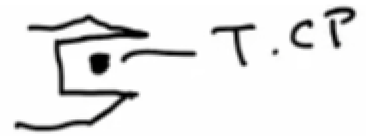
Notas - Parcial 2

Clase 06/10

Articulación prismática (o deslizante): son las que no rotan, que solo se extienden y contraen.

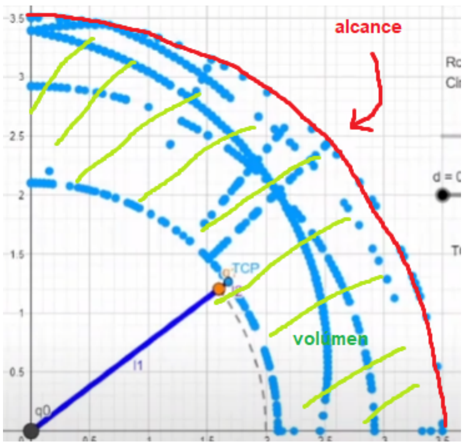
Articulación rotativa: rotan en un ángulo determinado.

Tool Center Point (TCP): es el centro de la herramienta u efector final.



Volúmen, área o espacio de trabajo: son todos aquellos lugares que el robot puede alcanzar.

Alcance del robot: es la máxima distancia que puede alcanzar o extenderse. Es la “película externa” del volúmen de trabajo.

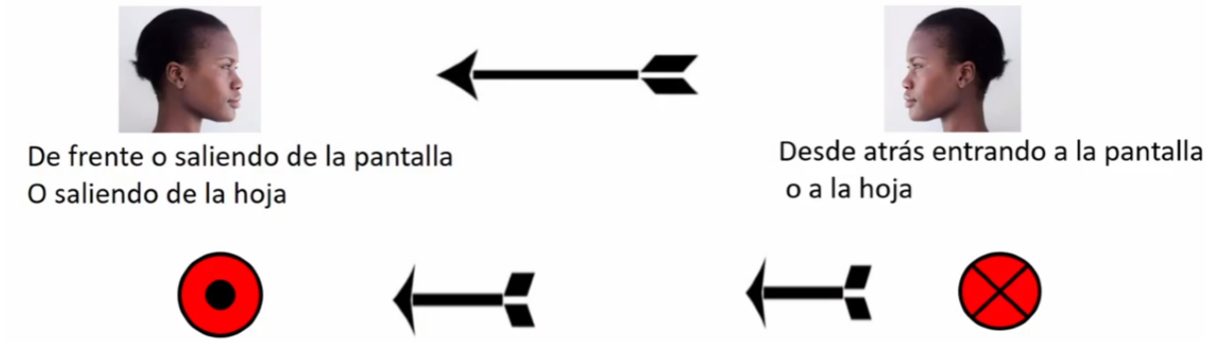


Localización del efector final:

La localización puede denominarse de dos formas:

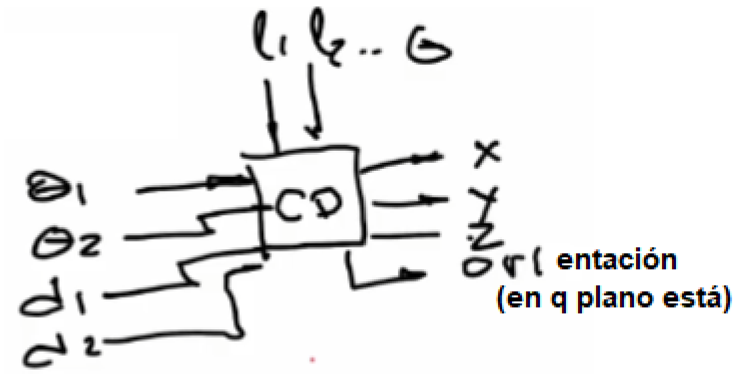
- Posición: dónde está el efectos final en el espacio (x, y, z) con respecto a la base.
- Orientación: cómo está orientado el efector final con respecto a
- Constraint: lugar donde el robot (o efector final) no puede ir.

Como representar a Z (u otros ejes) en gráficos cartesianos de 2 dimensiones:



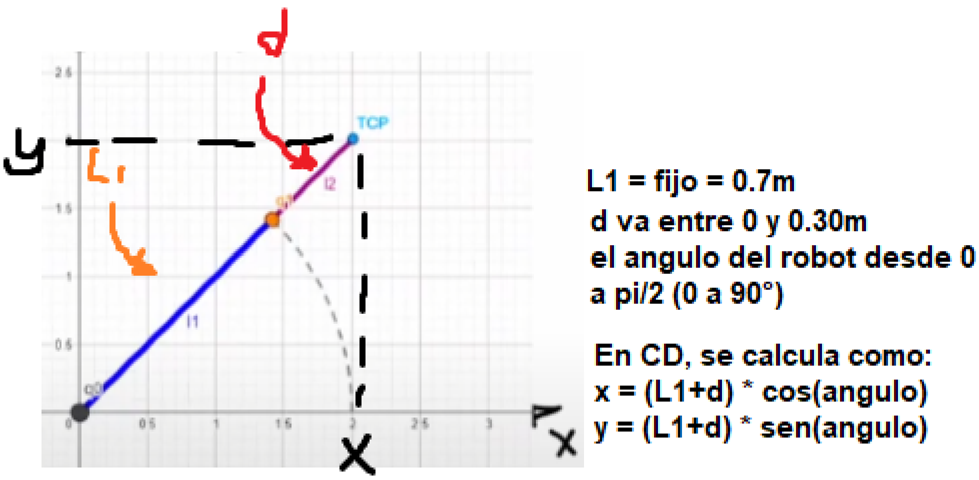
Cuando vemos la cruz X se dice **ENTRANTE** y cuando vemos el punto **SALIENTE**.

Cinemática directa: teniendo como entradas los **ángulos del robot** (q1, q2…) se obtienen las coordenadas (x, y, z) del efector final.



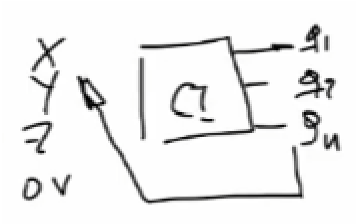
Puede tener otras entradas como datos, como las longitudes, desplazamientos, etc. La orientación se describe con la **tangente del ángulo**.

Para “un robot no tan elemental” la cinemática directa se obtiene mediante:



A la suma de (L1+d) se le llama **L**.

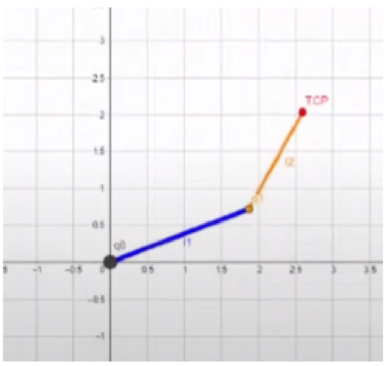
<u>Cinemática inversa</u>: teniendo como entradas las **coordenadas del robot** (x, y, z) se obtienen los ángulos y los desplazamientos (d).

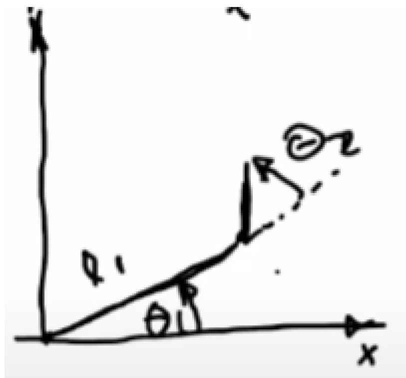


Entonces, para el “robot no tan elemental” de arriba:

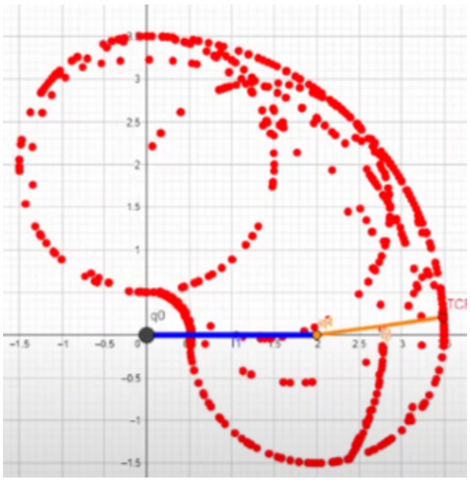
- L = l1+d
- L se puede calcular como $L = \sqrt{x^2 + y^2}$
- d = L - l1 [**solo depende de los parametros fijos**]
- El ángulo es $\theta = \operatorname{arctg} \frac{Y}{X}$ [**solo depende de Par. Fijo**]

<u>Otro robot “no tan elemental”</u>:





El área de trabajo de este robot tiene forma de “medialuna”:



El alcance, de nuevo, es toda la parte externa de la medialuna.

