Material Adicional: Cinemática Inversa

El problema de Cinemática Inversa hace referencia al cálculo de las variables articulares en función de los parámetros del extremo operativo. Este problema, para robots tipo serie, puede tener solución única, múltiples, infinitas o puede no tener solución.

Existen varios métodos para hallar estas soluciones. Uno es el método *Numérico*, que deriva en distintos tipos de algoritmos numéricos, donde, por lo general, se logra una solución (en el mejor de los casos) por cada ejecución de la búsqueda, normalmente la más "cercana" al vector inicial desde donde se comienza a iterar.

Otro método es el *Geométrico*, que consiste en hallar ecuaciones para las variables articulares, en función de los datos cartesianos del extremo, a partir del análisis de figuras geométricas que se plantean en diagramas cinemáticos. La ventaja de este método es que permite manipular y trabajar las ecuaciones de forma tal de obtener un sistema que permita hallar varias o todas las soluciones posibles, para cada situación. La desventaja es que requiere un trabajo dedicado para cada robot, y que para más de 3 GDL suele ser necesario trabajar en el espacio tridimensional, y los gráficos y análisis de figuras suele ser complejo.

Otra gran diferencia entre las ecuaciones y los métodos numéricos, es que las primeras tienen un tiempo de cálculo fijo y predecible, en el segundo caso podrían darse todas las situaciones conocidas sobre estos métodos (converger en distintos tiempos, divergir, etc.), además de que en los numéricos siempre se juega con un error aceptable, mientras más chico sea, más probable es que el algoritmo tenga demora o problemas.

Se recomienda revisar el capítulo 3.3 de Spong (Robot Modeling and Control) para mayor profundidad y ejemplos de lo que sigue.

Robots de más de 3GDL

Es muy común en la industria encontrarse con robots de 6GDL. En el caso general, el problema de C.I. para un robot serie de 6 GDL es:

$$(q_1, q_2, q_3, q_4, q_5, q_6) = F(T)$$

Donde T es la matriz de transformación homogénea que contiene la posición y orientación del extremo operativo. El mismo problema se podría reescribir de otras formas, representando la posición y orientación del extremo de alguna manera alternativa, como, por ejemplo:

$$(q_1, q_2, q_3, q_4, q_5, q_6) = G(x, y, z, \alpha, \beta, \gamma)$$

No hay que perder de vista que de los 16 elementos de la matriz T solo se pueden extraer 6 ecuaciones independientes entre sí.

En cualquier caso, hay que hallar las 6 ecuaciones para las 6 variables articulares. La tarea de establecer figuras geométricas para encontrar las relaciones entre cada articulación y las variables del extremo, puede ser complicada o sumamente engorrosa.

En este documento se explicará cómo se puede trabajar con el método geométrico para robots de 6 GDL, y se mostrarán 2 casos de aplicación que representan a la mayoría de los robots industriales del mercado.

Los mismos conceptos se pueden adaptar a robots de menos de 6GDL sin problema, y se pueden extender a más de 6GDL, pero en este último caso, hay que considerar que, si no se tienen suficientes restricciones o datos de entrada, el problema puede quedar indeterminado sin importar el planteo geométrico que se haga.

Desacoplo Cinemático

Lo primero a considerar es el método de desacople cinemático, también conocido como Método de Pieper. Este método propone dividir la cadena cinemática del robot en 2 cadenas separadas, más simples.

Este método aplica solo a robots con muñeca esférica, es decir, robots que tienen las últimas 3 articulaciones de rotación, con ejes que se cortan en un mismo punto. Aunque mecánicamente esto sea complejo de ver, lo cierto es que la gran mayoría de los robots industriales cumplen con esta característica.

En la Figura 1 se observa un esquema cinemático de un robot de 6 GDL que cumple con la condición anterior: los últimos 3 GDL son articulaciones de rotación que tienen un punto de cruce en común. Dicho punto suele denominarse como "centro de la muñeca" y siempre puede calcularse si se conoce la posición y orientación del extremo operativo.

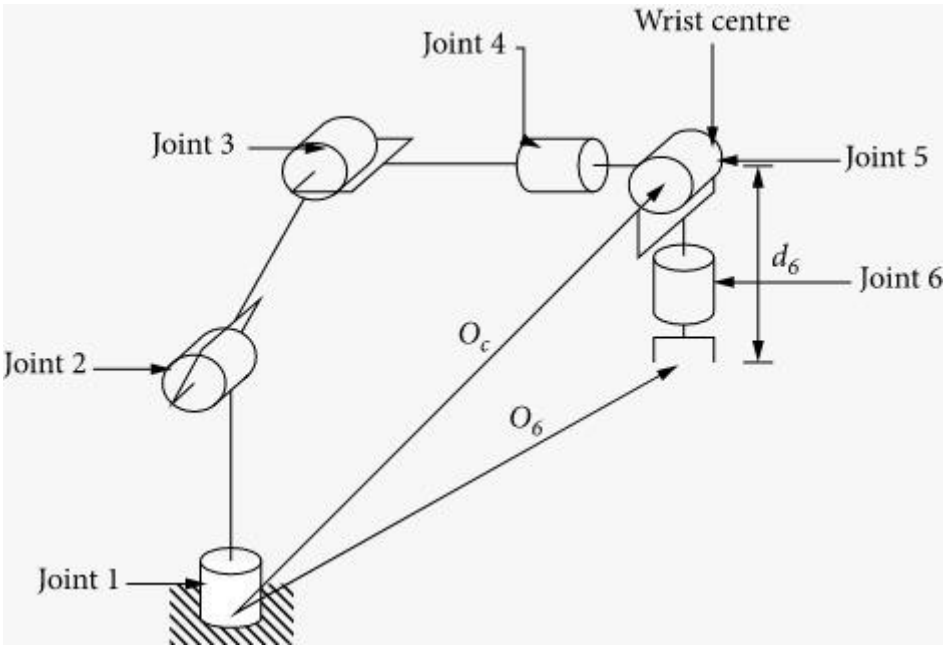

Figura 1: esquema cinemático de robot típico de 6GDL con muñeca.

Si se analiza la convención de DH, el centro de la muñeca coincide con el sistema de referencia 5, y el extremo operativo coincide con el 6, y siempre será constante la separación entre

ambos sistemas, específicamente la distancia d_6. Por lo tanto, en este contexto siempre se podrá calcular el centro de la muñeca con:

$$\bar{p}_c = \bar{d}_6 - d_6\bar{\boldsymbol{a}}_{\boldsymbol{6}}$$

Donde:

- $\bar{p}_c$: vector de posición del Centro de la Muñeca, respecto del origen.
- $\bar{d}_0^6$: vector de posición del extremo final respecto del origen (dato de la matriz de transformación homogénea del extremo).
- d_6: parámetro de DH que indica la distancia entre el extremo y la muñeca.
- $\bar{\boldsymbol{a}}_{\boldsymbol{6}}$: versor que indica la dirección del extremo y, por lo tanto, la dirección entre el extremo y la muñeca (dato de la matriz de transformación homogénea del extremo).

Una vez conocido este punto, se puede dividir el problema en 2, más simples y consecutivos.

Primer problema: Posición

Aplicando la ecuación anterior se puede calcular $\bar{p}_c$ y plantear el problema siguiente:

$$(q_1, q_2, q_3) = F_1(\bar{p}_c)$$

Que se representa con la Figura 2.

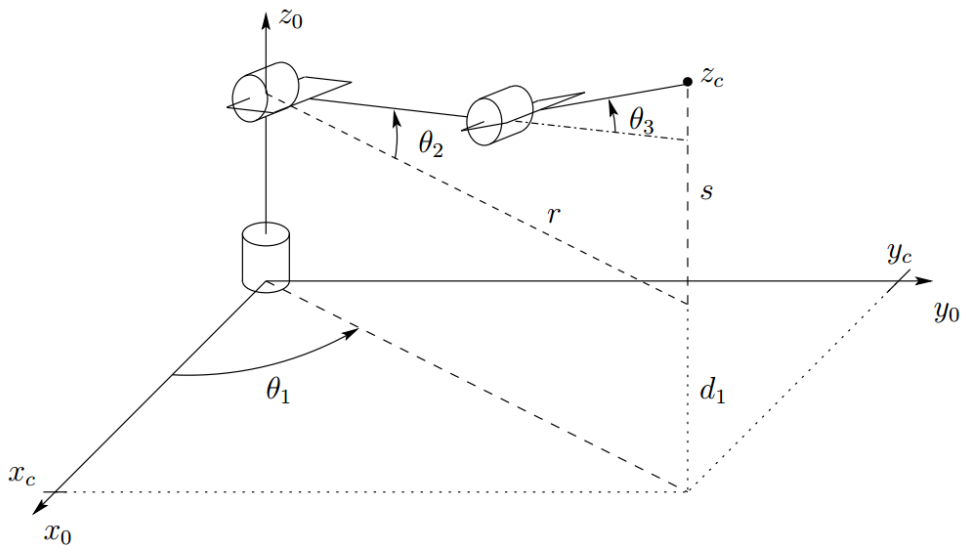

Figura 2: primeros 3 GDL.

