Intro To Robotics

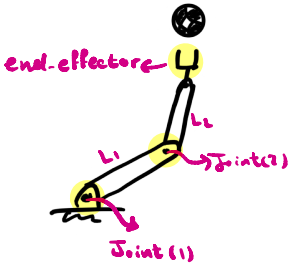
Coordinate Transformation

Q: Why do we need Kinematics & Dynamics?

- To study the behaviour and performance of robots

Q: what are manipulators?

- Set of links connected together with joints to perform a specific task (pick&place,welding,assembly,...)

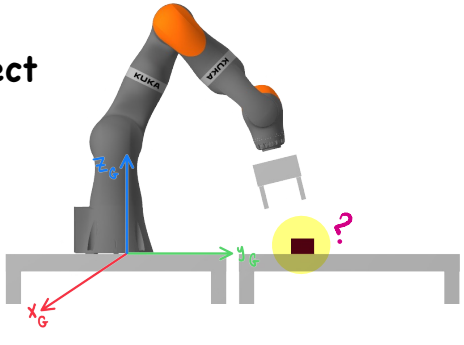


Problem : pick the ball using robot arm

- To study the behaviour and performance of robots

Challenges:

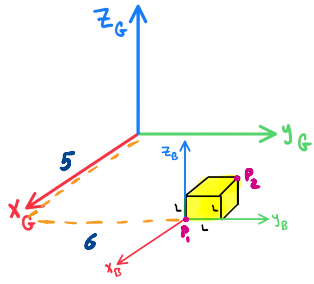
- Pose (position + orientation) of the object
- Weight of the object
- Size of the object
- World/environment around the object



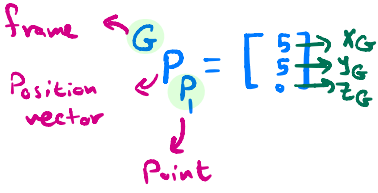
Q: Where is the box?

- We represent everything w.r.t the global/fixed/world frame

Q: How to represent point P1?



- In Global Frame {G}

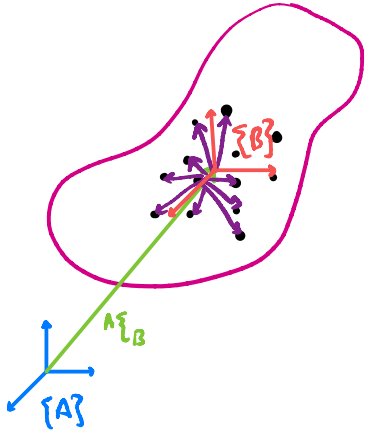


- In Body Frame {B}

$$^BP_{P_1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{X_B Y_B Z_B} \qquad ^BP_{P_2} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}_{X_B Y_B Z_B}$$

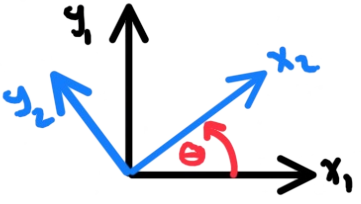
Rigid Body : a set of points with fixed distance between each pair

If ^A_BT is known and the positions of all points are known w.r.t {B}, then the position of every point in the rigid body is known in {A}



Rotation Matrix

Q: What is the relation between frame {1} and frame {2}?



- Using Rotation Matrix

Q: What is Rotation Matrix?

- A (3x3) matrix that describe rotational relationship between one frame w.r.t another frame.

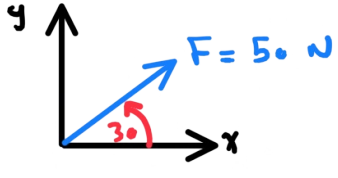
${}^{1}R_{2}$ Rotation of frame{2} w.r.t frame{1}

Dot product recap:

$$F_x = \vec{F} \cdot \vec{x} = |\vec{F}| \cdot |\vec{x}| \cdot \cos(\theta)$$

$$= 1_0 \cos(30)$$

Projection of F on X direction

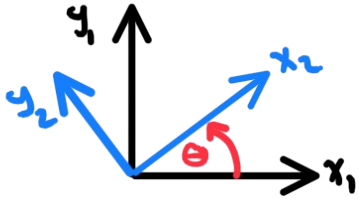


$$F_y = \vec{F} \cdot \vec{y} = |\vec{F}| \cdot |\vec{y}| \cdot \cos(90 - \theta)$$

$$= 1_0 \sin(30)$$

Projection of F on y direction

Let's do the same but with frames



$$
{}^1R_2 = \begin{matrix} x_1 \\ y_1 \\ z_1 \end{matrix}\overset{\overset{\text{new}}{\downarrow}}{\begin{bmatrix} x_2 \cdot x_1 & y_2 \cdot x_1 & z_2 \cdot x_1 \\ x_2 \cdot y_1 & y_2 \cdot y_1 & z_2 \cdot y_1 \\ x_2 \cdot z_1 & y_2 \cdot z_1 & z_2 \cdot z_1 \end{bmatrix}}
\quad \text{old}
$$

$$
\begin{aligned}
x_2 \cdot x_1 &= 1 \cdot 1 \cdot \cos(\theta) \\
x_2 \cdot y_1 &= 1 \cdot 1 \cdot \sin(\theta) \\
x_2 \cdot z_1 &= 1 \cdot 1 \cdot \cos(90) = 0
\end{aligned}
$$

$$
{}^1R_2 = \begin{matrix} x_1 \\ y_1 \\ z_1 \end{matrix}\overset{\begin{matrix} x_2 & y_2 & z_2 \end{matrix}}{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}} = \begin{bmatrix} {}^1x_2 & {}^1y_2 & {}^1z_2 \end{bmatrix}
$$

Standard Rotation Matrices

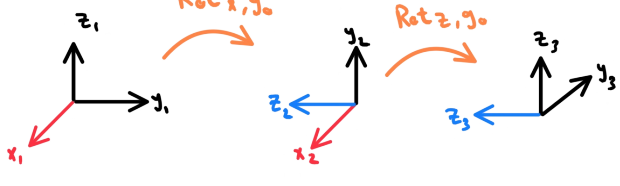
$$
R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{bmatrix} \qquad R_{y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}
$$

$$
R_{z,\theta} = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
$$

Notes:

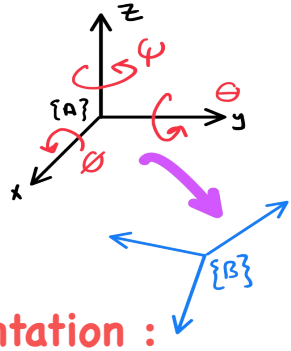
- $R^{-1} = R^T$
- $\det(R) = \pm 1$
- R is "Orthogonal"
- $s\theta = \sin(\theta)$
- $c\theta = \cos(\theta)$

Sequence of Rotations



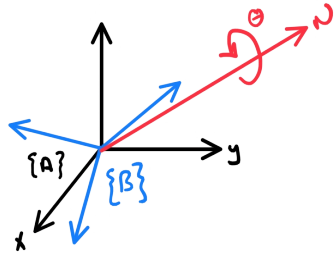
Euler's theorem:

- Any two independent orthogonal frames can be related by a sequence of Rotations (not more than 3) about coordinates, where no two successive rotations may be about the axis



Angel-axis representation :

- Any two independent orthogonal frames can be related by a single rotation about some axis. $(\theta, \hat{v})$ (θ: angel, $\hat{v}$: axis)



Angel-axis representation :

- Quaternion can be considered as a rotation of θ about the unit vector $\hat{v}$ which are related to quaternion components by $\vec{q} = \cos\frac{\theta}{2} < \hat{v} \sin\frac{\theta}{2} >$

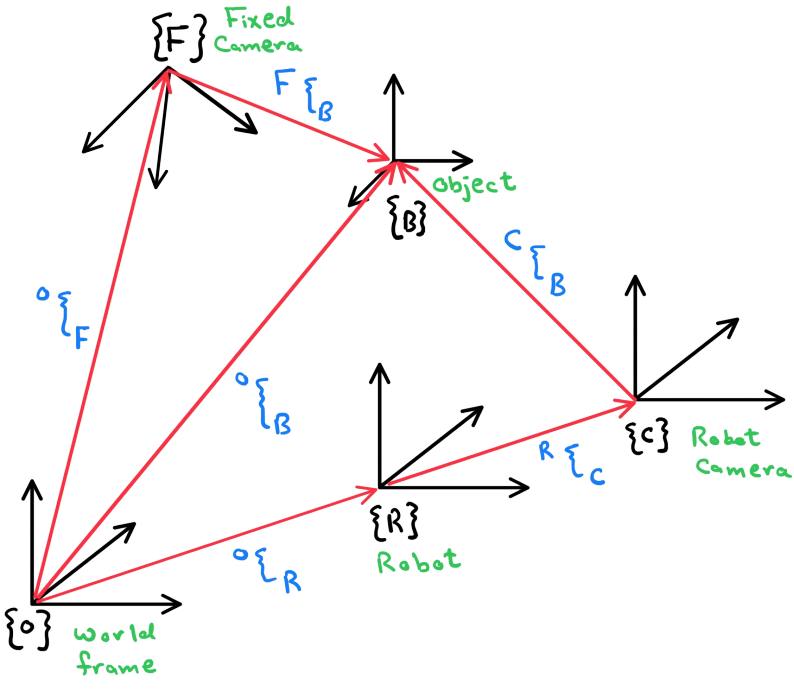
Different Representations of Rotation

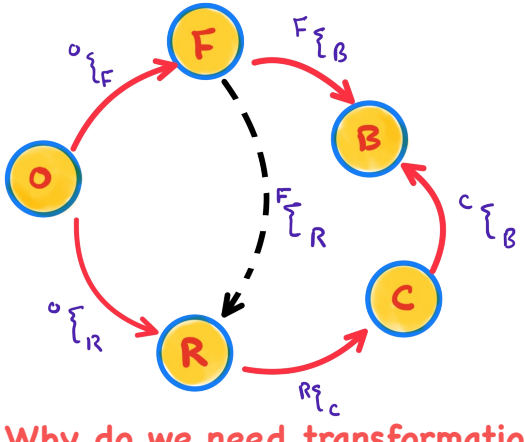
	Intuition	Chain	Memory	Numerical
Rotation Matrix	✓	✓	9	✗
Axis-Angle	✓	✗	4	
Euler Angels	✓		3	
Quaternion	✗	✓	4	✓

Notes:

- All representations can be used interchangeably
- Min number to represent Rotation -> 3
- Min number to represent Transformation -> 6

Relative Poses





° {R : Relative pose that describes frame {R} w.r.t frame {0}

Q: Why do we need transformations?

