

Laboratorio 2 - Modelo Geométrico Inverso

Andrés Felipe Medina Medina - Juan Santana Salgado - Osmar Toloza Bernal

Tabla de Contenido

3. Modelo Geométrico Inverso.....	1
3.1. Desarrollo del modelo geométrico inverso.....	1
3.1.1. Posición del efecto final.....	2
3.1.2. Orientación del efecto final.....	6
3.2. Multiplicidad de soluciones.....	10
3.3. Simulación y comprobación - Cinemática inversa (RVC).....	11
3.3.1 Comparación de funciones - RVC.....	18
3.3.2 Selección de función - RVC.....	19
3.4. Simulación y comprobación - Cinemática inversa (RST).....	19
3.5. Comparación de métodos.....	29
3.7. Posturas propuestas.....	29
3.8. Verificación de configuraciones calculadas.....	29
3.8.1. Pose 1.....	29
3.8.2. Pose 2:.....	31
3.8.3. Pose 3:.....	33
3.8.4. Pose 4:.....	35

3. Modelo Geométrico Inverso

3.1. Desarrollo del modelo geométrico inverso

Para el desarrollo de la cinemática inversa del robot ABB IRB2400, es pertinente emplear el método de desacople cinemático.

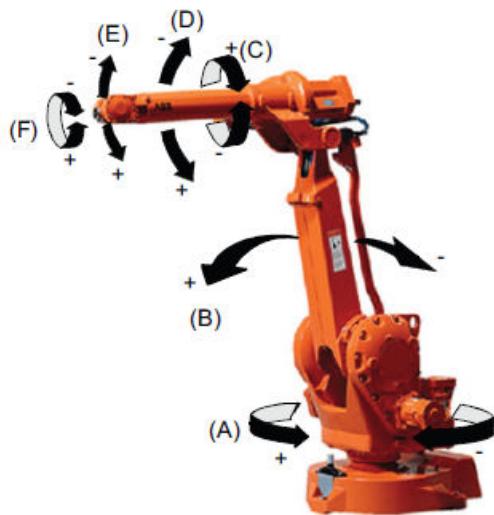


Fig.1. Diagrama de articulaciones - ABB IRB2400.

A partir de la figura 1, se observa que las articulaciones A, B y D se encargan de posicionar al efecto final en un punto en el espacio euclídeo; mientras que C, E y F se encargan de generar una orientación determinada del mismo. Por tanto, para resolver la posición del efecto final, se trabajará con los parámetros q_1, q_2 y

q_3 , correspondientes a las articulaciones A, B y D. Para la resolución de la orientación, se encontrarán los parámetros q_4, q_5 y q_6 , que corresponden a las articulaciones C, E y F.

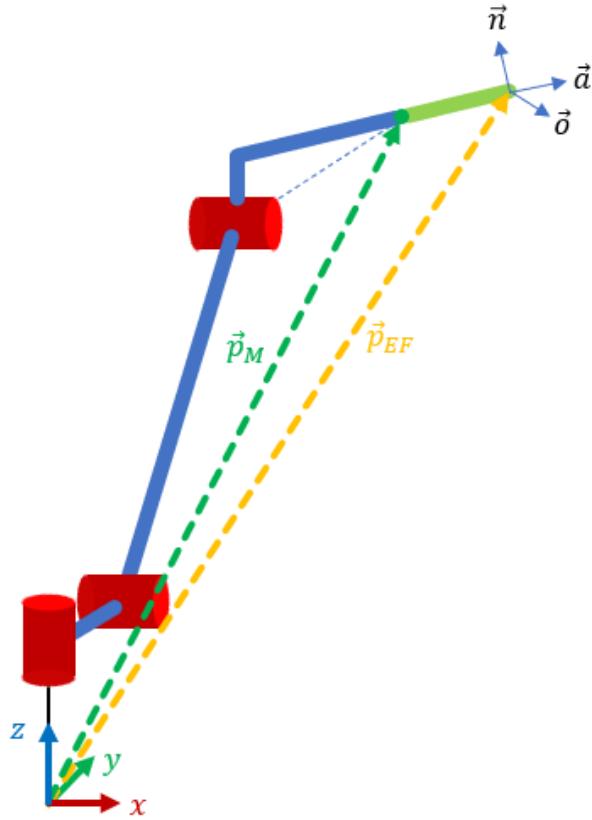


Fig.2. Método de desacople cinemático - ABB IRB2400.

Para el análisis que se presenta, todas las medidas están dadas en milímetros.

3.1.1. Posición del efecto final

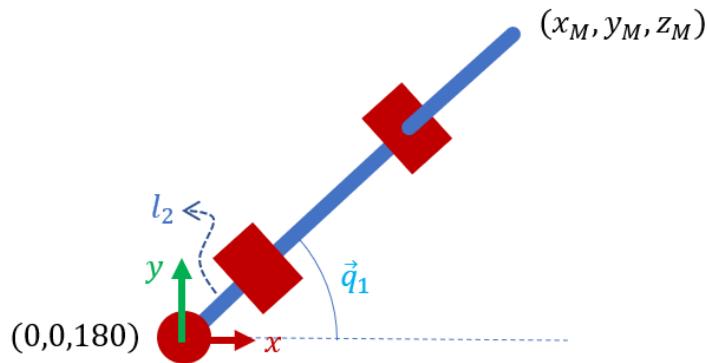
Para esta primera sección, se trabajará con la posición de la muñeca $\vec{p}_M = (x_M, y_M, z_M)$ del manipulador. Ésta se obtiene a partir de la posición $\vec{p}_{EE} = (x, y, z)$ del efecto final mediante la expresión:

$$(x_M, y_M, z_M) = (x, y, z) - L\hat{a}$$

en donde $L = 85\text{ mm}$ y \hat{a} corresponde al vector de aproximación de la trama NOA asignada al efecto final.

Configuración 1:

El análisis se inicia a partir del siguiente esquema:



Desde el plano xy , se encuentra que:

$$\tan(q_1) = \frac{y_M}{x_M}$$

$$q_1 = \text{atan}2(y_M, x_M)$$

A partir de la hoja de especificaciones, se sabe que $l_1 = 615\text{mm}$, $l_2 = 100\text{mm}$, $l_3 = 705\text{mm}$, y que $l_4 = \sqrt{135^2 + 755^2}\text{mm}$. De igual manera, se encuentra que las coordenadas del punto H (Fig.4) están dadas por $(l_2 \cos(q_1), l_2 \sin(q_1), l_1)$.

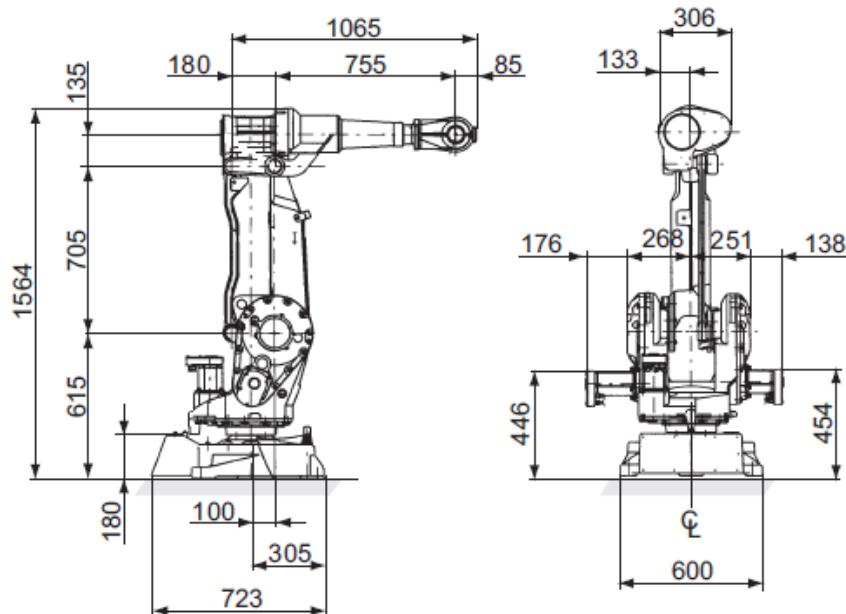


Fig.3. Dimensiones - ABB IRB2400.

Ahora, estudiando el manipulador desde el plano xz :

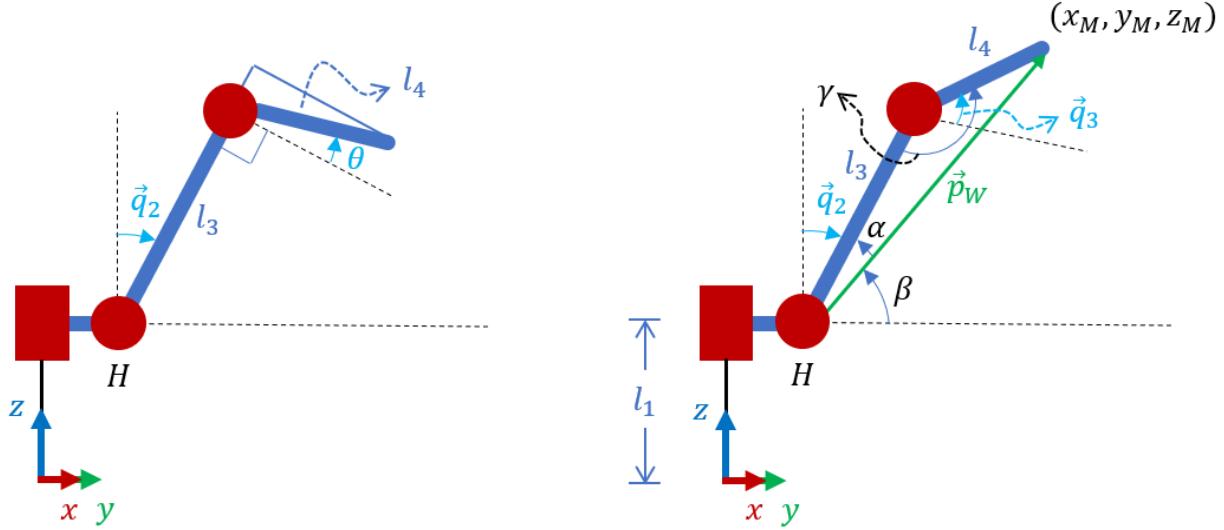


Fig.4. Cinemática inversa por método geométrico - ABB IRB2400.

Para la estimación de q_2 y q_3 se procede a aplicar la ley del coseno sobre el triángulo definido por l_3 , l_4 y p_w .

Se tiene que $p_w = \sqrt{(x_M - l_2 \cos(q_1))^2 + (y_M - l_2 \sin(q_1))^2 + (z_M - l_1)^2}$

Para empezar, se calcula α :

$$l_4^2 = l_3^2 + p_w^2 - 2l_3p_w \cos \alpha$$

$$\cos \alpha = \frac{l_3^2 + p_w^2 - l_4^2}{2l_3p_w}$$

A partir de ello se sabe que $\sin \alpha = \pm \sqrt{1 - (\cos \alpha)^2}$. Por tanto:

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha}$$

$$\alpha = \text{atan2}(\sin \alpha, \cos \alpha)$$

Ahora, β se puede calcular de la siguiente manera:

$$\sin \beta = \frac{z_M - l_1}{p_w}$$

de lo anterior se desprende que $\cos \beta = \pm \sqrt{1 - (\sin \beta)^2}$. Entonces:

$$\tan \beta = \frac{\sin \beta}{\cos \beta}$$

$$\beta = \text{atan2}(\sin \beta, \cos \beta)$$

De la figura 4 se observa que $q_2 = \alpha + \beta$. Según esto, $q_2 = \text{atan2}(\sin \alpha, \cos \alpha) + \text{atan2}(\sin \beta, \cos \beta)$.

Para la estimación de q_3 se aplica nuevamente la ley del coseno, esta vez para calcular γ :

$$p_W^2 = l_3^2 + l_4^2 - 2l_3l_4 \cos \gamma$$

$$\cos \gamma = \frac{l_3^2 + l_4^2 - p_W^2}{2l_3l_4}$$

Se entiende que $\sin \gamma = \pm \sqrt{1 - (\cos \gamma)^2}$. Según esto:

$$\tan \gamma = \frac{\sin \gamma}{\cos \gamma}$$

$$\gamma = \text{atan}(\sin \gamma, \cos \gamma)$$

Ahora bien, la ilustración a la izquierda de la figura 4 representa la posición *Home* del manipulador. El valor de θ se calcula a partir de las dimensiones provistas en la figura 3:

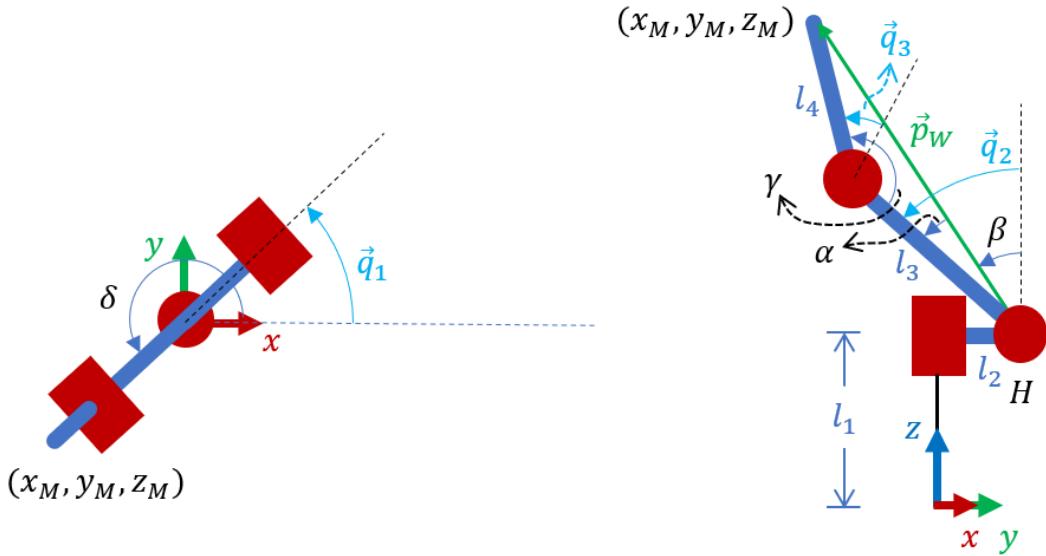
$$\theta = \text{atan}\left(\frac{135}{755}\right)$$

Considerando lo anterior, se concluye que $\gamma = 90^\circ + \theta + q_3$, Por lo que $q_3 = \gamma - 90^\circ - \theta$:

$$q_3 = \text{atan}(\sin \gamma, \cos \gamma) - 90^\circ - \text{atan}\left(\frac{135}{755}\right)$$

Configuración 2:

La segunda configuración se ilustra en la siguiente figura:



Para este caso, la expresión encontrada para q_1 ya no es válida. En su lugar, se observa que:

$$\delta = \text{atan}2(y_M, x_M)$$

Así mismo, se tiene que $q_1 = \delta - 180^\circ$. Entonces:

$$q_1 = \text{atan}2(y_M, x_M) - 180^\circ$$

Respecto a q_2 , se calculan primero las coordenadas del punto H , que están dadas por $(l_2 \cos(q_1), l_2 \sin(q_1), l_1)$. A partir de esto, se calcula p_W :

$$p_W = \sqrt{(x_M - l_2 \cos(q_1))^2 + (y_M - l_2 \sin(q_1))^2 + (z_M - l_1)^2}$$

A partir de la figura, se observa que:

$$\cos \beta = \frac{z_M - l_1}{p_W}$$

$$\sin \beta = \pm \sqrt{1 - (\cos \beta)^2}.$$

$$\tan \beta = \frac{\sin \beta}{\cos \beta}$$

$$\beta = \text{atan}2(\sin \beta, \cos \beta)$$

Con relación a α , se obtiene a partir de la ley del coseno. Con un procedimiento análogo al realizado en la configuración 1, se obtiene:

$$\cos \alpha = \frac{l_3^2 + p_W^2 - l_4^2}{2l_3 p_W}$$

$$\sin \alpha = \pm \sqrt{1 - (\cos \alpha)^2}$$

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha}$$

$$\alpha = \text{atan}2(\sin \alpha, \cos \alpha)$$

Según lo anterior, se encuentra que $q_2 = \alpha + \beta$.

Ahora, para estimar q_3 se procede a calcular γ a través de la ley del coseno, obteniéndose:

$$\cos \gamma = \frac{l_3^2 + l_4^2 - p_W^2}{2l_3 l_4}$$

$$\sin \gamma = \pm \sqrt{1 - (\cos \gamma)^2}$$

Por ende, $\gamma = \text{atan}2(\sin \gamma, \cos \gamma)$. Ahora, al igual que se definió en la configuración 1, θ es constante y viene dada por $\theta = \text{atan}\left(\frac{135}{755}\right)$.

Así pues, se tiene que $q_3 = \gamma - (90^\circ + \theta)$.

