Introduction to Robotics Toolbox for MATLAB

Yang Shen

Ph.D. Candidate, Bionics Lab, UCLA

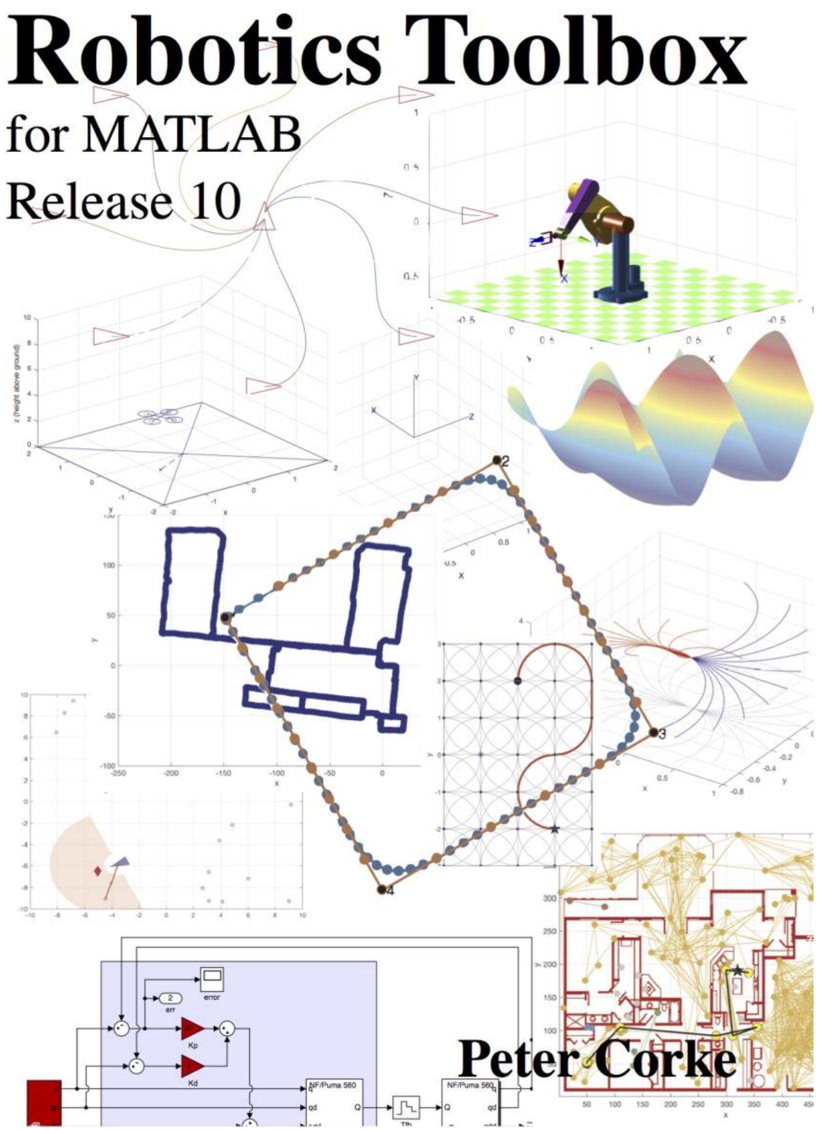
MAE 263B

Overview of today's lecture

- Robotics Toolbox for MATLAB: overview, online resources, basic operations, installation, built-in demo
- Serial-link manipulator example – Puma560: DH parameters, forward & inverse kinematics
- How to better use RTB manual
- Bugs – example, possible solutions
- Simulink – intro, RTB library for Simulink, RTB examples for Simulink

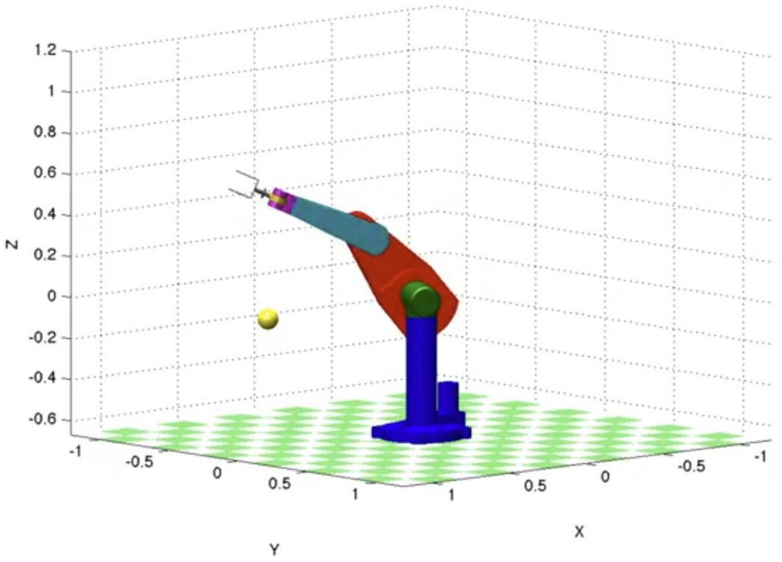
Overview of Robotics ToolBox

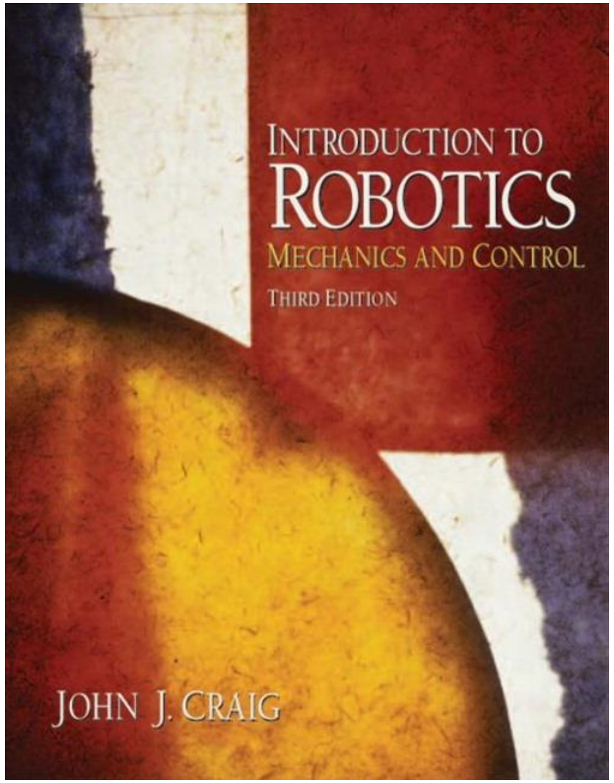
- Several releases of RTB could be found online. For this course, we will use the latest 10th release (http://petercorke.com/wordpress/toolboxes/robotics-toolbox)
- The reference book by the same author could be found here: http://petercorke.com/wordpress/rvc/
- You should be able to download both of them from the website, if you are using your UCLA VPN or connecting to a campus network.



Why RTB?

- The mathematical and visual expression of robots like serial-link robotic manipulator could be encapsulated as a reusable class/object. We do no need to spend time rebuilding these wheels and could focus on more complicated designs, either mechanical or algorithmic.
- Introduction to Robotics (3rd edition) used an earlier version of the Toolbox, you may find the difference in syntax and do some coding exercise on your own.





Basic operations

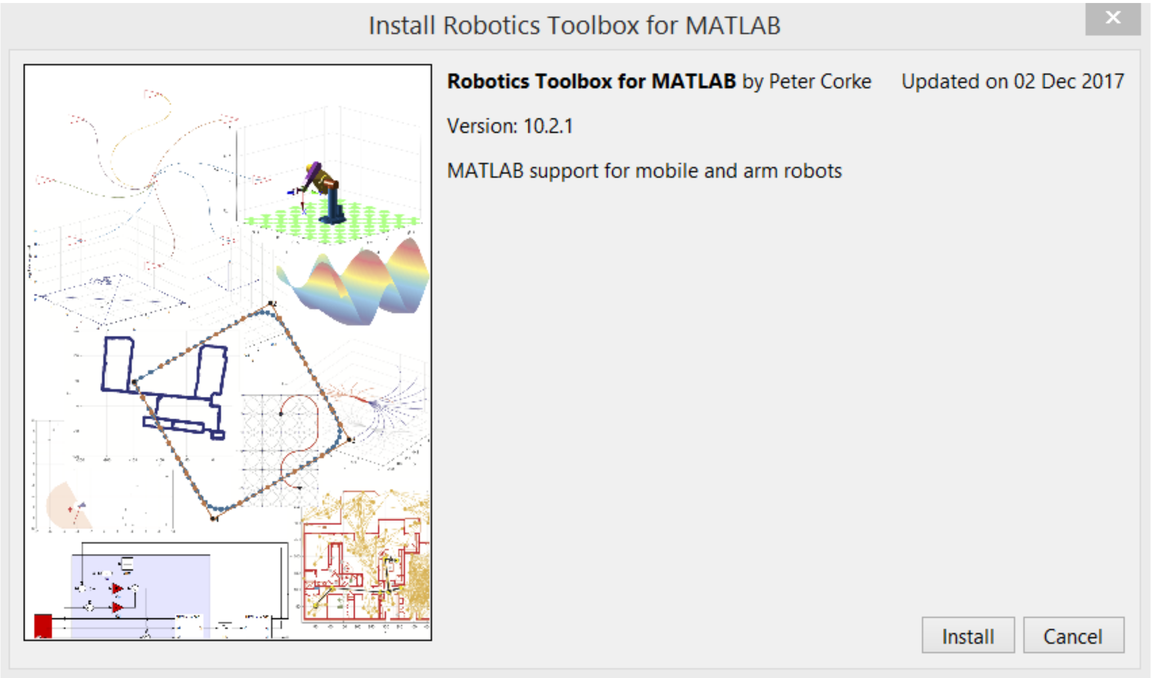
- Homogeneous transformation 2D/3D
- Differential motion
- Trajectory generation
- Pose representation
- Serial-link manipulator
- Classic robot models (e.g., Puma 560)
- Kinematics
- Dynamics
- Mobile robot
- Localization
- Path planning
- Graphics

