Robótica
2. Modelado Cinemática de Robots

F. Hugo Ramírez Leyva

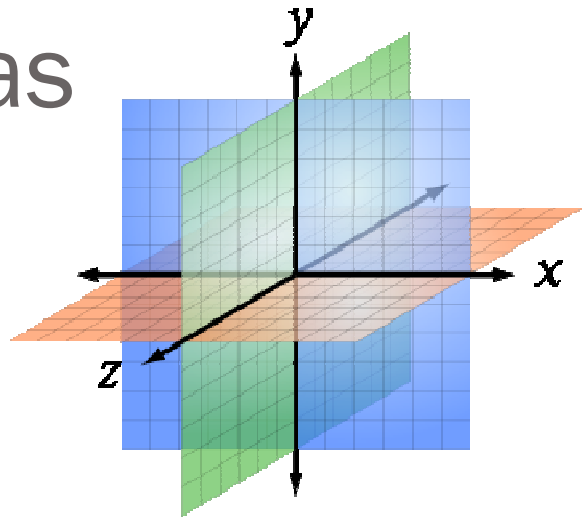
Cubículo 3

Instituto de Electrónica y Mecatrónica

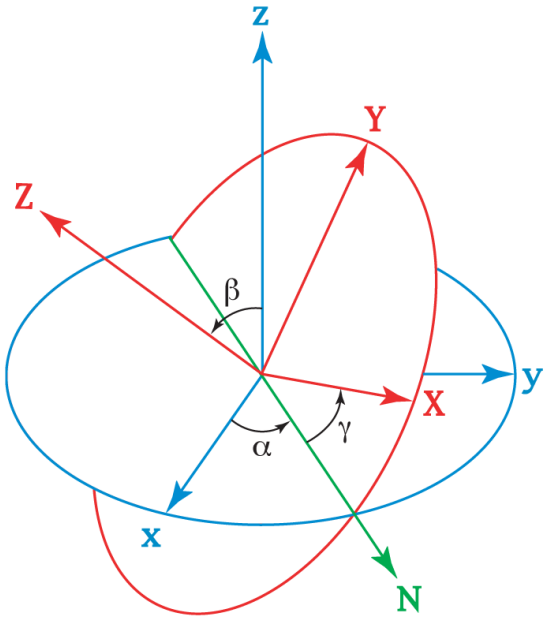
hugo@mixteco.utm.mx

Marzo 2012

Sistema de Coordenadas

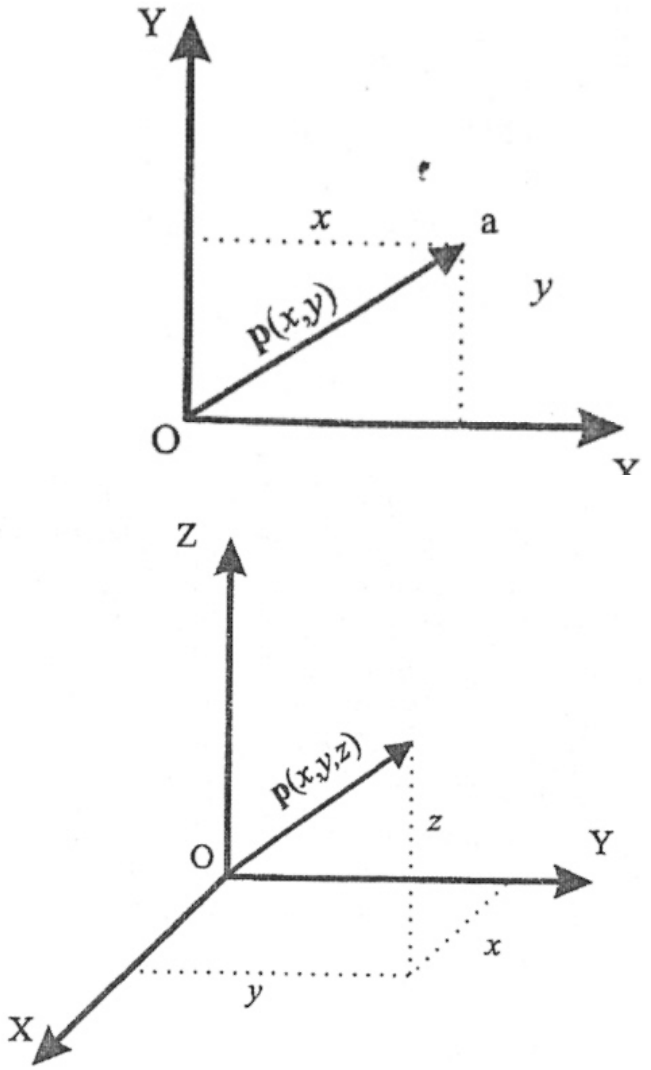


- La manipulación de piezas requiere el movimiento espacial de su extremo.
- Es necesario conocer su posición y orientación de ésta con respecto a la base del robot.
- Es necesario contar con herramientas que realicen este trabajo.
- Existe una teoría general para la localización de objetos en el espacio que puede aplicarse a otras áreas, estas son:
  - Sistemas de coordenadas: Cartesiano, cilíndrico, esférico
  - Matrices de transformaciones: Traslación y rotación. Método de Denavit- Hartenberg.
  - Cuaternios.



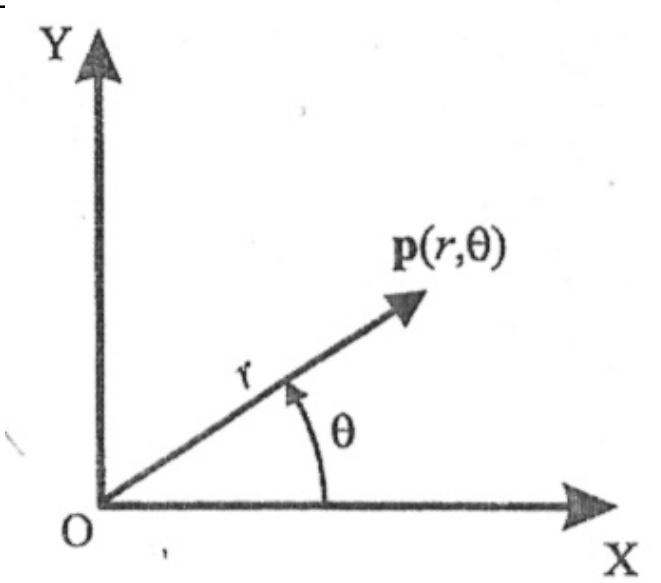
Coordenadas Cartesianas

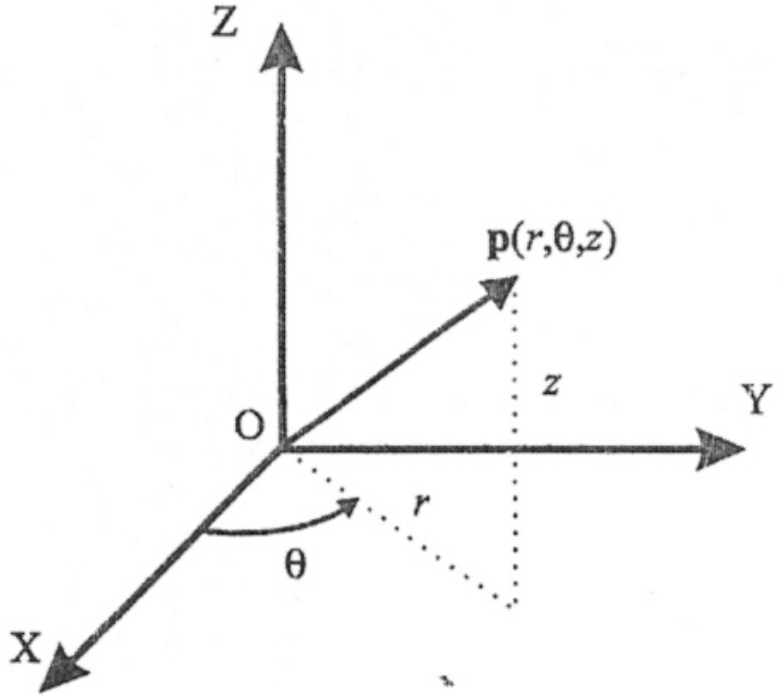
- Se trabaja en un sistema coordenado OXYZ.
- Cualquier punto a está expresado por las componentes (x,y,z).
- Este punto tiene asociado un vector **p**(x,y,z).



Cilíndrico

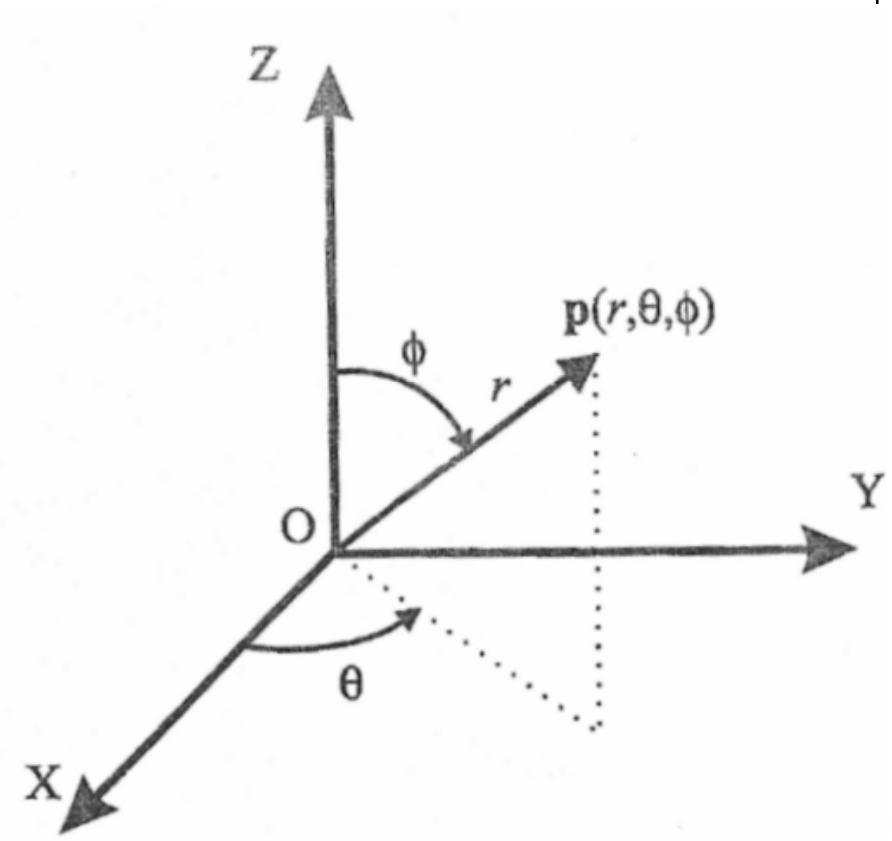
- Se trabaja en un sistema coordenado OXYZ.
- Se utilizan coordenadas polares **p**(r, ,z)
- r es la distancia del origen O al extremo del vector p.
- es el ángulo que forma el vetor p con el eje OX.
- z representa la proyección sobre el eje OZ





Esférico

- El vector p(r,) es el extremo del punto a.
- R es la distancia del origen hasta el extremo de p
- es el ángulo formado por la proyección del vector p sobre el plano =XY
- es el ángulo formado por el vector p con el eje OZ



Ejemplo

- Sea el vector en coordenadas cartesianas [2,3,5], encontrar su representación en coordenadas cilíndricas y esféricas

Coordenadas Cilíndricas.

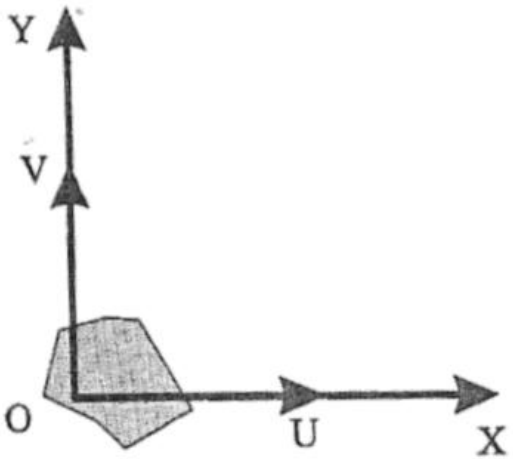
- $r = \sqrt{x^2 + y^2} = 3.605$
- $\theta = \tan^{-1}\left(\frac{y}{x}\right) = 56.3°$
- $z = z = 5$

Coordenadas esféricas:

- $r = \sqrt{x^2 + y^2 + z^2} = 6.164$
- $\theta = \tan^{-1}\left(\frac{y}{x}\right) = 56.3°$
- $\phi = \cos^{-1}\left(\frac{z}{r}\right) = 35.79°$

Representación de la Orientación

- Además de la posición es necesario definir la orientación con respecto al sistema de referencia.
- La orientación en un espacio tridimensional, viene definida por tres grados de libertad, linealmente independientes.
- Normalmente se usan 2 sistemas de referencia.
- Si se tiene 2 sistemas de referencia **OXY** y **OUV**, con el mismo origen, pero rotado un ángulo
- Cada vector del sistema de referencia es y deben ser equivalentes.
- Con una matriz de rotación **R** se define la orientación de **OUV** con respecto a **OXY**



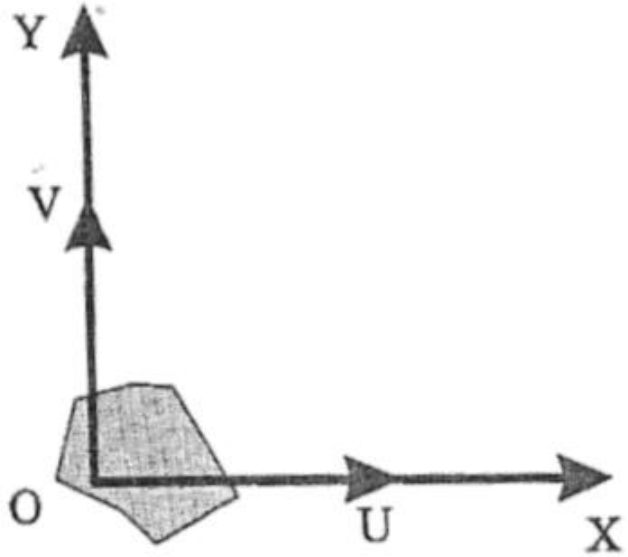
Matriz de rotación

- Sea el sistema de coordenadas OXY y OUV
- Vectores unitarios:

$$OXY \rightarrow i_x, i_y$$

$$OUV \rightarrow i_u, i_v$$

- Cualquier punto sobre el plano puede ser expresado como:



$$p_{xy} = \left[p_x, p_y\right]^T = p_x i_x + p_y j_y$$

$$p_{uv} = \left[p_u, p_v\right]^T = p_u \bullet i_u + p_v \bullet i_v$$

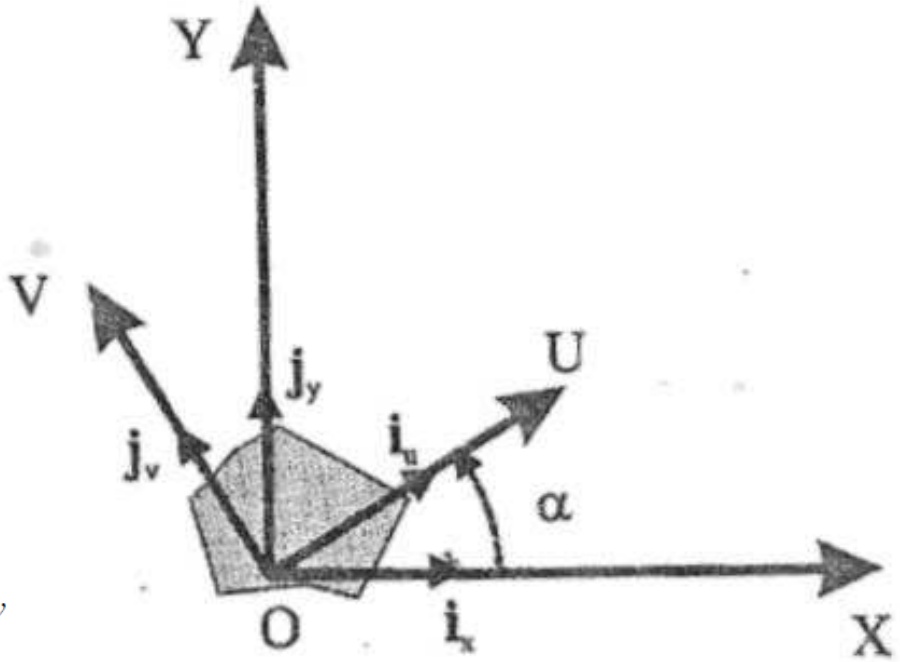
$$\text{Pxy=Puv} \Rightarrow \quad p_{uv} = p_{xy}$$

Matriz de rotación

- Si el sistema esta rotado

$$p_{xy} = [p_x, p_y]^T = p_x i_x + p_y j_y$$

$$p_{uv} = [p_u, p_v]^T = p_u i_u + p_v j_v$$



- Como

$$p_x = p_{uv} \bullet i_x = p_u i_x \bullet i_u + p_v i_x \bullet j_v$$

$$p_y = p_{uv} \bullet j_y = p_u j_y \bullet i_u + p_v j_y \bullet j_v$$

- Entonces

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = R \begin{bmatrix} p_u \\ p_v \end{bmatrix}$$

Matriz de rotación

- La matriz de rotación esta dada por:

$$R = \begin{bmatrix} i_x \bullet i_u & i_x \bullet j_v \\ j_y \bullet i_u & j_y \bullet j_v \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -sen(\alpha) \\ sen(\alpha) & \cos(\alpha) \end{bmatrix}$$

- Se usan la identidades trigonométricas

$$sen(x \pm y) = sen(x)\cos(y) \pm \cos(x)sen(y)$$

$$\cos(x \pm y) = \cos(x)\cos(y) \mp sen(x)sen(y)$$

Ejemplo

- Suponer que para un sistema de 2 dimensiones Puv=[1,2] y =90° y 45°. Encontrar en forma gráfica las coordenadas con respecto al sistema xy.
- Sol. Para θ=90°
  - Pxy=[-2, 1].
- Para θ=45°
  - Pxy=[-0.7071,2.1213]

- %Código de Matlab
- Puv=[1;2];
- teta=pi/2;
- R=[cos(teta),-sin(teta);sin(teta),cos(teta)];
- pxy=R*Puv
- pause
- teta2=pi/4;
- R2=[cos(teta2),-sin(teta2);sin(teta2),cos(teta2)];
- pxy2=R2*Puv

