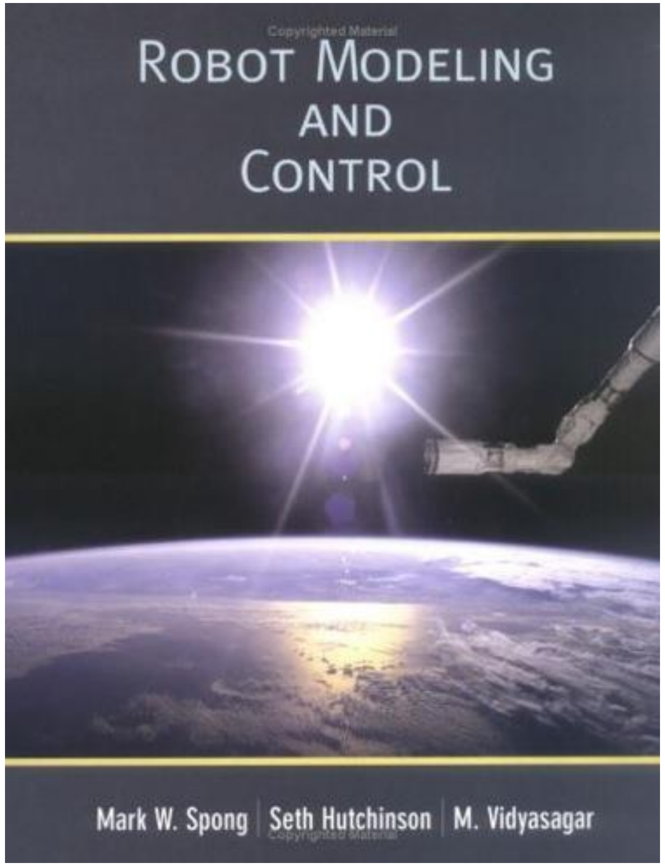
MEAM 520
Lecture 16: Velocity Kinematics

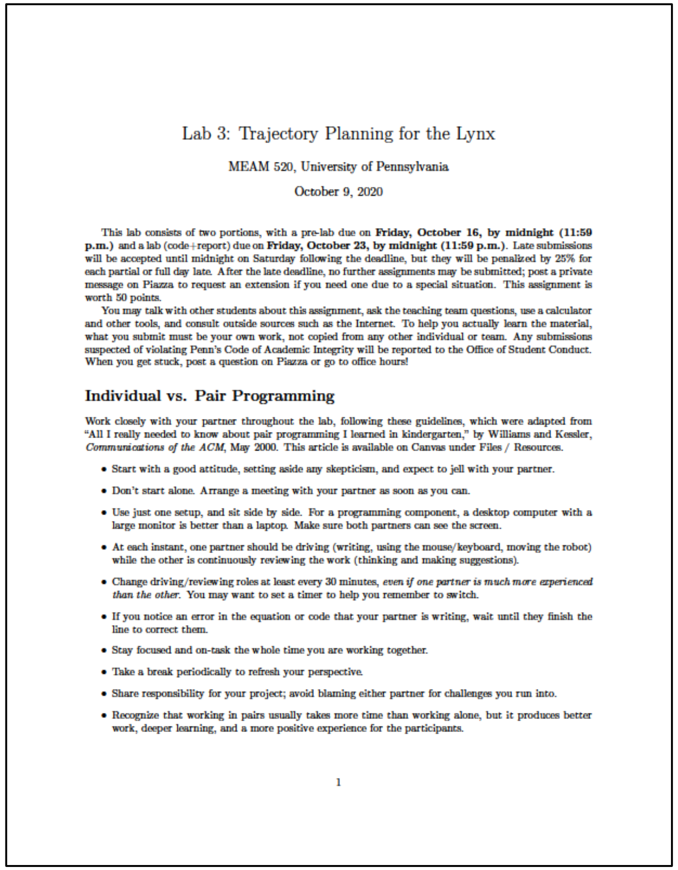
Cynthia Sung, Ph.D.
Mechanical Engineering & Applied Mechanics
University of Pennsylvania

Today: More Velocity Kinematics



Chapter 4: Velocity Kinematics

- Read Sec. 4.6, 4.9, 4.11-4.12



Lab 3: Trajectory Planning for the Lynx

MEAM 520, University of Pennsylvania

October 9, 2020

This lab consists of two portions, with a pre-lab due on **Friday, October 16, by midnight (11:59 p.m.)** and a lab (code+report) due on **Friday, October 23, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation. This assignment is worth 50 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

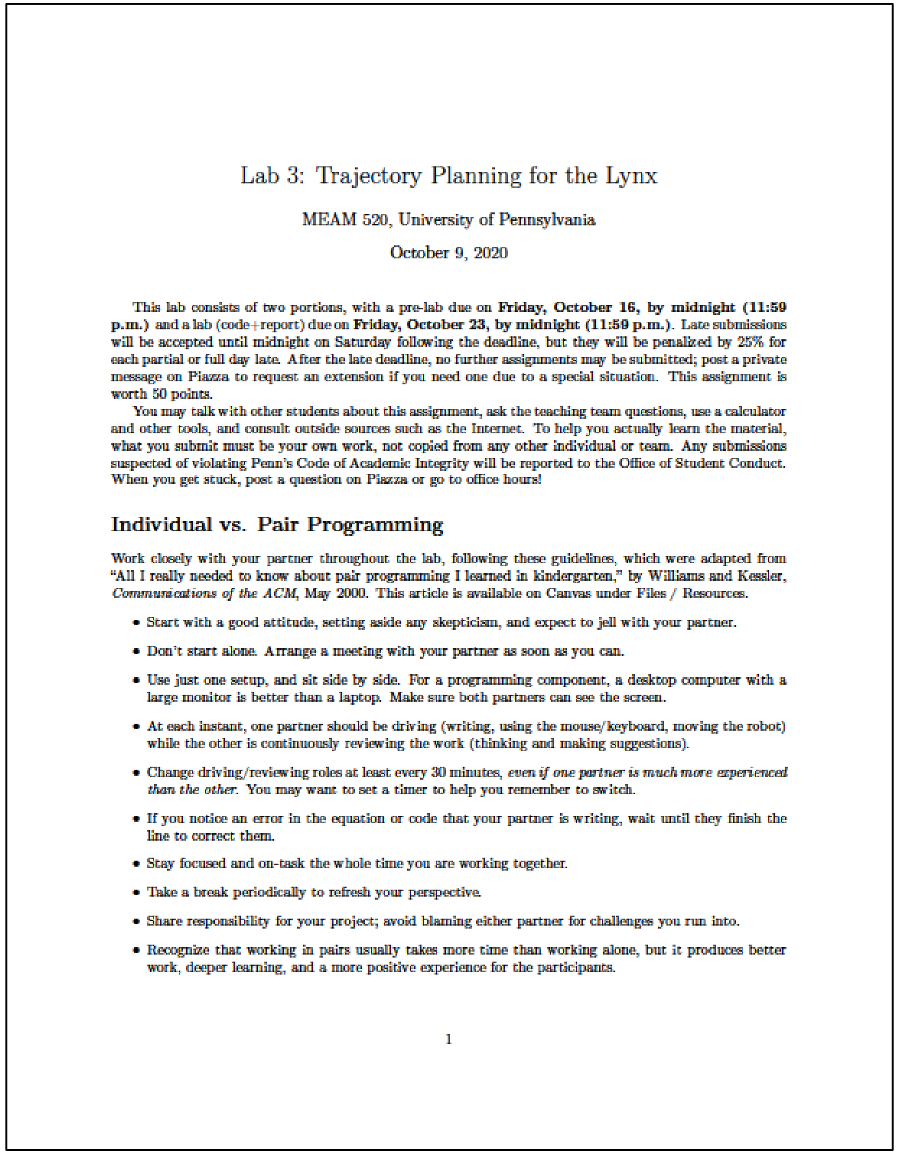
Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 3 due tomorrow

