WPI

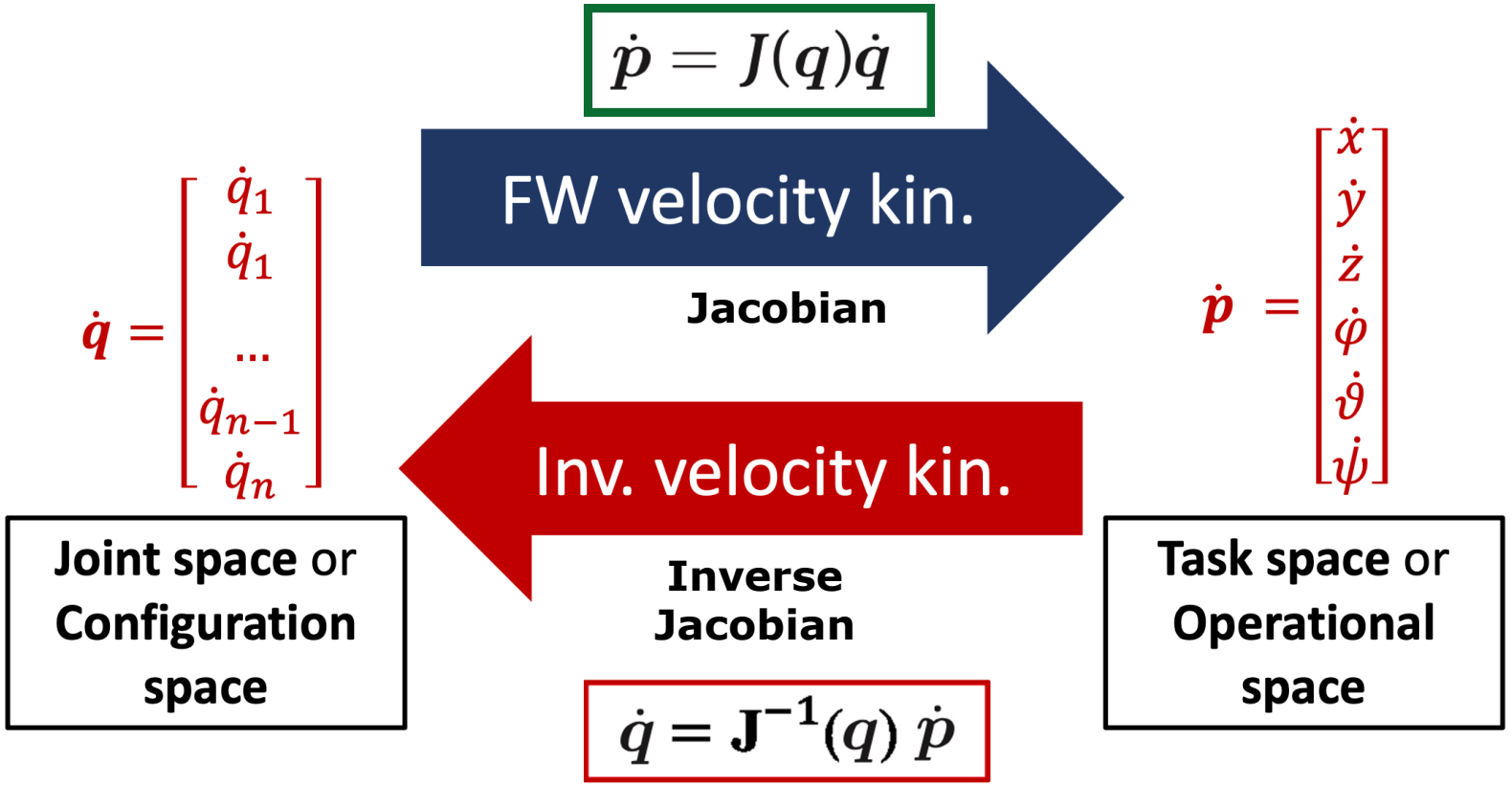
Lecture 10

Cont. Differential Kinematics, Jacobian for under-and-over-actuated manipulators and Velocity Manipulability Ellipsoids

Alexandros Lioulemes, PhD

Velocity (Differential) Kinematics

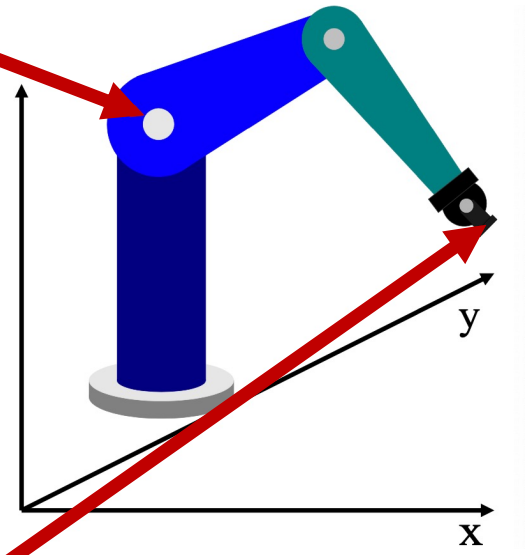
- Control the **velocity** of the **end effector**



https://en.wikipedia.org/wiki/Robot_kinematics#Velocity_kinematics

How to calculate the Geometric Jacobian?

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \dots \\ q_{n-1} \\ q_n \end{bmatrix}$$



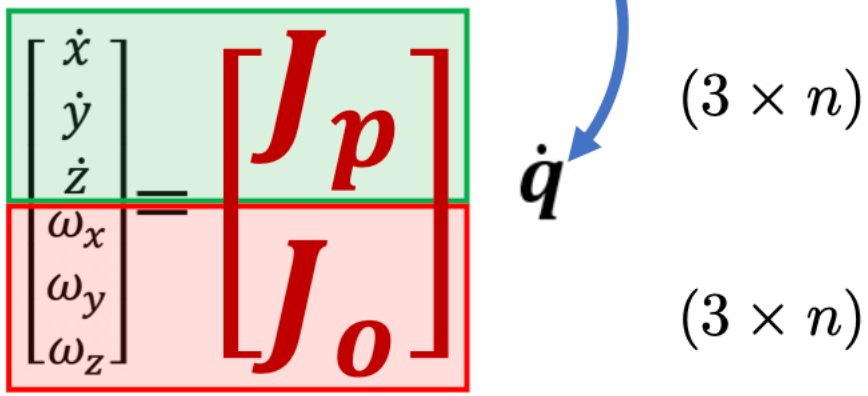
$$p = \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \vartheta \\ \psi \end{bmatrix}$$

- $\dot{p} = J(q)\dot{q}$
- $J(q) = \begin{bmatrix} J_p \\ J_o \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \begin{matrix} (3 \times n) \\ (3 \times n) \end{matrix}$ $(6 \times n)$
- $\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} J_p \\ J_o \end{bmatrix} \dot{q}$

How to calculate the Geometric Jacobian?

- Solve for Linear velocity and Angular velocity separately

$$\dot{\boldsymbol{p}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$



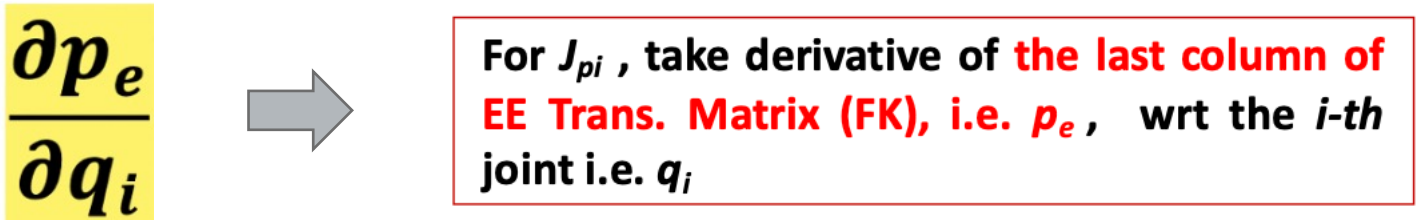
- Linear
- Angular

https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

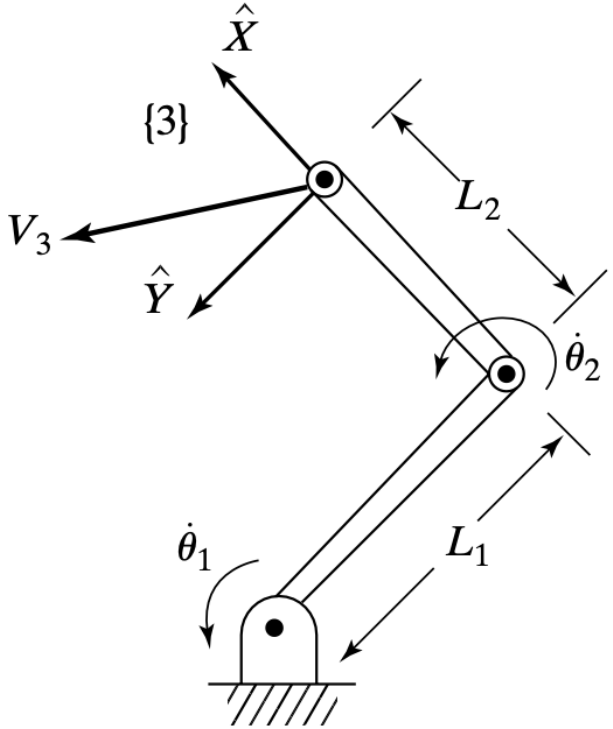
How to calculate the Linear velocity J_p

Procedure:

- Derive the Forward Kinematics (FK)
- Write position equations of the end-effector (p_e)
- Take derivative for velocity



How to calculate the Linear velocity J_p - Solution



3 x 2

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} -l_1\sin(\theta_1) - l_2\sin(\theta_1+\theta_2) & -l_2\sin(\theta_1+\theta_2) \\ l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) & l_2\cos(\theta_1+\theta_2) \\ 0 & 0 \\ & \\ & \\ & \end{bmatrix} \cdot \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

6 x 2

https://en.wikibooks.org/wiki/Robotics_Kinematics_and_Dynamics/Serial_Manipulator_Differential_Kinematics

How to calculate the Angular velocity J_o

Procedure:

- Last column of Rotation Matrix

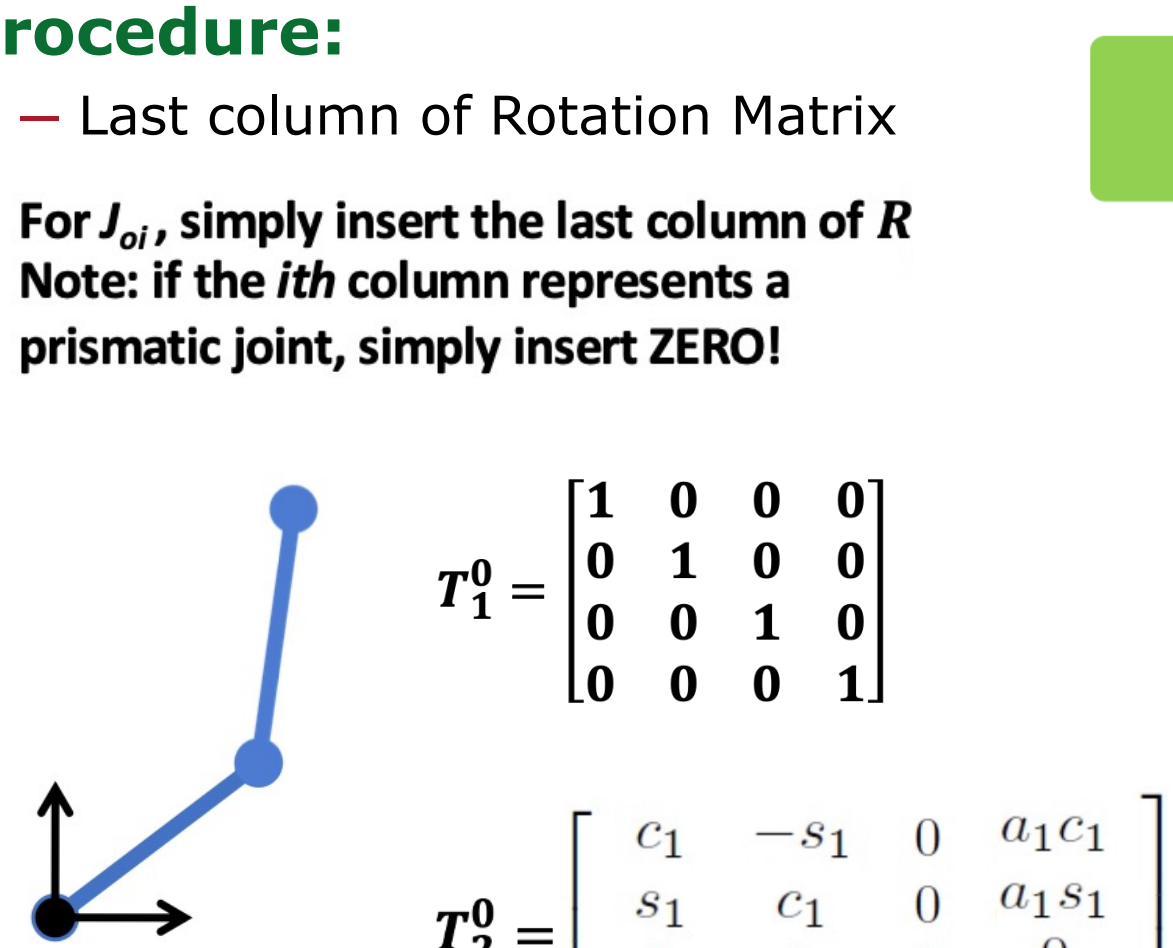
For J_{oi}, simply insert the last column of R
Note: if the *ith* column represents a prismatic joint, simply insert ZERO!

