April 20, 2023

# Advanced Mechanical Elements
## (Lecture 2)

***Kinematic analysis of spatial link mechanism with the systematic kinematic analysis method***

***-Expansion of the systematic kinematic analysis method to spatial mechanisms -***

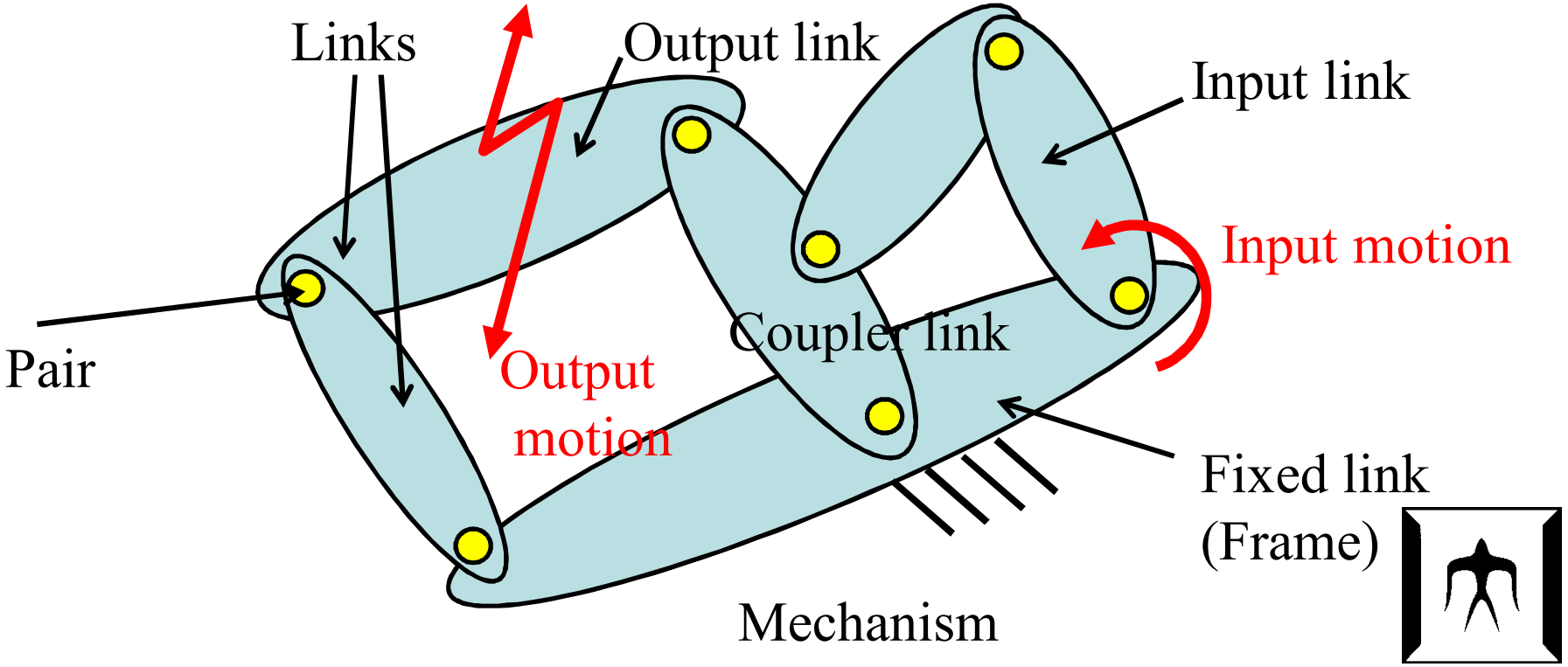

Tokyo Institute of Technology
Dept. of Mechanical Engineering
School of Engineering

**Prof. Nobuyuki Iwatsuki**

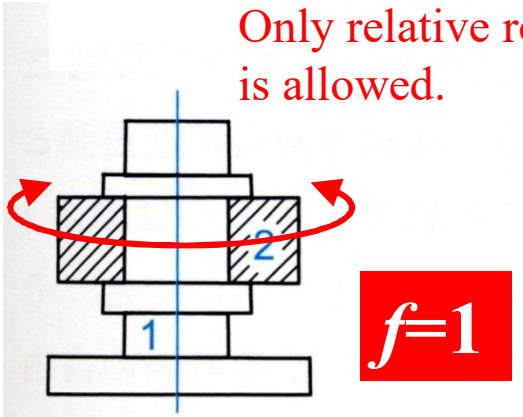
# 1. Construction and DOF of spatial link mechanisms

## Mechanism and kinematic pairs

Mechanism : One link in a link chain which is composed by many links with kinematic pairs should be fixed to frame. Motions of input links can be transformed to motion of output links.



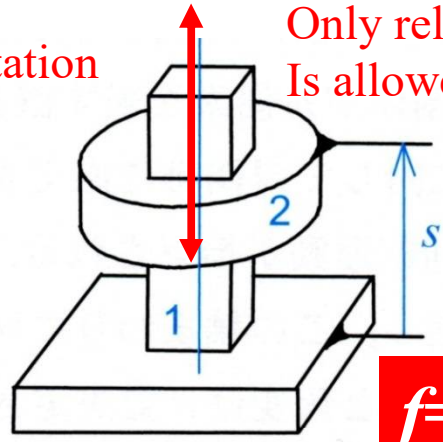
# Various kinematic pairs and their degree-of-freedom




Only relative rotation is allowed.

*f*=1

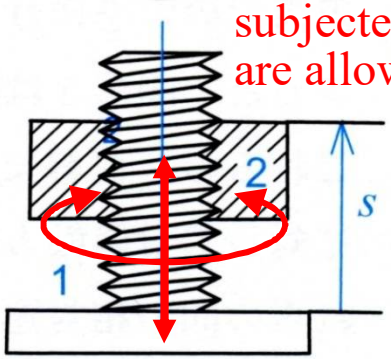
(a)Revolute pair




Only relative translation Is allowed.

*f*=1

(b)Prismatic pair

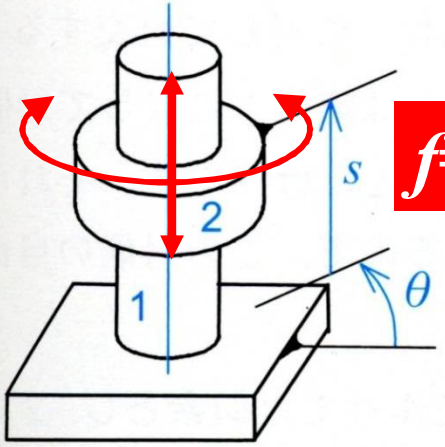



Only relative rotation and translation subjected to each other are allowed.

*f*=1

(c)Helical pair

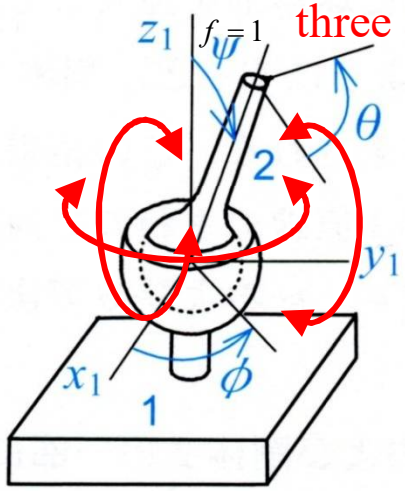
Independent relative rotation and translation are allowed.




*f*=2

(d)Cylindrical pair

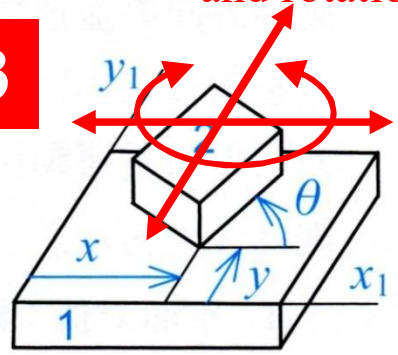
Relative rotations about three axes are allowed.




*f*=3

(e)Spherical pair

Relative 2D translation and rotation are allowed.

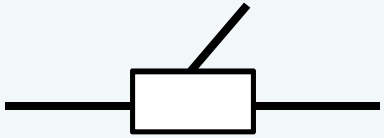
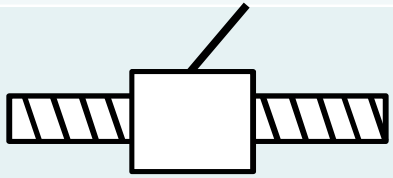
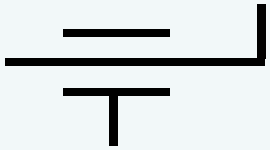
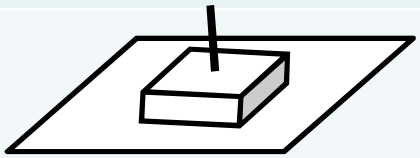


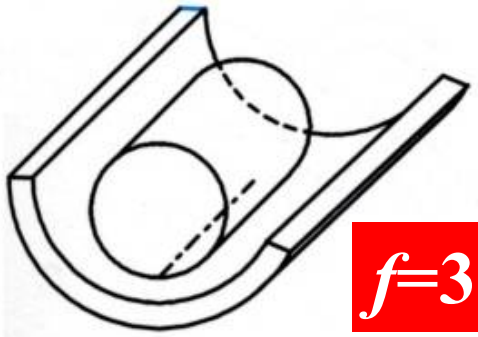

*f*=3

(f)Planar pair

Kinematic pairs with low DOF

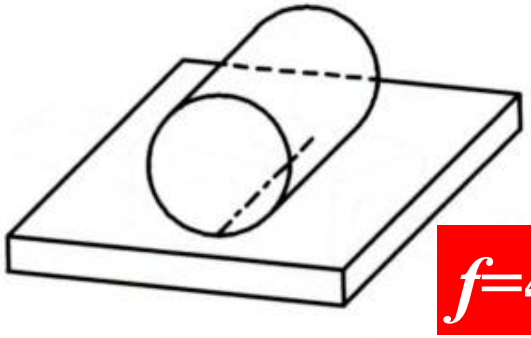
# Representation of kinematic pairs with lower DOF

| Name of pair | Symbol | Representation in mechanism diagram |
|---|---|---|
| Revolute pair | R |  |
| Prismatic pair | P |  |
| Helical pair | H |  |
| Cylindrical pair | C |  |
| Spherical pair | S |  |
| Planar pair | E |  |



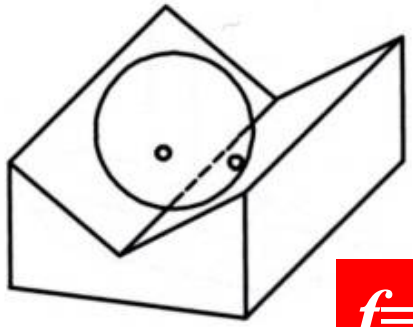
*f*=3

(a)Pillar-cylinder pair



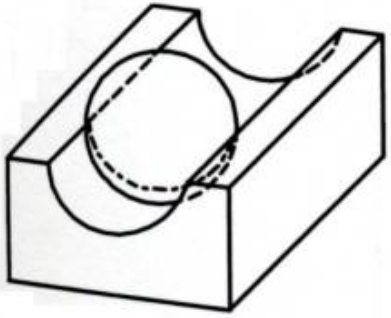
*f*=4

(b)Pillar-plane pair



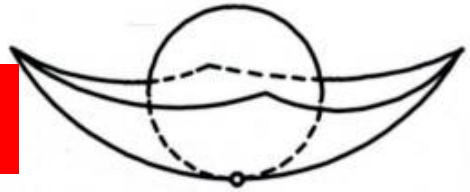
*f*=4

(c)Sphere - V slot pair



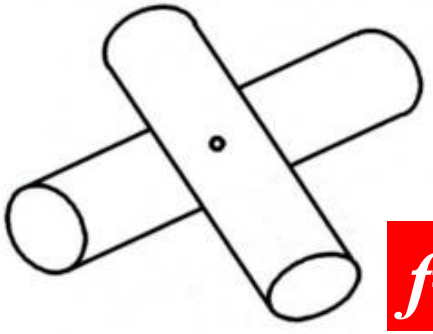
*f*=4

(d)Sphere–circular slot pair



*f*=5

(d)Sphere–spherical plane pair



*f*=5

(d)Two pillar pair

***They are hardly used in general mechanisms.***

Kinematic pairs with high DOF

# Degrees-of-Freedom (DOF) of Mechanism

Definition:

Number of kinematic parameters to determine position and posture of all links in the mechanism

Equation of DOF : (Gruebler's Equation)

$$F = F_S(N-1) - \sum_{f=1}^{F_s-1}(F_S - f)J_f$$

DOF of Mechanisms

DOF of space

Planar: $F_S$=3

Spherical : $F_S$=3

Spatial: $F_S$=6

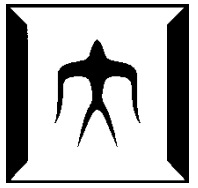
Number of links

DOF of pairs

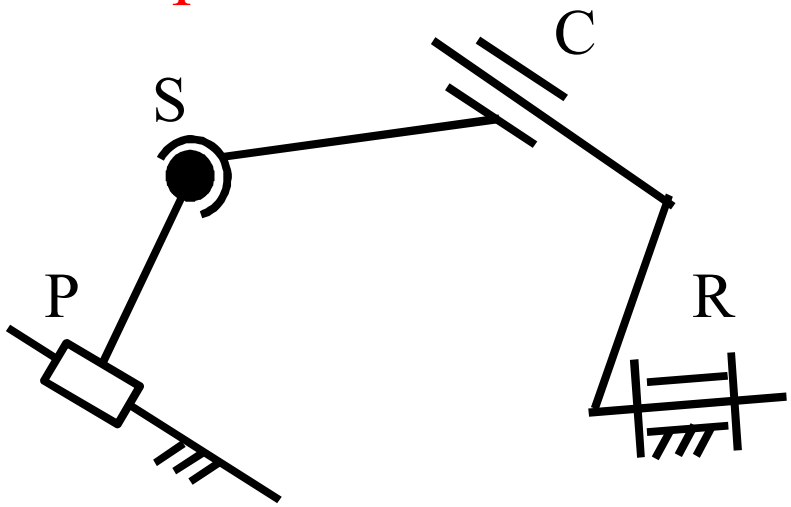
Number of pairs having $f$ DOF

**DOF which are constrained by kinematic pairs**

**Number of parameters to determine position and posture of all moving links which are not constrained with kinematic pairs**

