MATLAB® & SIMULINK®
Tech Forum & EXPO

16th MATLAB & SIMULINK技術高峰會

工業型機器人模擬與控制設計

Jerry Tung

Accelerating innovation
with MATLAB & Simulink

TeraSoft
www.terasoft.com.tw

MathWorks
Accelerating the pace of engineering and science

An Advanced Robotics System Requires Multiple Technologies

Connect to CAD Tool

Analysis of I

Actuator De

Joint Contro

Drivers

Safeguards

Motion Control

Inverse Kinen

Force/Impeda

Model-based

Sensor Prepr

Supervisory Logic

Sensor Fusion

Trajectory Planning

Obstacle Avoidance

Distributed Communication

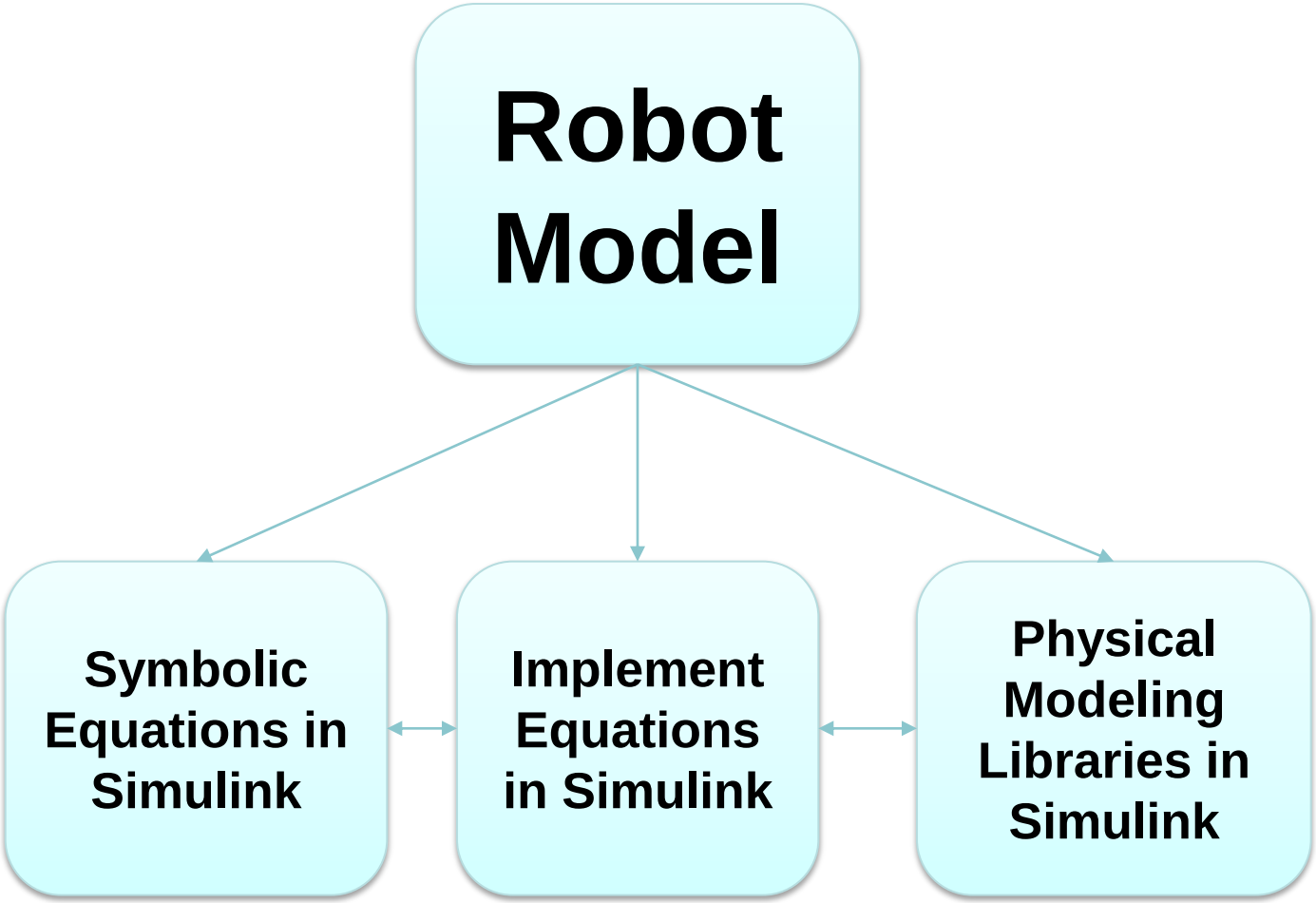
