KTH ROYAL INSTITUTE
OF TECHNOLOGY

Introduction to Robotics

DD2410 - Introduction to Robotics

Lecture 4 - Differential Kinematics & Dynamics

Schedule - Lectures

Sep 02 - 1. Intro, Course fundamentals, Topics, What is a Robot, History, Applications.

Sep 03 - 2 ROS Introduction (Scheduled as lab, will be in zoom only)

Sep 03 - 3 Manipulators, Kinematics
Sep 06 - 4. Differential kinematics, dynamics
Sep 08 - 5. Actuators, sensors I (force, torque, encoders, ...)
Sep 13 - 6. Grasping, Motion, Control

Sep 15 - 7. Planning (RRT, A*, ...)
Sep 20 - 8. Behavior Trees and Task Switching

Sep 22 - 9. Mobility and sensing II (distance, vision, radio, GPS, ...)
Sep 27 - 10. Localisation (where are we?)
Sep 29 - 11. Mapping (how to build the map to localise/navigate w.r.t.?)
Oct 04 - 12. Navigation (how do I get from A to B?)

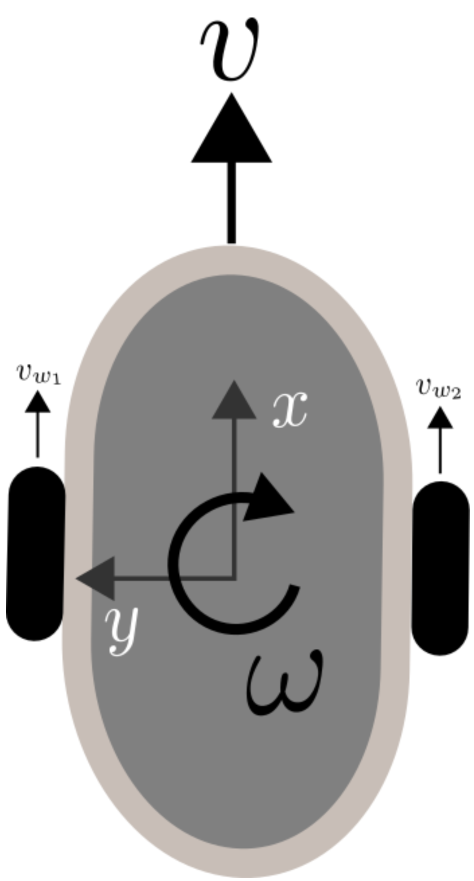
Oct 06 - Q/A - Open questions to your teachers.

Overview

- Differential kinematics
  - Jacobians
  - Singularities
  - Manipulability
  - Calculations
- Dynamics
  - Forces and accelerations
  - algorithms for calculations

- For many operations, we are not interested in the stationary kinematics, but rather the differential kinematics, mainly for the mapping between velocities in configuration space and cartesian space

Differential kinematics - Vacuum cleaner type



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$v = \frac{v_{w_1} + v_{w_2}}{2}$$

$$\omega = \frac{v_{w_2} - v_{w_1}}{2b}$$

$$v_{w_i} = \frac{2\pi r f \Delta_{\text{enc}}}{\text{ticks per rev}}$$

- The instantaneous transform between velocities in robot configuration space and cartesian space is given by the Jacobian:

$$\dot{X} = J(\Theta)\dot{\Theta}$$

- Where each element j_{mn} in J is defined as $\frac{\partial K(\Theta)_m}{\partial \Theta_n}$

Differential kinematics

- The instantaneous transform between velocities in robot configuration space and cartesian space is given by the Jacobian:

$$\dot{X} = J(\Theta)\dot{\Theta}$$

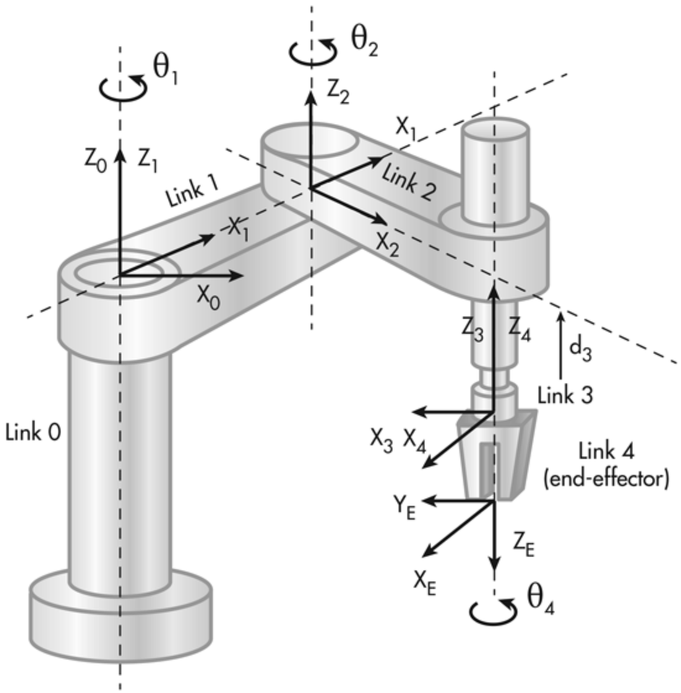
- Where each element j_{mn} in J is defined as $\dfrac{\partial K(\Theta)_m}{\partial \Theta_n}$