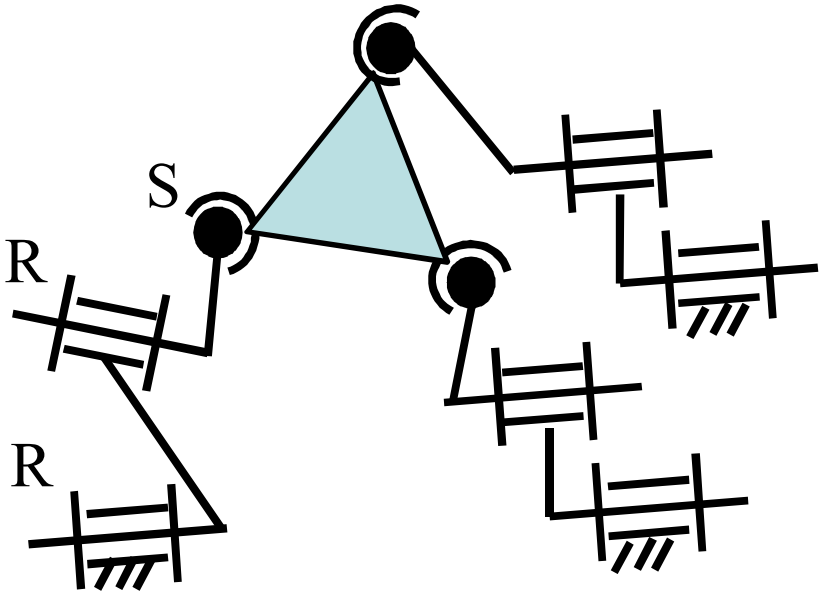


# Example of DOF calculation:



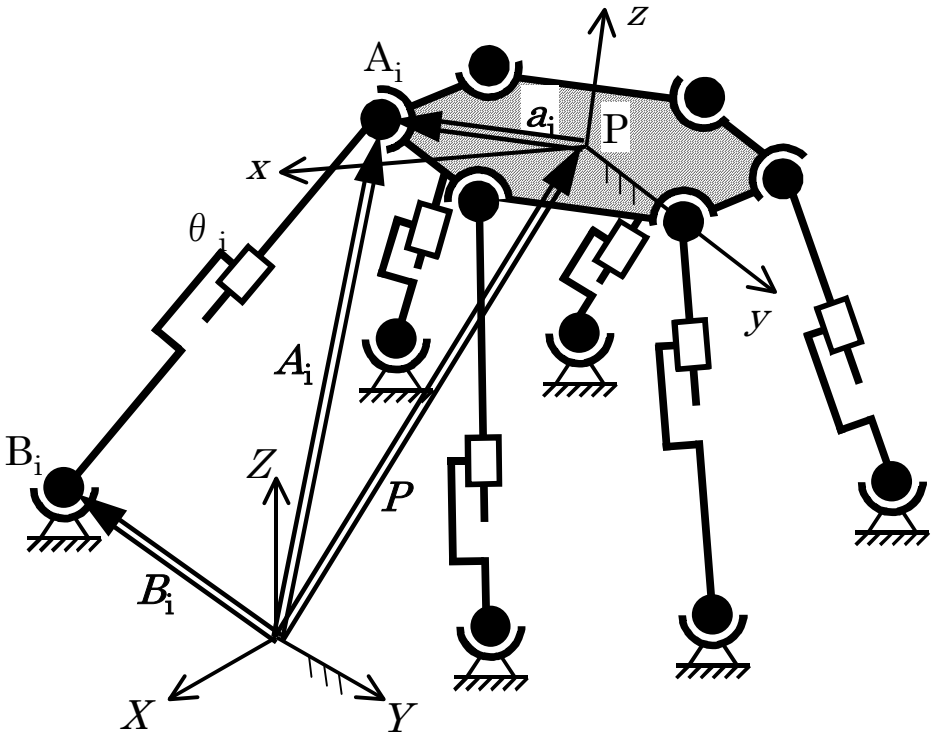

RCSP spatial 4-bar link mechanism

$F$=6(4-1)-(6-1)2-(6-2)-(6-3)=1




3RRS spatial manipulator

$F$=6(8-1)-(6-1)6-(6-3)3=3




Stewart platform Manipulator

$F$=6(14-1)-(6-1)6-(6-3)12=12

**DOF of platform is 6 while having 3 DOF of rotation of S-P-S chain**

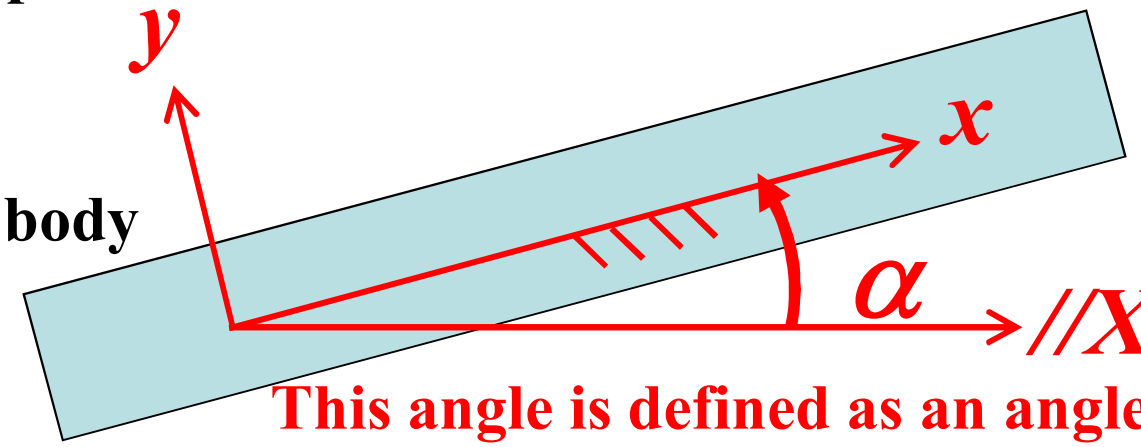
# 2. Understanding of Three-dimensional angular motion

***"You should have studied them in Engineering Dynamics!"***

**Comparison between planar and spatial motions:**

| | Planar motion | | Spatial motion |
|---|---|---|---|
| DOF : | 3 | ⇔ | 6 |
| Translational : | *X,Y coordinates of COG of a rigid body* | | *X,Y,Z coordinates of COG of a rigid body* |
| Angular: | 1 posture angle | | 3 posture angles? |

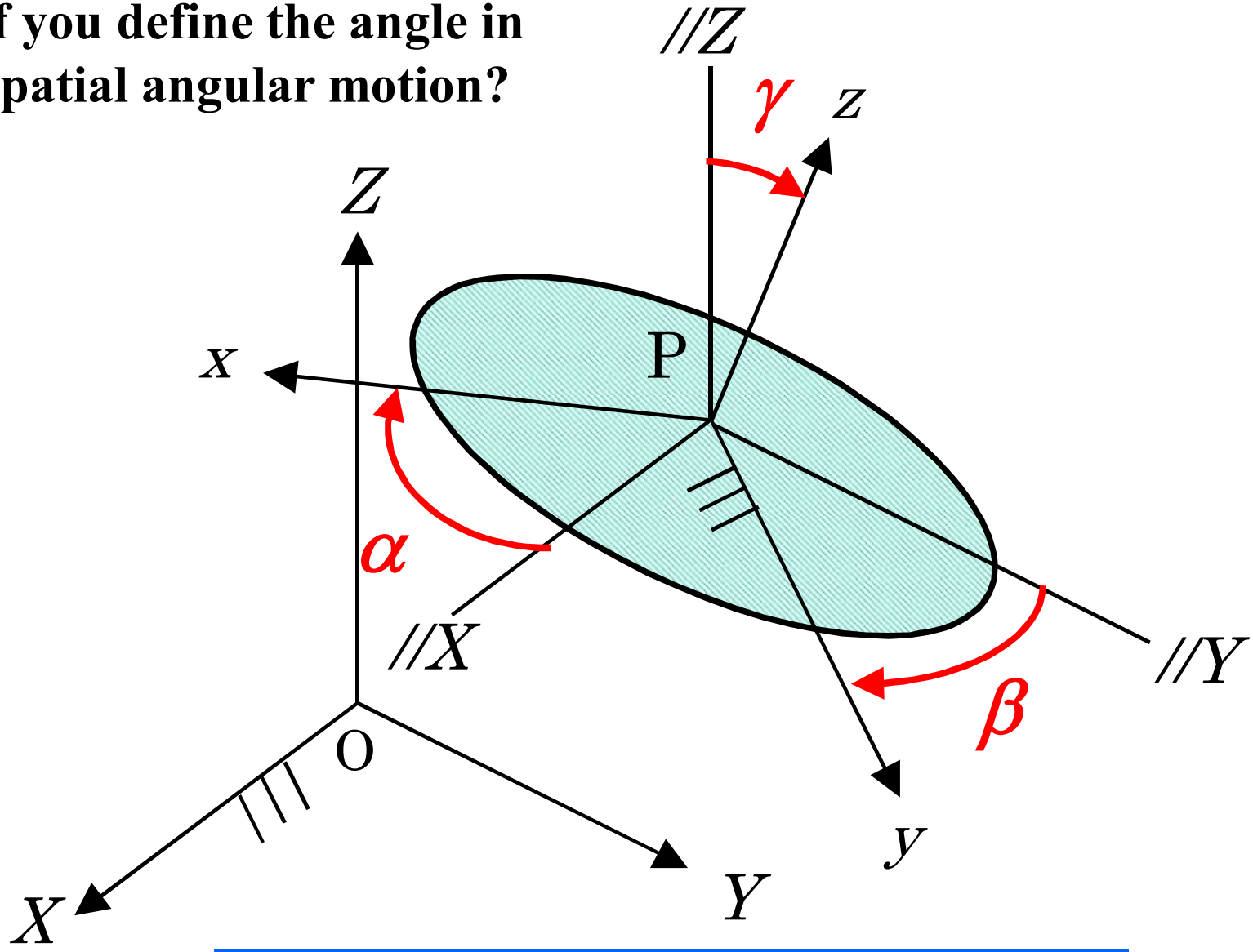
**Correct expression**




**A rigid body**

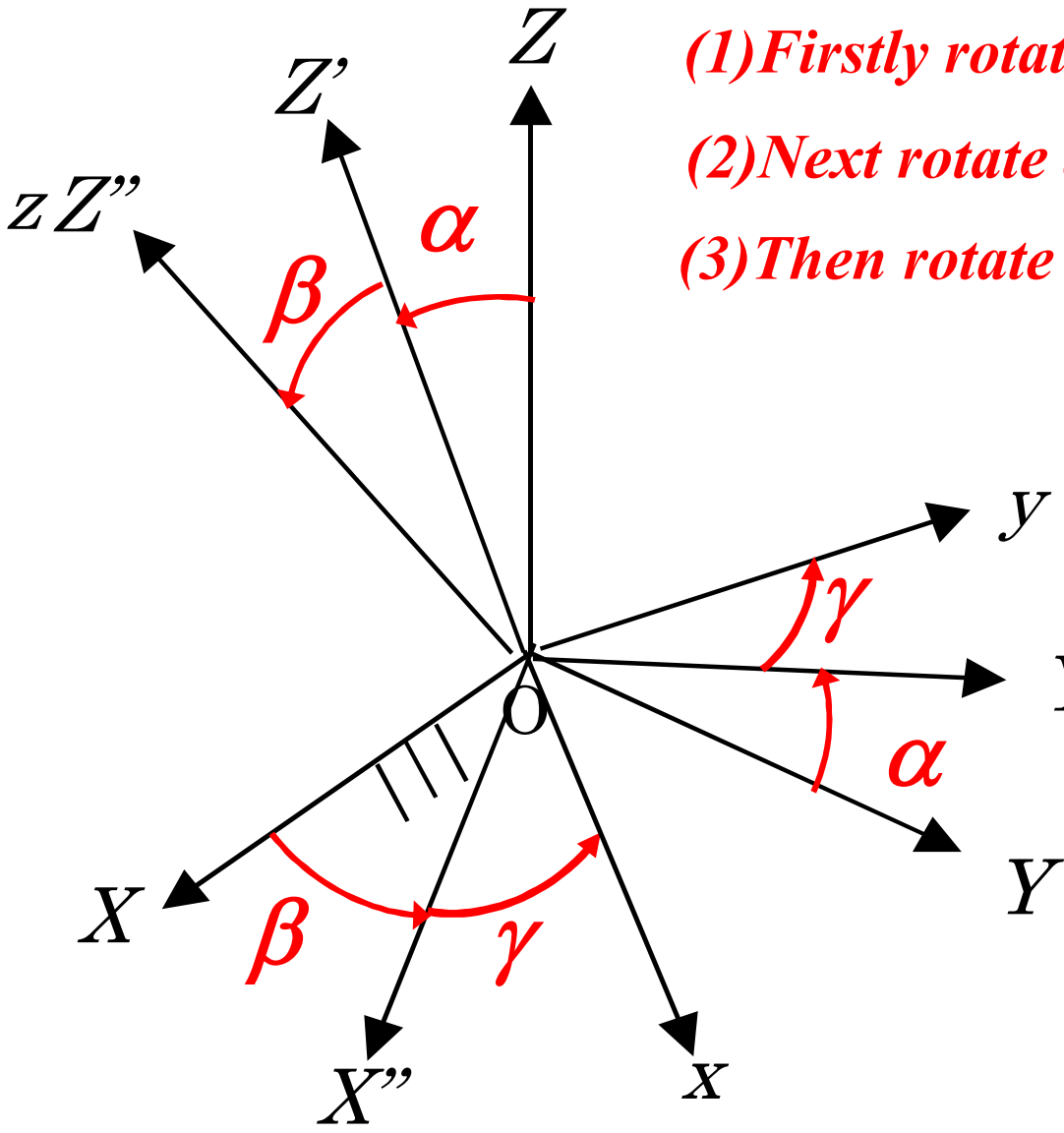
**This angle is defined as an angle between axes of moving coordinate system fixed on The rigid body and of fixed coordinate system**

**If you define the angle in Spatial angular motion?**




***It is possible to define posture angles. However it is unavailable!***

Therefore we use the angles to rotate coordinate axis
（Roll-,Pitch-,Yaw-angles）



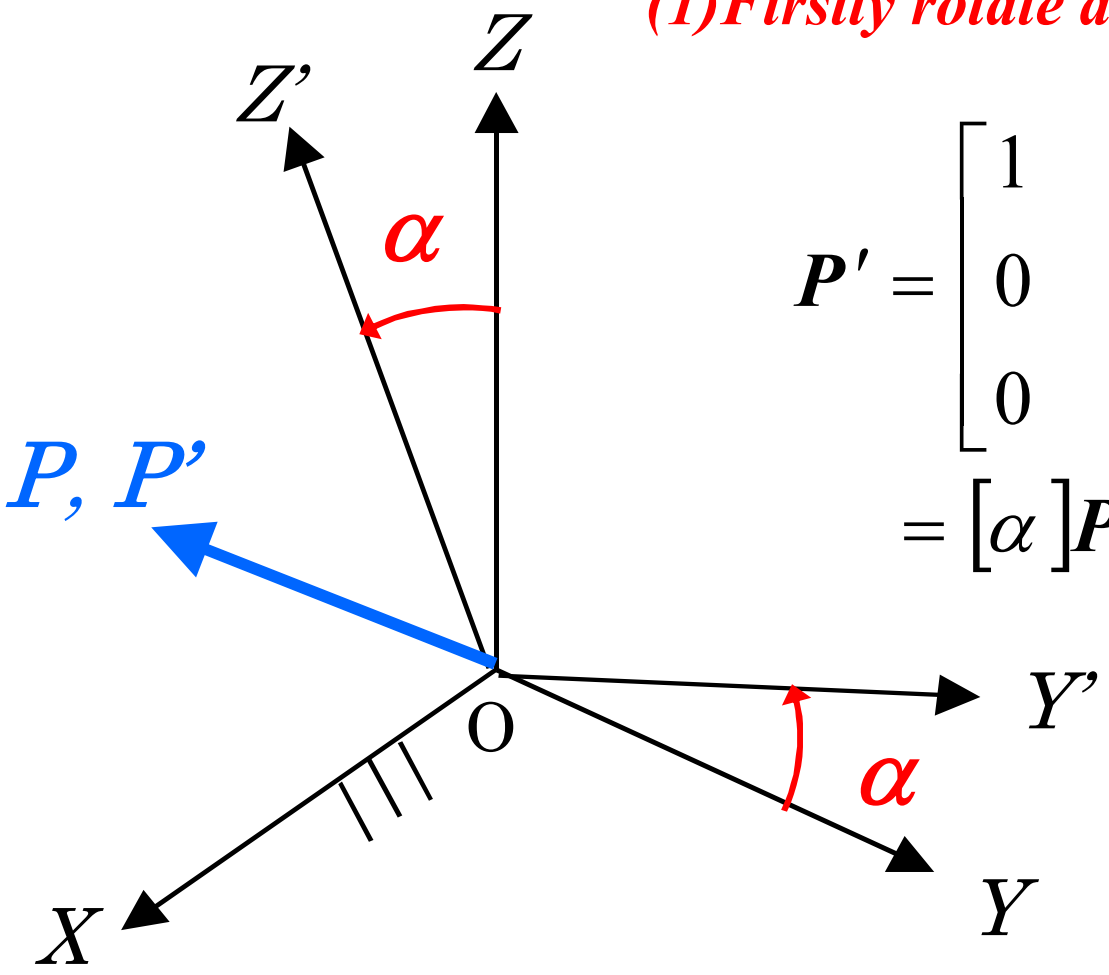

*(1)Firstly rotate about X-axis with angle α*

*(2)Next rotate about Y'-axis with angle β*

*(3)Then rotate about Z"-axis with angle γ*

***Directions of all axes correspond to those of moving coordinate system.***

**Let focus on a position vector on two coordinate systems**



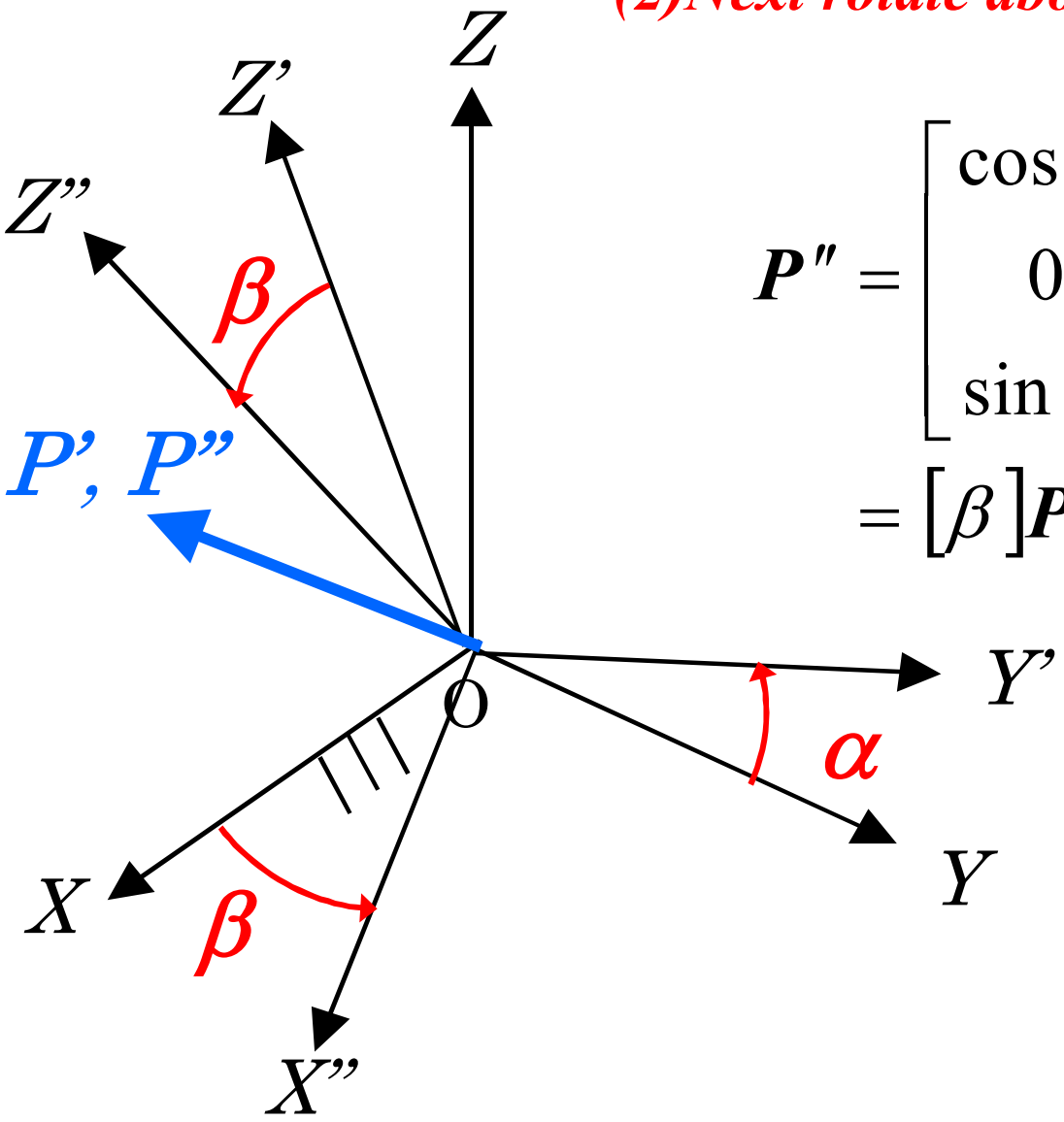

***(1)Firstly rotate about X-axis with angle α***

$$\boldsymbol{P}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \boldsymbol{P}$$

