ESTUDIO DE LA CINEMÁTICA DIRECTA Y CINEMÁTICA INVERSA DE UN MANIPULADOR PLANO DE 3 GRADOS DE LIBERTAD

Eduardo Vieira
Universidad Central de Venezuela
Facultad de Ingeniería
Escuela de Ingeniería Mecánica
eduardo.vieira@ucv.ve
Profesor: Arturo Gil

I. Introducción

La cinemática del robot estudia el movimiento del mismo con respecto a un sistema de referencia sin considerar las fuerzas que intervienen. Así, la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares.
Existen dos problemas fundamentales a resolver en la cinemática del robot, el primero de ellos se conoce como el problema cinemático directo, y consiste en determinar cuál es la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas que se toma como referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot; el segundo, denominado problema cinemático inverso, resuelve la configuración que debe adoptar el robot para una posición y orientación en su extremo conocidas.
En el presente trabajo se estudiará el problema cinemático directo e inverso para un manipulador plano de 3 grados de libertad. Además de desarrollará un algoritmo en python para la solución interactiva del problema.

II. El Problema

I. Se tiene

Se tiene: El siguiente manipulador plano

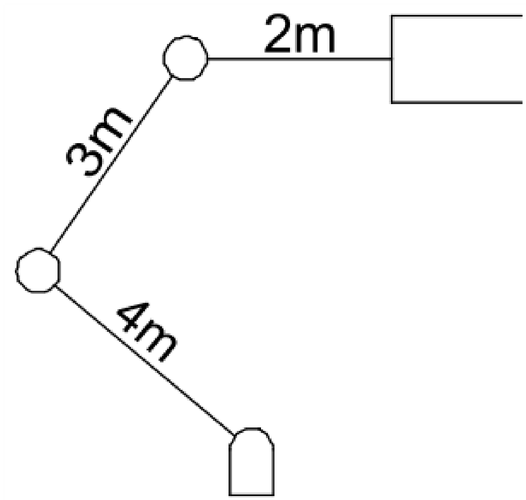

Figura 1: Manipulador plano de 3 grados de libertad

II. Se pide

1. Definir la matriz de Denvit-Hatenberg.
2. Estudiar la cinématica directa.
3. Estudiar la cinématica inversa.

III. Cinemática Directa

III.1 Algoritmo de Denavit-Hartenberg

1. Numerar los eslabones comenzando con 1 (primer eslabón móvil dela cadena) y acabando con n (último eslabón móvil). Se numerará como eslabón 0 a la base fija del robot.

2. Numerar cada articulación comenzando por 1 (la correspondiente al primer grado de libertad) y acabando en n.

3. Localizar el eje de cada articulación. Si esta es rotativa, el eje será su propio eje de giro. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

4. Para i de 0 a $n-1$, situar el eje Z_i, sobre el eje de la articulación $i+1$.

5. Situar el origen del sistema de la base (S_0) en cualquier punto del eje Z_0. Los ejes X_0 e Y_0 se situaran dé modo que formen un sistema dextrógiro con Z_0.

6. Para i de 1 a $n-1$, situar el sistema (S_i) (solidario al eslabón i) en la intersección del eje Z_i con la línea normal común a Z_{i-1} y Z_i. Si ambos ejes se cortasen se situaría (S_i) en el punto de corte. Si fuesen paralelos (S_i) se situaría en la articulación $i+1$.

7. Situar X_i en la línea normal común a Z_{i-1} y Z_i.

8. Situar Y_i de modo que forme un sistema dextrógiro con X_i y Z_i.

9. Situar el sistema (S_n) en el extremo del robot de modo que Z_n coincida con la dirección de Z_{n-1} y X_n sea normal a Z_{n-1} y Z_n.

10. Obtener θ_i como el ángulo que hay que girar en torno a Z_{i-1} para que X_{i-1} y X_i queden paralelos.

11. Obtener D_i como la distancia, medida a lo largo de Z_{i-1}, que habría que desplazar (S_{i-1}) para que X_i y X_{i-1} quedasen alineados.

