MAE263B: Dynamics of Robotic Systems
Discussion Section – Week3
: Robotics Toolbox for MATLAB

Seungmin Jung

01.24.2020.

UCLA

Contents

- **Robotics Toolbox for MATLAB**
  - Installation
  - Robotics Toolbox with examples
    - Modified DH parameter / Standard DH parameter
    - Forward Kinematics/ Transformation Matrix
    - Plot / Simulation – Animation
  - Useful functions in Robotics Toolbox and MATLAB
  - Review with PUMA 560

Overview of Robotics ToolBox

- The Robotics Toolbox provides many functions that are required in robotics and addresses areas such as kinematics, dynamics, and trajectory generation.

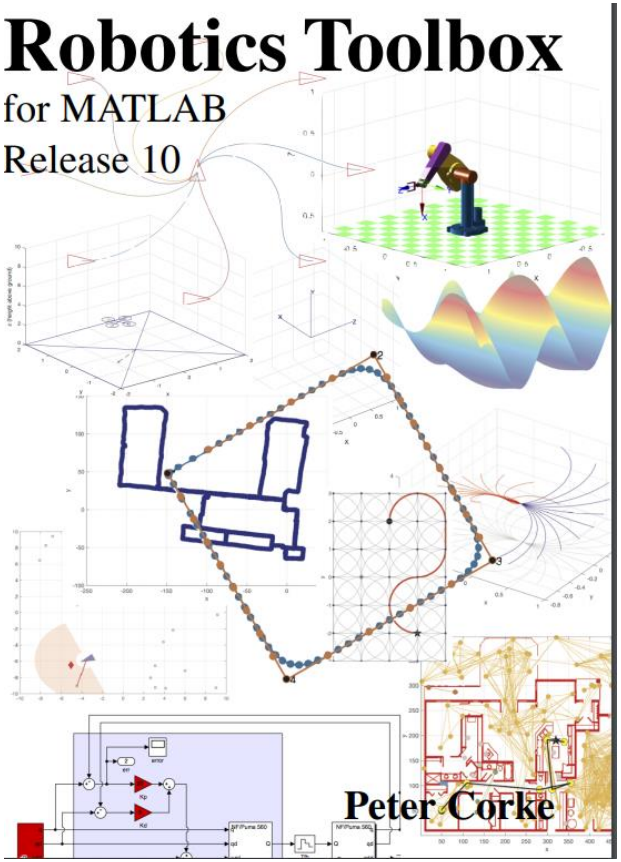
- The Toolbox uses a very general method of representing the kinematics and dynamics of serial-link manipulators as MATLAB® objects – robot objects can be created by the user for any serial-link manipulator and a number of examples are provided for well known robots from Kinova, Universal Robotics, Rethink as well as classical robots such as the Puma 560 and the Stanford arm.

- The Toolbox is useful for simulation as well as analyzing results from experiments with real robots, and can be a powerful tool for education.

Robotics Toolbox – Manual pdf file

Functions by category

- 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



Download the manual pdf file: click here

UCLA

Contents

- Robotics Toolbox for MATLAB
  - ✓ Installation
  - Example of PUMA 560
  - HW1 Examples
  - Examples of HW and Exam

Robotic Toolbox for MATLAB – Installation

- **Download the file: MATLAB**

https://softwarecentral.ucla.edu/matlab-getmatlab

MATLAB – Install and activate

https://softwarecentral.ucla.edu/matlab-getmatlab

- License can be obtained by generating an account **with UCLA email address**

How to Get MATLAB

MATLAB software is available as a self-service installation for individual user computers. If you have installed MATLAB in the past, please re-download and install to access the full set of tools available under the campus license.

How to get MATLAB for Single User

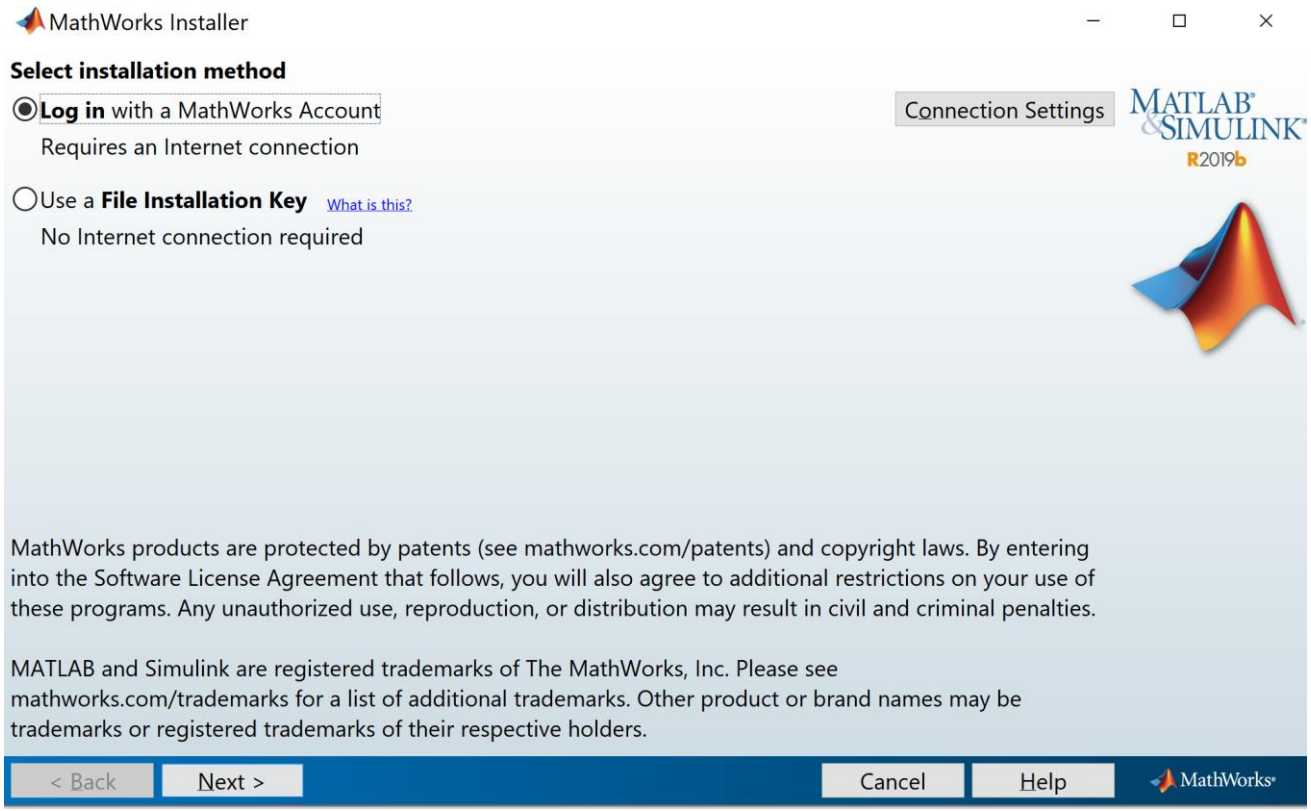
1. Visit the UCLA MATLAB Portal
Select 'Sign in to get started' under the Get MATLAB and Simulink section.
2. Create a MathWorks account using your UCLA email address which must have ".ucla.edu" as part of the domain. The MATLAB license is only for current UCLA Students, Faculty and Staff with an ucla.edu email address. (You may be asked to login with your UCLA BOL account to validate your eligibility first.)
3. Log in to your MathWorks account.
4. Click the download button for the current release (you can also download previous releases here).
5. Choose a supported platform and download the installer.

Install and activate

1. Run the installer
2. In the installer, select Log in with a MathWorks Account and follow the online instructions.
3. When prompted to do so, select the UCLA site-wide license.
4. Select the products you want to download and install.
5. After downloading and installing your products, keep the Activate MatLab checkbox selected and click "Next".
6. Follow the prompts to activate MATLAB.