Object Detection

MATLAB and Simulink are used to solve complex engineering problems

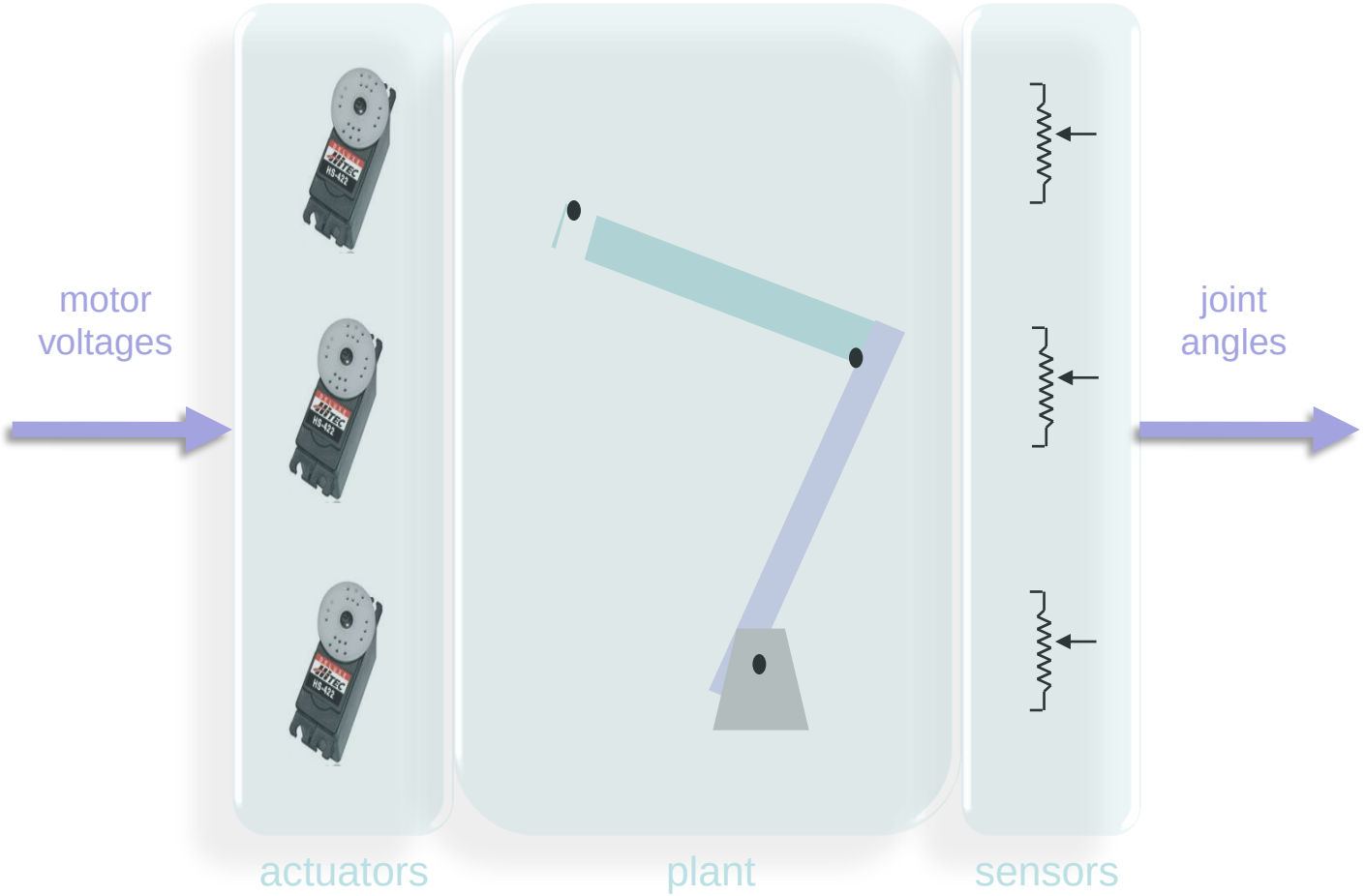
Agenda

- Modeling and Simulating the Manipulator
  - Euler Lagrange Equation
  - Simscape Multibody
- Robot Control
  - Jacobian and Inverse Kinematics Calculation
  - Path Planning
- Overall System Performance Simulation
  - Multi-domain Robot Simulation
  - .urdf/.sdf import, Simscape Multibody animator
  - Simulink Real-Time (Rapid Control Algorithm Prototyping)

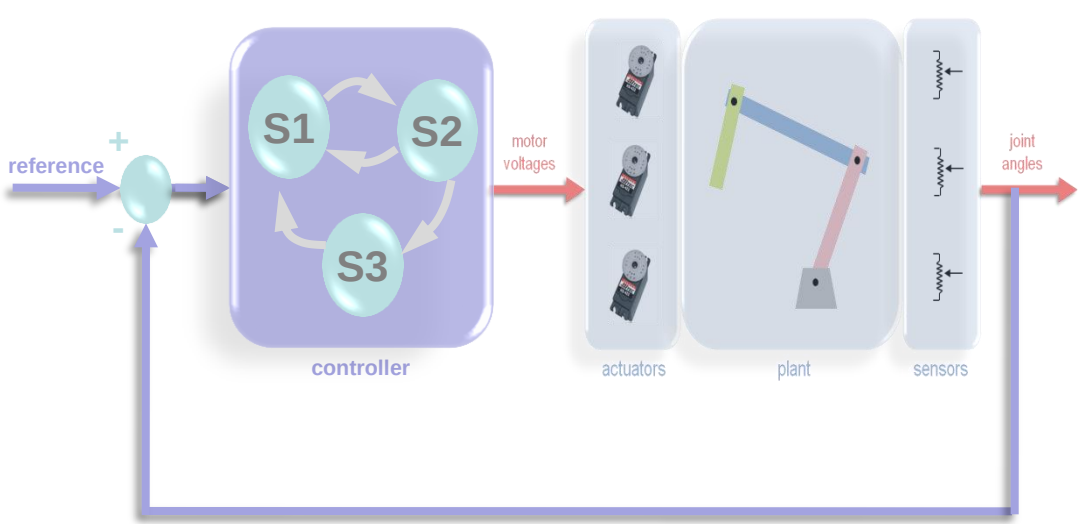
Modeling your Robotics System in MATLAB and Simulink