12. Obtener A_i como la distancia medida a lo largo de X_i (que ahora coincidiría con X_{i-1}) que habría que desplazar el nuevo (S_{i-1}) para que su origen coincidiese con (S_i).

13. Obtener α_i como el ángulo que habría que girar entorno a X_i (que ahora coincidiría con X_{i-1}), para que el nuevo (S_{i-1}) coincidiese totalmente con (S_i).

14. Obtener las matrices de transformación ${}^{i-1}A_i$.

15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot $T = {}^0 A_i^1 A_2 \ldots {}^{n-1}A_n$.

16. La matriz T define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido ala base en función de las n coordenadas articulares.

Al aplicar el algorítmo quedaría

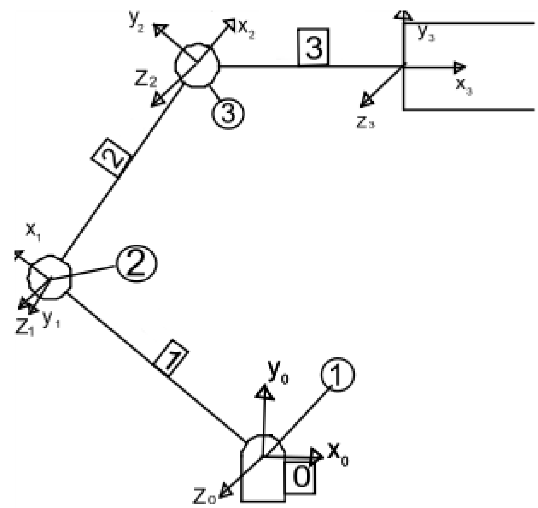

Figura 2: Algoritmo de Denavit-Hartenberg aplicado al manipulador

Los pafametros para cada articulación son

Articulación	θ	d	a	α
1	q_1	0	l_1	0
2	$q_2 - \pi$	0	l_2	0
3	$q_3 - \pi$	0	l_3	0

Tabla 1: Parámetros de Denavit-Hartenberg

La matriz de transformación ${}^0T_1$ será

$$\begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 4\cos(q_1) \\ \sin(q_1) & \cos(q_1) & 0 & 4\sin(q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

${}^1T_2$

$$\begin{bmatrix} -\cos(q_2) & \sin(q_2) & 0 & -3\cos(q_2) \\ -\sin(q_2) & -\cos(q_2) & 0 & -3\sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Y finalmente ${}^2T_3$

$$\begin{bmatrix} -\cos(q_3) & \sin(q_3) & 0 & -2\cos(q_3) \\ -\sin(q_3) & -\cos(q_3) & 0 & -2\sin(q_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

La matriz de transformación que relaciona el extremo de la herramienta del robot con el sistema definido en la base T será $T = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3$

$$\begin{bmatrix} \alpha & \beta & 0 & \gamma \\ \delta & \epsilon & 0 & \kappa \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Donde $\alpha = -\left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\cos(q_3) - \left(\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1)\right)\sin(q_3)$

$\beta = \left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\sin(q_3) - \left(\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1)\right)\cos(q_3)$

$\gamma = -2\left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\cos(q_3) - 2\left(\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1)\right)\sin(q_3) + 3\sin(q_1)\sin(q_2) - 3\cos(q_1)\cos(q_2) + 4\cos(q_1)$

$\delta = -\left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\sin(q_3) - \left(-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1)\right)\cos(q_3)$

$\epsilon = -\left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\cos(q_3) + \left(-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1)\right)\sin(q_3)$

$\kappa = -2\left(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2)\right)\sin(q_3) - 2\left(-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1)\right)\cos(q_3) - 3\sin(q_1)\cos(q_2) + 4\sin(q_1) - 3\sin(q_2)\cos(q_1)$

Sean x_r, y_r y z_r las coordenadas de un punto medidas desde el sistema de referencia ubicado en el extremo del robot.

$$P_r = \begin{bmatrix} x_r \\ y_r \\ z_r \\ 1 \end{bmatrix} \tag{5}$$

Entonces, el mismo punto medido en el sistema de referencia de la base será $T \cdot P_r$
$x = x_r(-(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\cos(q_3) - (\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1))\sin(q_3)) + y_r((\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\sin(q_3) - (\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1))\cos(q_3)) - 2(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\cos(q_3) - 2(\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1))\sin(q_3) + 3\sin(q_1)\sin(q_2) - 3\cos(q_1)\cos(q_2) + 4\cos(q_1)$

