MEAM 520

Trajectory Planning

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 10: October 1, 2013

Homework 4:
Forward Kinematics and DH Parameters

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

September 19, 2013

This paper-based assignment is due on **Thursday, September 26, 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 bin or under the door. Late submissions will be accepted until Sunday, September 29, 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 should be your own work, not copied from any other individual or a solution manual. 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!

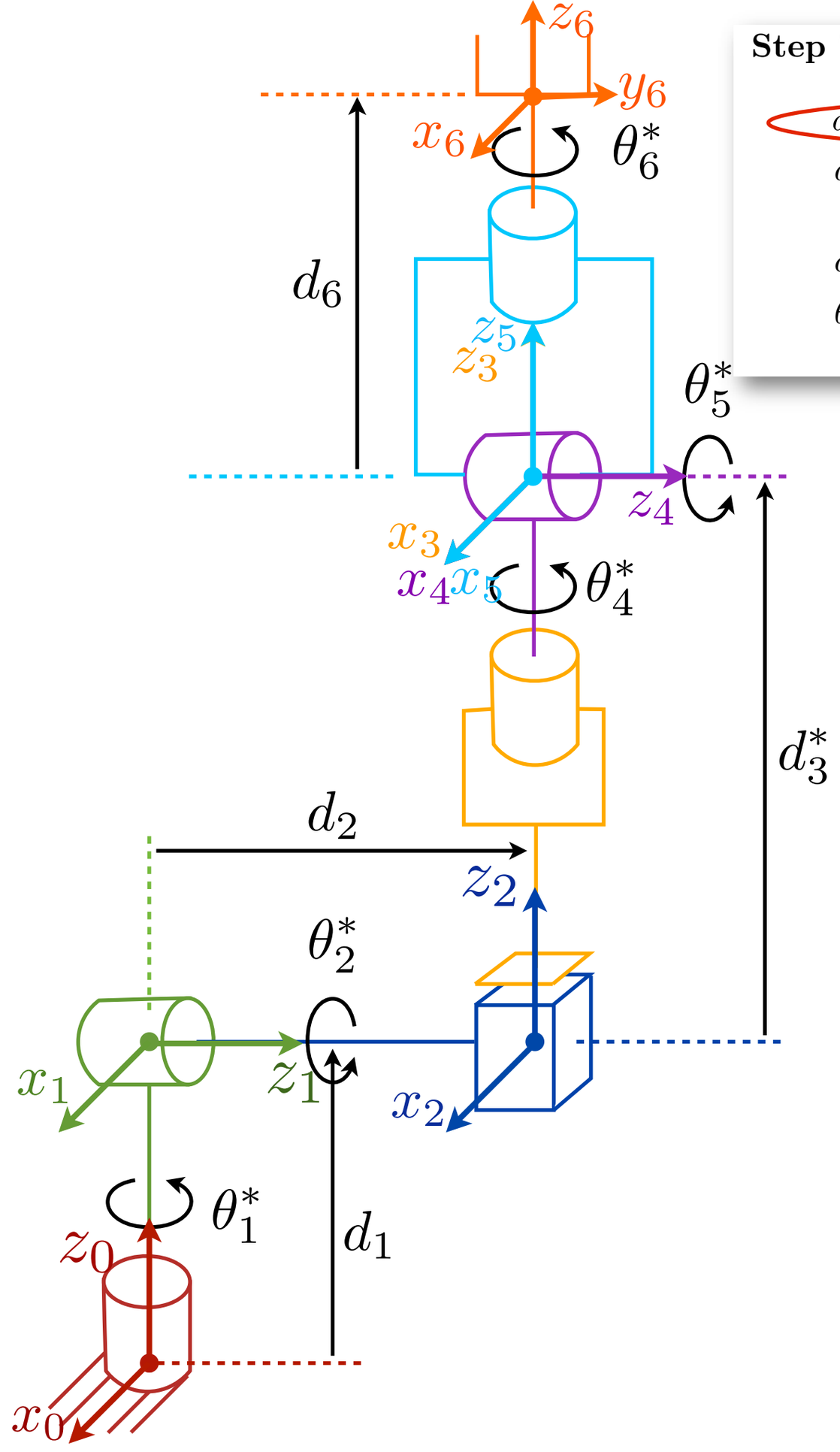
These problems are adapted from the printed version of the textbook, *Robot Modeling and Control* by Spong, Hutchinson, and Vidyasagar (SHV). Please follow the extra clarifications and instructions when provided. Write in pencil, show your work clearly, box your answers, and staple together all pages of your assignment. This assignment is worth a total of 20 points.

1. Custom problem – Kinematics of Baxter *(2 points)*
   Rethink Robotics sells a two-armed manufacturing robot named Baxter. Watch YouTube videos of Baxter (e.g., http://www.youtube.com/watch?v=rjPFqkFyrOY) to learn about its kinematics. Draw a schematic of the serial kinematic chain of Baxter's left arm (the one the woman is touching in the picture below.) Use the book's conventions for how to draw revolute and prismatic joints in 3D.

Homework 4 is being graded.

Common mistakes:

- Drawing Baxter incorrectly.
- Wrong sign on the DH parameters **alpha** and **a**.
- Not marking full extent of dimensions on diagrams.



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.

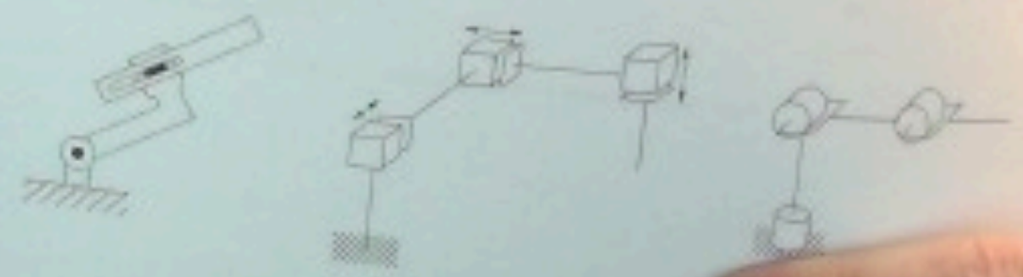
The right way to label variables on your diagram:

Precisely indicate start and end of measurement.

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

Solutions to Homework 4
Forward Kinematics and DH Parameters

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



hw04-
solutions.pdf

MEAM 520

https://piazza.com/class/hf935b0sz1m5r3?cid=86

piazza | MEAM 520 | Q & A | Course Page | Manage Class | Piazza Careers beta | Katherine J. Kuchenbecker

hw1 hw2 hw3 hw4 hw5 lecture5 lecture6 lecture7 lecture8 lecture9 other office_hours textbook matlab

note

stop following 56 views

How to Request a Regrade on Homework

Dear students,

With a class this large (95 students), it's inevitable that we will make some mistakes when grading your assignments and recording the grades in Canvas. I promise to follow through on any concerns you have and ensure that the grading is fair.

If you believe we have made a mistake in grading your assignment, or if you think that the number of points we took off for a certain mistake is too high, here is what you should do:

For a paper assignment, write out an explanation of your inquiry, attach it to the front of your graded assignment, and give it to Naomi during lecture or office hours. Do this within two weeks of the grade being posted on Canvas.

For a programming assignment, send an email to meam520@seas.upenn.edu to explain your concern. You do not need to resubmit your assignment as we already have it on file. Do this within two weeks of the grade being posted on Canvas.

Please do not contact any of the graders directly about your grades. They are not authorized to make changes unless you submit your request through the official procedures I've outlined above. We need to follow this procedure to be sure the system is fair to everyone in the class. I hope there will be only rare occasions when we make mistakes in grading, but I assure you we will handle them all appropriately.

Please let me know what questions and concerns you may have.

Questions?

hw1 hw2 hw3 hw4 hw5

edit · good note | 0

1 day ago by Katherine J. Kuchenbecker

Average Response Time: 10 min

Special Mentions: Katherine J. Kuchenbecker answered Zero Config for PUMA in 7 min. 1 day ago

Online Now | This Week: 14 | 105

I usually keep my office door open while I am working because I like seeing people.

But, I am often busy with other work; I hold office hours only at specified times, not all the time.

Please *ask* if I have time to talk with you; I apologize in advance if I am too busy.

Go to OH. Post on Piazza.

MEAM 520 | Class Profile | Piazza

https://piazza.com/upenn/fall2013/meam520/staff

Google

piazza | MEAM 520 | Q & A | Course Page | Manage Class | Katherine J. Kuchenbecker

University of Pennsylvania - Fall 2013

MEAM 520: Introduction to Robotics

+ Add Syllabus

Course Information | Staff | Resources

Edit

	Name	Office Hours
	Katherine J. Kuchenbecker	**When?** Tuesday 1:30 - 2:30 p.m. and Thursday 3:00 - 4:00 p.m. **Where?** Towne 224
	Naomi Fitter	**When?** Monday 4:30 - 5:30 p.m. and Wednesday 6:00 - 7:00 p.m. **Where?** Towne 205
	Tyler Barkin	**When?** **Where?**
	Yunkai Cui	**When?** **Where?**
	Samarth Manoj Brahmbhatt	**When?** Tuesday 3:00 - 4:15 p.m. **Where?** Towne 205
	Chaitanya Bhargava	**When?** Friday 2:00 - 3:00 p.m. **Where?** Towne 195
	Annie Mroz	**When?** Monday 5:30 - 6:30 p.m. **Where?** Towne 205

piazza

Share this

Piazza Report for:

MEAM 520

In total, students asked:
45 Questions

Either students, instructors, or both responded to
100% of Questions

Counting all posts, responses, edits, followups, and comments, there were:
237 Contributions

The average response time was:
10 Minutes

Top Contributors

Mabel
14 contributions

Marcus
9 contributions

Gaylord
6 contributions

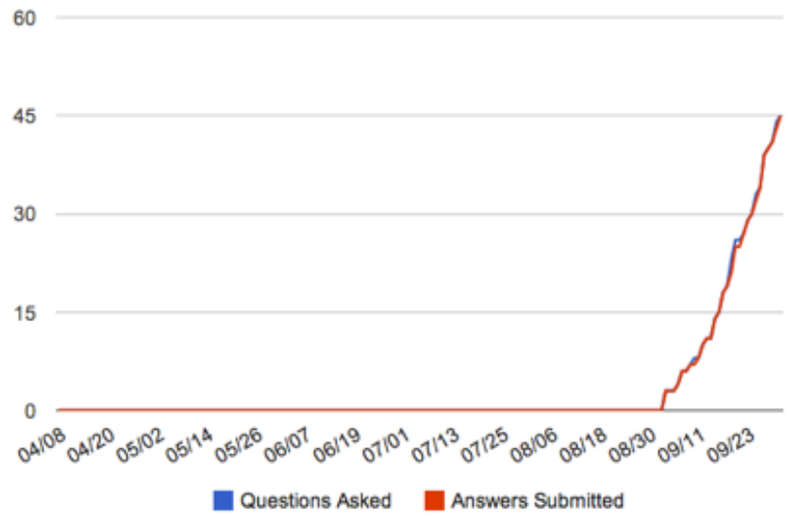
Siyao
6 contributions

Jianqiao
5 contributions

107 students were enrolled...

...and **32%** of them made at least one contribution (**34** in total).

Questions & Answers



100% of questions received instructors' responses (**45** in total).

18% of questions received students' responses (**8** in total).

And **50%** of those were endorsed by an instructor (**4** in total)!

Top Content

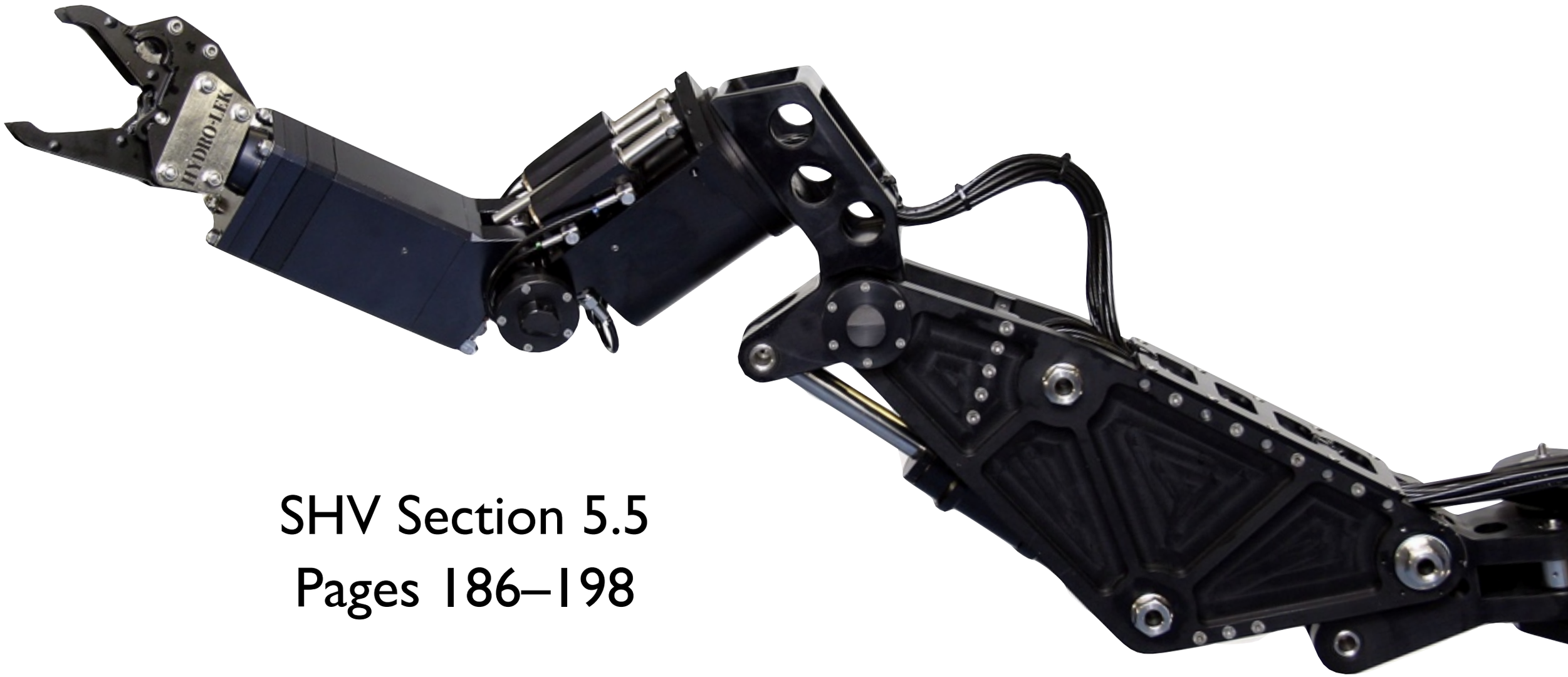
70 views
Doubt on Question 3
rotations.jpg Hi, I had a doubt with respect to rotating ...
1 Answerers
2 Followup Discussions
4 Followup Replies

79 views
TA Sessions for Homework 1
Hi everyone, I ll be holding TA hour tomorrow (Friday, ...
0 Answerers
2 Followup Discussions
0 Followup Replies

84 views
Area for red markers
It says that we have to plot red markers where ...
3 Answerers
1 Followup Discussions
2 Followup Replies

www.piazza.com

Trajectory Planning



SHV Section 5.5
Pages 186–198

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

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.

