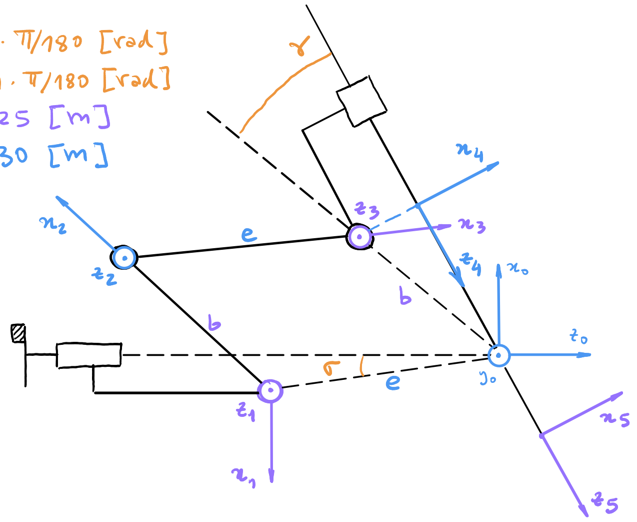
L9 - Constrained Kinematics (Da Vinci Xi) and Newton-Euler Algorithm

CARVALHO, A.
5/6/2023

Kinematics of Da Vinci's links (without wrist)

A rigid body frame is now attached to each link, according to the D-H convention.

$\sigma = 4 \cdot \pi/180$ [rad]
$\gamma = 24 \cdot \pi/180$ [rad]
$b = 0{,}25$ [m]
$e = 0{,}30$ [m]



d_i	θ_i	a_i	α_i
$-e\cos(\sigma)$	θ_1	$e\sin(\sigma)$	$\pi/2$
0	θ_2	b	0
0	θ_3	e	0
0	θ_4	$b\sin(\gamma)$	$\pi/2$
d_5	0	0	0

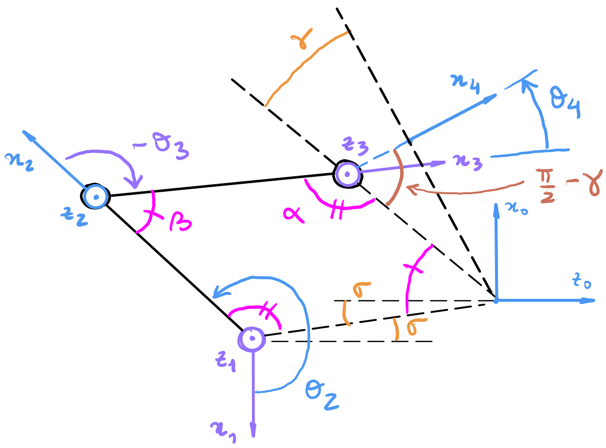
$$q = (\theta_1 \ \theta_2 \ \theta_3 \ \theta_4 \ d_5)^T$$

The motion of joints 2,3 and 4 is constrained to produce a "remote center of motion". That is, z_4 always intersects P_0 at a distance of $d_5 = b\cos\gamma$ from P_4. As a consequence, the parallelogram $P_0P_1P_2P_3$ is formed. The parallelogram's proprieties are now used to solve the kinematic constraints. In particular, we find that

(i) Adjacent angles add up to π, and

(ii) Opposite angles are equal. These angles are computed from $\theta_2, \theta_3, \theta_4$ as

$$\alpha = \theta_2 - \sigma - \pi/2 \ , \ \beta = \theta_3 + \pi \ , \ \alpha = \theta_4 + \gamma + \pi/2 \ .$$



Thus, from (i) and (ii), we find

$$\theta_3 = -\theta_2 + \sigma + \frac{\pi}{2} \quad , \quad \theta_4 = \theta_2 - \sigma - \gamma - \pi \ .$$

