MEAM 520

Midterm Review and More Robot Hardware

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception La ASP)
MEAM Department, SEAS, University of Pennsyl

GRASP LABORATORY

Lecture 18: October 31, 2013

MEAM 520 Action Item - PUMA Music Video Recording Times — Inbox

From: Naomi Fitter
Subject: **MEAM 520 Action Item - PUMA Music Video Recording Times**
Date: October 31, 2013 1:40:59 AM EDT
To: Katherine Kuchenbecker

Hi all,

I am reaching out personally to those who haven't booked a time for PUMA music video recording yet with an action item for today. By the end of the day, please make sure to, as a team, either:

1) Sign up for an existing time slot to record your PUMA music video on the YouCanBook.Me: http://penn_meam520.youcanbook.me/

-or if none of the existing time slots work for you-

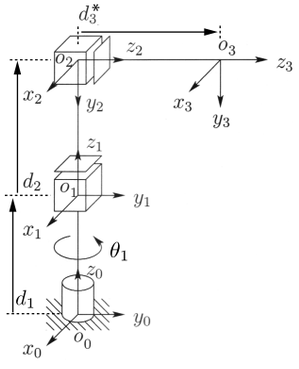
2) Reply to this email with a list of 3 hour-long blocks of time (before the midterm) that could work for your team.

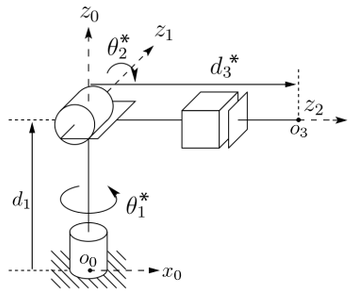
Recording your music video is an important part of Project 1 for this class, and I cannot guarantee the availability of time slots or recording equipment after the midterm exam. That being said, if you respond by midnight tonight, I will work with the other TAs to make sure we accommodate a proposed time for any group with limited availability. I will also be available after class today to field any questions or comments you may have about this process. Thanks for your efforts, and I look forward to seeing the final results of this project!

Thanks,
Naomi

Solutions to Homework 6
Velocity Kinematics and Jacobians

MEAM 520
Introduction to Robotics
University of Pennsylvania
Professor Kuchenbecker
Fall 2013





Solutions to Homework 6

Two copies are on reserve in the Engineering Library

Homework 7:
PUMA 260 Singularities and Manipulability

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

October 24, 2013

This assignment is due on **Sunday, November 3, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted until Wednesday, November 6, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 30%. After the late deadline, no further assignments may be submitted.

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 write down 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. If you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

You may do this assignment either individually or with a partner. If you do this homework with a partner, you may work with ***anyone except your partner from Project 1***. We want everyone in this class to gain experience working with a variety of partners. Consider using the "Search for Teammates!" tool on Piazza.

If you are in a pair, you should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one MATLAB script for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

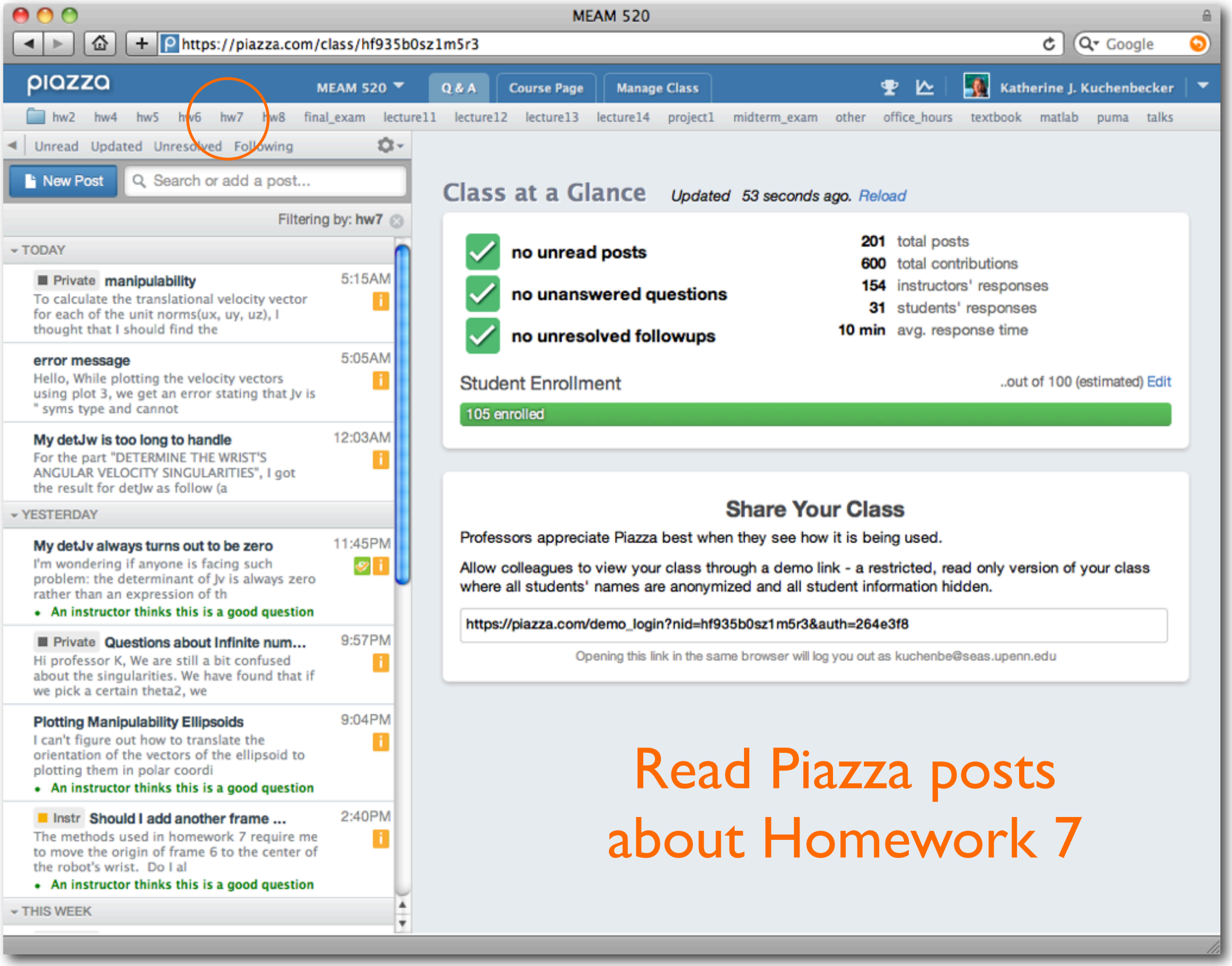
- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty 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 a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.

Homework 7

MATLAB Assignment on PUMA 260 Singularities and Manipulability

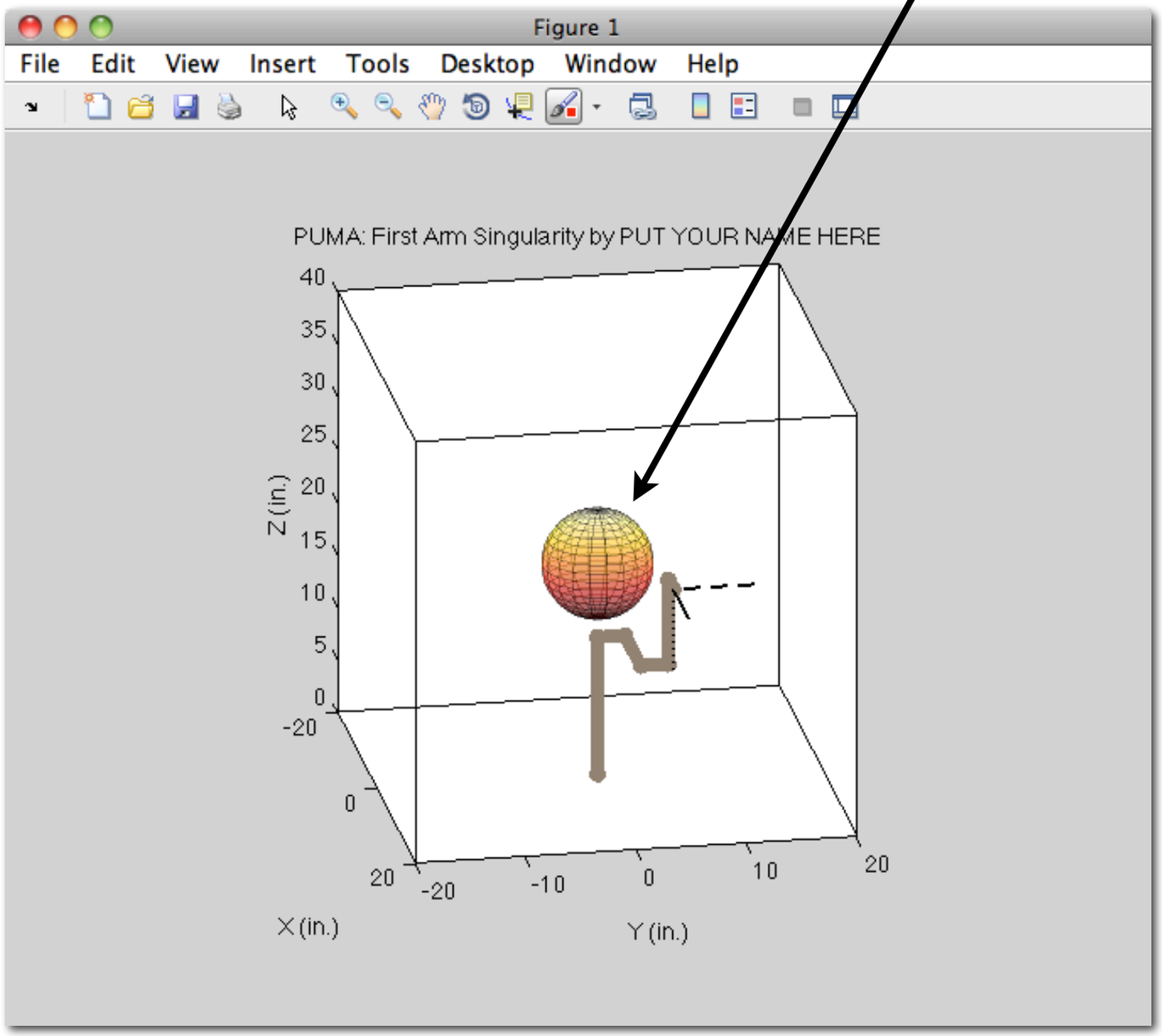
Done individually or with a partner *(not project 1 partner)*

Due Sunday 11/3



Read Piazza posts about Homework 7

The starter code's golden ball is there to show you two key pieces of how to plot the manipulability ellipsoid:



First, it creates a lot of three-element unit vectors for you to use as unit joint velocity vectors.

Second, it shows you how to plot a transparent surface using the surf command.

Editor - /Users/kuchenbe/Documents/teaching/meam 520/assignments/07 manipulability/matlab/plot_puma_starter.m

EDITOR | PUBLISH | VIEW

plot_puma_starter.m

```
110
111     %% CREATE SPHERE - FOR DEMONSTRATION
112
113     % Set the number of segments.
114 -   n = 21;
115
116     % Create theta as a column vector of n angles going between 0 and 2*pi rad.
117 -   theta = linspace(0,2*pi,n);
118
119     % Create phi as a row vector of n angles going between -pi/2 and pi/2 rad.
120 -   phi = linspace(-pi/2,pi/2,n)';
121
122     % Create a set of unit vectors from the angles theta and phi.  Each of the
123     % results ux, uy, and uz is an n x n matrix holding coordinates of a point
124     % on the sphere.  This is the format needed to use the surf command.
125 -   ux = cos(phi) * cos(theta);
126 -   uy = cos(phi) * sin(theta);
127 -   uz = sin(phi) * ones(1,length(theta));
128
129
130     %% PLOT SPHERE - FOR DEMONSTRATION
131
132     % Define an arbitrary center location, in inches.
133 -   cx = 0;
134 -   cy = 0;
135 -   cz = 20;
136
137     % Define an arbitrary sphere radius, in inches.
138 -   r = 5;
139
140     % Plot a scaled sphere at this location.  Make its edges and faces
141     % transparent so you can still see the robot.
142 -   surf(cx + r*ux, cy + r*uy, cz + r*uz,'FaceAlpha',.4,'EdgeAlpha',.4)
143
144     % Set the colormap to something fun.
145 -   colormap hot
146
147     % Call hold off because we are done plotting things.
148 -   hold off
149
150
151
```

plot_puma_starter | Ln 120 Col 31

SHV sections 1.1-1.3, 2.1-2.7, 3.1-3.2, 4.1-4.13, and 5.5,
minus the linear algebra in 4.11-4.12

Midterm Review

What schemes did you use?

What are their main distinguishing features?

motion commands vs. *exploration with feedback*

absolute position vs. *incremental position* vs. *velocity*

Cartesian space vs. *joint space*

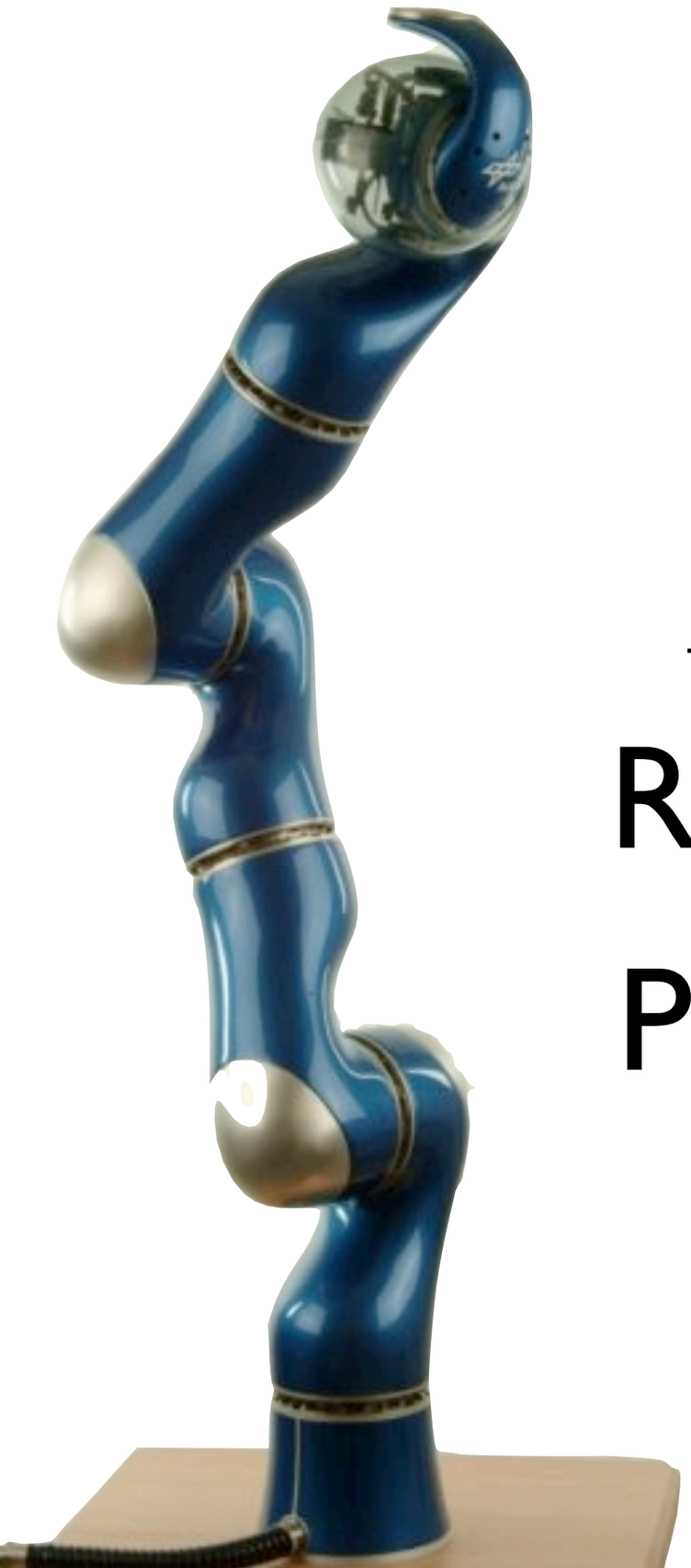
communication code and *error handling*

What is a robot?

a reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks

- Robot Institute of America (RIA)

Chapter 1 of Spong, Hutchinson, and Vidyasagar



Robot manipulators are composed of:

- Rigid **links**
- Connected by **joints**
- To form a **kinematic chain**

There are two types of **joints**:

R
- **Revolute** (rotary), like a hinge, allows relative rotation between two links

P
- **Prismatic** (linear), like a slider, allows a relative linear motion (translation) between two links

Articulated Manipulator (RRR)

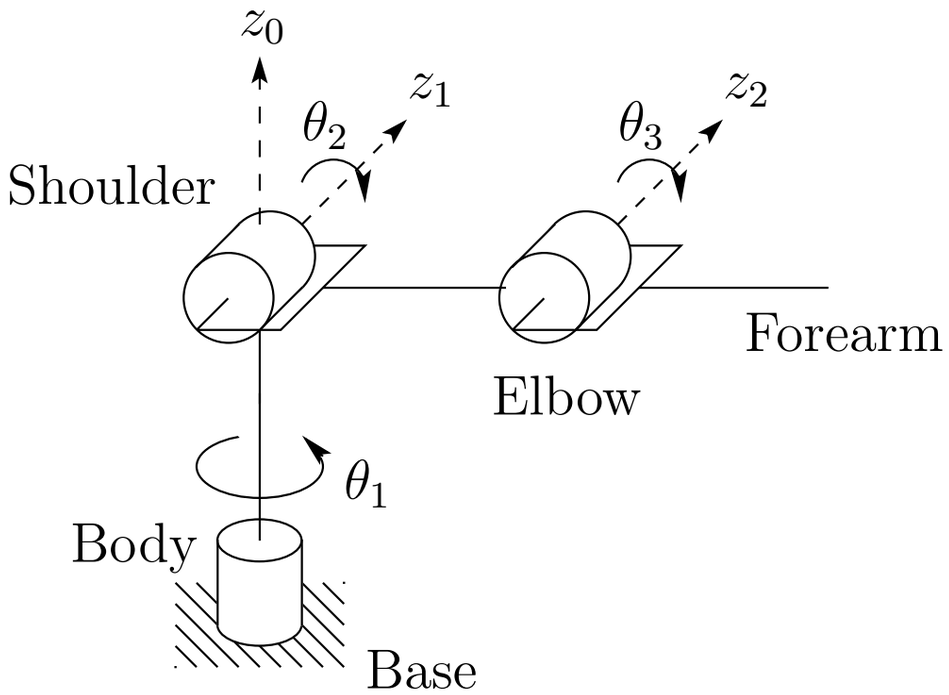


Fig. 1.8 Structure of the elbow manipulator.

