

คำถามข้อที่ ① 1.1) การนำ FK มาใช้งาน

สมการ Forward Kinematics เขียนอยู่ในรูปของฟังก์ชันดังนี้

$R, P, R_e, p_e = \text{FKHW2}(q)$

เมื่อ

$R(:, :, i) = R_i^0$

$P(:, i) = p_{0,i}^0$

$R_e = R_e^0$

$p_e = p_{0,e}^0$

$p_{0,i}^0 \in \mathbb{R}^{3 \times 1}$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึงตำแหน่งของจุดกำเนิดของเฟรมพิกัด F_i ที่สัมพันธ์กับจุดกำเนิดของเฟรมพิกัด F_0 และอ้างอิงกับเฟรมพิกัด F_0

$R_i^0 \in \mathbb{R}^{3 \times 3}$ เป็นเมตริกซ์การหมุนของ double ที่มีขนาดเท่ากับ 3×3 ที่แสดงถึงทิศทางหมุนของเฟรมพิกัด F_i ที่อ้างอิงกับเฟรมพิกัด F_0

$q \in \mathbb{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึง configuration ของหุ่นยนต์ (Joint Configuration)

$R, P, R_e, p_e = \text{FKHW2}([0, 0, 0])$

```
# R = Rotation matrix of each joint[i] via GlobalFrame [3x3x4]
# P = Position of Frame[i] via GlobalFrame [3x4]
# R_e = Rotation matrix of end effector via GlobalFrame [3x3]
# P_e = Position of end effector via GlobalFrame [3x1]
R, P, R_e, p_e = FKHW2(q)
```

...

```
R =
[[[111. 211. 311. 411.]
  [112. 212. 312. 412.]
  [113. 213. 313. 413.]]
```

```
[[[121. 221. 321. 421.]
  [122. 222. 322. 422.]
  [123. 223. 323. 423.]]
```

```
[[[131. 231. 331. 431.]
  [132. 232. 332. 432.]
  [133. 233. 333. 433.]]
]
```

```
R0_1 =
[[[111. 112. 113.]
  [121. 122. 123.]
  [131. 132. 133.]]
...
```

```
# Pick the rotation matrix of each joint (Frame[i]) via GlobalFrame
R0_1 = R[:, :, 0] # Frame[1] via GlobalFrame
R0_2 = R[:, :, 1] # Frame[2] via GlobalFrame
R0_3 = R[:, :, 2] # Frame[3] via GlobalFrame
```

ทำการดึง Rotation Matrix ของหุ่นยนต์ออกมาใช้งาน
โดยเขียนอยู่ในรูป R_i^j

และ Velocity Jacobian ของหุ่นยนต์เขียนในรูป

Velocity of Robot's Frame

$$\begin{bmatrix} \dot{\omega}_{o,i}^0 \\ \dot{v}_{o,i}^0 \end{bmatrix} = \begin{bmatrix} J_{\omega}^i(q) \\ J_v^i(q) \end{bmatrix} \dot{q}$$

manipulator Jacobian of frame i

$$\dot{\xi}_{o,i}^0 = J^i(q) \cdot \dot{q}$$

$\xi_{o,i}^0 = \begin{bmatrix} p_j \dot{z}_j^0 \end{bmatrix} : \text{if } j \leq i$

```
# Stack the Jacobian matrix of angular velocity and linear velocity to form the Jacobian matrix of End Effector [6x3]
J_e = np.concatenate((Jw, Jv), axis=0)

return J_e
```

1.2) การคำนวณหา Jacobian ของความเร็วเชิงมุม ω

```
# Since all the joints are revolute, the Jacobian p = 1
rho = [1,1,1]

# Find the Jacobian matrix of angular velocity for End Effector
Z = np.array([[0],[0],[1]]) # Z axis of Local Frame [3x1]
# Calculate the angular velocity Jacobian of each joint
JW0_1 = rho[0] * (R0_1 @ Z)
JW0_2 = rho[1] * (R0_2 @ Z)
JW0_3 = rho[2] * (R0_3 @ Z)
# Stack the angular velocity Jacobian of each joint to form the Jacobian matrix of angular velocity for End Effector [3x3]
JW = np.concatenate((JW0_1,JW0_2,JW0_3),axis=1)
```

Velocity of Robot's Frame
Angular Velocity of frame i

$$\omega_{o,i}^i = \begin{bmatrix} \rho_i \hat{z}_1^o & \rho_i \hat{z}_2^o & \dots & \rho_i \hat{z}_i^o & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_i \\ \dot{q}_{i+1} \\ \dot{q}_{i+2} \\ \vdots \\ \dot{q}_n \end{bmatrix} = \hat{z}_i^i \dot{q}$$