Donde:

$$\bar{p}_c = (x_c, y_c, z_c)$$

Este problema tiene 4 soluciones en el caso general. El punto $\bar{p}_c$ se encuentra en un plano que se puede describir con q_1 y con $q_1 + \pi$. Luego, con q_2 y q_3 se encuentran las soluciones "codo arriba" y "codo abajo", para cada q_1.

Es importante tener en cuenta que este subproblema tiene un punto singular cuando $\bar{p}_c$ está alineado con el eje z_0, ya que existirían infinitas soluciones en tal caso.

Otra cuestión a tener en cuenta es que los robots comerciales pueden tener algún offset o parámetro d distinto de cero, y el esquema cinemático varía en alguna articulación. Se recomienda revisar la sección 3.3.3 de Spong para más profundidad.

Segundo problema: Orientación

Una vez hallados los valores de las primeras 3 articulaciones, es posible calcular, por cinemática directa, la matriz ${}^0T_3$, y considerando la matriz de entrada T, se puede plantear el problema:

$$ {}^3T_6 = ({}^0T_3)^{-1}.T $$

Donde:

- ${}^3T_6$: matriz de cinemática directa con q_4, q_5 y q_6 como variables.
- $({}^0T_3)^{-1}.T$: producto de la inversa de la matriz obtenida por los resultados del problema 1, y matriz dato del problema.

Esta ecuación matricial debe llevar a 3 ecuaciones individuales, una para cada q desconocido. Es un problema general de orientación en el espacio que se puede resolver de múltiples maneras. En Spong se plantea el problema de forma general (sección 3.3.4), pero acá se presenta una alternativa geométrica específica.

La alternativa consiste en analizar los sistemas de referencia de la ecuación anterior con una herramienta computacional que asista en el graficado, y buscar las relaciones geométricas necesarias. El siguiente ejemplo se resuelve desacoplando posición de orientación, y en la segunda parte se aplica este método.

Ejemplo de Desacoplo Cinemático – Robot ABB IRB140

A) Desacople

El IRB140 de ABB es un robot de 6GDL con muñeca esférica muy popular en la industria. Posee la muñeca clásica de este tipo de robots, por lo tanto, se puede aplicar el método mencionado. En la Figura 3 se representa el robot con sus ejes principales.

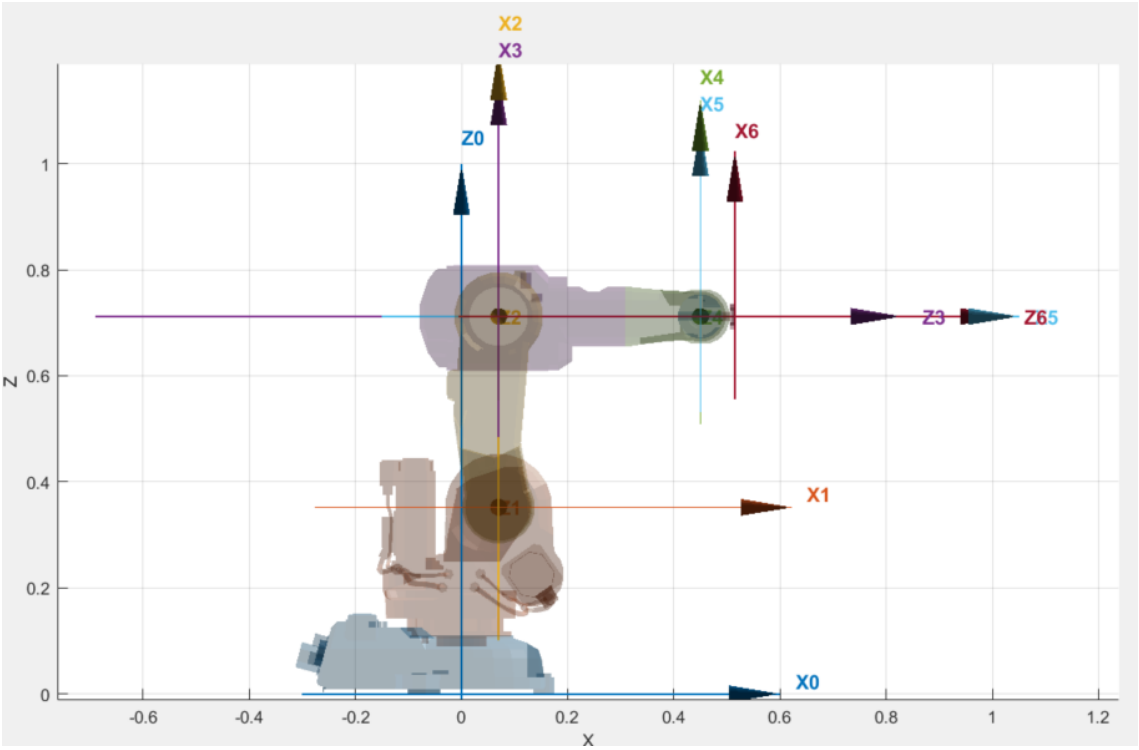

Figura 3: ABB IRB140 con los ejes principales según DH.

Vamos a suponer que tenemos una matriz T de dato, y queremos calcular todas las posiciones articulares posibles que cumplen con dicha matriz, es decir:

$$(q_1, q_2, q_3, q_4, q_5, q_6) = F(T)$$

Adicionalmente trabajaremos con límites articulares entre $-\pi$ y π, a pesar de que el robot tiene otros valores. Esto es así para poder tener todas las soluciones matemáticas en el círculo trigonométrico de cada articulación. Las soluciones reales son una consecuencia lógica de la diferencia entre estos límites y los reales.

En Matlab podemos representar el robot así:

```
>> dh = [
>>      0.000   0.352   0.070 -pi/2   0;
>>      0.000   0.000   0.360   0.000 0;
>>      0.000   0.000   0.000 -pi/2   0;
>>      0.000   0.380   0.000   pi/2   0;
>>      0.000   0.000   0.000 -pi/2   0;
```

```
>>      0.000   0.065   0.000   0.000 0];
>> R = SerialLink(dh);
```

Para tener una representación más clara vamos a adoptar variables articulares distintas de cero y a manipular un poco el plot, incluyendo los sistemas de referencia, como se ve en la Figura 4.

```
>> q =[35,-70,-35,35,-35,35]*pi/180;
```

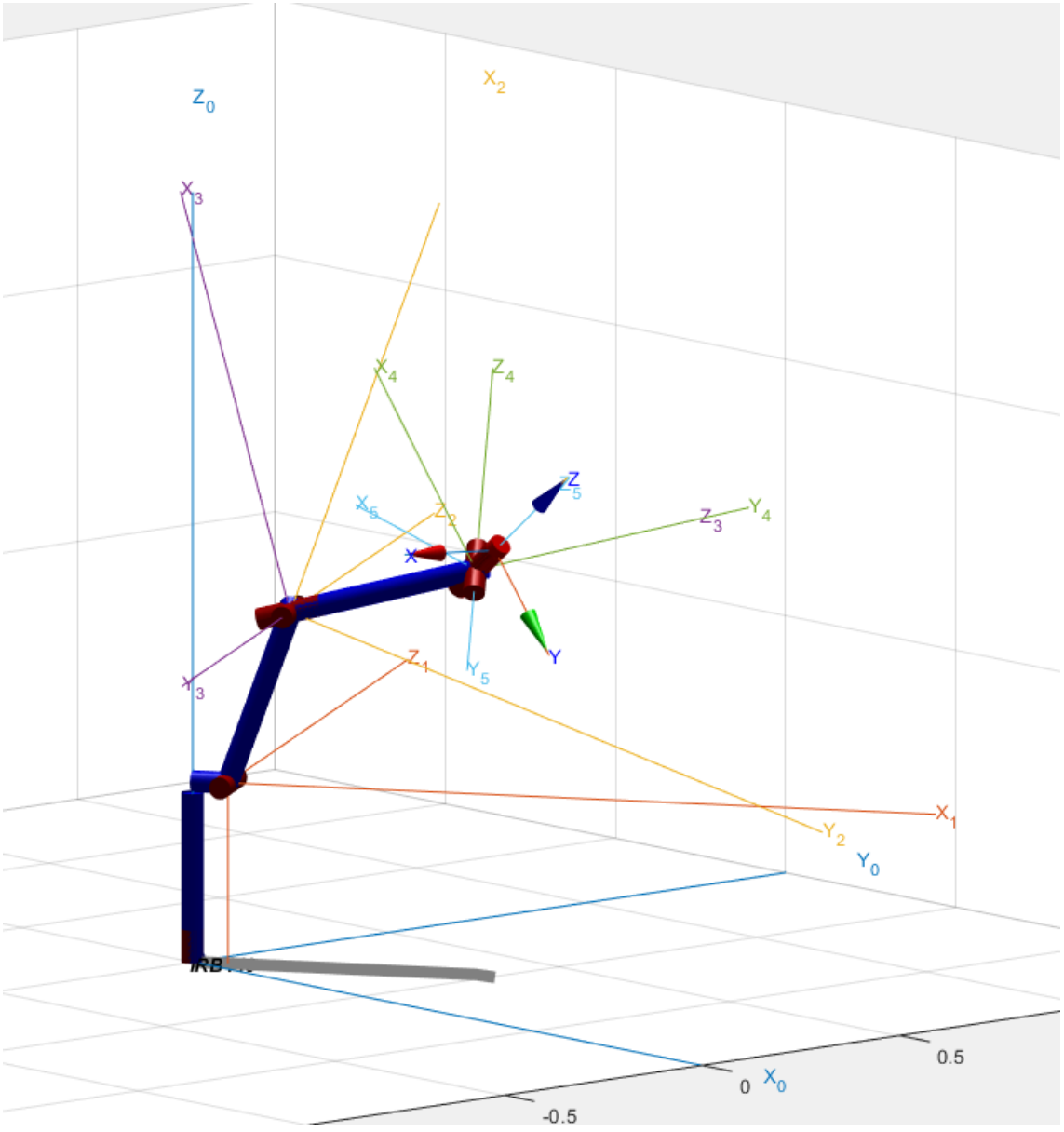

Figura 4: representación IRB140 en posición articular distinta de cero.