$$= [\alpha]\boldsymbol{P}$$

**Let focus on a position vector on two coordinate systems**

*(2)Next rotate about Y'-axis with angle $\beta$*

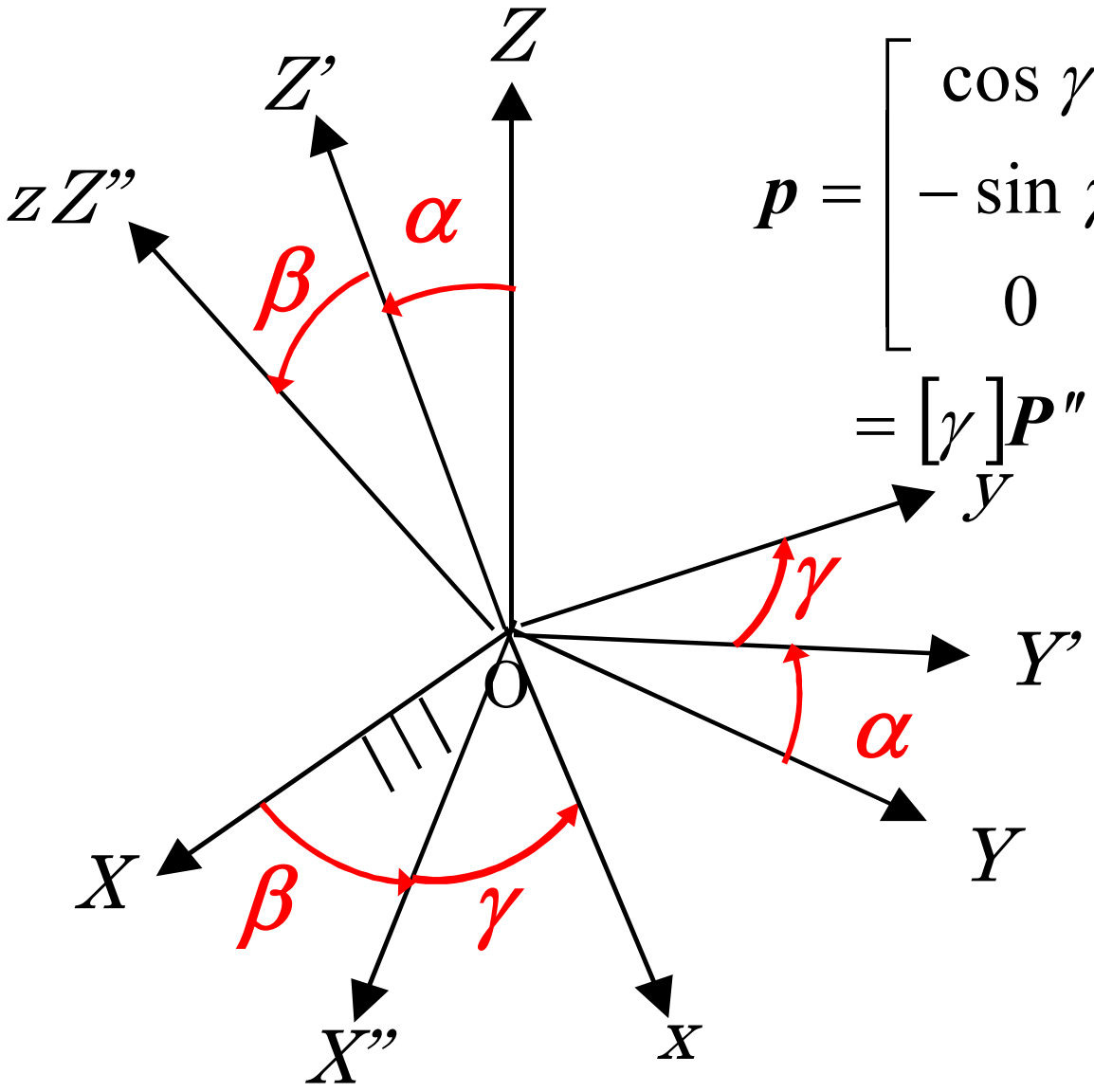



$$\boldsymbol{P}'' = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \boldsymbol{P}'$$

$$= [\beta]\boldsymbol{P}'$$

**Let focus on a position vector on two coordinate systems**

***(3)Then rotate about Z"-axis with angle γ***




$$\boldsymbol{p} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{P}''$$

$$= [\gamma]\boldsymbol{P}''$$

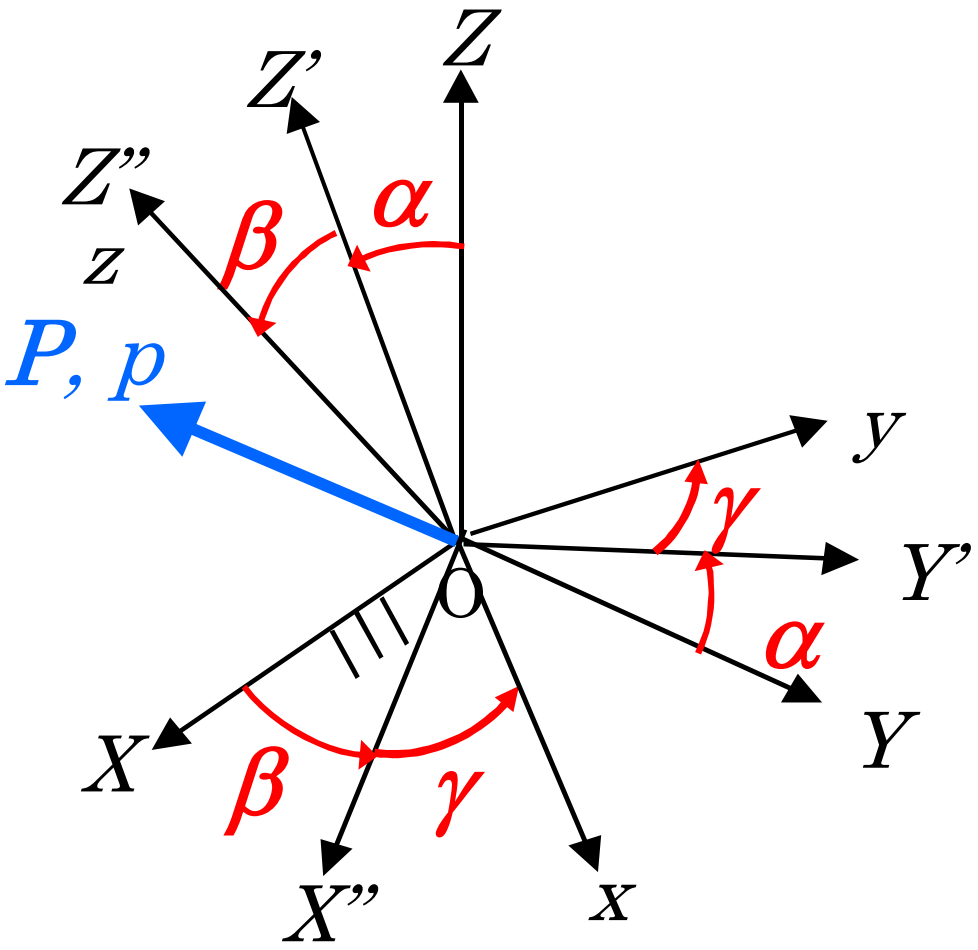
**Resultantly**

$$\boldsymbol{p} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \boldsymbol{P}$$

$$= [\gamma][\beta][\alpha]\boldsymbol{P}$$

$$= [T(\alpha, \beta, \gamma)]\boldsymbol{P}$$

***Coordinate transformation matrix***



**Inverse transformation**

$$\boldsymbol{P} = [T(\alpha,\beta,\gamma)]^{-1}\,\boldsymbol{p}$$

$$= [\alpha]^{-1}[\beta]^{-1}[\gamma]^{-1}\,\boldsymbol{p}$$

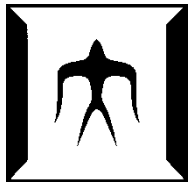
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix}^{-1} \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}^{-1} \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \boldsymbol{p}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{p}$$

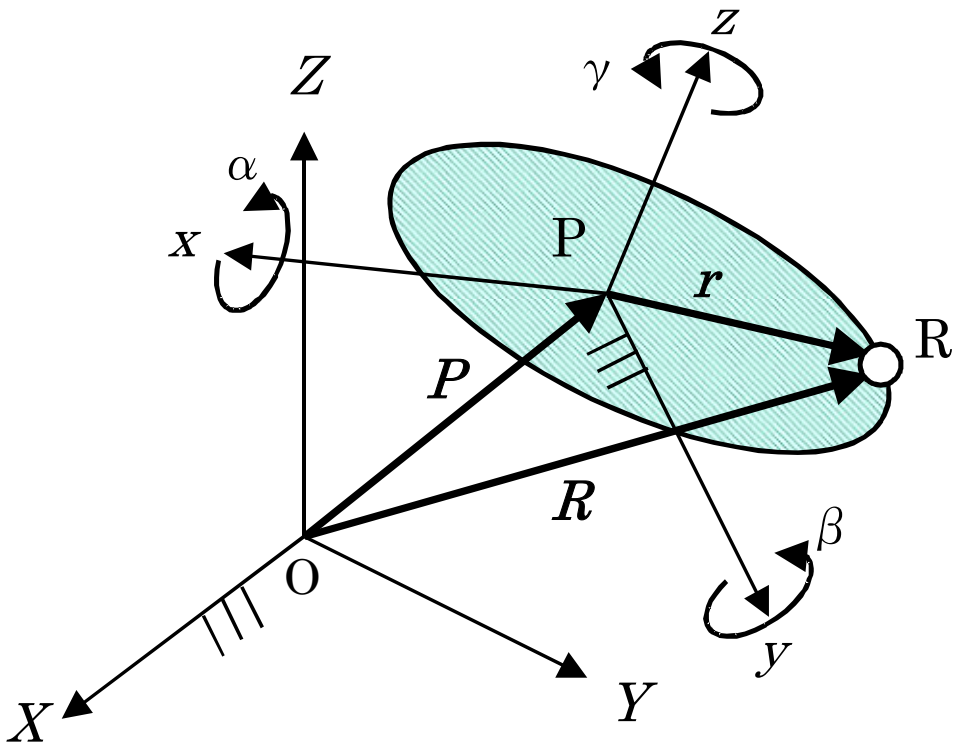
$$= [\alpha]^{T}[\beta]^{T}[\gamma]^{T}\,\boldsymbol{p}$$

$$= [T(\alpha,\beta,\gamma)]^{T}\,\boldsymbol{p}$$

***Inverse matrix can be given as transposed matrix.***



**Coordinate transformation from moving system on the rigid body to fixed system can be expressed as**




$$\boldsymbol{R} = [T]^T \boldsymbol{r} + \boldsymbol{P}$$

$$\dot{\boldsymbol{R}} = [\dot{T}]^T \boldsymbol{r} + \dot{\boldsymbol{P}}$$

$$\ddot{\boldsymbol{R}} = [\ddot{T}]^T \boldsymbol{r} + \ddot{\boldsymbol{P}}$$

***Spatial angular motion can be expressed with the coordinate transformation matrix.***

Differentiation of transformation matrix:

$$[\dot{T}]^T = [\dot{\alpha}]^T[\beta]^T[\gamma]^T + [\alpha]^T[\dot{\beta}]^T[\gamma]^T + [\alpha]^T[\beta]^T[\dot{\gamma}]^T$$

$$[\ddot{T}]^T = [\ddot{\alpha}]^T[\beta]^T[\gamma]^T + [\alpha]^T[\ddot{\beta}]^T[\gamma]^T + [\alpha]^T[\beta]^T[\ddot{\gamma}]^T + 2[\dot{\alpha}]^T[\dot{\beta}]^T[\gamma]^T + 2[\alpha]^T[\dot{\beta}]^T[\dot{\gamma}]^T + 2[\dot{\alpha}]^T[\beta]^T[\dot{\gamma}]^T$$

$$[\dot{\alpha}]^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\dot{\alpha}\sin\alpha & -\dot{\alpha}\cos\alpha \\ 0 & \dot{\alpha}\cos\alpha & -\dot{\alpha}\sin\alpha \end{bmatrix}$$

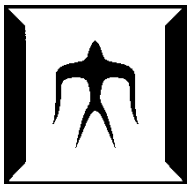
$$[\ddot{\alpha}]^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\ddot{\alpha}\sin\alpha - \dot{\alpha}^2\cos\alpha & -\ddot{\alpha}\cos\alpha + \dot{\alpha}^2\sin\alpha \\ 0 & \ddot{\alpha}\cos\alpha - \dot{\alpha}^2\sin\alpha & -\ddot{\alpha}\sin\alpha - \dot{\alpha}^2\cos\alpha \end{bmatrix}$$

$$[\dot{\beta}]^T = \begin{bmatrix} -\dot{\beta}\sin\beta & 0 & \dot{\beta}\cos\beta \\ 0 & 0 & 0 \\ -\dot{\beta}\cos\beta & 0 & -\dot{\beta}\sin\beta \end{bmatrix}$$

$$[\ddot{\beta}]^T = \begin{bmatrix} -\ddot{\beta}\sin\beta - \dot{\beta}^2\cos\beta & 0 & \ddot{\beta}\cos\beta - \dot{\beta}^2\sin\beta \\ 0 & 0 & 0 \\ -\ddot{\beta}\cos\beta + \dot{\beta}^2\sin\beta & 0 & -\ddot{\beta}\sin\beta - \dot{\beta}^2\cos\beta \end{bmatrix}$$

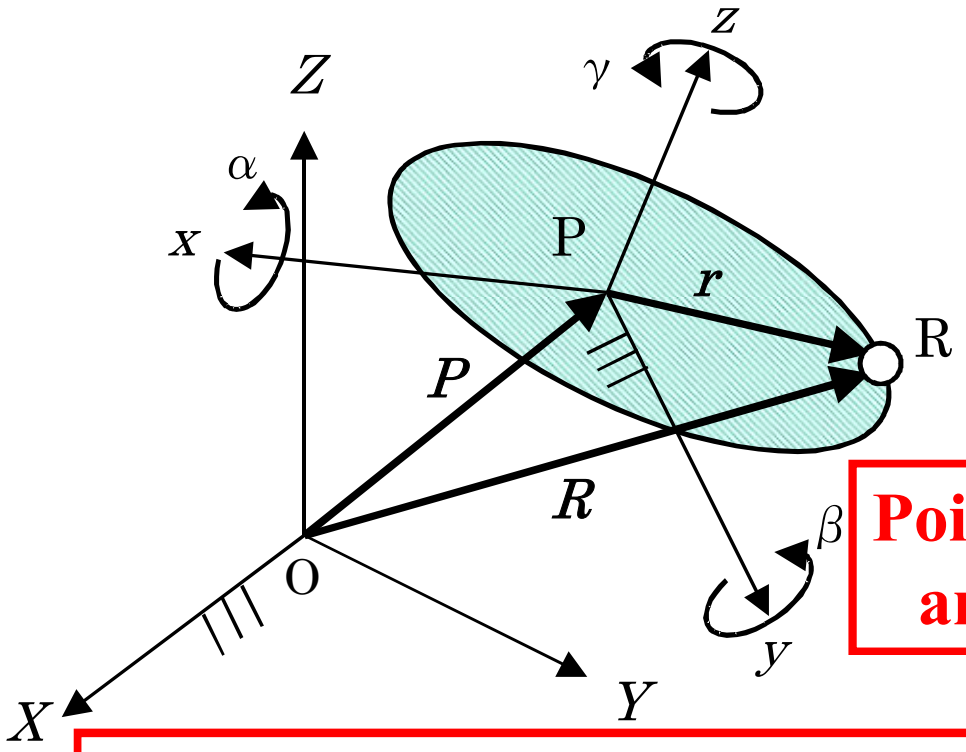
$$[\dot{\gamma}]^T = \begin{bmatrix} -\dot{\gamma}\sin\gamma & -\dot{\gamma}\cos\gamma & 0 \\ \dot{\gamma}\cos\gamma & -\dot{\gamma}\sin\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$[\ddot{\gamma}]^T = \begin{bmatrix} -\ddot{\gamma}\sin\gamma - \dot{\gamma}^2\cos\gamma & -\ddot{\gamma}\cos\gamma + \dot{\gamma}^2\sin\gamma & 0 \\ \ddot{\gamma}\cos\gamma - \gamma^2\sin\gamma & -\ddot{\gamma}\sin\gamma - \dot{\gamma}^2\cos\gamma & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