J_{ω}^i : angular Jacobian Matrix of frame i

$$J_{\omega}^i = \begin{bmatrix} J_{\omega,1}^i & J_{\omega,2}^i & \dots & J_{\omega,n}^i \\ \uparrow & \uparrow & & \uparrow \end{bmatrix}$$

$$J_{\omega,j}^i = \begin{cases} \rho_j \hat{z}_j^o & \text{if } j \leq i \\ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} & \text{if } j > i \end{cases}$$

เนื่องจาก Joint ของหุ่นยนต์เราเป็น
Revolute ทั้งหมด ดังนั้น $p = 1$
จากสูตร

โดย j คือ Frame ของ Joint ที่ j
 $j \rightarrow \{1, 2, 3\}$

$$J_{\omega,j}^e = \rho_j \hat{z}_j^o$$

และเนื่องจาก $\hat{z}_j^j = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

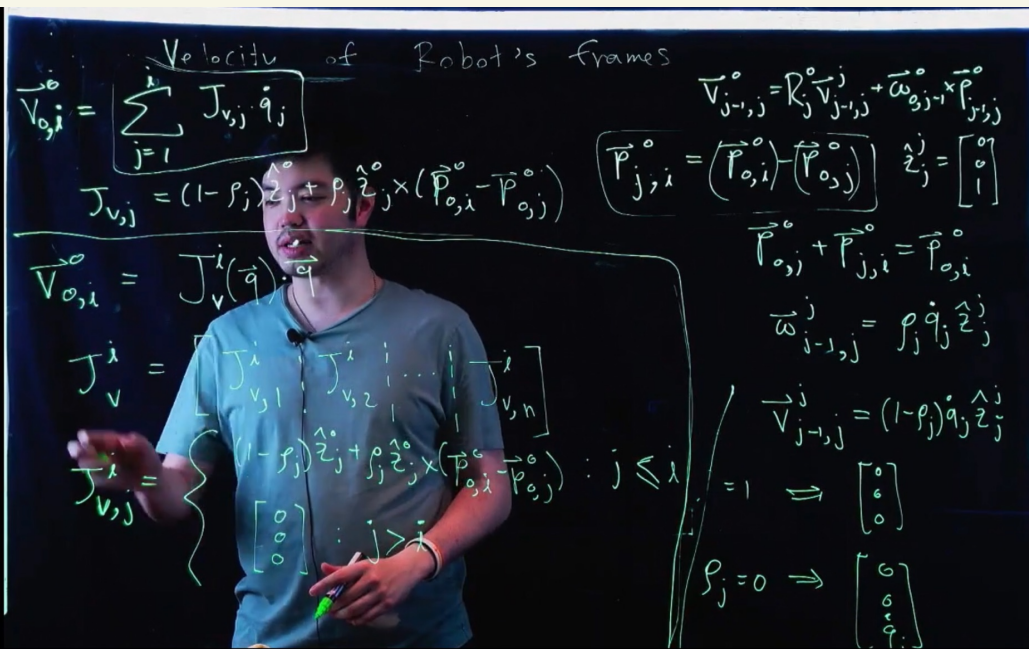
ดังนั้น เราจึงหาการแปลงให้ข้อต่อ
Frame ฐาน(0) โดยให้ Rotation Matrix

$$\hat{z}_j^o = R_j^o \hat{z}_j^j$$

จากสูตร

1.3) การคำนวณหา Jacobian ของความเร็วเชิงเส้น V

```
# Find the Jacobian matrix of linear velocity for End Effector
P0_01 = P[:,0] # Position of Frame[1] via GlobalFrame [3x1]
P0_02 = P[:,1] # Position of Frame[2] via GlobalFrame [3x1]
P0_03 = P[:,2] # Position of Frame[3] via GlobalFrame [3x1]
P0_E = p_e # Position of End Effector via GlobalFrame [3x1]
# Calculate the linear velocity Jacobian of each joint
JV0_1 = ((1-rho[0]) * (R0_1 @ Z)) + np.cross(JW0_1.reshape(1,3),(P0_E-P0_01)).reshape(3,1)
JV0_2 = ((1-rho[1]) * (R0_1 @ Z)) + np.cross(JW0_2.reshape(1,3),(P0_E-P0_02)).reshape(3,1)
JV0_3 = ((1-rho[2]) * (R0_1 @ Z)) + np.cross(JW0_3.reshape(1,3),(P0_E-P0_03)).reshape(3,1)
# Stack the linear velocity Jacobian of each joint to form the Jacobian matrix of linear velocity for End Effector [3x3]
JV = np.concatenate((JV0_1,JV0_2,JV0_3),axis=1)
```



