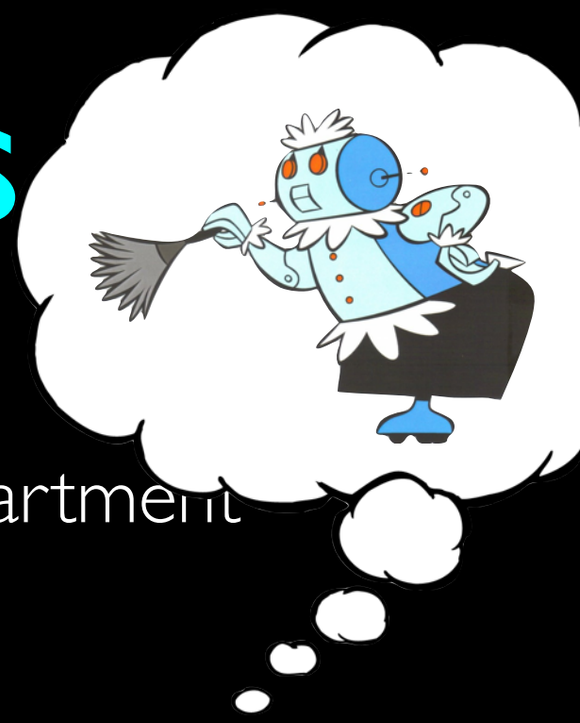
MEAM 520
Animating Robots

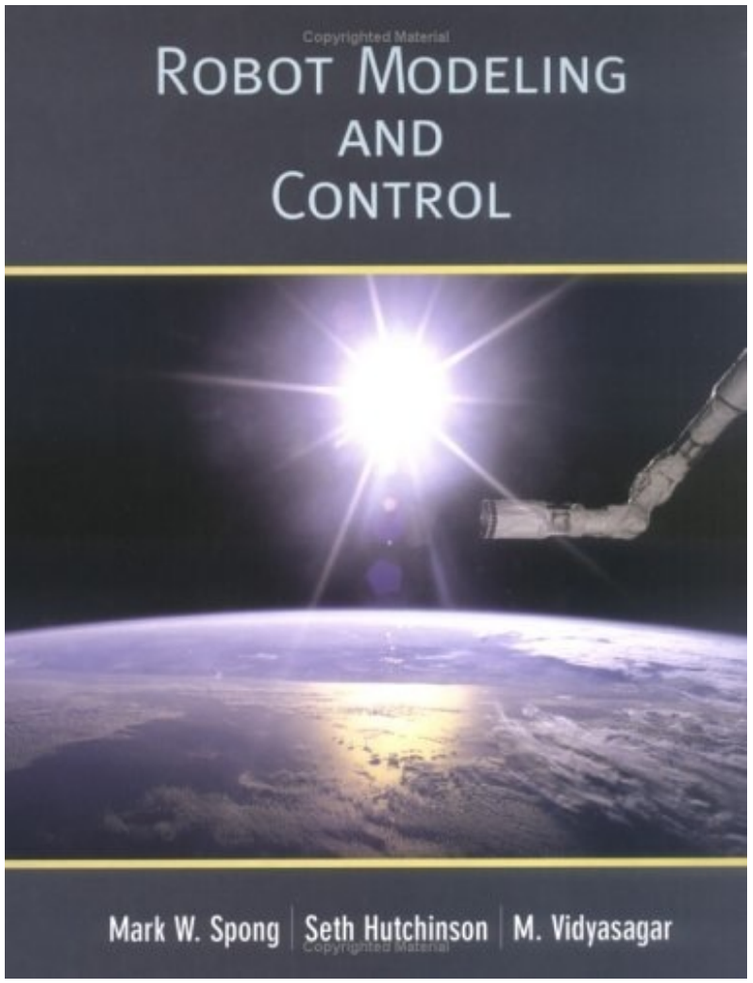


Katherine J. Kuchenbecker, Ph.D.
Mechanical Engineering and Applied Mechanics Department
SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 8: September 23, 2014

Reading



This week, read Section 5.5 in SHV, which covers Trajectory Planning.

Homework

Homework 4: Forward Kinematics
and the Denavit-Hartenberg Convention

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

September 18, 2014

This written assignment is due on **Thursday, September 25, by midnight (11:59:59 p.m.)** You should aim to turn it in during class that day. If you don't finish until later in the day, you can turn it in to Professor Kuchenbecker's office, Towne 224, in the assignment submission box or under the door. Late submissions will be accepted until Sunday, September 28, 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; post a private message on Piazza to request an extension if you need one due to a special situation such as illness.

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!

These problems are a mix of custom problems and problems adapted from the textbook, *Robot Modeling and Control* by Spong, Hutchinson, and Vidyasagar (SHV). All instructions are provided in this document, so you do not need to refer to the textbook for the questions. Write in pencil, show your work clearly, and staple together all pages of your assignment. This assignment is worth a total of 25 points.

1. **DH Convention** *(2 points)*
   Describing a rigid-body transformation in three dimensions generally requires six numbers. Why then are only four DH parameters (a, α, d, θ) needed to describe link i's pose relative to link $i-1$ in a serial manipulator? Be precise.

2. **Checking the 4-DOF SCARA's Transformation Matrix** *(4 points)*
   A SCARA robot is an RRP manipulator where all the joint axes are aligned. As in the bread-handling video shown in class, one more revolute joint is often added to third link of such robots, so that the end-effector can reorient in the horizontal plane. The resulting robot is an RRPR with all axes aligned, as shown in Figure 3.11 of SHV; this robot's joint variables are θ_1^*, θ_2^*, d_3^*, and θ_4^*. Note that Figure 3.11 shows frame $o_0x_0y_0z_0$ in the wrong location; it should be translated up along the z_0 axis until x_0 lies along the horizontal line that goes toward joint 1.
   Equation (3.24) on page 93 of SHV gives the 4-DOF SCARA manipulator's T_4^0 transformation matrix as the following, where s_i means $\sin\theta_i$, c_i means $\cos\theta_i$, and s_{ij} and c_{ij} mean $\sin(\theta_i+\theta_j)$ and $\cos(\theta_i+\theta_j)$:

$$T_4^0 = \begin{bmatrix} c_{12}^* c_4^* + s_{12}^* s_4^* & -c_{12}^* s_4^* + s_{12}^* c_4^* & 0 & a_1 c_1^* + a_2 c_{12}^* \\ s_{12}^* c_4^* - c_{12}^* s_4^* & -s_{12}^* s_4^* - c_{12}^* c_4^* & 0 & a_1 s_1^* + a_2 s_{12}^* \\ 0 & 0 & -1 & -d_3^* - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   Check every element of this matrix to determine whether it is correct. Write an explanation for the correctness or incorrectness of each element, r_{ij}, drawing on geometry and other practical knowledge about homogeneous transformations, the SCARA robot, and how the frames are defined.

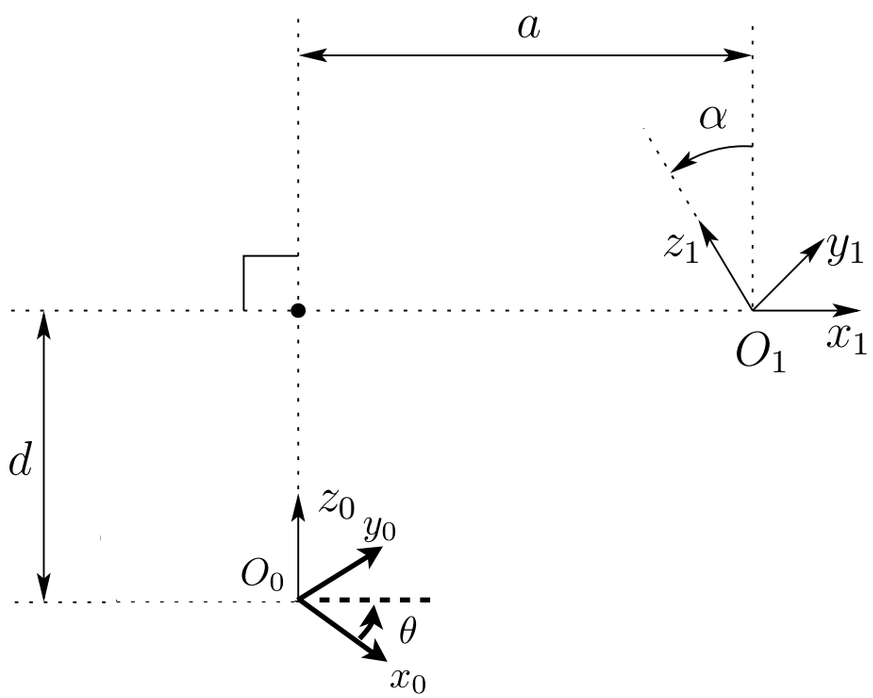
1

Homework 4 is due by 11:59 p.m. Thursday.

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?

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



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

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

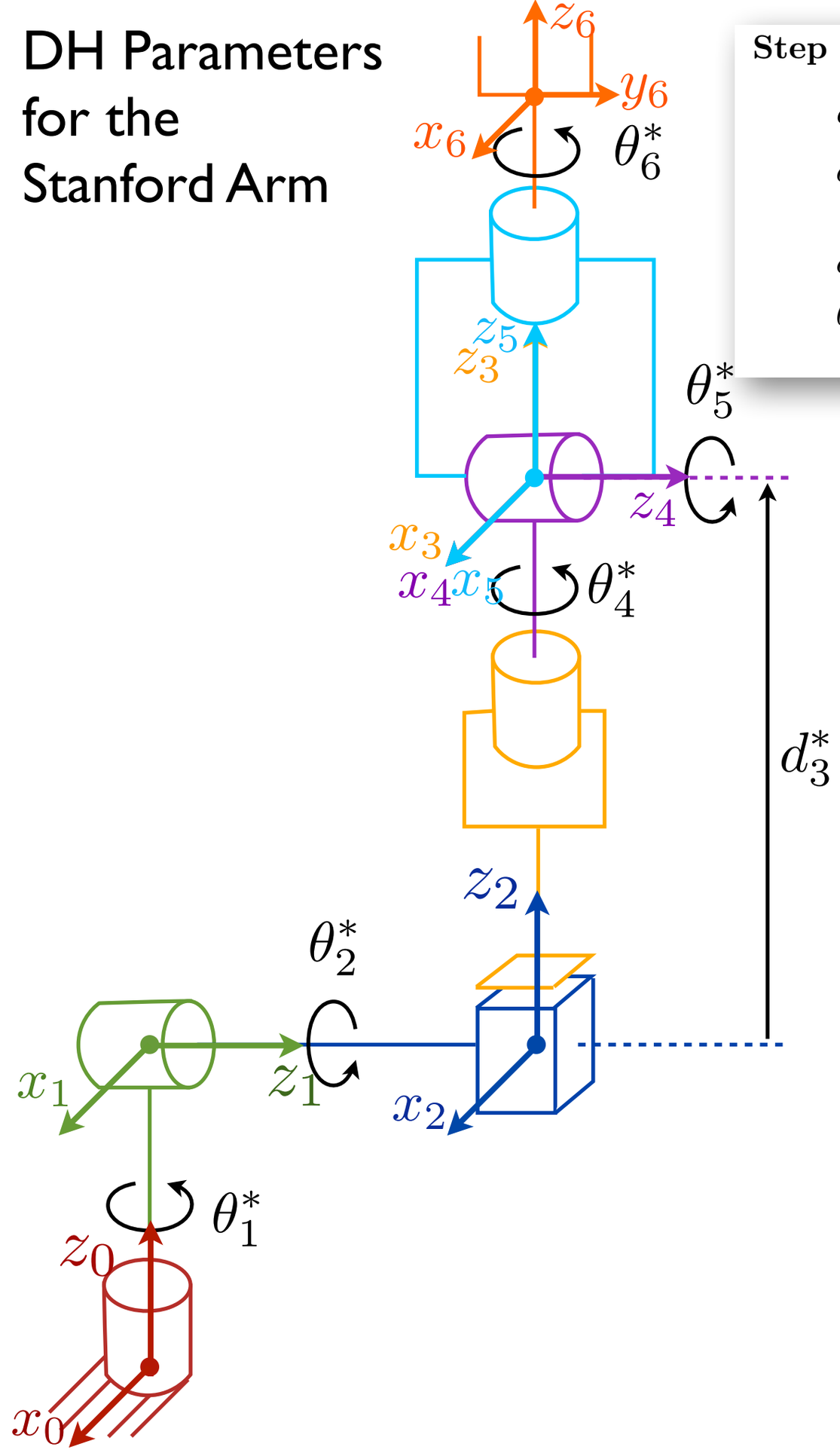
Link	a_i	α_i	d_i	θ_i

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.

DH Parameters for the Stanford Arm



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

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

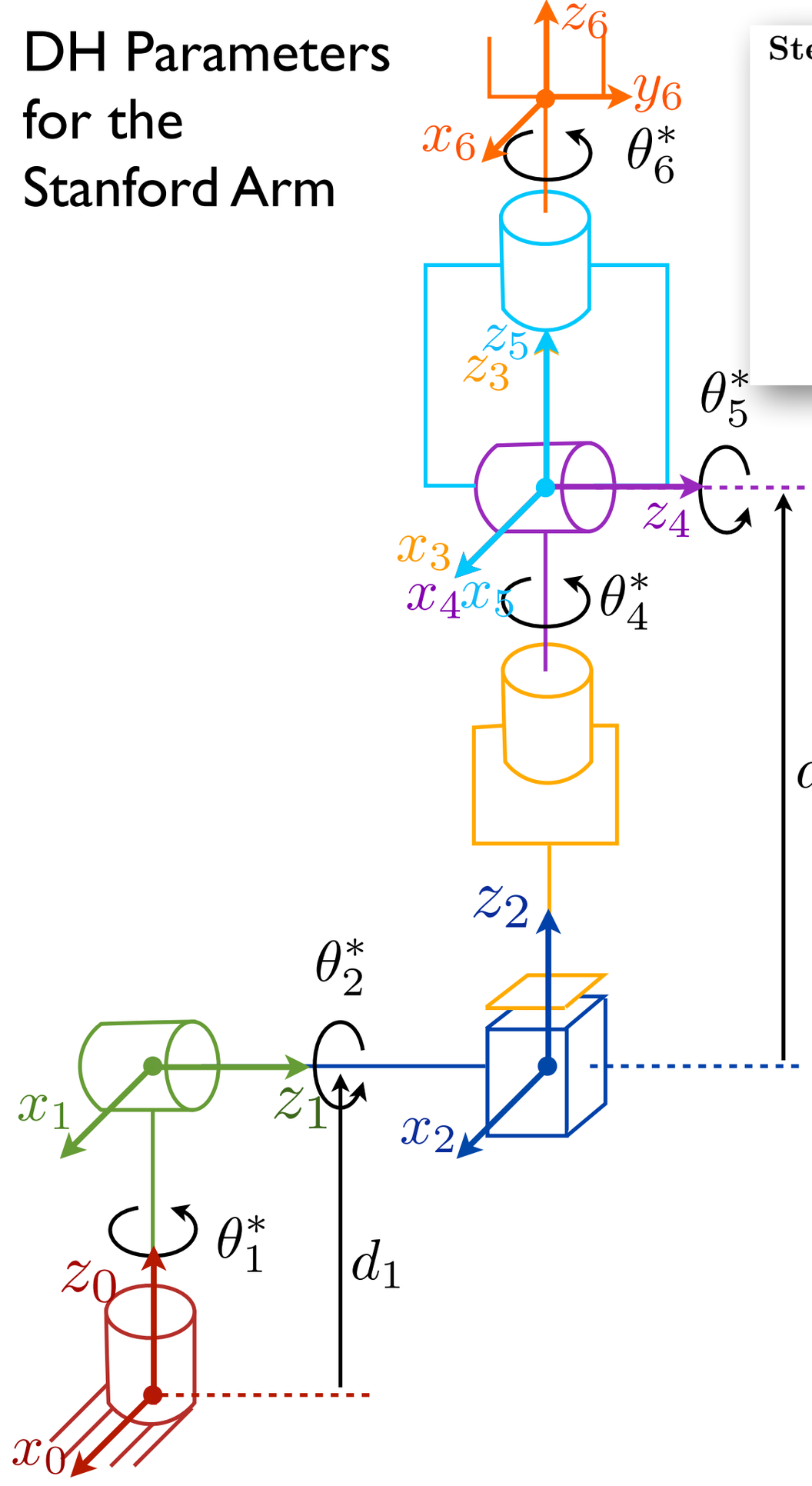
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	x-step		z-step	
Link	a_i	α_i	d_i	θ_i