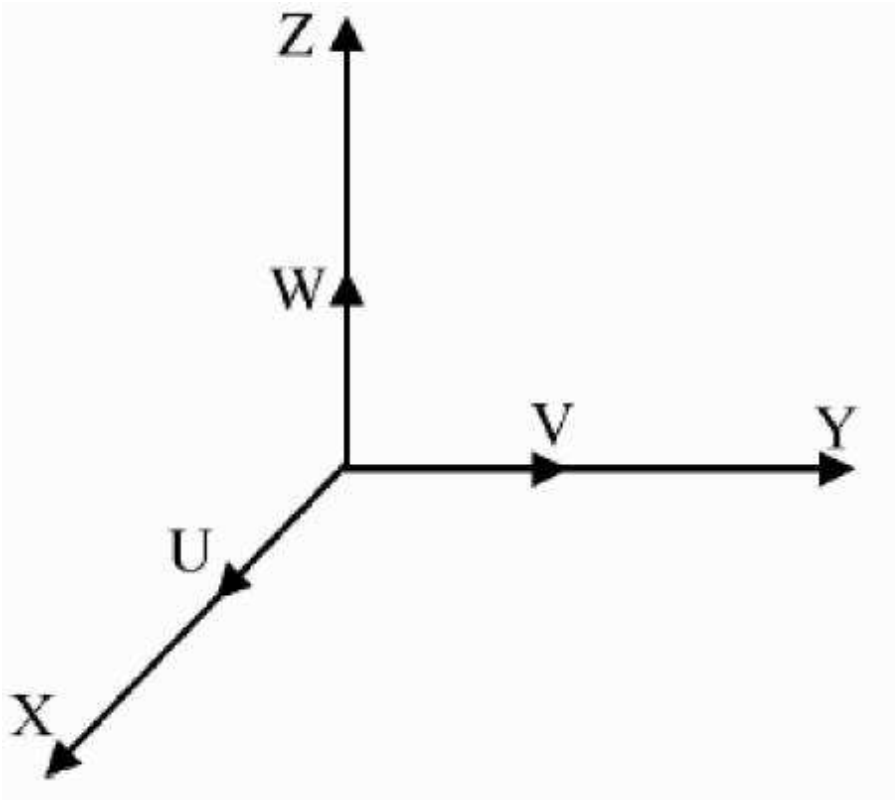
Rotación en 3d

- Sistema OXYZ
- Vectores unitarios ix, jy, kz.
- Ejes X, Y, Z.
- Sistema OUVW, Vectores unitarios iu, jv, kw.
- Ejes U, V, W. Vector con respecto al eje UVW

$$\vec{P}_{uvw} = \begin{bmatrix} p_u & p_v & p_w \end{bmatrix}$$

$$\vec{P}_{xyz} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}$$

$$\vec{P}_{xyz} = \mathbf{R}\vec{P}_{uvw}$$



Rotación en 3d

$$p_{xyz} = \left[p_x, p_y, p_z\right]^T = p_x i_x + p_y j_y + p_z k_z$$

$$p_{uvw} = \left[p_u, p_v, p_w\right]^T = p_u i_u + p_v j_v + p_w j_w$$

La proyección del vector **Puvw** sobre el sistema **xyz** es:

$$p_x = \vec{P}_{uvw} \bullet i_x = p_u\left(i_u \bullet i_x\right) + p_v\left(j_v \bullet i_x\right) + p_w\left(k_w \bullet i_x\right)$$

$$p_y = \vec{P}_{uvw} \bullet j_y = p_u\left(i_u \bullet j_x\right) + p_v\left(j_v \bullet j_y\right) + p_w\left(k_w \bullet j_y\right)$$

$$p_z = \vec{P}_{uvw} \bullet k_z = p_u\left(i_u \bullet k_z\right) + p_v\left(j_v \bullet k_z\right) + p_w\left(k_w \bullet k_z\right)$$

Rotación en 3d

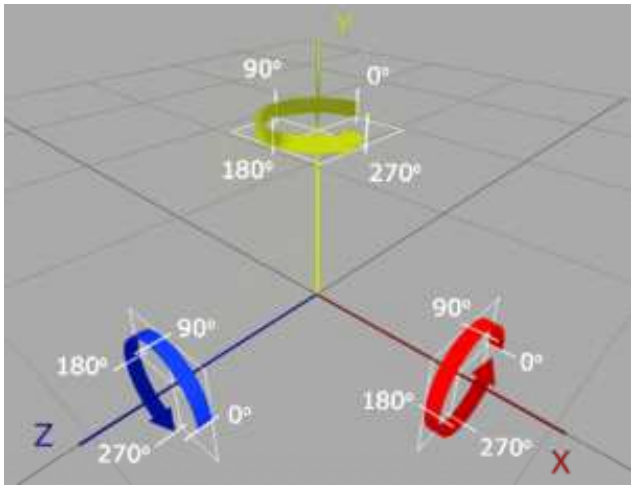
- La matriz de rotación R en tres dimensiones es:

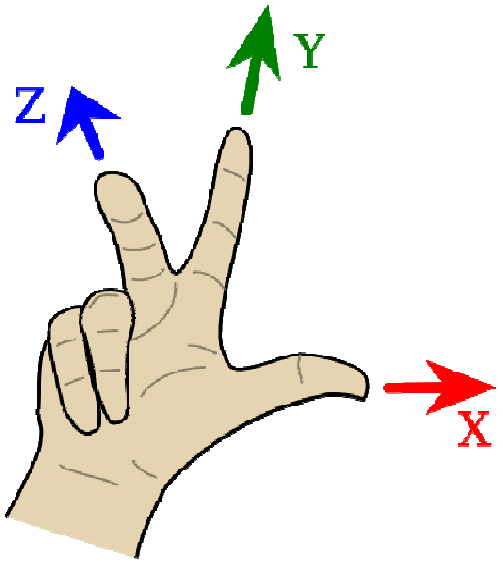
$$\vec{P}_{xyz} = \mathbf{R}\vec{P}_{uvw} = \begin{bmatrix} i_u \bullet i_x & j_v \bullet i_x & k_w \bullet i_x \\ i_u \bullet j_y & j_v \bullet j_y & k_w \bullet j_y \\ i_u \bullet k_z & j_v \bullet k_z & k_w \bullet k_z \end{bmatrix}$$

- Si el sistema ***uvw*** no esta rotado con respecto al **xyz** entonces **R** es una matriz diagonal.

Rotaciones con respecto a los ejes X, Y y Z

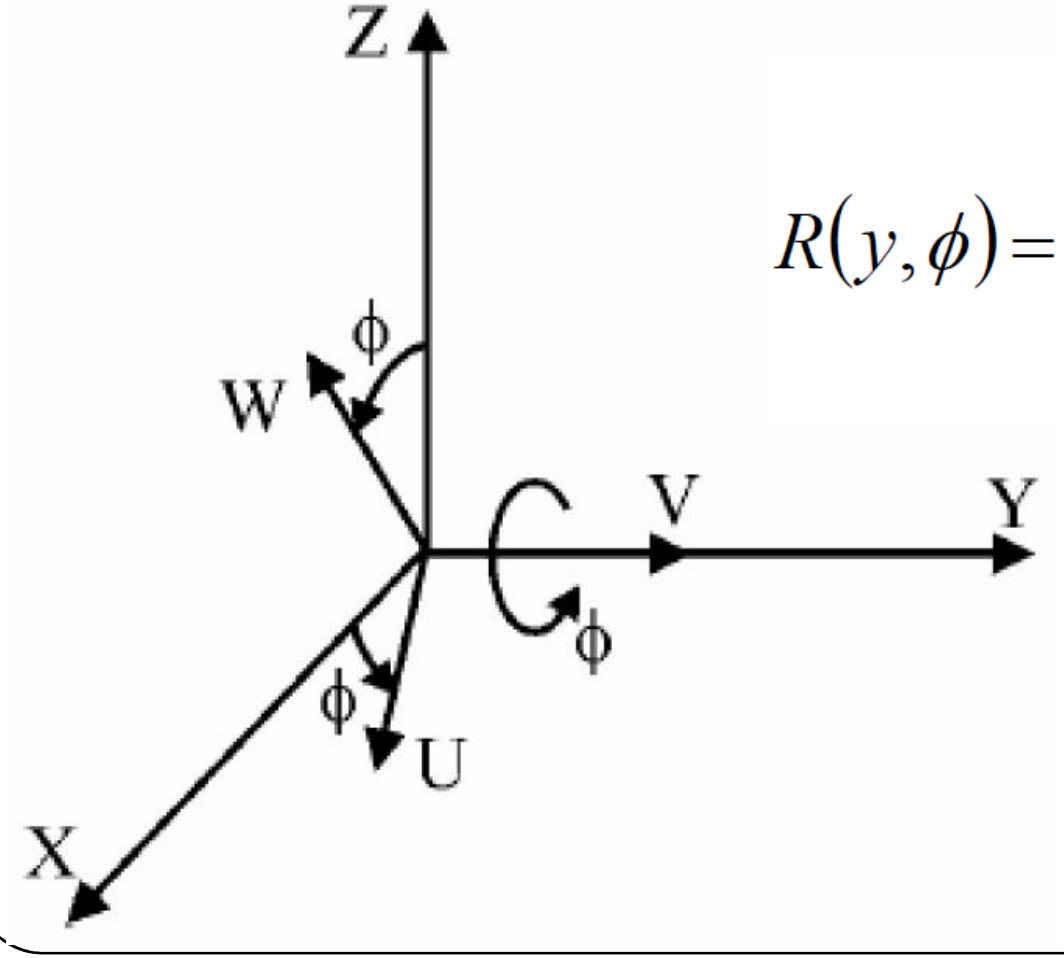
- En tres dimensiones se pueden hacer tres rotaciones diferentes:
- Rotación en OX
- Rotación en OY
- Rotación en OZ





Rotación con respecto al eje Y

- Si se hace una rotación tomando como eje de giro al Y, la matriz de rotación R(y,ϕ) es:



$$R(y,\phi)=\begin{bmatrix} \cos(\phi) & 0 & sen(\phi) \\ 0 & 1 & 0 \\ -sen(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

Ejemplo Rotación OY

- Encontrar el vector Pxyz, cuando el punto Puvw=[1,1,2], con φ=90° con respecto al eje OY

$$\vec{P}_{xyz} = R(y,90^0)\vec{P}_{uvw} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$$

- Sea el sistema OUVW rotado 45° con respecto a eje OY fijo. El punto Puvw= [1, 2, 3], encontrar el punto con respecto al sistema OXYZ

$$\vec{P}_{xyz} = R(y,45^0)\vec{P}_{uvw} = \begin{bmatrix} 0.7071 & 0 & 0.7071 \\ 0 & 1 & 0 \\ -0.7071 & 0 & 0.7071 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2.828 \\ 2 \\ 1.414 \end{bmatrix}$$

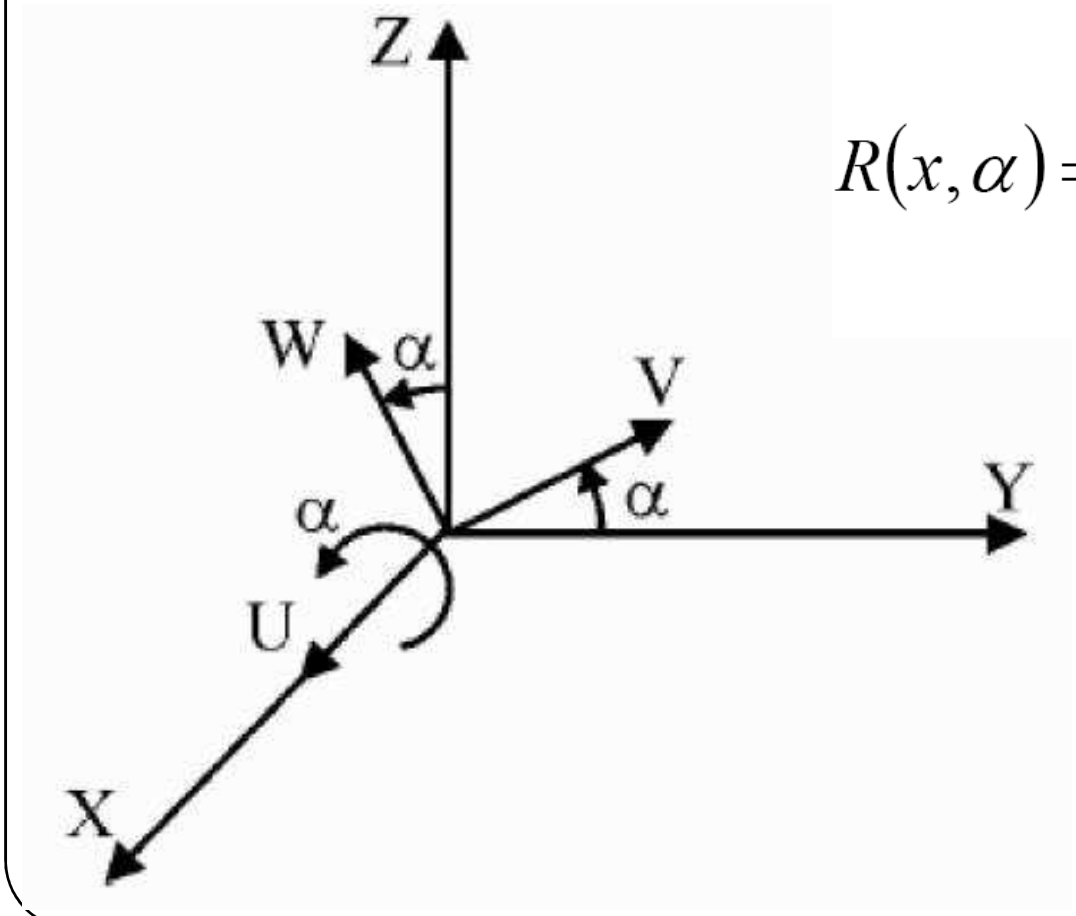
Eejemplo con Matlab

- clear; close;
- Puvw=[1;2;3];
- afi=pi/2
- Ry=[cos(afi),0,sin(afi);
- 0 1 0;
- -sin(afi),0,cos(afi)];
- Pxyz=Ry*Puvw

- clear; close;
- Puvw=[1;2;3];
- afi=pi/4
- Ry=[cos(afi),0,sin(afi);
- 0 1 0;
- -sin(afi),0,cos(afi)];
- Pxyz=Ry*Puvw

Rotación con respecto al eje X

- Si se hace una rotación tomando como eje de giro al OX, la matriz de rotación R(y,α) es:



$$R(x,\alpha)=\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) \\ 0 & sen(\alpha) & \cos(\alpha) \end{bmatrix}$$

Ejemplo Rotación OX

- Ejemplo: Sea el punto Puvw=[1,2,3], referenciado a un sistema rotado 90° con respecto al eje OX. Encontrar el punto Pxyz en el sistema fijo.

$$\text{Solución= } \vec{P}_{xyz} = R(x,90^0)\vec{P}_{uvw} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ -3 \\ 2 \end{bmatrix}$$

- Repetir el problema anterior con un ángulo de 30°.

$$\text{Solución= } \vec{P}_{xyz} = R(x,90^0)\vec{P}_{uvw} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.866 & -0.5 \\ 0 & 0.5 & 0.866 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.232 \\ 3.598 \end{bmatrix}$$

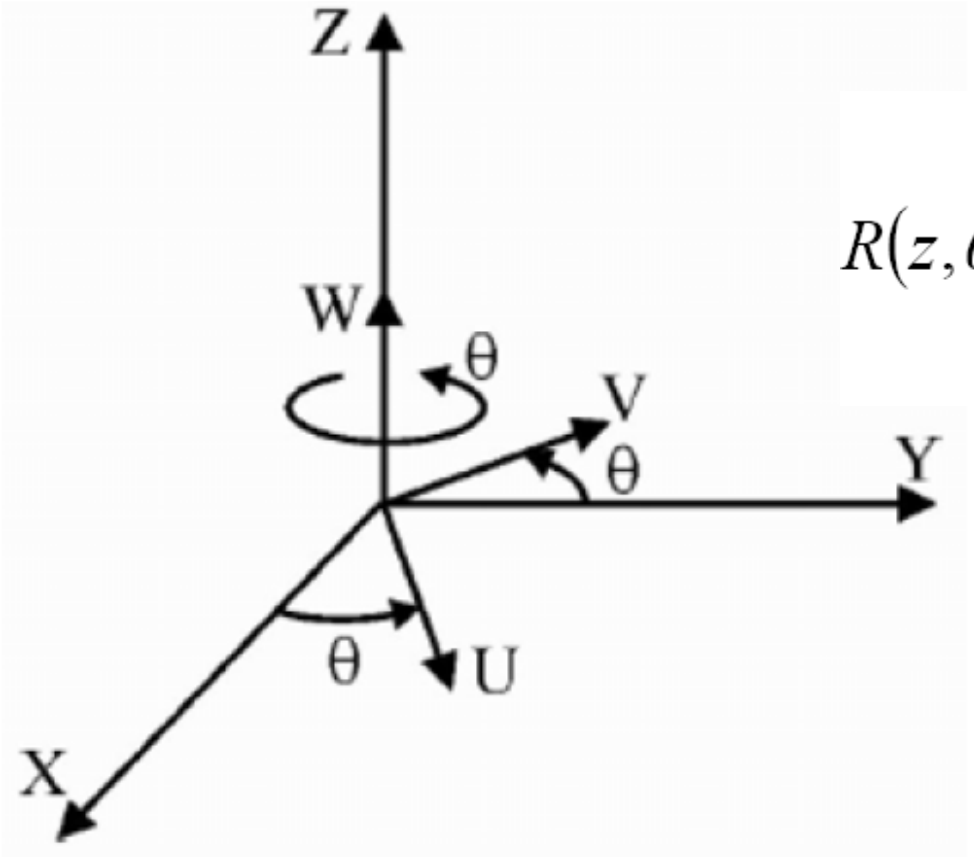
Eejemplo con Matlab

- clear; close;
- Puvw=[1;2;3];
- afi=pi/2
- Ry=[cos(afi),0,sin(afi);
- 0 1 0;
- -sin(afi),0,cos(afi)];
- Pxyz=Ry*Puvw

- clear; close;
- Puvw=[1;2;3];
- afi=pi/4
- Rx=[1,0,0;0,cos(afi),-sin(afi);0,sin(afi),cos(afi)];
- Pxyz=Rx*Puvw

Ejemplo Rotación OZ

- Si se hace una rotación tomando como eje de giro al OZ, la matriz de rotación R(z,θ) es



$$R(z,\theta)=\begin{bmatrix}\cos(\theta) & -sen(\theta) & 0\\ sen(\theta) & \cos(\theta) & 0\\ 0 & 0 & 1\end{bmatrix}$$

Ejemplo Rotación OZ

- Ejemplo: Sea el punto Puvw=[4,6,5], referenciado a un sistema rotado 90° con respecto al eje OZ. Encontrar el punto Pxyz en el sistema fijo.

$$\text{Solución= } \vec{P}_{xyz} = R(z,90^0)\vec{P}_{uvw} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix} = \begin{bmatrix} -6 \\ 4 \\ 5 \end{bmatrix}$$