En la Figura 4 se puede ver que los sistemas 4 y 5 coinciden en origen, y que el sistema 6 (sin número) está desplazado de dicho origen, pero el eje Z (eje de la articulación) está alineado con los centros de los sistemas anteriores. Por lo tanto, los ejes articulares Z_4, Z_5 y Z_6 se cortan en un punto, formando una muñeca esférica.

En la Figura 5 se representan solos los sistemas, y con un trazo en negro se representa el vector que debemos recorrer para llegar del extremo del robot (donde tenemos la matriz T de dato) al centro de la muñeca.

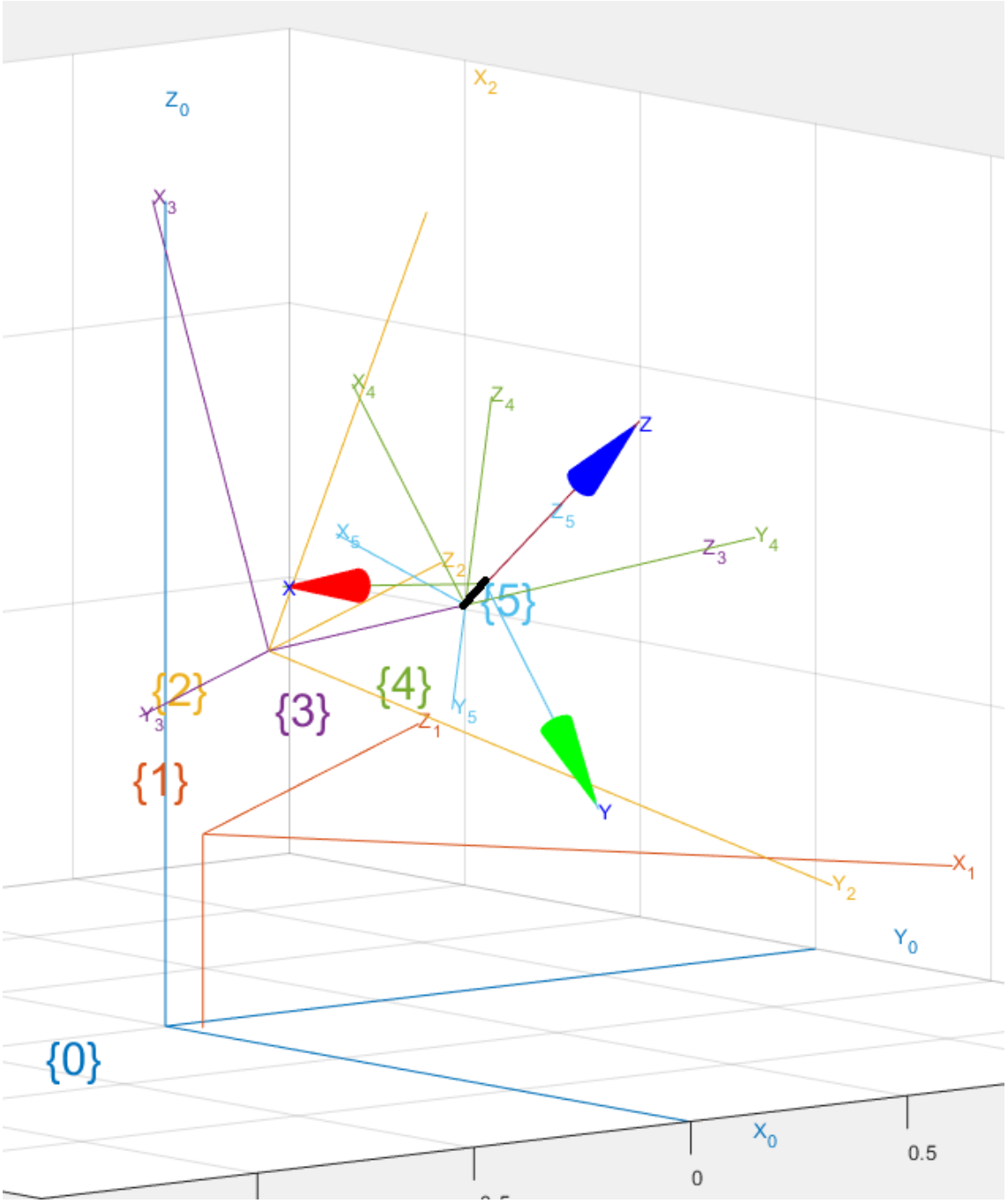

Figura 5: sistemas de referencia del IRB140 para una posición articular no nula.

Antes de empezar el cálculo vamos a desacoplar de la cadena cinemática las posibles matrices de base y de tool. Es decir, la matriz T de dato representa posición y orientación del extremo respecto de la base, pero dicho extremo está afectado por la matriz de "tool", y dicha base está afectada por la matriz de "base". Para lograr un trabajo genérico y que sirva para

cualquier caso de este robot, desacoplaremos dichas matrices, ya que son constantes y conocidas:

$$T = T_{base}.\,{}^{0}T_{6}.T_{tool} \rightarrow {}^{0}T_{6} = (T_{base})^{-1}.T - (T_{tool})^{-1}$$

En Matlab puedo pisar el valor de T sin cambiar el nombre:

```
>> T = invHomog(R.base) * T * invHomog(R.tool);
```

La función "`invHomog()`" calcula la inversa de una matriz homogénea de la forma particular, por lo que es más eficiente y precisa.

En estas condiciones se procede a calcular el centro de la muñeca:

```
>> p = T(1:3,4) - dh(6,2) * T(1:3,3);
```

Donde "`T(1:3,4)`" es el vector de posición del extremo, "`dh(6,2)`" es la distancia "d" de DH de la última articulación (trazo negro de la figura anterior), y "`T(1:3,3)`" es el versor Z del extremo.

De esta forma nos quedará desacoplado el sistema. El primer paso es resolver el problema de la Figura 6, donde se conoce el origen del sistema 4 (vector trazado en negro), que es el centro de la muñeca.

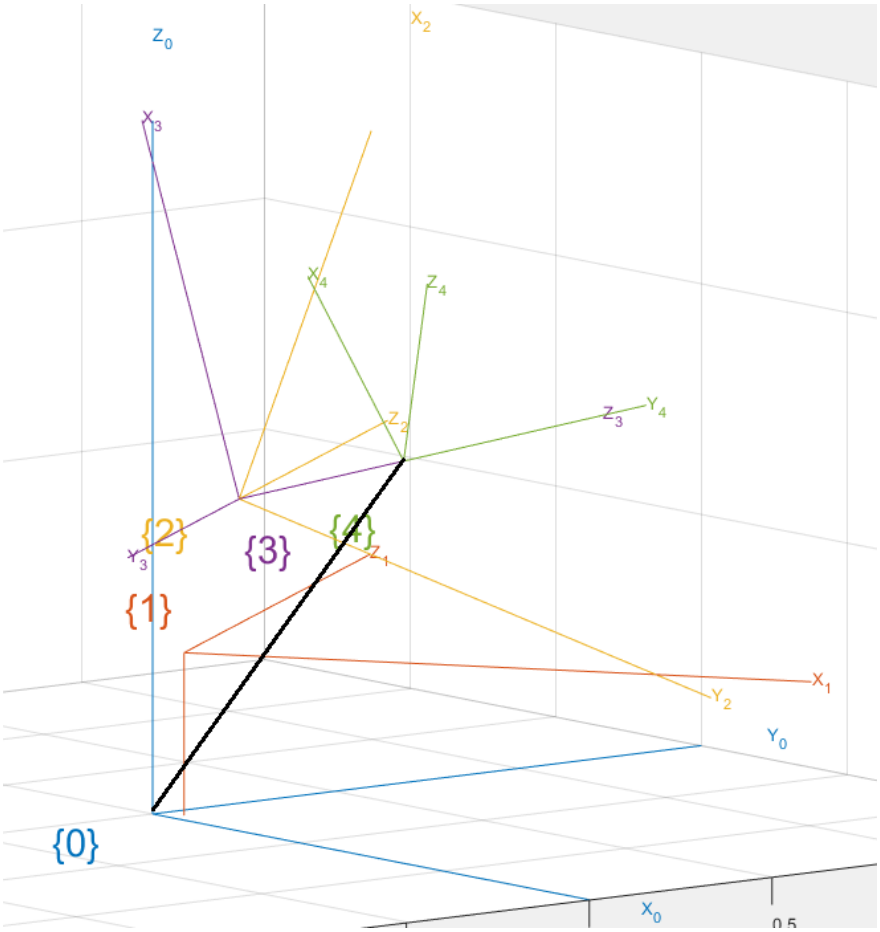

Figura 6: primer problema, posición de la muñeca.