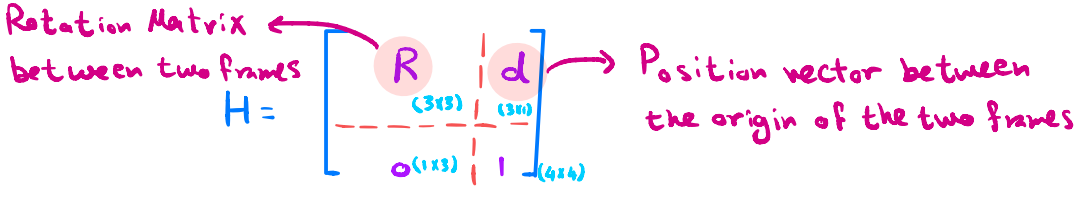
- Because it allow us to transfer information from one frame to another

Position Velocity Acceleration

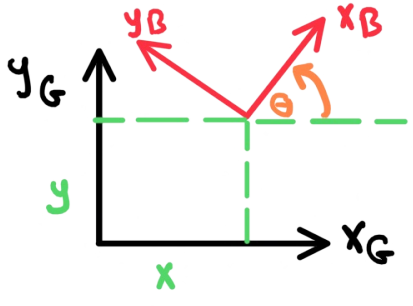
Homogeneous Transformation

Q: What is total homogeneous transformation?

- Total change of a frame w.r.t another frame



$$ {}^{G}H_{B} = \begin{bmatrix} c\theta & -s\theta & 0 & x \\ s\theta & c\theta & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



Note:

- This way, we have the ability to completely represent rigid Bodies w.r.t global frame.

Q: But how?

1- Assign body frame

2- Find Transformation Matrix between Body frame & Global frame

Q: What is the inverse of homogeneous transformation?

$$ {}^{1}H_{2} = \begin{bmatrix} {}^{1}R_{2} & {}^{1}d_{2} \\ 0 & 1 \end{bmatrix}, \quad \left({}^{1}H_{2}\right)^{-1} = \begin{bmatrix} \left({}^{1}R_{2}\right)^{T} & -\left({}^{1}R_{2}\right)^{T} {}^{1}d_{2} \\ 0 & 1 \end{bmatrix} $$

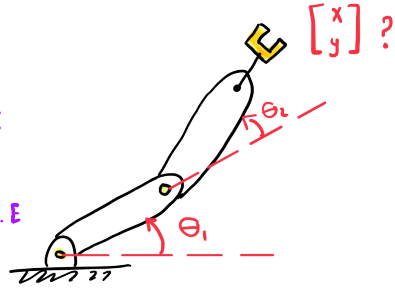
Forward Kinematics

Given:

- Joint Variables $\begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$

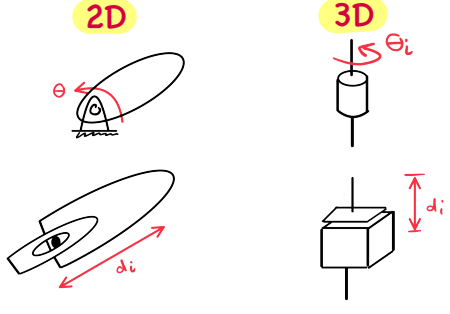
Find:

- Pose of the end-effector $\begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}$ Position of E.E / Orientation of E.E



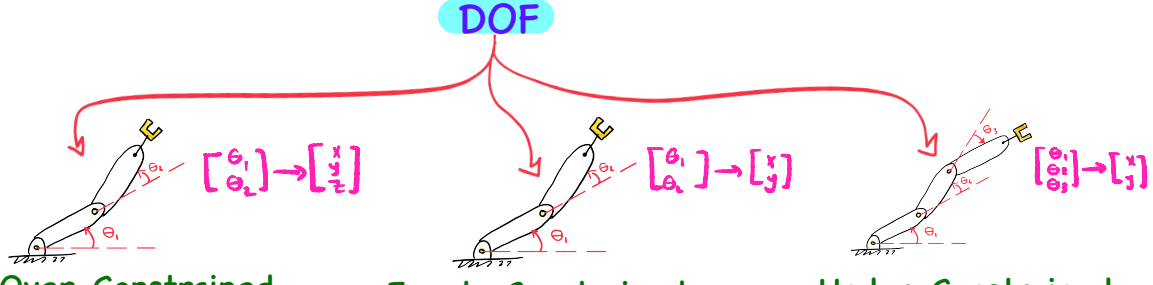
Q: But how? Using DH-Convention

Types of Joints:
- Revolute
- Prismatic



Task Space: pose (position+orientation) of end-effector

Q: What is degree of freedom?



- Over-Constrained
- Under-Actuated

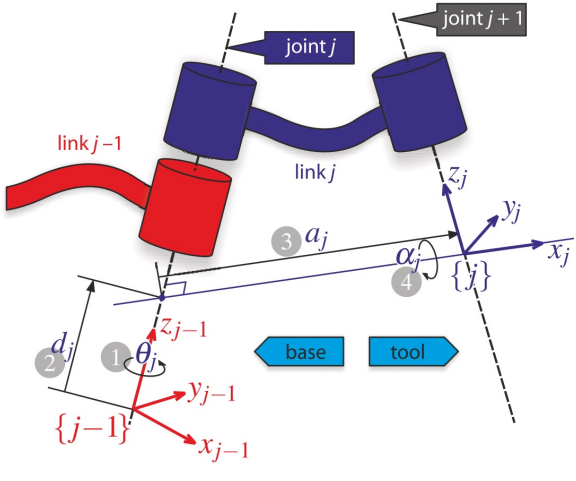
- Exact -Constrained

- Under-Constrained
- Over-Actuated

The map between Joint-space & Task-space:

DH-Convention :

- A systematic sequence of steps to solve the forward Kinematics problem for serial manipulators.

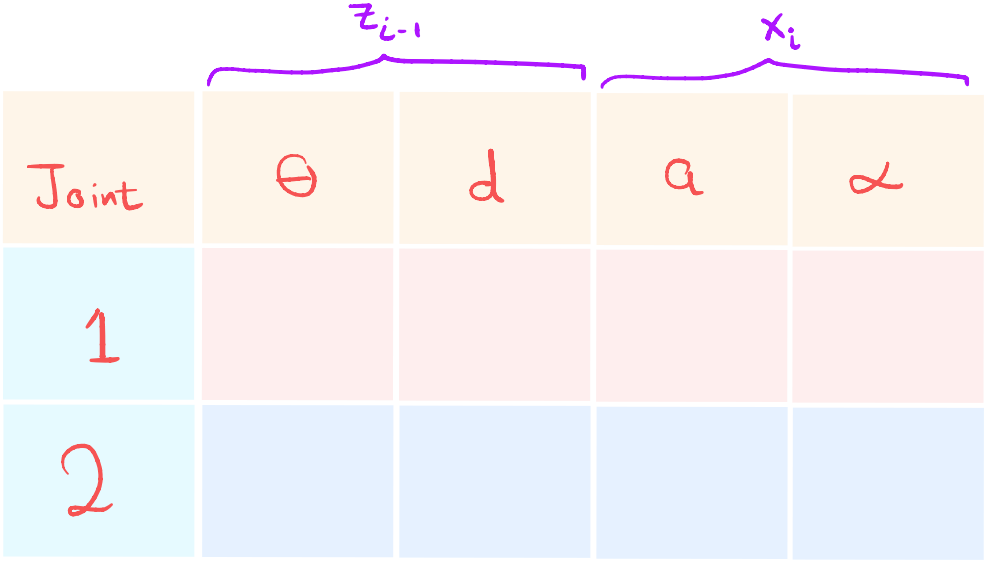


Joint angle	θ_j	the angle between the x_{j-1} and x_j axes about the z_{j-1} axis	revolute joint variable
Link offset	d_j	the distance from the origin of frame $j-1$ to the x_j axis along the z_{j-1} axis	prismatic joint variable
Link length	a_j	the distance between the z_{j-1} and z_j axes along the x_j axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$	constant
Link twist	α_j	the angle from the z_{j-1} axis to the z_j axis about the x_j axis	constant
Joint type	σ_j	$\sigma = R$ for a revolute joint, $\sigma = P$ for a prismatic joint	constant

Steps of DH

1- Assign axes z & x only

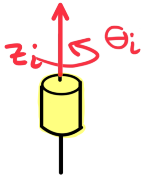
2- Fill DH Table



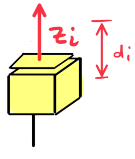
Step(1) : Assign Axes

I) Put all z-axis

Revolute Joint



Prismatic Joint



II) Assign x-axis

Rule(1): $x_0 \perp z_0$
↳ in any direction

Rule(2): $x_i \perp$ and $\cap$ z_i , z_{i-1}

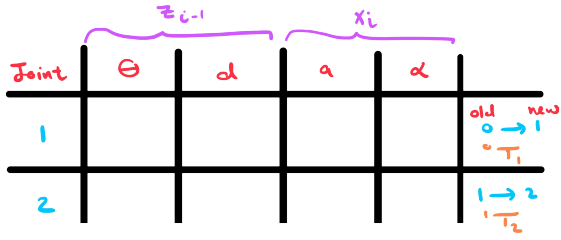
II) End-Effector Frame

Rule(1): $z_{E.E} \;//\; z_{last}$

Rule(2): $x_{E.E} \perp$ and $\cap$ $z_{E.E}$, z_{last}

Step(2) : Fill DH-Table

- Transform from one frame to another is done in 4 steps.



