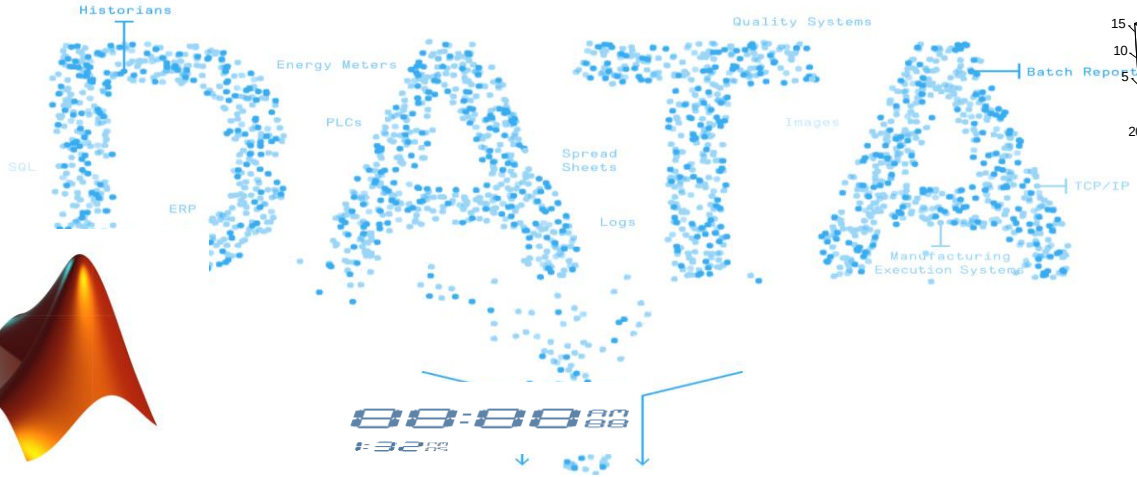
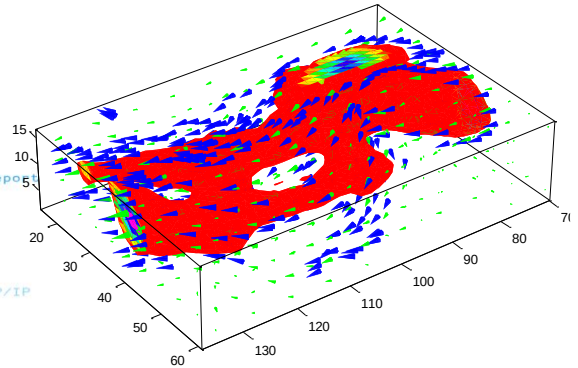
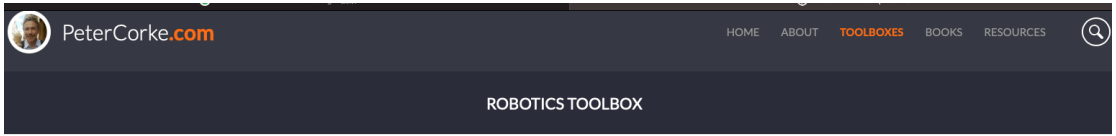
MATLAB Fundamental
Laboratory Handbook (MATLAB)

Chapter 13 MatLab Robotics Toolbox





MatLab Robotics Toolbox

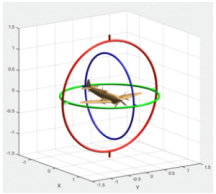


Introduction

This, the tenth release of the Toolbox, represents over twenty years of development and a substantial level of maturity. This version captures a large number of changes and extensions to support the **second edition** of my book "Robotics, Vision & Control".

For the first edition please go to this site to obtain the ninth release.

The Toolbox has always provided many functions that are useful for the study and simulation of classical arm-type robotics, for example such things as kinematics, dynamics, and trajectory generation.



The toolbox contains functions and classes to represent orientation and pose in 2D and 3D (SO(2), SE(2), SO(3), SE(3)) as matrices, quaternions, twists, triple angles, and matrix exponentials. The Toolbox also provides functions for manipulating and converting between datatypes such as vectors, homogeneous transformations and unit-quaternions which are necessary to represent 3-dimensional position and orientation.

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 also supports mobile robots with functions for robot motion models (unicycle, bicycle), path planning algorithms (bug, distance transform, D*, PRM), kinodynamic planning (lattice, RRT), localization (EKF, particle filter), map building (EKF) and simultaneous localization and mapping (EKF), and a Simulink model a of non-holonomic vehicle. The Toolbox also including a detailed Simulink model for a quadrotor flying robot.

Advantages of the Toolbox are that:

- the code is mature and provides a point of comparison for other implementations of the same algorithms;
- the routines are generally written in a straightforward manner which allows for easy understanding, perhaps at the expense of computational efficiency. If you feel strongly about computational efficiency then you can always rewrite the function to be more efficient, compile the M-file using the Matlab compiler, or create a MEX version;
- since source code is available there is a benefit for understanding and teaching.