- Transform ${}^{0}\mathbf{T}_E$ from end effector to base frame is dependant on configuration **Θ**
- The function that generates the end effector pose **X** given **Θ,** is called **forward kinematics, K**

$$\mathbf{X} = \mathbf{K}(\boldsymbol{\Theta}), \qquad \mathbf{r} = {}^{0}\mathbf{T}_E \mathbf{p}_E$$

where **p** is the position of the endpoint in the last frame

- Commonly, we define **K**(**Θ**) to output the pose vector **X** = [x y z α β γ]^T, where α β γ are the *Euler Angles*



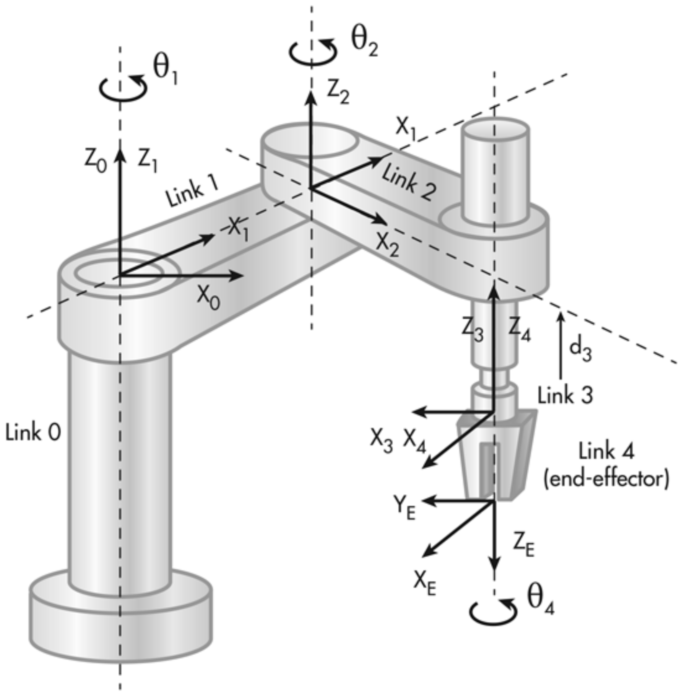
Forward kinematics

- Transform ${}^0\mathbf{T}_E$ from end effector to base frame is dependant on configuration **Θ**
- The function that generates the end effector pose **X** given **Θ,** is called **forward kinematics, K**

$$\mathbf{X} = \mathbf{K}(\mathbf{\Theta}), \qquad \mathbf{r} = {}^0\mathbf{T}_E\mathbf{p}_E$$

where **p** is the position of the endpoint in the last frame

- Commonly, we define **K**(**Θ**) to output the pose vector **X** = [x y z α β γ]^T, where α β γ are the *Euler Angles*



$${}^0\mathbf{T}_E = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$$

See R-MPC 2.4

- The instantaneous transform between velocities in robot configuration space and cartesian space is given by the Jacobian:

$$\dot{X} = J(\Theta)\dot{\Theta}$$

- Where each element j_{mn} in J is defined as $\frac{\partial K(\Theta)_m}{\partial \theta_n}$

- Thus, each column in J can be seen as the vector ΔX_i, or the motion in X caused by motion in the joint θ_i.

- The closed form of a typical manipulator Jacobian is not printable

- The closed form of a typical manipulator Jacobian is not printable

The Puma 560 can be seen in Figures 1 and 2.

The forward kinematics K_f can be formulated as:

$$\mathbf{X} = K_f(\boldsymbol{\Theta}) \tag{1}$$

where

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ z \\ p \\ t \\ a \end{bmatrix}, \qquad \boldsymbol{\Theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \tag{2}$$

we have:

$$\begin{aligned} x &= cos(\theta_1) * [a_2 cos(\theta_2) + a_3 cos(\theta_2+\theta_3) - d_4 sin(\theta_2+\theta_3)] - d_3 sin(\theta_1) \\ y &= sin(\theta_1) * [a_2 cos(\theta_2) + a_3 cos(\theta_2+\theta_3) - d_4 sin(\theta_2+\theta_3)] + d_3 cos(\theta_1) \\ z &= -a_3 sin(\theta_2+\theta_3) + a_2 sin(\theta_2) - d_4 cos(\theta_2+\theta_3) \\ p &= tan^{-1}\left(\frac{s1(c23c4s5+s23c5)-c1s4s5}{c1(c23c4s5+s23c5)+s1s4s5}\right) \\ t &= tan^{-1}\left(\frac{-s1(c23c4s5+s23c5)+c1s4s5}{sin(tan^{-1}(-s1(c23c4s5+s23c5)+c1s4s5/-c1(c23c4s5+s23c5)-s1s4s5))(s23c4s5-c23c5)}\right) \\ a &= tan^{-1}\left(\frac{-(s23(s4c6-c4c5s6)+c23s5s6)}{s23(s4s6-c4c5c6)-c23s5c6}\right) \end{aligned} \tag{3}$$