θ_i : Rotation around z_{i-1} between x_{i-1} and x_i

d_i : Translation along z_{i-1} from o_{i-1} to o_i

a_i : Translation along x_i from o_{i-1} to o_i

α_i : Rotation around x_i between z_{i-1} and z_i

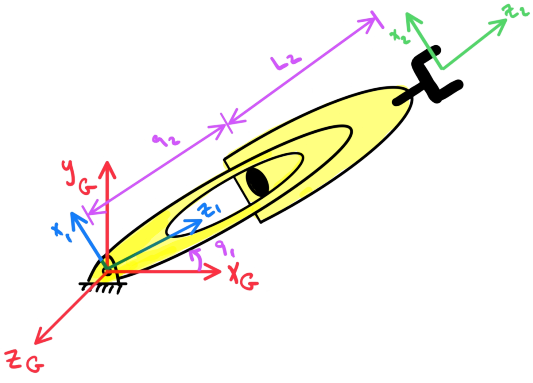
$$^{i-1}T_i = T_{rot\ z_{i-1},\theta} * T_{trans\ z_{i-1},d} * T_{trans\ x_i,a} * T_{rot\ x_i,\alpha_i}$$

$$= \begin{bmatrix} c\theta & -s\theta & 0 & 0 \\ s\theta & c\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha & -s\alpha & 0 \\ 0 & s\alpha & c\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c\theta & -s\theta c\alpha & s\theta s\alpha & a c\theta \\ s\theta & c\theta c\alpha & -c\theta s\alpha & a s\theta \\ 0 & s\alpha & c\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Example (1): RP Robot

Joint	θ	d	a	α	old → new
1	$(\pi/2 + q_1)$	0	0	$\pi/2$	$^0T_1$
2	0	$(L_2 + q_2)$	0	0	$^1T_2$



$$^0T_2 = {}^0T_1 * {}^1T_2 = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

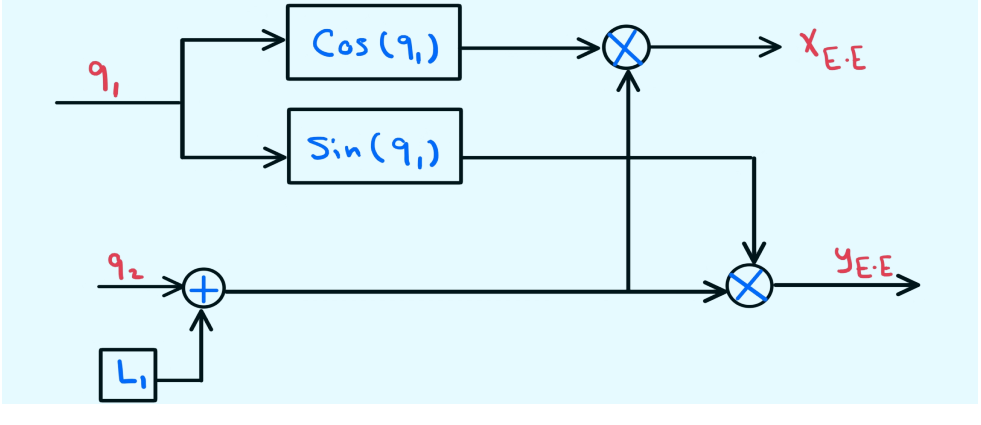
Orientation of E.E w.r.t global frame

Position of E.E w.r.t global frame

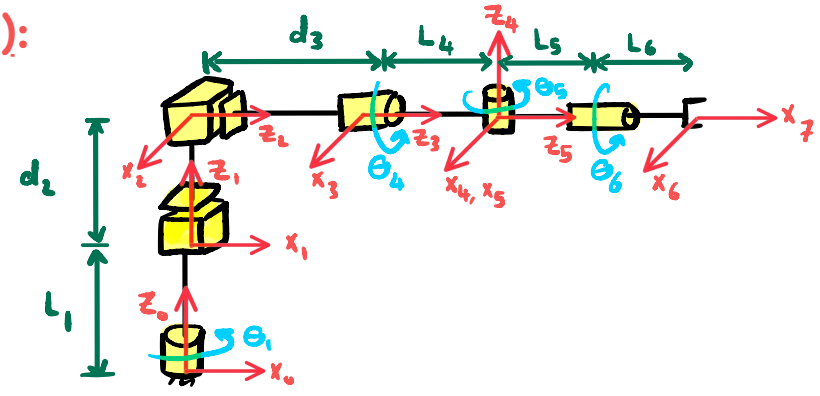
$$x_{E.E} = (L_2 + q_2)\cos(q_1)$$

$$y_{E.E} = (L_2 + q_2)\sin(q_1)$$

Simulink Model



Example (2):



Joint	θ	d	a	α	z_{i-1} / x_i	
1	θ_1	L_1	0	0	old → new: 0 → 1	${}^0T_1$
2	0	d_2	0	$-\pi/2$	1 → 2	${}^1T_2$
3	0	d_3	0	0	2 → 3	${}^2T_3$
4	θ_4	L_4	0	$\pi/2$	3 → 4	${}^3T_4$
5	θ_5	0	0	$-\pi/2$	4 → 5	${}^4T_5$
6	θ_6	$L_5 + L_6$	0	0	5 → 6	${}^5T_6$

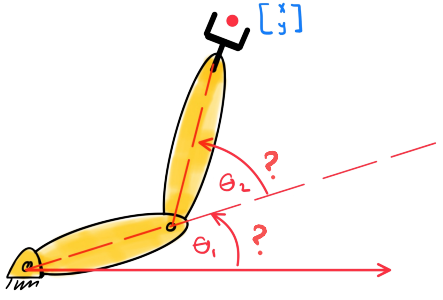
$${}^0T_6 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \cdot {}^5T_6$$

Inverse Kinematics

Given:

- Pose of the end-effector

$$\begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix}$$



Find:

- Joint-variables

$$\begin{bmatrix} q_1 \\ q_2 \\ | \\ | \\ q_n \end{bmatrix}$$

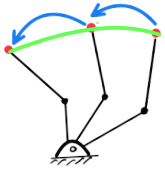
Q: But how?

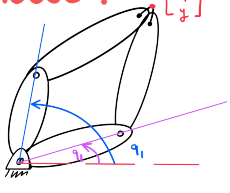
1- Geometrical Approach
2- Numerical Approach

- **Solution** could be → Unique / Multiple / No solution

Q: In case of multiple solutions which one to choose ?

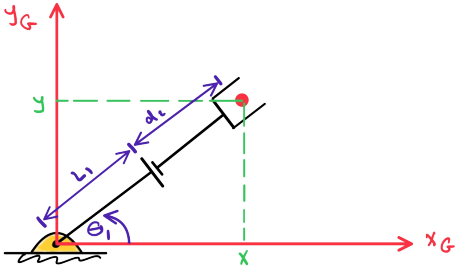
- Energy Efficiency
- Trajectory Smoothness
- Collision Avoidance

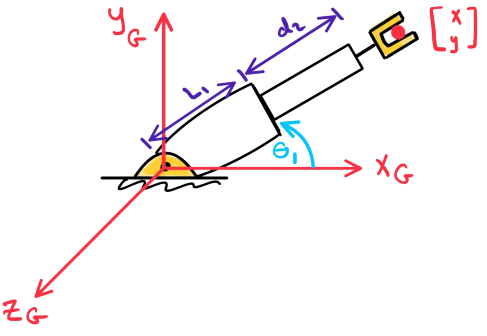




Inverse Kinematics (Geometric Approach)

Example (1): RP Robot using "geometric method"





Find: $\begin{bmatrix} \theta_1 \\ d_2 \end{bmatrix} = ?$

Given: $\begin{bmatrix} x \\ y \end{bmatrix}$

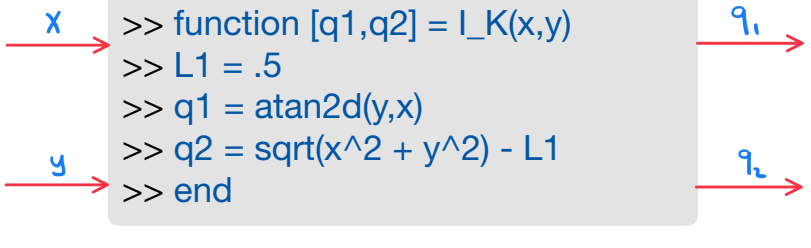
$$\tan(\theta_1) = \frac{y}{x} \Rightarrow \theta_1 = \tan^{-1}\left(\frac{y}{x}\right)$$

- Using Pythagorean Theorem

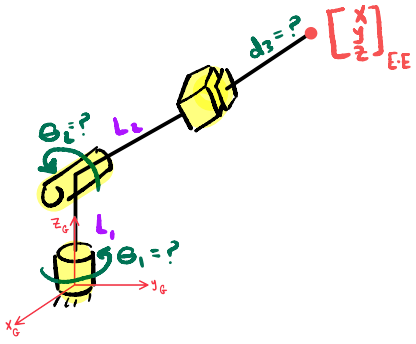
$$(L_1 + d_2)^2 = x^2 + y^2 \Rightarrow L_1 + d_2 = \sqrt{x^2 + y^2}$$

$$d_2 = \sqrt{x^2 + y^2} - L_1$$

Model Simulation Using "Matlab"