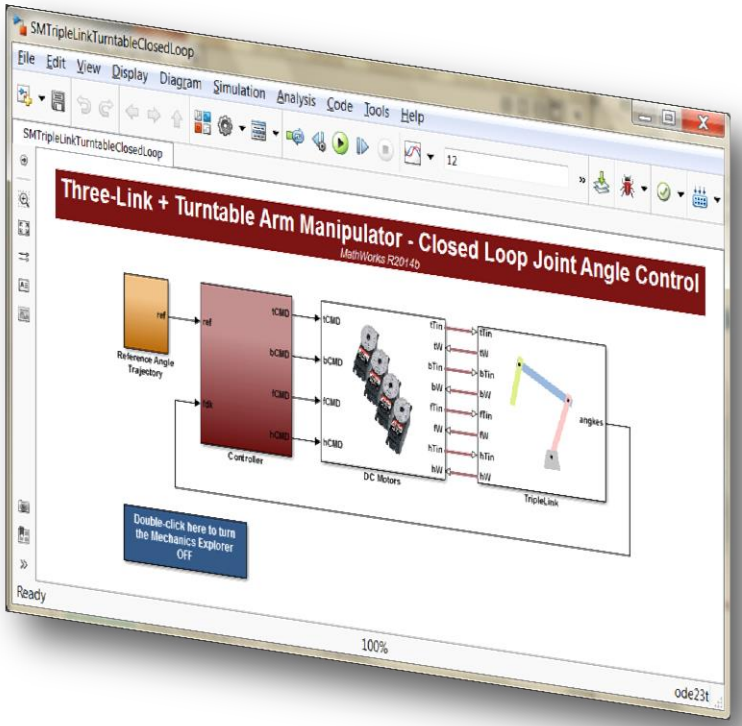


Robot Model with Actuators and Sensors



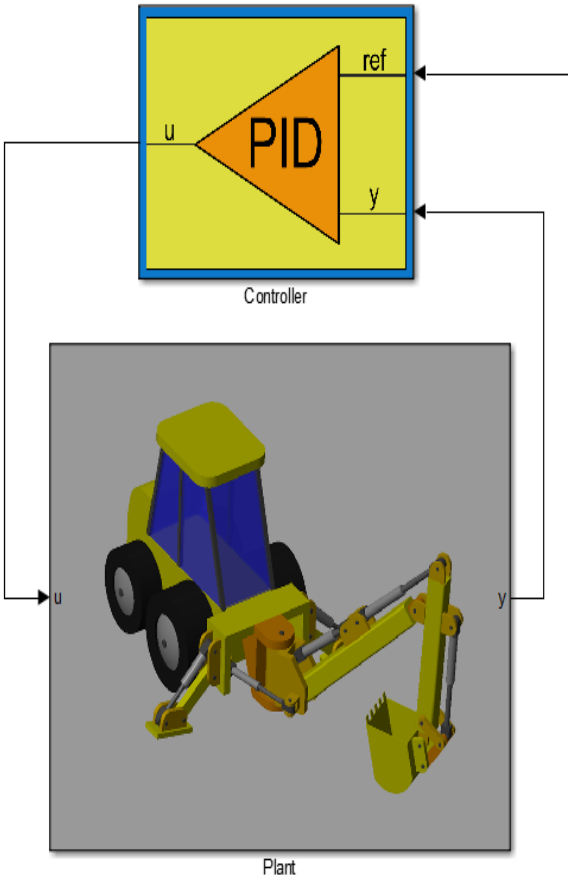
Use the model to design the Controller

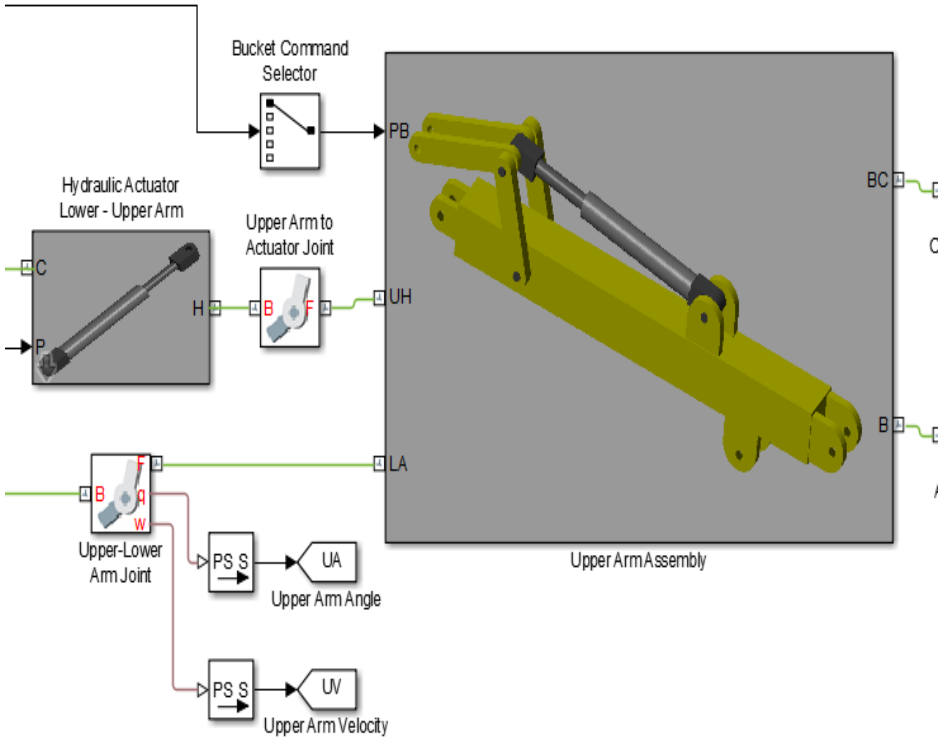




Modeling both, plant and controller in a single environment allows us to better understand and optimize the performance of the entire system