Varias Rotaciones

- Para encontrar la orientación con respecto a un punto, normalmente se utilizan varias rotaciones.
- Una de ellas es rotar α sobre el eje OX, después φ sobre el eje OY y finalmente θ sobre el eje OZ.
- A la matriz resultante se le conoce como matriz de transformación
- Es importante recordar el orden de las rotaciones, ya que no son conmutativas

$$\mathbf{T} = R(z,\theta)R(y,\phi)R(x,\alpha) =$$

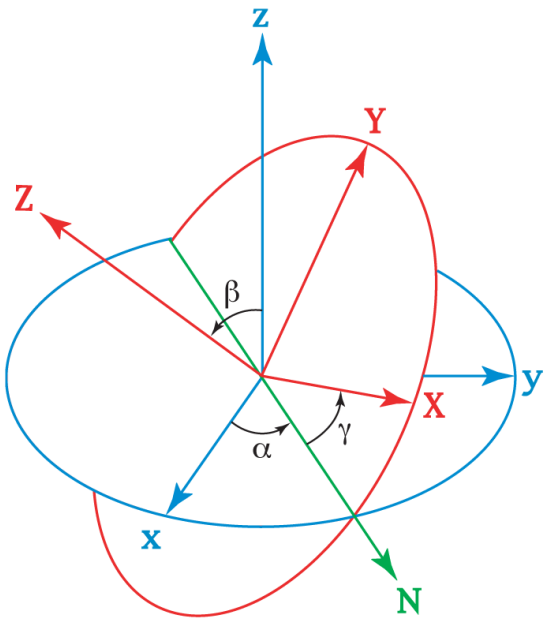
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\alpha) & -s(\alpha) \\ 0 & s(\alpha) & c(\alpha) \end{bmatrix} \begin{bmatrix} c(\phi) & 0 & s(\phi) \\ 0 & 1 & 0 \\ -s(\phi) & 0 & c(\phi) \end{bmatrix} \begin{bmatrix} c(\theta) & -s(\theta) & 0 \\ s(\theta) & c(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Varias Rotaciones

$$\mathbf{T} = \begin{bmatrix} c(\theta)c(\phi) & -s(\phi)c(\alpha)+c(\theta)s(\phi)s(\alpha) & -s(\theta)s(\alpha)+c(\theta)s(\phi)c(\alpha) \\ s(\theta)c(\phi) & c(\theta)c(\alpha)+s(\theta)s(\phi)s(\alpha) & -c(\theta)s(\alpha)+s(\theta)s(\phi)c(\alpha) \\ -s(\phi) & c(\phi)s(\alpha) & c(\phi)c(\alpha) \end{bmatrix}$$

Ángulos de Euler

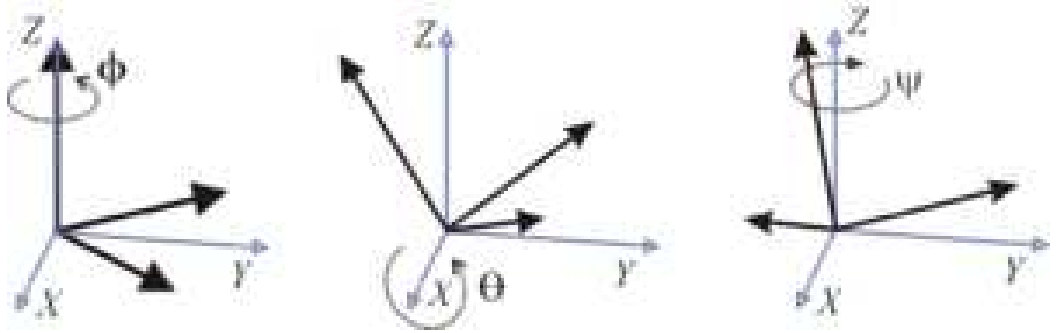
- Pare expresar las rotaciones y orientación se requieren 9 elementos.
- Una alternativa que solo usa 3 ángulos se le llaman ángulos de Euler.
- De lo ángulos de Euler, existen 3 posibilidades, estas son:
  - Ángulos de Euler ZXZ
  - Ángulos de Euler ZYZ
  - Roll, Pitch and Yaw (Alabeo, cabeceo guiñada) XYZ.



Ángulos de Euler ZXZ

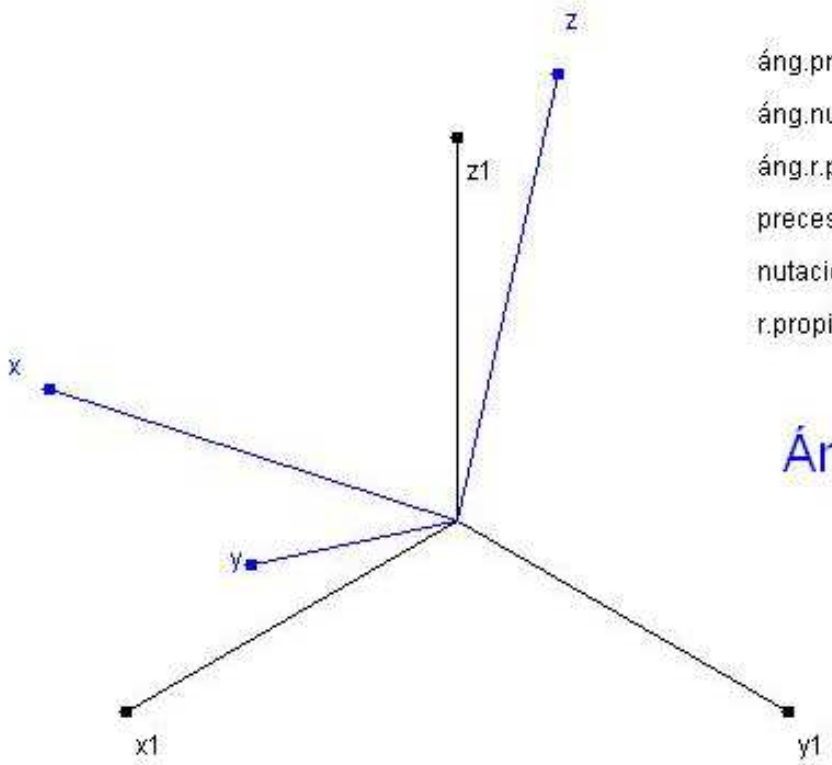
- Girar el sistema OUVW un ángulo φ (0° a 360°) con respecto ale eje OZ, convirtiéndose así en el OU'V'W'.
- Girar el sistema OU'V'W' un ángulo θ (0° a 180°) con respecto a OU', convirtiéndose así en el sistema OU"V"W".
- Girar el sistema OU"V"W" un ángulo ψ (0° a 360°) con respecto a OW", convirtiéndose finalmente en el OU'"V'"W'".
- $R(z,\psi) \rightarrow R(x,\theta) \rightarrow R(z,\phi)$

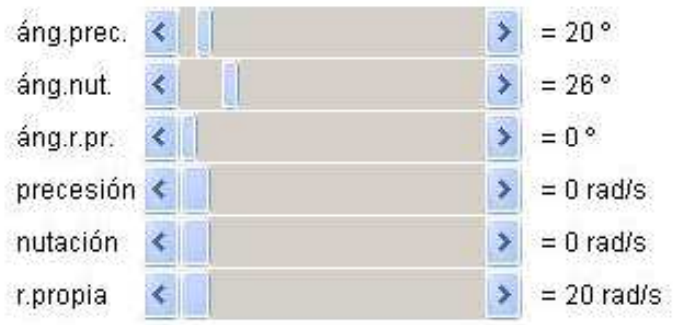
$$\vec{P}_{xyz} = R(z,\psi)R(x,\theta)R(z,\varphi)\vec{P}_{uvw}$$



Ángulos de Euler ZXZ

- http://mecfunnet.faii.etsii.upm.es/Xitami/webpages/euler.html

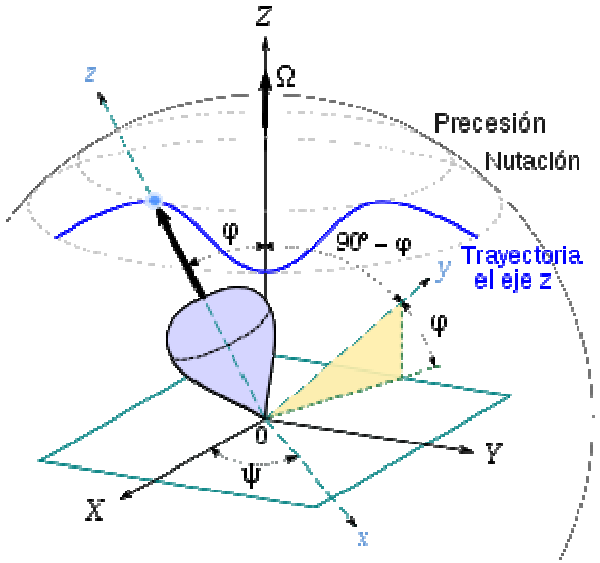




Ángulos y Rotaciones de Euler

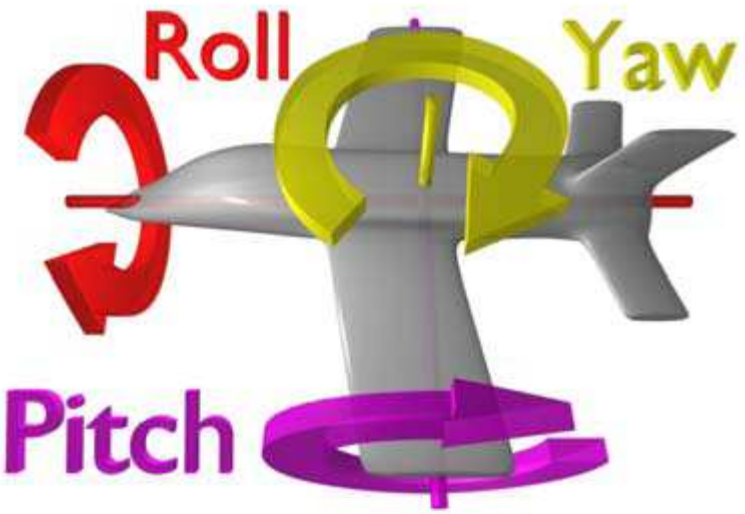
Ángulos de Euler ZYZ

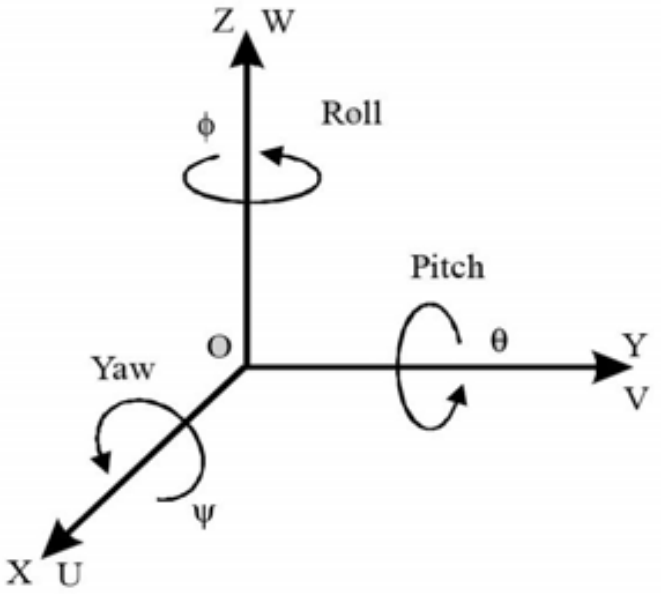
- Girar el sistema OUVW un ángulo φ (0° a 360°) con respecto ale eje OZ, convirtiéndose así en el OU'V'W'.
- Girar el sistema OU'V'W' un ángulo θ (0° a 180°) con respecto a OV', convirtiéndose así en el sistema OU"V"W".
- Girar el sistema OU"V"W" un ángulo ψ (0° a 360°) con respecto a OW", convirtiéndose finalmente en el OU'"V'"W'".
- $R(z,\psi) \rightarrow R(y,\theta) \rightarrow R(z,\phi)$



Roll, Pitch and Yaw (Alabeo, cabeceo, guiñada) XYZ

- Girar el sistema OUVW un ángulo φ con respecto ale eje OX (Guiñada)
- Girar el sistema OUVW un ángulo θ con respecto a OY (Cabeceo)
- Girar el sistema OUVW un ángulo ψ con respecto a OZ (Alabeo)





Ejemplo

- Encontrar la matriz de transformación con del punto Puvw=[1,2,3], con respecto al eje X,Y y Z (en ese orden) con los ángulos $\alpha = 90^{o}$, $\phi = 90^{a}$, $y\ \theta = 90^{a}$

$$\vec{P}_{xyz} = R(x,\alpha)R(y,\phi)R(z,\theta)\vec{P}_{uvw}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \\ 1 \end{bmatrix}$$

Coordenadas Homogéneas

- Las matrices de transformación homogénea se utilizan para:
  - Representar la posición y orientación de un sistema girado y trasladado con respecto a un sistema fijo.
  - Transformar un vector expresado en coordenadas movibles y su representación en un sistema fijo.
  - Rotar y trasladar un vector con respecto a un sistema fijo.
- Una matriz de transformación homogénea T es una matriz de dimensión 4x4
- Representan la transformación de un vector de coordenadas homogéneas de un sistema a otro.
- Esta compuesta por 4 términos: Escalamiento $w_{1\times1}$, traslación $P_{3\times1}$ rotación $R_{3\times3}$ y perspectiva $f_{1\times3}$.

Coordenadas Homogeneas

$$T = \begin{bmatrix} Rotación & Traslación \\ Perspectiva & Escala \end{bmatrix} = \begin{bmatrix} R_{3\times3} & P_{3\times1} \\ f_{1\times3} & w_{1\times1} \end{bmatrix}$$

- Para robótica $f_{1\times3} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ y $w_{1\times1} = 0$

- Para encontrar el punto con respecto al sistema fijo, se obtiene con:

$$\mathbf{p}_{xyz} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} p_u \\ p_v \\ p_w \\ 1 \end{bmatrix}$$

Traslación

- Supóngase un sistema O'UVW que se encuentra trasladado con respecto al sistema OXYZ.

$$p = p_x i + p_y j + p_z k$$

- La matriz T de traslación esta dada por:

$$T(p) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- El punto referenciado con respecto al eje OXYZ es

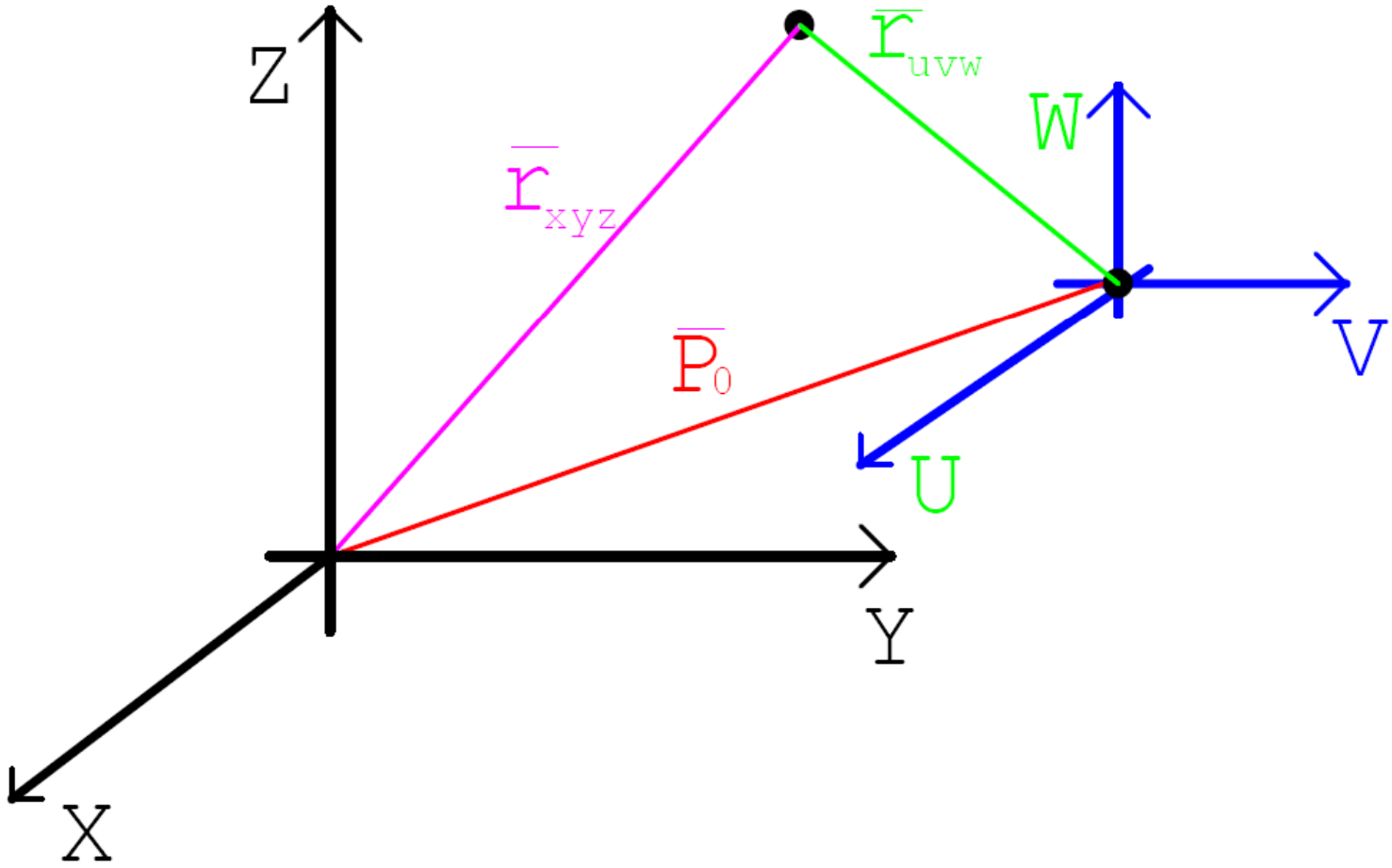
$$r_{xyz} = T(p) r_{uvw}$$

$$r_{xyz} = \left[r_x, r_y, r_z \right]^T$$

- El punto referenciado con respecto al eje OXYZ es

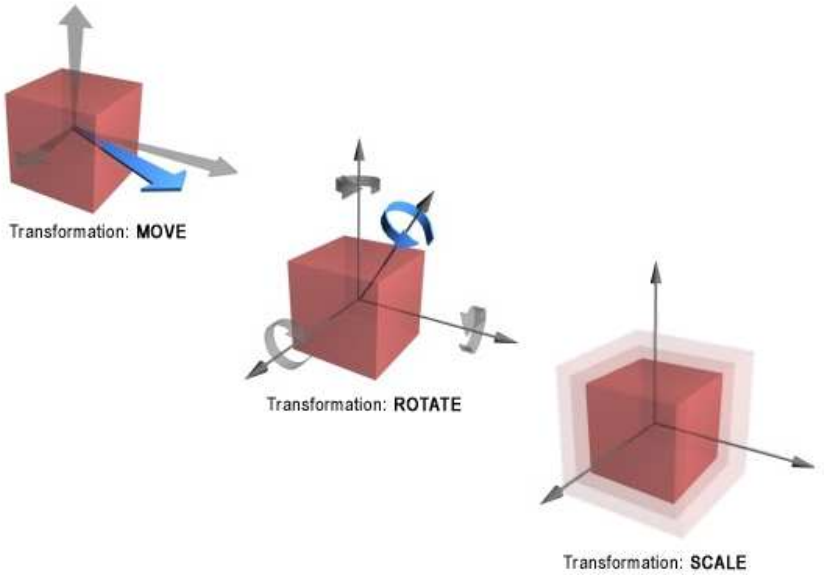
$$r_{uvw} = \left[r_u, r_v, r_w \right]^T = r_u i_u + r_v j_v + r_w j_w$$

Traslación



Rotación

- Cuando el sistema O'UVW solo esta rotado con respecto a OXYZ.
- Las matrices de rotación son las mismas que las que se vieron en la sección anterior.
- Existen 3 rotaciones, con respecto a OX, OY y OZ. Estas son: $T(x,\alpha)$, $T(y,\phi)$ y $T(z,\theta)$



Rotação

$$T(x,\alpha)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(\alpha) & -sen(\alpha) & 0\\ 0 & sen(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$T(y,\phi)=\begin{bmatrix}\cos(\phi) & 0 & sen(\phi) & 0\\ 0 & 1 & 0 & 0\\ -sen(\phi) & 0 & \cos(\phi) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$T(z,\theta)=\begin{bmatrix}\cos(\theta) & -sen(\theta) & 0 & 0\\ sen(\theta) & \cos(\theta) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

Ejemplo

- Ejemplo: Sistema girado 90° con respecto al eje OZ, encontrar r_{xyz} si $r_{uvw} = [4,8,12]^T$.