These constraints can now be introduced in the D-H Table. The reduced set of coordinates is here notated as

$$\bar{q} = (\theta_1 \ \theta_2 \ d_5)^T .$$

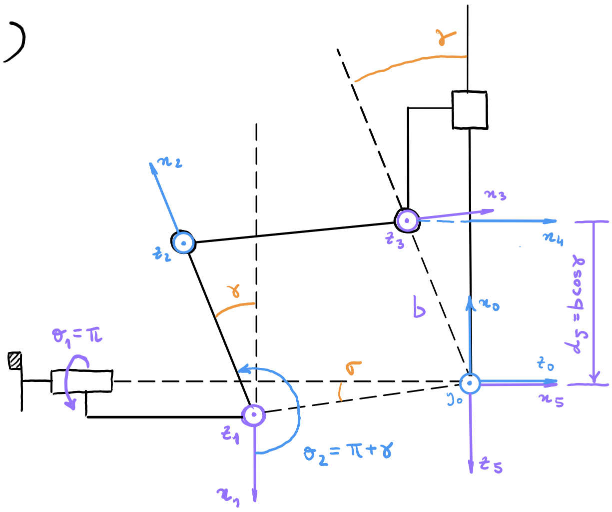
d_i	θ_i	a_i	α_i
$-e c_\sigma$	θ_1	$e s_\sigma$	$\pi/2$
0	θ_2	b	0
0	$-\theta_2 + \sigma + \frac{\pi}{2}$	e	0
0	$\theta_2 - \sigma - \gamma - \pi$	$b s_\gamma$	$\pi/2$
d_5	0	0	0

$$q = (\theta_1 \ \ \theta_2 \ \ -\theta_2 + \sigma + \tfrac{\pi}{2} \ \ \theta_2 - \sigma - \gamma - \pi \ \ d_5)^T$$

- Home Configuration (joint offsets)

It will be very useful to offset the previous D-H coordinates. Here, the offsetted coordinates, $\bar{q}'$, are related to the previous reduced coordinates through

$$\bar{q} = \bar{q}' + \text{offset} .$$



Placing the manipulator in the illustrated configuration and specifying $\bar{q}' = 0$ for this configuration, yields

$$\text{offset} = (\pi, \pi+\gamma, bc_\gamma)^T - (0,0,0) .$$

We may now replace $\bar{q} = \bar{q}' + \text{offset}$ in the D-H Table, as follows.

d_i	θ_i	a_i	α_i
$-e\cos(\sigma)$	$\theta'_1 + \pi$	$e\sin(\sigma)$	$\pi/2$
0	$\theta'_2 + \pi + \gamma$	b	0
0	$-(\theta'_2+\pi+\gamma) + \sigma + \frac{\pi}{2}$	e	0
0	$(\theta'_2+\pi+\gamma) - \sigma - \gamma - \pi$	$b\sin(\gamma)$	$\pi/2$
$d'_5 + b\cos\gamma$	0	0	0

$\bar{q} = (\theta'_1 + \pi \quad \theta'_2 + \pi + \gamma \quad d'_5 + bc_\gamma)^T$

=

d_i	θ_i	a_i	α_i
$-e\cos(\sigma)$	$\theta'_1 + \pi$	$e\sin(\sigma)$	$\pi/2$
0	$\theta'_2 + \pi + \gamma$	b	0
0	$-\theta'_2 - \gamma + \sigma - \frac{\pi}{2}$	e	0
0	$\theta'_2 - \sigma$	$b\sin(\gamma)$	$\pi/2$
$d'_5 + b\cos\gamma$	0	0	0

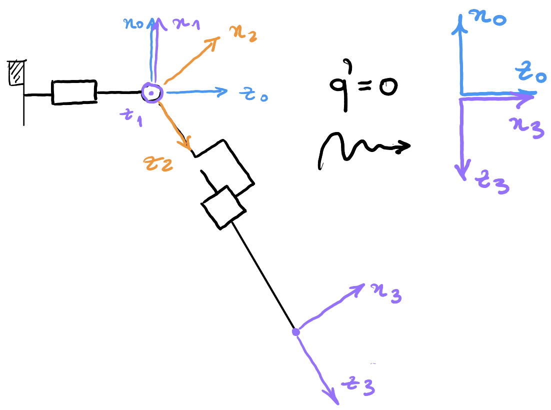
$q = (\theta'_1+\pi \quad \theta'_2+\pi+\gamma \quad -\theta'_2-\gamma+\sigma-\frac{\pi}{2} \quad \theta'_2-\sigma \quad d'_5+bc_\gamma)^T$

- Relation with the reduced links formulation

Note that the previous offset choice was not arbitrary.