Multi-domain Robot Simulation

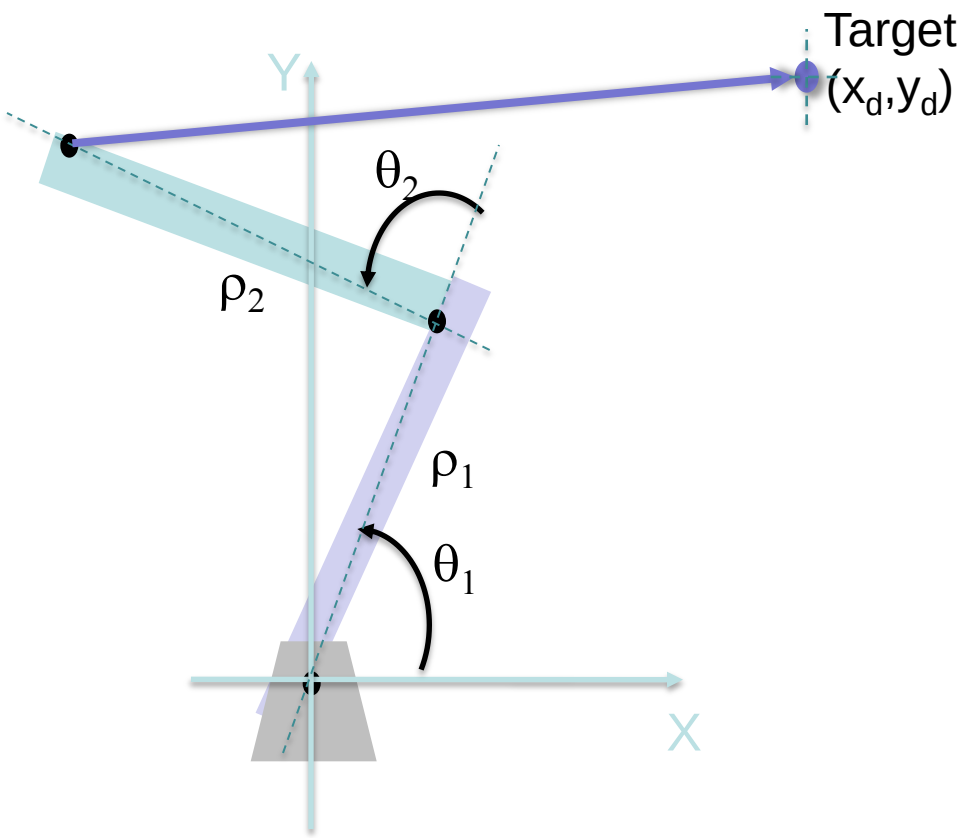




Agenda

- Modeling and Simulating the Manipulator
  - Euler Lagrange Equation
  - Simscape Multibody
- Robot Control
  - Jacobian and Inverse Kinematics Calculation
  - Path Planning
- Overall System Performance Simulation
  - Multi-domain Robot Simulation
  - .urdf/.sdf import, Simscape Multibody animator
  - Simulink Real-Time (Rapid Control Algorithm Prototyping)

Inverse Kinematics: Manipulator to reach a given target?



Three Options

1) Simscape Multibody

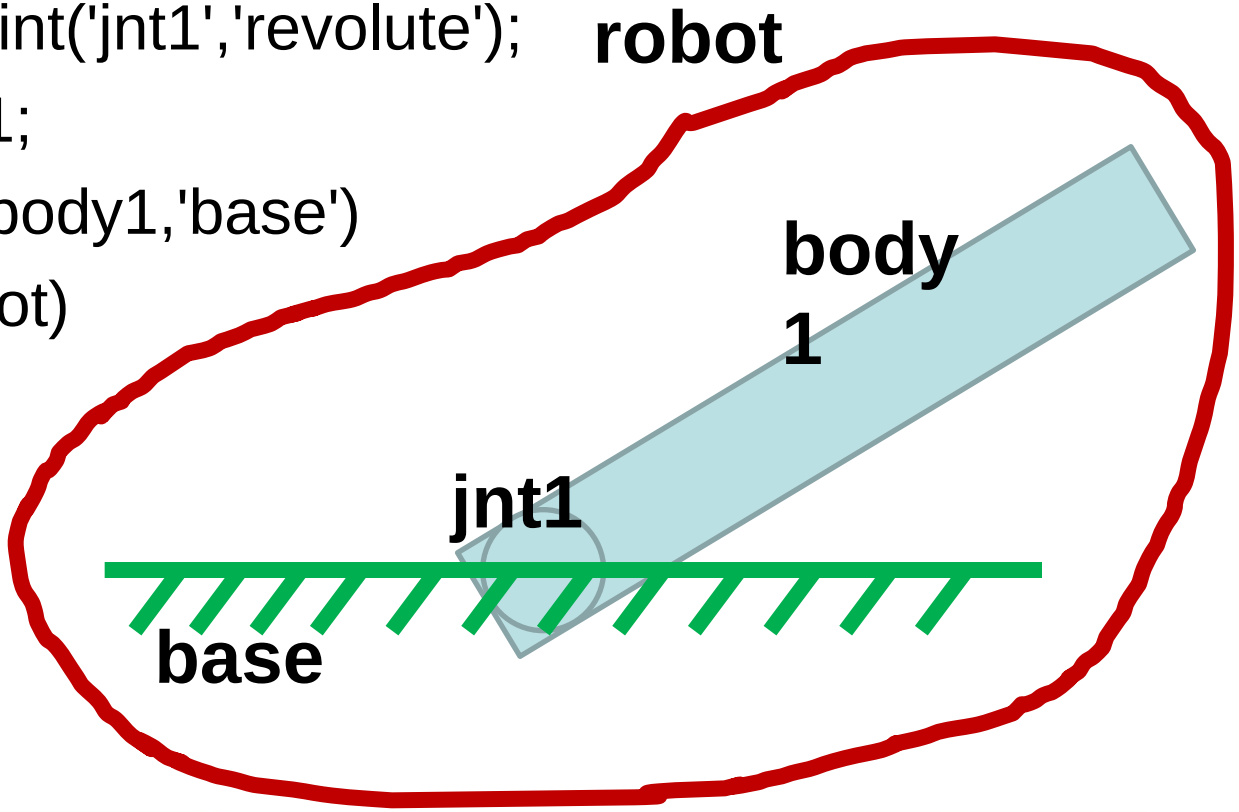
2) Constrained Optimization

3) Robotics System Toolbox

Find the joint angels to make the manipulator reach a given target

Rigid Body Tree (Building Robots in MATLAB)

```
robot=robotics.RigidBodyTree;
body1=robotics.RigidBody('body1');
jnt1=robotics.Joint('jnt1','revolute');
body1.Joint=jnt1;
addBody(robot,body1,'base')
showdetails(robot)
```



Rigid Body Tree (Building Robots in MATLAB)

```
% Description of a tree-structured robot
>> puma =
robotics.RigidBodyTree;
```

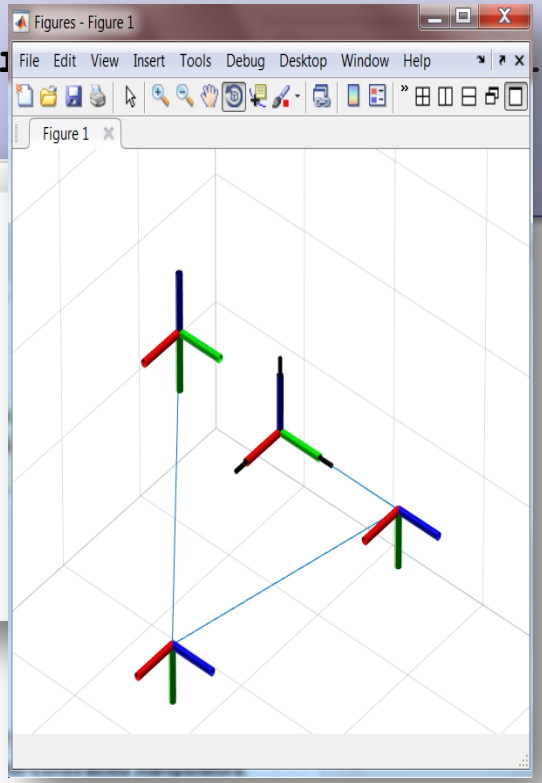
```
% Inverse kinematics for rigid body trees
>> ik =
robotics.Inver
n);
```