How to calculate the Angular velocity J_o- Solution

Procedure:

- Last column of Rotation Matrix

For J_{oi}, simply insert the last column of R
Note: if the *ith* column represents a prismatic joint, simply insert ZERO!



$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

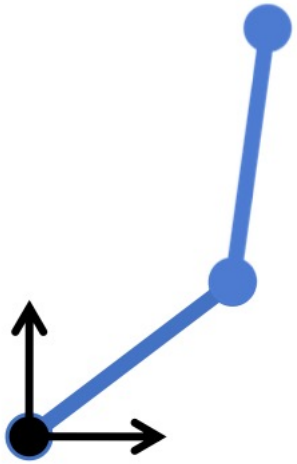
$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \\ & \end{bmatrix}$$

How to calculate the Angular velocity J_o- Solution

Procedure:

- Last column of Rotation Matrix

For J_{oi}, simply insert the last column of R
Note: if the *ith* column represents a prismatic joint, simply insert ZERO!



$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

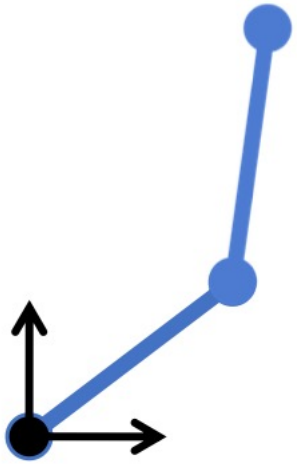
$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix}$$

How to calculate the Angular velocity J_o- Solution

Procedure:

- Last column of Rotation Matrix

For J_{oi}, simply insert the last column of R
Note: if the *ith* column represents a prismatic joint, simply insert ZERO!



$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

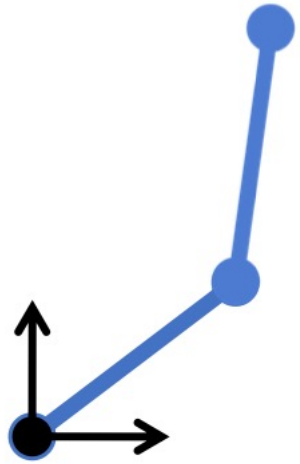
$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \end{bmatrix}$$

How to calculate the Angular velocity J_o- Solution

Procedure:

- Last column of Rotation Matrix

For J_{oi}, simply insert the last column of R
Note: if the *ith* column represents a prismatic joint, simply insert ZERO!

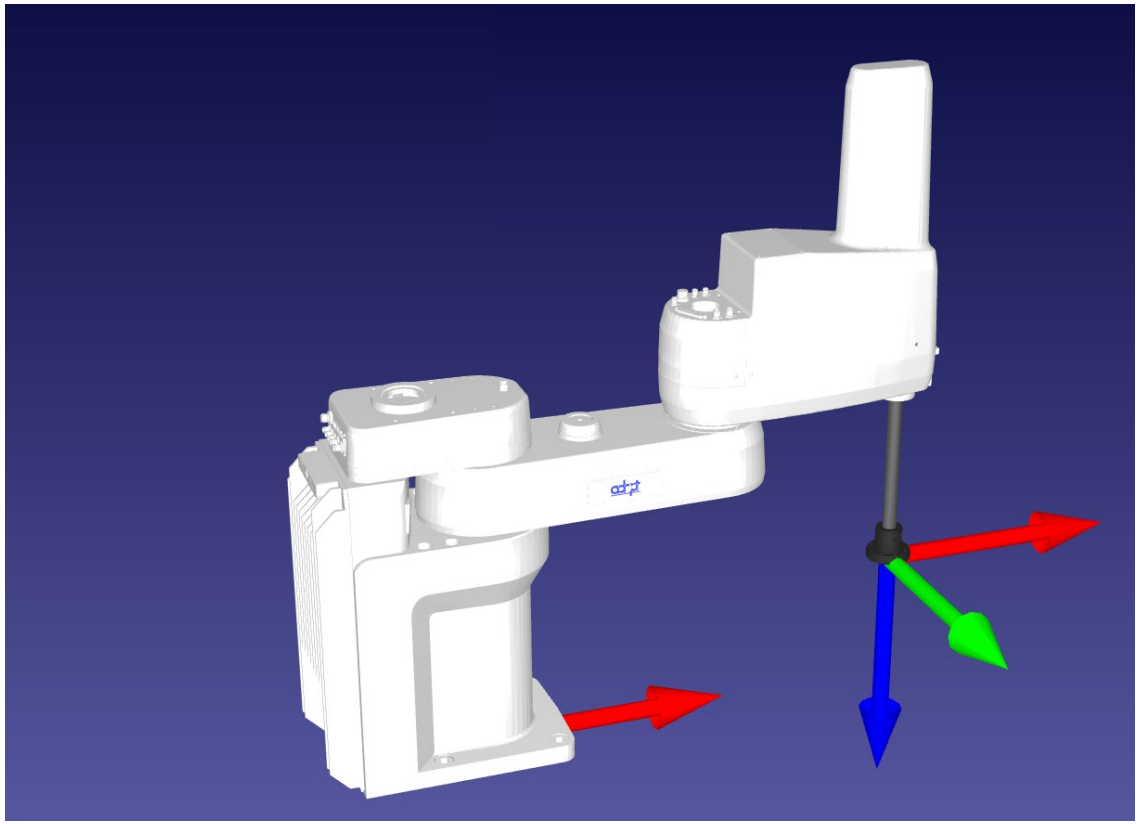


$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Adept Cobra s600

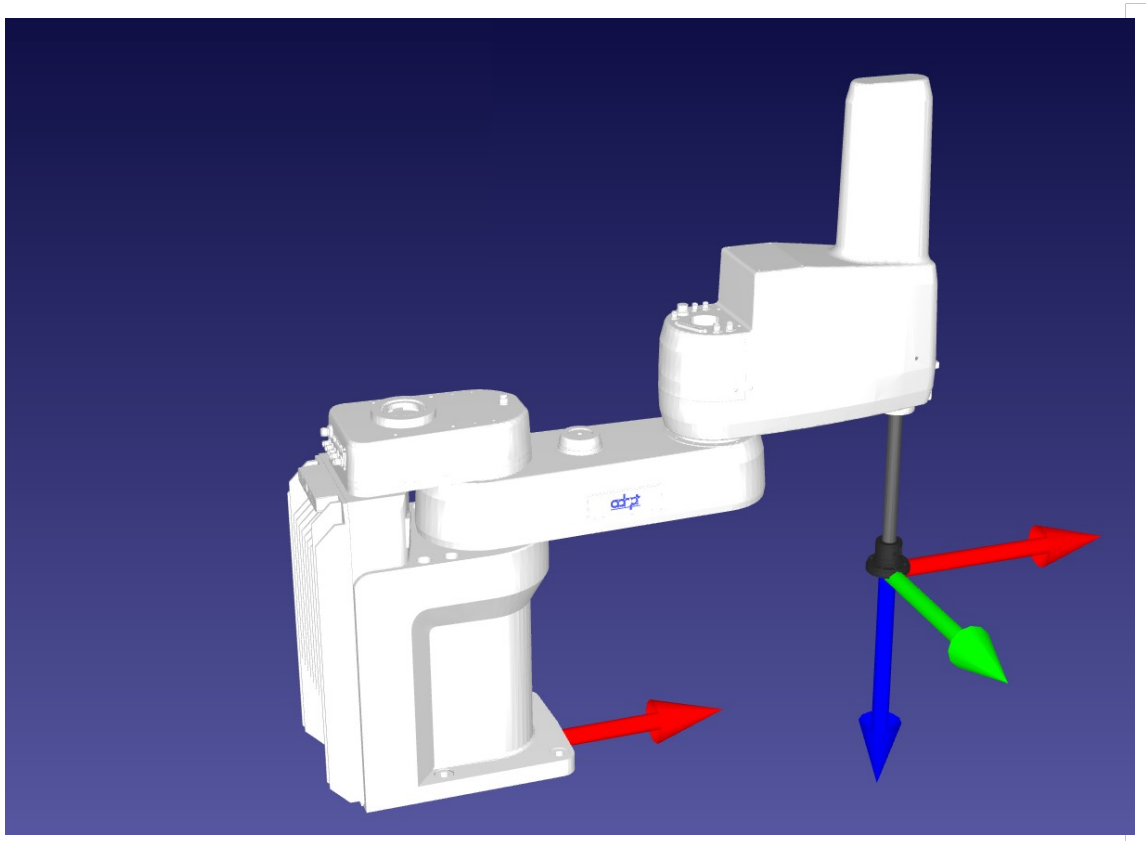


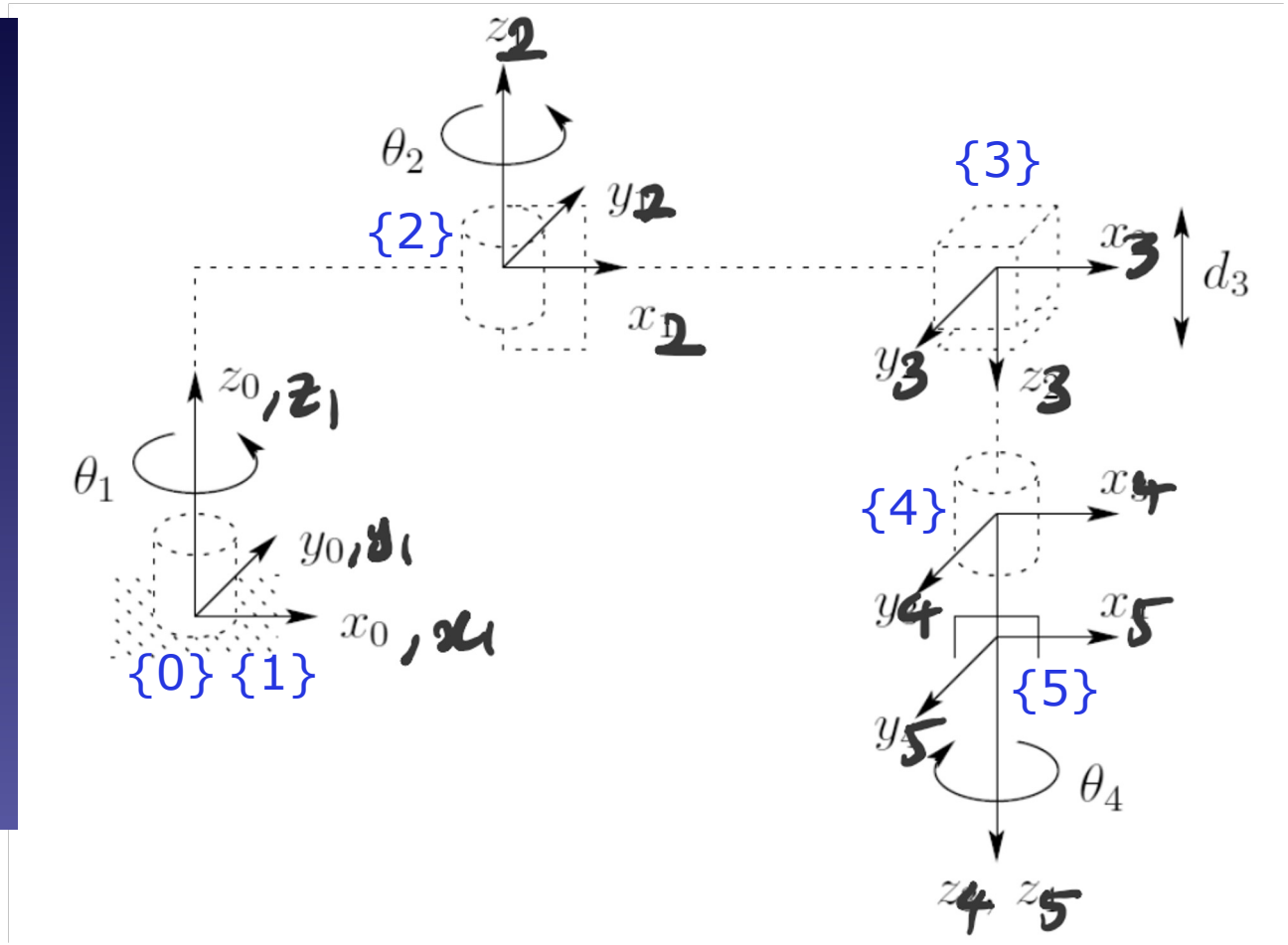
Specifications

Brand	Adept
Model	Cobra s600
Type	Scara Robot
Axes	4
Payload	2.0 kg
Reach	600 mm
Repeatability	0.018 mm
Weight	41 kg