Teniendo como parámetros:

l1 = fijo = 0.70m
l2 = fijo = 0.30m
q1 = de 0 a pi/2
q2 = de 0 a pi

<u>Cinemática directa</u>: para este robot, es:

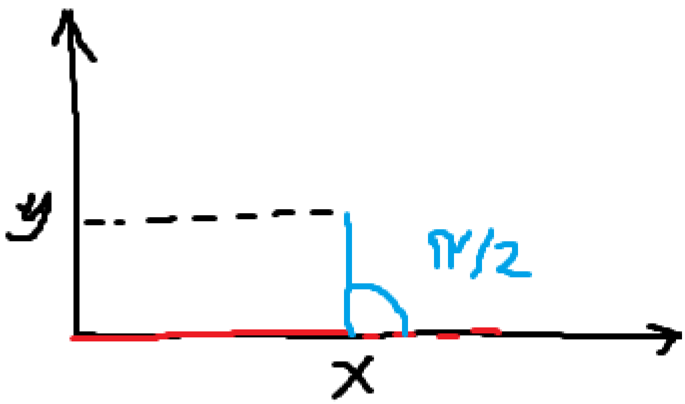
$$p_x = l_1 \cdot \cos\theta_1 + l_2 \cdot \cos(\theta_1 + \theta_2)$$
$$p_y = l_1 \cdot \text{sen}\,\theta_1 + l_2 \cdot \text{sen}(\theta_1 + \theta_2)$$

Supongamos que q1 = pi/2 y q2 = 0. ¿Cuanto vale x, y, z?

- Como el ángulo de l1 está en 90° significa que está parado en en eje Y.
- Recordar que el ángulo de l2 se toma DESDE el ángulo de l1 (NO desde el eje X) entonces un ángulo de 0 significa que está en la misma posición que l1.

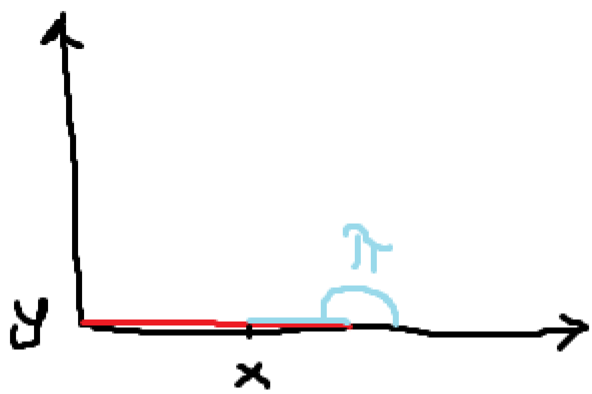
Entonces en x = 0. Como no hay z, z=0.
En y tenés que sumar las longitudes. Es y = l1+l2 = 1.

Entonces si fuese q1 = 0 y q2 = pi/2



En este caso x = 0.7m e y = 0.3m.
z = 0 (para estos robots en dos dimensiones siempre).

¿si fuese q1 = 0 y q2 = pi?



La CD siempre busca la posición del efector final.
Entonces y = 0; **x =** 0.7m-0.3m = **0.4m**; z = 0.

Más ejemplos:

- Para que el robot esté en el eje de las X completo entonces la cinemática directa debe ser: 0, 0
- Para que el robot esté en el eje de las -X completo entonces la cinemática debe ser: pi, 0
- Para que esté en el eje de las -Y (180°+90° = 270°) que sería 3/2 pi, 0
- Si está pi/4 (45°) y 0, la longitud para 0.7m y 0.3m es 0.7071.

MatLab para este robot: **CDdosG.m**
MatLab para este robot pero con funciones: **CDirecta.m**

- a la función se la llama escribiendo: CDirecta([q1,q2])

Cinemática inversa: del robot anterior es

Dados los parámetros del robot, l1 y l2.
También las coordenadas del robot, x, y, z..

Queremos hacer alguna función para obtener los ángulos q1 y q2 que ponen a ese robot en esa posición de x, y, z.

- $L = \sqrt{x^2+y^2}$
- lo que es lo mismo $L^2 = x^2 + y^2$
- Después se calcula q2 usando el teorema del coseno:

$$L^2 = l_1^2 + l_2^2 + 2 l_1 l_2 \cdot \cos\theta_2$$

$$P_x^2 + P_y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 \cdot \cos\theta_2$$

$$\cos\theta_2 = \frac{P_x^2 + P_y^2 - l_1^2 - l_2^2}{2 l_1 \cdot l_2}$$

- Tecnicamente ya tendríamos q2, pero como “computacionalmente hablando es mejor tener todo en función de la arcotangente y no del arcocoseno” se usa otra identidad trigonométrica para sacarlo

$$\cos^2\theta_2 + \operatorname{sen}^2\theta_2 = 1$$

$$\operatorname{sen}\theta_2 = \pm\sqrt{1-\cos^2\theta_2}$$

$$\boxed{\theta_2 = \operatorname{Artg}\frac{\operatorname{sen}\theta_2}{\cos\theta_2}}$$

(reemplazarías sen q2 y cos q2 con sus ecuaciones)

- Para calcular q1 es:

$$\beta = \operatorname{Arctg}\frac{P_Y}{P_X}$$

$$CA = l_1 + l_2 . \cos\theta_2$$

$$\alpha = \operatorname{Artg}\frac{l_2 . \operatorname{sen}\theta_2}{l_1 + l_2 . \cos\theta_2}$$

$$\boxed{\theta_1 = \beta - \alpha}$$

MatLab para este robot con funcion: **CInversa.m [No usar, vieja]**

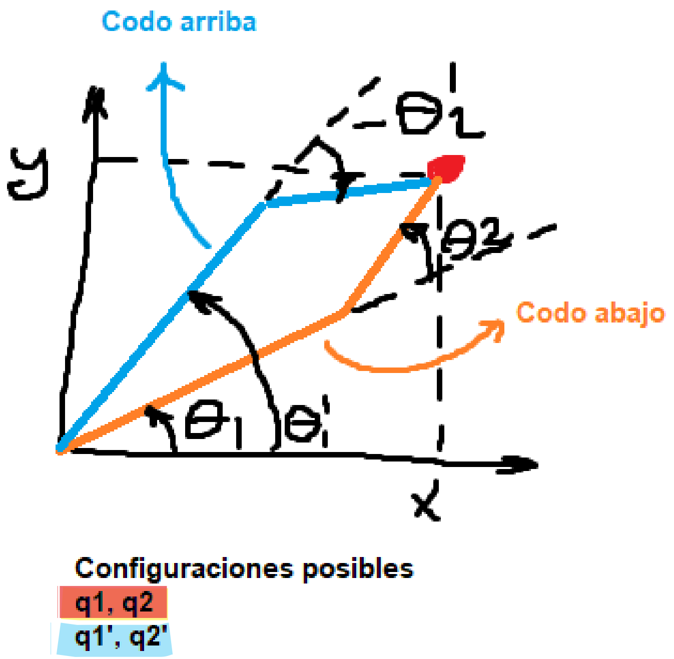
- a la función se la llama escribiendo: CInversa([x,y])
- esta versión sólo funciona para el primer cuadrante

$$\boxed{\text{sen } q2 = +-\sqrt{1-\cos 2\ q2}}$$

Qué significa la doble raíz (+-) en q2:

En el robot anterior, siempre hay **dos configuraciones** para alcanzar un cierto punto.

Una con q1 y q2 positivo y otra con q1 positivo y q2 negativo.



Como la cinemática inversa es un problema de síntesis, podemos tener diversos resultados.

La cinemática inversa es un problema de síntesis:

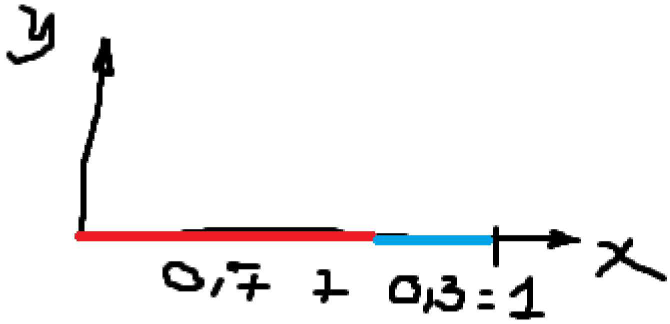
Significa:

- Puede tener 1 solución (ejemplo, **todo el brazo extendido** sobre un eje, tanto codo arriba como codo abajo dará la misma solución).
- Puede tener varias soluciones (ejemplo, codo arriba y codo abajo).
- Pueden no haber soluciones (ejemplo, fuera del alcance).