MATLAB – Install and activate

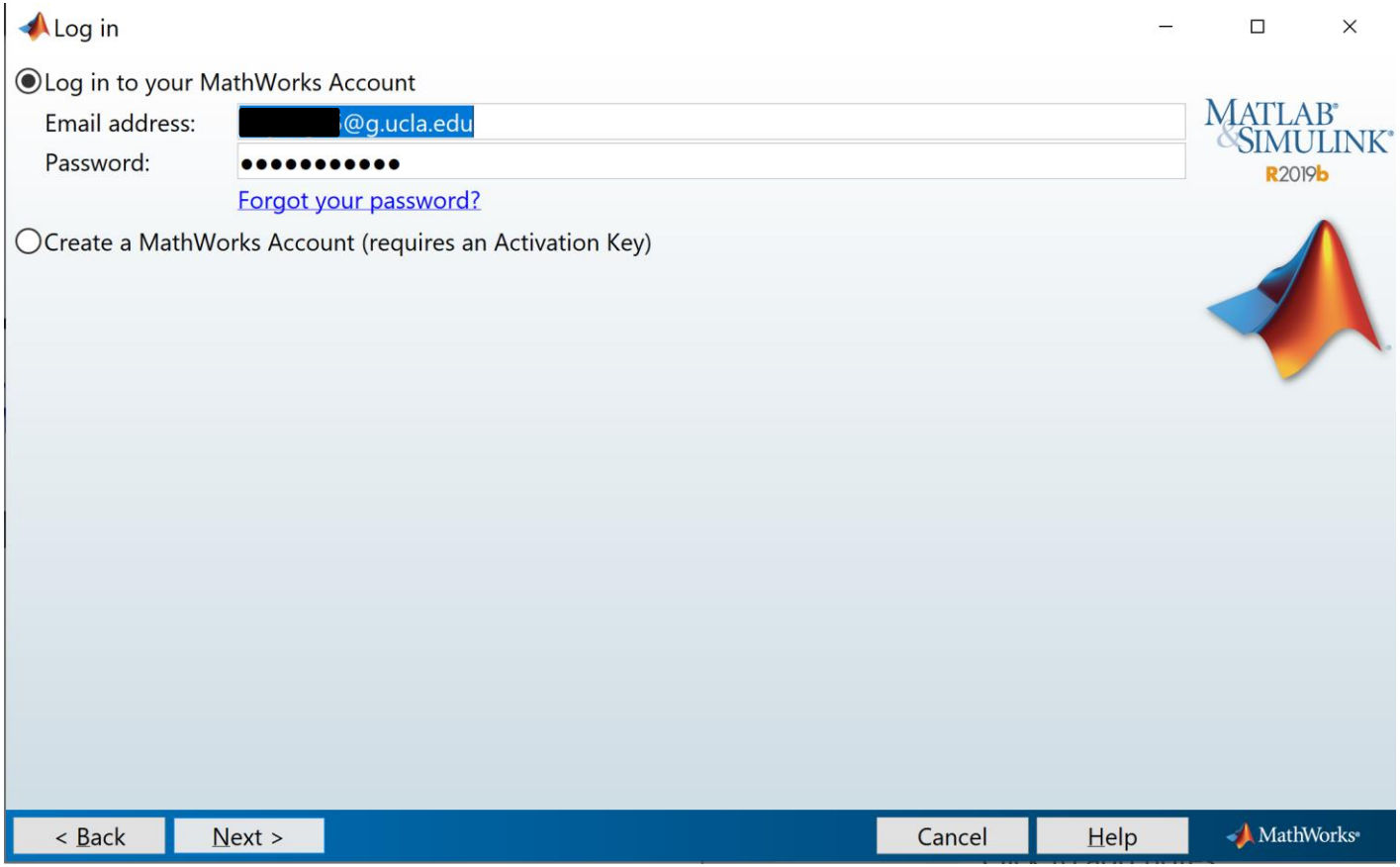
Log in with a MathWorks Account



UCLA

MATLAB – Install and activate

Create an account with **the UCLA email address** to get the license



UCLA

MATLAB – Install and activate

License Selection

Select a license or enter an Activation Key

The installer will determine which products to install based on your license.

MATLAB & SIMULINK R2019b

◉ Select a license:

License	Label	Option
[redacted]	Individual	Academic - Total Headcount

○ Enter an Activation Key for a license not listed:

You may have received the Activation Key from the Administrator of the license.

< Back | Next > | Cancel | Help | MathWorks

UCLA

MATLAB – Install and activate

Product Selection

Select products to install (recommended products are preselected)

Product	Notes
MATLAB 9.7	Download Required
Simulink 10.0	Download Required
5G Toolbox 1.2	Download Required
Aerospace Blockset 4.2	Download Required
Aerospace Toolbox 3.2	Download Required
Antenna Toolbox 4.1	Download Required
Audio Toolbox 2.1	Download Required
Automated Driving Toolbox 3.0	Download Required
AUTOSAR Blockset 2.1	Download Required
Bioinformatics Toolbox 4.13	Download Required
Communications Toolbox 7.2	Download Required
Computer Vision Toolbox 9.1	Download Required
Control System Toolbox 10.7	Download Required
Curve Fitting Toolbox 3.5.10	Download Required
Data Acquisition Toolbox 4.0.1	Download Required
Database Toolbox 9.2	Download Required
Datafeed Toolbox 5.9	Download Required
Deep Learning Toolbox 13.0	Download Required
DSP System Toolbox 9.9	Download Required

MATLAB & SIMULINK R2019b

< Back | Next > | Cancel | Help | MathWorks

UCLA

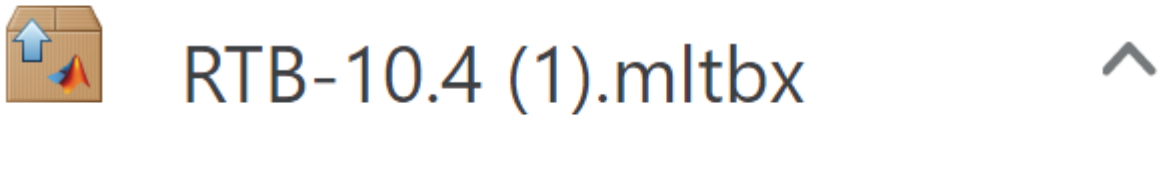
Robotics Toolbox for MATLAB – Installation

