MAE263B: Dynamics of Robotic Systems
Discussion Section – Week6
: Jacobian (SCARA)

Seungmin Jung

02.14.2020.

Homework

❑ Homework - Review

- On CCLE, you can check your score and download an annotated pdf version for your homework.
- All reasons for deduction are described in the pdf file.

❑ Solution will be uploaded on CCLE too

- The solution for homework2 on the bionics lab – course website is wrong.
- The solution for homework3 is not uploaded on the course website.

→ From this week, solution files will be uploaded on CCLE too.

Contents

- Jacobian with SCARA example
  - Force/Torque propagation
  - Review
    - Velocity propagation
    - Direct differentiation

Kinematics Relations - Joint & Cartesian Spaces

- The location of the robot end-effector may be specified using one of the following descriptions:

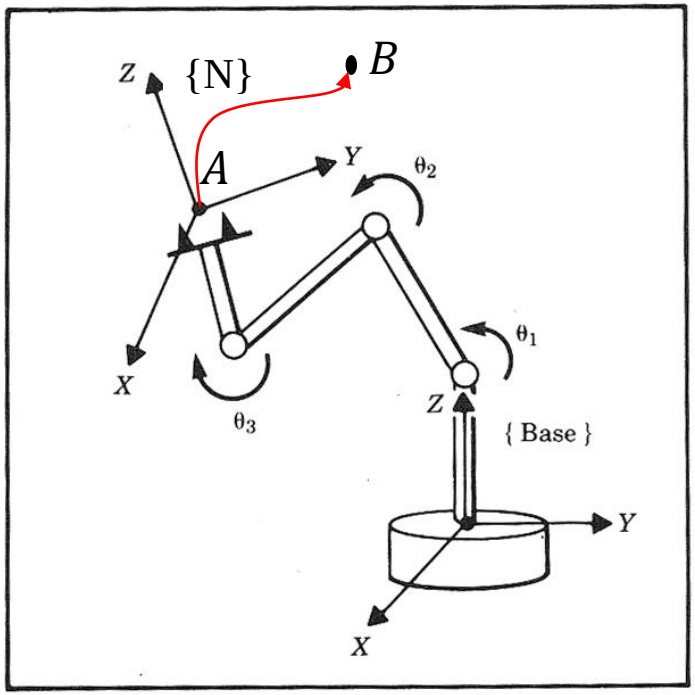
- Cartesian Space / Operational Space

$$X = \begin{bmatrix} {}^0P_N \\ {}^0r_N \end{bmatrix}$$

Euler Angles

- Joint Space

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}$$



Kinematics Relations - Forward & Inverse

- The robot kinematic equations relate the two description of the robot tip location

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}$$

$$\theta = IK(X)$$

$$X = FK(\theta)$$

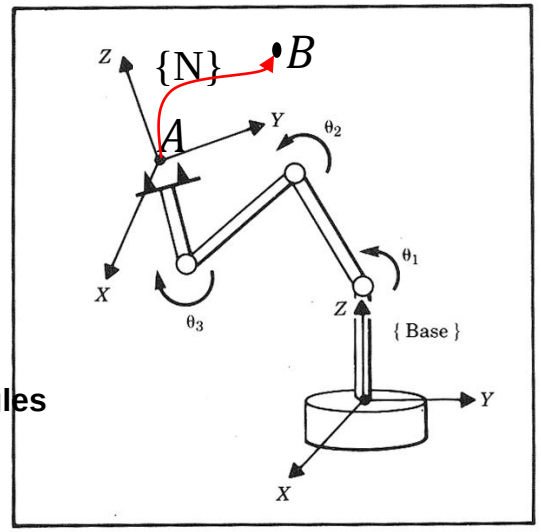
$$X = \begin{bmatrix} {}^0P_N \\ {}^0r_N \end{bmatrix}$$

Euler Angles

Tip Location in Joint Space

$${}^0_N T = \begin{bmatrix} {}^0_N R & {}^0P_N \\ 0 & 1 \end{bmatrix}$$

Tip Location in Cartesian Space



UCLA

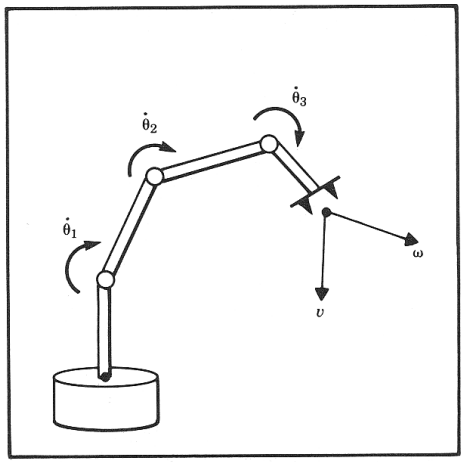
Velocity relationship
: Jacobian Matrix – Joint velocity / End-effector velocity

- **The velocity relationship**: The relationship between the joint angle rates ($\dot{X}$) and the linear and angular velocities of the end effector ($\dot{\theta}$).

$$\dot{X} = J\dot{\theta}$$

$$\dot{\theta} = \frac{d}{dt}[\theta] = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_N \end{bmatrix}$$

$$\dot{X} = \frac{d}{dt}[X] = \begin{bmatrix} \begin{bmatrix} v_N \end{bmatrix} \\ \begin{bmatrix} \omega_N \end{bmatrix} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$



Tip Velocity in Joint Space

$$\dot{\theta} = J^{-1}\dot{X}$$

Tip velocity in Cartesian Space

Jacobian Matrix - Introduction

- This expression can be expanded to: $\dot{\underline{x}} = J(\underline{\theta})\dot{\underline{\theta}}$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} & & \\ & J(\underline{\theta}) & \\ & & \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \cdots \\ \dot{\theta}_N \end{bmatrix}$$

Euler Angles

6x1 (N) 6xN Nx1 (N)

$$\dot{p}_e = J_P(q)\dot{q}$$

$$\omega_e = J_O(q)\dot{q}.$$

- Where:
  - $\dot{\underline{x}}$ is a 6x1 vector of the end effector linear and angular velocities
  - $J(\underline{\theta})$ is a 6xN Jacobian matrix
  - $\dot{\underline{\theta}}_N$ is a Nx1 vector of the manipulator joint velocities
  - N is the number of joints

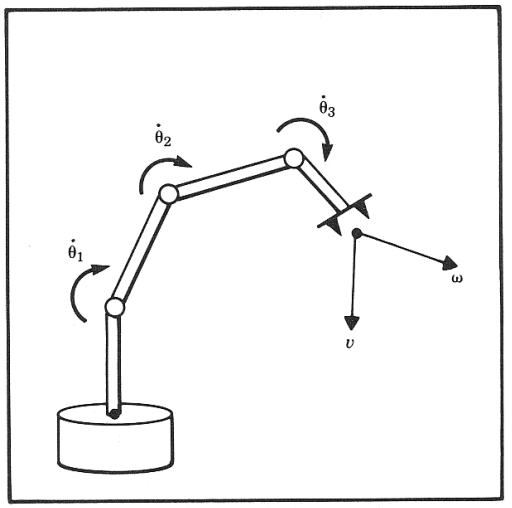
UCLA

Jacobian Matrix - Introduction

- **The velocity relationship**

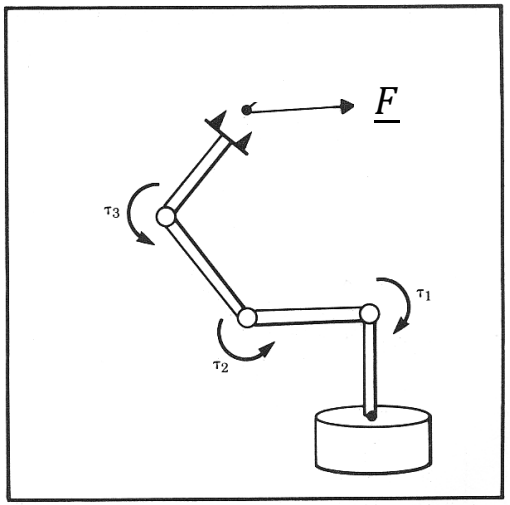
  : The relationship between
  the joint angle rates ($\underline{\dot{\theta}}_N$)
  and the translation and rotation velocities of the end effector ($\underline{\dot{x}}$).

$$\underline{\dot{x}} = J(\underline{\theta})\underline{\dot{\theta}}$$



- The relationship between
  the robot joint torques ($\underline{\tau}$)
  and the forces and moments ($\underline{F}$)
  at the robot end effector (**Static Conditions**).

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$



UCLA

Jacobian Matrix - Introduction

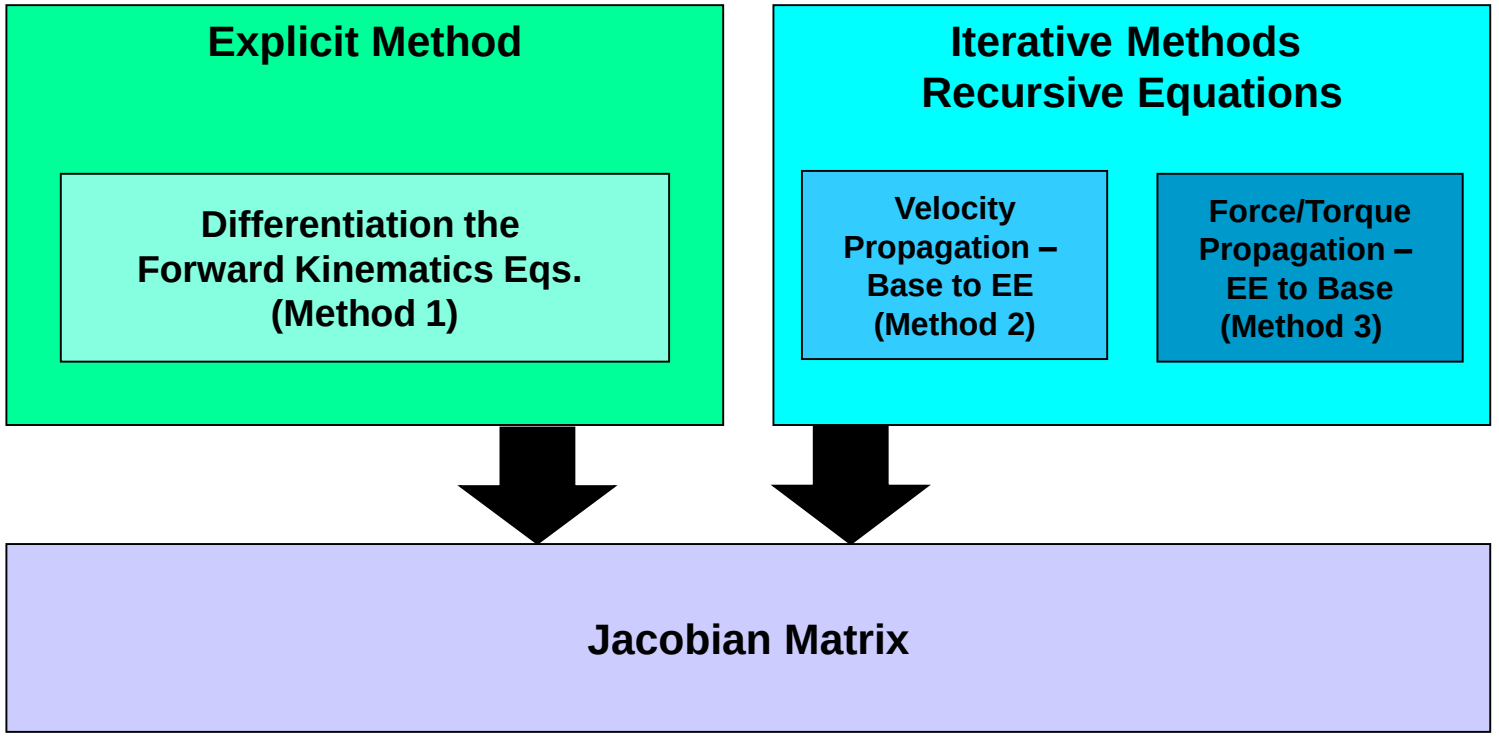
- This expression ($\underline{\tau} = J(\underline{\theta})^T \underline{F}$) can be expanded to:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \cdots \\ \tau_N \end{bmatrix} = \begin{bmatrix} & & \\ & J(\underline{\theta}) & \\ & & \end{bmatrix} \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}^T$$

Nx1 **Nx6** **6x1**

- Where:
  - $\underline{\tau}$ is a Nx1 vector of the robot joint torques
  - $J(\underline{\theta})^T$ is a Nx6 Transposed Jacobian matrix
  - $\underline{F}$ is a 6x1 vector of the forces and moments at the robot end effector
  - N is the number of joints