Where the latter uses shorthand. The full expression is:

$$\begin{aligned} p &= tan^{-1}\left(\frac{sin(\theta_1)(cos(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)+sin(\theta_2+\theta_3)cos(\theta_5))-cos(\theta_1)sin(\theta_4)sin(\theta_5)}{cos(\theta_1)(cos(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)+sin(\theta_2+\theta_3)cos(\theta_5))+sin(\theta_1)sin(\theta_4)sin(\theta_5)}\right) \\ t &= tan^{-1}\left(\frac{-sin(\theta_1)(cos(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)+sin(\theta_2+\theta_3)cos(\theta_5))+cos(\theta_1)sin(\theta_4)sin(\theta_5)}{sin\left(tan^{-1}\left(\frac{-sin(\theta_1)(cos(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)+sin(\theta_2+\theta_3)cos(\theta_5))+cos(\theta_1)sin(\theta_4)sin(\theta_5)}{-cos(\theta_1)(cos(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)+sin(\theta_2+\theta_3)cos(\theta_5))-sin(\theta_1)sin(\theta_4)sin(\theta_5)}\right)\right)(sin(\theta_2+\theta_3)cos(\theta_4)sin(\theta_5)-cos(\theta_2+\theta_3)cos(\theta_5))}\right) \\ a &= tan^{-1}\left(\frac{-(sin(\theta_2+\theta_3)(sin(\theta_4)cos(\theta_6)-cos(\theta_4)cos(\theta_5)sin(\theta_6))+cos(\theta_2+\theta_3)sin(\theta_5)sin(\theta_6))}{sin(\theta_2+\theta_3)(sin(\theta_4)sin(\theta_6)-cos(\theta_4)cos(\theta_5)cos(\theta_6))-cos(\theta_2+\theta_3)sin(\theta_5)cos(\theta_6)}\right) \end{aligned} \tag{4}$$

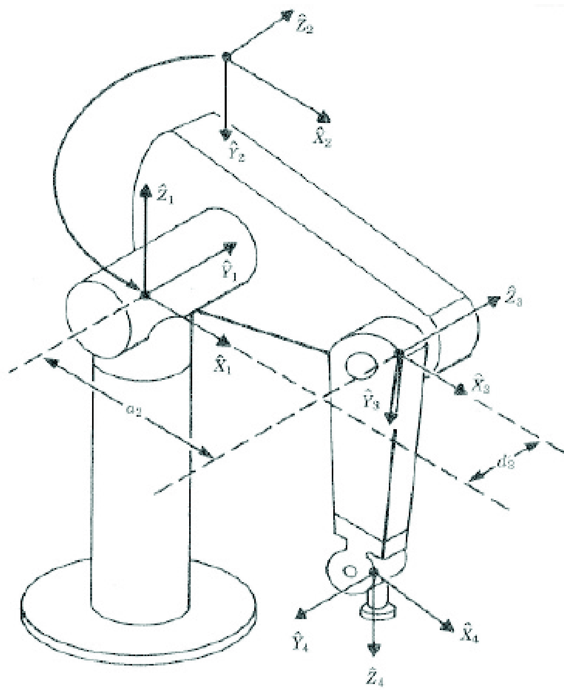
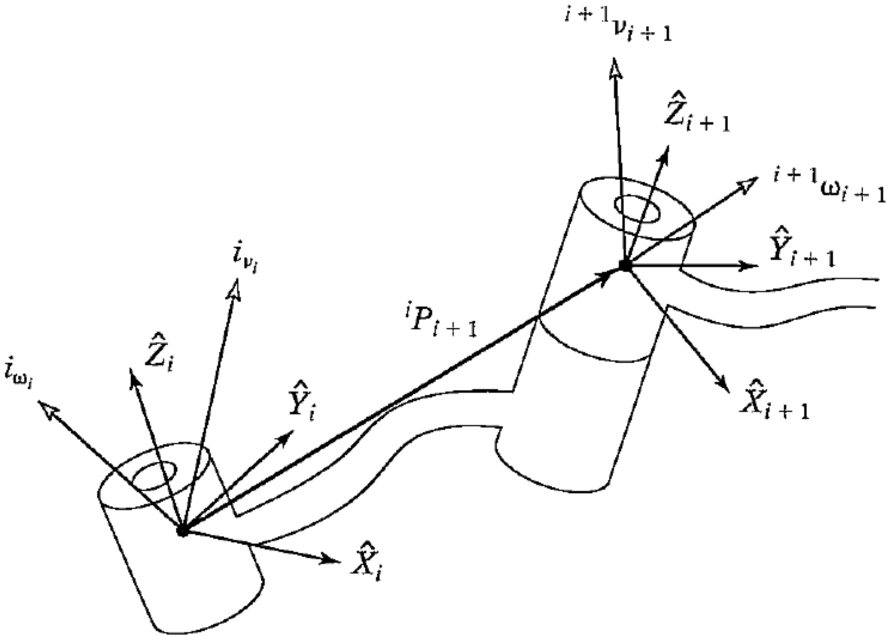


Figure 1: The puma 56

- The closed form of a typical manipulator Jacobian is often not printable, but can be derived by sequential application of frame transforms
- The motion of frame *i+1*, is a function of the motion of frame *i* and the motion of the joint between them.