- **Download the file: Robotics Toolbox**

http://petercorke.com/wordpress/toolboxes/robotics-toolbox

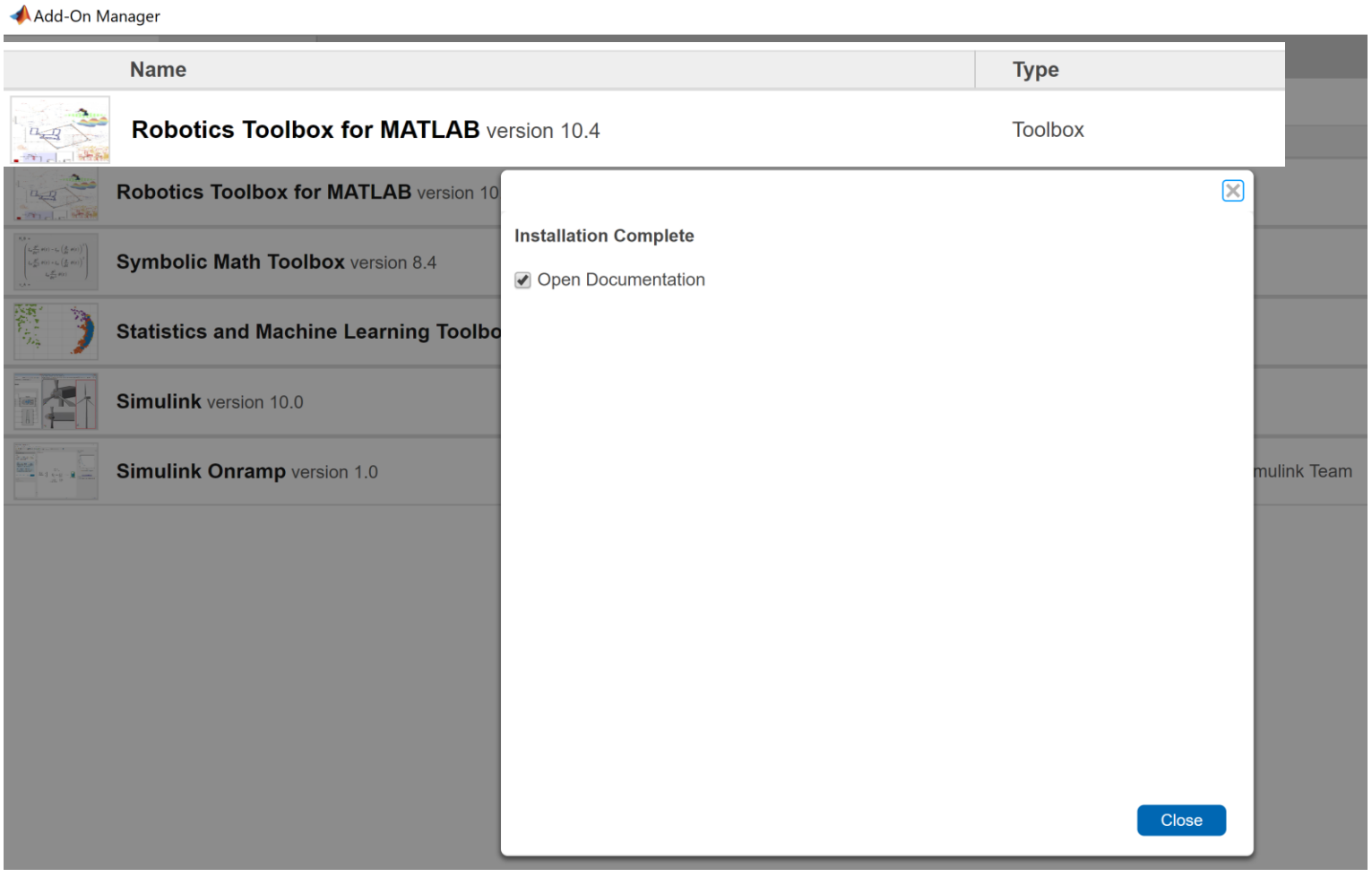
Install from .mltbx file

1. Download one of the following files:
   - RTB-10.1.mltbx (June 11, 2017)
   - RTB-10.2.mltbx (November 13, 2017)
   - RTB-10.3.1.mltbx (August 20, 2018)
   - RTB-10.4.mltbx (October 14, 2019)
2. From within the MATLAB file browser double click on this file, it will install and configure the paths correctly
3. Run the demo rtbdemo to see what it can do



UCLA

Robotics Toolbox for MATLAB – Installation



UCLA

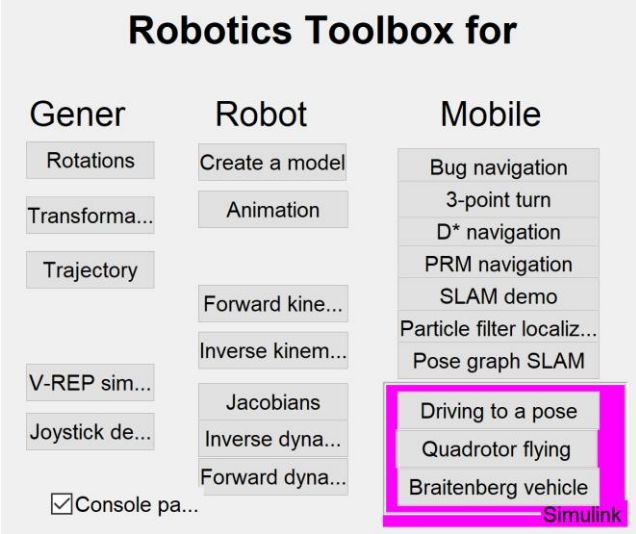
Robotics Toolbox for MATLAB – Installation Confirm

- In order to make sure the Robotics Toolbox is installed,
  go to the Command Window in MATLAB and then type “rtbdemo”

- You can get the following message and the popup window.

Command Window

```
>>  rtbdemo
------------------------------------------------------------
Many of these demos print tutorial text and MATLAB commmands
in the console window.  Read the text and press <enter> to move
on to the next command. At the end of the tutorial you can
choose the next one from the graphical menu, or close the menu
window.
------------------------------------------------------------
```



UCLA

MATLAB file

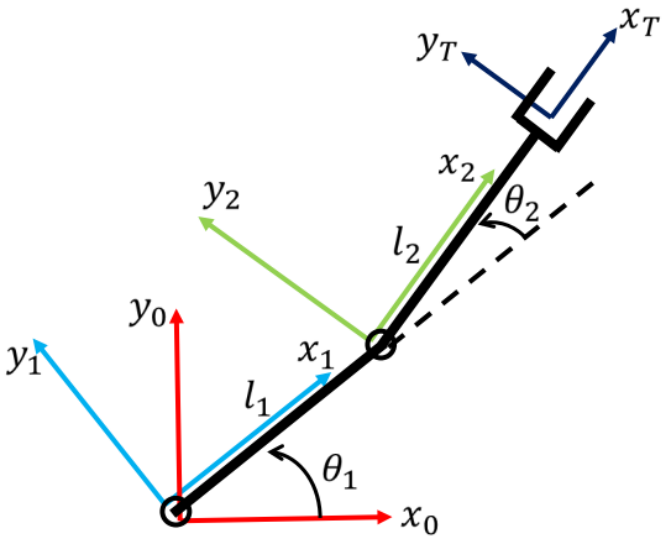
Examples in the Robotics Toolbox

MAE263B _ Dynamics of Robotics

Discussion Section Date: 01.24.2020

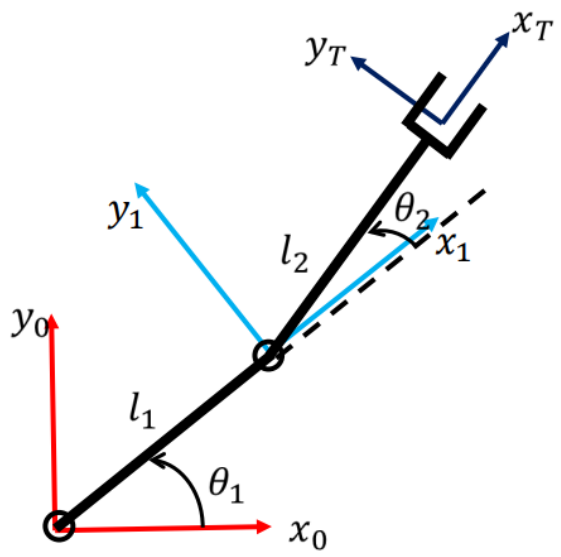
DH parameter in the Robotic Toolbox – homework 1

2D – 1 – RR (modified DH)



i-1	*i*	α_{i-1}	a_{i-1}	d_i	θ_i
0	1	0	0	0	θ_1
1	2	0	l_1	0	θ_2
2	3=T	0	l_2	0	0

2D – 1 – RR (standard DH)



i-1	*i*	α_i	a_i	d_i	θ_i
0	1	0	l_1	0	θ_1
1	2=T	0	l_2	0	θ_2

UCLA

DH parameters – Standard/ Modified approach Review

If the link frame have been attached to the links according to our convention, the following definitions of the DH parameters are valid:

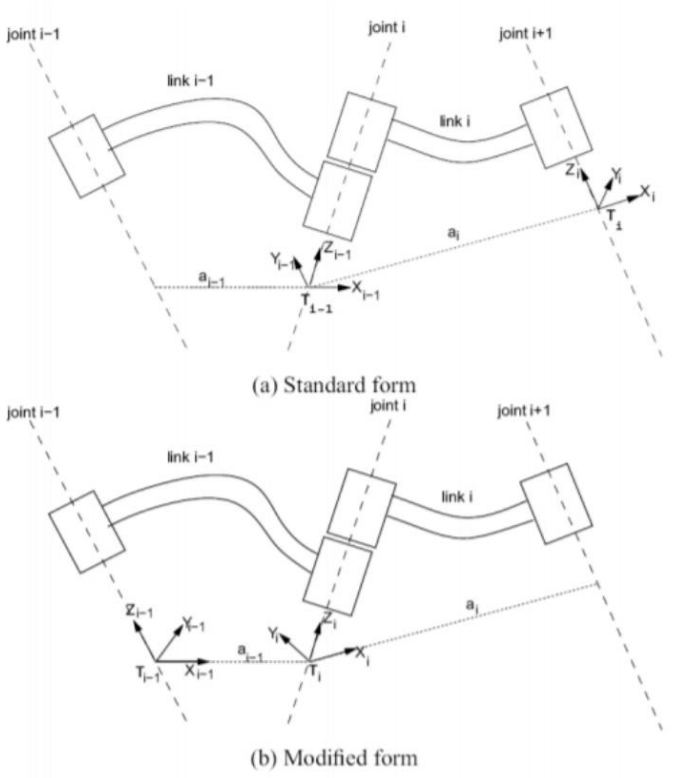
Standard form:

- a_i - The distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_i$
- α_i - The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_i$
- d_i - The distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_{i-1}$
- θ_i - The angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_{i-1}$

Modified form:

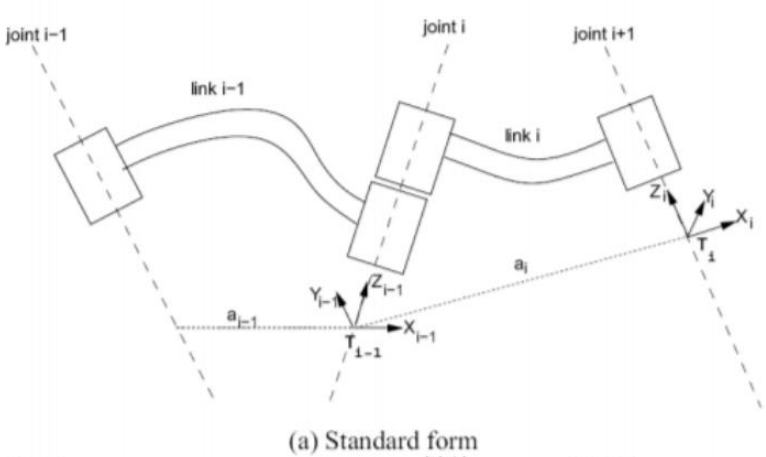
- a_{i-1} - The distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_{i-1}$
- α_{i-1} - The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_{i-1}$
- d_i - The distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$
- θ_i - The angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$

Note: $a_i \geq 0$ α_i d_i θ_i are signed quantities

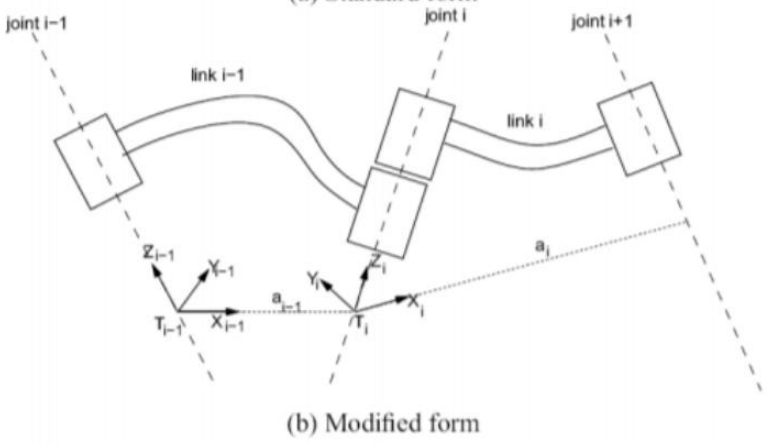


UCLA

DH parameters – Standard/ Modified approach Review



(a) Standard form



(b) Modified form

Standard Form

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

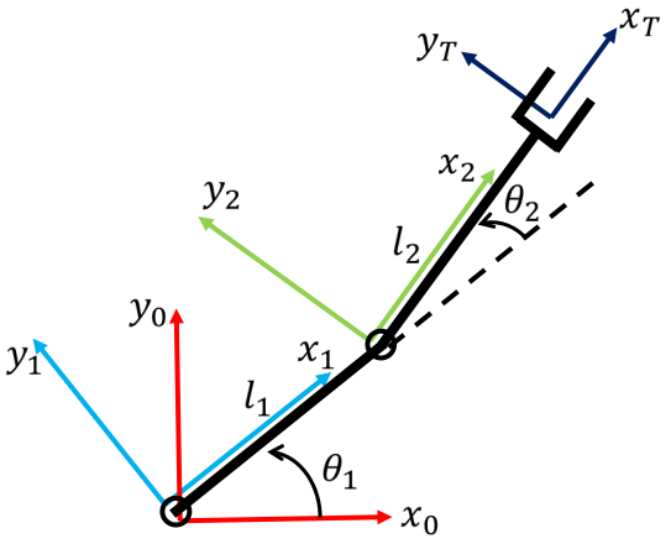
Modified Form

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

UCLA

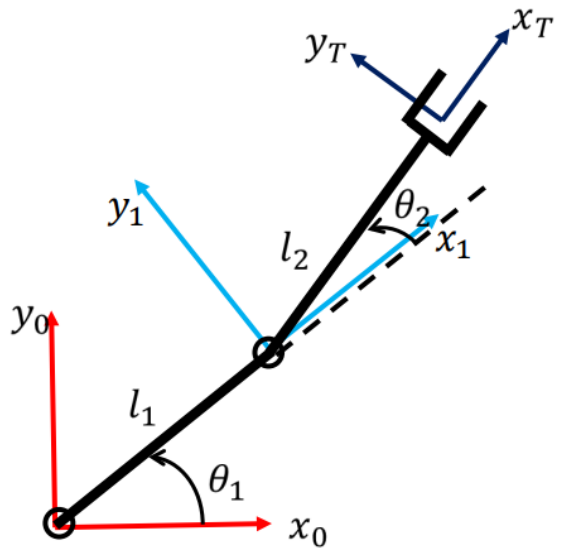
DH parameter in the Robotic Toolbox – homework 1&2

2D – 1 – RR (modified DH)



i-1	*i*	α_{i-1}	a_{i-1}	d_i	θ_i
0	1	0	0	0	θ_1
1	2	0	l_1	0	θ_2
2	3=T	0	l_2	0	0

2D – 1 – RR (standard DH)



i-1	*i*	α_i	a_i	d_i	θ_i
0	1	0	l_1	0	θ_1
1	2=T	0	l_2	0	θ_2

UCLA

Link: Robot manipulator Link class

A Link object holds all information related to a robot link such as kinematics parameters, rigid-body inertial parameters, motor and transmission parameters.