Last Minute Questions on Lab 3?



Lab 3: Trajectory Planning for the Lynx

MEAM 520, University of Pennsylvania

October 9, 2020

This lab consists of two portions, with a pre-lab due on **Friday, October 16, by midnight (11:59 p.m.)** and a lab (code+report) due on **Friday, October 23, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation. This assignment is worth 50 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Important notes:

- All robots are points in configuration space
- Not all robots are points in the workspace/task space
- Search algorithms can be applied to graphs of arbitrary dimension
- Collision checks are often conservative
- The purpose of the lab is for you to make choices. **Explain and evaluate those choices!**

Last Time: Linear Velocity Jacobians

How do the velocities of the joints affect the linear velocity of the end-effector?

$$v_n^0 = J_v \dot{q}$$

(3 x 1) (3 x n) (n x 1)

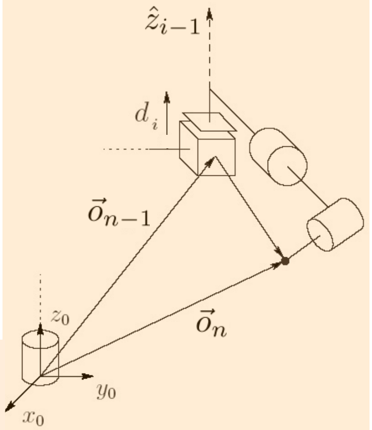
n joints

Two ways to get J_v

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

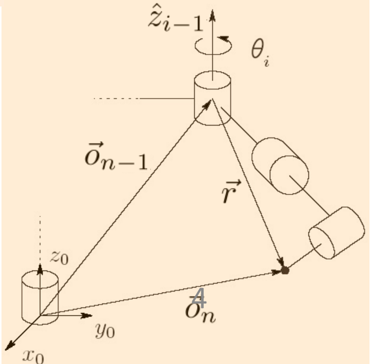
partial derivatives of the tip position with respect to the joint variables

geometric construction of the columns of Jv using the robot's forward kinematics



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

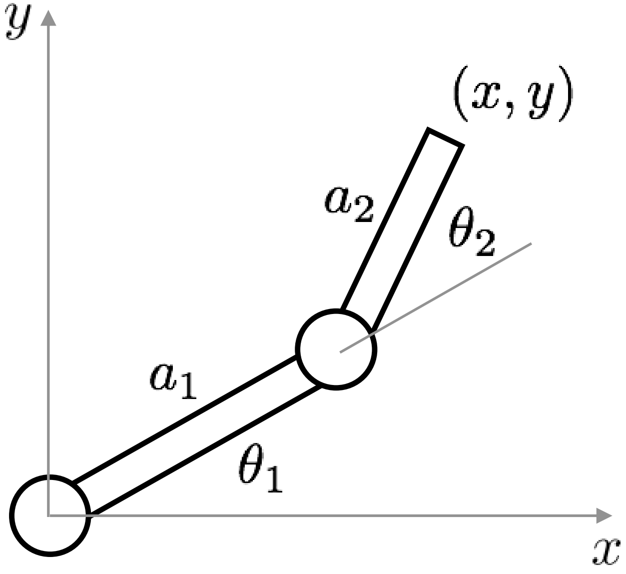
Revolute $J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$



Both methods yield the same Jv matrix

The mapping from joint velocities to the linear and angular velocity of the robot's tip depends on the robot's current pose!

Last Time: Planar RR Example of Partial Derivative Method



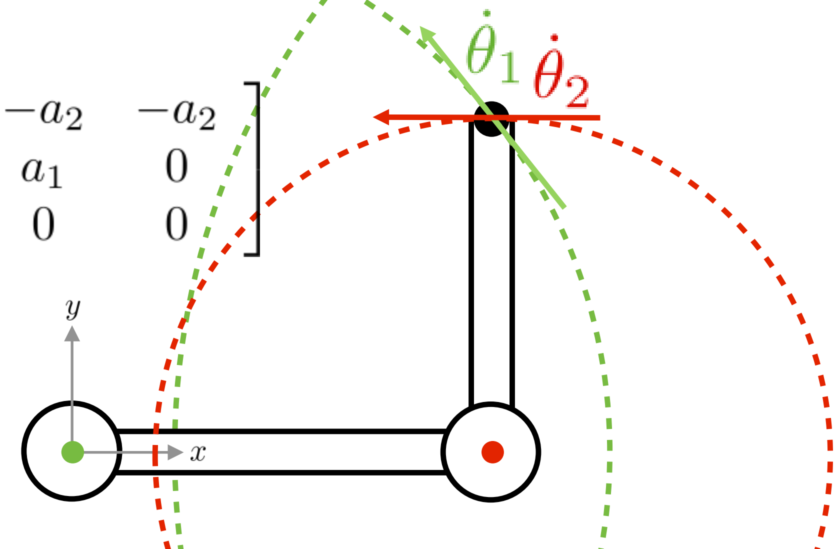
$$d_2^0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_2 c_{12} + a_1 c_1 \\ a_2 s_{12} + a_1 s_1 \\ 0 \end{bmatrix}$$

$$\begin{array}{cc} \frac{\partial}{\partial \theta_1} & \frac{\partial}{\partial \theta_2} \end{array}$$

