

**MEAM 520**

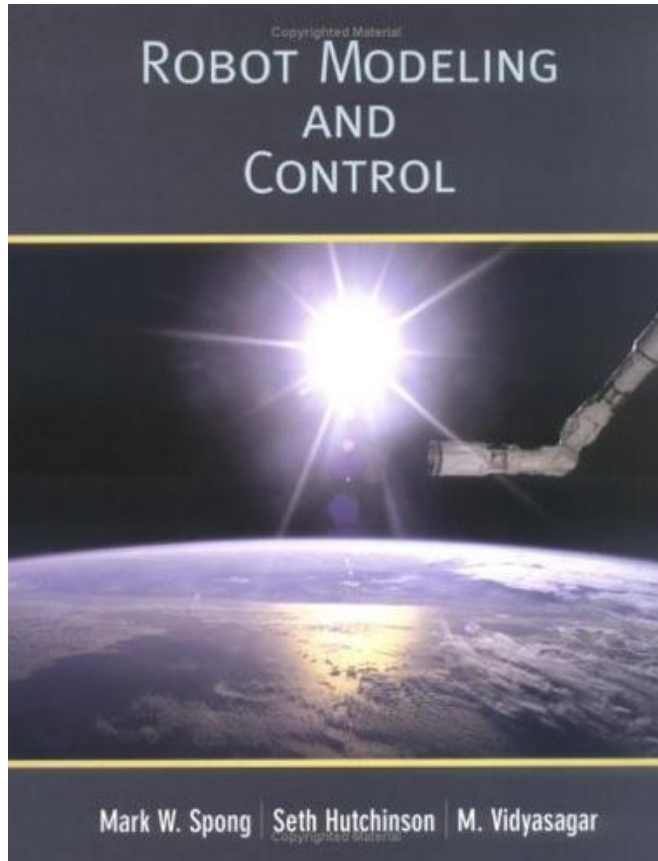
# **Lecture 17: Jacobians and Forces**

Cynthia Sung, Ph.D.

Mechanical Engineering & Applied Mechanics

University of Pennsylvania

# Today: Jacobians and Forces



## Ch 4: Velocity Kinematics – The Jacobian

- Read 4.8-4.13

### Lab 4: Jacobians and Velocity Kinematics

MEAM 520, University of Pennsylvania

October 23, 2020

This lab consists of two portions, with a pre-lab due on Friday, October 30, by midnight (11:59 p.m.) and a lab report due on Friday, November 6, by midnight (11:59 p.m.). Late submissions will be accepted until midnight on Monday 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.

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 Kosler, *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 4 posted (due Nov. 6)

# Quick announcement on paper readings

2007 IEEE International Conference on  
Robotics and Automation  
Roma, Italy, 10-14 April 2007

FrE4.2

## Grasping POMDPs

Kaijen Hsiao and Leslie Pack Kaelbling and Tomás Lozano-Pérez

**Abstract**—We provide a method for planning under uncertainty for robotic manipulation by partitioning the configuration space into a set of regions that are closed under compliant motions. These regions can be treated as states in a partially observable Markov decision process (POMDP), which can be solved to yield optimal control policies under uncertainty. We demonstrate the approach on simple grasping problems, showing that it can construct highly robust, efficiently executable solutions.

### I. INTRODUCTION

A great deal of progress has been made on the problem of planning motions for robots with many degrees of freedom through free space [10], [9], [13]. These methods enable robots to move through complex environments, as long as they are not in contact with the objects in the world. However, as soon as the robot needs to contact the world, in order to manipulate objects, for example, these strategies do not apply. The fundamental problem with planning for motion in contact is that the configuration of the robot and the objects in the world is not exactly known at the outset of execution, and, given the resolution of sensors, it cannot be exactly known. In such cases, traditional open-loop plans (even extended with simple feedback) are not reliable.

It is useful to distinguish between modes of uncertainty that can be effectively modeled, and those that cannot. In situations with unmodelable uncertainty, such as insertion of keys into locks, very fine-grained details of the surfaces can have large effects on the necessary directions of applied forces, and the available sensors can gain little or no information about those surfaces. When the uncertainty is unmodelable, we must fall back to strategies such as “wiggling” the key, which are highly successful without ever building a model of the underlying situation.

Modelable uncertainty, on the other hand, typically occurs at a coarser scale. In attempting to pick up a mug, for example, a robot with vision or range sensing might have a good high-level model of the situation, but significant remaining uncertainty about the pose or shape of the mug. Based on sensor feedback, it can reason about whether the hand is currently in contact with the handle or the cup body, and choose actions that will both gather more information about the situation and make progress toward a desired grasp with a multi-fingered hand.

This research was supported in part by DARPA IPTO Contract FA8750-05-2-0249, “Effective Bayesian Transfer Learning”, and in part by the Singapore-MIT Alliance agreement dated 11/6/98.  
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139  
{kjhhsiao, lpk, tlp}@csail.mit.edu

An early approach to planning in the presence of modelable uncertainty was developed in [15]. They used a worst-case model of sensor and motion error, and developed a framework for computing conservative plans under these assumptions. This method was computationally complex, and prone to failure due to overconservatism: if there was no plan that would work for all possible configurations consistent with the initial knowledge state, then the entire system would fail.

In this paper, we build on those ideas, addressing the weaknesses in the approach via abstraction and probabilistic representation. By modeling the initial uncertainty using a probability distribution, rather than a set, and doing the same for uncertainties in dynamics and sensing, we are in a position to make trade-offs when it is not possible to succeed in every possible situation. We can choose plans that optimize a variety of different objective functions involving those probabilities, including, most simply, the plan most likely to achieve the goal. The probabilistic representation also affords an opportunity for enormous computational savings through a focus on the parts of the space that are most likely to be encountered.

By building an abstraction of the underlying continuous configuration and action spaces, we lose the possibility of acting optimally, but gain an enormous amount in computational simplification, making it feasible to compute solutions to real problems. Concretely, we will use methods of model minimization to create an abstract model of the underlying configuration space, and then model the problem of choosing actions under uncertainty as a *partially observable Markov decision process* [25].

### II. BACKGROUND AND APPROACH

The approach we outline here applies to any domain in which a robot is moving or interacting with other objects and there is non-trivial uncertainty in the configuration. In this paper, we concentrate on the illustrative problem of a robot arm and hand performing pick-and-place operations. We assume that the robot’s position in the global frame is reasonably well known, but that there is some uncertainty about the relative pose and/or shape of the object to be manipulated. Additionally, we assume that there are tactile and/or force sensors on the robot that will enable it to perform compliant motions and to reasonably reliably detect when it makes or loses contacts. We frame this problem primarily as a planning problem. That is, we assume that a reasonably accurate model of the task dynamics and sensors is known, and that the principal uncertainty is in the configuration of the robot and the state of the objects in

## Next Tuesday:

Hsiao, K and Kaelbling, LP, and Lozano-Pérez, T. “Grasping POMDPs.” ICRA 2007.

Paper readings are now posted as optional on Canvas.

If you submit your answers, they will contribute an **extra** 2.5 pts to your grade.

If you don’t submit, no penalty.

We will still talk about them in class, though.

# Last Time: Manipulator 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) linear velocity Jacobian

(3 x n) angular velocity Jacobian

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

forward velocity kinematics

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

(6 x 1) body velocity

(6 x n) Jacobian

(n x 1) joint velocities

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

$$\rho_i = \begin{matrix} 0 & \text{for prismatic} \\ 1 & \text{for revolute} \end{matrix}$$

inverse velocity kinematics

$$\dot{q} = J^{-1} \xi$$

## Last Time: Singularities

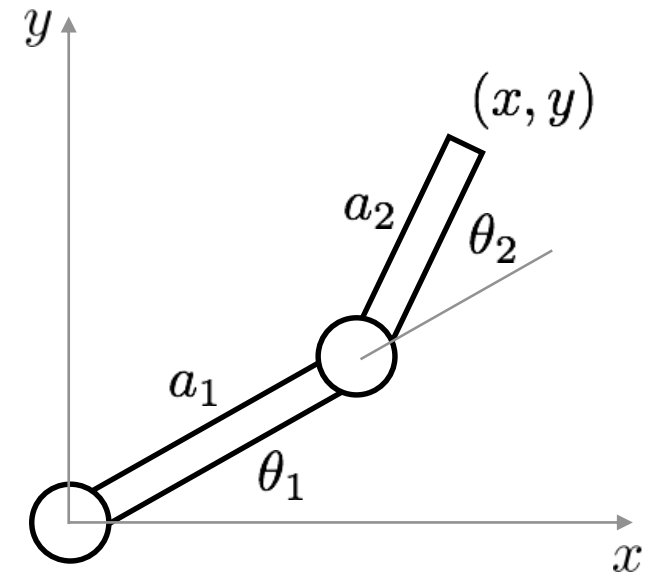
$$J_{v,\text{planar}}(\vec{q}) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$

$$\det(J_{v,\text{planar}}(\vec{q})) = a_1 a_2 (c_1 s_{12} - s_1 c_{12})$$

$$\det(\mathbf{J}) = 0 \text{ when } \theta_2 = \dots, -2\pi, -\pi, 0, \pi, 2\pi, \dots$$

$$\det(\mathbf{J}) = 0 \text{ when } a_1 = 0 \text{ or } a_2 = 0 \quad z_3 \text{ can become } || z_5$$

$$J_{\text{planar}}(q) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \\ 0 & 0 \end{bmatrix} \quad \text{rank}(\mathbf{J}) = 2$$