Examples

```
L = Link([0 1.2 0.3 pi/2]);
L = Link('revolute', 'd', 1.2, 'a', 0.3, 'alpha', pi/2);
L = Revolute('d', 1.2, 'a', 0.3, 'alpha', pi/2);
```

Properties (read/write)

theta	kinematic: joint angle
d	kinematic: link offset
a	kinematic: link length
alpha	kinematic: link twist
sigma	kinematic: 0 if revolute, 1 if prismatic
mdh	kinematic: 0 if standard D&H, else 1
offset	kinematic: joint variable offset
qlim	kinematic: joint variable limits [min max]

A	link transform matrix
RP	joint type: 'R' or 'P'

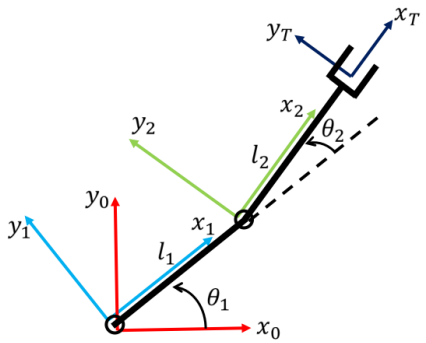
Link reference

Modified DH parameter

Modified DH parameter

```
fprintf('* ========== Modified DH parameter ========== *')
syms l1 l2  % Create symbolic variables and functions

L1 = Link('revolute','d',0,'a',0,'alpha',0,'modified')
%: cannot specify 'theta' for primatic link
%: cannot specify 'd' for primatic link
L1_2 = Link([0 0 0 0],'modified')
%: Link([theta, d, a, alpha, offset],'modified/standard');

L2 = Link('revolute','d',0,'a',l1,'alpha',0,'modified')

L3 = Link('d',0,'a',l2,'alpha',0,'modified') %default: revolute
L3_2 = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
L3_p = Link('prismatic','a',l2,'alpha',0,'theta', 0,'modified')

Robot_Mod_DH = SerialLink([L1 L2 L3], 'name', '2D-RR Modified DH parameter')
```

```
* ========== Modified DH parameter ========== *


L1 =
Revolute(mod): theta=q, d=0, a=0, alpha=0, offset=0

L1_2 =
Revolute(mod): theta=q, d=0, a=0, alpha=0, offset=0

L2 =
Revolute(mod): theta=q, d=0, a=l1, alpha=0, offset=0

L3 =
Revolute(mod): theta=q, d=0, a=l2, alpha=0, offset=0

L3_2 =
Revolute(mod): theta=q, d=0, a=l2, alpha=0, offset=0

L3_p =
Prismatic(mod): theta=0, d=q, a=l2, alpha=0, offset=0

Robot_Mod_DH =

2D-RR Modified DH parameter:: 3 axis, RRR, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|          0|          0|
|  2|         q2|          0|         l1|          0|          0|
|  3|         q3|          0|         l2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```



i-1	*i*	α_{i-1}	a_{i-1}	d_i	θ_i
0	1	0	0	0	θ_1
1	2	0	l_1	0	θ_2
2	3=T	0	l_2	0	0

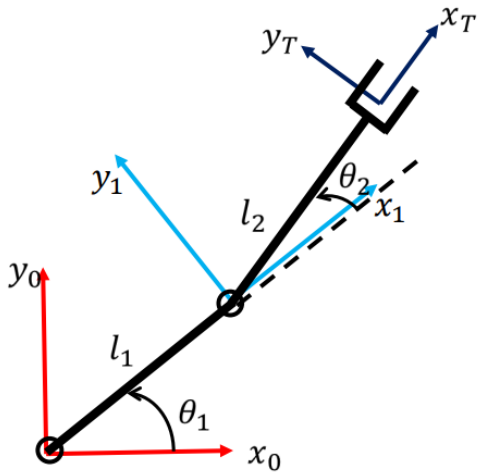
UCLA

Standard DH parameter

Standard DH parameter

```
fprintf('* ========== Standard DH parameter ========== *')
L_S(1) = Link('revolute','d',0,'a',l1,'alpha',0,'standard')
L_S(2)= Link('revolute','d',0,'a',l2,'alpha',0,'standard')

Robot_Stand_DH = SerialLink(L_S, 'name', '2D-RR Standard DH parameter')
% Robot_Standard_DH = SerialLink([L1 L2], 'name', '2D-RR-Standard')
% L_S = [L1, L2] <==  L_S(1)=L1, L_S(2)=L2
```

```
* ========== Standard DH parameter ========== *

L_S =
Revolute(std): theta=q, d=0, a=l1, alpha=0, offset=0

L_S =
Revolute(std):  theta=q1   d=0          a=l1 alpha=0          offset=0
Revolute(std):  theta=q2   d=0          a=l2 alpha=0          offset=0

Robot_Stand_DH =

2D-RR Standard DH parameter:: 2 axis, RR, stdDH, slowRNE, Symbolic
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|         l1|          0|          0|
|  2|         q2|          0|         l2|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```



i-1	*i*	α_i	a_i	d_i	θ_i
0	1	0	l_1	0	θ_1
1	2=T	0	l_2	0	θ_2

UCLA

Comparison - Modified vs Standard DH parameter

```
* ========== Comparison of DH parameter ========== *

Robot_Mod_DH =

2D-RR Modified DH parameter:: 3 axis, RRR, modDH, slowRNE
+---+----------+----------+----------+----------+----------+
| j |    theta |        d |        a |    alpha |   offset |
+---+----------+----------+----------+----------+----------+
|  1|        q1|         0|         0|         0|         0|
|  2|        q2|         0|        l1|         0|         0|
|  3|        q3|         0|        l2|         0|         0|
+---+----------+----------+----------+----------+----------+


Robot_Stand_DH =

2D-RR Standard DH parameter:: 2 axis, RR, stdDH, slowRNE, Symbolic
+---+----------+----------+----------+----------+----------+
| j |    theta |        d |        a |    alpha |   offset |
+---+----------+----------+----------+----------+----------+
|  1|        q1|         0|        l1|         0|         0|
|  2|        q2|         0|        l2|         0|         0|
+---+----------+----------+----------+----------+----------+
```

UCLA

Forward Kinematics

Forward Kinematics

- Robotname.fkine()

```
fprintf('* ========== Foward Kinematics ========== *')
t3 = 0;
th_Mod = [t1,t2,t3]
th_DH = [t1 t2]

fprintf('*** Foward Kinematics from Modified DH parameter')
Mod_DH_FK = Robot_Mod_DH.fkine(th_Mod)

fprintf('*** Foward Kinematics from Standard DH parameter')
Stand_DH_FK = Robot_Stand_DH.fkine(th_DH)
```

```
* ========== Foward Kinematics ========== *
```

th_Mod = $\begin{pmatrix} t_1 & t_2 & 0 \end{pmatrix}$

th_DH = $\begin{pmatrix} t_1 & t_2 \end{pmatrix}$

```
*** Foward Kinematics from Modified DH parameter
```

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
*** Foward Kinematics from Standard DH parameter
```

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Homogeneous Transformation Matrix

Homogeneous Transformation Matrix

- Robotname.A()

```
fprintf('* ========== Homogeneous Transformation Matri
t3 = 0;
th_Mod = [t1 t2 t3] %t3=0
th_Stand = [t1 t2]

fprintf('* T0_1_Mod')
T0_1_Mod = Robot_Mod_DH.A([1],th_Mod)
fprintf('* T0_1_Stand')
T0_1_Stand = Robot_Stand_DH.A([1],th_Stand)


fprintf('* T1_2_Mod')
T1_2_Mod = Robot_Mod_DH.A([2],th_Mod)
fprintf('* T1_2_Stand')
T1_2_Stand = Robot_Stand_DH.A([2],th_Stand)
fprintf('* T2_3_Mod')
T2_3_Mod = Robot_Mod_DH.A([3],th_Mod)