Fig. 1.6 The ABB IRB1400 Robot. Photo courtesy of ABB.

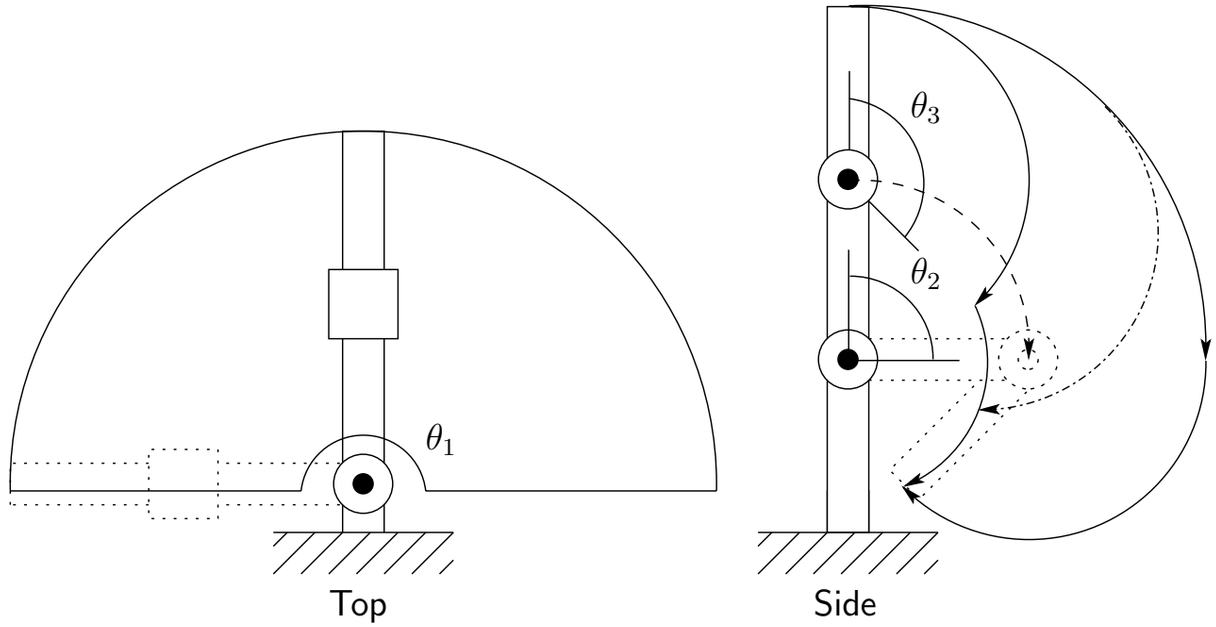


Fig. 1.9 Workspace of the elbow manipulator.

a.k.a. Revolute, Elbow, or Anthropomorphic

Spherical Manipulator (RRP)

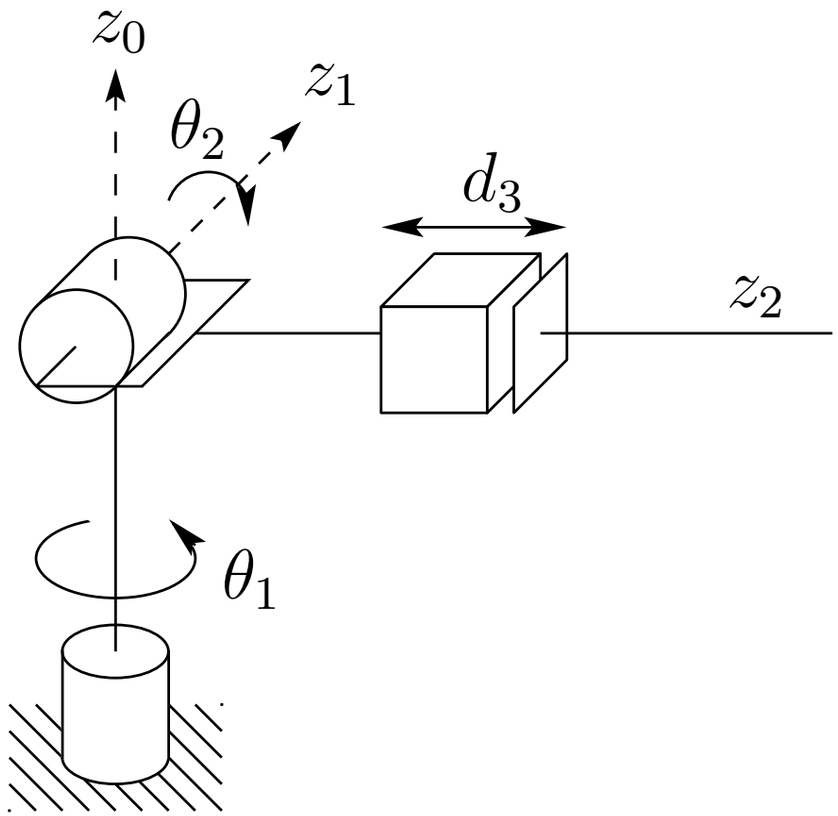


Fig. 1.10 The spherical manipulator.

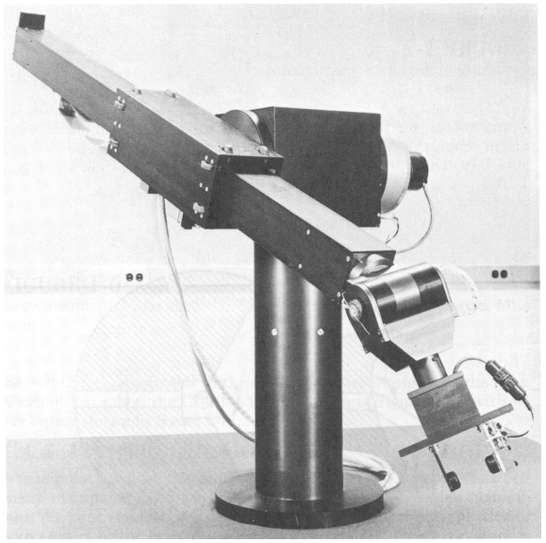


Fig. 1.11 The Stanford Arm. Photo courtesy of the Coordinated Science Lab, University of Illinois at Urbana-Champaign.

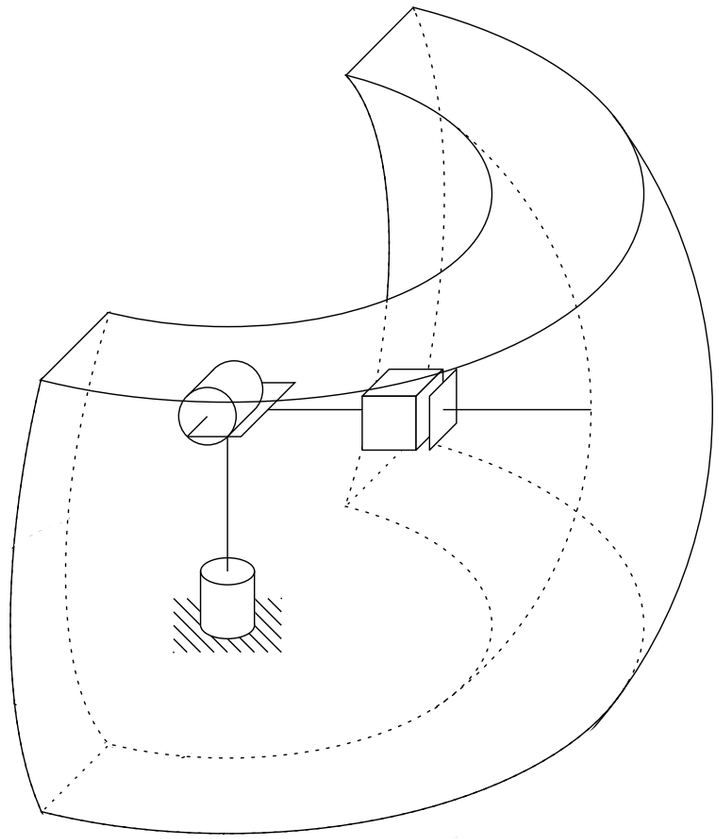


Fig. 1.12 Workspace of the spherical manipulator.

SCARA Manipulator (RRP)

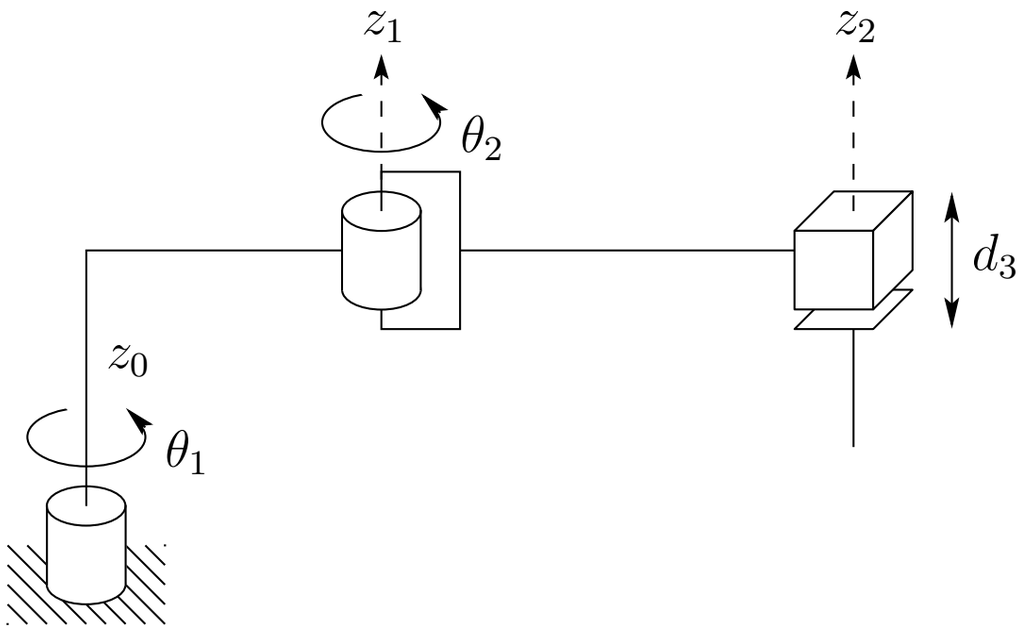


Fig. 1.13 The SCARA (Selective Compliant Articulated Robot for Assembly).

Fig. 1.14 The Epson E2L653S SCARA Robot. Photo Courtesy of Epson.

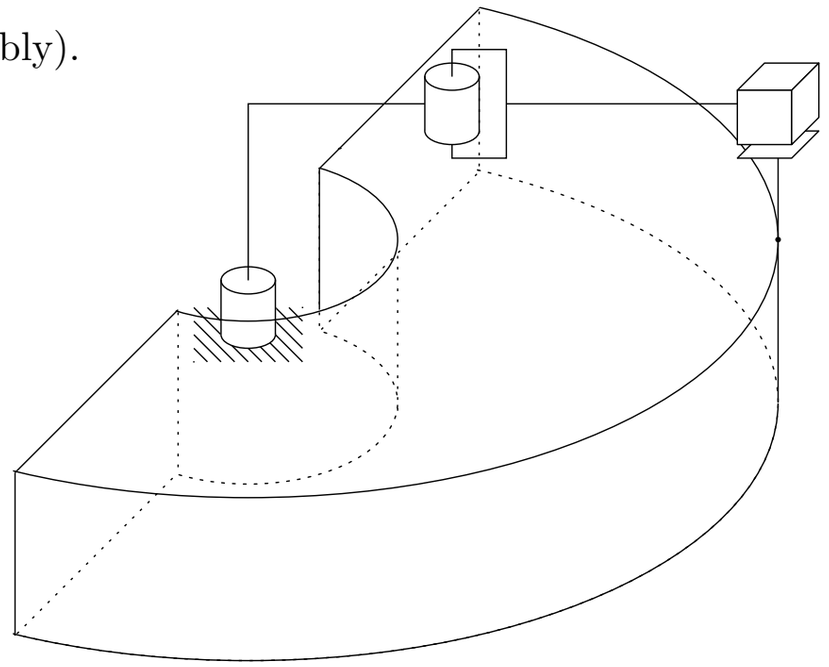


Fig. 1.15 Workspace of the SCARA manipulator.

Cylindrical Manipulator (RPP)

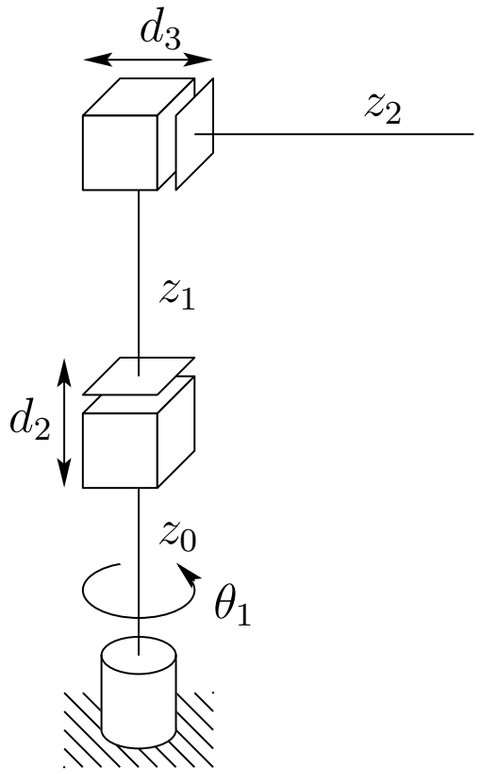


Fig. 1.16 The cylindrical manipulator.

Fig. 1.17 The Seiko RT3300 Robot. Photo courtesy of Seiko.

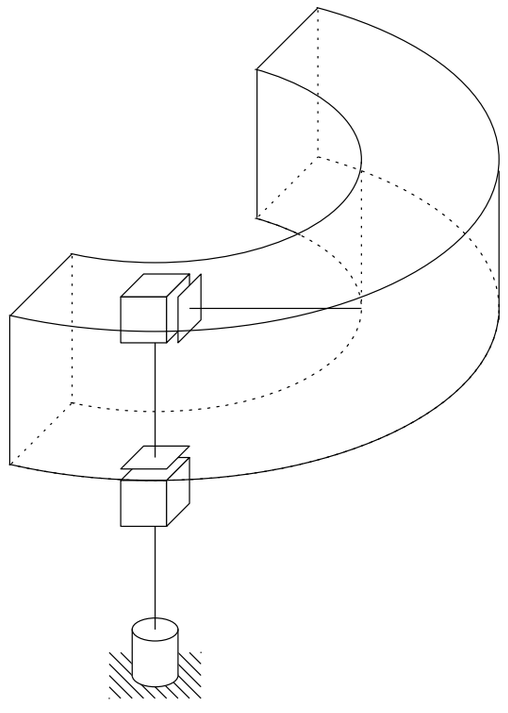


Fig. 1.18 Workspace of the cylindrical manipulator.

Cartesian Manipulator (PPP)

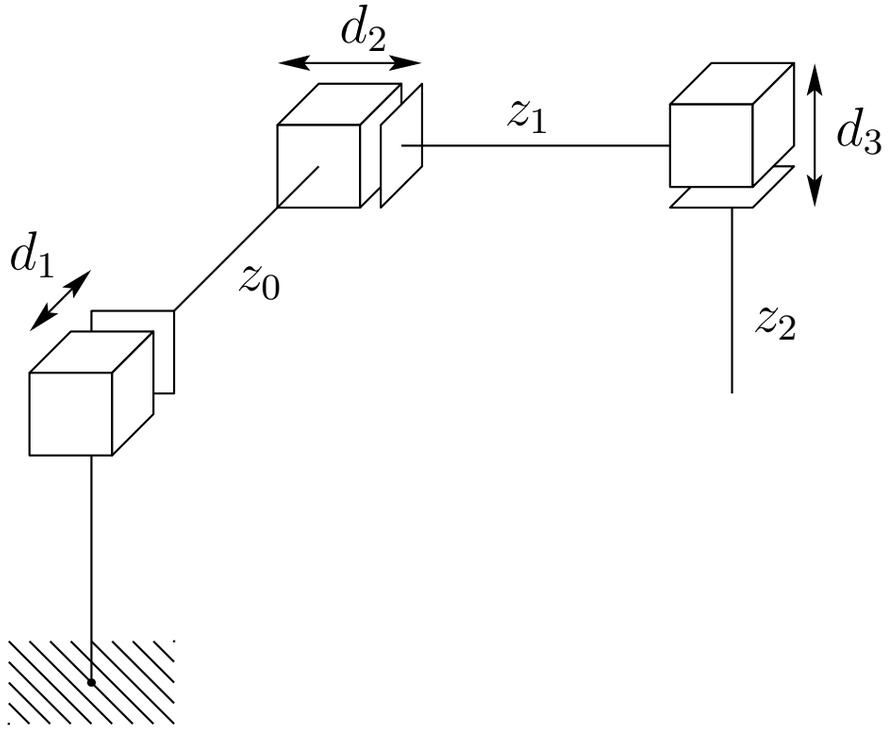


Fig. 1.19 The Cartesian manipulator.

Fig. 1.20 The Epson Cartesian Robot. Photo courtesy of Epson.