Clase 11/10

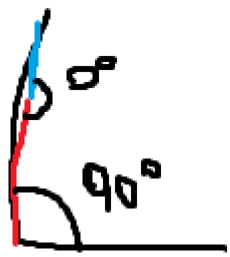
Algunos ejemplos de cinemática inversa:

- Si queremos que el robot esté en el eje de las X completo entonces a la función le tenemos que dar x = 1 e y = 0



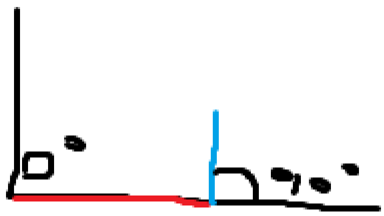
La cinemática inversa me va a dar los ángulos: x=0° e y = 0°.

- Supongamos que queremos que el robot esté en x = 0, y = 1. Esto significa que el robot está completamente vertical. ¿Qué ángulos debería darme la cinemática inversa?



x = 90° e y = 0°.

- Supongamos que queremos un x = 0.7 y un y = 0.3 ¿qué valores debería darme la cinemática inversa?



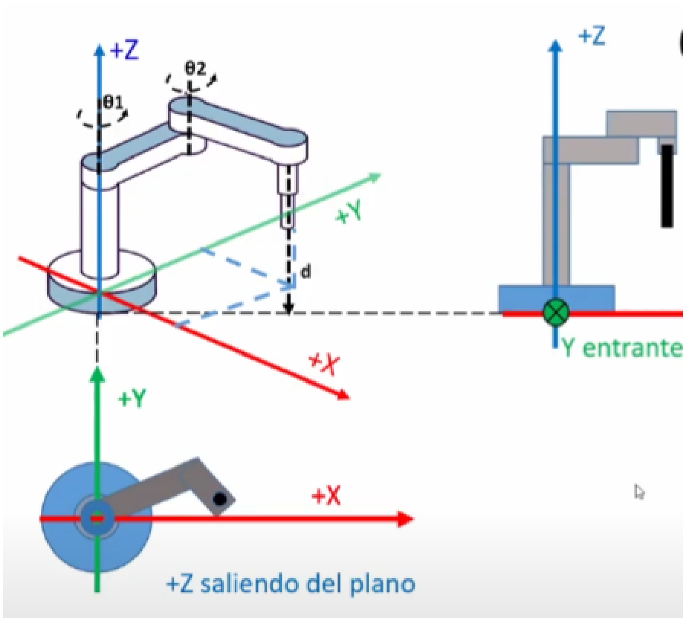
x = 0° e y = 90°.

- Supongamos que queremos un x = -0.3 y un y = 0.7.

x = 90° e y = 90°.

<u>Robot SCARA</u>

Posee tres articulaciones: dos rotativas y una de desplazamiento.



Queremos averiguar x, y, z (cinemática directa):

Articulaciones Parámetros

X= función de θ1, θ2, d, l1 y l2 y D

y= función de θ1, θ2, d, l1 y l2 y D

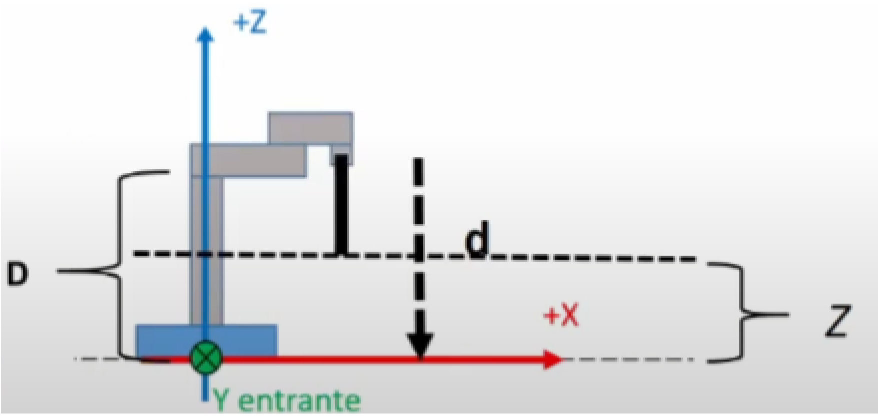
Z= función de θ1, θ2, d, l1 y l2 y D

O de alguno de ellos

Visto desde arriba, este robot es idéntico al “robot no tan elemental” por lo que para calcular x e y, se utiliza lo mismo:

$$\begin{cases} x = \ell_1 \cdot \cos\theta_1 + \ell_2 \cdot \cos(\theta_1 + \theta_2) \\ y = \ell_1 \cdot sen\theta_1 + \ell_2 \cdot sen(\theta_1 + \theta_2) \end{cases}$$

Lo único que cambia es que ahora tenemos z.



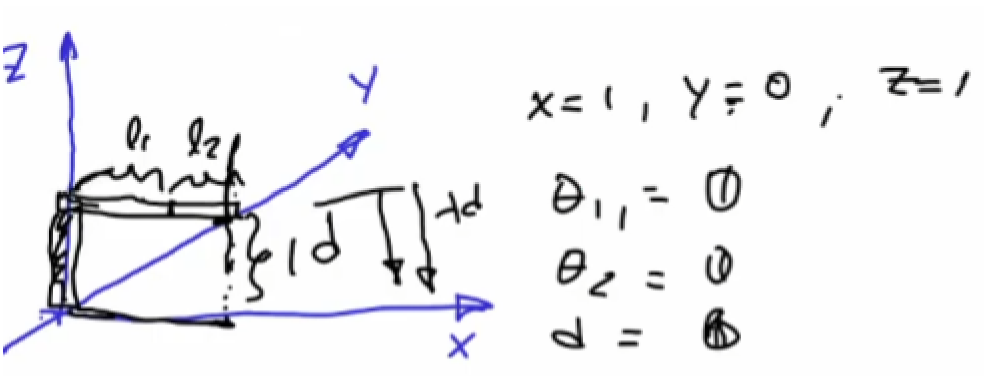
Z es simplemente la resta entre D y d. **Z = D - d**.

MatLab para este robot con funcion: **CDirectaSCARA.m**

- a la función se la llama escribiendo: CDirectaSCARA([q1,q2,d])

Ejemplos:

- ¿Que q1, q2 y d tengo que poner para obtener como resultado x= 1, y= 0, z=1 en la cinemática directa de SCARA?



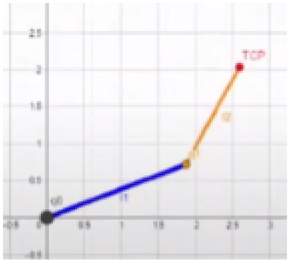
q1= 0°, q2 = 0° y d = 0

- ¿Que q1, q2 y d tengo que poner para obtener como resultado x= 0, y= 1, z= 0.3 en la cinemática directa de SCARA?

  q1 = pi/2, q2 = 0 y d = 0.7

Clase 18/10

Volviendo al robot pantógrafo de dos grados de libertad



Nuevo MatLab para este robot con funcion: **CInversa_03.m**

- a la función se la llama escribiendo: CInversa_03([x,y])
- calcula una matriz, ya que considera codo arriba y codo abajo.
- toma la que tenga menor error de 0.001 (1 milímetro).
- **requiere de parametros_fisicos.m** ya que ahí se edita l1 y l2.

Matrices de transformaciones:

Para representar traslaciones en robótica, no es conveniente hacerlo mediante suma de vectores. Es conveniente realizarlo con matrices.

Entonces, en vez de utilizar vectores:

$$\begin{bmatrix} p_x' \\ p_y' \\ p_z' \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

donde p’ es el destino, p el origen y v la traslación.

Lo representamos con matrices:

$$\underbrace{\begin{bmatrix} p_x' \\ p_y' \\ p_z' \\ 1 \end{bmatrix}}_{4x1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{4x4} \underbrace{\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}}_{4x1}$$

Es otra representación que es **idéntico** a lo anterior.