DH Parameters for the Stanford Arm



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

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

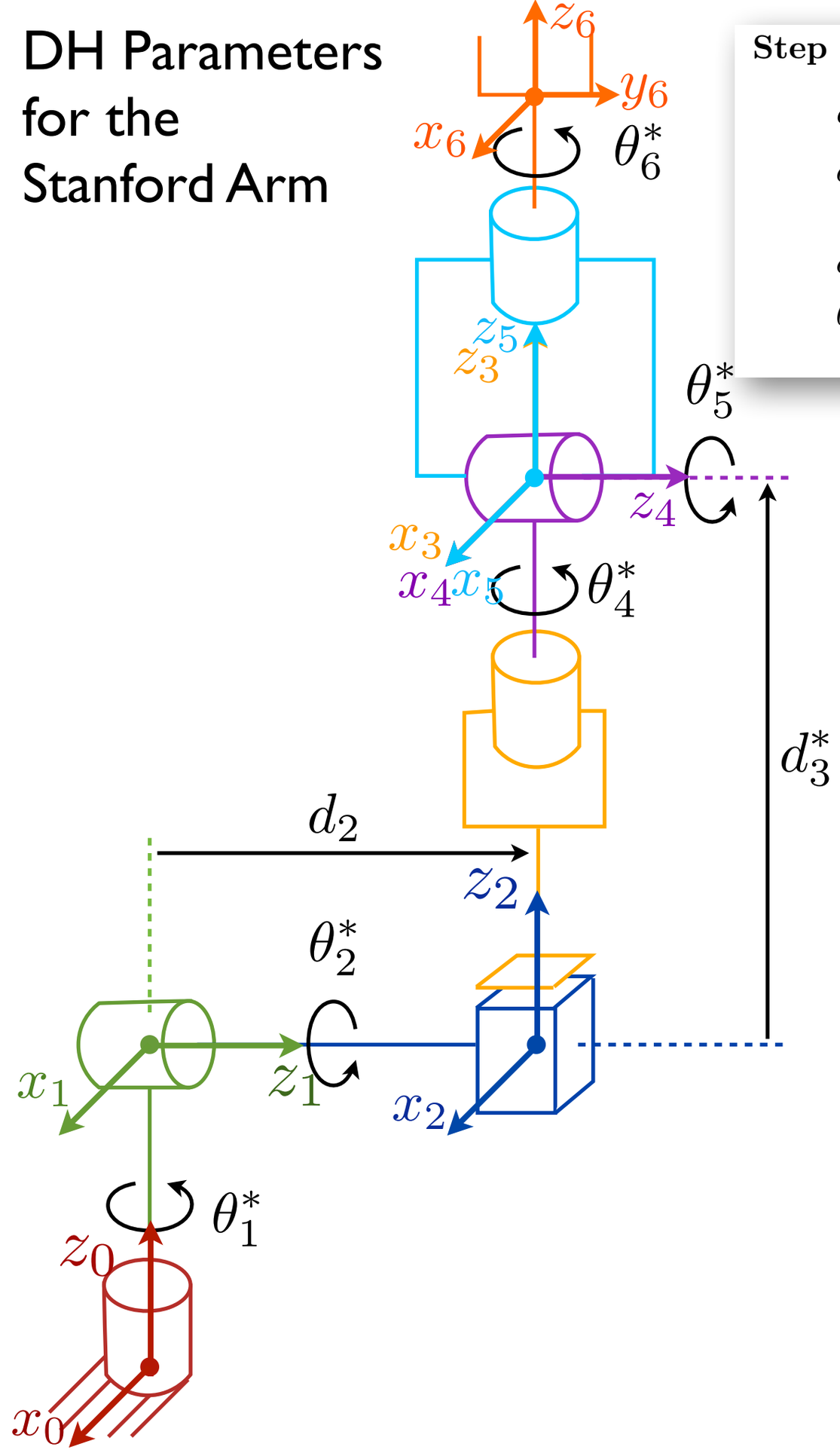
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	x-step		z-step	
Link	a_i	α_i	d_i	θ_i
1	0	$-90°$	d_1	θ_1^*

DH Parameters for the Stanford Arm



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

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

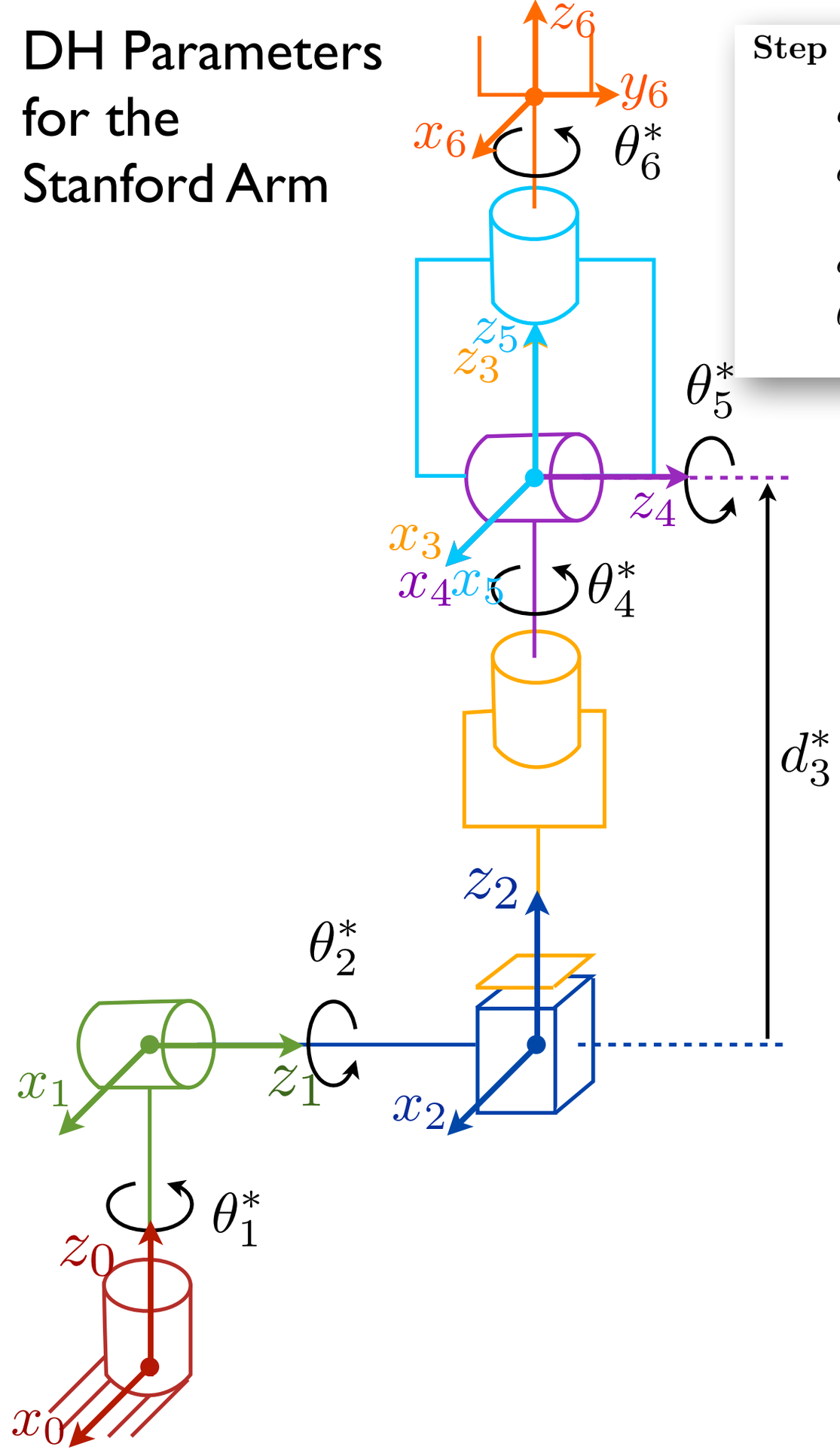
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	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^*

DH Parameters for the Stanford Arm



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

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

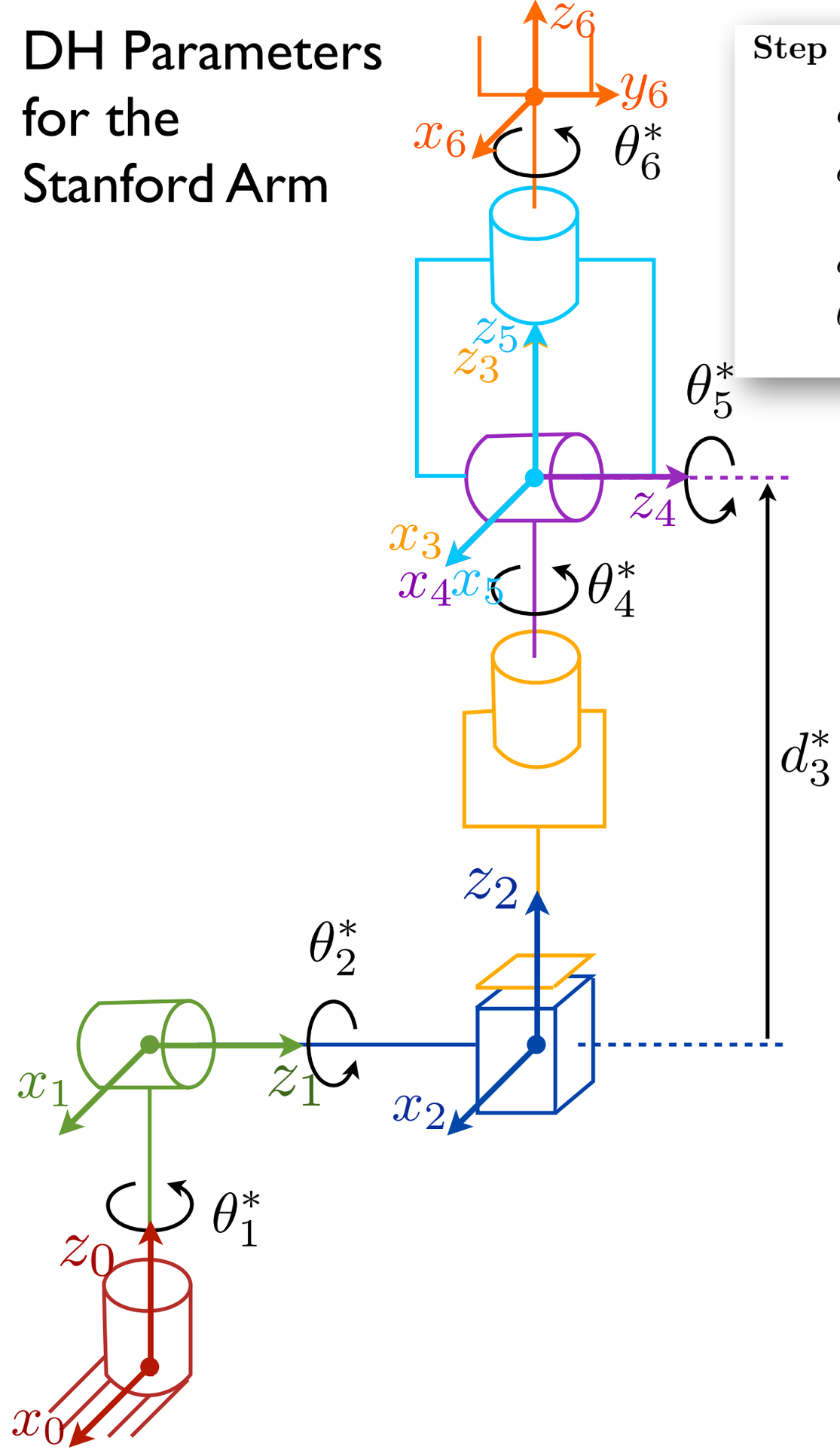
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	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^*	$0°$	●→●

DH Parameters for the Stanford Arm



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

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

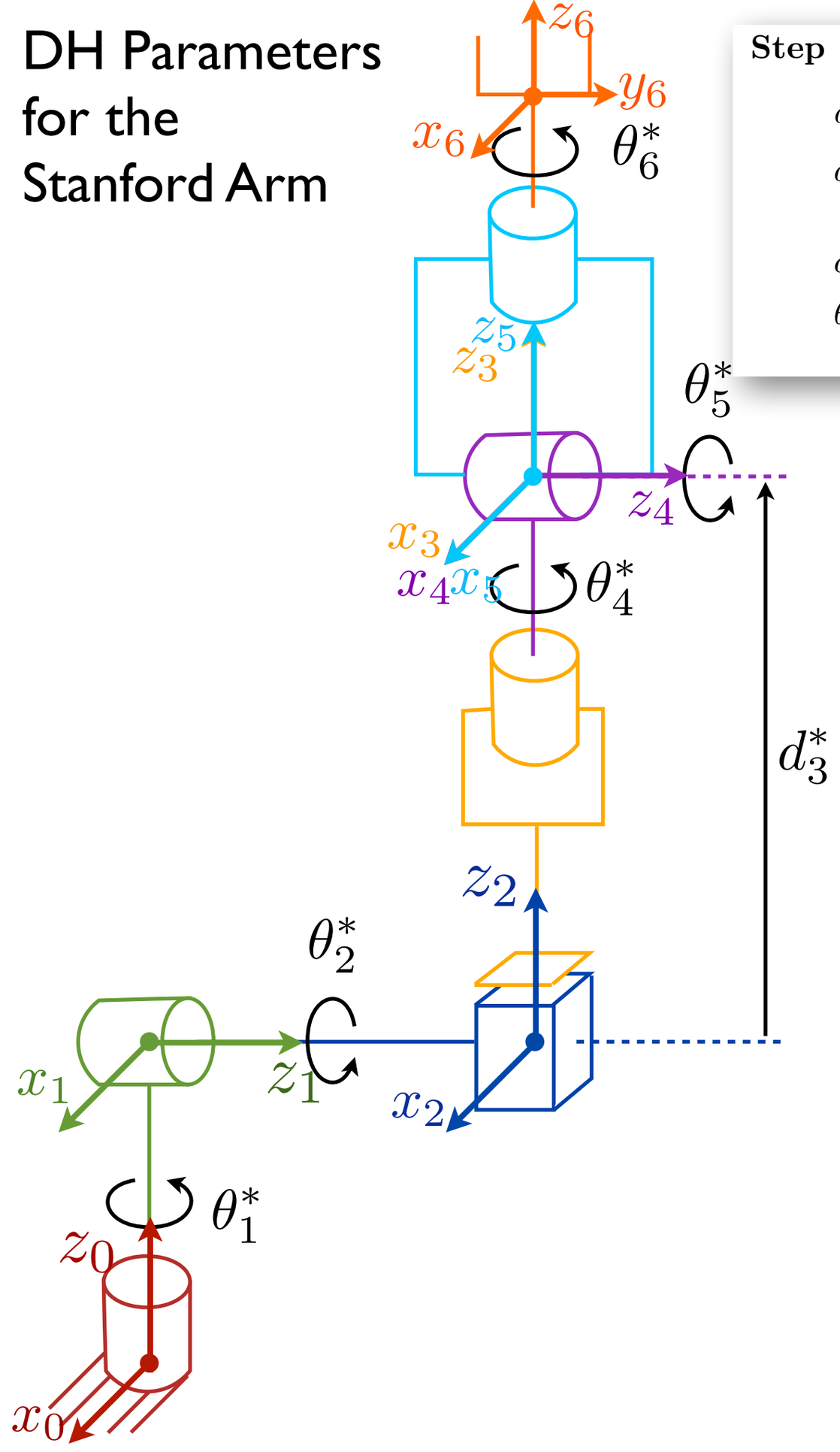
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	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^*	●→●

DH Parameters for the Stanford Arm



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

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

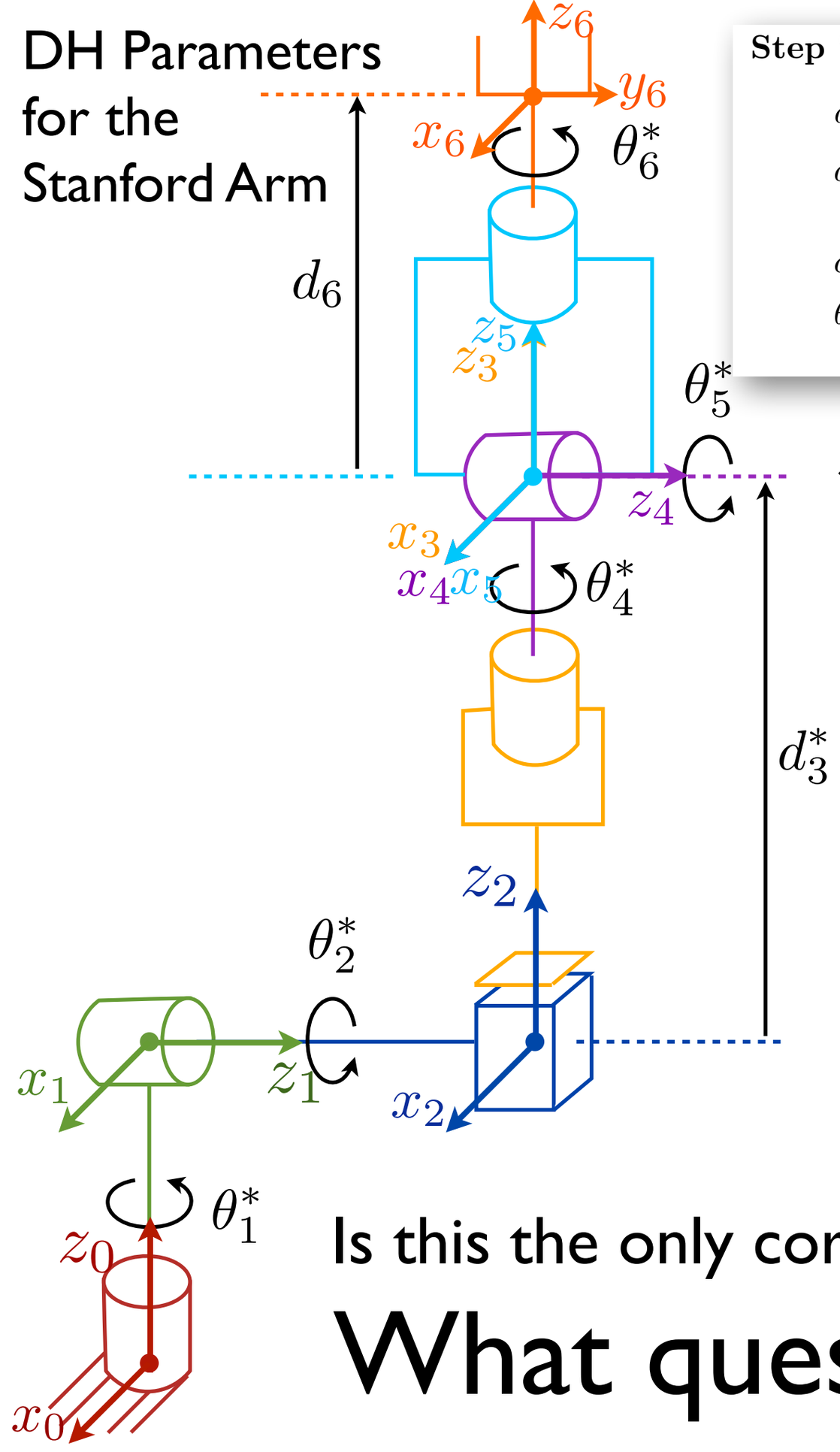
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint i is prismatic.

$\alpha_i =$ 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.

	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^*	●→●

DH Parameters for the Stanford Arm



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

$a_i =$ 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.