https://robodk.com/robot/Adept/Cobra-s600

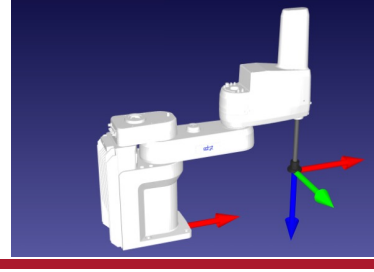
J_p plus J_o for Cobra - Example 1





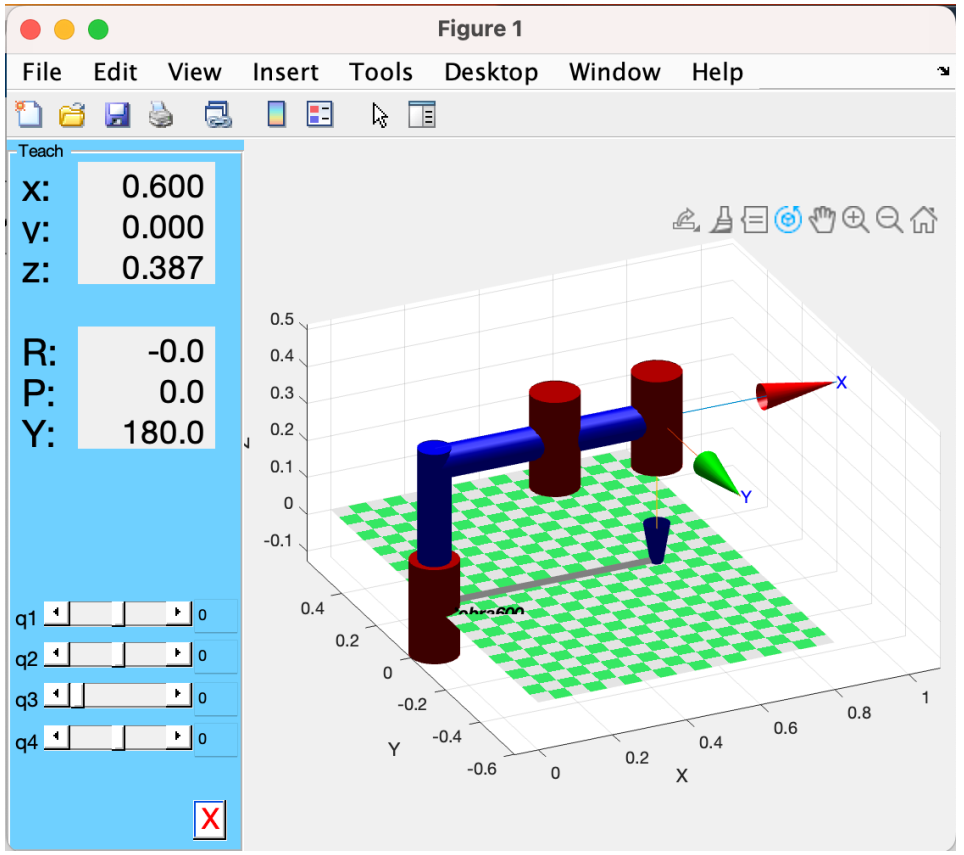
https://www.youtube.com/watch?v=vKD20BTkXhk

J_p plus J_o for Cobra - Example 1 - MATLAB

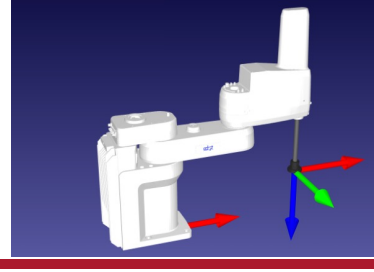


```
% Load SCARA robot
mdl_cobra600
c600
c600.teach()
```

```
c600 =

Cobra600 [Adept]:: 4 axis, RRPR, stdDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|      0.387|      0.325|          0|          0|
|  2|         q2|          0|      0.275|    3.14159|          0|
|  3|          0|         q3|          0|          0|          0|
|  4|         q4|          0|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```



J_p plus J_o for Cobra - Example 1

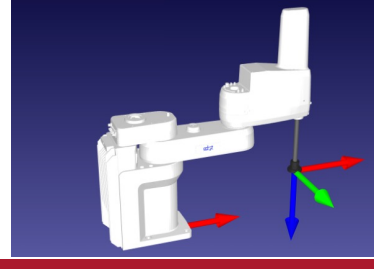


J_p — Derivative of EE Position → $\frac{\partial p_e}{\partial q_i}$

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$J(q) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

J_p plus J_o for Cobra - Example 1



$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad z_1$$

$$T_2^0 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad z_2$$

$$T_3^0 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} c_{12} & -s_{12} & 0 & a_1c_1 + a_2c_{12} \\ s_{12} & c_{12} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad z_4$$

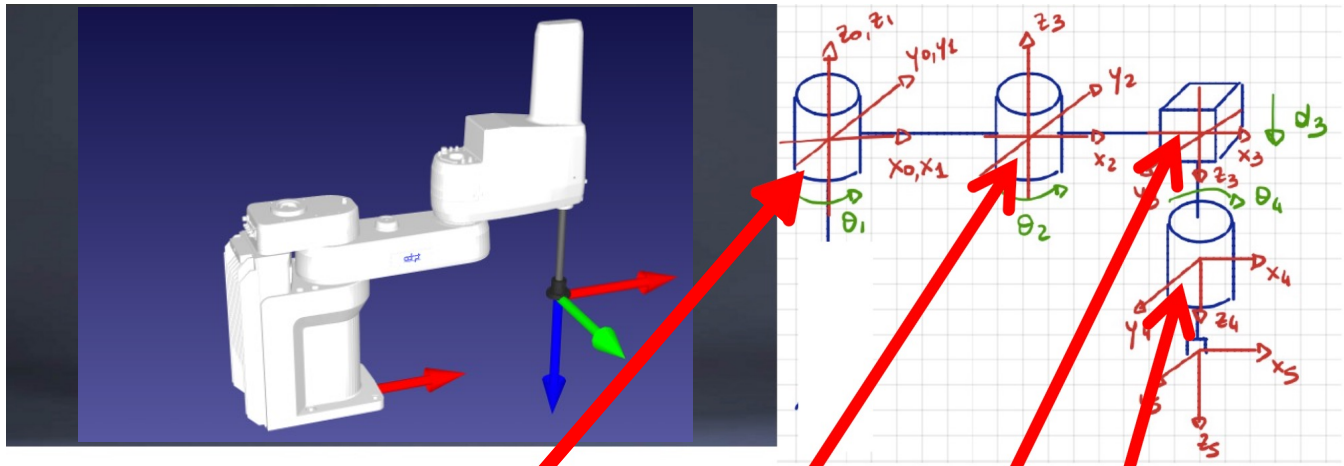
J_o — Last column of Rotation Matrix → $\hat{z}_i$ R_i, 0

$$J(q) = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}$$

J_{o1} J_{o2} J_{o3} J_{o4}

z_1 z_2 $\neq z_3$ (prismatic) z_4

J_p plus J_o for Cobra - Example 1 - Full Jacobian



$$J = \begin{bmatrix} J_1 & J_2 & J_3 & J_4 \end{bmatrix} = \begin{bmatrix} J_{p1} & J_{p2} & J_{p3} & J_{p4} \\ J_{o1} & J_{o2} & J_{o3} & J_{o4} \end{bmatrix} = \begin{bmatrix} J_p \\ J_o \end{bmatrix}$$

$$\begin{bmatrix} -a_1 s_1 - a_2 s_{12} & -a_2 s_{12} & 0 & 0 \\ a_1 c_1 + a_2 c_{12} & a_2 c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}$$

MATLAB
Full Jacobian of Cobra - Example 1

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                      -1,                                                      0.3870 - d3]
[                 0,                  0,                       0,                                                               1]
```

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                      -1,                                                      0.3870 - d3]
[                 0,                  0,                       0,                                                               1]
```

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                      -1,                                                      0.3870 - d3]
[                 0,                  0,                       0,                                                               1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                        0.3870 - d3
```

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t

% Jacobian
Jo = c600.jacob0([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                      -1,                                                      0.3870 - d3]
[                 0,                  0,                       0,                                                               1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                          0.3870 - d3
```

```
Jo =

[cos(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) - 1.2246e-16*sin(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) - sin(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
[1.2246e-16*cos(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) + cos(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + sin(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)),
[                      3.9801e-17*cos(q2) - 1.2246e-16*cos(q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + 1.2246e-16*sin(q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) + 3.3678e-17,
[                                                                                                                                                                                                1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
[                                                                                                                                                                                                1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
[                                                                                                                                                                                                                                                 1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,
```

$$J = [J_1 \quad \begin{bmatrix} -a_1s_1 - a_2s_{12} \\ a_1c_1 + a_2c_{12} \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t

% Jacobian
Jo = c600.jacob0([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                       -1,                                                      0.3870 - d3]
[                 0,                  0,                        0,                                                                1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                      0.3870 - d3
```

```
Jo =

[cos(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) - 1.2246e-16*sin(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) - sin(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
[1.2246e-16*cos(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) + cos(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + sin(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)),
[                3.9801e-17*cos(q2) - 1.2246e-16*cos(q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + 1.2246e-16*sin(q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) + 3.3678e-17,
[                                                                                                  1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
[                                                                                                  1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
[                                                                                                                                                  1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,
```

```
- 4.1243e-33*sin(q1 + q2) - cos(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)) - sin(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
  4.1243e-33*cos(q1 + q2) + cos(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - sin(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)),
          3.3678e-17 - 1.2246e-16*cos(q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - 1.2246e-16*sin(q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)),
                                 1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
                                 1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
                                                                                  1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,
```

$$J = [J_1 \quad J_2 \quad \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} \\ a_1c_1 + a_2c_{12} & a_2c_{12} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

MATLAB

Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t

% Jacobian
Jo = c600.jacob0([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                       -1,                                                          0.3870 - d3]
[                 0,                  0,                        0,                                                                    1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                      0.3870 - d3
```

```
Jo =

[cos(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) - 1.2246e-16*sin(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) - sin(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
[1.2246e-16*cos(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) + cos(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + sin(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)),
[                     3.9801e-17*cos(q2) - 1.2246e-16*cos(q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + 1.2246e-16*sin(q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) + 3.3678e-17,
[                                                                                                                      1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
[                                                                                                                      1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
[                                                                                                                                                         1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,
```

```
- 4.1243e-33*sin(q1 + q2) - cos(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)) - sin(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),  1.2246e-16*sin(q1 + q2),
  4.1243e-33*cos(q1 + q2) + cos(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - sin(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)), -1.2246e-16*cos(q1 + q2),
           3.3678e-17 - 1.2246e-16*cos(q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - 1.2246e-16*sin(q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)),                       -1,
                                       1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),                        0,
                                       1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),                        0,
                                                                                          1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,                        0,
```

$$J = [J_1 \quad J_2 \quad J_3 \quad \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t

% Jacobian
Jo = c600.jacob0([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                       -1,                                                       0.3870 - d3]
[                 0,                  0,                        0,                                                                 1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                      0.3870 - d3
```

```
Jo =

[cos(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) - 1.2246e-16*sin(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) - sin(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
[1.2246e-16*cos(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) + cos(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + sin(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)),
[                    3.9801e-17*cos(q2) - 1.2246e-16*cos(q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + 1.2246e-16*sin(q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) + 3.3678e-17,
[                                                                                              1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
[                                                                                              1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
[                                                                                                                                         1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,

- 4.1243e-33*sin(q1 + q2) - cos(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)) - sin(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),  1.2246e-16*sin(q1 + q2),                        0]
  4.1243e-33*cos(q1 + q2) + cos(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - sin(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)), -1.2246e-16*cos(q1 + q2),                        0]
           3.3678e-17 - 1.2246e-16*cos(q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - 1.2246e-16*sin(q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)),                       -1,                        0]
                                        1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),                        0,  1.2246e-16*sin(q1 + q2)]
                                        1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),                        0, -1.2246e-16*cos(q1 + q2)]
                                                                                     1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,                        0,                       -1]
```

$$J = [J_1 \quad J_2 \quad J_3 \quad J_4] = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}$$