La matriz que contiene ceros y unos, se denomina **Matriz H (Matriz de transformación homogénea). Esta es la matriz de traslación.**

- Para poder realizar una traslación de un punto a otro, basta con definir H como matriz de traslación y pre-multiplicar esa matriz con el vector p (el origen). Entonces p’ = Hp

Por ejemplo:

➔ Antes teníamos este vector.

$$(2,3,4)+(1,-6,-7)=(3,-3,-3)$$

Al punto de origen p= (2,3,4) le sumábamos el desplazamiento v=(1,-6,-7) y obteníamos como resultado el valor de destino p'=(3,-3,-3).

➔ A esto mismo ahora lo podemos representar con la matriz H.

$$\begin{bmatrix} 3 \\ -3 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -6 \\ 0 & 0 & 1 & -7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 1 \end{bmatrix}$$

Vimos desplazamiento ¿cómo hacemos que haya una **rotación**?

Tenemos que definir otra matriz, pero que nos permita hacer una rotación.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & 0 \\ sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Necesitamos el origen (x, y, z) y el ángulo que queremos rotar.

Con el origen, premultiplicamos por la **Matriz de Rotación** (que se obtiene con identidades trigonométricas). Se identifica que es de rotación por los senos y cosenos.

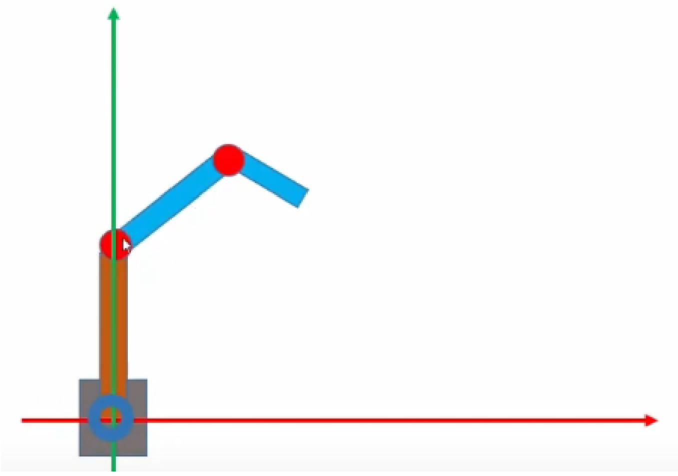
Obtenemos como resultado el (x', y', z') que sería el punto rotado.

Tanto traslación, como rotación tienen una última fila [0, 0, 0, 1] [1] que permite que la multiplicación sea posible.

Clase 20/10

<u>El tercer robot “no tan elemental”</u>:

Es un robot de 3 grados de libertad.



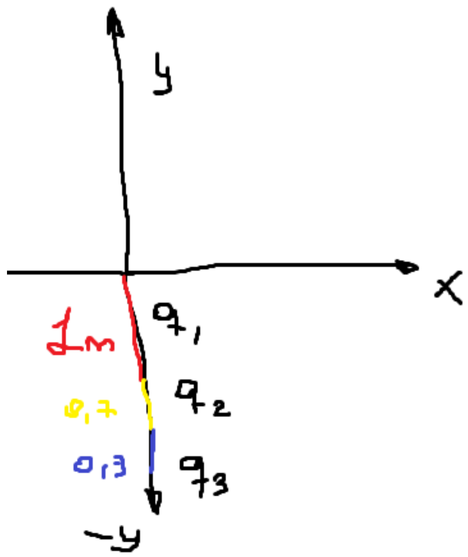
- Una articulación prismática que va desde cero a uno (la parte marrón), es decir, baja y sube en el eje de las y.
  - 1 metro.
- Una articulación rotativa que va desde cero a pi (primera parte celeste)
  - 0.7 metros
- Otra articulación rotativa que va desde cero a pi (segunda parte)
  - 0.3 metros

Cinemática directa:

El robot tiene tres parámetros.

q1= que es un desplazamiento (d).
q2 y q3 un ángulo. Donde q2 se mide desde el eje X y q3 desde q2.

Supongamos que queremos que el robot esté completamente recto sobre el eje de las -Y.

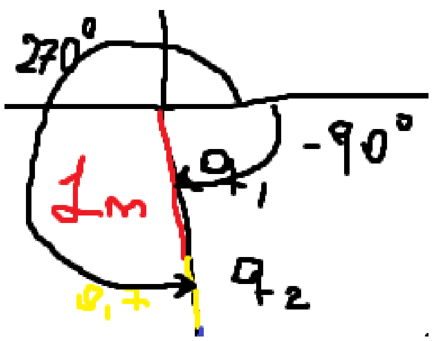


Esto implicaría que buscamos los ángulos para que el robot esté en:

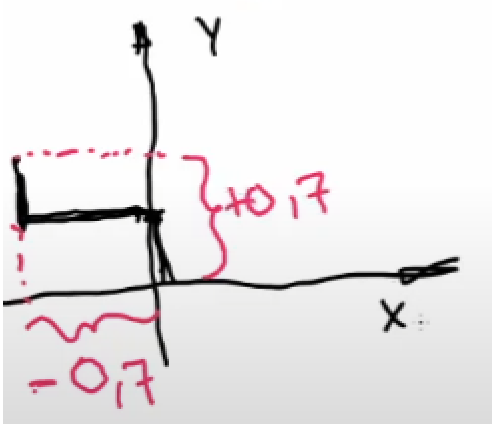
x = 0
y = -1m - 0.7m - 3m = -2

La respuesta es:

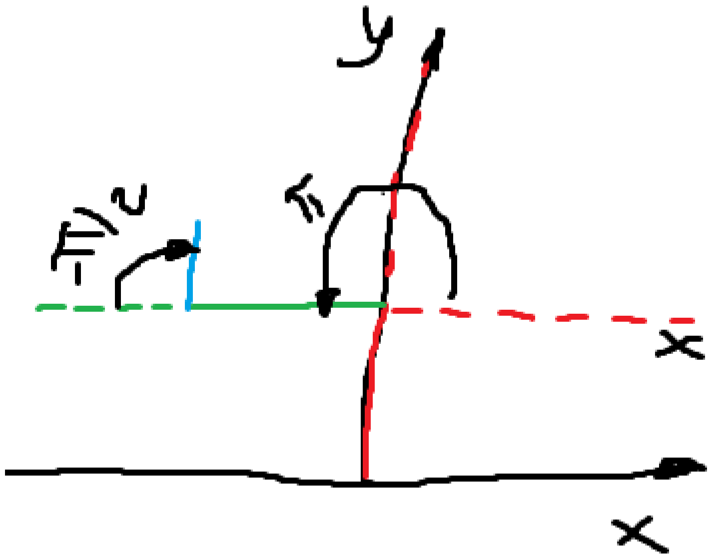
q1 = -1m
q2 = 3/2 pi o también -pi/2
q3 = 0



Supongamos ahora que queremos que el robot esté en x= -0.7 e y=0.7
¿qué ángulos tendría que tener?

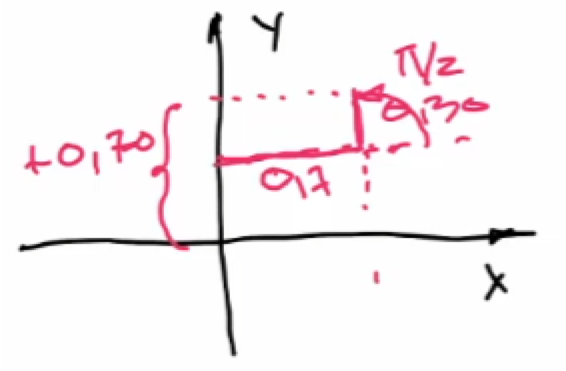


q1=0.4 (de restar 0.7-0.3) ; q2=pi ; q3=-pi/2



El q2 se toma desde el eje X (es como si en la punta de la articulación prismática se formase un nuevo X). El q3 se toma desde el q2.

Supongamos ahora x=0.7 e y=0.7



q1=0.4m ; q2=0 ; q3=pi/2

Rotación de ejes en el espacio

Las matrices de rotación permiten no sólo rotar un punto, sino también, por ejemplo, **todo un eje**.

Supongamos que queremos rotar el eje Z.

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

Sabemos que es de rotación porque tiene senos y cosenos, al eje que se rotará, le aparece un 1.

Se denomina como Rot(z, angulo).

Al rotar el eje completo, no solamente vamos a rotar el eje, sino también todos los puntos dentro.