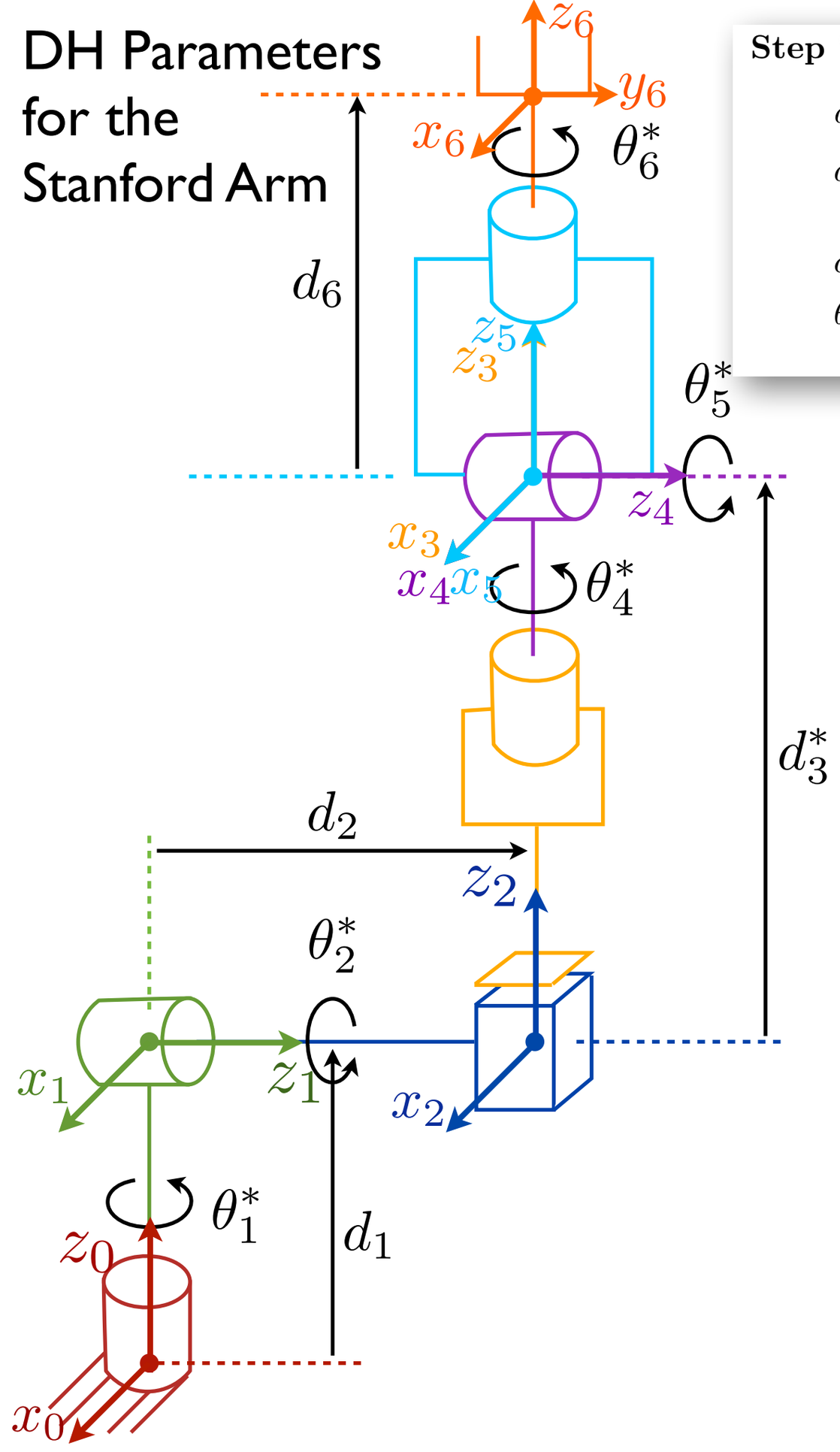
← Origins 3, 4, and 5 coincide

	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^*	●→●

Is this the only correct set of DH parameters? No.

What questions do you have ?

DH Parameters for the Stanford Arm



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

$a_i =$ 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.

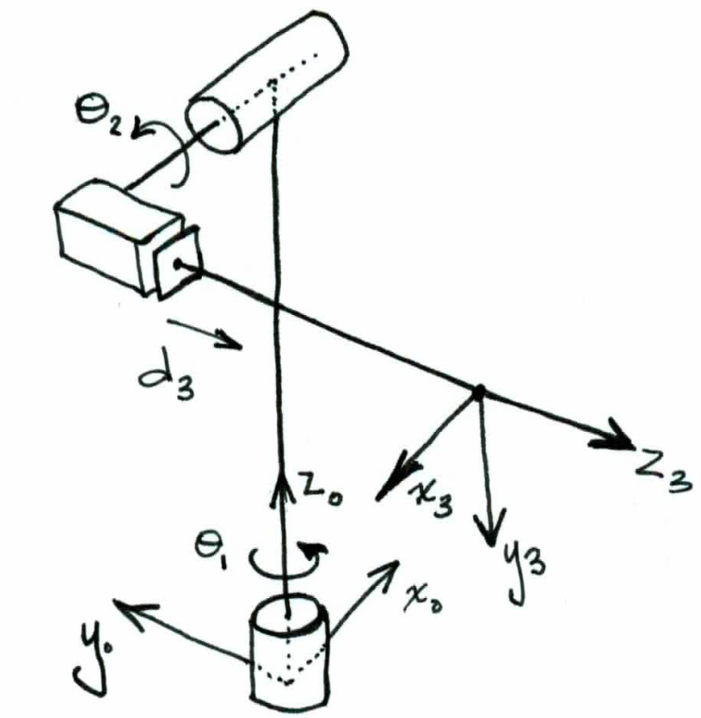
The right way to label variables on your diagram:

Precisely indicate start and end of measurement.

Use a directed arrow (only one arrowhead) for joint variables to show the positive direction.

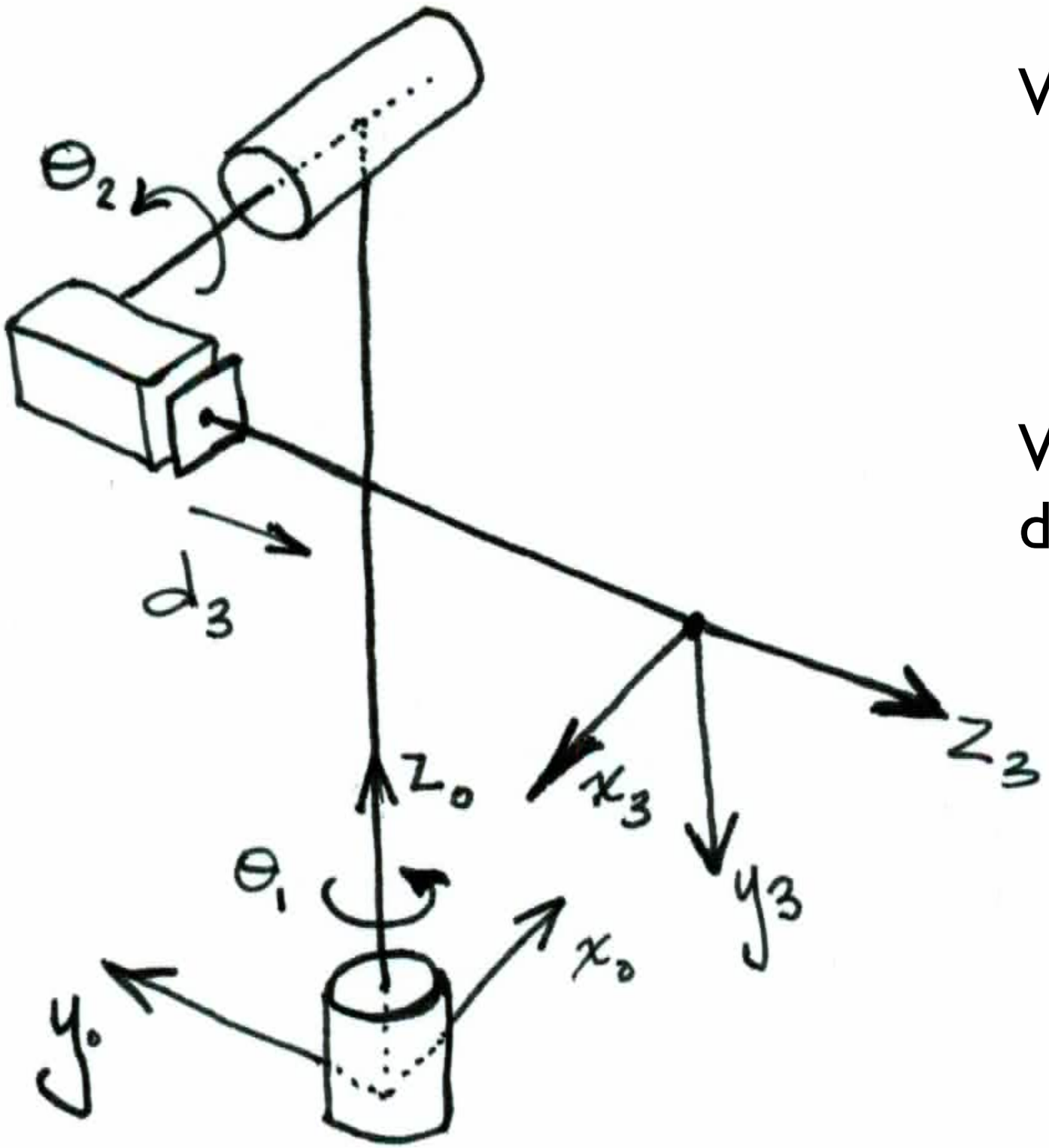
A Trickier DH Example

DH Parameters for Another Robot



Link	a_i	α_i	d_i	θ_i

Shown at $\theta_1^* = 0, \theta_2^* = 0,$ and $d_3* > 0$



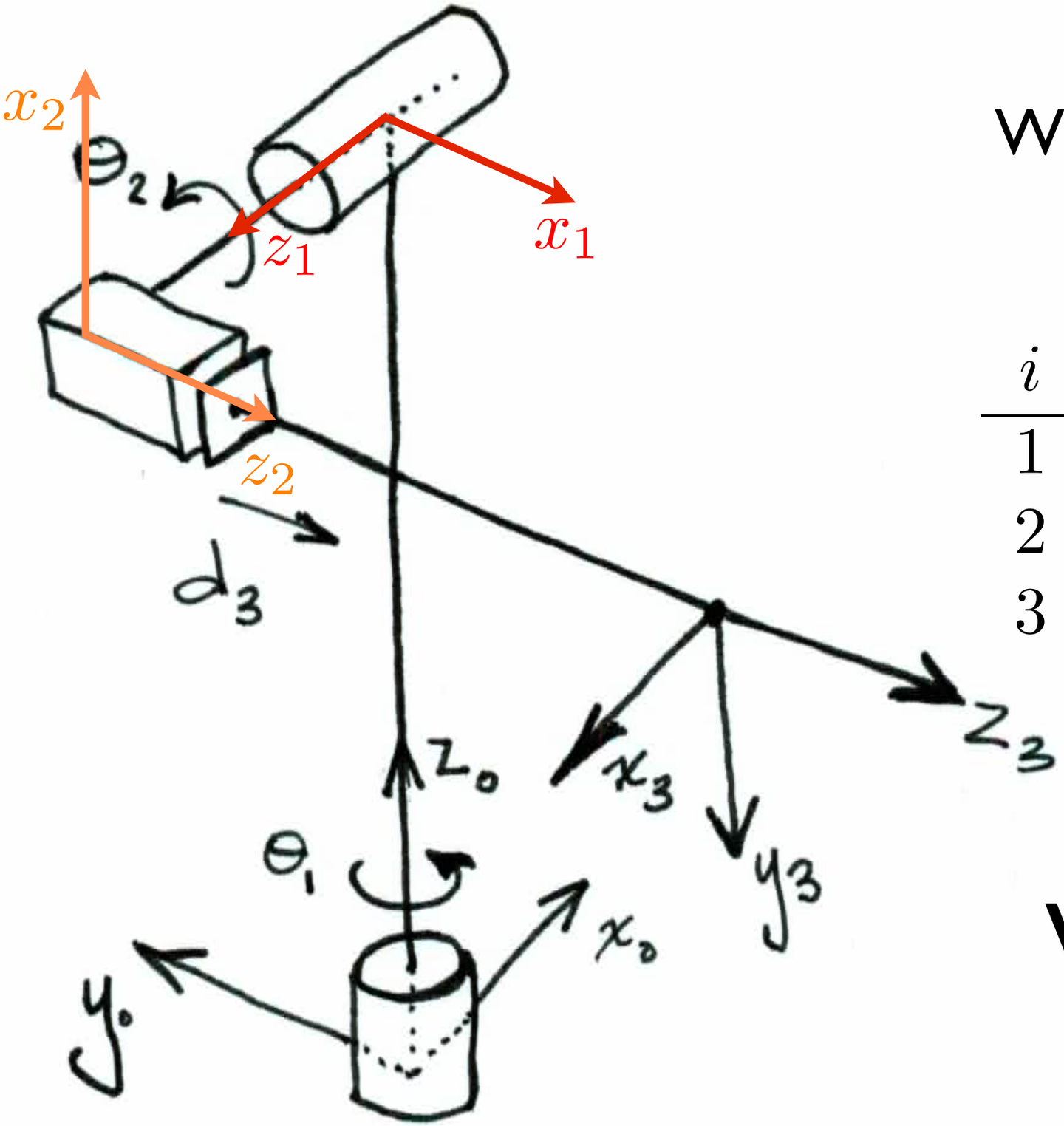
What type of robot is this?

- Stanford Arm in a different zero configuration!

Work with a partner to determine:

- Where should all the z-axes be?
- Where should all the frame origins be?
- Where should all the x-axes be?

Shown at $\theta_1^* = 0, \theta_2^* = 0,$ and $d_3* > 0$



What are the DH parameters?

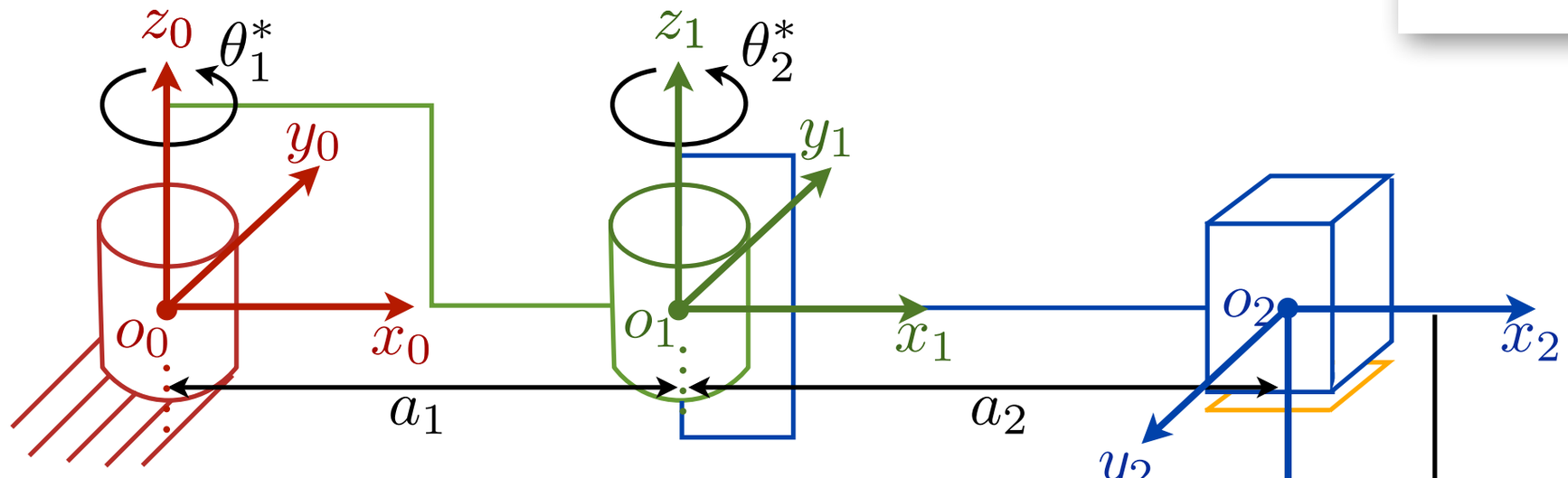
i	a	α	d	θ
1				
2				
3				

What questions do you have ?

Animating the SCARA Robot

Forward Kinematics of the SCARA Robot

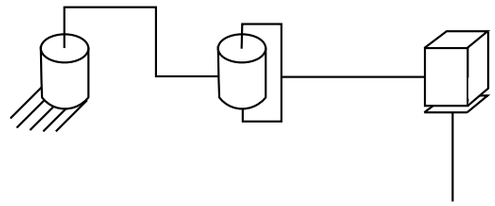
Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ_1^*
2	a_2	$180°$	0	θ_2^*
3	0	0	d_3^*	0



$$T_3^0 = A_1\, A_2\, A_3$$

$$T_3^0 = \begin{bmatrix} c_{12}^* & s_{12}^* & 0 & a_1 c_1^* + a_2 c_{12}^* \\ s_{12}^* & -c_{12}^* & 0 & a_1 s_1^* + a_2 s_{12}^* \\ 0 & 0 & -1 & -d_3^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

DH Parameters for the SCARA Robot



Link	a_i	α_i	d_i	θ_i

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 1: 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.

$a_i =$ 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.