3.1.2. Orientación del efecto final

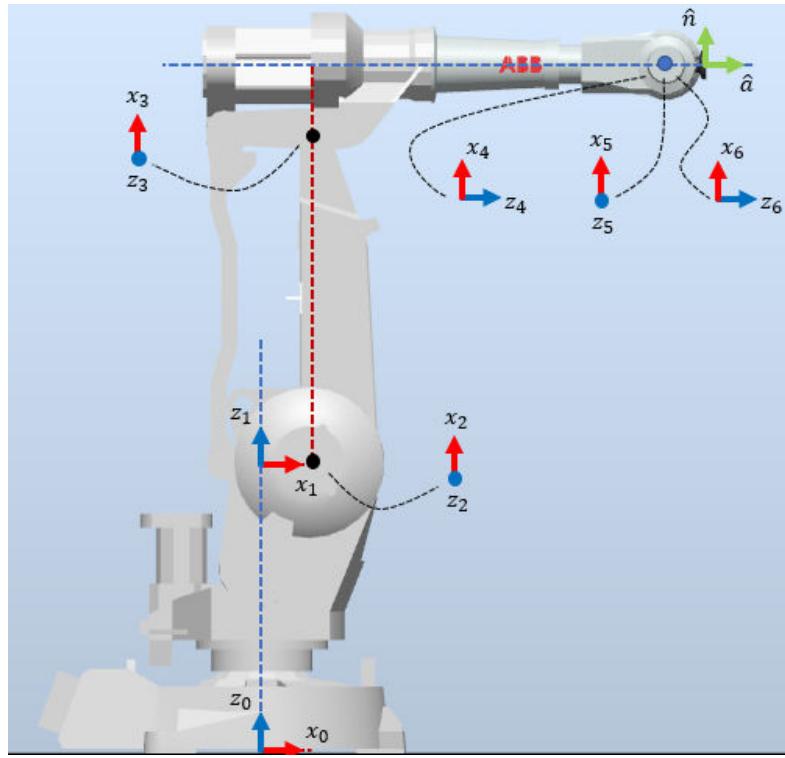


Fig.5. Asignación de tramas - ABB IRB2400.

Para resolver la orientación del efecto final, se conoce la matriz de rotación ${}^0R_{ef} = [\hat{n} \ \hat{o} \ \hat{d}]$ que indica la orientación del mismo respecto a la trama base. Ahora bien, a partir de la figura 5, se observa que las tramas de la articulación 6 y la del efecto final exhiben la misma orientación respecto a la trama base, por lo que no se requiere de una matriz de rotación adicional que describa la relación entre la trama {6} y la trama del efecto final; es decir, ${}^0R_6 = {}^0R_{ef}$.

Habiéndose calculado los valores requeridos para q_1, q_2 y q_3 , se procede a resolver para q_4, q_5 y q_6 de la siguiente forma:

$${}^0R_6 = {}^0R_3 {}^3R_6 = {}^0R_1 {}^1R_2 {}^2R_3 {}^3R_4 {}^4R_5 {}^5R_6$$

Dado que se conocen los valores de q_1, q_2 y q_3 , es posible encontrar ${}^0R_3 = {}^0R_1 {}^1R_2 {}^2R_3$. Posteriormente:

$$({}^0R_3)^{-1} {}^0R_6 = {}^3R_4 {}^4R_5 {}^5R_6$$

Sea $R = ({}^0R_3)^{-1} {}^0R_6$ una matriz de rotación cuyas entradas son conocidas. Entonces:

$$R = {}^3R_4 {}^4R_5 {}^5R_6$$

Ahora, a partir de la asignación de tramas propuesta en la figura 5, se calcula la tabla de parámetros DH para el manipulador en cuestión:

i	α_{i-1}	a_{i-1}	d_i	θ_i	offset
1	0	0	615	q_1	0
2	$\pi/2$	305	0	q_2	$\pi/2$
3	0	705	0	q_3	0
4	$\pi/2$	135	755	q_4	0
5	$-\pi/2$	0	0	q_5	0
6	$\pi/2$	0	0	q_6	0

Con el fin de obtener la estructura de las matrices de rotación correspondientes a ${}^3R_4, {}^4R_5$ y 5R_6 , se crea el robot y se recurre a la función A disponible en el Toolbox de Peter Corke:

```
%Parámetros del manipulador - dm
l1=6.15;
l2=1.00;
l3=7.05;
l4=sqrt(1.35^2+7.55^2);
l5=0.85;

%Creación del robot
L(1)=Link([0 l1 0 0],'modified');
L(2)=Link([0 0 l2 pi/2],'modified');
L(2).offset=pi/2;
L(3)=Link([0 0 l3 0],'modified');
L(4)=Link([0 7.55 1.35 pi/2],'modified');
L(5)=Link([0 0 0 -pi/2],'modified');
L(6)=Link([0 0 0 pi/2],'modified');

irb2400=SerialLink(L,'name','ABB IRB2400')
```

```
irb2400 =
ABB IRB2400 (6 axis, RRRRRR, modDH, fastRNE)

+-----+
| j | theta | d | a | alpha | offset |
+-----+
| 1| q1 | 6.15 | 0 | 0 | 0 |
| 2| q2 | 0 | 1 | 1.571 | 1.571 |
| 3| q3 | 0 | 7.05 | 0 | 0 |
| 4| q4 | 7.55 | 1.35 | 1.571 | 0 |
| 5| q5 | 0 | 0 | -1.571 | 0 |
| 6| q6 | 0 | 0 | 1.571 | 0 |
+-----+
grav = 0 base = 1 0 0 0 tool = 1 0 0 0
      0 0 1 0 0 1 0 0
      9.81 0 0 1 0 0 0 1 0
                  0 0 0 1
```

```
syms q4 q5 q6
```

```
th=1e-10;
T34=mapSymType(L(4).A(q4), 'rational', @(x) piecewise(abs(x)<=th,0,x))
```

T34 =

$$\begin{pmatrix} \cos(q_4) & -\sin(q_4) & 0 & \frac{27}{20} \\ 0 & 0 & -1 & -\frac{151}{20} \\ \sin(q_4) & \cos(q_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T45=mapSymType(L(5).A(q5), 'rational', @(x) piecewise(abs(x)<=th,0,x))
```

T45 =

$$\begin{pmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin(q_5) & -\cos(q_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
T56=mapSymType(L(6).A(q6), 'rational', @(x) piecewise(abs(x)<=th,0,x))
```

T56 =

$$\begin{pmatrix} \cos(q_6) & -\sin(q_6) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Según estos resultados, se tiene que:

$$R = \begin{bmatrix} cq_4 & -sq_4 & 0 \\ 0 & 0 & -1 \\ sq_4 & cq_4 & 0 \end{bmatrix} \begin{bmatrix} cq_5 & -sq_5 & 0 \\ 0 & 0 & 1 \\ -sq_5 & -cq_5 & 0 \end{bmatrix} \begin{bmatrix} cq_6 & -sq_6 & 0 \\ 0 & 0 & -1 \\ sq_6 & cq_6 & 0 \end{bmatrix}$$

$$\begin{bmatrix} cq_4 & -sq_4 & 0 \\ 0 & 0 & -1 \\ sq_4 & cq_4 & 0 \end{bmatrix}^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} cq_5 & -sq_5 & 0 \\ 0 & 0 & 1 \\ -sq_5 & -cq_5 & 0 \end{bmatrix} \begin{bmatrix} cq_6 & -sq_6 & 0 \\ 0 & 0 & -1 \\ sq_6 & cq_6 & 0 \end{bmatrix}$$

$$\begin{bmatrix} cq_4 & 0 & sq_4 \\ -sq_4 & 0 & cq_4 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} cq_5 & -sq_5 & 0 \\ 0 & 0 & 1 \\ -sq_5 & -cq_5 & 0 \end{bmatrix} \begin{bmatrix} cq_6 & -sq_6 & 0 \\ 0 & 0 & -1 \\ sq_6 & cq_6 & 0 \end{bmatrix}$$

$$\begin{bmatrix} cq_4r_{11} + sq_4r_{31} & cq_4r_{12} + sq_4r_{32} & cq_4r_{13} + sq_4r_{33} \\ -sq_4r_{11} + cq_4r_{31} & -sq_4r_{12} + cq_4r_{32} & -sq_4r_{13} + cq_4r_{33} \\ -r_{21} & -r_{22} & -r_{23} \end{bmatrix} = \begin{bmatrix} cq_5cq_6 & -cq_5sq_6 & sq_5 \\ sq_6 & cq_6 & 0 \\ -sq_5cq_6 & sq_5sq_6 & cq_5 \end{bmatrix}$$

Igualando las entradas 2,3:

$$-\text{sq}_4 r_{13} + \text{cq}_4 r_{33} = 0$$

$$\text{sq}_4 r_{13} = \text{cq}_4 r_{33}$$

$$\tan(q_4) = \frac{r_{33}}{r_{13}}$$

$$q_4 = \text{atan2}(r_{33}, r_{13})$$

Ahora, igualando las entradas 1,3 y 3,3:

$$\text{cq}_4 r_{13} + \text{sq}_4 r_{33} = \text{sq}_5$$

$$-r_{23} = \text{cq}_5$$

$$\tan(q_5) = \frac{\text{cq}_4 r_{13} + \text{sq}_4 r_{33}}{-r_{23}}$$

$$q_5 = \text{atan2}(\text{cq}_4 r_{13} + \text{sq}_4 r_{33}, -r_{23})$$

Finalmente, igualando las entradas 2,1 y 2,2:

$$-\text{sq}_4 r_{11} + \text{cq}_4 r_{31} = \text{sq}_6$$

$$-\text{sq}_4 r_{12} + \text{cq}_4 r_{32} = \text{cq}_6$$

$$\tan(q_6) = \frac{-\text{sq}_4 r_{11} + \text{cq}_4 r_{31}}{-\text{sq}_4 r_{12} + \text{cq}_4 r_{32}}$$

$$q_6 = \text{atan2}(-\text{sq}_4 r_{11} + \text{cq}_4 r_{31}, -\text{sq}_4 r_{12} + \text{cq}_4 r_{32})$$

3.2. Multiplicidad de soluciones

Al observar la tabla de parámetros DH, se encuentra que solamente $a_5 = 0$. Así mismo, $a_3 \neq 0$, lo cual nos indica que el criterio para estimar una cota máxima para el número de soluciones del problema cinemático inverso propuesto en el libro Robótica (Craig, John J.) no es la mejor aproximación para el problema en cuestión.

Para estimar la multiplicidad de soluciones, se analizarán las soluciones posibles una a una. Dando inicio al análisis con q_1 , se observa que su cálculo arroja una única solución, puesto que depende de la posición en el espacio de la muñeca en el plano xy , la cual está dada por dos valores únicos x_M y y_M .

En el caso de q_2 , se sabe que $q_2 = \alpha + \beta$. Dado que $\sin \alpha$ y $\cos \beta$ están definidos como raíces positivas o negativas, se entiende que, para cada ángulo, hay un total de 2 soluciones.

Puesto que q_2 depende de ambos ángulos, se concluye que puede tener un total de 4 posibles soluciones.

Respecto a q_3 , se encontró que su cálculo depende de los ángulos γ y θ (éste último, con valor constante). Dado que $\sin \gamma$ también está definido como una raíz positiva o negativa, este ángulo presenta dos posibles soluciones. Por ende, q_3 exhibirá también dos soluciones posibles.

De acuerdo con lo anterior, se contaría con un total de $4 \times 2 = 8$ posibles soluciones. No obstante, se deben considerar los límites articulares. En particular, las limitaciones que presenta la articulación 3 (eje D) impiden que, por ejemplo, los ángulos α y β sean negativos al mismo tiempo, puesto que generarían una configuración

codo abajo frente a la cual la articulación 3 no posee un rango de barrido lo suficientemente amplio como para alcanzar el punto deseado en el espacio. Una excepción del mismo tipo ocurrirá para ángulos α y β excesivamente positivos (se violarían los límites articulares de q_2 y, dependiendo del caso, también de q_3).

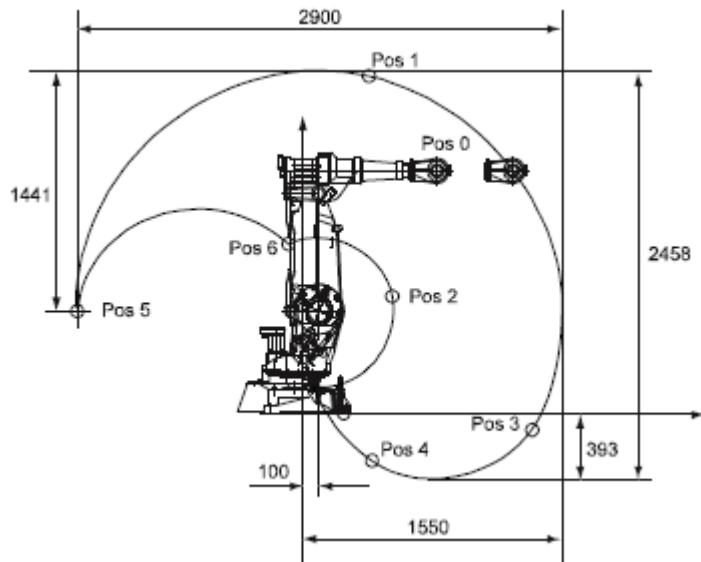


Fig.6. Espacio de trabajo - ABB IRB2400.

Nótese que, para ciertos puntos del espacio, las configuraciones codo arriba y codo abajo serán posibles, siempre que se respeten los límites articulares para las articulaciones 2 y 3.

3.3. Simulación y comprobación - Cinemática inversa (RVC)

%CINEMÁTICA DIRECTA

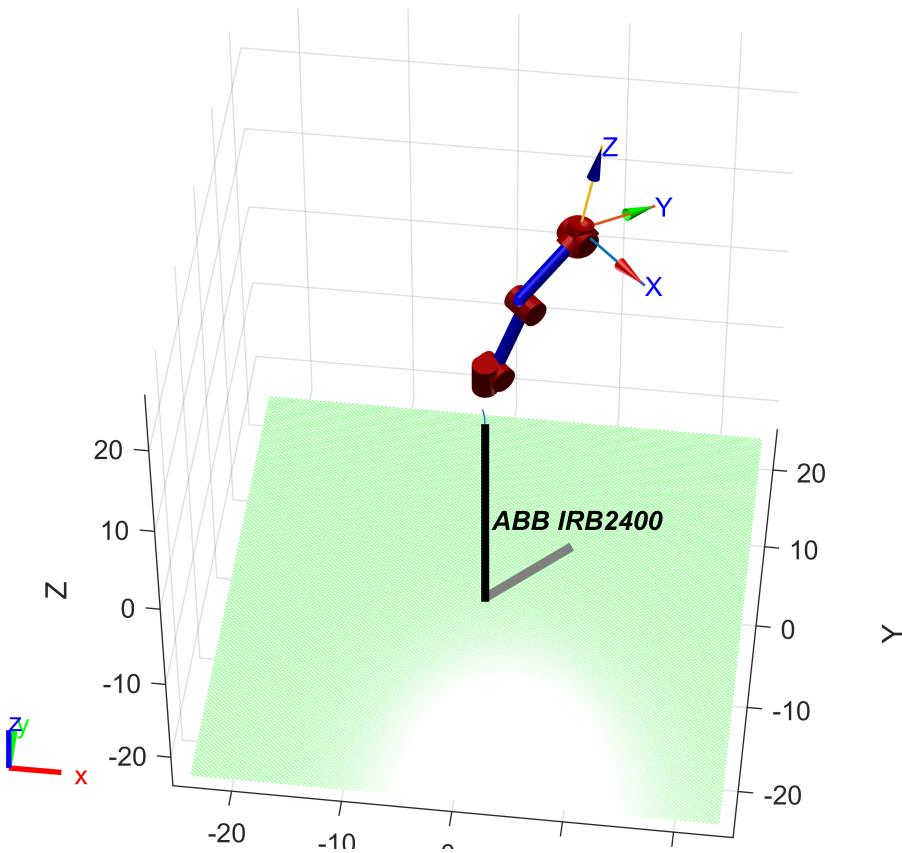
```
Rtool=eye(3);
ptool=[0 0 0.85]';
irb2400.tool=[Rtool ptool;0 0 0 1]
```

```
irb2400 =
ABB IRB2400 (6 axis, RRRRRR, modDH, fastRNE)

+-----+-----+-----+-----+
| j | theta | d | a | alpha | offset |
+---+-----+---+---+-----+-----+
| 1| q1 | 6.15 | 0 | 0 | 0 |
| 2| q2 | 0 | 1 | 1.571 | 1.571 |
| 3| q3 | 0 | 7.05 | 0 | 0 |
| 4| q4 | 7.55 | 1.35 | 1.571 | 0 |
| 5| q5 | 0 | 0 | -1.571 | 0 |
| 6| q6 | 0 | 0 | 1.571 | 0 |
+-----+-----+-----+-----+
grav =   0   base = 1 0 0 0   tool = 1      0      0      0
          0       0 1 0 0           0       1      0      0
         9.81     0 0 1 0           0       0      1      0.85
                  0 0 0 1           0       0      0      1
```