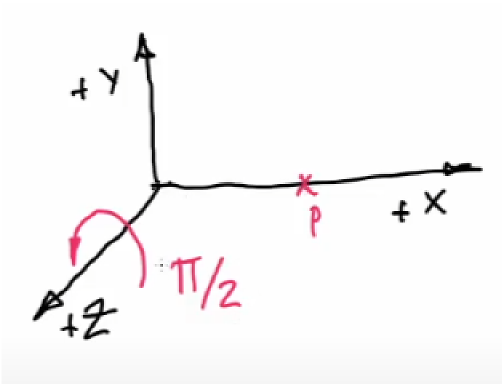
Supongamos que queremos rotar el eje X.

```
[ 1,          0,           0, 0]
[ 0, cos(alfa), -sin(alfa), 0]
[ 0, sin(alfa),  cos(alfa), 0]
[ 0,          0,           0, 1]
```

Supongamos que queremos rotar el eje Y.

Ejemplos:

Supongamos que tenemos el punto p = [x, y, z, 1] = [3, 0, 0, 1] y queremos rotar el eje Z en pi/2 (90°).



¿Dónde terminaría el punto?

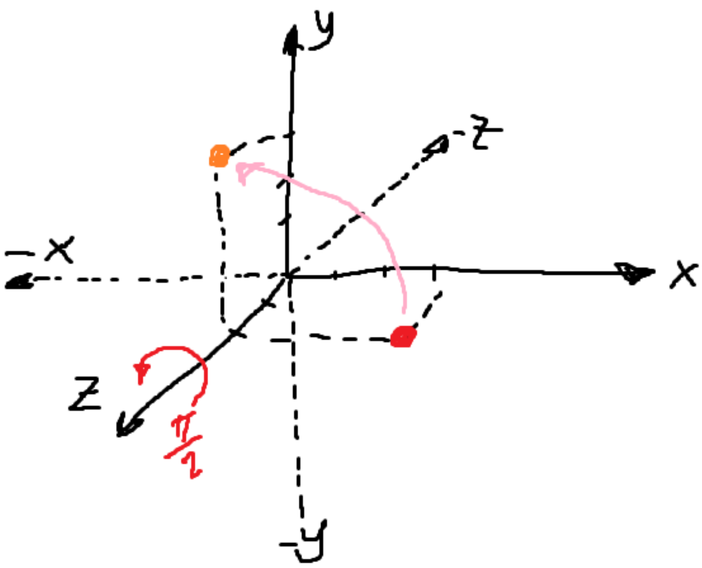
p’ = [0, 3, 0, 1]

Esto se obtuvo o se denomina como: **Rot(Z, pi/2) * p**

Ahora supongamos que queremos Rot(Z, pi/2) * p’

p’’ = [-3, 0, 0, 1]

Consideramos ahora el punto p = [3, 0, 2, 1] con Rot(Z, pi/2) * p. ¿Dónde va?

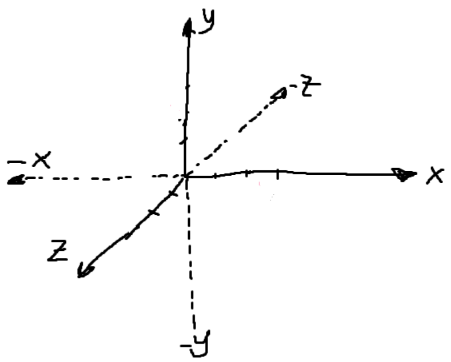


Se va a p’ = [0, 3, 2, 1]

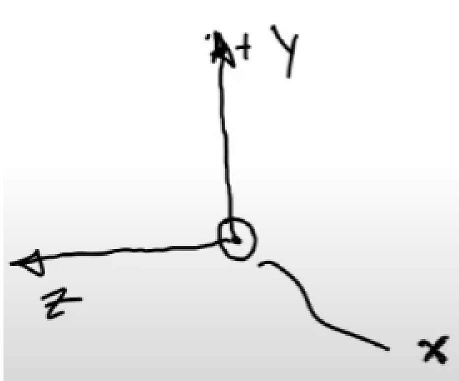
Supongamos que queremos rotar el eje X.

Siempre nos posicionamos "mirando" el eje que queremos rotar. Positivo será "antihorario" y negativo será "horario".

Si nuestro gráfico cartesiano es:



entonces viendo desde el eje X es:



Está bueno dibujar el eje de esta forma cuando tengas rotaciones / desplazamientos sucesivos de diferentes ejes. Entonces si primero tenes que rotar el eje X, dibujas el punto con el gráfico viendolo desde X. Lo rotas, obtenes el resultado y pasas al siguiente.

<u>Traslación y Rotación utilizando matrices en MatLab</u>

MatLab para ejecutar esto: **Tras_Rot.m**

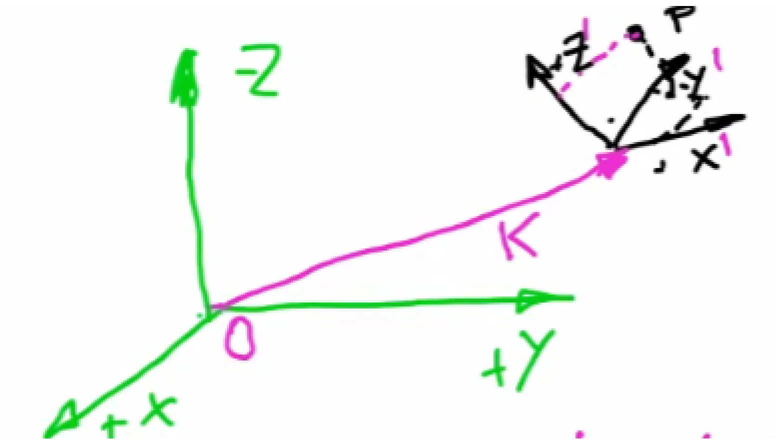
- se tienen que ir ejecutando paso a paso las "secciones" con Run Section.
- El punto se tiene que definir en donde dice p=[x y z 1]'
- Tenés que escribir el alpha donde dice %alfa = angulo
- El programa tiene para ir ejecutando rotaciones en cada eje, hay que elegir el eje que querramos.

Clase 25/10 - Presencial - Más ejercitación de rotaciones y traslaciones

Clase 27/10

<u>Matriz H para representar sistemas rotados y/o trasladados</u>

Supongamos que tenemos un sistema (x’, y’, z’) que con respecto al original se encuentra rotado y/o trasladado.



Las coordenadas del punto P en el sistema (x’, y’, z’) son:

- Px’, Py’, Pz’.

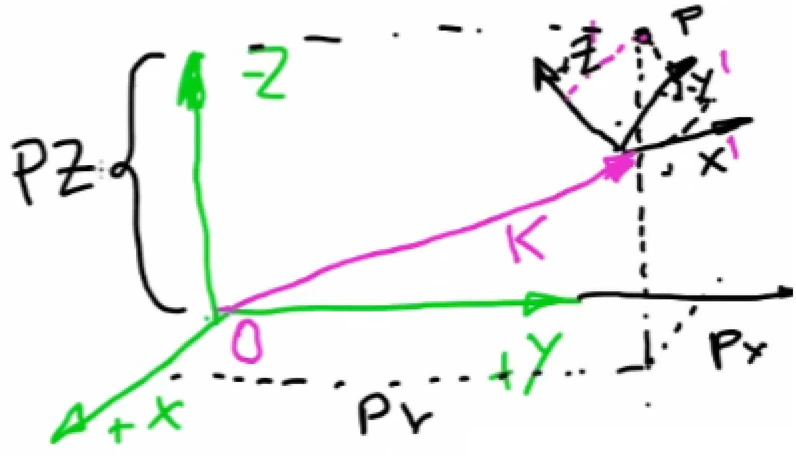
Podemos construir una **Matriz H** que represente este sistema rotado y trasladado que tenga la característica que cuando la multipliquemos por el punto en ese sistema rotado y/o trasladado, obtengamos como resultado las coordenadas del punto x, y, z pero con respecto al sistema original.

Entonces, se puede definir:

$$\begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = H * \begin{bmatrix} P_{x'} \\ P_{y'} \\ P_{z'} \\ 1 \end{bmatrix}$$

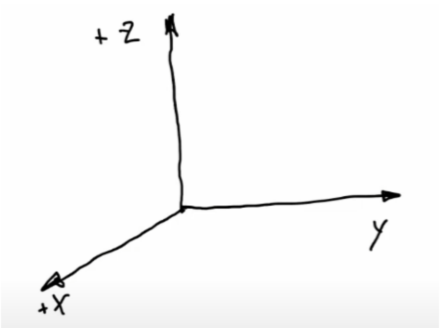
Para obtener Px, Py, Pz.