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/06 dh convention/scara_robot_kuchenbe.m

EDITOR | PUBLISH | VIEW

scara_robot_kuchenbe.m | dh_kuchenbe.m | +

```
%% scara_robot_kuchenbe.m
%
% This Matlab script shows an animation of a SCARA robot for MEAM 520 at
% the University of Pennsylvania. The original was written by Professor
% Katherine J. Kuchenbecker in September of 2012.



%% SETUP

% Clear all variables from the workspace.
clear all

% Clear the console, so you can more easily find any errors that may occur.
home

% Define our time vector.
tStart = 0;   % The time at which the simulation starts, in seconds.
tStep = 0.04; % The simulation's time step, in seconds.
tEnd = 10;    % The time at which the simulation ends, in seconds.
t = (tStart:tStep:tEnd)';  % The time vector (a column vector).

% Set whether to animate the robot's movement and how much to slow it down.
pause on;  % Set this to off if you don't want to watch the animation.
GraphingTimeDelay = 0.005; % The length of time that Matlab should pause bet


%% MOTION MODES

% Ask the user to get the mode of the robot motion.
[selection,ok] = listdlg('PromptString','Which robot joints should move?','S
```

script | Ln 5 Col 51

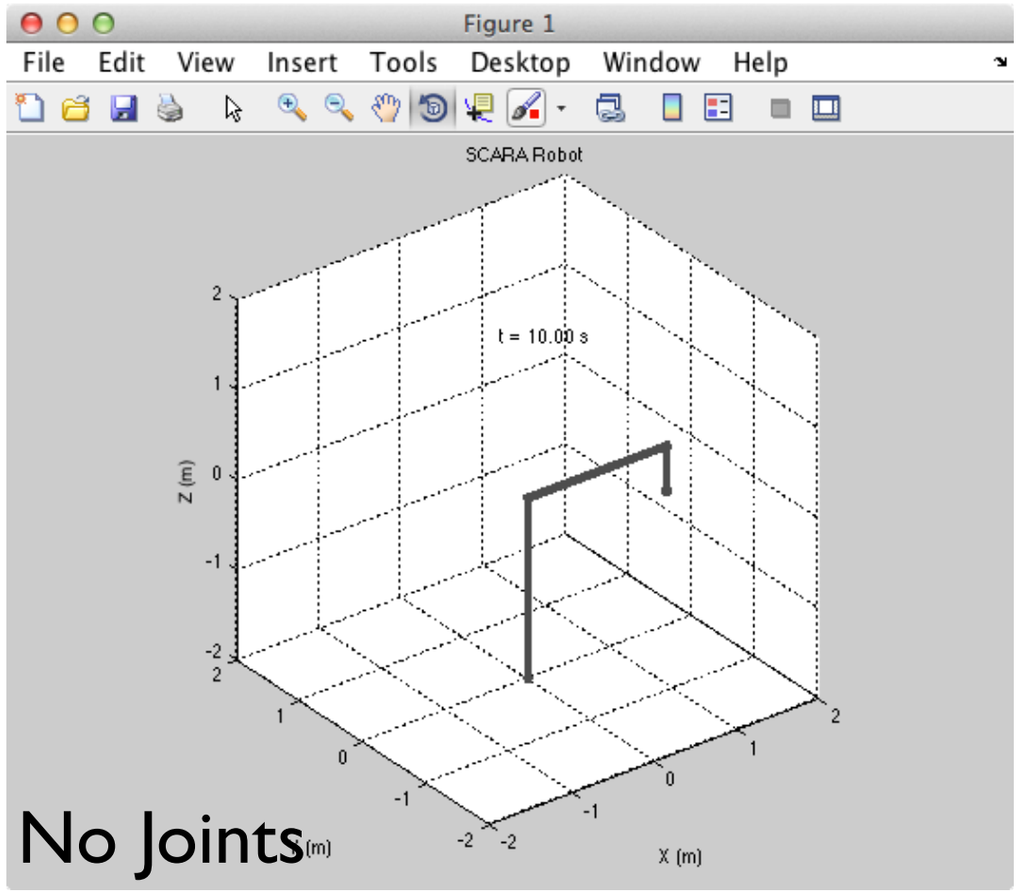


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot
t = 10.00 s
Z (m)
X (m)
No Joints

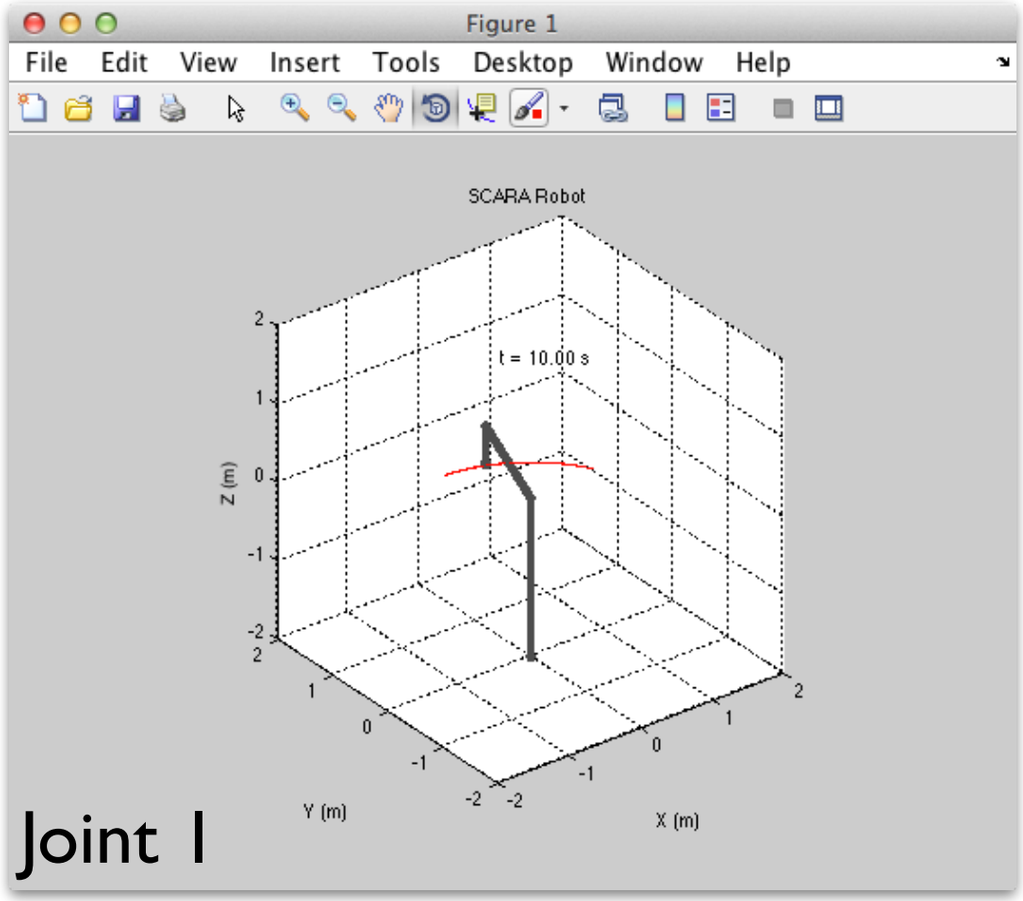
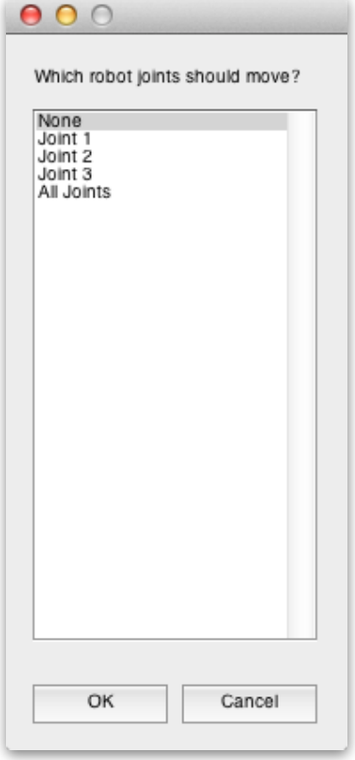


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot
t = 10.00 s
Z (m)
Y (m)
X (m)
Joint 1


Which robot joints should move?
None
Joint 1
Joint 2
Joint 3
All Joints
OK
Cancel

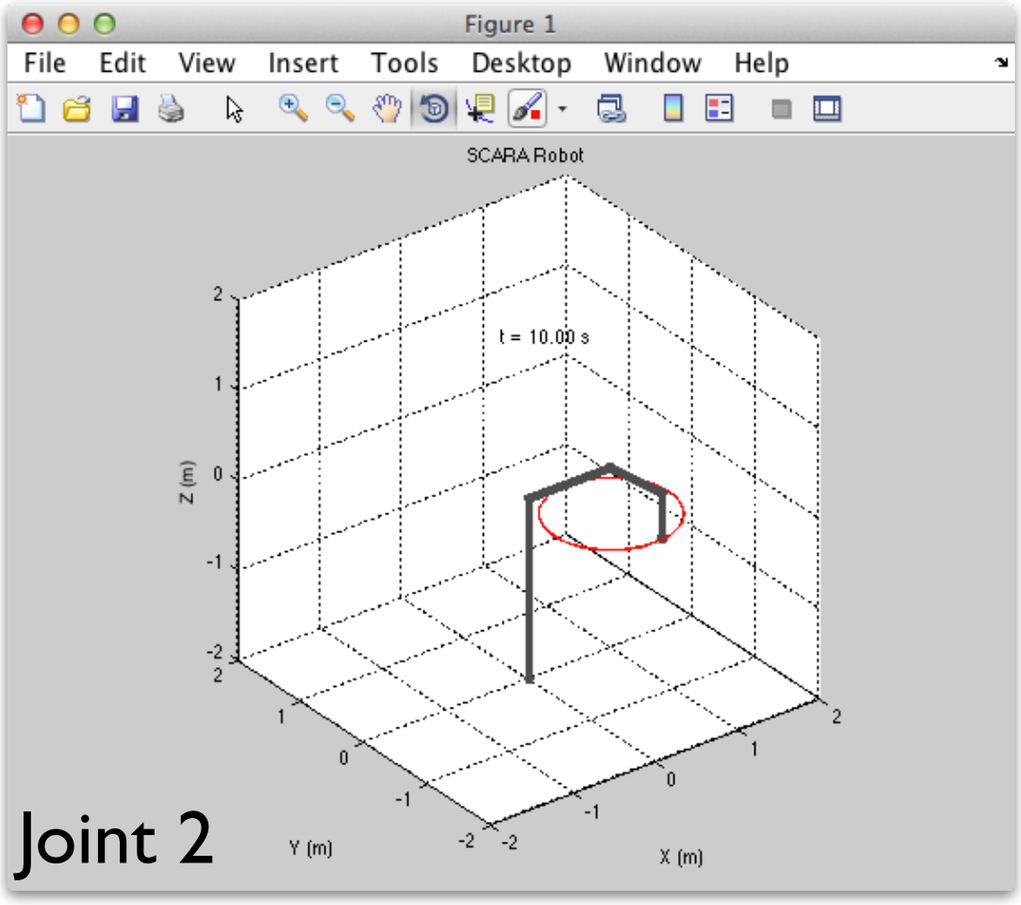


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot
t = 10.00 s
Z (m)
Y (m)
X (m)
Joint 2

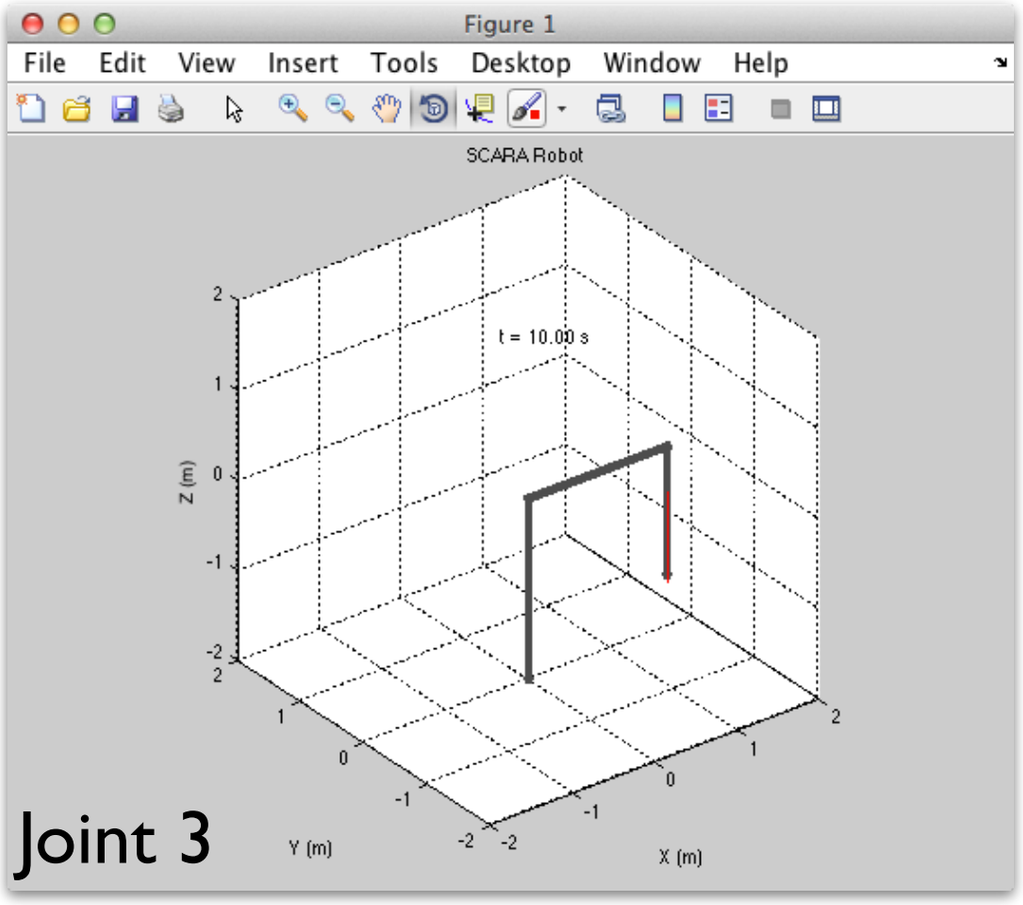


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot
t = 10.00 s
Z (m)
Y (m)
X (m)
Joint 3

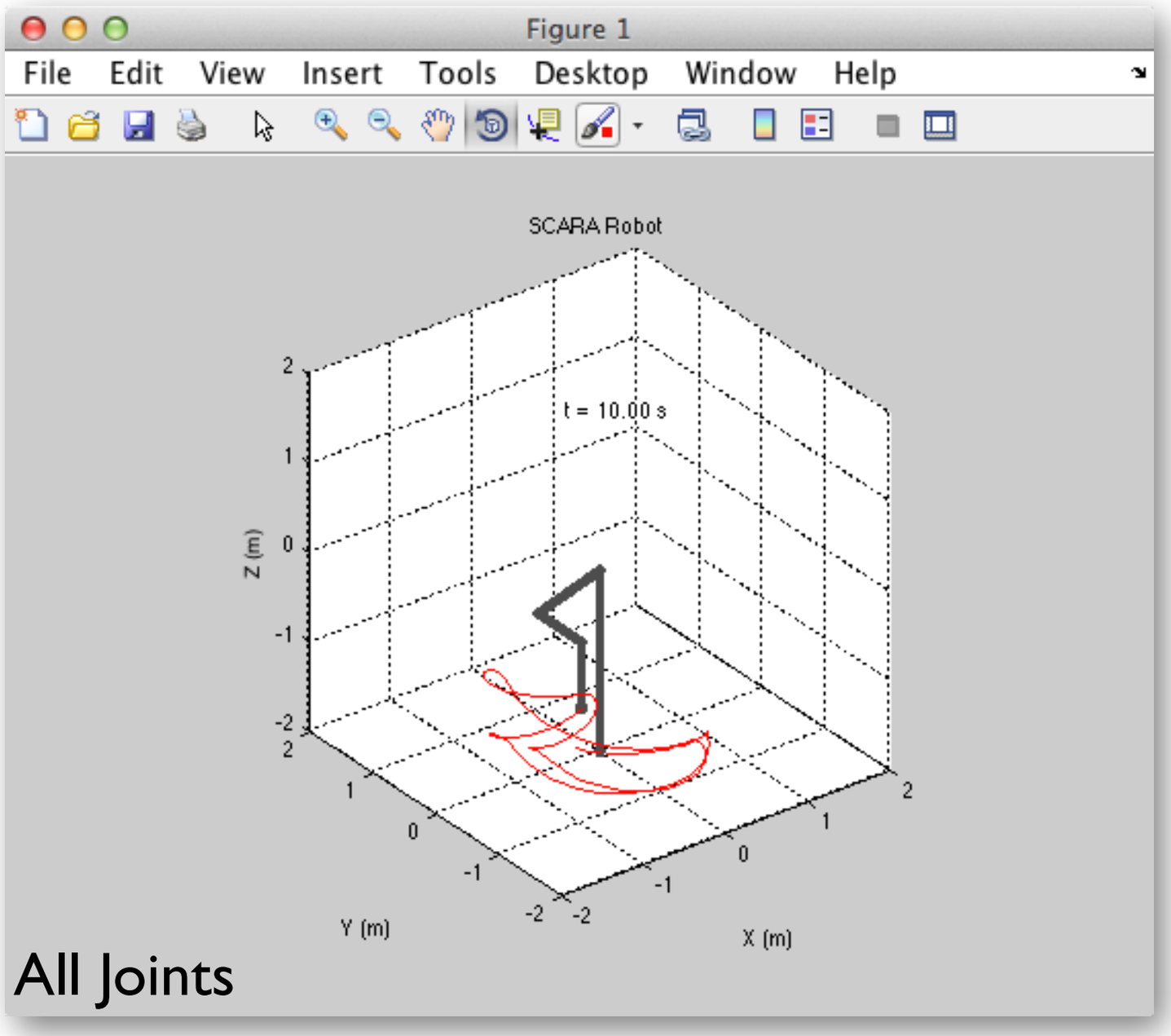
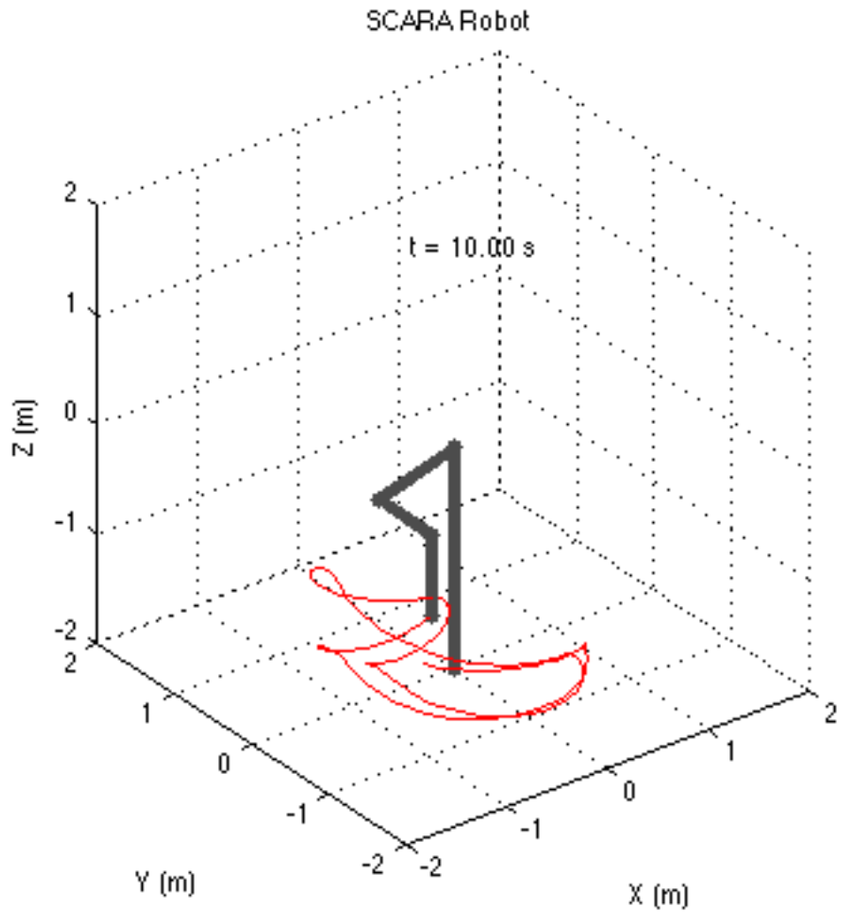