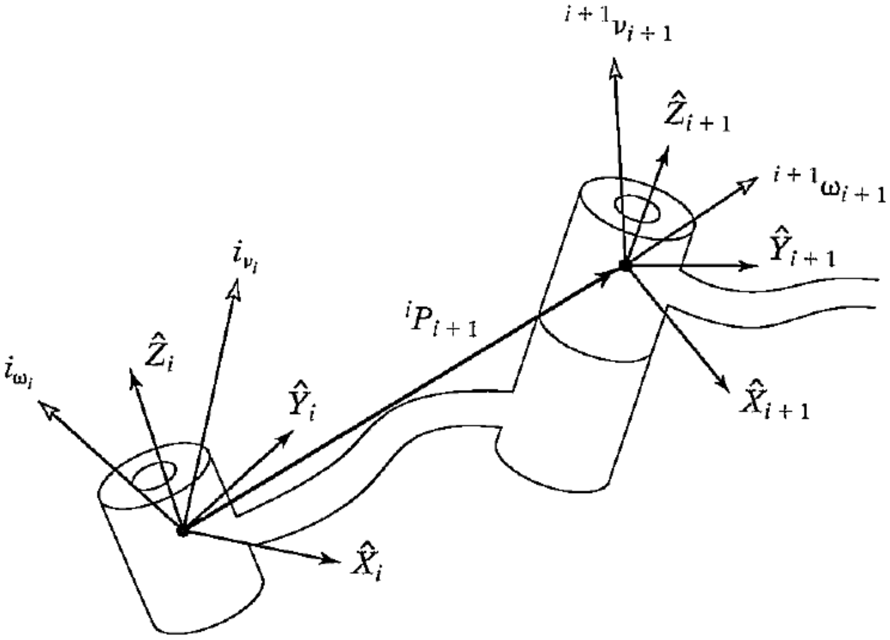
Differential kinematics (R-MPC chapter 3): Rotational joints



$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{\ i}R\, {}^{i}\omega_i + \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1} $$

$$ {}^{i+1}\upsilon_{i+1} = {}^{i+1}_{\ i}R({}^{i}\upsilon_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) $$

Differential kinematics (R-MPC chapter 3) - Prismatic joints

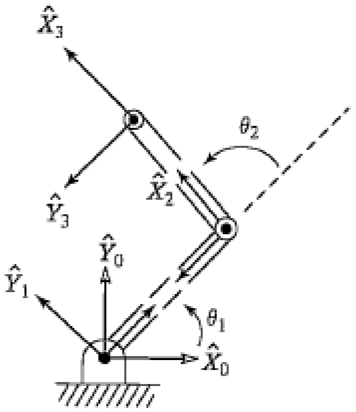


$$
{}^{i+1}\omega_{i+1} = {}^{i+1}_{\;\;i}R \, {}^{i}\omega_i ,
$$

$$
{}^{i+1}\upsilon_{i+1} = {}^{i+1}_{\;\;i}R({}^{i}\upsilon_i + {}^{i}\omega_i \times {}^{i}P_{i+1}) + \dot{d}_{i+1} \, {}^{i+1}\hat{Z}_{i+1}
$$

- Consequetive application of link transforms gives us velocities in end effector frame
- Note: resulting velocities are multilinear in joint velocities!
- Multiplying by rotation transform ${}^{B}R_{E}$ gives us velocities in base frame
- Thus we can derive $J(\Theta)$

Example: Planar robot



$$ {}^1\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}, $$

$$ {}^1v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, $$

$$ {}^2\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}, $$

$$ {}^2v_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{bmatrix}, $$

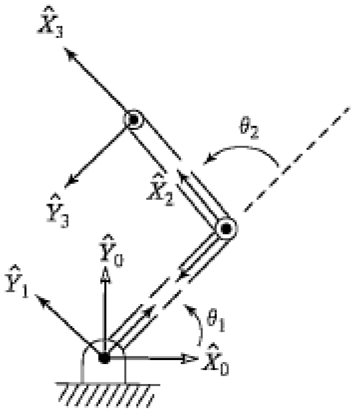
$$ {}^3\omega_3 = {}^2\omega_2, $$

$$ {}^3v_3 = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 + l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}. $$

$$ {}^0_3R = {}^0_1R \quad {}^1_2R \quad {}^2_3R = \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ {}^0v_3 = \begin{bmatrix} -l_1 s_1 \dot{\theta}_1 - l_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ l_1 c_1 \dot{\theta}_1 + l_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}. $$

Example: Planar robot



$$^1\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix},$$

$$^1v_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

$$^2\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix},$$

$$^2v_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{bmatrix},$$

$$^3\omega_3 = {}^2\omega_2,$$

$$^3v_3 = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 + l_2(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}.$$

$$^0_3R = {}^0_1R \quad {}^1_2R \quad {}^2_3R = \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$^0v_3 = \begin{bmatrix} -l_1 s_1 \dot{\theta}_1 - l_2 s_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ l_1 c_1 \dot{\theta}_1 + l_2 c_{12}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}.$$

$$^0J(\Theta) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}$$

Jacobians (R-MPC 3.1.3)

$$\dot{\mathbf{X}} = \mathbf{J}(\Theta)\,\dot{\Theta}$$

- Column j_i in **J** is the contribution of the i:th joint to the velocity of the end effector.
- Each column in **J** can be computed individually.