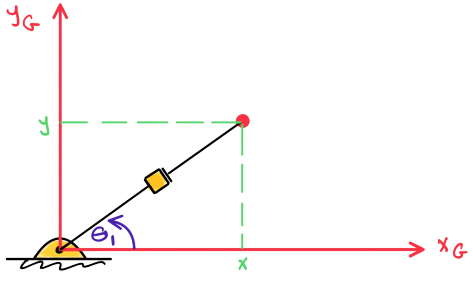


Example (2): RRP Robot



1 Find θ_1

- Project everything in x-y plane



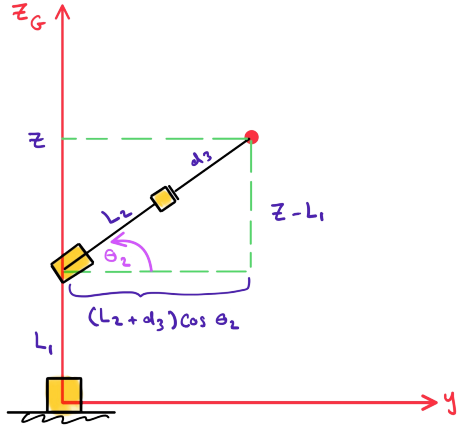
$$\tan\theta_1 = \frac{y}{x}$$

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right)$$

$$x^2 + y^2 = \left[(L_2 + d_3)\cos(\theta_2)\right]^2$$

2 Find θ_2, d_3

- Project on y-z plane



$$\theta_2 = \tan^{-1}\left(\frac{z - L_1}{\sqrt{x^2 + y^2}}\right)$$

- Using Pythagorean Theorem

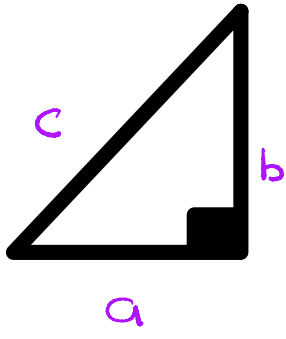
$$L_2 + d_3 = \sqrt{x^2 + y^2 + (z - L_1)^2}$$

$$d_3 = \sqrt{x^2 + y^2 + (z - L_1)^2} - L_2$$

Laws used in geometric approach

1 Pythagorean Theorem

$$a^2 + b^2 = c^2$$

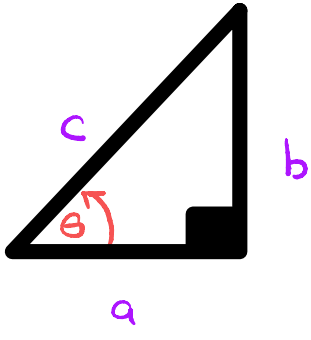


2 SOHCAHTOA

$$\sin(\theta) = \frac{b}{c}$$

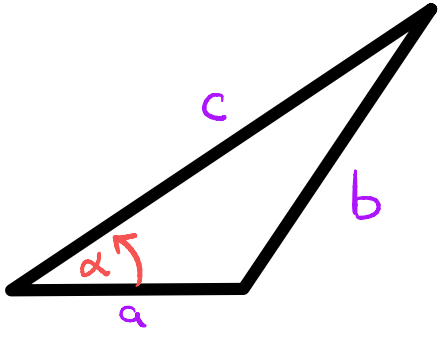
$$\cos(\theta) = \frac{a}{c}$$

$$\tan(\theta) = \frac{b}{c}$$



3 Law of cosines

$$b^2 = a^2 + c^2 - 2 \cdot a \cdot c \cdot \cos(\alpha)$$



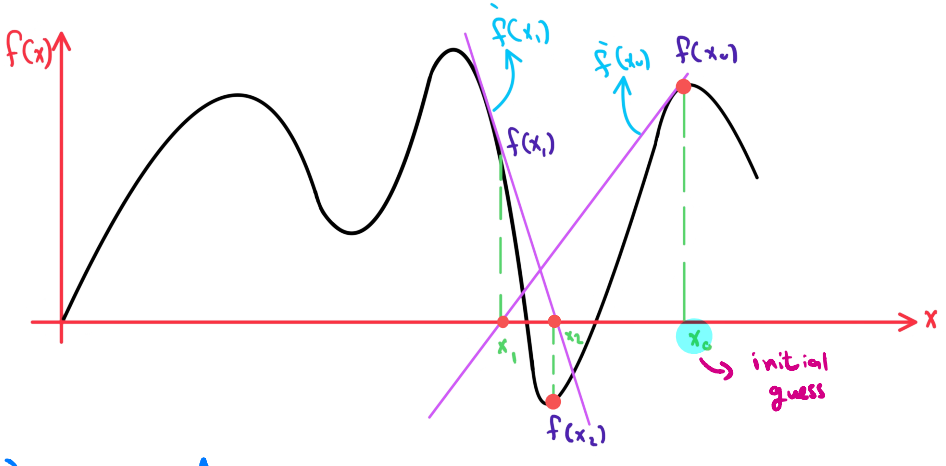
Inverse Kinematics (Numerical Approach)

Q: Why?

- Geometric approach is hard for robot with more than 4-dof.

Newton-Raphson Method:

- Iterative method to solve complex functions



$$\dot{f}(x_0) = \frac{\Delta y}{\Delta x} = \frac{f(x_0) - 0}{x_0 - x_1} = \frac{f(x_0)}{x_0 - x_1}$$

$$x_1 = x_0 - [\dot{f}(x_0)]^{-1} f(x_0)$$

"initial guess" → x_0; value at initial guess → $f(x_0)$

General Equation :

$$x_{n+1} = x_n - \dot{f}(x_n)^{-1} f(x_n)$$

Stopping Criteria:

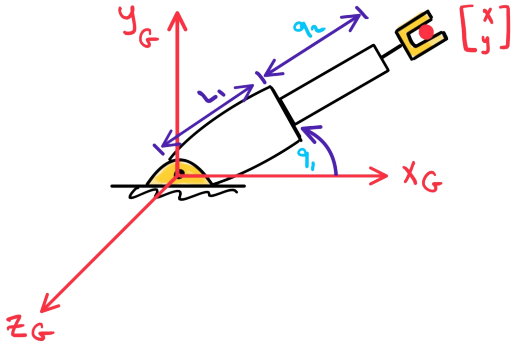
1- Convergence of $f(x_n)$ to a value close to $f(x_d)$

2- Max number of iterations reached (ex: 100 iter)

Example: RP-Robot using "numerical method"

Find: $\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = ?$

Given: $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$



- From forward kinematics

$$x_{E.E} = (L_1 + q_2)\cos(q_1)$$

$$y_{E.E} = (L_1 + q_2)\sin(q_1)$$

Recall: $x_{n+1} = x_0 - \dot{f}(x_0)^{-1} f(x_0)$

$$\vec{x} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad \vec{f} = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

$$\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_{n+1} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_n - \dot{f}(x_0)^{-1} \begin{bmatrix} (L_1+q_2)\cos(q_1) \\ (L_1+q_2)\sin(q_1) \end{bmatrix}$$

What? "Jacobian Matrix"

Jacobian Matrix :

$$J = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{cc} q_1 & q_2 \end{array} \\ \left[\begin{array}{c|c} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \hline \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{array}\right] \end{array}$$

$$F_1 = (L_1 + q_2)\cos(q_1)$$

$$F_2 = (L_1 + q_2)\sin(q_1)$$

$$\frac{\partial F_1}{\partial q_1} = -(L_1+q_2)\sin(q_1) \quad , \quad \frac{\partial F_1}{\partial q_2} = \cos(q_1)$$

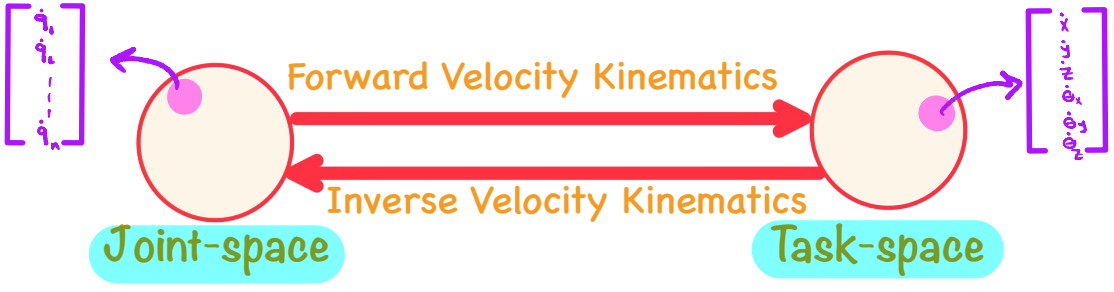
$$\frac{\partial F_2}{\partial q_1} = (L_1+q_2)\cos(q_1) \quad , \quad \frac{\partial F_2}{\partial q_2} = \sin(q_1)$$

$$J = \begin{array}{c} \\ F_1 \\ F_2 \end{array} \begin{array}{c} \begin{array}{cc} q_1 & q_2 \end{array} \\ \left[\begin{array}{c|c} -(L_1+q_2)\sin(q_1) & \cos(q_1) \\ \hline (L_1+q_2)\cos(q_1) & \sin(q_1) \end{array}\right] \end{array}$$

$$\underbrace{\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_{n+1}}_{\vec{x}_{n+1}} = \underbrace{\begin{bmatrix} q_1 \\ q_2 \end{bmatrix}_{n}}_{\vec{x}_n} - \underbrace{\begin{bmatrix} -(L_1+q_2)\sin(q_1) & \cos(q_1) \\ (L_1+q_2)\cos(q_1) & \sin(q_1) \end{bmatrix}^{-1}}_{\vec{J}^{-1}(\vec{x}_n)} \underbrace{\begin{bmatrix} (L_1+q_2)\cos(q_1) - 2 \\ (L_1+q_2)\sin(q_1) - 1 \end{bmatrix}}_{\vec{F}(x_n)}$$

Forward Velocity Kinematics

The map between Joint-space & Task-space:



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

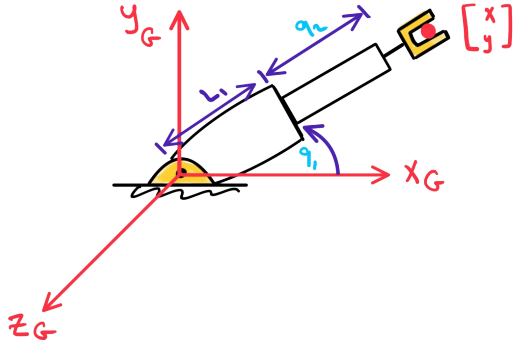
- $\dot{X}$: velocity of E.E
- J: Jacobian Matrix
- $\dot{q}$: Joint velocities

Differentiation Approach

- From forward kinematics

$$x_{E.E} = (L_1 + q_2)\cos(q_1)$$

$$y_{E.E} = (L_1 + q_2)\sin(q_1)$$



Find: $\begin{bmatrix} \dot{x}_{E\cdot E} \\ \dot{y}_{E\cdot E} \end{bmatrix} = ?$

Q: How? **Chain Rule**

$$\dot{X} = \frac{\partial x}{\partial q_1} \cdot \frac{\partial q_1}{\partial t} + \frac{\partial x}{\partial q_2} \cdot \frac{\partial q_2}{\partial t}$$

$\dot{q}_1$ "angular velocity" of joint (1)

$\dot{q}_2$ "Translational velocity" of joint (2)

$$\frac{\partial x}{\partial q_1} = -(L_1 + q_2)\sin(q_1)$$

$$\frac{\partial x}{\partial q_2} = \cos(q_1)$$

$$\dot{x} = \underbrace{-(L_1 + q_2)\sin(q_1)}_{\frac{\partial x}{\partial q_1}}\dot{q}_1 + \underbrace{\cos(q_1)}_{\frac{\partial x}{\partial q_2}}\dot{q}_2$$

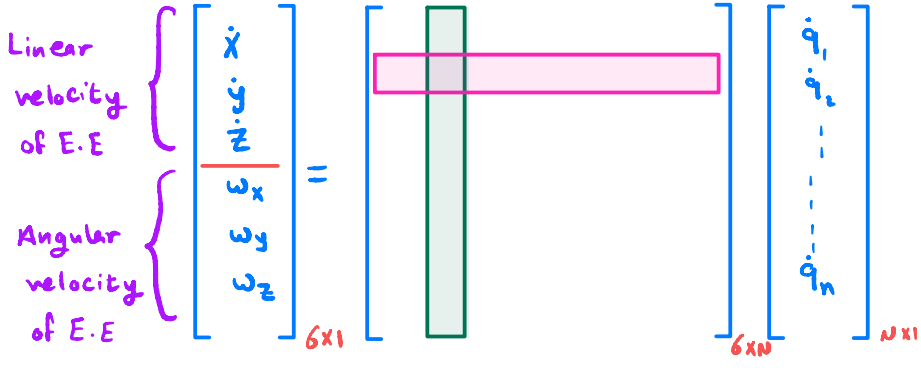
Similarly,

$$\dot{y} = \underbrace{(L_1 + q_2)\cos(q_1)}_{\frac{\partial y}{\partial q_1}}\dot{q}_2 + \underbrace{\sin(q_1)}_{\frac{\partial y}{\partial q_2}}\dot{q}_2$$

Matrix Form

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix}-(L_1+q_2)\cos(q_1)\\ (L_1+q_2)\cos(q_1)\end{bmatrix}}_{J} \underbrace{\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\end{bmatrix}}_{\vec{q}}$$

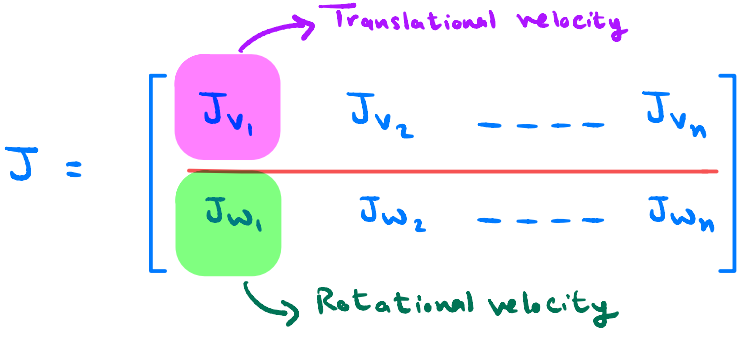
Jacobian Matrix "General Approach"



Each raw: represent the contribution/weights of each joint in the calculation of a specific velocity component of E.E

Each column: represent the contribution of specific joint in all velocity component of E.E

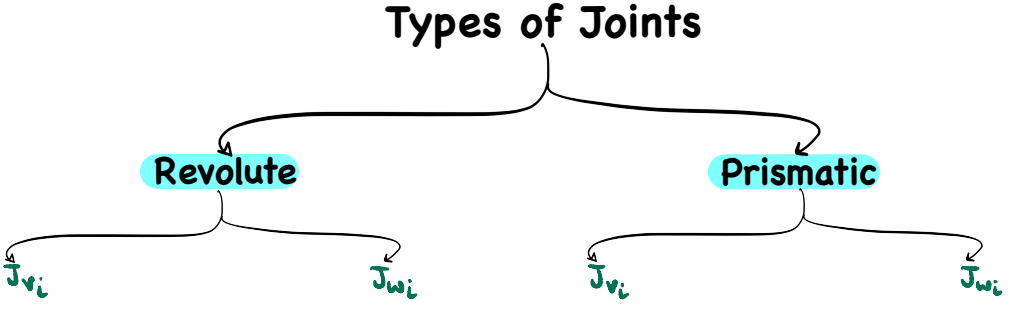
- We can rewrite Jacobian Matrix as follows :



J_{v_i} : describe the effect of the i^{th} joint on translational velocity

J_{w_i} : describe the effect of the i^{th} joint on rotational velocity

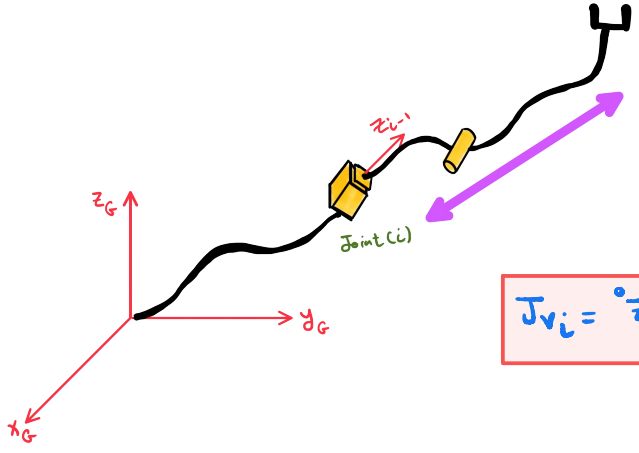
Q: How can we evaluate (Jv_i) & (Jw_i) for any joint based on its type?



=> Prismatic Joint

Note:

- when we investigate joint(i) we assume that the rest of the robot is rigid, and study the effect of this joint solely.



$J_{v_i} = {}^{0}z_{i-1}$ → Direction of velocity due to Ji

Q: what about Jw_i?

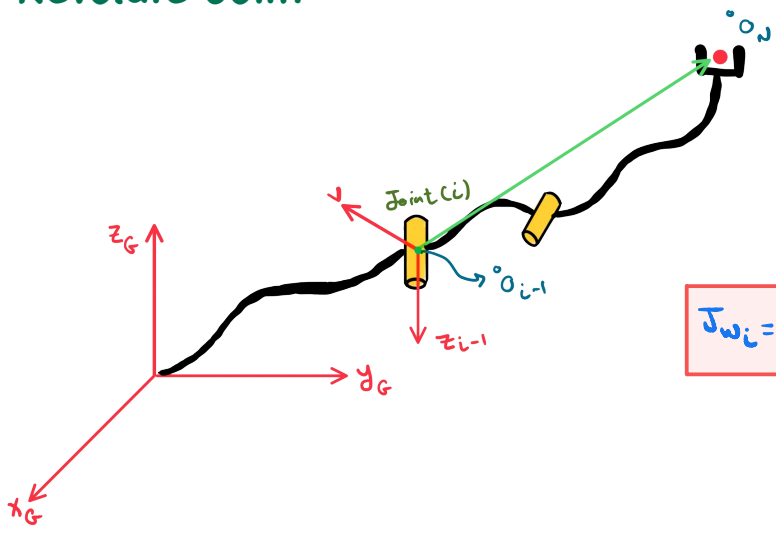
$J_{w_i} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ → Doesn't contribute to rotational velocity

Prismatic

$J_{v_i} = z_{i-1}$

$J_{w_i} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

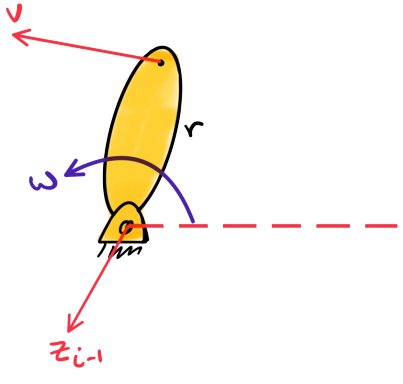
=> Revolute Joint



$$J_{w_i} = z_{i-1}$$

Q: what about Jv_i ?