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

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

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

Simplest Situation: Specifying Joint Value Only

Initial Condition

$$q(t_0) = q_0$$

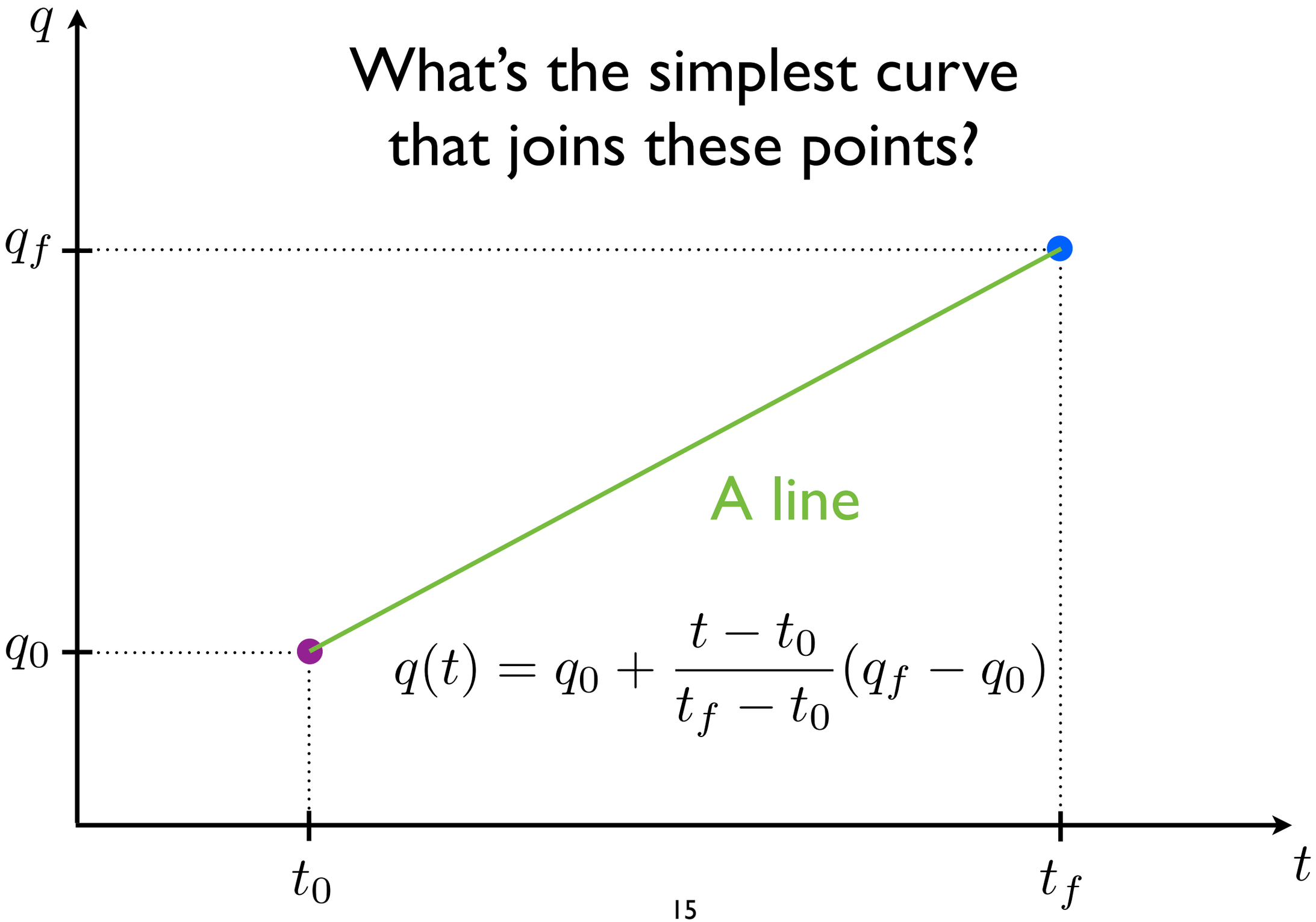
Final Condition

$$q(t_f) = q_f$$

Simplest Situation: Specifying Joint Value Only

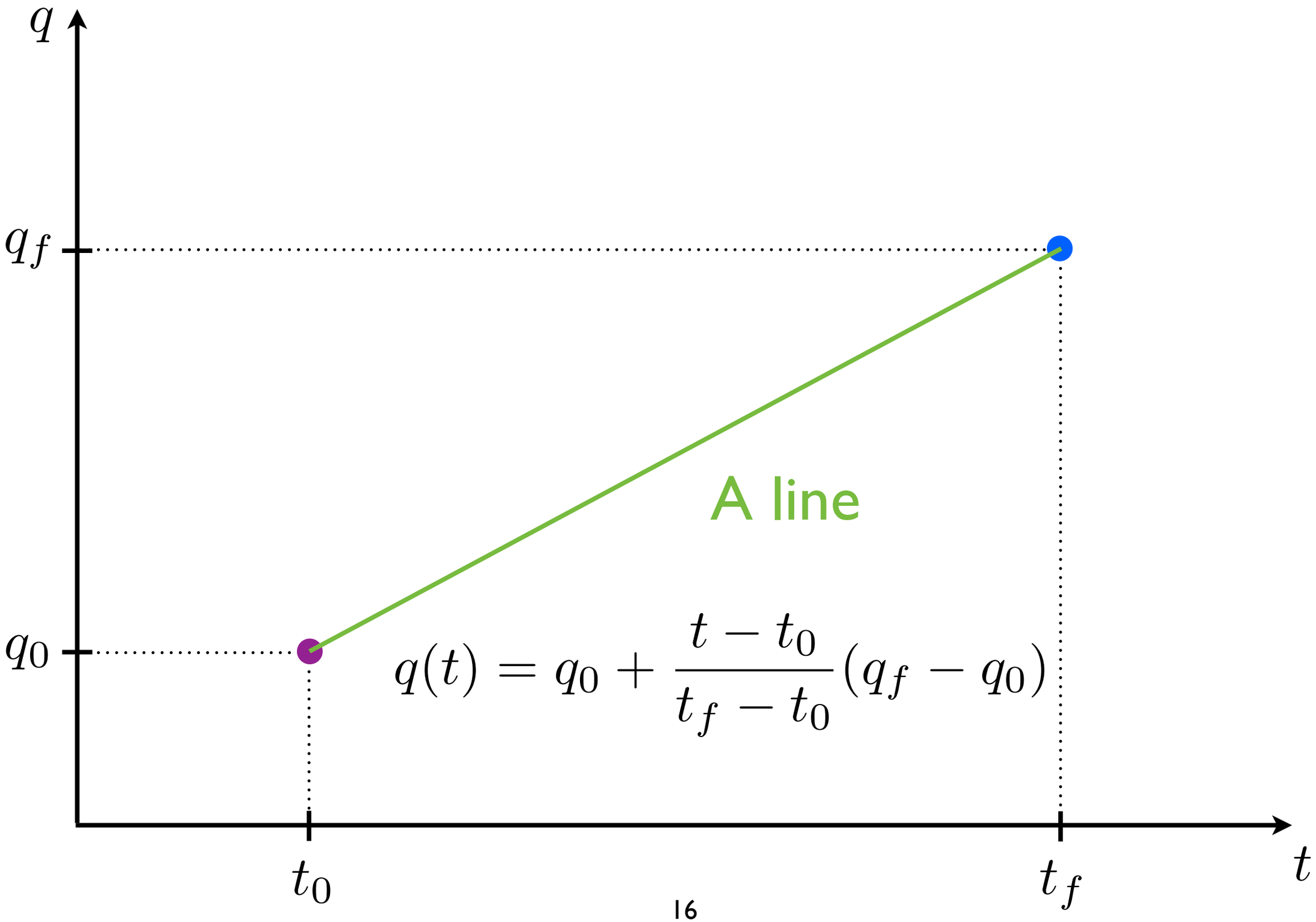
$$q(t_0) = q_0 \qquad\qquad q(t_f) = q_f$$

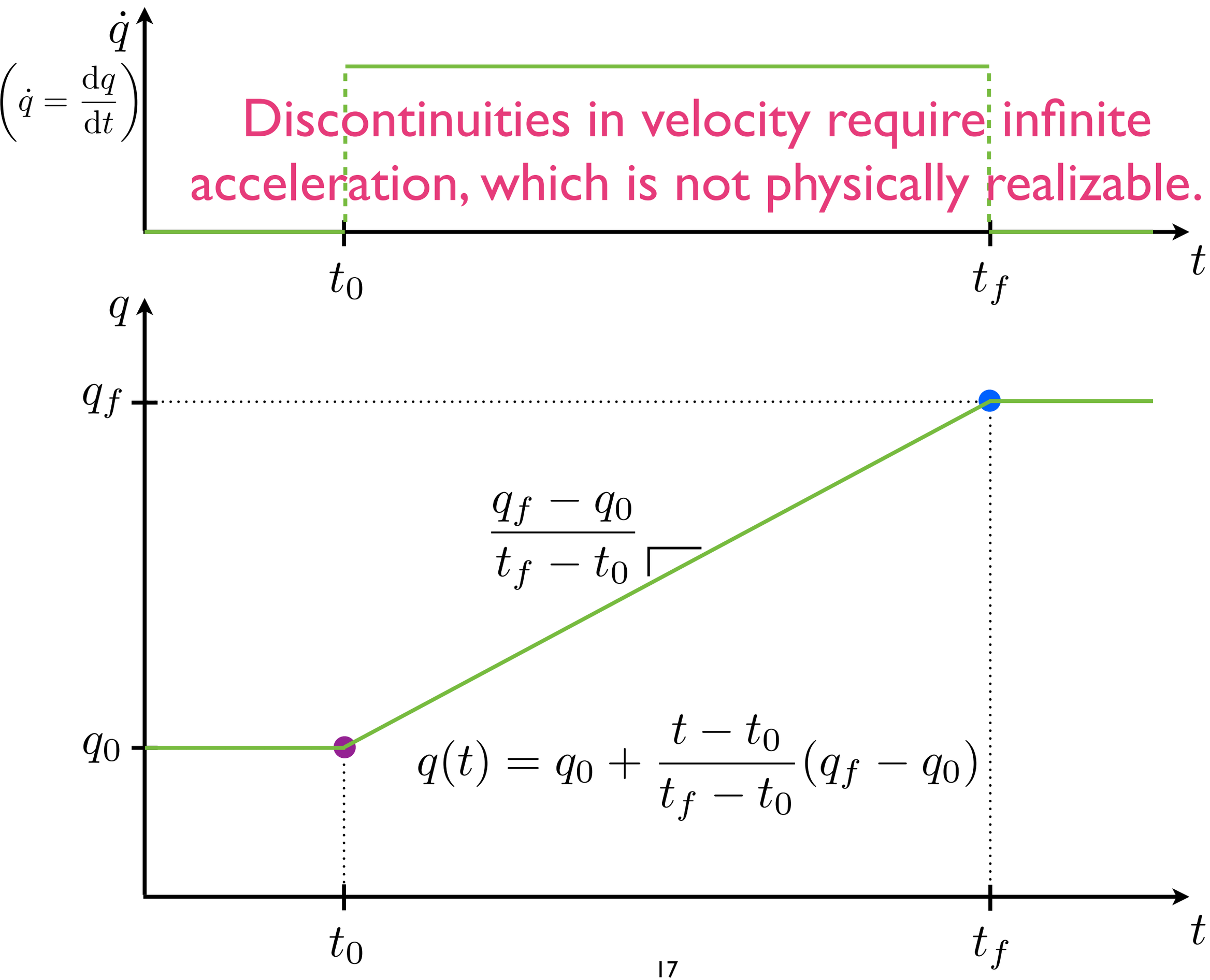
What's the simplest curve that joins these points?



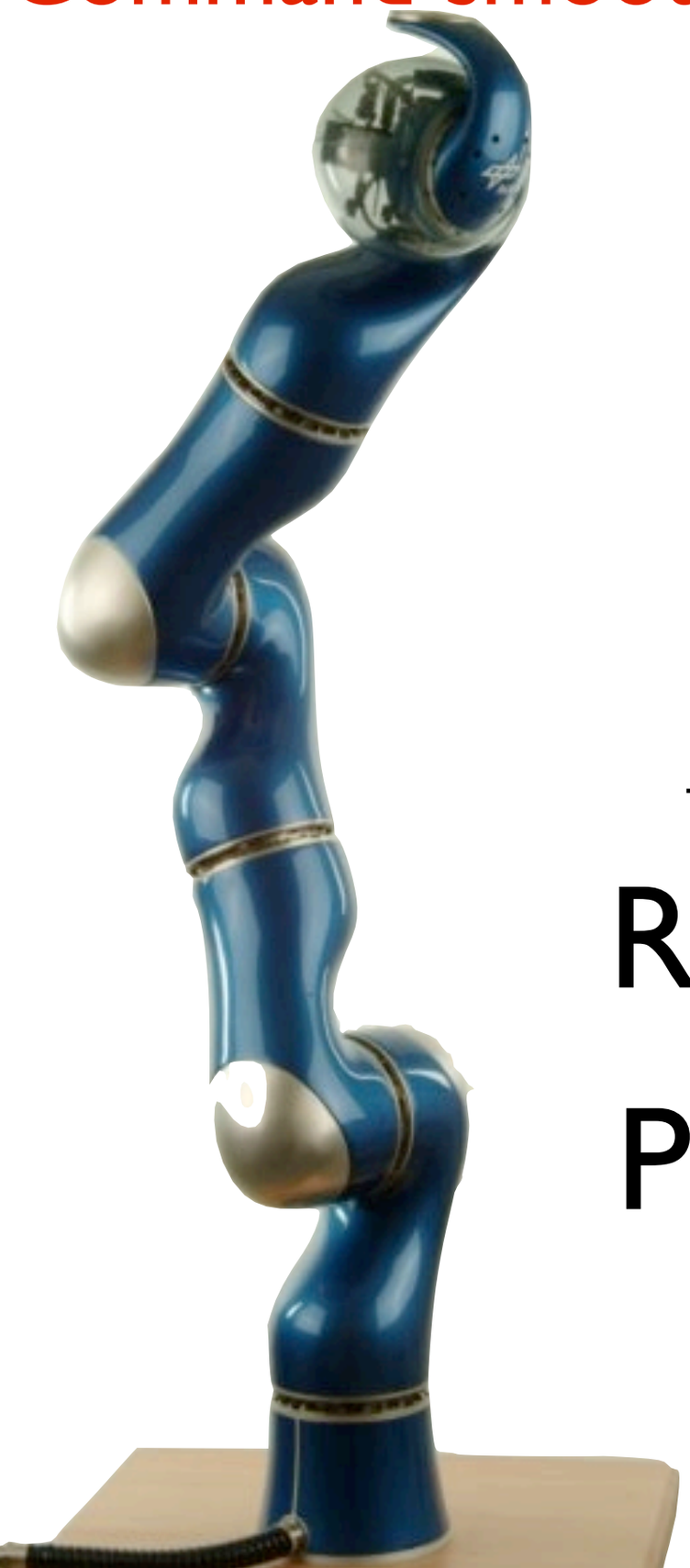
Linear interpolation is really useful!

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.

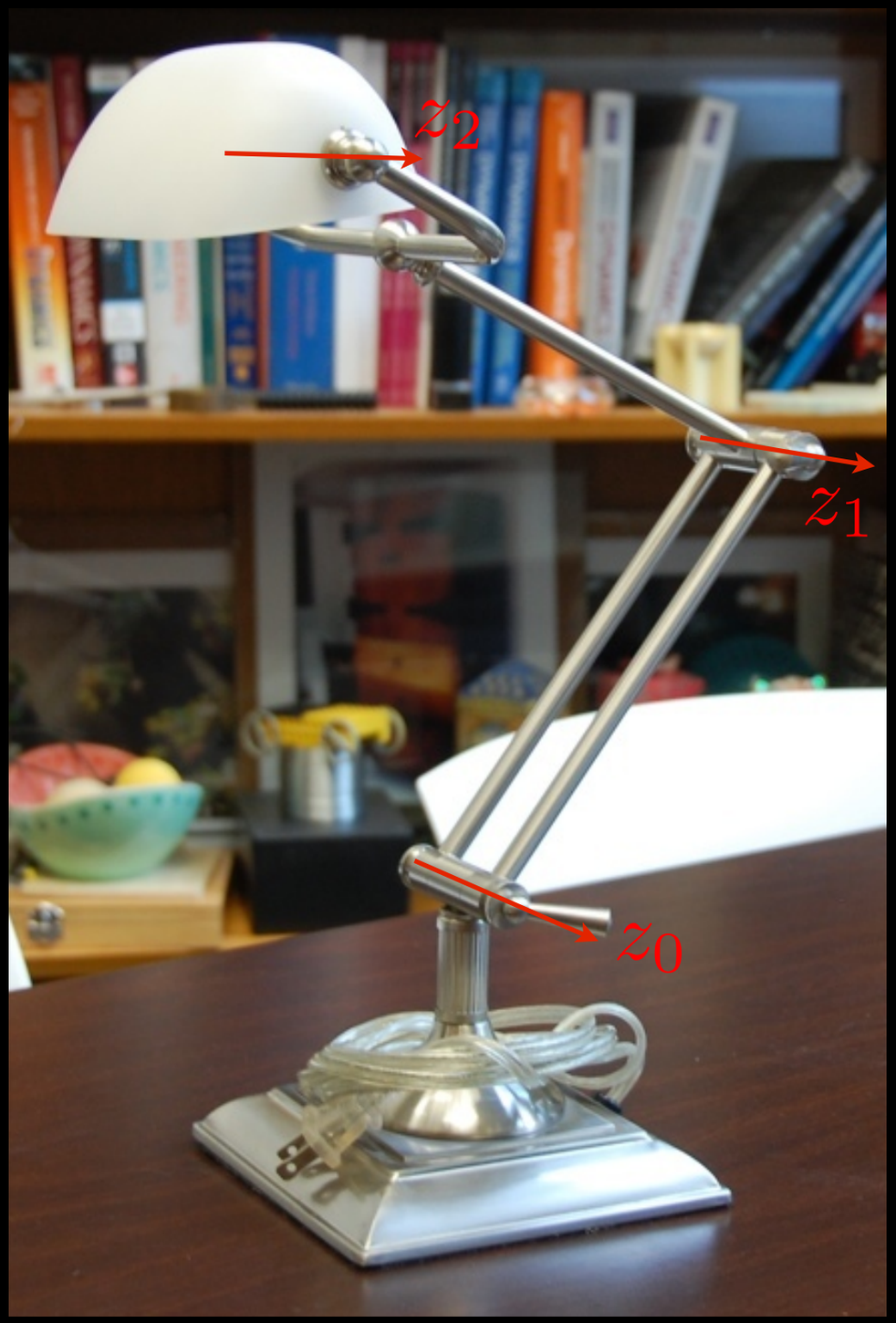
Robot manipulators are composed of:

- 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

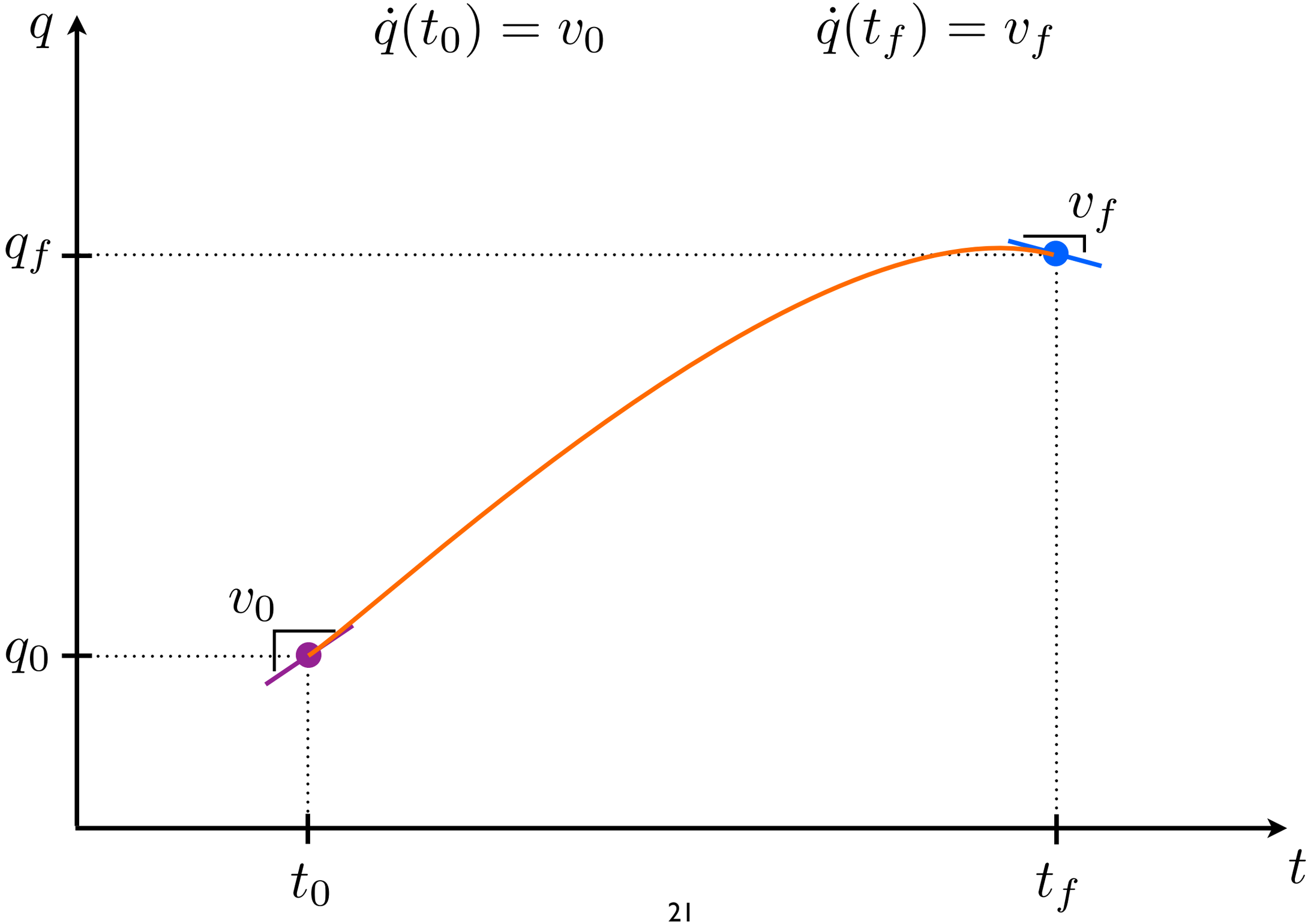
Final Conditions

$$q(t_f) = q_f$$

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

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

```
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 = 0;    % 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 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

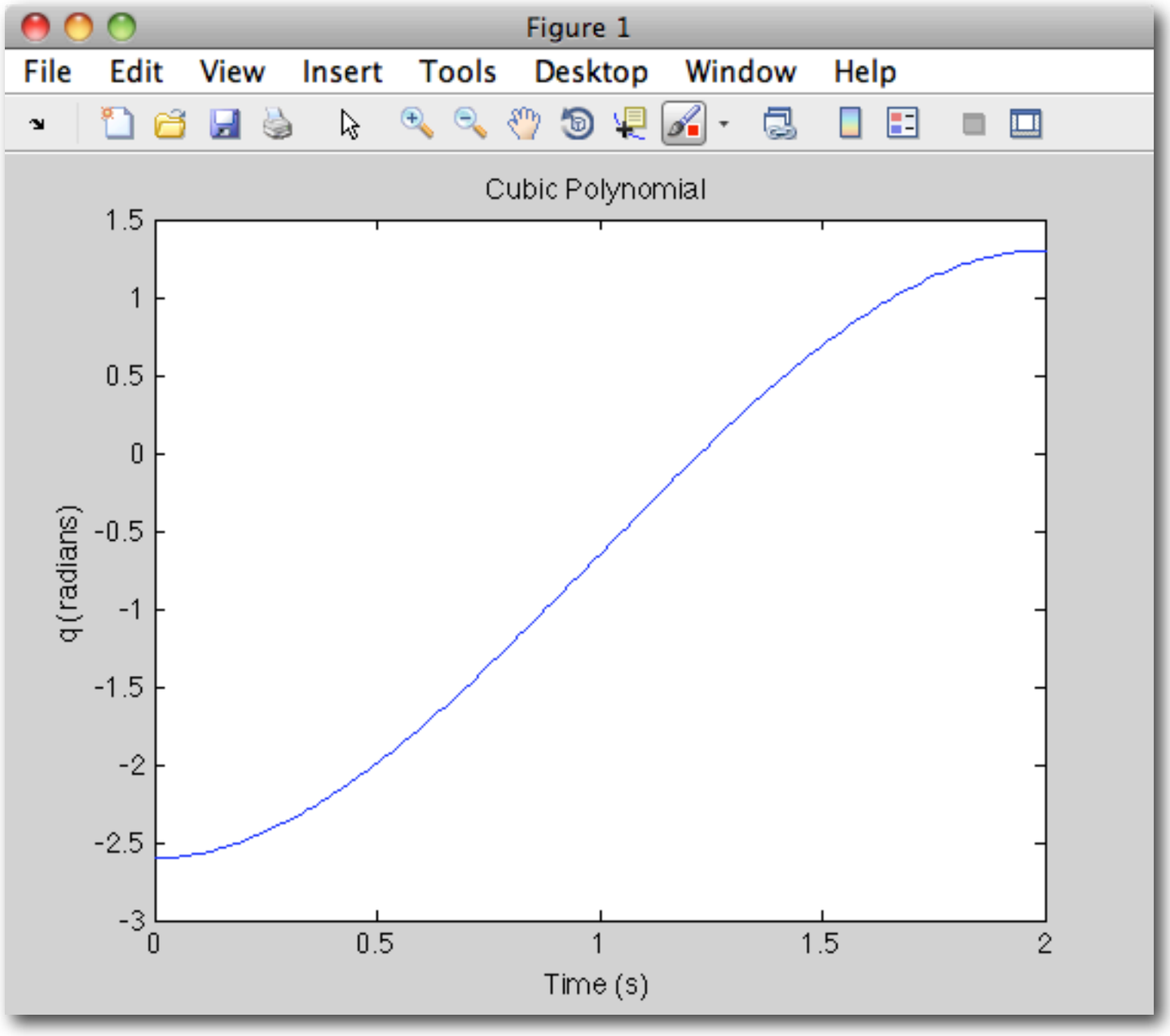


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.5
1
1.5
2

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

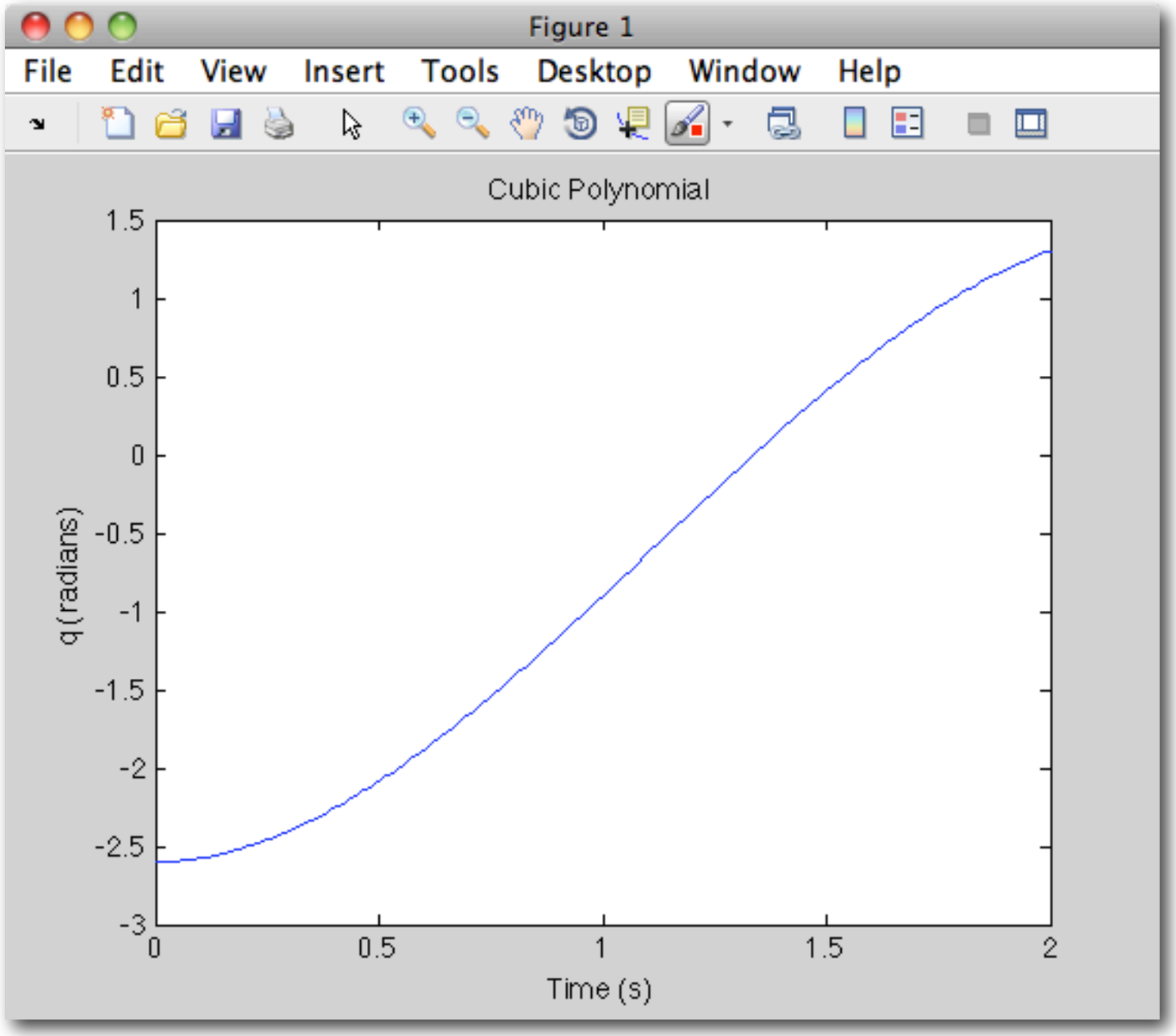


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.5
1
1.5
2

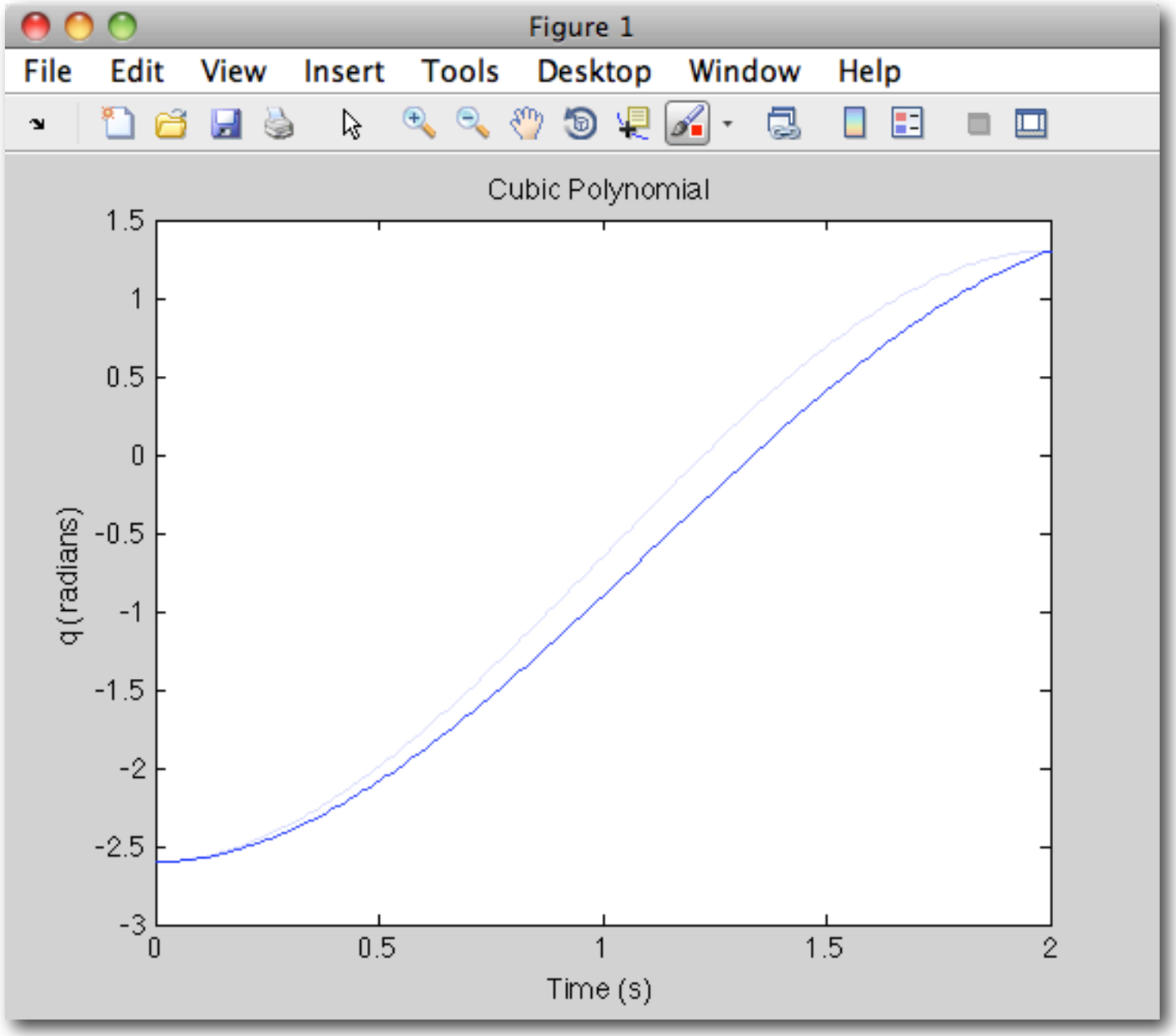


Figure 1
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.5
1
1.5
2

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

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m