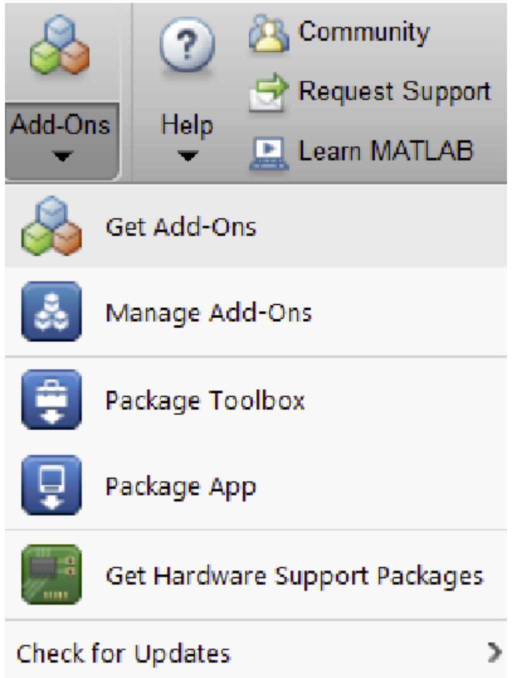
You are encouraged to read the first chapter of this manual pdf file!

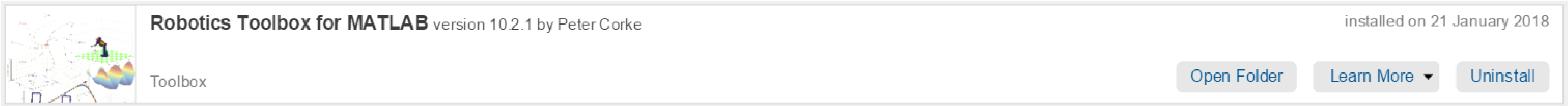
Contents

Installation

- My platform: Windows 8.1 (64 bit)
- My MATLAB version: R2016b, R2016a, R2015a. RTB installed via .mltbx

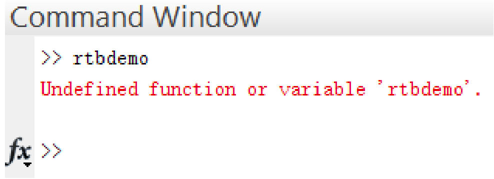




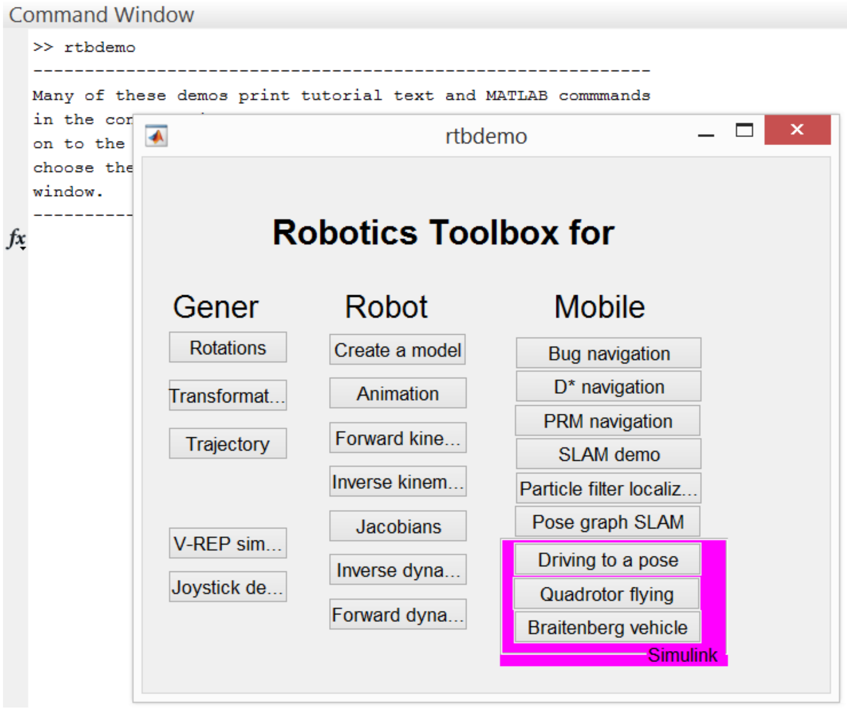


After installation – check via rtbdemo

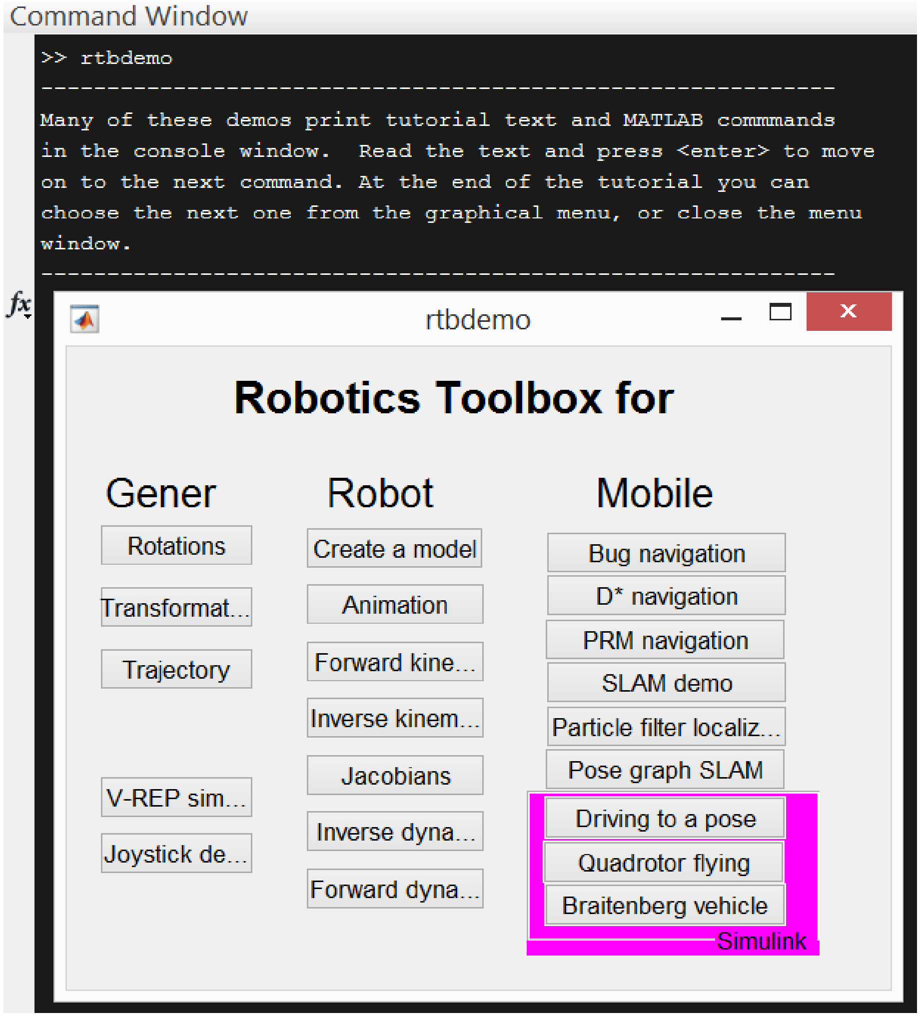
- R2015a (RTB not detected)