Configuración 1:

```
%Ejemplar - Configuración 1  
T0ef=irb2400.fkine([pi/4 -pi/6 pi/3 0 pi/4 pi/2]);  
  
irb2400.plot([pi/4 -pi/6 pi/3 0 pi/4 pi/2])  
axis('auto')  
view([6.8 46.2])  
hold off
```



```
%CINEMÁTICA INVERSA - Configuración 1  
[Ref,pef]=tr2rt(T0ef);
```

```
%Posición muñeca  
pm=pef-15*Ref(1:3,3)
```

```
pm = 3x1  
7.3458  
7.3458  
17.1996
```

```
%q1  
q1=atan2(pm(2),pm(1));  
q1d=rad2deg(q1)
```

```
q1d = 45
```

```

pw=sqrt((pm(1)-l2*cos(q1))^2+(pm(2)-l2*sin(q1))^2+(pm(3)-l1)^2);

%alpha
ca=(l3^2+pw^2-l4^2)/(2*l3*pw);
sa1=sqrt(1-ca^2);
sa2=-sa1;
a=atan2(sa1,ca);
% a=atan2(sa2,ca);

%beta
sb=(pm(3)-l1)/pw;
cb1=sqrt(1-sb^2);
cb2=-cb1;
b=atan2(sb,cb1);
% b=atan2(sb,cb2);

%q2
q2=-pi/2+a+b;
q2d=rad2deg(q2)

```

q2d = -30.0000

```

%gamma
cg=(l3^2+l4^2-pw^2)/(2*l3*l4);
sg1=sqrt(1-cg^2);
sg2=-sg1;
g=atan2(sg1,cg);
%g=atan2(sg2,cg);

```

```

%theta
theta=atan(1.35/7.55);

```

```

%q3
q3=g-pi/2-theta;
q3d=rad2deg(q3)

```

q3d = 60.0000

```

%Orientación del efecto final
R06=Ref;
[R03,~]=tr2rt(irb2400.A([1 2 3],[q1 q2 q3]));
R36=R03\R06;

```

```

%q4
q4=atan2(R36(3,3),R36(1,3));
q4d=rad2deg(q4)

```

q4d = -3.6726e-15

```

%q5
q5=atan2(cos(q4)*R36(1,3)+sin(q4)*R36(3,3),-R36(2,3));
q5d=rad2deg(q5)

```

q5d = 45.0000

```
%q6
```

```
q6=atan2(-sin(q4)*R36(1,1)+cos(q4)*R36(3,1), -sin(q4)*R36(1,2)+cos(q4)*R36(3,2));  
q6d=rad2deg(q6)
```

```
q6d = 90.0000
```

```
Tci=irb2400.fkine([q1 q2 q3 q4 q5 q6])
```

```
Tci = 4x4
```

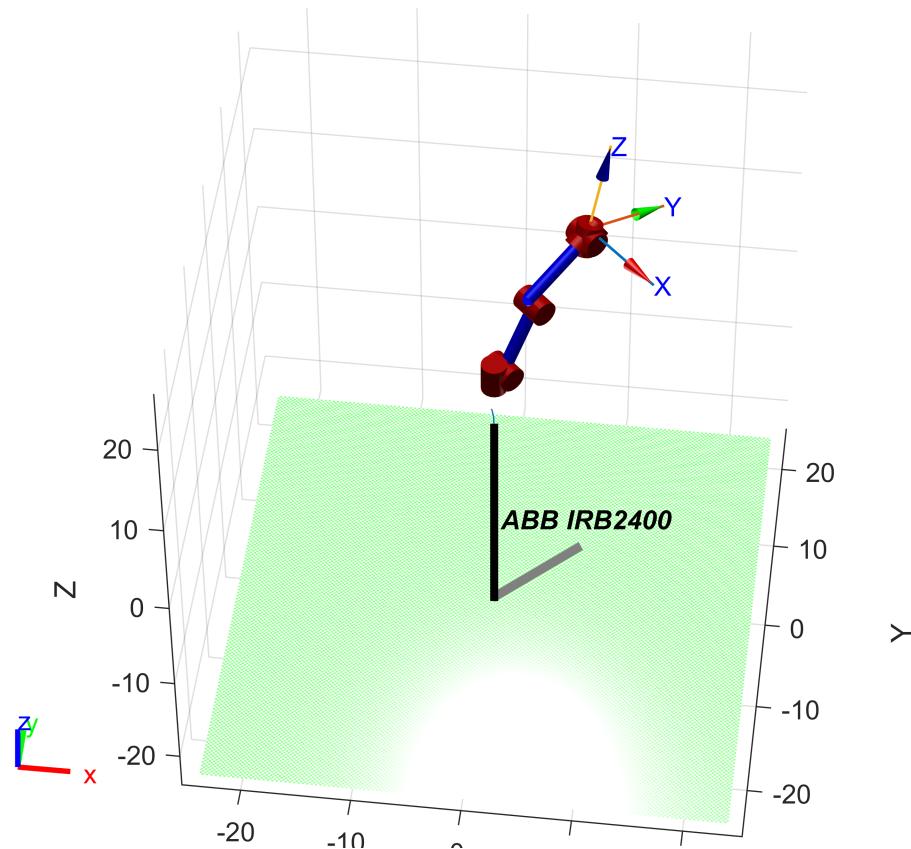
0.7071	0.6830	0.1830	7.5013
-0.7071	0.6830	0.1830	7.5013
0.0000	-0.2588	0.9659	18.0207
0	0	0	1.0000

```
irb2400.plot([q1 q2 q3 q4 q5 q6])
```

```
axis('auto')
```

```
view([6.8 46.2])
```

```
hold off
```



Comparando la MTH obtenida asignando las coordenadas generalizadas manualmente y aquella obtenida al utilizar los resultados de cinemática inversa:

```
T0ef % Manual
```

```
T0ef = 4x4
```

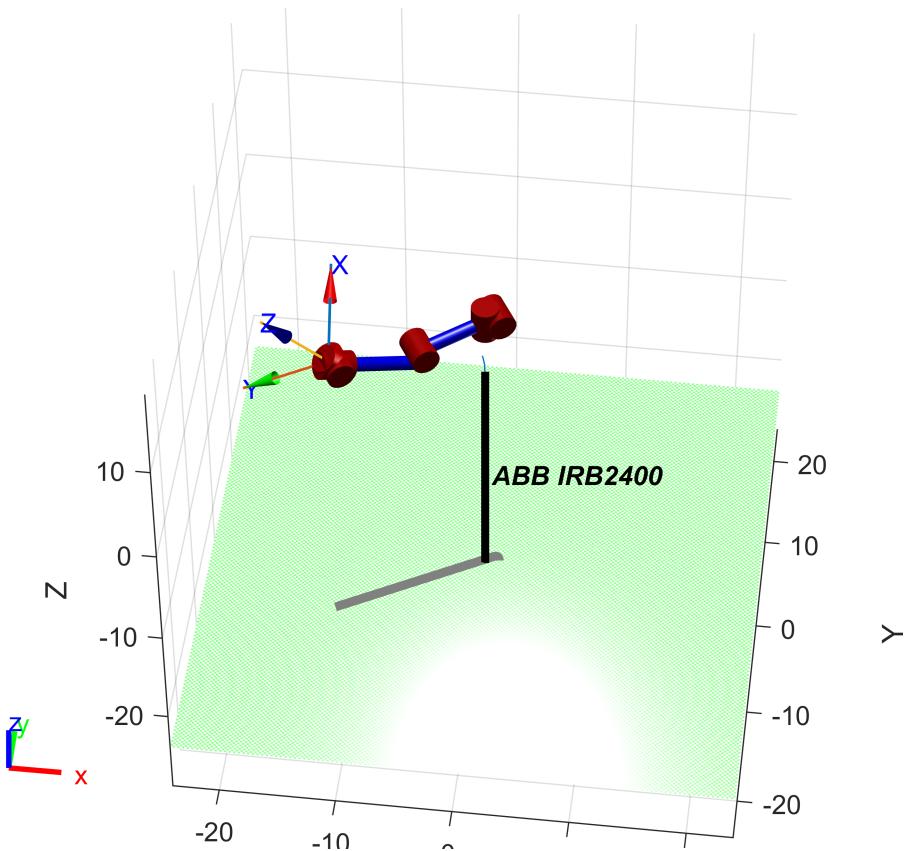
0.7071	0.6830	0.1830	7.5013
-0.7071	0.6830	0.1830	7.5013
0.0000	-0.2588	0.9659	18.0207
0	0	0	1.0000

Tci % Cinemática inversa

```
Tci = 4x4
0.7071    0.6830    0.1830    7.5013
-0.7071    0.6830    0.1830    7.5013
0.0000   -0.2588    0.9659   18.0207
0           0           0       1.0000
```

%Ejemplar - Configuración 2

```
T0ef=irb2400.fkine([pi/6 deg2rad(100) pi/3 pi pi/6 pi/3]);
irb2400.plot([pi/6 deg2rad(100) pi/3 pi pi/6 pi/3])
axis('auto')
view([6.8 46.2])
hold off
```



Configuración 2:

%CINEMÁTICA INVERSA - Configuración 2

```
[Ref,pef]=tr2rt(T0ef);
```

%Posición muñeca

```
pm=pef-15*Ref(1:3,3)
```

```
pm = 3x1
-11.6907
-6.7497
```

```
%delta
d=atan2(pm(2),pm(1));

%q1
q1=d-pi;
q1=q1+2*pi*(q1<-pi); %Evita violación de límites articulares para q1
q1d=rad2deg(q1);

pw=sqrt((pm(1)-l2*cos(q1))^2+(pm(2)-l2*sin(q1))^2+(pm(3)-l1)^2);

%alpha
ca=(l3^2+pw^2-l4^2)/(2*l3*pw);
sa1=sqrt(1-ca^2);
sa2=-sa1;
a=atan2(sa1,ca);
% a=atan2(sa2,ca);

%beta
cb=(pm(3)-l1)/pw;
sb1=sqrt(1-cb^2);
sb2=-sb1;
b=atan2(sb1,cb);
% b=atan2(sb2,cb);

%q2
q2=a+b;
q2d=rad2deg(q2);

%gamma
cg=(l3^2+l4^2-pw^2)/(2*l3*l4);
sg1=sqrt(1-cg^2);
sg2=-sg1;
g=atan2(sg1,cg);
%g=atan2(sg2,cg);

%theta
theta=atan(1.35/7.55);

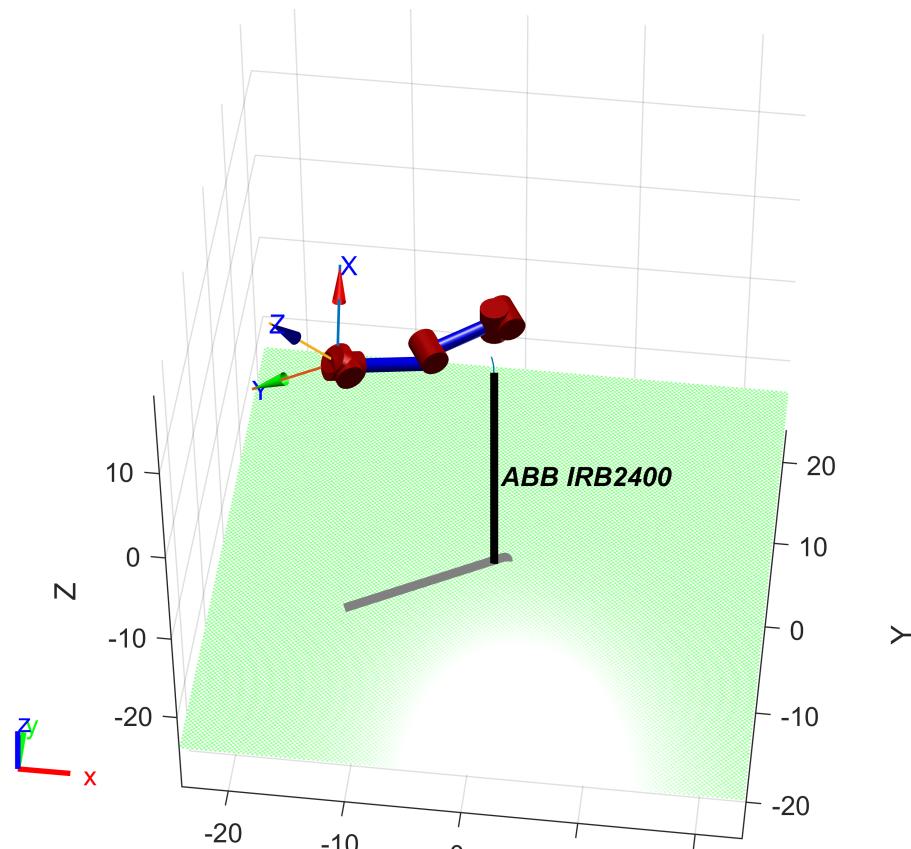
%q3
q3=g-pi/2-theta;
q3d=rad2deg(q3);

%Orientación del efecto final
R06=Ref;
[R03,~]=tr2rt(irb2400.A([1 2 3],[q1 q2 q3]));
R36=R03\R06;

%q4
q4=atan2(R36(3,3),R36(1,3));

%q5
q5=atan2(cos(q4)*R36(1,3)+sin(q4)*R36(3,3),-R36(2,3));
```

```
%q6
q6=atan2(-sin(q4)*R36(1,1)+cos(q4)*R36(3,1),-sin(q4)*R36(1,2)+cos(q4)*R36(3,2));
Tci=irb2400.fkine([q1 q2 q3 q4 q5 q6]);
irb2400.plot([q1 q2 q3 q4 q5 q6])
axis('auto')
view([6.8 46.2])
hold off
```



Comparando las MTH obtenidas asignando las coordenadas generalizadas manualmente y a partir de los resultados de cinemática inversa:

T0ef

T0ef = 4x4
-0.1013 -0.8245 -0.5567 -12.1639
0.9415 0.1013 -0.3214 -7.0228
0.3214 -0.5567 0.7660 6.8906
0 0 0 1.0000

Tci

Tci = 4x4
-0.1013 -0.8245 -0.5567 -12.1639
0.9415 0.1013 -0.3214 -7.0228
0.3214 -0.5567 0.7660 6.8906
0 0 0 1.0000

3.3.1 Comparación de funciones - RVC

- **ikcon:** Requiere para su funcionamiento de la función *fmincon* del Optimization Toolbox. Se puede utilizar para la cinemática inversa de robots con un número arbitrario de grados de libertad. Dado que es una función que ofrece una solución a través de métodos numéricos, el vector solución *q* presentará un error que se obtiene al comparar la MTH obtenida al realizar la cinemática directa utilizando *q* y la MTH que se deseaba obtener inicialmente. Así mismo, acepta como argumento un vector de valores iniciales a partir del cual empezará a converger hacia la solución; por tanto, la solución dependerá del vector de valores iniciales utilizado. Por último, los límites articulares definidos al crear el robot son tenidos en cuenta a la hora de encontrar la solución al problema cinemático inverso.
- **ikine:** Es utilizada con frecuencia para resolver la cinemática inversa de manipuladores de 6 o más grados de libertad. Al igual que ikcon, es una aproximación desde métodos numéricos de carácter iterativo. Ofrece un control personalizado del método numérico a emplear, al permitir seleccionar entre la traspuesta del Jacobiano o el método de pseudoinversa; así como ajustar la tolerancia en cuanto al error respecto al resultado deseado, limitar el número de iteraciones a realizar, el tamaño de paso entre iteraciones o el estado del proceso de convergencia en cada iteración. Una limitación se debe tener en cuenta al emplear esta función para manipuladores con menos de 6 grados de libertad. Dado que posición y orientación son definidas con 6 variables, para manipuladores con menos de 6 GDL, se debe emplear un *vector máscara* en el cual se indican cuáles de estos 6 parámetros se deben ignorar en el cálculo de la solución. Los parámetros a ignorar son asignados a cero, el resto a uno. La solución varía dependiendo del vector de valores iniciales empleado. Una ventaja es que este método permite obtener soluciones incluso en puntos de singularidad, aunque no tiene en cuenta los límites articulares.
- **ikine3:** Permite realizar la cinemática inversa para robots de 3GDL sin muñeca. Sus argumentos son la MTH que relaciona al efecto final con la base y un argumento de configuración opcional que permite seleccionar el tipo de solución a obtener (codo arriba/abajo, brazo derecho/izquierdo). El procedimiento de solución es equivalente al de *ikine6s* sin incluir los parámetros de la muñeca.
- **ikine6s:** Esta función resuelve la cinemática inversa de forma analítica para manipuladores de 6 ejes con muñeca esférica. Al igual que *ikine3*, sus argumentos son una MTH dada y, opcionalmente, una configuración para obtener una de las soluciones disponibles (codo arriba/abajo, brazo derecho/izquierdo, muñeca no rotada/rotada). Así mismo, puede trabajar con manipuladores que tengan offset en su hombro o codo, o cuya tercera articulación sea prismática (tipo Stanford). Una consideración importante es que **sólo es aplicable para manipuladores descritos con la convención DH estándar.**
- **ikine_sym:** Resuelve la cinemática inversa de forma simbólica. Su argumento principal *k* indica si se desea obtener la solución para posición en *x* y *y* (2), en los 3 ejes cartesianos (3), o la solución para posición y orientación (6).
- **ikinem:** Resuelve la cinemática inversa de forma numérica a través de la minimización de una función error calculada a partir de la diferencia entre la pose actual y la pose objetivo. Es recomendable emplearlo para resolver manipuladores complejos, pues por defecto emplear un solver para resolver problemas no lineales (Levenberg-Marquadt), el cual es menos eficiente que las funciones analíticas descritas previamente. En ocasiones, se utiliza un vector de valores iniciales, el cual debe ser revisado con cuidado para evitar que la solución corresponda a un caso en donde el algoritmo quede estancado en un mínimo local. Así mismo, puede tener en cuenta los límites articulares.