```
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

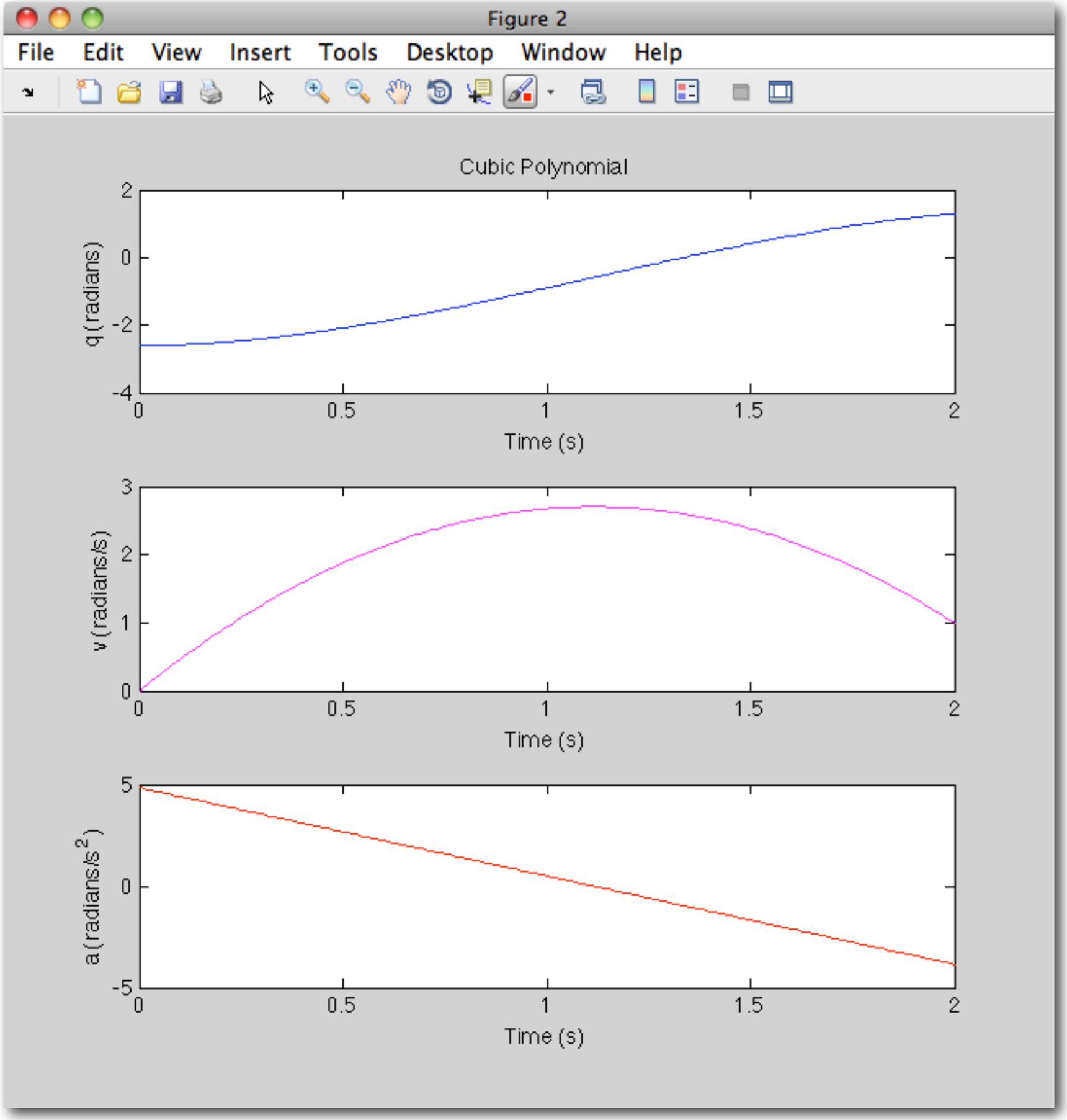
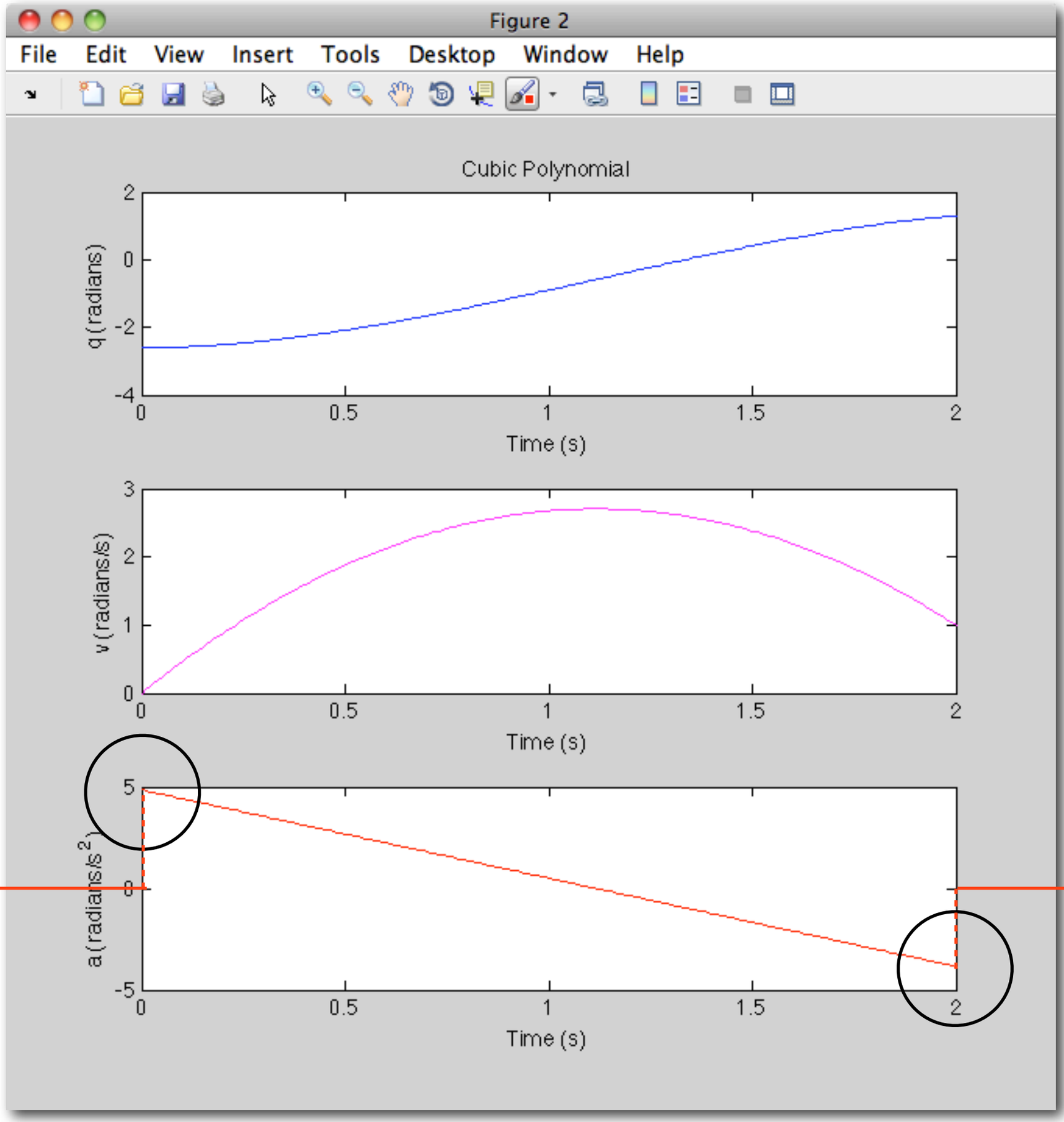


Figure 2
File Edit View Insert Tools Desktop Window Help
Cubic Polynomial
q (radians)
Time (s)
v (radians/s)
Time (s)
a (radians/s^2)
Time (s)

What questions do you have ?



Discontinuities in acceleration require **step changes in force/torque**, which excites vibrational modes in the robot.

Time derivative of acceleration is **jerk**.

We don't want **infinite jerk**.

Specifying Joint Values Plus First and Second 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

$$\ddot{q}(t_0) = \alpha_0$$ **Units** angle per time per time or distance per time per time, e.g., rad/s^2 or m/s^2

Not the same α as in DH!

Final Conditions

$$q(t_f) = q_f$$

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

$$\ddot{q}(t_f) = \alpha_f$$

What kind of curve to use?

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

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

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

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

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

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m | make_quintic_polynomial.m

```
%% 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
alpha0 = 0; % rad/s^2

% Define final conditions.
qf = 1.3;   % radians
vf = 1;     % rad/s
alphaf = 0; % rad/s^2


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

% Put initial and final conditions into a column vector.
conditions = [q0 v0 alpha0 qf vf alphaf]';

% Put time elements into matrix.
mat = [1    t0   t0^2     t0^3    t0^4    t0^5;
       0     1   2*t0   3*t0^2  4*t0^3  5*t0^4;
       0     0      2     6*t0 12*t0^2 20*t0^3;
       1    tf   tf^2     tf^3    tf^4    tf^5;
       0     1   2*tf   3*tf^2  4*tf^3  5*tf^4;
       0     0      2     6*tf 12*tf^2 20*tf^3];

% Solve for coefficients.
coeffs = mat \ conditions;

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

script | Ln 16 Col 27

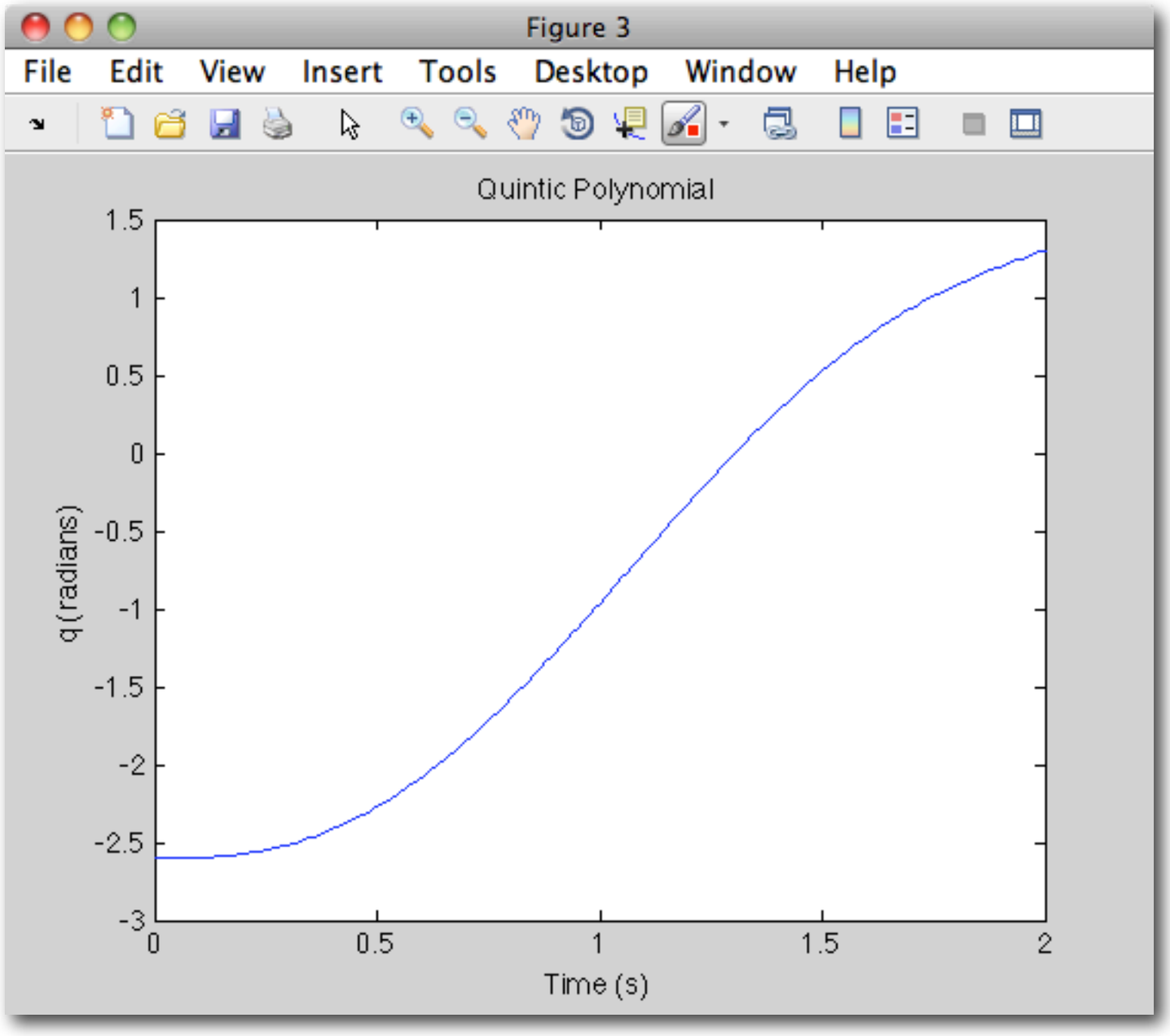


Figure 3
File Edit View Insert Tools Desktop Window Help
Quintic Polynomial
q (radians)
Time (s)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
0.5
1
1.5
2

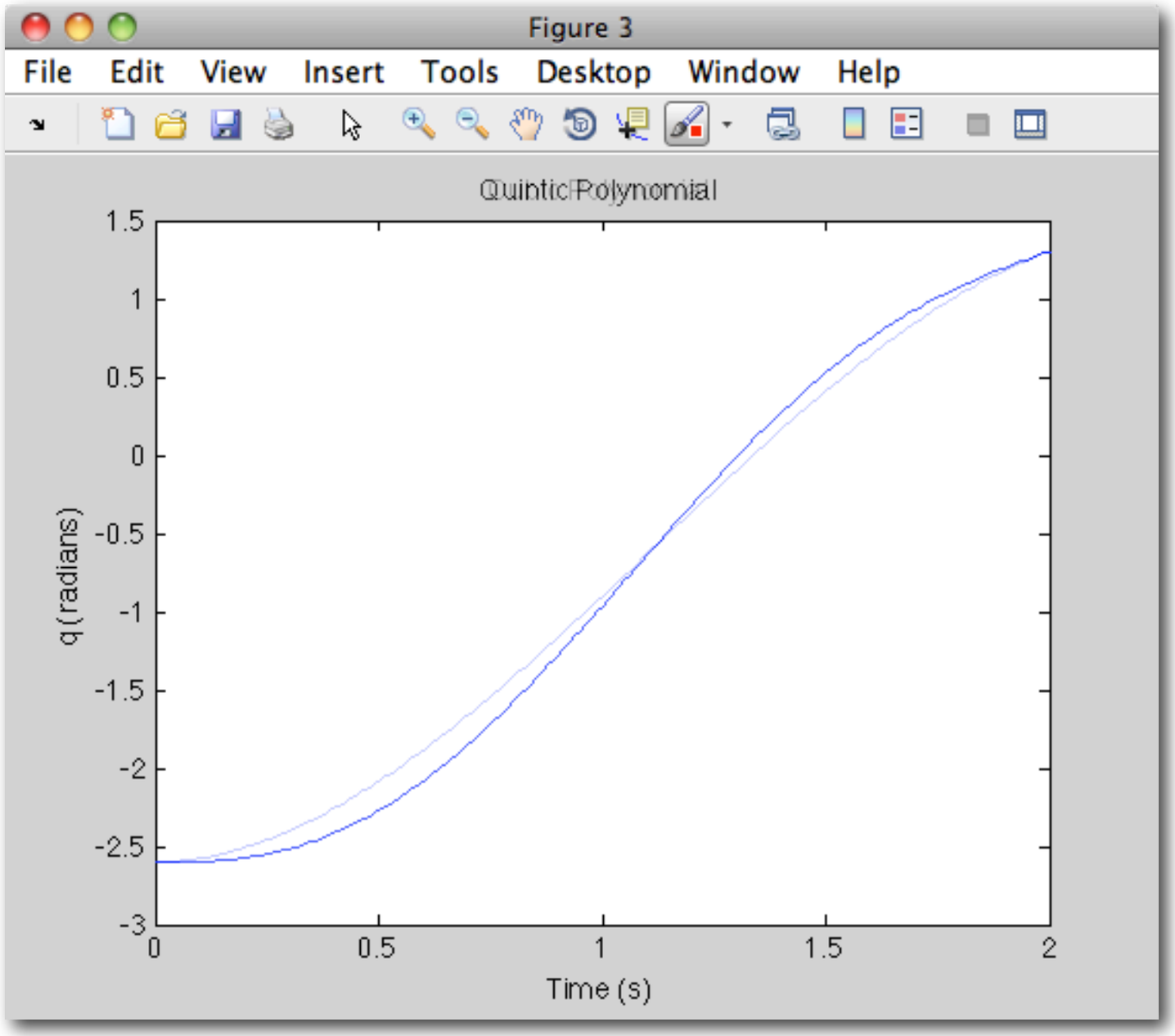


Figure 3
File Edit View Insert Tools Desktop Window Help
QuinticPolynomial
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
q (radians)
0
0.5
1
1.5
2
Time (s)

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

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m | make_quintic_polynomial.m