MATLAB
Full Jacobian of Cobra - Example 1

$$T_5^0 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
% Forward Kinematics
syms q1 q2 d3 q4 real

FK = c600.fkine([q1 q2 d3 q4])
Pe = FK.t

% Jacobian
Jo = c600.jacob0([q1 q2 d3 q4])
```

```
[ cos(q1 + q2 - q4),  sin(q1 + q2 - q4),  1.2246e-16*sin(q1 + q2), 0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)]
[ sin(q1 + q2 - q4), -cos(q1 + q2 - q4), -1.2246e-16*cos(q1 + q2), 0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)]
[1.2246e-16*sin(q4), 1.2246e-16*cos(q4),                       -1,                                                       0.3870 - d3]
[                 0,                  0,                        0,                                                                 1]


Pe =

0.2750*cos(q1 + q2) + 0.3250*cos(q1) + 1.2246e-16*d3*sin(q1 + q2)
0.2750*sin(q1 + q2) + 0.3250*sin(q1) - 1.2246e-16*d3*cos(q1 + q2)
                                                      0.3870 - d3
```

```
Jo =

[cos(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) - 1.2246e-16*sin(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) - sin(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),
[1.2246e-16*cos(q1 + q2)*(3.9801e-17*cos(q2) + 3.3678e-17) + cos(q1 + q2 - q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + sin(q1 + q2 - q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)),
[                       3.9801e-17*cos(q2) - 1.2246e-16*cos(q4)*(0.3250*cos(q2 - q4) + 0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) + 1.2246e-16*sin(q4)*(0.3250*sin(q2 - q4) - 0.2750*sin(q4) + 1.2246e-16*d3*cos(q4)) + 3.3678e-17,
[                                                                                                                         1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),
[                                                                                                                         1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),
[                                                                                                                                                                          1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,
```

```
- 4.1243e-33*sin(q1 + q2) - cos(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)) - sin(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)),  1.2246e-16*sin(q1 + q2),                        0]
  4.1243e-33*cos(q1 + q2) + cos(q1 + q2 - q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - sin(q1 + q2 - q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)), -1.2246e-16*cos(q1 + q2),                        0]
           3.3678e-17 - 1.2246e-16*cos(q4)*(0.2750*cos(q4) + 1.2246e-16*d3*sin(q4)) - 1.2246e-16*sin(q4)*(0.2750*sin(q4) - 1.2246e-16*d3*cos(q4)),                       -1,                        0]
                                          1.2246e-16*cos(q1 + q2 - q4)*sin(q4) - 1.2246e-16*sin(q1 + q2) + 1.2246e-16*sin(q1 + q2 - q4)*cos(q4),                        0,  1.2246e-16*sin(q1 + q2)]
                                          1.2246e-16*cos(q1 + q2) - 1.2246e-16*cos(q1 + q2 - q4)*cos(q4) + 1.2246e-16*sin(q1 + q2 - q4)*sin(q4),                        0, -1.2246e-16*cos(q1 + q2)]
                                                                                       1.4998e-32*cos(q4)^2 + 1.4998e-32*sin(q4)^2 + 1,                        0,                       -1]
```

$$J = [J_1 \quad J_2 \quad J_3 \quad J_4] = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}$$

NOTE : A lot of terms are equal to zero in the equation (Jo) above

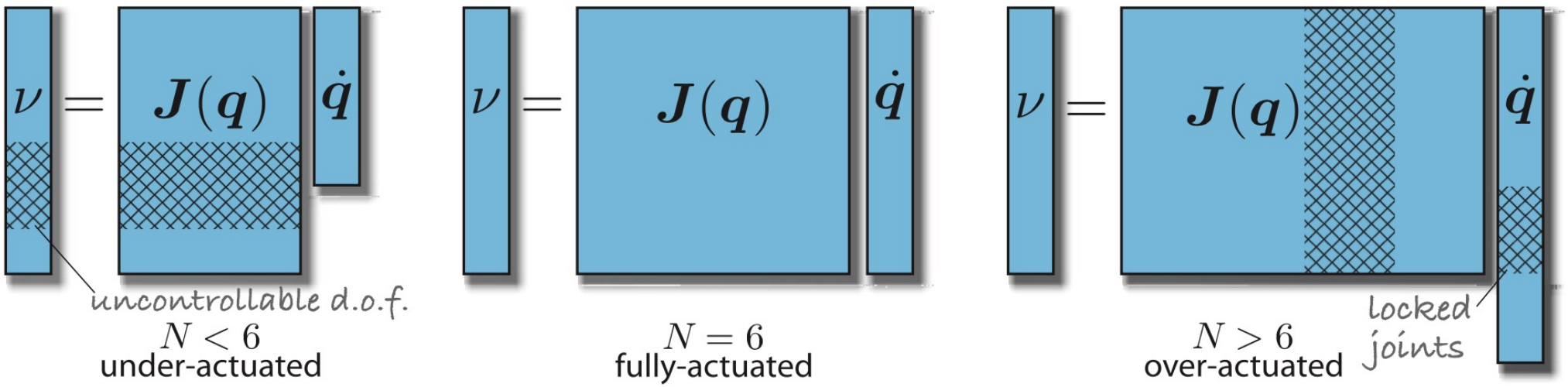
Recap: Geometric Jacobian

- For a robotic manipulator, the Jacobian is a 6xn matrix, where n is the number of joints:

$$J = \begin{bmatrix} J_1 & J_2 & J_3 & J_4 \end{bmatrix} = \begin{bmatrix} J_{p1} & J_{p2} & J_{p3} & J_{p4} \\ J_{o1} & J_{o2} & J_{o3} & J_{o4} \end{bmatrix} = \begin{bmatrix} J_p \\ J_o \end{bmatrix}$$

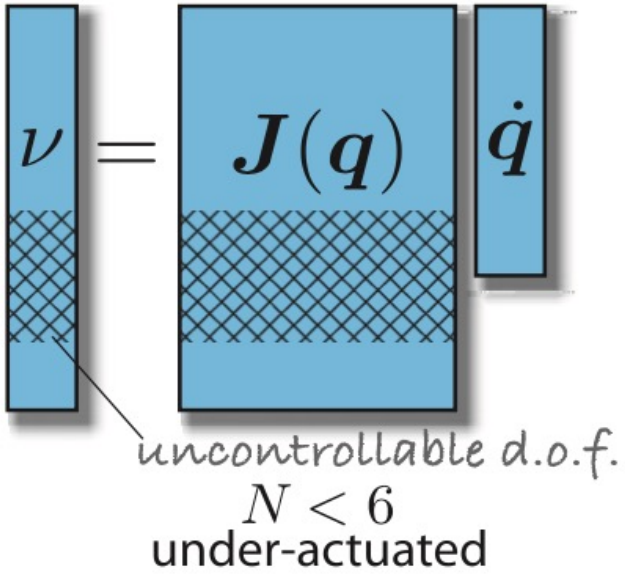
- We Solve Jacobian for each joint i.e., each column
- We Solve for *Position* ($\boldsymbol{J_p}$ i.e., upper half) Separately
- We Solve for *Orientation* ($\boldsymbol{J_o}$ i.e., lower half) Separately

Under / Fully / Over - actuated manipulator



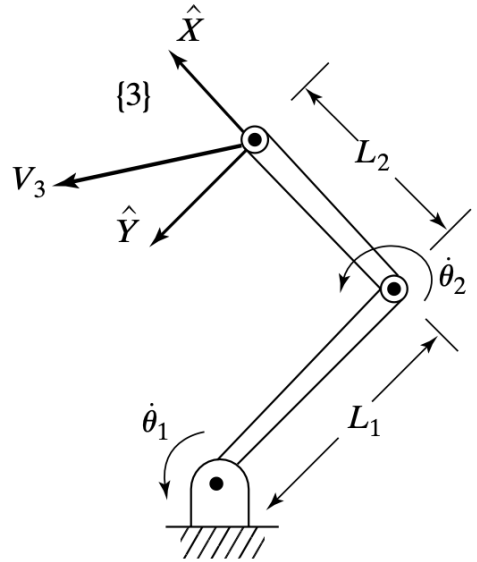
Jacobian for under-actuated manipulator

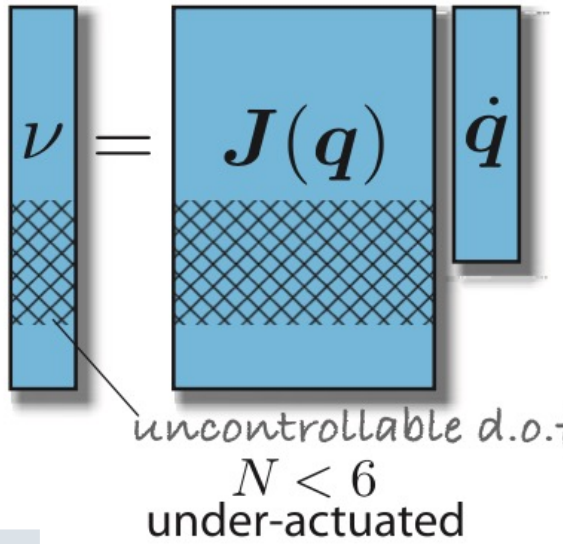
- An under-actuated robot has $N < 6$, and a Jacobian that is taller than it is wide.
- The system of equations is under -constrained but the system can be *squared up* by deleting some rows of v and J
  - accepting that some Cartesian degrees of freedom are not controllable given the low number of joints



Jacobian for under-Actuated "RR" - Example 2

$$\dot{q} = J(q)^{-1}\nu$$






```
>> mdl_planar2
>> qn = [1 1];
>> J = p2.jacob0(qn)
J =
   -1.7508   -0.9093
    0.1242   -0.4161
         0         0
         0         0
         0         0
    1.0000    1.0000
```

```
>> qd = pinv(J) * [0.1 0 0 0 0 0]'
qd =
   -0.0698
    0.0431
```

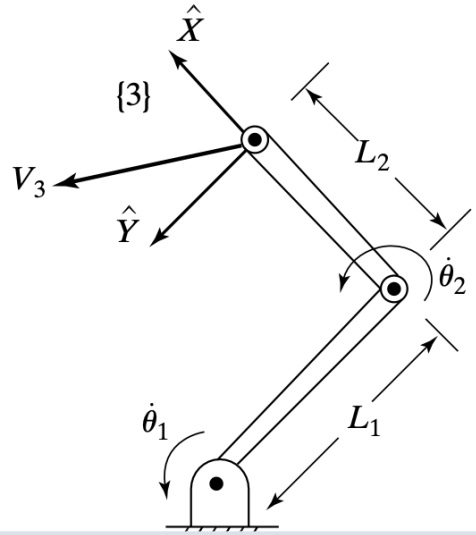
```
>> xd = J*qd;
>> xd'
ans =
    0.0829   -0.0266         0         0         0   -0.0266
```

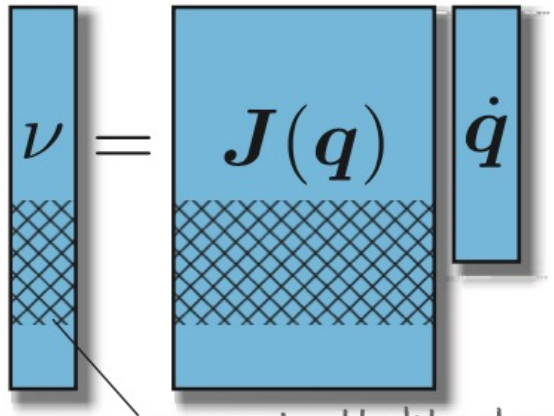
Jacobian for under-Actuated "RR" - Example 2

$$\begin{pmatrix} \nu_x \\ \nu_y \\ \hline \nu_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} J_{xy} \\ \hline J_0 \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

$$\dot{q} = J(q)^{-1}\nu$$

$${}^0\nu = {}^0J(q)\dot{q}$$






```
>> Jxy = J(1:2,:);
>> qd = inv(Jxy)* [0.1 0]'
qd =
   -0.0495
   -0.0148
```

$$\begin{pmatrix} \nu_x \\ \nu_y \end{pmatrix} = J_{xy} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = J_{xy}^{-1} \begin{pmatrix} \nu_x \\ \nu_y \end{pmatrix}$$