Las coordenadas del punto P con respecto al sistema original, son diferentes a con respecto al sistema rotado y/o trasladado anterior:



Ejercicios

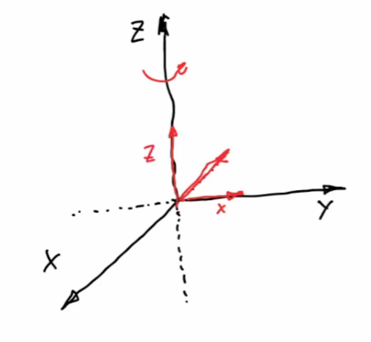
Supongamos que tenemos este sistema:



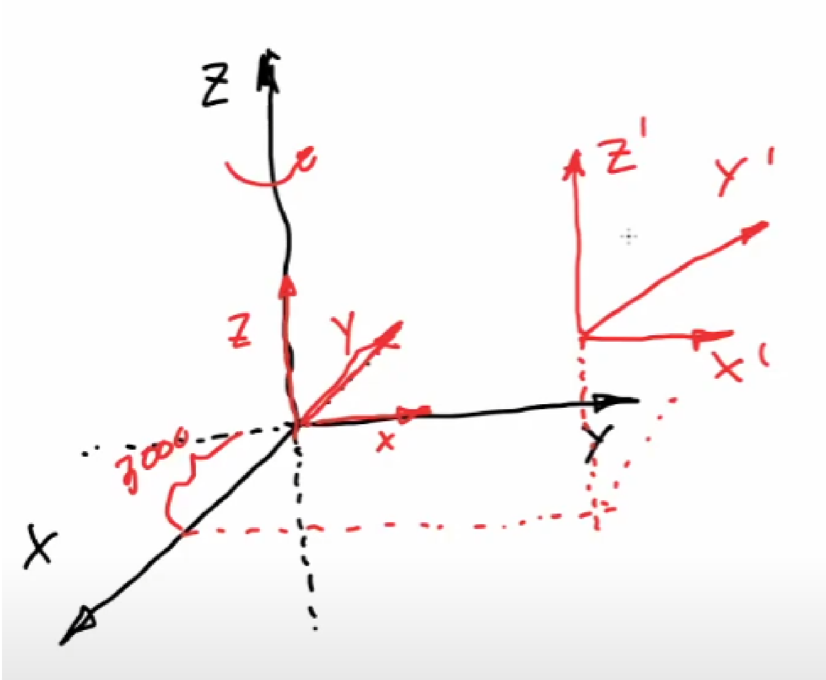
Realizamos una rotación **del sistema completo**: Rot(Z, pi/2)

Entonces nos posicionamos viendo al eje Z de arriba y lo rotamos antihorario pi/2 para formar **el nuevo sistema rotado.**

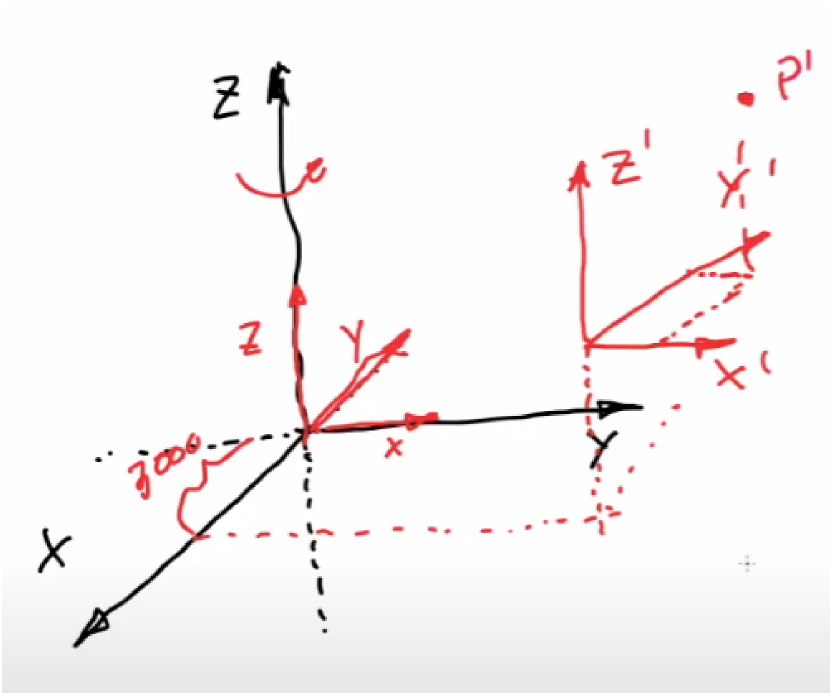
Tendremos



A este mismo sistema lo trasladamos Tras(3000, 5000, 350, 1) y esta traslación se hace **con respecto al sistema original, no con el rotado.**



Agregamos un punto en este sistema P’ = [3, 5, 10, 1]



¿Cómo expresamos este punto pero en función del sistema original?

- Lo que buscamos obtener es:

  P0 = Tras(3000, 5000, 350, 1)*Rot(Z, pi/2)*P’

  La parte “Tras*Rot” es la Matriz H.

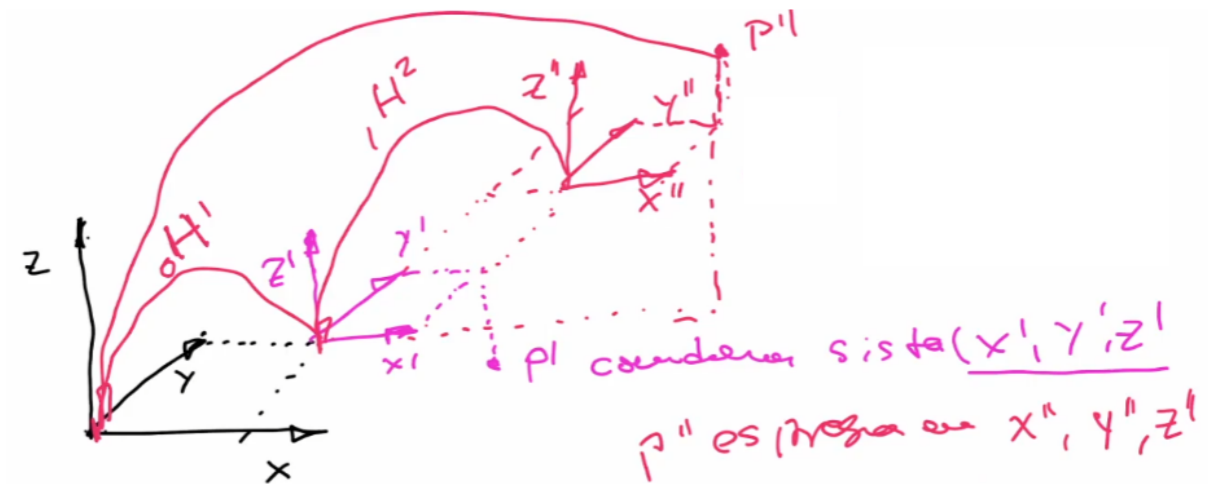
- El resultado (viendolo del gráfico y deduciendolo sería)

  Px0 = 2995 (3000-5)
  Py0 = 5003 (5000+3)
  Pz0 = 360 (350+10)

Para ejecutarlo en MatLab [**rotaciones_cuart.m**]:

- En **rotaciones_cuart.m**
- Línea 383 se ve rotación en Z.
- Cambiar gama por pi/2.
- Quitar punto y coma para ver la matriz.
- Después se ejecuta la traslación en línea 394.
- Se pone el punto p = [3 5 10 1]’ en línea 402.
- Se ejecuta la línea 407 para el resultado.

Entonces, esto se puede ir “complejizando” haciendo lo mismo, pero más veces.



Acá se hizo lo mismo que antes, se hizo un sistema nuevo (x’, y’, z’) pero luego se hizo otro nuevo, con respecto al nuevo anterior, (x’’, y’’, z’’).

Entonces, supongamos que queremos obtener el punto P’’ de (x’’, y’’, z’’) pero con respecto al (x’, y’, z’).

Esto se puede representar como 1H2 (del sistema 2 con respecto a 1) y surge de hacer:

P’ = 1H2 * P’’ [se premultiplica el punto p’’ por la matriz H]

De esa forma obtenemos el punto p’’ que estaba en el sistema nuevo nuevo pero con respecto a las coordenadas del sistema nuevo.