Jacobian Matrix - Derivation Methods

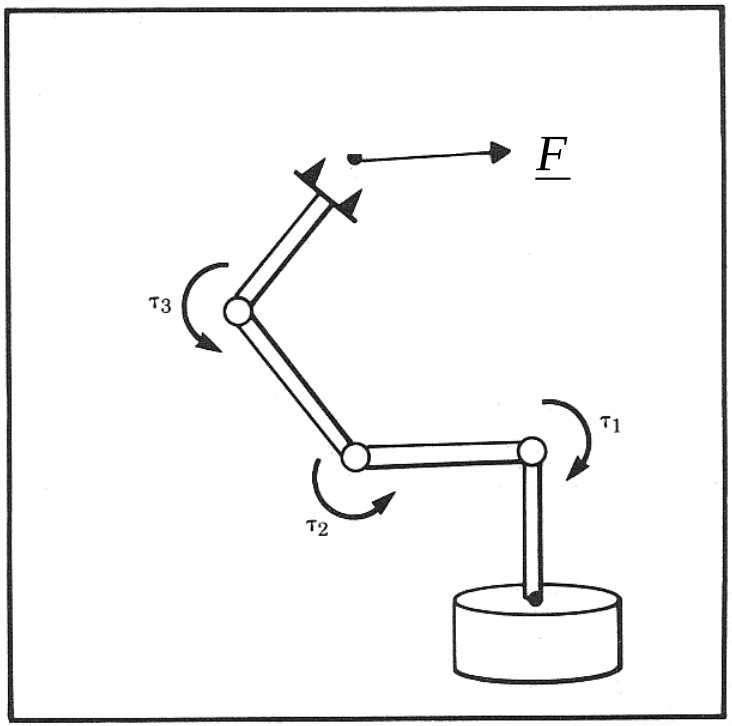


Jacobian – static force & torque

- In addition to the velocity relationship, we are also interested in developing a relationship between the robot joint torques ($\underline{\tau}$) and the forces and moments ($\underline{F}$) at the robot end effector (**Static Conditions**). This relationship is given by:

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \cdots \\ \tau_N \end{bmatrix} = J(\underline{\theta})^T \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$



UCLA

Jacobian – static force & torque

$$\underline{\tau} = J\left(\underline{\theta}\right)^T \underline{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \cdots \\ \tau_N \end{bmatrix} = \begin{bmatrix} \\ \\ J\left(\underline{\theta}\right) \\ \\ \\ \end{bmatrix}^T \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

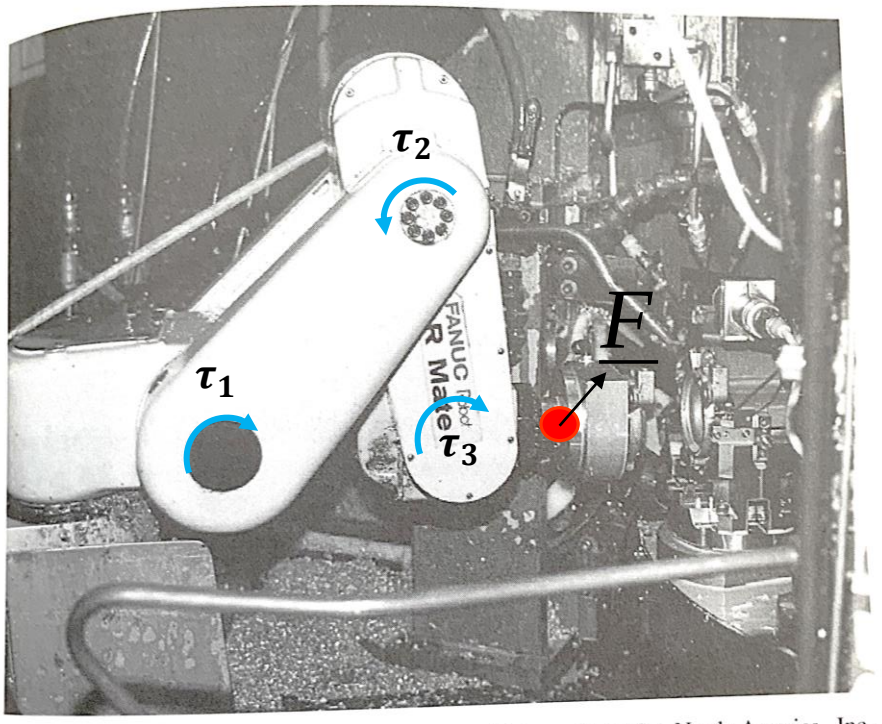


FIGURE 6.1. Fanuc LR Mate robot. (Courtesy of Fanuc Robotics North America, Inc., Rochester Hills, Michigan.)

- The force / moment equilibrium equations $\sum \mathrm{F} = 0 \quad \sum \mathrm{M} = 0$

UCLA

Jacobian – static force & torque

- Typically, the robot is **pushing on something** in the environment with the end-effector or is perhaps **supporting a load** at the hand.
- We wish to solve for the joint torques that must be acting to keep the system in static equilibrium.
- The force / moment equilibrium equations $\sum F = 0$ $\sum M = 0$

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \dots \\ \tau_N \end{bmatrix} = \begin{bmatrix} & & \\ & J(\underline{\theta}) & \\ & & \end{bmatrix}^T \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

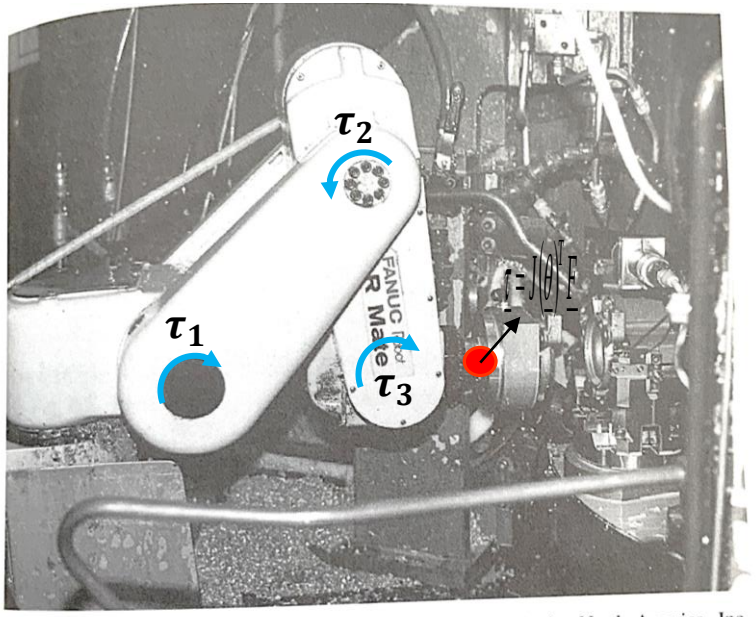


FIGURE 6.1. Fanuc LR Mate robot. (Courtesy of Fanuc Robotics North America, Inc., Rochester Hills, Michigan.)

Jacobian – static force & torque

- Static force analysis is of practical importance in determining the quality of force transmission through the various joints of a mechanism.
- It serves as a basis for sizing the links and bearings of a robot manipulator and for selecting appropriate actuators.
- The result also be used for force control of robot manipulator.

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \cdots \\ \tau_N \end{bmatrix} = \begin{bmatrix} & & \\ & J(\underline{\theta}) & \\ & & \end{bmatrix}^T \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

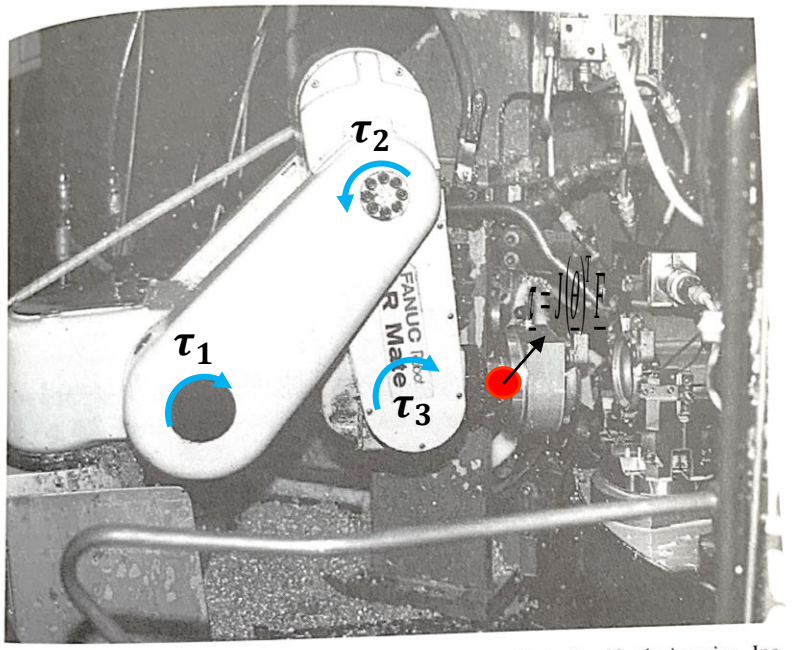


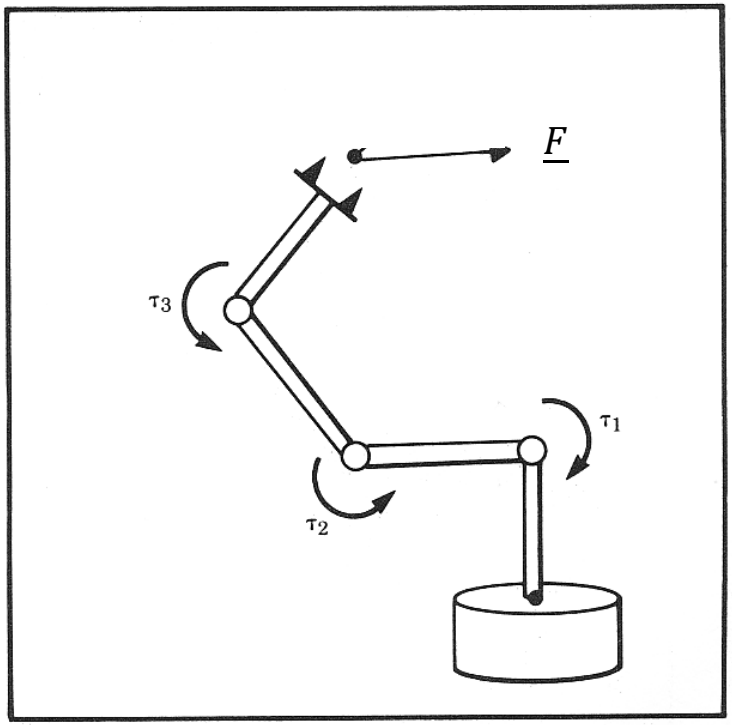
FIGURE 6.1. Fanuc LR Mate robot. (Courtesy of Fanuc Robotics North America, Inc., Rochester Hills, Michigan.)

Statics - Forces & Torques

Problem

Given: Typically the robot's end effector is applying forces and torques on an object in the environment or carrying an object (gravitational load).

Compute: The joint torques must be acting to keep the system in static equilibrium.



$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

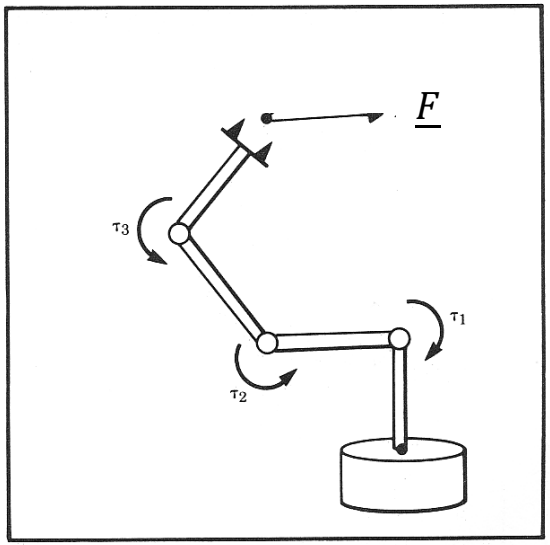
Statics - Forces & Torques

Solution

Jacobian - Mapping from the joint force/torques - $\underline{\tau}$ to forces/torque in the Cartesian space applied on the end effector) - $\underline{F}$.

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$

Free Body Diagram - The chain like nature of a manipulator leads to decompose the chain into individual links and calculate how forces and moments propagate from one link to the next.



$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix}$$

Static Analysis Protocol - Free Body Diagram 1/

Step 1

Lock all the joints - Converting the manipulator (mechanism) to a structure

Step 2

Consider each link in the structure as a free body and write the force / moment equilibrium equations

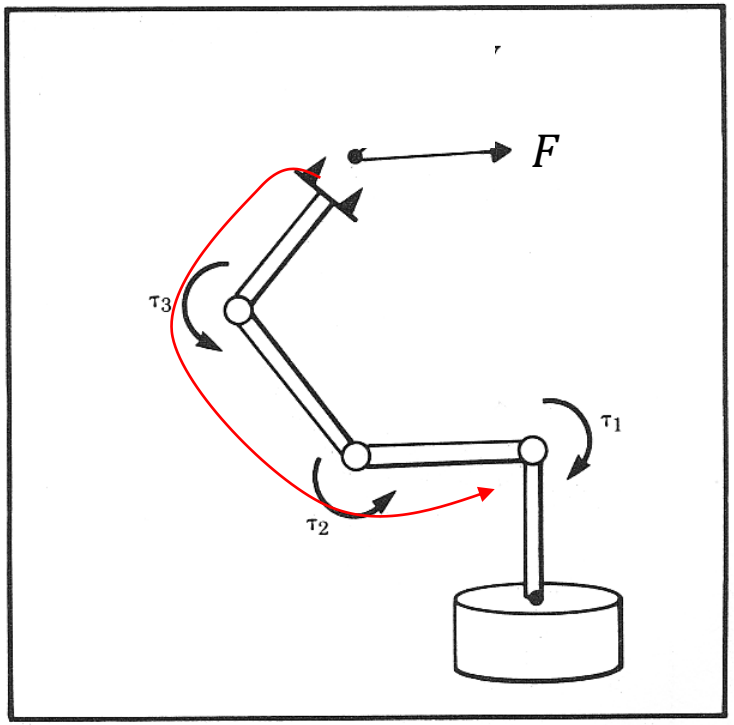
(3 Eqs.) $\sum \mathrm{F} = 0$

(3 Eqs.) $\sum \mathrm{M} = 0$

Step 3

Solve the equations - 6 Eq. for each link.