## 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:**  $\text{rank } J = \text{rank}[J \mid \xi]$  Check whether  $\xi$  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  $M \times N$  matrix with rank  $M$ , then      Happens, e.g., when  $N > 6$

- $JJ^T$  is  $M \times M$
- $(JJ^T)^{-1}$  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

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

$$I = J[J^T(JJ^T)^{-1}] = JJ^+$$

- 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 solutions



## Pseudoinverse: $N < 6$

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

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

If  $J$  is a  $M \times N$  matrix with rank  $N$ , then      Happens, e.g., when  $N < 6$

- $J^T J$  is  $N \times N$
- $(J^T J)^{-1}$  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

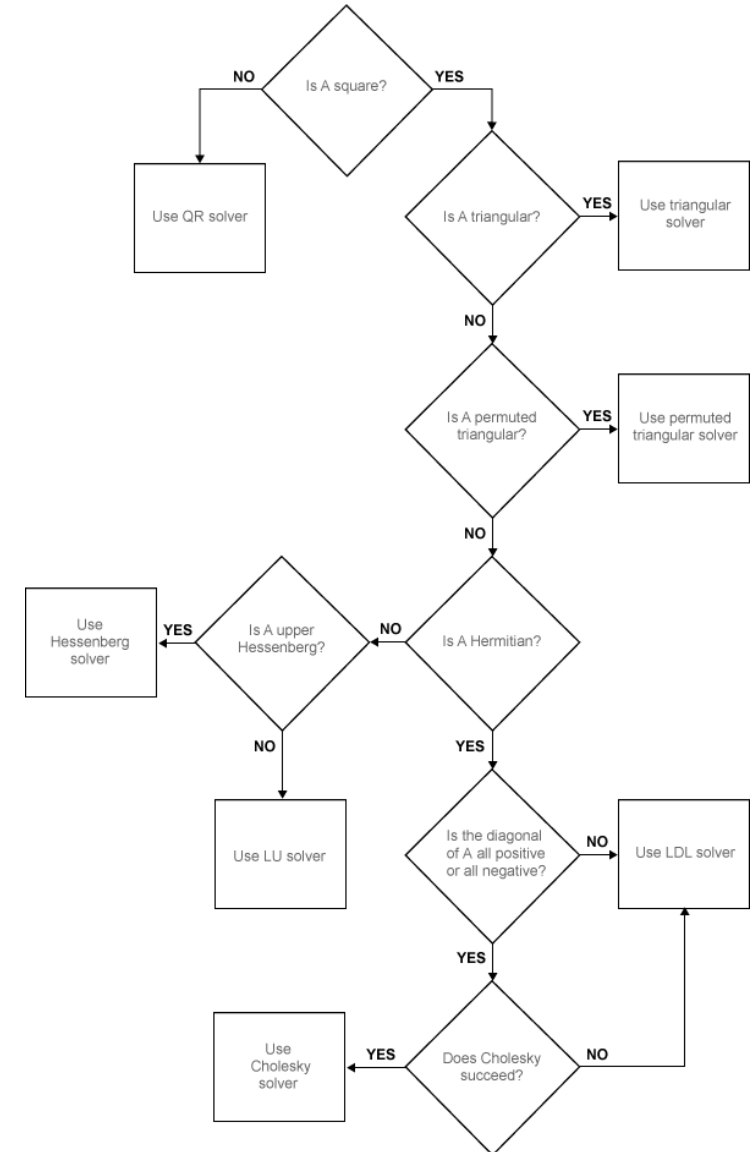
`pinv:`  $\dot{q} = \text{pinv}(J) * \dot{x}_i;$   
`\:`  $\dot{q} = J \setminus \dot{x}_i;$

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



# Manipulator 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) linear velocity Jacobian

(3 x n) angular velocity Jacobian

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

forward velocity kinematics

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

(6 x 1) body velocity      (6 x n) Jacobian      (n x 1) joint velocities

inverse velocity kinematics

$$\dot{q} = J^{-1}\xi$$

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

$$\rho_i = \begin{matrix} 0 & \text{for prismatic} \\ 1 & \text{for revolute} \end{matrix}$$

**NOTE:** All of this discussion has been for the end effector, but you can also write a Jacobian for any other point on the robot. Just change the location of the point you are tracking and check which joints affect that point.

# Manipulability (SHV 4.12)

For a specific configuration, the Jacobian scales the input (joint velocities) to the output (body velocity)

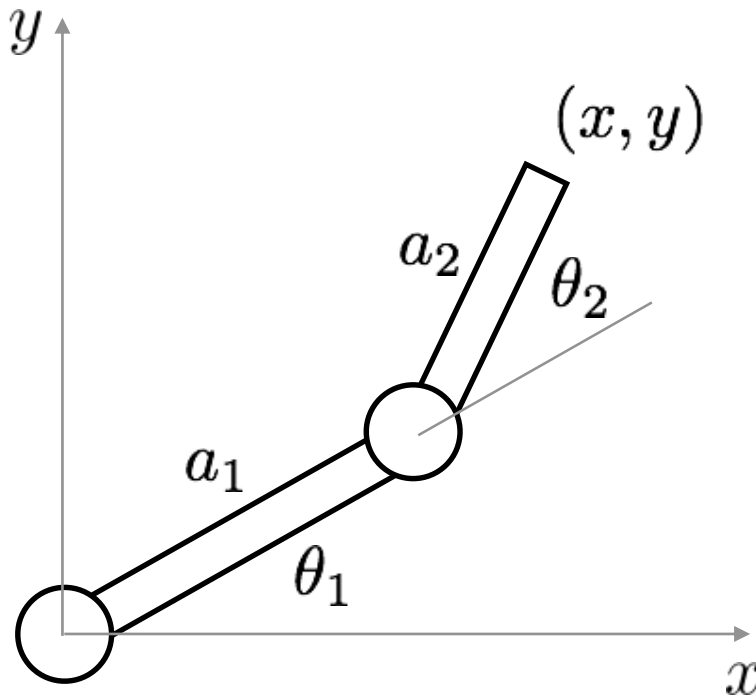
$$\xi = J(q)\dot{q}$$

If you put in a joint velocity vector with unit norm, you can calculate in which direction and how fast the robot's end-effector will translate and rotate.

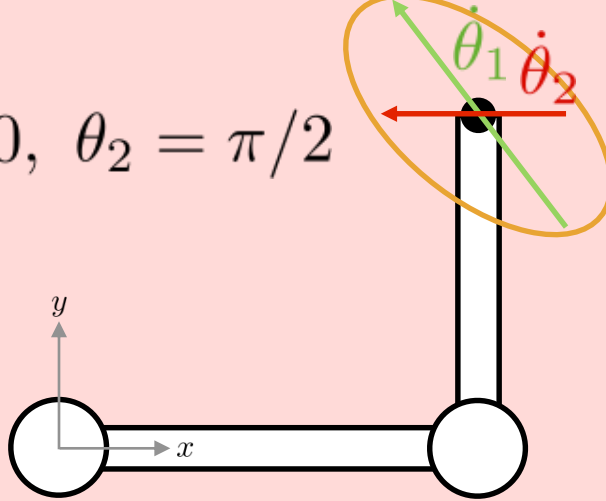
This approach allows you to calculate and plot the manipulability ellipsoid – a geometrical representation of all the possible tip velocities for a normalized joint velocity input.

A 6D ellipsoid is hard to visualize, but 2D and 3D ellipsoids are lovely and useful.

What does the manipulability ellipsoid look like  
for the planar RR robot?



$$\theta_1 = 0, \theta_2 = \pi/2$$



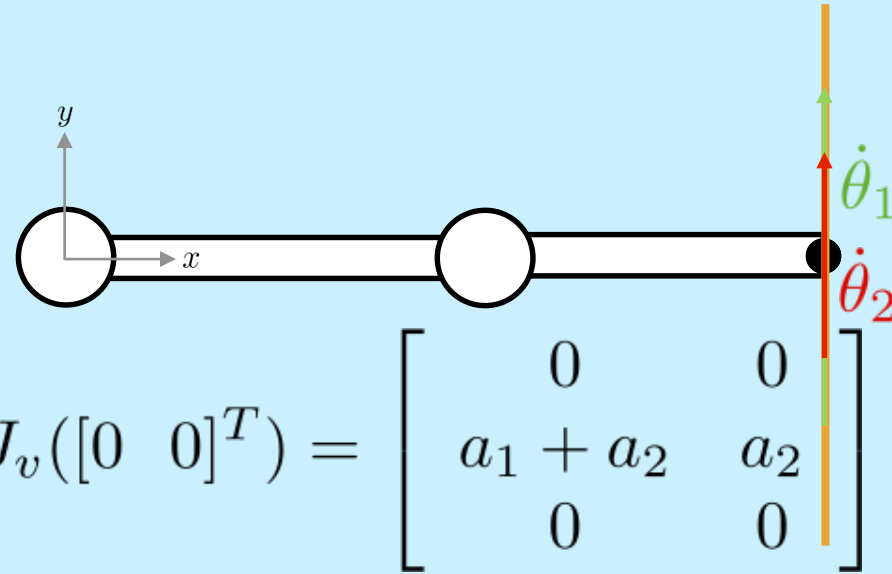
$$J_v([0 \ \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\dot{\vec{p}} = J_v(\vec{q}) \dot{\vec{q}}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -a_2\dot{\theta}_1 & -a_2\dot{\theta}_2 \\ a_1\dot{\theta}_1 \\ 0 \end{bmatrix}$$

The robot's tip cannot move in the z direction, but it can move in both x and y directions...