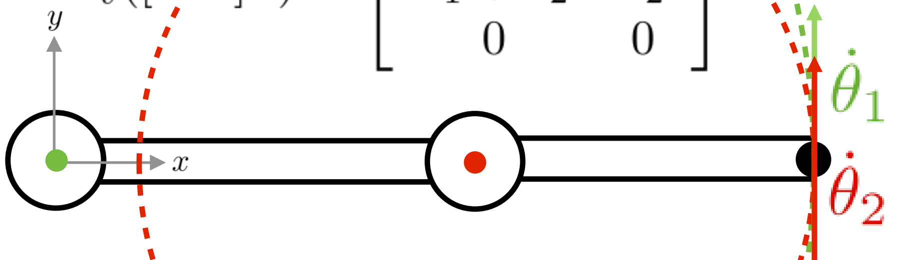
$$J_v(\vec{q}) = \begin{array}{c} \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow x \\ \leftarrow y \\ \leftarrow z \end{array}$$

$$v_n^0 = J_v \dot{q}$$

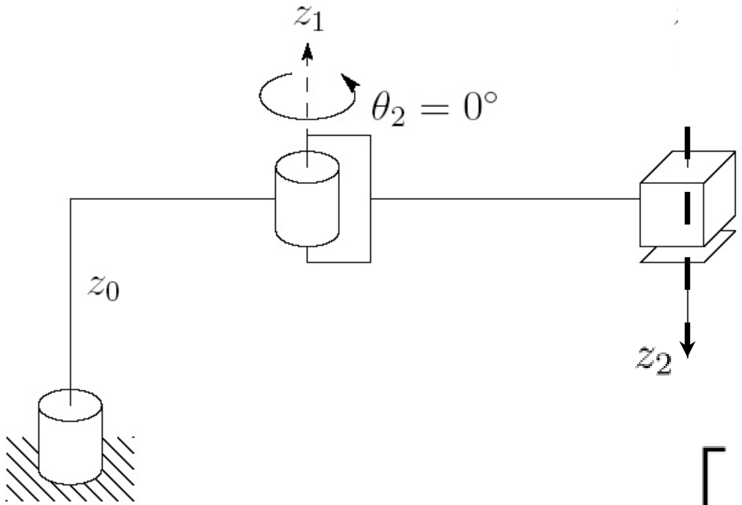
$$J_v([0 \quad \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$



$$J_v([0 \quad 0]^T) = \begin{bmatrix} 0 & 0 \\ a_1 + a_2 & a_2 \\ 0 & 0 \end{bmatrix}$$



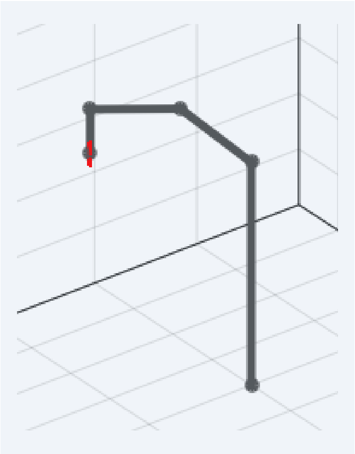
Last Time: SCARA Example of Geometric Method



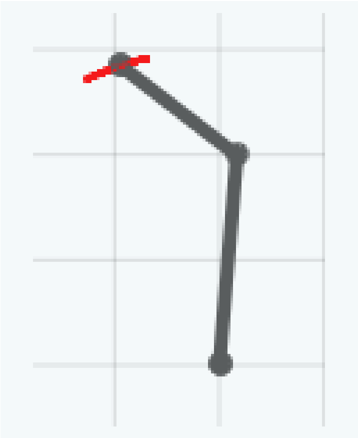
Prismatic $J_{v_i} = z_{i-1}$

Revolute $J_{v_i} = z_{i-1} \times (o_n - o_{i-1})$

express all vectors in frame zero



$$J_v = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$



$i = 1$

revolute

$$J_{v_1} = \hat{z}_0 \times (\vec{o}_3 - \vec{o}_0)$$

$i = 2$

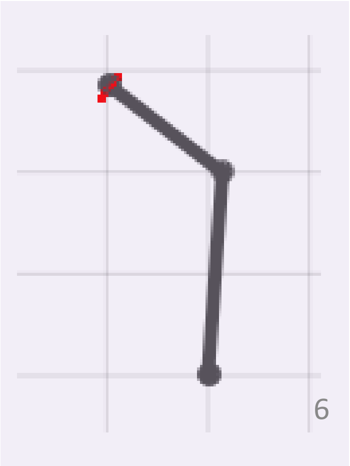
revolute

$$J_{v_2} = \hat{z}_1 \times (\vec{o}_3 - \vec{o}_1)$$

$i = 3$

prismatic

$$J_{v_3} = z_2$$



Angular Velocity Jacobians

$$\vec{\omega}_{0,n}^{0} = J_\omega(\vec{q})\,\dot{\vec{q}}$$

final frame angular velocity (3 x 1)

angular velocity Jacobian, evaluated at the robot's current pose (3 x n)

joint velocities (n x 1)

Prismatic joints **never** cause an angular velocity

Revolute joints **always** cause an angular velocity around the associated (previous) z-axis

$$\omega_{0,n}^{0} = \sum_{i=1}^{n} \left(\mathbf{R}_{i-1}^{0}\hat{z}\right)\dot{\theta}_i$$

$\rho_i =$ 0 for prismatic, 1 for revolute

Prismatic $J_{\omega_i} = 0$

Revolute $J_{\omega_i} = z_{i-1}$

angular velocity notation

$\vec{\omega}_{i,j}^{k}$ the angular velocity of frame **j** with respect to frame **i**, expressed in frame **k**

$$J_\omega(q) = \left[\rho_1\hat{\mathbf{z}} \quad \rho_2\mathbf{R}_1^0\hat{\mathbf{z}} \quad \rho_3\mathbf{R}_2^0\hat{\mathbf{z}} \quad \cdots \quad \rho_n\mathbf{R}_{n-1}^0\hat{\mathbf{z}}\right]$$

(3 x n) linear velocity Jacobian

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

Sometimes roboticists call this just the "Jacobian" because it's the most common one.

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

(3 x n) angular velocity Jacobian

The Jacobian is easily constructed from the manipulator's forward kinematics.

What do you need from the forward kinematics?

4.6.3 Combining the Linear and Angular Velocity Jacobians

As we have seen in the preceding section, the upper half of the Jacobian J_v is given as

$$J_v = [J_{v_1} \cdots J_{v_n}] \tag{4.56}$$

in which the i^{th} column J_{v_i} is

$$J_{v_i} = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \tag{4.57}$$

The lower half of the Jacobian is given as

$$J_\omega = [J_{\omega_1} \cdots J_{\omega_n}] \tag{4.58}$$

in which the i^{th} column J_{ω_i} is

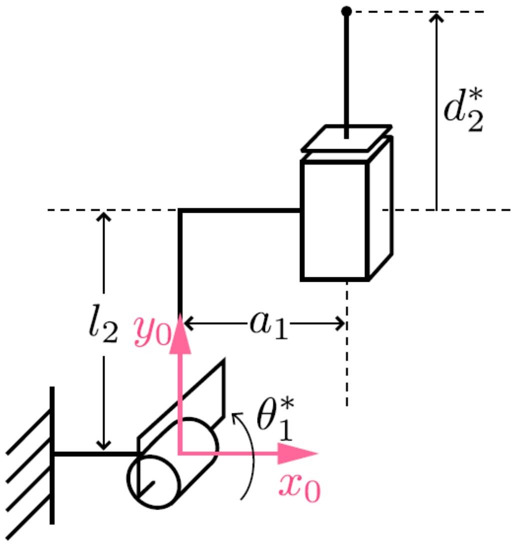
What questions do you have?