```
68      %% Plot the quintic polynomial and its derivatives in another figure.
69
70 -    showDerivativesPlot = true;
71 -    if (showDerivativesPlot)
72
73          % Open figure 4.
74 -        figure(4)
75 -        clf
76
77          % Plot quintic trajectory in first subplot.
78 -        subplot(3,1,1)
79 -        set(gca,'fontsize',14)
80 -        plot(t,q,'b')
81 -        xlabel('Time (s)')
82 -        ylabel('q (radians)')
83 -        title('Quintic Polynomial')
84
85          % Calculate first time-derivative of quintic trajectory with coefficients.
86 -        qdot = a1 + 2*a2*t + 3*a3*t.^2 + 4*a4*t.^3 + 5*a5*t.^4;
87
88          % Plot time derivative of quintic trajectory in second subplot.
89 -        subplot(3,1,2)
90 -        set(gca,'fontsize',14)
91 -        plot(t,qdot,'m')
92 -        xlabel('Time (s)')
93 -        ylabel('v (radians/s)')
94
95          % Calculate second time-derivative of cubic trajectory with coefficients.
96 -        qdoubledot = 2*a2 + 6*a3*t + 12*a4*t.^2 + 20*a5*t.^3;
97
98          % Plot second time derivative of cubic trajectory in third subplot.
99 -        subplot(3,1,3)
100 -       set(gca,'fontsize',14)
101 -       plot(t,qdoubledot,'r')
102 -       xlabel('Time (s)')
103 -       ylabel('alpha (radians/s^2)')
104
105 -   end
```

script | Ln 71 Col 25

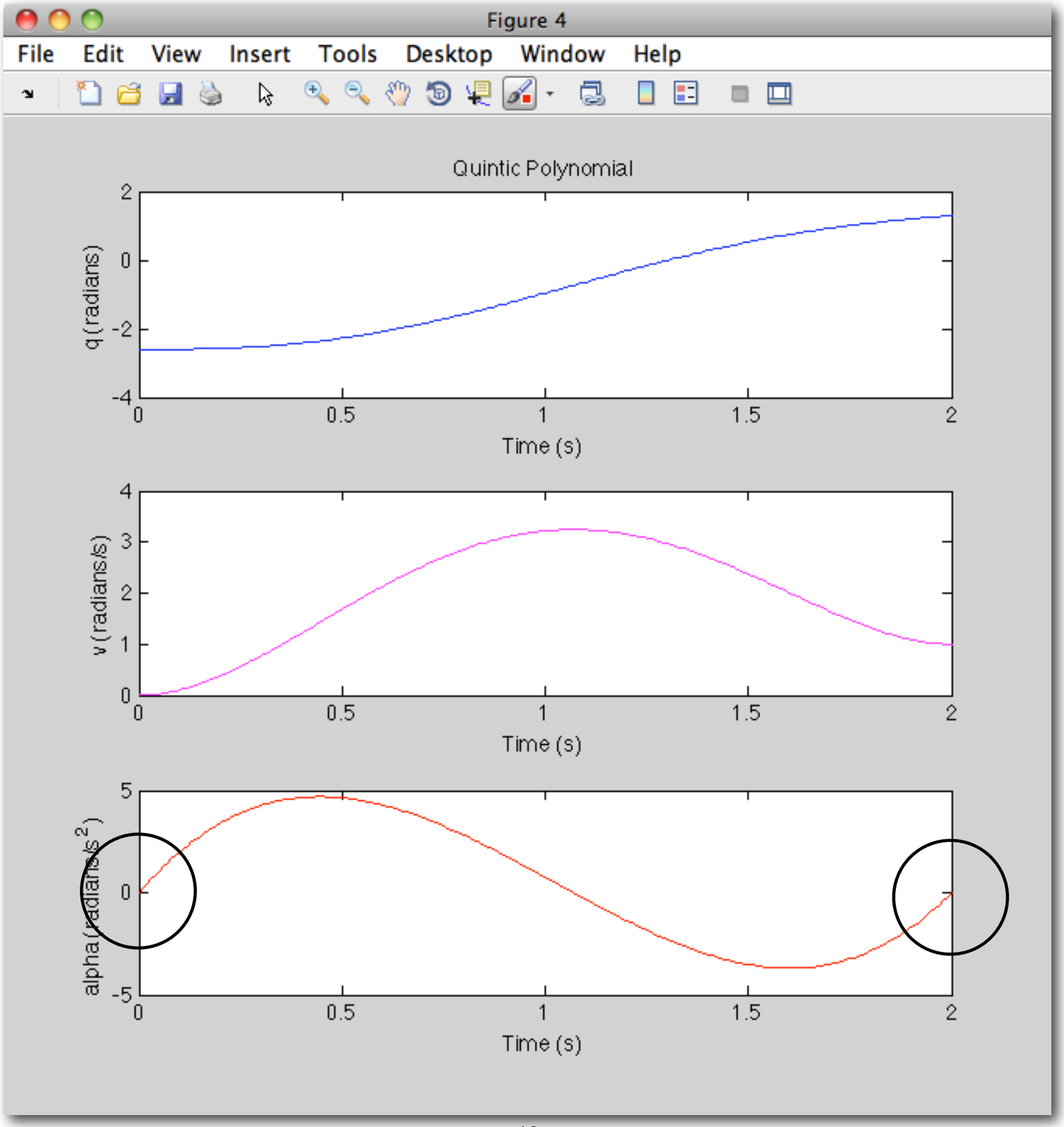


Figure 4
File Edit View Insert Tools Desktop Window Help
Quintic Polynomial
q (radians)
Time (s)
v (radians/s)
Time (s)
alpha (radians/s^2)
Time (s)

What questions do you have ?

0270-6474/85/0509-2318$02.00/0

The Journal of Neuroscience
Vol. 5, No. 9, pp. 2318–2330
September 1985

Kinematic Features of Unrestrained Vertical Arm Movements[1]

CHRISTOPHER G. ATKESON AND JOHN M. HOLLERBACH[2]

Artificial Intelligence Laboratory and Department of Psychology, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Abstract

Unrestrained human arm trajectories between point targets have been investigated using a three-dimensional tracking apparatus, the Selspot system. Movements were executed between different points in a vertical plane under varying conditions of speed and hand-held load. In contrast to past results which emphasized the straightness of hand paths, movement regions were discovered in which the hand paths were curved. All movements, whether curved or straight, showed an invariant tangential velocity profile when normalized for speed and distance. The velocity profile invariance with speed and load is interpreted in terms of simplification of the underlying arm dynamics, extending the results of Hollerbach and Flash (Hollerbach, J. M., and T. Flash (1982) Biol. Cybern. *44:* 67–77).

We have investigated unrestrained human arm trajectories between point targets using a three-dimensional tracking apparatus, the Selspot system. Our studies indicate the importance of examining natural unrestricted movements, as our results agree only in part with previous studies of arm movement. Past observations on multijoint human arm trajectories obtained from restricted horizontal planar movements measured with a gripped pantograph have shown in both humans and monkeys that point-to-point trajectories are essentially straight with bell-shaped velocity profiles (Morasso, 1981; Abend et al., 1982). Moreover, they satisfy a time scaling property that may be related to the underlying dynamics (Hollerbach and Flash, 1982). We sought to corroborate these observations for more natural unrestricted arm movements and also to examine the effects of different loads and of gravity on the arm trajectories. Our research on load effects has also led to the discovery of scaling laws for arm loads.

Path shape. A strategy for gaining insight into planning and control processes of the motor system is to look for kinematic invariances in trajectories of movement. The significance of straight-line movements of the hand during arm trajectories is that they imply movement planning at the hand or object level (Morasso, 1981; Hollerbach, 1982), that is to say, in terms of coordinates or variables that are external to the biological system and that could be matched to tasks or outside constraints.

If movements were planned in terms of joint variables, one would expect curved hand paths. The observed straight-line hand paths would seem to preclude this possibility (Morasso, 1981), yet in a series of papers examining unrestrained vertical arm movement (Soechting and Lacquaniti, 1981; Lacquaniti and Soechting, 1982; Lacquaniti et al., 1982), the hand trajectories were evidently straight at the same time that the joint rate ratio of shoulder and elbow tended toward a constant. This apparently contradictory situation of straight lines in both hand space and joint space has nevertheless been resolved recently in favor of hand space straight lines due to an artifact of two-joint kinematics near the workspace boundary (Hollerbach and Atkeson, 1984).

When hand movements are curved in response to task requirements or to internal control, it is not as clear what the planning variables are. For handwriting movements, Hollerbach (1981) proposed orthogonal task coordinates in the writing plane that yielded cursive script through coupled oscillation and modulation. Viviani and Terzuolo (1982) proposed hand variable planning for drawing as well as writing through proportional control of tangential velocity and radius of curvature. Morasso (1983) examined three-dimensional curved motion and proposed independent control of the curvature and torsion of the hand cartesian coordinates. Again arguing for joint-level planning but also for actuator-level planning, Soechting and Lacquaniti (1983) investigated curved movements resulting from change of target location during two-joint arm movement, and inferred both a linear relation between elbow and shoulder accelerations and stereotypical muscle electromyogram activity.

Time profile. In addition to the path of the arm, the other aspect of a trajectory is the time sequence along the path. This tangential velocity profile may through its shape also give insight into movement planning strategies. For motions under low spatiotemporal accuracy constraints, a common observation is a symmetrical and unimodal velocity profile. Crossman and Goodeve (1983) characterized these profiles as Gaussian for two different single degree of freedom movements: a pen-tapping movement constrained by a measurement wire and wrist rotation about the forearm axis. More recently, Hogan (1984) modeled the velocity profiles for single-joint elbow movement as fourth-order polynomials derived from a minimum-jerk cost function. In examining optimization criteria for single-joint movement, Nelson (1983) deduced that a minimum-jerk velocity profile is almost indistinguishable from simple harmonic motion for repetitive movement. Stein et al. (1985) modeled muscle activation and energetics for a single degree of freedom point-to-point movement, and showed that muscle force rise time or minimum energy yields a velocity profile very similar to minimum jerk.

The previous experiments involved single degree of freedom movement, either with one joint or an apparatus with one degree of freedom, for which the only independent parameter is the time dependence. Nevertheless, similar results have been found for multi-

Received June 26, 1984; Revised February 4, 1985;
Accepted March 13, 1985