$$\theta_1 = 0, \theta_2 = 0$$



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

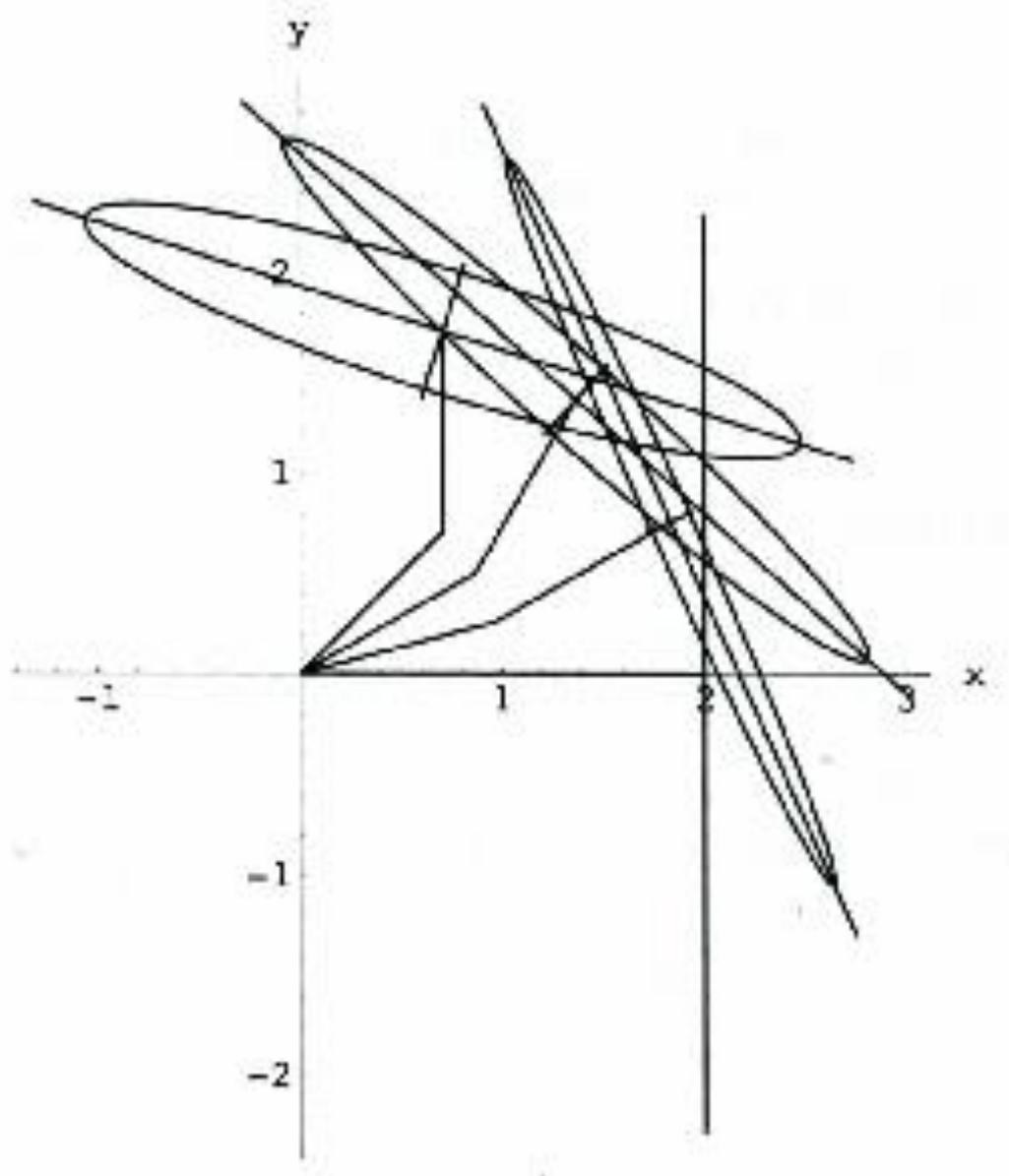
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ (a_1 + a_2)\dot{\theta}_1 & + a_2\dot{\theta}_2 \\ 0 \end{bmatrix}$$

## Manipulability

$$\mu = |\det(J)| = a_1 a_2 |\sin(\theta_2)|$$

Can be used to tell you where  
to perform certain tasks.

Also useful for deciding how to  
design a manipulator.



# Static Force/Torque Relationships

The transpose of the Jacobian relates joint forces and torques to Cartesian end-effector forces and torques

$$\begin{matrix} (n \times 1) & (n \times 6) & (6 \times 1) \\ \vec{\tau} = J^T(\vec{q}) \vec{F} \\ \uparrow & \uparrow & \uparrow \\ \text{joint} & & \text{endpoint} \\ \text{forces and} & & \text{forces and} \\ \text{torques} & & \text{torques} \\ & \uparrow & \\ & \text{Jacobian} & \\ & \text{matrix} & \\ & \text{transpose} & \end{matrix}$$

Simplest to think about for  
a 3-DOF robot with all  
revolute joints.  
We want to output a force  
at the tip.

$$\begin{matrix} (3 \times 1) & (3 \times 3) & (3 \times 1) \\ \vec{\tau} = J^T(\vec{q}) \vec{F} \\ \uparrow & \uparrow & \uparrow \\ \text{joint} & & \text{endpoint} \\ \text{torques} & & \text{forces} \\ & \uparrow & \\ & \text{Jacobian} & \\ & \text{matrix} & \\ & \text{transpose} & \end{matrix}$$



# Static Force/Torque Relationships

$$\vec{\tau} = J_v^\top \vec{F}$$

This relationship stems from virtual work:  
if we assume the arm has no frictional losses or compliance,  
we can equate work done at the joints with work done at the  
end-effector.

joints     $\vec{\tau} \cdot d\vec{q} = \vec{F} \cdot d\vec{x}$     end-effector

$$\dot{\vec{x}} = J_v \dot{\vec{q}}$$

or equivalently

$$\vec{\tau}^\top d\vec{q} = \vec{F}^\top d\vec{x}$$

$$\frac{d\vec{x}}{dt} = J_v \frac{d\vec{q}}{dt}$$

$$d\vec{x} = J_v d\vec{q}$$

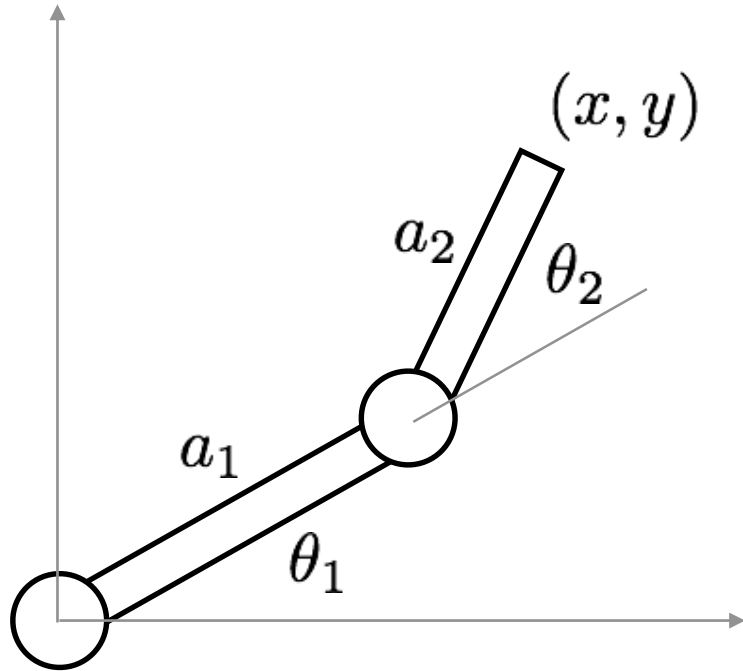
~~$$\vec{\tau}^\top d\vec{q} = \vec{F}^\top J_v d\vec{q}$$~~

$$\vec{\tau}^\top = \vec{F}^\top J_v$$

$$(\vec{\tau}^\top)^\top = (\vec{F}^\top J_v)^\top$$

$$\boxed{\vec{\tau} = J_v^\top \vec{F}}$$

## Example: Planar RR



Beginning with the 2 x 2 linear velocity Jacobian

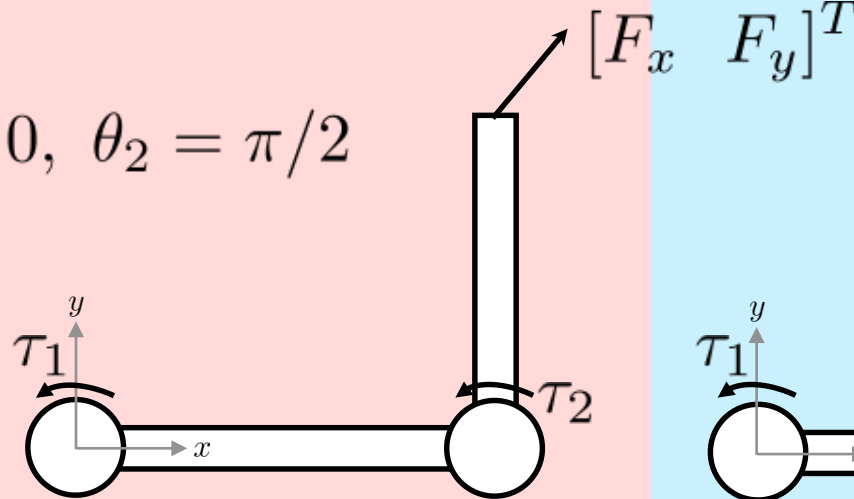
$$J_{v,\text{planar}}(\vec{q}) = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} \end{bmatrix}$$