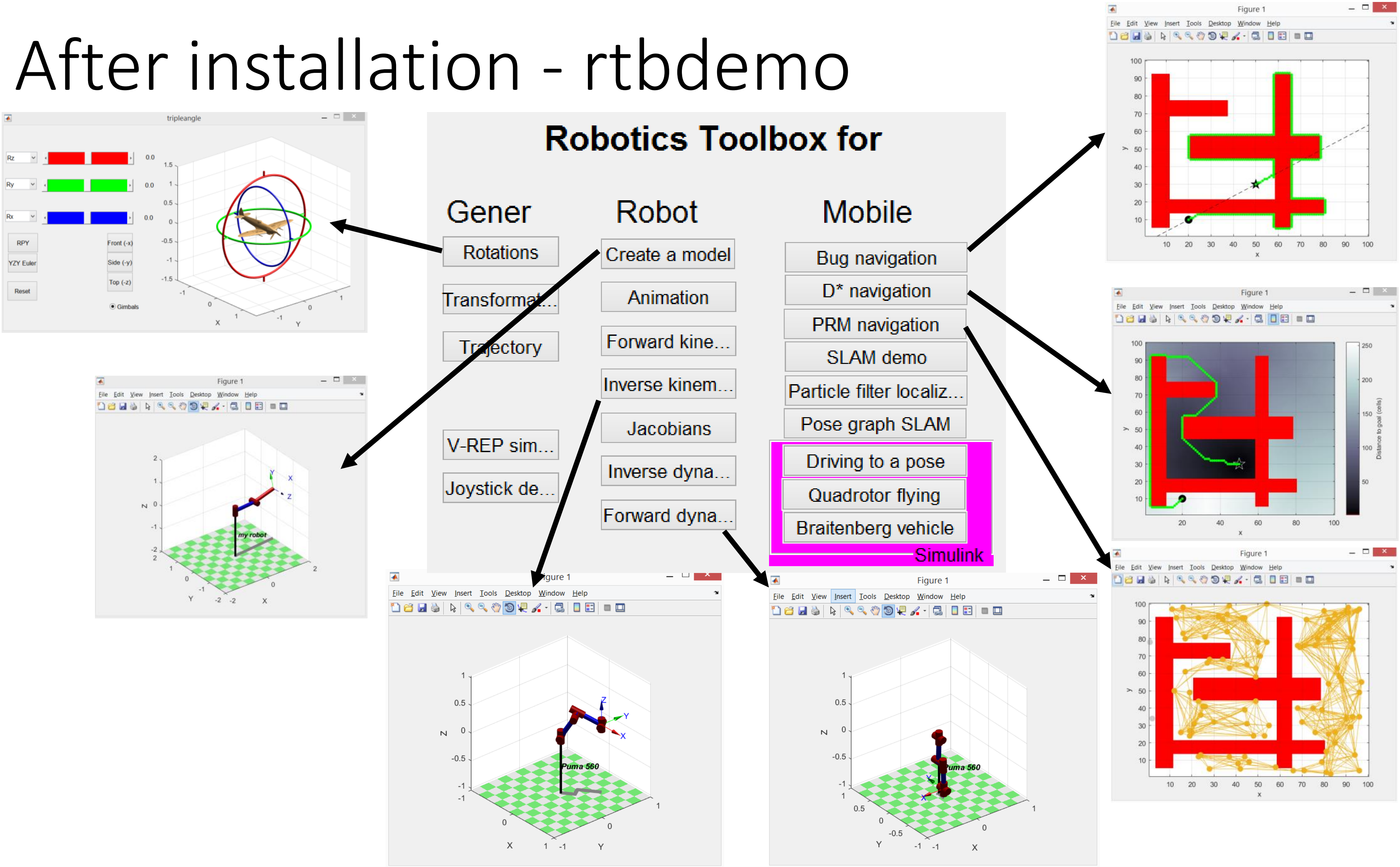
- R2016a (RTB installed)



- R2016b (RTB installed)



After installation - rtbdemo



Warning

- Be careful when you copy and test MATLAB codes directly from the manual – the quotation mark (’) is not in the correct format that MATLAB could read.

Examples

Create a 2-link robot

```
L(1) = Link([ 0     0    a1   pi/2], ’standard’);
L(2) = Link([ 0     0    a2   0], ’standard’);
twolink = SerialLink(L, ’name’, ’two link’);
```

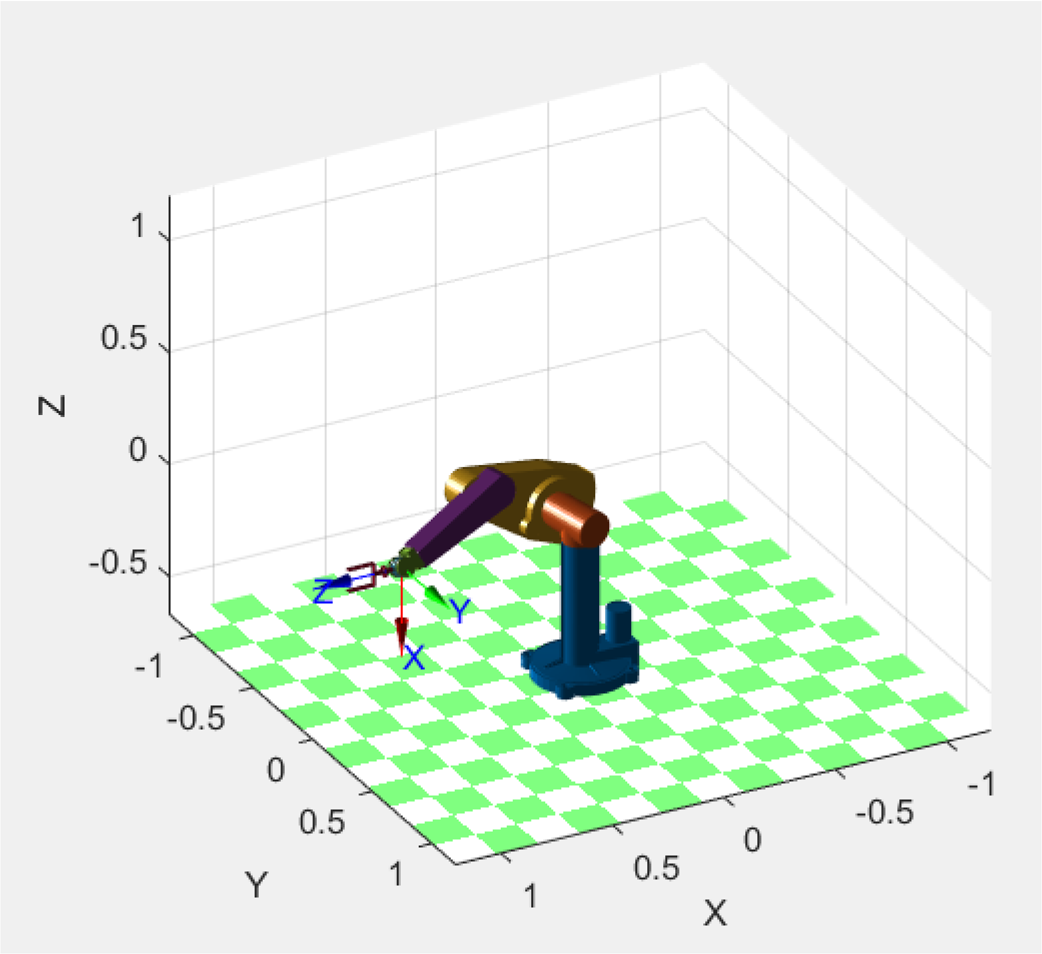
```
L(1) = Link([ 0 0 a1 pi/2], ’standard’);
L(2) = Link([ 0 0 a2 0], ’standard’);
twolink = SerialLink(L, ’name’, ’two link’);
```

```
L(1) = Link([ 0 0 a1 pi/2], 'standard');
L(2) = Link([ 0 0 a2 0], 'standard');
twolink = SerialLink(L, 'name', 'two link');
```

Serial-link manipulator example – Puma560

```
clear
mdl_puma560 % load puma560 model
p = [0.8 0 0]; % target position in task space ([x y z])
T = transl(p) * troty(pi/2); % transformation matrix
qr(1) = -pi/2; % initial position in joint space
qqr = p560.ikine6s(T, 'ru'); % target position in joint space
qrt = jtraj(qr, qqr, 50); % compute the joint space trajectory
ae = [138 8] % view angle
p560.plot3d(qrt, 'view', ae);
```

```
>> help mdl_puma560
 mdl_puma560 Create model of Puma 560 manipulator

  mdl_puma560 is a script that creates the workspace variable p560 which
  describes the kinematic and dynamic characteristics of a Unimation Puma
  560 manipulator using standard DH conventions.
```



https://youtu.be/4ddfhcbIr1Y

DH parameters

Variables - p560

p560

1x1 SerialLink

Property	Value
name	'Puma 560'
gravity	[0;0;9.8100]
base	*1x1 SE3*
tool	*1x1 SE3*
manufacturer	'Unimation'
comment	'viscous friction; ...
plotopt	*[ ]*
fast	0
interface	*[ ]*
ikineType	'puma'
trail	*[ ]*
movie	*[ ]*
delay	*[ ]*
loop	*[ ]*
model3d	'UNIMATE/puma...
faces	*[ ]*
points	*[ ]*
plotopt3d	*[ ]*
n	6
links	*1x6 Revolute*
mdh	0
T	*[ ]*
config	'RRRRRR'
offset	[0,0,0,0,0,0]
qlim	*6x2 double*
d	[0,0,0.1501,0.431...
a	[0,0.4318,0.0203,...
alpha	[1.5708,0,-1.5708...
theta	*[ ]*

Workspace

Name	Value
deg	0.0175
p560	*1x1 SerialLink*
qn	[0,0.7854,3.1416,0,0.7854,0]
qr	[0,1.5708,-1.5708,0,0,0]
qs	[0,0,-1.5708,0,0,0]
qz	[0,0,0,0,0,0]

```
>> p560

p560 =

Puma 560 [Unimation]:: 6 axis, RRRRRR, stdDH, slowRNE
 - viscous friction; params of 8/95;
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|     1.5708|          0|
|  2|         q2|          0|     0.4318|          0|          0|
|  3|         q3|    0.15005|     0.0203|    -1.5708|          0|
|  4|         q4|     0.4318|          0|     1.5708|          0|
|  5|         q5|          0|          0|    -1.5708|          0|
|  6|         q6|          0|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