[1] This paper describes research done at the Department of Psychology and at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided by National Institutes of Health Research Grant AM 26710, awarded by the National Institute of Arthritis, Metabolism, and Digestive Diseases, and by a National Science Foundation graduate fellowship (C. G. A.). We also acknowledge early contributions of Michael Propp and Jonathan Delatizky toward development of our Selspot system, and of Eric Saund for display software development.

[2] To whom correspondence should be addressed.

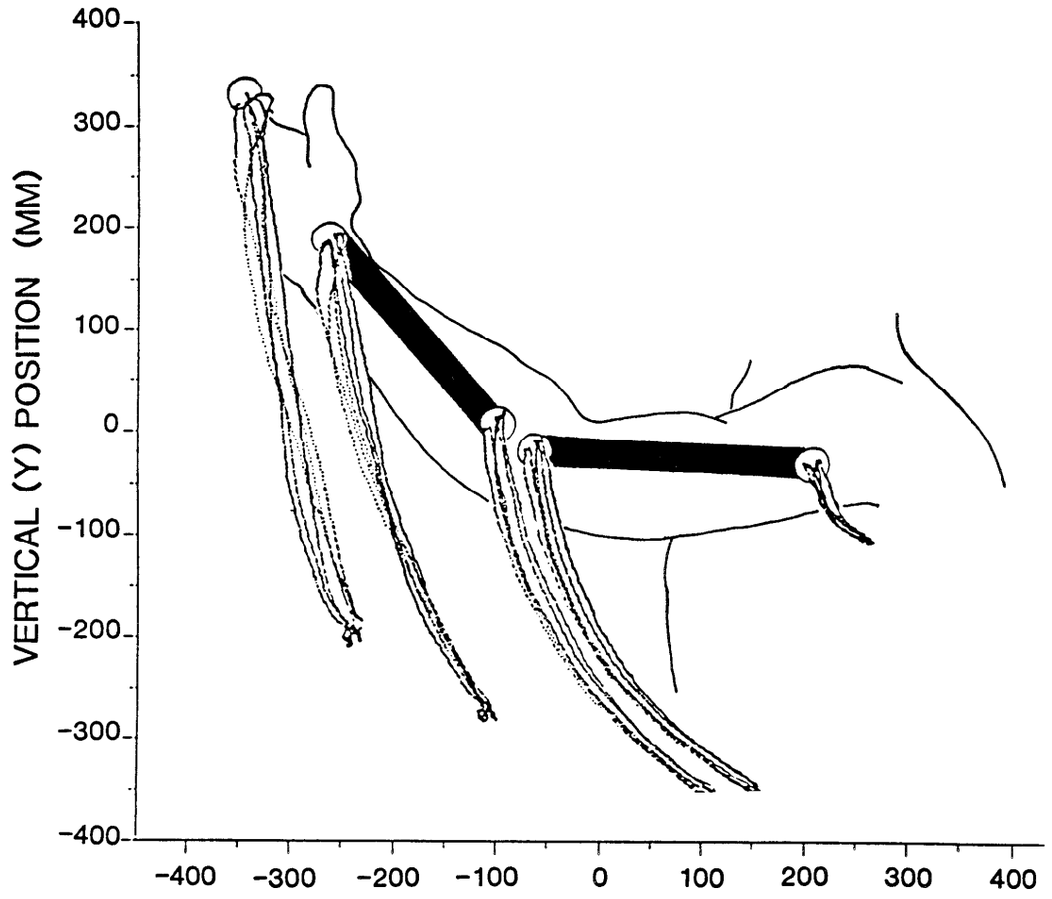


Figure 2. Attachment of Selspot markers and data presentation. Locations of the Selspot infrared LED markers and the typical format of the data presentation are shown. Note that the wrist and one of the elbow LEDs are connected by a rigid bar aligned with the forearm, and the shoulder and other elbow LEDs are similarly connected on a rigid bar aligned with the upper arm. The three-dimensional Selspot data are projected onto the *XY* plane. This projection shows most of the features of the path because these movements were almost planar (for the finger, wrist, and shoulder) and oriented parallel to the *XY* plane. In each data plot several movements are presented. Three upward movements (*dotted lines*) are indicated here by a *dot* at the location of the infrared LED for each sample (sampling frequency 315 Hz). Three downward movements are indicated by *solid lines* marking the path of each infrared LED.

camera and recording the average measured positions. Deviations between expected and actual measurements were calculated and mapped into a 25 × 25 correction table. Standard interpolation techniques were used to calculate the table originally and to read corrections from the table.

Three-dimensional positions of the LEDs were calculated from the corrected data using the known positions and orientations of the cameras and geometry. Points were marked as bad for which the vectors to the reconstructured LED position from each camera origin missed by greater than a certain threshold (3 cm), since with four parameters from the two cameras there is one redundant measurement. The Selspot system in our configuration can detect movements of the markers as small as 1 mm. Currently, the absolute accuracy of the system is within ±1 cm.

Normalization of tangential velocity profiles. To check invariance of tangential velocity profile shape, movements must be normalized for time and distance. Define $v(t)$ as the experimental tangential velocity profile as a function of time t, v_{max} as the maximum tangential velocity in $v(t)$, d as the experimental movement distance, v_{ref} as the reference velocity, and d_{ref} as the reference distance. The reference velocity and distance are chosen arbitrarily, and all data records are scaled to them. Since the tangential velocity profiles $v(t)$ are almost always unimodal, the maximum tangential velocity v_{max} is well defined. We use v_{max} rather than movement duration because of imprecision in determining movement start and stop points.

Now define time and distance scaling factors c and a as

$$c = \frac{v_{ref}}{v_{max}}, \qquad a = \frac{d_{ref}}{d}$$

The velocity profile $v'(t)$ normalized first for distance is $v'(t) = av(t)$. The maximum velocity for the new velocity profile is then $v'_{max} = av_{max}$. Define a new time scaling factor $c' = v_{ref}/v'_{max} = c/a$. Then the time-normalized velocity profile $v''(t)$ is

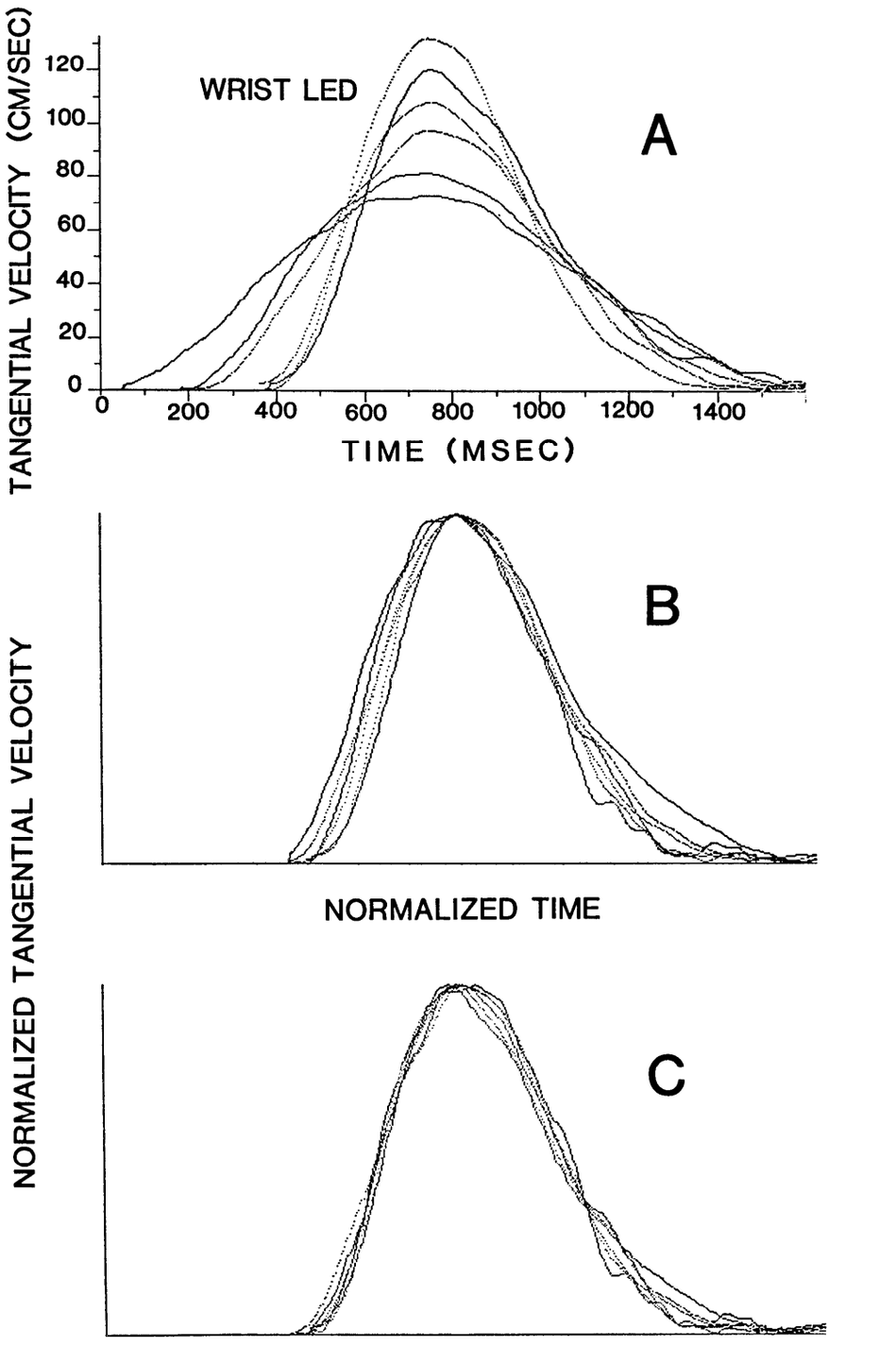


Figure 3. Normalization of tangential velocity profiles. *A*, A set of typical tangential velocity profiles of the wrist LED, derived from six unloaded movements at slow speeds between targets 3 and 7. *B*, Normalized tangential velocity profiles indicating substantial overlap. The units for the normalized tangential velocity profile have no physical meaning and are therefore not indicated. *C*, Realigned profiles through minimization of the similarity measure *w* with respect to the averaged velocity profile.

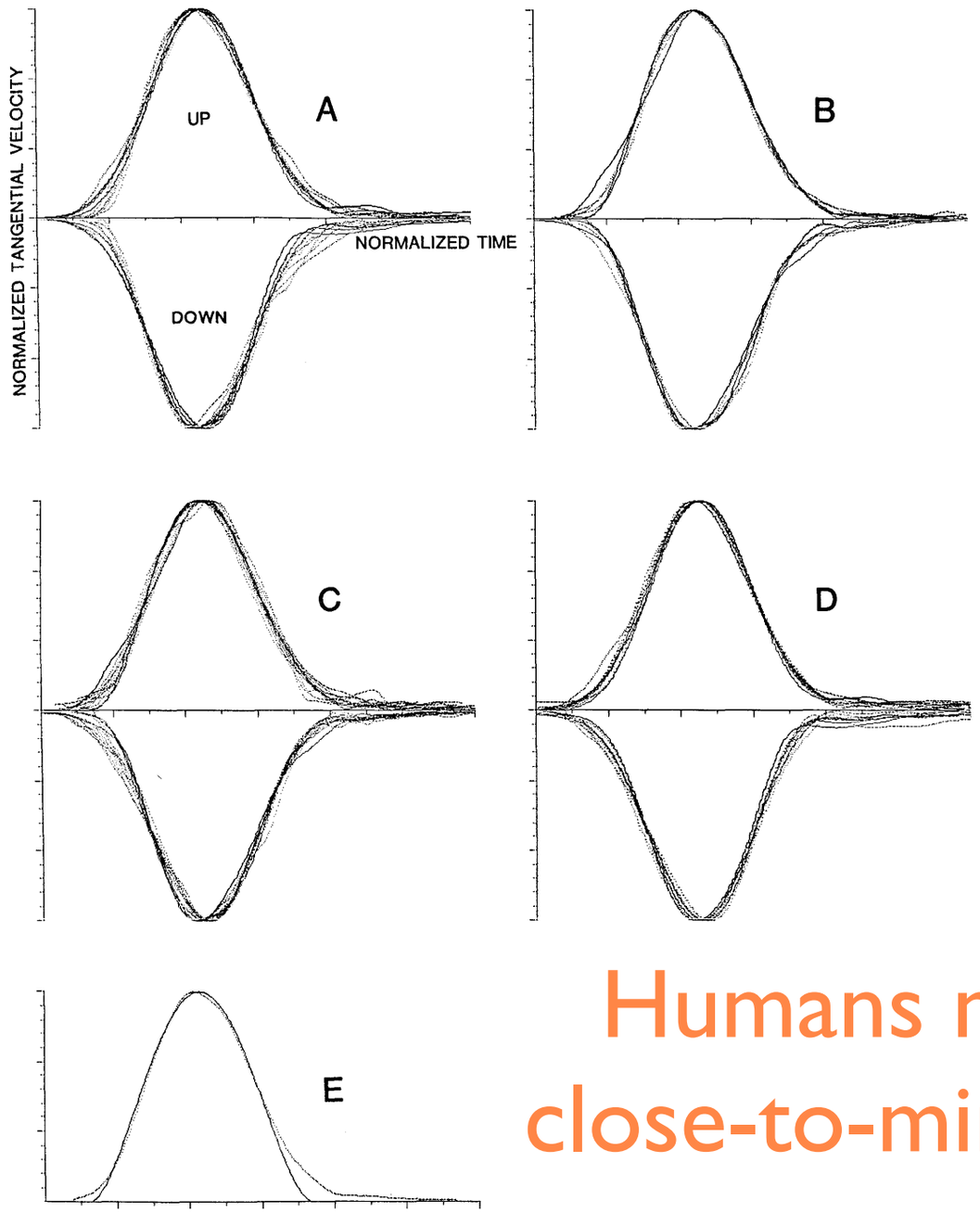


Figure 7. Tangential velocity profile shape invariance across different conditions. *A*, Different speeds. One subject's unloaded movements between targets 3 and 7 at slow, medium, and fast speeds are shown. Slow and fast movements are *dotted lines* here, whereas medium-speed movements are presented as *solid lines*. *B*, Different loads. One subject's medium-speed movements between targets 3 and 7 were executed unloaded (*solid lines*) and loaded by a 4-lb. hand-held load (*dotted lines*). *C*, Different movements. One subject's medium-speed unloaded movements between all target pairs are shown. Movements between targets 3 and 7 are shown as *solid lines*, and those between targets 1-5, 2-6, and 4-8 are shown as *dotted lines*. *D*, Different subjects. Fast speed unloaded movements between targets 3 and 7 from three subjects. Movements of the same subject as presented in *A*-*C* are shown as *solid lines*; movements of the other two subjects are shown as *dotted lines*. *E*, Comparison of reference profile (*dotted lines*) to minimum-jerk profile (*solid line*).

Humans move with close-to-minimum jerk.

What other kinds of trajectories can you think of?

Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

Ramp up velocity to desired value for a short time at start.

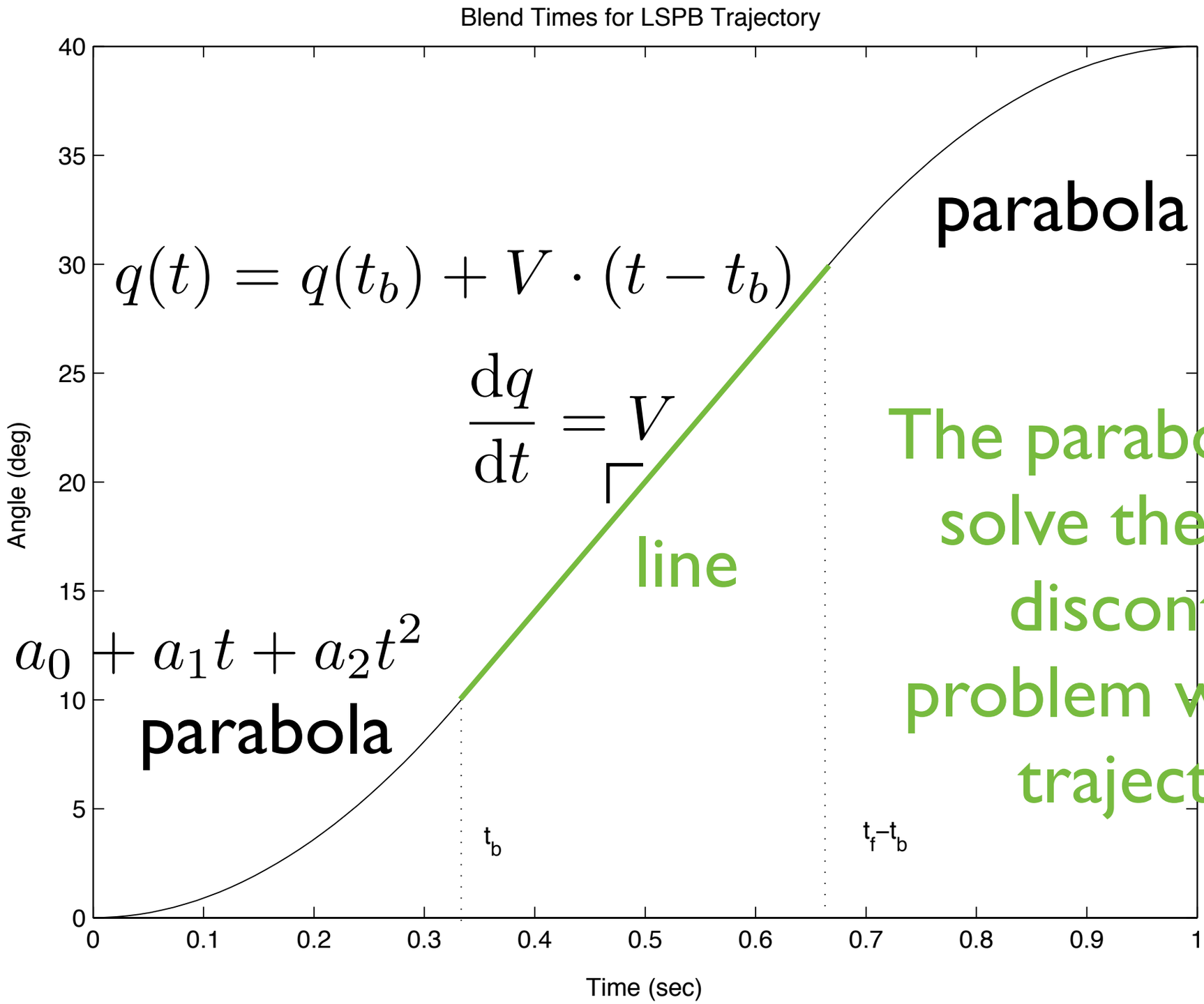
Move at constant velocity for a while.

Ramp down velocity to final value for a short time at end.

Start and end blend times are usually equal.

Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

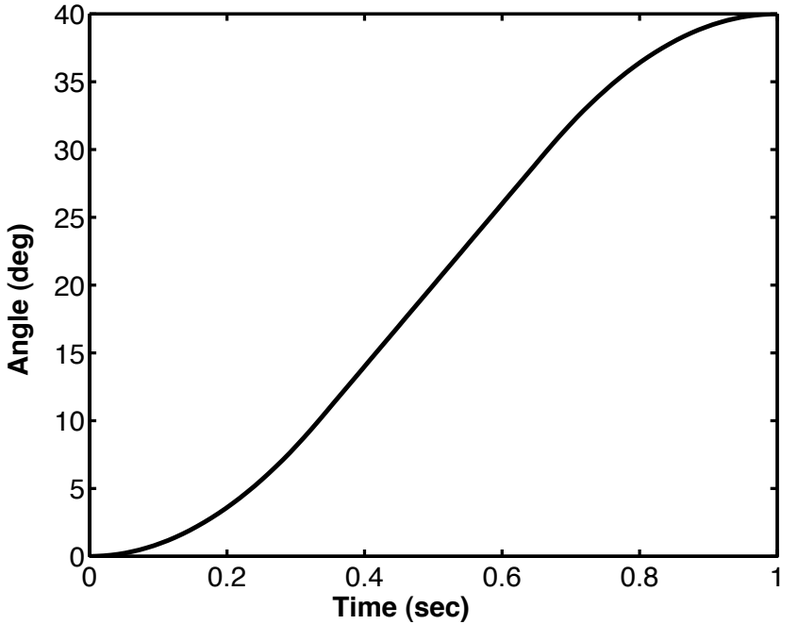


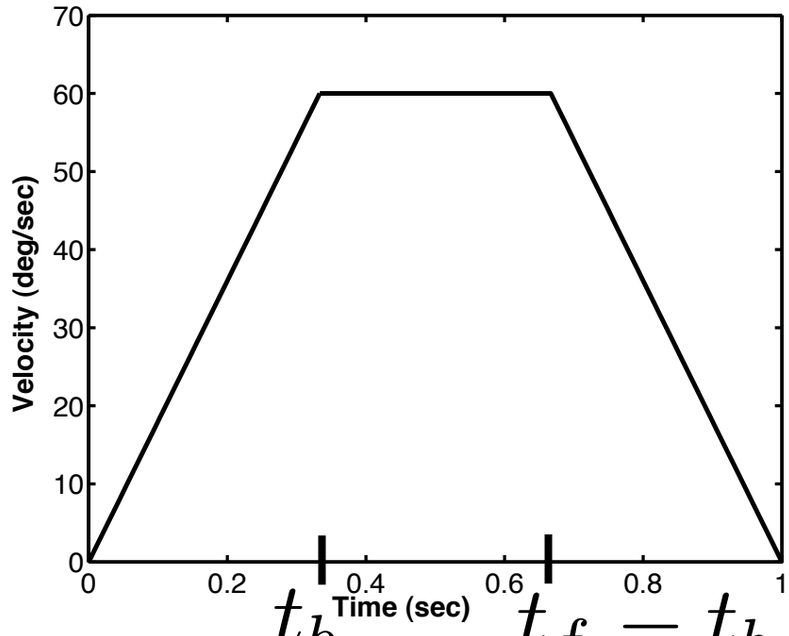
The parabolic blends solve the velocity discontinuity problem with linear trajectories.

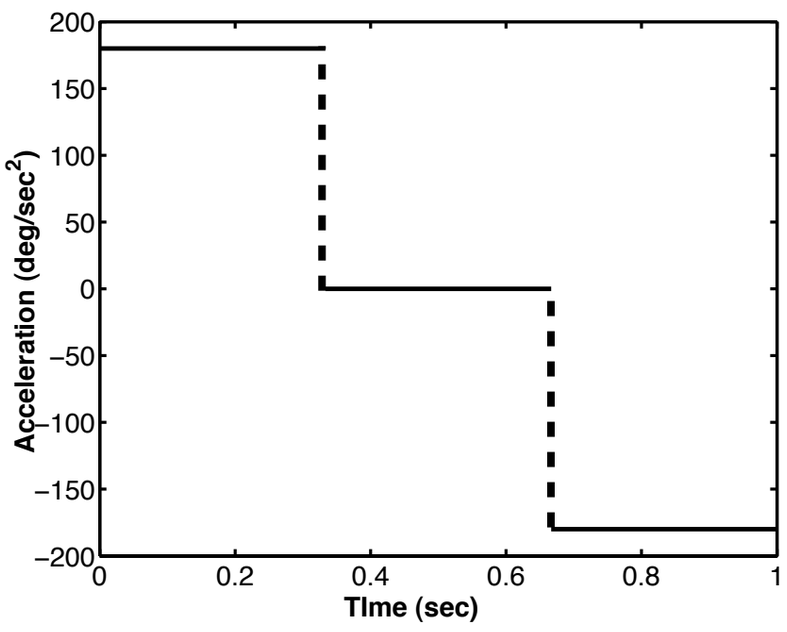
Specifying Constant Velocity for Central Portion

Linear Segments with Parabolic Blends (LSPB)

Piecewise constant accelerations







Trapezoidal velocity profile

Limits

$$0 < t_b \leq \frac{t_f}{2}$$

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}$$

Not minimum jerk...

Getting There As Fast As Possible

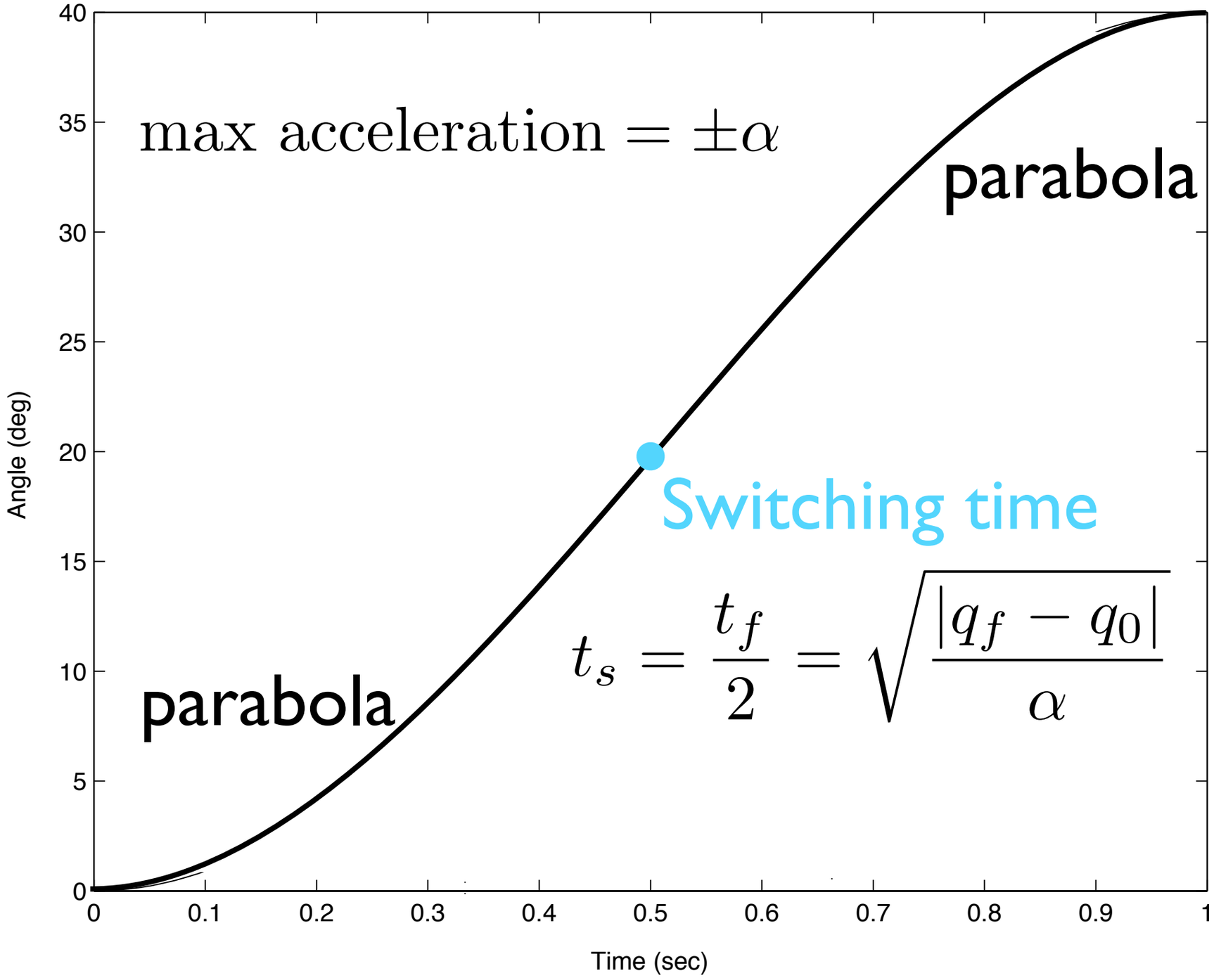
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories

Leave final time unspecified.
Specify the maximum acceleration possible,
typically set by actuator limit.
Apply maximum acceleration in one direction,
then abruptly switch to negative maximum acceleration.

Typically starting and ending at rest.
Switching time is halfway through the trajectory.

Getting There As Fast As Possible

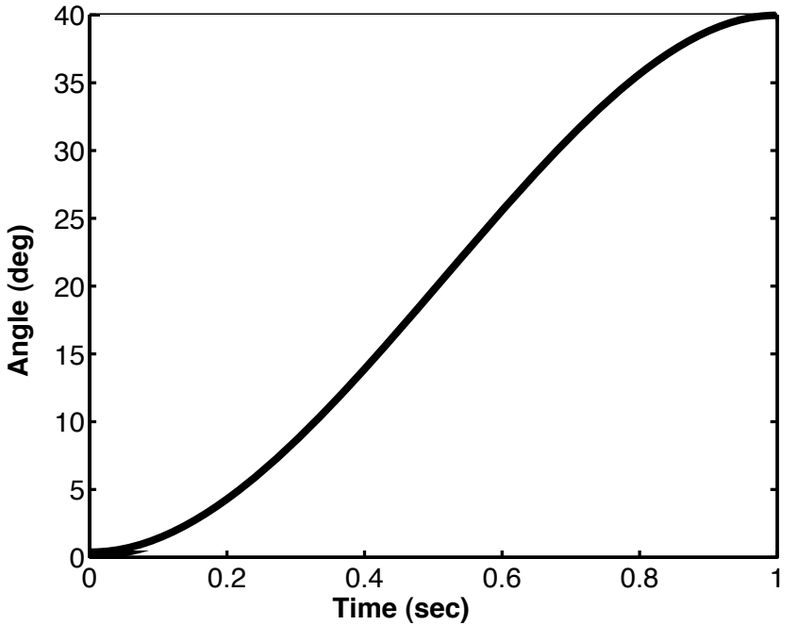
Minimum Time Trajectories, a.k.a. Bang-Bang Trajectories

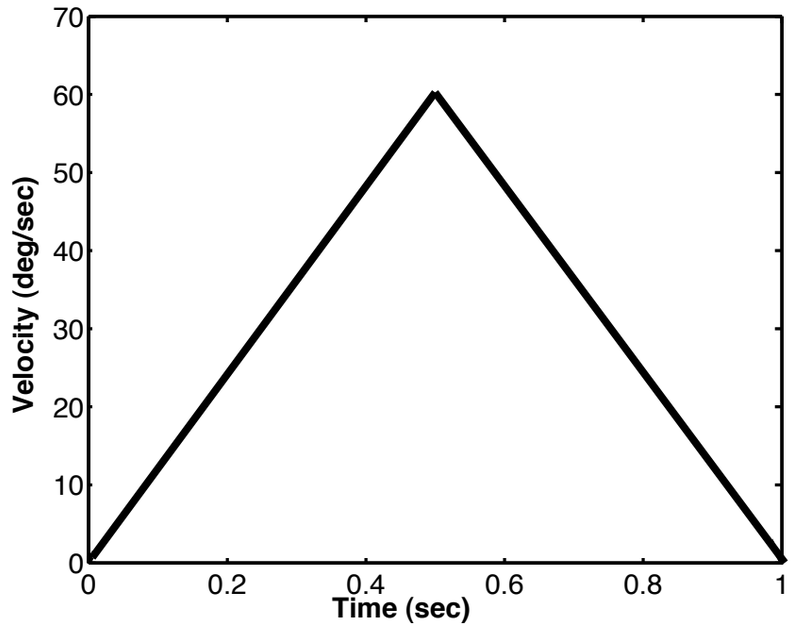


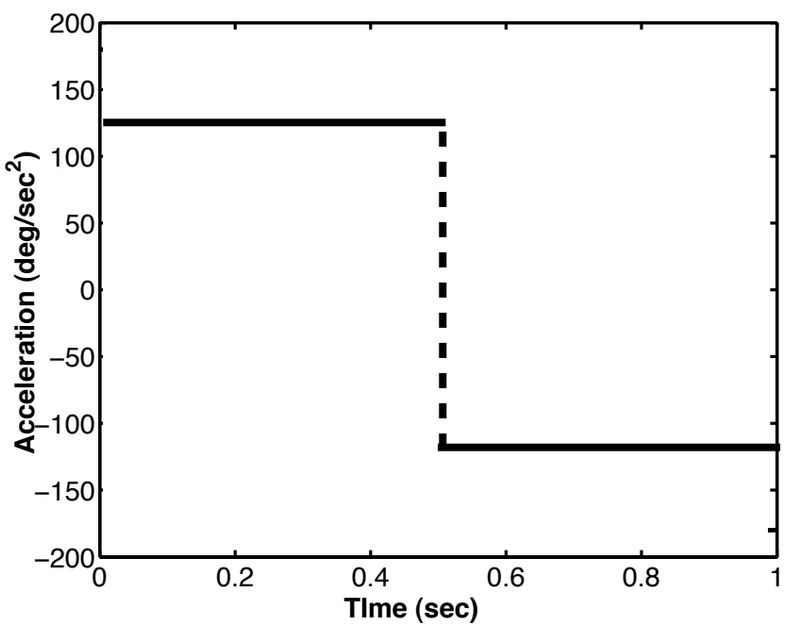
Getting There As Fast As Possible

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

Piecewise constant max accelerations







Triangular velocity profile

Not minimum jerk...
...but fast!

What questions do you have ?

Moving Through Via Points

You could solve for a single high-order polynomial that hits all your via points.

This approach yields a nice continuously differentiable curve.

However, it is intractable when many via points are used because the linear system's dimension become very large.

Moving Through Via Points

Instead, use low-order polynomials for trajectory segments between adjacent via points.

Ensure that position, velocity, and acceleration constraints are satisfied at the via points, where we switch from one polynomial to the next.

Final conditions for one polynomial become the initial conditions for the next!

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

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m | make_quintic_polynomial.m | make_two_cubic_polynomials.m