- **ikunc**: Requiere de la función *fminunc* del Optimization Toolbox. Su funcionalidad es la misma de *ikcon*, con la salvedad de que **no tiene en cuenta los límites articulares**.

3.3.2 Selección de función - RVC

Para el presente problema se trabaja con un robot de 6GDL para el cual es posible obtener una solución analítica, por lo cual no es necesario recurrir a funciones que empleen métodos numéricos debido a su alta exigencia a nivel computacional. Ahora, dado que el robot ha sido descrito mediante la convención DH modificada, no es posible emplear la función *ikine6s*, que sería la más adecuada para este caso, pues brinda una solución analítica para el robot de 6 GDL, la cual implica un costo computacional notablemente menor.

Según lo anterior, la función más adecuada es *ikine_sym*, la cual nos brinda una solución analítica de forma simbólica, que además reporta las distintas posibles soluciones de manera detallada. Lo anterior permite identificar con claridad todas las soluciones existentes, las cuales se podrían utilizar mediante una función que permita evaluar expresiones simbólicas con valores reales; por ejemplo, la función *subs*.

3.4. Simulación y comprobación - Cinemática inversa (RST)

Modelo del robot ABB IRB 2400 utilizando Robotics System Toolbox de Matlab.

```
L0=6.15;
L1=1.00;
L2=7.05;
L3=1.35;
L4=7.55;
L5=0.85;

robotabb_rst=rigidBodyTree;

dhparams=[  0      0      0      0   ;
            0.1*L1  pi/2    0      0   ;
            0.1*L2    0      0      0   ;
            0.1*L3  pi/2    0.1*L4    0   ;
            0     -pi/2   0      0   ;
            0     pi/2    0.1*L5    0 ];

link1=rigidBody('link1');
link1.Joint=rigidBodyJoint('art1','revolute');
link2=rigidBody('link2');
link2.Joint=rigidBodyJoint('art2','revolute');
link3=rigidBody('link3');
link3.Joint=rigidBodyJoint('art3','revolute');
link4=rigidBody('link4');
link4.Joint=rigidBodyJoint('art4','revolute');
link5=rigidBody('link5');
link5.Joint=rigidBodyJoint('art5','revolute');
link6=rigidBody('link6');
link6.Joint=rigidBodyJoint('art6','revolute');

link1.Joint.PositionLimits=[deg2rad(-180) deg2rad(180)];
link2.Joint.PositionLimits=[deg2rad(-100) deg2rad(110)];
```

```

link3.Joint.PositionLimits=[deg2rad(-60) deg2rad(65)];
link4.Joint.PositionLimits=[deg2rad(-200) deg2rad(200)];
link5.Joint.PositionLimits=[deg2rad(-120) deg2rad(120)];
link6.Joint.PositionLimits=[deg2rad(-400) deg2rad(400)];

setFixedTransform(link1.Joint,dhparams(1,:),'mdh');
setFixedTransform(link2.Joint,dhparams(2,:),'mdh');
setFixedTransform(link3.Joint,dhparams(3,:),'mdh');
setFixedTransform(link4.Joint,dhparams(4,:),'mdh');
setFixedTransform(link5.Joint,dhparams(5,:),'mdh');
setFixedTransform(link6.Joint,dhparams(6,:),'mdh');

addBody(robotabb_RST,link1,'base')
addBody(robotabb_RST,link2,'link1')
addBody(robotabb_RST,link3,'link2')
addBody(robotabb_RST,link4,'link3')
addBody(robotabb_RST,link5,'link4')
addBody(robotabb_RST,link6,'link5')

figure(1)
showdetails(robotabb_RST)

```

Robot: (6 bodies)

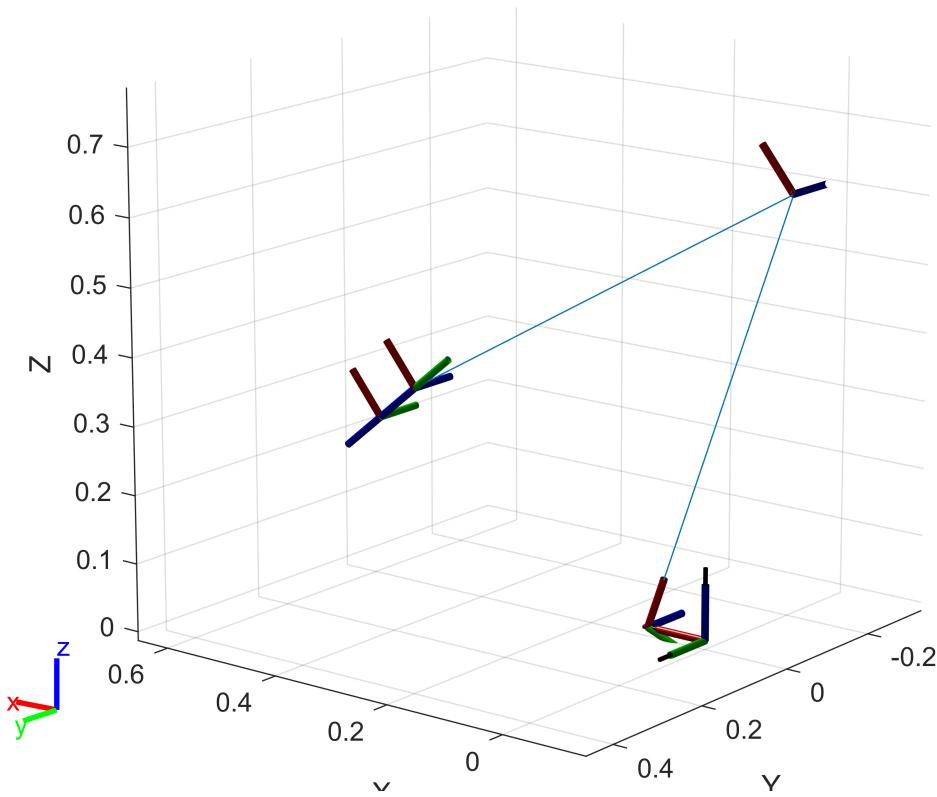
Idx	Body Name	Joint Name	Joint Type	Parent Name(Idx)	Children Name(s)
1	link1	art1	revolute	base(0)	link2(2)
2	link2	art2	revolute	link1(1)	link3(3)
3	link3	art3	revolute	link2(2)	link4(4)
4	link4	art4	revolute	link3(3)	link5(5)
5	link5	art5	revolute	link4(4)	link6(6)
6	link6	art6	revolute	link5(5)	

```

config = homeConfiguration(robotabb_RST);
config(1).JointPosition = 0;
config(2).JointPosition = 110*pi/180;
config(3).JointPosition = -pi/3;
config(4).JointPosition = 0;
config(5).JointPosition = 0;
config(6).JointPosition = 0;

show(robotabb_RST,config);
xlim([-0.140 0.656])
ylim([-0.336 0.459])
zlim([-0.013 0.783])
view([-142.069 14.790])

```



Cinemática inversa

Para la solución de la cinemática inversa se partió primero de la cinemática directa. Se definió un conjunto de ángulos q y luego se encontró cual era la posición y orientación del efecto final. Luego se realizó la cinemática inversa para dicha POSE, sin embargo los valores de q pueden ser diferentes cuando la POSE está dentro del espacio de trabajo diestro. Esto implica que, debido a la multiplicidad de resultados, Matlab encuentre un resultado que dé dicha POSE y que sea diferente de los valores de q del que partió.

Procedimiento

Se encuentra la pose de los 7 puntos de calibración y de 4 puntos adicionales, definidos por nosotros. Luego, por medio de la función *fun_pose*, se encuentra la pose para un conjunto de ángulos q . Estos datos son luego utilizados en la función *cinematicalnversa* para encontrar los valores q que den dicha pose. La función del Toolbox de Robotics es *ik=inverse kinematics* y toma como valores semilla la posición de home del robot. Cuando existen más de una respuesta para una misma pose, se pueden encontrar al variar los valores semilla. Este proceso es iterativo, de prueba y error.

```
q_cal=[0,0,0,0,0,0;
       0,0,-60,0,0,0;
       0,0,65,0,0,0;
       0,110,-60,0,0,0;
       0,110,18.3,0,0,0;
       0,-100,-60,0,0,0;
       0,-100,65,0,0,0];
```

```

q=[q_cal;
 45,-30,-60,0,45,90;
 135,-60,15,30,60,45;
 -45,60,30,60,0,-135;
 120,90,45,135,-60,90];

```

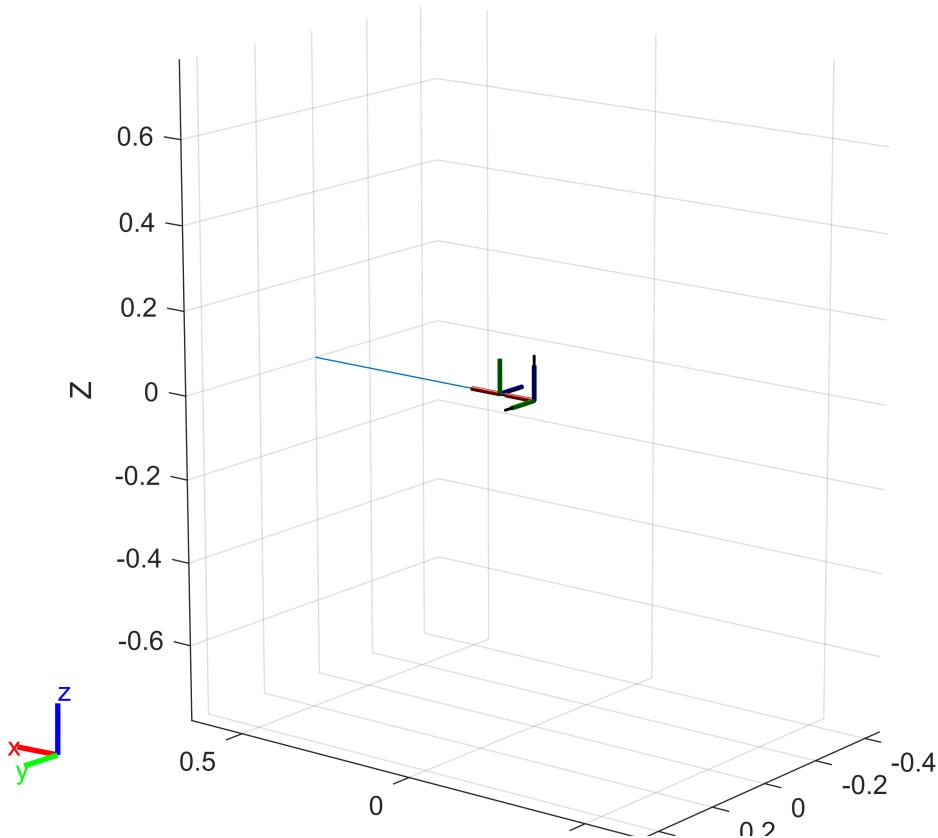
La cinemática inversa para las 11 POSE's son

```

for m=1:11
V_pose_P2=fun_pose(q(m,1:6),robotabb_RST);
disp("POSE #"+m+": ")
disp(V_pose_P2)
[configSoln_P2,solnInfo_P2,q_resul] = cinematicaInversa(V_pose_P2,robotabb_RST);
tform2 = getTransform(robotabb_RST,configSoln_P2,'link6','base'); %Es la MTH del EF en la posic
disp("Angulos q resultantes")
disp(q_resul')
disp(" ")
figure()
show(robotabb_RST,configSoln_P2);
xlim([-0.656 0.656])
ylim([-0.459 0.459])
zlim([-0.783 0.783])
view([-142.069 14.790])
end

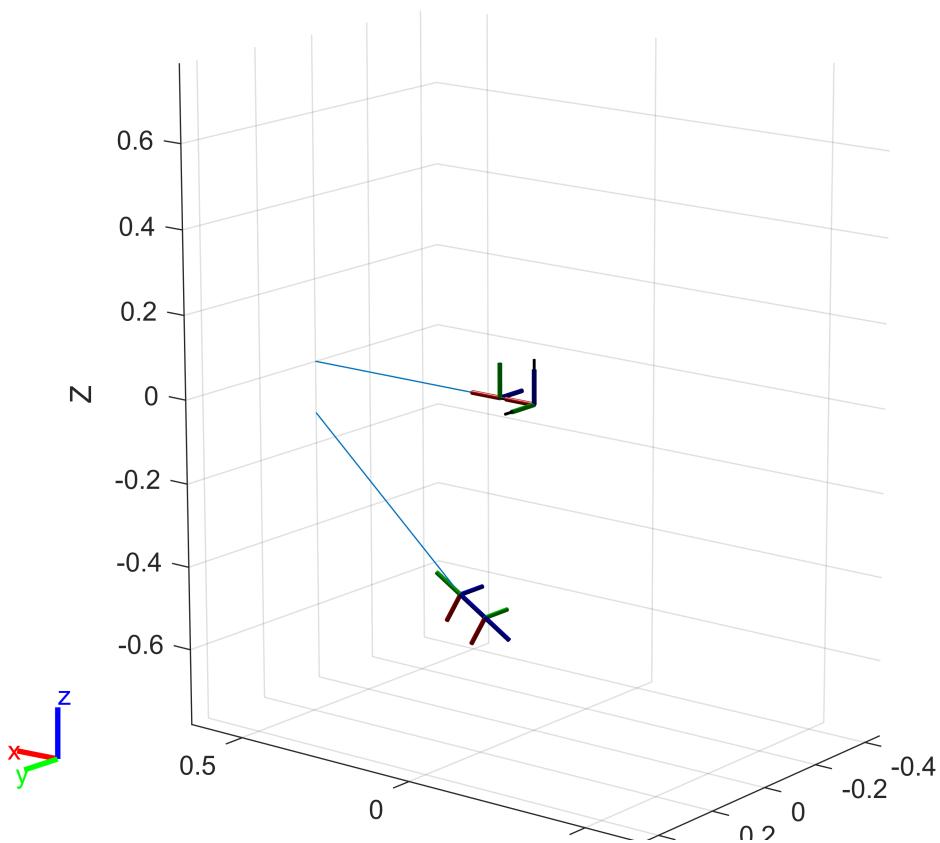
```

POSE #1:
 3.1416 0 0 0.9400 -0.0000 -0.8400
 Angulos q resultantes
 0 0 0 0 0 0



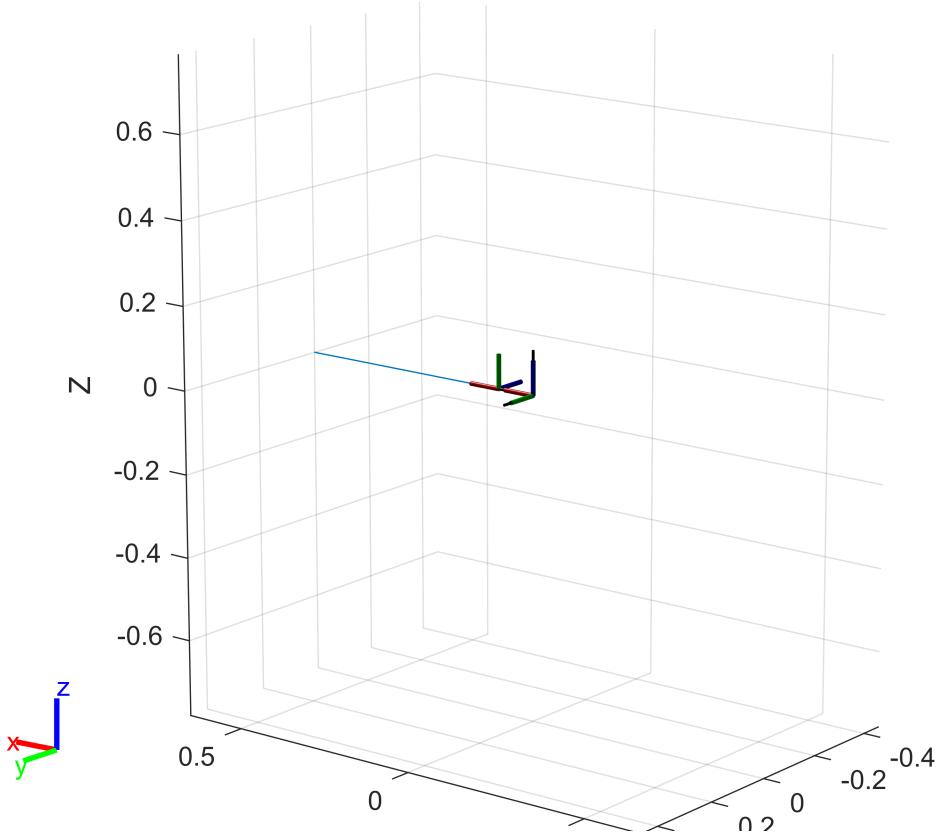
POSE #2:

3.1416	-1.0472	-0.0000	0.1450	-0.0000	-0.5369
Angulos q resultantes					
-0.0000	0.0000	-60.0000	-0.0000	-0.0000	-0.0000