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolute joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \tag{4.59}$$

You need the **third column (*z*)** of the homogeneous transformation matrix for all frames except the end-effector, plus the **end-effector frame's origin position** (o_n). If using geometry, you also need **origin positions for all revolute joints (fourth column)**.

$$T_n^0 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Your turn: RP Manipulator with Offset



$$A_1 = H_1^0 = \begin{bmatrix} c_1 & 0 & -s_1 & a_1 c_1 \\ s_1 & 0 & c_1 & a_1 s_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2 = H_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_2 + d_2^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = \begin{bmatrix} c_1 & 0 & -s_1 & a_1 c_1 - (l_2 + d_2^*)\, s_1 \\ s_1 & 0 & c_1 & a_1 s_1 + (l_2 + d_2^*)\, c_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c_1 = \cos\theta_1^*$ and $s_1 = \sin\theta_1^*$

What is this robot's manipulator Jacobian for motion in 3D?

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

(3 x n) linear velocity Jacobian

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

(3 x n) angular velocity Jacobian

$$\xi = J(q)\dot{q}$$

(6 x 1) body velocity

(6 x n) Jacobian

(n x 1) joint velocities

Notice that the body velocity is not the time derivative of a body position vector because of the angular velocity.

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

Analytical Jacobian (SHV 4.8)

Alternative to the Geometric Jacobian: use a different representation for orientation

Instead of calculating the angular velocity of the end-effector's frame, calculate the time derivatives of three values that represent the orientation of the end-effector frame

$$\dot{X} = \begin{bmatrix} \dot{d} \\ \dot{\alpha} \end{bmatrix} = J_a(q)\dot{q}$$

Euler angles are the most commonly used minimal representation.

$$R = R_{z,\psi} R_{y,\theta} R_{z,\phi}$$

Note this is inconsistent with Chapter 2's definition of ZYZ Euler angles...

$$\alpha = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}$$

We won't use the analytical Jacobian in this class, but you may encounter it elsewhere.

$$\omega = \begin{bmatrix} c_\psi s_\theta & -s_\psi & 0 \\ s_\psi s_\theta & c_\psi & 0 \\ c_\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = B(\alpha)\dot{\alpha}$$

$$J_a(q) = \begin{bmatrix} I & 0 \\ 0 & B^{-1}(\alpha) \end{bmatrix} J(q)$$

Summary: Velocity Forward Kinematics

(3 x n) linear velocity Jacobian

$$J = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$$

(6 x n) Jacobian
a.k.a. manipulator Jacobian
a.k.a. geometric Jacobian

(3 x n) angular velocity Jacobian

$$J_v(\vec{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

$$J_\omega(q) = [\rho_1 \hat{\mathbf{z}} \quad \rho_2 \mathbf{R}_1^0 \hat{\mathbf{z}} \quad \rho_3 \mathbf{R}_2^0 \hat{\mathbf{z}} \quad \cdots \quad \rho_n \mathbf{R}_{n-1}^0 \hat{\mathbf{z}}]$$

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}$$

Questions?

A Use for the Linear Velocity Jacobian

$$v_n^0 = J_v \dot{q}$$

What joint velocities should I choose to cause a desired end-effector velocity?
(**inverse velocity kinematics**)

$$\dot{q} = J_v^{-1} v_n^0$$

Can a robot always achieve all end-effector velocities?

No. This works only when the Jacobian is square and invertible (non-singular).

Position Singularities

Singularities are points in the configuration space where infinitesimal motion in a certain direction is not possible and the manipulator loses one or more degrees of freedom

$$\dot{q} = J_v^{-1} v_n^0$$

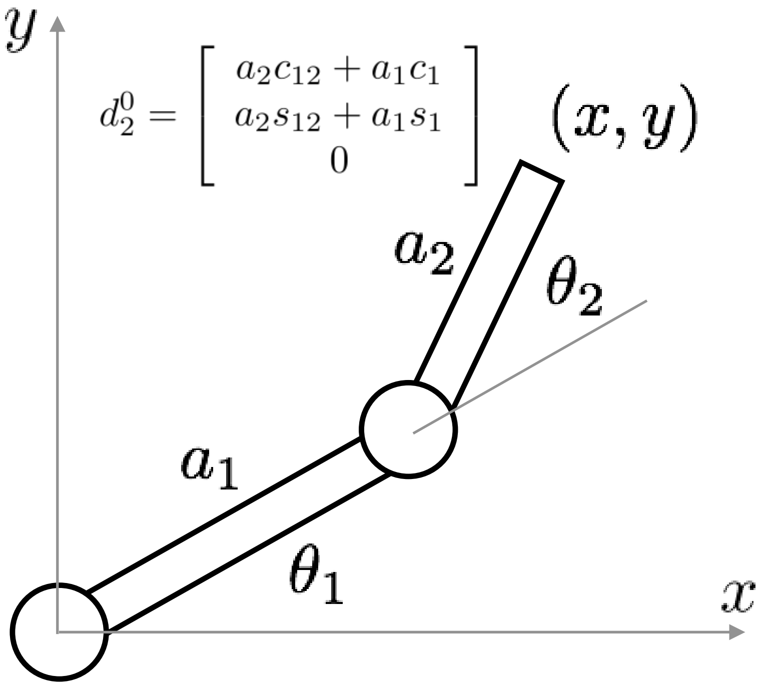
Mathematically, singularities exist at any point in the workspace where the Jacobian matrix loses rank.

Let's look at square J_v first

a matrix is singular if and only if its determinant is zero:

$$\det(J_v) = 0$$

Planar RR



$$J_v(\vec{q}) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix}$$

$$J_{v,\text{planar}}(\vec{q}) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \end{bmatrix}$$

$$\det(J_{v,\text{planar}}(\vec{q})) = ?$$

$$= (-a_1s_1 - a_2s_{12})(a_2c_{12}) - (-a_2s_{12})(a_1c_1 + a_2c_{12})$$

$$\det(J_{v,\text{planar}}(\vec{q})) = a_1a_2(\underline{c_1s_{12} - s_1c_{12}})$$

When does $\det(\mathbf{J}) = 0$?

$\det(\mathbf{J}) = 0$ when $\theta_2 = 0$

Any other times?