# 3. Systematic kinematic analysis of spatial link mechanisms

**(1) Coordinate transformation matrix and motion of coupler point**




$$\boldsymbol{R} = [T]^T \boldsymbol{r} + \boldsymbol{P}$$

$$\dot{\boldsymbol{R}} = [\dot{T}]^T \boldsymbol{r} + \dot{\boldsymbol{P}}$$

$$\ddot{\boldsymbol{R}} = [\ddot{T}]^T \boldsymbol{r} + \ddot{\boldsymbol{P}}$$

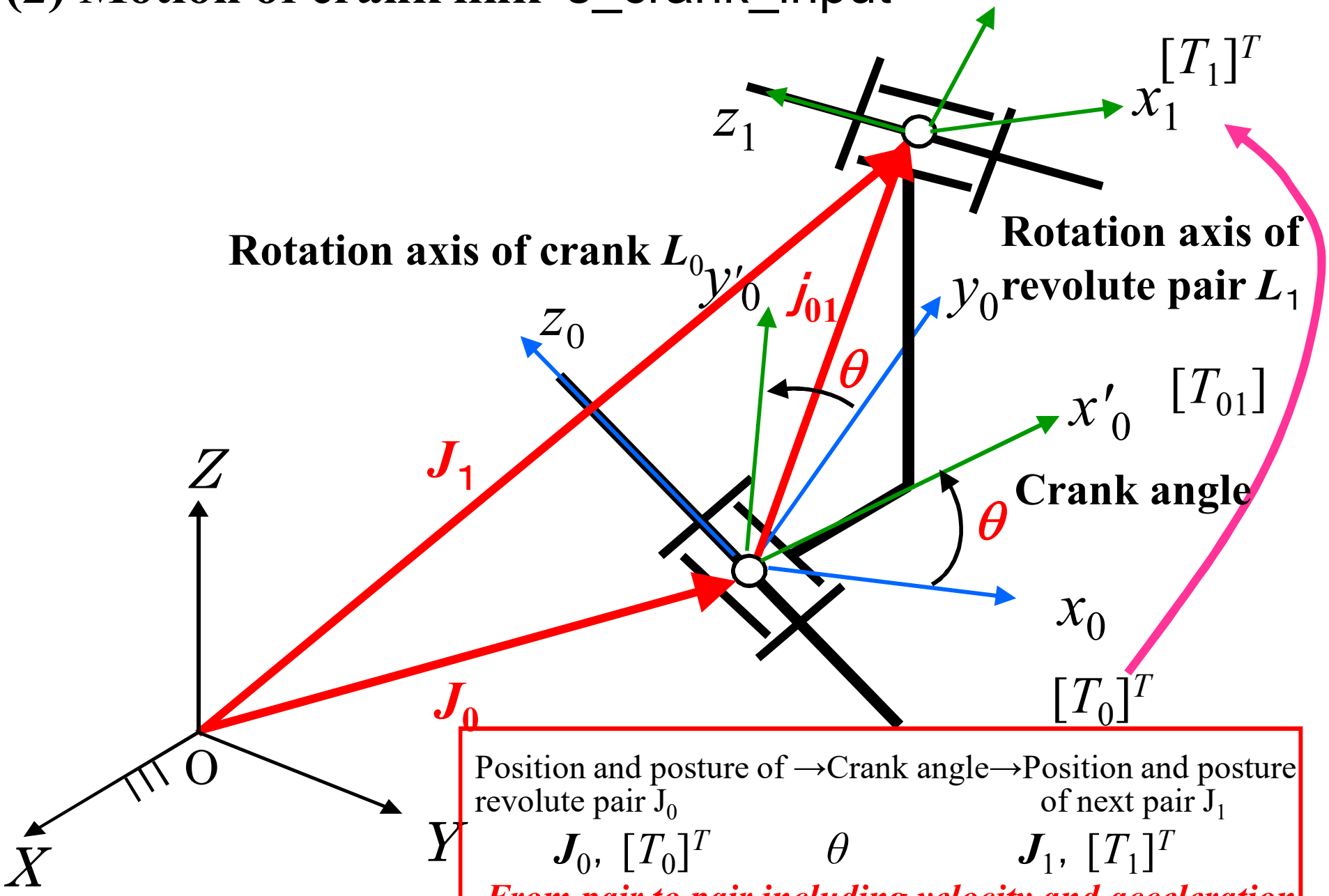
**Point on a spatially moving link: analysis program: Trans**

**Coordinate transformation matrix and posture angles**

$$\alpha,\ \beta,\ \gamma \rightleftharpoons [T]^T$$

**analysis program: RPYTT,TTRPY**

## (2) Motion of crank link s_crank_input




Position and posture of revolute pair $J_0$ → Crank angle → Position and posture of next pair $J_1$

$J_0,\ [T_0]^T$ $\quad\theta\quad$ $J_1,\ [T_1]^T$

***From pair to pair including velocity and acceleration***

**Equations to calculate:**

$$\boldsymbol{J}_1 = \boldsymbol{J}_0 + \left[T_0\right]^T \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{j}_{01} = \boldsymbol{J}_0 + \left[T_0\right]^T \left[\theta\right]^T \boldsymbol{j}_{01}$$

$$\dot{\boldsymbol{J}}_1 = \dot{\boldsymbol{J}}_0 + (\left[\dot{T}_0\right]^T \left[\theta\right]^T + \left[T_0\right]^T \left[\dot{\theta}\right]^T)\, \boldsymbol{j}_{01}$$

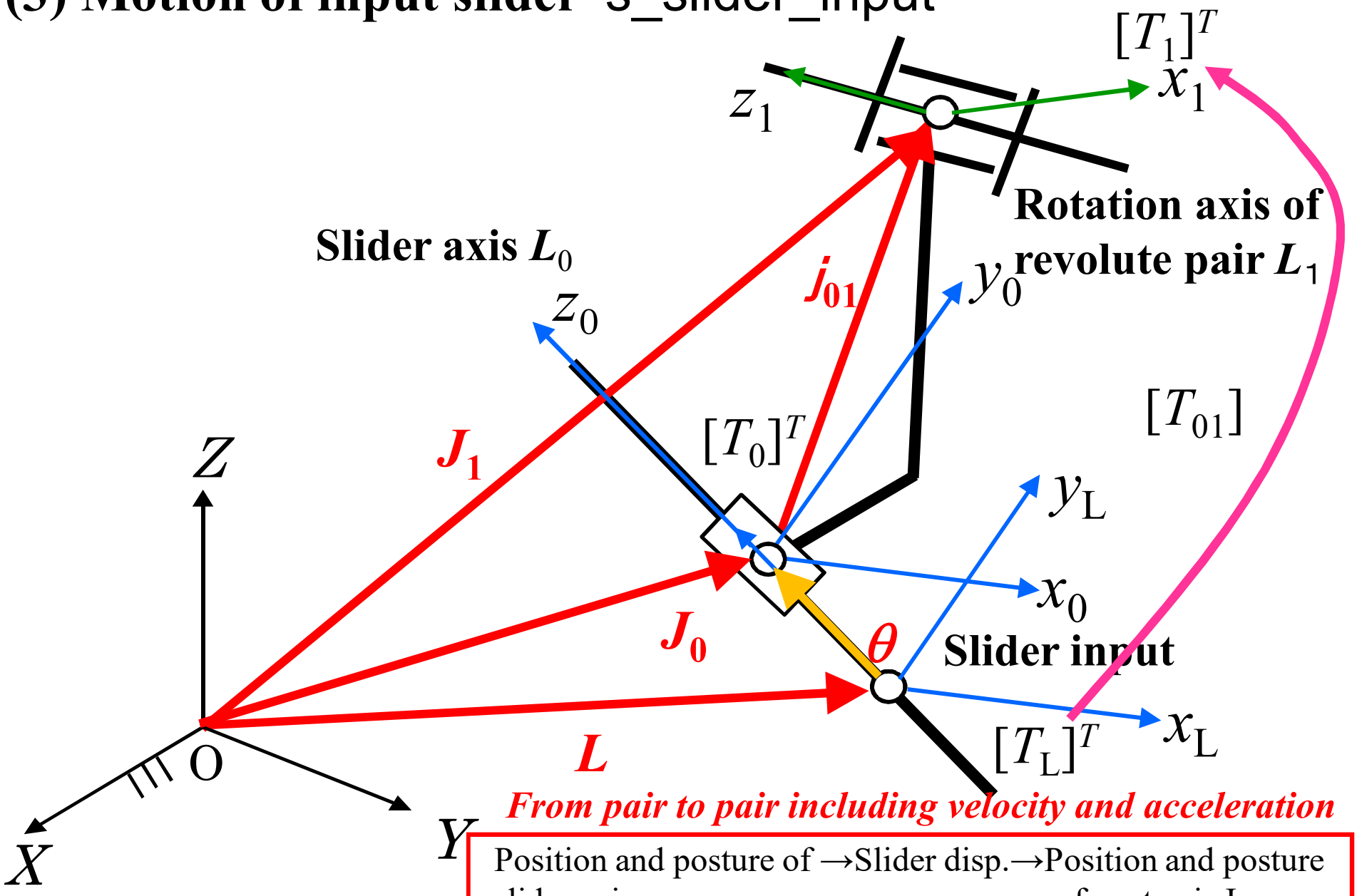
$$\ddot{\boldsymbol{J}}_1 = \ddot{\boldsymbol{J}}_0 + (\left[\ddot{T}_0\right]^T \left[\theta\right]^T + 2\left[\dot{T}_0\right]^T \left[\dot{\theta}\right]^T + \left[T_0\right]^T \left[\ddot{\theta}\right]^T)\, \boldsymbol{j}_{01}$$

$$\left[T_1\right]^T = \left[T_0\right]^T \left[\theta\right]^T \left[T_{01}\right]^T$$

$$\left[\dot{T}_1\right]^T = \left[\dot{T}_0\right]^T \left[\theta\right]^T \left[T_{01}\right]^T + \left[T_0\right]^T \left[\dot{\theta}\right]^T \left[T_{01}\right]^T$$

$$\left[\ddot{T}_1\right]^T = \left[\ddot{T}_0\right]^T \left[\theta\right]^T \left[T_{01}\right]^T + 2\left[\dot{T}_0\right]^T \left[\dot{\theta}\right]^T \left[T_{01}\right]^T + \left[T_0\right]^T \left[\ddot{\theta}\right]^T \left[T_{01}\right]^T$$

# (3) Motion of input slider s_slider_input




***From pair to pair including velocity and acceleration***

| Position and posture of slider axis | → | Slider disp. | → | Position and posture of next pair $J_1$ |
|---|---|---|---|---|
| $L$, $[T_L]^T$ | | $\theta$ | | $J_1$, $[T_1]^T$ |

**Equations to calculate:**

$$\boldsymbol{J}_1 = \boldsymbol{L} + [T_L]^T \left( [0 \quad 0 \quad \theta]^T + \boldsymbol{j}_{01} \right)$$

$$\dot{\boldsymbol{J}}_1 = \dot{\boldsymbol{L}} + [\dot{T}_L]^T \left( [0 \quad 0 \quad \theta]^T + \boldsymbol{j}_{01} \right) + [T_L]^T [0 \quad 0 \quad \theta]^T$$

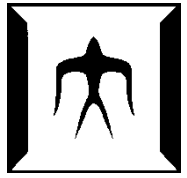
$$\ddot{\boldsymbol{J}}_1 = \ddot{\boldsymbol{L}} + [\ddot{T}_L]^T \left( [0 \quad 0 \quad \theta]^T + \boldsymbol{j}_{01} \right) + 2[\dot{T}_L]^T [0 \quad 0 \quad \dot{\theta}]^T + [T_L]^T [0 \quad 0 \quad \ddot{\theta}]^T$$

$$[T_1]^T = [T_L]^T [T_{01}]^T$$

$$[\dot{T}_1]^T = [\dot{T}_L]^T [T_{01}]^T$$

$$[\ddot{T}_1]^T = [\ddot{T}_L]^T [T_{01}]^T$$

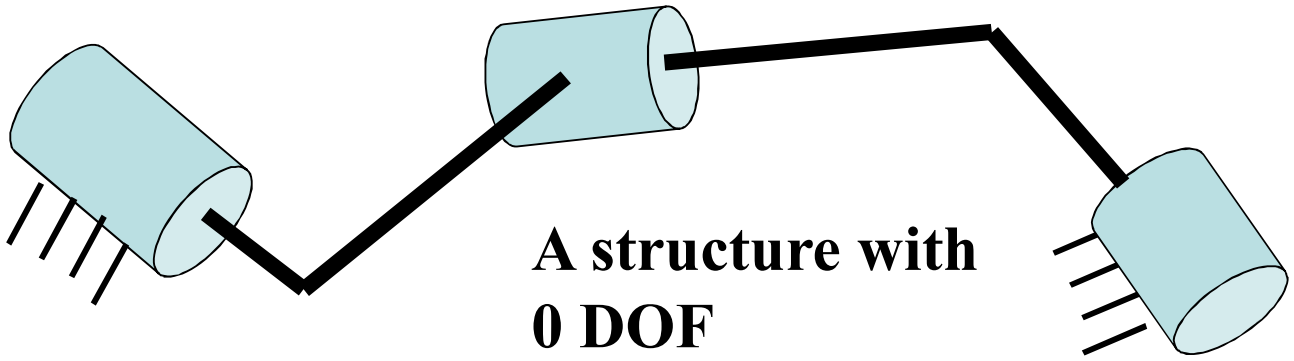
$$\because [T_1] = [T_L]$$



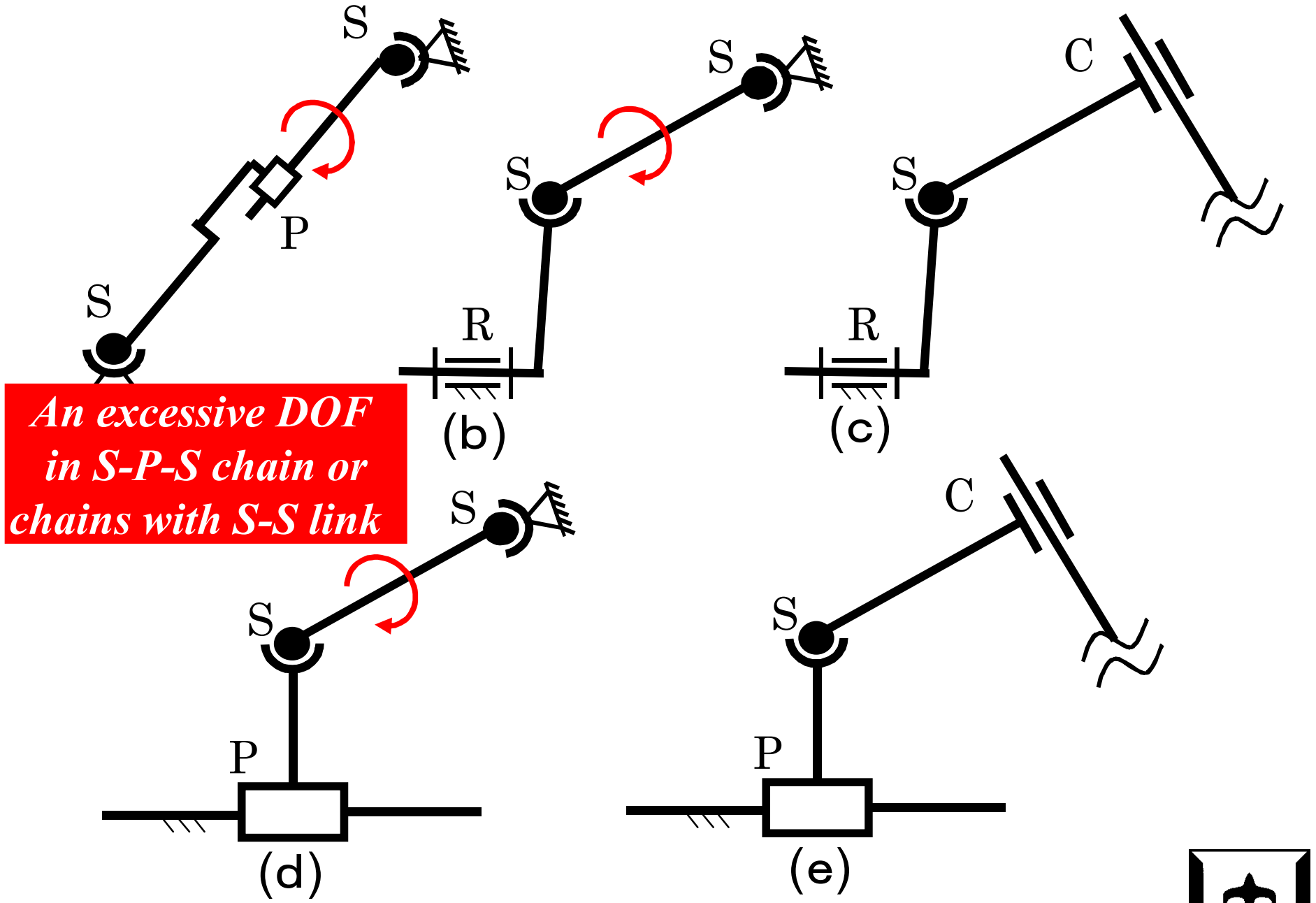
**Next let's consider motion of spatial two adjacent links**