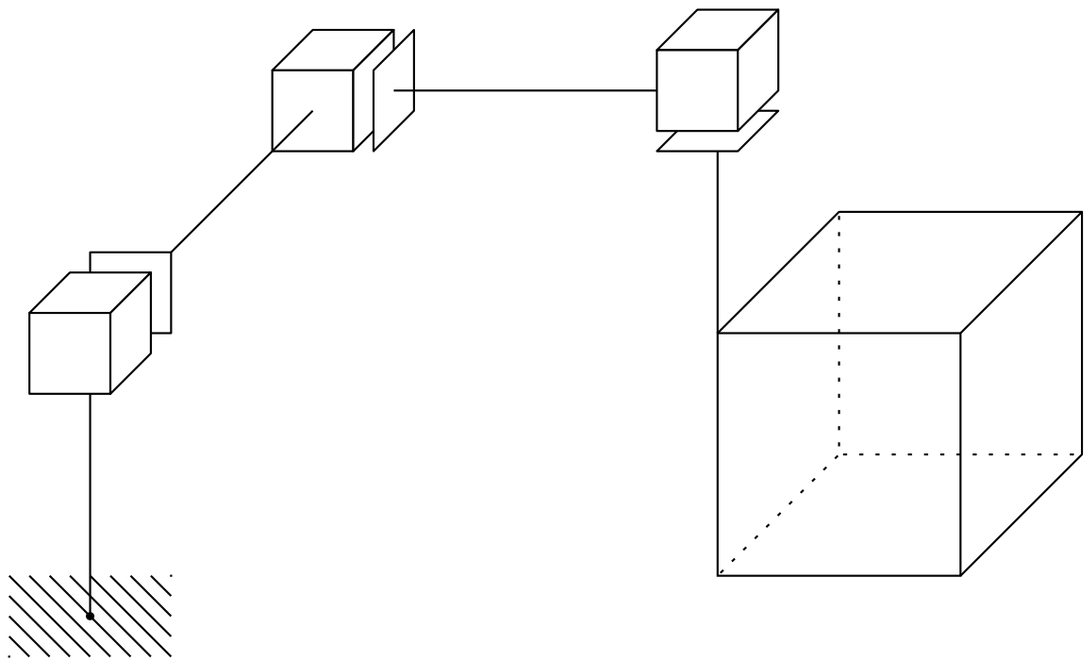
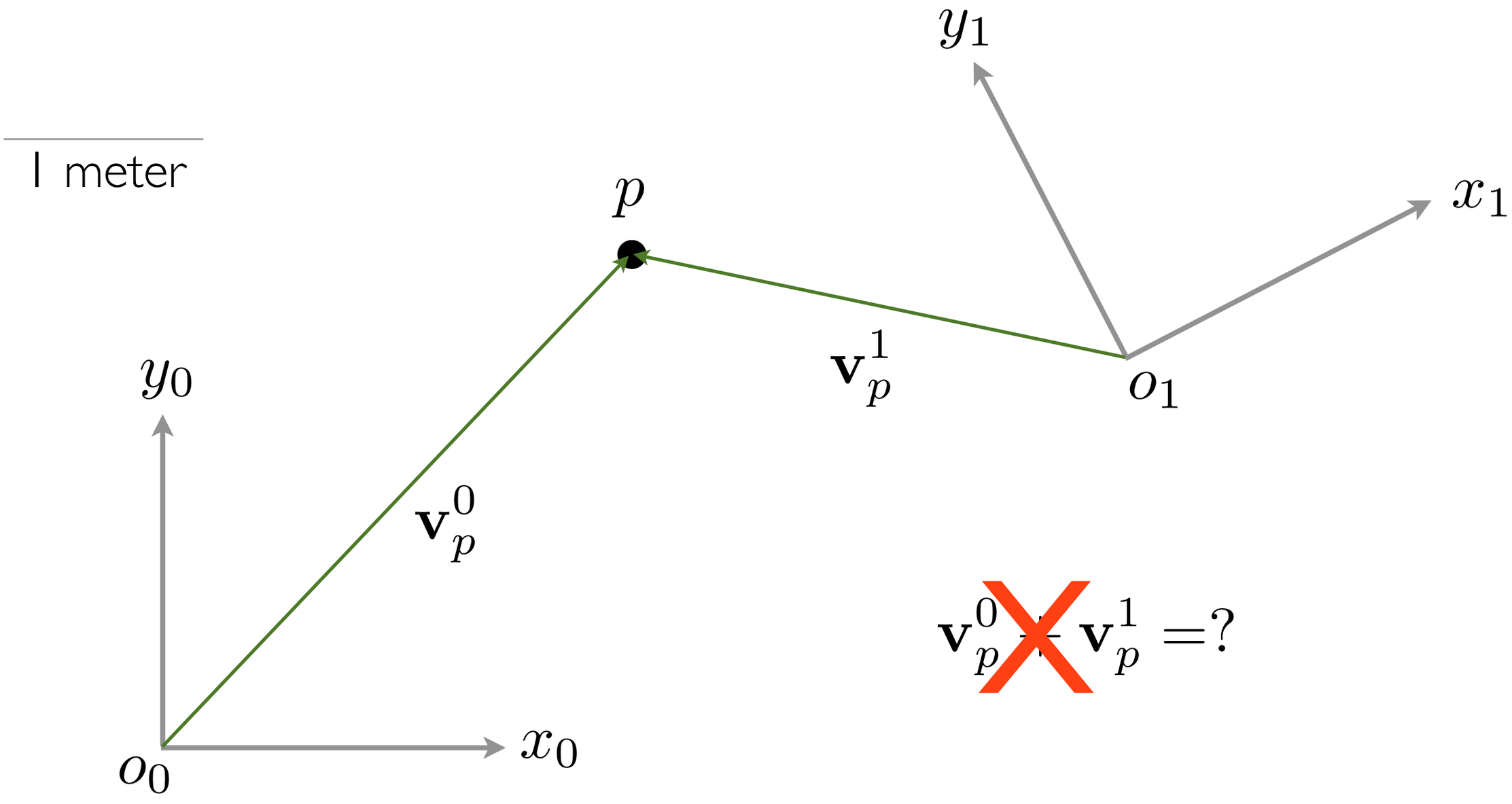


Fig. 1.21 Workspace of the Cartesian manipulator.

Multiple coordinate frames

The coordinate frame being used is designated using **superscript** notation



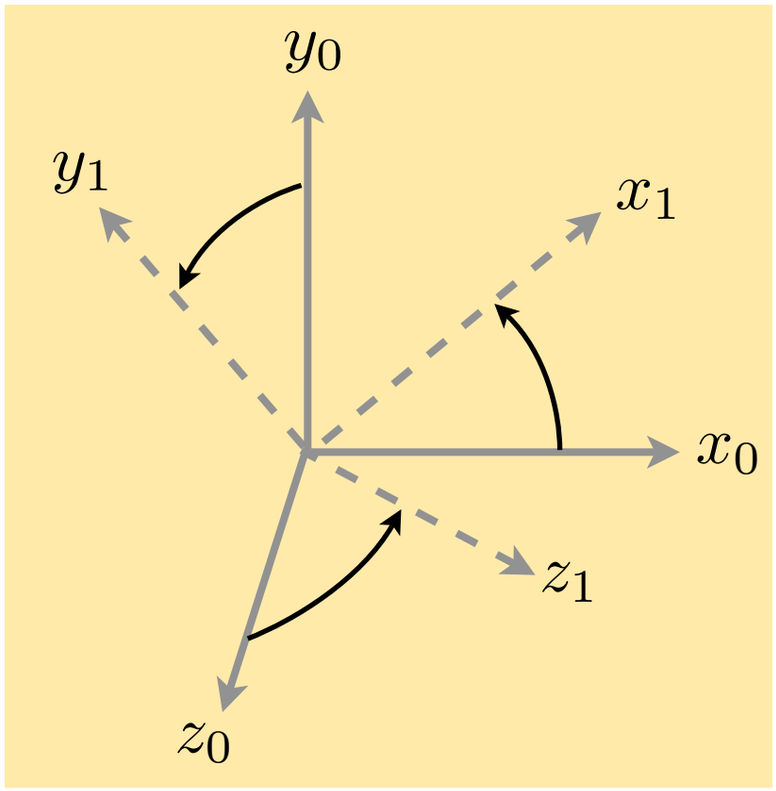
$$\mathbf{v}_p^0 \neq \mathbf{v}_p^1 \; =?$$

To perform algebraic manipulation, you must express vectors in the **same frame** or in **parallel frames**

Rotation Matrices

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Rotation Matrices - Interpretation 1 of 3



Represents the orientation of one coordinate frame with respect to another frame

$$\mathbf{R}_1^0 = \begin{bmatrix} x_1 \cdot x_0 & y_1 \cdot x_0 & z_1 \cdot x_0 \\ x_1 \cdot y_0 & y_1 \cdot y_0 & z_1 \cdot y_0 \\ x_1 \cdot z_0 & y_1 \cdot z_0 & z_1 \cdot z_0 \end{bmatrix}$$

Orientation of frame 1 w.r.t. frame 0

columns are of unit length
columns are mutually orthogonal
$\det \mathbf{R}_1^0 = +1$

$\mathbf{R}_0^1 = ?$ $\quad \mathbf{R}_0^1 = \left(\mathbf{R}_1^0\right)^T$

$\left(\mathbf{R}_1^0\right)^T = \left(\mathbf{R}_1^0\right)^{-1}$

Three-Dimensional Coordinate Rotations

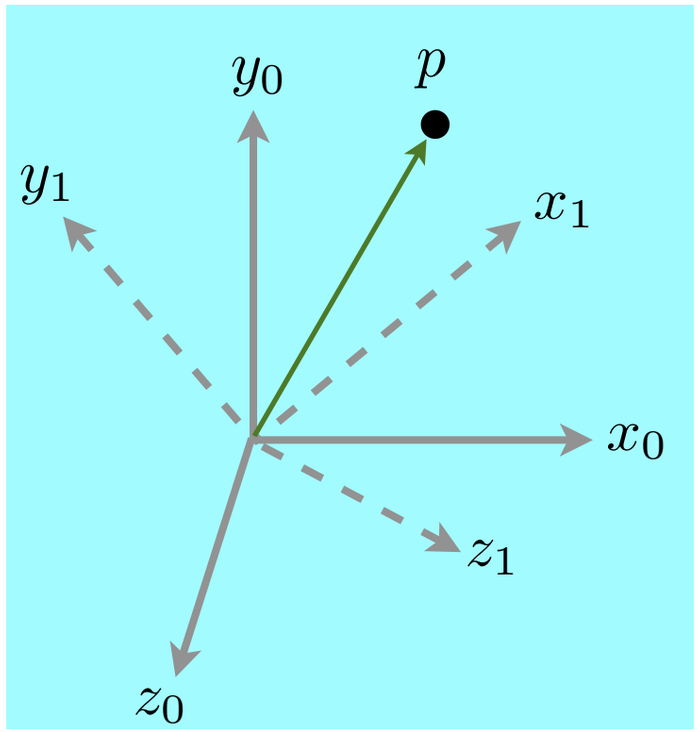
The **basic rotation matrices**
define rotations about the three
coordinate axes

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Matrices - Interpretation 2 of 3



Coordinate transformation relating the coordinates of a point p in two different frames

$\mathbf{R}_1^0 \qquad \mathbf{v}_p^1$

$\mathbf{v}_p^0 = ?$

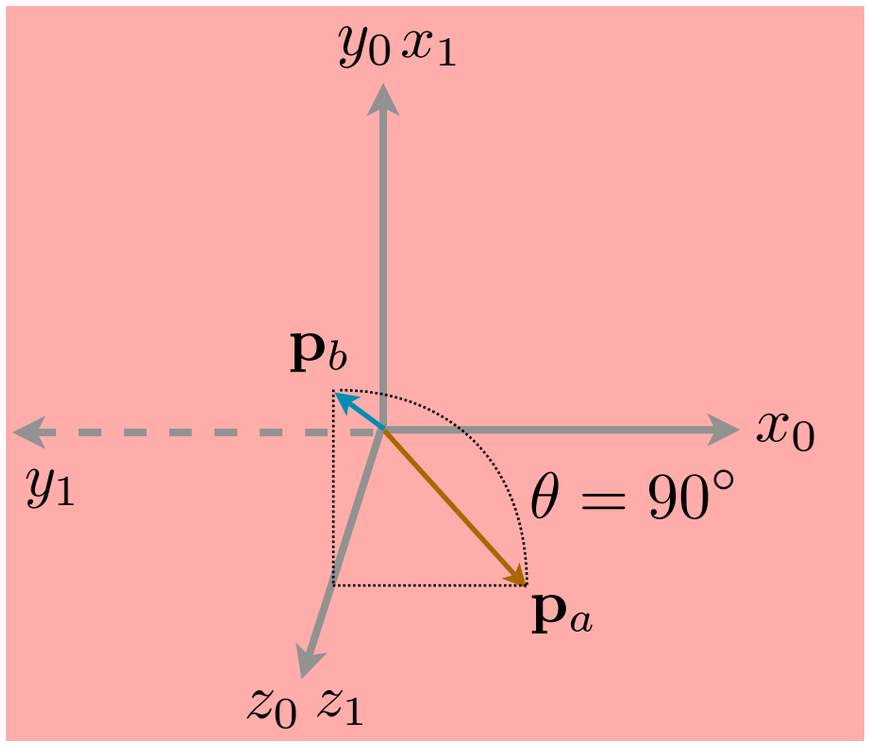
$$\boxed{\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1}$$

Subscript and superscript cancel

$\mathbf{v}_p^1 = ?$ $\qquad \left(\mathbf{R}_1^0\right)^{-1} \mathbf{v}_p^0 = \cancel{\left(\mathbf{R}_1^0\right)^{-1} \mathbf{R}_1^0} \mathbf{v}_p^1$ $\qquad \mathbf{v}_p^1 = \left(\mathbf{R}_1^0\right)^T \mathbf{v}_p^0$

$$\boxed{\mathbf{v}_p^1 = \mathbf{R}_0^1 \mathbf{v}_p^0}$$

Rotation Matrices - Interpretation 3 of 3



Operator taking a vector and rotating it to yield a new vector in the same coordinate frame

$$\mathbf{p}_a^0 \qquad\qquad \mathbf{R}_1^0$$

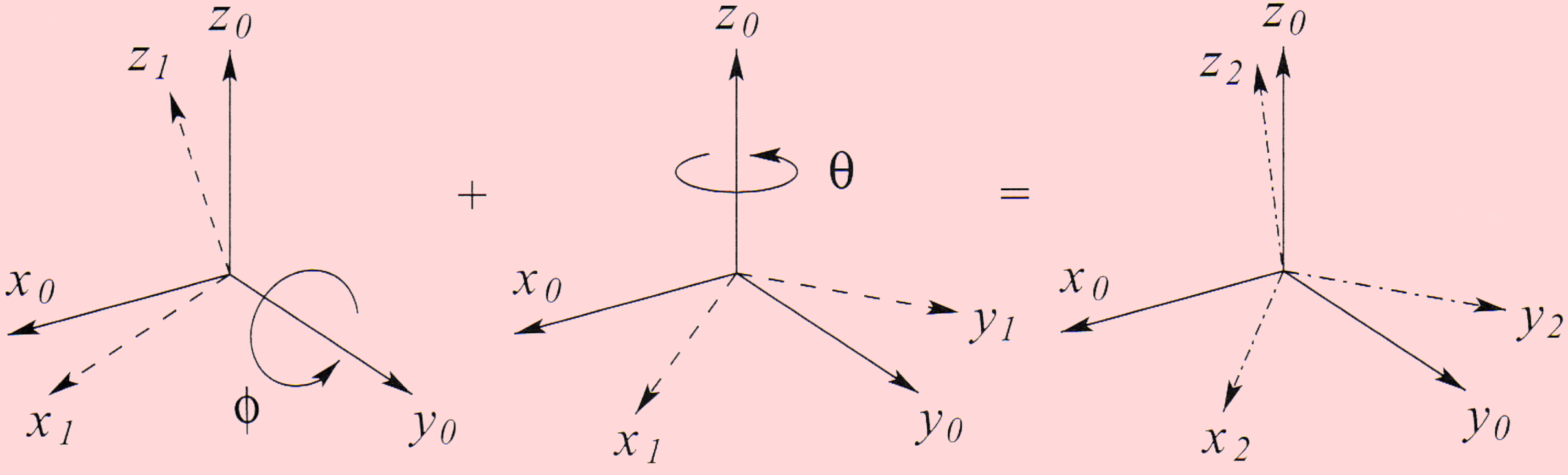
$$\mathbf{p}_b^0 = ?$$

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_b^1$$

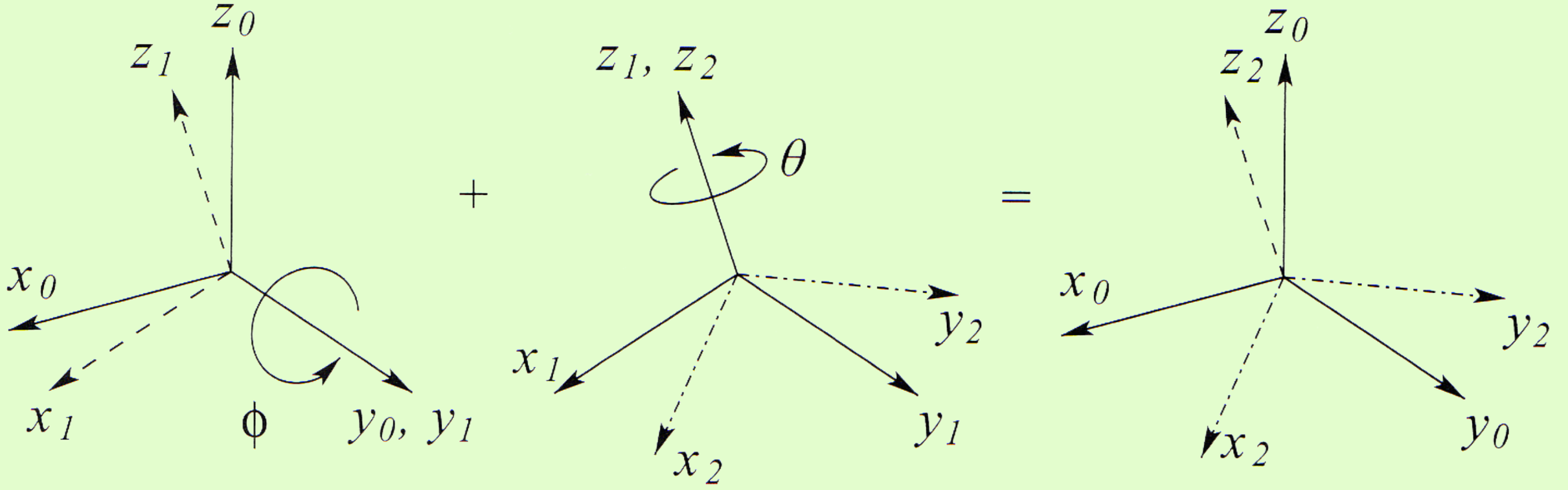
$$\mathbf{p}_b^1 = \mathbf{p}_a^0$$

$$\mathbf{p}_b^0 = \mathbf{R}_1^0\, \mathbf{p}_a^0 \qquad \boxed{\mathbf{p}_b^0 = \mathbf{R}\, \mathbf{p}_a^0}$$

successive rotation about fixed frame? **pre-multiply** $\mathbf{R}_2^0 = \mathbf{R}\,\mathbf{R}_1^0$