B) Primer problema: Posición

Se puede observar en la Figura 6 que el extremo de este nuevo problema (sistema 4, dato) se mueve en un plano que gira alrededor de Z_0, con dicho giro dado por q_1. Analizado también en la Figura 2. Luego, en este plano giratorio vemos que la posición del punto extremo depende de q_2 y q_3. El plan es calcular primero q_1 y luego trabajar sobre dicho plano.

Cálculo de q_1

Para el cálculo de q_1 vemos que simplemente debemos aplicar un atan2 a las coordenadas XY del extremo (origen del sistema 4), ya que éstas están respecto del sistema de la base, y nos dan exactamente el ángulo entre el eje X_0 y X_1, debido a que el X_1 forma parte del plano mencionado. Esto se puede observar en la Figura 7.

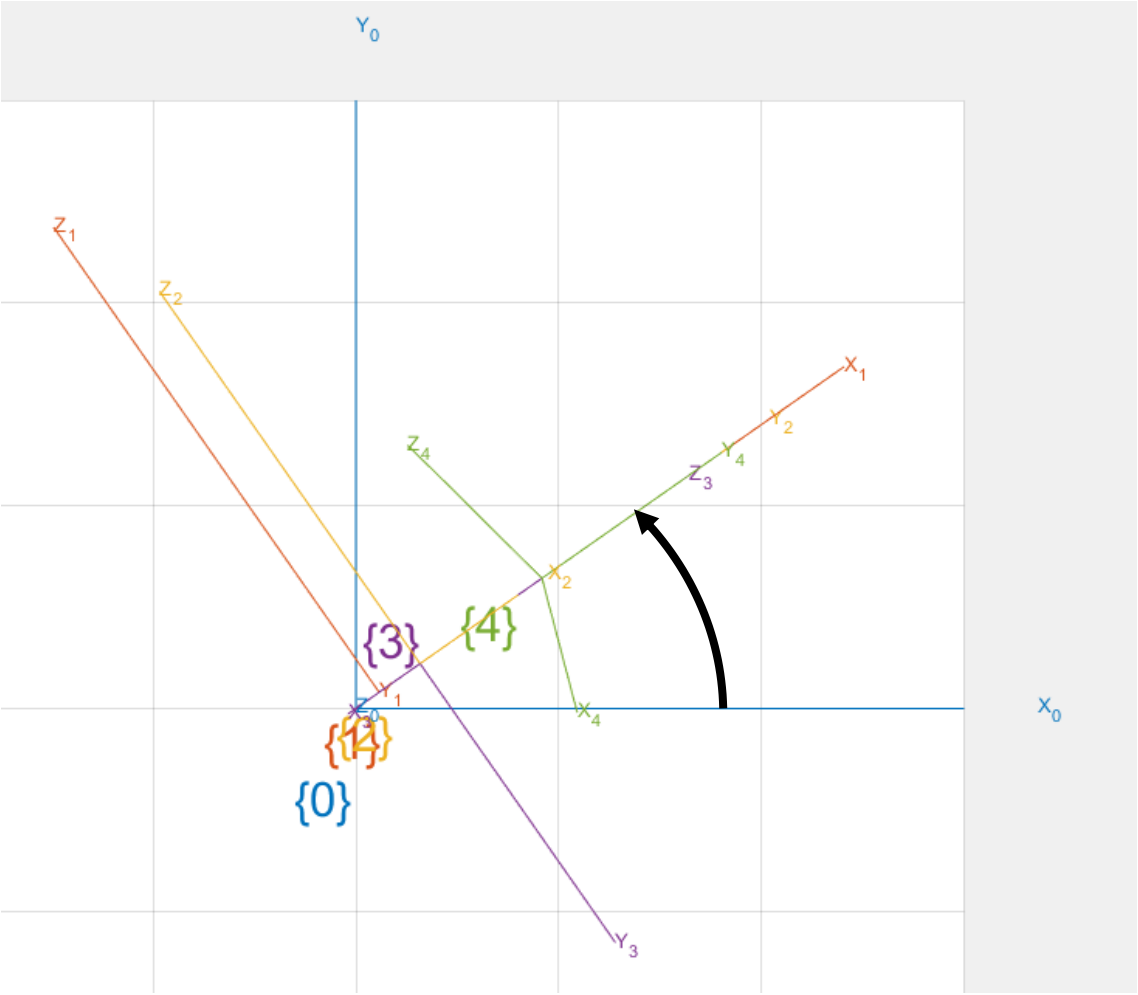

Figura 7: plano para q_1.

Por lo tanto:

```
>> q1(1) = atan2(p(2), p(1)); % -pi >= atan2 <= pi
```

Es importante notar que existen 2 valores de q_1 que pueden cumplir con la condición geométrica planteada: el calculado y el opuesto a 180º. Por lo tanto:

```
>> if q1(1) > 0
>>     q1(2) = q1(1) - pi;
>> else
>>     q1(2) = q1(1) + pi;
>> end
```

Y de esta forma tenemos 2 valores de q_1 que son consistentes con el análisis geométrico, y que están en el rango $-\pi$ a π.

También es importante saber que algunos robots presentan un offset en el hombro, la estructura no está completamente alineada con el eje principal de rotación, sino que tienen un pequeño desplazamiento, a veces de muy pocos milímetros. Esta situación se observa en la figura 3.16 de Spong, y su resolución no afecta las conclusiones obtenidas aquí, solo requiere una consideración extra en el cálculo.

Cálculo de q_2

Para el cálculo de q_2, y para todos los que sigan, adoptaremos el criterio de ir moviéndonos en la cadena cinemática de acuerdo a lo que ya calculamos. Es decir, ahora que tenemos q_1 podemos calcular la matriz ${}^0T_1$ y simplificar el problema restante referenciando nuestro punto de dato respecto del sistema 1, y no del sistema 0. Como hay 2 valores de q_1, el problema se bifurca. Acá se presenta el cálculo del punto "p" (centro de muñeca) respecto de uno de los valores de q_1, en la implementación real del código se debe calcular todo con los dos valores.

```
>> T1 = R.links(1).A(q1).double;
>> p = invHomog(T1) * [p; 1];
```

El código "`R.links(1).A(q1).double`" calcula una matriz de transformación homogénea de acuerdo a los 4 parámetros de DH del link 1 y el valor calculado de "`q1`". Ahora, el vector "p" es la posición del extremo respecto del sistema 1, por lo tanto, si observa el sistema 1 en las figuras anteriores (naranja) se puede concluir que la situación actual es la de la Figura 8.

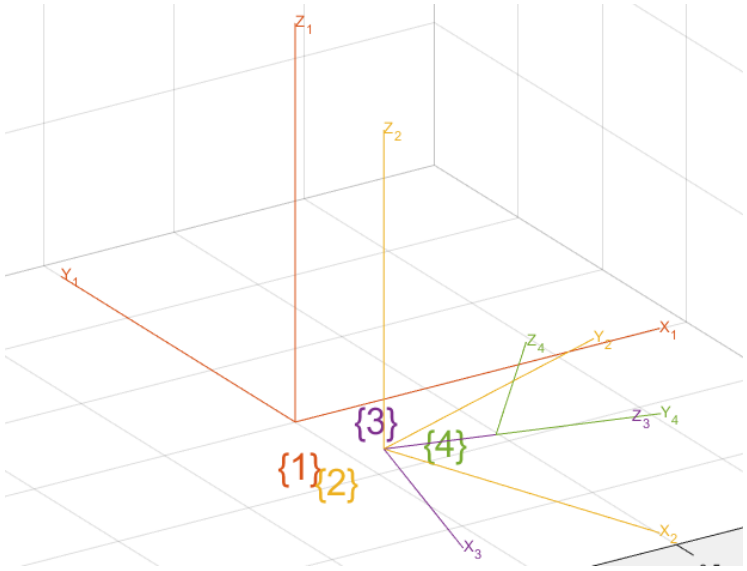

Figura 8: sistema 4 respecto del sistema 1.

Este esquema puede resultar confuso inicialmente, pero analizándolo detenidamente se debe concluir que corresponde a lo mostrado en la Figura 6. Es recomendable detenerse hasta comprenderlo.