$$v = w \cdot r$$



Similarly,

$$Jv_i = {}^{\circ}z_{i-1} \times ({}^{\circ}O_N - {}^{\circ}O_{i-1})$$

cross product

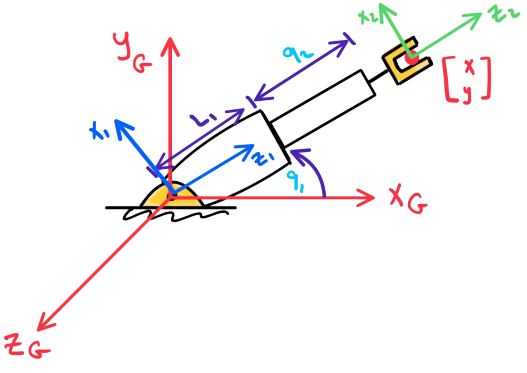
Recall: Dot product between 2-vectors

$$v_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \Rightarrow v_1 \times v_2 = \begin{bmatrix} + & - & + \\ i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{bmatrix} = \begin{bmatrix} y_1 z_2 - z_1 y_2 \\ -(x_1 z_2 - z_1 x_2) \\ x_1 y_2 - y_1 x_2 \end{bmatrix}$$

Types of Joints

- **Revolute**
  - $J_{v_i} = {}^0z_{i-1} \times ({}^0O_N - {}^0O_{i-1})$
  - $J_{\omega_i} = {}^0z_{i-1}$
- **Prismatic**
  - $J_{v_i} = {}^0z_{i-1}$
  - $J_{\omega_i} = [0 \;\; 0 \;\; 0]^T$

Example(1): RP-Robot



- From forward kinematics

$${}^0T_1 = \begin{bmatrix} -sq_1 & 0 & cq_1 & 0 \\ cq_1 & 0 & sq_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ({}^0z_1, \; {}^0O_1)$$

$${}^0T_2 = \begin{bmatrix} -sq_1 & 0 & cq_1 & (L_1+q_2)\,cq_1 \\ cq_1 & 0 & sq_1 & (L_1+q_2)\,sq_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ({}^0z_2, \; {}^0O_2)$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix}_{6\times 1} = \left[\begin{array}{cc} J_{v_1} & J_{v_2} \\ \hline J_{\omega_1} & J_{\omega_2} \end{array}\right]_{6\times 2} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}_{2\times 1}$$

I) For joint(2) -> prismatic

$$J_{\omega_2} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$ Doesn't contribute in rotational velocity

$$J_{v_2} = {}^0z_{i-1} = {}^0z_1 = \begin{bmatrix} c q_1 \\ s q_1 \\ 0 \end{bmatrix}$$

II) For joint(1) -> revolute

$$J_{\omega_1} = {}^0z_{i-1} = {}^0z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{v_1} = {}^0z_{i-1} \times ({}^0O_N - {}^0O_{i-1})$$

$$= {}^0z_0 \times ({}^0O_2 - {}^0O_0)$$

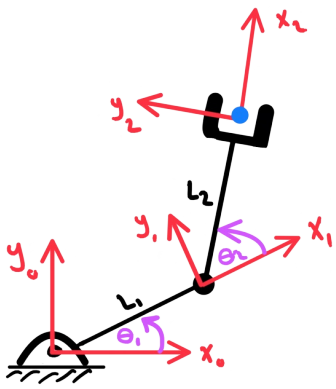
$$= \begin{bmatrix} + & - & + \\ \hat{i} & \hat{j} & \hat{k} \\ 0 & 0 & 1 \\ (L_1+q_2)c_1 & (L_1+q_2)s_1 & 0 \end{bmatrix} = \begin{bmatrix} -(L_1+q_2)s_1 \\ (L_1+q_2)c_1 \\ 0 \end{bmatrix}$$

Putting all together

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \hline \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{6\times1} = \begin{bmatrix} -(L_1+q_2)c_1 & c_1 \\ (L_1+q_2)s_1 & s_1 \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}_{6\times2} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}_{2\times1}$$

J_{v_1}, J_{v_2}, J_{ω_1}, J_{ω_2}

Example(2): RR-Robot



Method(1): Diffrentiation

- From forward kinematics

$$x_{E\cdot E} = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$y_{E\cdot E} = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

- Using chain rule

$$\dot{x}_{E\cdot E} = \frac{\partial x}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial x}{\partial \theta_2}\dot{\theta}_2$$

$$\dot{y}_{E\cdot E} = \frac{\partial y}{\partial \theta_1}\dot{\theta}_1 + \frac{\partial y}{\partial \theta_2}\dot{\theta}_2$$

- Computing partial derivative

$$\frac{\partial x}{\partial \theta_1} = -L s_1 - L_2 s_{12} \qquad \frac{\partial y}{\partial \theta_1} = L_1 c_1 + L_2 c_{12}$$

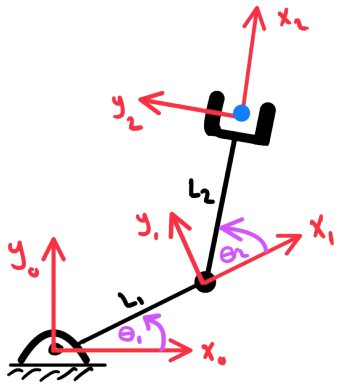
$$\frac{\partial x}{\partial \theta_2} = -L_2 s_{12} \qquad \frac{\partial y}{\partial \theta_2} = L_2 c_{12}$$

- Putting all together

$$\begin{bmatrix} \dot{x}_{E\cdot E} \\ \dot{y}_{E\cdot E} \end{bmatrix} = \left[\begin{array}{cc} -L_1 s_1 - L_2 s_{12} & -L_2 s_{12} \\ L_1 c_1 + L_2 c_{12} & L_2 c_{12} \end{array}\right] \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

Method(2): Using General Approach

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \hline \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{6\times1} = \begin{bmatrix} J_{v_1}{}_{(3\times1)} & J_{v_2}{}_{(3\times1)} \\ \hline J_{w_1}{}_{(3\times1)} & J_{w_2}{}_{(3\times1)} \end{bmatrix}_{6\times2} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}_{2\times1}$$



- From forward kinematics

$$^0T_1 = \left[\begin{array}{c|c|c|c} C_1 & -S_1 & 0 & L_1C_1 \\ S_1 & C_1 & 0 & L_1S_1 \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$^0T_2 = \left[\begin{array}{c|c|c|c} -C_{12} & S_{12} & 0 & L_1C_1+L_2C_{12} \\ S_{12} & C_{12} & 0 & L_1S_1+L_2S_{12} \\ \hline 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

I) For joint(1) -> revolute

$$J_{w_1} = {}^0Z_{i-1} = {}^0Z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{v_1} = {}^0Z_{i-1} \times ({}^0O_N - {}^0O_{i-1})$$

$$= {}^0Z_0 \times ({}^0O_2 - {}^0O_0)$$

$$= \begin{vmatrix} + & - & + \\ i & j & k \\ 0 & 0 & 1 \\ L_1C_1+L_2C_{12} & L_1S_1+L_2S_{12} & 0 \end{vmatrix} = \begin{bmatrix} -L_1S_1 - L_2S_{12} \\ L_1C_1 + L_2C_{12} \\ 0 \end{bmatrix}$$

II) For joint(2) -> revolute

$$J\omega_2 = {}^0z_{i-1} = {}^0z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_{v_2} = {}^0z_{i-1} \times ({}^0O_N - {}^0O_{i-1})$$

$$= {}^0z_1 \times ({}^0O_2 - {}^0O_0)$$

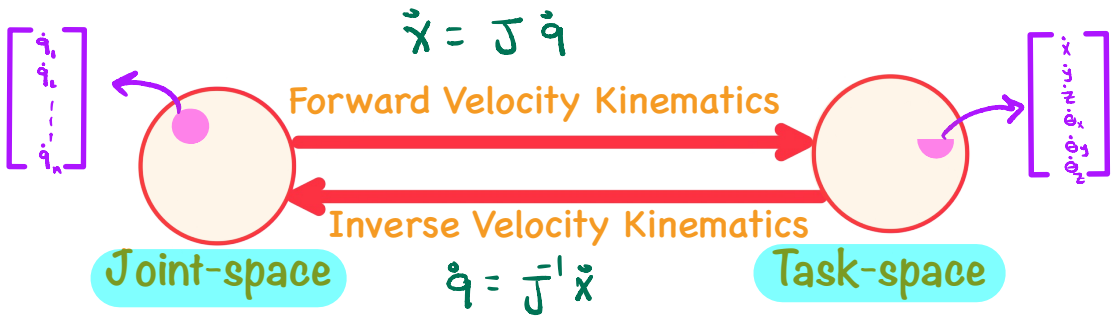
$$= \begin{bmatrix} + & - & + \\ \hat{i} & \hat{j} & \hat{k} \\ 0 & 0 & 1 \\ L_2 C_{12} & L_2 S_{12} & 0 \end{bmatrix} = \begin{bmatrix} -L_2 S_{12} \\ L_2 C_{12} \\ 0 \end{bmatrix}$$

- Putting all together

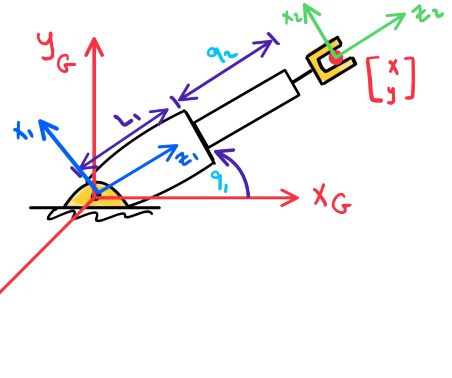
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \hline \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}_{6\times1} = \left[\begin{array}{cc} -L_1 S_1 - L_2 S_{12} & -L_2 S_{12} \\ L_1 C_1 + L_2 C_{12} & L_2 C_{12} \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{array}\right]_{6\times2} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}_{2\times1}$$

Inverse Velocity Kinematics

The map between Joint-space & Task-space:



Example: RP-Robot



- Forward velocity kinematics

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}}_{\dot{\vec{X}}} = \underbrace{\left[\begin{array}{c|c} -(L_1+q_2)s_1 & c_1 \\ (L_1+q_2)c_1 & s_1 \end{array}\right]}_{J} \underbrace{\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}}_{\dot{q}}$$

- Inverse velocity kinematics

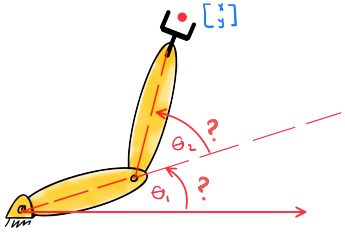
$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \left[\begin{array}{c|c} -(L_1+q_2)s_1 & c_1 \\ (L_1+q_2)c_1 & s_1 \end{array}\right]^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$

Q: What if the Jacobian has no inverse ?