Apply backward solution starting from the last link (end effector) and end up at the first link (base)



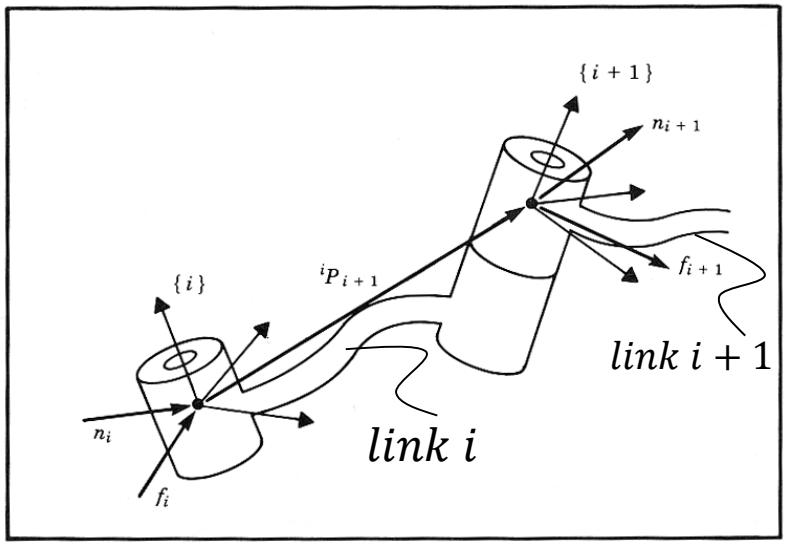
UCLA

Static Analysis Protocol - Free Body Diagram 2/

- Special Symbols are defined for the force and torque exerted by the neighbor link

f_i - Force exerted on link ***i*** by link ***i-1***

n_i - Torque exerted on link ***i*** by link ***i-1***



Reference coordinate system {B}

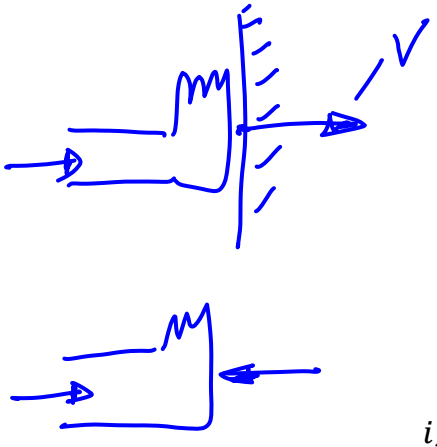
$^B f_A$ Force ***f*** or torque ***n***

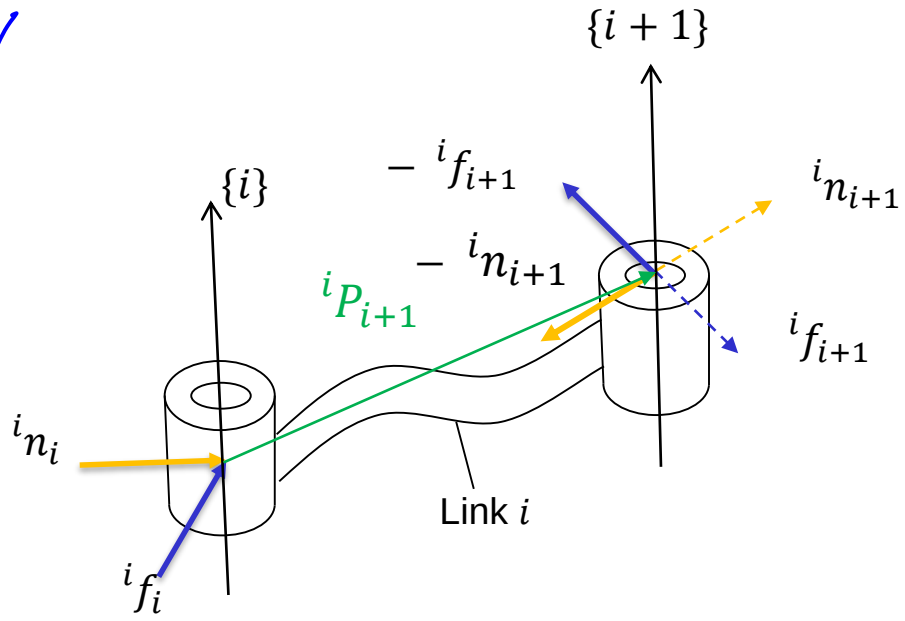
$^B n_A$ Exerted on link A by link A-1

- For easy solution superscript index (B) should the same as the subscript (A)

Static Analysis Protocol - Free Body Diagram 3/

- For serial manipulator in static equilibrium (joints locked), the sum the forces and torques acting on Link i in the link frame $\{i\}$ are equal to zero.





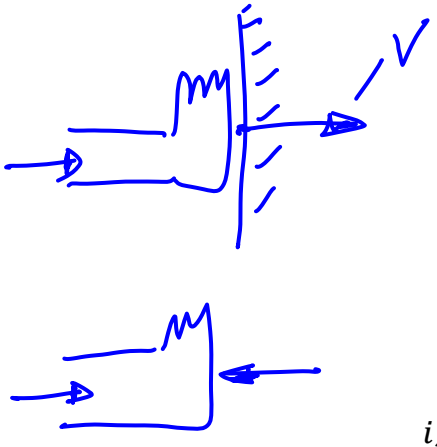
$$\sum F = 0 \quad \Rightarrow \quad \sum F = {}^{i}f_i - {}^{i}f_{i+1} = 0$$

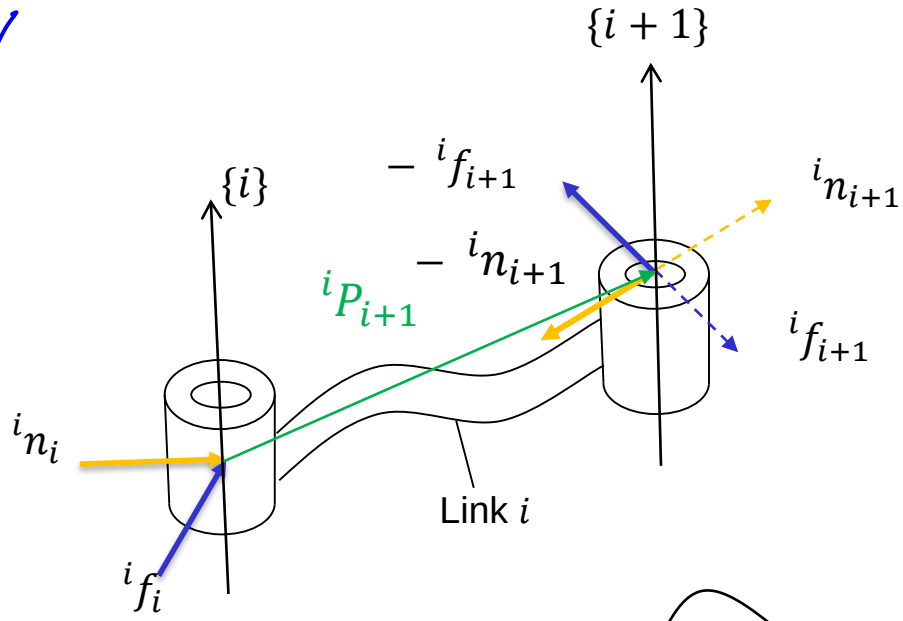
$$\sum M = 0 \quad \Rightarrow \quad \sum M = {}^{i}n_i - {}^{i}n_{i+1} - {}^{i}P_{i+1} \times {}^{i}f_{i+1} = 0$$

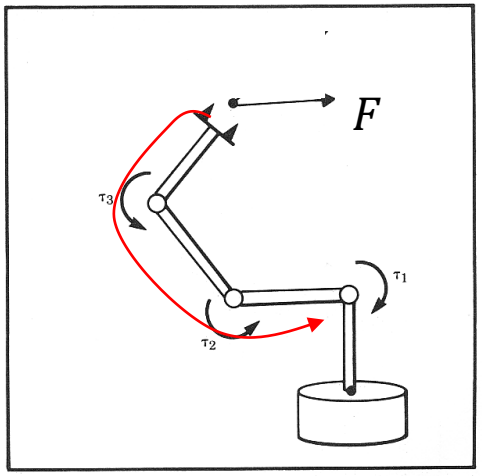
UCLA

Static Analysis Protocol - Free Body Diagram 4/

- For serial manipulator in static equilibrium (joints locked), the sum the forces and torques acting on Link i in the link frame $\{i\}$ are equal to zero.







$$\sum F = 0 \quad \Rightarrow \quad \sum F = {}^{i}f_i - {}^{i}f_{i+1} = 0 \qquad \rightarrow \quad {}^{i}f_i = {}^{i}f_{i+1}$$

$$\sum M = 0 \quad \Rightarrow \quad \sum M = {}^{i}n_i - {}^{i}n_{i+1} - {}^{i}P_{i+1} \times {}^{i}f_{i+1} = 0 \qquad \rightarrow \quad {}^{i}n_i = {}^{i}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_{i+1}$$

UCLA

Static Analysis Protocol - Free Body Diagram 5/

- Changing the reference frame such that each force (and torque) is expressed upon their link's frame, we find the static force (and torque) propagation from link ***i+1*** to link ***i***

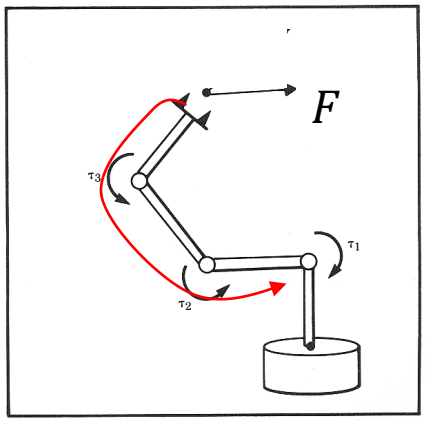
$$ {}^{i}f_i = {}^{i}f_{i+1} = {}^{i}_{i+1}R\ {}^{i+1}f_{i+1} $$

$$ {}^{i}n_i = {}^{i}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_{i+1} = {}^{i}_{i+1}R\ {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i $$

$$ {}^{i}f_i = {}^{i}_{i+1}R\ {}^{i+1}f_{i+1} $$

$$ {}^{i}n_i = {}^{i}_{i+1}R\ {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i $$

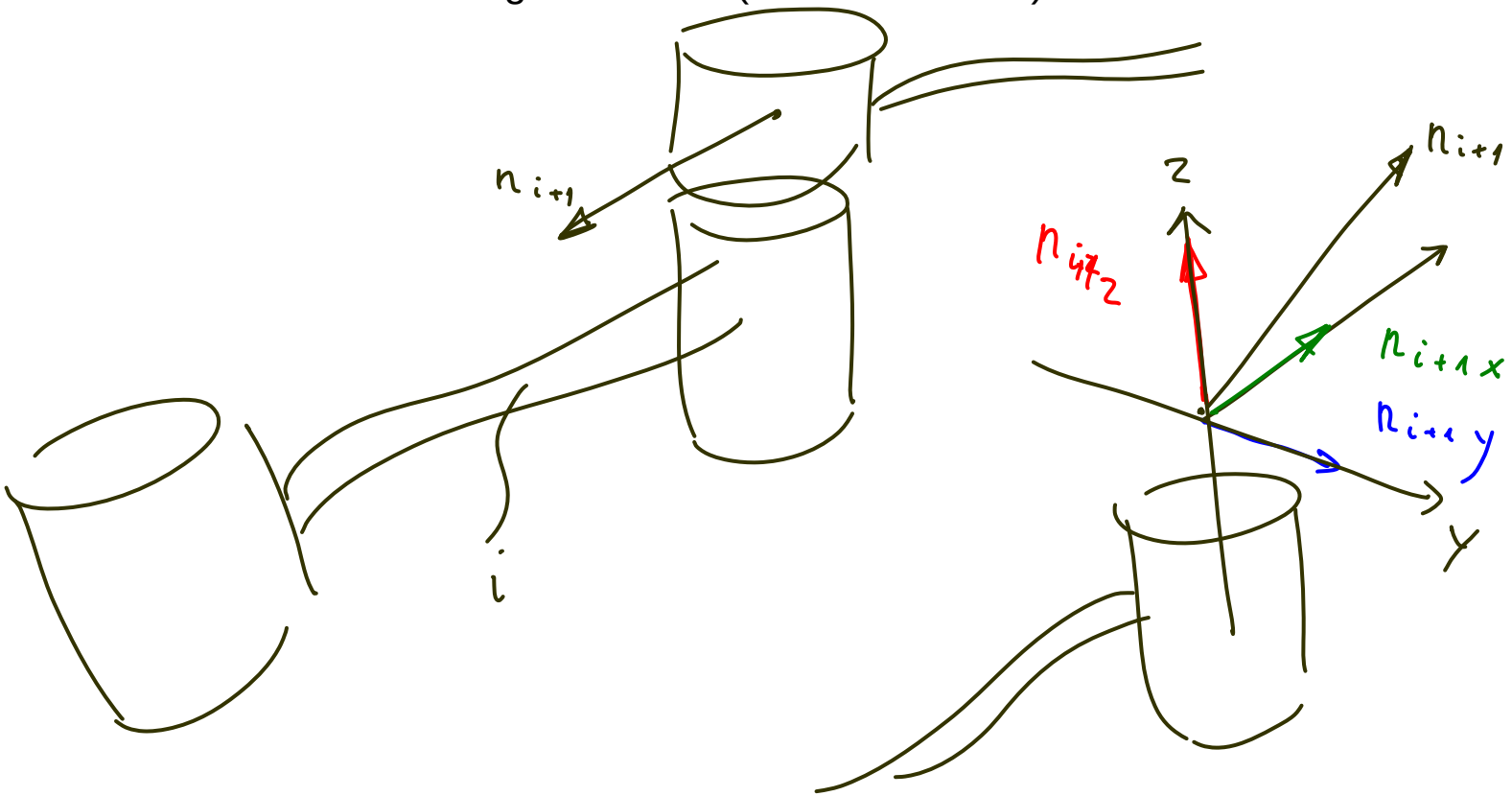
- These equations provide the static force (and torque) propagation from link to link.
- They allow us to start with the force and torque applied at the end effector and calculate the force and torque at each joint all the way back to the robot base frame.