Jacobians (R-MPC 3.1.3)

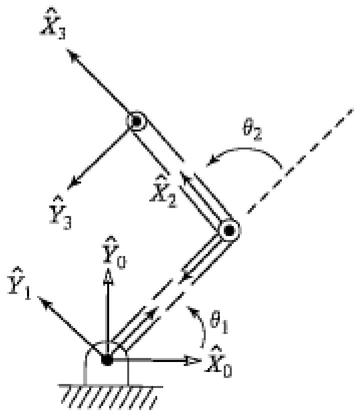
$$\dot{\mathbf{X}} = \mathbf{J}(\Theta)\,\dot{\Theta}$$

- Column j_i in **J** is the contribution of the i:th joint to the velocity of the end effector.

- Each column in **J** can be computed individually.

Assignment 2: second part

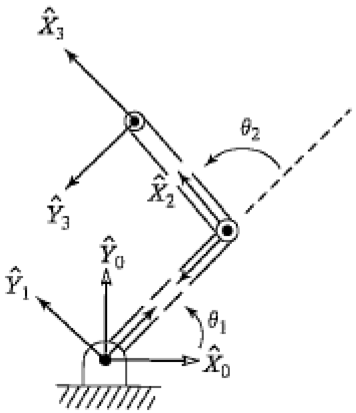
Example: Planar robot



$$
{}^0J(\Theta) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}
$$

What happens if both angles are 0?

Example: Planar robot



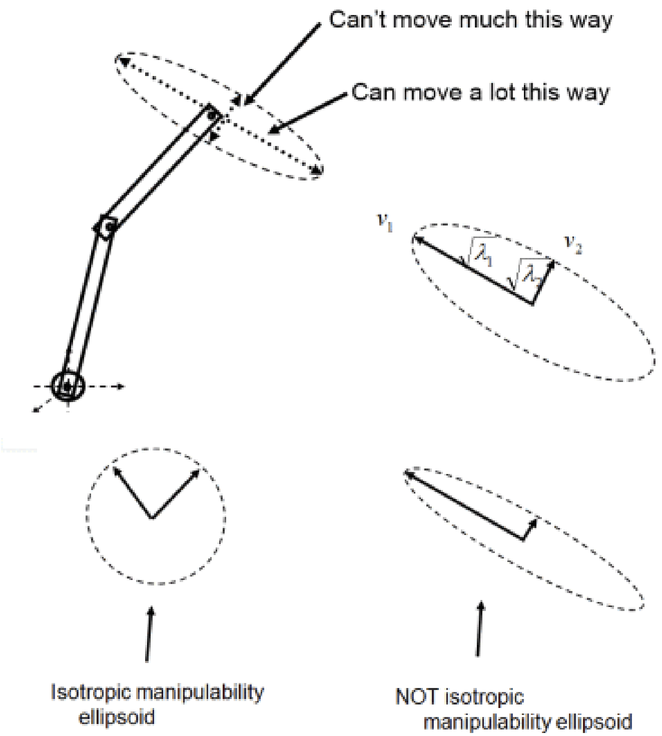
$$\dot{X} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

What happens if both angles are 0?

When the Jacobian loses rank, we get a kinematic singularity - we lose the ability to generate motion in some direction!