```
>> xd = J*qd;
>> xd'
ans =
    0.1000    0.0000         0         0         0   -0.0642
```

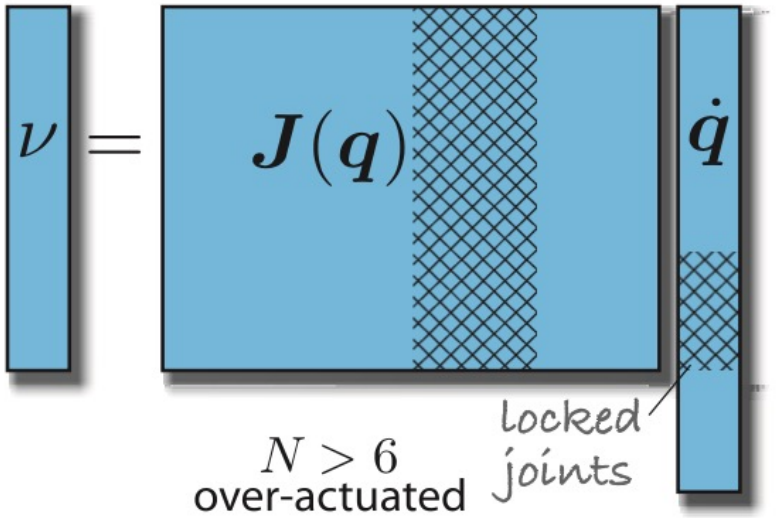
Over-actuated "Redundant" manipulators

- An over-actuated or redundant robot has $N > 6$, and a Jacobian that is wider than it is tall
- The system of equations is under -constrained and the best we can do is find a least-squares solution. Alternatively, we can *square up* the Jacobian to make it invertible by deleting some columns – effectively *locking* the corresponding joints

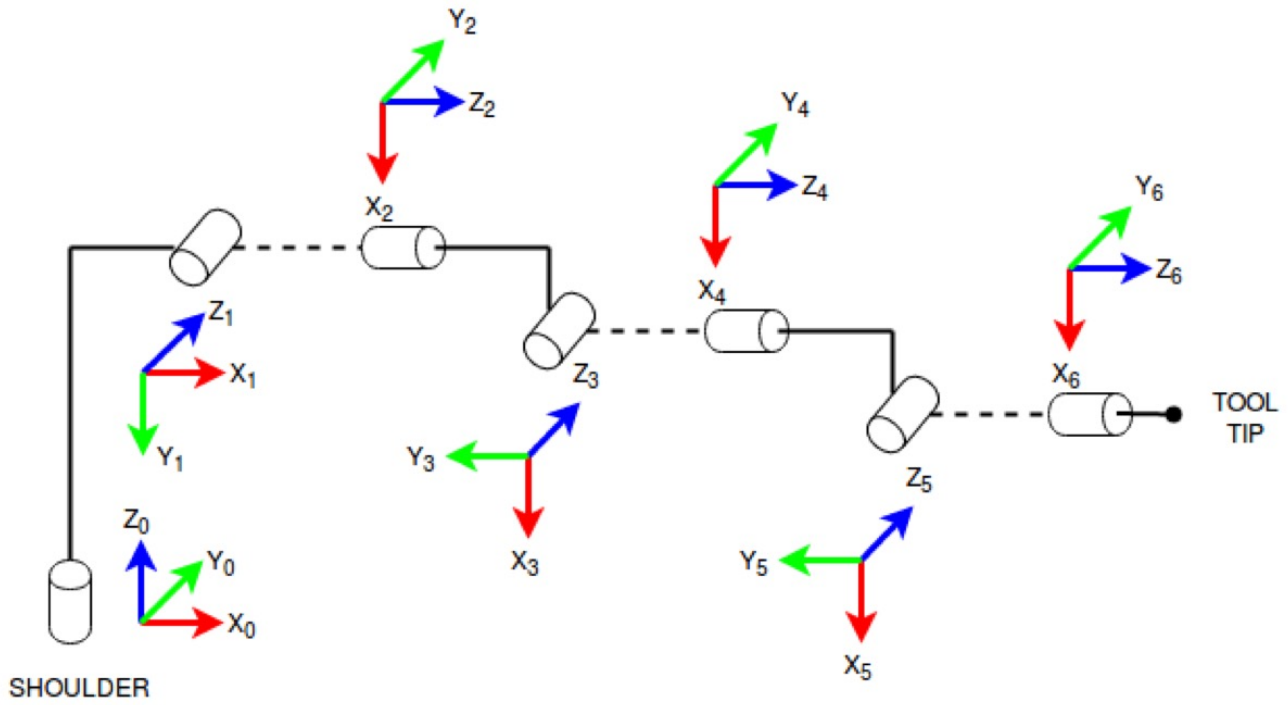
$$\dot{q} = J(q)^{-1}\nu$$

$$\dot{q} = J(q)^{+}\nu$$

$$\dot{q} = \underbrace{J(q)^{+}\nu}_{\text{end-effector motion}} + \underbrace{NN^{+}\dot{q}_{\text{null}}}_{\text{null-space motion}}$$



Baxter (7 DoF)

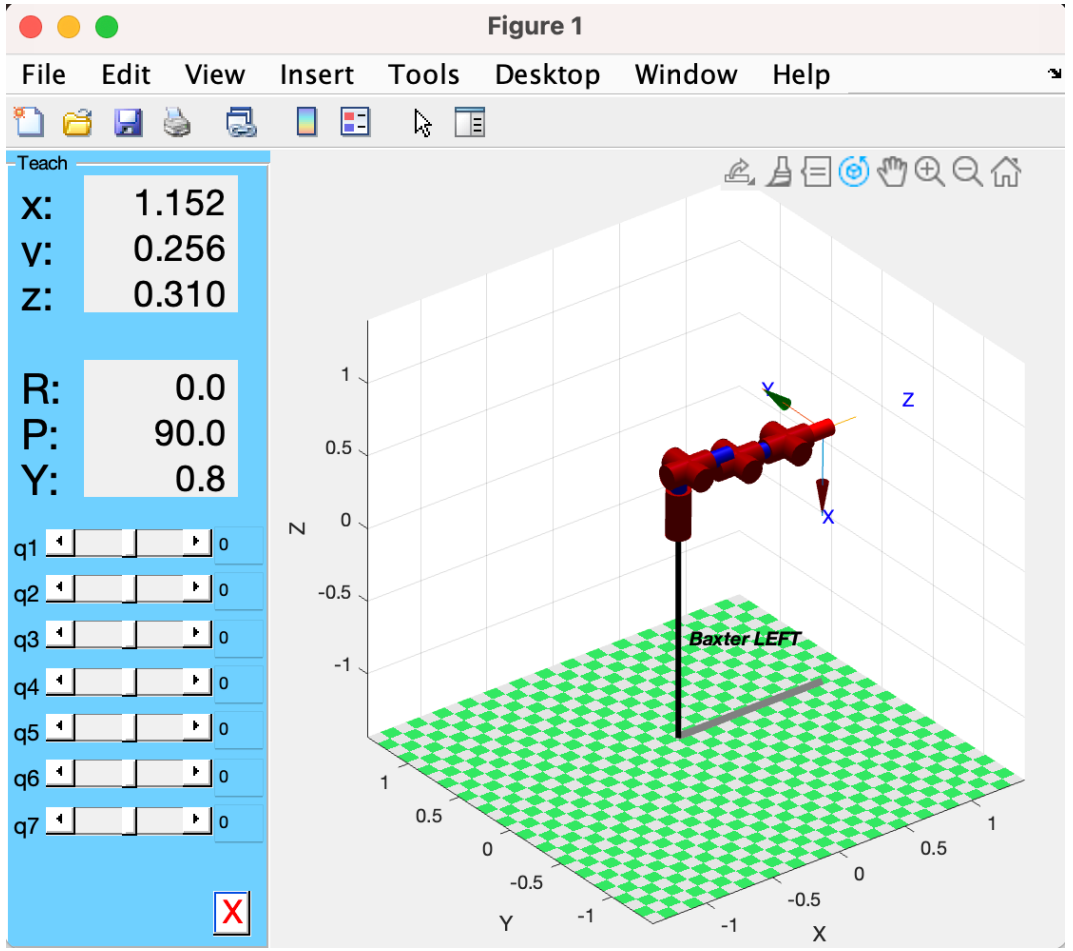


https://en.wikipedia.org/wiki/Baxter_(robot)
https://sanjuksha.github.io/BaxterProject/baxter_report.pdf

Baxter (7 DoF) - Robotics Toolbox (RTB)

```
>> mdl_baxter
>> TE = SE3(0.8, 0.2, -0.2) * SE3.Ry(pi);
>> q = left.ikine(TE)
```

```
>> J = jacob0(left, q);
>> about J
J [double] : 6x7 (336 bytes)
```



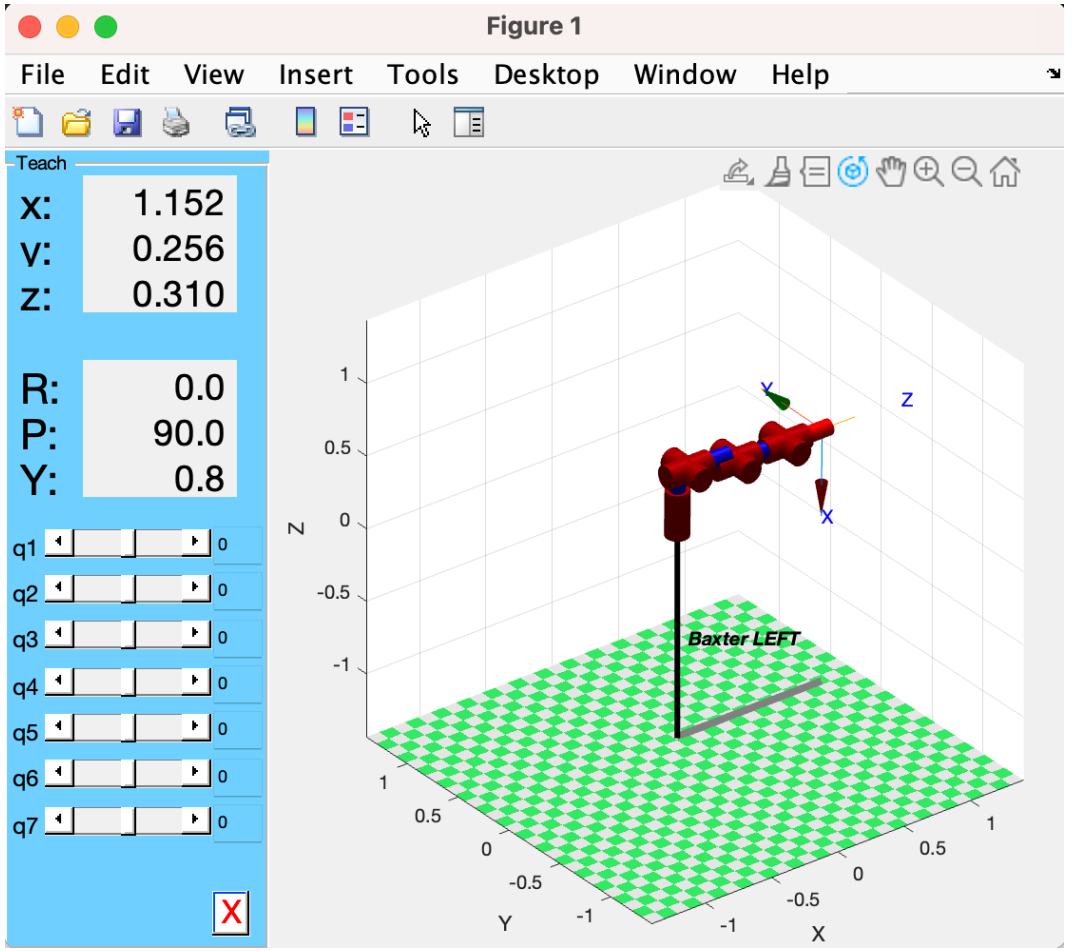
Jacobian for Over-actuated "Baxter" - Example 3

```
>> mdl_baxter
>> TE = SE3(0.8, 0.2, -0.2) * SE3.Ry(pi);
>> q = left.ikine(TE)
```