$y = x_r(-(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\sin(q_3) - (-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1))\cos(q_3)) + y_r(-(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\cos(q_3) + (-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1))\sin(q_3)) - 2(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\sin(q_3) - 2(-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1))\cos(q_3) - 3\sin(q_1)\cos(q_2) + 4\sin(q_1) - 3\sin(q_2)\cos(q_1)$

Si colocamos el sistema de referencia en la base (x_r y y_r iguales a cero) el punto que determina la ubicación de la herramienta se obtiene de la siguiente manera
$x = -2(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\cos(q_3) - 2(\sin(q_1)\cos(q_2) + \sin(q_2)\cos(q_1))\sin(q_3) + 3\sin(q_1)\sin(q_2) - 3\cos(q_1)\cos(q_2) + 4\cos(q_1)$

$y = -2(\sin(q_1)\sin(q_2) - \cos(q_1)\cos(q_2))\sin(q_3) - 2(-\sin(q_1)\cos(q_2) - \sin(q_2)\cos(q_1))\cos(q_3) - 3\sin(q_1)\cos(q_2) + 4\sin(q_1) - 3\sin(q_2)\cos(q_1)$

III.2 Código en python

En la siguiente entrada determinaremos la posición de la herramienta modificando los valores de q_1, q_2 y q_3 en los widgets interactivos

```
In [1]: import matplotlib.pyplot as plt
        #plt.style.use('bmh')
        from numpy import sin, cos, pi, arctan2
        from IPython.html.widgets import interact, FloatSlider, interactive
        from IPython.display import display
        %matplotlib inline
        fact = 180 / pi
        def punto(q1,q2,q3):
            x = -2*(sin(q1)*sin(q2) - cos(q1)*cos(q2))*cos(q3) - 2 * \
            (sin(q1)*cos(q2) + sin(q2)*cos(q1))*sin(q3) + 3*sin(q1)*sin(q2) - \
```

```
    3*cos(q1)*cos(q2) + 4*cos(q1)

    y = -2*(sin(q1)*sin(q2) - cos(q1)*cos(q2))*sin(q3) - 2 * \
    (-sin(q1)*cos(q2) - sin(q2)*cos(q1))*cos(q3) - 3*sin(q1)*cos(q2) + \
    4*sin(q1) - 3*sin(q2)*cos(q1)

    x1 = 4*cos(q1)
    y1 = 4*sin(q1)
    x2 = 3*sin(q1)*sin(q2)-3*cos(q1)*cos(q2)+4*cos(q1)
    y2 = -3*sin(q1)*cos(q2)+4*sin(q1)-3*sin(q2)*cos(q1)
    plt.plot(x,y,'r+')
    plt.ylim(-9.5,9.5)
    plt.xlim(-9.5,9.5)
    plt.plot([0,x1],[0,y1],'k-')
    plt.plot([x1,x2],[y1,y2],'r-')
    plt.plot([x2,x],[y2,y],'b-')
    plt.plot([-1,1],[0,0],'k-')
    plt.grid(True)
    print "x [m] = ", x
    print "y [m] = ",y
    print "Ángulo [deg] = ", fact * arctan2((y-y2),(x-x2))
q1_slider = FloatSlider(min=-2*pi, max=2*pi, step=0.1, value=3*pi/4, \
                        description='Angulo $q_1$')
q2_slider = FloatSlider(min=-2*pi, max=2*pi, step=0.1, value=pi/2, \
                        description='Angulo $q_2$')
q3_slider = FloatSlider(min=-2*pi, max=2*pi, step=0.1, value=2*pi/3, \
                        description='Angulo $q_3$')
w=interactive(punto,q1=q1_slider,q2=q2_slider,q3=q3_slider)
display(w)
```