Manipulability

- We can generalize this into a concept of manipulability w

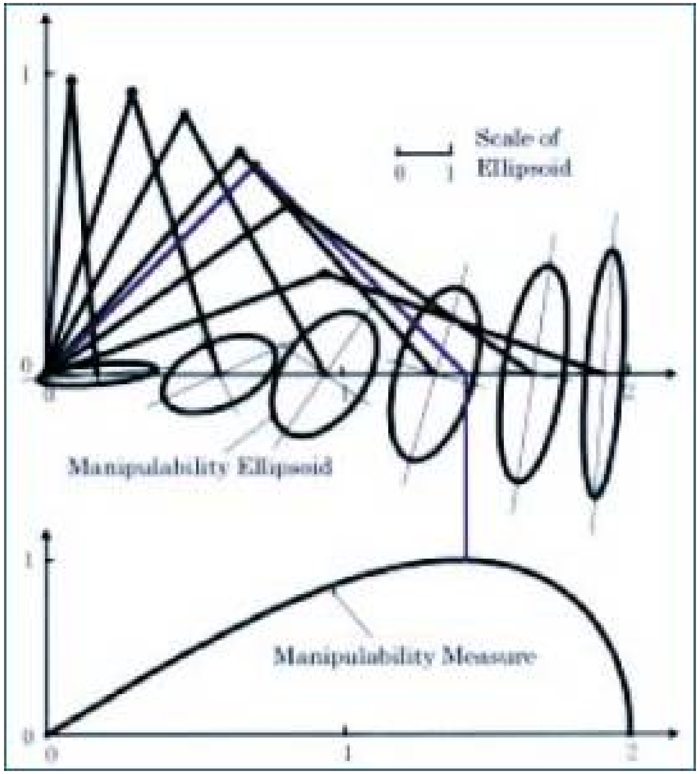


$$w = \sqrt{det(JJ^T)}$$

w is proportional to the volume of the *manipulability ellipsoid*.

Manipulability

- We can generalize this into a concept of manipulability w



$$w = \sqrt{det\left(JJ^T\right)}$$

w is proportional to the volume of the *manipulability ellipsoid*.

Manipulability example

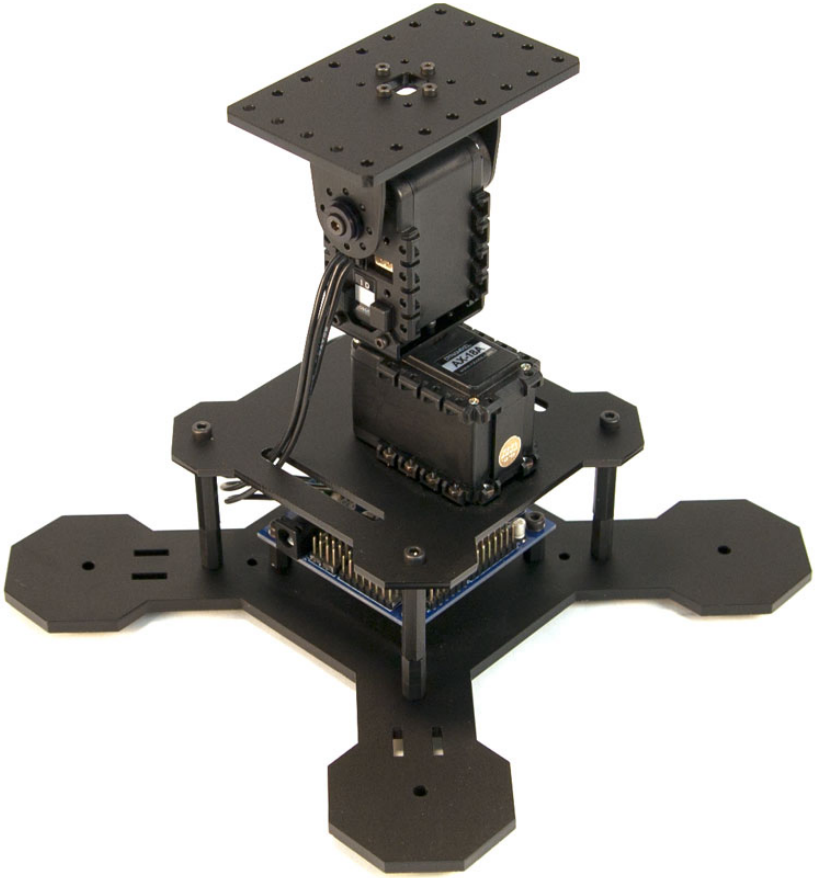


Image: Trossen Robotics

Jacobians

- The inverse Jacobian is trivial to calculate, as long as the Jacobian matrix is invertible.

- If J is not invertible, we can often use pseudo-inverse instead.

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We start with an approximation

$$\hat{\Theta} = \Theta + \epsilon_{\Theta}$$

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We start with an approximation

$$\hat{\Theta} = \Theta + \epsilon_{\Theta}$$

$$X + \epsilon_X = K(\Theta + \epsilon_{\Theta})$$

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We start with an approximation

$$\hat{\Theta} = \Theta + \epsilon_{\Theta}$$

$$X + \epsilon_X = K(\Theta + \epsilon_{\Theta})$$

$$K(\Theta) + \epsilon_X = K(\Theta + \epsilon_{\Theta})$$

Jacobians for numerical inverse kinematics

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We start with an approximation

$$\hat{\Theta} = \Theta + \epsilon_{\Theta}$$

$$X + \epsilon_X = K(\Theta + \epsilon_{\Theta})$$

$$K(\Theta) + \epsilon_X = K(\Theta + \epsilon_{\Theta})$$

- With linear approximation, we get

$$\epsilon_X \approx J(\Theta)\,\epsilon_{\Theta}$$

Jacobians for numerical inverse kinematics

- We want to find the inverse kinematics

$$\Theta = K^{-1}(X)$$

- We start with an approximation

$$\hat{\Theta} = \Theta + \epsilon_\Theta$$

$$X + \epsilon_X = K(\Theta + \epsilon_\Theta)$$

$$K(\Theta) + \epsilon_X = K(\Theta + \epsilon_\Theta)$$

- With linear approximation, we get (assuming invertible J)