Puma 560 described with **MDH** parameters

index	a (meter)	alpha (radians)	d (meters)	θ (rad
1	0	0	0	-
2	0	-π/2	0.2435	-
3	0.4318	0	-0.0934	-
4	-0.0203	π/2	0.4331	-
5	0	-π/2	0.0	-
6	0	π/2	0.0	-

Command Window

```
--------------------
Robot: (6 bodies)

 Idx    Body Name   Joint Name   Joint Type    Parent Name(Idx)
 ---    ---------   ----------   ----------    ----------------
   1          L1         jnt1     revolute            base(0)
   2          L2         jnt2     revolute            L1(1)
   3          L3         jnt3     revolute            L2(2)
   4          L4         jnt4     revolute            L3(3)
   5          L5         jnt5     revolute            L4(4)
   6          L6         jnt6     revolute            L5(5)
--------------------
```

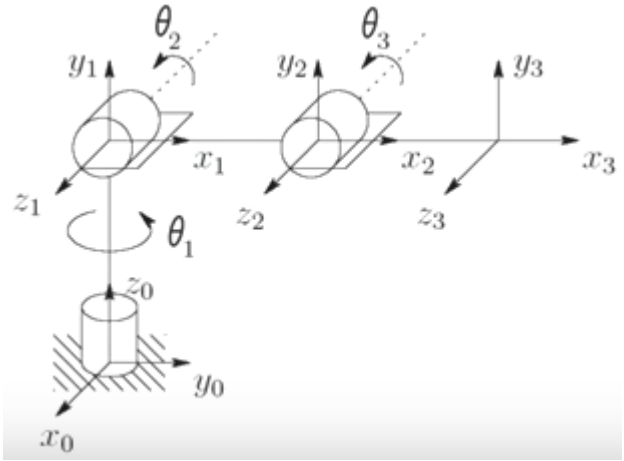


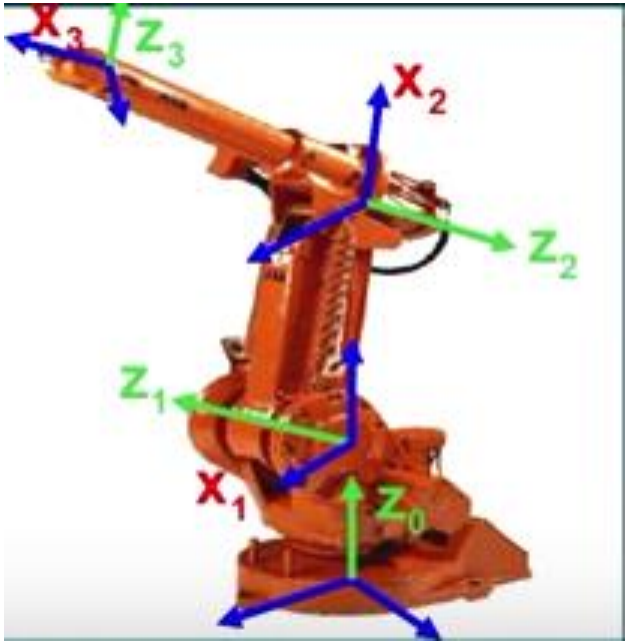
Redundant and over-constrained manipulators.

Robotics System Toolbox: represent rigid body trees in MATLAB and solve inverse kinematics

Rigid Body Tree Example

```
dhparams=[
        0       pi/2    0.25    pi/2;
        0.5     0       0       0;
        0.5     0       0       0 ];

robot=robotics.RigidBodyTree;

body1=robotics.RigidBody('body1');
jnt1=robotics.Joint('jnt1','revolute');
setFixedTransform(jnt1,dhparams(1,:),'dh');
body1.Joint=jnt1;
addBody(robot,body1,'base')
…..
```





Robot Configuration

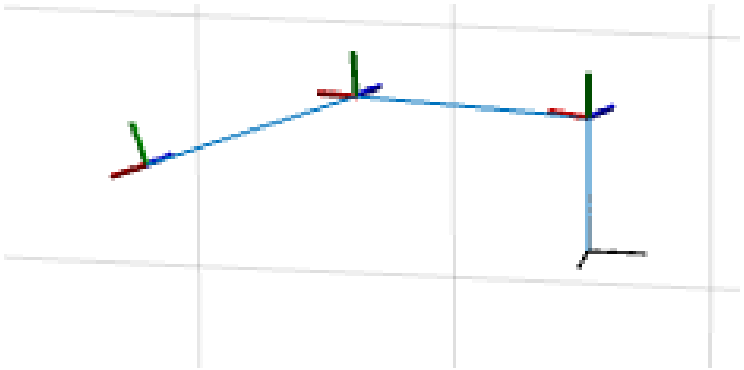
```
conf1=homeConfiguration(robot)
```

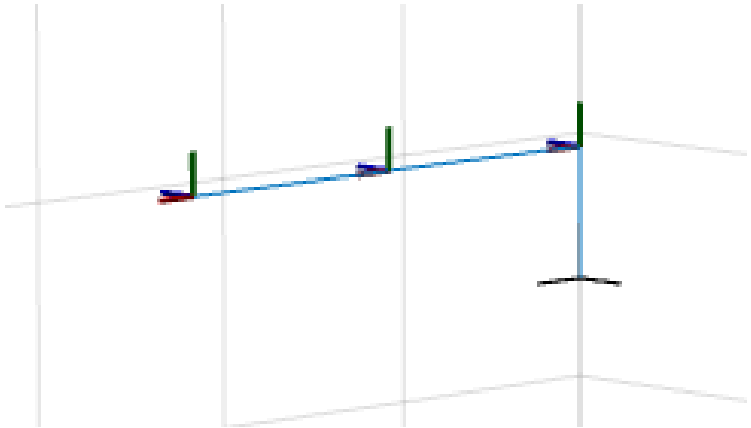
The configuration when the robot is constructed

```
conf2=randomConfiguration(robot)
```

Manually assign the robot configuration

```
conf3=conf1;conf3(1).JointPosition=pi/4;
```