UCLA

Static Analysis Protocol - Free Body Diagram 6/

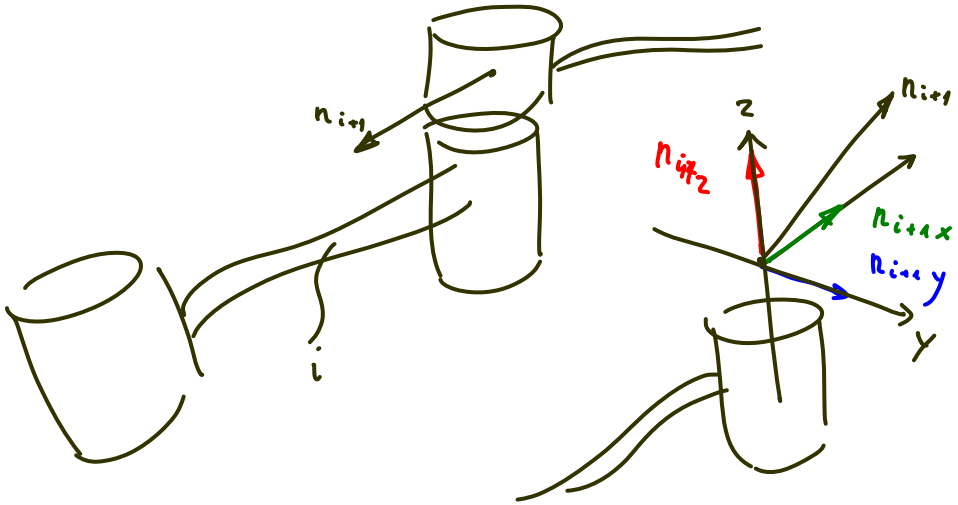
- **Question:** What torques are needed at the joints in order to balance the reaction moments acting on the link (**Revolute Joint**).



Static Analysis Protocol - Free Body Diagram 7/

- **Question:** What torques are needed at the joints in order to balance the reaction moments acting on the link (**Revolute Joint**).

- **Answer:** All the components of the moment vectors are resisted by the structure of mechanism itself, except for the torque about the joint axis (**Revolute Joint**).

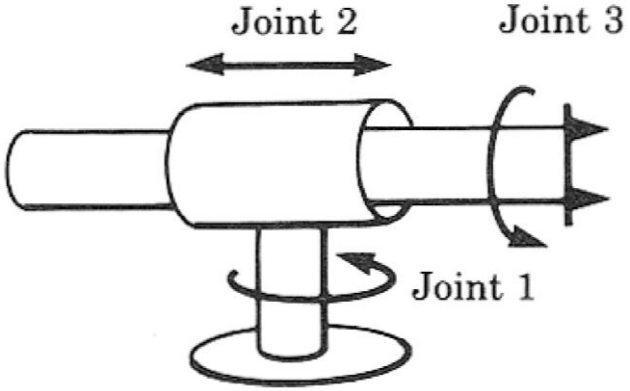


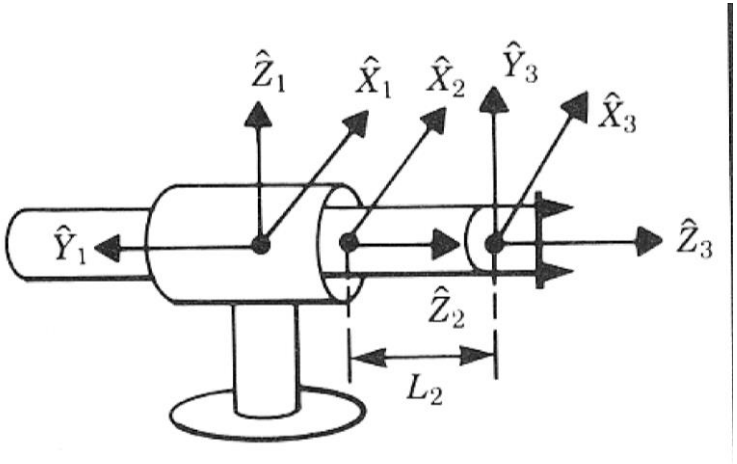
$$\tau_i = {}^i n_i^T \hat{Z}_i$$

$$= \begin{bmatrix} {}^i n_{ix} & {}^i n_{iy} & {}^i n_{iz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Static Analysis Protocol - Free Body Diagram 8/

- **Question:** What forces are needed at the joints in order to balance the reaction forces acting on the link (**Prismatic Joint**).

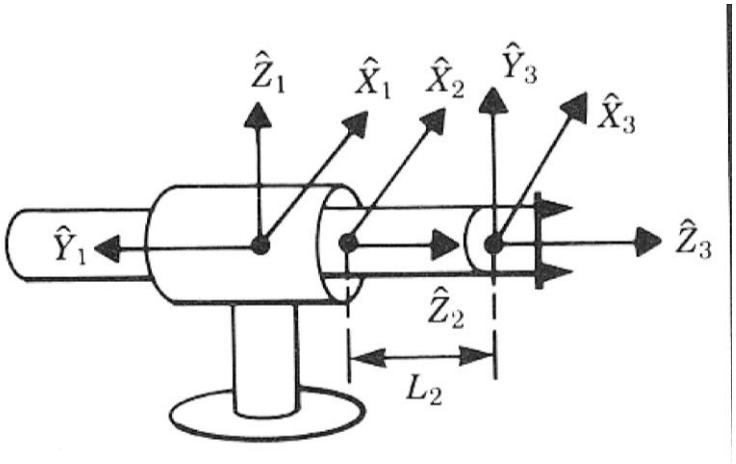




Static Analysis Protocol - Free Body Diagram 9/

- **Question:** What forces are needed at the joints in order to balance the reaction forces acting on the link (**Prismatic Joint**).

- **Answer:** All the components of the force vectors are resisted by the structure of mechanism itself, except for the force along the joint (**Prismatic joint**).



$$f_i = {}^i f_i^T \hat{Z}_i$$

$$= \begin{bmatrix} {}^i f_{ix} & {}^i f_{iy} & {}^i f_{iz} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

UCLA

Static Analysis Protocol - Free Body Diagram 10/

- **Answer:** All the components of the force and moment vectors are resisted by the structure of mechanism itself, except for the torque about the joint axis (revolute joint) or the force along the joint (prismatic joint).

- Therefore, to find the joint the torque or force required to maintain the static equilibrium, the dot product of the joint axis vector with the moment vector or force vector acting on the link is computed

Revolute Joint

$$\tau_i = {}^i n_i^T \hat{z}_i = [{}^i n_{ix} \quad {}^i n_{iy} \quad {}^i n_{iz}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Prismatic Joint

$$f_i = {}^i f_i^T \hat{z}_i = [{}^i f_{ix} \quad {}^i f_{iy} \quad {}^i f_{iz}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Static Analysis

Solution

- Apply the static equilibrium equations starting from the end effector and going toward the base

Static force "propagation" from link to link

$$^{i}f_{i} = {}^{i}_{i+1}R \; {}^{i+1}f_{i+1}$$

$$^{i}n_{i} = {}^{i}_{i+1}R \; {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_{i}$$

- To find **the joint torque or force required to maintain the static equilibrium**, the dot product of the joint axis vector with the moment vector or force vector acting on the link is computed

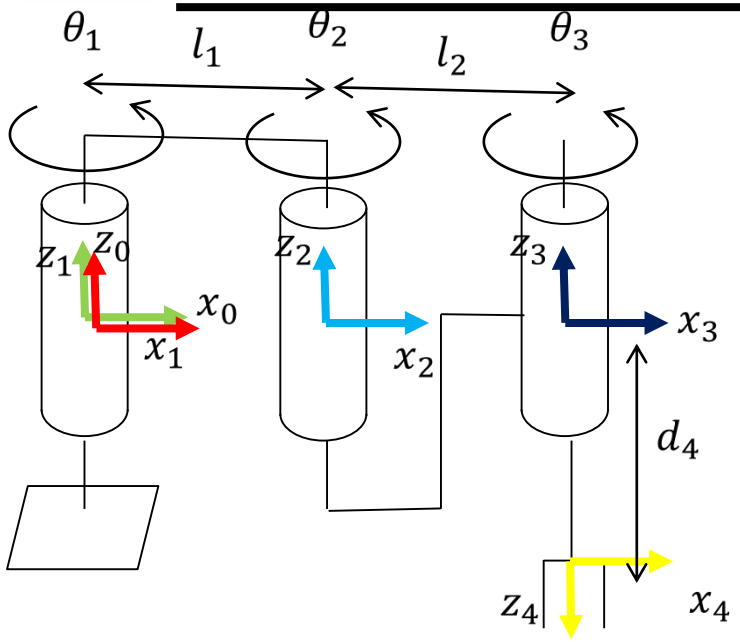
$$\tau_i = {}^{i}n_{i}^{T}\hat{z}_i = [{}^{i}n_{ix} \quad {}^{i}n_{iy} \quad {}^{i}n_{iz}]\begin{bmatrix}0\\0\\1\end{bmatrix}$$

Revolute Joint

$$f_i = {}^{i}f_{i}^{T}\hat{z}_i = [{}^{i}f_{ix} \quad {}^{i}f_{iy} \quad {}^{i}f_{iz}]\begin{bmatrix}0\\0\\1\end{bmatrix}$$

Prismatic Joint

SCARA – RRRP – DH Parameter (Modified form)



Modified form:

a_{i-1} - The distance from $\hat{Z}_{i-1}$ to $\hat{Z}_i$ measured along $\hat{X}_{i-1}$

α_{i-1} - The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured about $\hat{X}_{i-1}$

d_i - The distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$

θ_i - The angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$

i-1	*i*	α_{i-1}	a_{i-1}	d_i	θ_i
0	1	0	0	0	θ_1
1	2	0	l_1	0	θ_2
2	3	0	l_2	0	θ_3
3	4	180°	0	d_4	0

SCARA – RRRP

Modified DH parameter

```
syms pi

L(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
L(2) = Link('revolute','d',0,'a',l1,'alpha',0,'modified');
L(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
L(4) = Link('prismatic','alpha',pi,'theta', 0 ,'a',0,'modified')

SCARA = SerialLink(L,'name','SCARA')
```

```
SCARA:: 4 axis, RRRP, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|        q1|          0|          0|          0|          0|
|  2|        q2|          0|         l1|          0|          0|
|  3|        q3|          0|         l2|          0|          0|
|  4|         0|         q4|          0|         pi|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

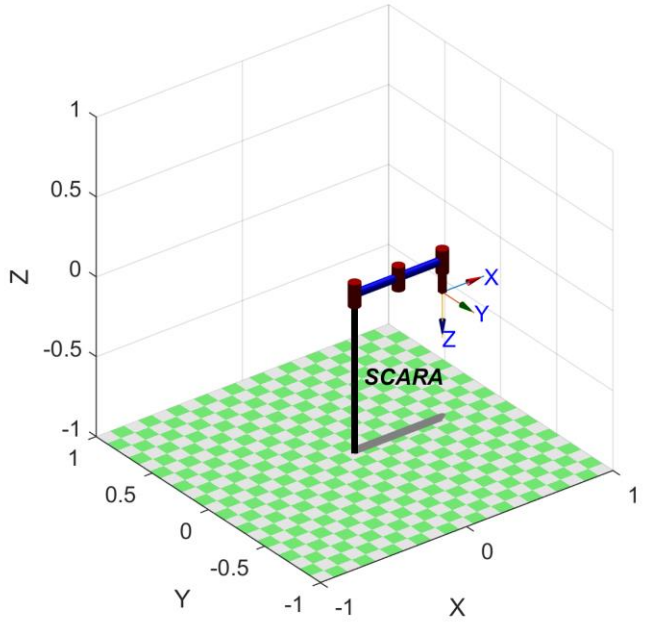
For plot [mm]