```
>> p560.mdh

ans =

     0
```

Properties (read only)

n	number of joints
config	joint configuration string, eg. 'RRRRRR'
mdh	kinematic convention boolean (0=DH, 1=MDH)
theta	kinematic: joint angles (1xN)
d	kinematic: link offsets (1xN)
a	kinematic: link lengths (1xN)
alpha	kinematic: link twists (1xN)

DH parameters - review

DH Parameters - Review

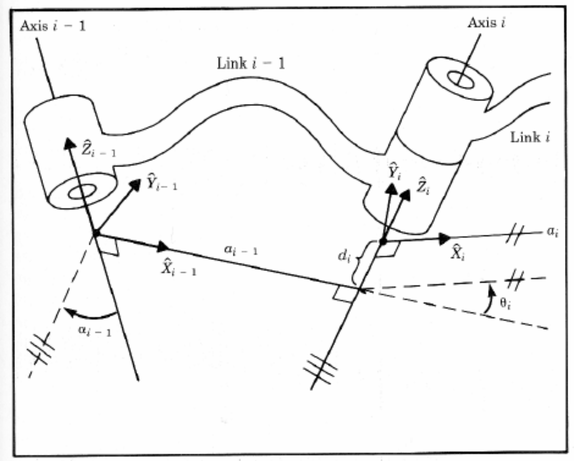
a_{i-1} - **Link Length** - The distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_{i-1}$

α_{i-1} - **Link Twist** - The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_{i-1}$

d_i - **Link Offset** - The distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$

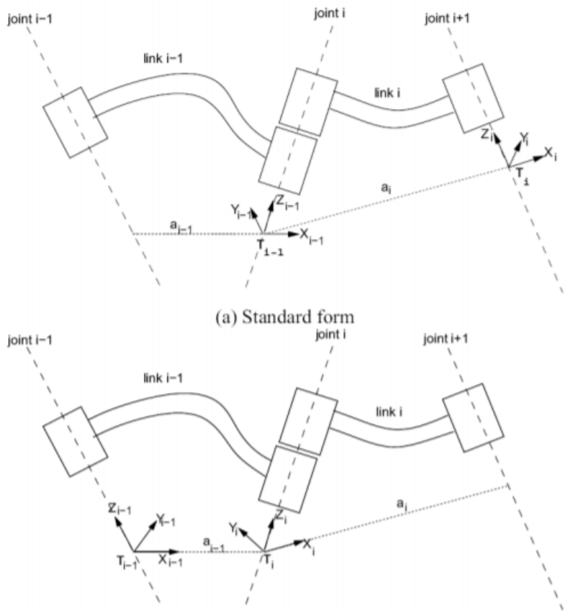
θ_i - **Link Angle** - The angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$

Note: $a_i \geq 0 \quad \alpha_i \quad d_i \quad \theta_i$ are singed quantities



Instructor: Jacob Rosen
Advanced Robotic - MAE 263D - Department of Mechanical & Aerospace Engineering - UCLA
UCLA

DH Parameters – Standard / Modified Approach



(a) Standard form

(b) Modified form

Standard Form

$$
{}^{i-1}_{\ i}T = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\alpha_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Modified Form

$$
{}^{i-1}_{\ i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Instructor: Jacob Rosen
Advanced Robotic - MAE 263D - Department of Mechanical & Aerospace Engineering - UCLA
UCLA

DH parameters – standard vs. modified

- Standard

```
clear
L(1) = Link([ 0 0 1 pi/2], 'standard');
L(2) = Link([ 0 1 2 0], 'standard');
L(3) = Link([ 0 2 3 pi/3], 'standard');
std3link = SerialLink(L, 'name', 'two link');
```

```
>> std3link

std3link =

two link:: 3 axis, RRR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          1|     1.5708|          0|
|  2|         q2|          1|          2|          0|          0|
|  3|         q3|          2|          3|     1.0472|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