- Solución: $r_{xyz} = [-8,4,12]^T$

Traslación y Rotación

- Es la matriz homogenes que se consigue después de trasladar y posteriormente rotar con respecto a uno de los ejes fijos.
- La matriz de transformaciones homogénea esta dada por:

$$T(p,(z,\theta)) = T(p)T(z,\theta) =$$

$$= \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & 0 \\ sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & p_x \\ sen(\theta) & \cos(\theta) & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Traslación y Rotación

$$T(p,(y,\phi)) = T(p)T(y,\phi) = \begin{bmatrix} \cos(\phi) & 0 & sen(\phi) & p_x \\ 0 & 1 & 0 & p_y \\ -sen(\phi) & 0 & \cos(\phi) & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T(p,(x,\alpha)) = T(p)T(z,\theta) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & \cos(\alpha) & -sen(\alpha) & p_y \\ 0 & sen(\alpha) & \cos(\alpha) & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotación y Traslación

- Es el resultado de rotar con respecto a un eje fijo y posteriormente trasladar.
- Son 3 las matrices de transformación homogénea. Con respecto al eje z es:

$$T((z,\theta),p) = T(z,\theta)T(p) = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & 0 \\ sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & p_x\cos(\theta) - p_y sen(\theta) \\ sen(\theta) & \cos(\theta) & 0 & p_x sen(\theta) - p_y\cos(\theta) \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ejemplos de Matlab (Rotaciones)

```
%Programa que validar operaciones de rotación 6/Abril/2010
clear; close
%Genera un cuadro con rotaciones
P1=[1.1;1.1;2;1];
P2=rotz(pi/2)*P1;
X=[P1(1),P2(1)];Y=[P1(2),P2(2)]; Z=[P1(3),P2(3)];
L1=line(X,Y,Z,'Color','r','LineWidth',4)
xlabel('x'); ylabel('y'); zlabel('z')
view(3)
grid
pause
 P1=P2;
P2=rotz(pi/2)*P1;
X=[P1(1),P2(1)];Y=[P1(2),P2(2)]; Z=[P1(3),P2(3)];
L2=line(X,Y,Z,'Color','g','LineWidth',4)
Pause
P1=P2;
P2=rotz(pi/2)*P1;
X=[P1(1),P2(1)];Y=[P1(2),P2(2)]; Z=[P1(3),P2(3)];
L3=line(X,Y,Z,'Color','b','LineWidth',4)
pause
P1=P2;
P2=rotz(pi/2)*P1;
X=[P1(1),P2(1)];Y=[P1(2),P2(2)]; Z=[P1(3),P2(3)];
L4=line(X,Y,Z,'Color','c','LineWidth',4)
pause
%Modifica un objeto
Z=get(L4,'zdata');
Z=Z+1;
set(L4,'zdata',Z)
pause
Z=Z-1;
set(L4,'zdata',Z)
%Objeto Cuadro. Lo manipula
Cuadro=[L1;L2;L3;L4];
manipulaObjeto(Cuadro,rotz(pi/4));
for r=0:pi/10:2*pi
   manipulaObjeto(Cuadro,rotz(r));
end
```

Función: manipulaObjeto

```
function Objeto = manipulaObjeto(Dato1,Transfor)
%Transfor=rotz(pi/4);
%Dato1=Cuadro
N=size(Dato1);
for i=1:N
   Li=Dato1(i);
   Xi=get(Li,'xdata');
  Yi=get(Li,'ydata');
   Zi=get(Li,'zdata');
   Pi1=[Xi(1);Yi(1);Zi(1);1];
   Pi2=[Xi(2);Yi(2);Zi(2);1];
   Pi1=Transfor*Pi1;
   Pi2=Transfor*Pi2;
   Xi=[Pi1(1);Pi2(1)];
  Yi=[Pi1(2);Pi2(2)];
   Zi=[Pi1(3);Pi2(3)];
   set(Li,'xdata',Xi, 'ydata',Yi,'zdata',Zi);
   pause(0.2)
end
Objeto=1;
```

Curvas paramétricas y línea recta

```
%Este Programa genera la línea recta y un circulo
%4 de Mayo 2010
clear; close
N=50;
dt=4*pi/(N);
i=1;
for t=0:dt:4*pi
   X(i)=cos(t);
   Y(i)=sin(t);
   Tiempo(i)=t;
   i=i+1;
   subplot(3,1,1), plot(Tiempo,X)
   grid
   Subplot(3,1,2), plot(Tiempo,Y)
   grid
   Subplot(3,1,3), plot(X,Y)
   grid
   pause(0.1)
end
pause
```

```
P1=[1;2;3];
P2=-P1;
N=100;
dt=1/N
i=1;
Close
for t=0:dt:1
   temp=(P2-P1)*t+P1;
   X2(i)=temp(1);
   Y2(i)=temp(2);
   Z2(i)=temp(3);
   Tiempo2(i)=t;
   i=i+1;
   subplot(2,2,1), plot(Tiempo2,X2)
   grid
   subplot(2,2,2), plot(Tiempo2,Y2)
   grid
   subplot(2,2,3), plot(Tiempo2,Z2)
   grid
   subplot(2,2,4), plot3(X2,Y2,Z2)
   grid
   pause(0.1)
end
```

Ejemplo de Aplicación

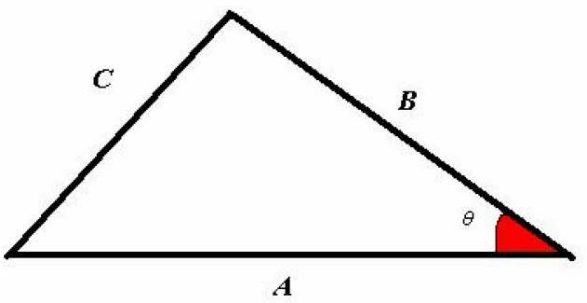
Solución geométrica:

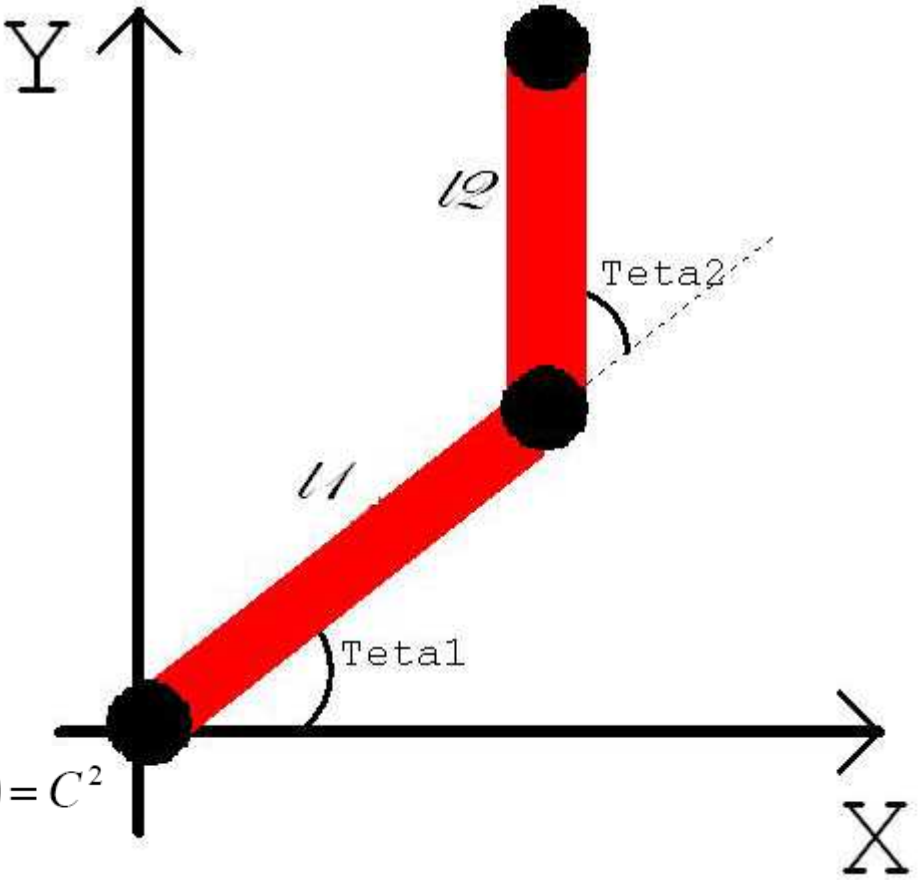
La coordenada en x es:

$$p_x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

La coordenada en y es:

$$p_y = l_1 sen(\theta_1) + l_2 sen(\theta_1 + \theta_2)$$





Ley de los cosenos : $A^2 + B^2 - 2AB\cos(\theta) = C^2$

$$p_y = l_1 sen(\theta_1) + l_2 sen(\theta_1 + \theta_2)$$

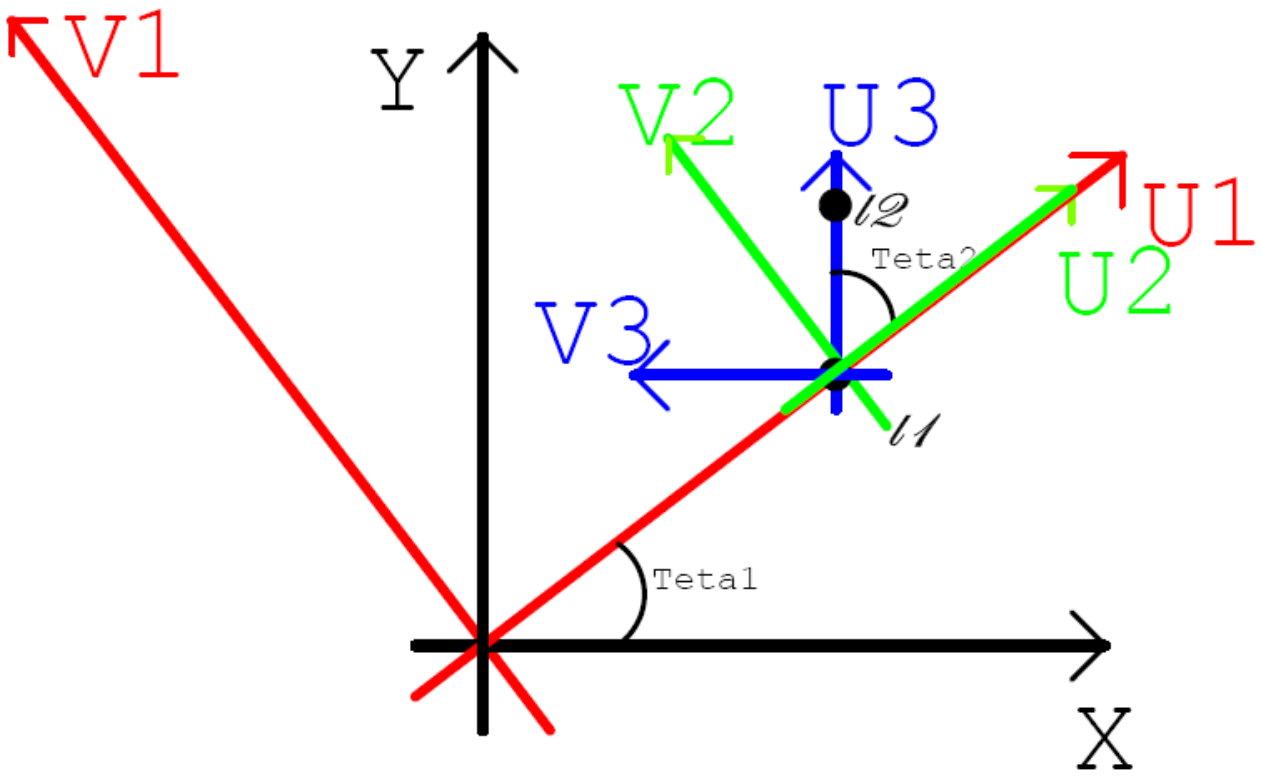
Solución por Matrices de Rotación y Traslación

$$P_3 = \begin{bmatrix} l_2 & 0 & 0 \end{bmatrix}^T$$

$$P_{32} = T\left(R\left(z,\theta_2\right)\right)P_3 = T_3 P_3$$

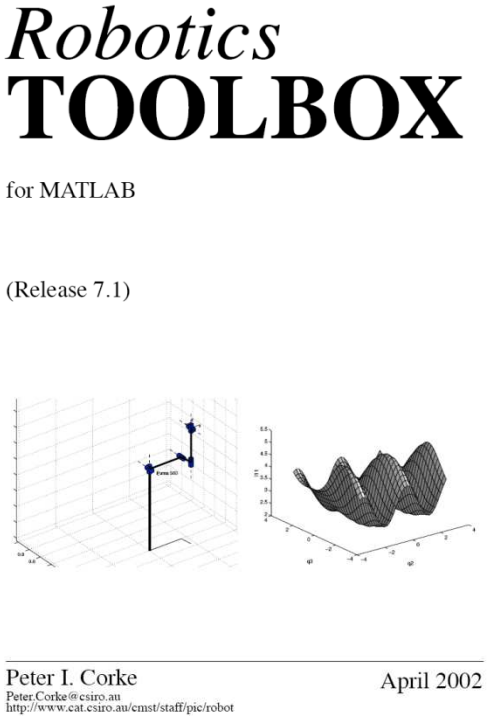
$$P_{31} = T\left(R\left(0\right),P_{01}\right)P_{32} = T_2 P_{32}$$

$$P_{3xy} = T\left(R\left(z,\theta_1\right)\right)P_{31} = T_1 P_{31}$$



Toolbox de Robótica

- Provee varias funciones que son utiles en robótica incluyendo cinemática, dinámica y generación de trayectoria
- Suministra objetos en Matlab que permite generar y manipular cualquier cadena serial de eslabones encapsulados en funciones de Matlab
- El toolbox también tiene funciones para manipular y convertir tipos de datos vectoriales, transformaciones homogéneas y cuaterniones para representar la posición y orientación en 3 dimensiones
- Tiene un manual de ayuda en PDF con las sus funciones
- Fue desarrollado por Peter I. Corke, investigador de Queensland Center for Advanced Technologies en Australia



Instalación del Toolbox

- El toolbox viene en una carpeta llamada robot
- Copiarale en la carpeta de toolbox de matlab:
  - C:\Archivos de programa\MATLAB\R2009a\toolbox\robot
- Dar de lata la carpeta en el Matlab.
  - Dentro de Matlab ir a: File→Set→ Path
  - Seleccionar: La carpeta de robot que esta en la trayectoria especificada
  - Presionar el boton: Add Folder
  - Queda listor
- Para probar que ha quedado bien instalado dar en la línea de comandos: >> rotx(0)
- No debe marcar errores y regresa una matriz de 4x4

Funciones del Toolbox

Homogeneous Transforms	
`eul2tr`	Euler angle to homogeneous transform
`oa2tr`	orientation and approach vector to homogeneous transform
`rotvec`	homogeneous transform for rotation about arbitrary vector
`rotx`	homogeneous transform for rotation about X-axis
`roty`	homogeneous transform for rotation about Y-axis
`rotz`	homogeneous transform for rotation about Z-axis
`rpy2tr`	Roll/pitch/yaw angles to homogeneous transform
`tr2eul`	homogeneous transform to Euler angles
`tr2rot`	homogeneous transform to rotation submatrix
`tr2rpy`	homogeneous transform to roll/pitch/yaw angles
`transl`	set or extract the translational component of a homogeneous transform
`trnorm`	normalize a homogeneous transform

Funciones del Toolbox

Rotación en X

```
>> rotx(pi/2)

ans =

    1.0000         0         0         0
         0    0.0000   -1.0000         0
         0    1.0000    0.0000         0
         0         0         0    1.0000

>>
```

Traslación

```
>> transl([1 2 3])

ans =

     1     0     0     1
     0     1     0     2
     0     0     1     3
     0     0     0     1

>>
```

Funciones del Toolbox

Trajectory Generation	
`ctraj`	Cartesian trajectory
`jtraj`	joint space trajectory
`trinterp`	interpolate homogeneous transforms

Manipulator Models	
`link`	construct a robot link object
`nofriction`	remove friction from a robot object
`perturb`	randomly modify some dynamic parameters
`puma560`	Puma 560 data
`puma560akb`	Puma 560 data (modified Denavit-Hartenberg)
`robot`	construct a robot object
`showlink`	show link/robot data in detail
`stanford`	Stanford arm data
`twolink`	simple 2-link example

Funciones del Toolbox

Kinematics	
`diff2tr`	differential motion vector to transform
`fkine`	compute forward kinematics
`ftrans`	transform force/moment
`ikine`	compute inverse kinematics
`ikine560`	compute inverse kinematics for Puma 560 like arm
`jacob0`	compute Jacobian in base coordinate frame
`jacobn`	compute Jacobian in end-effector coordinate frame
`tr2diff`	homogeneous transform to differential motion vector
`tr2jac`	homogeneous transform to Jacobian