Cases of no Jacobian inverse:

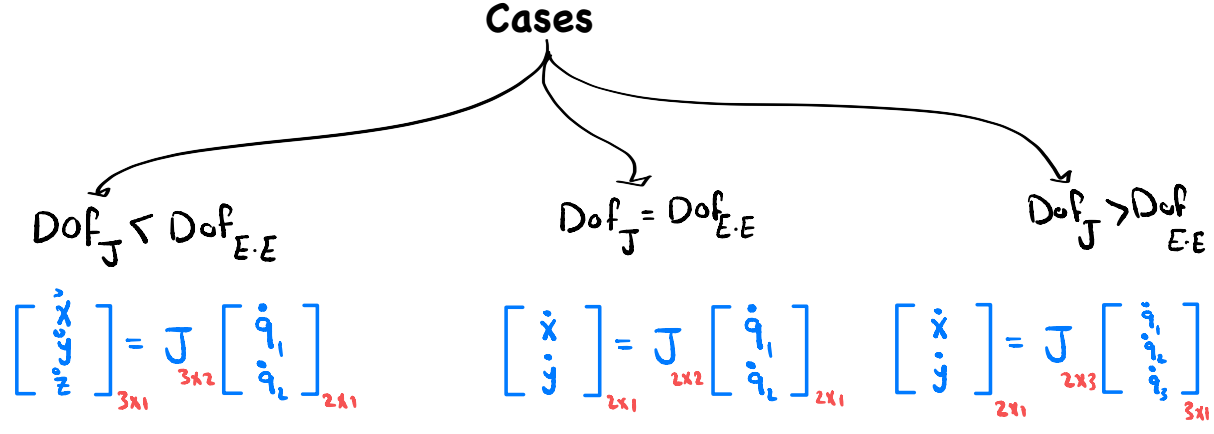
1- det(J) = 0

$$\left[\begin{array}{c|c} -L_1 S_1 - L_2 S_{12} & -L_2 S_{12} \\ L_1 C_1 + L_2 C_{12} & L_2 C_{12} \end{array}\right]$$



Let's $\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ➡ det(J)=0 ➡ Singularity

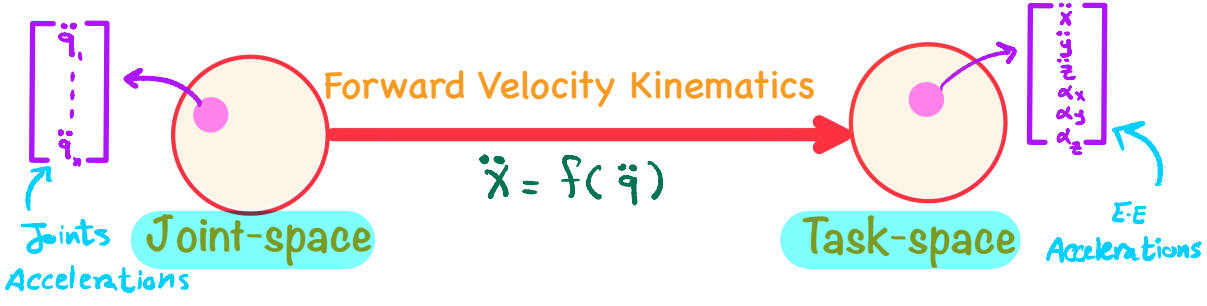
2- According to DOF of Joint space & Task space



Under-Actuated	Exact-Actuated	Over-Actuated
J is (M X N), M > N	J is (N X N)	J is (M X N), M < N
Pseudo-inverse	Inverse	Pseudo-inverse
$J^{\dagger} = (J^T J)^{-1} J^T$	J^{-1}	$J^{\dagger} = J^T (J J^T)^{-1}$

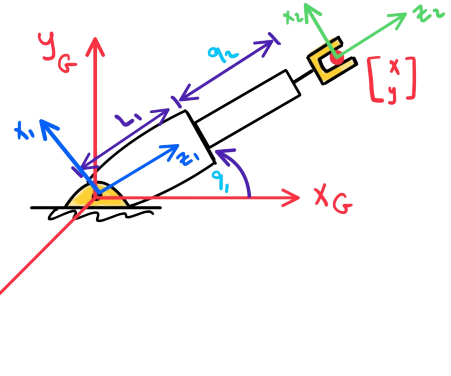
Acceleration Kinematics

The map between Joint-space & Task-space:



Example: RP-Robot

- Forward velocity kinematics



$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}}_{\dot{X}} = \underbrace{\begin{bmatrix} -(L_1+q_2)s_1 & c_1 \\ (L_1+q_2)c_1 & s_1 \end{bmatrix}}_{J} \underbrace{\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}}_{\dot{q}}$$

Given: $\dot{X} = J\dot{q}$

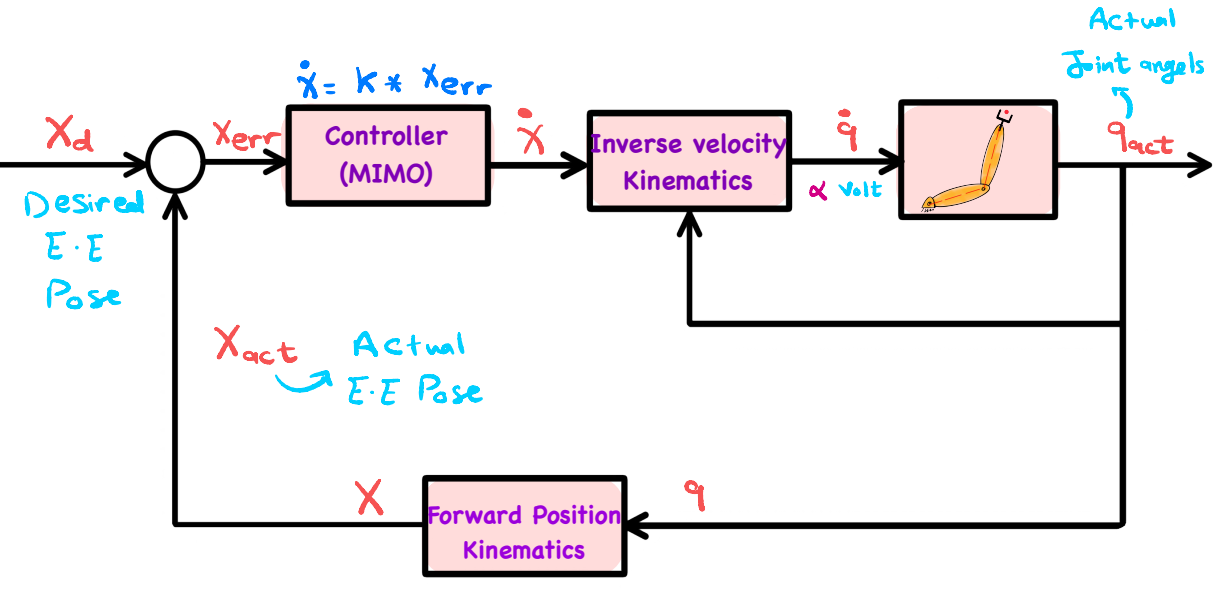
Then $\ddot{X} = J\ddot{q} + \dot{J}\dot{q}$

Inverse Acceleration Solution

Given: $\ddot{X} = J\ddot{q} + \dot{J}\dot{q}$

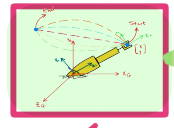
Then: $\ddot{q} = J^{-1}(\ddot{X} - \dot{J}\dot{q})$

Putting All Together



Trajectory Planning

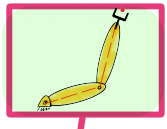
The Big Picture

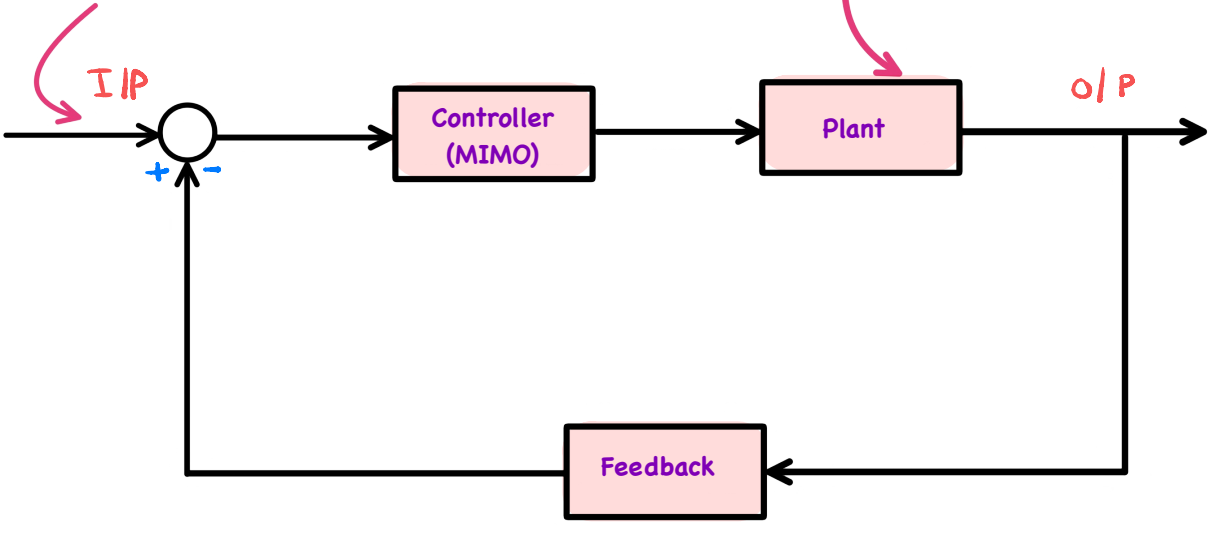


Task-space

Joint-space

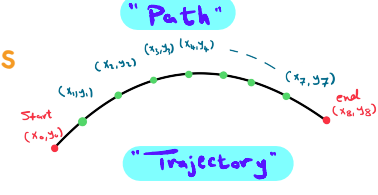
Mathematical Model → Kinematics, Dynamics



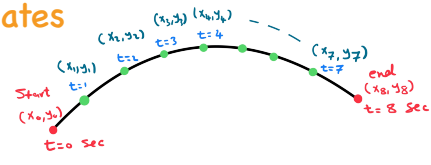


Q: what is the difference between "path" and "trajectory" planning ?