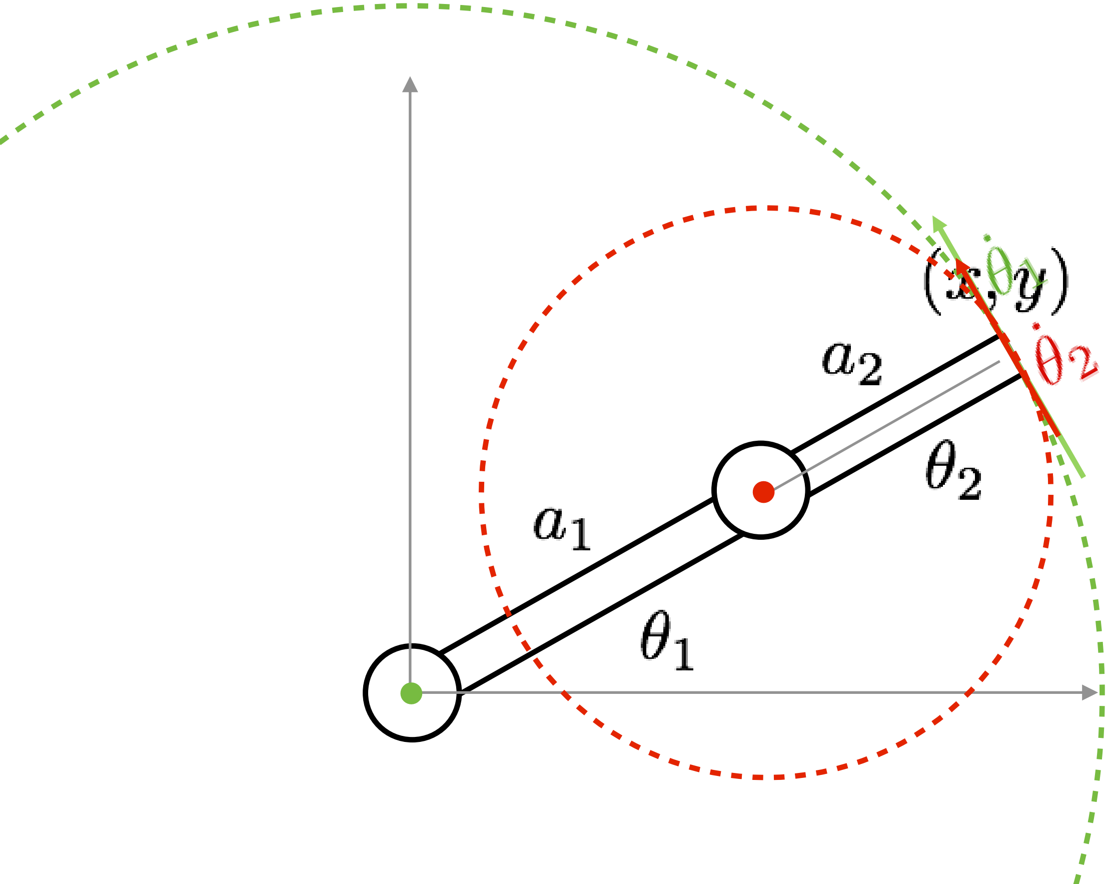
$\det(\mathbf{J}) = 0$ when $a_1 = 0$ or $a_2 = 0$

if $\theta_2 = 0, c_1s_{12} - s_1c_{12} = c_1s_1 - s_1c_1 = 0$

Is that the only time?

No... $\det(\mathbf{J}) = 0$ when $\theta_2 = \ldots, -2\pi, -\pi, 0, \pi, 2\pi, \ldots$

Planar RR



For $\theta_2 = 0$

The Jacobian collapses to have linearly dependent rows

$$\mathbf{J}_{\theta_2=0} = \begin{bmatrix} -a_1 s_1 - a_2 s_1 & -a_2 s_1 \\ a_1 c_1 + a_2 c_1 & a_2 c_1 \end{bmatrix}$$

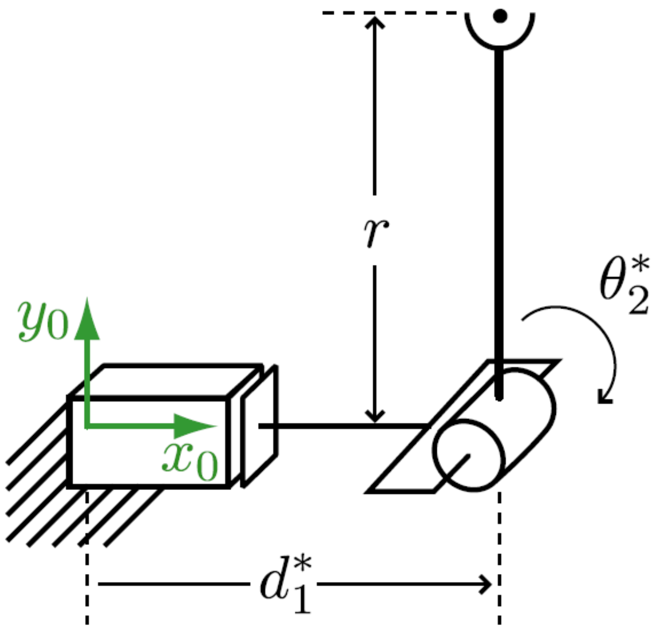
This means that actuating either joint causes motion in the same direction

We often try to avoid singularities.

What questions do you have?

Your turn: PR Manipulator

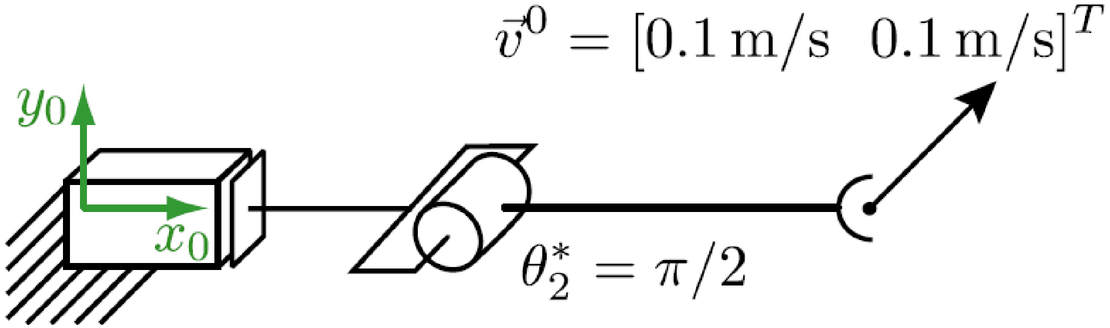
What are the singular configurations of this robot?