Ahora, si a este P’ que calculamos, lo premultiplicamos por:

P0 = 0H1 * **P’** [se premultiplica el punto P’ de antes por la matriz H de la base]

Entonces obtenemos el punto p’’ pero expresado en función de **“la base”** (siendo la base el sistema originalísimo)

Basicamente es una recursión.

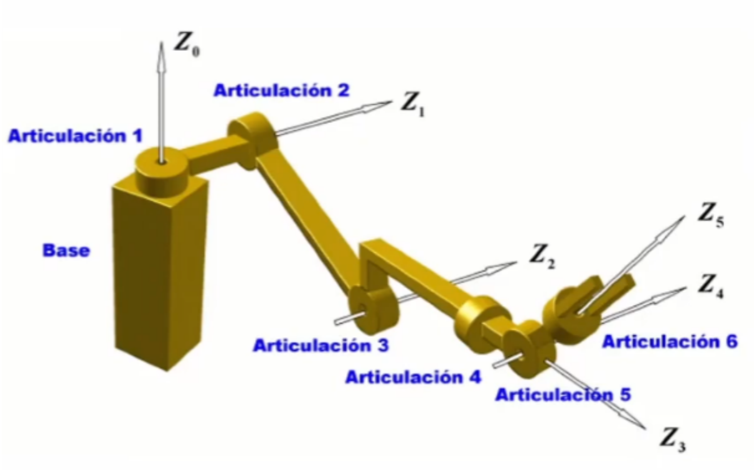
Se puede hacer todo de un solo golpe como:

P0 = 0H1 * 1H2 * p’’

(Se puede decir que; H total = 0H1 * 1H2)

De esa forma obtenemos el punto p’’ que estaba en el sistema nuevo nuevo pero con respecto a las coordenadas del sistema base.

Método Denavit-Hartenberg



Continuando lo anterior, un robot se puede pensar como una concatenación de sistemas hasta llegar a la base del mismo.

Esto nos permite, por ejemplo, poder expresar la posición del efector final con respecto a la base.

El método de Denavit-Hartenberg tiene como objetivo expresar la posición o coordenadas del efector final con respecto a la base y para lograrlo realiza una sucesión de transformaciones lineales hasta llegar a la base. Basicamente es una manera de lograr la cinemática directa solo dandole ciertos parámetros como datos de cada eslabón del robot.

Entonces por cada articulación tenemos un sistema:

Basta con conocer cuatro cosas de cada eslabón del robot (eslabón: cada una de las partes rígidas que componen la estructura mecánica del robot) para determinar todo.

a_i: Largo del eslabón

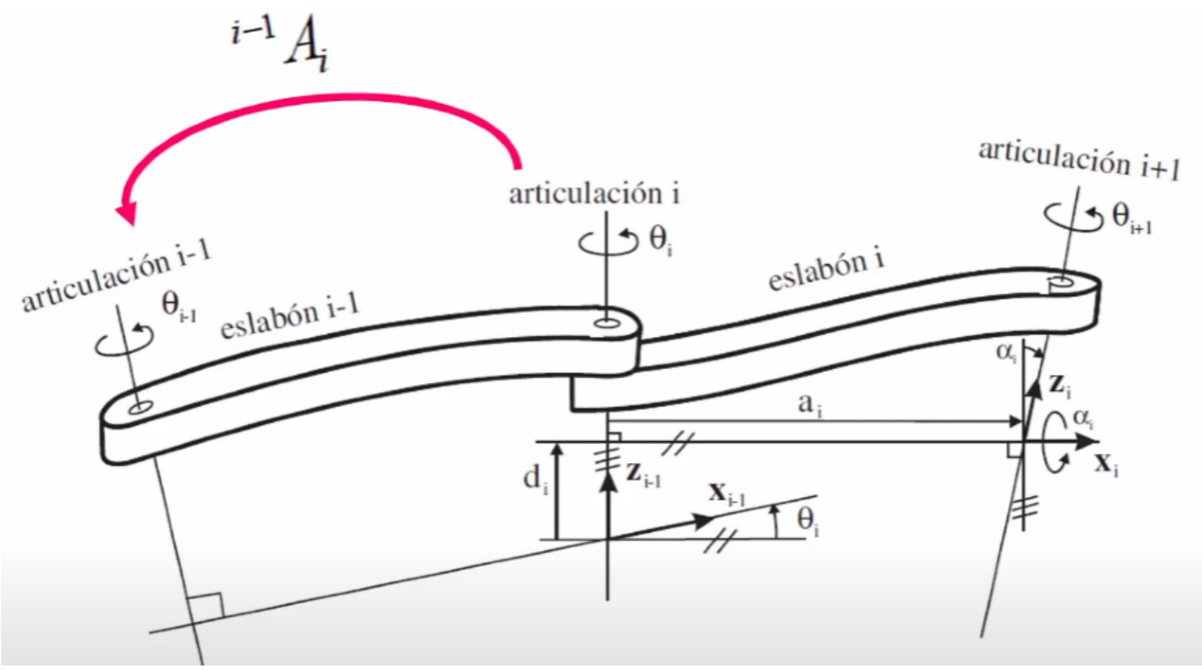
α_i : Torsión del eslabón

θ_i :Eslabón tipo rotativo

d_i : Eslabón tipo prismático

El ai es la longitud del eslabón, el alphai es el chingueo del eslabón (cuán mal orientado está con respecto al anterior eje de rotación), el titai si es una articulación rotativa o di si es una articulación deslizante o prismática.

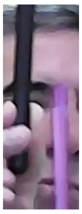
Denavit-Hartenberg va a determinar que el **eje Z pase por donde haya articulaciones rotativas o deslizantes (prismáticas)**.



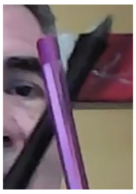
Para el método, Denavit-Hartenberg propone **cuatro transformaciones:**

$$^{i-1}A_i = T(x,\alpha_{i-1})T(a_{i-1},0,0)T(z,\theta_i)T(0,0,d_i)$$

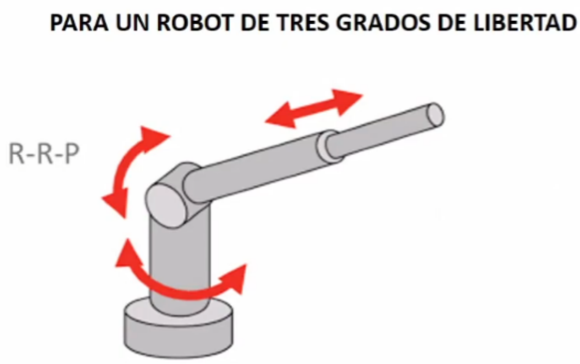
- di = si la articulación tiene un grosor o se modifica su largo. Es fijo el ancho también.



- titai = con respecto al eje z.
- desplazamiento ai = que no va a variar, es fijo el largo del eslabón.
- alphai = una torsión entre las articulaciones, tampoco varía.



Tendremos **tantas matrices como articulaciones tenga el robot. Si tenemos un robot de 3 grados de libertad, tendremos 3 matrices.**



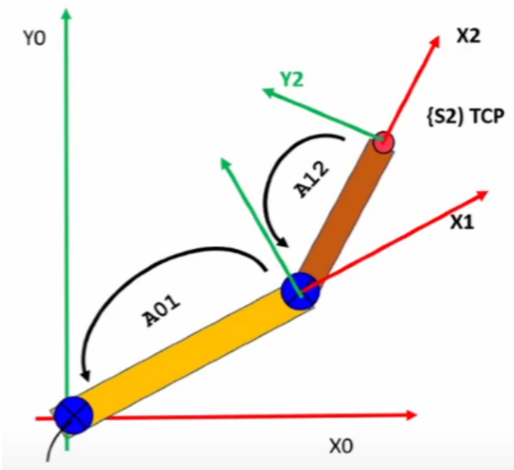
$$^{0}\mathbf{A}_3 = {}^{0}\mathbf{A}_1\,{}^{1}\mathbf{A}_2\,{}^{2}\mathbf{A}_3$$

(de 3 a 2, de 2 a 1, de 1 a 0, siendo 0 la base)

Clase 01/11

<u>Pasos para Denavit-Hartenberg</u>

Supongamos el siguiente robot “no tan elemental”:



1. Definir todos los ejes del robot.
2. Definir todos los parámetros (son datos que nos van a dar).

Una vez definidos todos los ejes llenaremos la tabla de D-H

	θ	d	a	α
Artic 1	q1 *	0	L1	0
Artic 2	q2 *	0	L2	0

En MatLab se pueden definir como:

```
q=[q1 q2]
d=[0 0]
a=[3 1.5]
alfa=[0 0]
```

3. Crear las matrices (una por cada articulacion, que también nos van a dar).
4. Multiplicarlas DESDE LA PUNTA DEL EFECTOR FINAL (va al final de la multiplicación) HASTA LA BASE (que va al principio de la multiplicación).