This Toolbox, the Robotics Toolbox for MATLAB, is different to the MathWorks's own Robotic Systems Toolbox. Hear a bit more about how this came about in this video.

https://petercorke.com/toolboxes/robotics-toolbox/

MatLab Robotics Toolbox

- How to download and install?
  - https://petercorke.com/download/27/rtb/1045/rtb10-4-mltbx.mltbx
  - (Also available in your learningmall)
  - Drag it into the MatLab, then, install it.

MatLab Robotics Toolbox

Parameters

- alpha: α, the link twist;
- a: a, the link length;
- theta: θ, the joint angle;
- d: d, the link offset;
- offset —> set as zeros

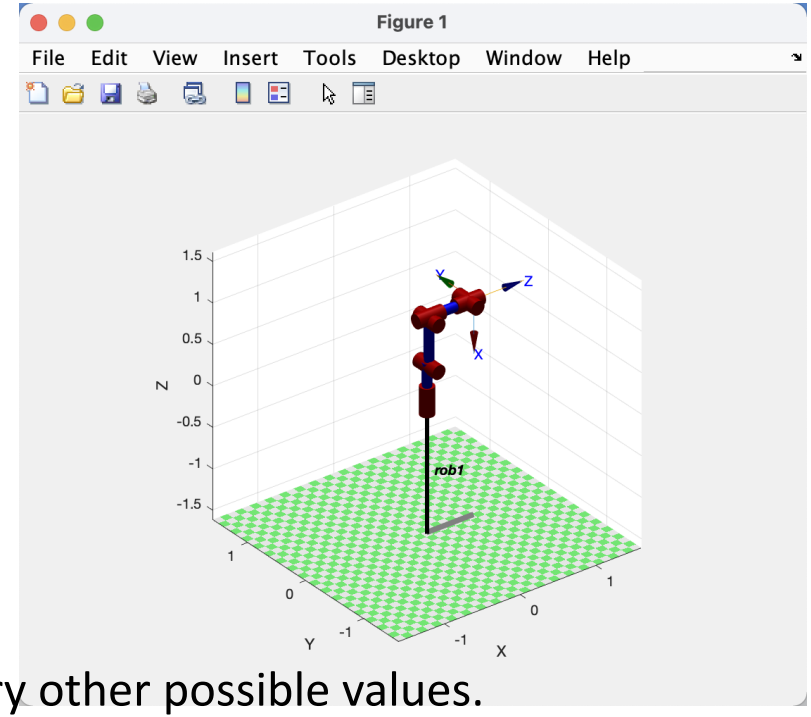
MatLab Robotics Toolbox

Building the robotics model

```
clear;
clc;
%建立机器人模型
%       theta    d        a        alpha     offset
L1=Link([0       0.4      0.025    pi/2      0     ]);  % Define the link D-H parameters
L2=Link([pi/2    0        0.56     0         0     ]);
L3=Link([0       0        0.035    pi/2      0     ]);
L4=Link([0       0.515    0        pi/2      0     ]);
L5=Link([pi      0        0        pi/2      0     ]);
L6=Link([0       0.08     0        0         0     ]);
robot=SerialLink([L1 L2 L3 L4 L5 L6],'name','rob1');    % Connect the links, and name the manipulator as rob1
robot.plot([0,pi/2,0,0,pi,0]);                          % Output the manipulator model, and input thetas for each joint
```

MatLab Robotics Toolbox

Building the robotics model



- Please try other possible values.

MatLab Robotics Toolbox

Display robotics parameters

Command Window

New to MATLAB? See resources for Getting Started.

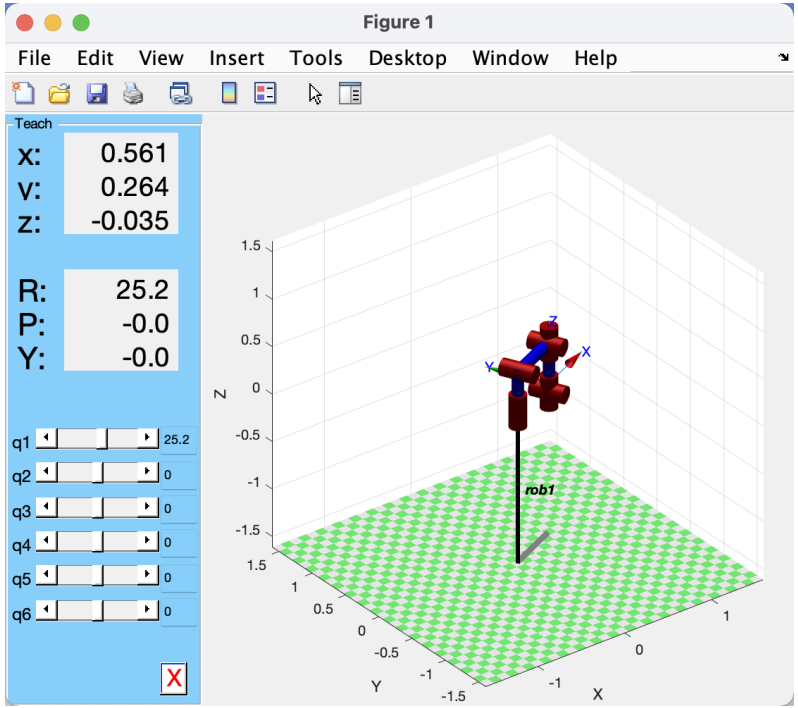
```
>> robot.display();

robot =

rob1:: 6 axis, RRRRRR, stdDH, slowRNE
+---+----------+----------+----------+----------+----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+----------+----------+----------+----------+----------+
|  1|         q1|       0.4|     0.025|    1.5708|         0|
|  2|         q2|         0|      0.56|         0|         0|
|  3|         q3|         0|     0.035|    1.5708|         0|
|  4|         q4|     0.515|         0|    1.5708|         0|
|  5|         q5|         0|         0|    1.5708|         0|
|  6|         q6|      0.08|         0|         0|         0|
+---+----------+----------+----------+----------+----------+

fx >>
```