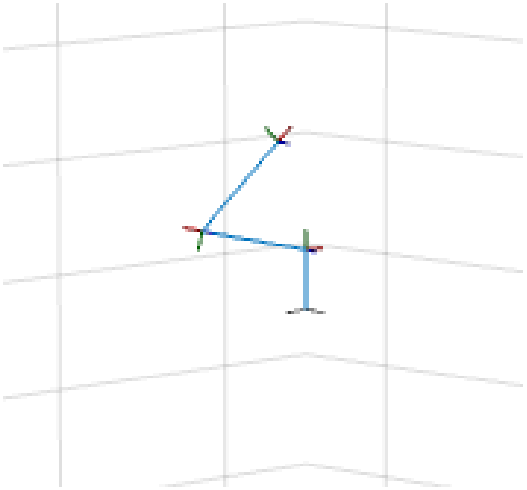


Jacobian Calculation

```
show(robot)

conf1=randomConfiguration(robot)
conf1.JointPosition
show(robot,conf1)

jacobian = geometricJacobian(robot,conf1,'body3')
```

```
         0   -0.9224   -0.9224
   -0.0000    0.3863    0.3863
    1.0000    0.0000    0.0000
    0.8839   -0.0887   -0.0763
   -0.3702   -0.2119   -0.1822
   -0.0000    0.9583    0.4593
```

IK Calculation



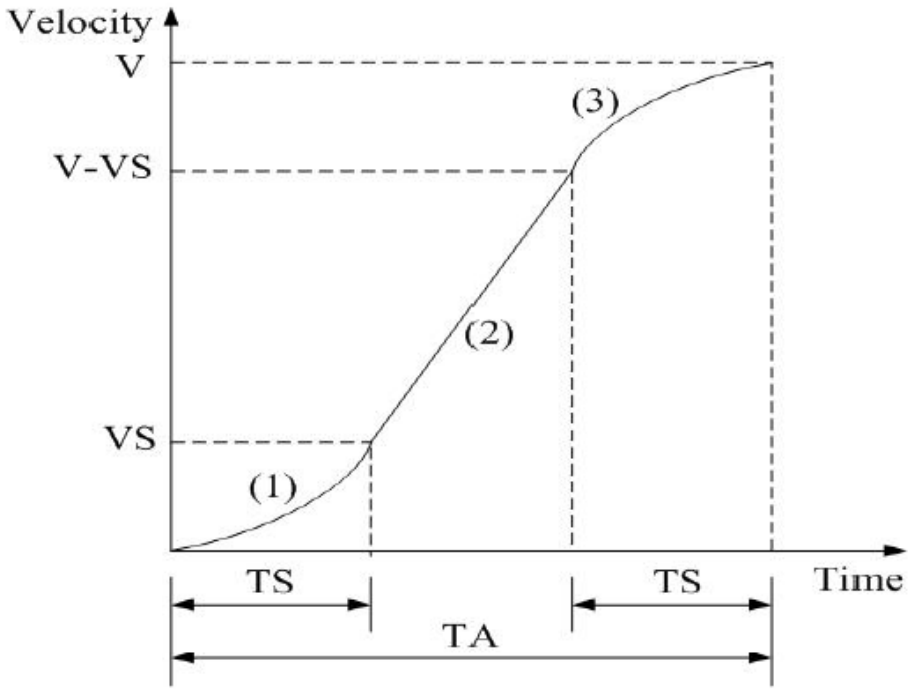
```
conf1=randomConfiguration(robot);
show(robot,conf1);

ik = robotics.InverseKinematics('RigidBodyTree',robot);

tform = getTransform(robot,conf1,'body3','base');
weights = ones(6,1);
[configSoln, solnInfo] = ik('body3',tform,weights,robot.homeConfiguration)
configSoln(1).JointPosition/pi*180
configSoln(2).JointPosition/pi*180
configSoln(3).JointPosition/pi*180

show(robot,configSoln)
```

```
tform =

   -0.6927    0.6713    0.2638    0.1243
    0.1895   -0.1836    0.9646   -0.0340
    0.6959    0.7181    0.0000    0.7074
         0         0         0    1.0000
```

```
ans =

  164.7025

  167.3599

 -123.2593
```

Path Planning

- 三階多項式
- 五階多項式
- 梯形速度規劃
- 高階多項式
- Spline
- …

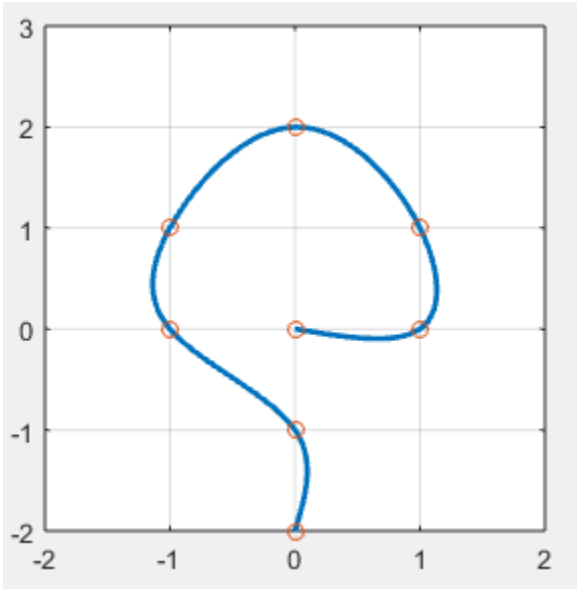


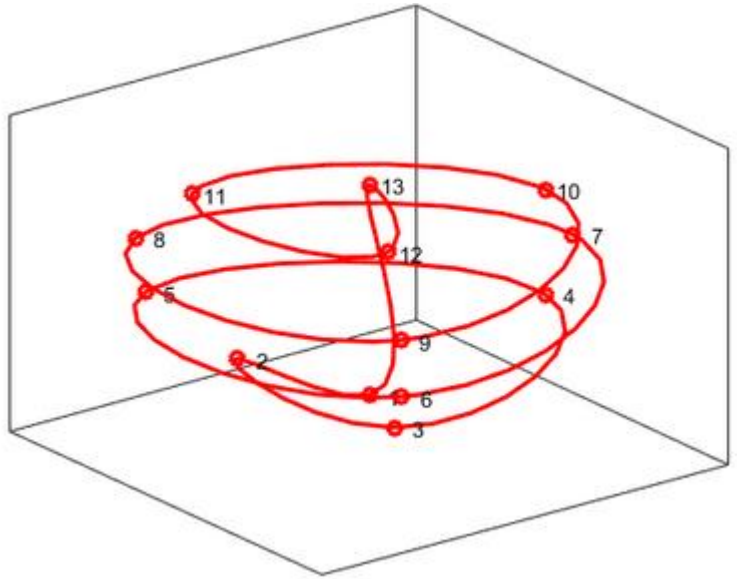
$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$
$$\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$

