponibile al variare di $z \in \mathbb{R}^m$. È la somma della soluzione dell'eq. non omogenea (cioè $x^* = A_W^+ y$) e di quella dell'eq. omogenea associata (che sono ee $(I_m - A_W^+ A)z$ al variare di z in $\mathbb{R}^m$). Graficamente in $\mathbb{R}^2$:



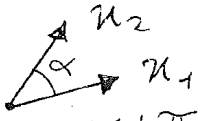
Curve di livello della funzione quadratica $z \in \mathbb{R}^2$ $x^T W x$

Soluzione a minima norma pesata.

Come vediamo z varia in $\mathbb{R}^m$ (cioè in $\mathbb{R}^2$) ma genera delle soluzioni $\tilde{z}$ limitate alla sola retta del vincolo. In robotica ee $\tilde{z}$ sono gli automovimenti cioè ee velocità di giunto che danno velocità cartesiane = 0.

OSS: $\rho(I_m - A_W^+ A) = \mathcal{N}(A) = m - m$ ed è $\perp$ dalla matrice di peso W. Tale matrice (W) interviene poi sul modo in cui viene effettuata la proiezione (se è geometricamente ortogonale o meno).

Infatti, quando definisco un prodotto scalare tra due vettori: $\langle x_1, x_2 \rangle = x_1^T W x_2 \Rightarrow$ varia il concetto classico di ortogonalità legato all'angolo di $\pi/2$ tra i due vettori. Mediante una matrice W opportuna, i vettori a DX potrebbero risultare ortogonali (avere prodotto scalare nullo) pur non essendo geometricamente perpendicolari.



$\alpha \neq \pi/2$

ES: Si vuole minimizzare una $\gamma(x) = x_1^2 + 2x_2^2$ (si pesa di più la componente x_2 rispetto alla x_1.

OSS: $\sqrt{x^T W x} = \|x\|_W$ e $\langle x_A, x_B \rangle = x_A^T W x_B \Rightarrow \|x\| = \sqrt{\langle x, x \rangle}$

$x_A \perp x_B \iff x_A^T W x_B = 0 = \langle x_A, x_B \rangle$

L'ortogonalità classica si ha per: $W = W \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} w & 0 \\ 0 & w \end{bmatrix}$

Nell'esempio: $\langle x_A, x_B \rangle = x_A^T \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} x_B = x_{A1} x_{B1} + 2 x_{A2} x_{B2} = 0$

$\iff x_A = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \rightarrow x_B = \begin{pmatrix} 0 \\ \alpha \end{pmatrix}$: ho ancora perpendicolarità geom.

Se però considero $x_A = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow x_B = \begin{pmatrix} 2\alpha \\ -\alpha \end{pmatrix}$ $\forall \alpha$ perché:

$\langle x_A, x_B \rangle = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{pmatrix} x_{B1} \\ x_{B2} \end{pmatrix} = 0 \iff (1 \; 2) \begin{pmatrix} x_{B1} \\ x_{B2} \end{pmatrix} = 0$

Graficamente i due casi si rappresentano:

$\gamma \neq \pi/2$

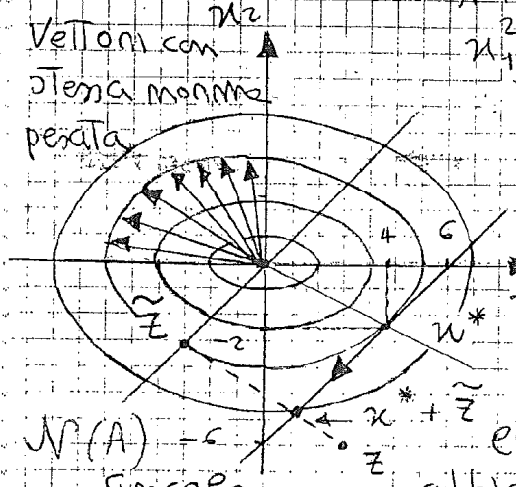
L'angolo tra due vettori ortogonali, secondo il peso W, non è sempre ottuso, ma dipende dalla posizione dei vettori stessi. Tornando all'esempio il simcolo che risolve impone è