We can solve for the joint torques necessary to exert a desired force at the end-effector using the Jacobian transpose

$$\vec{\tau} = J_v^\top \vec{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -a_1 s_1 - a_2 s_{12} & a_1 c_1 + a_2 c_{12} \\ -a_2 s_{12} & a_2 c_{12} \end{bmatrix} \begin{bmatrix} F_x \\ F_y \end{bmatrix}$$

$$\theta_1 = 0, \theta_2 = \pi/2$$



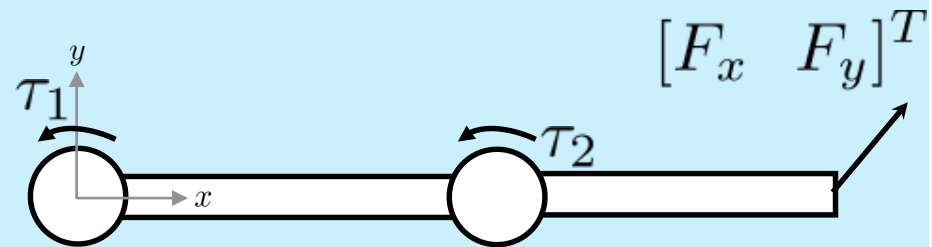
$$J_v([0 \ \pi/2]^T) = \begin{bmatrix} -a_2 & -a_2 \\ a_1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\tau_1 = -a_2 F_x + a_1 F_y$$

$$\tau_2 = -a_2 F_x$$

Can create forces in both x and y directions.

$$\theta_1 = 0, \theta_2 = 0$$



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

$$\tau_1 = (a_1 + a_2) F_y$$

$$\tau_2 = a_2 F_y$$

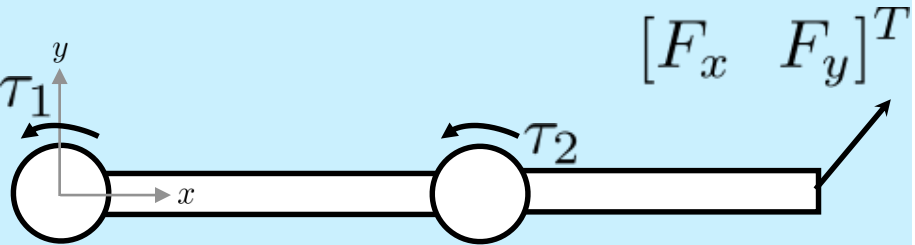
Can't create forces in the x direction!

At singularities, the manipulator is unable to move in certain directions.

Near singularities, the manipulator can only **actively** apply forces in certain directions.

**Q:** Can the manipulator apply forces in other directions?

$\theta_1 = 0, \theta_2 = 0$


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

$$\tau_1 = (a_1 + a_2)F_y$$

$$\tau_2 = a_2 F_y$$

Can't create forces in the x direction!

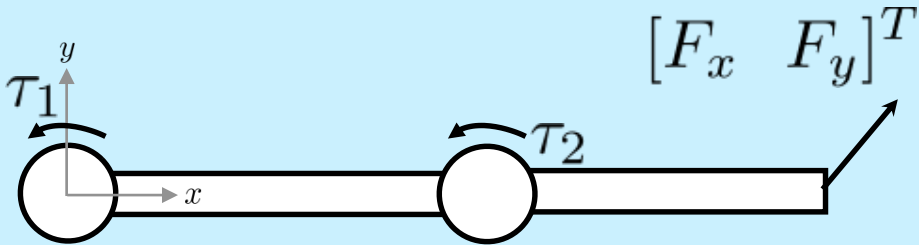
At singularities, the manipulator is unable to move in certain directions.

Near singularities, the manipulator can only **actively** apply forces in certain directions.

**Q:** Can the manipulator apply forces in other directions?

The robot can resist arbitrary externally applied forces/torques in  $\text{Null}(J^T)$  without moving.

$\theta_1 = 0, \theta_2 = 0$


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

$$\tau_1 = (a_1 + a_2)F_y$$

$$\tau_2 = a_2 F_y$$

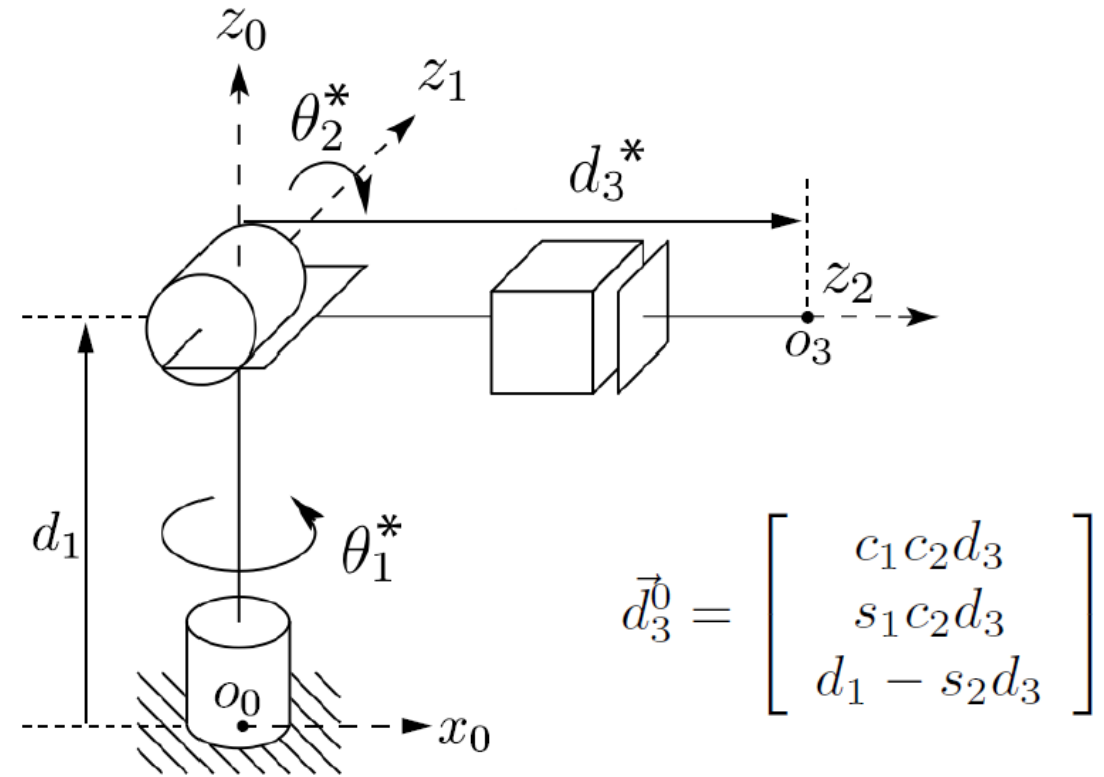
Can't create forces in the x direction!

# Spherical Manipulator

Find the Jacobian.

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

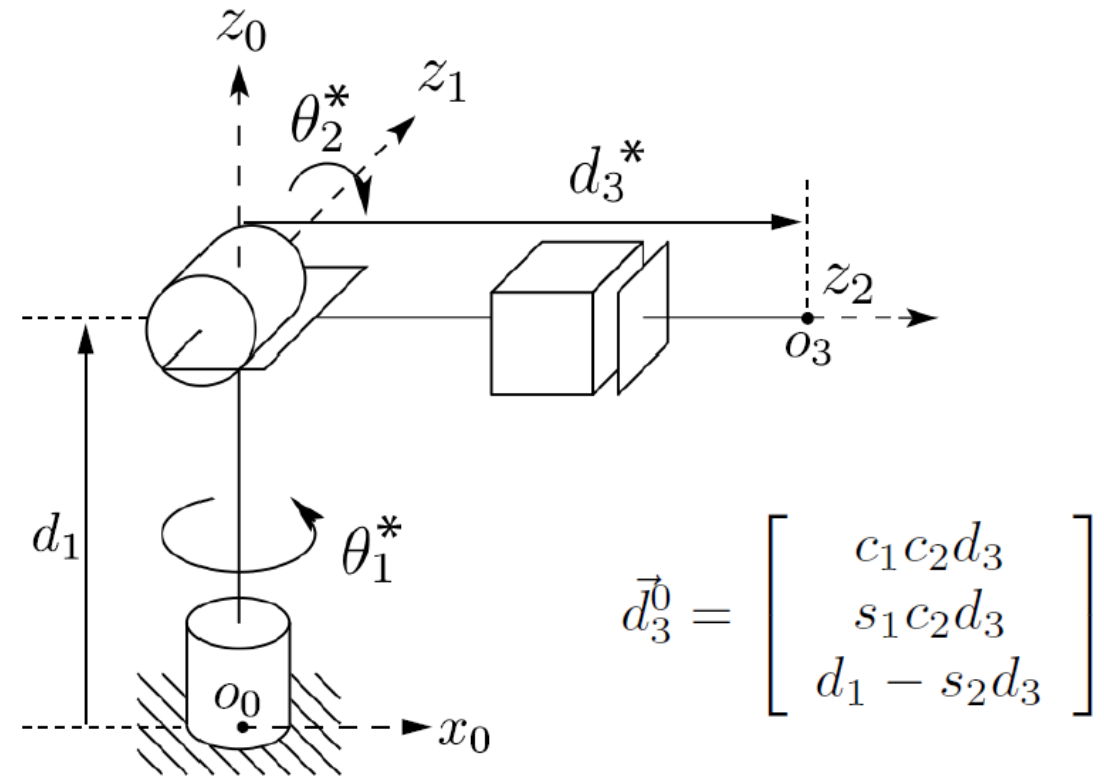
$$= \begin{bmatrix} -s_1 c_2 d_3 & -c_1 s_2 d_3 & c_1 c_2 \\ c_1 c_2 d_3 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & -c_2 d_3 & -s_2 \end{bmatrix}$$



$$J_\omega = [\rho_1 \hat{\mathbf{z}} \quad \rho_2 \mathbf{R}_1^0 \hat{\mathbf{z}} \quad \rho_3 \mathbf{R}_2^0 \hat{\mathbf{z}} \quad \cdots \quad \rho_n \mathbf{R}_{n-1}^0 \hat{\mathbf{z}}] = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

# Practice: Spherical Manipulator

- Under what conditions is there no solution to the IK?
- Where are the singularities?