```
 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);
39
40
41      %% Plot the cubic polynomial we calculated.
```

script | Ln 14 Col 1

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

EDITOR | PUBLISH | VIEW

make_cubic_polynomial.m | make_quintic_polynomial.m | make_two_cubic_polynomials.m

```
 94 -        set(gca,'fontsize',14)
 95 -        plot(t,qdoubledot,'r')
 96 -        xlabel('Time (s)')
 97 -        ylabel('a (radians/s^2)')
 98
 99 -    end
100
101     %% Define the second problem.
102
103     % Define initial and final times.
104 -   t0 = tf; % s
105 -   tf = 5; % s
106
107     % Define initial conditions.
108 -   q0 = qf; % radians
109 -   v0 = vf;    % rad/s
110
111     % Define final conditions.
112 -   qf = -2.3;  % radians
113 -   vf = -.5;    % rad/s
114
115
116     %% Solve for the cubic polynomial coefficients that meet these conditions.
117
118     % Put initial and final conditions into a column vector.
119 -   conditions = [q0 v0 qf vf]';
120
121     % Put time elements into matrix.
122 -   mat = [1    t0   t0^2     t0^3;
123            0     1   2*t0   3*t0^2;
124            1    tf   tf^2     tf^3;
125            0     1   2*tf   3*tf^2];
126
127     % Solve for coefficients.
128 -   coeffs = mat \ conditions;
129
130     % Pull individual coefficients out.
131 -   a0 = coeffs(1);
```

script | Ln 113 Col 21

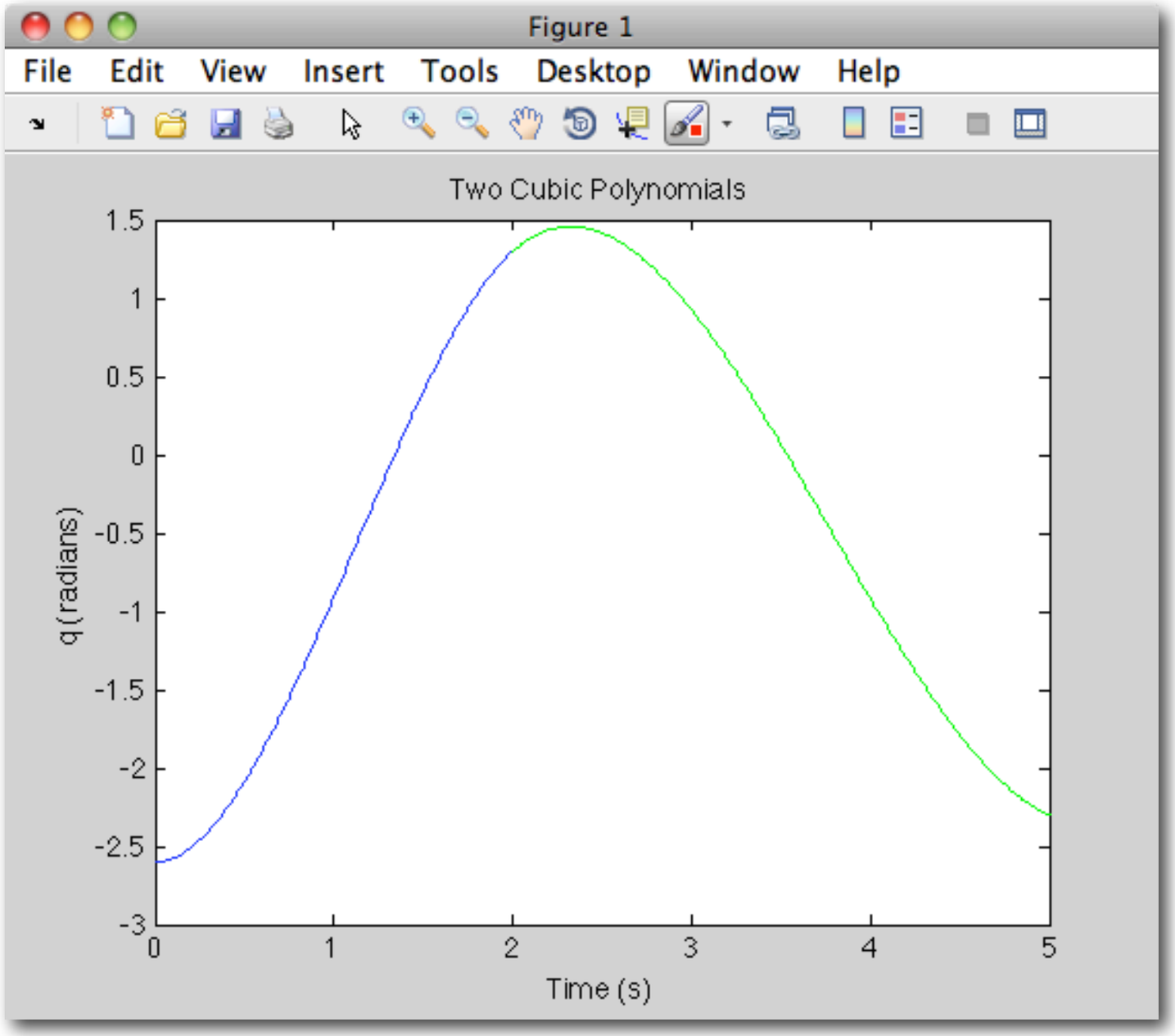


Figure 1
File Edit View Insert Tools Desktop Window Help
Two Cubic Polynomials
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
q (radians)
0
1
2
3
4
5
Time (s)

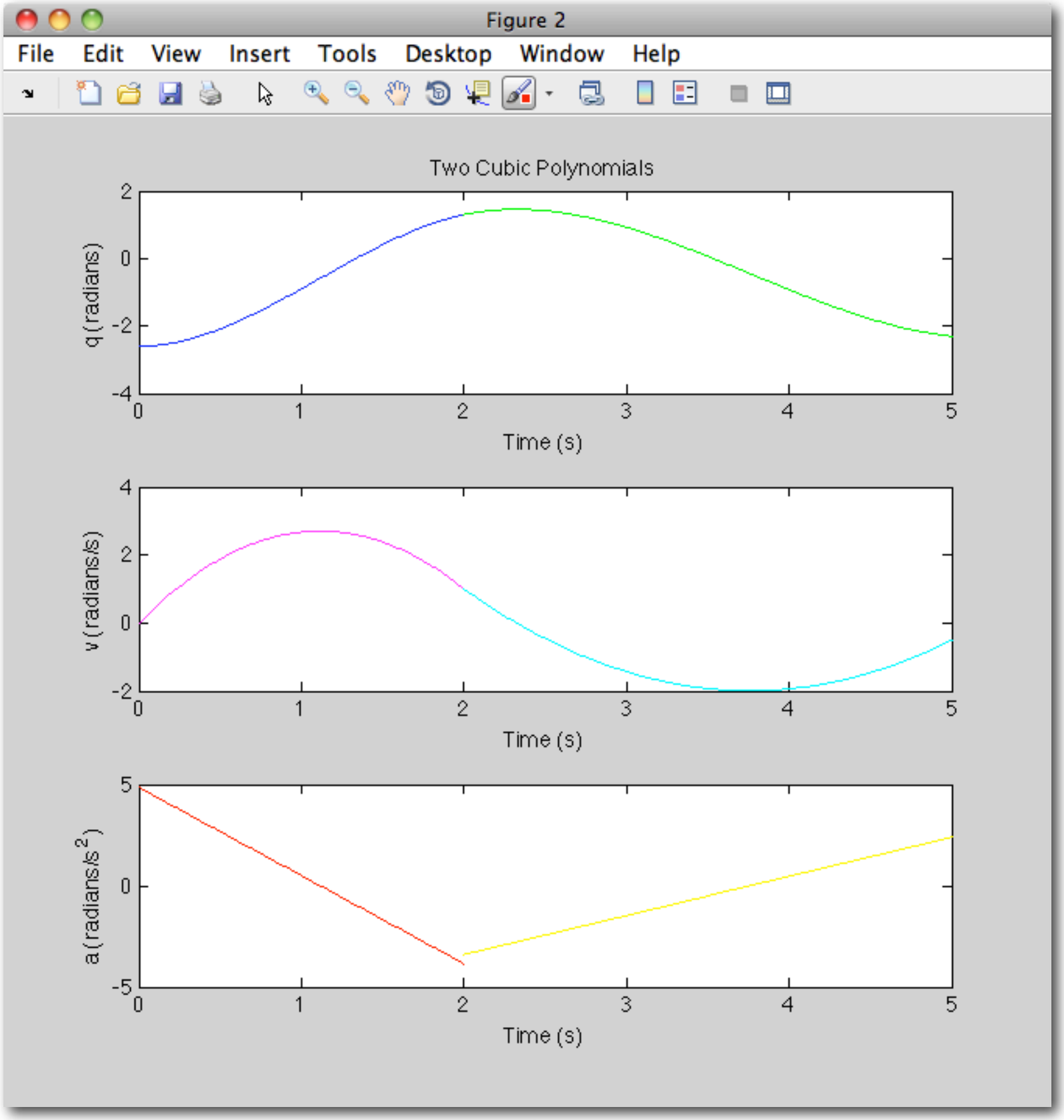


Figure 2
File Edit View Insert Tools Desktop Window Help
Two Cubic Polynomials
q (radians)
Time (s)
v (radians/s)
Time (s)
a (radians/s^2)
Time (s)

What questions do you have ?

Homework 5:
PUMA 260

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

September 26, 2013

This assignment is due on **Tuesday, October 1, 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 Thursday, October 3, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 20%. 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 should 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

We encourage you to do this assignment with a partner. If you prefer, you may also do this assignment individually. If you do this homework with a partner, you may work with anyone you choose, even someone with substantial MATLAB experience. If you are looking for a partner, 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 5 (MATLAB)

Due tonight by midnight.

Late deadline is Thursday 10/3 at midnight.

Penalty of -10% per day, up to 20%.

$a = 13.0$ in.

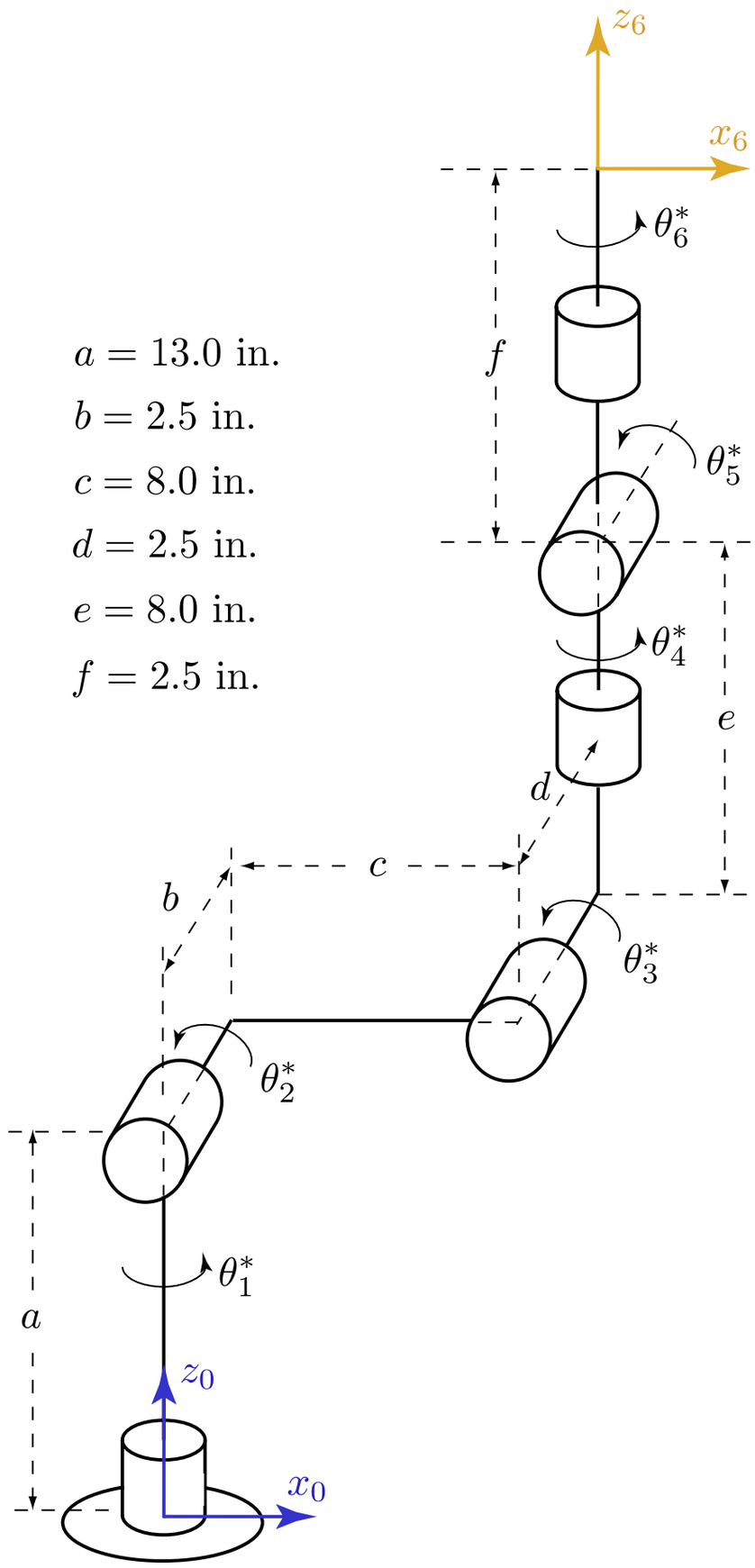
$b = 2.5$ in.

$c = 8.0$ in.

$d = 2.5$ in.

$e = 8.0$ in.

$f = 2.5$ in.



$$-180^\circ < \theta_1^* < 110^\circ$$

$$-75^\circ < \theta_2^* < 240^\circ$$

$$-235^\circ < \theta_3^* < 60^\circ$$

$$-580^\circ < \theta_4^* < 40^\circ$$

$$-120^\circ < \theta_5^* < 110^\circ$$

$$-215^\circ < \theta_6^* < 295^\circ$$

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/07 dh/scara_robot_kuchenbe.m

EDITOR | PUBLISH | VIEW

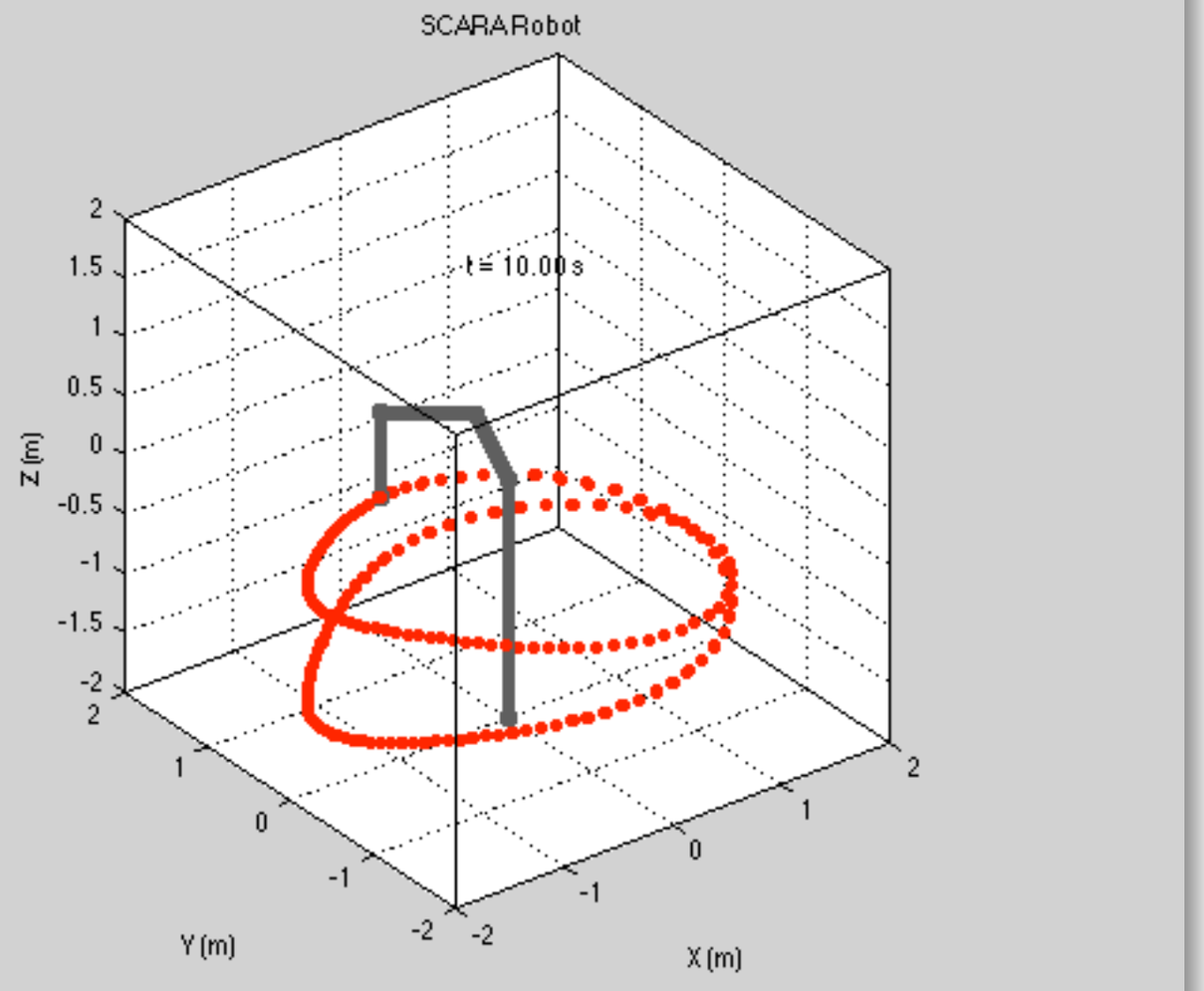
scara_robot_kuchenbe.m | dh_kuchenbe.m