Path Planning: find collision free points/coordinates from start to goal some **without taking time into consideration.**

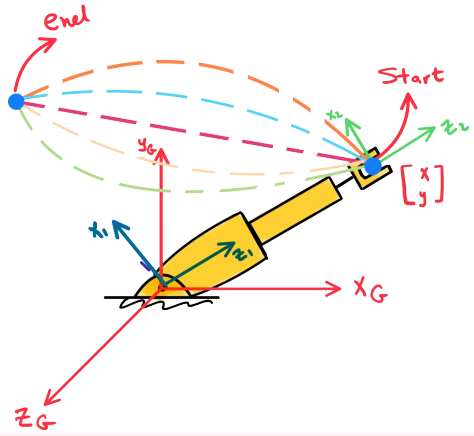


Trajectory Planning: follow some points/coordinates with **each point coupled with time stamp.**



Q:what is the effect of time?

- Introduce more complexity due to the need of discretisation of time and motion



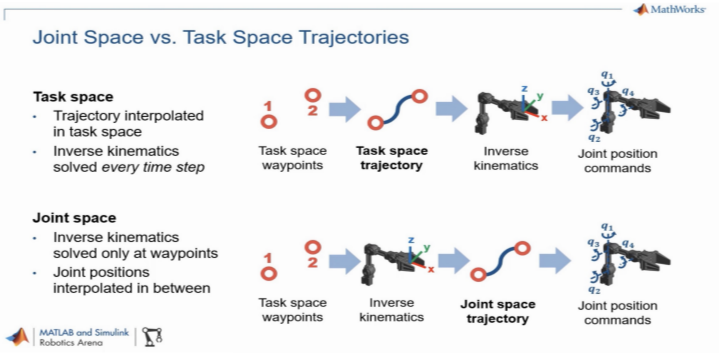
Sampling Time (Ts)

- Inverse Kinematics
- Inverse Velocity Kinematics

...

Minimum Time needed to complete all steps/calculations to move the robot from one pose to another's

Types of Trajectory Planning



1) Task-space Planning

- we want the end-effector to follow a specific path in a specific duration

=> Applications

1- Welding

2- Assembly

3- Painting

2) Joint-space Planning

- we don't care about the end-effector path just move from A -> B smoothly

=> Applications

1- Pick&Place

Steps to solve Task-Space Trajectory Planning

Given:

1- start point
2- destination point
3- duration

Do the following :

1- Decompose the trajectory to equations (functions in time) representing (x,y,z)
2- calculate inverse kinematics to compute joint values at each time step

Steps to solve Joint-Space Trajectory Planning

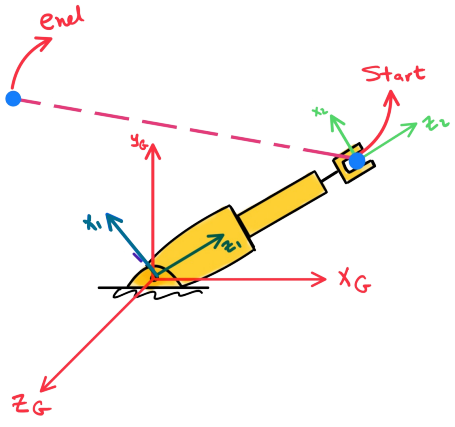
Given:

1- start point
2- destination point
3- duration

Do the following :

1- calculate values of joints at start & goal points using inverse kinematics
2- define interpolation equation for each joint

Example: RP-Robot (Task-space Planning)



Given:

- start point : $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$
- destination point : $\begin{bmatrix} -2 \\ 2,5 \end{bmatrix}$
- duration : 10 sec

Find:

- Joint angels that make the E.E move in **straight line** from start to goal in the required duration

Solution

- Start with x-axis (2 -> -2)

$$X(t) = x_0 + \alpha_x \cdot t$$

α : Factor by which we will change the value of x(t) as time grows

$$\alpha_x = \frac{X(t) - X(0)}{t}$$

(goal → $X(t)$, initial → $X(0)$, t → duration)

$$\alpha_x = \frac{-2 - 2}{10} = -,4$$

$$X(t) = 2 - ,4\ t$$

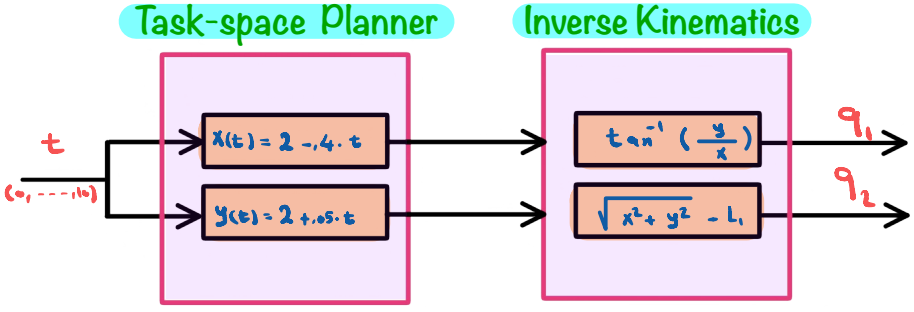
- For y-axis (2 -> 2.5)

$$y(t) = y_0 + \alpha_y \cdot t$$

$$\alpha_y = \frac{2,5 - 2}{10} = ,05$$

$$y(t) = 2 + ,05 \cdot t$$

- Use table to compute joint velocities at each time step



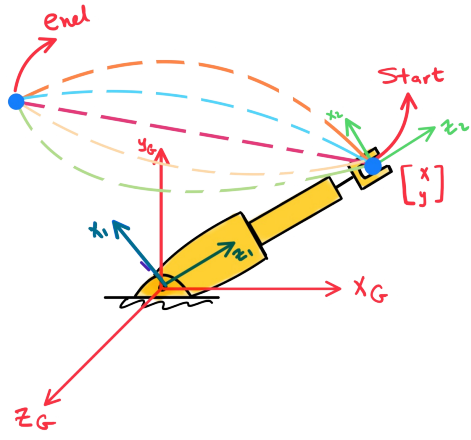
time	x(t)	y(t)	θ_1	θ_2
0	2	2	45	2,3
1	1,6	2,05		
⋮				
10	-2	2,5		

Solution / values of joints to follow the path in 10 sec

Example: RP-Robot (Joint-space Planning)

Given:

- start point : $\begin{bmatrix} 2 \\ 2 \end{bmatrix}$
- destination point : $\begin{bmatrix} -2 \\ 1.5 \end{bmatrix}$
- duration : 10 sec



Find:

- Joint angels that will move the robot from start to goal **smoothly**

Q: Why we want position and orientation to vary smoothly with time ?

1) reduce the peak acceleration of the robot
2) reduce the size of the motors

Q: But how we guarantee smoothness ?

1) position $q(t)$ is continuous
2) velocity $\dot{q}(t)$ is continuous
1) acceleration $\ddot{q}(t)$ is continuous
2) jerk $\dddot{q}(t)$ is continuous

Solution

- Let's use polynomial function that allow us to put conditions/constrains on position,velocity,acceleration on start and destination

Fifth-order Polynomial

- has 6-coefficients which allows us to specify 6 boundary conditions (position,velocity,acceleration) at t = 0 and t = T

$$q(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3 + a_4 \cdot t^4 + a_5 \cdot t^5$$

Time	Position	Velocity	Acceleration
t = 0	$q(0)$	$\dot{q}(0)$	$\ddot{q}(0)$
t = T	$q(T)$	$\dot{q}(T)$	$\ddot{q}(T)$

Third-order Polynomial

- for simplicity we will use 3rd-order polynomial. Always remember that we need to balance between smoothness and execution time

- has 4-coefficients which allows us to specify 4 boundary conditions (position,velocity,acceleration) at t = 0 and t = T

$$q(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3$$ (Joint (1))

Time	Position	Velocity
t = 0	$q(0)$	$\dot{q}(0)$
t = T	$q(T)$	$\dot{q}(T)$

$$q(t) = a_0 + a_1 \cdot t + a_2 \cdot t^2 + a_3 \cdot t^3$$

$$\dot{q}(t) = a_1 + 2a_2 \cdot t + 3a_3 \cdot t^2$$

Q: But how calculate the coefficients ?

at $t=0$

$q(0) = a_0 = 45$

$\dot{q}(0) = a_1 = 0$

at $t=10$

$q(10) = 45 + a_2(10)^2 + a_3(10)^3$

$\dot{q}(10) = 2a_2(10) + 3a_3(10)^2$

$a_2 = 2,509$

$a_3 = -,167$

- equation of joint(1)

$$q(t) = 45 + 2,50 \cdot t - ,167 \cdot t^2$$