เนื่องจาก Joint ของหุ่นยนต์เราเป็น
Revolute ทั้งหมด ดังนั้น $\rho = 1$
จากสูตร

$$\textcircled{1} \quad J_{v,j}^e = (1-\rho_j) \hat{z}_j^o + \overbrace{\rho_j \hat{z}_j^o}^{J_{\omega,j}^e} \times (\vec{p}_{0,e}^o - \vec{p}_{0,j}^o)$$

เนื่องจาก เราต้องการคือ Jacobian ของ End effector

$$j \rightarrow \{1, 2, 3\}$$

② จากนั้นหา Jacobian ของแต่ละ Joint มาประกอบ

$$J_{\omega,3}^e, J_{v,3}^e = [J_{\omega,1}^e \parallel J_{\omega,2}^e \parallel J_{\omega,3}^e], [J_{v,1}^e \parallel J_{v,2}^e \parallel J_{v,3}^e]$$

คำถาม ข้อที่ ②

2.1) Երկրորդ Jacobian Matrix

Reduced Jacobian Matrix & Singularity

Example: 2DOF PR Robot

$$J^e(\vec{q}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{6 \times 2}$$

$$\vec{\xi}_{o,e} = J^e(\vec{q}) \dot{\vec{q}}$$

$$\dot{\vec{q}} = J^e(\vec{q})^{-1} \vec{\xi}_{o,e}$$

FK
IK

$$\dot{\vec{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}$$

Question 2

```
def checkSingularityHW2(q):
    ...
    q : format list 1x3
    q unit: rad
    ...

# J_e = Jacobian matrix
# J_e_star = Reduced Jac
J_e = endEffectorJacobian
# We only need the last
J_e_star = J_e[3:6,:]
```

6. คำนวณ Task space Variable เป็น $p_{0,e}^0 = [p_x, p_y, p_z]$ คำนวณ Jacobian ของ $p_{0,e}^0$ เป็น $J_{0,e}$ (3x3)

$$J = \begin{bmatrix} \cancel{J_{wx}} \\ \cancel{J_{wy}} \\ \cancel{J_{wz}} \\ J_{vx} \\ J_{vy} \\ J_{vz} \end{bmatrix} \quad p_{o,e} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (1)$$

๑. ความเป็นมาของงานวิจัย
 ๒. วัตถุประสงค์ของการวิจัย
 ๓. ขอบเขตของการวิจัย
 ๔. วิธีการวิจัย
 ๕. ผลการวิจัย
 ๖. สรุปและข้อเสนอแนะ

2.2) การคำนวณหาว่าหุ่นยนต์อยู่ในสถานะ Singularity

จากโจทย์

คำถามที่ 2

กำหนดให้ Taskspace Variable เป็น $\mathbf{p}_{0,e}^0 = [p_x, p_y, p_z]$ ในการควบคุมแขนกล RRR พบว่ามีหลายตำแหน่งใน configuration space ที่จะทำให้เกิดสถานะ Singularity ทำให้ไม่สามารถหา inverse velocity kinematics ของสมการได้ กำหนดให้หุ่นยนต์อยู่ในสถานะ Singularity ก็ต่อเมื่อ

$$|\det(\mathbf{J}^*(\mathbf{q}))| < \varepsilon$$

โดยที่ค่า ε มีค่า 0.001 จงเขียนฟังก์ชันในการหาสถานะ Singularity

ทำ การ คำนวณ Det ของ Matrix และเปรียบเทียบกับ ε

```
# Find the determinant of Reduced Jacobian matrix of End Effector to check singularity
# If value of the determinant is undefined(subtrack by zero) the return value is 0.0 So, it always below the threshold
J_e_star_det = np.linalg.det(J_e_star)

...
    return format bool
...

# Set threshold of Singularity = 0.001
threshold = 0.001
# If the absolute value of determinant of Reduced Jacobian matrix is 0 or below threshold, then the robot is in singularity
if np.abs(J_e_star_det) < threshold:
    return True
else:
    return False
```