```
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 fi
14  home
15
16  % Define our time vector.
17  tStart = 0;    % The time at which the simulati
18  tStep = 0.04; % The simulation's time step, ir
19  tEnd = 10;     % The time at which the simulati
20  t = (tStart:tStep:tEnd)';  % The time vector (
21
22  % Set whether to animate the robot's movement
23  pause on;  % Set this to off if you don't want
24  GraphingTimeDelay = 0.001; % The length of tim
25
26
27  %% MOTION MODES
28
29  % Ask the user to get the mode of the robot mo
30  [selection,ok] = listdlg('PromptString','Which
31
32  % Define the joint coordinates given the selec
33  % setting the values for theta1, theta2, and d
34  % vector of the same dimensions as t.  We use
35  % meters for the displacement.
36  if ~ok
37      selection = -1;
38  end
```

script

Figure 1

File Edit View Insert Tools Desktop Window Help



Similar to SCARA animation

note

New PUMA Motions Posted for Homework 5

I just posted a new version of puma_motions.zip. It contains two additional joint angle histories for you to test with the PUMA animation you are creating for Homework 5.

In puma_motion8.mat, the PUMA should always point its final link in the positive x-direction and should draw a vertical circle in the air. The circle should have a radius of 4 inches (diameter of 8 inches) and occur in a vertical plane located at x = 10 inches. The center of the circle should be at y = 0 in. and z = 21 in.

In puma_motion9.mat, the PUMA should always point its final link in the positive z-direction and should draw a horizontal circle in the air. The circle should have a radius of 3 inches (diameter of 6 inches) and occur in a horizontal plane located at z = 22 inches. The center of the circle should be at x = 3 in. and y = 12 in.

Title: puma_motions.zip
http://www.piazza.com/class_profile/get_resource/hf935b0sz1m5r3/hm8kcwn18lk3nk
(updated once more since this posting, see @82 and @91)

You can also view it on the course page:
https://piazza.com/upenn/fall2013/meam520/resources

I hope these new files help! Please post any follow-up questions.

hw5

MATLAB R2012b

HOME | PLOTS | APPS | Search Documentation

/ ▸ Users ▸ kuchenbe ▸ Documents ▸ teaching ▸ meam 520 ▸ assignments ▸ 05 puma ▸ matlab ▸

Current Folder

Name ▲

- ik
- puma_motions
- dh_kuchenbe.m
- puma_motion0.mat
- puma_motion1.mat
- puma_motion2.mat
- puma_motion3.mat
- puma_motion4.mat
- puma_motion5.mat
- puma_motion6.mat
- puma_motion7.mat
- puma_motion8.mat
- puma_motion9.mat
- puma_motion10.mat
- puma_motions.zip
- puma_robot_kuchenbe.m
- puma_robot_kuchenbe_alt.m
- puma_robot_kuchenbe_alt2.m
- puma_robot_kuchenbe_load.m

puma_robot_kuchenbe.m (MATLAB Scri

Command Window

```
>> load puma_motion8
fx >>
```

Workspace

Name	Value	Min	Max
T06_history	<4x4x36...	-4.0000	25
t_history	<361x1...	0	18
theta1_his...	<361x1...	-1.1188	-0.1389
theta2_his...	<361x1...	-0.5056	0.5385
theta3_his...	<361x1...	-0.3781	0.6827
theta4_his...	<361x1...	-3.0013	-2.0188
theta5_his...	<361x1...	1.6322	1.7217
theta6_his...	<361x1...	-3.1219	-3.0046

Command History

```
clear
clc
load puma_motion8
```

MATLAB R2012b

HOME | PLOTS | APPS | Search Documentation

/ ▸ Users ▸ kuchenbe ▸ Documents ▸ teaching ▸ meam 520 ▸ assignments ▸ 05 puma ▸ matlab ▸

Current Folder

Name ▲

- ik
- puma_motions
- dh_kuchenbe.m
- puma_motion0.mat
- puma_motion1.mat
- puma_motion2.mat
- puma_motion3.mat
- puma_motion4.mat
- puma_motion5.mat
- puma_motion6.mat
- puma_motion7.mat
- puma_motion8.mat
- puma_motion9.mat
- puma_motion10.mat
- puma_motions.zip
- puma_robot_kuchenbe.m
- puma_robot_kuchenbe_alt.m
- puma_robot_kuchenbe_alt2.m
- puma_robot_kuchenbe_load.m

puma_robot_kuchenbe.m (MATLAB Scri

Command Window

```
>> load puma_motion8
>> help plot3
 plot3  Plot lines and points in 3-D space.
    plot3() is a three-dimensional analogue of PLOT().

    plot3(x,y,z), where x, y and z are three vectors of the same length,
    plots a line in 3-space through the points whose coordinates are the
    elements of x, y and z.

    plot3(X,Y,Z), where X, Y and Z are three matrices of the same size,
    plots several lines obtained from the columns of X, Y and Z.

    Various line types, plot symbols and colors may be obtained with
    plot3(X,Y,Z,s) where s is a 1, 2 or 3 character string made from
    the characters listed under the PLOT command.

    plot3(x1,y1,z1,s1,x2,y2,z2,s2,x3,y3,z3,s3,...) combines the plots
    defined by the (x,y,z,s) fourtuples, where the x's, y's and z's are
    vectors or matrices and the s's are strings.

    Example: A helix:

        t = 0:pi/50:10*pi;
        plot3(sin(t),cos(t),t);

    plot3 returns a column vector of handles to lineseries objects, one
    handle per line. The X,Y,Z triples, or X,Y,Z,S quads, can be
    followed by parameter/value pairs to specify additional
    properties of the lines.

    See also plot, line, axis, view, mesh, surf.

    Overloaded methods:
       gpuArray/plot3

    Reference page in Help browser
       doc plot3

fx >>
```

Workspace

Name	Value
T06_history	<4x4>
t_history	<361>
theta1_his...	<361>
theta2_his...	<361>
theta3_his...	<361>
theta4_his...	<361>
theta5_his...	<361>
theta6_his...	<361>

Command His...

```
load puma_m
help plot3
```

Help

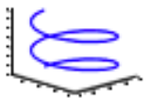
plot3

Search Documentation

MATLAB › Graphics › 2-D and 3-D Plots › Line Plots

plot3

3-D line plot



Syntax

```
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h = plot3(...)
```

Description

The `plot3` function displays a three-dimensional plot of a set of data points.

`plot3(X1,Y1,Z1,...)`, where `X1`, `Y1`, `Z1` are vectors or matrices, plots one or more lines in three-dimensional space through the points whose coordinates are the elements of `X1`, `Y1`, and `Z1`.

`plot3(X1,Y1,Z1,LineSpec,...)` creates and displays all lines defined by the `Xn,Yn,Zn,LineSpec` quads, where `LineSpec` is a line specification that determines line style, marker symbol, and color of the plotted lines.

`plot3(...,'PropertyName',PropertyValue,...)` sets properties to the specified property values for all line graphics objects created by `plot3`. See `lineseries properties` for a description of the properties you can set.

`h = plot3(...)` returns a column vector of handles to lineseries graphics objects, with one handle per object.

Examples

Plot a three-dimensional helix.

MATLAB does exactly what you tell it to do and no more.

Turn on MATLAB's beep (beep on) so you can hear when there is an error in your code.

Read the error message carefully.
(MATLAB has bad social skills and speaks cryptically, but these messages give you clues!)

Click the link to where the error is in your code.

Examine your variables in the workspace.

Test commands at the command line.

What questions do you have ?