- Modified

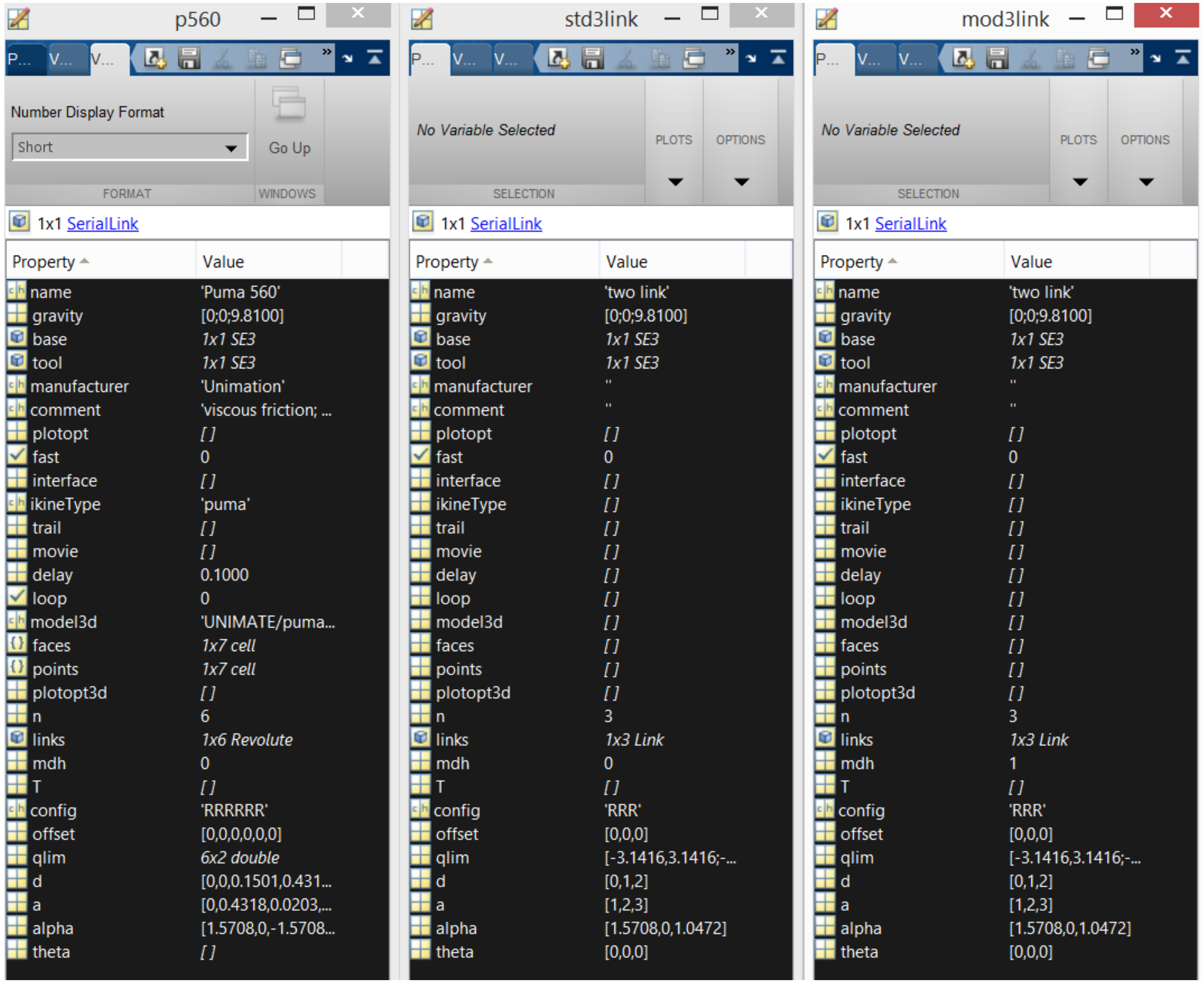
```
clear
L(1) = Link([ 0 0 1 pi/2], 'modified');
L(2) = Link([ 0 1 2 0], 'modified');
L(3) = Link([ 0 2 3 pi/3], 'modified');
mod3link = SerialLink(L, 'name', 'two link');
```

```
>> mod3link

mod3link =

two link:: 3 axis, RRR, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          1|     1.5708|          0|
|  2|         q2|          1|          2|          0|          0|
|  3|         q3|          2|          3|     1.0472|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

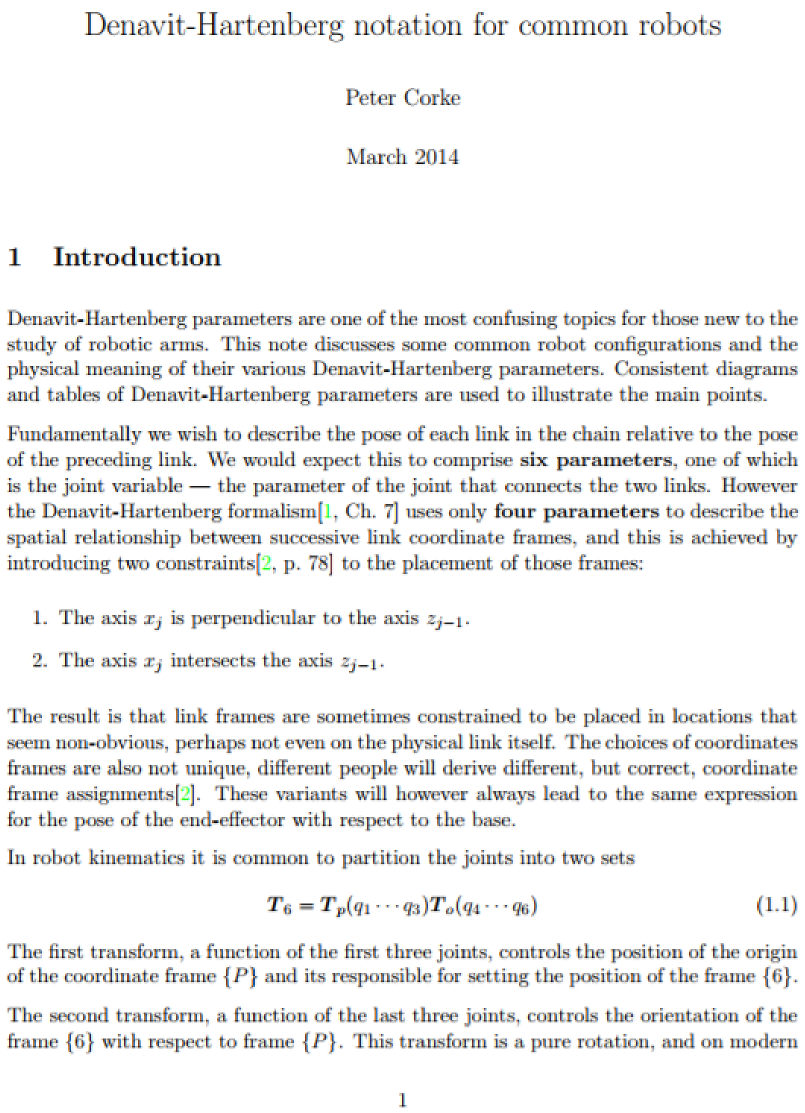
*Note: the built-in Puma 560 model uses standard DH parameters only, but some other built-in models like the Stanford Arm has modified DH parameter option.

DH parameters - more

A good reference:
http://www.petercorke.com/doc/rtb_dh.pdf



Property	Value (p560)	Value (std3link)	Value (mod3link)
name	'Puma 560'	'two link'	'two link'
gravity	[0;0;9.8100]	[0;0;9.8100]	[0;0;9.8100]
base	1x1 SE3	1x1 SE3	1x1 SE3
tool	1x1 SE3	1x1 SE3	1x1 SE3
manufacturer	'Unimation'	''	''
comment	'viscous friction; ...	''	''
plotopt	[]	[]	[]
fast	0	0	0
interface	[]	[]	[]
ikineType	'puma'	[]	[]
trail	[]	[]	[]
movie	[]	[]	[]
delay	0.1000	[]	[]
loop	0	[]	[]
model3d	'UNIMATE/puma...	[]	[]
faces	1x7 cell	[]	[]
points	1x7 cell	[]	[]
plotopt3d	[]	[]	[]
n	6	3	3
links	1x6 Revolute	1x3 Link	1x3 Link
mdh	0	0	1
T	[]	[]	[]
config	'RRRRRR'	'RRR'	'RRR'
offset	[0,0,0,0,0,0]	[0,0,0]	[0,0,0]
qlim	6x2 double	[-3.1416,3.1416;-...	[-3.1416,3.1416;-...
d	[0,0,0.1501,0.431...	[0,1,2]	[0,1,2]
a	[0,0.4318,0.0203,...	[1,2,3]	[1,2,3]
alpha	[1.5708,0,-1.5708...	[1.5708,0,1.0472]	[1.5708,0,1.0472]
theta	[]	[0,0,0]	[0,0,0]



Denavit-Hartenberg notation for common robots

Peter Corke

March 2014

1 Introduction

Denavit-Hartenberg parameters are one of the most confusing topics for those new to the study of robotic arms. This note discusses some common robot configurations and the physical meaning of their various Denavit-Hartenberg parameters. Consistent diagrams and tables of Denavit-Hartenberg parameters are used to illustrate the main points.

Fundamentally we wish to describe the pose of each link in the chain relative to the pose of the preceding link. We would expect this to comprise **six parameters**, one of which is the joint variable — the parameter of the joint that connects the two links. However the Denavit-Hartenberg formalism[1, Ch. 7] uses only **four parameters** to describe the spatial relationship between successive link coordinate frames, and this is achieved by introducing two constraints[2, p. 78] to the placement of those frames:

1. The axis x_j is perpendicular to the axis z_{j-1}.
2. The axis x_j intersects the axis z_{j-1}.

The result is that link frames are sometimes constrained to be placed in locations that seem non-obvious, perhaps not even on the physical link itself. The choices of coordinates frames are also not unique, different people will derive different, but correct, coordinate frame assignments[2]. These variants will however always lead to the same expression for the pose of the end-effector with respect to the base.

In robot kinematics it is common to partition the joints into two sets

$$\boldsymbol{T}_6 = \boldsymbol{T}_p(q_1 \cdots q_3)\boldsymbol{T}_o(q_4 \cdots q_6) \qquad (1.1)$$

The first transform, a function of the first three joints, controls the position of the origin of the coordinate frame $\{P\}$ and its responsible for setting the position of the frame $\{6\}$.

The second transform, a function of the last three joints, controls the orientation of the frame $\{6\}$ with respect to frame $\{P\}$. This transform is a pure rotation, and on modern

1

Kinematics

Forward kinematics

- Joint space -> End-effector space
- Useful commands:
  - SerialLink.jtraj
  - SerialLink.fkine
  - …

Inverse kinematics

- End-effector space -> Joint space
- Useful commands:
  - SerialLink.ikine
  - SerialLink.ikine3
  - SerialLink.ikine6s
  - …

How to better use the manual

- Get familiar with MATLAB Objects (especially if you have no experience in object-oriented programming): http://petercorke.com/wordpress/a-quick-introduction-to-matlab-objects
- I found some class descriptions in the manual are not self-contained. You may want to work with rtbdemo and example codes first.

What about bugs?

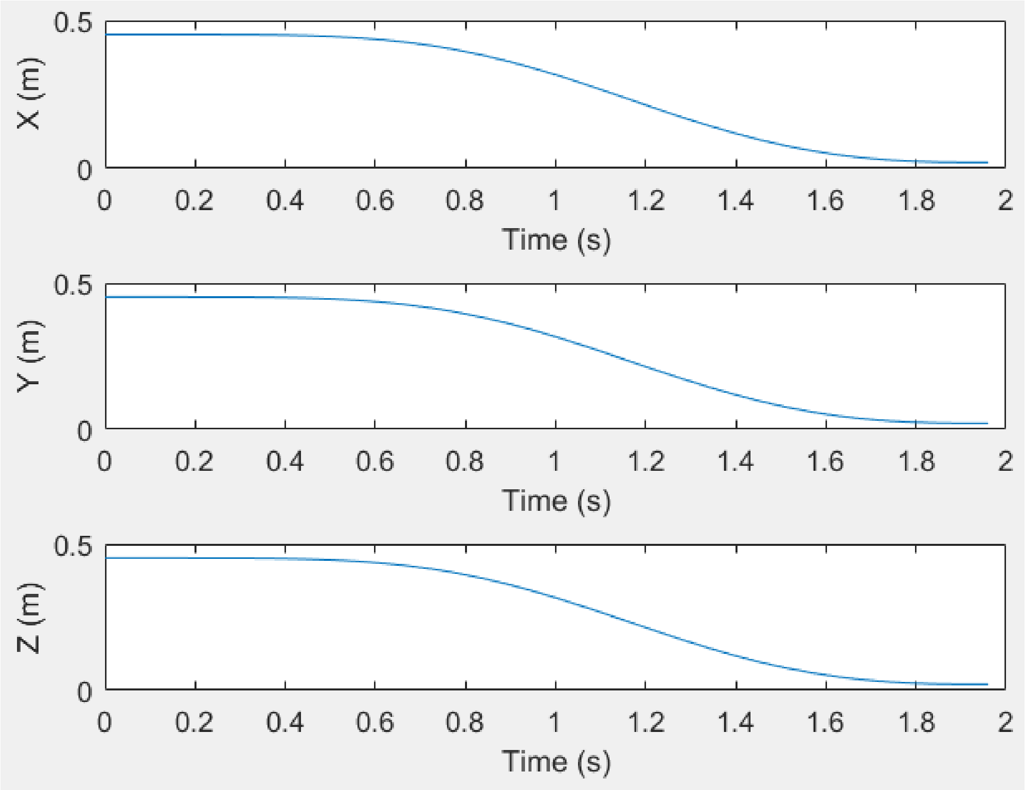
*According to the author, the Toolbox is tested with MATLAB R2016b. I suggest you use the tested version.

If you find any strange bugs and have no clue after spending hours on it, you have several options:

- Although there is no official support, this google group works as a reliable reference and an active online community, try to find the answer there first: https://groups.google.com/forum/#!forum/robotics-tool-box
- You are encouraged to ask for help on our course CCLE forum, hopefully your knowledgeable classmates could give you a hand.
- You may also send emails to the course TA (yangshen@ucla.edu), Yang would try to come up with a brief solution in 1-2 business days.

One bug example

- When you run rtbdemo, click Robot->Forward kinematics
- Everything works fine until…



Easy to find out that all subplots are plotting the x-t relationship.

You may choose to change the code in the demo libraries.