$y = Ax \Rightarrow \begin{cases} A = [1\ -1] \\ y = 6 \end{cases} \Rightarrow [1\ -1]\,x = 6$ con $W = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

Il problema completo è:

$\min_{[1\ -1]x=6} x_1^2 + 2x_2^2$

Le curve di livello della funzione $x^T W x$ si ricavano facendo variare c nella eq: $x_1^2 + 2x_2^2 = c$. Ad esempio per $c = 1$ si ha $x_A = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ed $x_B = \begin{pmatrix} 0 \\ 1/\sqrt{2} \end{pmatrix}$. Se $W = I \Rightarrow$ le curve di livello sono circonferenze con centro che, se W è una matrice diagonale con pesi generici $\Rightarrow$ sono ellissi, se W è generica $\Rightarrow$ le curve di livello sono ellissi traslati. In questo caso abbiamo delle ellissi ed i vettori con la stessa norma pesata sono quelli che congiungono l'origine con una determinata curva di livello. Dal simcol $\Rightarrow$



$[1\ -1]\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 6 \Rightarrow x_1 - x_2 = 6 \Rightarrow x_2 = x_1 - 6 \rightarrow$ retta rossa

Per calcolare la soluzione che soddisfa il simcol e minimizza la funzione indice si calcola a partire dalla pseudoinversa:

$(AW^{-1}A^T) = (1\ -1)\begin{pmatrix} 1 & 0 \\ 0 & 1/2 \end{pmatrix}\begin{pmatrix} 1 \\ -2 \end{pmatrix} = \frac{3}{2} \Rightarrow W^{-1}A^T = \begin{pmatrix} 1 \\ -1/2 \end{pmatrix}$

$A_W^\dagger = W^{-1}A^T(AW^{-1}A^T) = \begin{pmatrix} 1 \\ -1/2 \end{pmatrix}\frac{3}{2} = \begin{pmatrix} 2/3 \\ -1/3 \end{pmatrix}$

Si verifica che $A A_W^\dagger = [1\ -1]\begin{pmatrix} 2/3 \\ -1/3 \end{pmatrix} = 1$ OK

La soluzione è:

$x^* = A_W^\dagger y = \begin{pmatrix} 2/3 \\ -1/3 \end{pmatrix} 6 = \begin{bmatrix} 4 \\ -2 \end{bmatrix}$ infatti è verificato che:

$Ax^* = [1\ -1]\begin{bmatrix} 4 \\ -2 \end{bmatrix} = 6 = y$. La proiezione nel nucleo della matrice A si ricava:

$z \rightarrow \tilde{z} = (I - A_W^\dagger A)\,z$; con:

$I - A_W^\dagger A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 2/3 \\ -1/3 \end{bmatrix}[1\ -1] = \begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix} \Rightarrow$ Hanango $m = 1$.

Infatti in questo esempio ho una equazione di simcolo ($m = 1$) in 2 incognite ($n = 2$).

$R\left(\begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix}\right) = R\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right)$ Quindi i vettori nel $N(A)$ sono del tipo $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$

La generica soluzione è:

$$x_s = A_W^+ y + (I - A_W^+ A) z = \begin{bmatrix} 4 \\ -2 \end{bmatrix} + \begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = x^* + \tilde{z}$$

Se $z = \begin{pmatrix} 3 \\ -6 \end{pmatrix} \Rightarrow$ proiettando nel nucleo di A ottengo:

$$\tilde{z} = \begin{bmatrix} 1/3 & 2/3 \\ 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} 3 \\ -6 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$ → La proiezione non è geometricamente ortogonale. Quindi $[1 \ -1]x = 6$ ammette ∞^1 soluzioni, tra le quali $\begin{pmatrix} 4 \\ -2 \end{pmatrix}$ è quella a minima norma pesata (secondo W). La soluzione più intuitiva $x = \begin{pmatrix} 3 \\ -3 \end{pmatrix}$ si ha quando $W = I$. Se avessi dato un peso → ∞ alla componente $x_2 \Rightarrow$ avrei avuto $x = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$ viceversa dando peso nullo alla $x_2 \Rightarrow$ la soluzione sarebbe $x = \begin{pmatrix} 0 \\ -6 \end{pmatrix}$. L'ellisse tende a schiacciarsi lungo x_2 per $x \to \begin{pmatrix} 6 \\ 0 \end{pmatrix}$ mentre tende a schiacciarsi lungo x_1 per $x \to \begin{pmatrix} 0 \\ -6 \end{pmatrix}$.