$$p^0 = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} d_1^* + r\sin\theta_2^* \\ r\cos\theta_2^* \end{bmatrix}$$

$$\boxed{J_v = \begin{bmatrix} 1 & r\cos\theta_2^* \\ 0 & -r\sin\theta_2^* \end{bmatrix}}$$

PR Manipulator

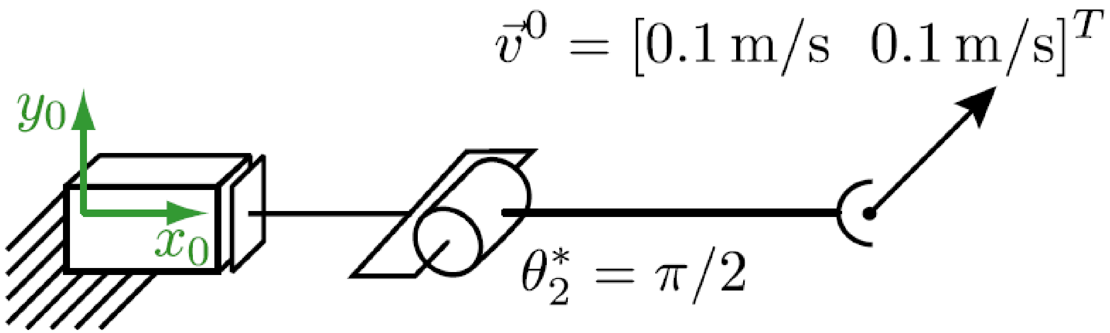


$$\boxed{J_v = \begin{bmatrix} 1 & r\cos\theta_2^* \\ 0 & -r\sin\theta_2^* \end{bmatrix}}$$

$$\dot{d}_1^* = ? \qquad \dot{\theta}_2^* = ?$$

When the robot is at the pose shown above, what joint velocities are needed to make the gripper move with the indicated velocity vector?

PR Manipulator



$$\boxed{J_v = \begin{bmatrix} 1 & r\cos\theta_2^* \\ 0 & -r\sin\theta_2^* \end{bmatrix}}$$

$$\dot{d}_1^* = ? \qquad \dot{\theta}_2^* = ?$$

When the robot is at the pose shown above, what joint velocities are needed to make the gripper move with the indicated velocity vector?

$$v_n^0 = J_v \dot{q}$$

$$\begin{bmatrix} \dot{x}^0 \\ \dot{y}^0 \end{bmatrix} = J_v \begin{bmatrix} \dot{d}_1^* \\ \dot{\theta}_2^* \end{bmatrix}$$

$$J_v(\theta_2^* = \pi/2) = \begin{bmatrix} 1 & 0 \\ 0 & -r \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}^0 \\ \dot{y}^0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -r \end{bmatrix} \begin{bmatrix} \dot{d}_1^* \\ \dot{\theta}_2^* \end{bmatrix}$$

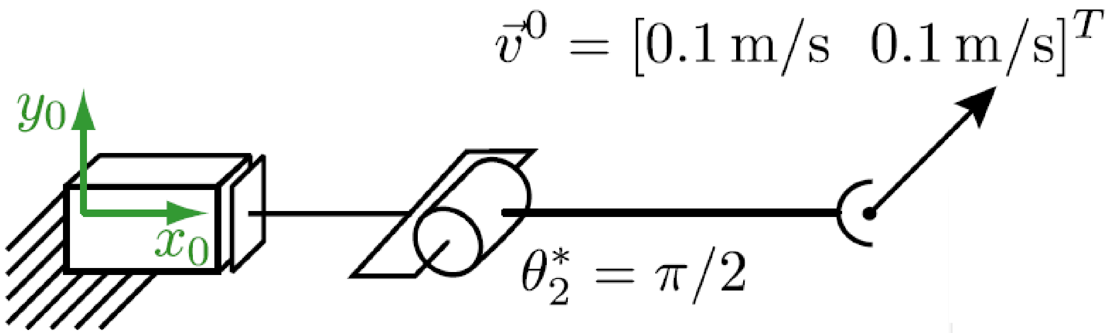
$$\dot{x}^0 = \dot{d}_1^* \qquad \dot{y}^0 = -r\dot{\theta}_2^*$$

Forward velocity kinematics

$$\boxed{\dot{d}_1^* = 0.1\,\mathrm{m/s} \quad \dot{\theta}_2^* = \frac{-0.1\,\mathrm{m/s}}{r}}$$

Inverse velocity kinematics

PR Manipulator



$$J_v = \begin{bmatrix} 1 & r\cos\theta_2^* \\ 0 & -r\sin\theta_2^* \end{bmatrix}$$

General forward velocity kinematics

$$\dot{d}_1^* = ? \qquad \dot{\theta}_2^* = ?$$

When the robot is at the pose shown above, what joint velocities are needed to make the gripper move with the indicated velocity vector?

A more general approach

$$v_n^0 = J_v \dot{q} \qquad \dot{q} = J_v^{-1} v_n^0$$

$$J_v^{-1} = \begin{bmatrix} 1 & \cos\theta_2^* / \sin\theta_2^* \\ 0 & -1/(r\sin\theta_2^*) \end{bmatrix}$$

General inverse velocity kinematics

$$\begin{bmatrix} \dot{d}_1^* \\ \dot{\theta}_2^* \end{bmatrix} = \begin{bmatrix} 1 & \cos\theta_2^* / \sin\theta_2^* \\ 0 & -1/(r\sin\theta_2^*) \end{bmatrix} \begin{bmatrix} \dot{x}^0 \\ \dot{y}^0 \end{bmatrix}$$

Will inverse velocity kinematics always return a solution?

No. It will fail when the robot is at a singular configuration!

$$r = 0 \qquad \sin\theta_2^* = 0$$

6-DOF Manipulators

$$\xi = J(q)\dot{q}$$

It is mathematically challenging to find all of the singularities for a 6-DOF manipulator; the determinant of the Jacobian gets very complicated!

For a 6-DOF manipulator with a spherical wrist, we can decouple the determination of singular configurations into two simpler problems.

$$\xi = J(q)\dot{q}$$

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}]$$

$$\text{(the book calls this } J = [J_P \mid J_O]\text{)}$$

$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\begin{array}{c|c} J_{11} & J_{12} \\ \hline J_{21} & J_{22} \end{array}\right]$$

$$J_{\text{wrist}} = \begin{bmatrix} z_3 \times (o_6 - o_3) & z_4 \times (o_6 - o_4) & z_5 \times (o_6 - o_5) \\ z_3 & z_4 & z_5 \end{bmatrix}$$

Put the origin of the effector-frame at the center of the wrist so that wrist rotations cause no translation of the end-effector. Of course, wrist rotations do actually move the tip, but this is convenient for analysis.