```
l1 = 0.3; l2=0.3; pi= 3.14;
t1 =0; t2=0; t3=0; d4=0.2;
th= [t1 t2 t3 d4];

L_P(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
L_P(2) = Link('revolute','d',0,'a',l1,'alpha',0,'modified');
L_P(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
L_P(4) = Link('prismatic','alpha',pi,'theta', 0 ,'a',0,'modified')

SCARA_P = SerialLink(L_P,'name','SCARA')

figure()
SCARA_P.plot(th,'workspace',[-1 1 -1 1 -1 1])
```



Simplify Function

```
T0_4 = SCARA.A([1 2 3 4],q)
```

T0_4 =

$$\begin{pmatrix} \cos(t_3)\,\sigma_2 - \sin(t_3)\,\sigma_3 & \sigma_1 & 0 & l_2\,\sigma_2 + l_1\cos(t_1) \\ \sigma_1 & \sin(t_3)\,\sigma_3 - \cos(t_3)\,\sigma_2 & 0 & l_2\,\sigma_3 + l_1\sin(t_1) \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \cos(t_3)\,\sigma_3 + \sin(t_3)\,\sigma_2$$

$$\sigma_2 = \cos(t_1)\cos(t_2) - \sin(t_1)\sin(t_2)$$

$$\sigma_3 = \cos(t_1)\sin(t_2) + \cos(t_2)\sin(t_1)$$

```
simplified_T0_4 = simplify(SCARA.A([1 2 3 4],q))
```

simplified_T0_4 =

$$\begin{pmatrix} \cos(t_1+t_2+t_3) & \sin(t_1+t_2+t_3) & 0 & l_2\cos(t_1+t_2) + l_1\cos(t_1) \\ \sin(t_1+t_2+t_3) & -\cos(t_1+t_2+t_3) & 0 & l_2\sin(t_1+t_2) + l_1\sin(t_1) \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

UCLA

syms pi

3D-1-RRR Modi

```
clear all; close all;
syms t1 t2 t3 l2 l3

 L1(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
 L1(2) = Link('revolute','d',0,'a',0,'alpha',-pi/2,'modified');
 L1(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
 L1(4) = Link('revolute','d',0,'a',l3,'alpha',0,'modified');
 Robot_3D_1_RRR = SerialLink(L1, 'name', '3D-1-RRR Modi')
```

```
Robot_3D_1_RRR =

3D-1-RRR Modi:: 4 axis, RRRR, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|          0|          0|
|  2|         q2|          0|          0|    -1.5708|          0|
|  3|         q3|          0|         l2|          0|          0|
|  4|         q4|          0|         l3|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

3D-1-RRR Modi_Direct Kinematic wiq Robotics Toolbox

T0_4 =

$$\begin{pmatrix} \sigma_4+\sigma_2 & -\sigma_3-\sigma_1 & -\sin(t_1) & l_3\,(\sigma_4+\sigma_2)+l_2\left(\frac{811296384146066767280314051225 53\cos(t_1-t_2)}{16225927682921336339157801028 8128}+\frac{811296384146066866635466051655 75\cos(t_1+t_2)}{16225927682921336339157801028 8128}\right) \\ \sigma_3-\sigma_1 & \sigma_4-\sigma_2 & \cos(t_1) & l_3\,(\sigma_3-\sigma_1)+l_2\left(\frac{811296384146066767280314051225 53\sin(t_1-t_2)}{16225927682921336339157801028 8128}+\frac{811296384146066866635466051655 75\sin(t_1+t_2)}{16225927682921336339157801028 8128}\right) \\ -\sin(t_2+t_3) & -\cos(t_2+t_3) & \frac{4967757600021511}{811296384146066816957890051 44064} & -l_3\sin(t_2+t_3)-l_2\sin(t_2) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1=\frac{811296384146066767280314051225 53\sin(t_2-t_1+t_3)}{16225927682921336339157801028 8128}$$

$$\sigma_2=\frac{811296384146066767280314051225 53\cos(t_2-t_1+t_3)}{16225927682921336339157801028 8128}$$

$$\sigma_3=\frac{811296384146066866635466051655 75\sin(t_1+t_2+t_3)}{16225927682921336339157801028 8128}$$

$$\sigma_4=\frac{811296384146066866635466051655 75\cos(t_1+t_2+t_3)}{16225927682921336339157801028 8128}$$

3D-1-RRR Modi

```
clear all; close all;
syms t1 t2 t3 l2 l3
syms pi

 L1(1) = Link('revolute','d',0,'a',0,'alpha',0,'modified');
 L1(2) = Link('revolute','d',0,'a',0,'alpha',-pi/2,'modified');
 L1(3) = Link('revolute','d',0,'a',l2,'alpha',0,'modified');
 L1(4) = Link('revolute','d',0,'a',l3,'alpha',0,'modified');
 Robot_3D_1_RRR = SerialLink(L1, 'name', '3D-1-RRR Modi')
```

```
Robot_3D_1_RRR =

3D-1-RRR Modi:: 4 axis, RRRR, modDH, slowRNE
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|          0|          0|          0|          0|
|  2|         q2|          0|          0|      -pi/2|          0|
|  3|         q3|          0|         l2|          0|          0|
|  4|         q4|          0|         l3|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
```

3D-1-RRR Modi_Direct Kinematics

T0_4 =

$$\begin{pmatrix} \cos(t_2+t_3)\cos(t_1) & -\sin(t_2+t_3)\cos(t_1) & -\sin(t_1) & \cos(t_1)\,\sigma_1 \\ \cos(t_2+t_3)\sin(t_1) & -\sin(t_2+t_3)\sin(t_1) & \cos(t_1) & \sin(t_1)\,\sigma_1 \\ -\sin(t_2+t_3) & -\cos(t_2+t_3) & 0 & -l_3\sin(t_2+t_3)-l_2\sin(t_2) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = l_3\cos(t_2+t_3)+l_2\cos(t_2)$$

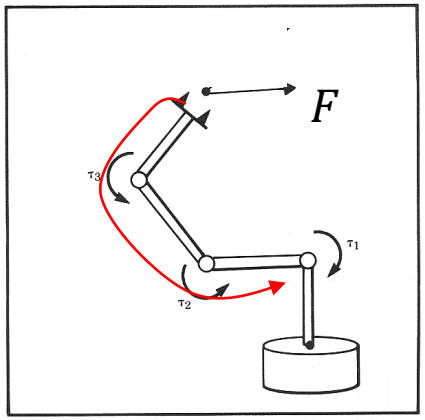
SCARA example _ Jacobian: Static force propagation

- The recursive equations for static force and torque from link *i+1* to link *i*

$$^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1}$$

$$^{i}n_i = {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i$$

- It allows us to start with the force and torque applied at the end effector.
- By using this recursive equation, we can calculate the force and torque at each joint all the way back to the robot base frame



Starting with $^{4}f_4 = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$ $^{4}n_4 = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$

$$^{3}f_3 = {}^{3}_{4}R\,{}^{4}f_4 \qquad ^{3}n_3 = {}^{3}_{4}R\,{}^{4}n_4 + {}^{3}P_4 \times {}^{3}f_3$$

$$^{2}f_2 = {}^{2}_{3}R\,{}^{3}f_3 \qquad ^{2}n_2 = {}^{2}_{3}R\,{}^{3}n_3 + {}^{2}P_3 \times {}^{2}f_2$$

$$^{1}f_1 = {}^{1}_{2}R\,{}^{2}f_2 \qquad ^{1}n_1 = {}^{1}_{2}R\,{}^{2}n_2 + {}^{1}P_2 \times {}^{1}f_1$$

SCARA example _ Jacobian: Static force propagation

- **Force propagation**

$$ {}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} $$

$$ {}^{4}f_4 = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} $$

$$ {}^{3}f_3 = {}^{3}_{4}R\,{}^{4}f_4 $$

$$ {}^{2}f_2 = {}^{2}_{3}R\,{}^{3}f_3 $$

$$ {}^{1}f_1 = {}^{1}_{2}R\,{}^{2}f_2 $$

```
f4 = [fx;fy;fz]
f3 = R3_4*f4;
f3 = collect(f3,[fx fy fz])
f2 = R2_3*f3;
f2 = collect(f2,[fx fy fz])
f1 = R1_2*f2;
f1 = simplify(collect(f1,[fx fy fz]))
```

SCARA example _ Jacobian: Static force propagation

- **Force propagation**

$$ {}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} $$

$$ {}^{4}f_4 = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} $$

$$ {}^{3}f_3 = {}^{3}_{4}R\,{}^{4}f_4 $$

$$ {}^{2}f_2 = {}^{2}_{3}R\,{}^{3}f_3 $$

$$ {}^{1}f_1 = {}^{1}_{2}R\,{}^{2}f_2 $$

```
f4 = [fx;fy;fz]
f3 = R3_4*f4;
f3 = collect(f3,[fx fy fz])
f2 = R2_3*f3;
f2 = collect(f2,[fx fy fz])
f1 = R1_2*f2;
f1 = simplify(collect(f1,[fx fy fz]))
```

```
R0_1 = t2r(T0_1)
R1_2 = t2r(T1_2)
R2_3 = t2r(T2_3)
R3_4 = t2r(T3_4)
```

R0_1 =

$$ \begin{pmatrix} \cos(t_1) & -\sin(t_1) & 0 \\ \sin(t_1) & \cos(t_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

R1_2 =

$$ \begin{pmatrix} \cos(t_2) & -\sin(t_2) & 0 \\ \sin(t_2) & \cos(t_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

R2_3 =

$$ \begin{pmatrix} \cos(t_3) & -\sin(t_3) & 0 \\ \sin(t_3) & \cos(t_3) & 0 \\ 0 & 0 & 1 \end{pmatrix} $$

R3_4 =

$$ \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} $$

```
T0_1 = SCARA.A([1],q)
T1_2 = SCARA.A([2],q)
T2_3 = SCARA.A([3],q)
T3_4 = SCARA.A([4],q)
```

T0_1 =

$$ \begin{pmatrix} \cos(t_1) & -\sin(t_1) & 0 & 0 \\ \sin(t_1) & \cos(t_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

T1_2 =

$$ \begin{pmatrix} \cos(t_2) & -\sin(t_2) & 0 & l_1 \\ \sin(t_2) & \cos(t_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

T2_3 =

$$ \begin{pmatrix} \cos(t_3) & -\sin(t_3) & 0 & l_2 \\ \sin(t_3) & \cos(t_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

T3_4 =

$$ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

SCARA example _ Jacobian: Static force propagation

- **Force propagation**

$${}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1}$$

$${}^{4}f_4 = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}$$

$${}^{3}f_3 = {}^{3}_{4}R\,{}^{4}f_4$$

$${}^{2}f_2 = {}^{2}_{3}R\,{}^{3}f_3$$

$${}^{1}f_1 = {}^{1}_{2}R\,{}^{2}f_2$$

```
f4 = [fx;fy;fz]
f3 = R3_4*f4;
f3 = collect(f3,[fx fy fz])
f2 = R2_3*f3;
f2 = collect(f2,[fx fy fz])
f1 = R1_2*f2;
f1 = simplify(collect(f1,[fx fy fz]))
```

f4 =

$$\begin{pmatrix} \mathrm{fx} \\ \mathrm{fy} \\ \mathrm{fz} \end{pmatrix}$$

f3 =

$$\begin{pmatrix} \mathrm{fx} \\ -\mathrm{fy} \\ -\mathrm{fz} \end{pmatrix}$$

f2 =

$$\begin{pmatrix} \cos(t_3)\ \mathrm{fx} + \sin(t_3)\ \mathrm{fy} \\ \sin(t_3)\ \mathrm{fx} + (-\cos(t_3))\ \mathrm{fy} \\ -\mathrm{fz} \end{pmatrix}$$

f1 =

$$\begin{pmatrix} \mathrm{fx}\cos(t_2+t_3) + \mathrm{fy}\sin(t_2+t_3) \\ \mathrm{fx}\sin(t_2+t_3) - \mathrm{fy}\cos(t_2+t_3) \\ -\mathrm{fz} \end{pmatrix}$$

SCARA example _ Jacobian: Static force propagation

- **Torque propagation** $${}^{i}n_i = {}_{i+1}^{i}R\,{}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i$$

$${}^{4}n_4 = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

$${}^{3}n_3 = {}_{4}^{3}R\,{}^{4}n_4 + {}^{3}P_4 \times {}^{3}f_3$$

$${}^{2}n_2 = {}_{3}^{2}R\,{}^{3}n_3 + {}^{2}P_3 \times {}^{2}f_2$$

$${}^{1}n_1 = {}_{2}^{1}R\,{}^{2}n_2 + {}^{1}P_2 \times {}^{1}f_1$$

```
n4 = [nx;ny;nz]
n3 = R3_4*n4 + cross(P3_4,f3);
n3 = collect(n3,[fx fy fz nx ny nz])
n2 = R2_3*n3 + cross(P2_3,f2);
n2 = simplify(collect(n2,[fx fy fz nx ny nz]))
n2 = collect(n2,[fx fy fz nx ny nz])
n1 = R1_2*n2 + cross(P1_2,f1);
```