$$\epsilon_X \approx J(\Theta)\,\epsilon_\Theta$$

$$\epsilon_\Theta \approx J^{-1}(\Theta)\,\epsilon_X$$

- **Algorithm for finding inverse kinematics**

  Given target **X** and initial approximation $\hat{\Theta}$

- **Algorithm for finding inverse kinematics**

  Given target **X** and initial approximation $\widehat{\Theta}$

  repeat

  $$\widehat{X} = K\left(\widehat{\Theta}\right)$$

  until $\epsilon_X \leqslant tolerance$

- **Algorithm for finding inverse kinematics**

Given target **X** and initial approximation $\hat{\Theta}$

repeat

$$\hat{X} = K\left(\hat{\Theta}\right)$$
$$\epsilon_X = \hat{X} - X$$

until $\epsilon_X \leqslant tolerance$

- **Algorithm for finding inverse kinematics**

  Given target **X** and initial approximation $\widehat{\Theta}$

  repeat

  $$\widehat{X} = K\left(\widehat{\Theta}\right)$$
  $$\epsilon_X = \widehat{X} - X$$
  $$\epsilon_\Theta = J^{-1}\left(\widehat{\Theta}\right)\epsilon_x$$

  until $\epsilon_X \leqslant tolerance$

- **Algorithm for finding inverse kinematics**

  Given target **X** and initial approximation $\hat{\Theta}$

  repeat

$$\hat{X} = K\left(\hat{\Theta}\right)$$
$$\epsilon_X = \hat{X} - X$$
$$\epsilon_\Theta = J^{-1}\left(\hat{\Theta}\right)\epsilon_x$$
$$\hat{\Theta} = \hat{\Theta} - \epsilon_\Theta$$

  until $\epsilon_X \leqslant tolerance$

Jacobians for static forces

- Virtual work must be same independent of coordinates

$$\mathcal{F}^T \delta\chi = \tau^T \delta\Theta$$

- We remember that:

$$\delta\chi = J\delta\Theta$$

- Which gives us:

$$\mathcal{F}^T J = \tau^T$$

$$\tau = J^T \mathcal{F}$$

Jacobians for static forces

$$\tau = J^T \mathcal{F}$$

- We can now see that for singular configurations, there will be directions where the required torque for a given force goes to zero, or inversely, **the forces generated by a given torque tend to infinity**. This may cause damage to the robot or the environment.

Jacobians for static forces

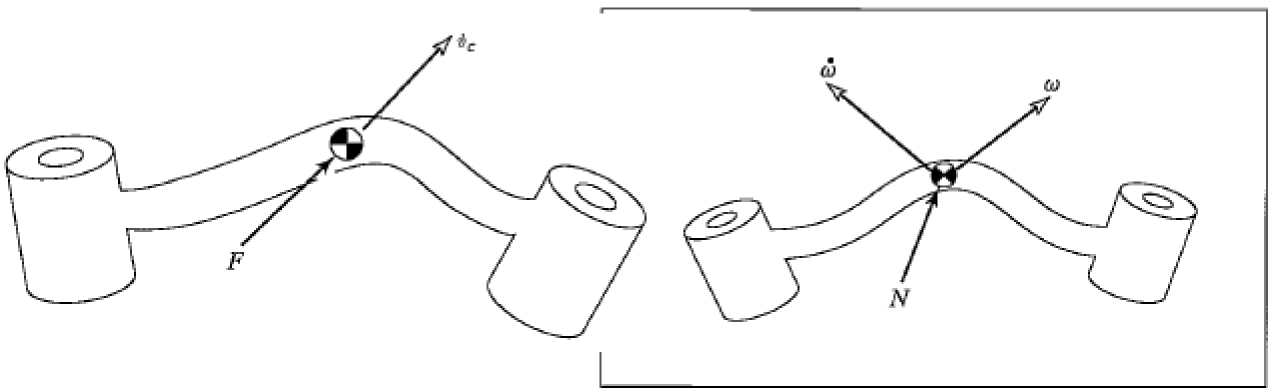
$$\tau = J^T \mathcal{F}$$

- We can also calculate inverse kinematics by virtual forces and torques. We apply a "force" correcting the end effector position, calculate the torques this would generate, and move the robot accordingly. This gives us the update step:

$$\epsilon_{\Theta} = J^T\left(\widehat{\Theta}\right)\epsilon_x$$

- This is useful when inverse of J does not exist, but typically converges slower.