**Principle →** **Two adjacent links in which position and posture of all of links can be determined when position and posture of pairs at both ends**

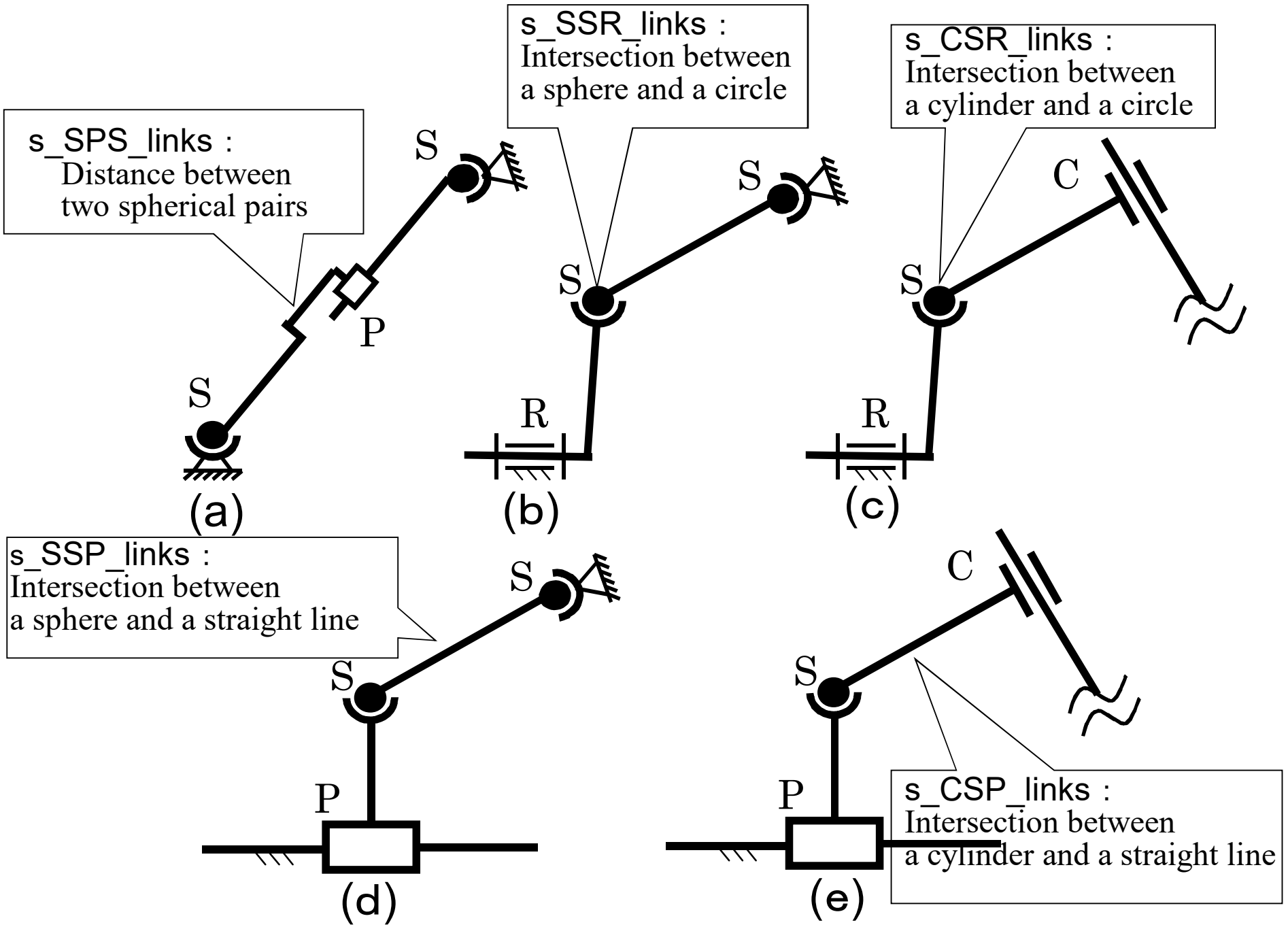
**→ It means the link chain which becomes a structure when pairs at both ends become as fixed pairs.**



**A structure with 0 DOF**



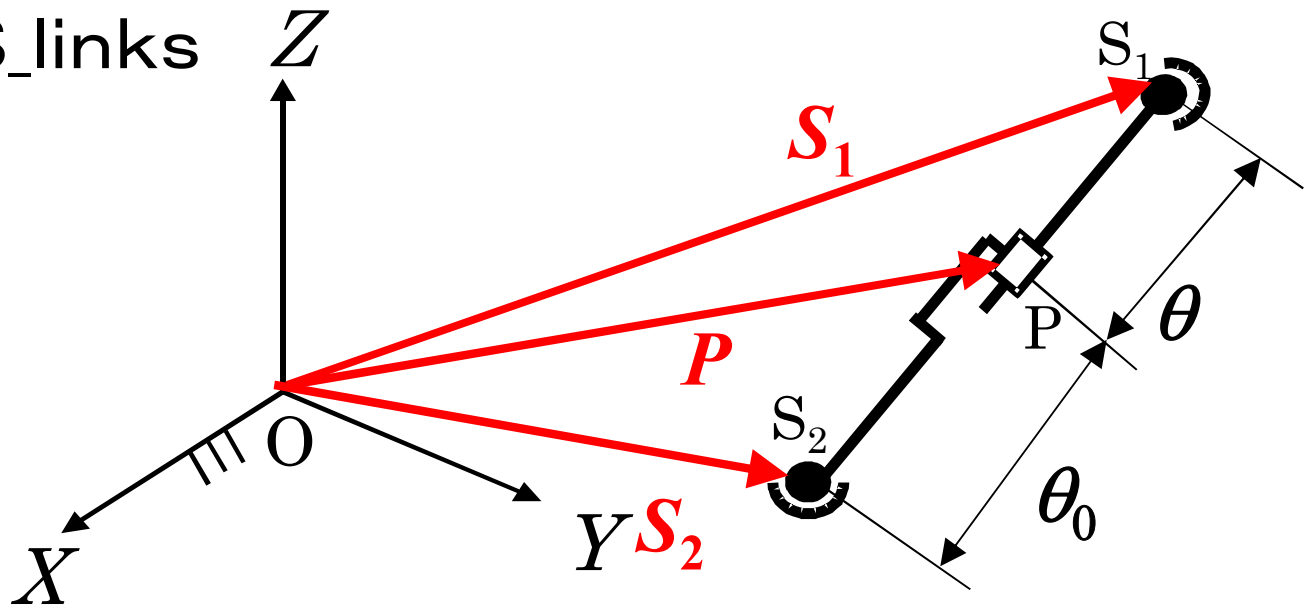

Two adjacent links in spatial mechanisms




Kinematic analysis program for two adjacent links

## Several examples:

(4)SPS links s_SPS_links



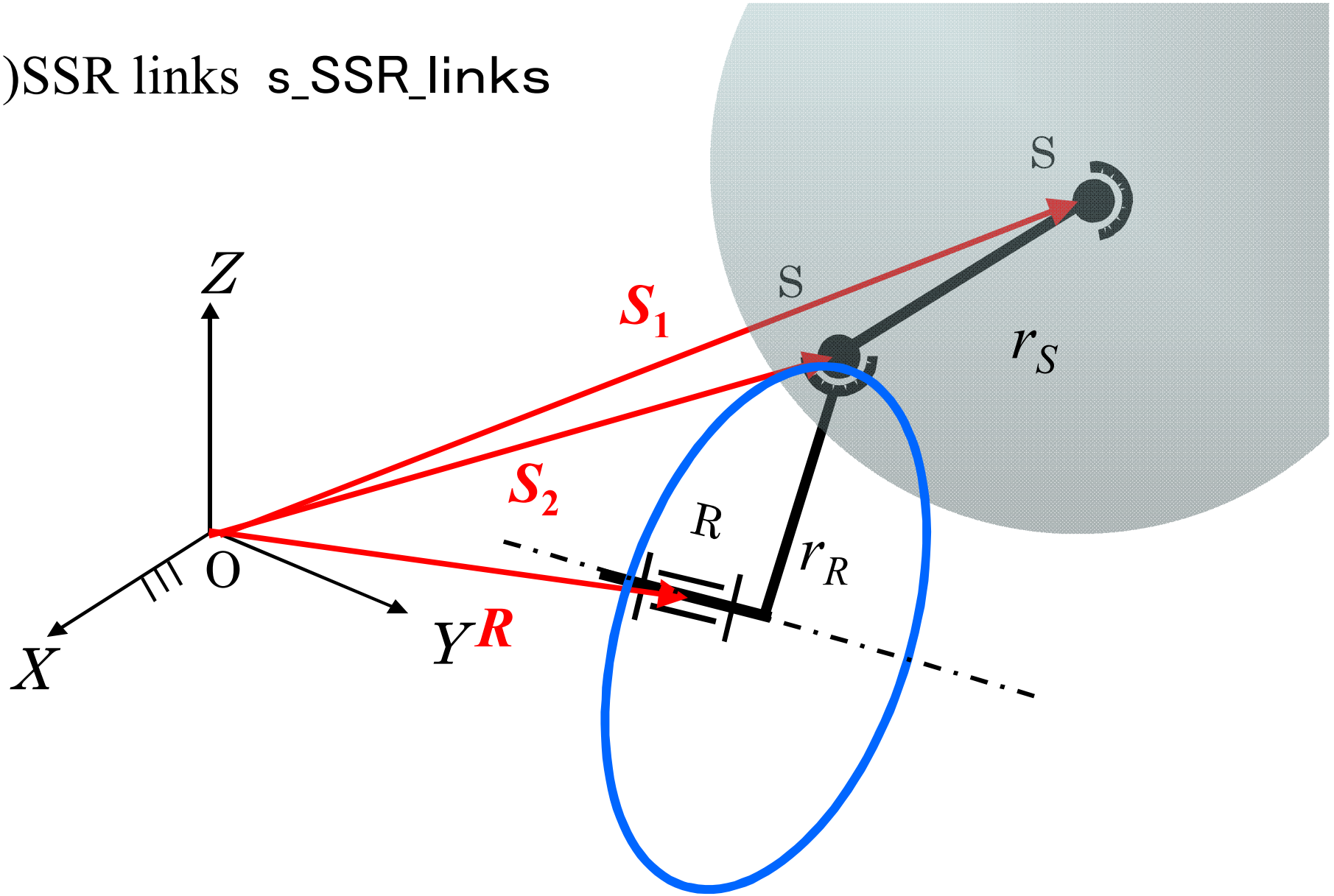

$$\theta+\theta_0=\left|\boldsymbol{S}_2-\boldsymbol{S}_1\right|=\sqrt{\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)}$$

$$\theta=\sqrt{\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)}-\theta_0$$

$$\dot{\theta}=\frac{\left(\dot{\boldsymbol{S}}_2-\dot{\boldsymbol{S}}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)}{\sqrt{\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)}}=\frac{\left(\dot{\boldsymbol{S}}_2-\dot{\boldsymbol{S}}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)}{\theta+\theta_0}$$

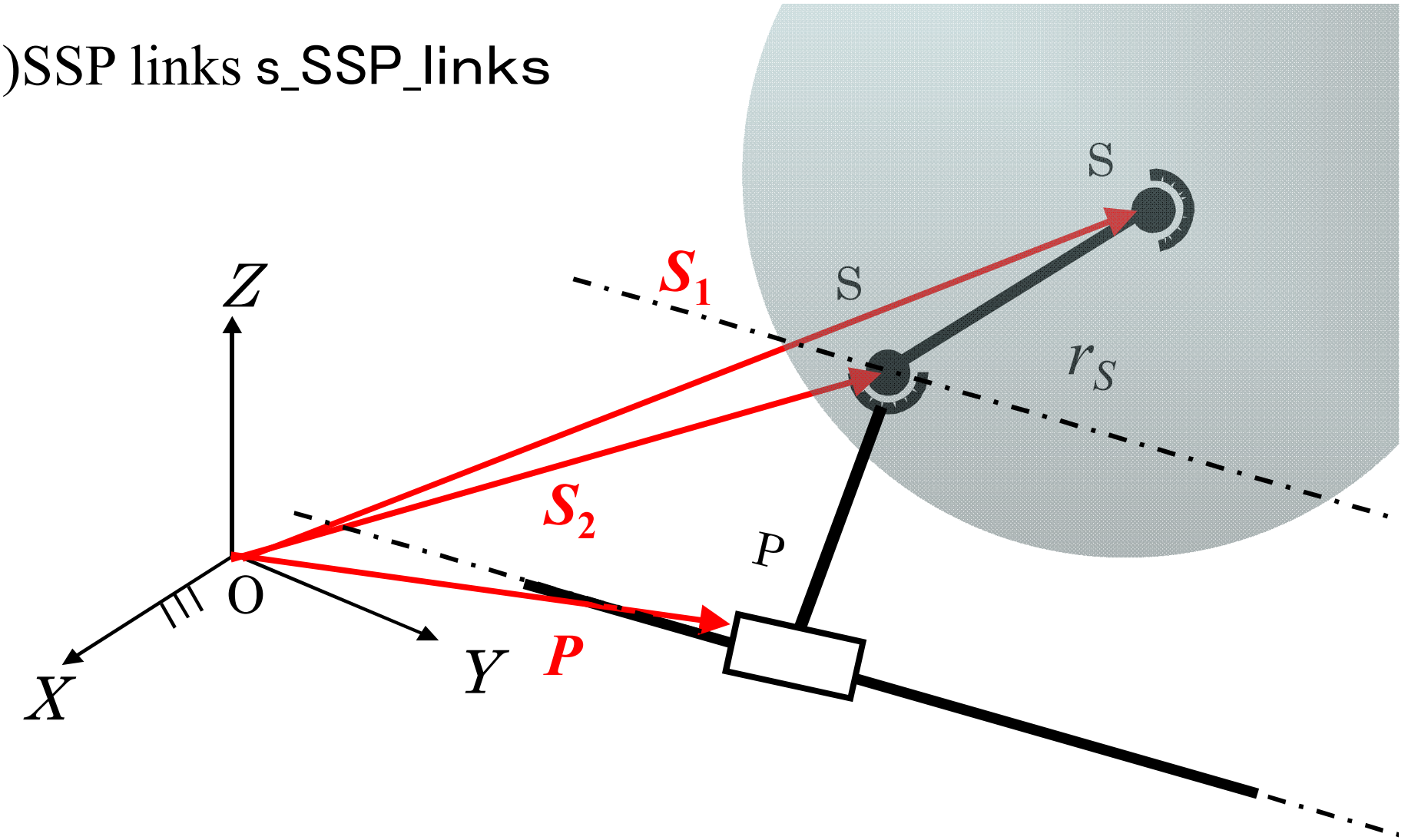
$$\ddot{\theta}=\frac{\left[\left(\ddot{\boldsymbol{S}}_2-\ddot{\boldsymbol{S}}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)+\left(\dot{\boldsymbol{S}}_2-\dot{\boldsymbol{S}}_1\right)\bullet\left(\dot{\boldsymbol{S}}_2-\dot{\boldsymbol{S}}_1\right)\right]\left(\theta+\theta_0\right)-\left(\dot{\boldsymbol{S}}_2-\dot{\boldsymbol{S}}_1\right)\bullet\left(\boldsymbol{S}_2-\boldsymbol{S}_1\right)\dot{\theta}}{\left(\theta+\theta_0\right)^2}$$

(5)SSR links s_SSR_links




**Position of pair $S_2$ can be calculated as an Intersection between a sphere with radius $r_s$ and a circle with radius $r_R$**