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

X = linsolve(A,B)

2D/3D Cubic Spline





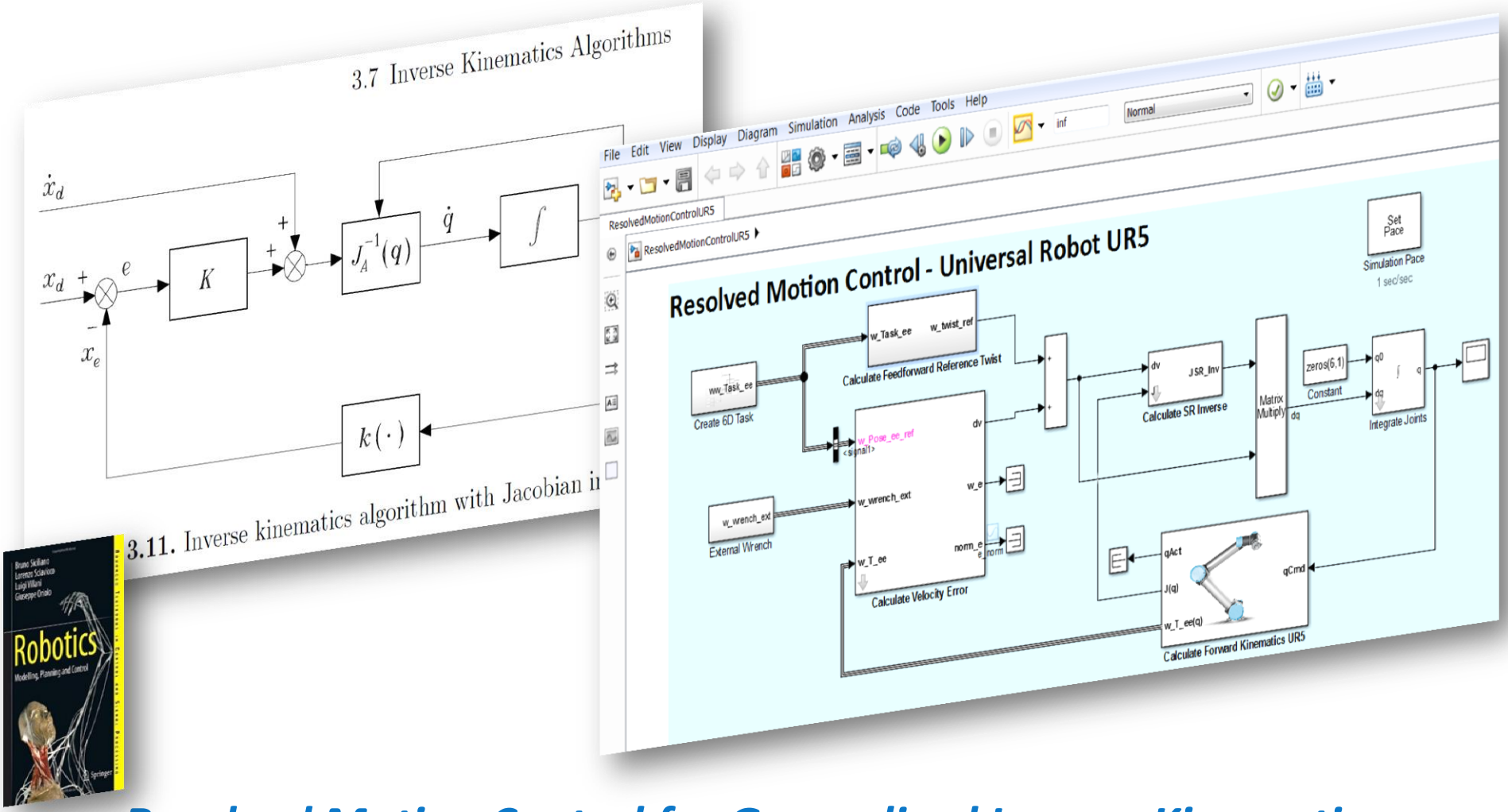

```
cs1=cscvn(xyz(:,[1:end]))
fnplt(cs1,'r',2)
x1=polyval(cs1.coefs(1,:),0.1)
y1=polyval(cs1.coefs(2,:),0.1)
z1=polyval(cs1.coefs(3,:),0.1)
```

```
cs1
  struct with fields:

    form: 'pp'
  breaks: [1×13 double]
   coefs: [36×4 double]
  pieces: 12
   order: 4
     dim: 3
```