Graphics	
`drivebot`	drive a graphical robot
`plot`	plot/animate robot

Generación de un Robot

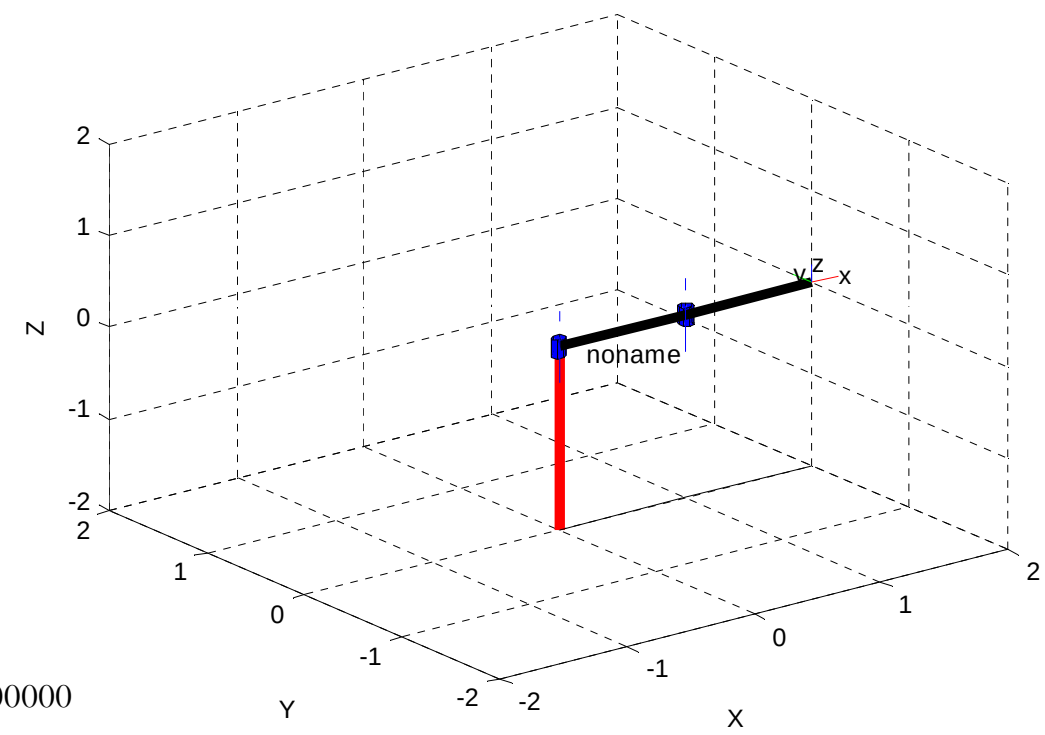
```
>> L1=link([0 1 0 0 0],'standard');
>> L2=link([0 1 0 0 0], 'standard');
>> r=robot({L1 L2});
>> L1=link([0 1 0 0 0],'standard');
>> L2=link([0 1 0 0 0], 'standard');
>> r=robot({L1 L2})

r =

noname (2 axis, RR)
                grav = [0.00 0.00 9.81]
                standard D&H parameters

 alpha          A              theta          D
                R/P
0.000000    1.000000    0.000000    0.000000
                R           (std)
0.000000    1.000000    0.000000    0.000000
                R           (std)

>> plot(r, [0 0])
```



Manipulación de un Cubo

```
%Este Ejemplo manipula un cubo en 3 dimensiones
clear; close;
largo=4; Color='r'; Ancho=4;
Cubo=CreaCubo(largo,Color,Ancho)
xlabel('x'); ylabel('y'); zlabel('z');
view(3)
grid
v=[-largo,largo,-largo,largo,-largo,largo];
axis(10*v);
hold on
 rango1=1;
mmm=1
while mmm
   r=LimitaRango(rand,rango1)*pi;
   r1=LimitaRango(rand,rango1)*pi;
   r2=LimitaRango(rand,rango1)*pi;
   rx=LimitaRango(rand,rango1)*largo/10;
   ry=LimitaRango(rand,rango1)*largo/10;
   rz=LimitaRango(rand,rango1)*largo/10;
   Pxyz=[rx,ry,rz];
   RotaCubo(Cubo,transl(Pxyz)*rotx(r)*roty(r1)*rotz(r2));
   pause(0.3)
end
```

```
 pause
for r=0:pi/6:2*pi
   for r1=0:pi/6:2*pi
      for r2=0:pi/10:2*pi
         RotaCubo(Cubo,rotx(r)*roty(r1)*rotz(r2));
         pause(0.2)
      end
   end
end
```

Manipulación de un Cubo

Función CreaCubo.m

```
%Esta función crea un cubo
function Cubo= CreaCubo(largo, Color, Ancho)
P1=[largo,-largo,-largo,1]';
Cuadro1=CreaCuadro2(P1,rotz(pi/2),Color,Ancho);

P2=P1;
P2(3)=P2(3)+2*largo;
Cuadro2=CreaCuadro2(P2,rotz(pi/2),'r',4);
pause(0.2)

for i=1:4
   La=Cuadro1(i);
   Lb=Cuadro2(i);
   Xa=get(La,'xdata'); Xb=get(Lb,'xdata');
   Ya=get(La,'ydata'); Yb=get(Lb,'ydata');
   Za=get(La,'zdata'); Zb=get(Lb,'zdata');
   P1=[Xa(1),Ya(1),Za(1),1]'; P2=[Xb(1),Yb(1),Zb(1),1]';
   Li(i)=CreaLinea(P1,P2,'b',Ancho);

end
Cubo=[Cuadro1,Cuadro2,Li'];
```

Función CreaCuadro2.m

```
function Cuadro = CreaCuadro2(P1,Transfor,Color,Ancho)
P2=Transfor*P1;
L1=CreaLinea(P1,P2,Color,Ancho);
P1=P2;
P2=Transfor*P1;
L2=CreaLinea(P1,P2,Color,Ancho);
P1=P2;
P2=Transfor*P1;
L3=CreaLinea(P1,P2,Color,Ancho);
P1=P2;
P2=Transfor*P1;
L4=CreaLinea(P1,P2,Color,Ancho);

Cuadro=[L1;L2;L3;L4];
```

Manipulación de un Cubo

Función CreaLinea.m

- function Linea = CreaLinea(P1,P2,Color,Ancho)
- X=[P1(1),P2(1)];
  Y=[P1(2),P2(2)];
  Z=[P1(3),P2(3)];
- Linea=line(X,Y,Z,'Color',Color,'LineWidth',Ancho)

Función RotaCubo.m

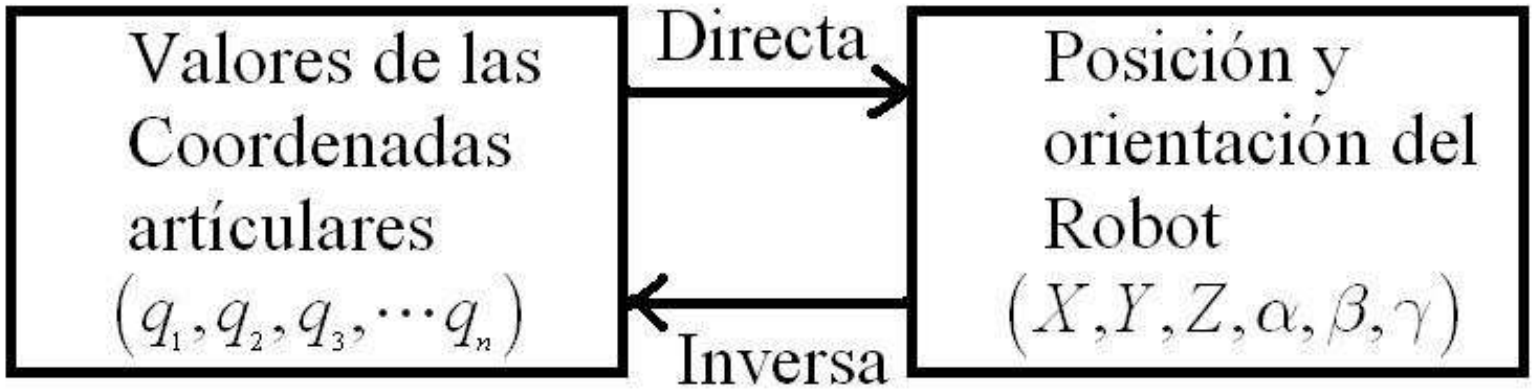
```
function dato = RotaCubo(Cubo,Transform)

%Transform=rotz(pi/4);
for i=1:3
   for j=1:4
      La=Cubo(j,i);
Xa=get(La,'xdata');Ya=get(La,'ydata');Za=get(La,'zdata');
      P1=[Xa(1),Ya(1),Za(1),1]';
P2=[Xa(2),Ya(2),Za(2),1]';
      P1=Transform*P1; P2=Transform*P2;
Xa=[P1(1),P2(1)];Ya=[P1(2),P2(2)];Za=[P1(3),P2(3)];
      set(La,'xdata',Xa,'ydata',Ya,'zdata',Za);
      %pause(0.1)
   end
end
```

Cinemática

- La cinemáica del robot estudia el movimiento con respecto a un sistema de referencia
- Existen 2 problemas fundamentales, cinemática directa e inversa.
- La cinemática directa consiste en determinar la posición y orientación del extremo final del robot en función de las coordenadas articulares
- La cinemática inversa determina las coordenadas articulares en función de la posición final del robot.

Cinemática



- La cinemática del robot trata también de encontrar las relaciones entre las velocidades de cada articulación y la del extremo. A esto se le conoce como modelo diferencial.
- Denavit y Hartenberg propusieron un método sistemático para representar la geometría espacial de los elementos de una cadena cinemática de un robot.

Problema Cinemático Directo

- Un robot se puede considerar como una cadena cinemática formada por eslabones unidos por articulaciones
- Se establece un sistema de referencia fijo solidario a la base
- Se describe la localización de cada uno de los eslabones con respecto a dicho sistema de referencia
- El problema cinemático directo se reduce a encontrar la matriz de transformación T
- Es función de las coordenadas articulares.
- Para sistemas de hasta 3 grados de libertad se puede usar un método trigonométrico

$$x = f_x(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$y = f_y(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$z = f_z(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\alpha = f_\alpha(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\beta = f_\beta(q_1, q_2, q_3, q_4, q_5, q_6)$$
$$\gamma = f_\gamma(q_1, q_2, q_3, q_4, q_5, q_6)$$

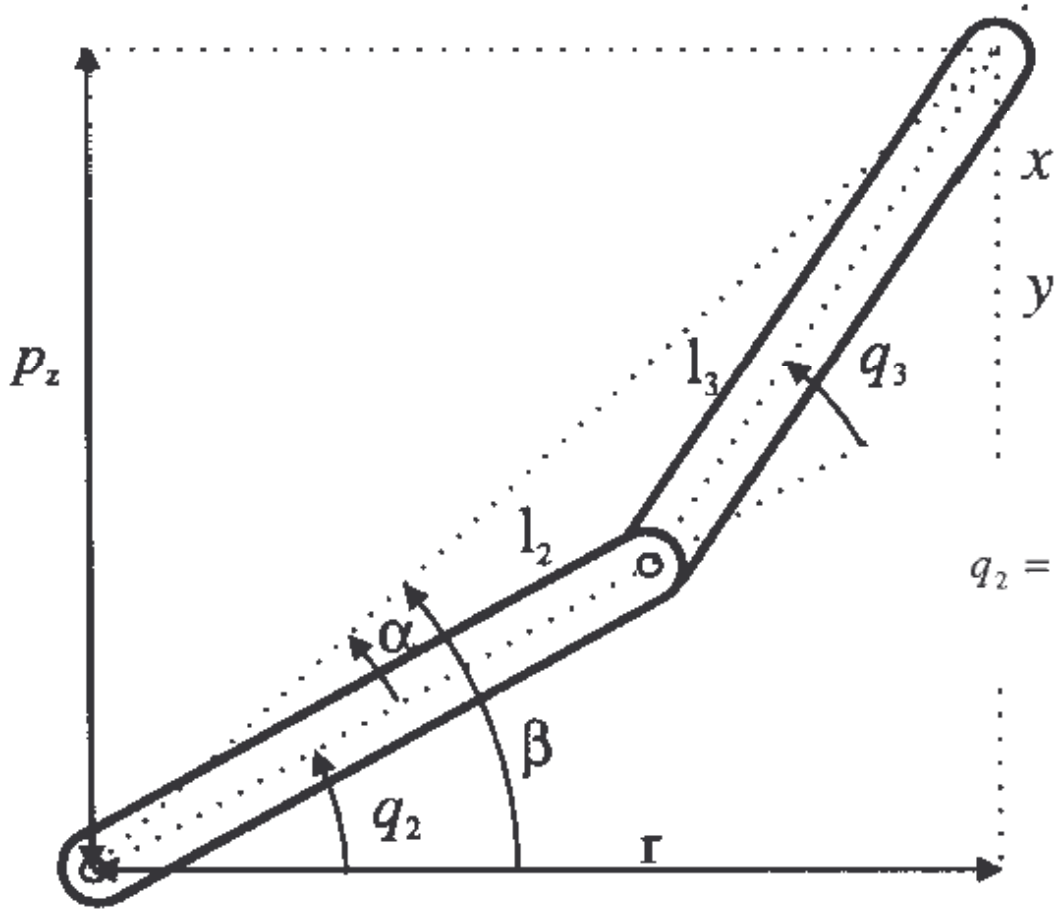
Problema Cinemático Inverso

- El problema cinemático inverso consiste en encontrar los valores que debe adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial
- Existen mas de una solución
- Se puede despejar de la matriz de transformación homogénea
- Se puede obtener usando trigonometría
- Lo adecuado es encontrar una solución cerrada, es decir encontrar una relación matemática explícita de la forma

$$q_k = f_k(x, y, z, \alpha, \beta, \gamma)$$

$$k = 1 \ldots n \qquad \text{(GDL)}$$

Codo Abajo



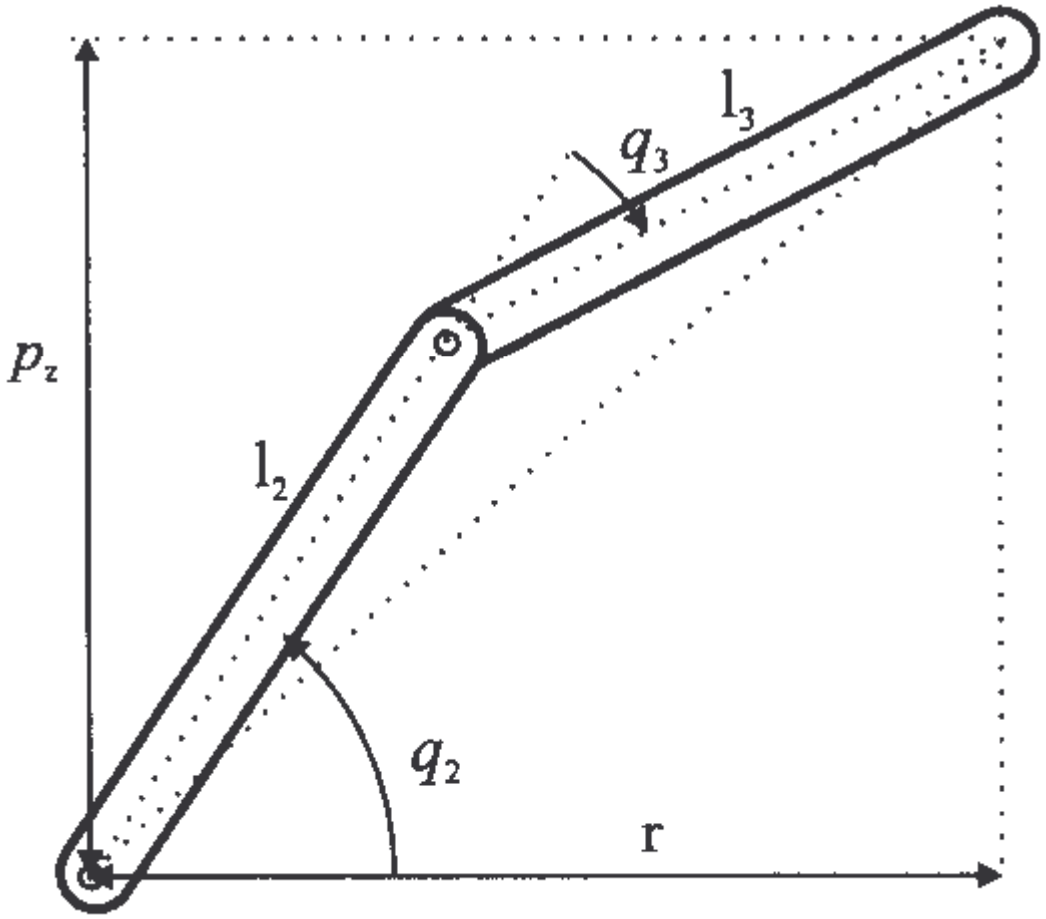
$$x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$y = l_1 \operatorname{sen} q_1 + l_2 \operatorname{sen}(q_1 + q_2)$$

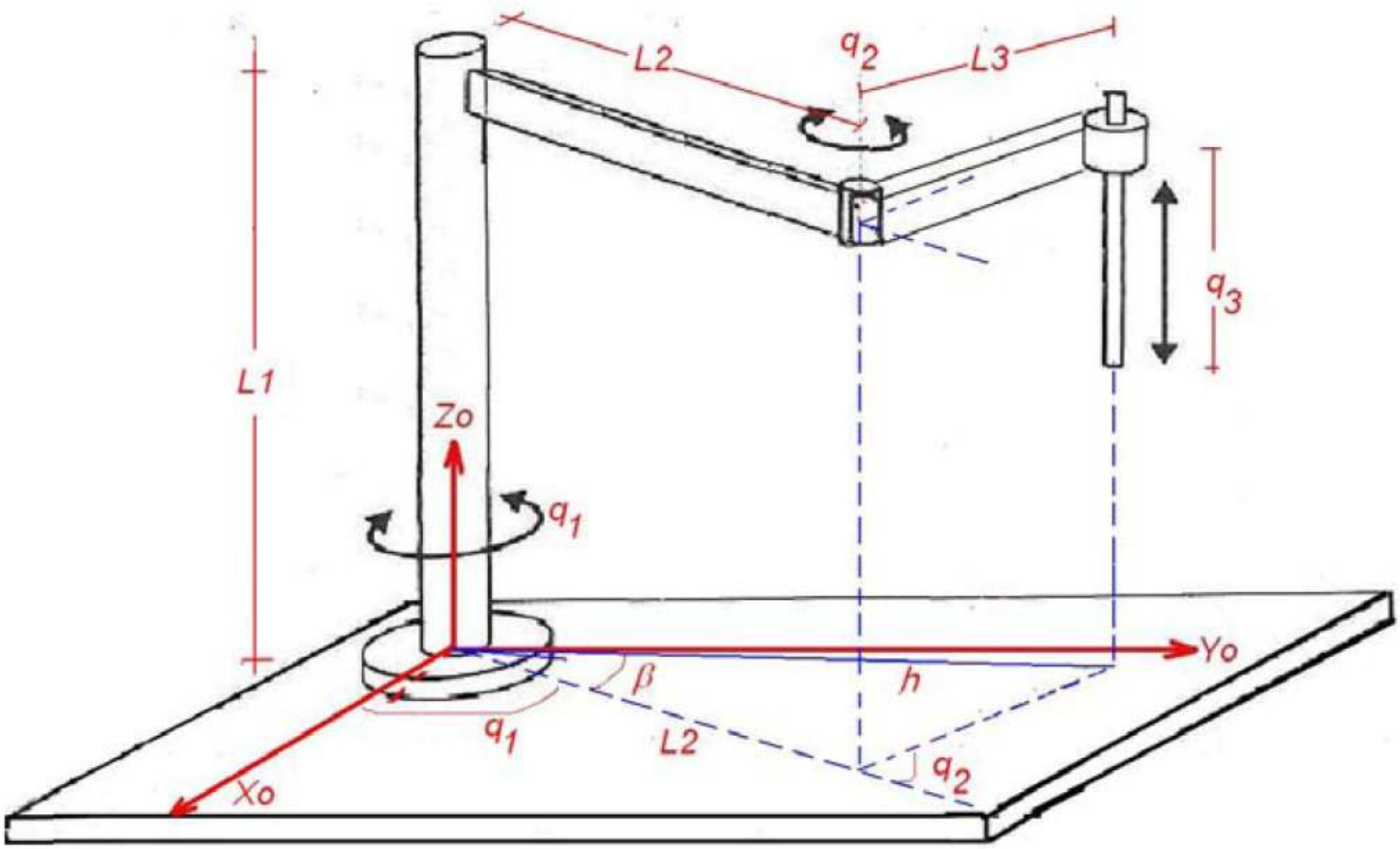
$$q_2 = \operatorname{arctg}\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) - \operatorname{arctg}\left(\frac{l_3 \operatorname{sen} q_3}{l_2 + l_3 \cos q_3}\right)$$

$$q_3 = \operatorname{arctg}\left(\frac{\pm\sqrt{1-\cos^2 q_3}}{\cos q_3}\right)$$