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot
t = 10.00 s
Z (m)
Y (m)
X (m)

All Joints

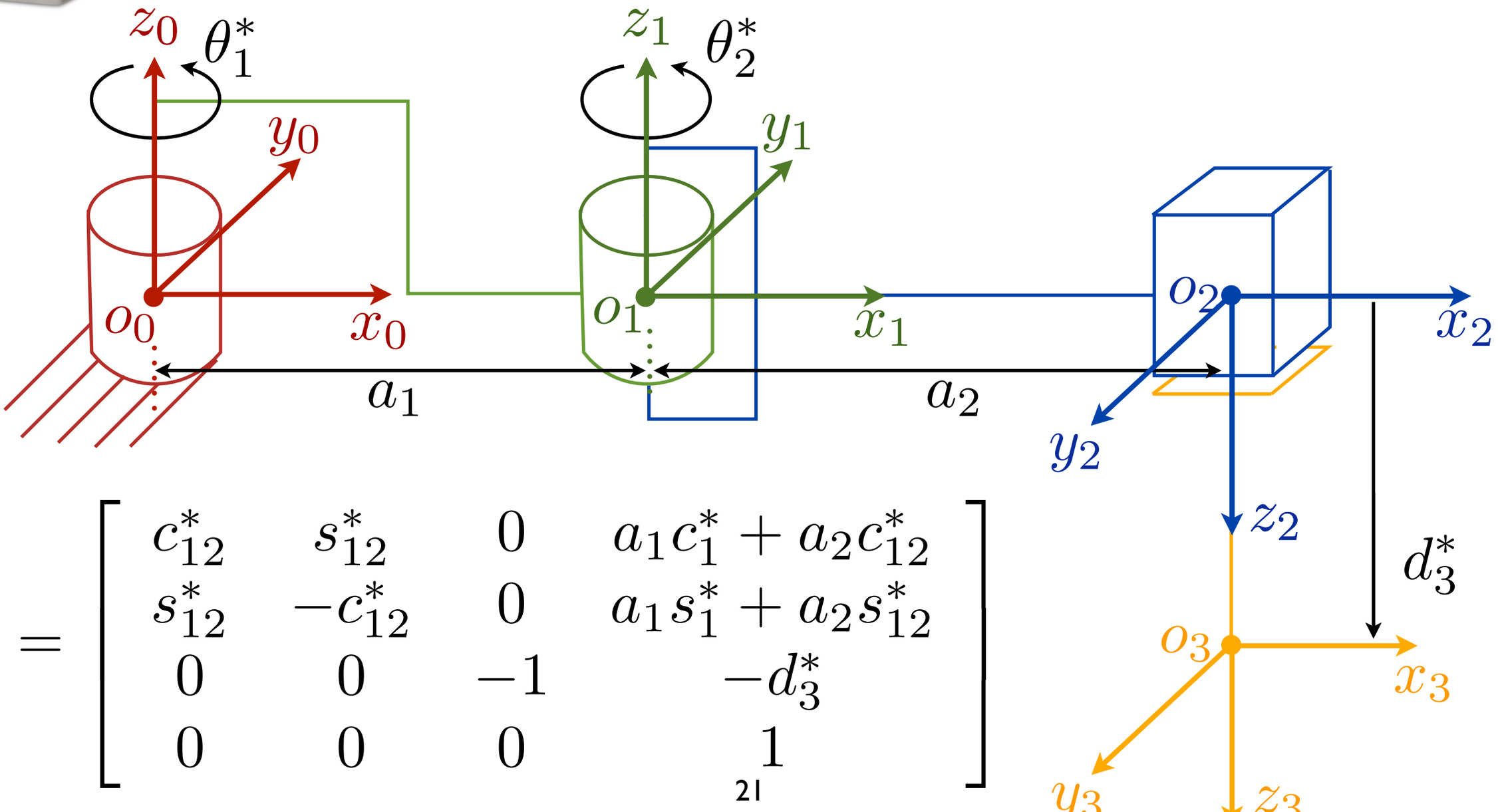
How did I plot the whole robot, instead of just the tip?



Talk to a partner.

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.

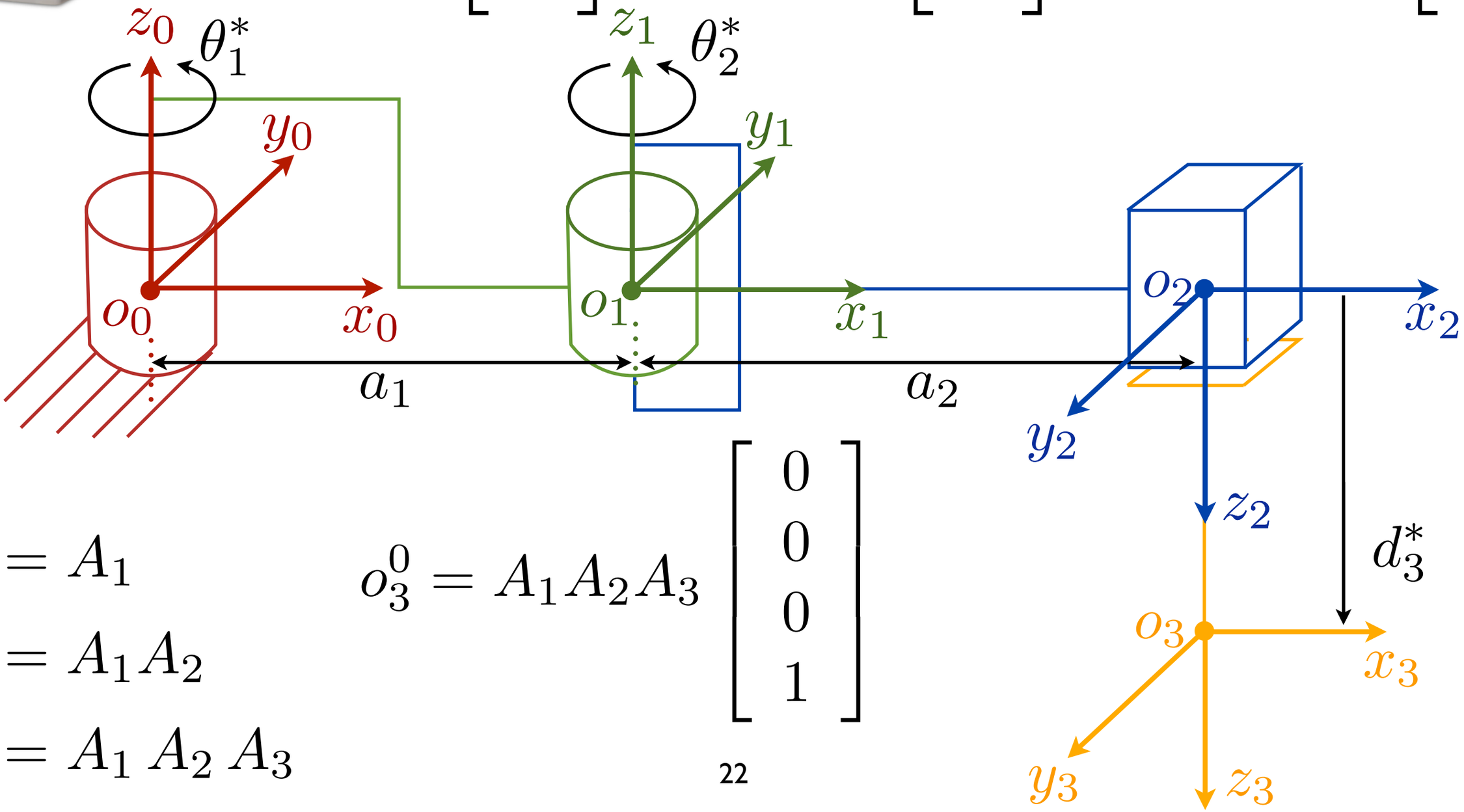
$$T_3^0 = A_1\, A_2\, A_3$$



$$T_3^0 = \begin{bmatrix} c_{12}^* & s_{12}^* & 0 & a_1 c_1^* + a_2 c_{12}^* \\ s_{12}^* & -c_{12}^* & 0 & a_1 s_1^* + a_2 s_{12}^* \\ 0 & 0 & -1 & -d_3^* \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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.

$$o_0^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad o_1^0 = A_1 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad o_2^0 = A_1 A_2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$



$$T_1^0 = A_1$$

$$T_2^0 = A_1 A_2$$

$$T_3^0 = A_1\, A_2\, A_3$$

$$o_3^0 = A_1 A_2 A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/06 dh convention/scara_robot_kuchenbe.m

EDITOR PUBLISH VIEW

scara_robot_kuchenbe.m | dh_kuchenbe.m | +

```
140         % Create the homogeneous representation of the origin of a frame.
141 -       o = [0 0 0 1]';
142
143         % Calculate the position of the origin of frame 0 expressed in frame 0.
144         % By definition, this is just o.
145 -       o0 = o;
146
147         % Calculate the position of the origin of frame 1 expressed in frame 0.
148         % We multiply A1 into the origin vector to find the position o1.
149 -       o1 = A1*o;
150
151         % Calculate the position of the origin of frame 2 expressed in frame 0.
152         % We multiply A1 and A2 into the origin vector to find the position o2.
153 -       o2 = A1*A2*o;
154
155         % Calculate the position of the origin of frame 3 expressed in frame 0.
156         % We multiply A1, A2, and A3 into the origin vector to find the position
157 -       o3 = A1*A2*A3*o;
158
159         % Put the points together.  Each column is the homogeneous
160         % representation of a point in the robot.  I have added an extra point
161         % at [0 0 -2 1]' to give the visual illusion of a base; this is not
162         % needed.  The points should go in order along the robot.
163 -       points_to_plot = [[0 0 -2 1]' o0 o1 o2 o3];
164
165         % Our list of points to plot is in points_to_plot, going from the base
166         % out to the tip.  Each column represents the x, y, and z coordinates
167         % of one point.  The last column of points_to_plot should be the
168         % position of the tip of the robot.
169
```

script Ln 17 Col 12

What questions do you have?

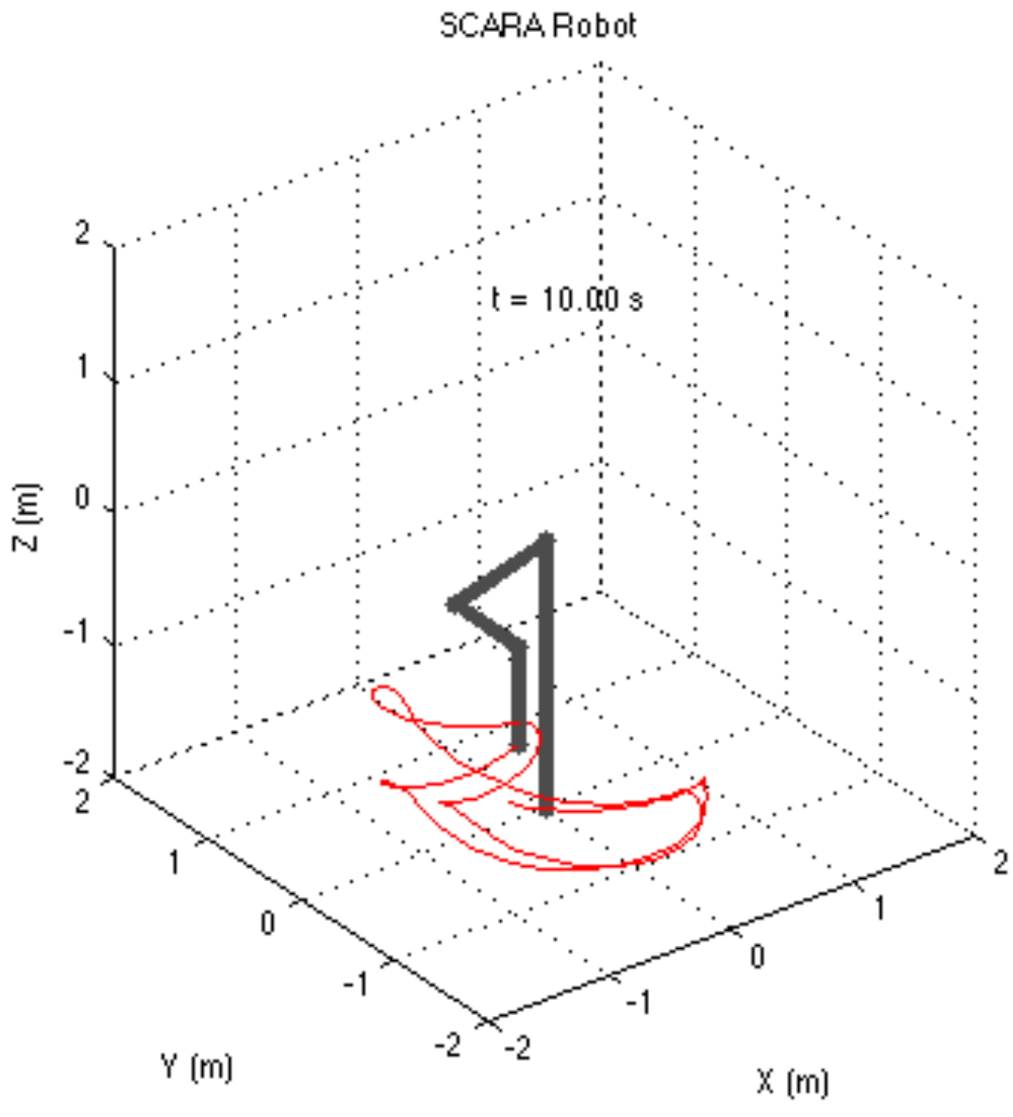
Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/06 dh convention/scara_robot_kuchenbe.m

EDITOR | PUBLISH | VIEW

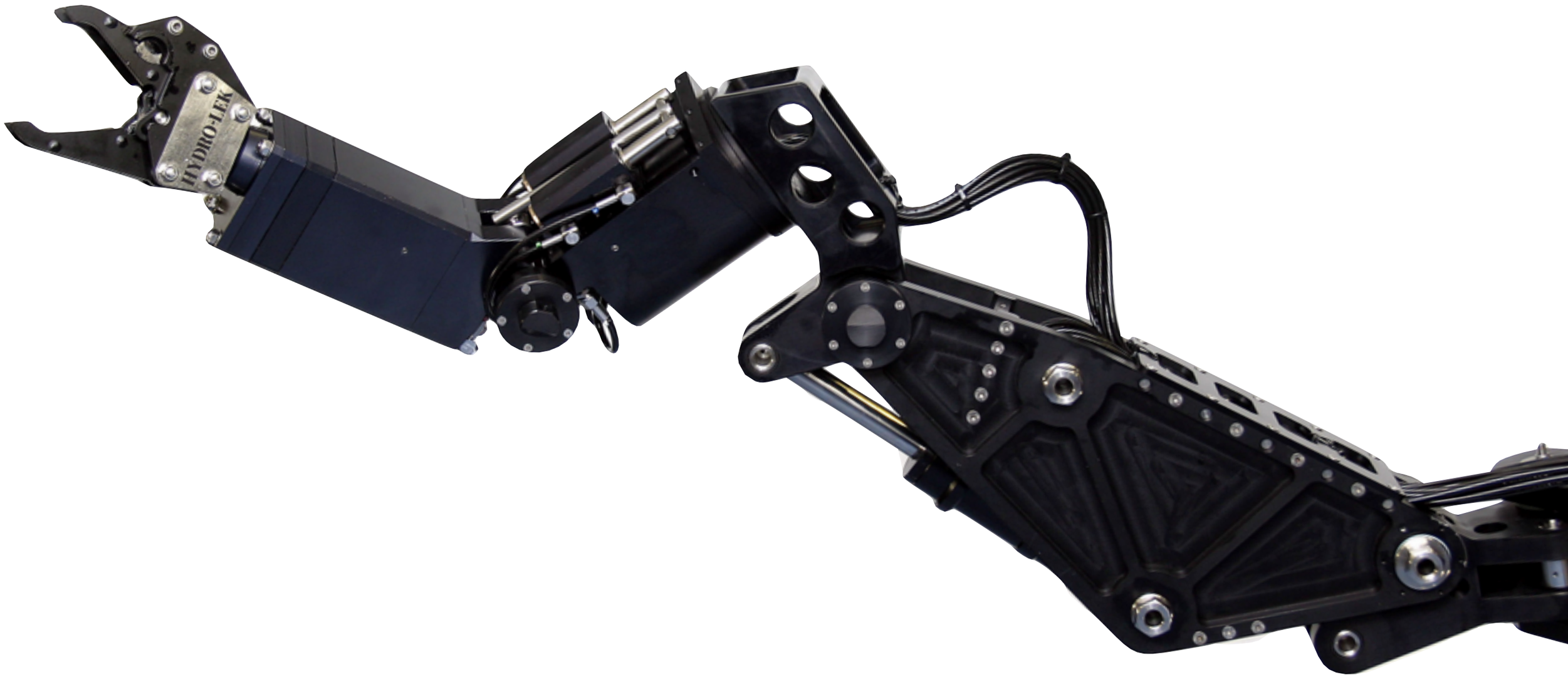
scara_robot_kuchenbe.m | dh_kuchenbe.m

```
1   %% scara_robot_kuchenbe.m
2   %
3   % This Matlab script shows an animation of a SCARA robot for MEAM 520 at
4   % the University of Pennsylvania. The original was written by Professor
5   % Katherine J. Kuchenbecker in September of 2012.
6
7
8   %% SETUP
9
10  % Clear all variables from the workspace.
11  clear all
12
13  % Clear the console, so you can more easily find any errors that may occur.
14  home
15
16  % Define our time vector.
17  tStart = 0;   % The time at which the simulation starts, in seconds.
18  tStep = 0.04; % The simulation's time step, in seconds.
19  tEnd = 10;    % The time at which the simulation ends, in seconds.
20  t = (tStart:tStep:tEnd)';  % The time vector (a column vector).
21
22  % Set whether to animate the robot's movement and how much to slow it down.
23  pause on;  % Set this to off if you don't want to watch the animation.
24  GraphingTimeDelay = 0.005; % The length of time that Matlab should pause bet
25
26
27  %% MOTION MODES
28
29  % Ask the user to get the mode of the robot motion.
30  [selection,ok] = listdlg('PromptString','Which robot joints should move?','S
```