```
>> J = jacob0(left, q);
>> about J
J [double] : 6x7 (336 bytes)
```

```
>> xd = [0.2 0.2 0.2 0 0 0]';
>> qd = pinv(J) * xd;
>> qd'
ans =
       0.0895   -0.0464   -0.4259    0.6980   -0.4248    1.0179    0.2998
```

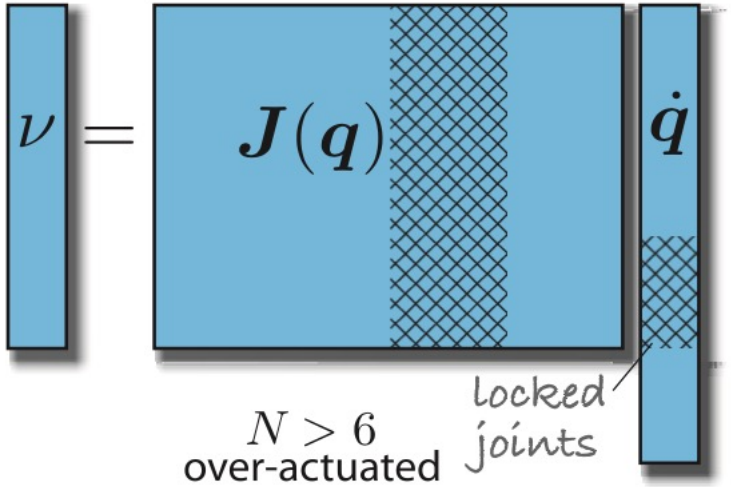
$$\dot{q} = J(q)^{+}\nu$$

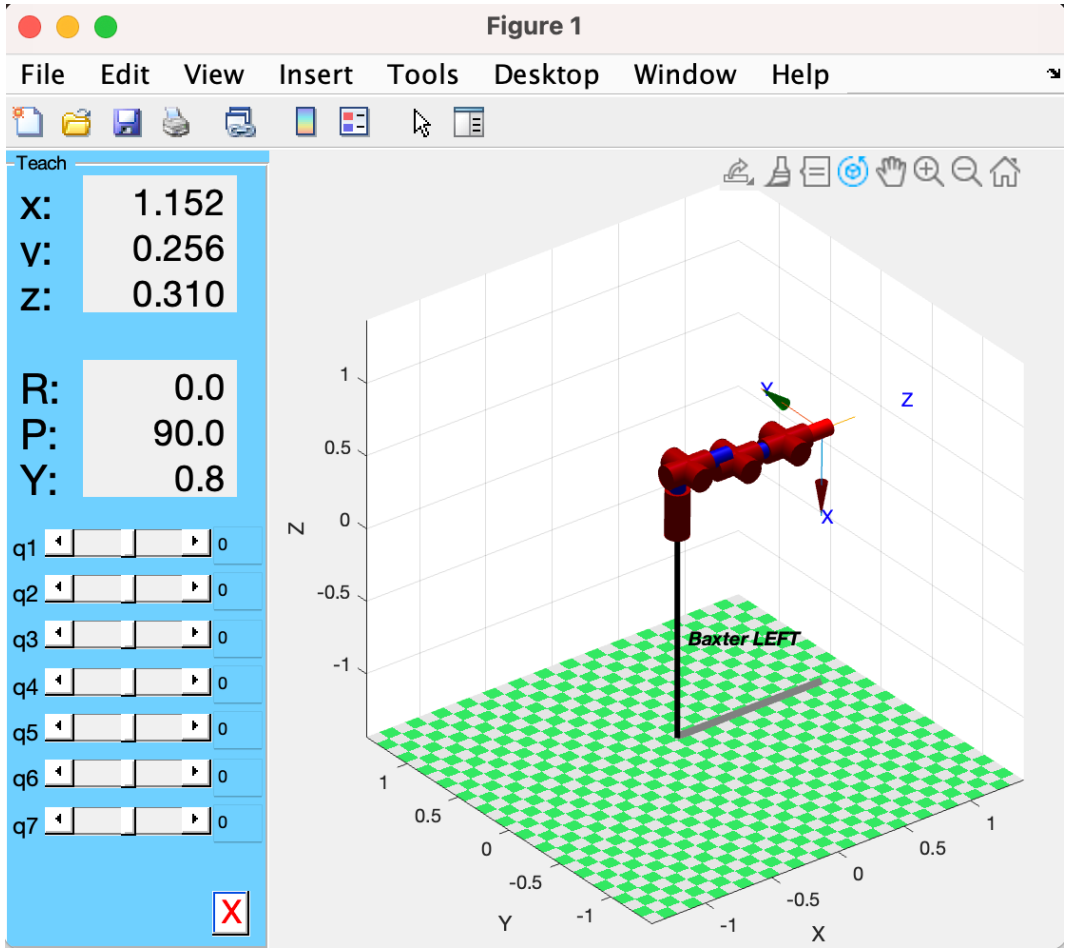


Jacobian for Over-actuated "Baxter" - Example 3

```
>> rank(J)
ans =
     6
```

```
>> N = null(J)
N =
   -0.2244
   -0.1306
    0.6018
    0.0371
   -0.7243
    0.0653
    0.2005
```



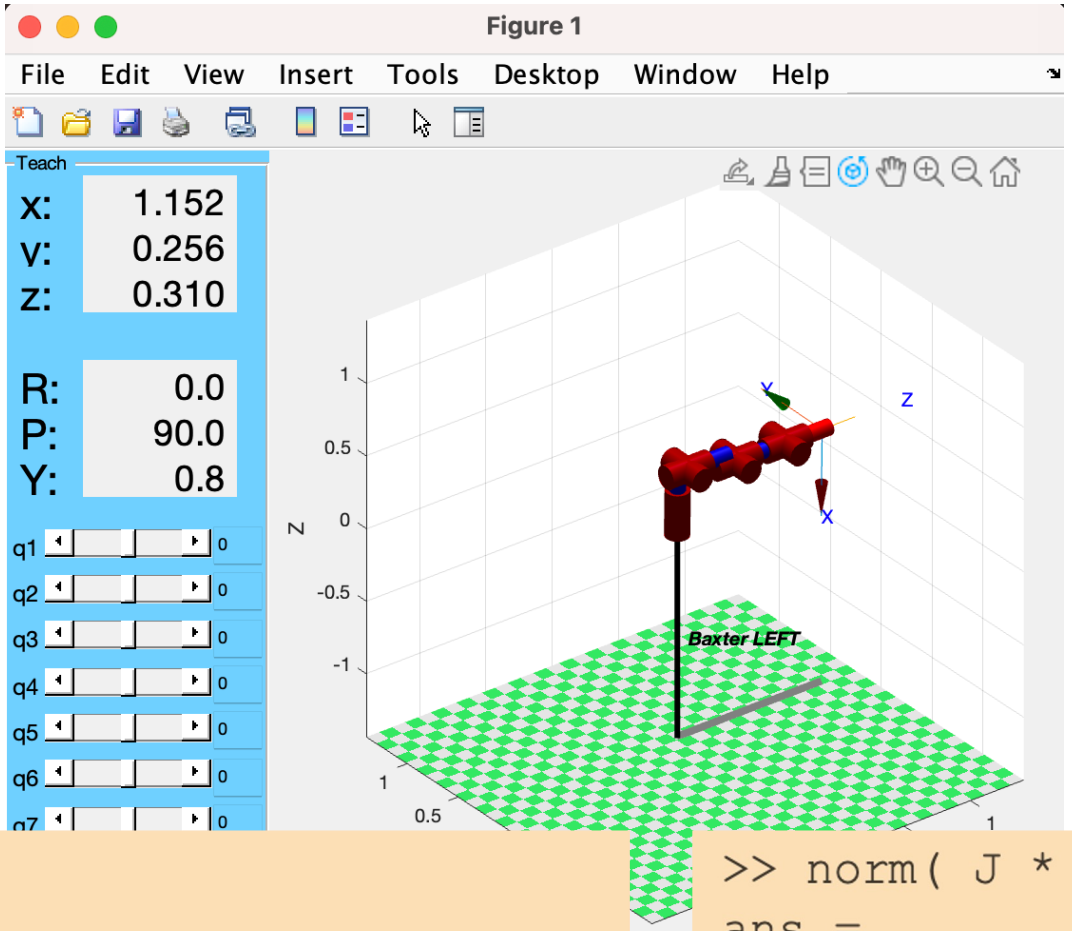


Jacobian for Over-actuated "Baxter" - Example 3

```
>> norm( J * N(:,1))
ans =
   2.6004e-16
```

$$\dot{q} = \underbrace{J(q)^{+}\nu}_{\text{end-effector motion}} + \underbrace{NN^{+}\dot{q}_{\text{null}}}_{\text{null-space motion}}$$

```
>> qd_null = [0 0 0 0 1 0 0]';
```

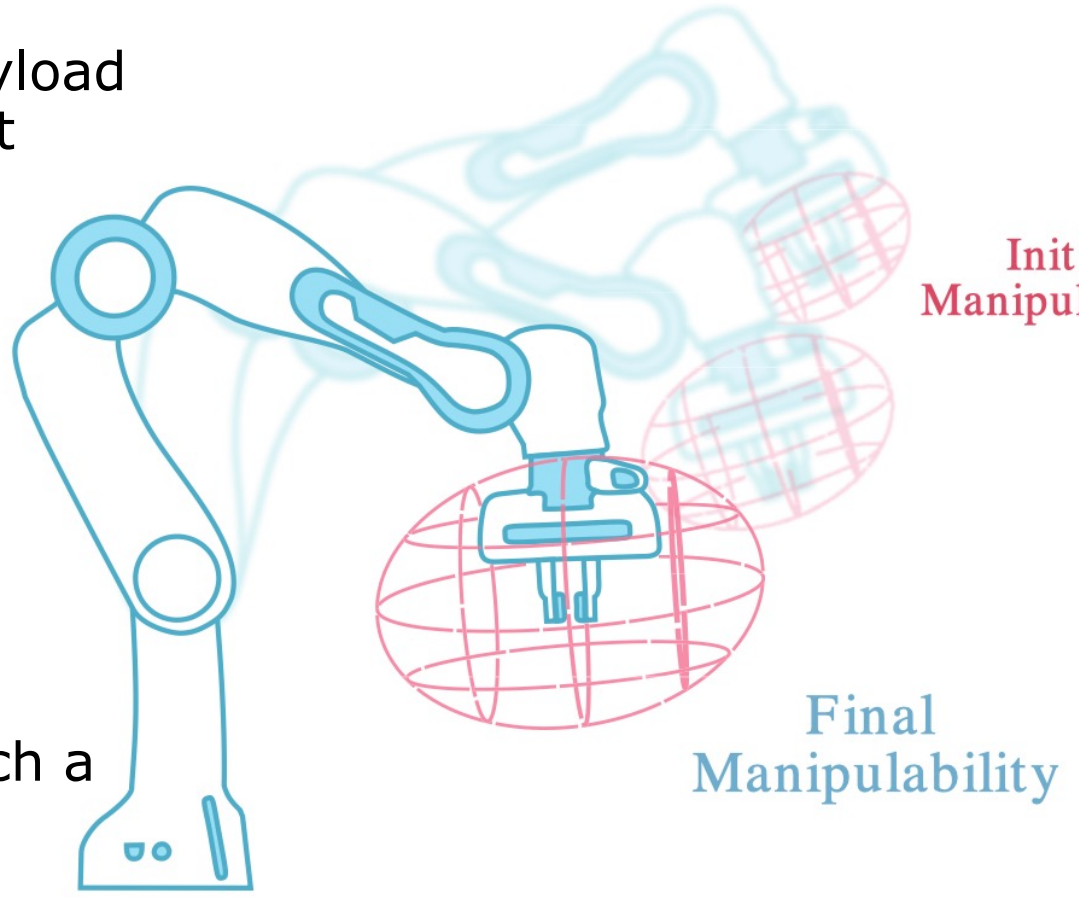



```
>> qp = N * pinv(N) * qd_null;
>> qp'
   0.1625    0.0946   -0.4359   -0.0269    0.5246   -0.0473   -0.1452
```