En la Figura 9 se presenta el plano dado por X_1 e Y_1 con las indicaciones necesarias para el cálculo de q_2.

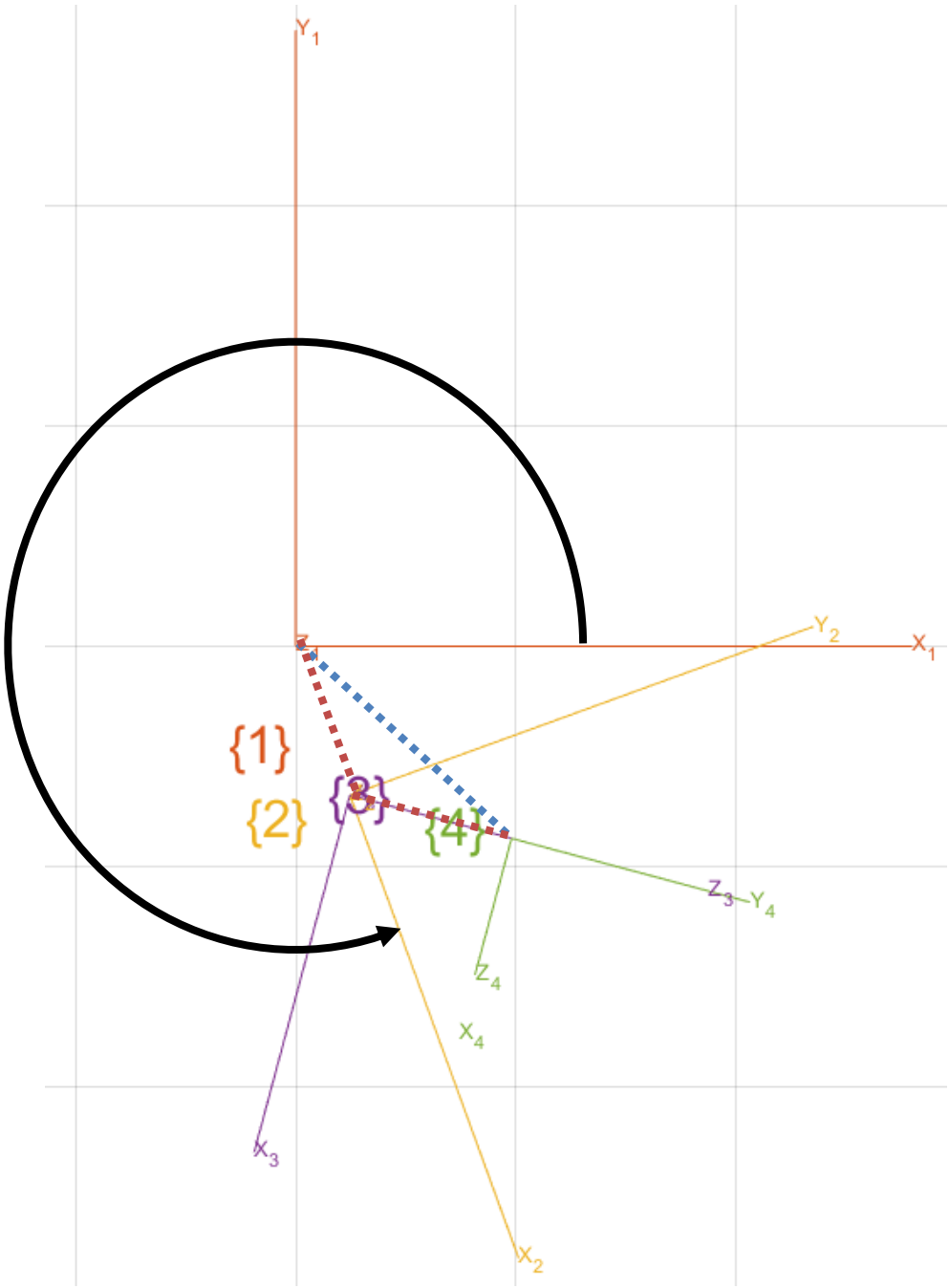

Figura 9: plano para q_2.

Se observa el clásico problema de “codo arriba” y “codo abajo”. El ángulo buscado ahora es q_2, que es el medido desde X_1 hasta X_2, y se puede hallar con:

```
>> B = atan2(p(2), p(1));
>> r = sqrt(p(1)^2 + p(2)^2);
>> L2 = R.links(2).a;
>> L3 = R.links(4).d;
>> G = (acos((L2^2 + r^2 - L3^2) / (2 * r * L2)));
>> q2(1) = B - real(G);
>> q2(2) = B + real(G);
```

El código anterior da los 2 valores de q_2 posibles para el punto "p" calculado, pero hay que recordar que este punto se obtuvo para el primer valor de q_1, por lo tanto, hay que realizar todo el mismo cálculo para el segundo valor de q_1. En conclusión, deberíamos tener en este punto 2 valores de q_1 y 2 valores de q_2 para cada q_1.

Es importante recordar también que "G" puede ser imaginario; en tal caso el punto buscado no es alcanzable por las dimensiones del robot, y continuar el procedimiento ignorando la parte imaginaria (con "`real(G)`") nos lleva a resultados que deben ser tratados con criterio.

Por último, también es importante considerar que algunos robots presentan un codo fijo en la segunda articulación, debido a un parámetro DH distinto de cero, normalmente a_3. Esto modifica un poco el esquema e introduce no-perpendicularidades, pero el problema sigue siendo igual de definido; solo se requieren algunos cálculos intermedios.

Cálculo de q_3

El cálculo de q_3 se puede realizar a partir del gráfico mostrado en la Figura 9, pero para continuar con el trabajo matricial, lo recomendable es lo siguiente:

```
>> T1 = R.links(1).A(q1).double;
>> T2 = T1 * R.links(2).A(q2).double;
>> p = invHomog(T2) * [p; 1]; % p respecto de {2}
```

Observe que al haber 2 valores de q_1 y 4 de q_2, en total habrá 4 matrices ${}^0T_2$, 4 puntos "p", y por lo tanto 4 cálculos de q_3, que se representan en la Figura 10, donde se observa un nuevo gráfico, donde vemos que q_3 va de X_2 a X_3 alrededor de Z_2.

En este caso vemos que el plano que contiene a "p" está conteniendo también a Z_3, y no a X_3, por lo tanto será necesario restar $\pi/2$ como sigue:

```
>> q3 = atan2(p(2), p(1)) - pi/2;
```

Entonces, el valor de q_3 debe ser calculado 4 veces: una vez para cada q_2, que son 2 valores por cada q_1. De esta manera estaría resuelta la primera parte del desacople. Debido a que cada conjunto (q_1, q_2, q_3) tendrá dos soluciones (q_4, q_5, q_6), se puede ordenar una matriz de solución como sigue:

```
>> qq(1,:) = [q1(1) q1(1) q1(1) q1(1) q1(2) q1(2) q1(2) q1(2)];
>> qq(2,:) = [q2(1) q2(1) q2(2) q2(2) q2(3) q2(3) q2(4) q2(4)];
>> qq(3,:) = [q3(1) q3(1) q3(2) q3(2) q3(3) q3(3) q3(4) q3(4)];
```

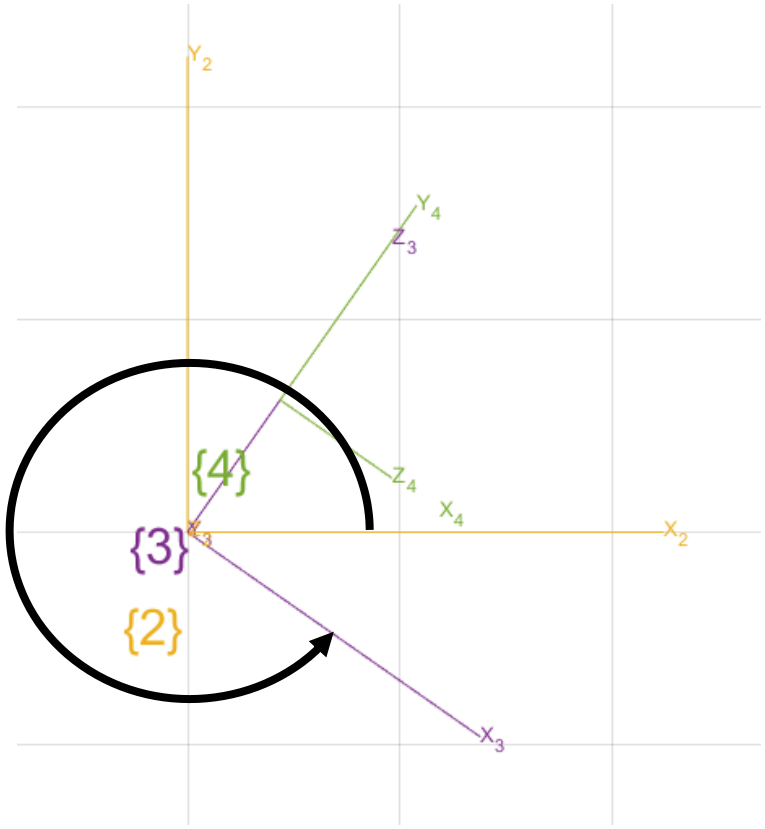

Figura 10: plano para q_3.