Recall the previous reduced DH*, without the wrist, as illustrated. Then, since the home configurations, $\bar{q}'=0$ and $q'=0$, represent identical configurations, in both joint diagrams, and the joint directions are equal, we can conclude that $\bar{q}' = q'$, or,

$$\theta'_1 = \theta'_1 \quad , \quad \theta'_2 = \theta'_2 \quad , \quad d'_5 = d'_3 .$$



d_i	θ_i	a_i	α_i
0	θ'_1	0	$-\pi/2$
0	$\theta'_2 - \pi/2$	0	$\pi/2$
d'_3	0	0	0

$q = (\theta'_1 \quad \theta'_2 - \frac{\pi}{2} \quad d'_3)^T$

* See the class notes from Lab.3 .

Newton - Euler Formulation

- Forward Recursion $(\omega_i^i, \dot\omega_i^i, \ddot p_{c_i}^i)$, $i = 1, \dots, n$

$$\omega_i^i = \begin{cases} R_i^{i-1\,T}\,\omega_{i-1}^{i-1} & ,\ q_i \text{ is prismatic} \\ R_i^{i-1\,T}(\omega_{i-1}^{i-1} + \dot q_i \hat k) & ,\ q_i \text{ is revolute} \end{cases}$$

$$\dot\omega_i^i = \begin{cases} R_i^{i-1\,T}\,\dot\omega_{i-1}^{i-1} & ,\ q_i \text{ is prismatic} \\ R_i^{i-1\,T}(\dot\omega_{i-1}^{i-1} + \ddot q_i \hat k + \dot q_i\, \omega_{i-1}^{i-1} \times \hat k) & ,\ q_i \text{ is revolute} \end{cases}$$

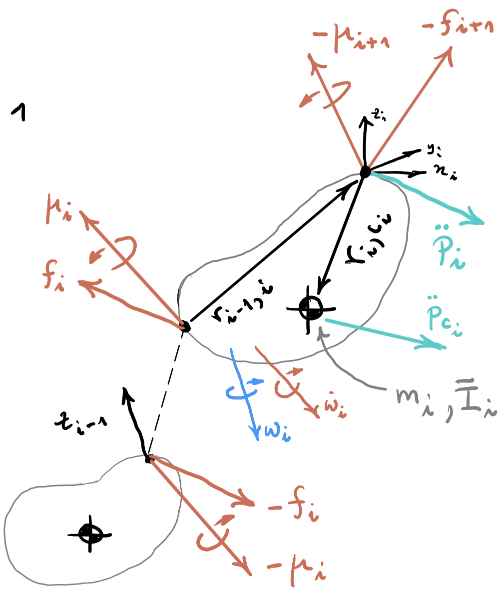
$$\ddot p_i^i = \begin{cases} R_i^{i-1\,T}(\ddot p_{i-1}^{i-1} + \ddot q_i \hat k) + 2\dot q_i\, \omega_i^i \times (R_i^{i-1\,T}\hat k) \\ \quad + \dot\omega_i^i \times (R_i^{i-1\,T} r_{i-1,i}^{i-1}) + \omega_i^i \times \left(\omega_i^i \times (R_i^{i-1\,T} r_{i-1,i}^{i-1})\right) & ,\ q_i \text{ is prism.} \\ R_i^{i-1\,T}\,\ddot p_{i-1}^{i-1} + \dot\omega_i^i \times (R_i^{i-1\,T} r_{i-1,i}^{i-1}) \\ \quad + \omega_i^i \times \left(\omega_i^i \times (R_i^{i-1\,T} r_{i-1,i}^{i-1})\right) & ,\ q_i \text{ is rev.} \end{cases}$$

$$\ddot p_{c_i}^i = \ddot p_i^i + \dot\omega_i^i \times r_{i,c_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,c_i}^i)$$