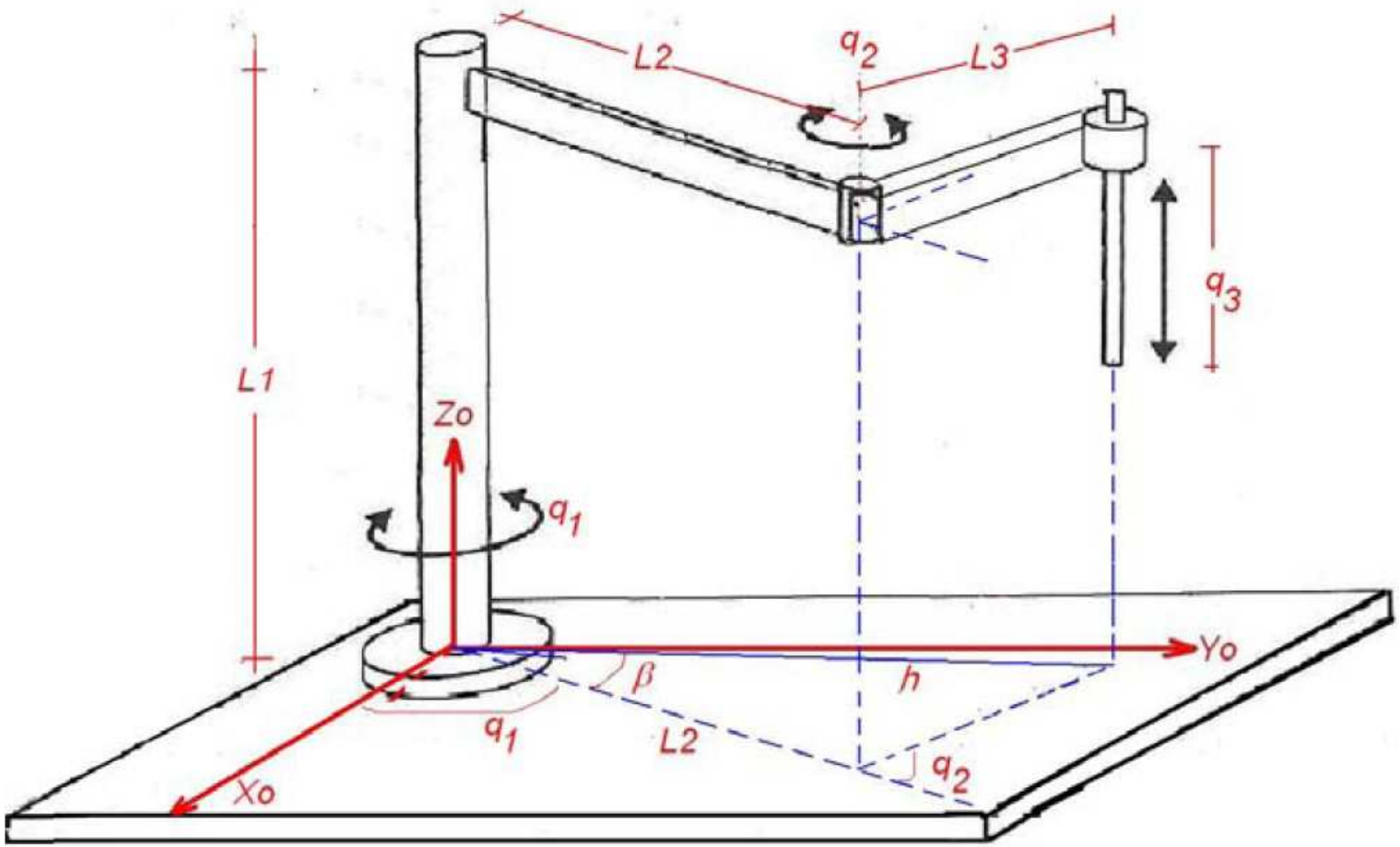
Codo Arriba



Robot Scara



Robot Scara



Robot Scara

Cinemática directa

$$h^2 = L_2^2 + L_3^2 - 2\cos(180^\circ - q_2)$$
$$= L_2^2 + L_3^2 + 2\cos(q_2)$$

$$L_3^2 = L_2^2 + h^2 - 2L_2 h\cos(\beta)$$
$$\Rightarrow$$
$$\beta = \cos^{-1}\left(\frac{L_2^2 + h^2 - L_3^2}{2L_2 h}\right)$$

$$\begin{cases} x = h\cos(\beta + q_1) \\ y = h\sin(\beta + q_1) \\ z = L_1 - q_3 \end{cases}$$

Cinemática Inversa

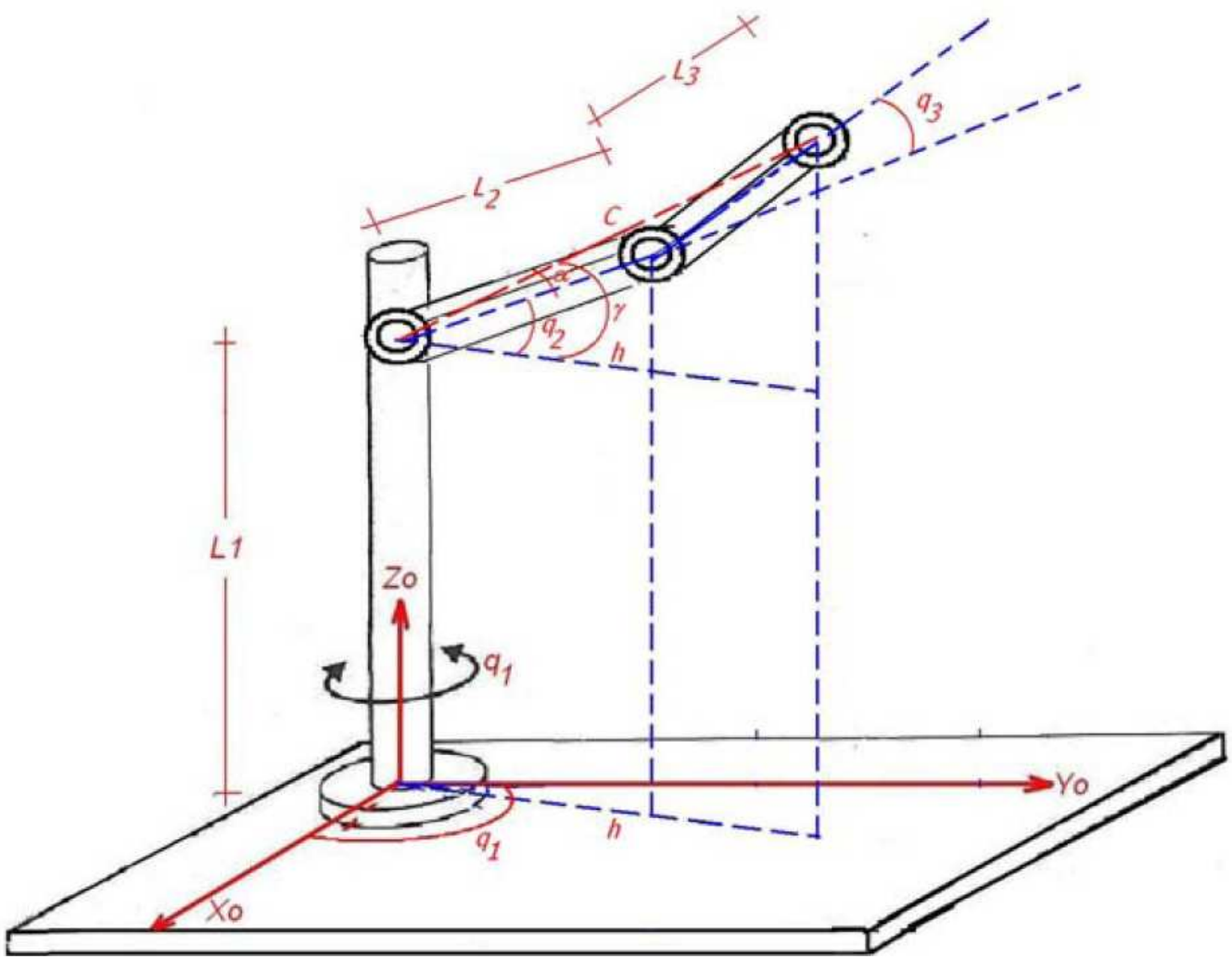
$$h = \sqrt{x^2 + y^2}$$

$$L_3^2 = L_2^2 + h^2 - 2L_2 h\cos(\beta)$$
$$\Rightarrow$$
$$\beta = \cos^{-1}\left(\frac{L_2^2 + h^2 - L_3^2}{2L_2 h}\right)$$

$$\gamma = \beta + q_1 = \tan^{-1}\left(\frac{y}{x}\right)$$

$$\begin{cases} q_1 = \gamma - \beta \\ q_2 = \cos^{-1}\left(\dfrac{h^2 - L_2^2 - L_3^2}{2L_2 L_3}\right) \\ q_3 = L_1 - z \end{cases}$$

Robot Antropomórfico



Robot Antropomórfico

Cinemática directa

$$h = L_2 \cos(q_2) + L_3 \cos(q_2 + q_3)$$

$$c^2 = L_2^2 + L_3^2 + 2L_2 L_3 \cos(q_3)$$

$$\gamma = \alpha + q_2$$

$$\begin{cases} x = h\cos(q_1) \\ y = h\sin(q_1) \\ z = L_1 + L_2 \sin(q_2) + L_3 \sin(q_2 + q_3) \end{cases}$$

Cinemática Inversa

$$h = \sqrt{x^2 + y^2}$$

$$c^2 = x^2 + y^2 + (z - L_1)^2$$

$$\alpha = \cos^{-1}\left(\frac{L_2^2 + c^2 - L_3^2}{2L_2 c}\right)$$

$$\gamma = \alpha + q_2 = \tan^{-1}\left(\frac{z - L_1}{x}\right)$$

$$\begin{cases} q_1 = \tan^{-1}\left(\frac{y}{x}\right) \\ q_2 = \gamma - \alpha = \tan^{-1}\left(\frac{z - L_1}{h}\right) - \alpha \\ q_3 = \cos^{-1}\left(\frac{c^2 - L_2^2 - L_3^2}{2L_2 L_3}\right) \end{cases}$$

Resolución del problema cinemático directo

- Mediante matrices de transformación homogénea
  - A cada eslabón se le asocia un sistema de referencia solidario
  - Es posible representar las traslaciones y rotaciones relativas entre distintos eslabones
  - La matriz ${}^{i-1}A_i$ representa la posición y orientación relativa entre los sistemas asociados a dos eslabones consecutivos del robot.

$$ {}^0A_3 = {}^0A_1\,{}^1A_2\,{}^2A_3 $$

$$ T = {}^0A_6 = {}^0A_1\,{}^1A_2\,{}^2A_3\,{}^3A_4\,{}^4A_5\,{}^5A_6 $$

  - Existen métodos sistemáticos para situar los sistemas de coordenadas asociados a cada eslabón y obtener la cadena cinemática del robot. Modelado de Denavit-Hartenberg (D-H).

Resolución del problema cinemático directo

- Sistematiza la obtención de las matrices entre eslabones
- Los sistemas de coordenadas asociados a cada eslabón deben se escogidos con condiciones concretas
- Con 4 movimientos o transformaciones simples (rotaciones y traslaciones). Los cuales implican un parámetro D-H.

$$(q_i, d_i, a_i, a_i)$$

-

$$^{i-1}A = T(z, q_i)T([0,0, d_i])T([a_i, 0,0])T(x, a_i)$$

Resolución del problema cinemático directo

$$
{}^{i-1}A = \begin{bmatrix} cq_i & -sq_i & 0 & 0 \\ sq_i & cq_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & ca_i & -sa_i & 0 \\ 0 & sa_i & ca_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{i-1}A = \begin{bmatrix} cq_i & -ca_i sq_i & sa_i cq_i & a_i cq_i \\ sq_i & ca_i cq_i & -sa_i cq_i & a_i sq_i \\ 0 & sa_i & ca_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Método de Denavit – Hartenberg (D-H)

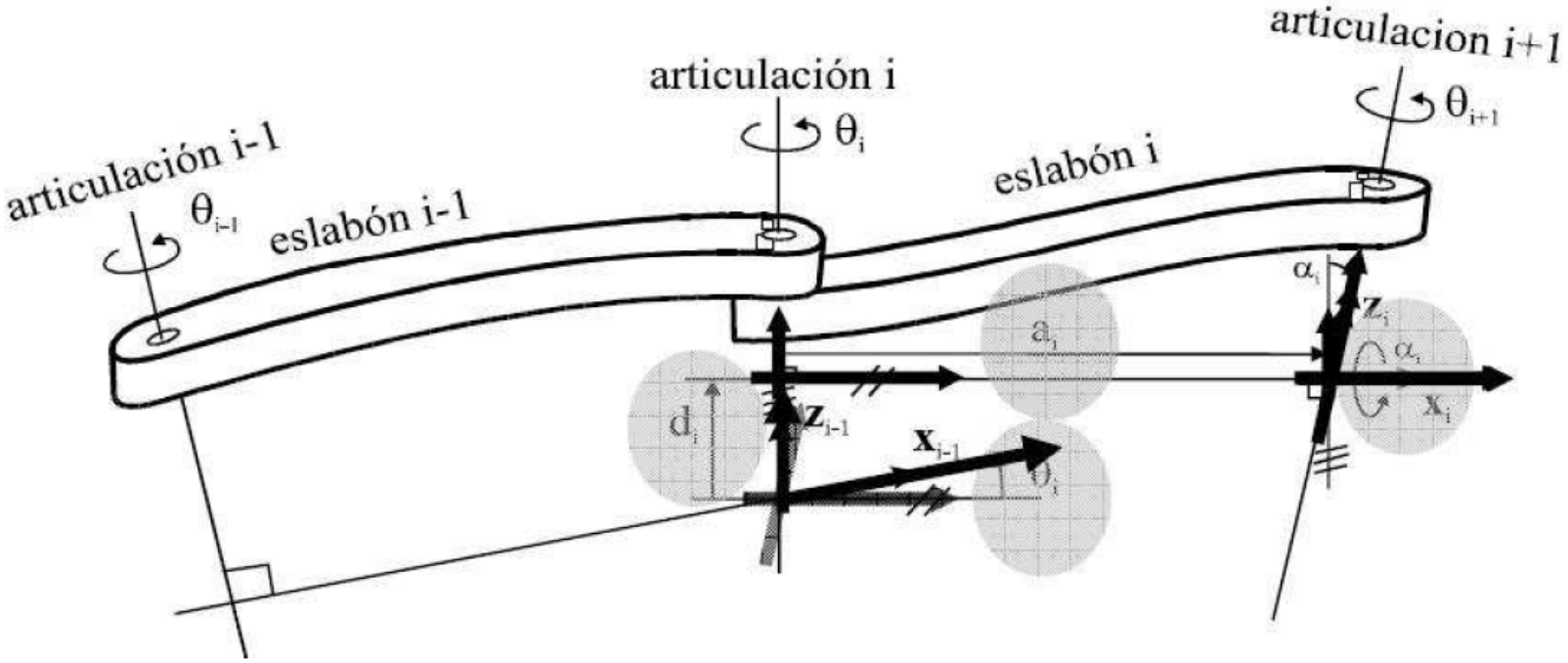
- Permite el paso de un eslabón al siguiente mediante 4 matrices de transformación básicas, que dependen exclusivamente de las características constructivas del robot.
- Las transformaciones básicas que relacionan el sistema de referencia del elemento i con el sistema i-1 son:
  - Rotación θ_i alrededor del eje Z_{i-1}
  - Traslación d_i a lo largo del eje Z_{i-1}
  - Traslación a_i a lo largo de X_i
  - Rotación α_i alrededor del eje X_i

Método de Denavit – Hartenberg (D-H)

$$^{i-1}A = T(z, \theta_i)T([0,0, d_i])T([a_i, 0,0])T(x, \alpha_i)$$

$$^{i-1}A = \begin{bmatrix} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Método de Denavit – Hartenberg (D-H)



Algoritmo de Denavit – Hartenberg I.

1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la cadena) y acabando con n (último eslabón móvil). Se numera como eslabón 0 a la base fija del robot.
2. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.
3. Localizar el eje de cada articulación. Si ésta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.
4. Para i de 0 a n-1 situar el eje Z_i sobre el eje de la articulación i+1.
5. Situar el origen del sistema de la base $\{S_0\}$ en cualquier punto del eje Z_0. Los ejes X_0 e Y_0 se situarán de modo que formen un sistema dextrógiro con Z_0.
6. Para i de 1 a n-1, situar el sistema $\{S_i\}$ (solidario al eslabón i) en la intersección del eje Z_i con la línea normal común a Z_{i-1} y Z_i. Si ambos ejes se cortasen se situaría $\{S_i\}$ en el punto de corte. Si fuesen paralelos $\{S_i\}$ se situaría en la articulación i+1.
7. Para i de 1 a a n-1, situar X_i en la línea normal común a Z_{i-1} y Z_i.

Algoritmo de Denavit – Hartenberg I.

8. Para i de 1 a n-1, situar Y_i de modo que formen un sistema dextrógiro con X_i y Z_i
9. Situar el sistema $\{S_n\}$ en el extremo del robot de modo que Z_n coincida con la dirección de Z_{n-1} y X_n sea normal a Z_{n-1} y Z_n.
10. Obtener como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y X_i queden paralelos.
11. Obtener como la distancia medida a lo largo del eje Z_{i-1}, que habría que desplazar $\{S_{i-1}\}$ para que X_i y X_{i-1} quedasen alineados.
12. Obtener como la distancia medida a lo largo de X_i, que ahora coincidiría con X_{i-1}, que habría que desplazar el nuevo $\{S_{i-1}\}$ para que su origen coincidiese con $\{ S_i \}$.
13. Obtener como el ángulo que habría que girar en torno a Xi, que ahora coincidiría con X_{i-1}, para que el nuevo $\{ S_{i-1}\}$ coincidiese totalmente con $\{ S_i\}$
14. Obtener las matrices de transformación
15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot:
16. La matriz T define la orientación (matriz de rotación) y posición (matriz de traslación) del extremo referidas a la base en función de las n coordenadas articulares.