```
>> subplot(3,1,1)

>> plot(t, p(:,1))

>> xlabel('Time (s)');

>> ylabel('X (m)')

>> subplot(3,1,2)

>> plot(t, p(:,1))

>> xlabel('Time (s)');

>> ylabel('Y (m)')

>> subplot(3,1,3)

>> plot(t, p(:,1))

>> xlabel('Time (s)');

>> ylabel('Z (m)')
```

Another bug example

mdl_cobra600

Create model of Puma 560 manipulator

MDL_PUMA560 is a script that creates the workspace variable p560 which describes the kinematic and dynamic characteristics of a Unimation Puma 560 manipulator using standard DH conventions.

Also define the workspace vectors:

qz	zero joint angle configuration
qr	vertical 'READY' configuration
qstretch	arm is stretched out in the X direction
qn	arm is at a nominal non-singular configuration

Notes

- SI units are used.
- The model includes armature inertia and gear ratios.

Reference

- "A search for consensus among model parameters reported for the PUMA 560 robot", P. Corke and B. Armstrong-Helouvry, Proc. IEEE Int. Conf. Robotics and Automation, (San Diego), pp. 1608-1613, May 1994.

Variables - c600

c600

1x1 SerialLink

Property	Value
name	'Cobra600'
gravity	[0;0;9.8100]
base	1x1 SE3
tool	1x1 SE3
manufacturer	'Adept'
comment	''
plotopt	1x2 cell
fast	0
interface	[]
ikineType	[]
trail	[]
movie	[]
delay	[]
loop	[]
model3d	[]
faces	[]
points	[]
plotopt3d	[]
n	4
links	1x4 Link
mdh	0
T	[]
config	'RRPR'
offset	[0,0,0,0]
qlim	[-0.8727,0.8727;-...
d	[0.3870,0,0]
a	[0.3250,0.2750,0,0]
alpha	[0,3.1416,0,0]
theta	0

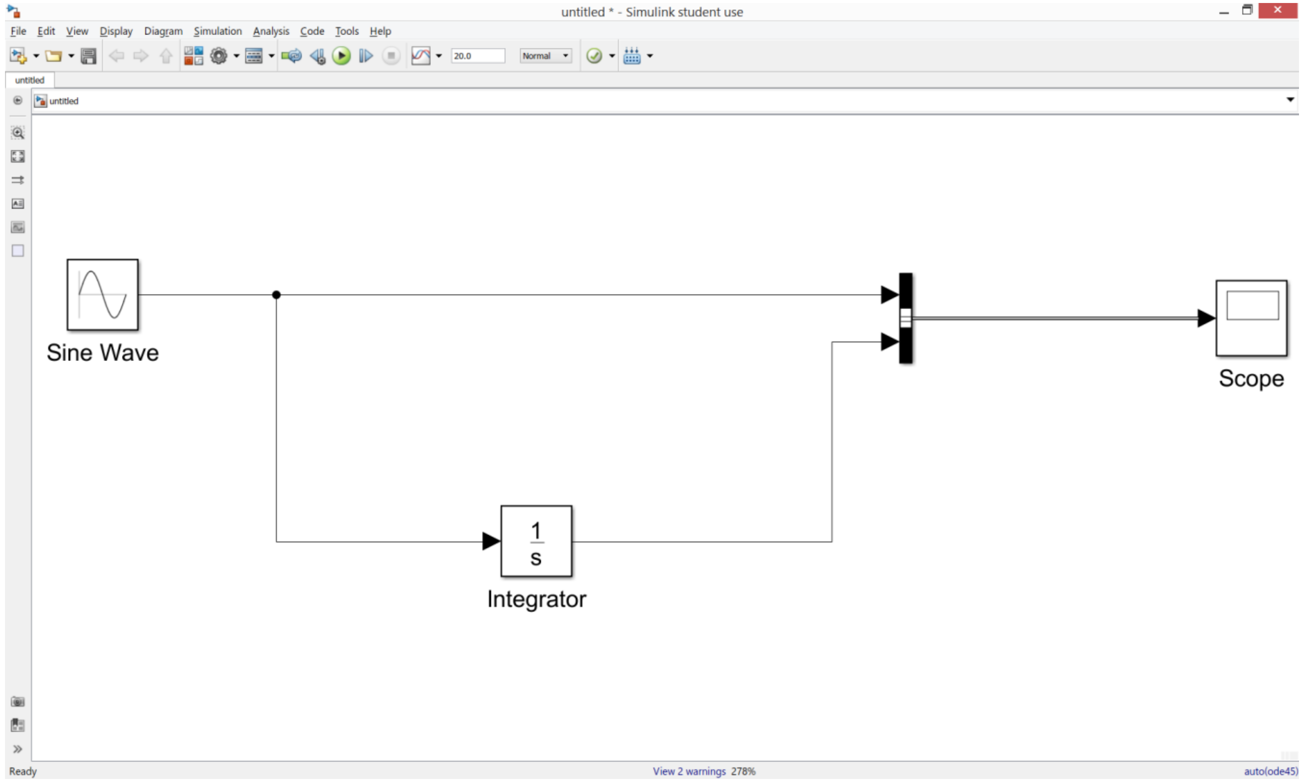
Workspace

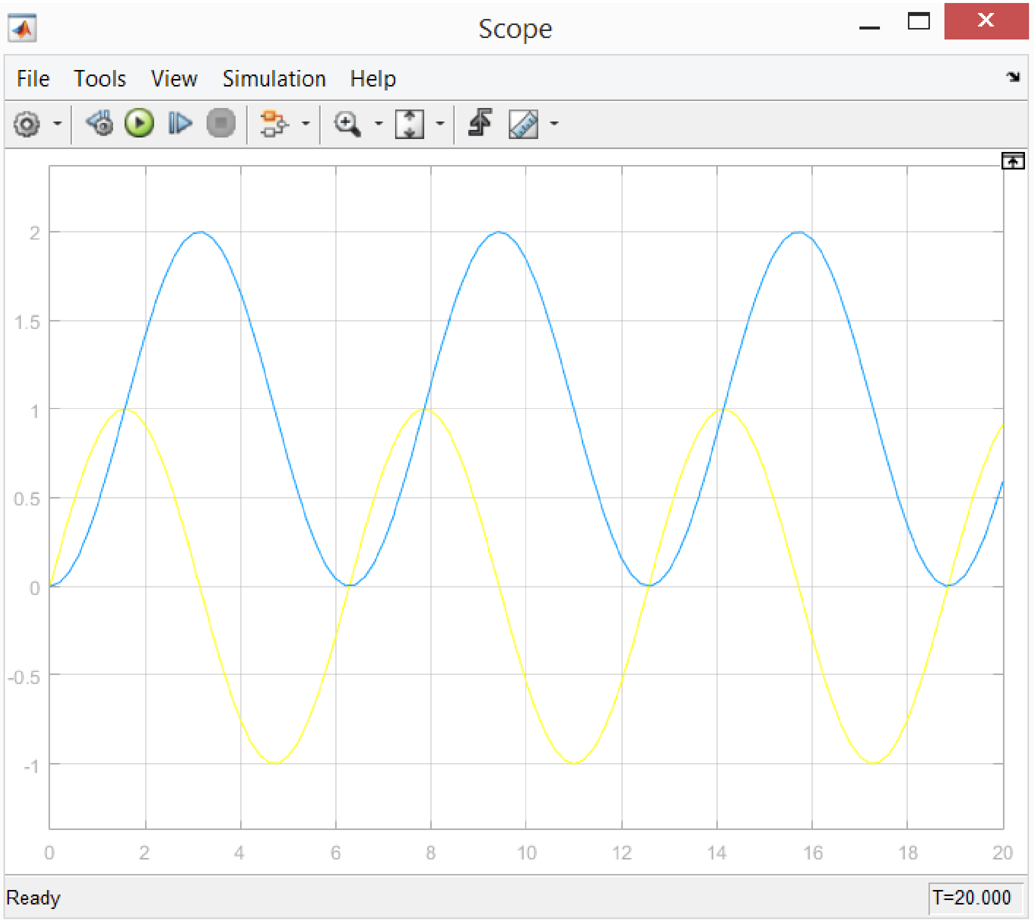
Name	Value
c600	1x1 SerialLink
links	1x4 Link
qz	[0,0,0,0]

Intro to Simulink

- Simulink: a graphical programming environment for modeling, simulating and analyzing multidomain dynamical systems
- MathWorks provides videos, examples, and tutorials on getting started with Simulink. You are encouraged to read and practice with following official links.
- https://www.mathworks.com/products/simulink/getting-started.html
- This is a good reference to show what MATLAB and Simulink could do in robotics: https://www.mathworks.com/solutions/robotics.html

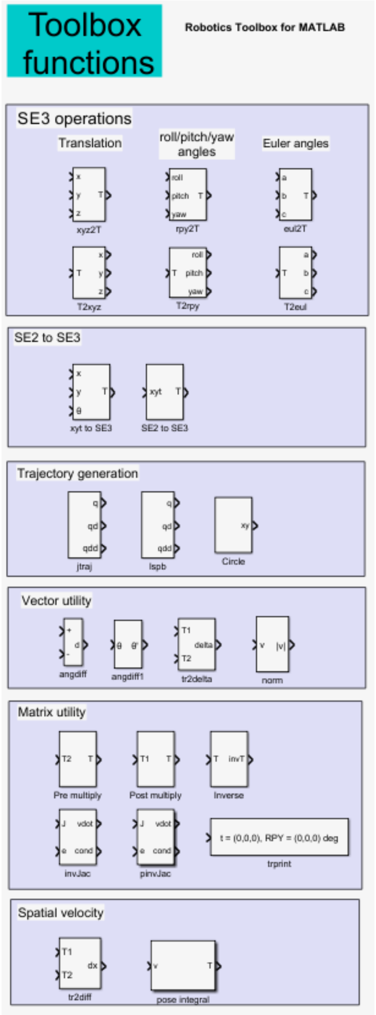
Simulink example

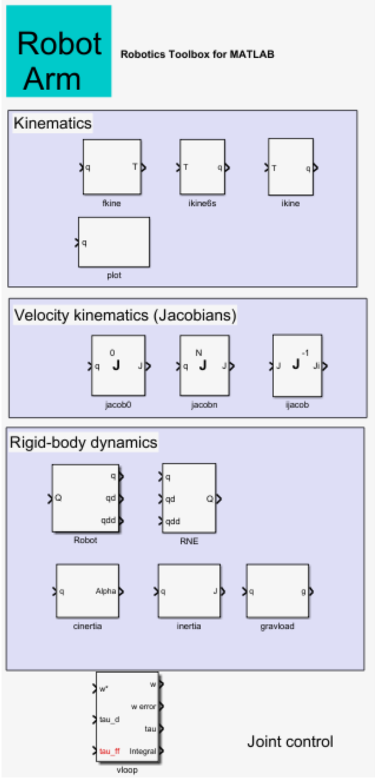


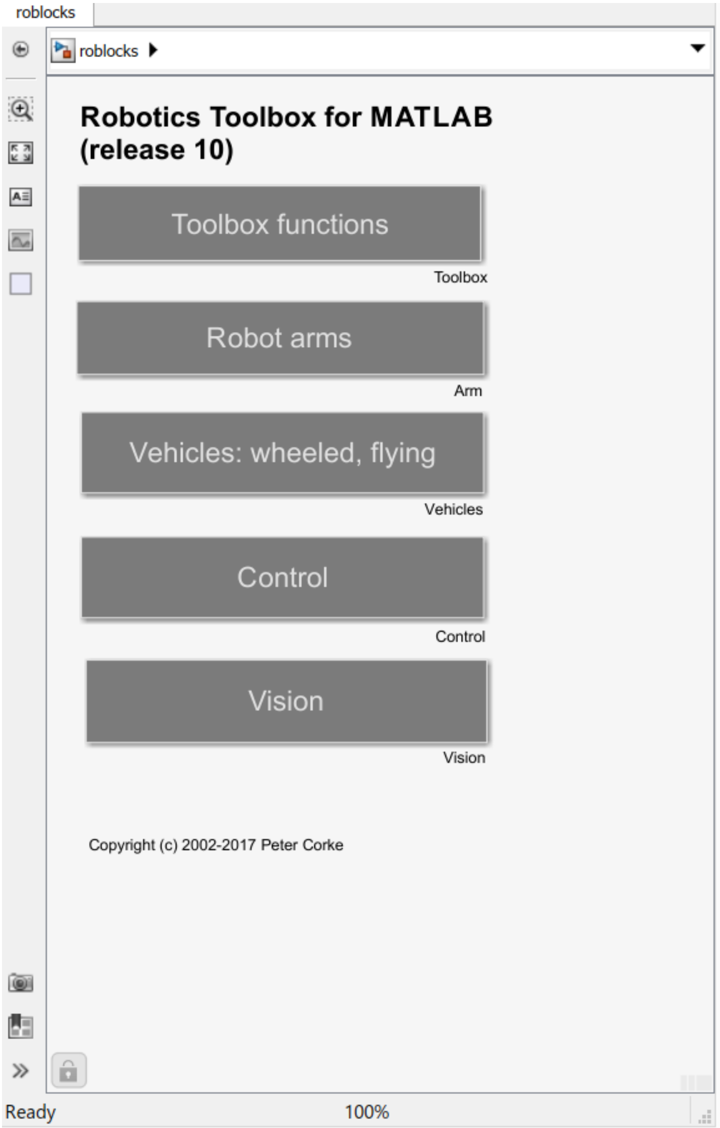


Some useful shortcuts: Ctrl, Shift, spacebar, right click

Simulink with RTB – library

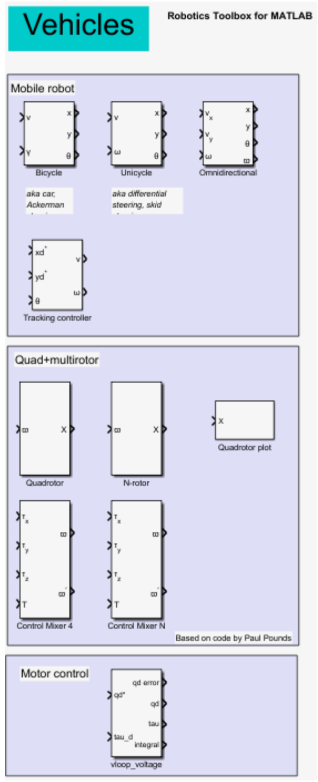


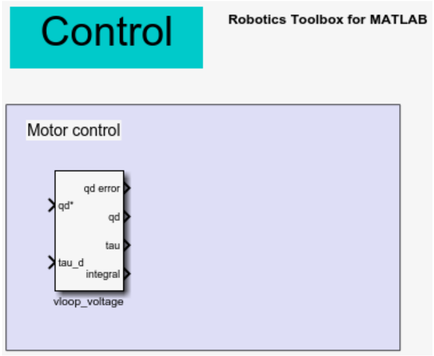


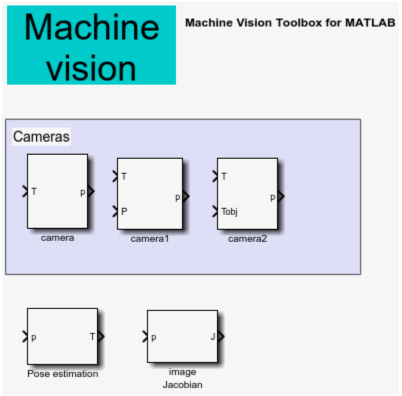