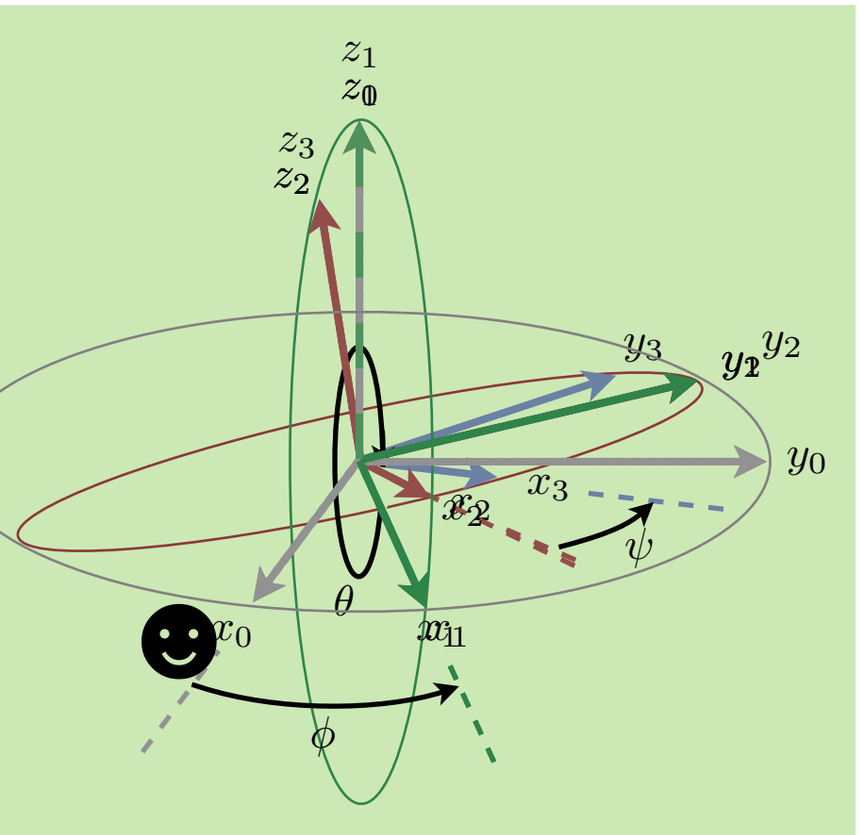
successive rotation about intermediate frame? **post-multiply** $\mathbf{R}_2^0 = \mathbf{R}_1^0\,\mathbf{R}_2^1$



Parameterizations of Rotation Matrices

Euler Angles

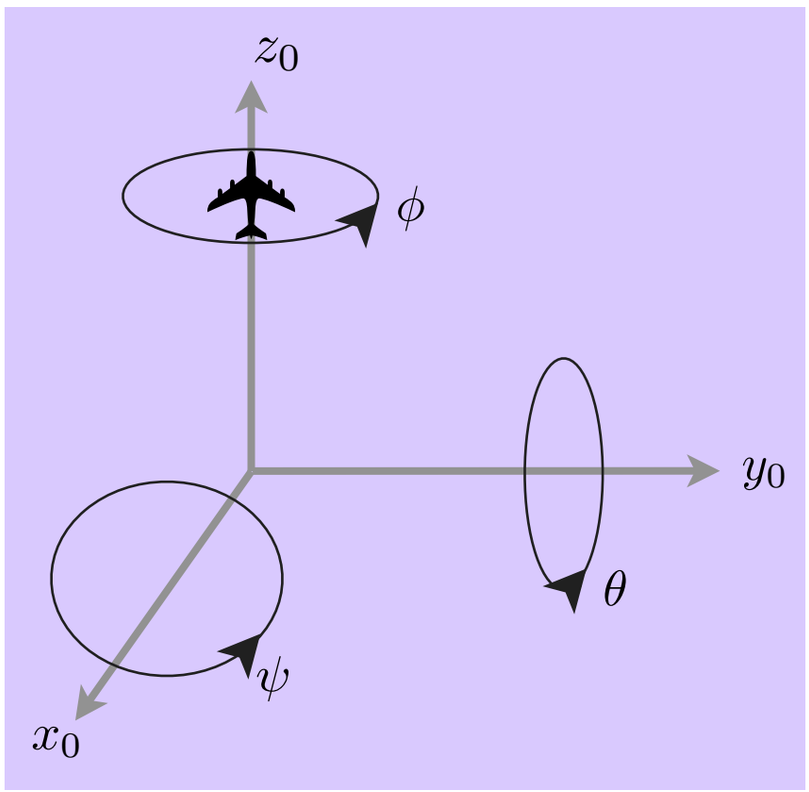
three rotations
about intermediate
frames



Our book uses ZYZ

Roll, Pitch, Yaw Angles

three rotations
about the fixed
frame



Our book uses XYZ

Axis/Angle

a unit vector
(axis)
and one angle

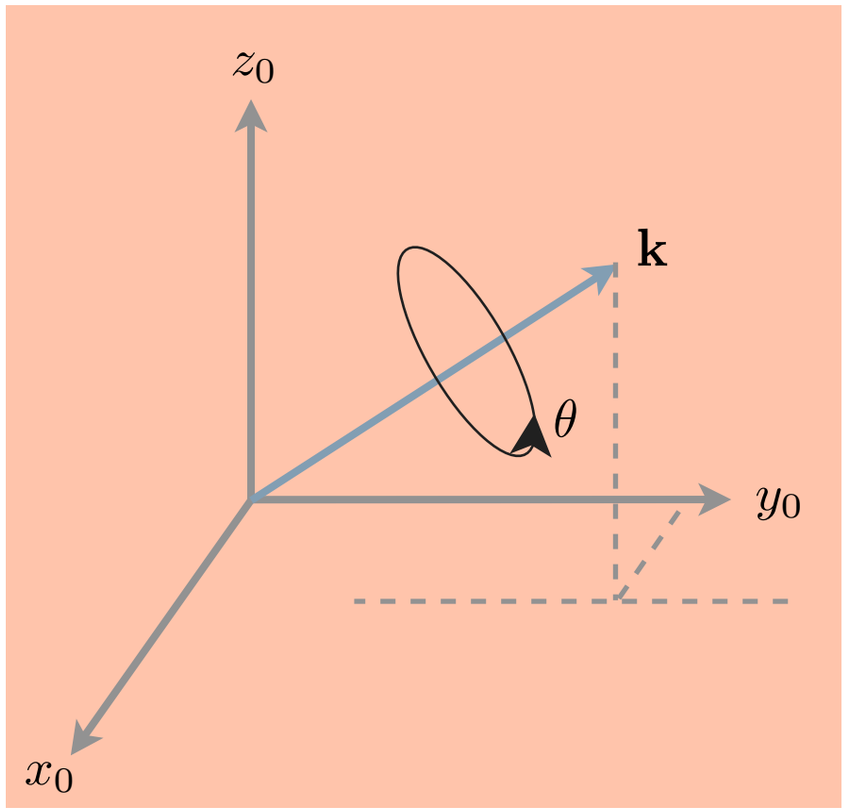


Figure 1

File Edit View Insert Tools Desktop Window Help

Press any key to continue.

$$R_1^0 = \begin{matrix} -0.1620 & -0.9324 & -0.3231 \\ 0.7479 & -0.3296 & 0.5762 \\ -0.6437 & -0.1483 & 0.7507 \end{matrix}$$

φ_a = 119.3 degrees

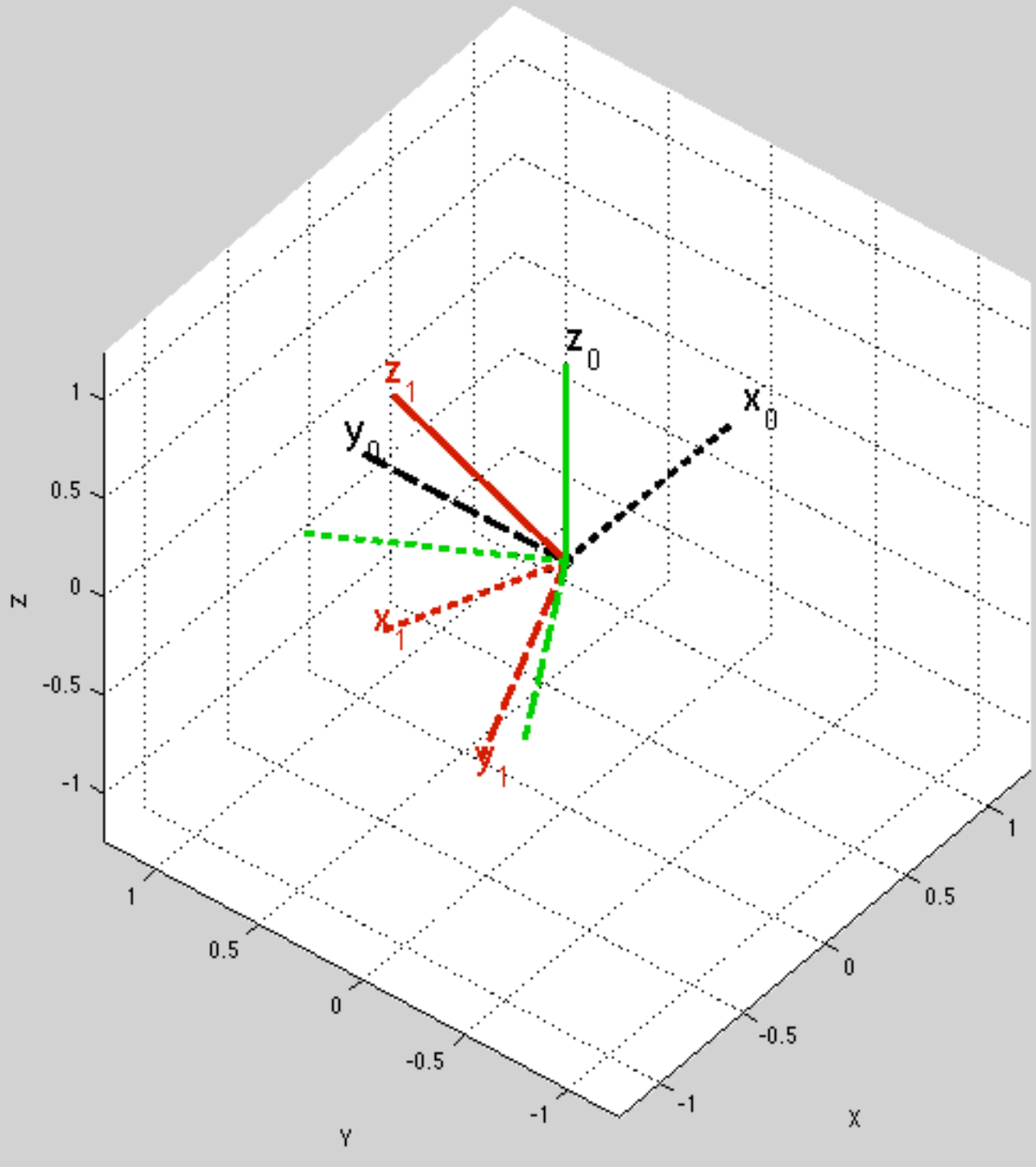
θ_a = 41.3 degrees

ψ_a = -13.0 degrees

φ_b = -60.7 degrees

θ_b = -41.3 degrees

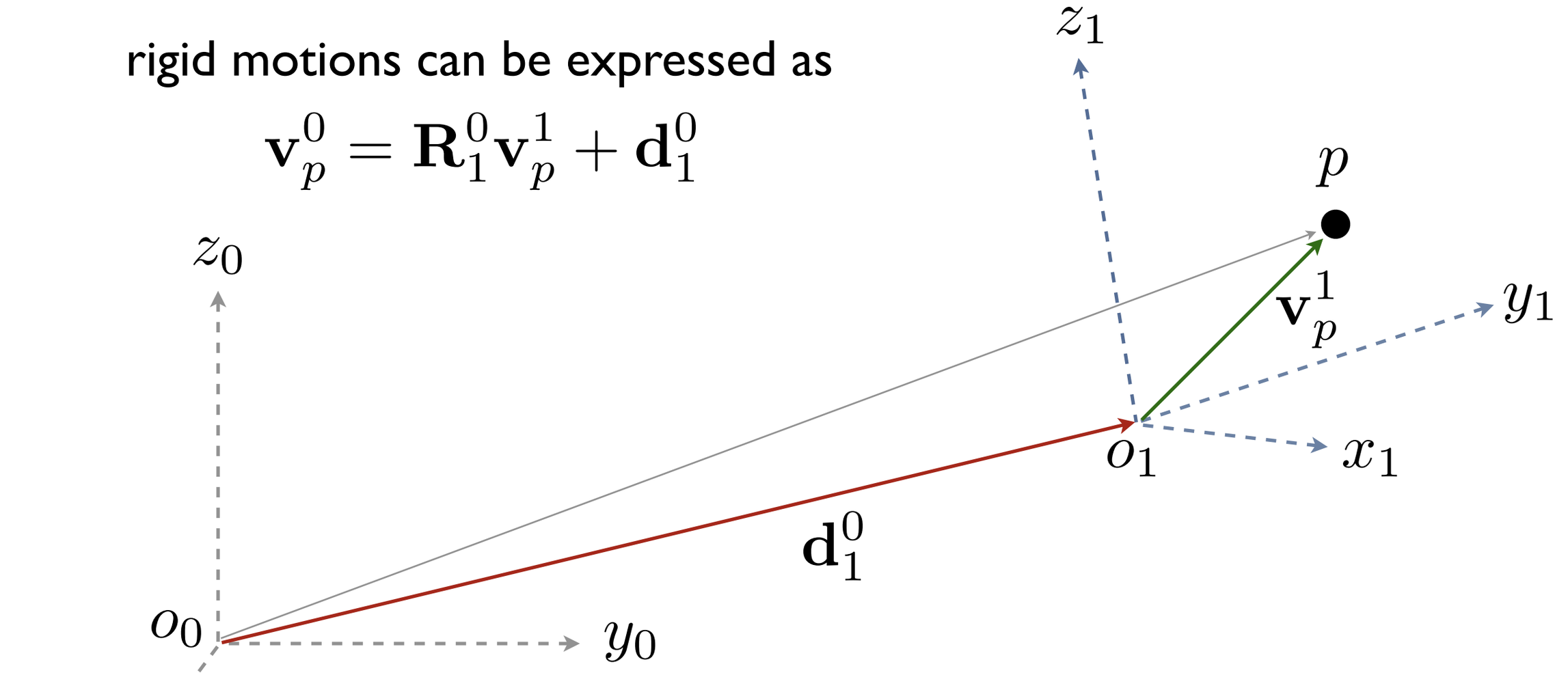
ψ_b = 167.0 degrees



a **rigid motion** couples pure translation with pure rotation

rigid motions can be expressed as

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \mathbf{v}_p^1 + \mathbf{d}_1^0$$



$$\mathbf{H} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \quad \mathbf{v}_p^0 = \mathbf{H}_1^0 \, \mathbf{v}_p^1$$