SCARA example _ Jacobian: Static force propagation

- **Torque propagation** $\quad {}^{i}n_i = {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i$

$$ {}^{4}n_4 = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} $$

$$ {}^{3}n_3 = {}^{3}_{4}R\,{}^{4}n_4 + {}^{3}P_4 \times {}^{3}f_3 $$

$$ {}^{2}n_2 = {}^{2}_{3}R\,{}^{3}n_3 + {}^{2}P_3 \times {}^{2}f_2 $$

$$ {}^{1}n_1 = {}^{1}_{2}R\,{}^{2}n_2 + {}^{1}P_2 \times {}^{1}f_1 $$

```
n4 = [nx;ny;nz]
n3 = R3_4*n4 + cross(P3_4,f3);
n3 = collect(n3,[fx fy fz nx ny nz])
n2 = R2_3*n3 + cross(P2_3,f2);
n2 = simplify(collect(n2,[fx fy fz nx ny nz]))
n2 = collect(n2,[fx fy fz nx ny nz])
n1 = R1_2*n2 + cross(P1_2,f1);
```

n4 =

$$\begin{pmatrix} \text{nx} \\ \text{ny} \\ \text{nz} \end{pmatrix}$$

n3 =

$$\begin{pmatrix} (-d_4)\ \text{fy} + \text{nx} \\ (-d_4)\ \text{fx} - \text{ny} \\ -\text{nz} \end{pmatrix}$$

n2 =

$$\begin{pmatrix} (d_4 \sin(t_3))\ \text{fx} + \sigma_1\ \text{fy} + \cos(t_3)\,\text{nx} + \sin(t_3)\,\text{ny} \\ \sigma_1\ \text{fx} + (-d_4 \sin(t_3))\ \text{fy} + l_2\ \text{fz} + \sin(t_3)\,\text{nx} + (-\cos(t_3))\,\text{ny} \\ (l_2 \sin(t_3))\ \text{fx} + (-l_2 \cos(t_3))\ \text{fy} - \text{nz} \end{pmatrix}$$

where

$$\sigma_1 = -d_4 \cos(t_3)$$

n1 =

$$\begin{pmatrix} \sigma_2\ \text{fx} + \sigma_1\ \text{fy} + (-l_2 \sin(t_2))\ \text{fz} + \cos(t_2 + t_3)\,\text{nx} + \sin(t_2 + t_3)\,\text{ny} \\ \sigma_1\ \text{fx} + (-\sigma_2)\ \text{fy} + (l_1 + l_2 \cos(t_2))\ \text{fz} + \sin(t_2 + t_3)\,\text{nx} + (-\cos(t_2 + t_3))\,\text{ny} \\ (l_1 \sin(t_2 + t_3) + l_2 \sin(t_3))\ \text{fx} + (-l_1 \cos(t_2 + t_3) - l_2 \cos(t_3))\ \text{fy} - \text{nz} \end{pmatrix}$$

where

$$\sigma_1 = -d_4 \cos(t_2 + t_3)$$

$$\sigma_2 = d_4 \sin(t_2 + t_3)$$

- To find the joint torque required to maintain the static equilibrium for a revolute joint,

$$\tau_i = {}^i n_i{}^T \hat{z}_i = \begin{bmatrix} {}^i n_{i,x} & {}^i n_{i,y} & {}^i n_{i,z} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- To find the joint force required to maintain the static equilibrium for a prismatic joint,

$$f_i = {}^i f_i{}^T \hat{z}_i = \begin{bmatrix} {}^i f_{i,x} & {}^i f_{i,y} & {}^i f_{i,z} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- We are only interested in ${}^i n_{i,z}$ and ${}^i f_{i,z}$

$$[\tau] = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} = \begin{bmatrix} {}^1 n_{1,z} \\ {}^2 n_{2,z} \\ {}^3 n_{3,z} \\ {}^4 f_{4,z} \end{bmatrix}$$

SCARA example _ Jacobian: Static force propagation

$$[\tau] = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} = \begin{bmatrix} {}^1n_{1,z} \\ {}^2n_{2,z} \\ {}^3n_{3,z} \\ {}^4f_{4,z} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} {}^1n_{1,x} & {}^1n_{1,y} & {}^1n_{1z} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ {}^2n_{2,z} \\ {}^3n_{3,z} \\ \begin{bmatrix} {}^4f_{4,x} & {}^4f_{4,y} & {}^4f_{4,z} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix}$$

```
tq1 = collect(transpose(n1)*[0;0;1],[fx fy fz nx ny nz])
tq2 = collect(transpose(n2)*[0;0;1],[fx fy fz nx ny nz])
tq3 = collect(transpose(n3)*[0;0;1],[fx fy fz nx ny nz])
tq4 = collect(transpose(f4)*[0;0;1],[fx fy fz nx ny nz])
```

tq1 = $(l_1 \sin(t_2 + t_3) + l_2 \sin(t_3))\ \text{fx} + (-l_1 \cos(t_2 + t_3) - l_2 \cos(t_3))\ \text{fy} - \text{nz}$

tq2 = $(l_2 \sin(t_3))\ \text{fx} + (-l_2 \cos(t_3))\ \text{fy} - \text{nz}$

tq3 = $-\text{nz}$

tq4 = fz

UCLA

SCARA example _ Jacobian: Static force propagation

The Jacobian $\left({}^{4}J_{4}\right)$ is defined as

$$[\tau] = \left[\,{}^{4}J_{4}\,\right]^{T} \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

According to the definition $\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} = \begin{bmatrix} {}^{1}n_{1,z} \\ {}^{2}n_{2,z} \\ {}^{3}n_{3,z} \\ {}^{4}f_{4,z} \end{bmatrix}$,

$$\begin{bmatrix} {}^{1}n_{1,z} \\ {}^{2}n_{2,z} \\ {}^{3}n_{3,z} \\ {}^{4}f_{4,z} \end{bmatrix} = \left[\,{}^{4}J_{4}\right]^{T} \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

UCLA

SCARA example _ Jacobian: Static force propagation

$$\begin{bmatrix} {}^{1}n_{1,z} \\ {}^{2}n_{2,z} \\ {}^{3}n_{3,z} \\ {}^{4}f_{4,z} \end{bmatrix} = \left[{}^{4}J_{4} \right]^{T} \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

```
TQ1 = equationsToMatrix([tq1],[fx fy fz nx ny nz])
TQ2 = equationsToMatrix([tq2],[fx fy fz nx ny nz])
TQ3 = equationsToMatrix([tq3],[fx fy fz nx ny nz])
TQ4 = equationsToMatrix([tq4],[fx fy fz nx ny nz])

J4_FP_T = [TQ1;TQ2;TQ3;TQ4]
J4_FP = transpose(J4_FP_T)
```

TQ1 = $\begin{pmatrix} l_1 \sin(t_2+t_3)+l_2 \sin(t_3) & -l_1 \cos(t_2+t_3)-l_2 \cos(t_3) & 0 & 0 & 0 & -1 \end{pmatrix}$

TQ2 = $\begin{pmatrix} l_2 \sin(t_3) & -l_2 \cos(t_3) & 0 & 0 & 0 & -1 \end{pmatrix}$

TQ3 = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$

TQ4 = $\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$

J4_FP_T =

$$\begin{pmatrix} l_1 \sin(t_2+t_3)+l_2 \sin(t_3) & -l_1 \cos(t_2+t_3)-l_2 \cos(t_3) & 0 & 0 & 0 & -1 \\ l_2 \sin(t_3) & -l_2 \cos(t_3) & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

UCLA

SCARA example _ Jacobian: Static force propagation

$$\left[{}^{4}J_4 \right] = \left[{}^{4}J_4{}^{T} \right]^{T}$$

```
J4_FP_T = [TQ1;TQ2;TQ3;TQ4]
J4_FP = transpose(J4_FP_T)
```

J4_FP_T =

$$\begin{pmatrix} l_1 \sin(t_2+t_3) + l_2 \sin(t_3) & -l_1 \cos(t_2+t_3) - l_2 \cos(t_3) & 0 & 0 & 0 & -1 \\ l_2 \sin(t_3) & -l_2 \cos(t_3) & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

J4_FP =

$$\begin{pmatrix} l_1 \sin(t_2+t_3) + l_2 \sin(t_3) & l_2 \sin(t_3) & 0 & 0 \\ -l_1 \cos(t_2+t_3) - l_2 \cos(t_3) & -l_2 \cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

UCLA

Jacobian Methods of Derivation & the Corresponding Reference Frame – Summary

Method	Jacobian Matrix Reference Frame	Transformation to Base Frame (Frame 0)
Explicit (Diff. the Forward Kinematic Eq.)	${}^{0}J_{N}$	None
Iterative Velocity Eq.	${}^{N}J_{N}$	Transform Method 1: ${}^{0}v_{N} = {}^{0}_{N}R\,{}^{N}v_{N}$; ${}^{0}\omega_{N} = {}^{0}_{N}R\,{}^{N}\omega_{N}$; Transform Method 2: ${}^{0}J_{N}(\theta) = \begin{bmatrix} {}^{0}_{N}R & 0 \\ 0 & {}^{0}_{N}R \end{bmatrix} {}^{N}J_{N}(\theta)$
Iterative Force Eq.	${}^{N}J_{N}^{T}$	Transpose ${}^{N}J_{N} = [{}^{N}J_{N}^{T}]^{T}$; Transform ${}^{0}J_{N}(\theta) = \begin{bmatrix} {}^{0}_{N}R & 0 \\ 0 & {}^{0}_{N}R \end{bmatrix} {}^{N}J_{N}(\theta)$

UCLA

Jacobian: Frame of Representation

$$
{}^{0}J_4(\theta) = \begin{bmatrix} \begin{bmatrix} {}^{0}_{4}R \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} {}^{0}_{4}R \end{bmatrix} \end{bmatrix} {}^{4}J_4(\theta)
$$

```
% convert 4J4_FP to 0J4_FP
J0_FP=[R0_4 zeros(3,3); zeros(3,3) R0_4]*J4_FP;
J0_FP=simplify(J0_FP)
```

J4_FP =

$$
\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & l_2\sin(t_3) & 0 & 0 \\ -l_1\cos(t_2+t_3)-l_2\cos(t_3) & -l_2\cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}
$$

J0_FP =

$$
\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}
$$

UCLA

SCARA example _ Jacobian Matrix

Differentiation the Forward Kinematics Eqs. (Method 1)

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}$$

Velocity Propagation – Base to EE (Method 2)

J4_VP =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & l_2\sin(t_3) & 0 & 0 \\ -l_1\cos(t_2+t_3)-l_2\cos(t_3) & -l_2\cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

Force/Torque Propagation – EE to Base (Method 3)

J4_FP_T =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & -l_1\cos(t_2+t_3)-l_2\cos(t_3) & 0 & 0 & 0 & -1 \\ l_2\sin(t_3) & -l_2\cos(t_3) & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

J4_FP =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & l_2\sin(t_3) & 0 & 0 \\ -l_1\cos(t_2+t_3)-l_2\cos(t_3) & -l_2\cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

J0_DD =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

J0_VP =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

J0_FP =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

UCLA

Kinematics Relations - Forward & Inverse

- The robot kinematic equations relate the two description of the robot tip location

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}$$

$$\theta = IK(X)$$

$$X = FK(\theta)$$

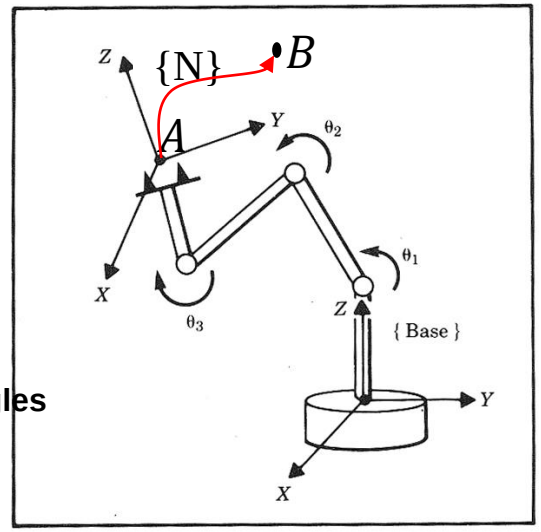
$$X = \begin{bmatrix} {}^0P_N \\ {}^0r_N \end{bmatrix}$$

Euler Angles

Tip Location in Joint Space

$${}^0_N T = \begin{bmatrix} {}^0_N R & {}^0P_N \\ 0 & 1 \end{bmatrix}$$

Tip Location in Cartesian Space



UCLA

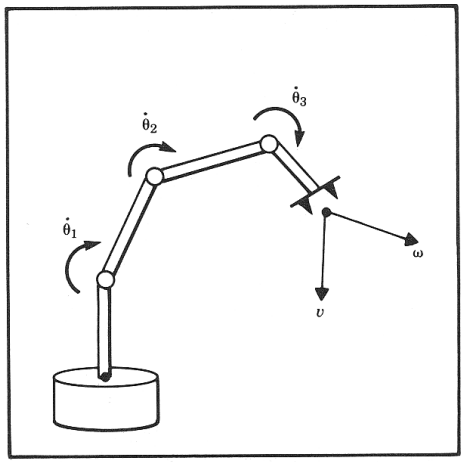
Velocity relationship
: Jacobian Matrix – Joint velocity / End-effector velocity