(6)SSP links s_SSP_links




**Position of pair $S_2$ can be determined as an intersection between a sphere with radius $r_s$ and a straight line parallel to the slider axis**

```
//-------------------------------------------------------------
//
//    ロール・ピッチ・ヨー角から座標変換行列を求める
//
//   RPY_ANGLES
//          : ロール・ピッチ・ヨー角とその微分
//   TT[3][3], dTT[3][3], ddTT[3][3]
//          : 座標変換行列とその微分
//
//-------------------------------------------------------------

void RPYTT ( POSTURE RPY_ANGLES, double TT[3][3],
      double dTT[3][3], double ddTT[3][3] )
{
                  double CR, SR, CP, SP, CY, SY;
                  int i, j;
                  double WA1[3][3], WA2[3][3], WA3[3][3],
                                    WB1[3][3], WB2[3][3], WB3[3][3],
                                    WC1[3][3], WC2[3][3], WC3[3][3];

                  CR = cos( RPY_ANGLES.RPY.x );
                  SR = sin( RPY_ANGLES.RPY.x );
                  CP = cos( RPY_ANGLES.RPY.y );
                  SP = sin( RPY_ANGLES.RPY.y );
                  CY = cos( RPY_ANGLES.RPY.z );
                  SY = sin( RPY_ANGLES.RPY.z );

                  WA1[0][0] = 0.0;
                  WA1[1][0] = -SR*SY + CR*SP*CY;
                  WA1[2][0] = CR*SY + SR*SP*CY;
                  WA1[0][1] = 0.0;
                  WA1[1][1] = -SR*CY - CR*SP*SY;
                  WA1[2][1] = CR*CY - SR*SP*SY;
                  WA1[0][2] = 0.0;
                  WA1[1][2] = -CR*CP;
                  WA1[2][2] = -SR*CP;

                  WA2[0][0] = -SP*CY;
                  WA2[1][0] =  SR*CP*CY;
                  WA2[2][0] = -CR*CP*CY;
                  WA2[0][1] =  SP*SY;
                  WA2[1][1] = -SR*CP*SY;
                  WA2[2][1] =  CR*CP*SY;
                  WA2[0][2] =  CP;
                  WA2[1][2] =  SR*SP;
                  WA2[2][2] = -CR*SP;

                  WA3[0][0] = -CP*SY;
                  WA3[1][0] =  CR*CY - SR*SP*SY;
                  WA3[2][0] =  SR*CY + CR*SP*SY;
                  WA3[0][1] = -CP*CY;
                  WA3[1][1] = -CR*SY - SR*SP*CY;
                  WA3[2][1] = -SR*SY + CR*SP*CY;
                  WA3[0][2] =  0.0;
                  WA3[1][2] =  0.0;
                  WA3[2][2] =  0.0;

                  WB1[0][0] =  0.0;
                  WB1[1][0] = -CR*SY - SR*SP*CY;
                  WB1[2][0] = -SR*SY + CR*SP*CY;
                  WB1[0][1] =  0.0;
                  WB1[1][1] = -CR*CY + SR*SP*SY;
                  WB1[2][1] = -SR*CY - CR*SP*SY;
                  WB1[0][2] =  0.0;
                  WB1[1][2] =  SR*CP;
                  WB1[2][2] = -CR*CP;

                  WB2[0][0] = -CP*CY;
                  WB2[1][0] = -SR*SP*CY;
                  WB2[2][0] =  CR*SP*CY;
                  WB2[0][1] =  CP*SY;
                  WB2[1][1] =  SR*SP*SY;
                  WB2[2][1] = -CR*SP*SY;
                  WB2[0][2] =  -SP;
                  WB2[1][2] =  SR*CP;
                  WB2[2][2] = -CR*CP;

                  WB3[0][0] = -CP*CY;
                  WB3[1][0] = -CR*SY - SR*SP*CY;
                  WB3[2][0] = -SR*SY + CR*SP*CY;
                  WB3[0][1] =  CP*SY;
                  WB3[1][1] = -CR*CY + SR*SP*SY;
                  WB3[2][1] = -SR*CY - CR*SP*SY;
                  WB3[0][2] = 0.0;
                  WB3[1][2] = 0.0;
                  WB3[2][2] = 0.0;

                  WC1[0][0] = 0.0;
                  WC1[1][0] =  CR*CP*CY;
                  WC1[2][0] =  SR*CP*CY;
                  WC1[0][1] = 0.0;
                  WC1[1][1] = -CR*CP*SY;
                  WC1[2][1] = -SR*CP*SY;
```

C language program of systematic kinematics analysis

```
WC2[0][0] =  SP*SY;
WC2[1][0] = -SR*CP*SY;
WC2[2][0] =  CR*CP*SY;
WC2[0][1] =  SP*CY;
WC2[1][1] = -SR*CP*CY;
WC2[2][1] =  CR*CP*CY;
WC2[0][2] = 0.0;
WC2[1][2] = 0.0;
WC2[2][2] = 0.0;

WC3[0][0] = 0.0;
WC3[1][0] = -SR*CY - CR*SP*SY;
WC3[2][0] =  CR*CY - SR*SP*SY;
WC3[0][1] = 0.0;
WC3[1][1] =  SR*SY - CR*SP*CY;
WC3[2][1] = -CR*SY - SR*SP*CY;
WC3[0][2] = 0.0;
WC3[1][2] = 0.0;
WC3[2][2] = 0.0;

TT[0][0] =  CP*CY;
TT[1][0] =  CR*SY + SR*SP*CY;
TT[2][0] =  SR*SY - CR*SP*CY;
TT[0][1] = -CP*SY;
TT[1][1] =  CR*CY - SR*SP*SY;
TT[2][1] =  SR*CY + CR*SP*SY;
TT[0][2] =  SP;
TT[1][2] = -SR*CP;
TT[2][2] =  CR*CP;
for(i = 0; i < 3; i++){
        or(j = 0; j < 3; j++){
              dTT[i][j]  = WA1[i][j]*RPY_ANGLES.DRPY.x
                         + WA2[i][j]*RPY_ANGLES.DRPY.y
                         + WA3[i][j]*RPY_ANGLES.DRPY.z;
              dTT[i][j] = WA1[i][j]*RPY_ANGLES.DDRPY.x
                        + WA2[i][j]*RPY_ANGLES.DDRPY.y
                        + WA3[i][j]*RPY_ANGLES.DDRPY.z
                        +  WB1[i][j]*RPY_ANGLES.DRPY.x*
                                RPY_ANGLES.DRPY.x
                        + WB2[i][j]*RPY_ANGLES.DRPY.y*
                                RPY_ANGLES.DRPY.y
                        + WB3[i][j]*RPY_ANGLES.DRPY.z*
                                RPY_ANGLES.DRPY.z
                       + 2.0*WC1[i][j]*RPY_ANGLES.DRPY.x*
                                RPY_ANGLES.DRPY.y
                      +  2.0*WC2[i][j]*RPY_ANGLES.DRPY.y*
                                RPY_ANGLES.DRPY.z
                                        + 2.0*WC3[i][j]*RPY_ANGLES.DRPY.z*
                                              RPY_ANGLES.DRPY.x;
        }
     }
}


//-----------------------------------------------------
//
//    座標変換行列に基づく3次元座標変換
//
//   PM       : 動座標系上の位置ベクトル
//   P0       : 動座標系原点の静止系上の位置ベクトル
//              とその微分
//   TT[3][3], dTT[3][3], ddTT[3][3]
//            : 座標変換行列とその微分
//   PF       : 静止系上の位置ベクトルとその微分
//
//-----------------------------------------------------
void TRANS( VECTOR3 PM, POSITION P0,
          double TT[3][3], double dTT[3][3], double ddTT[3][3],
            POSITION *PF )
{
                        MPV3( TT, PM, &PF->P  );
                        MPV3( dTT, PM, &PF->DP );
                        MPV3( ddTT, PM, &PF->DDP );

                        PF->P.x = PF->P.x + P0.P.x;
                        PF->P.y = PF->P.y + P0.P.y;
                        PF->P.z = PF->P.z + P0.P.z;

                        PF->DP.x = PF->DP.x + P0.DP.x;
                        PF->DP.y = PF->DP.y + P0.DP.y;
                        PF->DP.z = PF->DP.z + P0.DP.z;

                        PF->DDP.x = PF->DDP.x + P0.DDP.x;
                        PF->DDP.y = PF->DDP.y + P0.DDP.y;
                        PF->DDP.z = PF->DDP.z + P0.DDP.z;
}
```

```
//-------------------------------------------------------------
//
//    空間 球－直進－球対偶2連節の運動
//
//   s1, s2   : 両端の球対偶の位置ベクトルとその微分
//   p0       : s1からpまでの固定距離
//   p        : 直進対偶の位置ベクトルとその微分
//   the      : 直進対偶変位とその微分
//
//-------------------------------------------------------------

int s_SPS_links ( POSITION s1, POSITION s2,   double p0,   POSITION *p,  VARIABLE *the )
{
                    double X, Y, Z, dX, dY, dZ, ddX, ddY, ddZ, TH, dTH, ddTH, w1, w2, w3;

                    X = s2.P.x - s1.P.x;
                    Y = s2.P.y - s1.P.y;
                    Z = s2.P.z - s1.P.z;
                    dX = s2.DP.x - s1.DP.x;
                    dY = s2.DP.y - s1.DP.y;
                    dZ = s2.DP.z - s1.DP.z;
                    ddX = s2.DDP.x - s1.DDP.x;
                    ddY = s2.DDP.y - s1.DDP.y;
                    ddZ = s2.DDP.z - s1.DDP.z;

                    TH = sqrt( X*X + Y*Y + Z*Z );
                    if( fabs( TH ) < 1.0e-30 ) {
                                printf( "++ Error in s_SPS_links (特異状態です)¥n");
                                return ERROR;
                    }
                    dTH = ( dX*X + dY*Y + dZ*Z )/TH;
                    ddTH = ( ( ddX*X + dX*dX + ddY*Y + dY*dY + ddZ*Z + dZ*dZ )*TH –
                      ( dX*X + dY*Y + dZ*Z )*dTH )/( TH*TH );
                    w1 = p0/TH;
                    w2 = -p0*dTH/( TH*TH );
                    w3 = -p0*( ddTH  - 2.0*dTH*dTH*TH )/( TH*TH*TH*TH );
                    p->P.x = w1*X + s1.P.x;
                    p->P.y = w1*Y + s1.P.y;
                    p->P.z = w1*Z + s1.P.z;

                    p->DP.x = w2*X + w1*dX + s1.DP.x;
                    p->DP.y = w2*Y + w1*dY + s1.DP.y;
                    p->DP.z = w2*Z + w1*dZ + s1.DP.z;

                    p->DDP.x = w3*X + 2.0*w2*dX + w1*ddX + s1.DDP.x;
                    p->DDP.y = w3*Y + 2.0*w2*dY + w1*ddY + s1.DDP.y;
                    p->DDP.z = w3*Z + 2.0*w2*dZ + w1*ddZ + s1.DDP.z;
                    the->v = TH - p0;
                    the->dv = dTH;
                    the->ddv = ddTH;

                    return SUCCESS;

}
```

```
//-------------------------------------------------------------
//
//    空間 球－球－直進対偶2連節の運動
//
//   s1       : 端の球対偶の位置ベクトルとその微分
//   rs       : 2球対偶間の距離
//   g        : 直進対偶の直動ガイドを表す空間直線上の１点の
//              位置ベクトルとその微分
//   TLT[3][3], dTLT[3][3], ddTLT[3][3]
//            : g3を原点，直進対偶の直動ガイドを表す空間直線を
//              Z軸とする動座標系の座標変換行列とその微分（転置）
//   s2p      : 中央の球対偶の位置ベクトル
//            （直進対偶に固定した動座標系）
//   minv     : 機構の反転の指示インジケータ
//   the      : 直進対偶の変位とその微分
//   p        : 直進対偶の位置ベクトルとその微分
//   s2       : 中央の球対偶の位置ベクトルとその微分
//
//-------------------------------------------------------------

int s_SSP_links ( POSITION s1,  double rs,   POSITION g,
                  double TLT[3][3], double dTLT[3][3], double ddTLT[3][3],
                  VECTOR3 s2p,   int minv,   VARIABLE *the,
                        POSITION *p,   POSITION *s2 )
{
                        VECTOR3 UVLF, PLS1F, dUVLF, dPLS1F, ddUVLF, ddPLS1F,
                        ww, dww, ddww;
                        double a, da, dda, b, db, ddb, d, rootd, w1, w2, w3, w4, w5;

                        UVLF.x = TLT[0][2];
                        UVLF.y = TLT[1][2];
                        UVLF.z = TLT[2][2];
                        PLS1F.x = g.P.x - s1.P.x;
                        PLS1F.y = g.P.y - s1.P.y;
                        PLS1F.z = g.P.z - s1.P.z;

                        VIV3 ( UVLF, PLS1F, &a );
                        a = a + s2p.z;

                        MPV3 ( TLT, s2p, &ww );
                        VIV3 ( ww, PLS1F, &b );
                         b = b*2.0 + PLS1F.x*PLS1F.x + PLS1F.y*PLS1F.y
                    + PLS1F.z*PLS1F.z + s2p.x*s2p.x + s2p.y*s2p.y + s2p.z*s2p.z - rs*rs;

                        d = a*a - b;
```

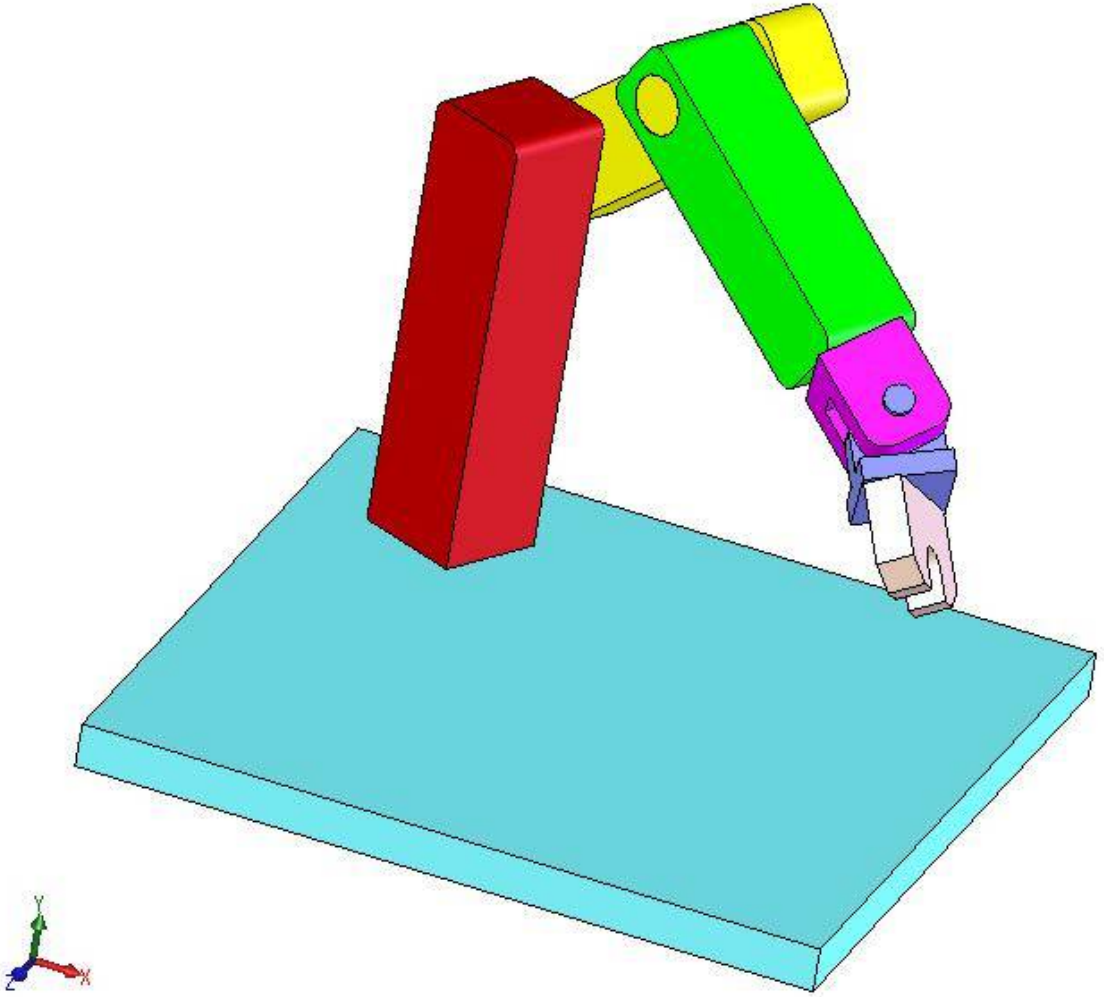
```
if( d<0 ) {
    rintf("++ Error in s_SSP_links (連鎖が構成できません.) ¥n");
                return ERROR;
}
rootd = sqrt( d );
dUVLF.x = dTLT[0][2];
dUVLF.y = dTLT[1][2];
dUVLF.z = dTLT[2][2];
dPLS1F.x = g.DP.x - s1.DP.x;
dPLS1F.y = g.DP.y - s1.DP.y;
dPLS1F.z = g.DP.z - s1.DP.z;
ddUVLF.x = ddTLT[0][2];
ddUVLF.y = ddTLT[1][2];
ddUVLF.z = ddTLT[2][2];
ddPLS1F.x = g.DDP.x - s1.DDP.x;
ddPLS1F.y = g.DDP.y - s1.DDP.y;
ddPLS1F.z = g.DDP.z - s1.DDP.z;

VIV3 ( dUVLF, PLS1F, &w1 );
VIV3 ( UVLF, dPLS1F, &da );
da = da + w1;
VIV3 ( ddUVLF, PLS1F, &w1 );
VIV3 ( dUVLF, dPLS1F, &w2 );
VIV3 ( UVLF, ddPLS1F, &dda );
dda = dda + w1 + 2.0*w2;

MPV3 ( dTLT, s2p, &dww );
MPV3 ( ddTLT, s2p, &ddww );
VIV3 ( dPLS1F, PLS1F,  &w1 );
VIV3 ( dww, PLS1F,  &w2 );
VIV3 ( ww, dPLS1F,  &w3 );

db = 2.0*( w1 + w2 + w3 );
VIV3 ( ddPLS1F, PLS1F,  &w1 );
VIV3 ( dPLS1F, dPLS1F,  &w2 );
VIV3 ( ddww, PLS1F,  &w3 );
VIV3 ( dww, dPLS1F,  &w4 );
VIV3 ( ww, ddPLS1F,  &w5 );
ddb = 2.0*( w1 + w2 + w3 + 2.0*w4 + w5 );

                the->dv = da - ( 2.0*da*a - db )/( 2.0*rootd );
                the->ddv = -dda - ( 2.0*(2.0*dda*a + 2.0*da*da
                  - ddb )*d - ( 2.0*da*a - db )*( 2.0*da*a - db ) )
                  /( 4.0*d*rootd );
}
```

```
p->P.x = the->v*UVLF.x + g.P.x;
                p->P.y = the->v*UVLF.y + g.P.y;
                p->P.z = the->v*UVLF.z + g.P.z;
                p->DP.x = the->dv*UVLF.x + the->v*dUVLF.x + g.DP.x;
                p->DP.y = the->dv*UVLF.y + the->v*dUVLF.y + g.DP.y;
                p->DP.z = the->dv*UVLF.z + the->v*dUVLF.z + g.DP.z;
                p->DDP.x = the->ddv*UVLF.x + 2.0*the->dv*dUVLF.x
                        + the->v*ddUVLF.x + g.DDP.x;
                p->DDP.y = the->ddv*UVLF.y + 2.0*the->dv*dUVLF.y
                        + the->v*ddUVLF.y + g.DDP.y;
                p->DDP.z = the->ddv*UVLF.z + 2.0*the->dv*dUVLF.z
                        + the->v*ddUVLF.z + g.DDP.z;

                s2->P.x = p->P.x + ww.x;
                s2->P.y = p->P.y + ww.y;
                s2->P.z = p->P.z + ww.z;
                s2->DP.x = p->DP.x + dww.x;
                s2->DP.y = p->DP.y + dww.y;
                s2->DP.z = p->DP.z + dww.z;
                s2->DDP.x = p->DDP.x + ddww.x;
                s2->DDP.y = p->DDP.y + ddww.y;
                s2->DDP.z = p->DDP.z + ddww.z;

                return SUCCESS;
}
```