```
>> norm( J * qp)
ans =
   1.9541e-16
```

Manipulability

- Ability to move and control the tool or payload attached to the end of the arm in different directions and orientations
- Quantifies how effectively the robot can manipulate objects within its workspace
- Influenced by several factors:
  - kinematics
  - dynamics
  - joint configuration
- A highly manipulable robotic arm can reach a wide range of positions and orientations



Manipulability

- Represents the attitude of a manipulator to arbitrary change end-effector position and orientation using the *velocity manipulability ellipsoid*

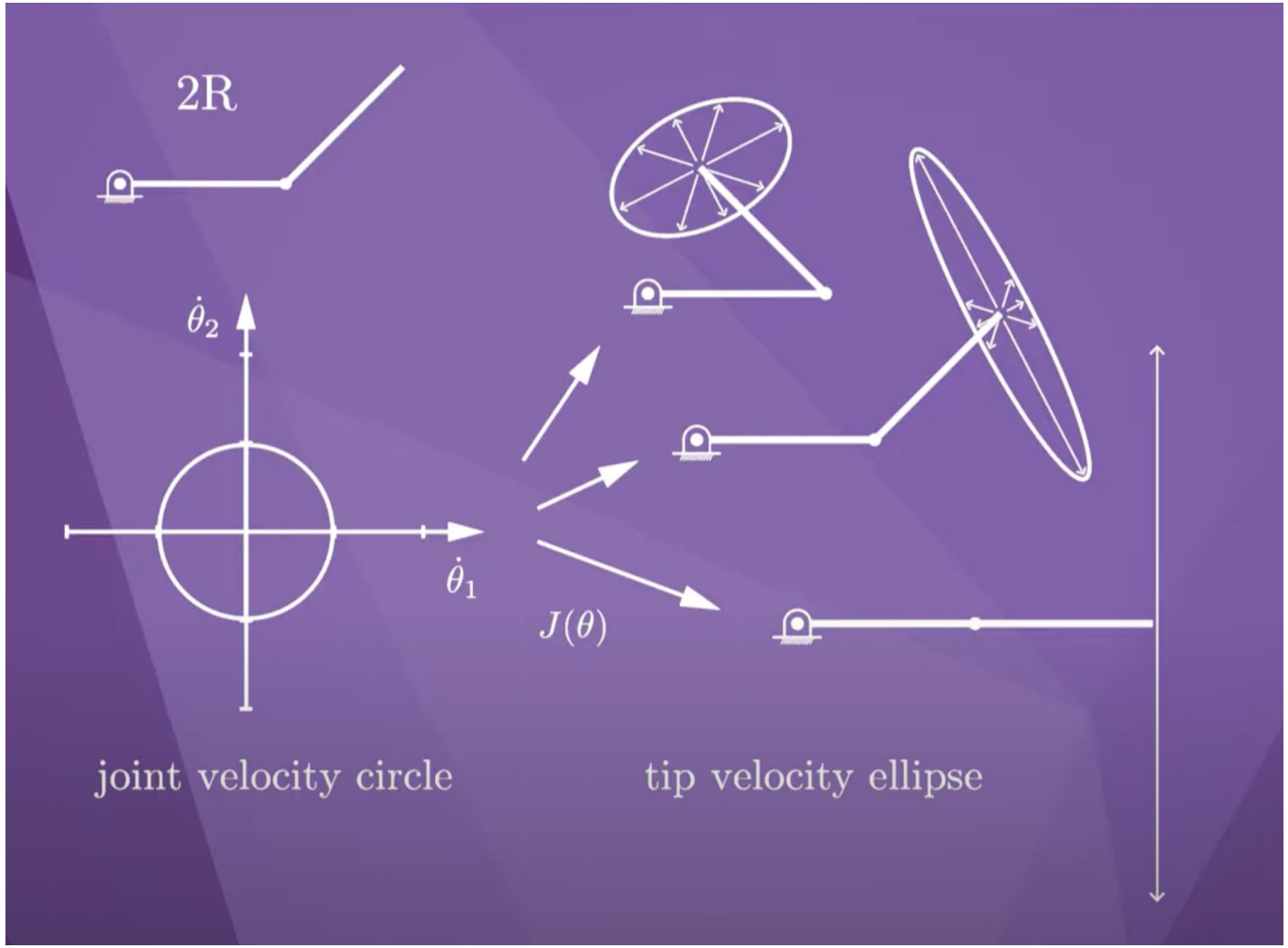
- Consider the set of joint velocities of constant (unit) norm:

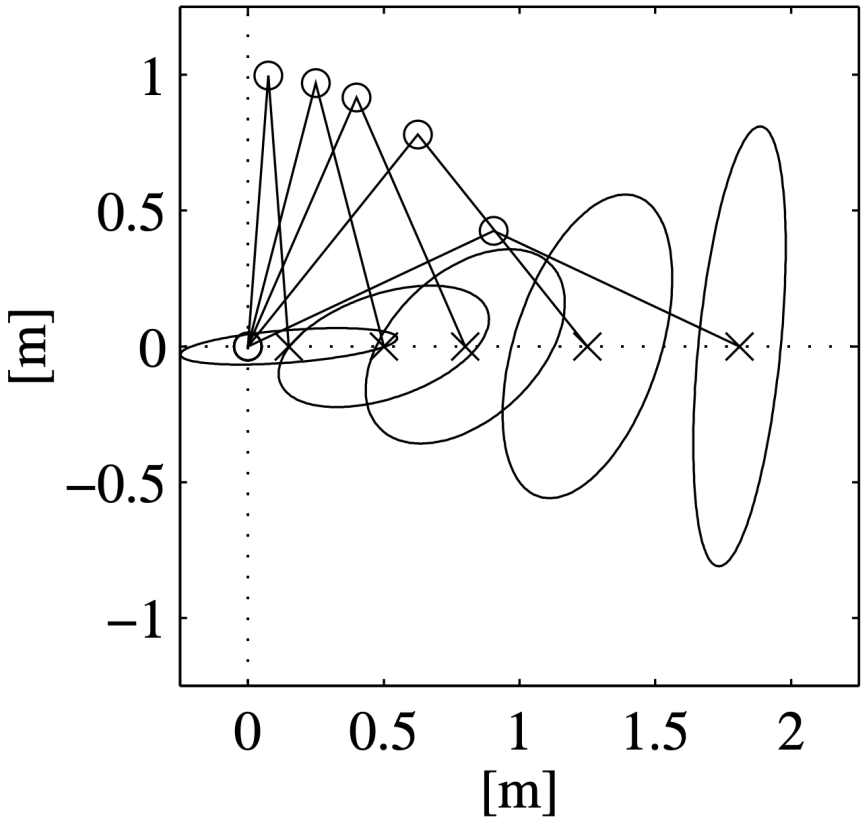
$$\dot{q}^T \dot{q} = 1$$

(*this equation describes the points on the surface of a sphere in the joint velocity space*)

$$\dot{q} = J(q)^{-1}\nu \longrightarrow \nu^T \left(J(q)J(q)^T\right)^{-1} \nu = 1$$

Manipulability Ellipsoid - RR planar



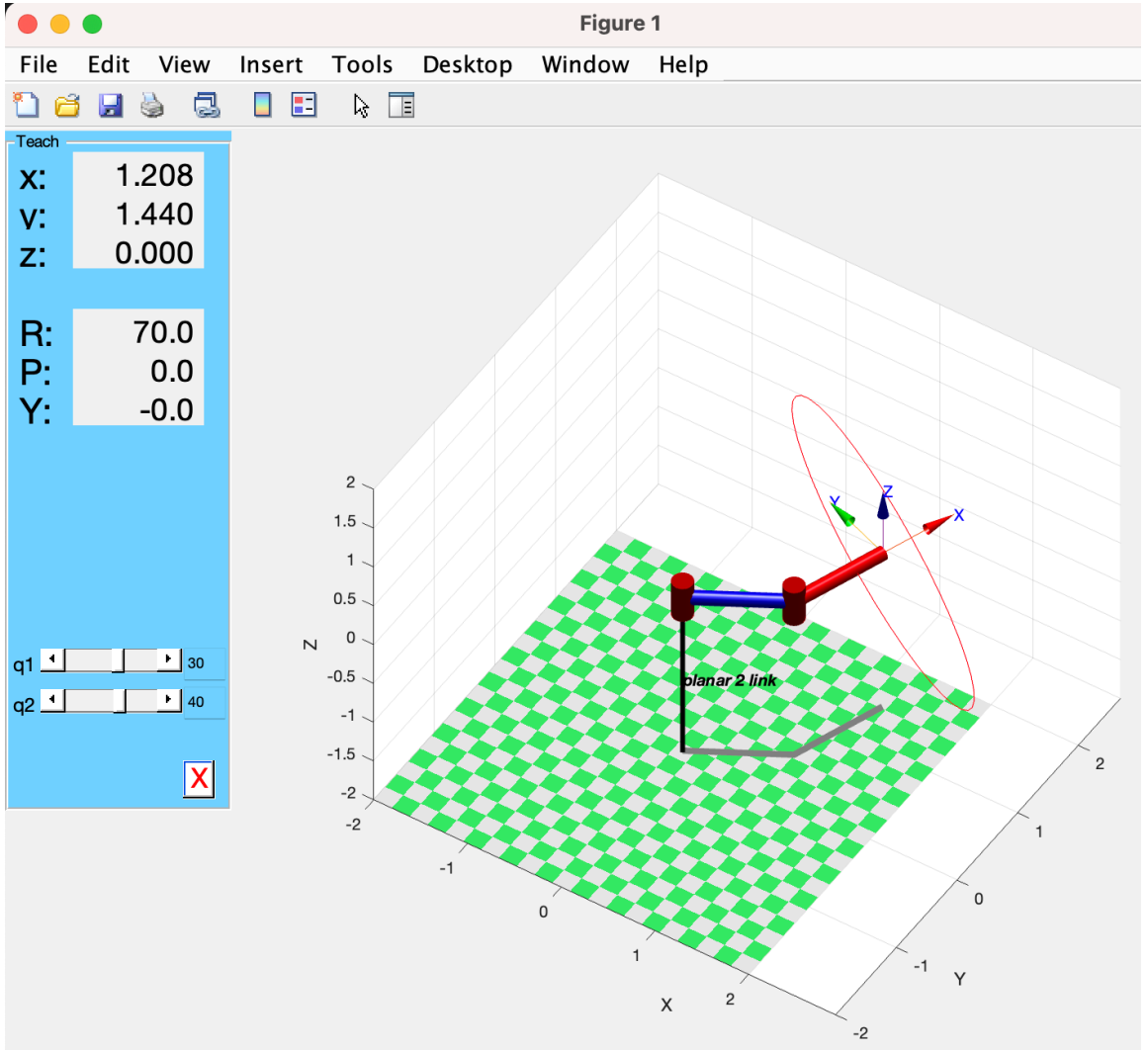


https://modernrobotics.northwestern.edu/nu-gm-book-resource/5-4-manipulability/

Manipulability Ellipsoid - RR planar

$$\nu^T \left(J(q) J(q)^T \right)^{-1} \nu = 1$$

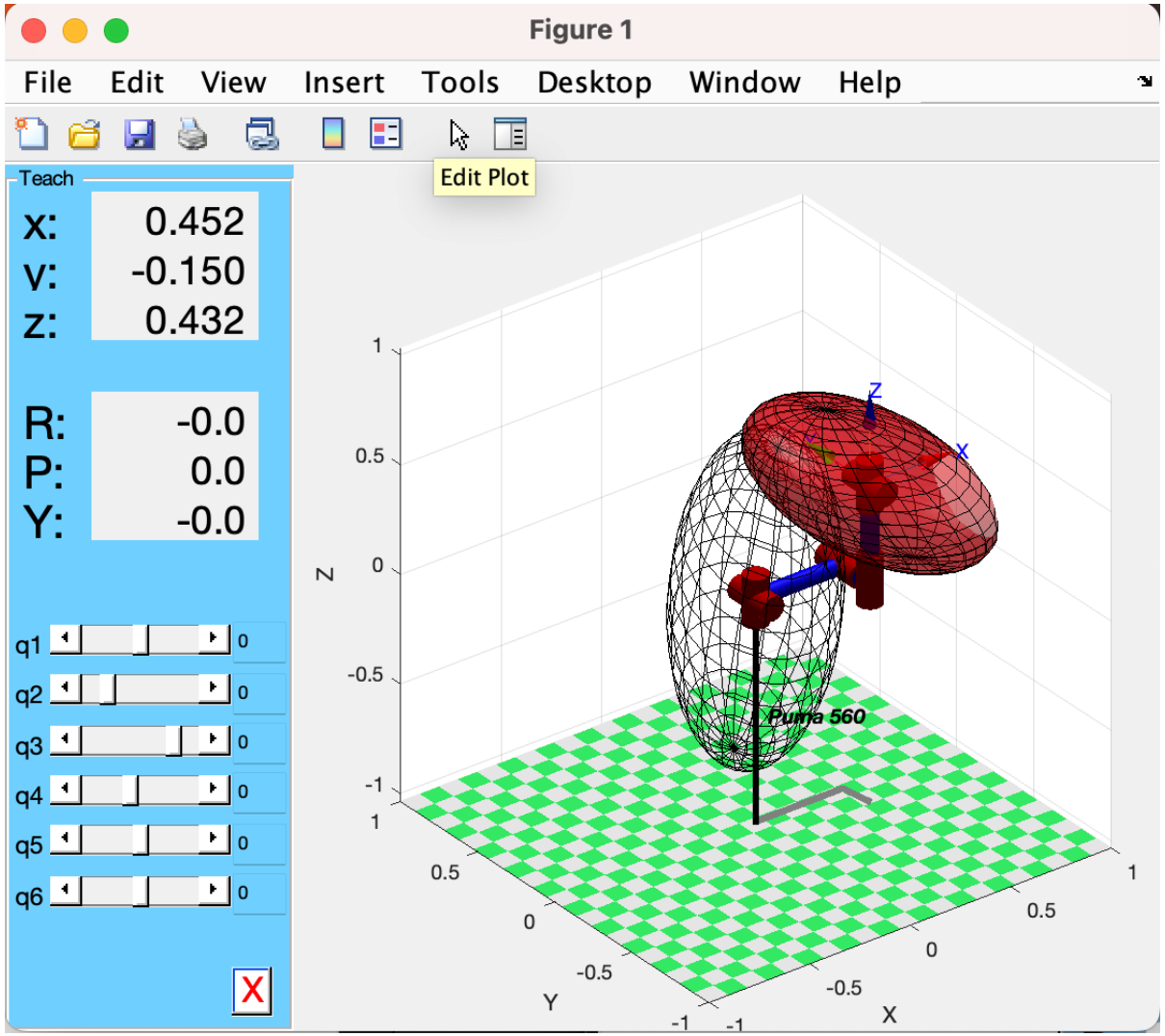
```
mdl_planar2
p2

qns = [30 40];
p2.vellipse(qns, 'deg')

p2.teach('callback', @(r,q) r.vellipse(q))
```



Manipulability Ellipsoid - PUMA 560

$$\nu^T \left(J(q) J(q)^T \right)^{-1} \nu = 1$$

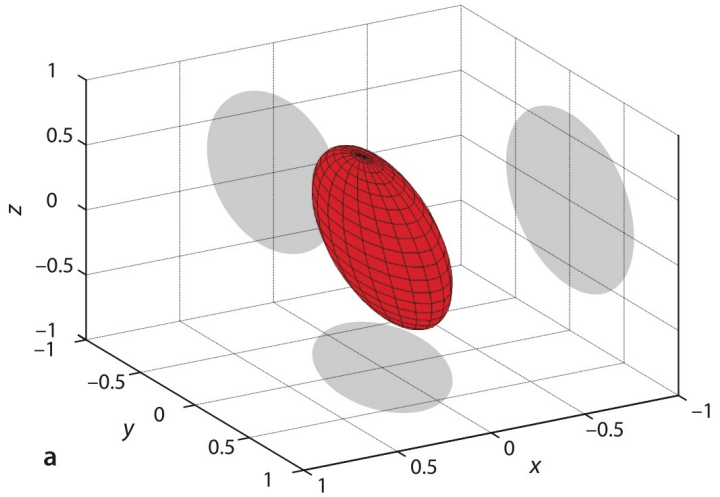
```
mdl_puma560
p560