Dynamics (R-MPC Chapter 7)



$$ {}^{A}I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}, $$

$$ I_{xx} = \iiint_V (y^2 + z^2)\rho dv, $$

$$ I_{yy} = \iiint_V (x^2 + z^2)\rho dv, $$

$$ I_{zz} = \iiint_V (x^2 + y^2)\rho dv, $$

$$ I_{xy} = \iiint_V xy\rho dv, $$

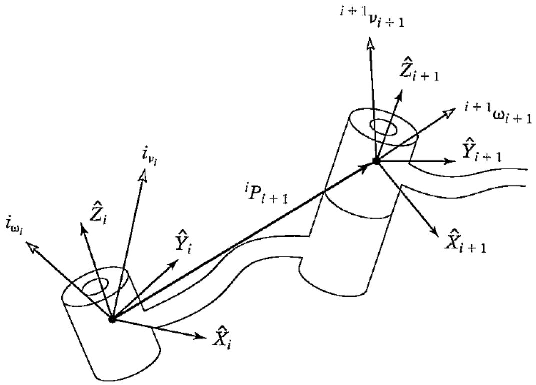
$$ I_{xz} = \iiint_V xz\rho dv, $$

$$ I_{yz} = \iiint_V yz\rho dv, $$

$$ F = m\dot{v}_C, $$

$$ N = {}^{C}I\dot{\omega} + \omega \times {}^{C}I\omega, $$

Dynamics (R- MPC chapter 7) - Rotational joints



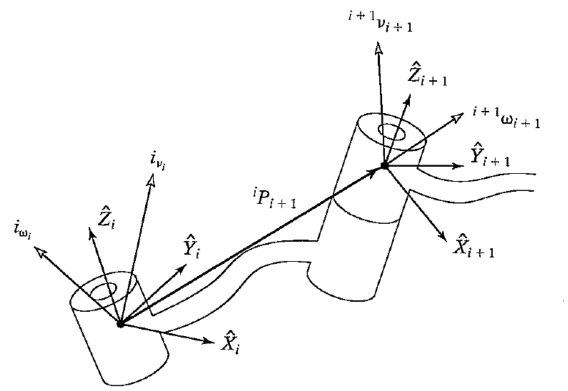
$$^{i+1}\omega_{i+1} = {}^{i+1}_{\ \ i}R\, {}^{i}\omega_i + \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1}$$

$$^{i+1}v_{i+1} = {}^{i+1}_{\ \ i}R({}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1})$$

$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{\ \ i}R\, {}^{i}\dot{\omega}_i + {}^{i+1}_{\ \ i}R\, {}^{i}\omega_i \times \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1}.$$

$$^{i+1}\dot{v}_{i+1} = {}^{i+1}_{\ \ i}R[{}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i]$$

Dynamics (R-MPC chapter 7) - Prismatic joints



$${}^{i+1}\omega_{i+1} = {}^{i+1}_{\ \ i}R\,{}^{i}\omega_{i},$$

$${}^{i+1}v_{i+1} = {}^{i+1}_{\ \ i}R({}^{i}v_{i} + {}^{i}\omega_{i} \times {}^{i}P_{i+1}) + \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}$$

$${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{\ \ i}R\,{}^{i}\omega_{i}.$$

$$\begin{aligned}{}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{\ \ i}R({}^{i}\dot{\omega}_{i} \times {}^{i}P_{i+1} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_{i}) \\ + 2\,{}^{i+1}\omega_{i+1} \times \dot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{d}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}\end{aligned}$$

$${}^{i}\dot{v}_{C_i} = {}^{i}\dot{\omega}_{i} \times {}^{i}P_{C_i} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} + {}^{i}P_{C_i}) + {}^{i}\dot{v}_{i}$$

Dynamics

Newton - Euler approach:

- Find the acceleration and velocity of each joint, working outwards
- Find the necessary torque/force to generate that acceleration, adding the external forces and torques, working inwards

Dynamics

Outward iterations: $i : 0 \to 5$

$$^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\, {}^{i}\omega_i + \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1},$$

$$^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\, {}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\, {}^{i}\omega_i \times \dot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\, {}^{i+1}\hat{Z}_{i+1},$$

$$^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i),$$

$$\begin{aligned} {}^{i+1}\dot{v}_{C_{i+1}} &= {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} \\ &\quad + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1}, \end{aligned}$$

$$^{i+1}F_{i+1} = m_{i+1}\, {}^{i+1}\dot{v}_{C_{i+1}},$$

$$^{i+1}N_{i+1} = {}^{C_{i+1}}I_{i+1}\, {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C_{i+1}}I_{i+1}\, {}^{i+1}\omega_{i+1}.$$

Inward iterations: $i : 6 \to 1$

$$^{i}f_i = {}^{i}_{i+1}R\, {}^{i+1}f_{i+1} + {}^{i}F_i,$$

$$\begin{aligned} {}^{i}n_i &= {}^{i}N_i + {}^{i}_{i+1}R\, {}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i \\ &\quad + {}^{i}P_{i+1} \times {}^{i}_{i+1}R\, {}^{i+1}f_{i+1}, \end{aligned}$$

$$\tau_i = {}^{i}n_i^T\, {}^{i}\hat{Z}_i.$$

Dynamics (R-MPC chapter 7)

The resulting dynamic equations can be written on the form (state-space equation):

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) + J^T f$$

Dynamics (DLR)

$$\tau = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) + J^T f$$

Dynamics (DLR)

Dynamics (DLR)

State of the art - industrial manipulation

Dynamics (DLR)