$\mathbf{R}_1^0$ $\mathbf{d}_1^0$ $\mathbf{v}_p^1$

composition of multiple transforms is the same as for rotation matrices:

post-multiply when successive rotations are relative to intermediate frames

$$\mathbf{H}_2^0 = \mathbf{H}_1^0 \, \mathbf{H}_2^1$$

pre-multiply when successive rotations are relative to the first fixed frame

$$\mathbf{H}_2^0 = \mathbf{H} \, \mathbf{H}_1^0$$

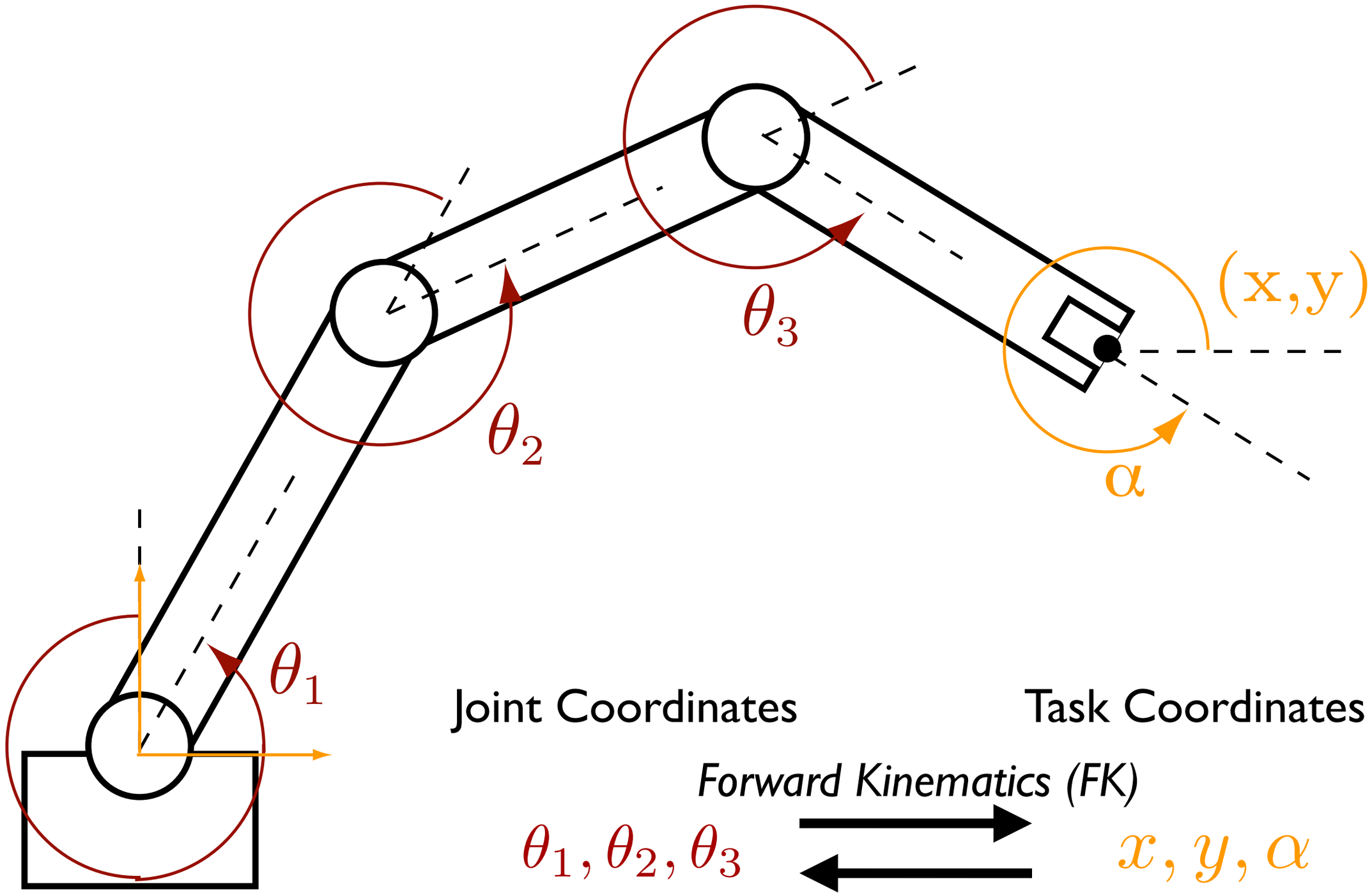
Composition (intermediate frame)

$$\mathbf{H}_2^0 = \mathbf{H}_1^0\,\mathbf{H}_2^1 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_2^1 & \mathbf{d}_2^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{R}_1^0\mathbf{d}_2^1 + \mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Inverse Transform

$$\mathbf{H}_0^1 = \begin{bmatrix} \mathbf{R}_0^1 & \mathbf{d}_0^1 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_1^0)^\top & -(\mathbf{R}_1^0)^\top\mathbf{d}_1^0 \\ \mathbf{0} & 1 \end{bmatrix}$$

Joint and Task Coordinates



Joint Coordinates

Task Coordinates

Forward Kinematics (FK)

$\theta_1, \theta_2, \theta_3$ ⇄ x, y, α

Inverse Kinematics (IK)

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains

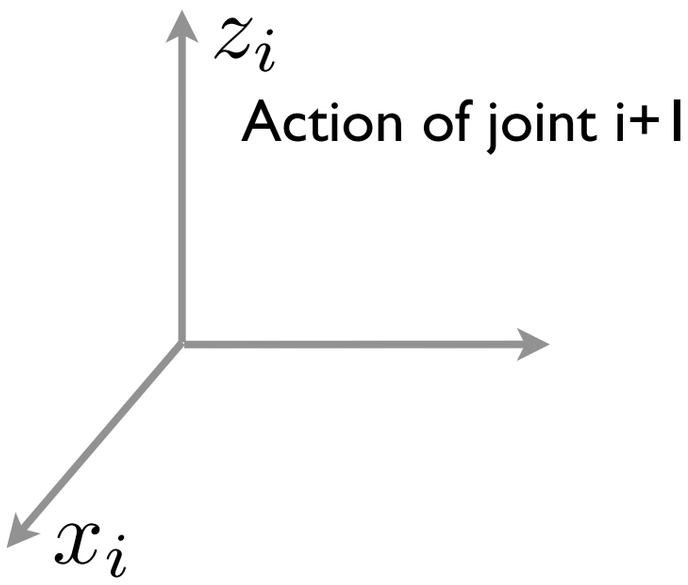
Start by drawing a schematic of the robot in its zero pose.

Then attach one frame to each link:

the joint variable for joint i+1 acts along/around z_i

the orientation of z_i determines the joint angle's positive direction

the axis x_i is perpendicular to, and intersects, z_{i-1}



Must also choose a location for the base (0) frame:

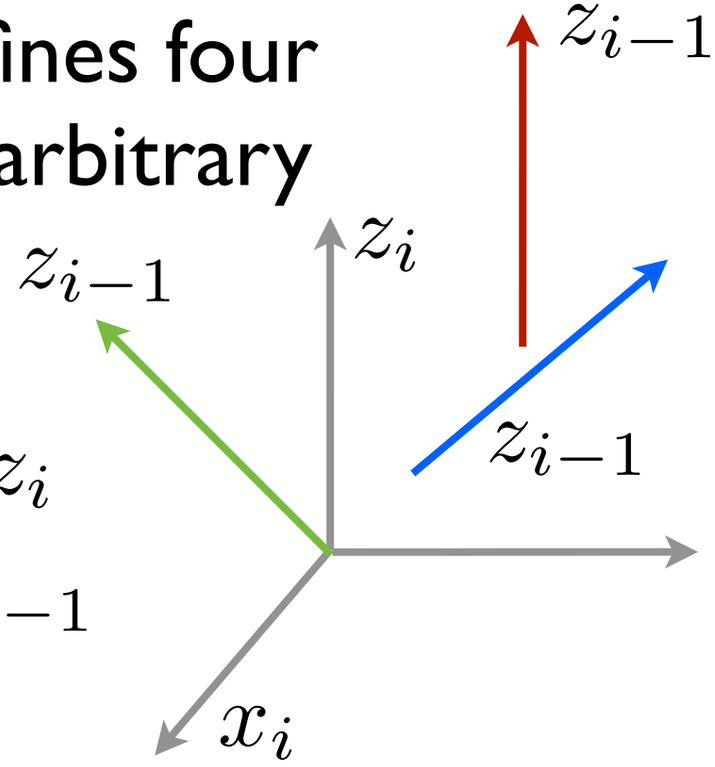
Origin must be on z_0.

x_0 and y_0 are chosen for convenience.

Takes you from previous (i-1) frame to this (i) frame

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

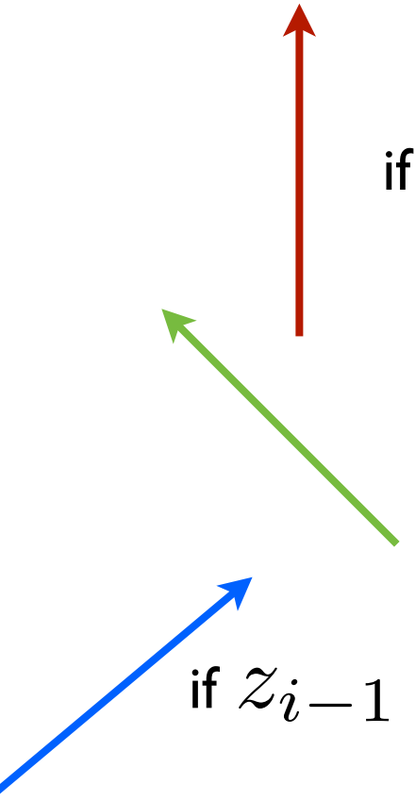
The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains



the joint variable for joint i+1 acts along/around z_i

the axis x_i is perpendicular to, and intersects, z_{i-1}

the following conventions make x placement easier (p. 82 in SHV):



if z_{i-1} is parallel to z_i	orient x_i toward z_i
if z_{i-1} intersects z_i	orient x_i normal to the plane formed by z_{i-1} and z_i
if z_{i-1} is not coplanar with z_i	orient x_i along normal with z_{i-1}, toward z_i

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

The **Denavit-Hartenberg convention** defines four parameters and some rules to help characterize arbitrary kinematic chains (see page 110-111 for parameter definitions)

DH parameters are usually written in this order, but I prefer the opposite order.

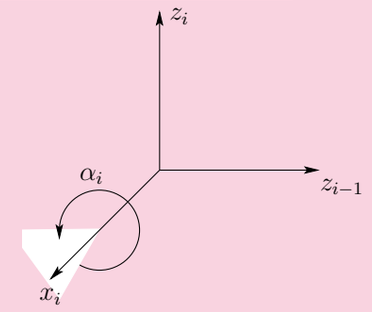
x step

a_i

Link Length the distance between z_{i-1} and z_i, measured along x_i

α_i

Link Twist the angle between z_{i-1} and z_i, measured in the plane normal to x_i (right-hand rule around x_i)



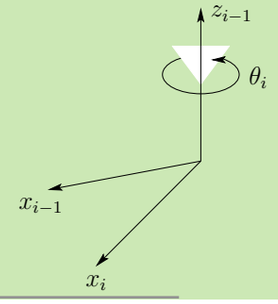
z step

d_i

Link Offset the distance between x_{i-1} and x_i, measured along z_{i-1}

θ_i

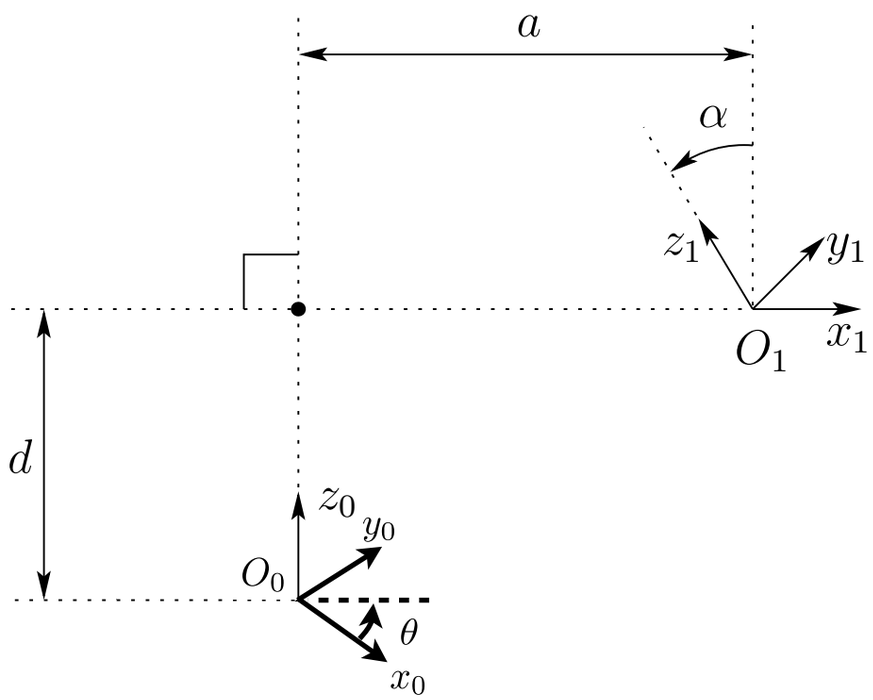
Joint Angle the angle between x_{i-1} and x_i, measured in the plane normal to z_{i-1} (right-hand rule around z_{i-1})



The Denavit-Hartenberg (DH) Convention for Manipulator Forward Kinematics

Given the design of my robot and the current values for its joint variables, where is the end-effector, and how is it oriented?

$$\mathbf{T}_n^0 = A_1(q_1) \cdots A_n(q_n)$$



$$A_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,\mathrm{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Link	a_i	α_i	d_i	θ_i

DH Coordinate Frame Assumptions

(DH1) The axis x_1 is perpendicular to the axis z_0.

(DH2) The axis x_1 intersects the axis z_0.

In what order are the transformations applied?

Intermediate frames
Post-multiply

$$A_i = Rot_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,Rot_{x,\alpha_i}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate around z by theta

Translate along z by d

Translate along new x by a

Rotate around new x by alpha

Denavit & Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," ASME Journal of Applied Mechanics, June 1955

Great List of DH Steps on page 110-111 in SHV

Procedure for Deriving the Forward Kinematics of a Manipulator
Following the Denavit-Hartenberg (DH) Convention
From "Robot Modeling and Control" by Spong, Hutchinson, and Vidyasagar

Step l: Locate and label the joint axes $z_0, \ldots, z_{n-1}$.

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$, perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1} locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n-th joint is revolute, set $z_n = a$ along the direction z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the center of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

Step 7: Create a table of link parameters a_i, d_i, α_i, θ_i.

(fixed!)

$a_i =$ distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i

$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

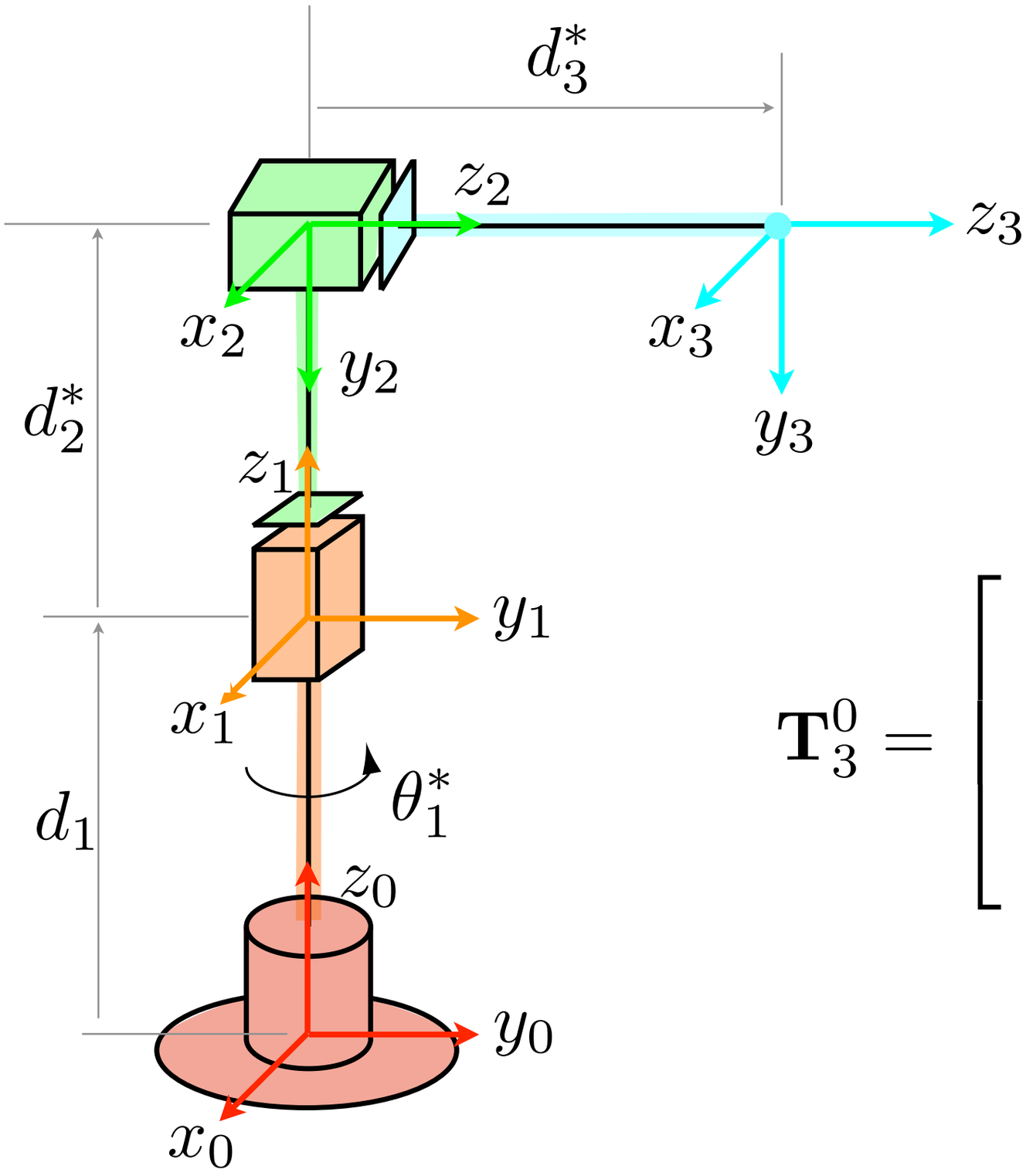
$\alpha_i =$ the angle between z_{i-1} and z_i measured about x_i.

$\theta_i =$ the angle between x_{i-1} and x_i measured about z_{i-1}. θ_i is variable if joint i is revolute.

