คำถามข้อที่ 1    1.1) การนำ FK มาใช้งาน

สมการ Forward Kinematics เขียนอยู่ในรูปของฟังก์ชันดังนี้

```
R,P,R_e,p_e = FKHW2(q)
```

เมื่อ

$\mathbf{R}(:,:,i) = \mathbf{R}_i^0$
$\mathbf{P}(:,i) = \mathbf{p}_0^{0,i}$
$\mathbf{R}_e = \mathbf{R}_e^0$
$\mathbf{p}_e = \mathbf{p}_{0,e}^0$
$\mathbf{p}_{0,i}^0 \in \mathbf{R}^{3\times1}$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึงตำแหน่งของจุดกำเนิดของเฟรมพิกัด $\mathrm{F}_i$ ที่สัมพัทธ์กับจุดกำเนิดของเฟรมพิกัด $\mathrm{F}_0$ และอ้างอิงกับเฟรมพิกัด $F_0$
$\mathbf{R}_i^0 \in \mathbf{R}^{3\times3}$ เป็นเมตริกซ์การหมุนของ double ที่มีขนาดเท่ากับ 3x3 ที่แสดงถึงทิศทางหมุนของเฟรมพิกัด $\mathrm{F}_i$ ที่อ้างอิงกับเฟรมพิกัด $\mathrm{F}_0$
$\mathrm{q} \in \mathrm{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึง configuration ของหุ่นยนต์ (Joint Configuration)

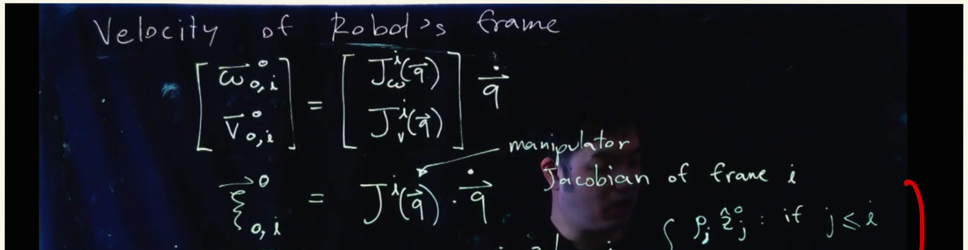
```
R,P,R_e,p_e = FKHW2([0,0,0])
```

```
# 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 $R_j^i$

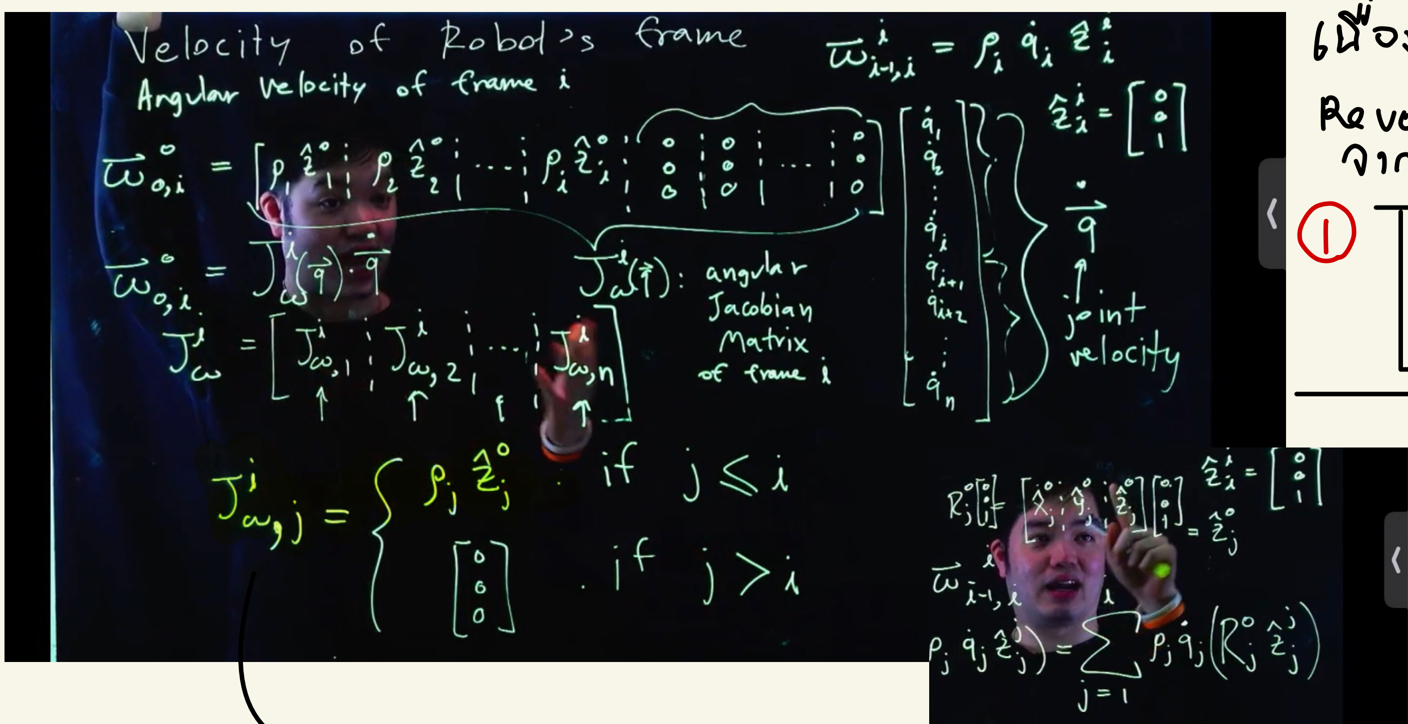
และ Velocity Jacobian ของหุ่นยนต์เขียนในรูป