Agenda

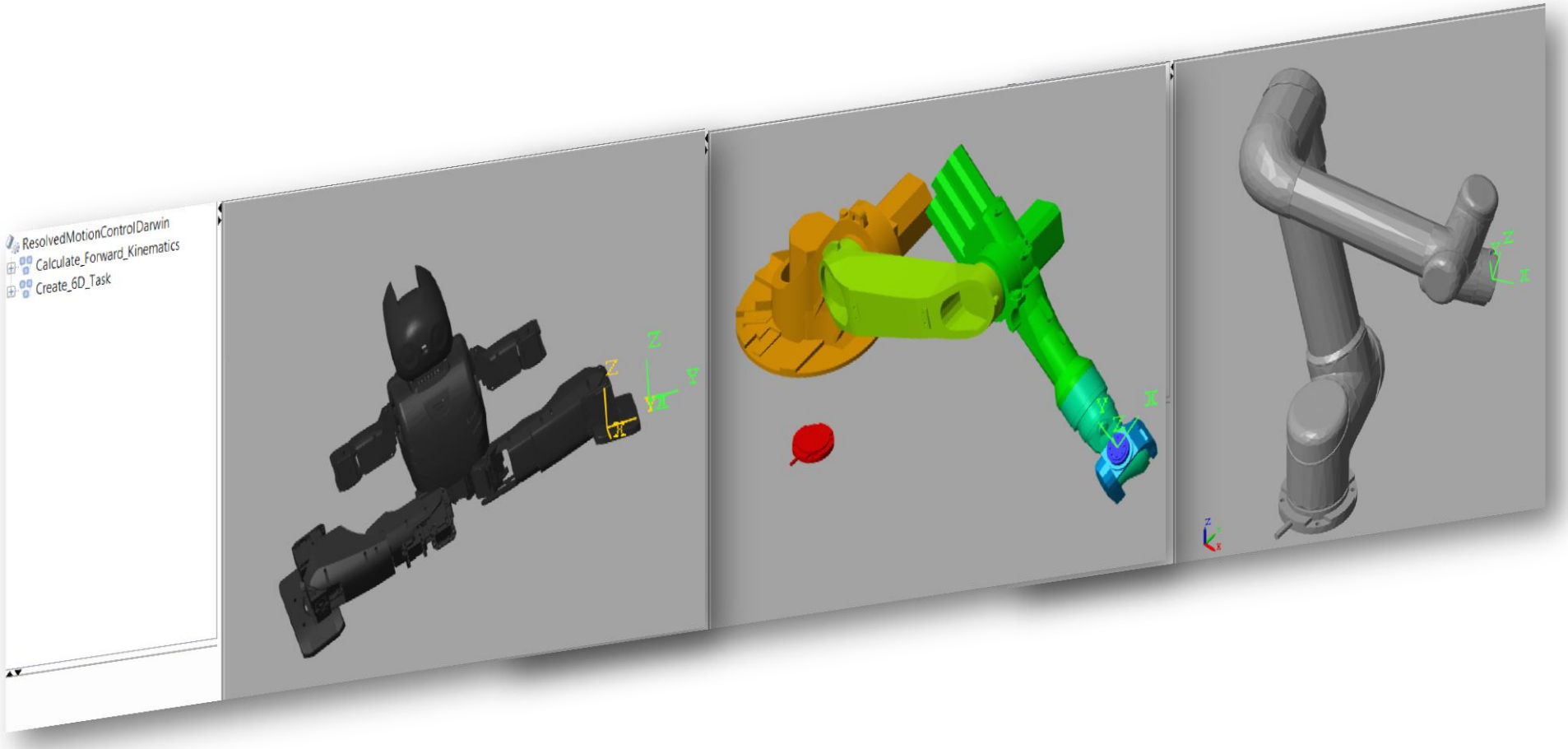
- Modeling and Simulating the Manipulator
  - Euler Lagrange Equation
  - Simscape Multibody
- Robot Control
  - Jacobian and Inverse Kinematics Calculation
  - Path Planning
- Overall System Performance Simulation
  - Multi-domain Robot Simulation
  - .urdf/.sdf import, Simscape Multibody animator
  - Simulink Real-Time (Rapid Control Algorithm Prototyping)

Resolved Motion Control



Resolved Motion Control for Generalized Inverse Kinematics

Resolved Motion Control



Resolved Motion Control for Generalized Inverse Kinematics

Simulink Real Time for algorithm prototyping

NATLAB/Simulink

Robot Control Algorithm

C code generation

Simulink Real Time OS

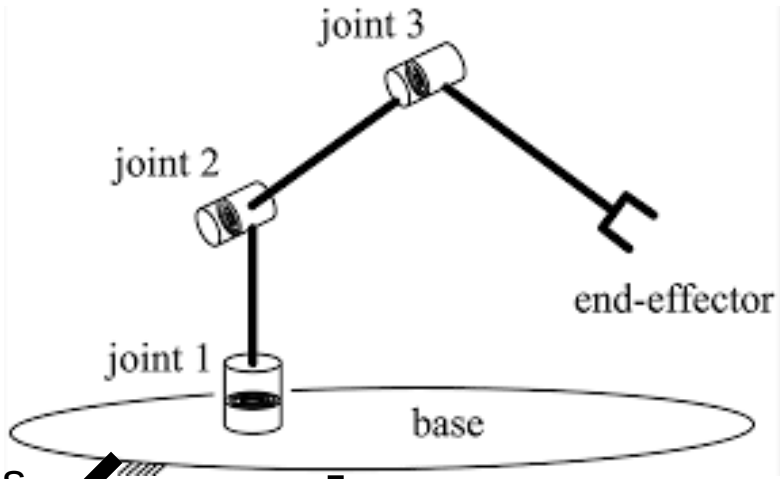
Robot Control Algorithm

Other Interfaces

EtherCAT

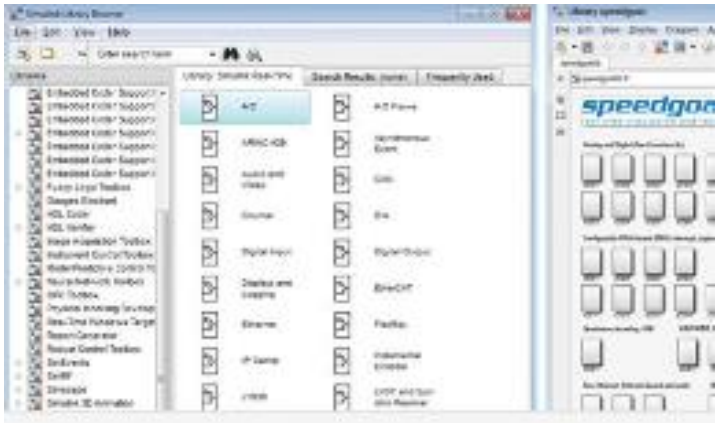
Motion Cards

Drivers



Simulink Real Time

- Real Time Kernel
- Multicore CPU
- FPGA
- I/O and protocol interfaces



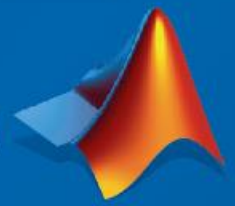
Summary

- In MATLAB, Manipulator can be modeled by mathematical equation, CAD model, or Simscape Multibody blocks
- MATLAB provides many Jacobian and IK calculation, and path planning functions
- Simulink is a perfect environment to integrate everything together to evaluate the system performance (animation)
- Using SLRT for rapid algorithm prototyping

2016

MATLAB® & SIMULINK® Tech Forum & EXPO

Accelerating innovation
with MATLAB & Simulink



TeraSoft 鈦思科技 MathWorks®