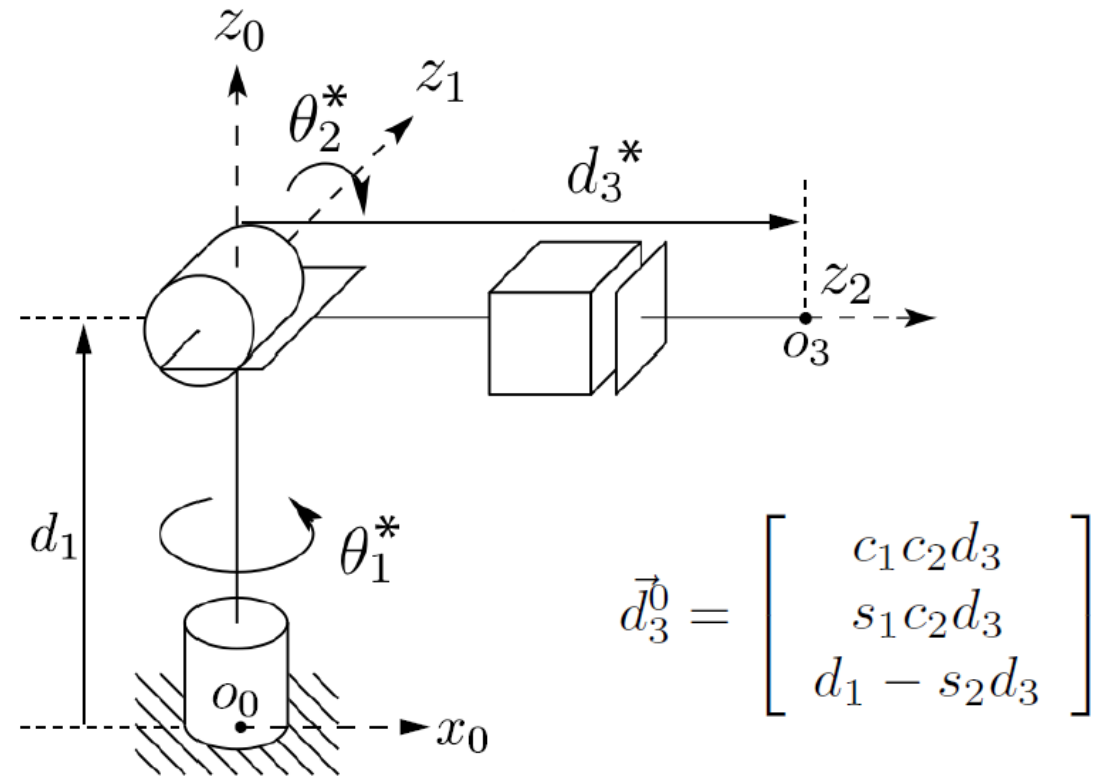


$$J_v = \begin{bmatrix} -s_1 c_2 d_3 & -c_1 s_2 d_3 & c_1 c_2 \\ c_1 c_2 d_3 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & -c_2 d_3 & -s_2 \end{bmatrix}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

# Practice: Spherical Manipulator

- Under what conditions is there no solution to the IK?
- Where are the singularities?



$$d_3 = 0 \quad \text{or} \quad \theta_2 = \frac{\pi}{2} + k\pi \quad \text{with } k = \dots, -2, -1, 0, 1, 2, \dots$$

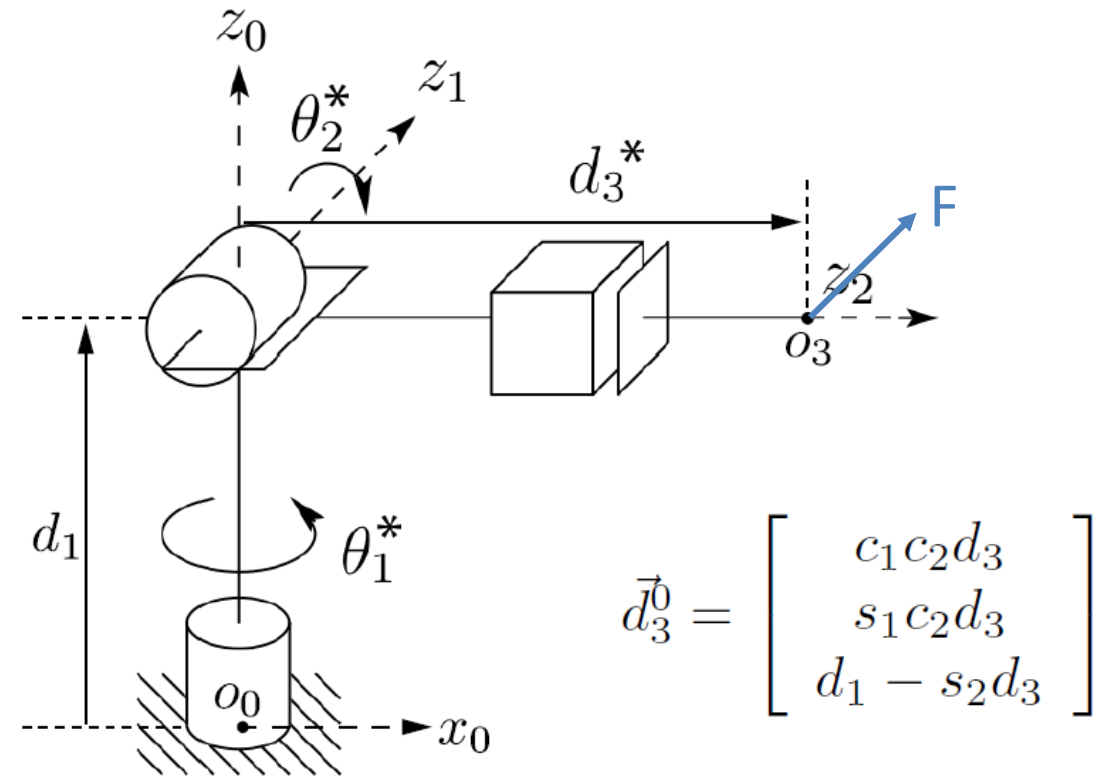
$$J_v = \begin{bmatrix} -s_1 c_2 d_3 & -c_1 s_2 d_3 & c_1 c_2 \\ c_1 c_2 d_3 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & -c_2 d_3 & -s_2 \end{bmatrix}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$



## Practice: Spherical Manipulator

- If the robot has the following joint values, what joint torques to choose if the tip should apply [0 N, 2.0 N, 2.0 N]?  
 $\theta_1 = \frac{\pi}{4} \text{ rad}, \theta_2 = 0 \text{ rad}, d_3 = 1 \text{ m}$



$$J_v = \begin{bmatrix} -s_1 c_2 d_3 & -c_1 s_2 d_3 & c_1 c_2 \\ c_1 c_2 d_3 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & -c_2 d_3 & -s_2 \end{bmatrix}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

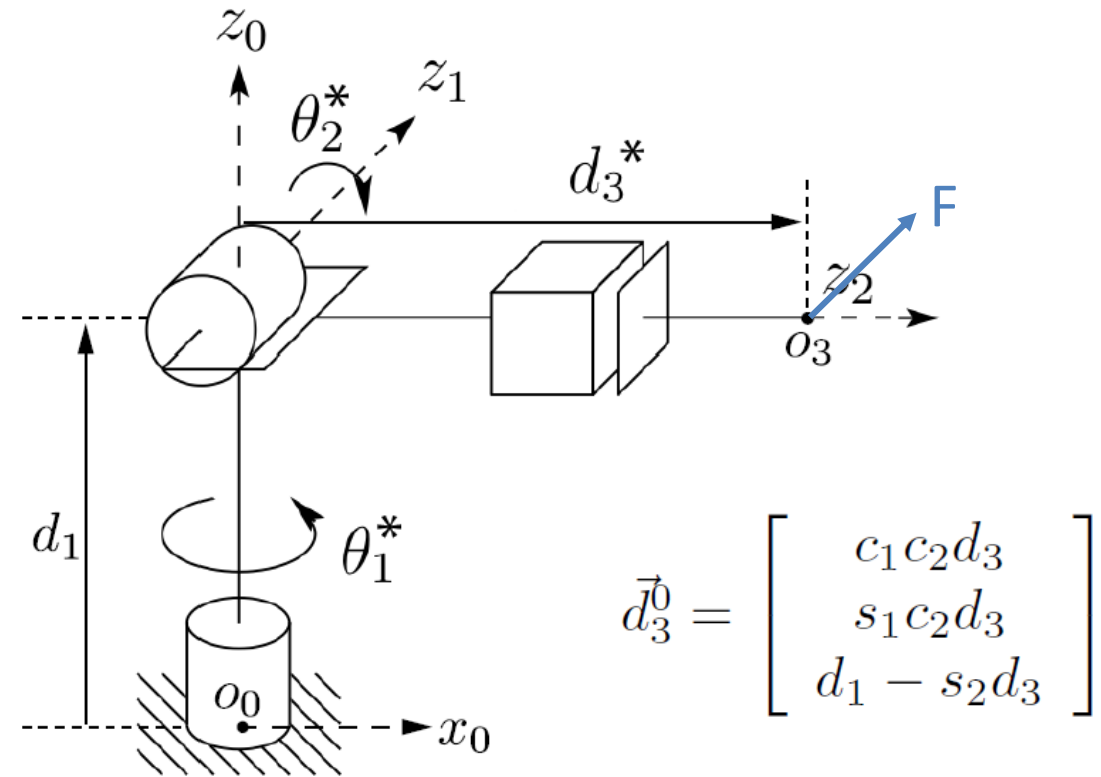
## Practice: Spherical Manipulator

- If the robot has the following joint values, what joint torques to choose if the tip should apply [0 N, 2.0 N, 2.0 N]?  
 $\theta_1 = \frac{\pi}{4} \text{ rad}, \theta_2 = 0 \text{ rad}, d_3 = 1 \text{ m}$