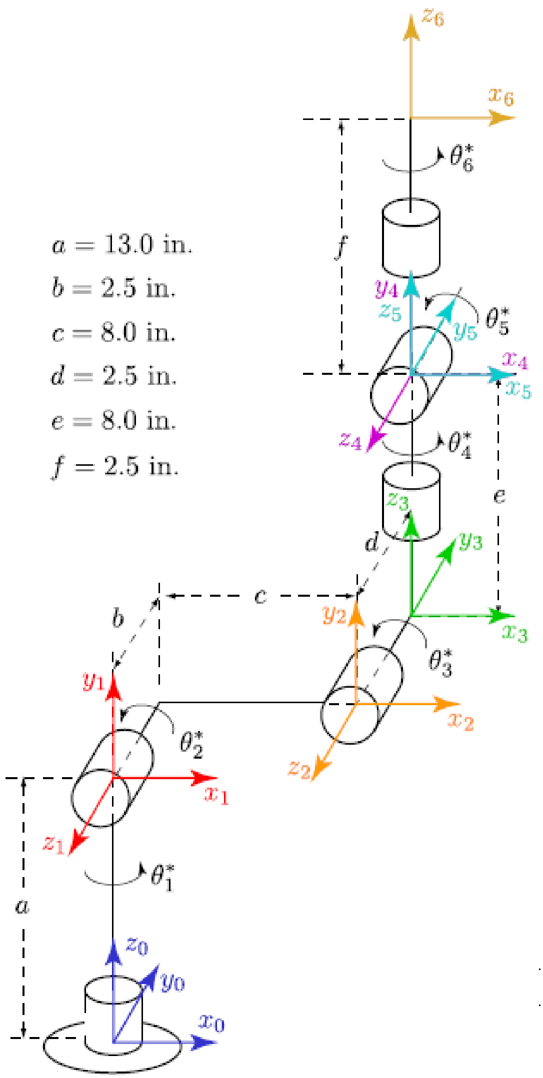
$$\text{if we choose } o_4 = o_5 = o_6$$

$$J_{\text{wrist}} = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix}$$

$$J = \left[\begin{array}{c|c} J_{11} & 0 \\ \hline J_{21} & J_{22} \end{array}\right]$$

$$\det(J) = \det(J_{11})\det(J_{22})$$

arm wrist



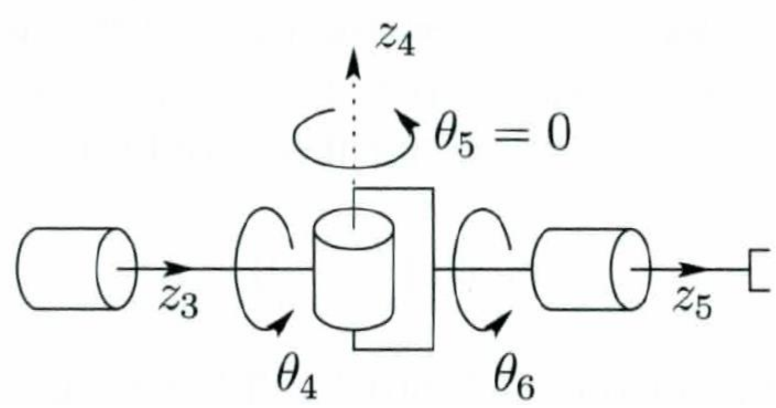
The book erroneously says that o3 must also be at the wrist center. Why isn't that needed? Because z3 is the first axis of the wrist; the wrist center lies along z3 away from o3.

$$\det(J) = \det(J_{11})\det(J_{22})$$

$$J_{22} = \begin{bmatrix} z_3 & z_4 & z_5 \end{bmatrix}$$

When will this matrix be singular?

Singular when any two wrist axes align



Questions?

$z_3 \perp z_4$

$z_4 \perp z_5$

z_3 can become $\|$ z_5

$\theta_5 = 0, \pi$ are singular configurations

Non-Square Jacobians (SHV 4.11)

$$N \neq 6$$

J is not square – cannot be inverted

Q: Does a solution to $\dot{q} = J^{-1}\xi$ exist?

Def: **matrix rank** – maximum number of linearly independent columns

Rank test: $\mathrm{rank}\, J = \mathrm{rank}[J \,|\xi]$

Check whether ξ is a linear combination of the columns of *J*

Pseudoinverse: N>6

For nonsquare matrices, we can define a pseudoinverse J^+ such that

$$\dot{q} = J^+ \xi$$

If J is a MxN matrix with rank M, then Happens, e.g., when N>6

- JJ^T is MxM
- $(JJ^T)^{-1}$ exists

Notice: $$I = JJ^T(JJ^T)^{-1} = J\underbrace{[J^T(JJ^T)^{-1}]}_{J^+ \in \mathbb{R}^{N\times M}}$$

Uses of the Pseudoinverse: N>6

SHV 4.11 tells you how to compute J^+ using SVD

$$\xi = J\dot{q} \qquad\qquad I = J[J^T(JJ^T)^{-1}] = JJ^+$$

- If a solution $\dot{q}$ exists, then $\dot{q}' = J^+\xi$ is a solution
- $\dot{q}' = J^+\xi$ is the solution that minimizes $||\dot{q}'||_2$

- With *N*>6, there may be more than one solution
  - $J^+J \in \mathbb{R}^{N\times N}$ Note: $J^+J \neq I$ even though $JJ^+ = I$
  - All vectors $(I - J^+J)b$, with $b \in \mathbb{R}^N$, are in the null space of J
  - If the joints move with velocity $(I - J^+J)b$, then the end effector frame **does not change**
  - All $\dot{q}' = J^+\xi + (I - J^+J)b$ are min norm solutions

Pseudoinverse: N<6

For nonsquare matrices, we can define a pseudoinverse J⁺ such that

$$\dot{q} = J^{+}\xi$$

If J is a MxN matrix with rank N, then

Happens, e.g., when N<6

- $J^T J$ is NxN
- $(J^T J)^{-1}$ exists

Notice: $I = (J^T J)^{-1} J^T J = \underbrace{[(J^T J)^{-1} J^T]}_{J^{+} \in \mathbb{R}^{N \times M}} J$