script | Ln 5 Col 51



Trajectory Planning



SHV 5.5

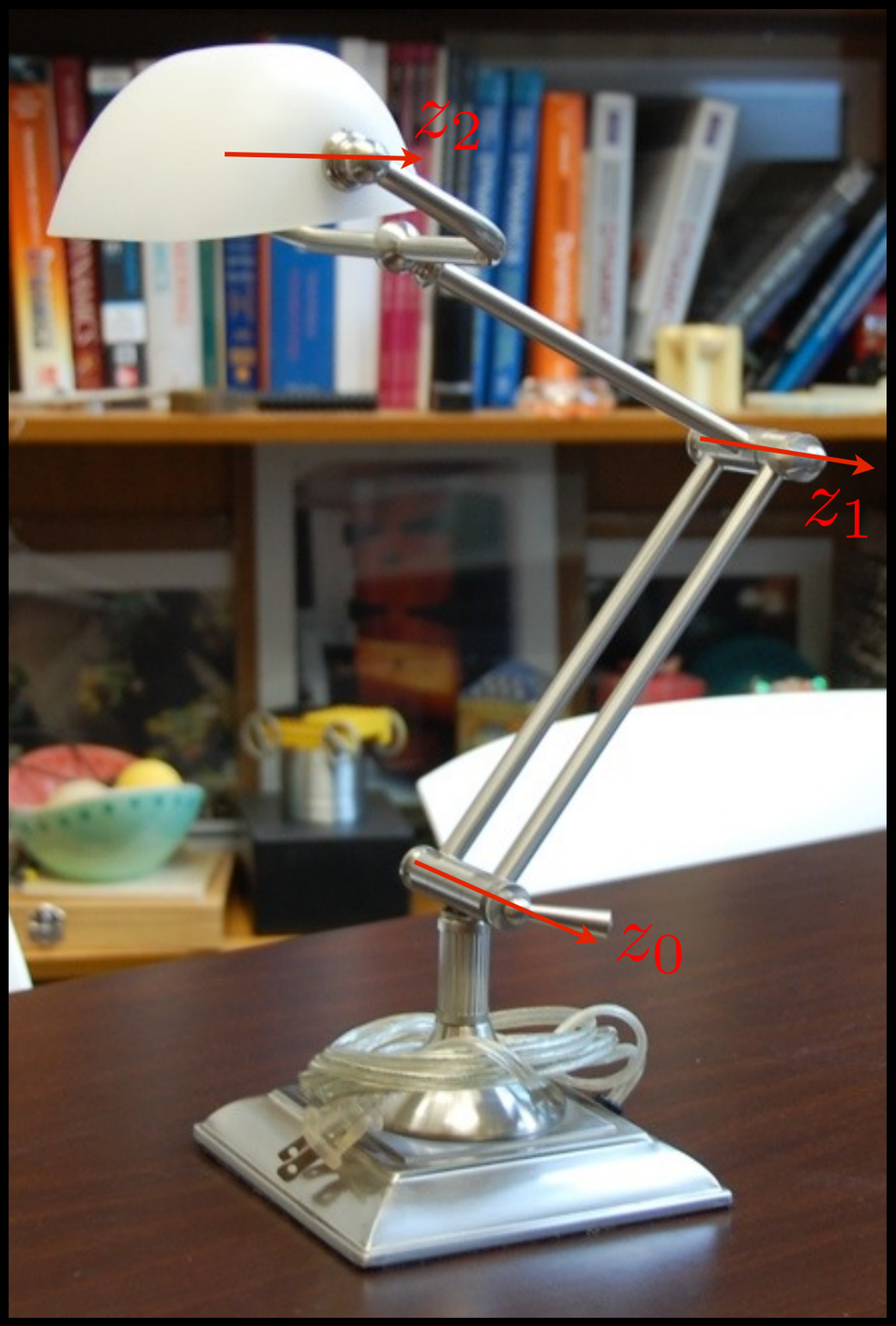
Rationale

Forward Kinematics (FK) is beautiful but can take a little while to master.

Inverse Kinematics (IK) is often ugly; we should practice FK more to prepare for it.

We will come back to IK (Sec. 3.3) after studying trajectory planning (Sec. 5.5), which nicely builds on FK.

Two arm poses... How do I move between them?



A manipulator's **configuration** is a complete specification of the location of every point on the manipulator.

$$\vec{q} = \begin{bmatrix} q_i \\ \vdots \\ q_n \end{bmatrix}$$

- We use a **joint variable** to denote each joint's position.
- Value defines the joint's displacement from a **zero configuration**.
- Use **θ** for revolute joints.
- Use **d** for prismatic joints.
- Axis orientation defines the **positive direction**.
- Knowing all joint variable values defines the configuration.

A **trajectory** is a function of time $\vec{q}(t)$

Such that $\vec{q}(t_0) = \vec{q}_s$

and $\vec{q}(t_f) = \vec{q}_f$

Parameterized by time, so we can compute velocities and accelerations along the trajectory by differentiation.

A **via point** is a configuration that the robot needs to achieve.

Via points could be obtained in many ways:

Hand programming by picking joint angles.

Physically leading the robot through the desired motion and recording joint angles.

Specified as a T_6^0 pose, using IK to calculate the corresponding joint angles.

May 1984
New Information
Mailed To: E, C, D/22-505A, C, E

A compact, computer-controlled robot for high speed, close-tolerance assembly, light materials handling, and inspection applications.

UNIMATE® PUMA® Series 200 Industrial Robot

IROS 2011 Montage | Willow Garage

http://www.willowgarage.com/blog/2011/10/11/iros-2011-montage

Google

ABOUT US | DOWNLOAD | JOBS | CONTACT | SUPPORT

Willow Garage

ROBOTS SOFTWARE RESEARCH BLOG

SEARCH

Recent Posts

Willow Garage employees join Suitable Technologies

August 21, 2013

Read More »

Pick and Place with MoveIt! on the PR2 Robot

August 20, 2013

Read More »

Enabling robots to see better through improved camera calibration

August 9, 2013

Read More »

Intuitive Robot Interfaces with Web Robotics

August 1, 2013

IROS 2011 Montage

Submitted by admin on Tue, 10/11/2011 - 18:44

We had a busy and fun time at this year's IROS 2011. Thanks to all of you who stopped by our booth and talks. A lot of great robotics research was on display and we had a great time at the PR2 Workshop, talks, and interactive presentations. The exhibition was very exciting and we put together a montage video to celebrate all of the robots in action. Enjoy!


powered by
pcl
PR-J
PR2

A **trajectory** is a function of time $\vec{q}(t)$

Such that $\vec{q}(t_0) = \vec{q}_s$

and $\vec{q}(t_f) = \vec{q}_f$

How many trajectories exist that satisfy these constraints?

Infinitely many.

What if I also specify starting and final velocities?

There are still infinitely many trajectories.

Roboticists typically choose trajectories from a **finitely parameterizable family**, such as **polynomials of degree n**.

How many constraints may we impose when calculating an nth-order polynomial?

cubic polynomial

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

We may impose **n+1 constraints** because there are n+1 coefficients in an nth-order polynomial.

For point-to-point motion, each joint's motion is typically planned independently, so we'll consider just a single joint angle.

instead of $\vec{q}(t)$

$$q(t) = \theta_i(t) \quad \text{or} \quad q(t) = d_i(t)$$

Simplest Situation: Specifying Joint Value Only

Initial Condition

$$q(t_0) = q_0$$

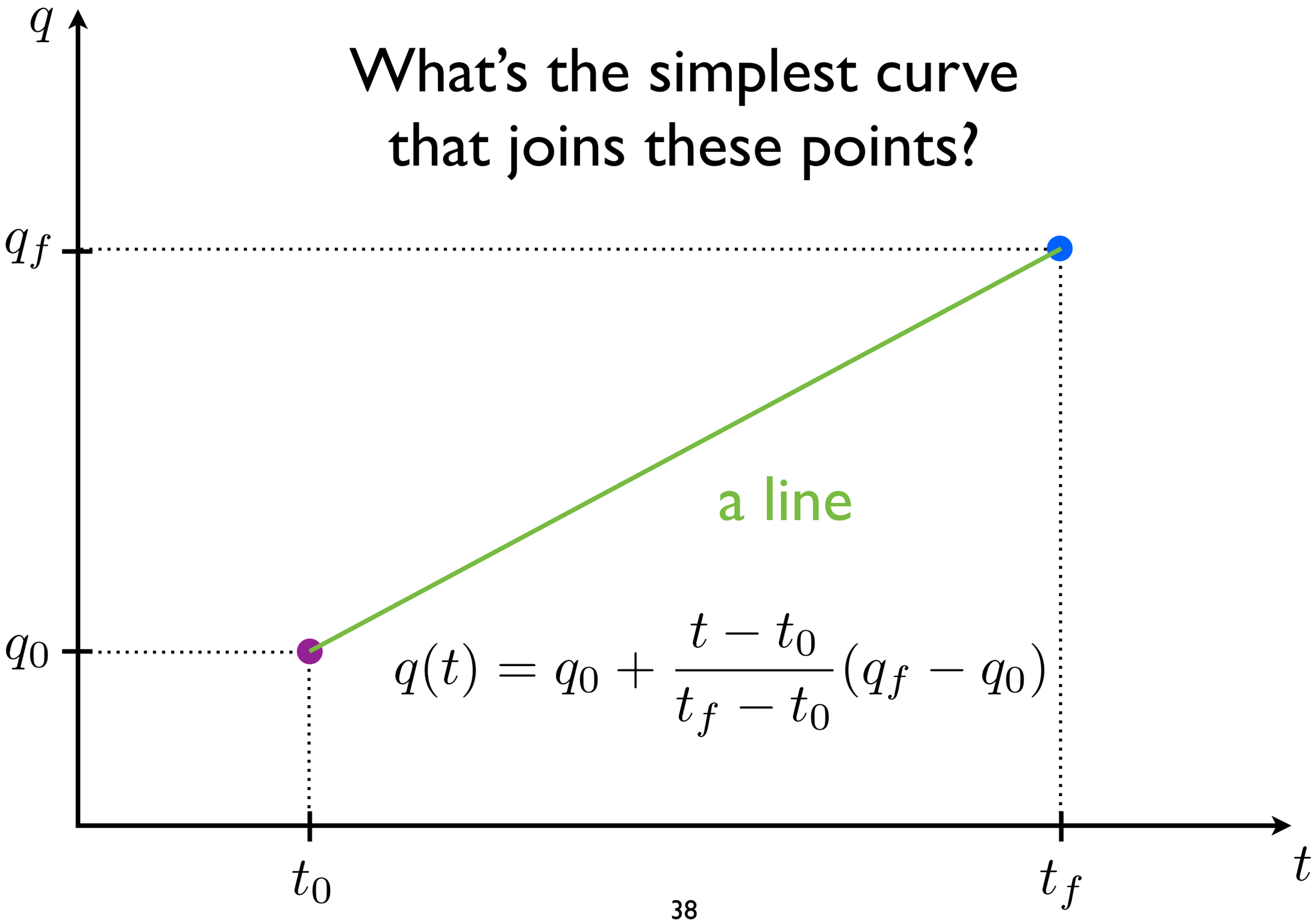
Final Condition

$$q(t_f) = q_f$$

Simplest Situation: Specifying Joint Value Only

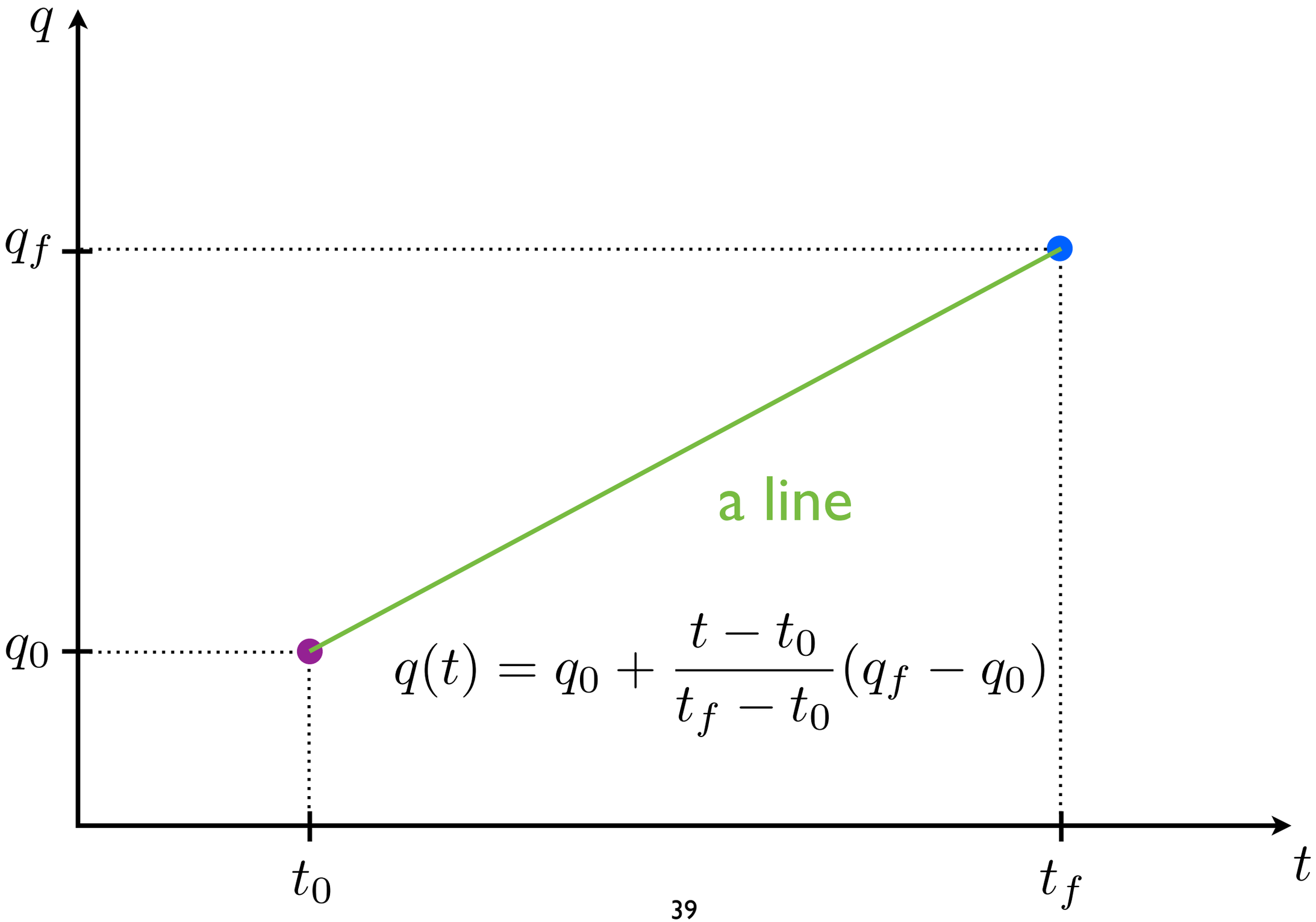
$$q(t_0) = q_0 \qquad\qquad q(t_f) = q_f$$

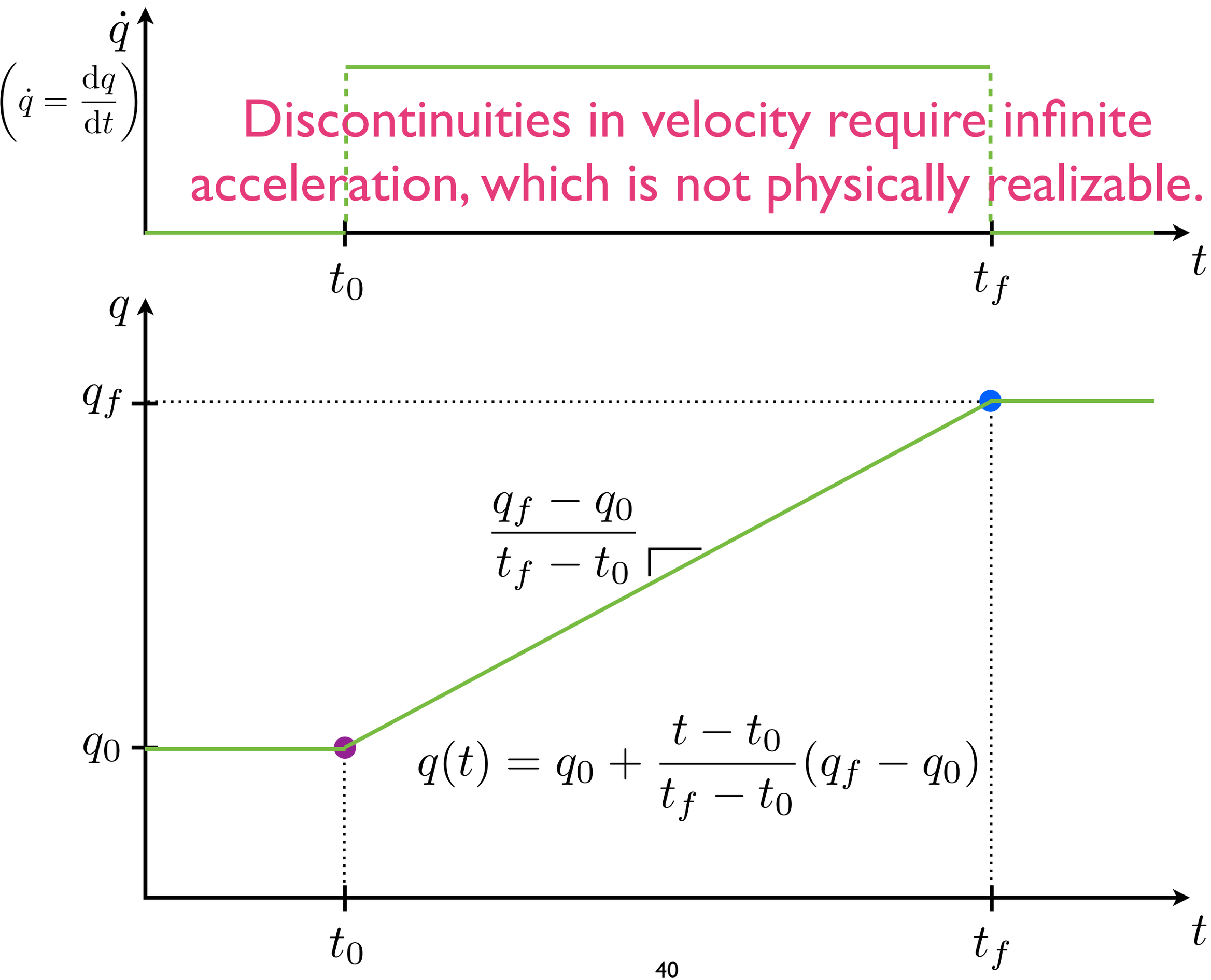
What's the simplest curve that joins these points?