Siendo "qq" una matriz de 6x8 que tendrá en cada fila las soluciones del problema (que serán 8). Hasta este punto se repite de a pares cada columna, pero en la siguiente sección determinaremos una solución distinta por columna.

C) Segundo problema: Orientación

En este punto está resuelto el problema de posición, se han hallado todos los valores de q_1, q_2 y q_3 que dejan el centro de la muñeca del robot en el punto exacto desde el que se puede variar q_4, q_5 y q_6, para lograr que el extremo quede en la posición final y con la orientación final.

La situación actual es la de la Figura 11. El sistema de referencia $\{3\}$ está completamente determinado en posición y orientación, para 4 casos diferentes (por los 4 conjuntos de soluciones del paso anterior), y por lo tanto podemos expresar toda la posición y orientación del extremo (matriz T de dato inicial) respecto del sistema 3, como se comentó anteriormente:

$$ {}^3T_6 = ({}^0T_3)^{-1}.T $$

Es decir:

```
>> T1 = R.links(1).A(q1).double;
>> T2 = R.links(2).A(q2).double;
>> T3 = R.links(3).A(q3).double;
>> T36 = invHomog(T3) * invHomog(T2) * invHomog(T1) * T;
```

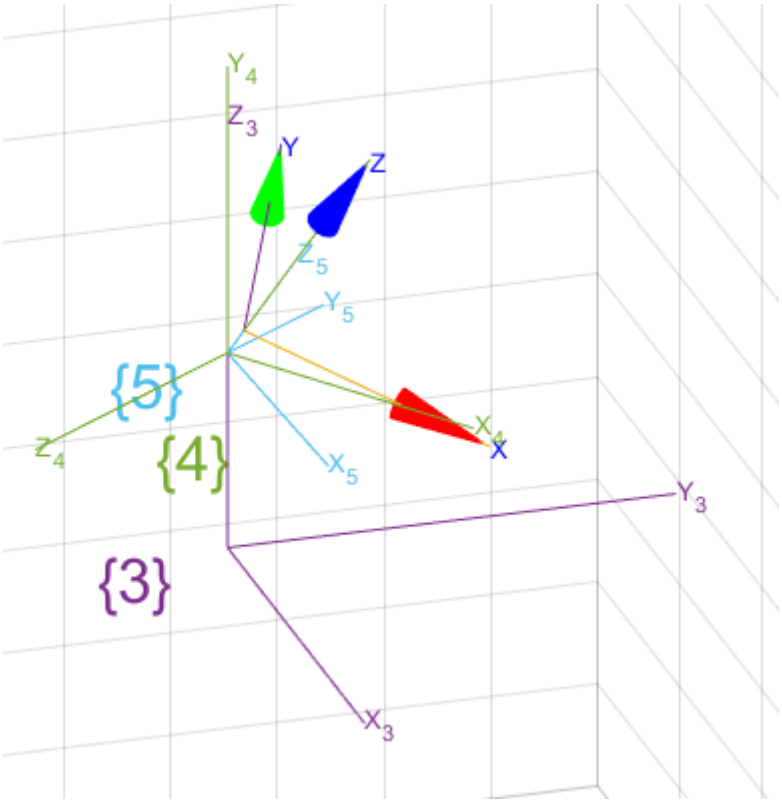

Figura 11: representación de ${}^{3}T_{6}$.

Donde "T36" es la matriz conocida que vamos a usar para hallar los valores articulares restantes, y que se representó en la figura 11. Si dicha figura la observamos respecto del plano $X_3 - Y_3$, vemos que el ángulo q_4, que va desde X_3 a X_4, se puede calcular simplemente como el ángulo de la proyección del Z en el plano en cuestión, ya que dicha proyección coincide siempre con el X_4. Para asegurarse de esto se puede analizar la asignación de DH o se pueden probar otras posiciones del robot. En la Figura 12 se observa el plano para q_4.

Observe que para que el cálculo de "T36" tenga sentido, los valores de q_1, q_2 y q_3 Deben ser de los índices "[1, 3, 5, 7]" de "qq", así no están repetidos.

Como en la matriz "T36" tenemos todos los datos del sistema del extremo, incluyendo el eje Z, respecto del sistema 3, entonces las coordenadas xy del eje Z de dicha matriz pueden ser usadas para calcular el ángulo en cuestión:

```
>> q4(1) = atan2(T36(2,3), T36(1,3));
```

Es muy importante notar que el problema de orientación tendrá 2 soluciones para cada matriz "T36" definida, debido a que se puede buscar un "codo arriba codo abajo", que sería más bien un "muñeca arriba muñeca abajo". Ya que si q_4 gira 180º desde cualquier posición, q_5 puede efectuar una rotación para dejar el extremo en la misma posición, y luego q_6 hace rotar el sistema del extremo para orientarlo adecuadamente. Por lo tanto:

```
>> if q4(1) > 0, q4(2) = q4(1) - pi; else, q4(2) = q4(1) + pi; end
```

Que es lo mismo que se hizo con q_1.

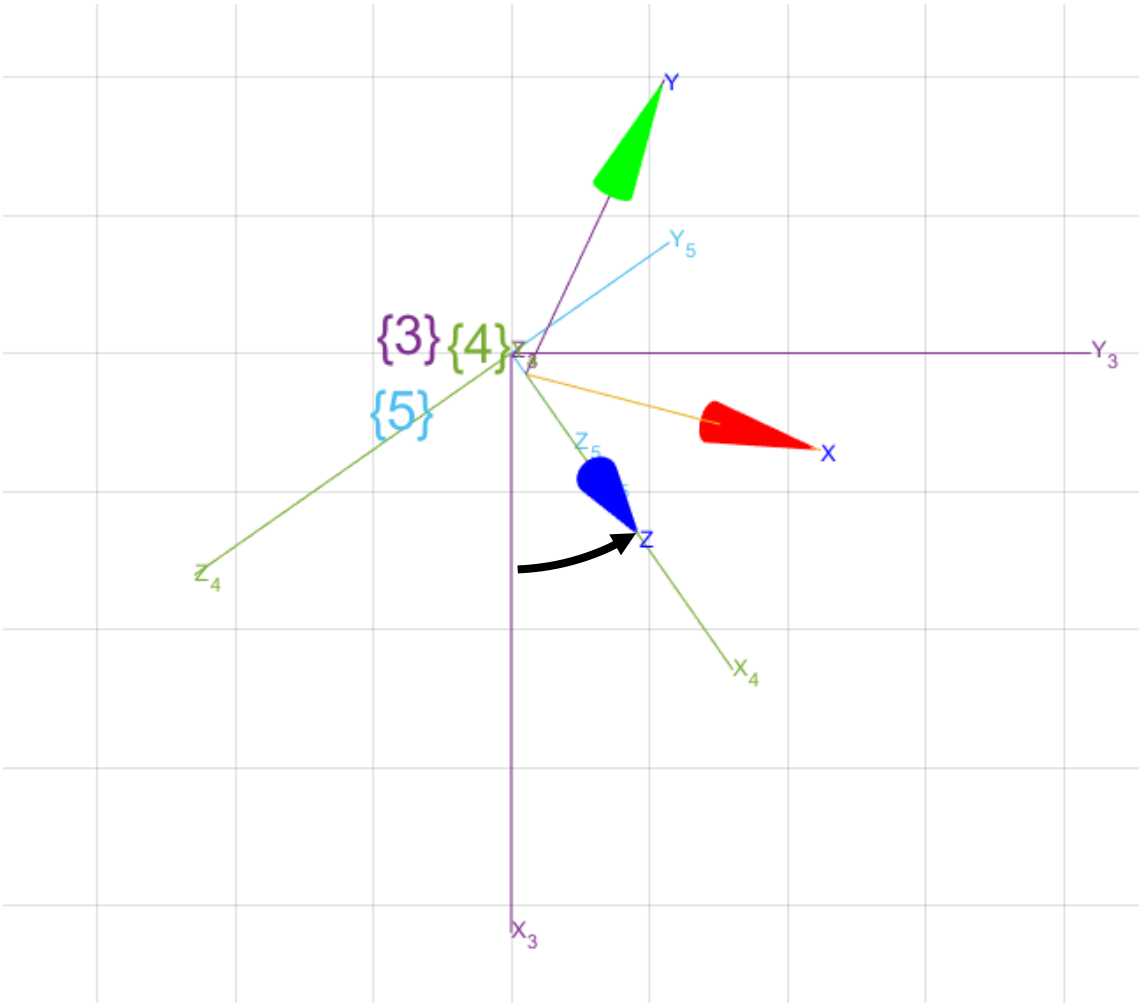

Figura 12: plano para q_4.