- It gives the model parameters

MatLab Robotics Toolbox

Play the robot

- teach(robot);
- Enable adjusting joint angels



MatLab Robotics Toolbox

Forward and Inverse Kinematics

- What is the forward and inverse kinematics?

MatLab Robotics Toolbox

Forward and Inverse Kinematics

```
clear;
clc;
%建立机器人模型
%       theta    d        a        alpha     offset
L1=Link([0       0.4      0.025    pi/2      0     ]);  % Define the link D-H Parameters
L2=Link([pi/2    0        0.56     0         0     ]);
L3=Link([0       0        0.035    pi/2      0     ]);
L4=Link([0       0.515    0        pi/2      0     ]);
L5=Link([pi      0        0        pi/2      0     ]);
L6=Link([0       0.08     0        0         0     ]);
robot=SerialLink([L1 L2 L3 L4 L5 L6],'name','rab1');    % Connect the links, and name it as rob1
theta=[0,0,0,0,0,0];                                    % define the joint angles

p=robot.fkine(theta)
% forward kinematics, given theta, calculate end-effector position and orientation

q=robot.ikine(p)
% inverse kinematics, given end-effector position and orientation, calculate thetas
```

MatLab Robotics Toolbox

Forward and Inverse Kinematics

```
p =
     1     0     0    0.62
     0     1     0       0
     0     0     1  -0.035
     0     0     0       1

q =
     0     0     0     0     0     0

fx >>
```

- Please try other possible values.

MatLab Robotics Toolbox

Trajectory Planning

- In practice, we make use of the start + end positions to plan the trajectory.

```
clear;
clc;
%建立机器人模型
%       theta    d        a        alpha     offset
L1=Link([0       0.4      0.025    pi/2      0     ]);  % Define the D-H Parameters
L2=Link([pi/2    0        0.56     0         0     ]);
L3=Link([0       0        0.035    pi/2      0     ]);
L4=Link([0       0.515    0        pi/2      0     ]);
L5=Link([pi      0        0        pi/2      0     ]);
L6=Link([0       0.08     0        0         0     ]);
robot=SerialLink([L1 L2 L3 L4 L5 L6],'name','rpob1');   % Connect the links, and name the manipulator is rob1
T1=transl(0.5,0,0);                                     % Define the start position
T2=transl(0,0.5,0);                                     % Define the end position
q1=robot.ikine(T1);                                     % calculate the start joint angles
q2=robot.ikine(T2);                                     % calculate the end joint angles
[q ,qd, qdd]=jtraj(q1,q2,50);                           % 5-th polymonial, sample 50 times
grid on
T=robot.fkine(q);                                       % Get the positions of each of the 5 samples
position1 = T(1);
position2 = T(2);
position3 = T(3);
hold on
robot.plot(q);                                          % Display
```

MatLab Robotics Toolbox

Trajectory Planning

- In practice, we make use of the start + end positions to plan the trajectory.

```
clear;
clc;
%建立机器人模型
%       theta    d        a        alpha     offset
L1=Link([0       0.4      0.025    pi/2      0     ]);  % Define the D-H Parameters
L2=Link([pi/2    0        0.56     0         0     ]);
L3=Link([0       0        0.035    pi/2      0     ]);
L4=Link([0       0.515    0        pi/2      0     ]);
L5=Link([pi      0        0        pi/2      0     ]);
L6=Link([0       0.08     0        0         0     ]);
robot=SerialLink([L1 L2 L3 L4 L5 L6],'name','rpob1');   % Connect the links, and name the manipulator is rob1
T1=transl(0.5,0,0);                                     % Define the start position
T2=transl(0,0.5,0);                                     % Define the end position
q1=robot.ikine(T1);                                     % calculate the start joint angles
q2=robot.ikine(T2);                                     % calculate the end joint angles
[q ,qd, qdd]=jtraj(q1,q2,50);                           % 5-th polymonial, sample 50 times
grid on
T=robot.fkine(q);                                       % Get the positions of each of the 5 samples
position1 = T(1);
position2 = T(2);
position3 = T(3);
hold on
robot.plot(q);                                          % Display
```

MatLab Robotics Toolbox

Trajectory Planning