In robotica, la possibilità di aggiungere la proiezione di z nel $N(A)$ ($\tilde{z}$) alla soluzione x^*, viene sfruttata per cercare la $\dot{q}$ a minima norma pesata, ma quanto più possibile vicina ad una $\dot{q}_0$ desiderata (che si vorrebbe il robot avesse):

$$\boxed{\dot{q}^S = J_W^+(q)\, {}^0v_d + (I - J_W^+(q) J(q))\, \dot{q}_0}$$ → Soluzione che realizza sempre la ${}^0v_d$.

Risolve Pb minimo e soddisfa il vincolo — Si vuole imporre una $\dot{q} \cong \dot{q}_0$ (se possibile) con:

$\dot{q}_0 = \frac{\partial Z(q)}{\partial q}$ → Velocità dei giunti che massimizza la manipolabilità del robot $Z(q)$ o meglio la velocità di giunto che porta in una configurazione di massima manipolabilità (oppure che dà luogo ad una posizione più vicina ---), ma che non realizza la velocità cartesiana desiderata ${}^0v_d$. Questo è il motivo per cui si proietta tale $\dot{q}_0$ nel nullo dello Jacobiano in modo che dia contributo nullo a ${}^0v_d$ cioè risulti che $\tilde{\dot{q}}_0 = J(q)\, \tilde{\dot{q}}_0 = 0$ → ES: Se voglio che il robot appoggi un oggetto su un tavolo senza sbattere il polso combino il movimento voluto per appoggiare l'oggetto (la soluzione x^*) con il automovimento che allontana il polso dal tavolo senza che dia contributo alla ${}^0v_d$ (soluzione $\tilde{z} \in N(A)$) mettendo in condizione di sicurezza

OSS: una approx che si fa per questi studi è considerare il robot fisicamente e calcolare la distanza di 1 braccio del robot da 1 oggetto che lo circonda.

OSS: La $Z(q)$ matrice di manipolabilità (è una matrice di qualità) $\Rightarrow$ la soluzione migliore sarà quella che massimizza $Z(q)$. Bisogna minimizzare gli indici di penalità e massimizzare quelli di qualità.

OSS: Abbiamo risolto il pb della cinematica inversa per i robot non ridondanti mediante l'algoritmo dello Jacobiano inverso $J(q)^{-1}$. Analogamente risolviamo il pb della cinematica inversa per i robot ridondanti mediante l'algoritmo dello Jacobiano pseudo inverso $J_W^{\dagger}(q)$ $\rightarrow$ con minimizzazione/massimizzazione di un eventuale indice di penalità/qualità.

$$\boxed{\dot{q}^* = W_q^{-1} J^T(q) \left(J(q) W_q^{-1} J^T(q) \right)^{-1} {}^0v_d = J_W^{\dagger}(q) \, {}^0v_d}$$

Questa è una formula teorica perché nella pratica si preferisce una implementazione numerica che è molto più rapida.