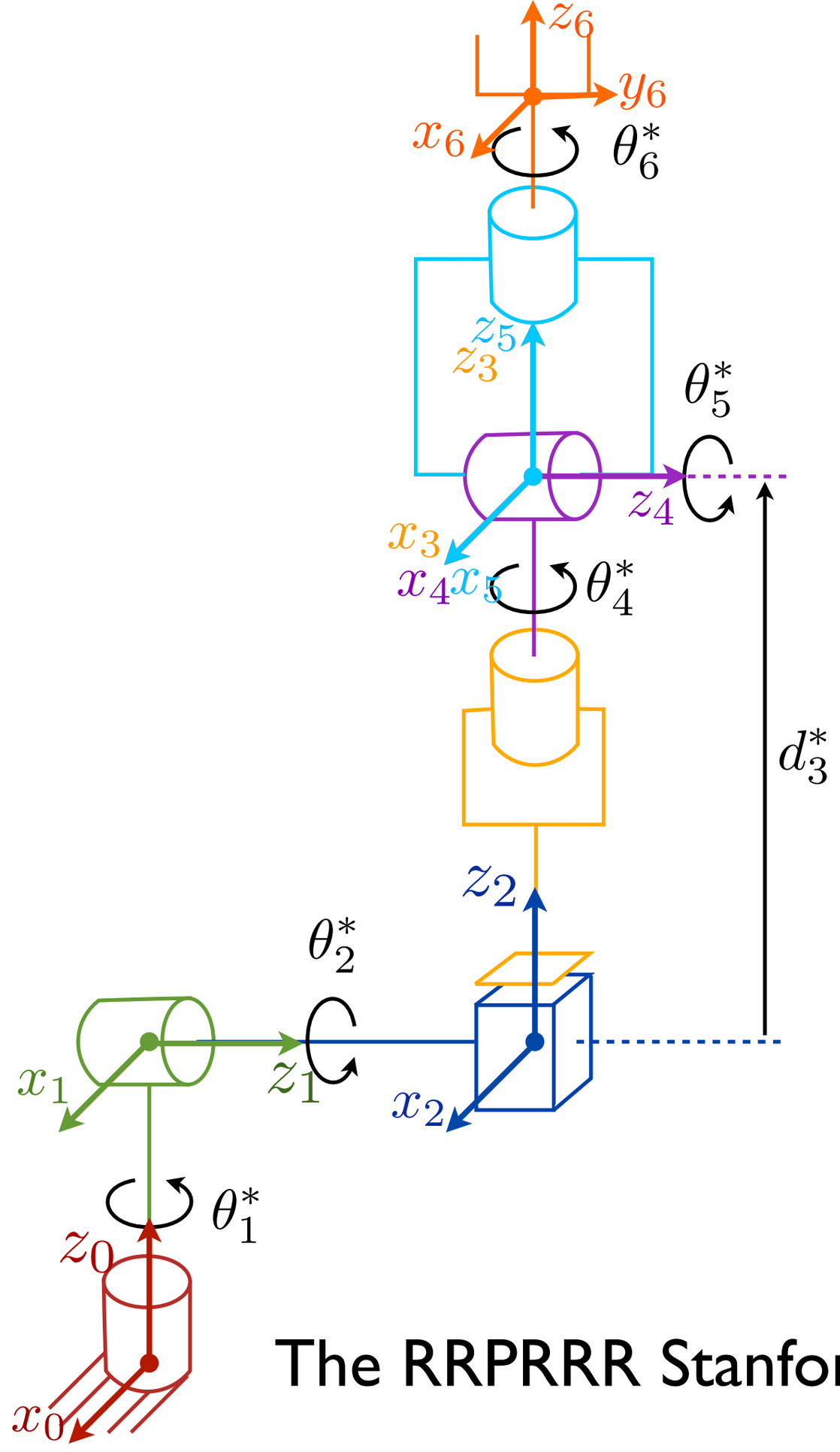
Step 8: Form the homogeneous transformation matrices A_i by substituting the above parameters into (3.10).

Step 9: Form $T_n^0 = A_1 \cdots A_n$. This then gives the position and orientation of the tool frame expressed in base coordinates.

The RPP Cylindrical Robot



$$\mathbf{T}_3^0 = \begin{bmatrix} c_1^* & 0 & -s_1^* & -d_3^* s_1^* \\ s_1^* & 0 & c_1^* & d_3^* c_1^* \\ 0 & -1 & 0 & d_1 + d_2^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



	x-step		z-step		
Link	a_i	α_i	d_i	θ_i	
1	0	$-90°$	d_1	θ_1^*	●→●
2	0	$+90°$	d_2	θ_2^*	●→●
3	0	$0°$	d_3^*	$0°$	●→●
4	0	$-90°$	0	θ_4^*	●→●
5	0	$+90°$	0	θ_5^*	●→●
6	0	$0°$	d_6	θ_6^*	●→●

The RRPRRR Stanford Manipulator with Spherical Wrist

Specifying Joint Values - Line

$\dot{q}$

$\left(\dot{q} = \frac{\mathrm{d}q}{\mathrm{d}t}\right)$

Discontinuities in velocity require infinite acceleration, which is not physically realizable.

t_0 t_f t

q

q_f

q_0

$$\frac{q_f - q_0}{t_f - t_0}$$

$$q(t) = q_0 + \frac{t - t_0}{t_f - t_0}(q_f - q_0)$$

t_0 t_f t

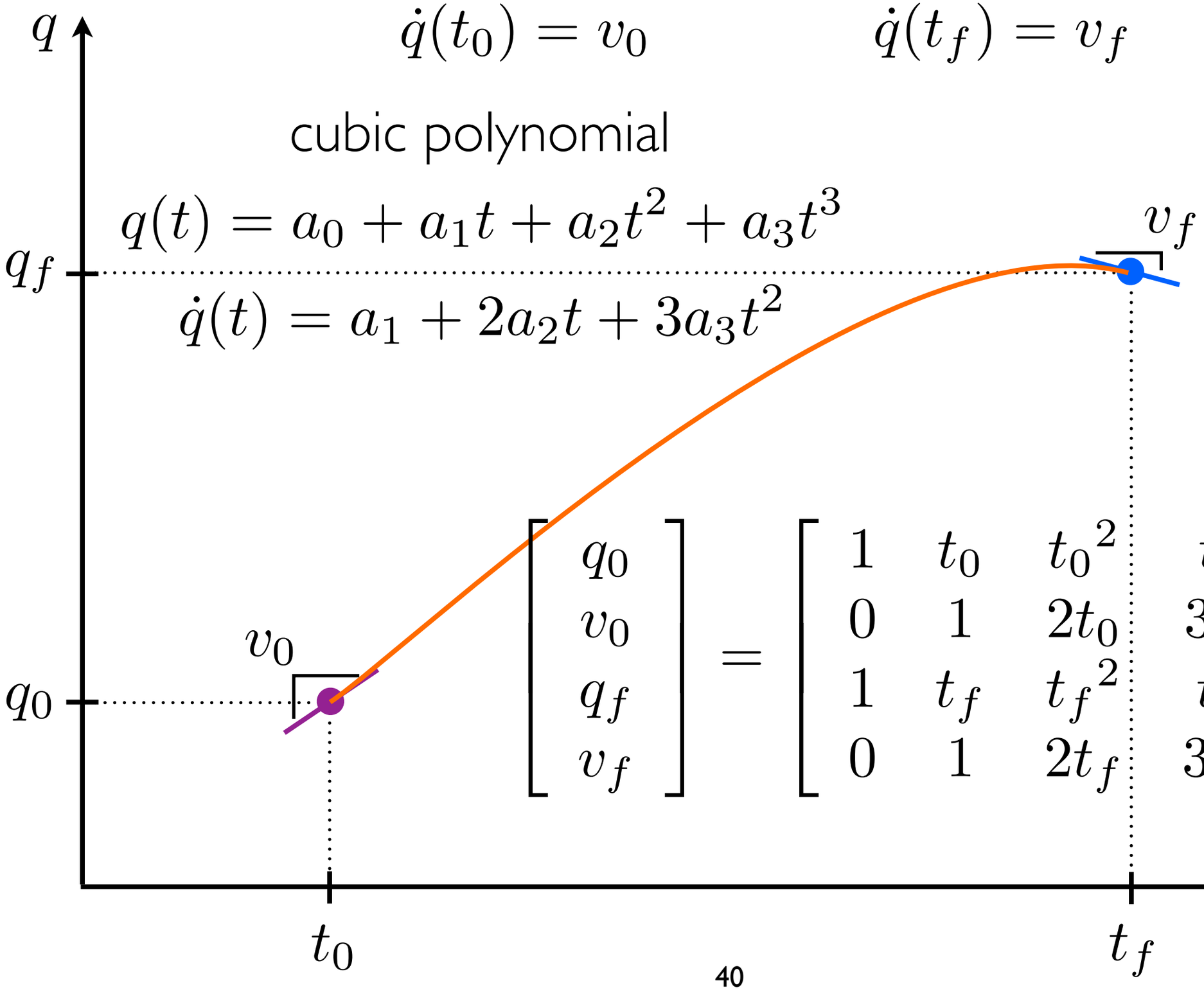
Specifying Joint Values and First Time Derivatives - Cubic

$$q(t_0) = q_0 \qquad q(t_f) = q_f$$
$$\dot{q}(t_0) = v_0 \qquad \dot{q}(t_f) = v_f$$

cubic polynomial

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

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



$$\begin{bmatrix} q_0 \\ v_0 \\ q_f \\ v_f \end{bmatrix} = \begin{bmatrix} 1 & t_0 & {t_0}^2 & {t_0}^3 \\ 0 & 1 & 2t_0 & 3{t_0}^2 \\ 1 & t_f & {t_f}^2 & {t_f}^3 \\ 0 & 1 & 2t_f & 3{t_f}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Specifying Joint Values Plus First and Second Time Derivatives

Quintic Polynomial Trajectories

start		end
$q(t_0) = q_0$	$\longrightarrow$	$q(t_f) = q_f$
$\dot{q}(t_0) = v_0$	$\longrightarrow$	$\dot{q}(t_f) = v_f$
$\ddot{q}(t_0) = \alpha_0$	$\longrightarrow$	$\ddot{q}(t_f) = \alpha_f$

quintic polynomial

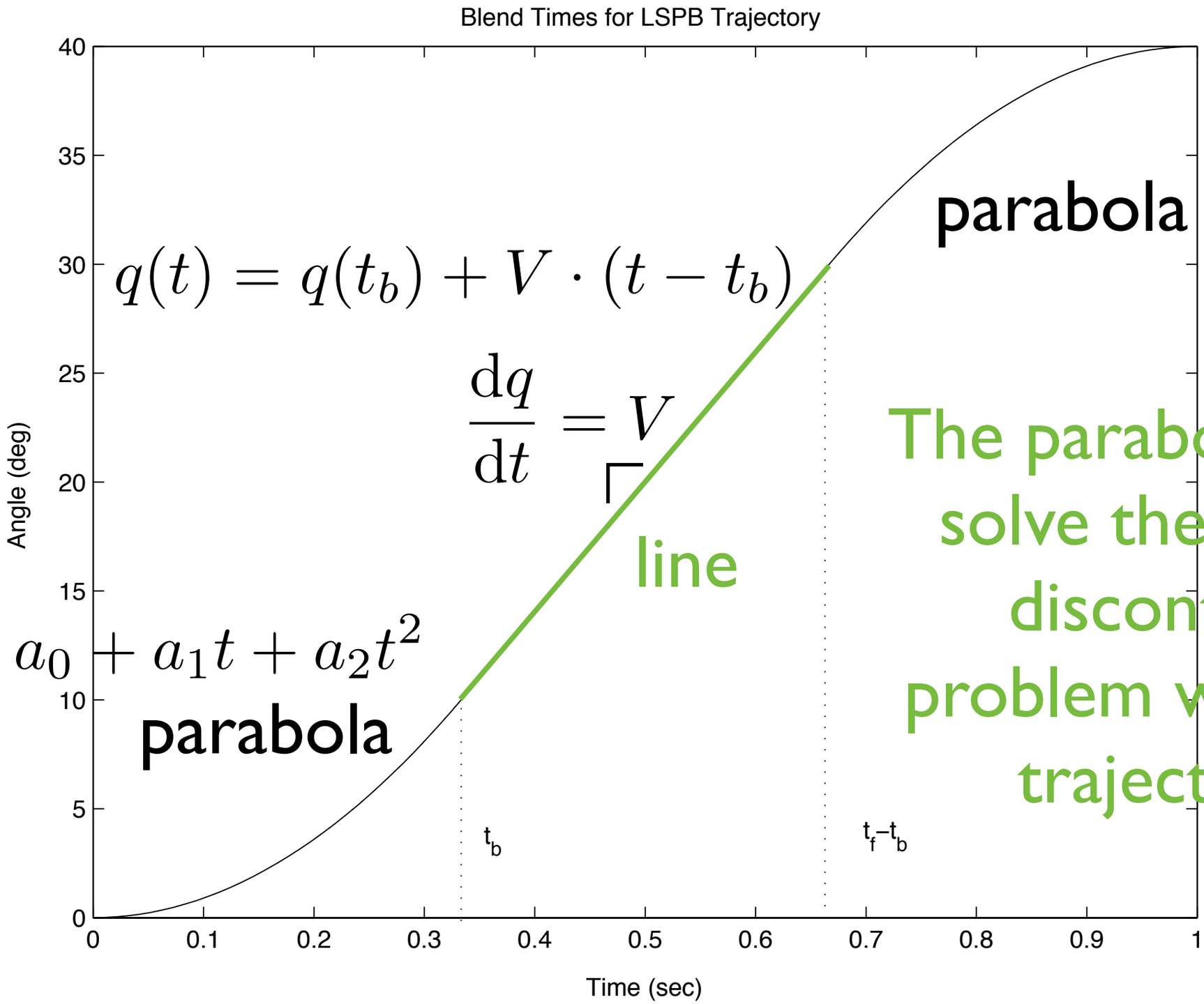
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

Specifying Constant Velocity for Central Portion

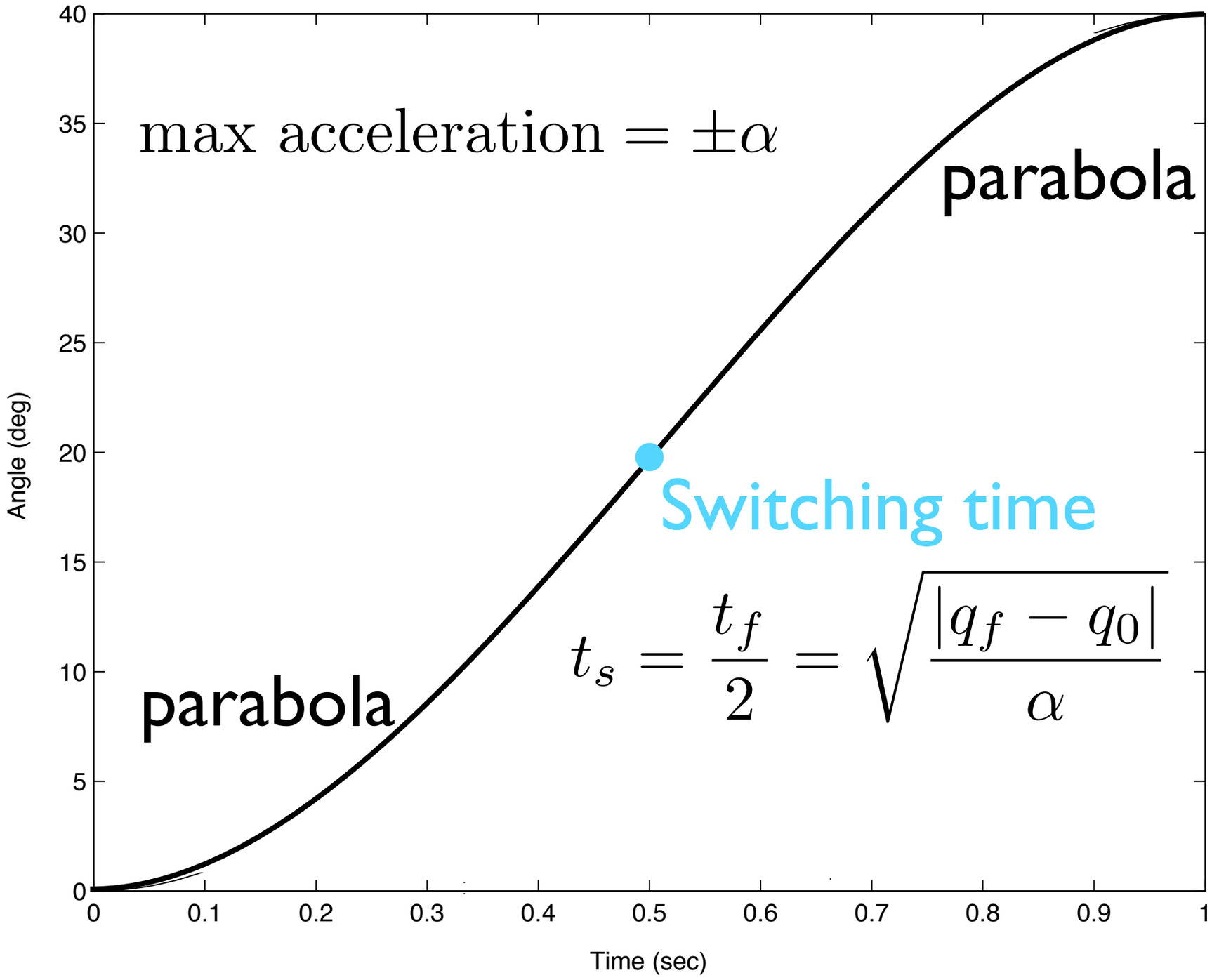
Linear Segments with Parabolic Blends (LSPB)



The parabolic blends solve the velocity discontinuity problem with linear trajectories.