คำถาม ข้อที่ 3

คำถามข้อที่ 3
ถ้าหากนักศึกษาติดตั้ง Force Sensor รุ่น FT300 ที่สามารถวัดแรงและแรงควบคุมในพิกัดสามมิติ โดยติดตั้งในตำแหน่งกึ่งกลางปลายมือของหุ่นยนต์ RRR ในเฟรมพิกัด สามารถอ่านค่าแรงได้จาก Sensor ดังนี้

$$w^e = \begin{bmatrix} n^e f^e \end{bmatrix}$$

จงเขียนฟังก์ชันในการหา effort ของแต่ละข้อต่อเมื่อมี wrench มากระทำกับจุดกึ่งกลางของเฟรมพิกัด F^e

```
tau = computeEffortHW2(q,w)
```

$$\tau = computeEffort(q,w)$$

$\tau \in \mathbb{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึงค่า Effort ของแต่ละข้อต่อ
 $q \in \mathbb{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึง configuration ของหุ่นยนต์ (Joint Configuration)
 $w \in \mathbb{R}^6$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 6 ที่แสดงโมเมนต์และแรงที่อ้างอิงกับเฟรมพิกัด F_e
หมายเหตุ:

$$f^A = R_B^A \cdot f^B$$
$$n^A = R_B^A \cdot n^B$$

Static Analysis
Principle of Virtual Work

$\delta W = \vec{w} \cdot \delta \vec{x} - \vec{\tau} \cdot \delta \vec{q}$
 $\delta W = \vec{w}^T \delta \vec{x} - \vec{\tau}^T \delta \vec{q}$
 $\delta \vec{x} = J^e(\vec{q}) \delta \vec{q}$
 $\delta W = \vec{w}^T J^e(\vec{q}) \delta \vec{q} - \vec{\tau}^T \delta \vec{q}$
 $\delta W = (\vec{w}^T J^e(\vec{q}) - \vec{\tau}^T) \delta \vec{q}$
 $\delta W = 0$
 $\vec{w}^T J^e(\vec{q}) - \vec{\tau}^T = 0$
 $J^e(\vec{q})^T \vec{w} - \vec{\tau} = 0$

$\vec{w} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$
 $\vec{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$
 $W = \vec{w} \cdot \vec{x}$
 $W = \vec{\tau} \cdot \vec{q}$
 $W : \text{work} [J]$
 $W = \int \vec{F} \cdot d\vec{s}$
 $= \int \vec{\tau} \cdot d\vec{\theta}$
 $\vec{\tau} = J^T \vec{f}$
 $\vec{\tau} = J^e(\vec{q})^T \vec{w}_e$
[1x1] [1x6] [6x1]

จากสูตร

$$\vec{\tau} = J^e(\vec{q})^T \vec{w}_e$$

①

เนื่องจาก ค่าที่อ่านได้จาก Sensor จะบันทึก
ของ End effector ดังนั้น เรา จำเป็น ต้อง แปลง
wrench ของ หุ่นยนต์ ให้ เทียบ กับ เฟรม โลก (0)

จาก

$$\vec{w}_e = \begin{bmatrix} R_e^0 f^e \\ \dots \dots \dots \\ R_e^0 n^e \end{bmatrix}$$

②

จากฉบับจริง ทำการแก้สมการ

```
# Question 3
def computeEffortHW2(q,w):
    ...

    q : format list 1x3 [[i_11, i_12, i_13]]
    q unit: rad
    ...

    # Get the Rotation matrix of each Frame[i] via GlobalFrame and Position of Frame[i] via GlobalFrame and End Effector via GlobalFrame
    R,P,R_e,p_e = FKHW2(q) } ฝึกหัด Rotation Matrix  $R_e$ 
    # Get the Jacobian matrix of End Effector
    J_e = endEffectorJacobianHW2(q) } ฝึกหัด Jacobian

    # Get the Moement and Force of End Effector
    w_t = np.transpose(w) # Since w is a 1x6 vector, we need to transpose it to 6x1
    N = w_t[0:3] # Moment of End Effector [3x1]
    F = w_t[3:6] # Force of End Effector [3x1] } ฝึกหัด Moment และ Force  $\rightarrow f^e, n^e$ 

    # Find the effort of each joint
    # Since Sensor is attached to End Effector, the given Moment and Force of End Effector need to be transformed to GlobalFrame
    # Get Rotation matrix that transforms the values of w from End Effector Frame to Global Frame
    R0_e = R_e # R0_4 = R_e
    W0_e = np.concatenate((R0_e @ N, R0_e @ F),axis=0) # Stack Moment and Force of End Effector to form the vector of w in GlobalFrame [6x1]
    # Calculate the effort of each joint
    J_e_transpose = np.transpose(J_e) # Since J_e is a 6x3 matrix, we need to transpose it to 3x6 so that we can multiply it with W0_e [6x1]
    effort = J_e_transpose @ W0_e # [3x6] @ [6x1] = [3x1]
    ...

    return format list 1x3
    [ [i_11, i_12, i_13] ]
    ...

    return np.transpose(effort) # Since effort is a 3x1 vector, we need to transpose it to 1x3
```