```
# 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 ของ ความเร็วเชิงมุม $\omega$

```python
# 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)
```




จาก สูตร

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

จากสูตร

①
$$\boxed{J^e_{\omega,j} = \rho_j \hat{Z}^0_j}$$

โดย j คือ Frame ของ Joint ที่ j

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

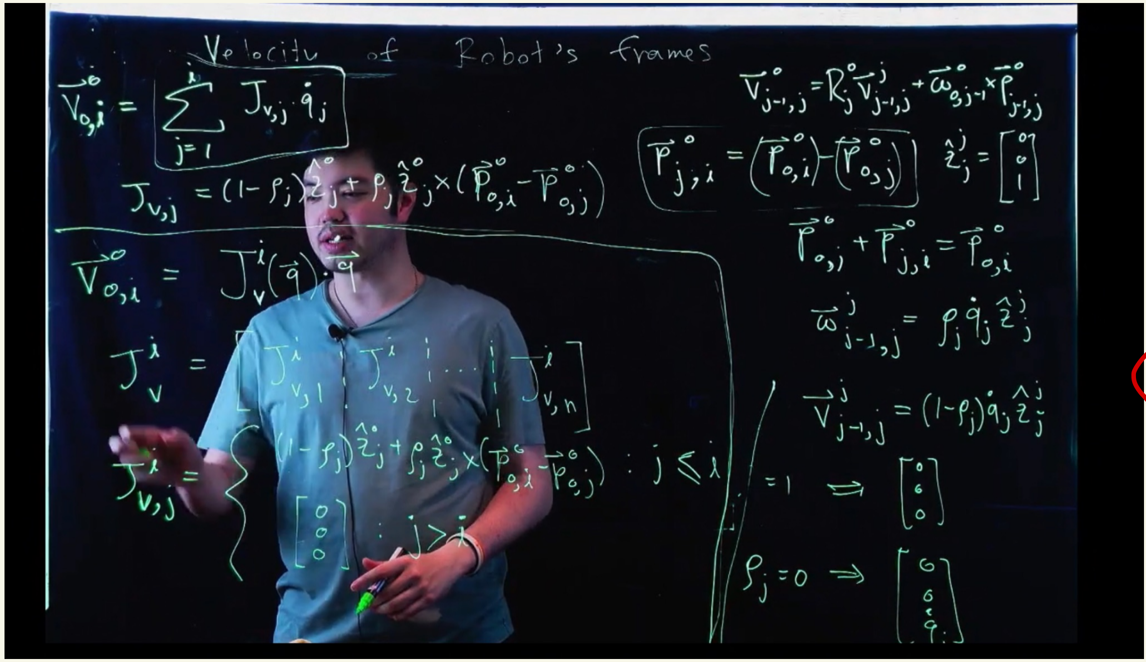
และเนื่องจาก $\hat{Z}^j_j = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

ดังนั้น เราต้องทำการแปลงให้อยู่ใน Frame โลก (0) โดยใช้ Rotation Matrix

②
$$\boxed{\hat{Z}^0_j = R^0_j \hat{Z}^j_j}$$

## 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

1. $$\boxed{J^e_{v,j} = (1-\rho_j)\hat{Z}^0_j + \overbrace{\rho_j \hat{Z}^0_j}^{J^e_{w,j}} \times (P^0_{0,e} - P^0_{0,j})}$$