Getting There As Fast As Possible

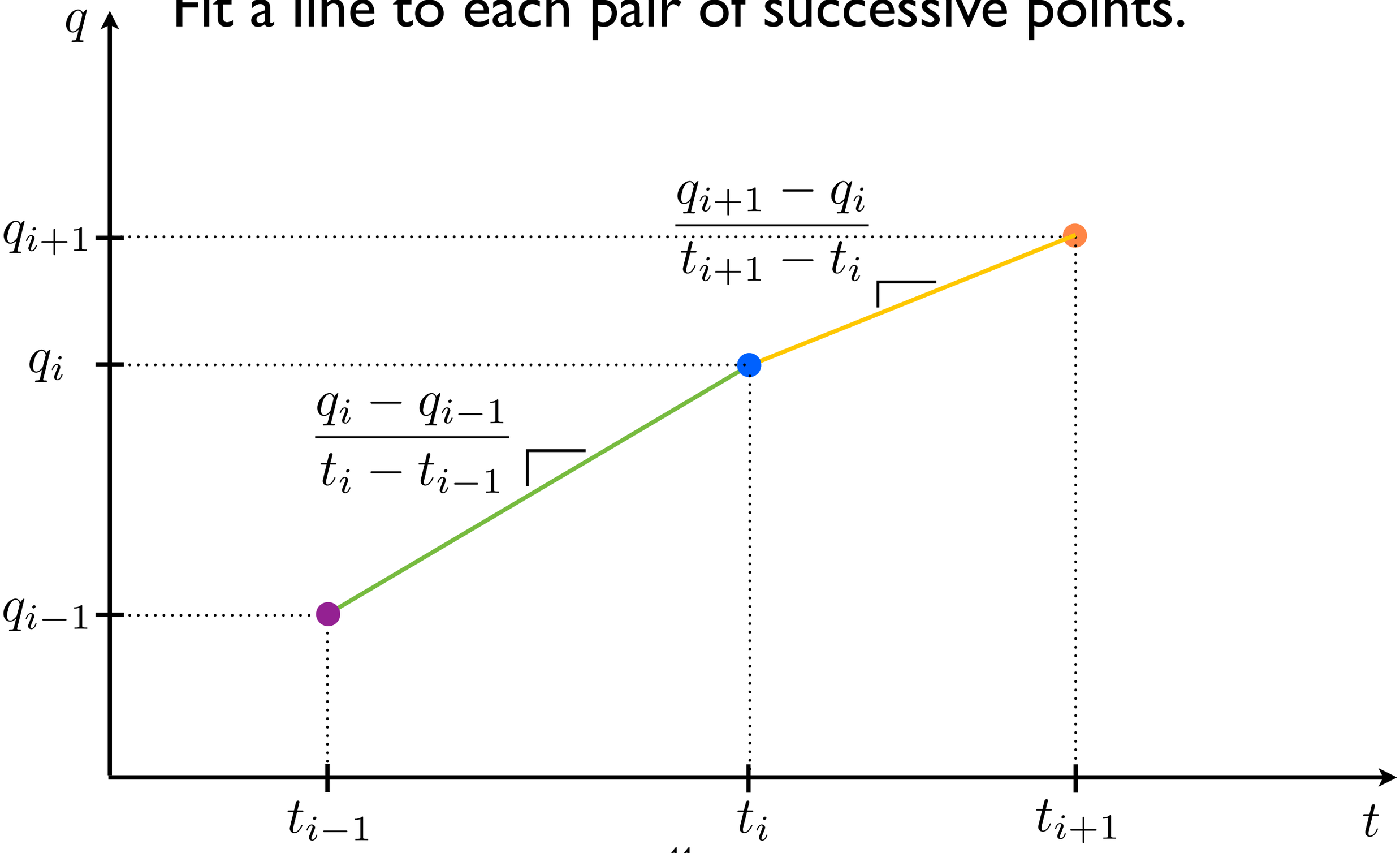
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories



You have a list of times and associated joint angles.

How to estimate joint velocity?

Fit a line to each pair of successive points.



Derivative of a Rotation Matrix

$$\dot{R} = \frac{dR}{dt} = \frac{dR}{d\theta}\frac{d\theta}{dt}$$

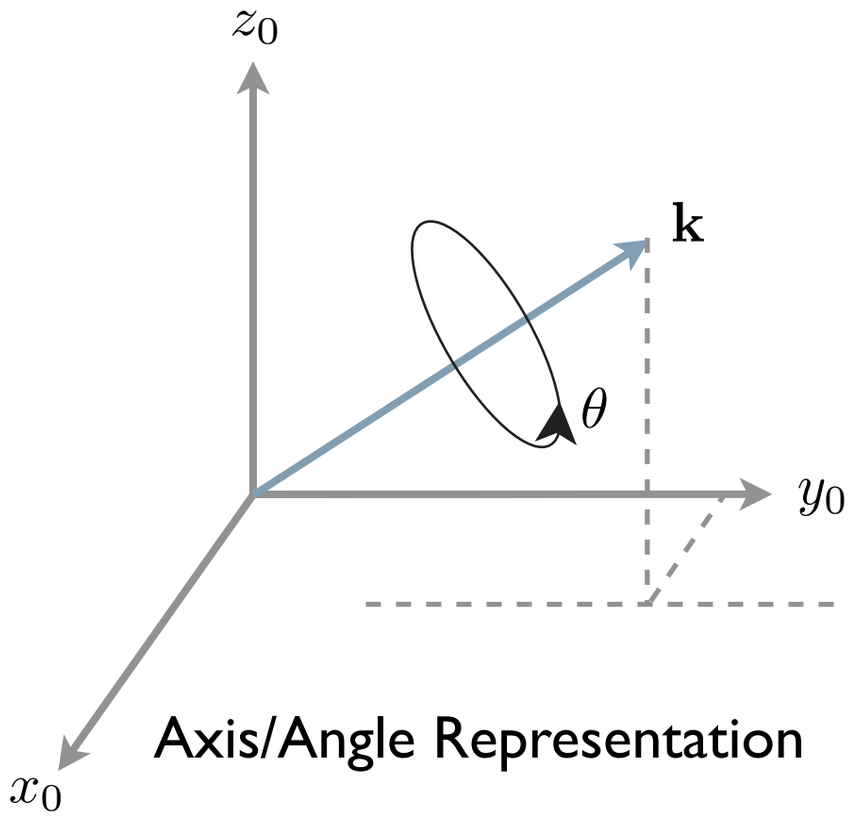
✓

?

$$S + S^T = 0$$

$$S(\vec{a}) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$S(\vec{a})\vec{p} = \vec{a} \times \vec{p}$$



Axis/Angle Representation

$$\frac{dR}{d\theta} = S(\hat{\omega})\,R$$

The skew-symmetric matrix S defines the axis about which rotation is occurring.

$$\frac{dR}{dt} = S(\vec{\omega})\,R$$

In general, you simply form S from the angular velocity vector and don't need to differentiate the matrix.

$$\vec{\omega}_2^0 = \vec{\omega}_{0,1}^0 + R_1^0\vec{\omega}_{1,2}^1$$

$$\dot{\vec{p}}^0 = S(\vec{\omega}^0)R_1^0\vec{p}^1 + \dot{\vec{o}}_1^0$$

Jacobians

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

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

$$\omega_n^0 = J_\omega \dot{q}$$

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

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

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

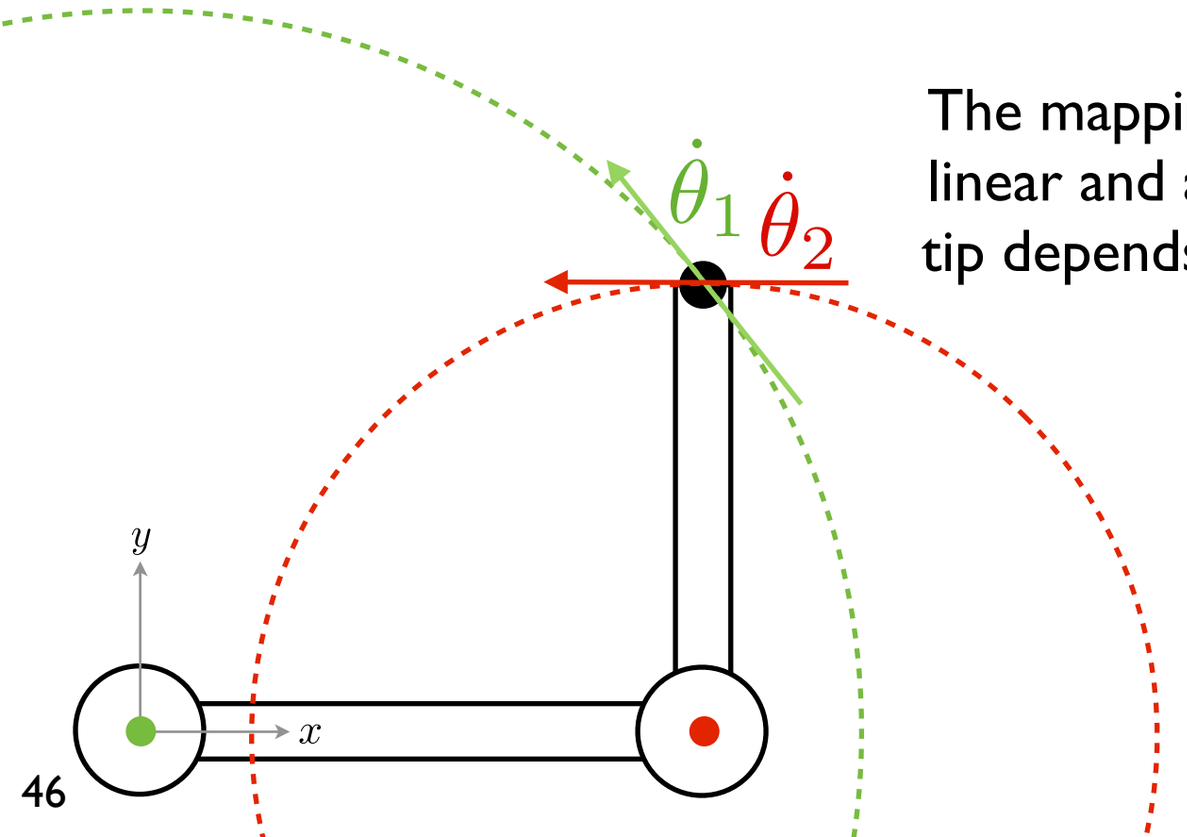
$$\rho_i = \begin{matrix} \text{0 for prismatic} \\ \text{1 for revolute} \end{matrix}$$

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

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

Prismatic $J_{\omega_i} = 0$

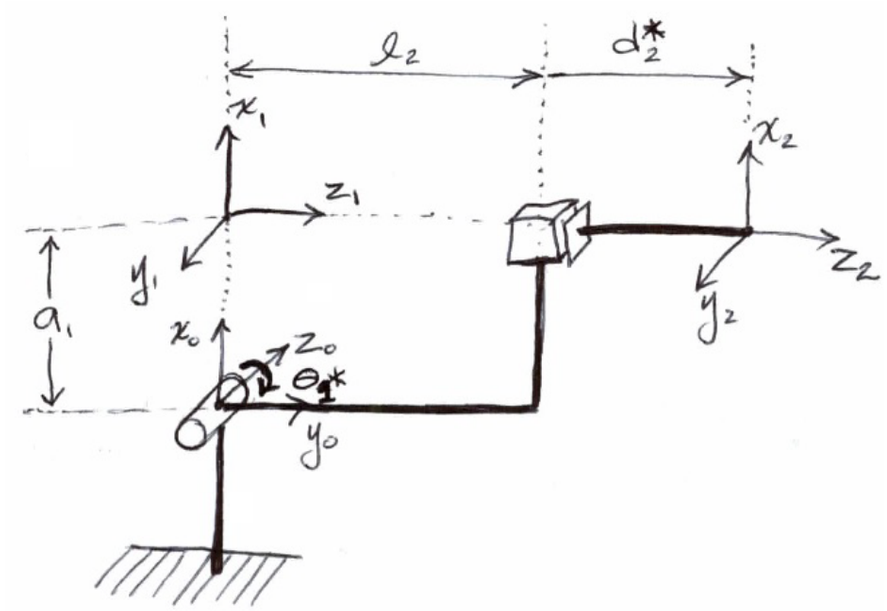
Revolute $J_{\omega_i} = z_{i-1}$



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

A robot loses the ability to move its end-effector in certain directions when $\det(J_v) = 0$

Singular configurations or singularities



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

(6 x 1) body velocity

but this is not the derivative of any vector

(6 x n) Jacobian

(n x 1) joint velocities

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

(6 x n) Analytical Jacobian

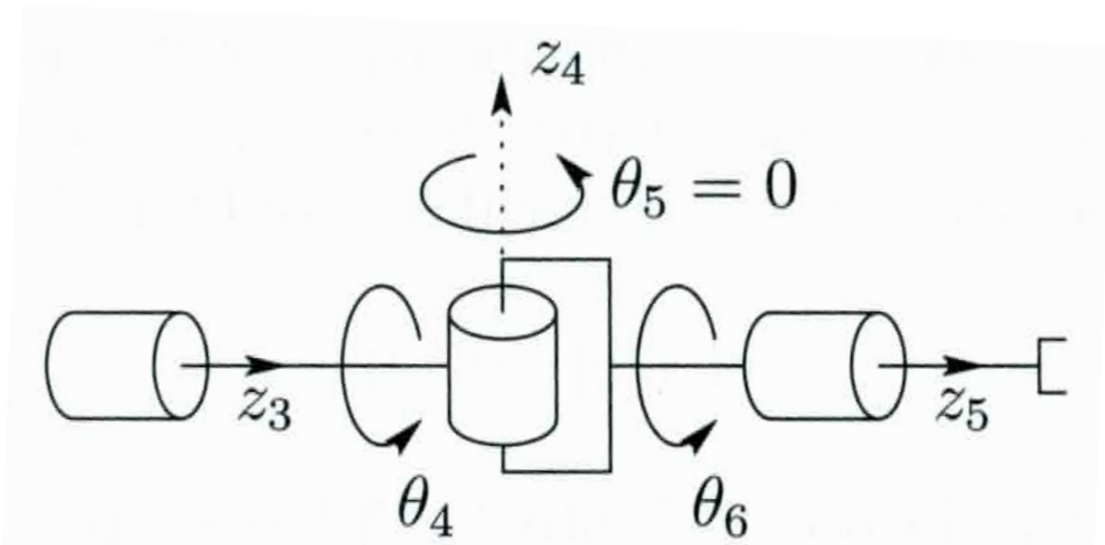
$$J = [J_{\text{arm}} \mid J_{\text{wrist}}] = \left[\frac{J_{11}}{J_{21}} \middle| \frac{J_{12}}{J_{22}}\right]$$

if we choose $o_4 = o_5 = o_6$

$$J = \left[\frac{J_{11}}{J_{21}} \middle| \frac{0}{J_{22}}\right]$$

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

arm wrist



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

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

$$\underset{(n \times 1)}{\vec{\tau}} = \underset{(n \times 6)}{J^T(\vec{q})} \underset{(6 \times 1)}{\vec{F}}$$

joint forces and torques

Jacobian matrix transpose

endpoint forces and torques

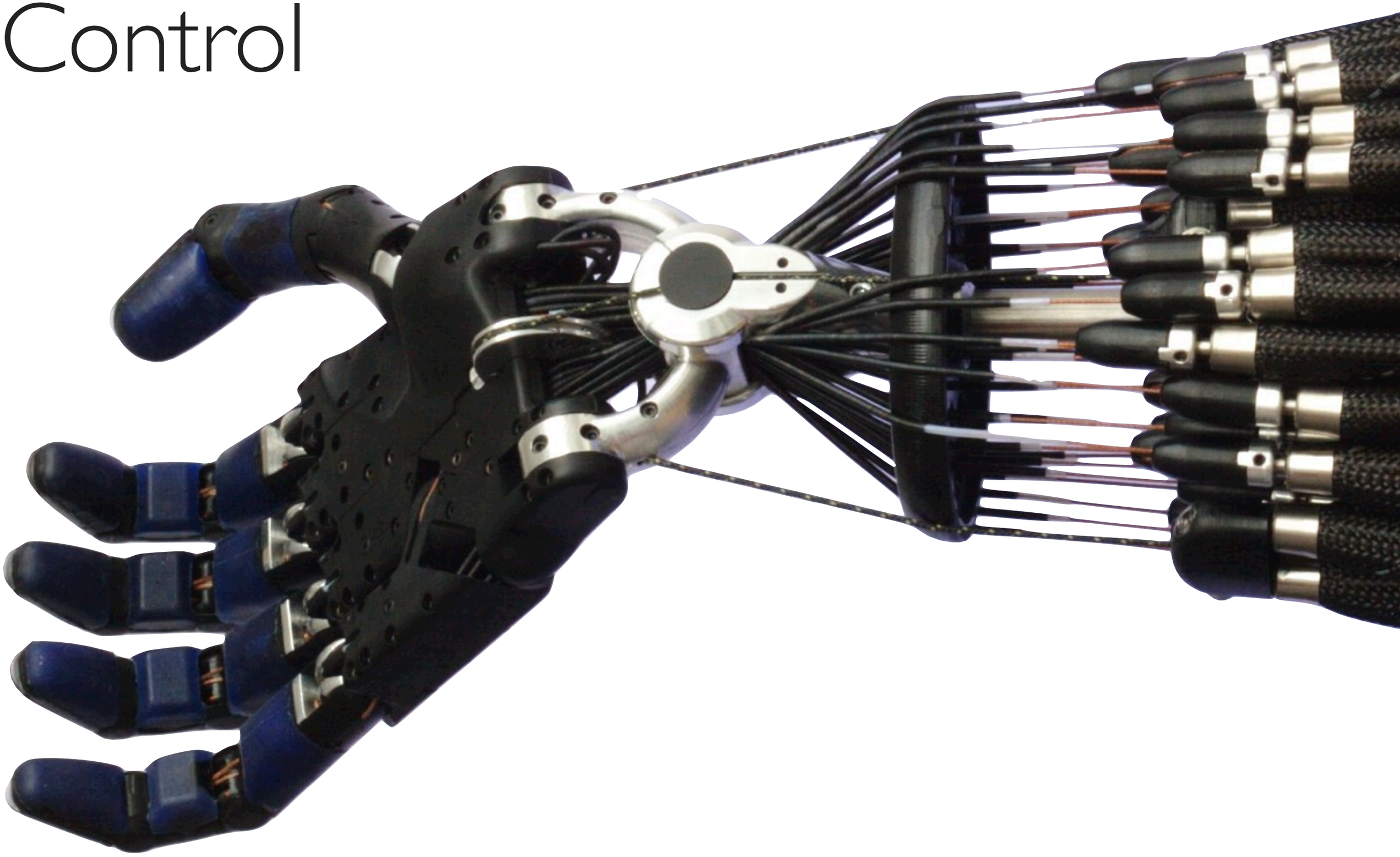
For a specific configuration, the Jacobian scales the input (joint velocities) to the output (body velocity)

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

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.