Ahora, para cada valor de q_4 buscaremos el q_5 calculando primero la matriz ${}^3T_4$ y referenciando la matriz final T (y su sistema final asociado) al sistema $\{4\}$:

```
>> T4 = R.links(4).A(q4(i)).double;
>> T6 = invHomog(T4) * T36;
```

Como vemos en la Figura 13, en ángulo q_5, que va desde X_4 hasta X_5 alrededor de Z_4, se puede calcular con proyecciones. Usaremos nuevamente una proyección del eje Z del extremo, por consideraciones que veremos posteriormente. Dicho eje difiere siempre en 90º de X_5, entonces:

```
>> q5(i) = atan2(T46(2,3), T46(1,3)) - pi/2;
```

Por último, el cálculo de q_6 queda muy simple considerando la Figura 14, donde podemos ver que q_6, que es el ángulo desde X_5 hasta X_6, alrededor de Z_5, se calcula simplemente con:

```
>> T5 = R.links(5).A(q5(i)).double;
>> T6 = invHomog(T5) * T6;
>> q6(i) = atan2(T6(2,1), T6(1,1));
```

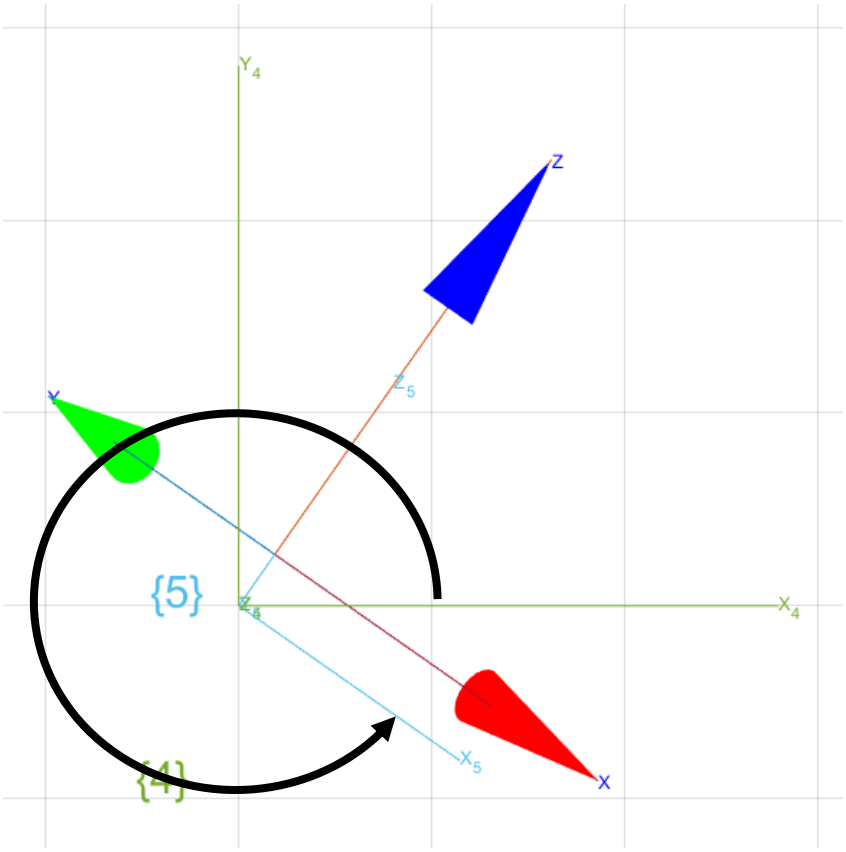

Figura 13: plano para q_5.

Donde se debe notar que se usó la proyección del eje X_6 para el cálculo del arco tangente.

De esta forma se pueden calcular todos los valores de q_4, q_5 y q_6, haciendo uso de arco tangentes y algunos cálculos simples de matrices.

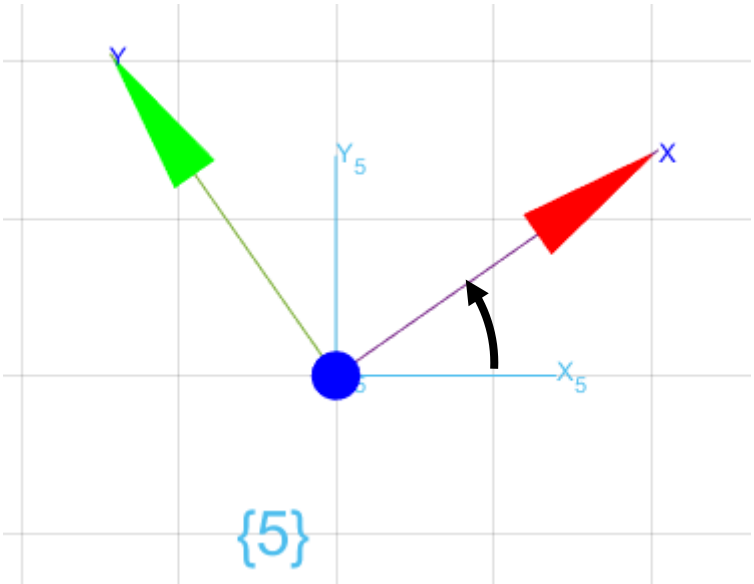

Figura 14: plano para q_6.

Es importante recordar el ordenamiento en la matriz de soluciones "`qq`". Si en cada columna había un conjunto q_1, q_2, q_3, entonces cada columna deberá tener un conjunto q_4, q_5, q_6 consistente.

D) Aplicación de Offsets

Para finalizar el cálculo de las 8 soluciones de la Cinemática Inversa, es importante aplicar los offsets del robot. El offset en una articulación es un valor constante que se le suma al valor real de dicha articulación. Por lo tanto, si un robot tiene un offset en alguna articulación q_i, cuando usamos el robot en el valor $q_i = X$, en realidad está en $q_i = X - offset_i$

Por ejemplo, en la Figura 15 se observa el IRB140 con todos sus valores articulares en cero y sin offsets, y en la Figura 16 se observa con el mismo vector articular, pero con un offset de $-pi/2$ en la segunda articulación.

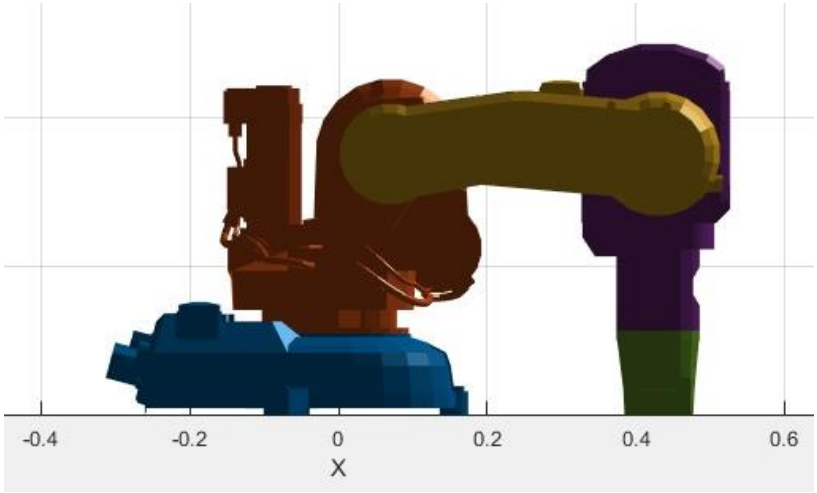

Figura 15: IRB140 con "`q=[0,0,0,0,0,0]`" y "`offset=[0,0,0,0,0,0]`"

Hay que tener cuidado con esto en los cálculos porque, por ejemplo, el método "`R.links(i).A(x)`" de la clase SerialLink sí tiene en cuenta el offset. Esto es adecuado para el trabajo interno del objeto, pero en nuestro desarrollo debemos llevar el robot a su forma base y genérica. Algo similar a lo que se hizo con las matrices de "`base`" y "`tool`", para el offset se puede rescatar el valor y anularlo, al inicio de la función:

```
>> offsets = R.offset;
>> R.offset = zeros(6,1);
```

Pero luego, al final, las soluciones deben tener este offset incluido, sino no serían compatibles con el resto del código:

```
>> R.offset = offsets;
>> qq = qq - R.offset' * ones(1,8);
```

La forma de asignar los offsets en la línea anterior no es única, y el objetivo es restarle a cada articulación el offset que corresponde, dentro de la matriz "`qq`".