$\dot{q}' = J^{+}\xi$ is a least squares solutions

MATLAB has the following functions

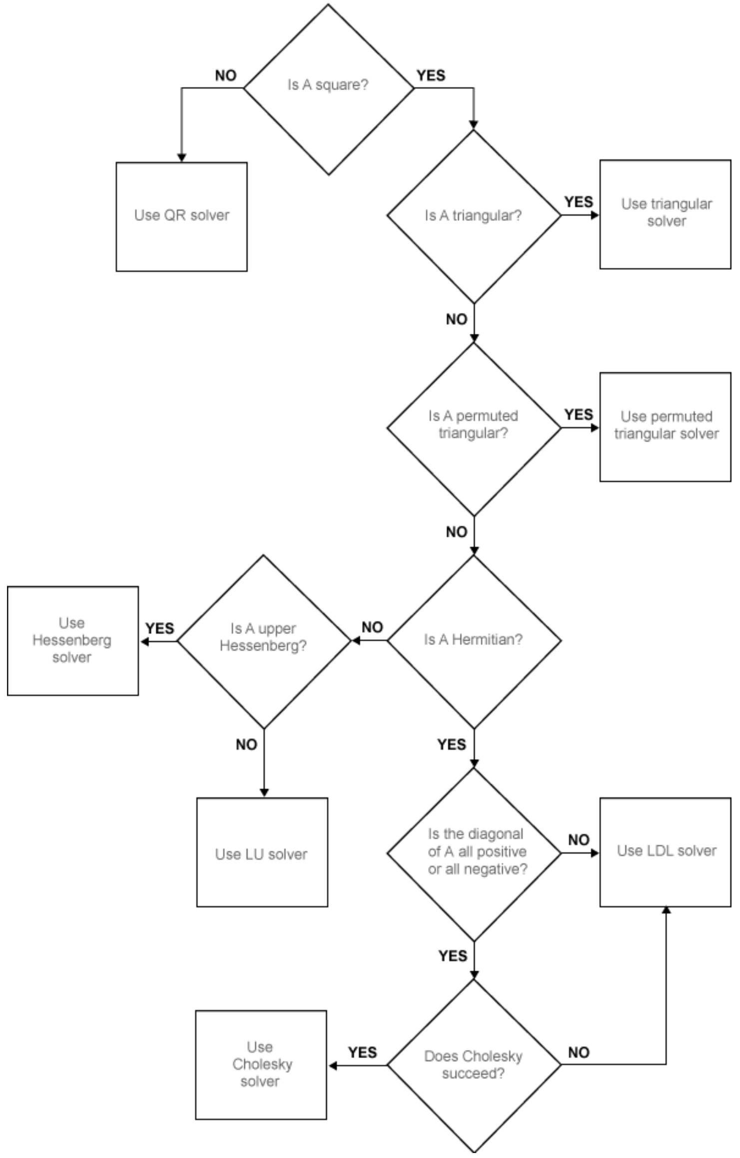
```
pinv:          qdot = pinv(J)*xi;
\:             qdot = J \ xi;
```

Backslash doesn't always return the same solution as `pinv(J)*xi`

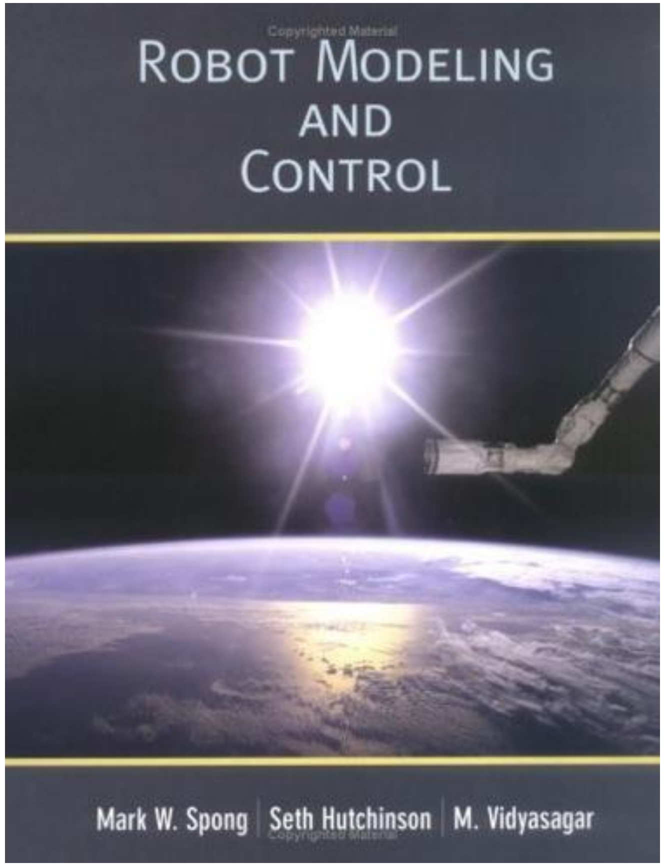
pinv is computed using SVD.
\ solves using (usually) QR.

QR is (usually) faster. SVD is more stable.

Find out more from your linear algebra or computational mathematics classes

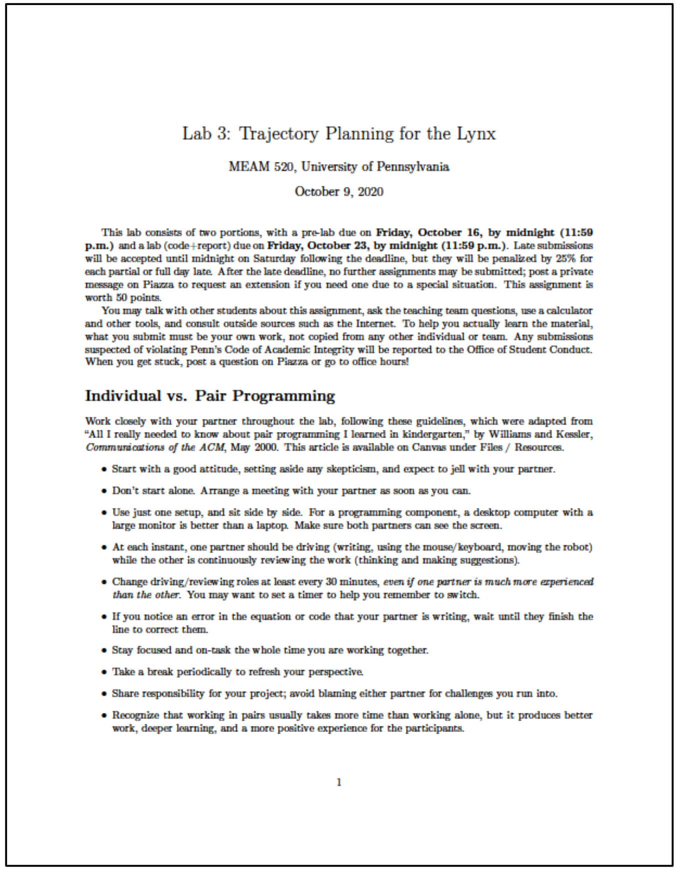


Next time: Jacobians and Forces



Ch 4: Velocity Kinematics – The Jacobian

- Read 4.8-4.13



Lab 3: Trajectory Planning for the Lynx

MEAM 520, University of Pennsylvania

October 9, 2020

This lab consists of two portions, with a pre-lab due on **Friday, October 16, by midnight (11:59 p.m.)** and a lab (code+report) due on **Friday, October 23, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation. This assignment is worth 50 points.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

Work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 3 due tomorrow