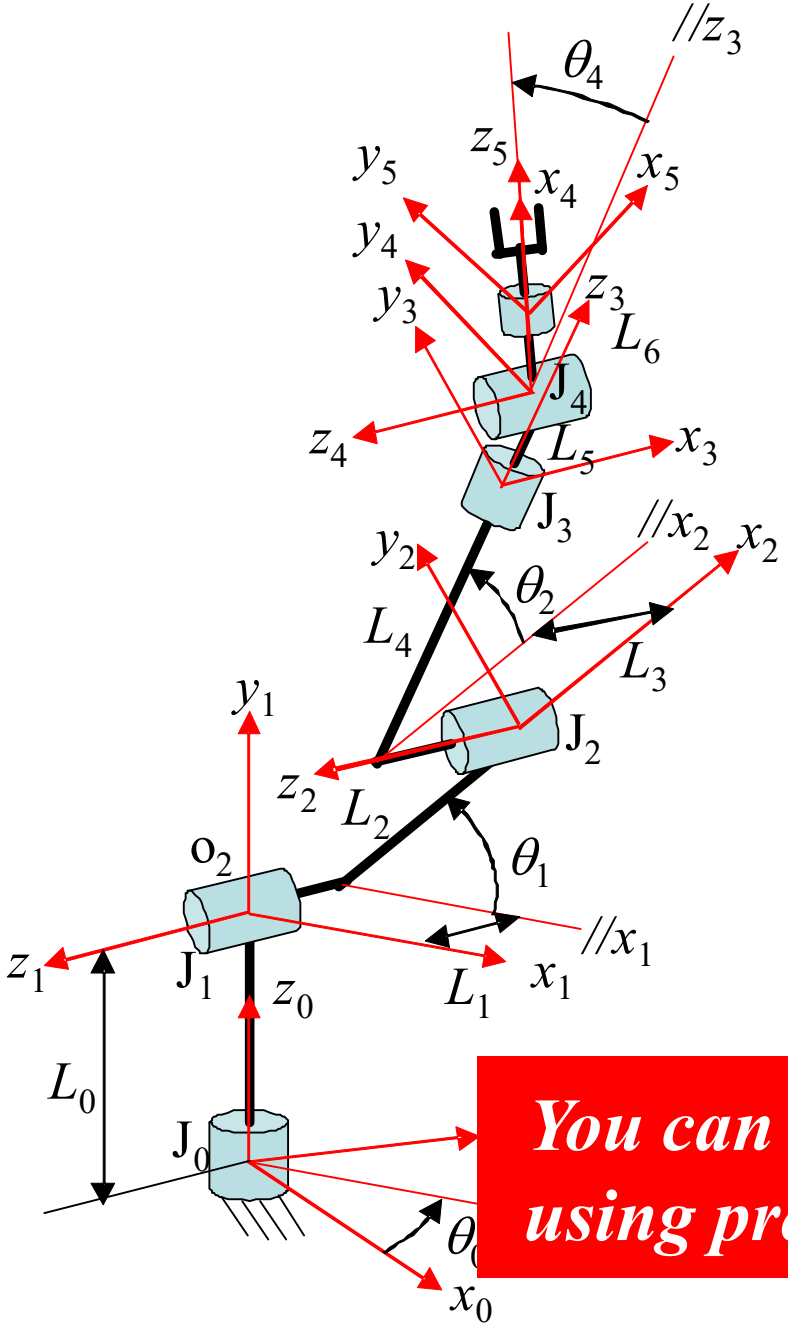
**You can download these programs from WEB-site:**
**http://www.dynamics.mep.titech.ac.jp/japanese/download.html**