- Backward Recurssion (f_i^i, μ_i^i), $i = n, \dots, 1$

$$f_i^i = m_i\, \ddot p_{c_i}^i + R_{i+1}^i f_{i+1}^{i+1}$$

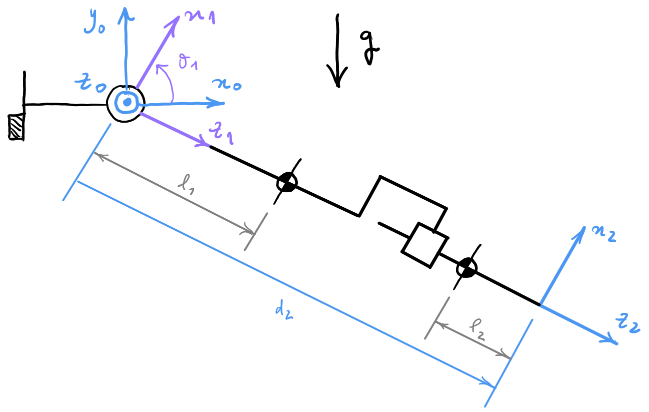
$$\mu_i^i = I_i^i\, \dot\omega_i^i + \omega_i^i \times (I_i^i\, \omega_i^i) + R_{i+1}^i \mu_{i+1}^{i+1} - r_{i,c_i}^i \times (R_{i+1}^i f_{i+1}^{i+1}) + (R_i^{i-1\,T} r_{i-1,i}^{i-1} + r_{i,c_i}^i) \times f_i^i$$



- Joint axis projection

$$\tau_i = \begin{cases} \hat k^T R_i^{i-1} f_i^i & ,\ q_i \text{ is prismatic} \\ \hat k^T R_i^{i-1} \mu_i^i & ,\ q_i \text{ is revolute} \end{cases}$$

- Example – Polar Manipulator (9th Set of Problems)



d_i	θ_i	a_i	α_i
0	ϑ_1	0	$\pi/2$
d_2	0	0	0

$$r^1_{1,C_1} = \ell_1 \hat{k} \quad , \quad r^2_{2,C_2} = -\ell_2 \hat{k}$$

- Implementation in matlab/simulink

The equations of motion can be partitioned as follows,

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \;\Rightarrow\; \underbrace{B(q)\ddot{q}}_{\text{acceleration dynamics}} + \underbrace{\phi(q,\dot{q})}_{\text{velocity dynamics}} + \underbrace{g(q)}_{\text{gravity dynamics}} = \tau \,.$$

A function can be created to simbolically compute τ, through

$$\tau = \mathrm{Tau\_NewtonEuler}(DH, M, \dot{q}, \ddot{q}, \omega_0^0, \dot{\omega}_0^0, \ddot{p}_0^0(|g|)) \,.$$

Then, for efficiency sake, the previous partitions can be created through

$$B(q)\ddot{q} = \tau_a = \mathrm{Tau\_NewtonEuler}(DH, M, 0, \ddot{q}, 0, \dot{\omega}_0^0, \ddot{p}_0^0(0)) \,,$$

$$\phi(q,\dot{q}) = \mathrm{Tau\_NewtonEuler}(DH, M, \dot{q}, 0, \omega_0^0, 0, 0) \,,$$

$$g(q) = \mathrm{Tau\_NewtonEuler}(DH, M, 0, 0, 0, 0, \ddot{p}_0^0(|g|) - \ddot{p}_0^0(0)) \,.$$

Then, the mass matrix is determined through $B = \partial \tau_a / \partial \ddot{q}$.

matlabFunctionBlock( ) is now used to create a simulink model.