fprintf('* T0_2_Mod')
T0_2_Mod = Robot_Mod_DH.A([1 2],th_Mod)
fprintf('* T0_2_Stand')
T0_2_Stand = Robot_Stand_DH.A([1 2],th_Stand) ;
sim_T0_2_Stand = simplify(T0_2_Stand)
fprintf('* T0_3_Mod')
T0_3_Mod = simplify(Robot_Mod_DH.A([1 2 3],th_Mod))
```

* T0_1_Mod

$$\begin{pmatrix} \cos(t_1) & -\sin(t_1) & 0 & 0 \\ \sin(t_1) & \cos(t_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T0_1_Stand

$$\begin{pmatrix} \cos(t_1) & -\sin(t_1) & 0 & l_1\cos(t_1) \\ \sin(t_1) & \cos(t_1) & 0 & l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T1_2_Mod

$$\begin{pmatrix} \cos(t_2) & -\sin(t_2) & 0 & l_1 \\ \sin(t_2) & \cos(t_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T1_2_Stand

$$\begin{pmatrix} \cos(t_2) & -\sin(t_2) & 0 & l_2\cos(t_2) \\ \sin(t_2) & \cos(t_2) & 0 & l_2\sin(t_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T2_3_Mod

$$\begin{pmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T0_2_Mod

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T0_2_Stand

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

* T0_3_Mod

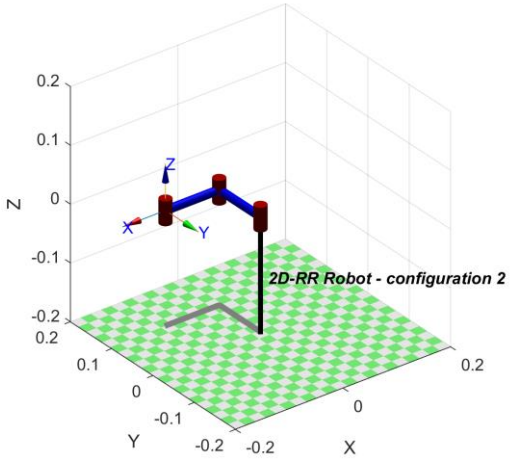
$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

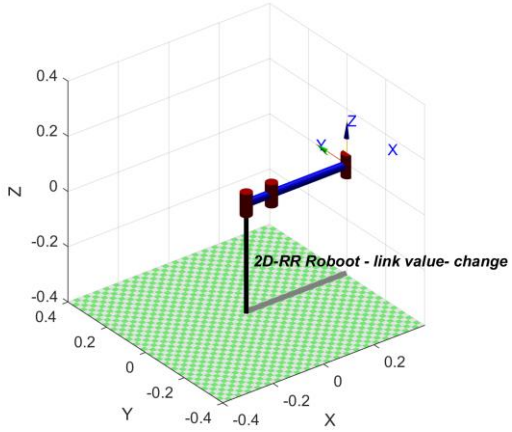
Plot robots

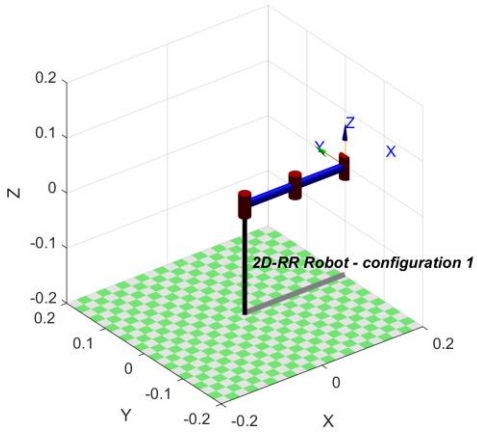
Plot robots

- Robotname.plot()

```
fprintf('* ========== Plot robots ========== *')
l1 = 0.1
l2 = 0.1
l2_2 = 0.3

L(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified') ;
L(2) = Link('revolute','d',0,'a',l1,'alpha',0,'modified');
L(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');

Robot_1 = SerialLink(L, 'name', '2D-RR Robot - configuration 1');
Robot_2 = SerialLink(L, 'name', '2D-RR Robot - configuration 2');

L2(1) = L(1); L2(2) = L(2);
L2(3) = Link('revolute','d',0,'a',l2_2,'alpha',0,'modified');
Robot_3 = SerialLink(L2, 'name', '2D-RR Roboot - link value- change')

figure()
Robot_1.plot([0 0 0]) %[t1 t2 t3]
figure()
Robot_2.plot([pi/2 pi/2 0])
figure()
Robot_3.plot([0 0 0])
```



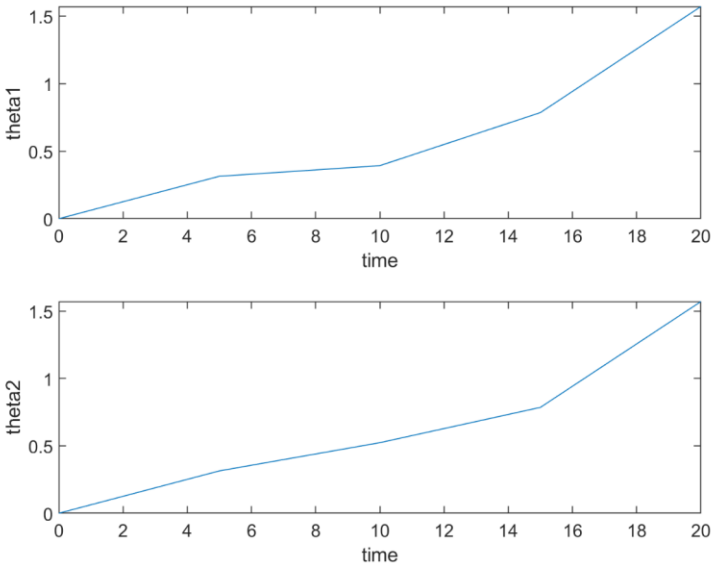


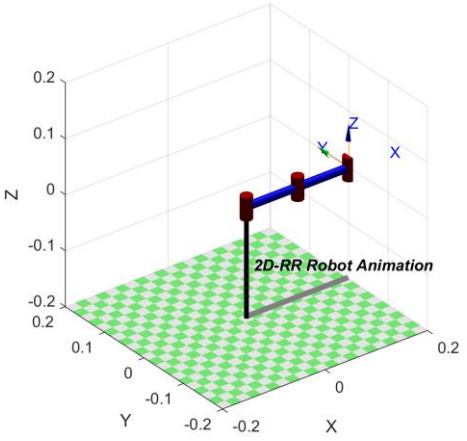


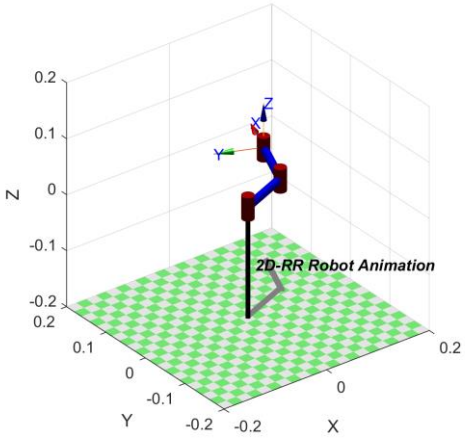
Simulation - Animation

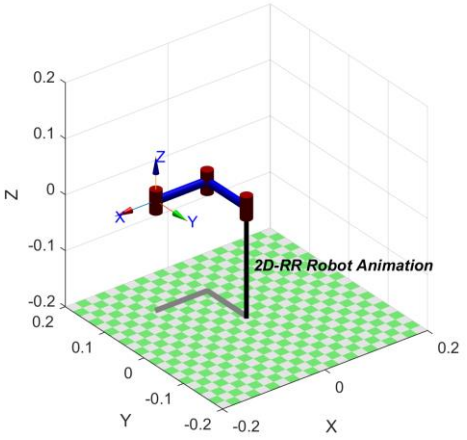
Simulation - Animation

```
l1 = 0.1;
l2 = 0.1; %0.3
L(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
L(2) = Link('revolute','d',0,'a',l1,'alpha',0,'modified');
L(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');

figure()
Robot = SerialLink(L, 'name', '2D-RR Robot Animation')

time = [0 5 10 15 20];
t1 = [0 pi/10 pi/8 pi/4 pi/2];
t2 = [0 pi/10 pi/6 pi/4 pi/2];

figure()
for i = 1:5
    Robot.plot([t1(i) t2(i) 0])
end

figure()
subplot(2,1,1)
plot(time,t1); xlabel('time'); ylabel('theta1');
subplot(2,1,2)
plot(time,t2); xlabel('time'); ylabel('theta2');
```









Useful functions in Robotics Toolbox and MATLAB

Useful functions in Robotics Toolbox

t2r : rotation matrix from the homogeneous matrix

.t : translation matrix from the homogeneous matrix

Useful functions in Robotics Toolbox and MATLAB

- Useful functions in Robotics Toolbox: tr2, .t :Extract the rotatioinal matrix and the position matrix