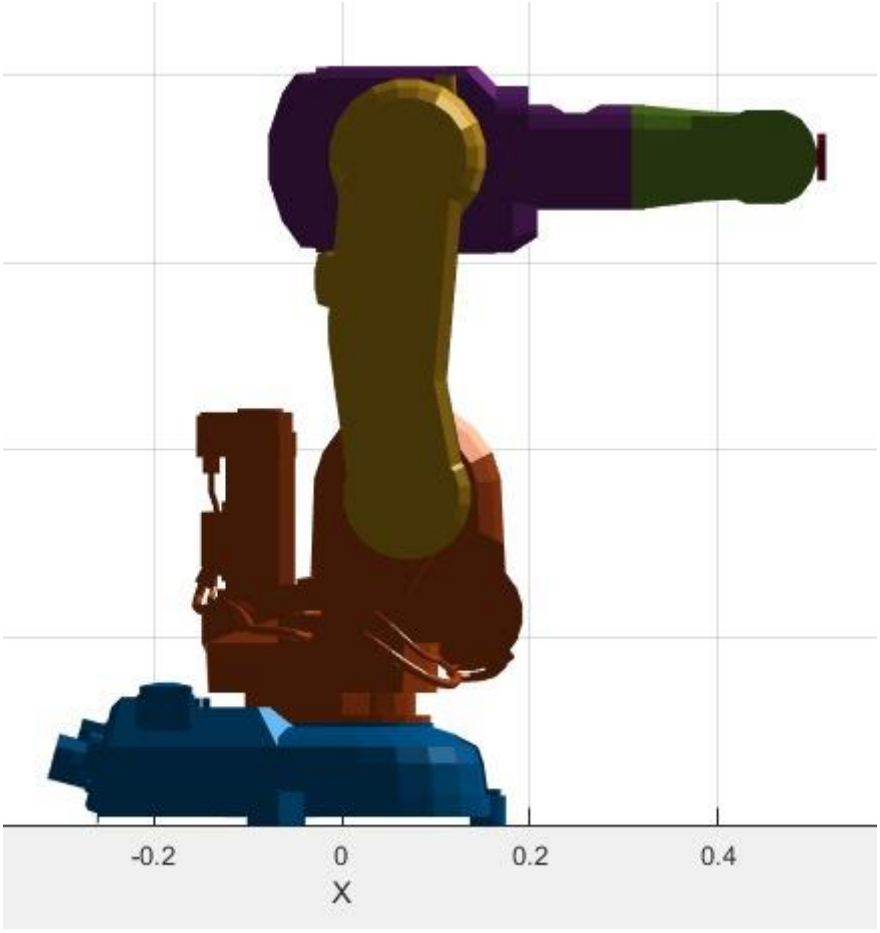

Figura 16: IRB140 con "`q=[0,0,0,0,0,0]`" y "`offset=[0,-pi/2,0,0,0,0]`"

En resumen, el procedimiento descripto consiste en:

1. Anular "`base`" y "`tool`".
2. Calcula el punto de la muñeca.
3. Calcular los 2 valores de q_1.
4. Calcular 2 valores de q_2 para cada q_1.
5. Calcular el valor de q_3 para cada par q_1 y q_2.
6. Determinar las 4 matrices ${}^3T_6$, correspondientes a las ternas q_1, q_2 y q_3.
7. Calcular los 2 valores de q_4 para cada matriz.
8. Calcular el valor de q_5 para cada q_4.
9. Calcular el valor de q_6 para cada q_5.
10. Restar el offset a cada una de los 8 vectores solución.

CONSIDERACIONES IMPORTANTES

Caso degenerado y puntos singulares

Existirán configuraciones singulares en el espacio articular que pueden presentar ciertos problemas en los cálculos. Por ejemplo, la más conocida es la de $q_5 = 0$. Sin importar el valor de las demás variables articulares, cuando $q_5 = 0$ no se puede realizar el cálculo de q_4 porque no existen las componentes XY del extremo sobre el plano $X_3 - Y_3$. Esto se debe a que si la

articulación $q_5 = 0$, entonces los ejes Z_4 y Z_6 están alineados. Uno podría tentarse a buscar otras relaciones para determinar el valor de q_4, pero en realidad el problema está en que si los ejes Z_4 y Z_6 están alineados, ambas articulaciones q_4 y q_6 producen el mismo giro en el efector final. Es decir, se ha perdido un grado de libertad ya que 2 de las articulaciones están contribuyendo exactamente al mismo movimiento y mismo grado de libertad.

Para este caso se propone usar una solución alternativa, conocida como "degenerada" en parte de la bibliografía, que consiste en fijar el valor de q_4 y solo calcular el valor de q_6. En este caso, por una cuestión de utilidad, definiremos q_4 con un valor inicial pasado como parámetro a la función de CI. Dicho valor podría ser el de la posición anterior en una trayectoria, por ejemplo. La determinación de la condición de $q_5 = 0$ se puede realizar así:

```
>> if abs(T36(3,3) - 1) < eps
```

Donde evaluamos que la componente Z del eje Z del extremo, respecto del sistema 3, se exactamente 1. Es decir, estamos viendo si el eje Z del extremo respecto del sistema 3 está completamente alineado. Específicamente le restamos 1, calculamos el valor absoluto y lo comparamos con el "`eps`", que sería la precisión de punto flotante en Matlab. Esta condición se puede evaluar de distintas formas, la presentada tiene algunas ventajas que no es necesario abordar en este momento.

Luego, las soluciones del problema de orientación son:

```
>> q4(1) = q0(4);
>> q5(1) = 0;
>> q6(1) = atan2(T36(2,1), T36(1,1)) - q4(1);
>> q4(2) = q4(1);
>> q5(2) = 0;
>> q6(2) = q6(1);
```

Donde se puede ver $q_5 = 0$ y que el segundo valor de las 3 variables es igual al primero. Se observa que q_4 toma el valor correspondiente del vector inicial, y que el único cálculo se da en q_6, para que se cumpla la orientación del extremo. A dicho valor se le debe restar q_4 ya que es ángulo calculado es desde el eje X_3 hasta el X_6, y q_4 rota en el mismo eje, como se comentó anteriormente.

Verificación total y parcial

Otro punto importante a tener en cuenta es la forma correcta de verificar los cálculos. Sobre todo, en el momento del desarrollo e implementación de la función. Para esto, se recomienda recurrir a la cinemática directa tantas veces como sea necesario.

Por ejemplo, un procedimiento básico sería tomar un vector q cualquiera, calcula su cinemática directa, y pasar el resultado a la función en desarrollo. En las 8 soluciones halladas debería encontrarse el q inicial, por un lado y, por otro lado, las 8 soluciones deberían tener la misma matriz de cinemática directa, igual a la pasada de dato.

En caso de que la función no esté funcionando correctamente (que no aparezca q en las soluciones, o que las 8 soluciones no tengan la misma cinemática directa), o que se decida

proceder desarrollando y verificando parcialmente, lo que se puede hacer es verificar la primera parte del desacoplo cinemático. Es decir, verificar el cálculo de las variables q_1, q_2 y q_3 antes de proceder con el resto. Este cálculo se puede realizar calculando parcialmente la cinemática directa, hasta la 4º articulación, dándole a q_4 cualquier valor, cero, por ejemplo. Este cálculo parcial debería llevar a matrices con cualquier orientación, pero todas con el mismo punto de posición, y tal punto debe coincidir con la muñeca, ya que justamente eso estamos calculando.

Q mejor

Por último, es importante tener en cuenta que muchas veces lo que se requiere es solo una solución del problema. Por ejemplo, si se desea que el robot siga una trayectoria recta en el espacio cartesiano, se debe interpolar dicha recta en varios puntos, y calcula la CI para cada punto, pero para cada punto se requiere una única respuesta, así cada articulación se posiciona, y dicha respuesta debería ser la más cercana al punto anterior, así el recorrido articular es mínimo, y el seguimiento de la recta es suave. Por lo tanto, lo correcto sería, en principio, obtener la solución más cercana al punto anterior. Pueden existir otros criterios más adecuados para otras situaciones.

La más cercana se puede hallar de varias formas, una de ellas sería calculando la distancia euclidiana entre vectores y ver cuál tiene la menor respecto del vector inicial pasado a la función, por ejemplo:

```
>> Qaux = qq  - q0' * ones(1,8);
>> normas = zeros(1,8);
>> for i=1:8
>>     normas(i) = norm(Qaux(:,i));
>> end
>> [~,pos] = min(normas);
>> Q = qq(:, pos);
```

Conclusión

En conclusión, con este método se pueden obtener las 8 soluciones de un robot de 6 grados de libertad, tipo serie, con muñeca esférica.

Las soluciones son 8 porque se consideró un círculo trigonométrico de $\pm 180°$ para cada articulación. En la realidad, el robot puede tener límites articulares inferiores o superiores a dicha amplitud, o incluso no centrados en $\pm 180°$. Cualquier caso de estos se resuelve descartando algunas de las soluciones obtenidas, o agregando otras por sucesiones.

Si la muñeca no es esférica no se puede realizar el desacople y el procedimiento difiere, siendo un poco más complejo de visualizar.

El script adjunto tiene la implementación detallada de este texto, con una función como la que sigue:

```
>> function Q = cin_inv_IRB140(R, T, q0, mejor)
```

Donde:

1. `R`: es el objeto SerialLink del robot en cuestión.
2. `T`: es la matriz de dato que se desea alcanzar.
3. `q0`: es el vector de posiciones articulares inicial o previo, con el que se calculará el caso degenerado o se usará para obtener el más cercano.
4. `mejor`: es un booleano para indicar si se desea obtener las 8 soluciones, o la más cercana al `q0`.
5. `Q`: será una matriz 6x8 con las 8 soluciones en columnas, o será de 6x1 si `mejor=true`.

Ejemplo de Caso General – Robot Universal UR10

(Próximamente)