เนื่องจาก เราต้องการ คิด Jacobain กับ End effector

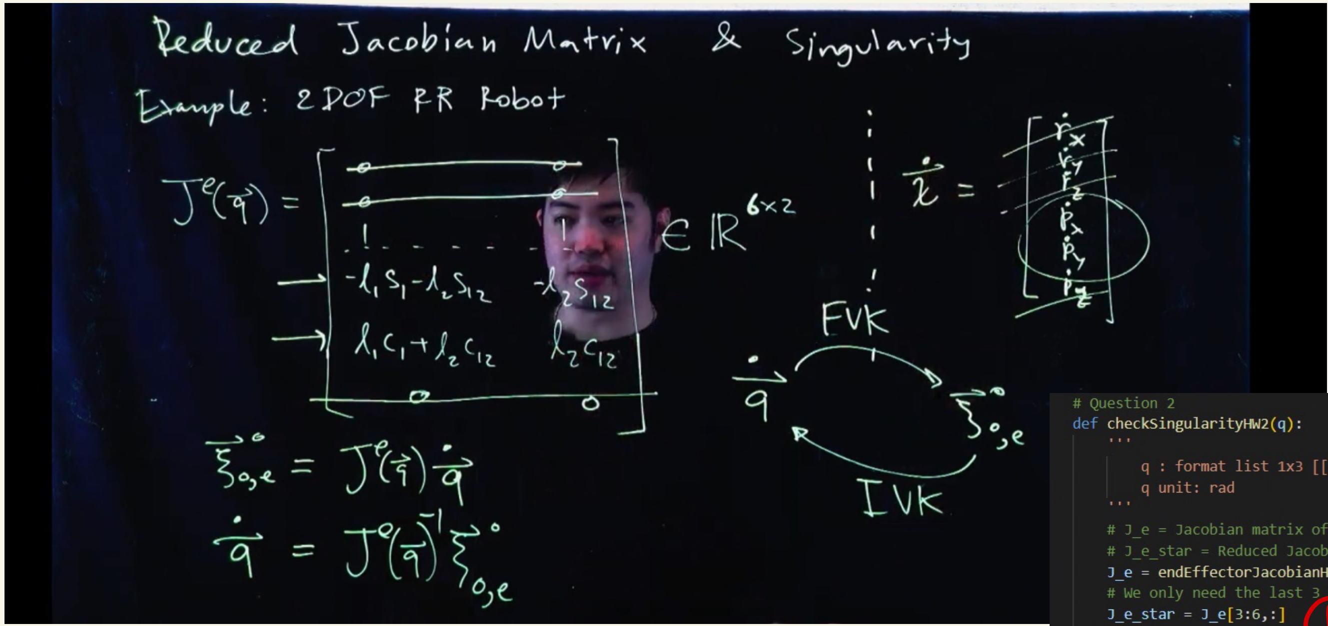
$$j \longrightarrow \{1, 2, 3\}$$

2. จากนั้นหา Jacobain ของแต่ละ Joint มาต่อกัน

$$J^e_{w,3},\ J^e_{v,3} = [\, J^e_{w,1} \vdots J^e_{w,2} \vdots J^e_{w,3} \,],\ [\, J^e_{v,1} \vdots J^e_{v,2} \vdots J^e_{v,3} \,]$$

คำถามข้อที่ 2

## 2.1) การลดรูป Jacobian Matrix




```python
# Question 2
def checkSingularityHW2(q):
    '''
        q : format list 1x3 [[i_11, i_12, i_13]]
        q unit: rad
    '''
    # J_e = Jacobian matrix of End Effector [6x3]
    # J_e_star = Reduced Jacobian matrix of End Effector [3x3]
    J_e = endEffectorJacobianHW2(q)
    # We only need the last 3 rows of the Jacobian matrix of End Effector it represents the Task Space linear velocity in the Gl
    J_e_star = J_e[3:6,:]
```
1

เนื่องจาก Task space Varible เป็น $p^0_{0,e} = [p_x, p_y, p_z]$ ดังนั้นเราจึงต้องลดรูป Jacobain ให้เหลือเท่ากับ Task space varible [3x3] โดยเราจะเลือกเก็บ Jacobian ของความเร็วเชิงเส้นไว้ เพื่อให้เราสามารถหา $p^0_{0,e}$ ได้

$$J = \begin{bmatrix} \cancel{J_{\omega x}} \\ \cancel{J_{\omega y}} \\ \cancel{J_{\omega z}} \\ J_{vx} \\ J_{vy} \\ J_{vz} \end{bmatrix} \qquad p^0_{0,e} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad 1$$