- **The velocity relationship**: The relationship between the joint angle rates ($\dot{X}$) and the linear and angular velocities of the end effector ($\dot{\theta}$).

$$\dot{X} = J\dot{\theta}$$

$$\dot{\theta} = \frac{d}{dt}[\theta] = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_N \end{bmatrix}$$

$$\dot{X} = \frac{d}{dt}[X] = \begin{bmatrix} \begin{bmatrix} v_N \end{bmatrix} \\ \begin{bmatrix} \omega_N \end{bmatrix} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$



Tip Velocity in Joint Space

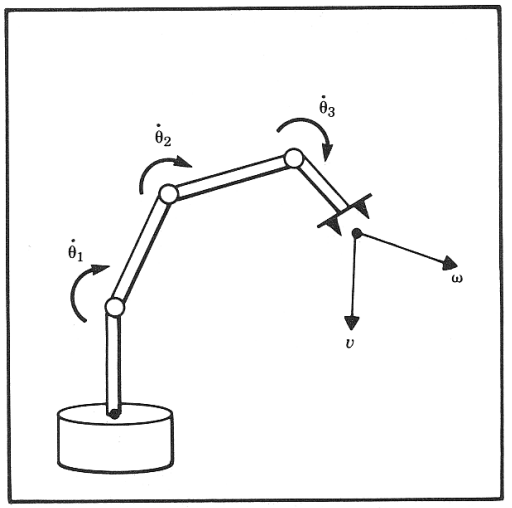
$$\dot{\theta} = J^{-1}\dot{X}$$

Tip velocity in Cartesian Space

Jacobian Matrix - Introduction

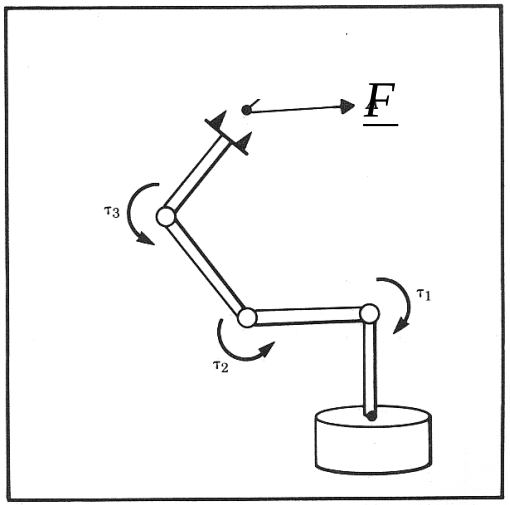
- The velocity relationship
  : The relationship between
  the joint angle rates ($\underline{\dot{\theta}}_N$)
  and the translation and rotation velocities of the end effector ($\underline{\dot{x}}$).

$$\underline{\dot{x}} = J(\underline{\theta})\underline{\dot{\theta}}$$



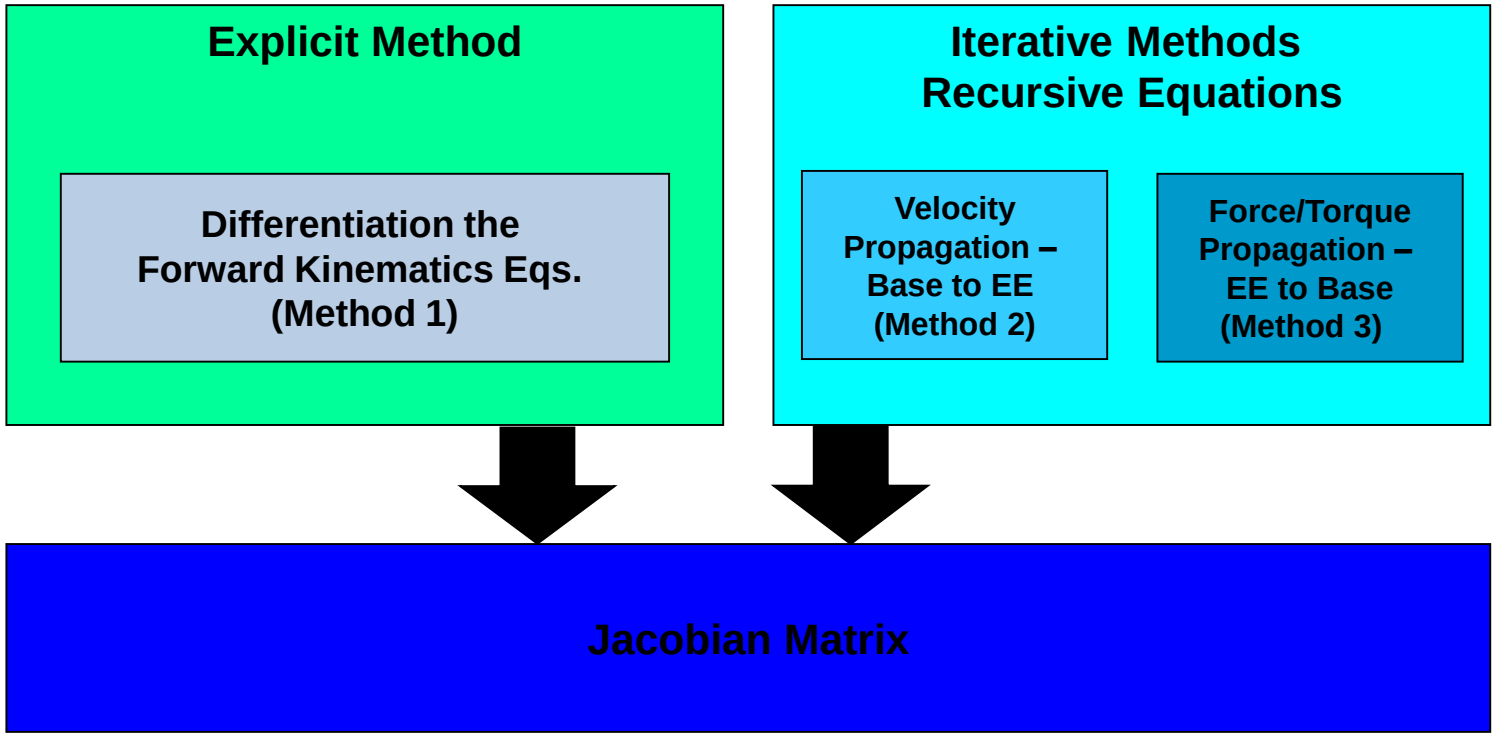
- The relationship between
  the robot joint torques ($\underline{\tau}$)
  and the forces and moments ($\underline{F}$)
  at the robot end effector (**Static Conditions**).

$$\underline{\tau} = J(\underline{\theta})^T \underline{F}$$



UCLA

Jacobian Matrix - Derivation Methods



Jacobian: Direct Differentiation

$$\dot{p}_e = J_P(q)\dot{q} \tag{3.2}$$

$$\omega_e = J_O(q)\dot{q}. \tag{3.3}$$

In (3.2) J_P is the $(3 \times n)$ matrix relating the contribution of the joint velocities $\dot{q}$ to the end-effector *linear* velocity $\dot{p}_e$, while in (3.3) J_O is the $(3 \times n)$ matrix relating the contribution of the joint velocities $\dot{q}$ to the end-effector *angular* velocity ω_e. In compact form, (3.2), (3.3) can be written as

$$v_e = \begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = J(q)\dot{q} \tag{3.4}$$

which represents the manipulator *differential kinematics equation*. The $(6 \times n)$ matrix J is the manipulator *geometric Jacobian*

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}, \tag{3.5}$$

which in general is a function of the joint variables.

Jacobian: Direct Differentiation

In summary, the Jacobian in (3.5) can be partitioned into the (3×1) column vectors $\jmath_{Pi}$ and $\jmath_{Oi}$ as

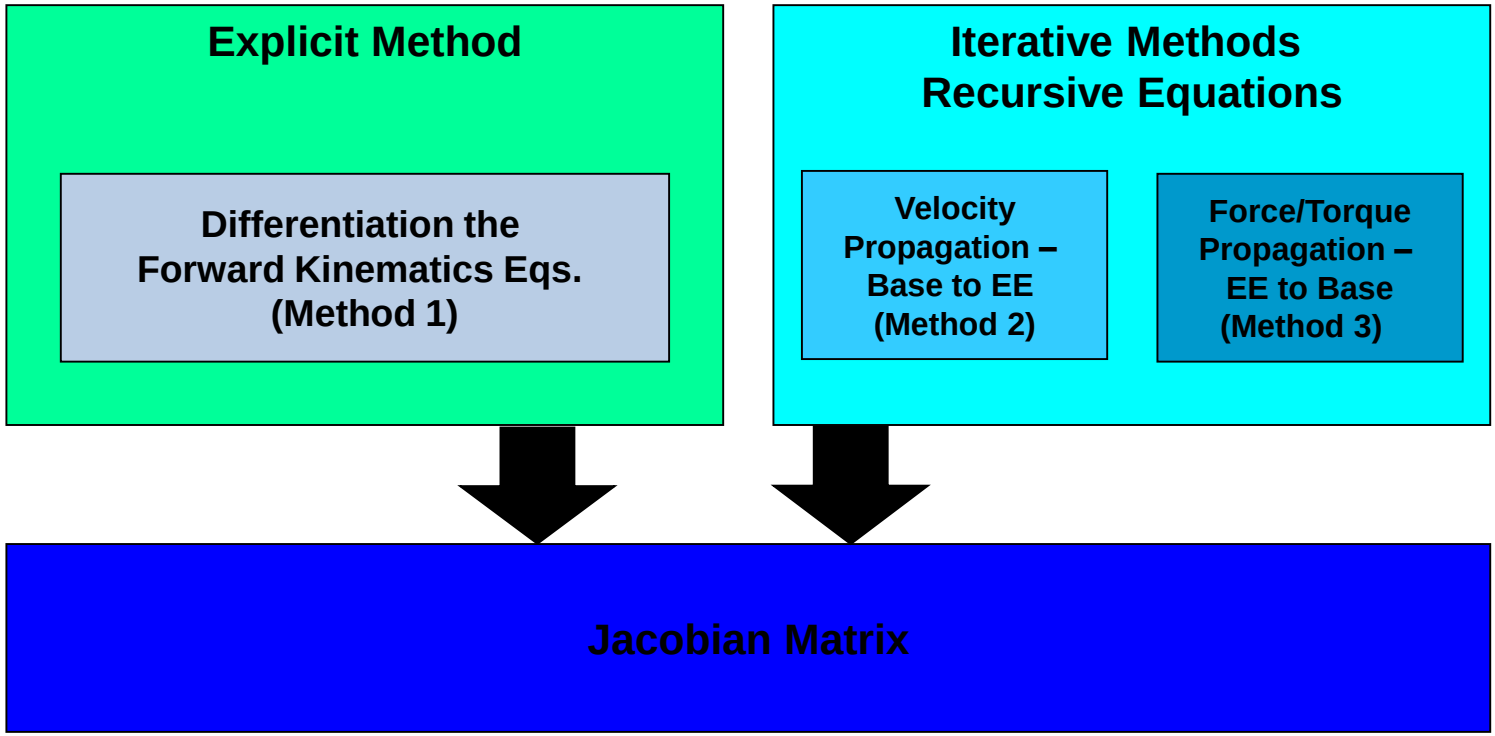
$$J = \begin{bmatrix} \jmath_{P1} & \cdots & \jmath_{Pn} \\ \jmath_{O1} & & \jmath_{On} \end{bmatrix}, \tag{3.29}$$

where

$$\begin{bmatrix} \jmath_{Pi} \\ \jmath_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ \mathbf{0} \end{bmatrix} & \text{for a } \textit{prismatic} \text{ joint} \\ \begin{bmatrix} z_{i-1} \times (p_e - p_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{for a } \textit{revolute} \text{ joint.} \end{cases} \tag{3.30}$$

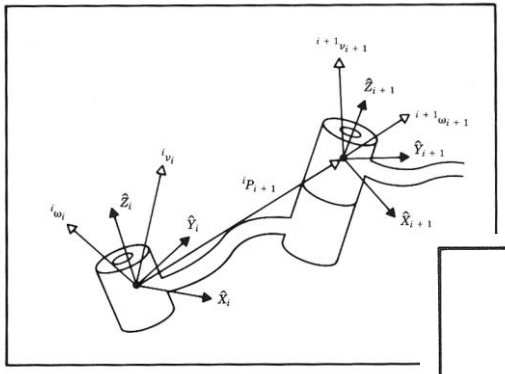
The expressions in (3.30) allow Jacobian computation in a simple, systematic way on the basis of direct kinematics relations. In fact, the vectors z_{i-1}, p_e and p_{i-1} are all functions of the joint variables. In particular:

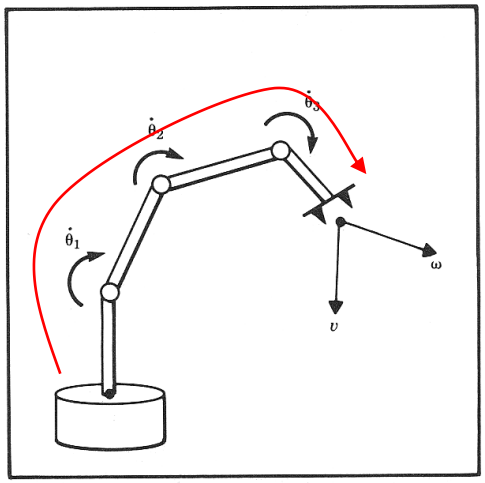
Jacobian Matrix - Derivation Methods



Velocity Propagation

- ***Given:*** A manipulator - A chain of rigid bodies each one capable of moving relative to its neighbor
- ***Problem:*** Calculate the linear and angular velocities of the link of a robot
- ***Solution (Concept):*** Due to the robot structure (serial mechanism) we can **compute the velocities** of each link in order **starting from the base**.