Manipulator Hardware and Control



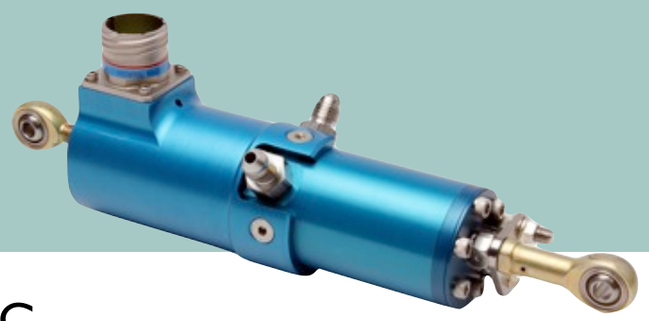
A Biological Inspiration

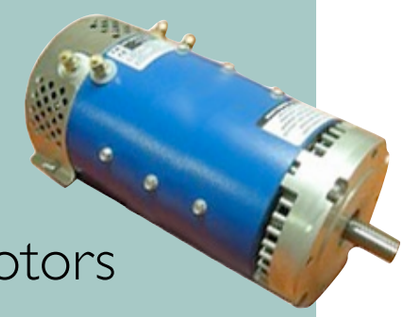
Mechanical Structure

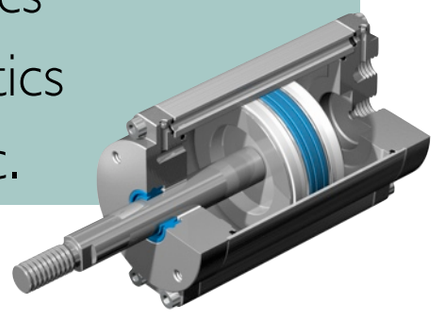
Bones	Frame / Links
Joints	Joints

Actuators

Muscles	Electric Motors
	Hydraulics
	Pneumatics
	SMA, etc.





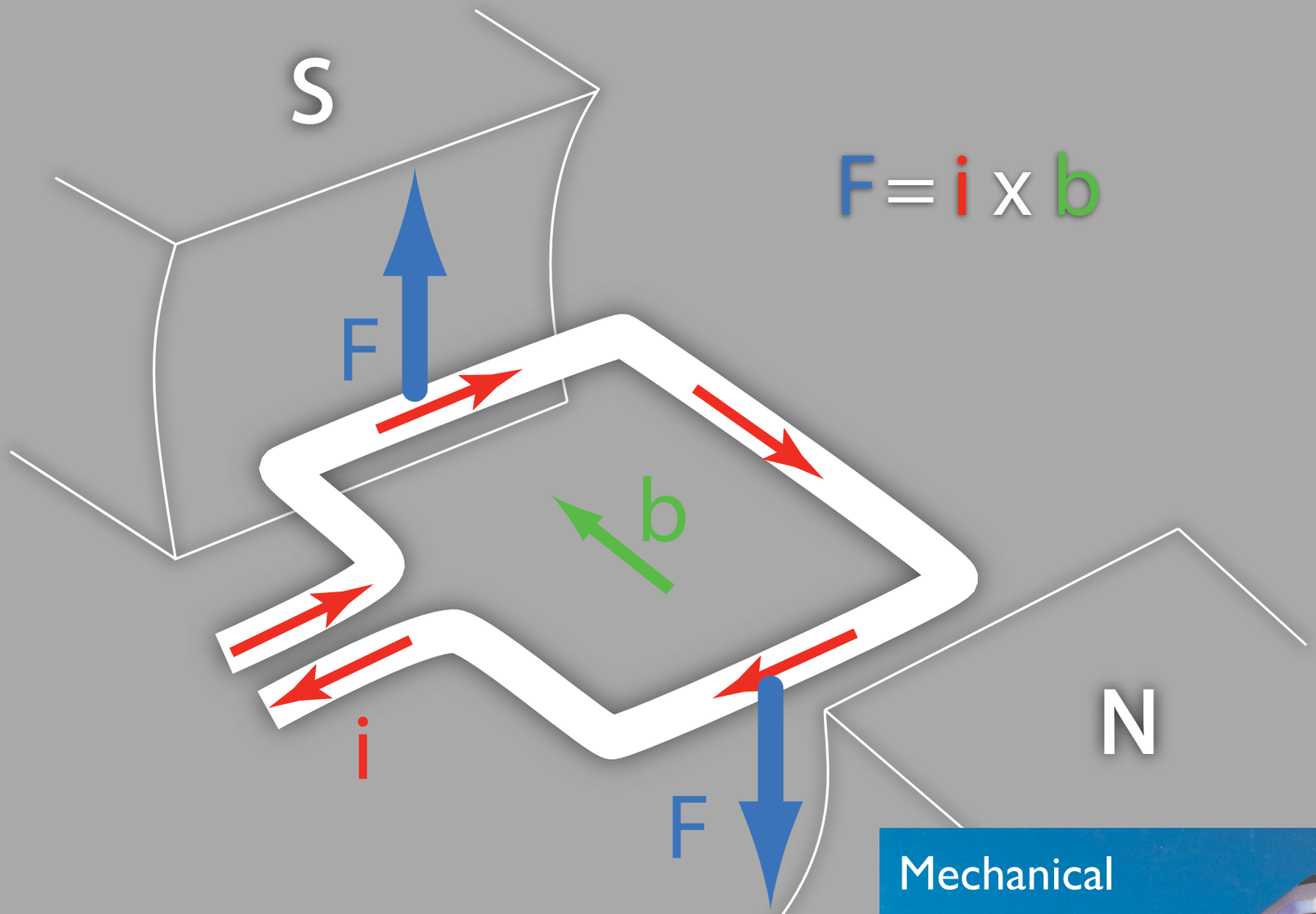


Sensors

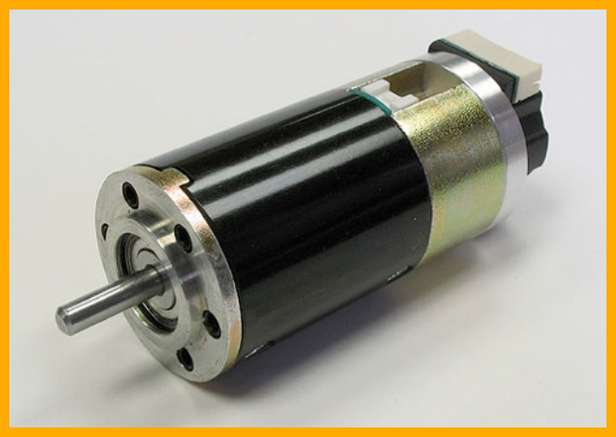
Kinesthetic	Encoders
Tactile	Load Cells
Vision	Vision
Vestibular	Accelerometers

Controller

Brain	Computer
Spinal Cord Reflex	Local feedback



$F = i \times b$



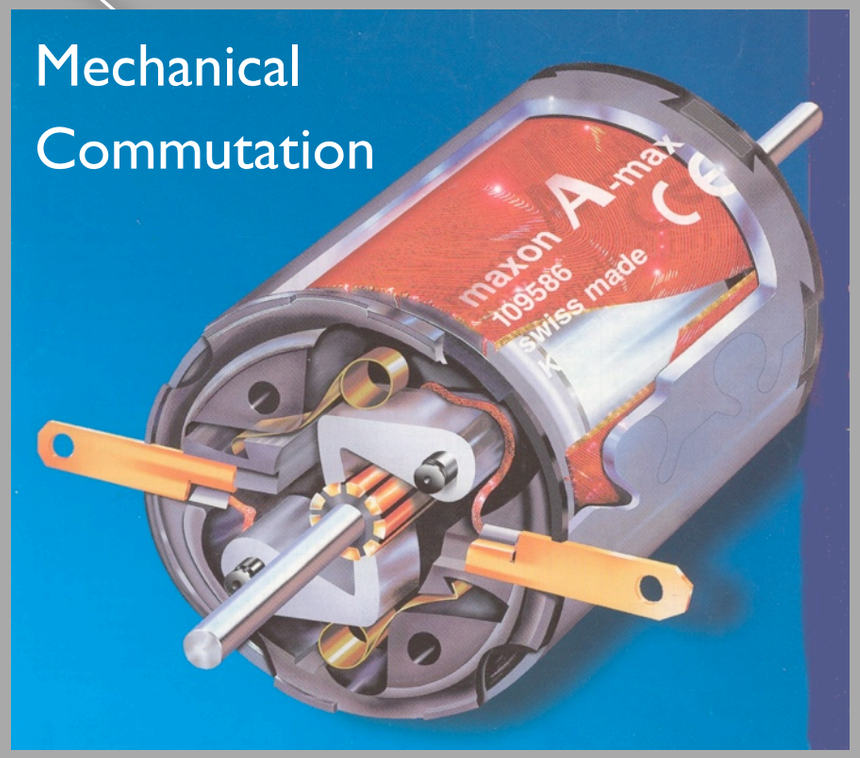
DC Brushed

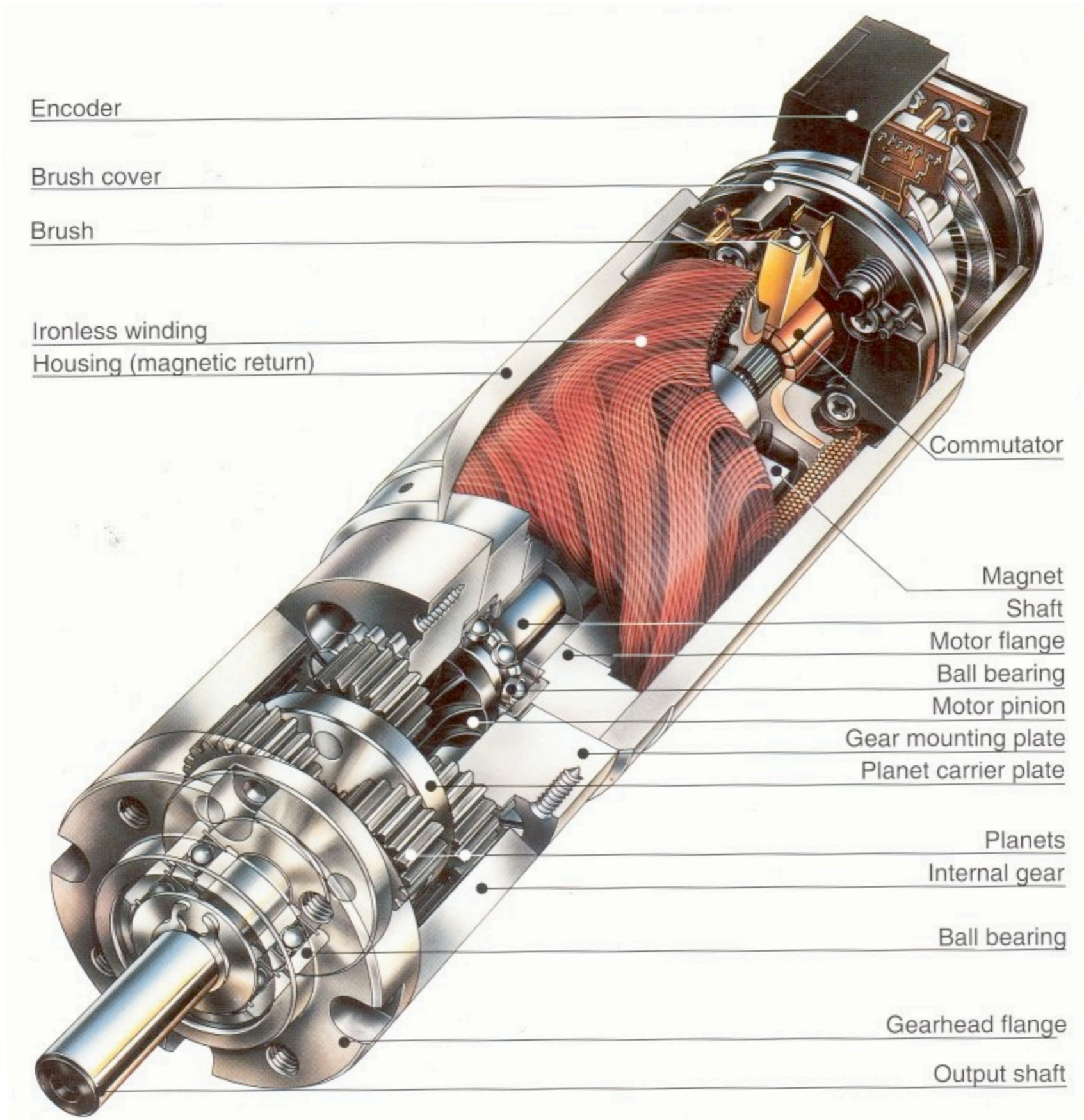
Most common!

Coil Rotor

Magnetic Stator

Brushes carry current to the rotor





torque constant (N•m/A)

$$\tau_m = k_t \, i_a$$

generated torque (N•m)

armature current (A)

back emf (V)

back-emf constant (V•s)

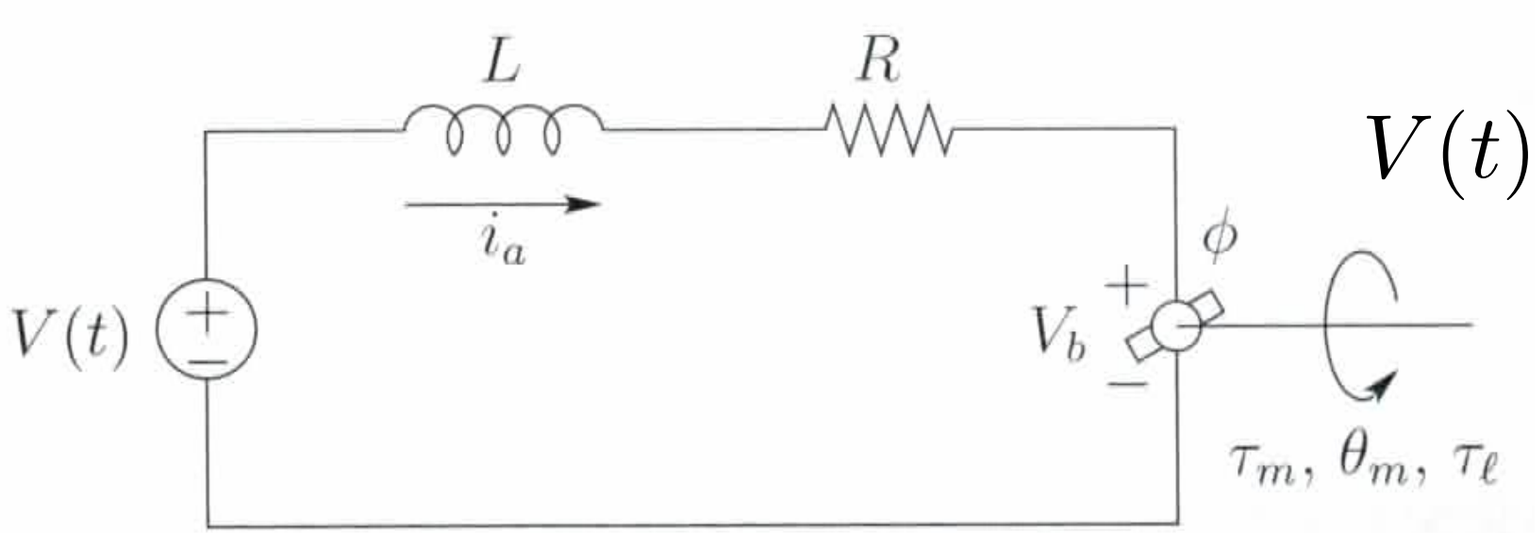
$$V_b = k_v \, \omega_m$$

motor velocity (rad/s)

$$k_t = k_v$$

if using meters, kilograms and seconds

Electrical Circuit Diagram of a DC Brushed Motor



Use Kirchoff's Voltage Law:

$$V(t) = L\frac{di_a}{d_t} + R\,i_a + \overbrace{k_v\,\omega_m}^{V_b}$$

If I hold the motor's shaft stationary so it cannot rotate,
the motor is merely an inductor in series with a resistor.

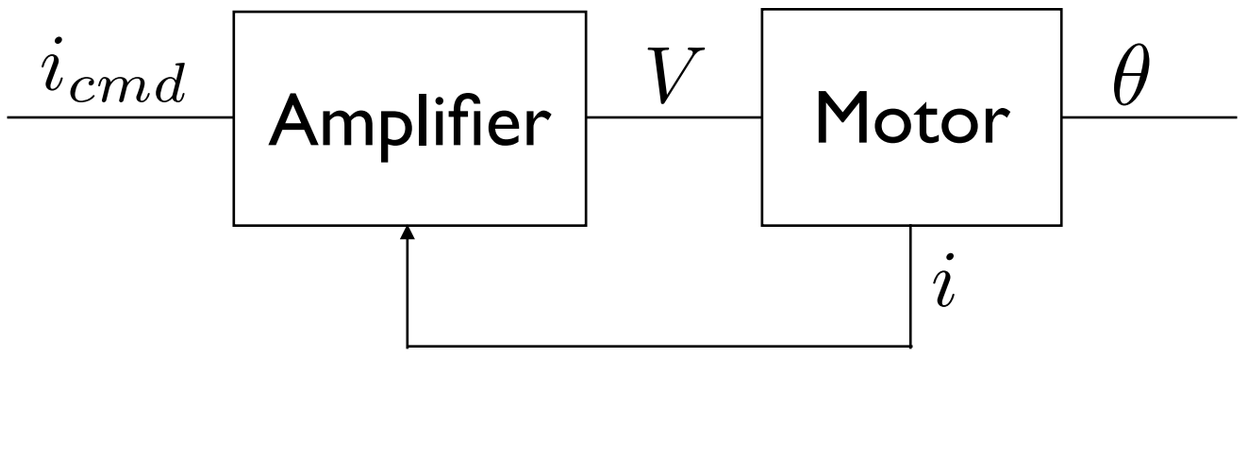
Usually we want to allow motors to rotate!

Want to control torque output.... $\tau_m = k_t\,i_a$

I could just ignore the inductance and back-emf, assuming $i_a \approx \frac{V(t)}{R}$

Not a terrible solution, but inductance is often non-negligible,
back-emf is large when speeds are high, and R changes with temperature.

These electrical dynamics can affect performance, so engineers sometimes try to overcome them
by using a current controlled circuit (current drive) instead of voltage drive.


i_{cmd}
Amplifier
V
Motor
θ
i