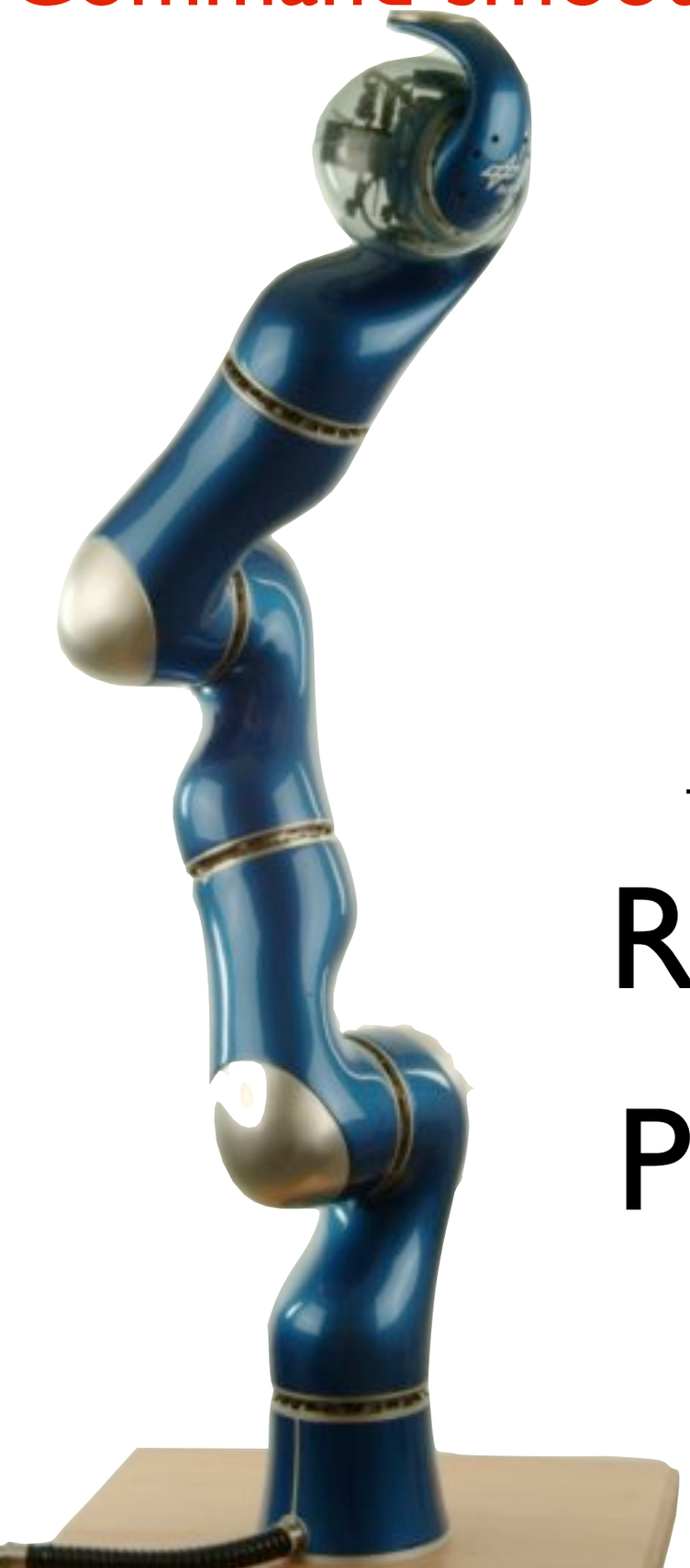
Linear interpolation is really useful!

Why do you think SHV doesn't present lines?




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

Robots are actually flexible!
Command smooth trajectories to avoid exciting flexibilities.



Hanging slinky example.

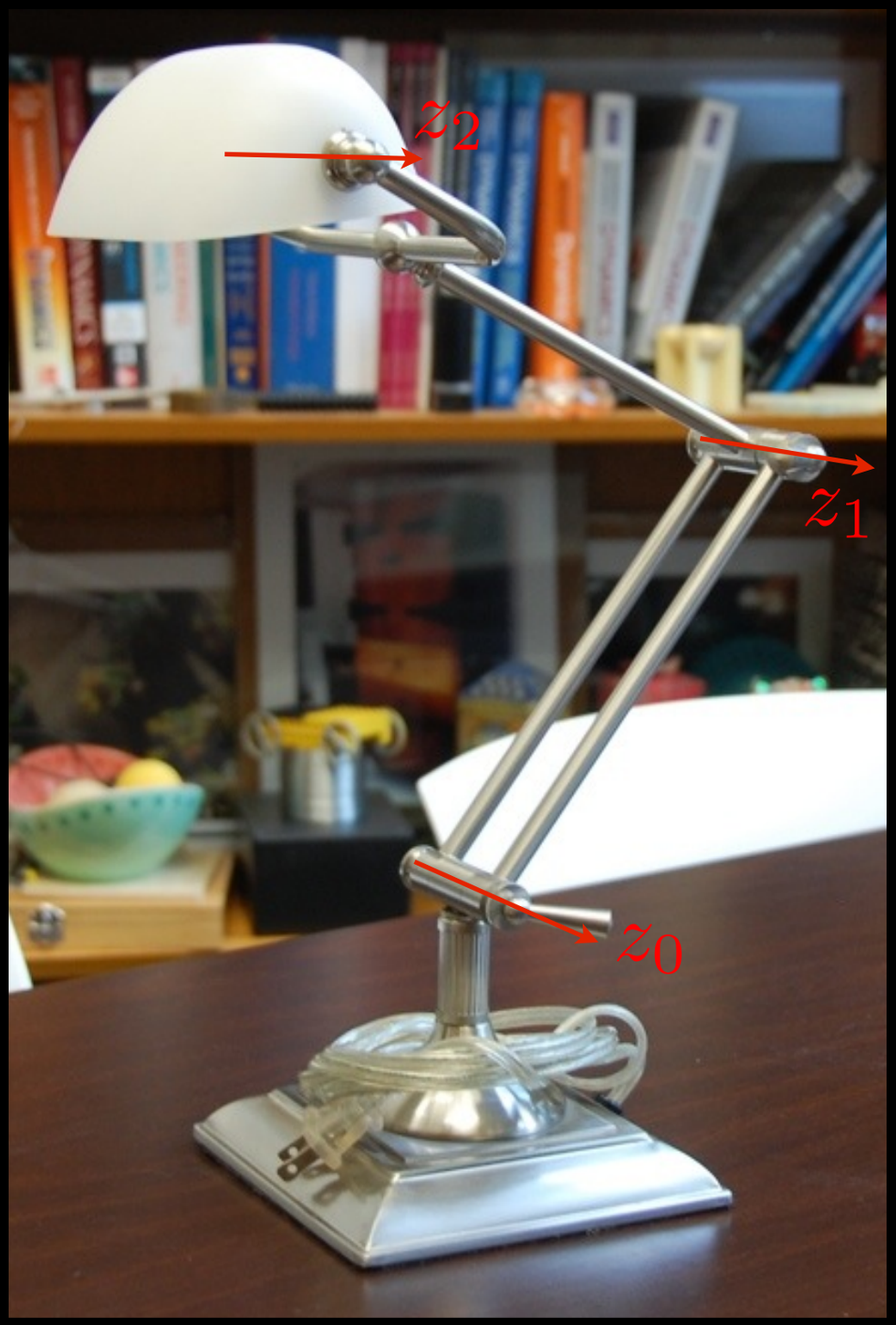
Robot manipulators are composed of:

- 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



Does the **configuration** of a manipulator fully define how it will move in the future?

- No. It only gives you an **instantaneous description** of the geometry.

- The manipulator's **state** is a set of variables that is sufficient to tell you its future time response when combined with dynamics and future inputs.

- State requires both the joint variables and their **derivatives**.

Specifying Joint Values and First Time Derivatives

Initial Conditions

$$q(t_0) = q_0$$ **Units** angle or distance, e.g., rad or m

$$\dot{q}(t_0) = v_0$$ **Units** angle per time or distance per time, e.g., rad/s or m/s

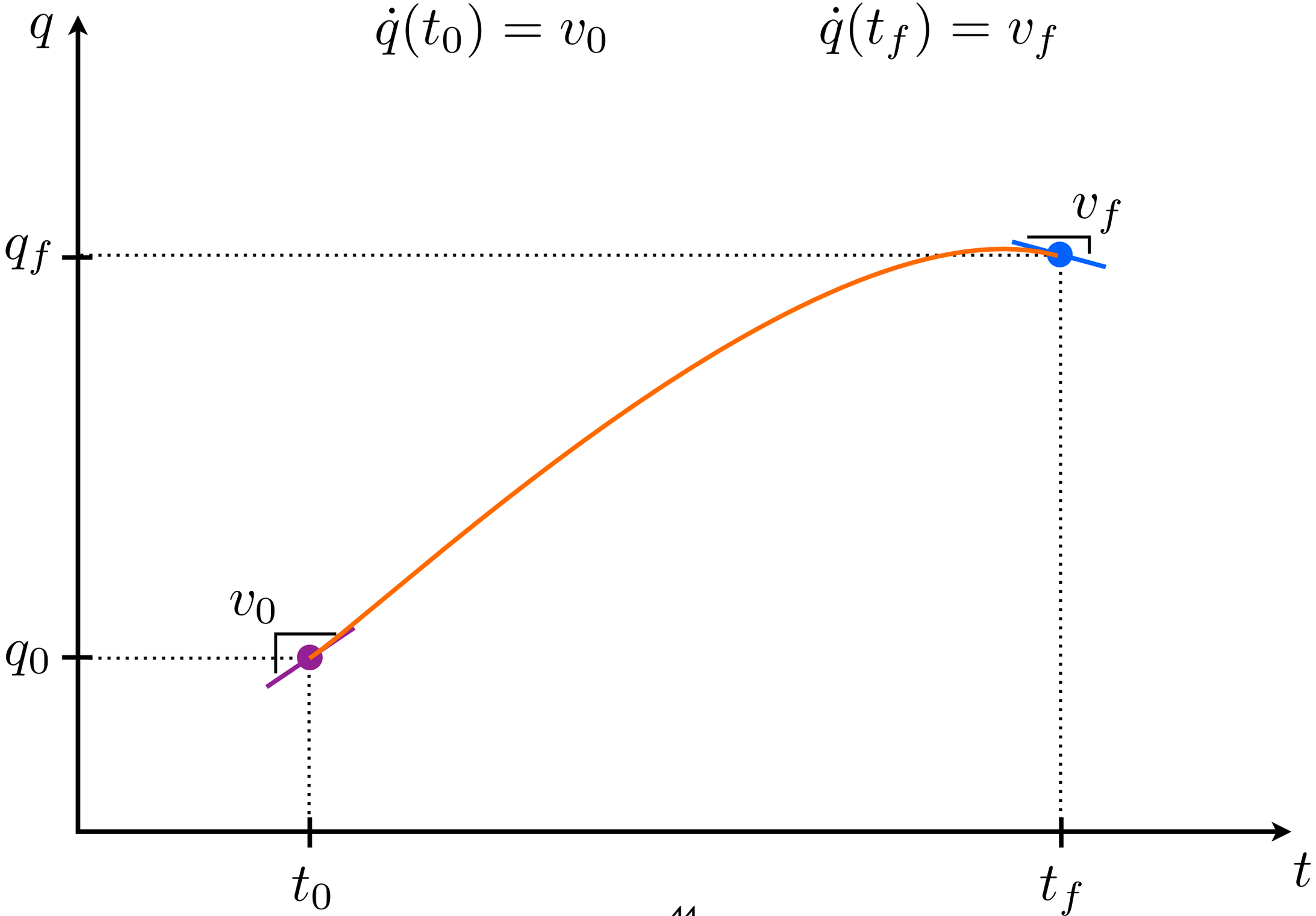
Final Conditions

$$q(t_f) = q_f$$

$$\dot{q}(t_f) = v_f$$

Specifying Joint Values and First Time Derivatives

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



Specifying Joint Values and First Time Derivatives

Cubic Polynomial Trajectories

start → end

$$q(t_0) = q_0 \longrightarrow q(t_f) = q_f$$

$$\dot{q}(t_0) = v_0 \longrightarrow \dot{q}(t_f) = v_f$$

cubic polynomial

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

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

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

Specifying Joint Values and First Time Derivatives
Cubic Polynomial Trajectories

System of Four Equations

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$

$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$

$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

$$\vec{x} = A^{-1}\vec{b}$$

time matrix

conditions ↓ unknown coefficients

Reformulate in $\vec{b} = A\vec{x}$ form

$$\vec{x} = A \backslash \vec{b}$$

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

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/10 trajectories/make_cubic_polynomial.m

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m

```
%% Clear the workspace.
clear


%% Define the problem.

% Define initial and final times.
t0 = 0; % s
tf = 2; % s

% Define initial conditions.
q0 = -2.6; % radians
v0 = 0;    % rad/s

% Define final conditions.
qf = 1.3;  % radians
vf = 0;    % rad/s


%% Solve for the cubic polynomial coefficients that meet these conditions.

% Put initial and final conditions into a column vector.
conditions = [q0 v0 qf vf]';

% Put time elements into matrix.
mat = [1    t0   t0^2     t0^3;
       0     1   2*t0   3*t0^2;
       1    tf   tf^2     tf^3;
       0     1   2*tf   3*tf^2];

% Solve for coefficients.
coeffs = mat \ conditions;

% Pull individual coefficients out.
a0 = coeffs(1);
a1 = coeffs(2);
a2 = coeffs(3);
a3 = coeffs(4);
```

script | Ln 5 Col 23

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/10 trajectories/make_cubic_polynomial.m

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m

```
20      %% Solve for the cubic polynomial coefficients that meet these conditions.
21
22      % Put initial and final conditions into a column vector.
23 -    conditions = [q0 v0 qf vf]';
24
25      % Put time elements into matrix.
26 -    mat = [1    t0   t0^2     t0^3;
27             0     1   2*t0   3*t0^2;
28             1    tf   tf^2     tf^3;
29             0     1   2*tf   3*tf^2];
30
31      % Solve for coefficients.
32 -    coeffs = mat \ conditions;
33
34      % Pull individual coefficients out.
35 -    a0 = coeffs(1);
36 -    a1 = coeffs(2);
37 -    a2 = coeffs(3);
38 -    a3 = coeffs(4);
39
40
41      %% Plot the cubic polynomial we calculated.
42
43      % Create time vector.
44 -    tstep = 0.01;
45 -    t = (t0:tstep:tf)';
46
47      % Calculate cubic trajectory with coefficients.
48 -    q = a0 + a1*t + a2*t.^2 + a3*t.^3;
49
50      % Open figure 1.
51 -    figure(1)
52 -    clf
53
54      % Plot cubic trajectory.
55 -    set(gca,'fontsize',14)
56 -    plot(t,q,'b')
57 -    xlabel('Time (s)')
```

script | Ln 21 Col 1

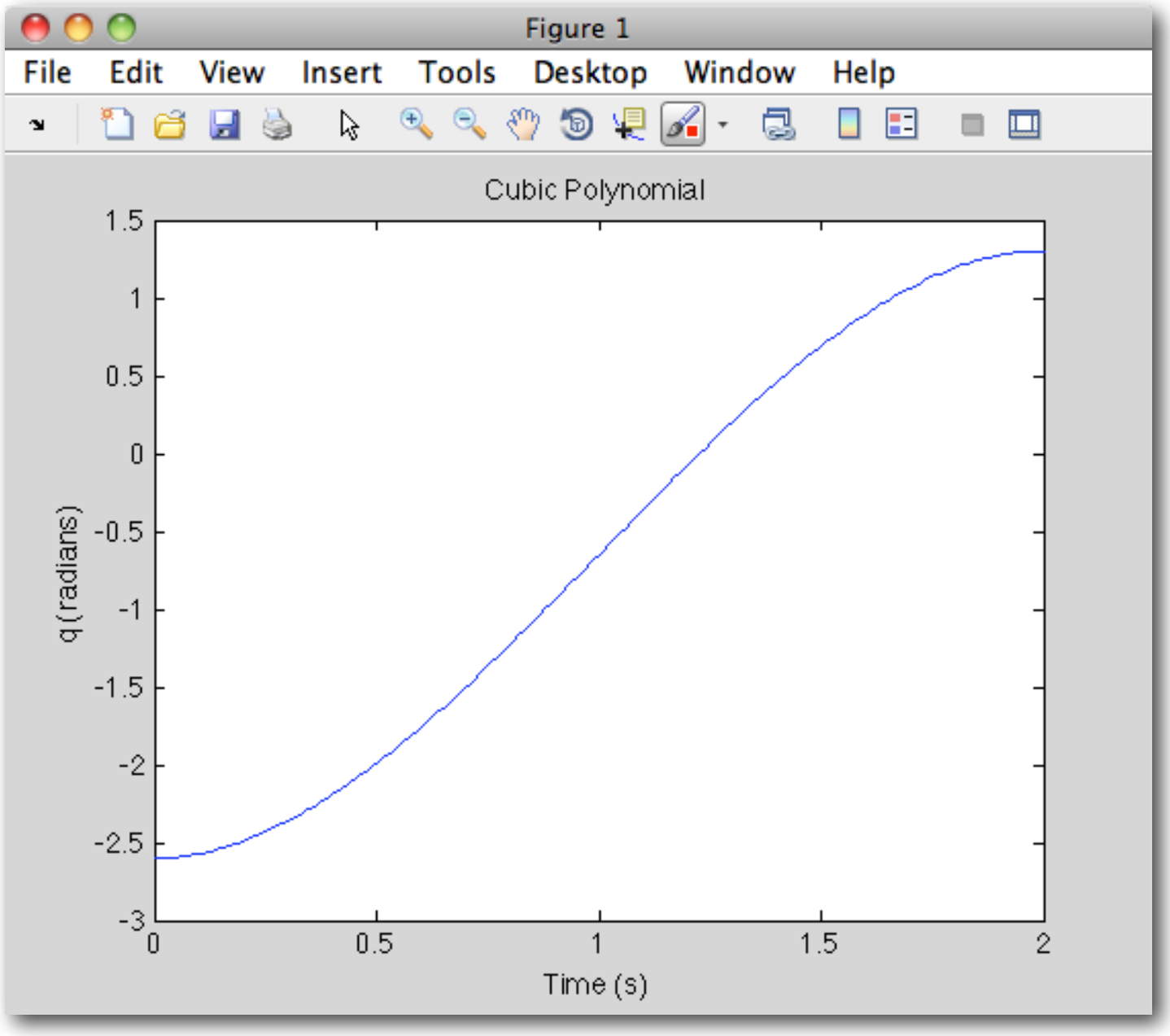


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.5
1
1.5
2

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/10 trajectories/make_cubic_polynomial.m