L'aggiunta di automovimenti deriva dall'esigenza di raggiungere obbiettivi secondari:

$$\boxed{\dot{q}^S = J_W^{\dagger}(q) \, {}^0v_d + \left(I - J_W^{\dagger}(q) J(q) \right) \dot{q}_0} \Longrightarrow \left\| \begin{array}{l} \dot{q}^* = J_W^{\dagger}(q) {}^0v_d \\ J(q) J_W^{\dagger}(q) = I \end{array} \right.$$

$$\Rightarrow \dot{q}^S = \dot{q}^* + \left(I - J_W^{\dagger}(q) J(q) \right) \left(\dot{q}_* - \dot{q}_0 \right) \Rightarrow$$

$$\Rightarrow \dot{q}_S = \dot{q}^* + \left(I - J_W^{\dagger}(q) J(q) \right) \dot{q}_0$$

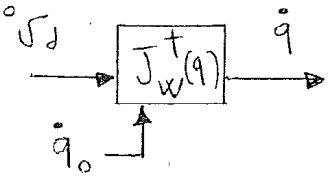
Mi piacerebbe avere una velocità di giunto $\dot{q}_0$ (ma non realizza la ${}^0v_d$) ma è $\dot{q}^*$ che realizza la ${}^0v_d \Rightarrow$ si aggiunge allora a $\dot{q}^*$ la proiezione di $\dot{q}_0$ nel $N(J(q))$.

Dalle relazioni scritte si vede come se la $\dot{q}^* = \dot{q}_0 \Rightarrow$ la correzione da effettuare è nulla.

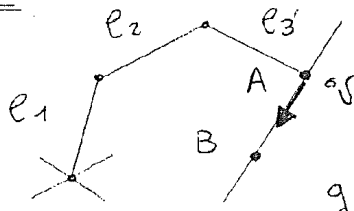
OSS: Le eq. scritte risolvono il pb: di cercare la velocità di giunto $\dot{q}^*$ il più vicino possibile a $\dot{q}_0$ e che realizzi contemporaneamente la velocità desiderata ${}^0v_d$. Si perviene allo stesso risultato utilizzando i moltiplicatori di Lagrange (è un inseguimento di traiettoria vincolato). La specifica completa potrebbe richiedere di 130 minimizzare la traiettoria completa. Infatti se com

$$\boxed{{}^0v_d = J(q)\dot{q} \quad \min_{\dot{q}} (\dot{q} - \dot{q}_0)^T W (\dot{q} - \dot{q}_0)}$$

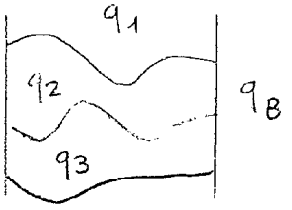
videno $W = B(q)$ si determina localmente lo spostamento a minima energia cinetica, ma non siamo sicuri che sull'intera traiettoria è energia è minima perché potrebbero esserci delle traiettorie ad energia minore. È necessario effettuare un controllo ottimo non lineare (in campo lineare abbiamo il controllo LQ). Noi abbiamo risolto il problema di determinazione della traiettoria in ambito locale perché q è giunta e determiniamo di volta in volta $\dot{q}$. Si parla dell'analisi classica che dà una soluzione $\dot{q}^* \in \mathbb{R}^n$ alla analisi funzionale con $\dot{q}^* \in C(\mathbb{R}^m, (0,T))$. L'algoritmo locale trovato va inserito nello schema a blocchi al posto dello Jacobiano inverso. Se siamo in ambito della robotica avanzata c'è anche il termine di correzione per realizzare la velocità dei giunti che ottimizza la manipolabilità.



OSS:



Data una traiettoria in posizione q_A, determinare gli andamenti delle variabili di giunto per muovere la mano nel punto B con configurazione q_B.



Oppure può interessante potrebbe essere, realizzare un programma in MATLAB che data la descrizione geometrica di un robot in termini di parametri di Denavit-Hartenberg (più un parametro booleano per sapere se il giunto i-mo è prismatico o rotoidale) e l'orientamento + posizionamento in ^0S dell'organo terminale, implementi degli algoritmi di cinematica diretta, di inversione cinematica...

Finisce così la cinematica. Il prossimo argomento è la STATICA del robot, cioè lo studio delle leggi esistenti tra le forze prodotte dai motori sui giunti e le forze applicate dall'esterno: questo studio ci fornirà una interpretazione dell'algoritmo dello Jacobiano Trasposto per il calcolo della cinematica inversa. La dinamica infine ci fornirà le leggi di controllo mediante le quali sarà possibile controllare il robot.