- Los 4 parámetros de D-H dependen únicamente de las características geométricas de cada eslabón y de las articulaciones que le unen con el anterior y el siguiente.

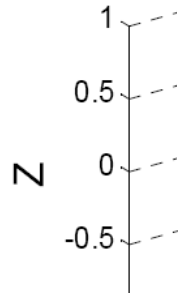
Representación en Matlab D-H

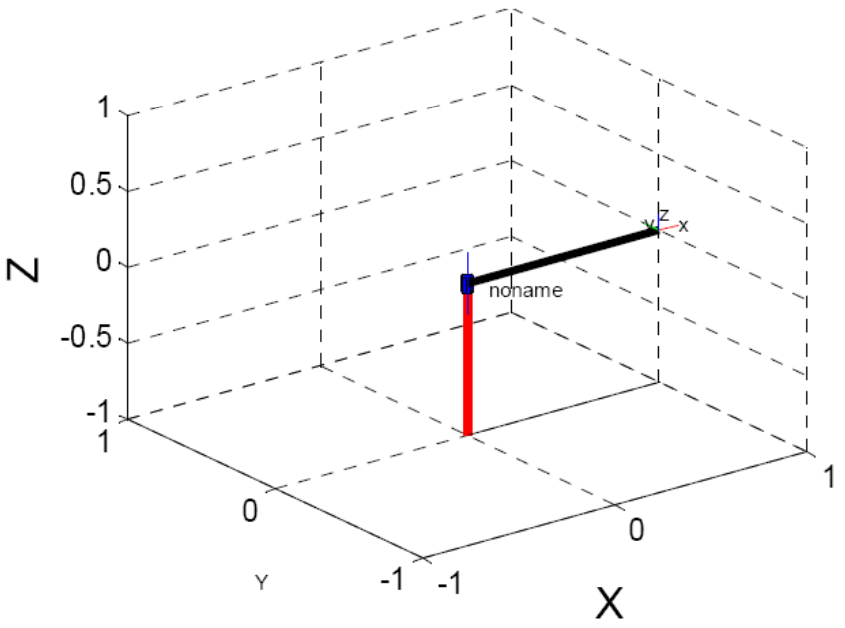
- La herramienta de robot para Matlab permite definir un robot usando la notación D-H.

L1=link($[\alpha_i, a_i, \theta_i, d_i, Articulación]$,'standard')

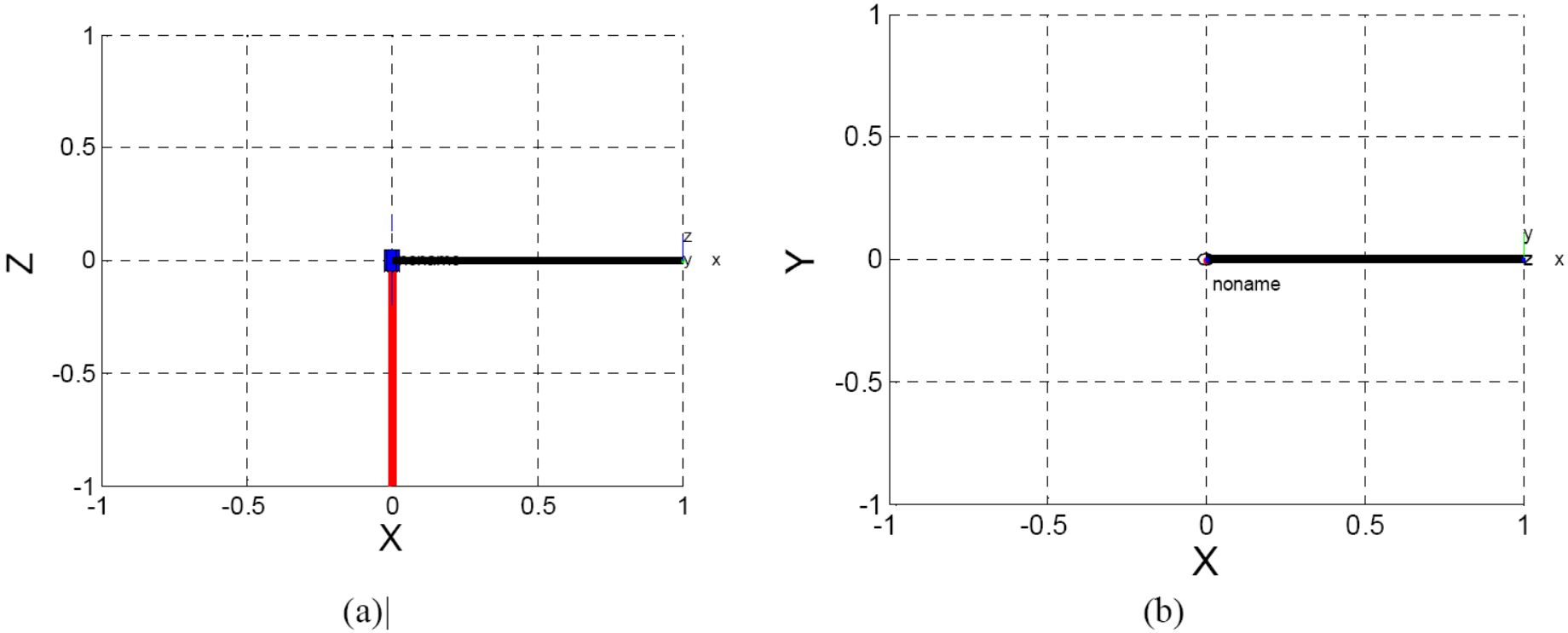
- α_i ángulo de rotación con respecto al eje X_i
- a_i Traslación a lo largo del eje X_i
- θ_i rotación alrededor del eje Z_{i-1}
- d_i traslación a lo largo del eje Z_{i-1}

```
L1=link([0,1,0,0,0],'standard')
L={L1}
r =robot(L)
Plot(r,[0])
view([0, 0]
pause
view(0, 90)
```





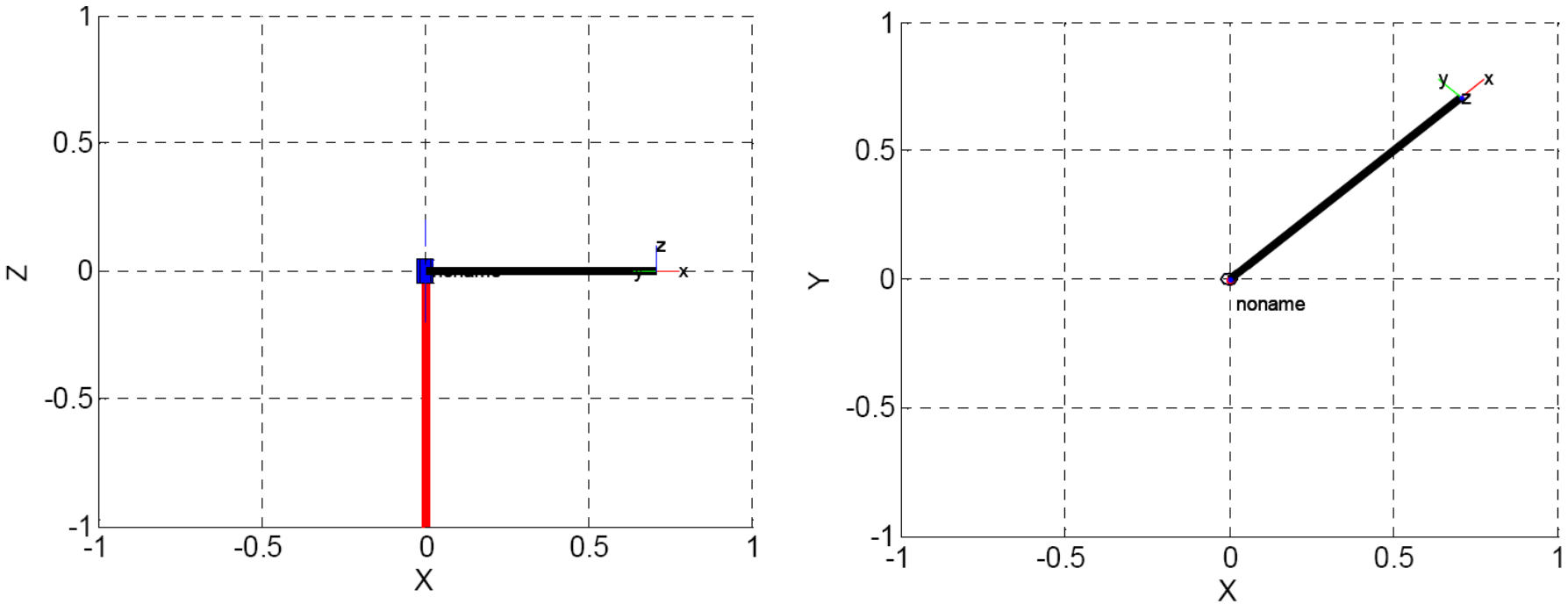
Representación en Matlab D-H



(a)| (b)

Figura (a) Representación del robot con el ángulo de 0° en el plano XZ. (b) Representación del robot en el plano XY.

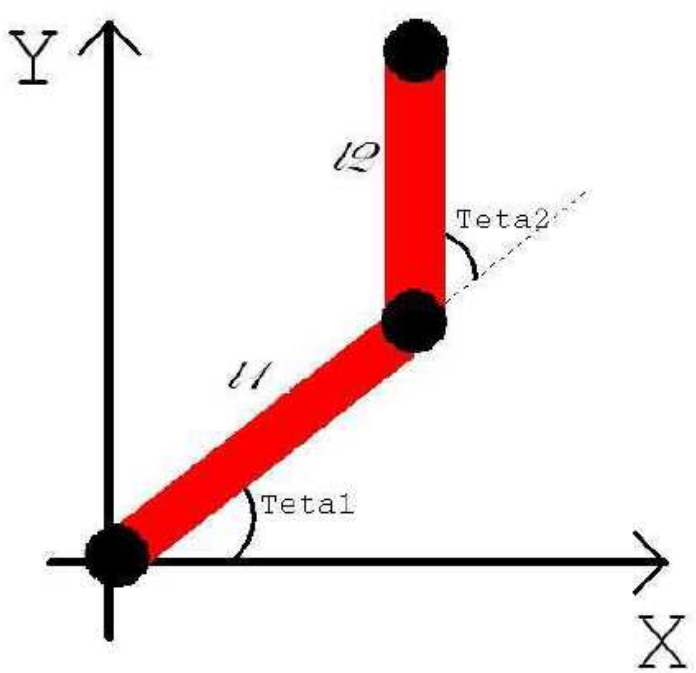
Representación en Matlab D-H

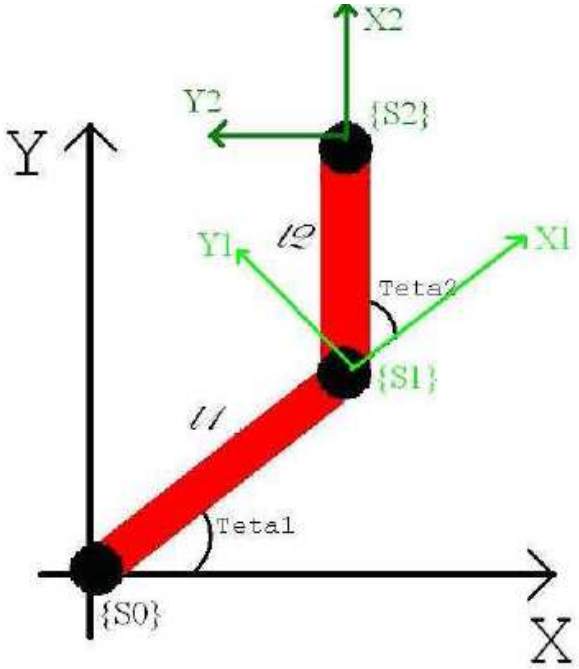


(a)| (b)

Figura (a) Representación del robot con el ángulo de 45° en el plano XZ. (b) Representación del robot en el plano XY.

Robot de 2 Grados de Libertad

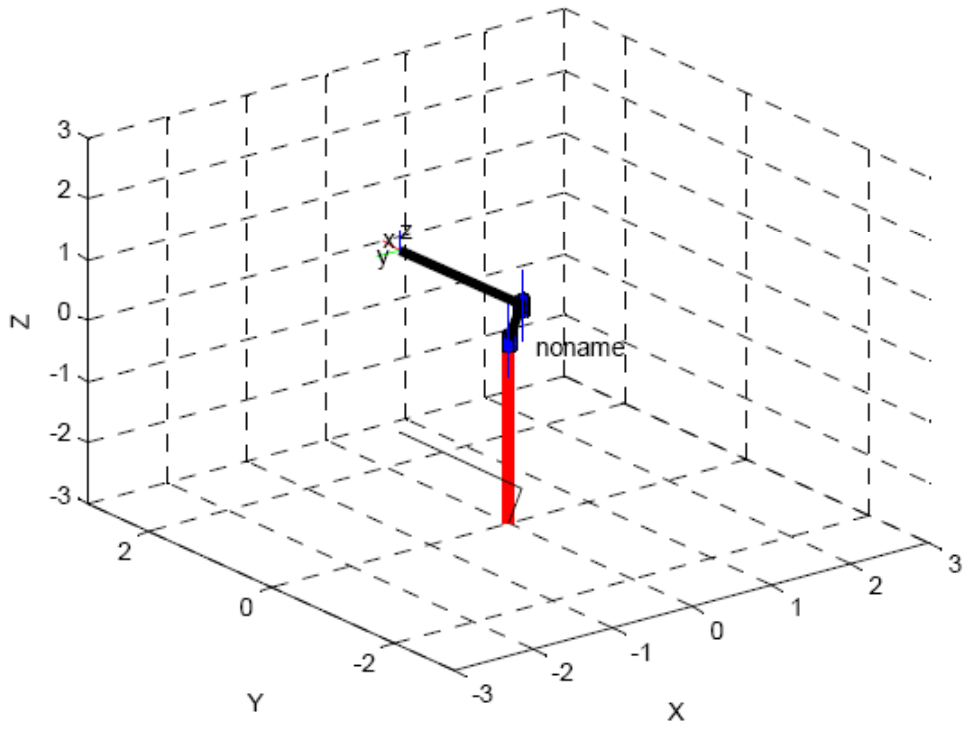




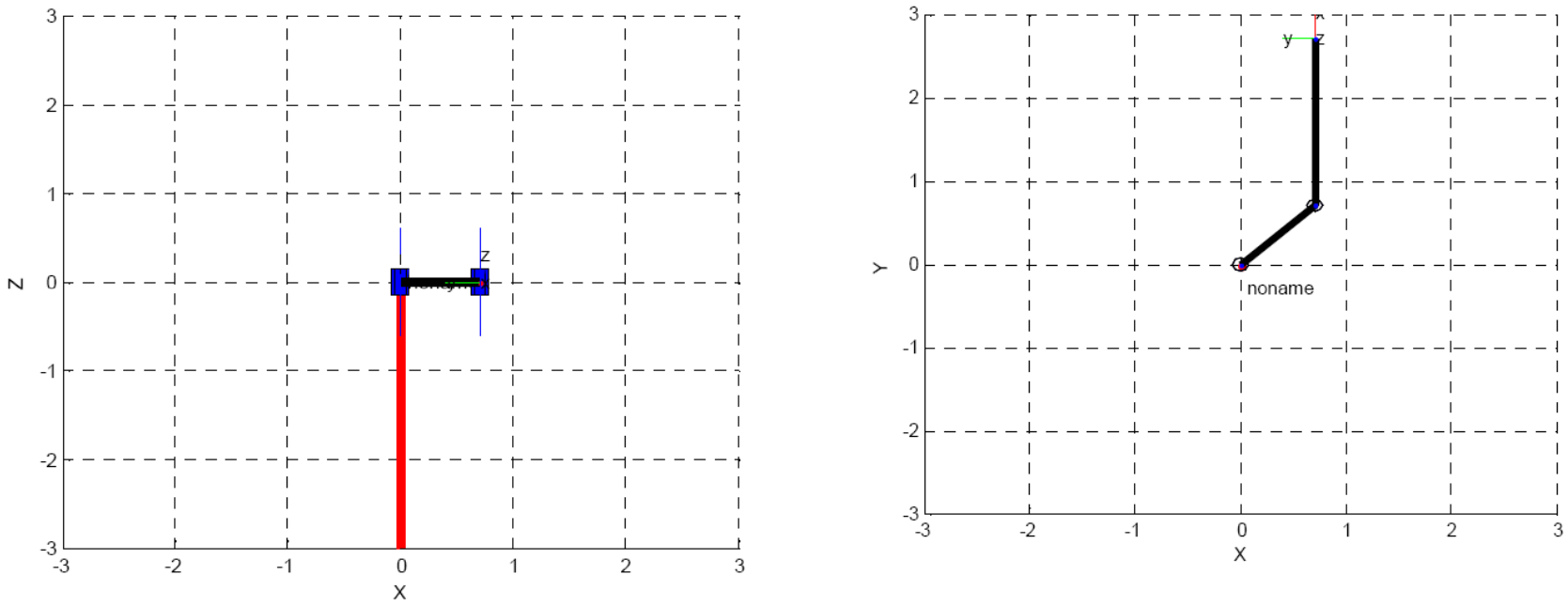
Articulación	θ_i	d_i	a_i	α_i
1	Teta1	0	l_1	0
2	Teta2	0	l_2	0

Robot de 2 Grados de Libertad

```
l1=1; l2=2
L1=link([0,l1,0,0],'standard')
L2=link([0,l2,0,0],'standard')
L={L1,L2}
r =robot(L)
Plot(r,[pi/4, pi/4])
view([0, 0]
pause
view(0, 90)
```



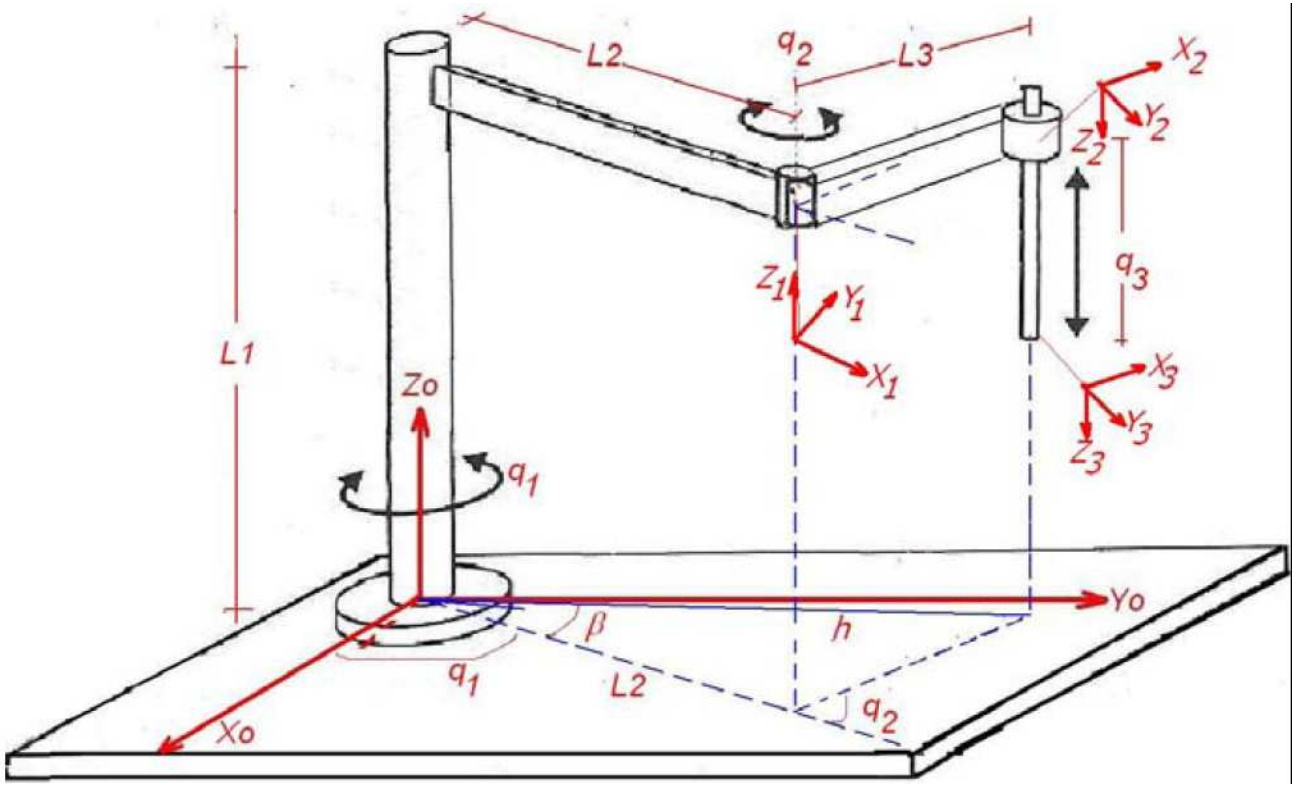
Robot de 2 Grados de Libertad



(a)| (b)

Figura (a) Representación del robot con el ángulo de 45° en el plano XZ. (b) Representación del robot en el plano XY.