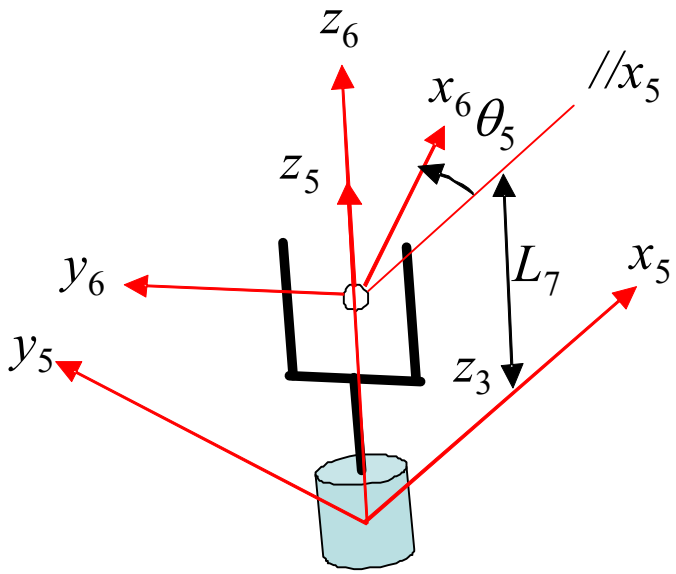
**Examples of analysis：**

(1)PUMA type spatial serial manipulator



**Spatial serial mechanism with only revolute pairs and 6 DOF**

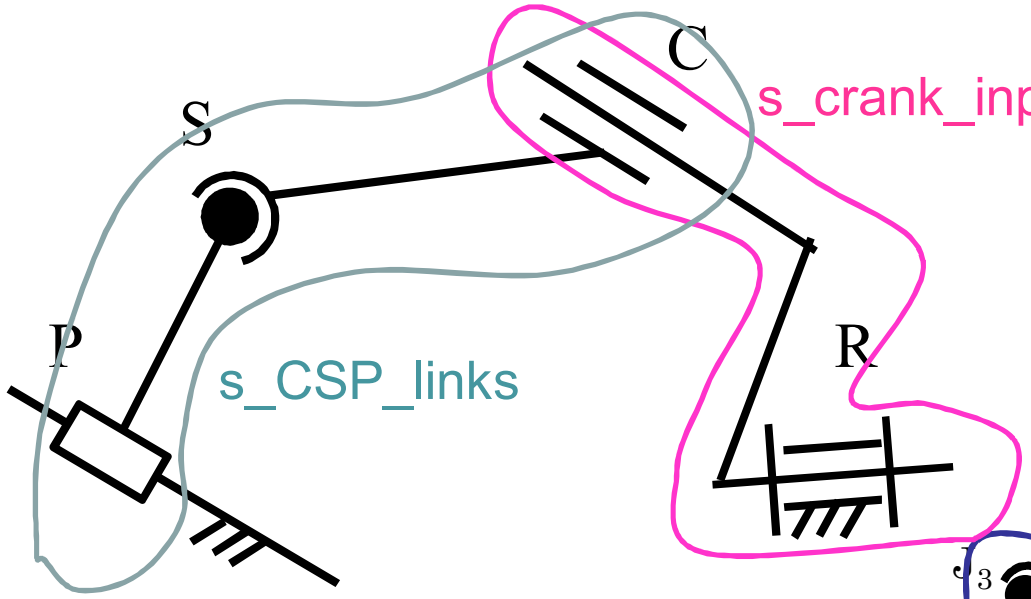


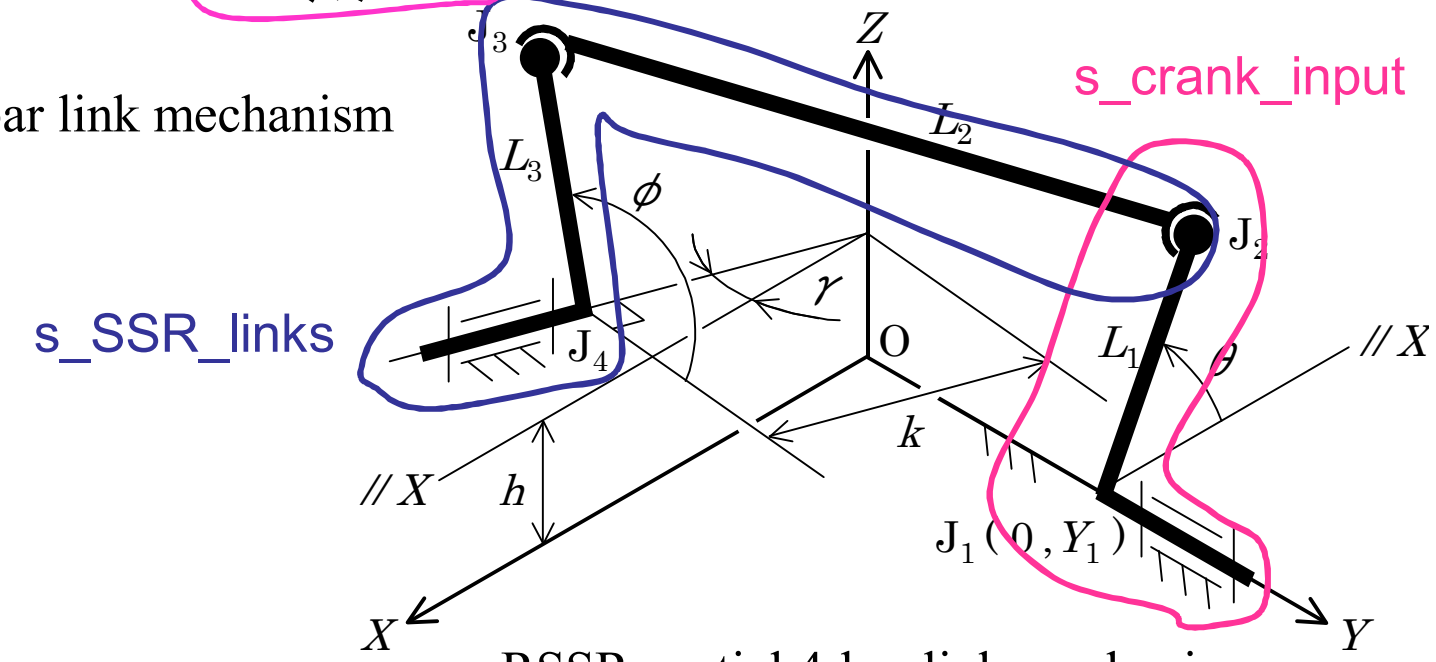

**Location of coordinate systems**

***You can calculate forward kinematics by using program :s_crank_input only.***

# (2)Spatial 4-bar link mechanisms with 1 DOF

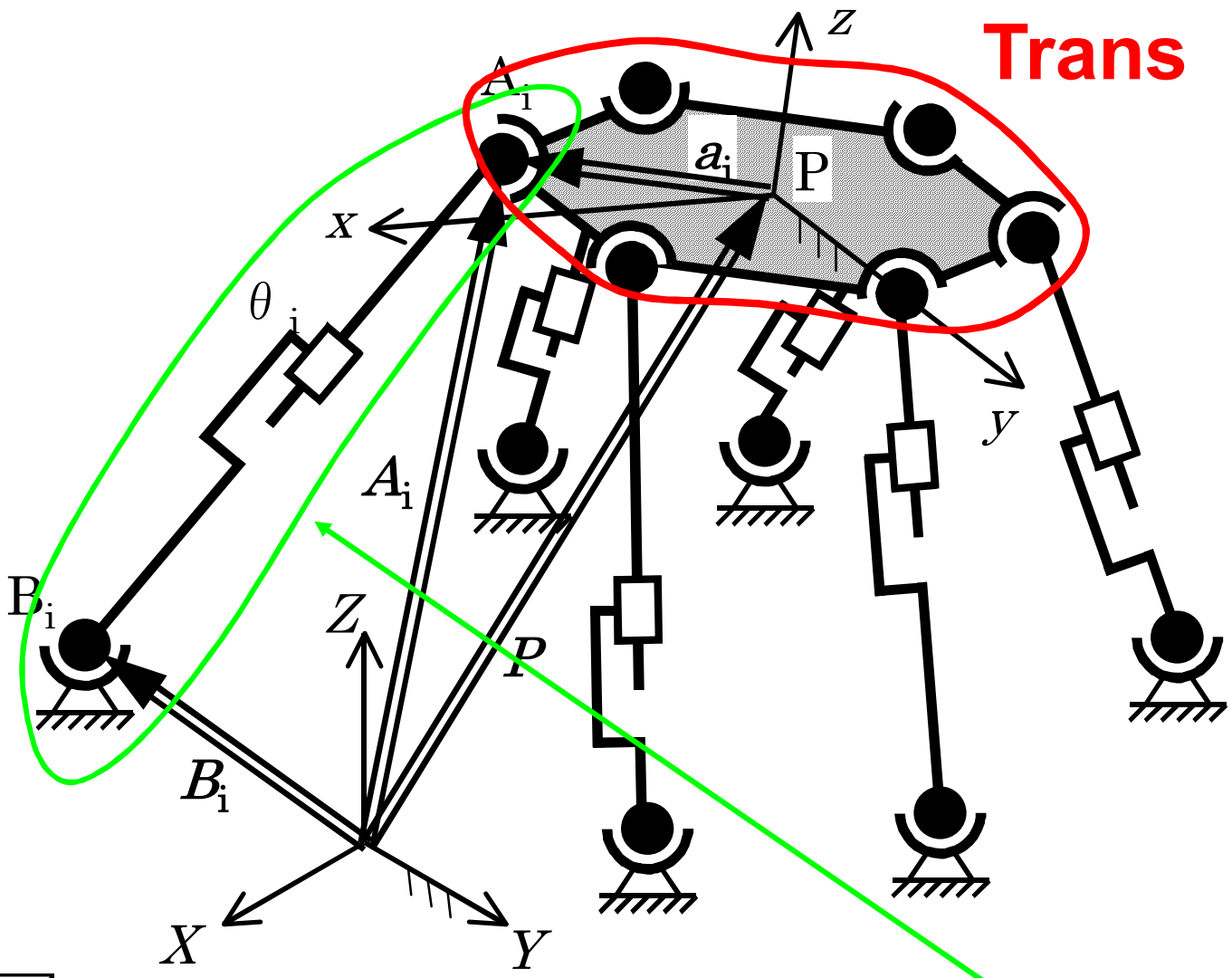


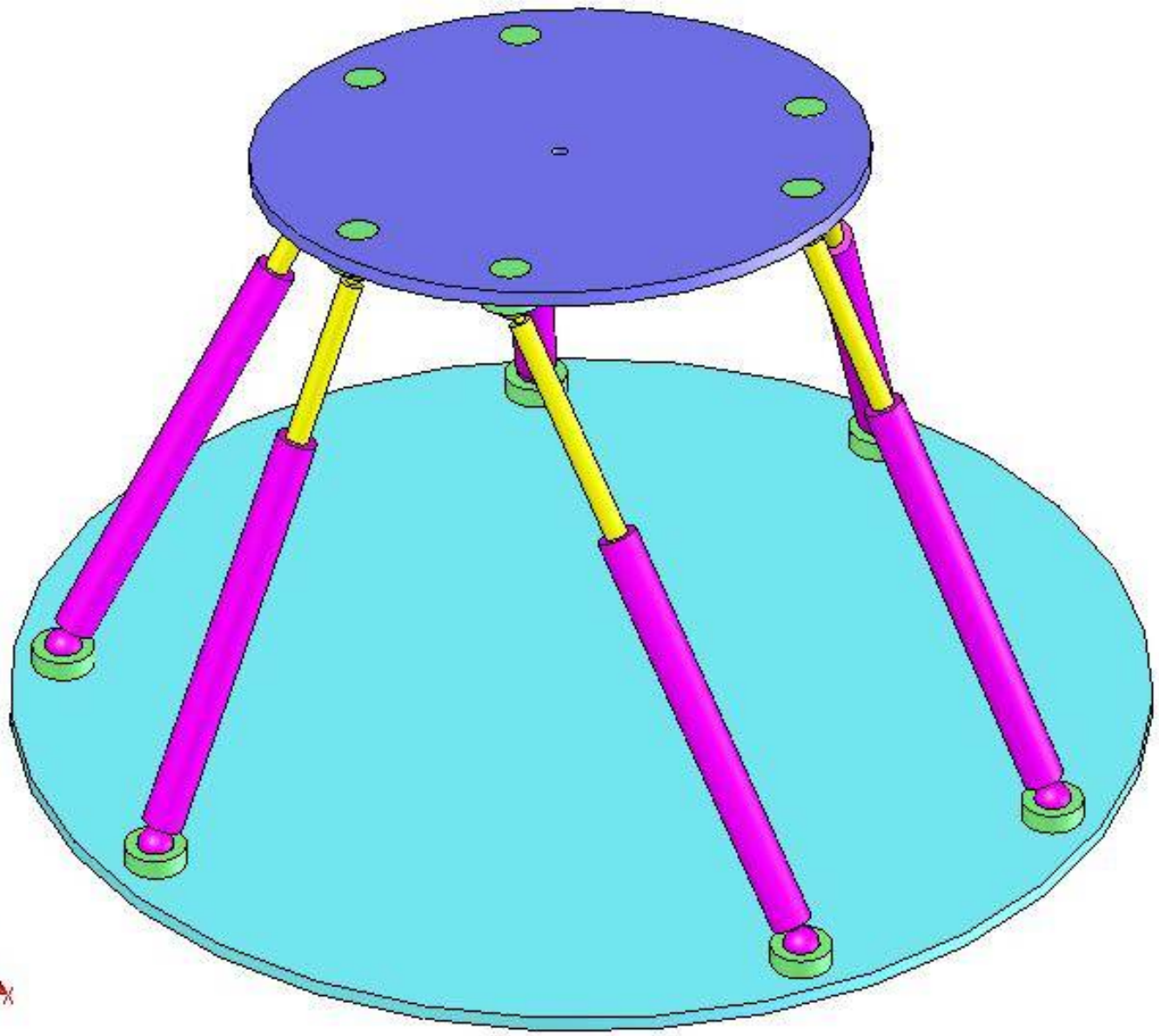

RCSP spatial 4-bar link mechanism




RSSR spatial 4-bar link mechanisms

(3)Inverse kinematics of Stewart platform manipulator





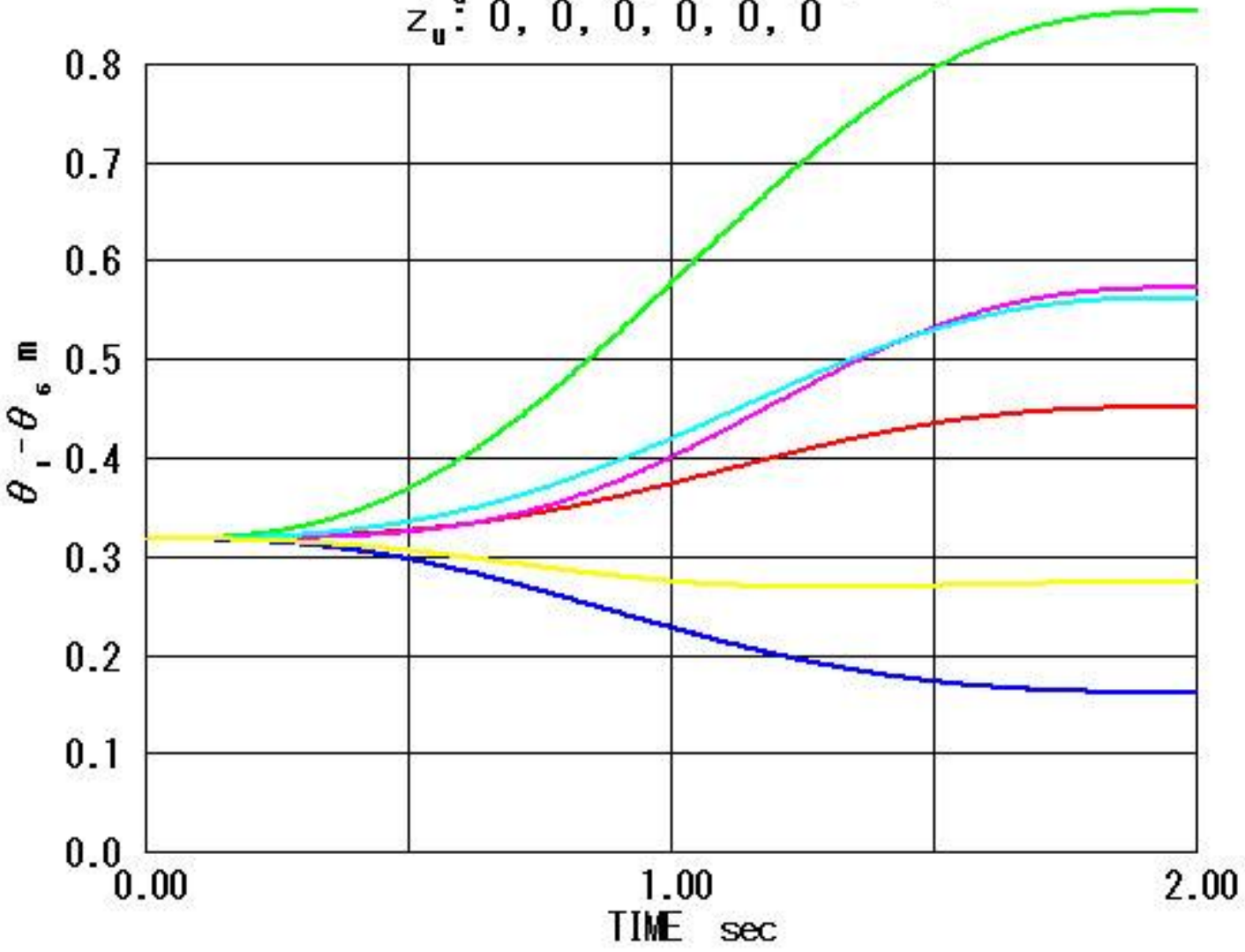
# Example

スチュワートプラットフォームマニピュレータの逆運動学解析 with RPIKSTEWARTP.BAS

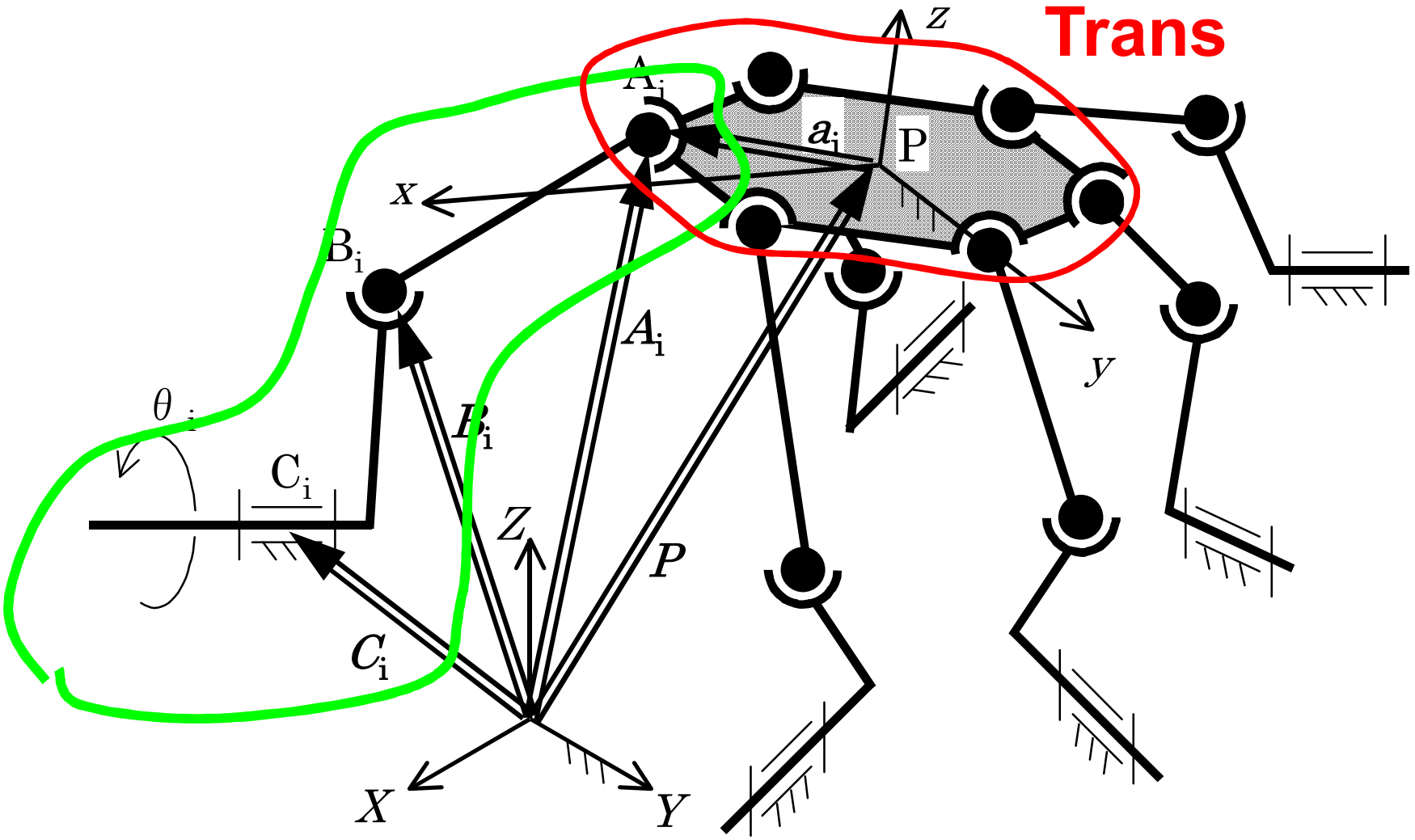
入力関節変位

Example of calculation

$R_L$ : 1, 1, 1, 1, 1, 1
$\psi_L$ : 20, 100, 140, 220, 260,-20
$Z_L$ : 0, 0, 0, 0, 0, 0
$R_u$ : .5, .5, .5, .5, .5, .5
$\psi_u$ : 40, 80, 160, 200,-80,-40
$z_u$ : 0, 0, 0, 0, 0, 0



# (4)Inverse kinematics of 6RSS spatial parallel manipulator



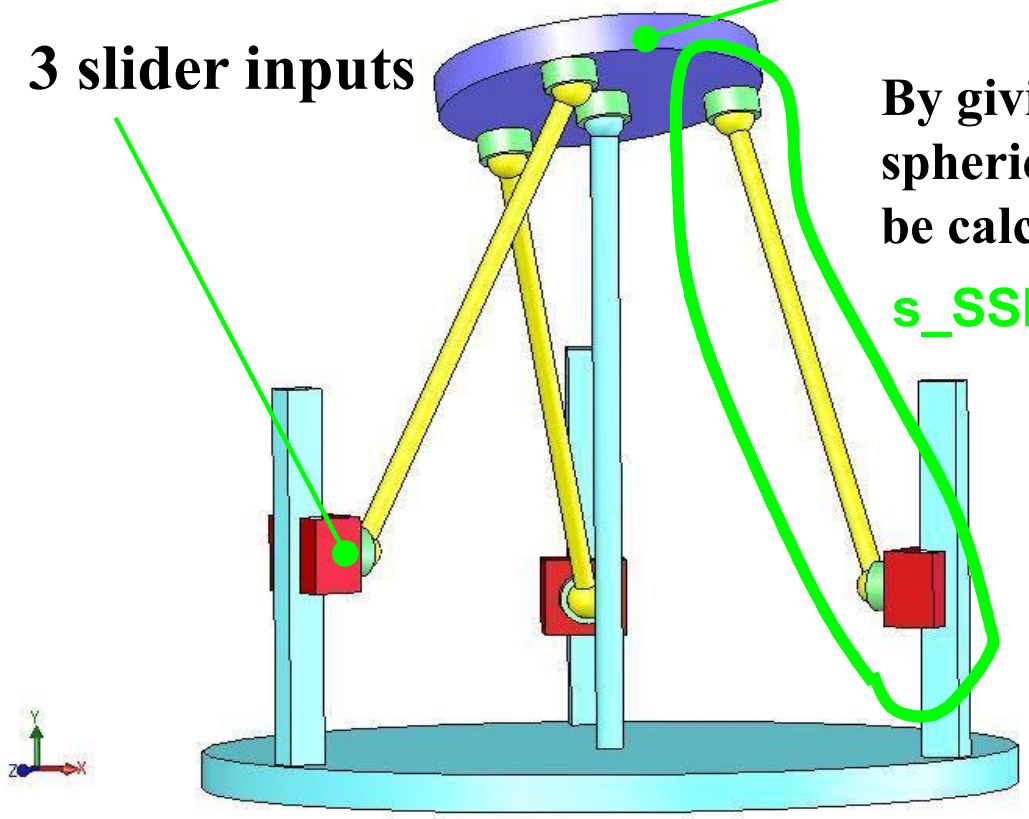

**s_SSR_LINKS**

**By changing two adjacent links**

# (5)Inverse kinematics of a camera head driving mechanism with 3 DOF

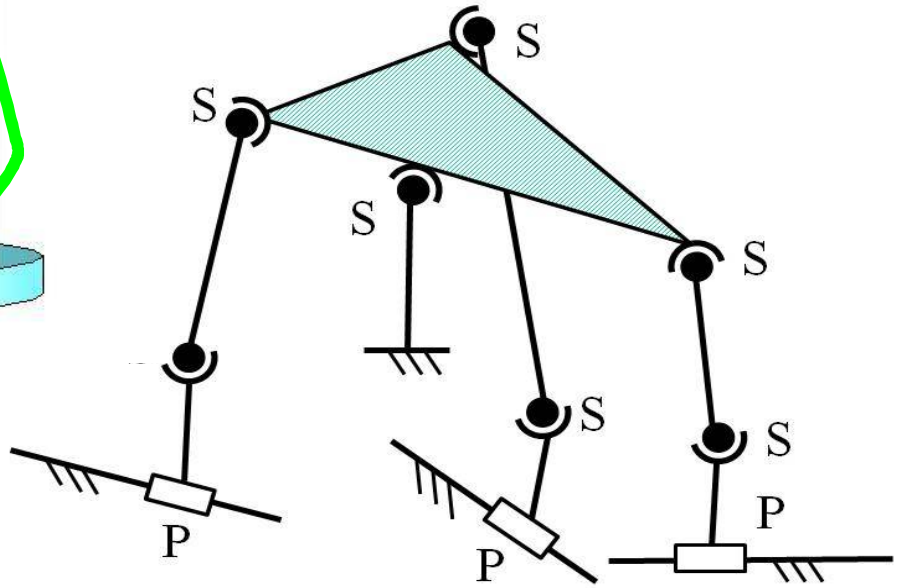
**A camera is mounted on a moving platform and its 3 posture angles can be controlled.**

**3 slider inputs**



**By giving 3 posture angles, positions of spherical pairs on the moving platform can be calculated.**

**s_SSP_links**



*You can analyze various spatial link mechanisms by using the offered programs.*

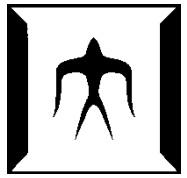
*Principal analyses:*

*(1)Spatial angular motion expressed by coordinate transformation matrix*

*(2)Crank/slider input links*

*(3)Two adjacent links*

*(4)Coupler point motion as coordinate transformation*



# 4. Concluding remarks

**As an available tool to analyze or design link mechanisms is expanded to spatial mechanisms.**

**(1)Spatial angular motion can be expressed with coordinate transformation.**

**(2)Principal analysis:**

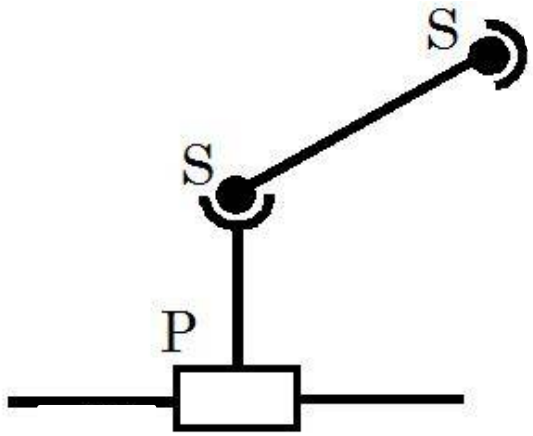
**Input link/Two adjacent links/Coupler point**

**(3)Not only displacement but also velocity and acceleration can be calculated for various mechanisms**

**(4)Students are expected to experience to analyze some mechanisms with the offered programs.**

# Homework 2

**Derive the position vector of spherical pair $S_2$ in SSP links when position and posture of prismatic pair P and position of pair $S_1$ at both ends and mechanism constants are given.**




**The result will be summarized in A4 size PDF with less than 5 pages and sent to Prof. Iwatsuki via T2SCHOLA by May 2, 2023.**