<u>En MatLab</u>:

1. En **ejercicioDH_1.m** (para el robot de arriba)
2. Escribir y ejecutar los parámetros (línea 4)
3. Tomaremos la matriz generica de Denavit-Hartenberg (escrita ya en ese .m)
4. Hay i=1, i=2, una por cada articulación.
5. Ejecutar cada una.
6. Ejecutá la multiplicación de las matrices.
7. Definí los valores de q1, q2 según lo que se te pida, se hace un eval en esa misma sección. Ejecutá la sección.
8. Te da como resultado una matriz como la que sigue:

```
ans =

    0.0000   -1.0000         0    0.0000
    1.0000    0.0000         0    4.5000
         0         0    1.0000         0
         0         0         0    1.0000
```

La última columna es la importante, que **nos da los valores de x; y; z**.

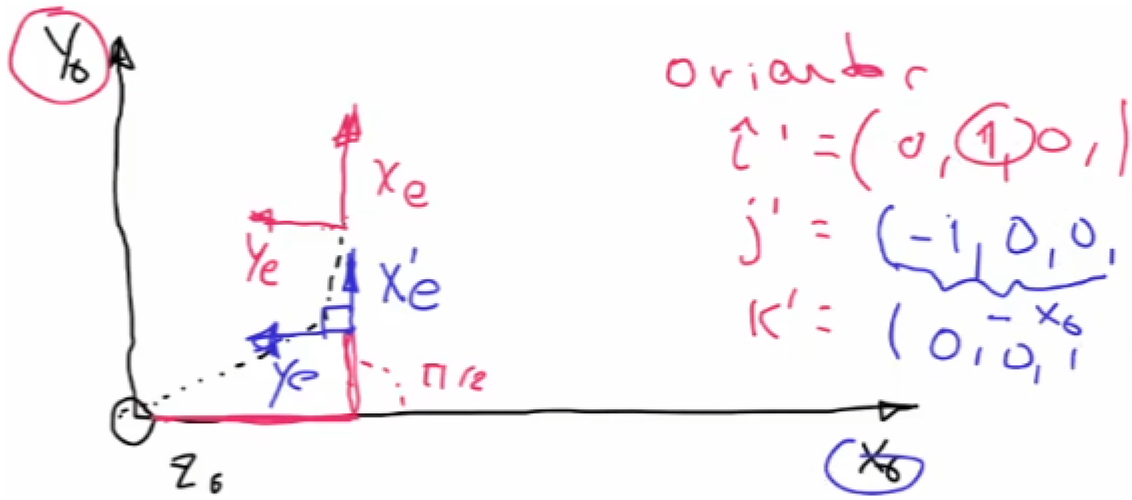
La primera, segunda y tercer columna, que corresponden a los versores i, j y k que **están en el efector final** nos dicen a dónde apuntan en relación a la base.
Las filas corresponden al x, y, z de la base.

Por ejemplo acá:

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

- El versor i/x del efector final está apuntando a "y" de la base (por eso hay un 1 en la segunda fila de la primer columna).
- El versor j/y del efector final está apuntando a "-x" de la base (por eso hay un -1 en la primer fila de la segunda columna).
- El versor k/z del efector final está apuntando a "z" de la base (por eso hay un 1 en la tercer fila de la tercer columna).

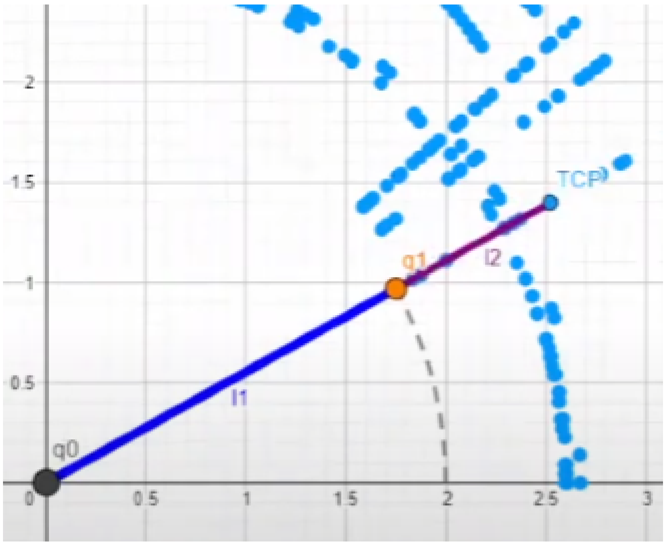
Por ejemplo, los versores en este ejemplo valen:



Entonces el método Denavit-Hartenberg no solo nos permite saber a dónde está el efector final en términos de x, y, z sino también saber la orientación del TCP.

Normalizamos todos los robot del planeta. Existe una única matriz 4x4 (16 elementos) **que puede describir cualquier robot.** Incluso si el robot tiene 128 articulaciones, por ejemplo, ya que se hará una multiplicación sucesiva de 128 matrices 4x4 que se irán multiplicando una a una para llegar a la final, que resume todo el robot.

Denavit-Hartenberg para otro robot "no tan tan elemental"



En MatLab:

1. En **robntan.m** (para el robot de arriba)
2. p= es la articulación deslizante.
3. q1 = articulación rotativa.
4. l1 = longitud del eslabón rotativo.
5. L = es l1

Clase 03/11

<u>El robot asesino de la página 100</u> (re contra mil no elemental, seguramente no lo va a tomar):

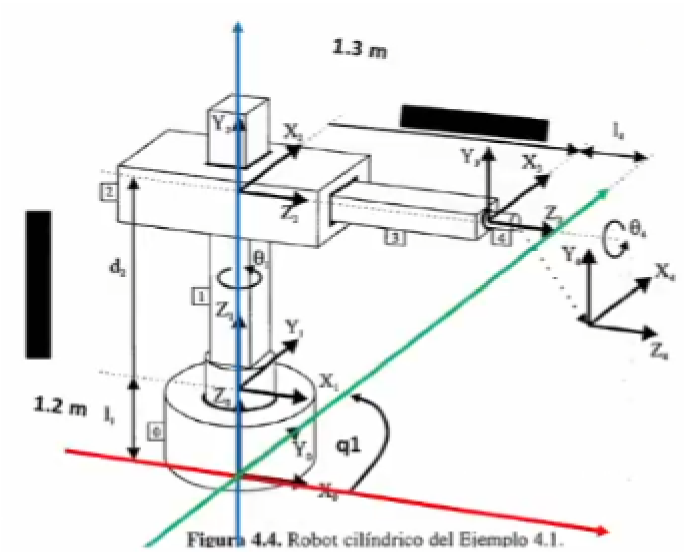


Figura 4.4. Robot cilíndrico del Ejemplo 4.1.

- q1 es la rotativa de la base del robot.
- d2 es la articulación prismática que sale de la base (puede estar completamente plegada para abajo, es decir, en 0)
- d3 es la otra articulación prismática que entra y sale desde la parte superior del robot (que muestra un 3).
- q4 es el destornillador, siempre será 0 (o debería).

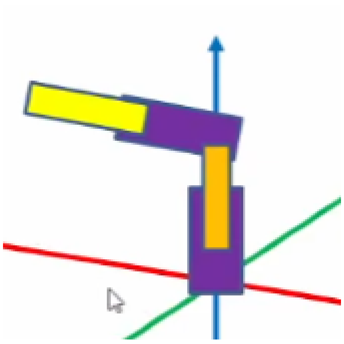
Ejemplo Denavit-Hartenberg (CD):

q1 = pi, q4 = 0, d2 = 0.8, d3 = 0.7

El robot gira 180° al eje de -X. Como la longitud del destornillador es fija de 1.3, y decimos que d3 es 0.7, entonces será a -2.

En y, el robot tiene una base fija de 1.2m, como d2 es 0.8, la altura será de y=2.

q4 siempre es cero.



-0.0000	0.0000	-1.0000	-2.0000
-1.0000	-0.0000	0.0000	0.0000
0	1.0000	0.0000	2.0000
0	0	0	1.0000

En MatLab:

1. En **ejercicio_de_barrientos.m** (para el robot de arriba)
2. Cambiar los parámetros q1, d2, d3, q4

- Estudiar PDFs teóricos de tipos de robots (los que dio en las primeras clases)

-- EL FIN --