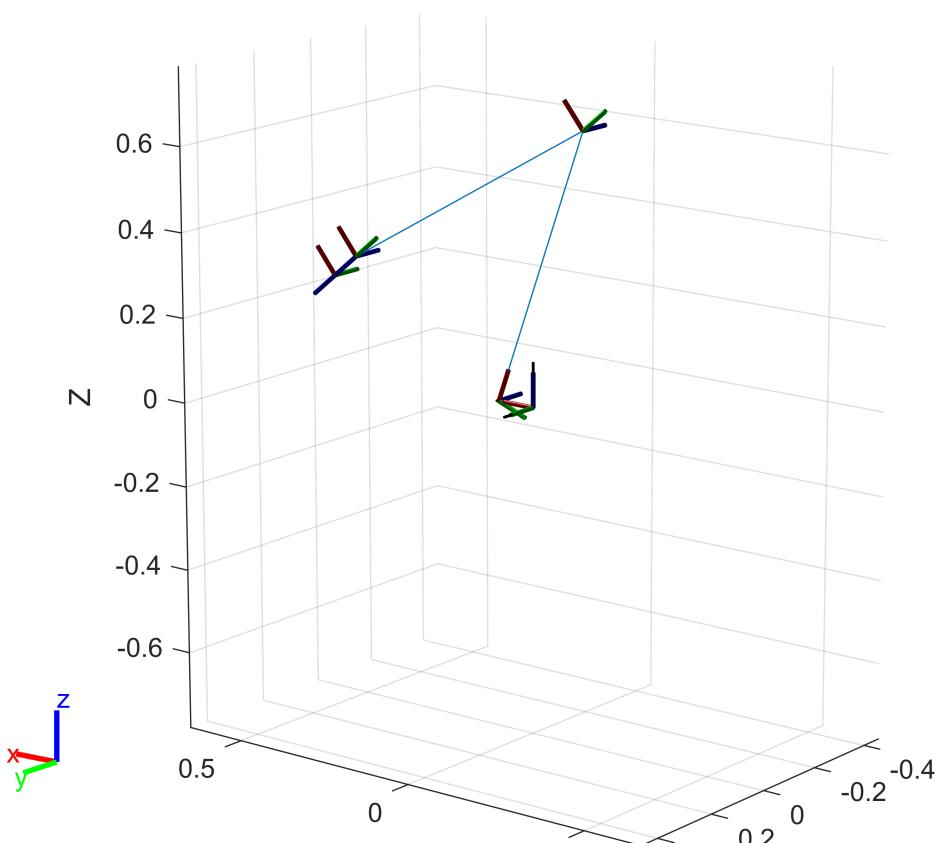


POSE #3:

3.1416	1.1345	0.0000	1.6234	-0.0000	-0.2326
Angulos q resultantes					
-0.0000	-0.0000	65.0000	-0.0000	0.0000	0.0000

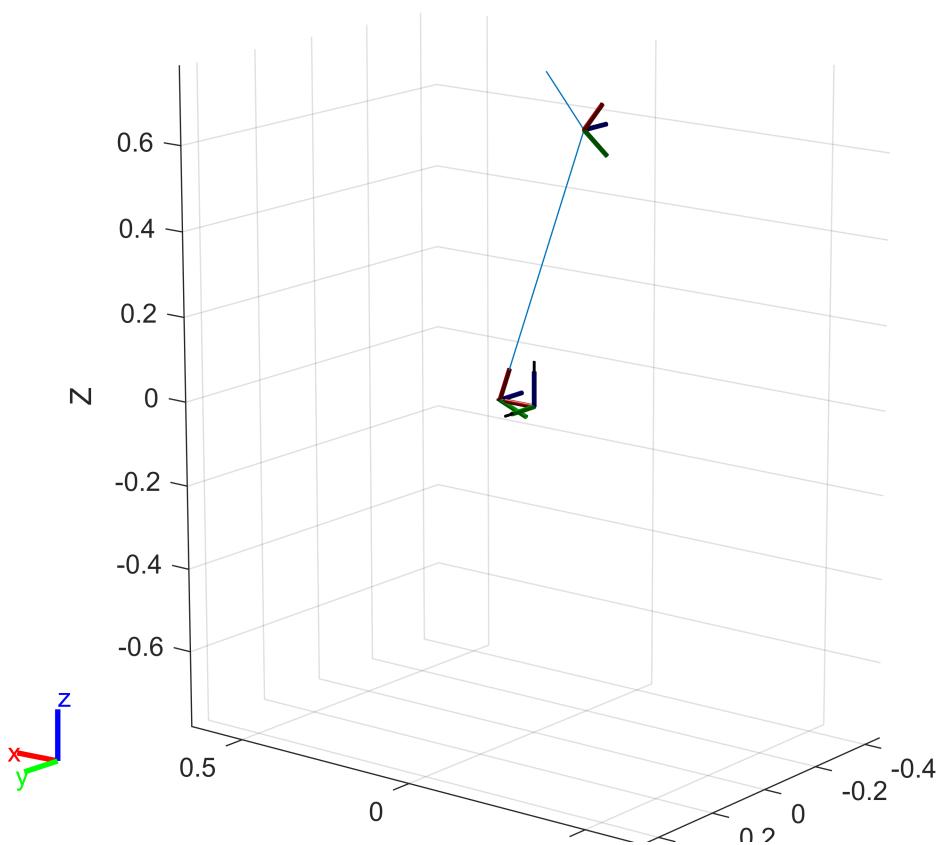


POSE #4:
 3.1416 0.8727 0.0000 0.5891 -0.0000 0.2260
 Angulos q resultantes
 -0.0000 110.0000 -60.0000 0.0000 0.0000 0.0000



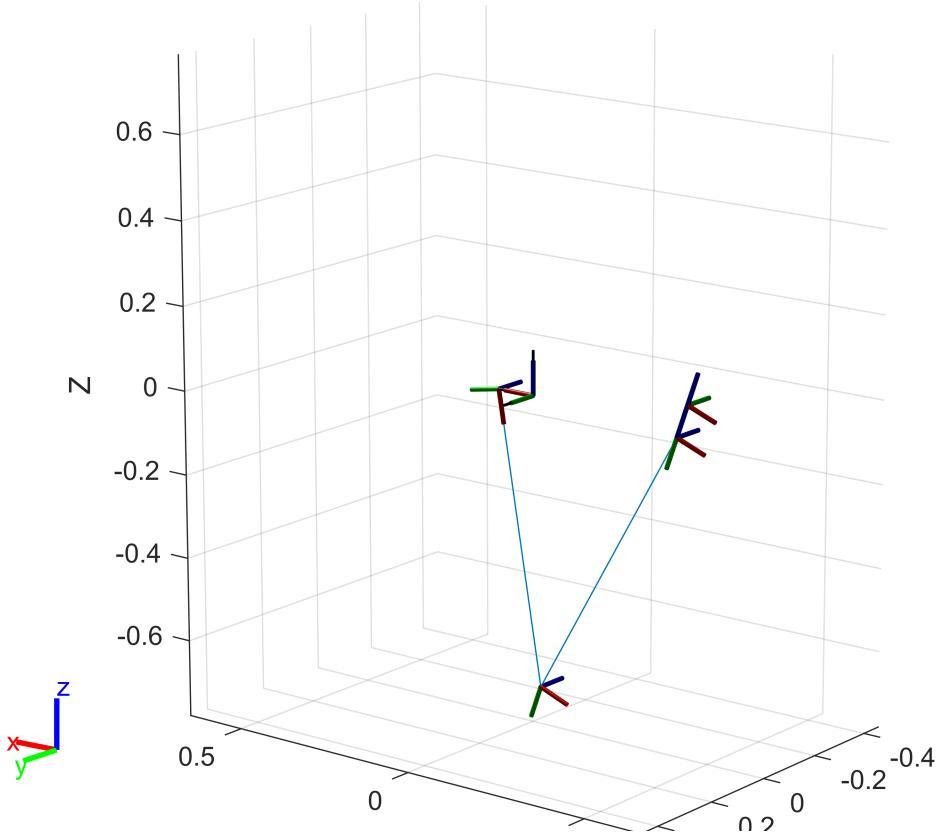
POSE #5:

0.0000	0.9023	3.1416	0.4344	0.0000	1.2890
Angulos q resultantes			0.0000	0.0000	-0.0000
0.0000	110.0000	18.3000	0.0000	0.0000	-0.0000

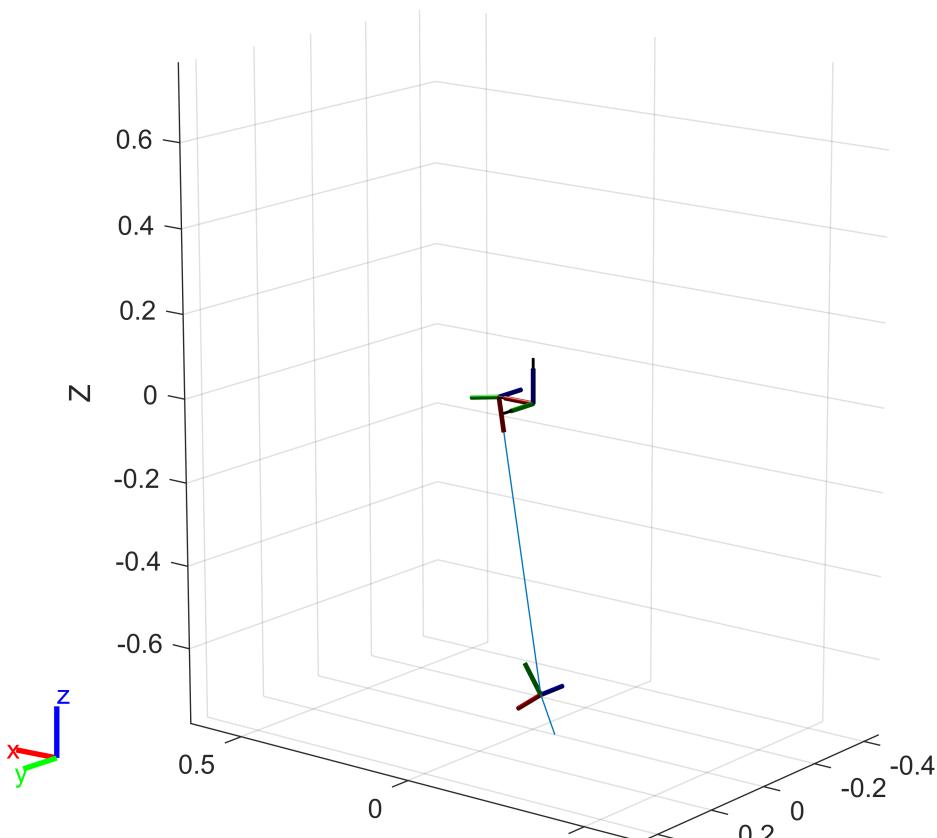


POSE #6:

0.0000	-0.3491	-3.1416	-0.4366	-0.0000	0.0489
Angulos q resultantes			0.0000	0.0000	0.0000
0.0000	-100.0000	-60.0000	0.0000	0.0000	0.0000

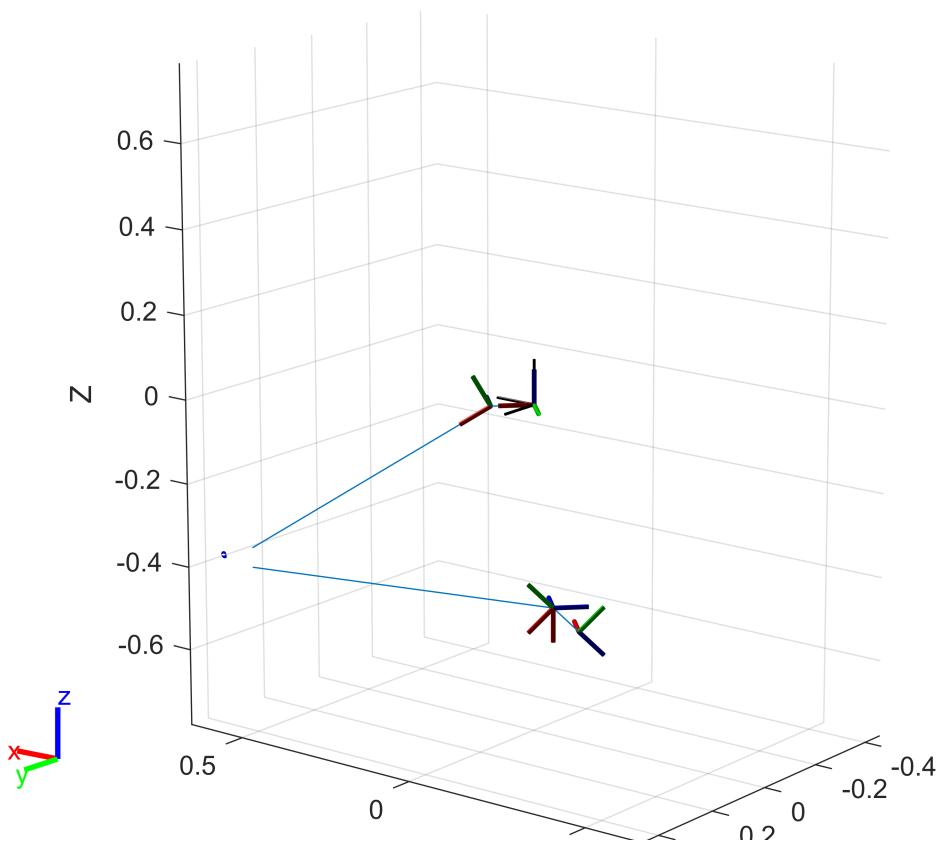


POSE #7:
 3.1416 -0.6109 -0.0000 -0.3936 -0.0000 -1.4598
 Angulos q resultantes
 -0.0000 -100.0000 65.0000 0.0000 -0.0000 -0.0000



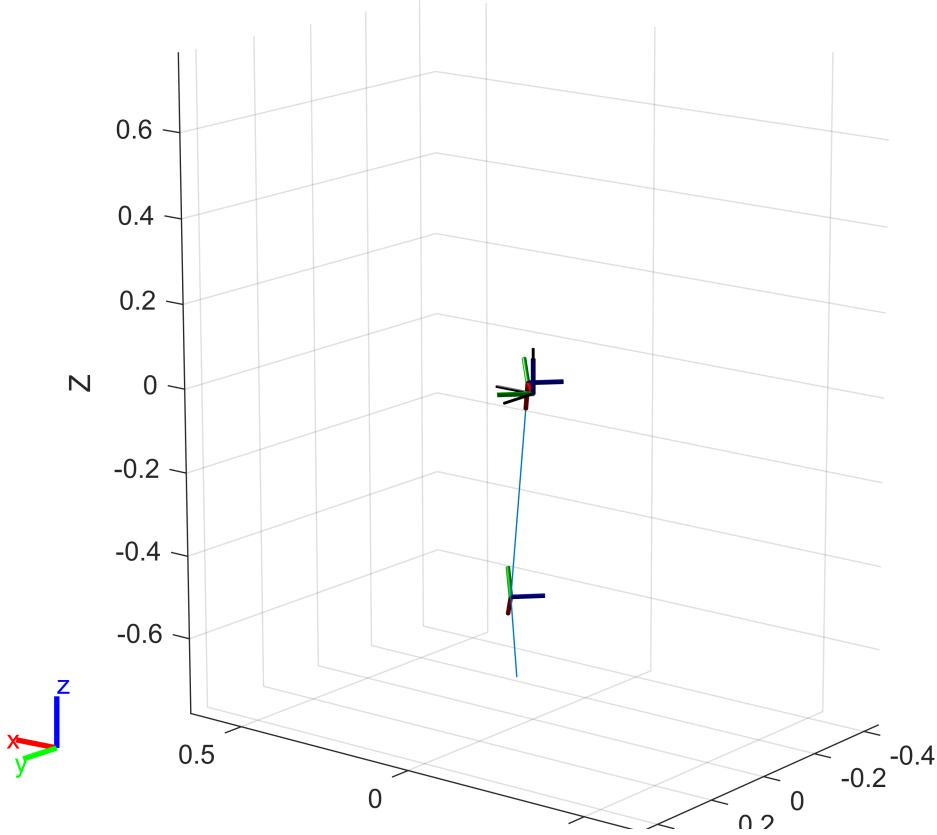
POSE #8:

2.5261	-0.5236	0.6155	-0.0739	-0.0739	-0.5476
Angulos q resultantes					
45.0000	-30.0000	-60.0000	0.0000	45.0000	90.0000