$$\vec{\tau} = J_v^\top \vec{F}$$

$$J_v = \begin{bmatrix} -\sqrt{2}/2 \text{ m} & 0 & \sqrt{2}/2 \\ \sqrt{2}/2 \text{ m} & 0 & \sqrt{2}/2 \\ 0 & -1 \text{ m} & 0 \end{bmatrix}$$

$$\tau = [\sqrt{2} \text{ N m} \quad -2 \text{ N m} \quad \sqrt{2} \text{ N}]$$



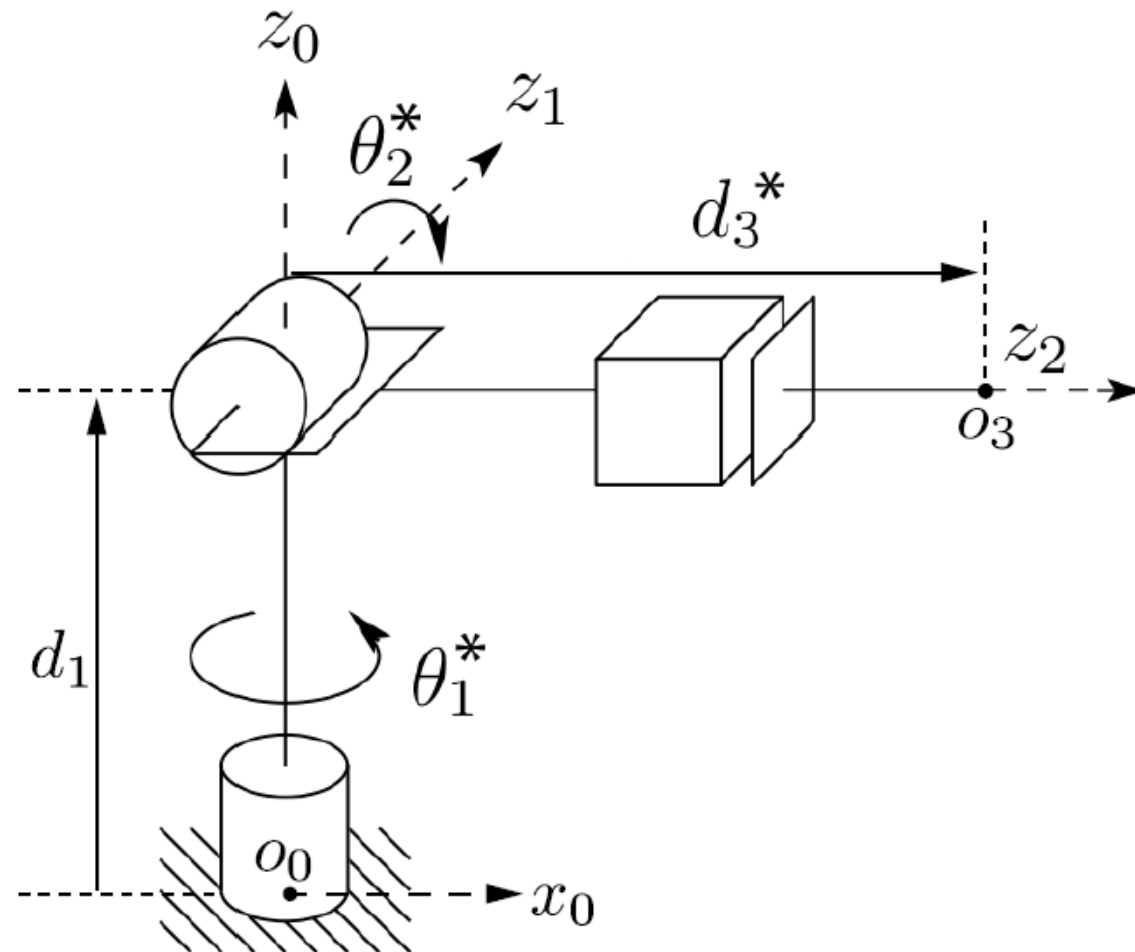
$$\vec{d}_3^0 = \begin{bmatrix} c_1 c_2 d_3 \\ s_1 c_2 d_3 \\ d_1 - s_2 d_3 \end{bmatrix}$$

$$J_v = \begin{bmatrix} -s_1 c_2 d_3 & -c_1 s_2 d_3 & c_1 c_2 \\ c_1 c_2 d_3 & -s_1 s_2 d_3 & s_1 c_2 \\ 0 & -c_2 d_3 & -s_2 \end{bmatrix}$$

$$J_\omega = \begin{bmatrix} 0 & -s_1 & 0 \\ 0 & c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

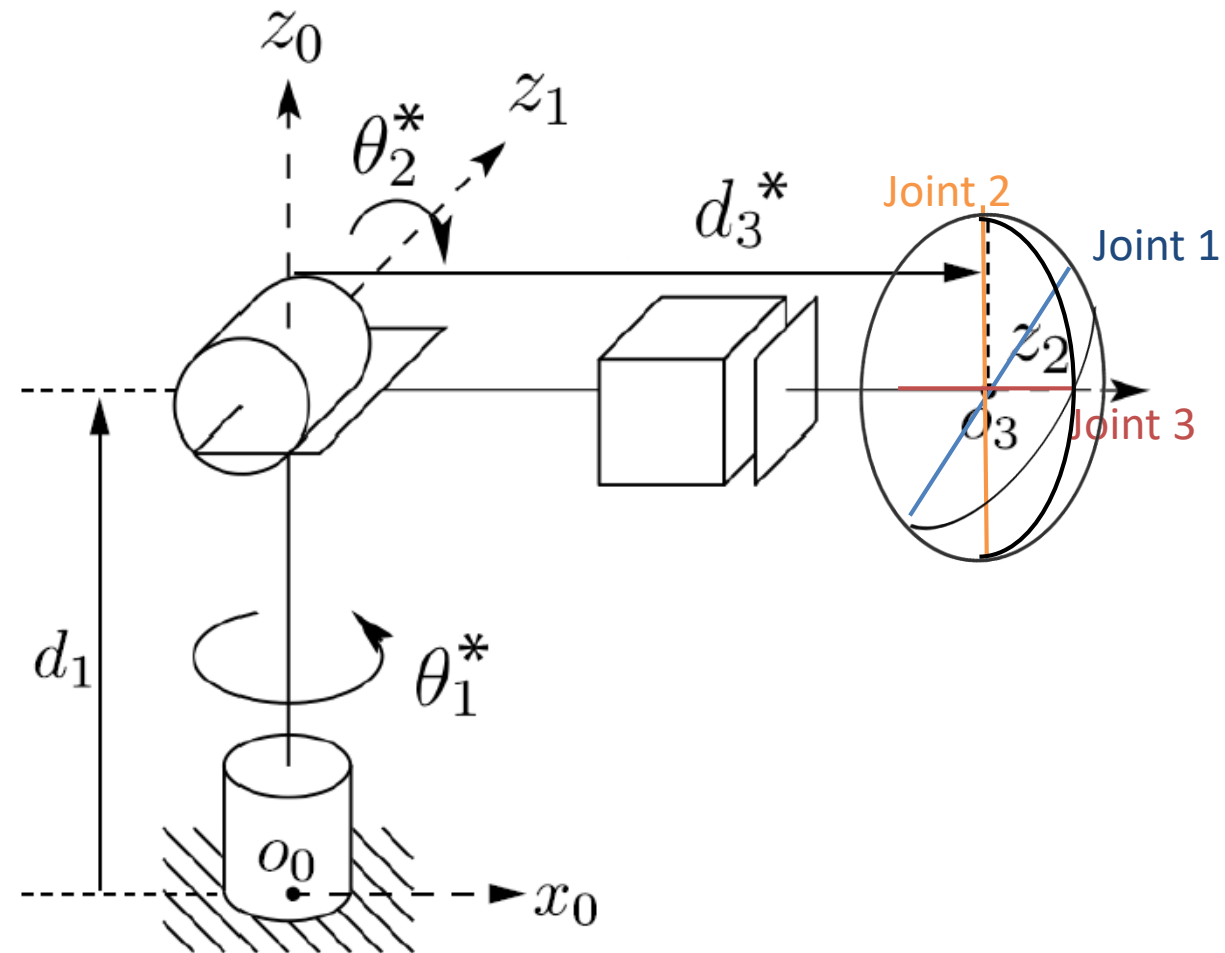
## Practice: Spherical Manipulator

- What does the manipulability ellipsoid look like in the zero configuration?