Parámetros D-H Robot Scara



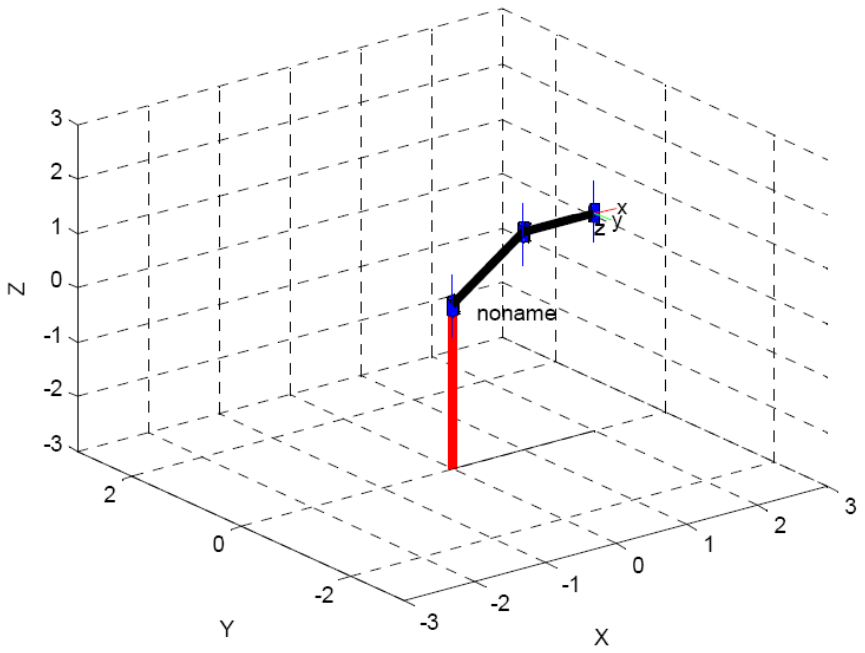
Parámetros Denavit Hartenberg

i	θ_i	d_i	a_i	α_i
1	Q1	L1	L2	0
2	q2	0	L3	180°
3	0	Q3	0	0

Parámetros D-H Robot Scara

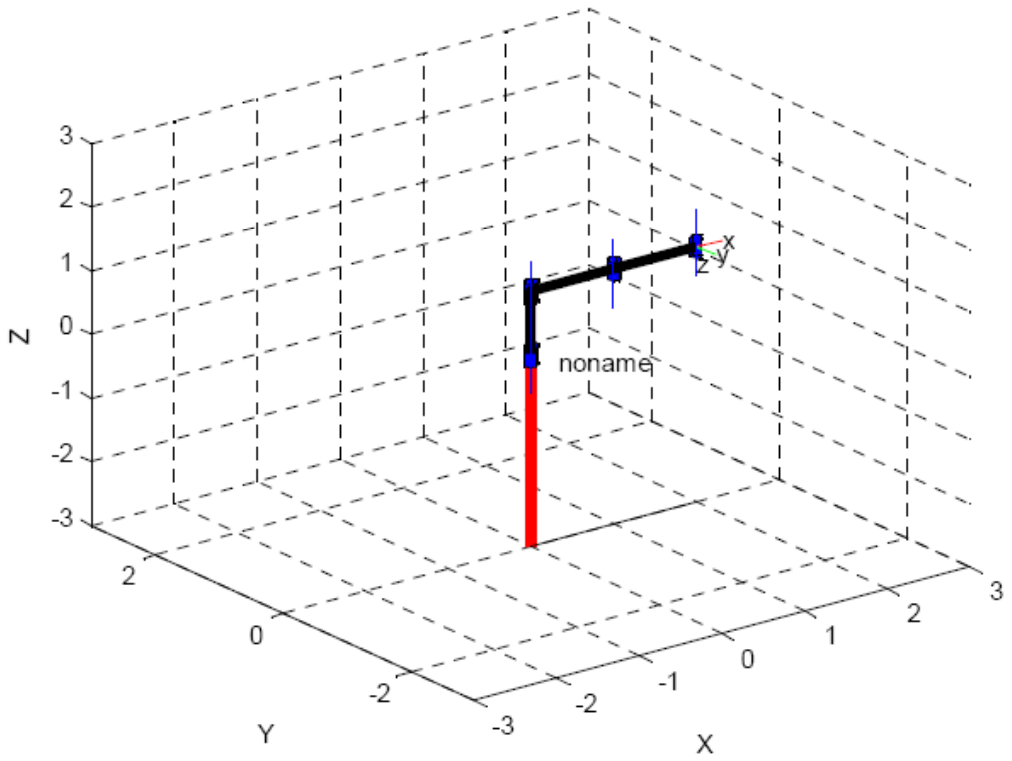
Robot Scara en MATLAB

```
%Ejemplo de como se genera un robot
antropomórfico  de 3 grados de
libertad con
%El tool box de robótica. 14 de
abril de 08
clear; close
h=0.5;
l1=1;
l2=1;
l3=1;
q1=0;
q2=0;
q3=0;
L1=link([0 l1 q1 l2 0],
'standard');
L2=link([pi l3 q2 0 0],
'standard');
L3=link([0 0 0 q3 1], 'standard');
r=robot({L1 L2 L3 })
plot(r,[q1,q2,q3])
```



Parámetros D-H Robot Scara

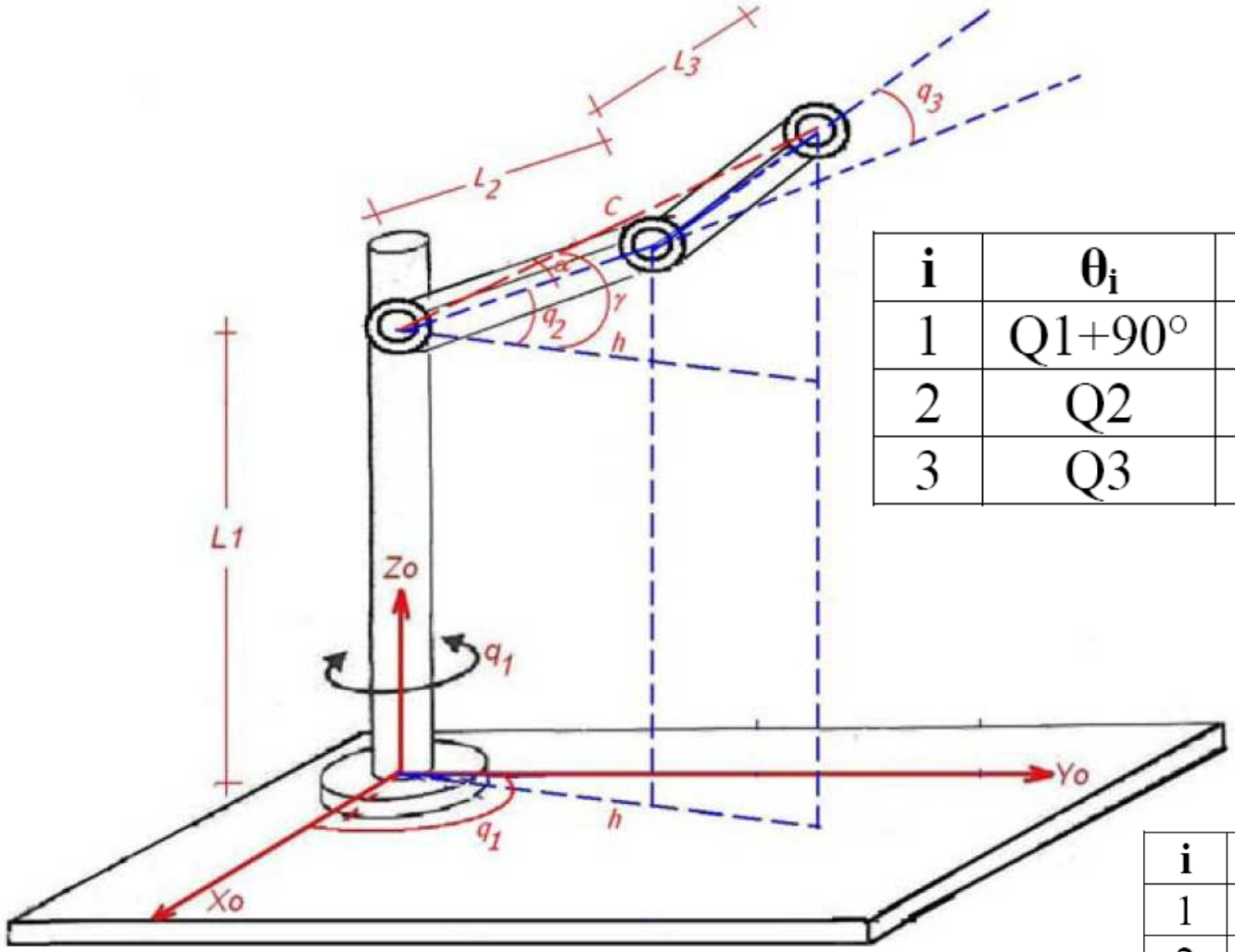
```
%Ejemplo del Robot SCARA con
una artícualación más
% Para simular el ángulo recto.
%20 de Abril de 2009
clear; close;
h=0.5; l1=1;
l2=1; l3=1;
q1=0; q2=0; %Artículación
artificial
q3=0; q4=0;
L1=link([0 0 (q1) l1 0],
'standard');
L2=link([0 l2 (q2+pi/2) 0 0],
'standard');
L3=link([pi l3 (q3) 0 0],
'standard');
L4=link([0 0 0 q3 1],
'standard');
r=robot({L1 L2 L3 L4})
plot(r,[q1,q2,q3,q4])
```



Parámetros Denavit Hartenberg

i	θ_i	d_i	a_i	α_i	Art.
1	Q1	L1	0	0	0
2	q2+90°	0	L2	0	0
3	Q3	0	L3	180	0
4	0	Q4	0	0	1

Robot Antropomorfico



i	θ_i	d_i	a_i	α_i
1	Q1+90°	L1	0	90°
2	Q2	0	L2	0°
3	Q3	0	L3	0

i	θ_i	d_i	a_i	α_i
1	Q1	L1	L2	0
2	-q2	0	L3	180°
3	0	Q3	0	0

Prueba del Robot Antropomórfico

```
%Este ejemplo hace que el robot antropomorfico genere un cuadrado y lo
%recorra de ida y vuelta
%26/Mayo/09
clear; close;
Long=[50,50,50];0
r=generaAntropo(Long);
q1=0; q2=0; q3=0;
plot(r,[0,0,0]);
xo=40; yo=-40; zo=60; radio=50;
i=1;
Puntos=[0,0,0;xo,yo,zo;40,-90,60];
M=size(Puntos);
k=1;
N=20
for i=1:M(1)-1
   N=30;
   P1=Puntos(i,:); P2=Puntos(i+1,:);
   Pxy=Traylinea(P1,P2,N);
   for j=1:N+1
      clf
      x(k)=Pxy(j,1); y(k)=Pxy(j,2); z(k)=Pxy(j,3);
      P0=Pxy(j,:);
      Q=CinversaAntropo(P0,Long);
      plot(r,Q);
      hold on
      plot3(x,y,z)
      pause(0.1)
      k=k+1;
   end
end
for t=0:pi/20:2*pi
      clf
      x(k)=xo+radio*cos(t-pi/2);
      y(k)=yo+radio*sin(t-pi/2); z(k)=zo;
      P0=[x(k),y(k),z(k)];
      Q=CinversaAntropo(P0,Long);
      plot(r,Q);
      hold on
      plot3(x,y,z)
      pause(0.1)
      k=k+1;
end
P1=[x(k-1),y(k-1),z(k-1)]; P2=[0,0,0];
 Pxy=Traylinea(P1,P2,N);
 for j=1:N+1
      clf
      x(k)=Pxy(j,1); y(k)=Pxy(j,2); z(k)=Pxy(j,3);
      P0=Pxy(j,:);
      Q=CinversaAntropo(P0,Long);
      plot(r,Q);
      hold on
      plot3(x,y,z)
      pause(0.1)
      k=k+1;
   end
```

Prueba del Robot Antropomórfico

```
clear; close;
l1=1;l2=1,l3=1;
Long=[l1,l2,l3];
r=generaAntropo(Long);
plot(r,[0,0,0]);
i=1;

q1=0; q2=0; q3=0;
escala=10;
P1=[0,0,0]; P2=[1,-1,1];
i=1;
for t=0:escala/20:escala
   clf
   P=generalinea(P1,P2,t,escala);
   X(i)=P(1); Y(i)=P(2); Z(i)=P(3);
   Q=CinversaAntropo(P,Long);
   tiempo(i)=t;
   aq1(i)=Q(1)*180/pi; aq2(i)=Q(2)*180/pi; aq3(i)=Q(3)*180/pi;
   %subplot(1,2,1),
   plot(r,Q); Hold on
   plot3(X,Y,Z)
   grid
   %subplot(1,2,2), plot(tiempo,aq1,tiempo,aq2,tiempo,aq3);
   grid
   i=i+1;
   pause(0.5)
end
```

Funciones

Cinemática Inversa del Robot Antropomórfico

- %Esta función genera al robot antropomorfico. Recibe la longitud de cad eslabon
- function Q=CinversaAntropo(P0,Lados)
- l1=Lados(1); l2=Lados(2); l3=Lados(3);
- x=P0(1); y=P0(2); z=P0(3);
- h=sqrt(x^2+y^2);
- c=sqrt(h^2+(z-l1)^2);
- gama=atan2((z-l1),h);
- alfa=acos((l2^2+c^2-l3^2)/(2*l2*c+1e-9));
-
- q1=atan2(y,x);
- q2=gama-alfa;
- q3=acos((c^2-l2^2-l3^2)/(2*l2*l3));
- Q=[q1,q2,q3];

Cinemática Directa del Robot Antropomórfico

- %Esta función genera la cinemática directa del robot Antropomórfico
- %Los parámetrios que recibe son los ángulos y la longitud de los lados
- function P0=CdirectaAntropo(Q,Lados)
- l1=Lados(1); l2=Lados(2); l3=Lados(3);
- q1=Q(1); q2=Q(2); q3=Q(3);
- h=l2*cos(q2)+l3*cos(q2+q3);
- c=sqrt(l2^2+l3^2+2*l2*l3*cos(q3));
- x=h*cos(q1);
- y=h*sin(q1);
- z=l1+l2*sin(q2)+l3*sin(q2+q3);
- P0=[x,y,z];
-

Ejemplo de movimiento de un robot Antropomórfico

```
%Este ejemplo hace que el robot
    antropomorfico genere un cuadrado y
    lo
%recorra de ida y vuelta
%26/Mayo/09
clear; close;
Long=[40,40,40];
r=generaAntropo(Long);
q1=0; q2=0; q3=0;
plot(r,[0,0,0]);
i=1;
Puntos=[0,0,0;0,-50,0;0,-
    50,70;0,0,70;0,0,0;0,0,70;0,-
    50,70;0,0,0];
M=size(Puntos);
k=1;
```

-

```
for i=1:M(1)-1
   N=30;
   P1=Puntos(i,:); P2=Puntos(i+1,:);
   Pxy=Traylinea(P1,P2,N);
   for j=1:N+1
      clf
      x(k)=Pxy(j,1); y(k)=Pxy(j,2);
    z(k)=Pxy(j,3);
      P0=Pxy(j,:);
      Q=CinversaAntropo(P0,Long);
      plot(r,Q);
      hold on
      plot3(x,y,z)
      pause(0.1)
      k=k+1;
   end
end
```

Ejemplo de movimiento de un robot Antropomórfico

Funcion generaAntopo.m

```
%Esta función genera al robot
    antropomorfico. Recibe la longitud de
    cad eslabon
function
    ObjRobot=generaAntropo(Lados)
l1=Lados(1); l2=Lados(2); l3=Lados(3);
q1=0; q2=0, q3=0;
L1=link([pi/2 0 (q1) l1 0], 'standard');
L2=link([0 l2 q2 0 0], 'standard');
L3=link([0 l3 q3 0 0], 'standard');
r=robot({L1 L2 L3});

ObjRobot=r;
```

Función TrayLinea.m

- %Esta función genera la trayectoria de una linea
- function Tray=TrayLinea(P1,P2,N)
- dP=P2-P1;
- for i=1:N+1
- Tray(i,:)=(i-1)*dP/N+P1;
- end

Ejemplo de movimiento de un robot Antropomórfico

Función CinversaAntropo

```
%Esta función genera al robot antropomorfico. Recibe la longitud de cad eslabon
function Q=CinversaAntropo(P0,Lados)
l1=Lados(1); l2=Lados(2); l3=Lados(3);
x=P0(1); y=P0(2); z=P0(3);
h=sqrt(x^2+y^2);
c=sqrt(h^2+(z-l1)^2);
gama=atan2((z-l1),h);
alfa=acos((l2^2+c^2-l3^2)/(2*l2*c+1e-9));

q1=atan2(y,x);
q2=gama-alfa;
q3=acos((c^2-l2^2-l3^2)/(2*l2*l3));
Q=[q1,q2,q3];
```