The velocity of link *i*+1
= The velocity of link *i*
+ whatever new velocity components were added by joint *i*+1

UCLA

Velocity Propagation – Intuitive Explanation

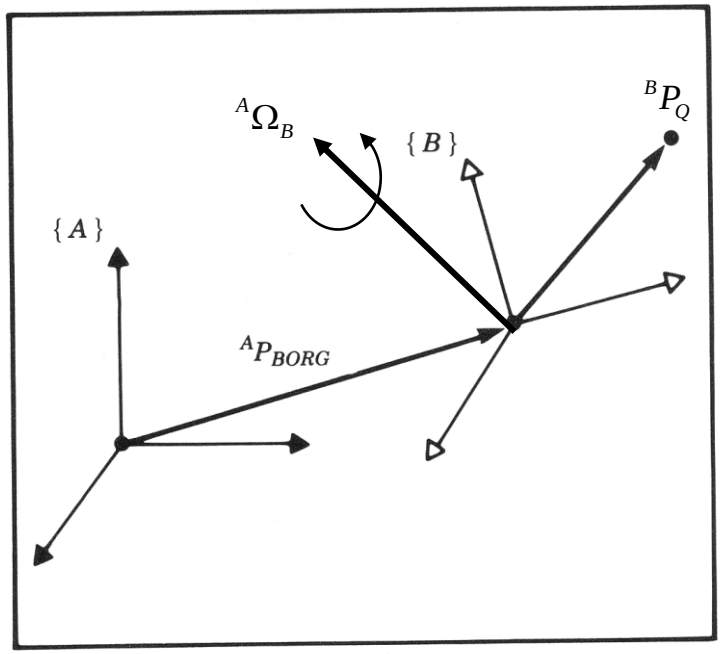
- Three Actions
  - The origin of frame B moves with respect to the origin of frame A
  - Point Q moves with respect to frame B
  - Frame B rotates with respect to frame A about an axis defined by ${}^{A}\Omega_{B}$

- Vector Form

$$ {}^{A}V_{Q} = {}^{A}V_{BORG} + {}^{A}_{B}R\,{}^{B}V_{Q} + {}^{A}\Omega_{B} \times {}^{A}_{B}R\,{}^{B}P_{Q} $$

- Matrix Form

$$ {}^{A}V_{Q} = {}^{A}V_{BORG} + {}^{A}_{B}R\,{}^{B}V_{Q} + {}^{A}_{B}\dot{R}_{\Omega}\left({}^{A}_{B}R\,{}^{B}P_{Q}\right) $$



UCLA

Jacobian: Velocity propagation

- The recursive equation for the Angular Velocity is

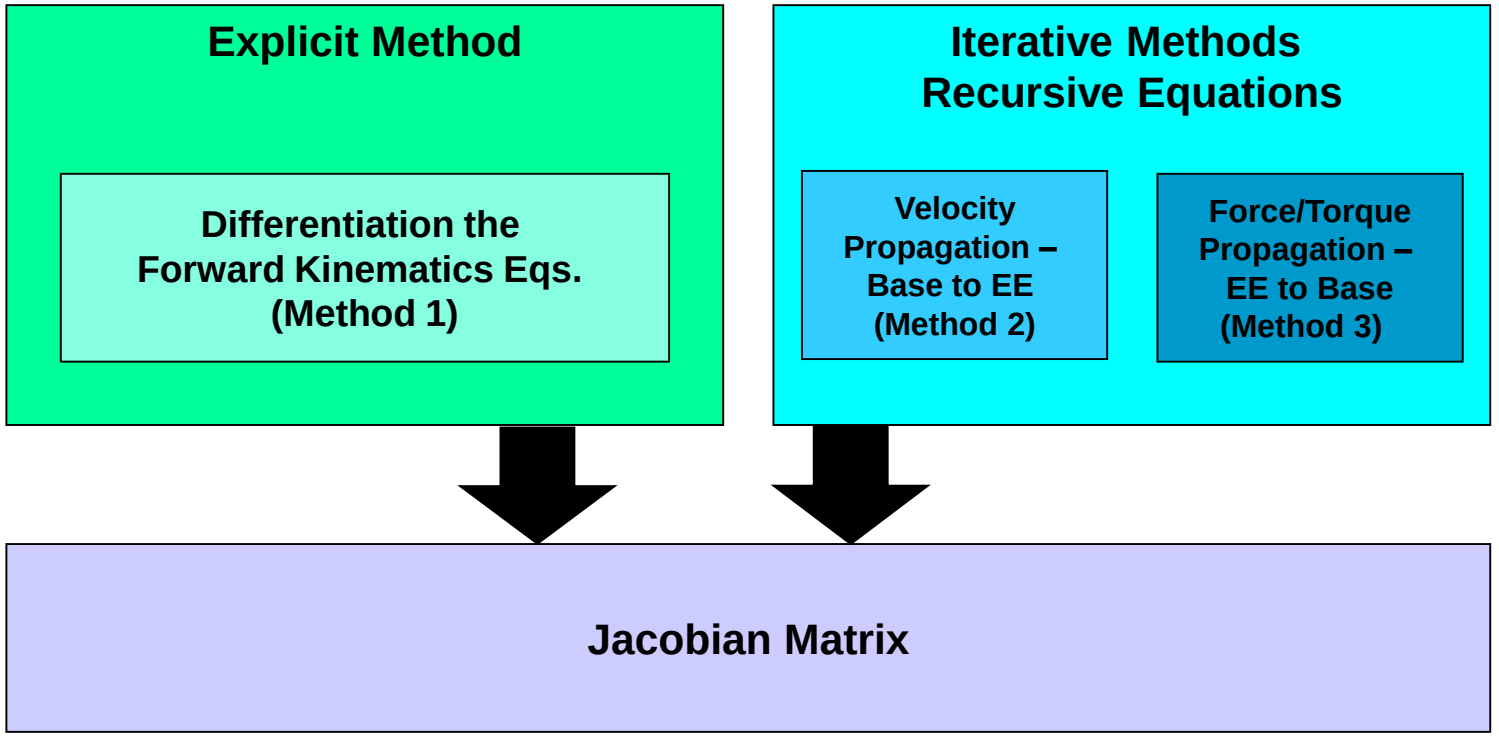
$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\ {}^{i}\omega_i + \rho \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}, \quad \rho = 0\ in\ the\ \ prismatic\ joint,\ \boldsymbol{\rho = 1\ in\ the\ \ revolute\ joint} $$

- The recursive equation for Linear Velocity is

$$ {}^{i+1}v_{i+1} = {}^{i+1}_{i}R\left({}^{i}\omega_i \times {}^{i}P_{i+1} + {}^{i}v_i \right) + \rho \begin{bmatrix} 0 \\ 0 \\ \dot{d} \end{bmatrix}, \quad \rho = 1\ in\ the\ \ prismatic\ joint,\ \boldsymbol{\rho = 0\ in\ the\ \ revolute\ joint} $$

- According to the definition $\begin{bmatrix} {}^{4}v_4 \\ {}^{4}\omega_4 \end{bmatrix} = \begin{bmatrix} {}^{4}J_4 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{d}_4 \end{bmatrix}$, $\begin{bmatrix} {}^{4}v_4 \\ {}^{4}\omega_4 \end{bmatrix} = \begin{bmatrix} {}^{4}v_{4,x} \\ {}^{4}v_{4,y} \\ {}^{4}v_{4,z} \\ {}^{4}w_{4,1} \\ {}^{4}w_{4,2} \\ {}^{4}w_{4,3} \end{bmatrix}$

Jacobian Matrix - Derivation Methods



Jacobian: Force/Torque propagation

Solution

Apply the static equilibrium equations starting from the end effector and going toward the base

Static force "propagation" from link to link

$$ {}^{i}f_i = {}^{i}_{i+1}R \; {}^{i+1}f_{i+1} $$

$$ {}^{i}n_i = {}^{i}_{i+1}R \; {}^{i+1}n_{i+1} + {}^{i}P_{i+1} \times {}^{i}f_i $$

- To find **the joint torque or force required to maintain the static equilibrium**, the dot product of the joint axis vector with the moment vector or force vector acting on the link is computed

$$ \tau_i = {}^{i}n_i^T \hat{z}_i = [{}^{i}n_{ix} \quad {}^{i}n_{iy} \quad {}^{i}n_{iz}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

Revolute Joint

$$ f_i = {}^{i}f_i^T \hat{z}_i = [{}^{i}f_{ix} \quad {}^{i}f_{iy} \quad {}^{i}f_{iz}] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

Prismatic Joint

SCARA example _ Jacobian: Static force propagation

The transpose of Jacobian $\left({}^{4}J_{4}^{T} \right)$ is defined as

$$[\tau] = \left[{}^{4}J_{4} \right]^{T} \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

According to the definition $\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} = \begin{bmatrix} {}^{1}n_{1,z} \\ {}^{2}n_{2,z} \\ {}^{3}n_{3,z} \\ {}^{4}f_{4,z} \end{bmatrix}$,

$$\begin{bmatrix} {}^{1}n_{1,z} \\ {}^{2}n_{2,z} \\ {}^{3}n_{3,z} \\ {}^{4}f_{4,z} \end{bmatrix} = \left[{}^{4}J_{4} \right]^{T} \begin{bmatrix} f_x \\ f_y \\ f_z \\ n_x \\ n_y \\ n_z \end{bmatrix}$$

UCLA

SCARA example _ Jacobian Matrix

Differentiation the Forward Kinematics Eqs. (Method 1)

$$J = \begin{bmatrix} J_P \\ J_O \end{bmatrix}$$

Velocity Propagation – Base to EE (Method 2)

J4_VP =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & l_2\sin(t_3) & 0 & 0 \\ -l_1\cos(t_2+t_3)-l_2\cos(t_3) & -l_2\cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

Force/Torque Propagation – EE to Base (Method 3)

J4_FP_T =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & -l_1\cos(t_2+t_3)-l_2\cos(t_3) & 0 & 0 & 0 & -1 \\ l_2\sin(t_3) & -l_2\cos(t_3) & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

J4_FP =

$$\begin{pmatrix} l_1\sin(t_2+t_3)+l_2\sin(t_3) & l_2\sin(t_3) & 0 & 0 \\ -l_1\cos(t_2+t_3)-l_2\cos(t_3) & -l_2\cos(t_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

J0_DD =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

J0_VP =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

J0_FP =

$$\begin{pmatrix} -l_2\sin(t_1+t_2)-l_1\sin(t_1) & -l_2\sin(t_1+t_2) & 0 & 0 \\ l_2\cos(t_1+t_2)+l_1\cos(t_1) & l_2\cos(t_1+t_2) & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

UCLA

Jacobian Methods of Derivation & the Corresponding Reference Frame – Summary

Method	Jacobian Matrix Reference Frame	Transformation to Base Frame (Frame 0)
Explicit (Diff. the Forward Kinematic Eq.)	${}^{0}J_{N}$	None
Iterative Velocity Eq.	${}^{N}J_{N}$	Transform Method 1: ${}^{0}v_{N} = {}^{0}_{N}R\,{}^{N}v_{N}$ ${}^{0}\omega_{N} = {}^{0}_{N}R\,{}^{N}\omega_{N}$ Transform Method 2: ${}^{0}J_{N}(\theta) = \begin{bmatrix} {}^{0}_{N}R & 0 \\ 0 & {}^{0}_{N}R \end{bmatrix} {}^{N}J_{N}(\theta)$
Iterative Force Eq.	${}^{N}J_{N}^{T}$	Transpose ${}^{N}J_{N} = [{}^{N}J_{N}^{T}]^{T}$ Transform ${}^{0}J_{N}(\theta) = \begin{bmatrix} {}^{0}_{N}R & 0 \\ 0 & {}^{0}_{N}R \end{bmatrix} {}^{N}J_{N}(\theta)$

Summary

- ✓ Jacobian with SCARA example
  - Velocity propagation
  - Direct differentiation
  - Force/Torque propagation
- ✓ Frame of Representation

Next Discussion Section

- **Manipulator Dynamics - Concept**
  - Forward Dynamics
  - Inverse Dynamics
- **Manipulator Dynamics - Solution**
  - Newton-Euler Equations
  - Lagrangian Dynamics

- **Or Lab section with DENSO robot**