```
%t2r : the rotation matrix from the transformation matrix
%.t : the translation matrix from the transformation matrix
fprintf('* ========== Examples ========== *')
syms l1 l2 t1 t2 c1 c2

L(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
L(2) = Link('revolute','d',0,'a',l1,'alpha',0,'modified');
L(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
Robot = SerialLink(L, 'name', '2D-RR Robot1')

fprintf('* Tranformation matrix T')
T = Robot.fkine([t1 t2 0])
Rot_m = t2r(T)
tlans_m = T.t
```

```
* Tranformation matrix T
```

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
Rot_m =
```

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
tlans_m =
```

$$\begin{pmatrix} l_2\cos(t_1+t_2)+l_1\cos(t_1) \\ l_2\sin(t_1+t_2)+l_1\sin(t_1) \\ 0 \end{pmatrix}$$

UCLA

Useful functions in Robotics Toolbox and MATLAB

Useful function in MATLAB

Transpose
cross
simplify
collect
equationToMatrix

- Useful functions in MATLAB: **simplify** , collect, equationToMatrix, Transpose, cross

```
fprintf('* T0_3_Mod')
T0_3_Mod = Robot.A([1 2 3],th_Mod)
fprintf('* T0_3_Mod')
sim_T0_3_Mod = simplify(Robot_Mod_DH.A([1 2 3],th_Mod))

equation = c1*t1 + c2*t2 + c2*t1 + c2*t2
collect = collect(equation,[t1 t2])
e2m = equationsToMatrix(equation,[t1 t2])
```

* T0_3_Mod

$$\begin{pmatrix} \sigma_1 & -\cos(t_1)\sin(t_2) - \cos(t_2)\sin(t_1) & 0 & l_2\,\sigma_1 + l_1\cos(t_1) \\ \sigma_2 & \sigma_1 & 0 & l_2\,\sigma_2 + l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \cos(t_1)\cos(t_2) - \sin(t_1)\sin(t_2)$$

$$\sigma_2 = \cos(t_1)\sin(t_2) + \cos(t_2)\sin(t_1)$$

* T0_3_Mod

$$\begin{pmatrix} \cos(t_1+t_2) & -\sin(t_1+t_2) & 0 & l_2\cos(t_1+t_2) + l_1\cos(t_1) \\ \sin(t_1+t_2) & \cos(t_1+t_2) & 0 & l_2\sin(t_1+t_2) + l_1\sin(t_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

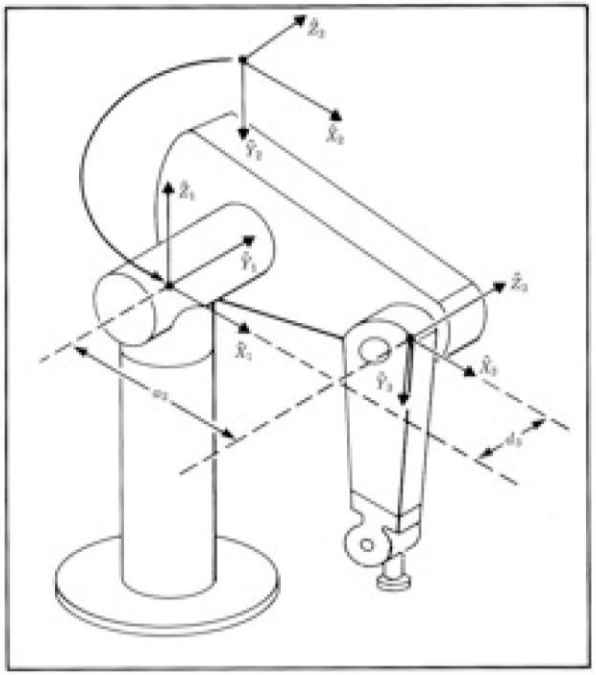
equation = $c_1\,t_1 + c_2\,t_1 + 2\,c_2\,t_2$

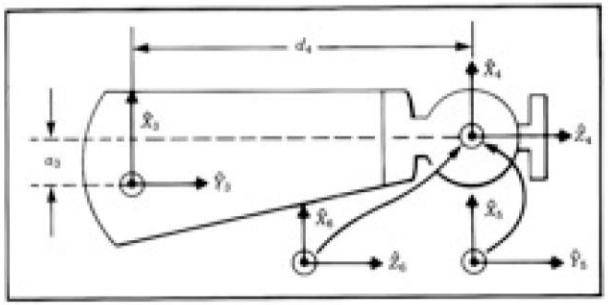
collect = $(c_1 + c_2)\,t_1 + (2\,c_2)\,t_2$

e2m = $\begin{pmatrix} c_1 + c_2 & 2\,c_2 \end{pmatrix}$

UCLA

Review with PUMA 560





i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-90°$	0	0	θ_2
3	0	a_2	d_3	θ_3
4	$-90°$	a_3	d_4	θ_4
5	$90°$	0	0	θ_5
6	$-90°$	0	0	θ_6

UCLA

PUMA 560

Puma 560

```
syms t1 t2 t3 t4 t5 t6 a2 a3 d3 d4

L1 = Link('revolute','d', 0,'a',0,'alpha',0,'modified')
L2 = Link('revolute','d', 0,'a',0,'alpha',-pi/2,'modified')
L3 = Link('revolute','d', d3,'a',a2,'alpha',0,'modified')
L4 = Link('revolute','d', d4,'a',a3,'alpha',-pi/2,'modified')
L5 = Link('revolute','d', 0,'a',0,'alpha',pi/2,'modified')
L6 = Link('revolute','d', 0,'a',0,'alpha',-pi/2,'modified')

th = [t1 t2 t3 t4 t5 t6]
Puma560 = SerialLink([L1 L2 L3 L4 L5 L6], 'name','Puma 560')
```

Forward Kinematics

```
Puma_fk = simplify(Puma560.fkine(th))
```

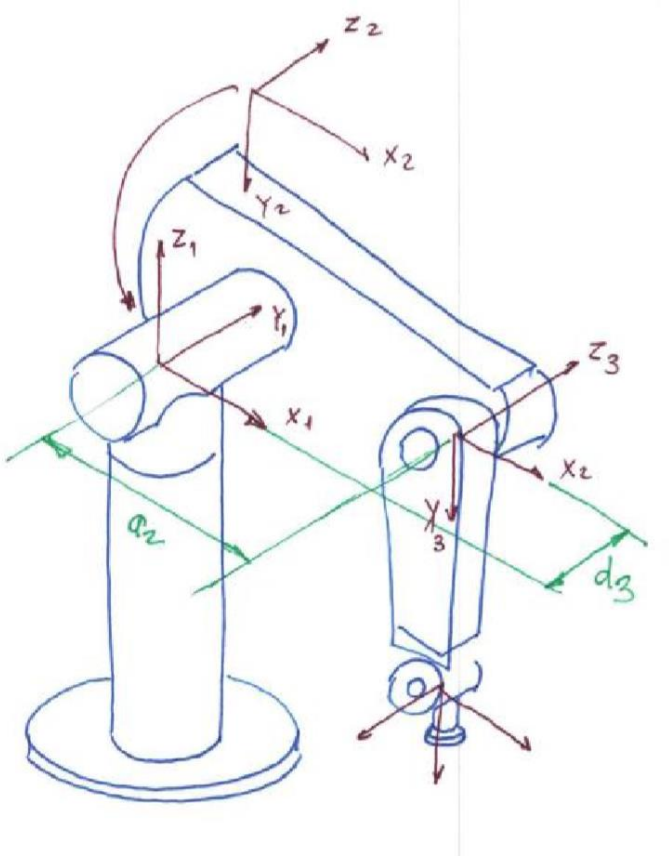
```
L1 =
Revolute(mod): theta=q, d=0, a=0, alpha=0, offset=0