## Practice: Spherical Manipulator

- What does the manipulability ellipsoid look like in the zero configuration?



# Application: Gravity Compensation



## Strategy:

Determine expected form for gravitational force/torque and cancel out using statics analysis

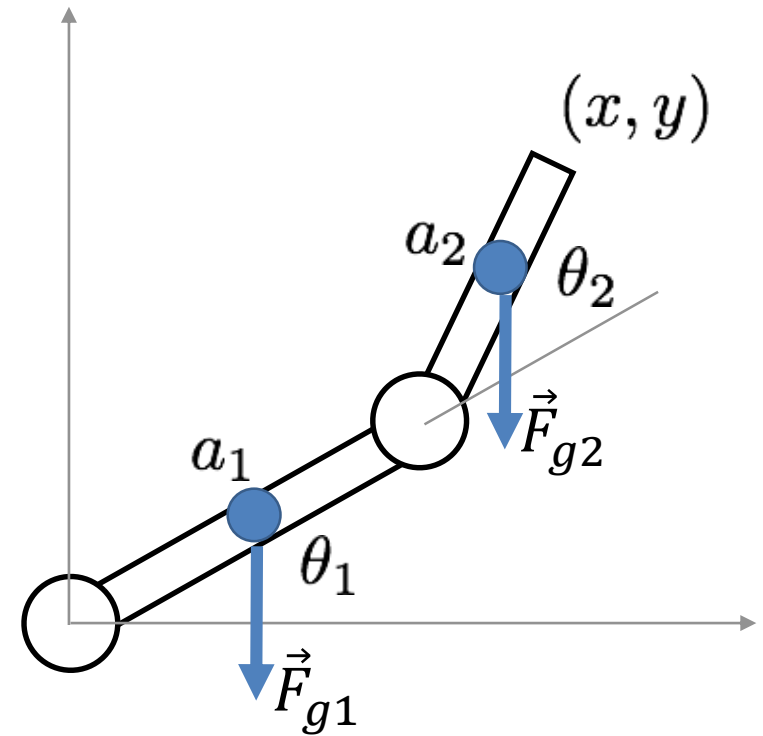
# Gravitational Force/Torque

$$\vec{\tau}^\top d\vec{q} = \vec{F}^\top d\vec{x}$$

$$\vec{\tau}^\top d\vec{q} = \sum_{i=1}^n \vec{F}_{gi}^\top d\vec{x}_i$$

$$\vec{\tau}^\top d\vec{q} = \sum_{i=1}^n \vec{F}_{gi}^\top J_i d\vec{q}$$

$$\vec{\tau} = \sum_{i=1}^n J_i^\top \vec{F}_{gi}$$



## Example: Planar RR

$$\vec{\tau} = \sum_{i=1}^n J_i^\top \vec{F}_{gi} \quad \vec{F}_{gi} = -m_i g \hat{y}$$

$$COM_1 = \begin{bmatrix} (a_1/2)c_1 \\ (a_1/2)s_1 \\ 0 \end{bmatrix}$$

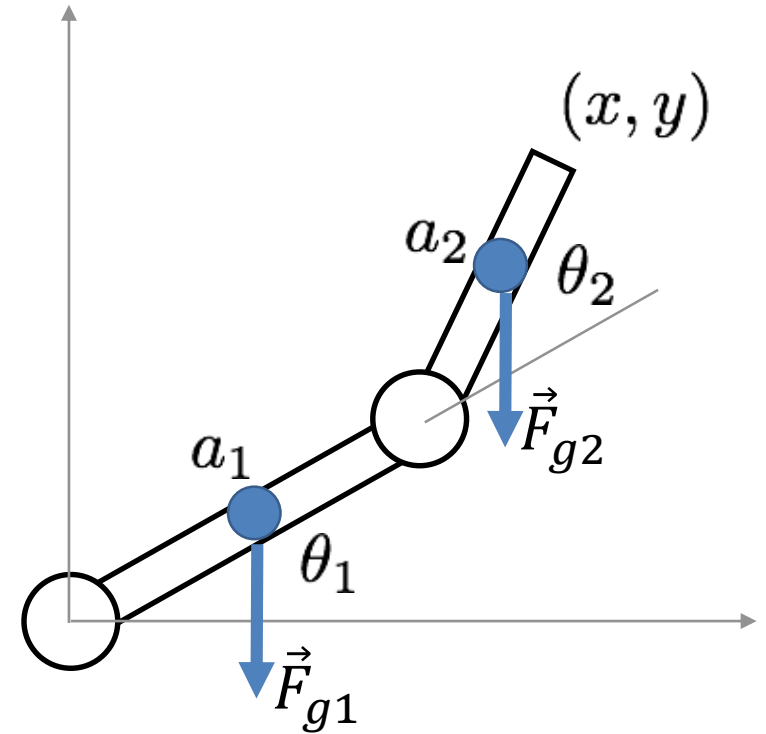
$$COM_2 = \begin{bmatrix} (a_2/2)c_{12} + a_1c_1 \\ (a_2/2)s_{12} + a_1s_1 \\ 0 \end{bmatrix}$$

$$J_1 = \begin{bmatrix} -(a_1/2)s_1 & 0 \\ (a_1/2)c_1 & 0 \\ 0 & 0 \end{bmatrix}$$

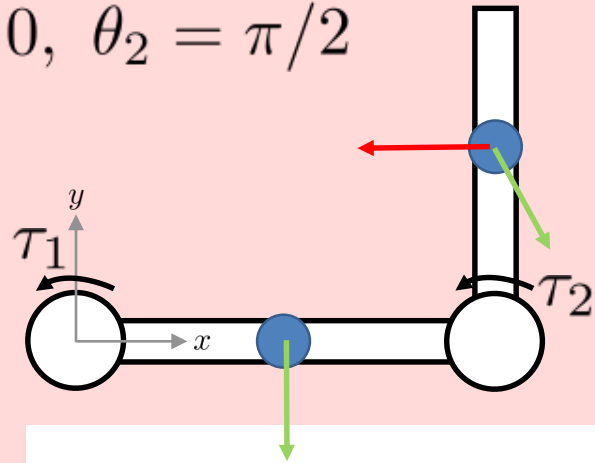
$$J_2 = \begin{bmatrix} -(a_2/2)s_{12} - a_1s_1 & -(a_2/2)s_{12} \\ (a_2/2)c_{12} + a_1c_1 & (a_2/2)c_{12} \\ 0 & 0 \end{bmatrix}$$

$$\vec{\tau} = J_1^\top \vec{F}_{g1} + J_2^\top \vec{F}_{g2}$$

$$\vec{\tau} = \begin{bmatrix} -(a_1/2)c_1 m_1 g \\ 0 \end{bmatrix} + \begin{bmatrix} -((a_2/2)c_{12} + a_1c_1)m_2 g \\ -(a_2/2)c_{12}m_2 g \end{bmatrix}$$



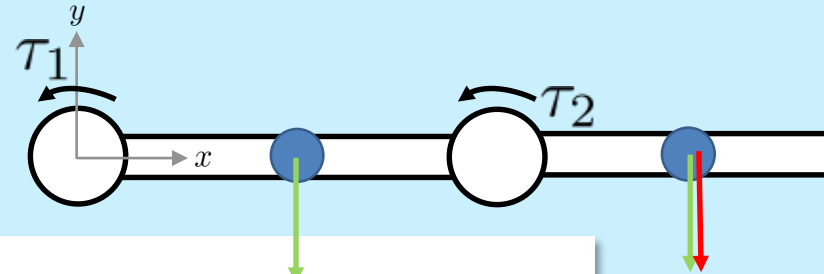
$$\theta_1 = 0, \theta_2 = \pi/2$$



$$\vec{\tau} = \begin{bmatrix} -(a_1/2)c_1 m_1 g \\ 0 \end{bmatrix} + \begin{bmatrix} -((a_2/2)c_{12} + a_1 c_1) m_2 g \\ -(a_2/2)c_{12} m_2 g \end{bmatrix}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -(a_1/2)m_1 g \\ 0 \end{bmatrix} + \begin{bmatrix} -a_1 m_2 g \\ 0 \end{bmatrix}$$

$$\theta_1 = 0, \theta_2 = 0$$



$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} -(a_1/2)m_1 g \\ 0 \end{bmatrix} + \begin{bmatrix} -((a_2/2) + a_1)m_2 g \\ -(a_2/2)m_2 g \end{bmatrix}$$



# Application: Gravity Compensation

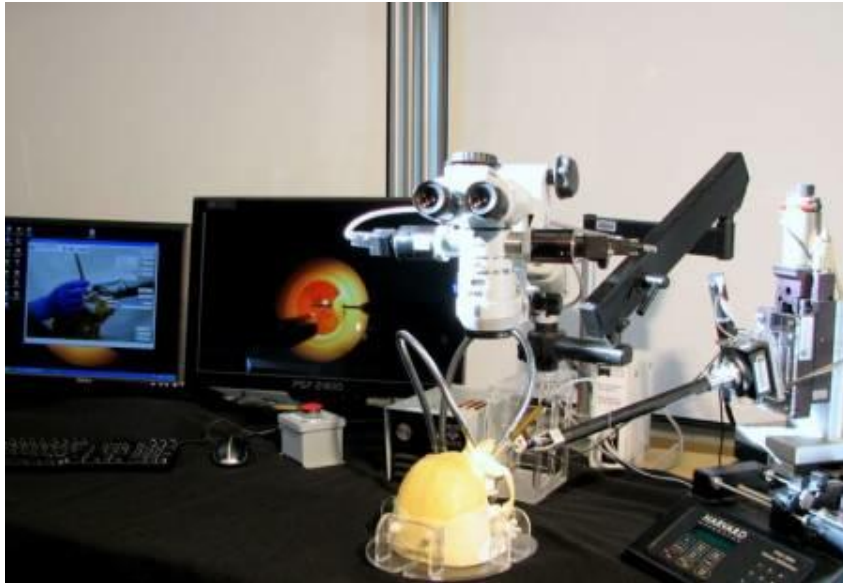
## Strategy:

Determine expected form for gravitational force/torque and cancel out using statics analysis

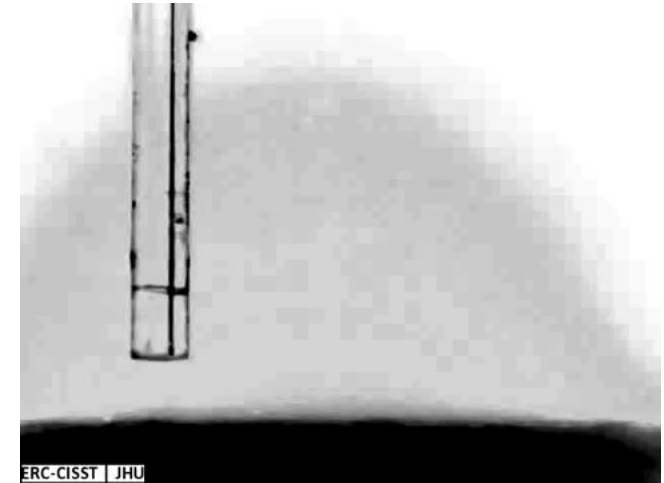
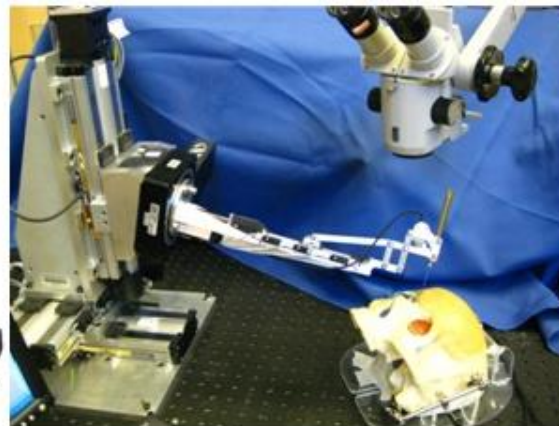
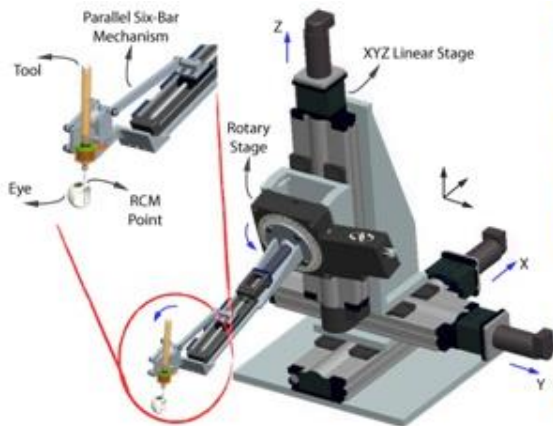
$$\vec{\tau}_{app} = - \sum_{i=1}^n J_i^T \vec{F}_{gi}$$



# Real Example: JHU Steady-Hand Eye Robot

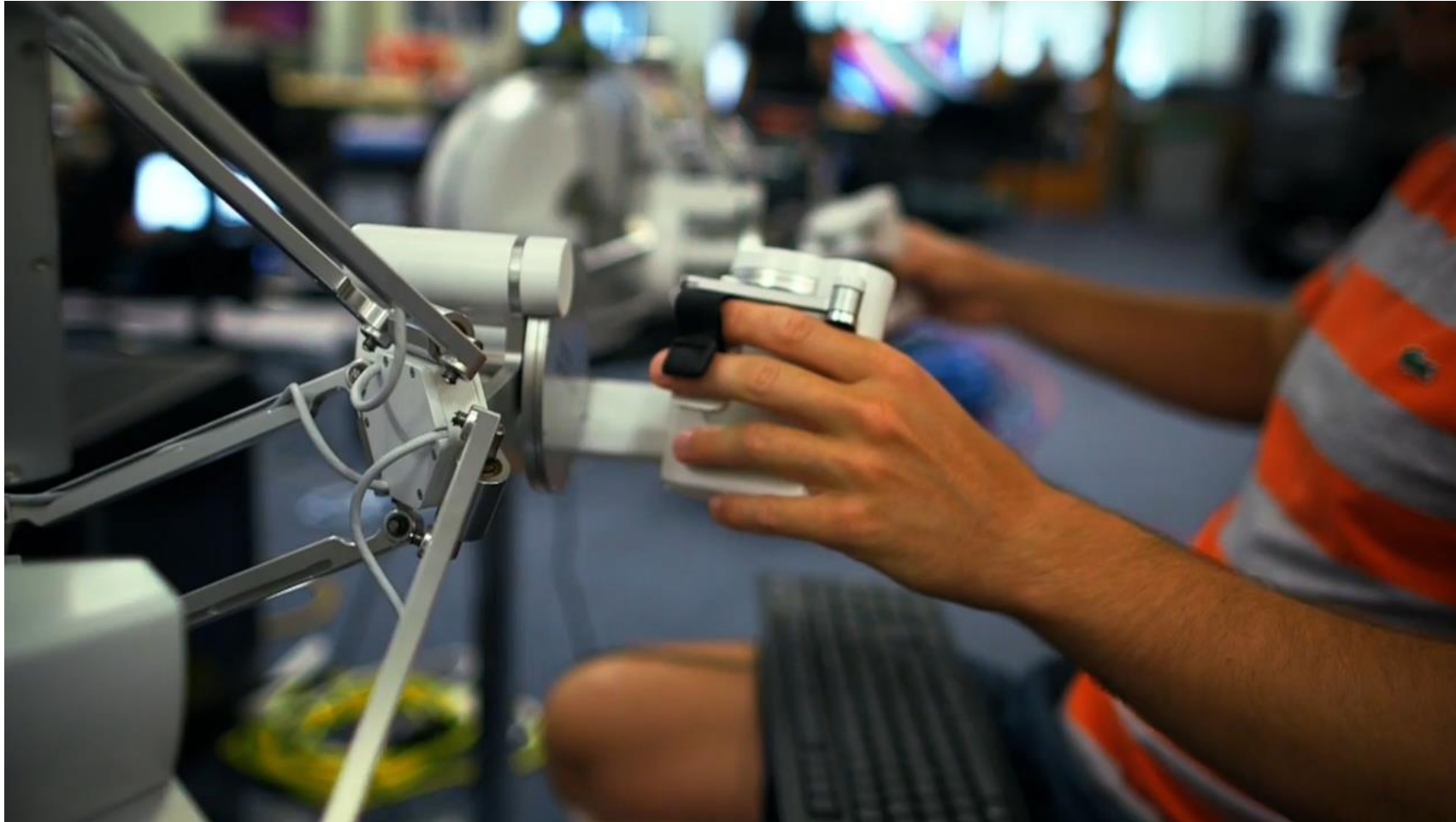


<https://www.youtube.com/watch?v=WiK-CSr-FBs>



<https://www.youtube.com/watch?v=ML45iQB2oU0>

# Teleop Example: Stanford Ocean One

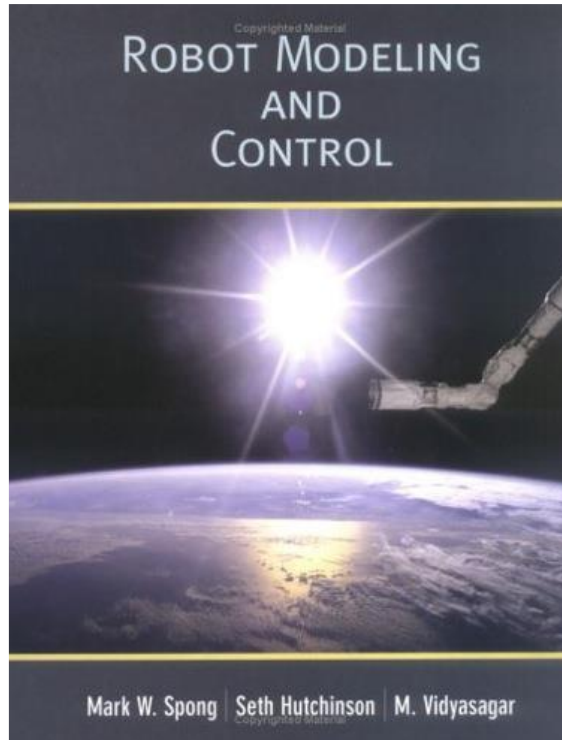




# Simulation Example: 3D CAD with Phantom Omni



# Next time: Potential Fields (Using Jacobians in Planning)



## Chapter 5: Trajectory Planning

- Read 5.2

### Lab 4: Jacobians and Velocity Kinematics

MEAM 520, University of Pennsylvania

October 23, 2020

This lab consists of two portions, with a pre-lab due on Friday, October 30, by midnight (11:59 p.m.) and a lab report due on Friday, November 6, by midnight (11:59 p.m.). Late submissions will be accepted until midnight on Monday 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.

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 4 due 11/6