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m

```
1     %% Clear the workspace.
2 -   clear
3
4
5     %% Define the problem.
6
7     % Define initial and final times.
8 -   t0 = 0; % s
9 -   tf = 2; % s
10
11    % Define initial conditions.
12 -  q0 = -2.6; % radians
13 -  v0 = 0;    % rad/s
14
15    % Define final conditions.
16 -  qf = 1.3;  % radians
17 -  vf = 1;    % rad/s
18
19
20    %% Solve for the cubic polynomial coefficients that meet these conditions.
21
22    % Put initial and final conditions into a column vector.
23 -  conditions = [q0 v0 qf vf]';
24
25    % Put time elements into matrix.
26 -  mat = [1    t0   t0^2     t0^3;
27           0     1   2*t0   3*t0^2;
28           1    tf   tf^2     tf^3;
29           0     1   2*tf   3*tf^2];
30
31    % Solve for coefficients.
32 -  coeffs = mat \ conditions;
33
34    % Pull individual coefficients out.
35 -  a0 = coeffs(1);
36 -  a1 = coeffs(2);
37 -  a2 = coeffs(3);
38 -  a3 = coeffs(4);
```

script | Ln 17 Col 19

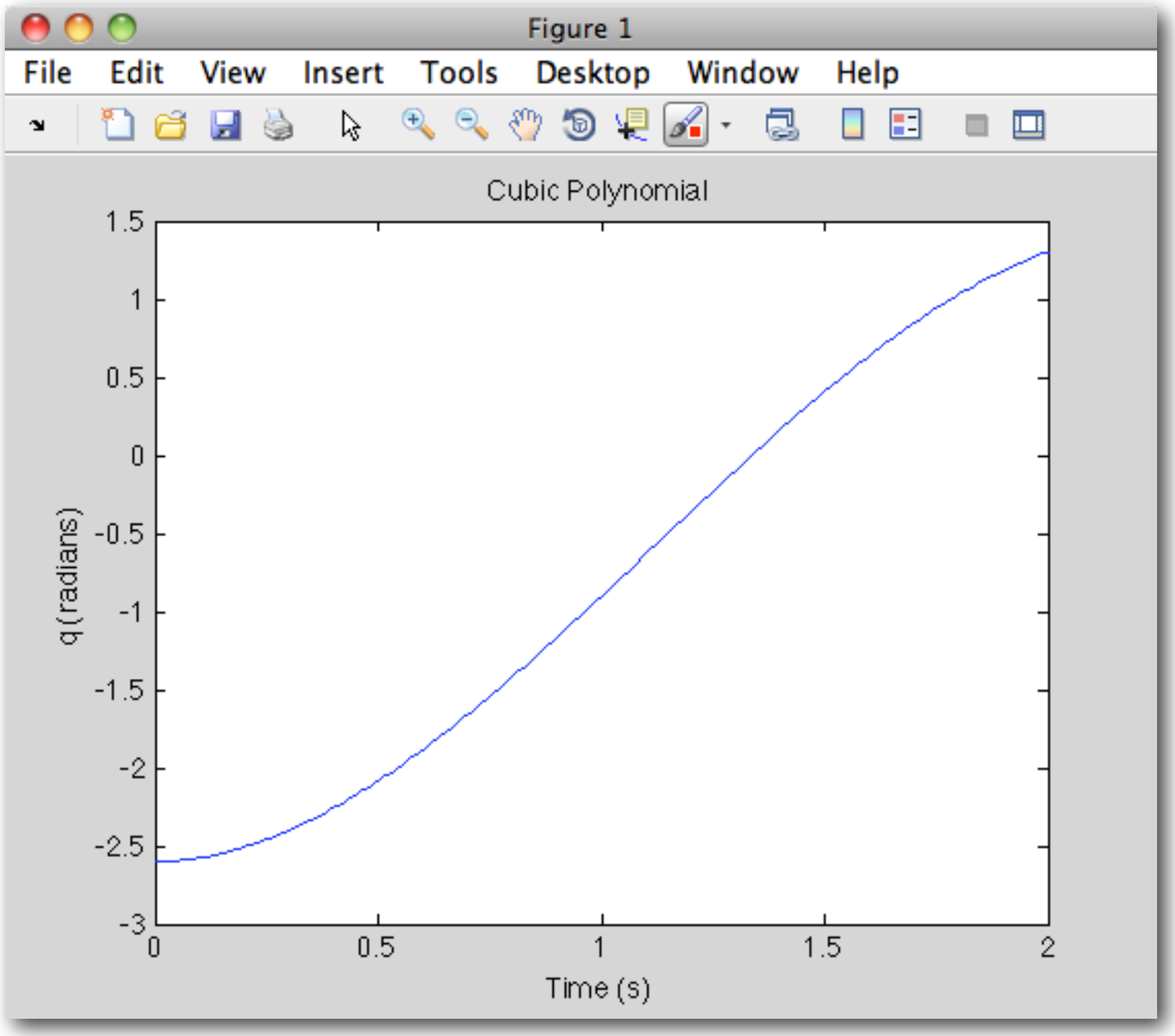


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
q (radians)
0
0.5
1
1.5
2
Time (s)

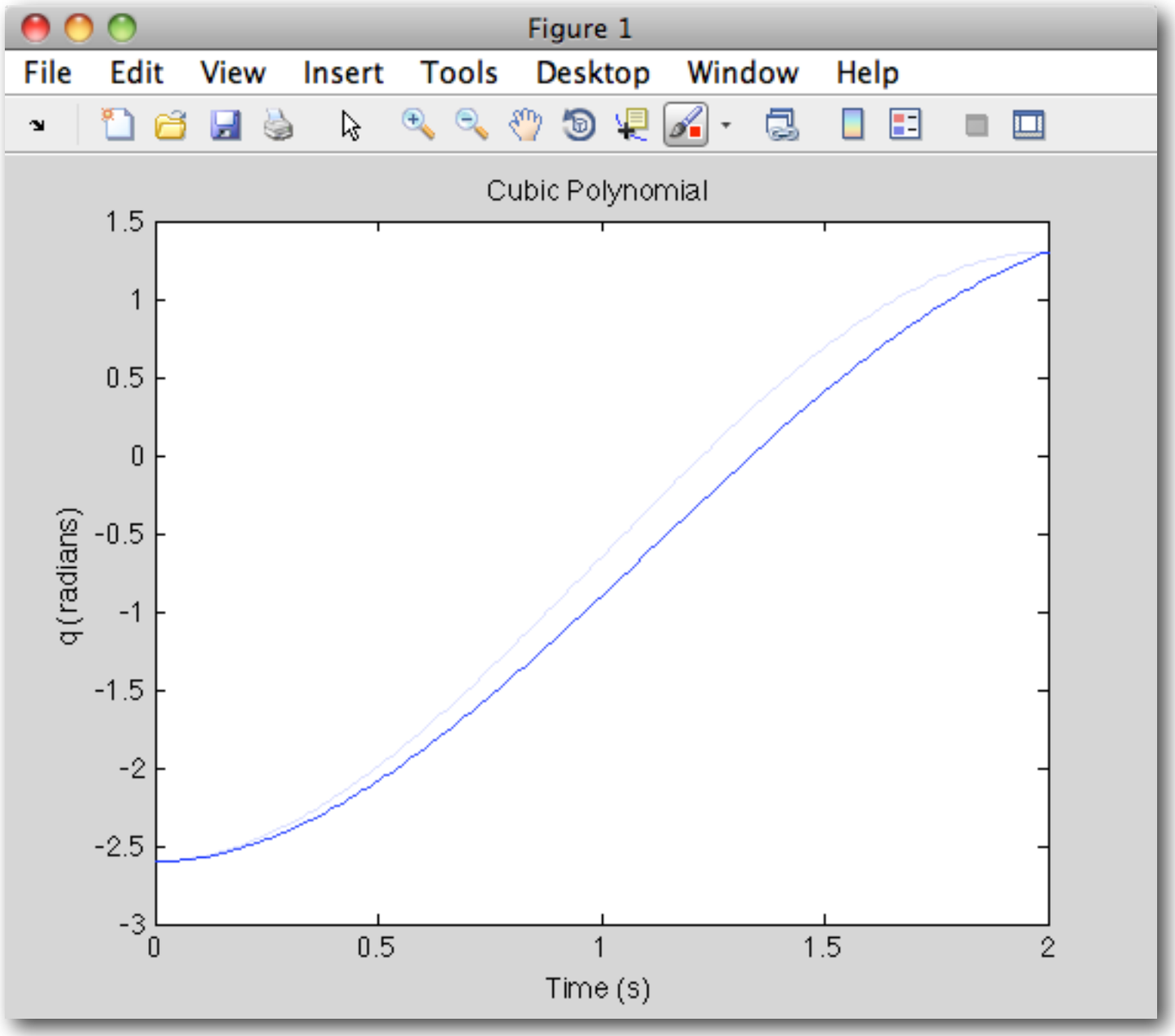


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
q (radians)
0
0.5
1
1.5
2
Time (s)

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/10 trajectories/make_cubic_polynomial.m

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m

```
60
61      %% Plot the cubic polynomial and its derivatives in another figure.
62
63 -    showDerivativesPlot = true;
64 -    if (showDerivativesPlot)
65
66          % Open figure 2.
67 -        figure(2)
68 -        clf
69
70          % Plot cubic trajectory in first subplot.
71 -        subplot(3,1,1)
72 -        set(gca,'fontsize',14)
73 -        plot(t,q,'b')
74 -        xlabel('Time (s)')
75 -        ylabel('q (radians)')
76
77          % Calculate first time-derivative of cubic trajectory with coefficients.
78 -        qdot = a1 + 2*a2*t + 3*a3*t.^2;
79
80          % Plot time derivative of cubic trajectory in second subplot.
81 -        subplot(3,1,2)
82 -        set(gca,'fontsize',14)
83 -        plot(t,qdot,'m')
84 -        xlabel('Time (s)')
85 -        ylabel('v (radians/s)')
86
87          % Calculate second time-derivative of cubic trajectory with coefficients.
88 -        qdoubledot = 2*a2 + 6*a3*t;
89
90          % Plot second time derivative of cubic trajectory in third subplot.
91 -        subplot(3,1,3)
92 -        set(gca,'fontsize',14)
93 -        plot(t,qdoubledot,'r')
94 -        xlabel('Time (s)')
95 -        ylabel('a (radians/s^2)')
96
97 -    end
```

script | Ln 70 Col 32

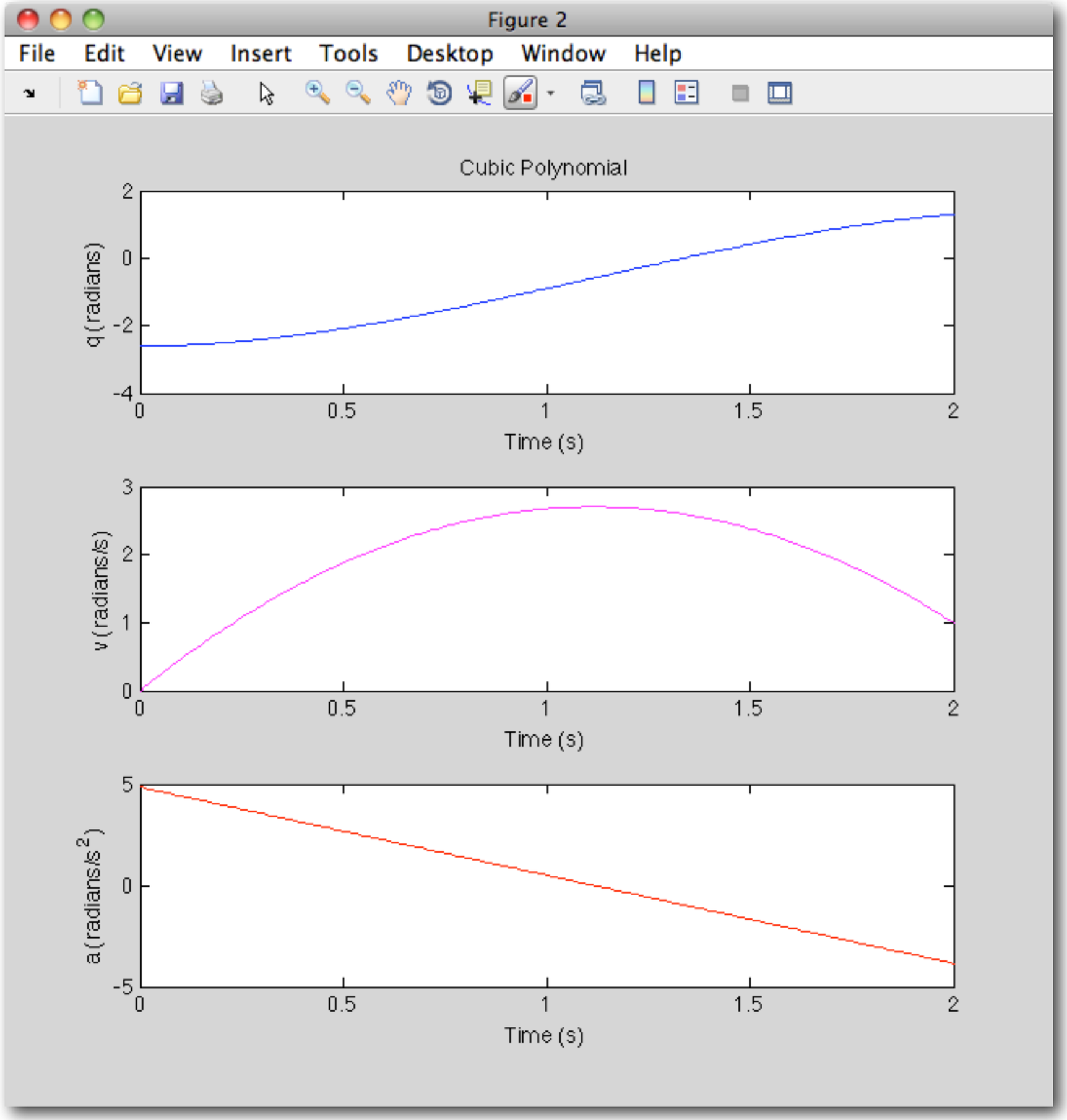
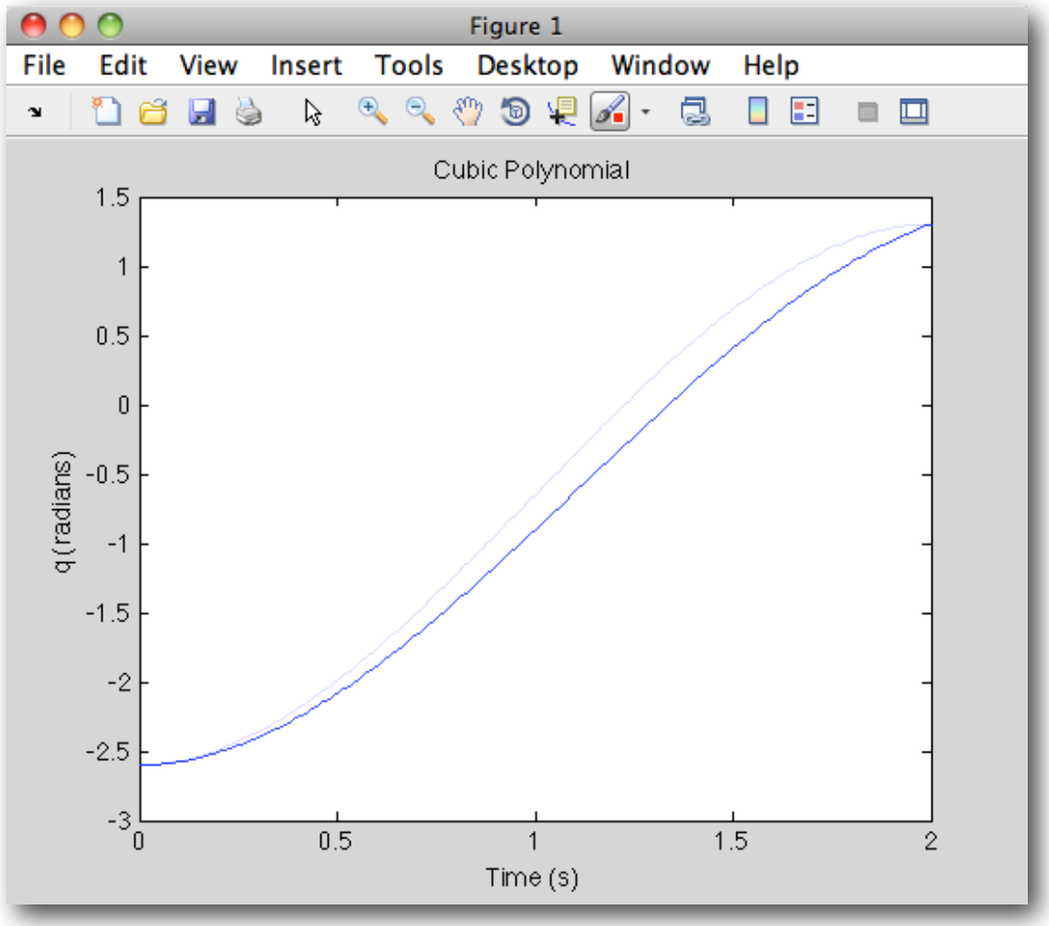


Figure 2
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
2
0
-2
-4
0
0.5
1
1.5
2
Time (s)
v (radians/s)
3
2
1
0
0
0.5
1
1.5
2
Time (s)
a (radians/s^2)
5
0
-5
0
0.5
1
1.5
2
Time (s)

What questions do you have ?



start

$$q(t_0) = q_0 \longrightarrow q(t_f) = q_f$$

$$\dot{q}(t_0) = v_0 \longrightarrow \dot{q}(t_f) = v_f$$

end

cubic polynomial

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

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

Reading Assignment

Within Chapter 5: Path and Trajectory Planning

- Finish reading Section 5.5, which covers Trajectory Planning (pages 186–198)

Deadline: Thursday lecture