```
roblocks:
```

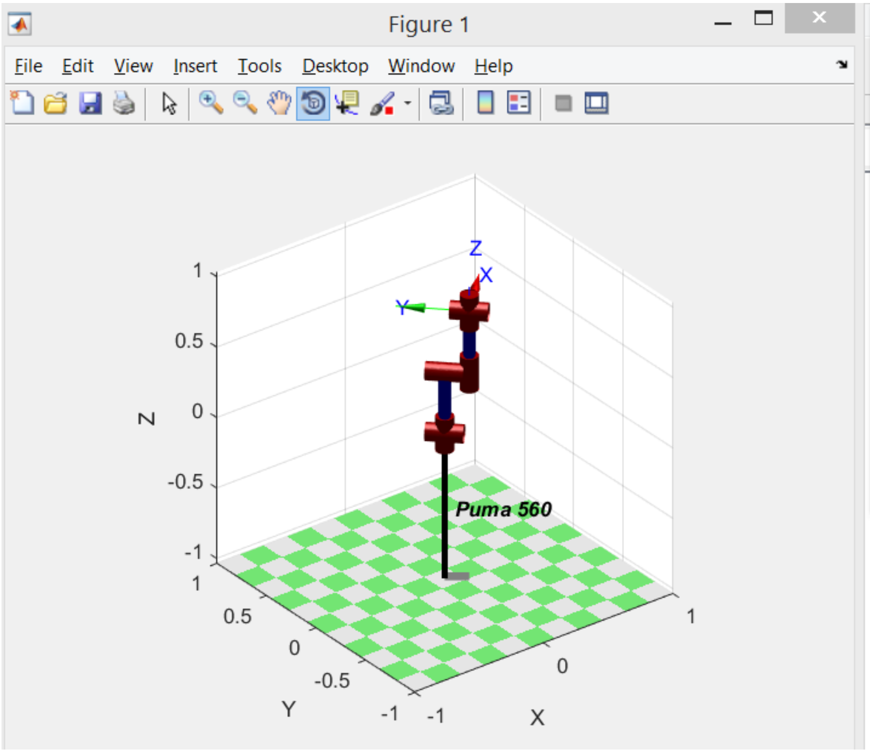
Only works with R2016b

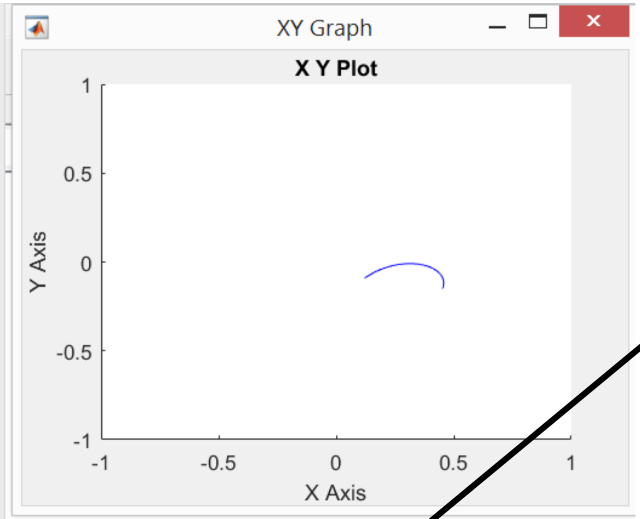


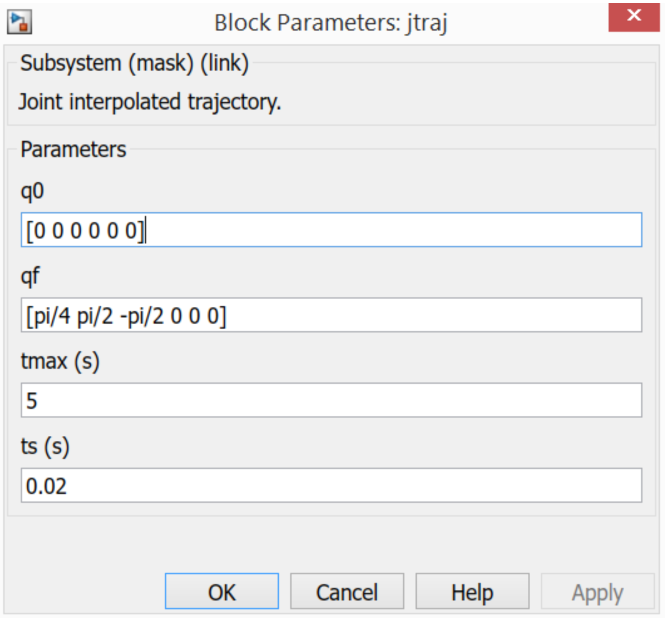


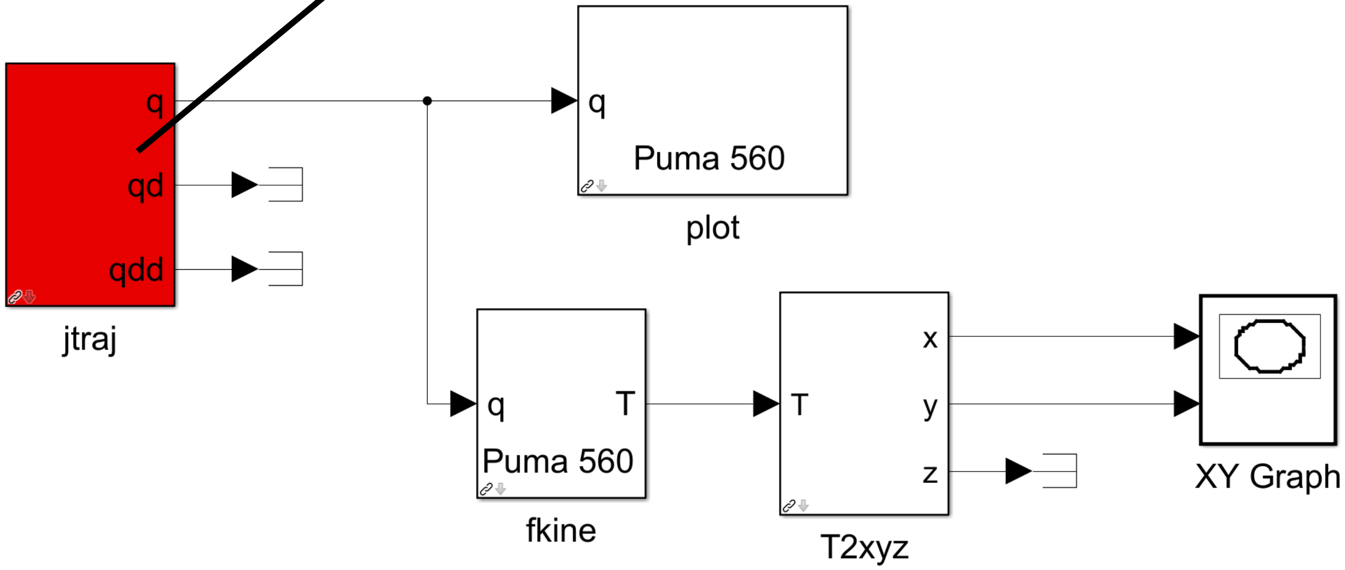


Simulink with RTB – example 1









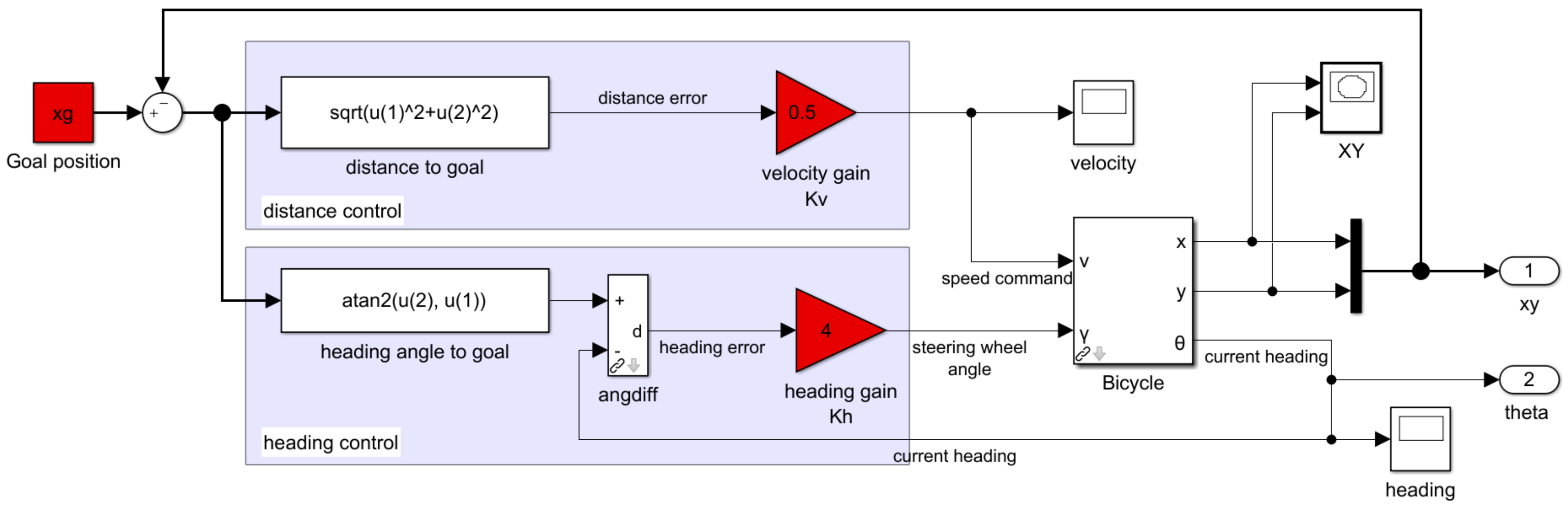