## 2.2) การคำนวณหาว่าหุ่นยนต์อยู่ในสภาวะ Singularity

จากโจทย์

***คำถามที่ 2***

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

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

โดยที่ค่า ε มีค่า 0.001 จงเขียนฟังก์ชันในการหาสภาวะ Singularity

ทำการคำนวณ Det ของ Jacobain Matrix ที่ถูกลดรูป และเปรียบเทียบกับ $\varepsilon$

```python
# 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 ที่สามารถวัดแรงและแรงควบคู่ในพิกัดสามมิติ โดยติดตั้งในตำแหน่งกึ่งกลางปลายมือของหุ่นยนต์ RRR ในเฟรมพิกัด สามารถอ่านค่าแรงได้จาก Sensor ดังนี้

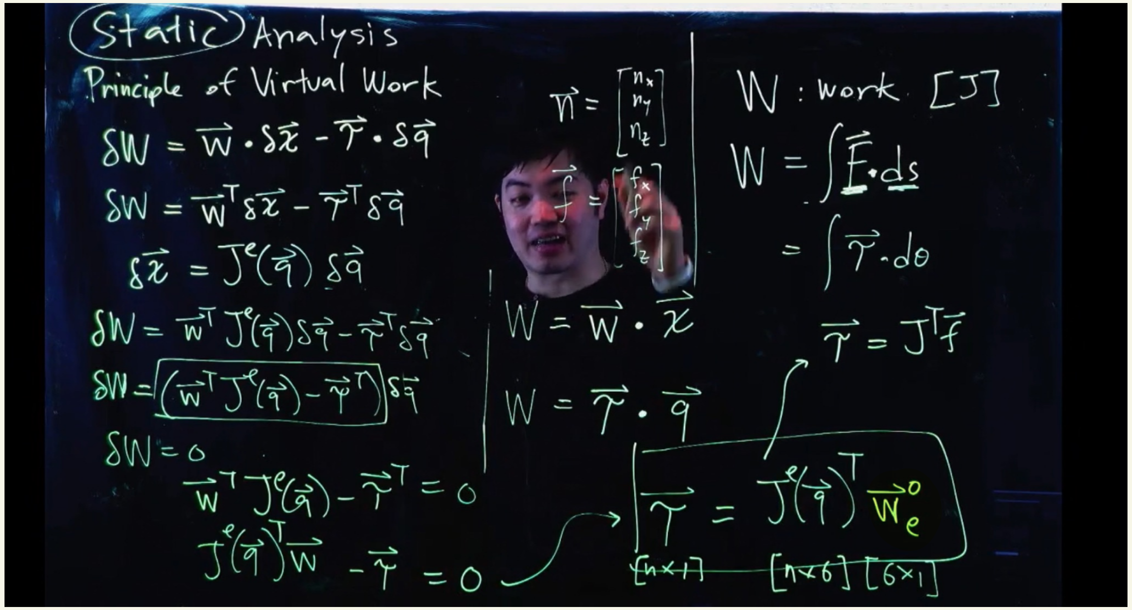
$$w^e = [[n^e f^e]]$$

จงเขียนฟังก์ชันในการหาeffortของแต่ละข้อต่อเมื่อมี wrench มากระทำกับจุดกึ่งกลางของเฟรมพิกัด $F^e$

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

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

> $\mathbf{tau} \in \mathbf{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึงค่า Effort ของแต่ละข้อต่อ
> $\mathbf{q} \in \mathbf{R}^3$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 3 ที่แสดงถึง configuration ของหุ่นยนต์ (Joint Configuration)
> $\mathbf{w} \in \mathbf{R}^6$ เป็นเวกเตอร์หลักของ double ที่มีขนาดเท่ากับ 6 ที่แสดงโมเมนท์และแรงที่อ้างอิงกับเฟรมพิกัด $\mathbf{F_e}$
> ***หมายเหตุ:***
>
> $$\mathbf{f^A} = \mathbf{R_B^A} \cdot \mathbf{f^B}$$
>
> $$\mathbf{n^A} = \mathbf{R_B^A} \cdot \mathbf{n^B}$$




โดยที่

$\vec{w}_e^o$ = wrench ของ End effector เทียบเฟรมโลก(0)

จากสูตร การคำนวณ หา effort

$$\boxed{\vec{\tau} = J^e(\vec{q})^T \vec{w}_e^o} \quad (1)$$

เนื่องจาก ค่าที่อ่านได้จาก Sensor จะเป็นหลัก ของ End effector ดังนั้น เรา จำเป็นต้องแปลง wrench ของ หุ่นยนต์ ให้เทียบ กับ เฟรม โลก (0) จาก

$$\vec{w}_e^o = \begin{bmatrix} R_e^o f^e \\ \hline R_e^o n^e \end{bmatrix} \quad (2)$$

จากนั้นจึงทำการคำนวณตามสูตร

```python
# 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)
    # Get the Jacobian matrix of End Effector
    J_e = endEffectorJacobianHW2(q)

    # 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]

    # 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
```

} คำนวณหา Rotation Matrix $R^0_e$

} คำนวณหา Jacobian

} แยก Moment และ Force → $f^e, n^e$

2

1