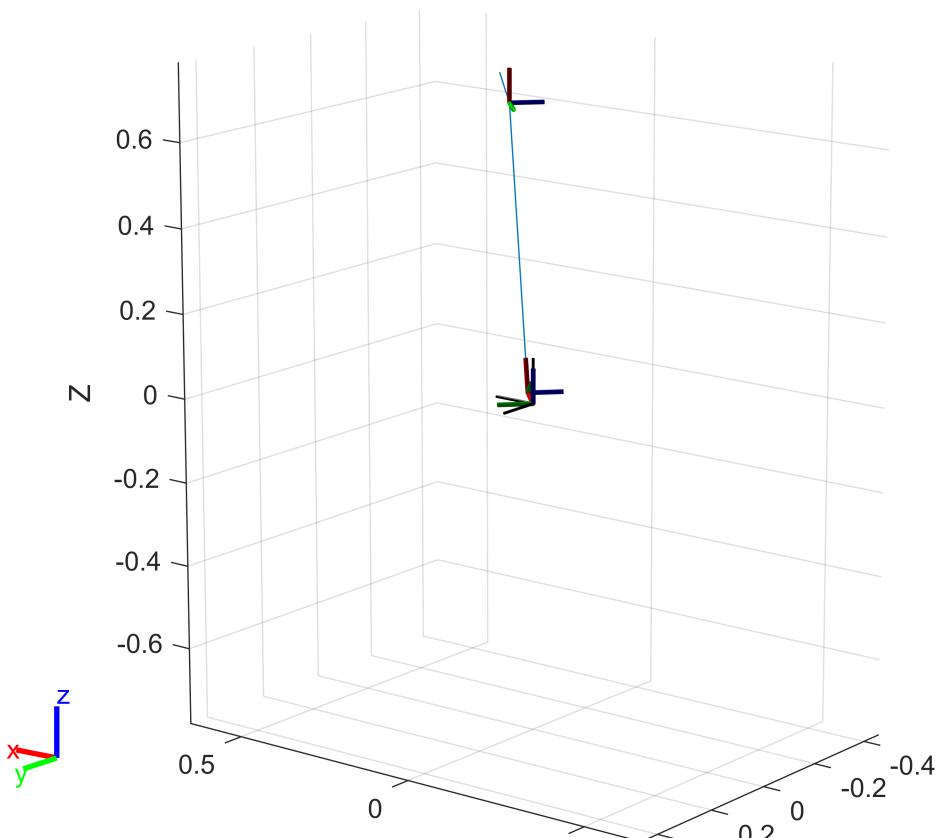


POSE #9:

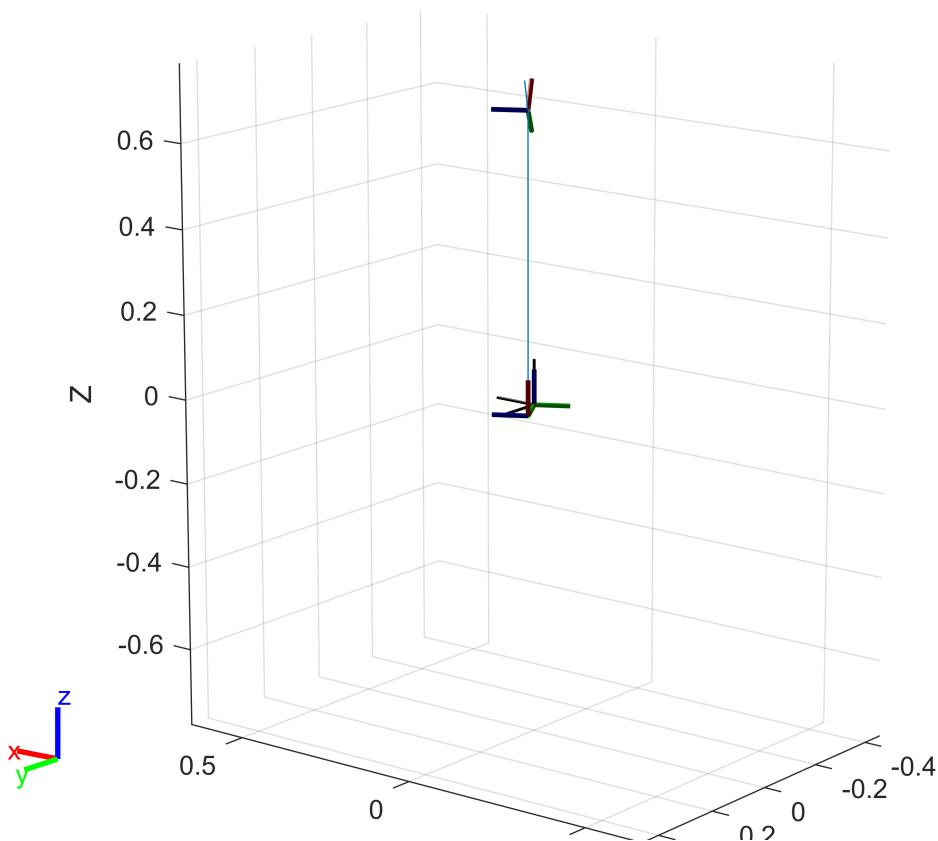
-2.6877	0.1822	-1.2868	0.0054	0.0466	-1.3150
Angulos q resultantes					
-45.0000	-61.4690	15.3304	-40.0678	42.2757	-87.0015



POSE #10:
 1.5708 0.7854 0.2618 0.9139 -0.9139 0.7455
 Angulos q resultantes
 -45.0000 60.0000 30.0000 -36.1034 0.0000 -38.8966



POSE #11:
 0.4449 -0.5128 -2.9976 -0.3109 0.4344 1.4012
 Angulos q resultantes
 120.0000 90.0000 45.0000 -45.0000 60.0000 270.0000



3.5. Comparación de métodos

Teniendo en cuenta la facilidad de programación, así como las opciones disponibles en cuanto a los métodos para calcular la cinemática inversa, se observa una mayor versatilidad con el RVC Toolbox, puesto que permite una creación del robot relativamente sencilla. Igualmente, ofrece una gama de opciones para optimizar la estimación de la cinemática inversa, de acuerdo con las necesidades y restricciones existentes.

Respecto a la manipulación de MTH's, las funciones disponibles son muy similares, aunque el acceso y manipulación de las MTH se facilita más con RVC Toolbox, dada la sencillez de su notación.

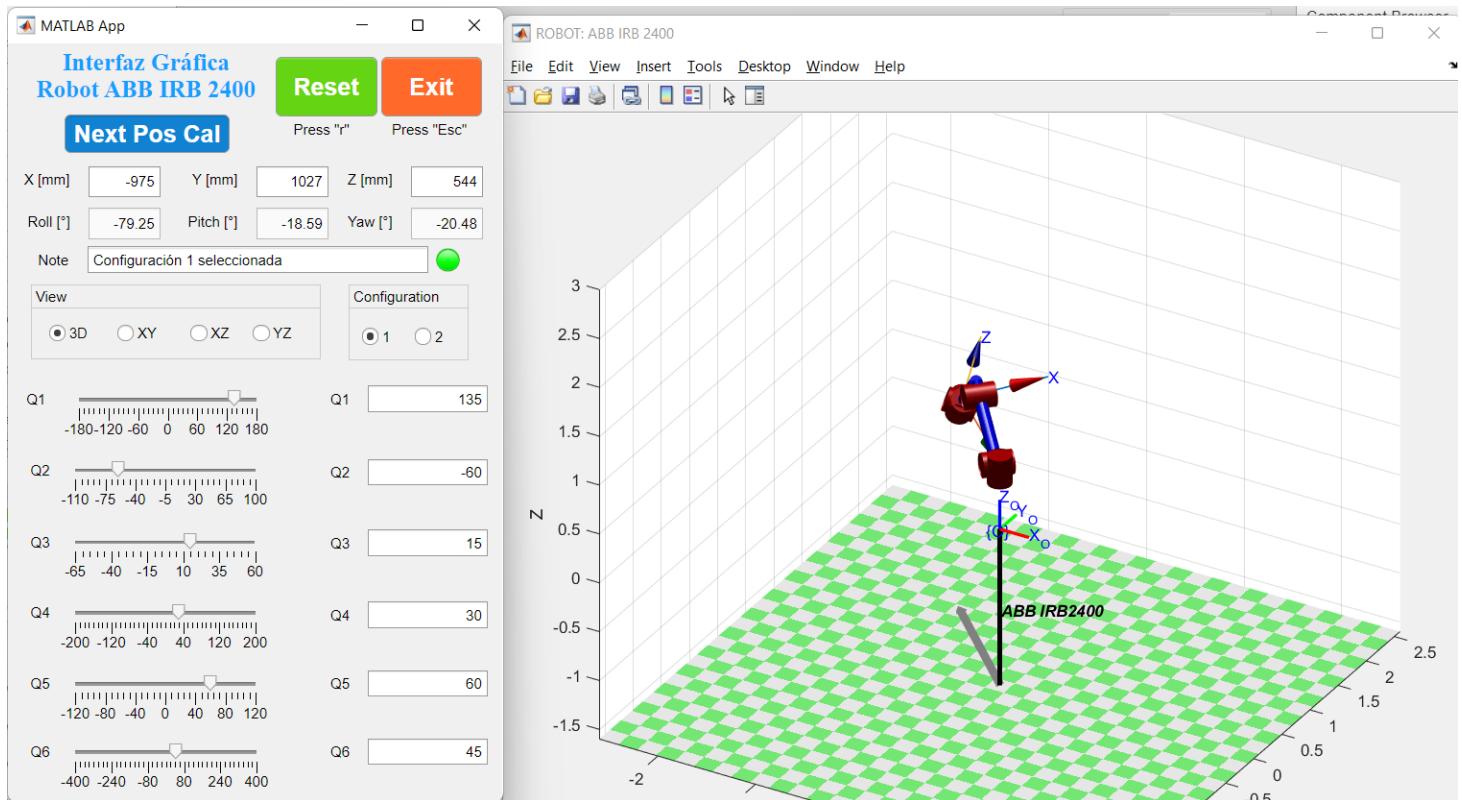
3.7. Posturas propuestas

Se propusieron las siguientes 4 posturas, obteniéndose los resultados ilustrados en la siguiente tabla:

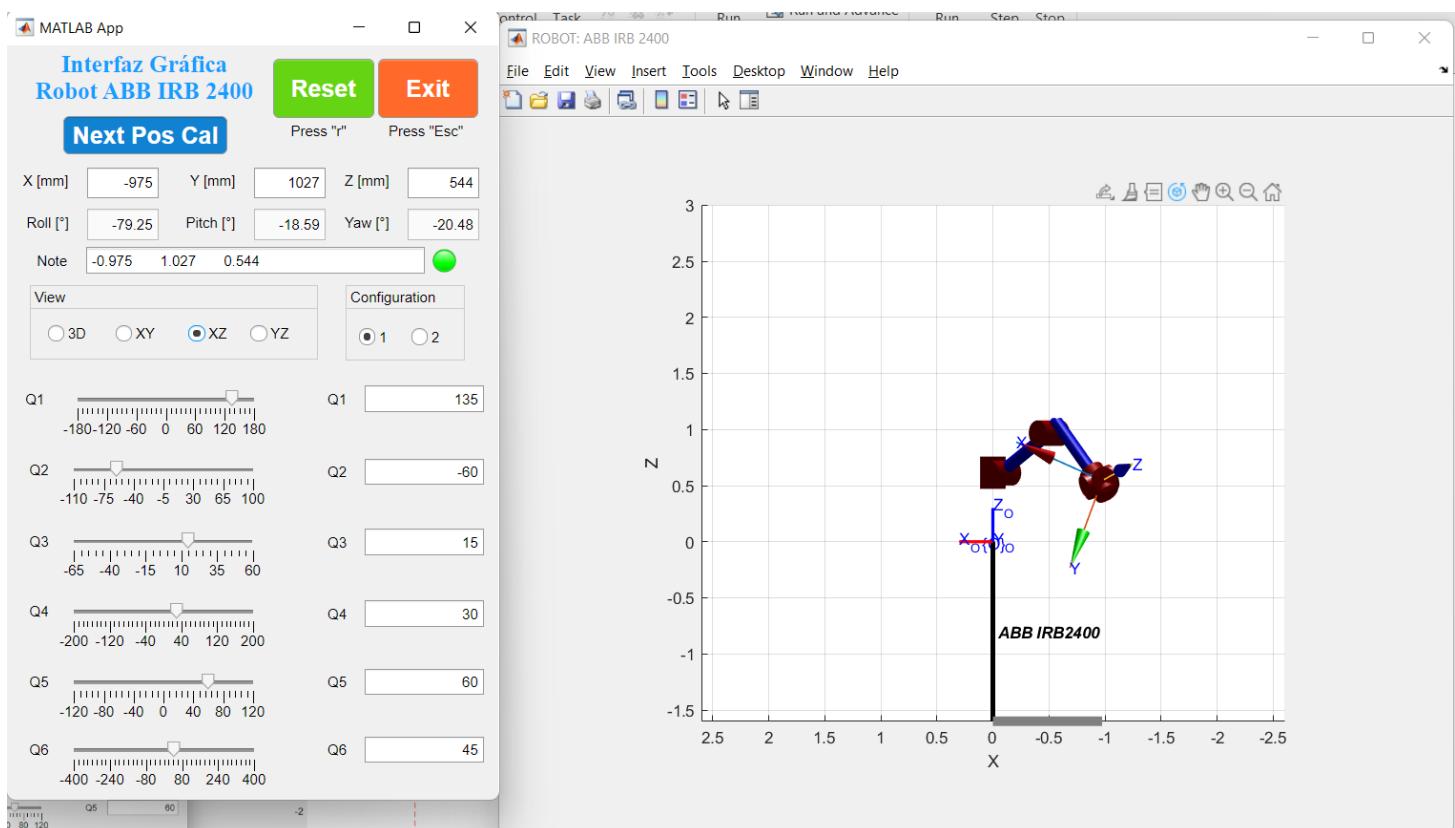
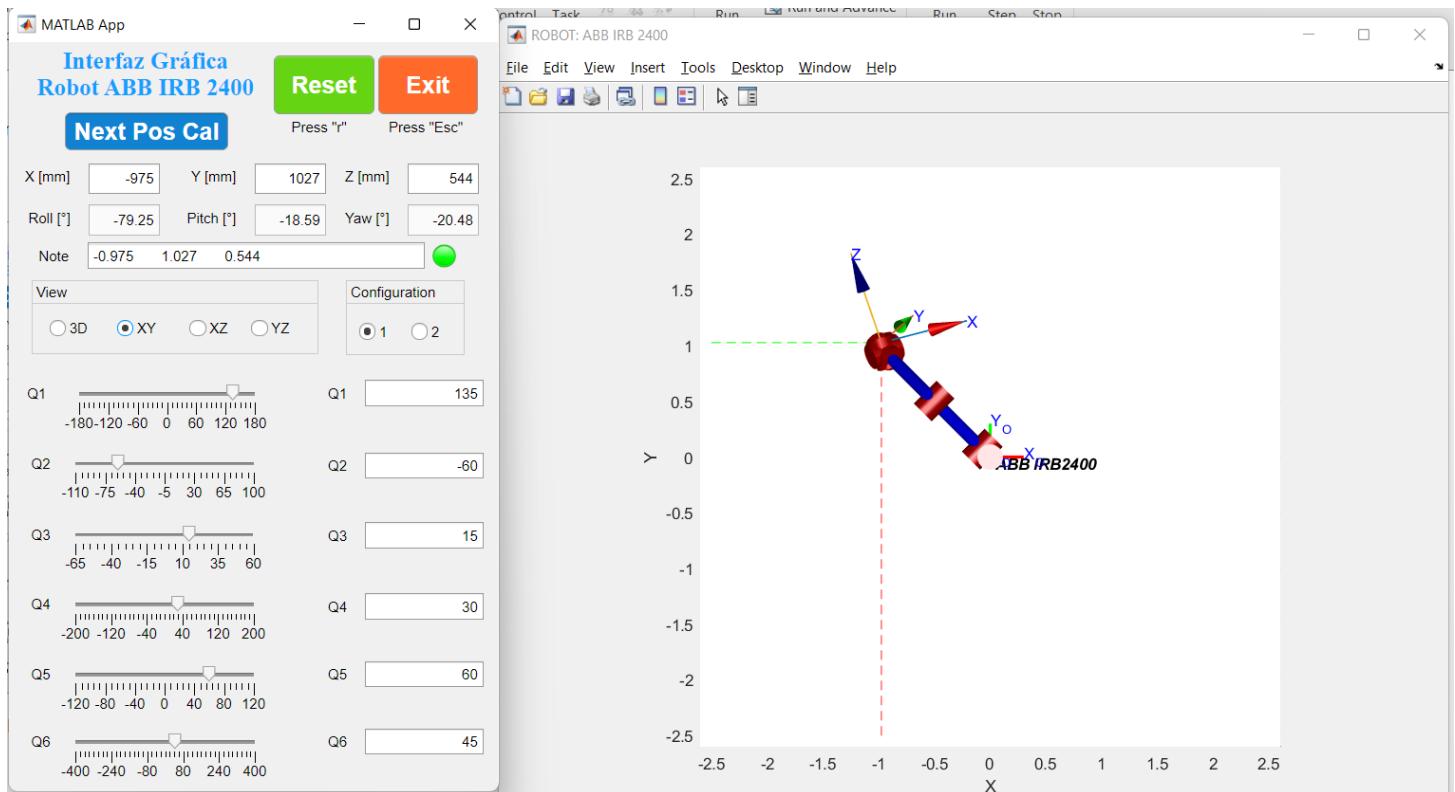
3.8. Verificación de configuraciones calculadas

3.8.1. Pose 1

Ingresando los valores de Q1, Q2, Q3, Q4, Q5, y Q6, se obtiene los siguientes valores:

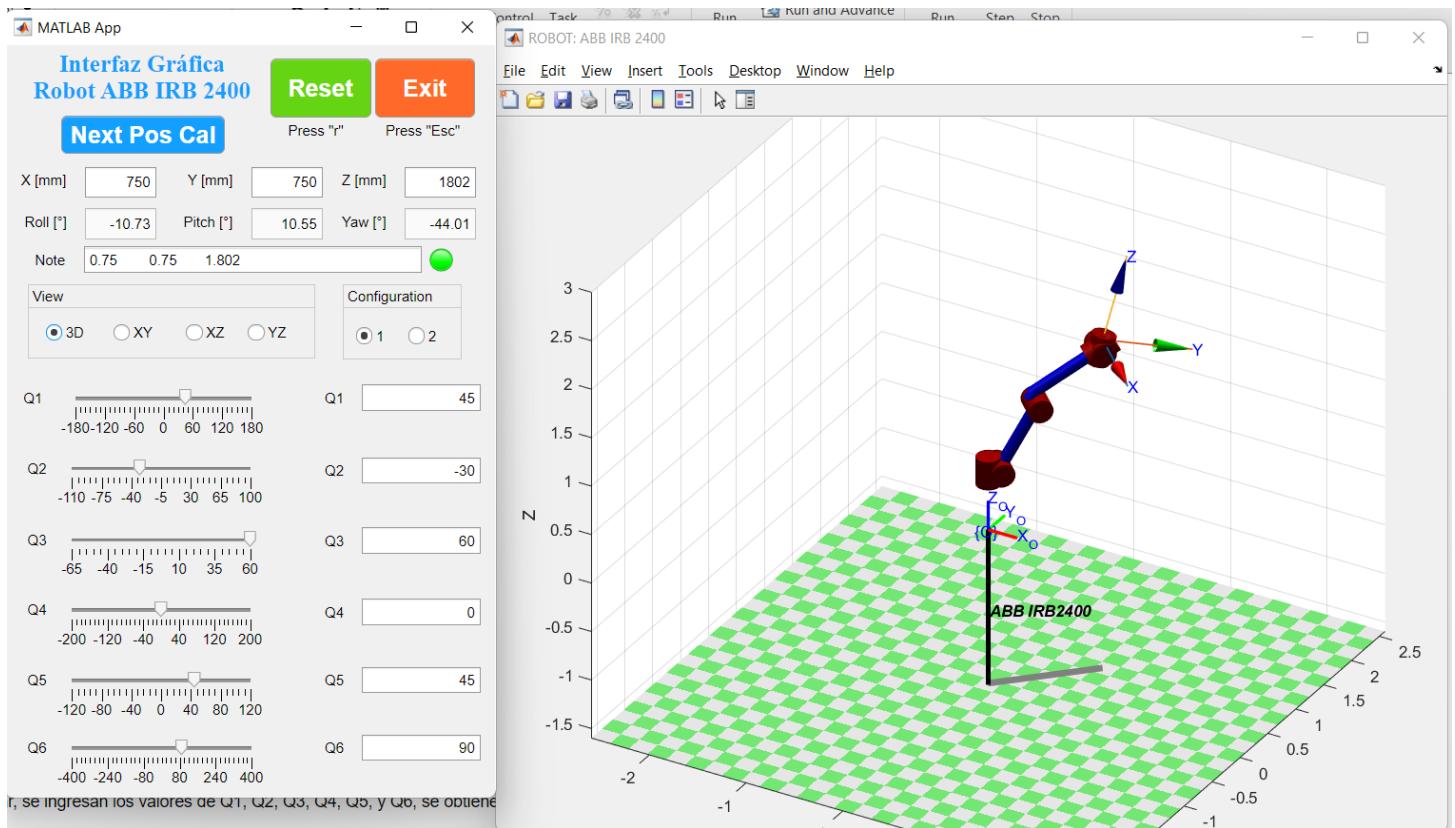


Para validar la posición en la posición x,y,z se tiene:

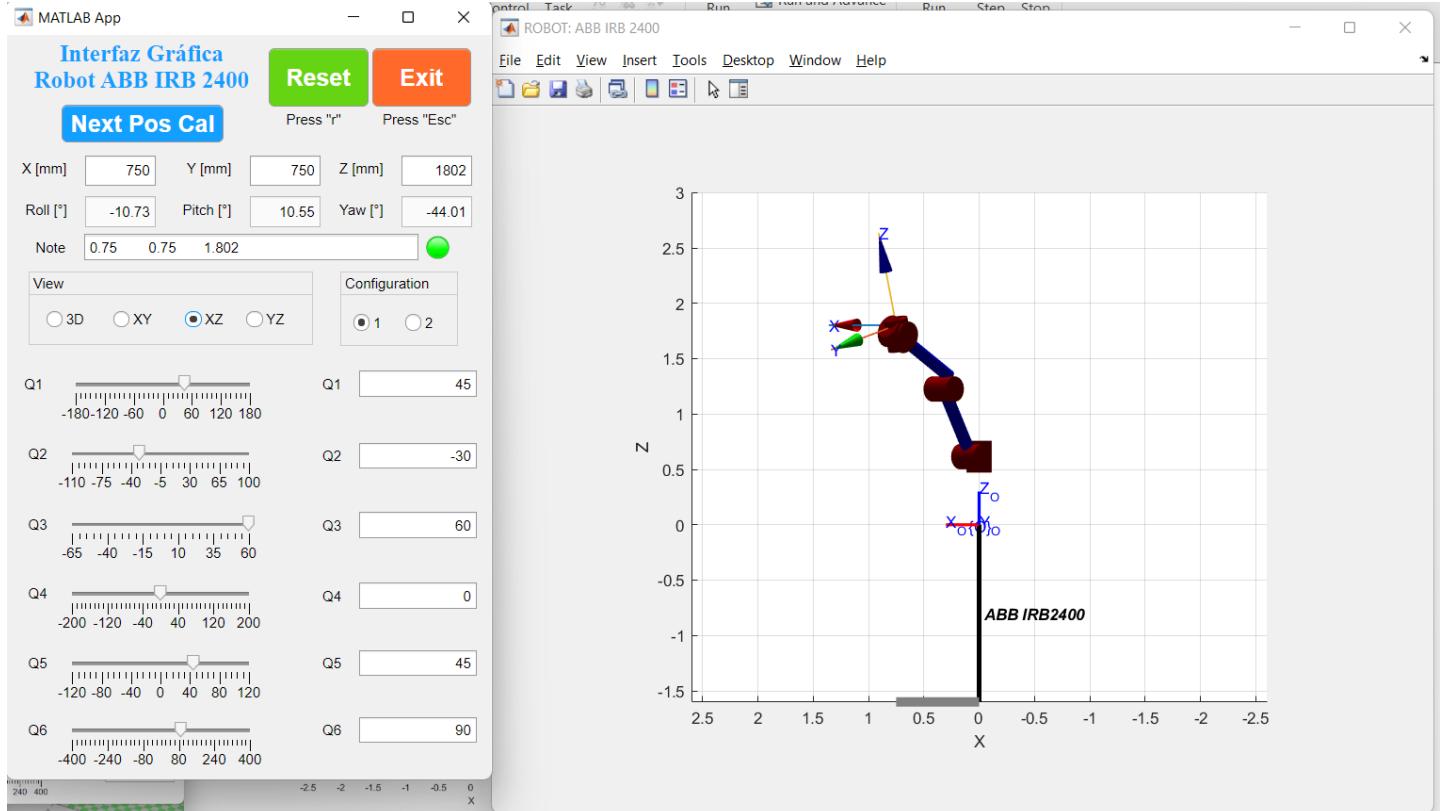
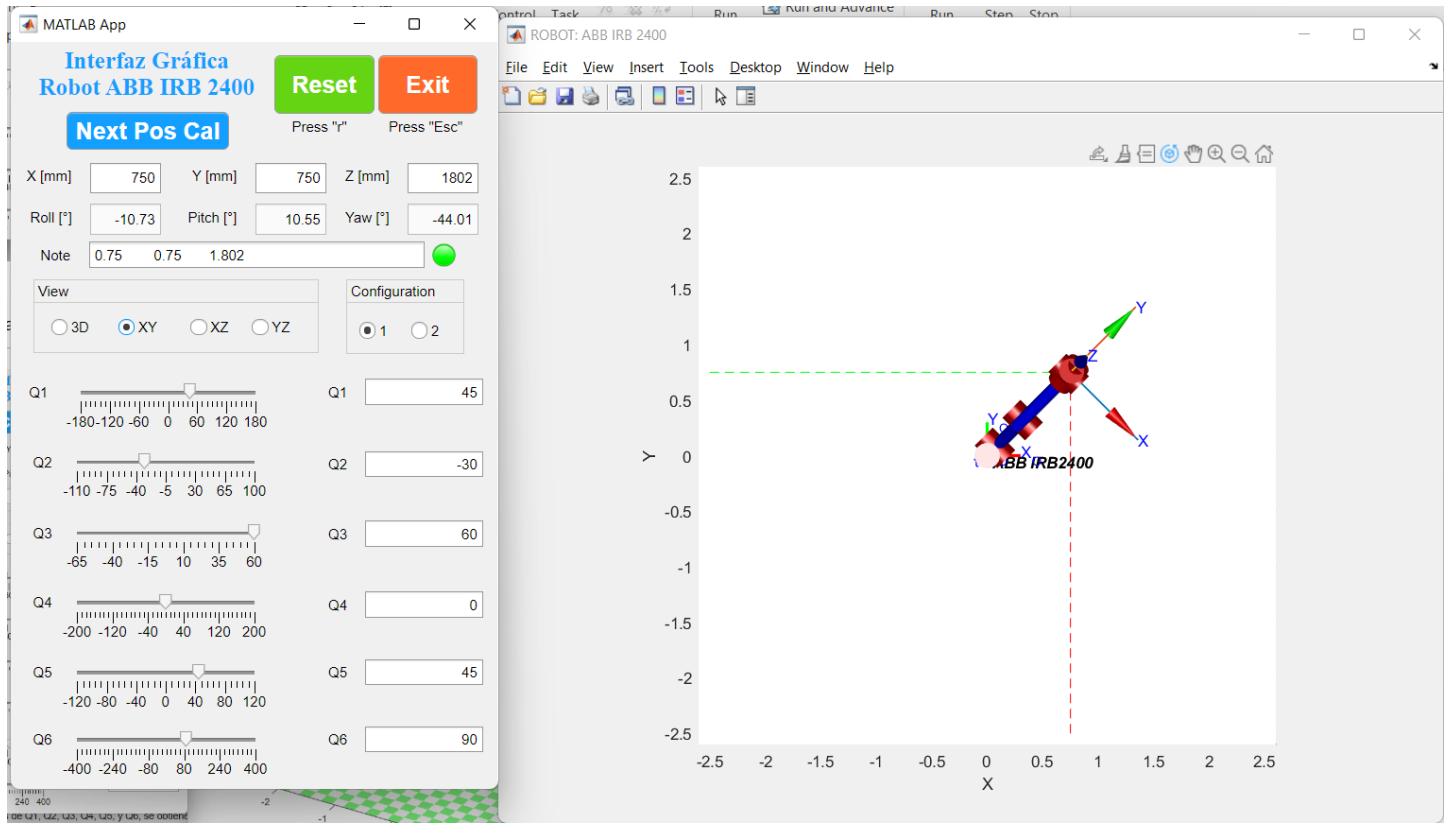


3.8.2. Pose 2:

De forma similar, se ingresan los valores de Q1, Q2, Q3, Q4, Q5, y Q6, se obtiene los siguientes valores:

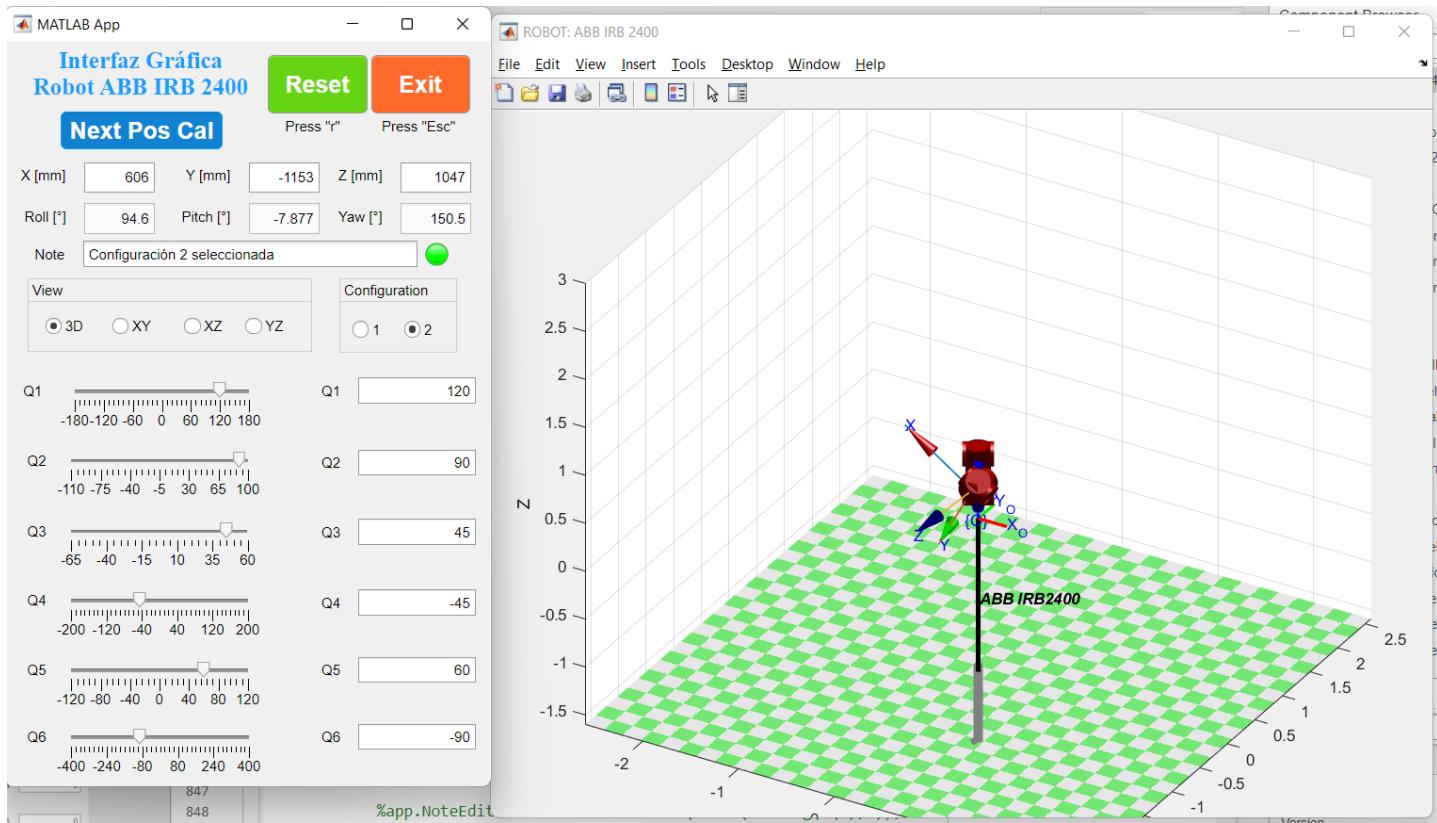


Validando los valores de x,y,z se tiene:

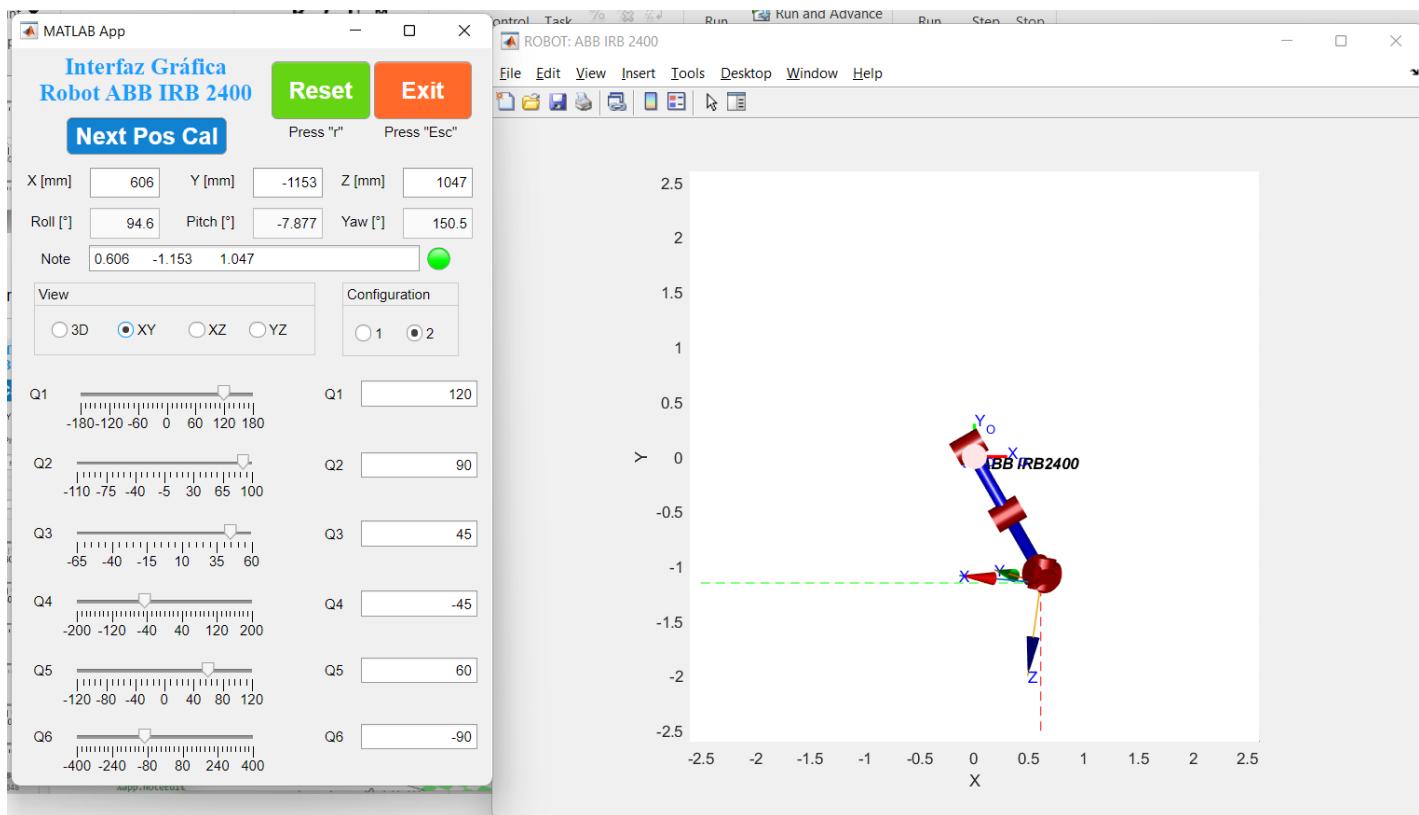


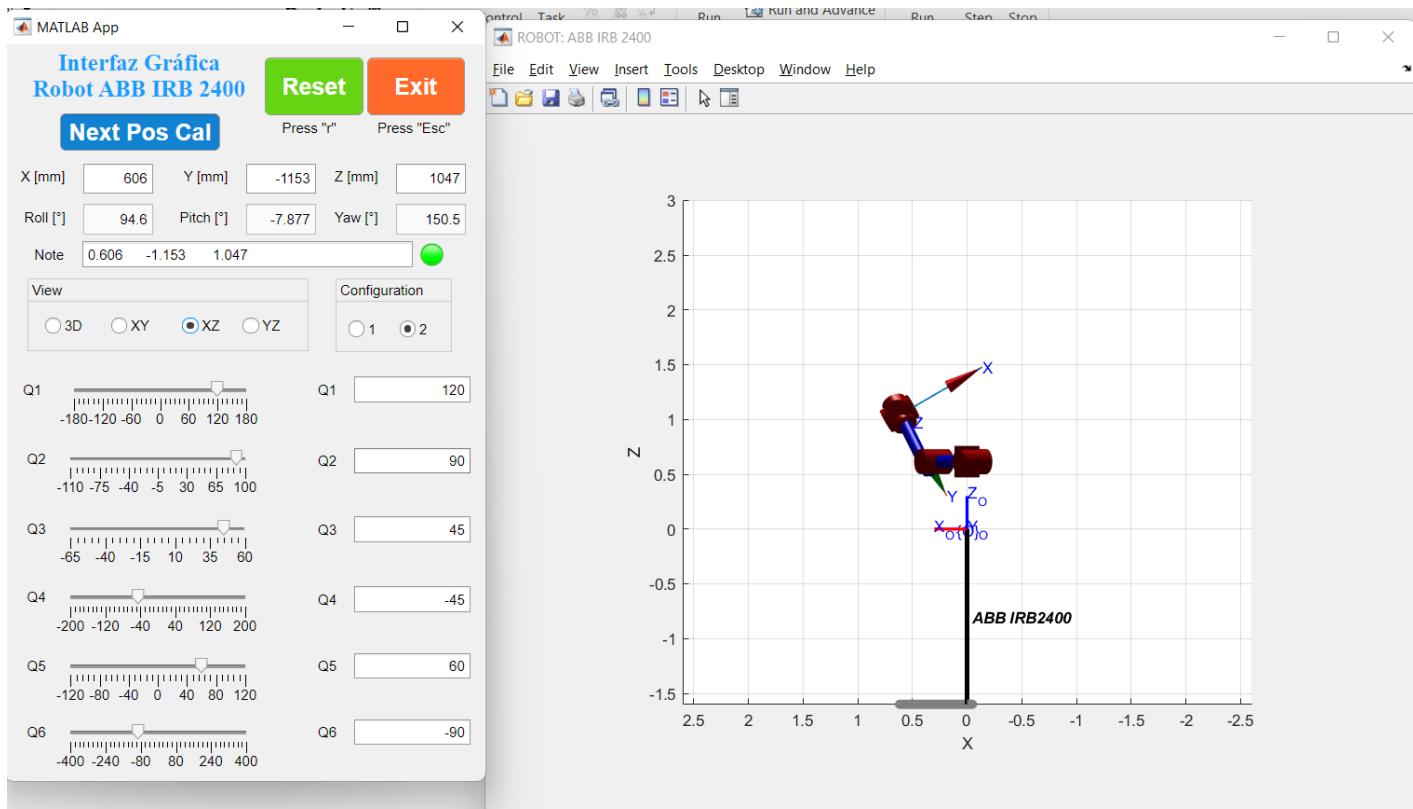
3.8.3. Pose 3:

De igual manera, se ingresan los valores de Q1, Q2, Q3, Q4, Q5, y Q6, se obtiene los siguientes valores:



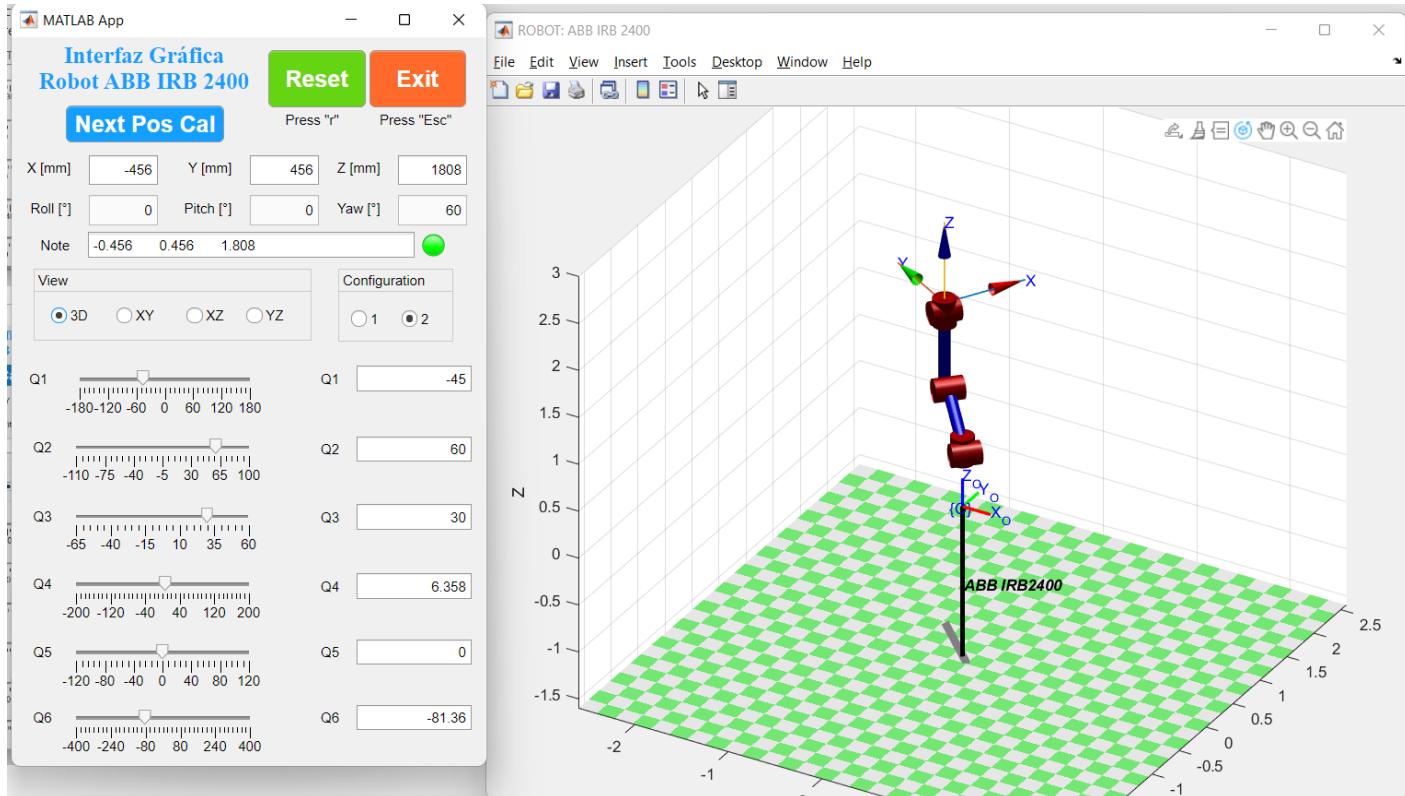
Validando los valores de x,y,z se tiene:



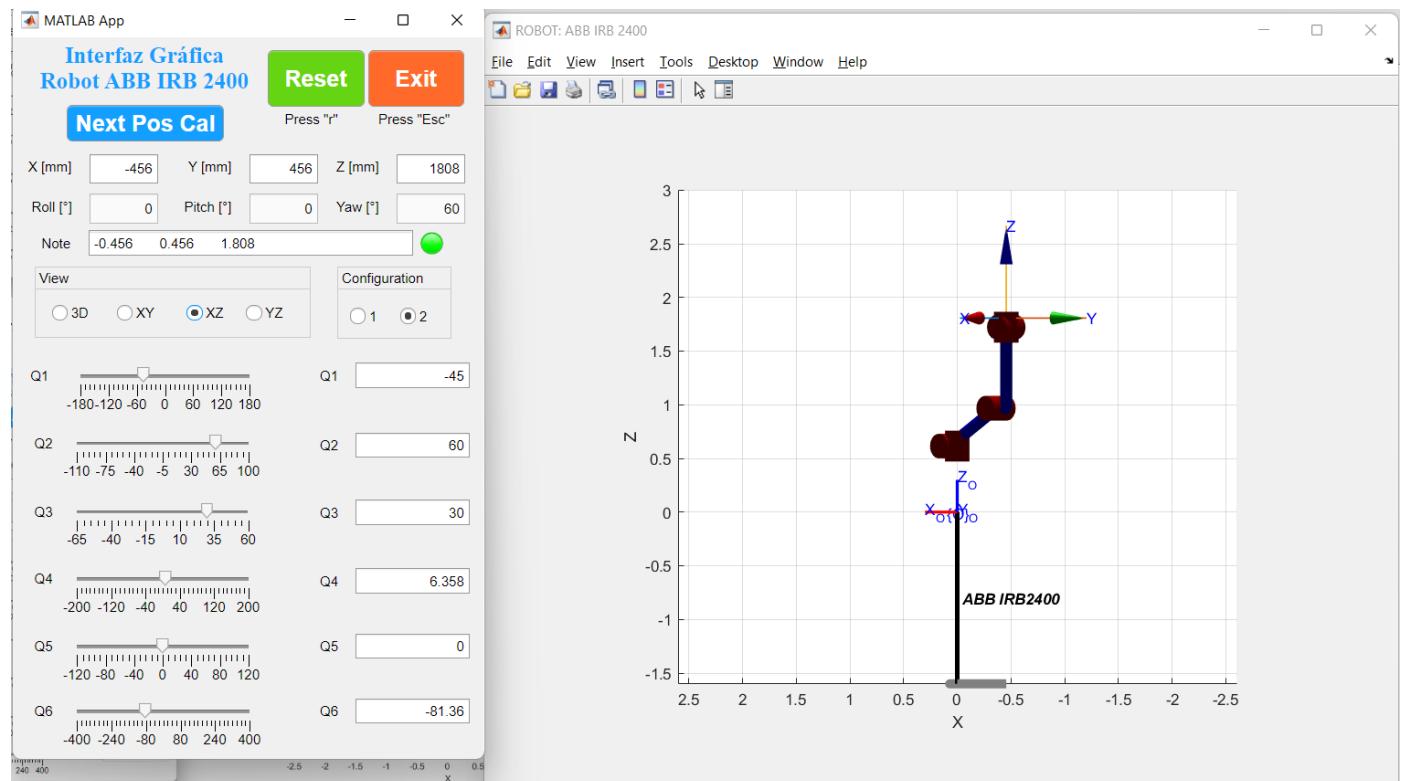
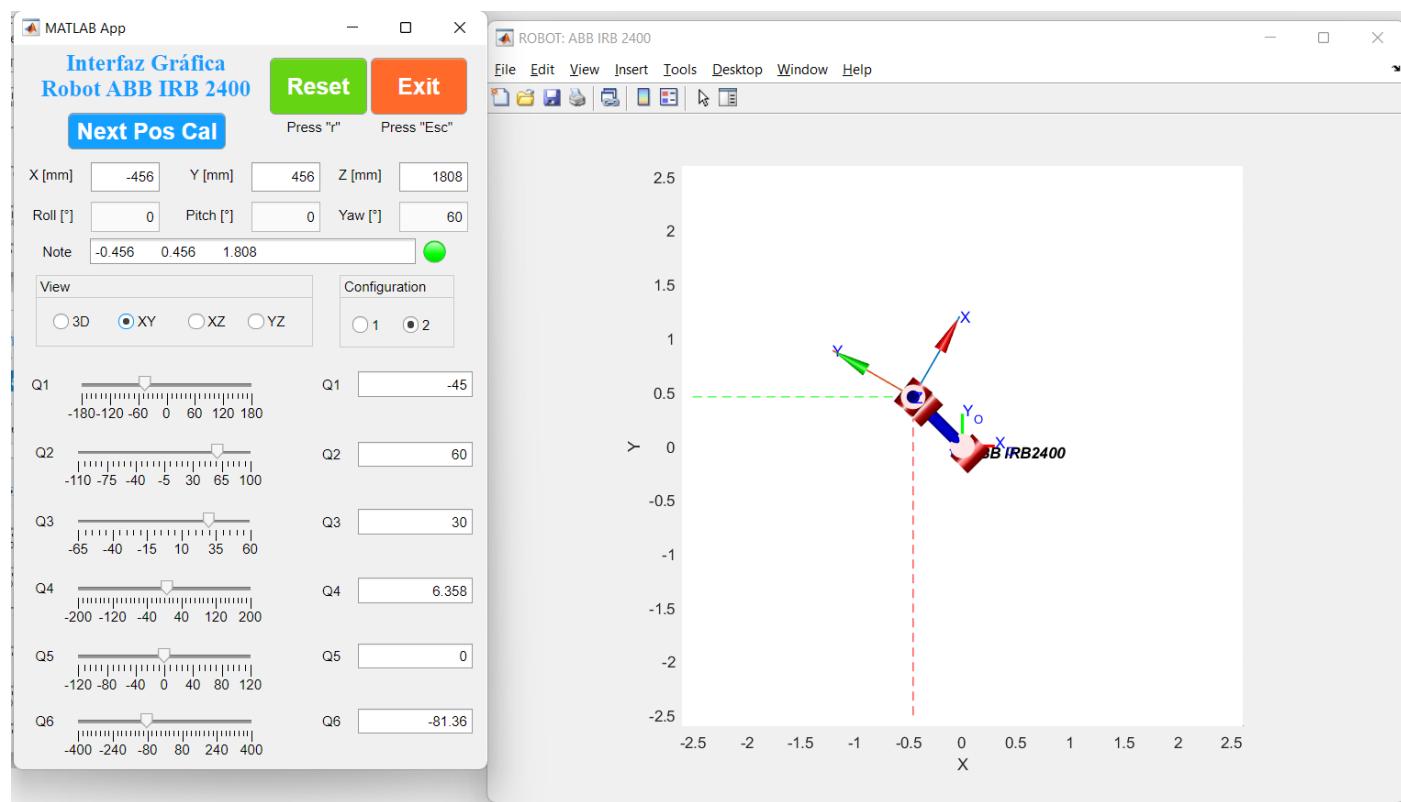


3.8.4. Pose 4:

Continuando, se ingresan los valores de Q1, Q2, Q3, Q4, Q5, y Q6, se obtiene los siguientes valores:



Validando los valores de x,y,z se tiene:



Funciones **fun_pose** y **cinematical inversa**:

```

function [POSE]=fun_pose(q,robotabb_rst)
config = homeConfiguration(robotabb_rst);
config(1).JointPosition = deg2rad(q(1));
config(2).JointPosition = deg2rad(q(2));
config(3).JointPosition = deg2rad(q(3));
config(4).JointPosition = deg2rad(q(4));
config(5).JointPosition = deg2rad(q(5));
config(6).JointPosition = deg2rad(q(6));
tform = getTransform(robotabb_rst,config,'link6','base'); %Es la MTH del EF en la configuracion
angulos=tr2rpy(tform);
POSE=[angulos(1),angulos(2),angulos(3),tform(1,4),tform(2,4),tform(3,4)]; % Roll Pitch Yaw
end

function [configSoln,solnInfo,q]=cinematicaInversa(V_pose,robotabb_rst)

q=zeros(6,1);
tform_t=transl1(V_pose(4),V_pose(5),V_pose(6));
tform_r=trotx(V_pose(1))*trot(y(V_pose(2)))*trot(z(V_pose(3)));
tform=[tform_r(1:3,1:3),tform_t(1:3,4);0,0,0,1];

ik = inverseKinematics('RigidBodyTree',robotabb_rst);
weights = [0.1 0.1 0.1 0.1 0.1 0.1];
initialguess = robotabb_rst.homeConfiguration;

[configSoln,solnInfo] = ik('link6',tform,weights,initialguess);

for i=1:6
    q(i)=rad2deg(configSoln(i).JointPosition(1));
end

end

```