```
sl_jspace:
```

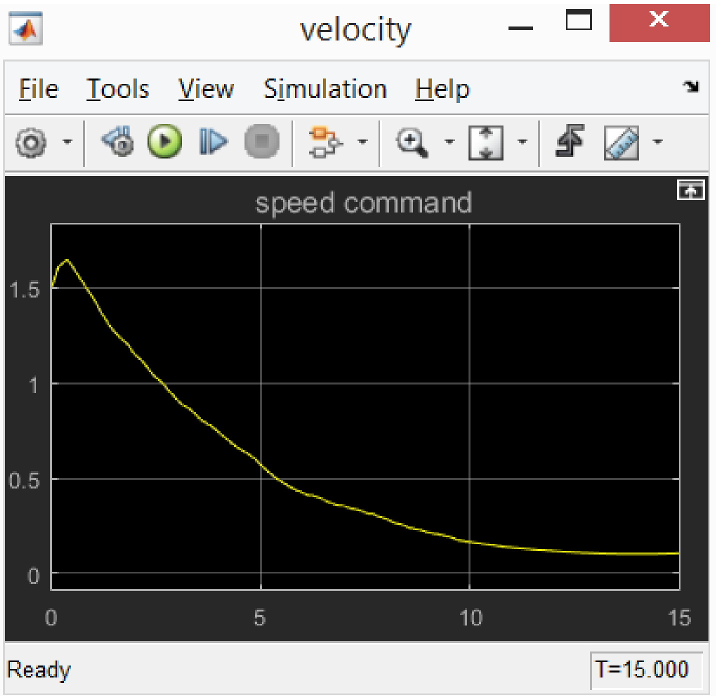
Joint space control

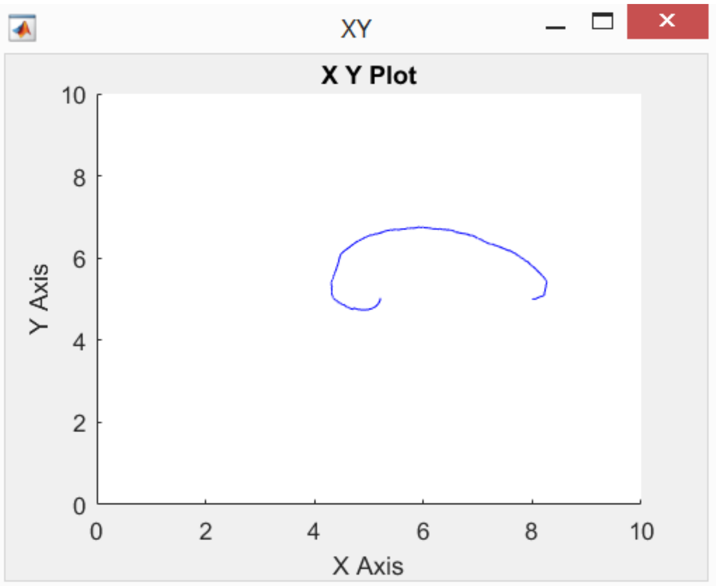
Simulink with RTB – example 2

`sl_drivepoint:`

Drive to a point







Simulink with RTB – example 3 (RVC #4.2)

`sl_quadrotor:`

Quadrotor control