L2 =
Revolute(mod): theta=q, d=0, a=0, alpha=-1.5708, offset=0

L3 =
Revolute(mod): theta=q, d=d3, a=a2, alpha=0, offset=0

L4 =
Revolute(mod): theta=q, d=d4, a=a3, alpha=-1.5708, offset=0

L5 =
Revolute(mod): theta=q, d=0, a=0, alpha=1.5708, offset=0

L6 =
Revolute(mod): theta=q, d=0, a=0, alpha=-1.5708, offset=0
```

th = $\begin{pmatrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{pmatrix}$

```
Puma560 =

Puma 560:: 6 axis, RRRRRR, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|          0|          0|
|  2|         q2|          0|          0|    -1.5708|          0|
|  3|         q3|         d3|         a2|          0|          0|
|  4|         q4|         d4|         a3|    -1.5708|          0|
|  5|         q5|          0|          0|     1.5708|          0|
|  6|         q6|          0|          0|    -1.5708|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

UCLA

PUMA 560 geometrical dimensions



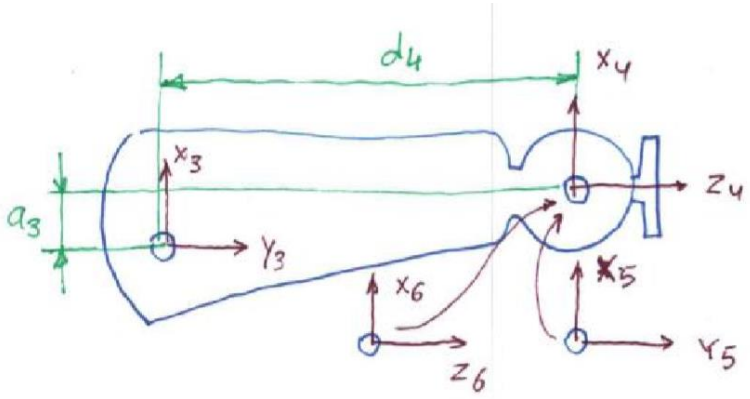


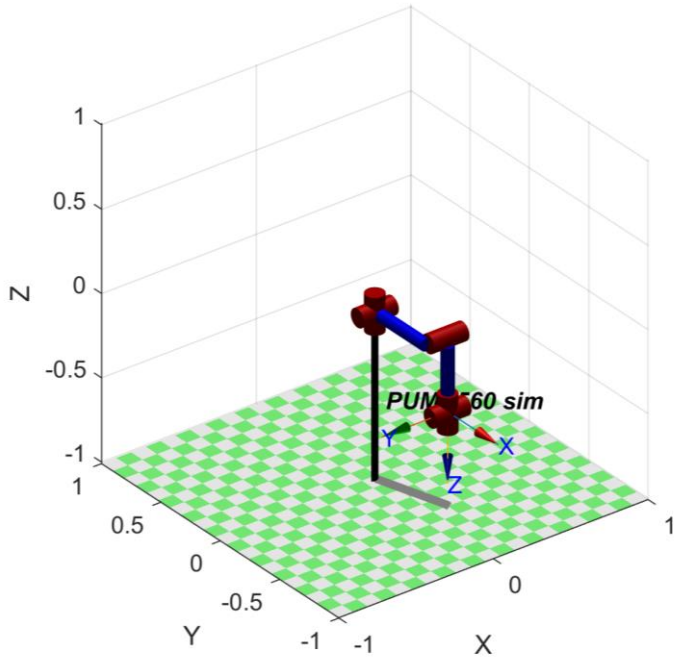
Fig. 2 - The PUMA 560 Coordinate system and geometrical dimensions - $a_2 = 0.4318m; a_3 = 0.0191m\ d_3 = 0.1254m\ d_2 = 0.4318m$

UCLA

PUMA560

Simulation

```
th = [0 0 0 0 0 0];
a2 = 0.4318; a3 = 0.019; d3 = 0.1254; d4=0.4318;

L1 = Link('revolute','d', 0,'a',0,'alpha',0,'modified')
L2 = Link('revolute','d', 0,'a',0,'alpha',-pi/2,'modified')
L3 = Link('revolute','d', d3,'a',a2,'alpha',0,'modified')
L4 = Link('revolute','d', d4,'a',a3,'alpha',-pi/2,'modified')
L5 = Link('revolute','d', 0,'a',0,'alpha',pi/2,'modified')
L6 = Link('revolute','d', 0,'a',0,'alpha',-pi/2,'modified')

Puma560_sim = SerialLink([L1 L2 L3 L4 L5 L6], 'name','Puma 560')
% Puma560_sim.plot(th)

time = [0 5 10];
t1 = [0 -pi/4 -pi/2];
t2 = [0 0 0]; t3 = [0 0 0];t4 = [0 0 0];t5 = [0 0 0];t6 = [0 0 0];
for i = 1:3
    th = [t1(i) t2(i) t3(i) t4(i) t5(i) t6(i)]
    Puma560_sim.plot(th)
end

plot(time,t1)
xlabel ('time'); ylabel('t1');
```



UCLA

Summary

- **Robotics Toolbox for MATLAB**
  - ✓ Installation
  - ✓ Robotics Toolbox with examples
    - Modified DH parameter / Standard DH parameter
    - Forward Kinematics/ Transformation Matrix
    - Plot / Simulation – Animation
  - ✓ Useful functions in Robotics Toolbox and MATLAB
  - ✓ Review with PUMA 560

Summary

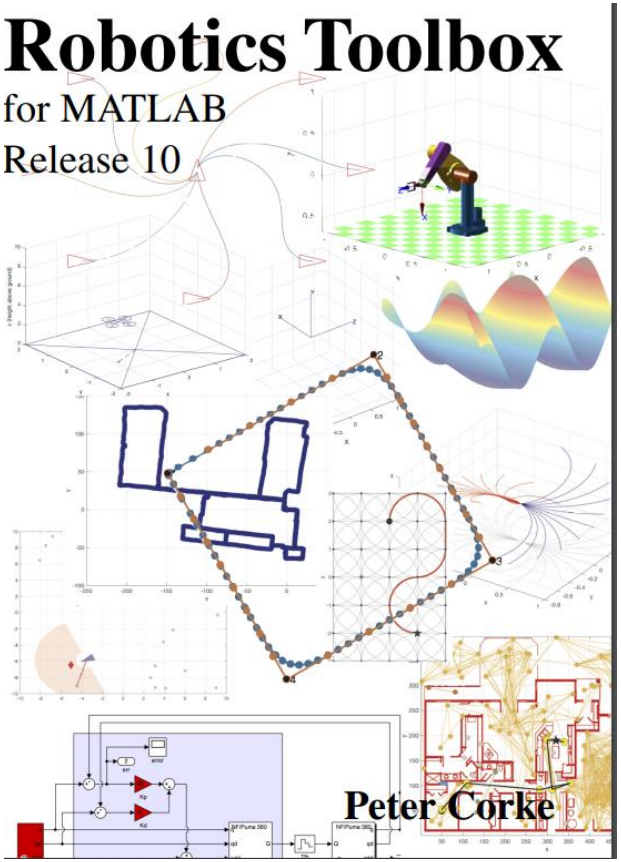
- **Useful functions in Robotics Toolbox**
  - Link
  - SerialLink
  - .A()
  - .fkine()
  - .plot
  - t2r : extract the rotation matrix from the homogeneous matrix
  - .t : extract the translation matrix from the homogeneous matrix

- **Useful function in MATLAB**
  - Transpose
  - cross
  - simplify
  - collect
  - equationToMatrix

Robotics Toolbox

Functions by category

- 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



Download the manual pdf file: click here

UCLA