J = p560.jacob0(qn)
Jp = J(1:3, :)

plot_ellipse(Jp * Jp')
```

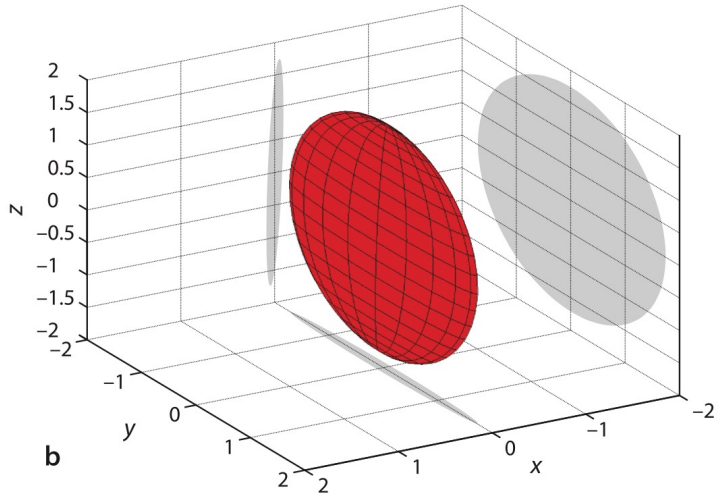


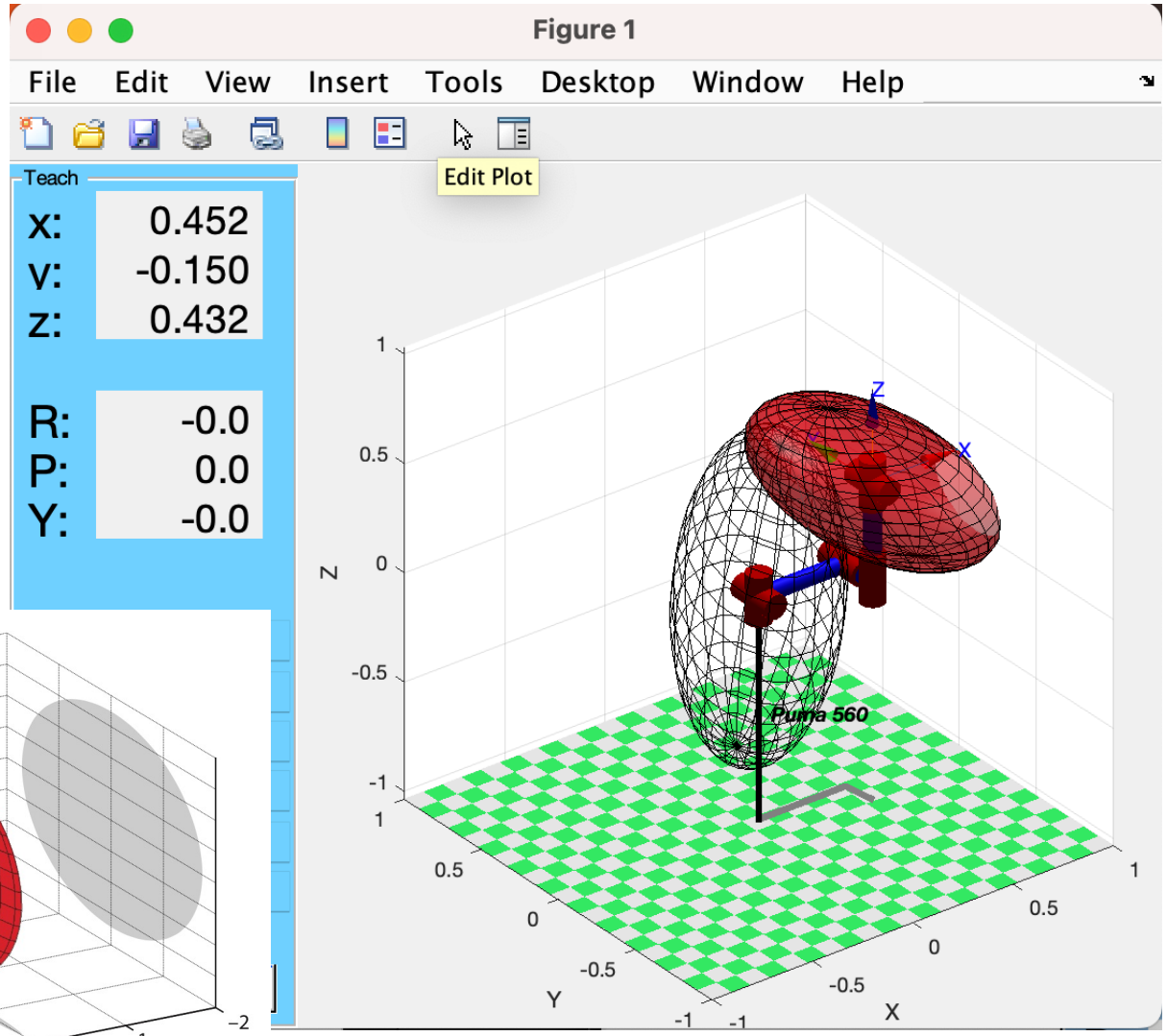
Manipulability Ellipsoid - PUMA 560

$$\nu^T \left(J(q) J(q)^T \right)^{-1} \nu = 1$$

```
mdl_puma560
p560

J = p560.jacob0(qn)
Jp = J(1:3, :)

plot_ellipse(Jp * Jp')

p560.vellipse(qn, 'trans');
p560.vellipse(qn, 'rot')
```







Manipulability measure

- A global representative measure of manipulation ability can be obtained by considering the volume of the ellipsoid. This volume is proportional to the quantity:

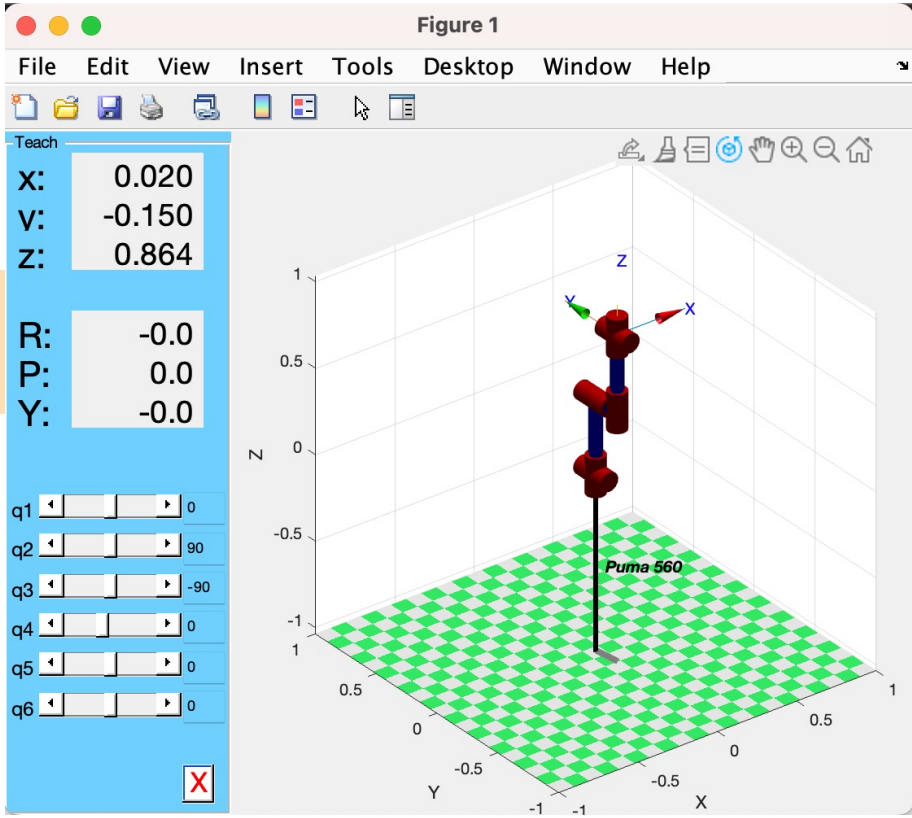
$$m = \sqrt{\det\left(\boldsymbol{J}\boldsymbol{J}^T\right)}$$

Manipulability measure - PUMA 560

- A global representative measure of manipulation ability can be obtained by considering the volume of the ellipsoid. This volume is proportional to the quantity:

$$m = \sqrt{\det\left(JJ^T\right)}$$

```
>> p560.manipulty(qr)
Manipulability: translation 0.00017794, rotation 0
```

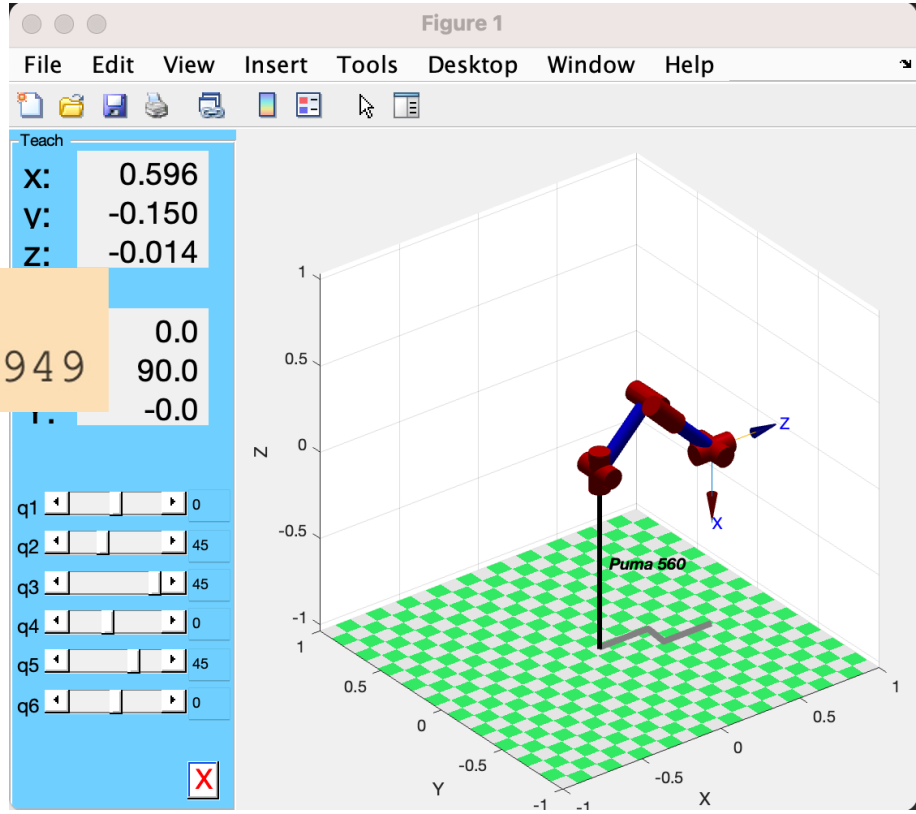


Manipulability measure - PUMA 560

- A global representative measure of manipulation ability can be obtained by considering the volume of the ellipsoid. This volume is proportional to the quantity:

$$m = \sqrt{\det\left(JJ^T\right)}$$

```
>> p560.maniplty(qn)
Manipulability: translation 0.111181, rotation 2.44949
```



… end of Lecture 10