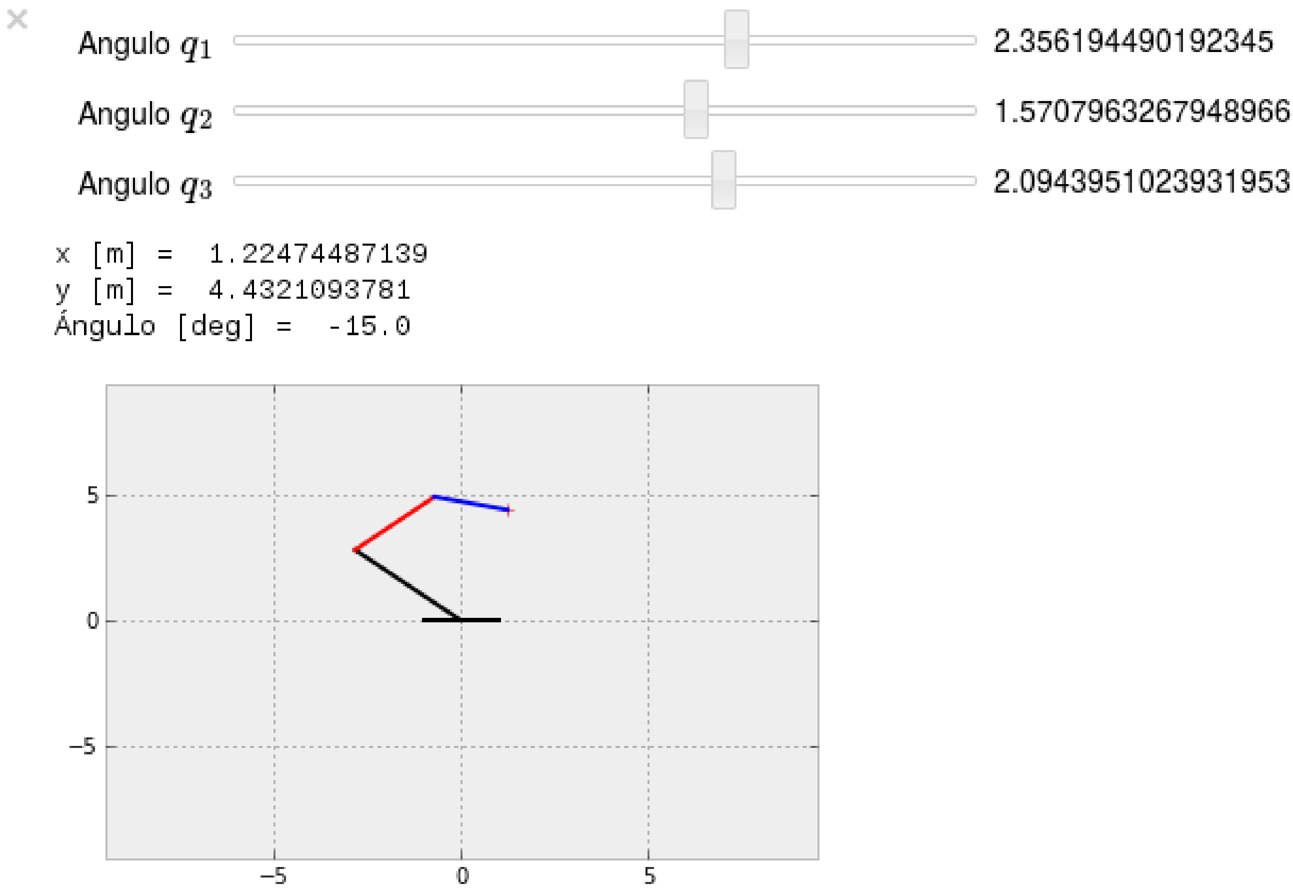

Figura 3: Captura de la ejecución del código en un Notebook de IPython

IV. Cinemática Inversa

Realizaremos la cinemática inversa mediate métodos geométricos. Sea Pher el punto desado de la herramienta con coordenadas x_h y y_h ($z_h = 0$ al tratarse de un robot plano) y formando un ángulo θ_h con la horizontal.

Por trigonometría las coordenadas de la articulación 3 serán $x_3 = x_h - l_3 * cos(\theta)$ y $y_3 = y_h - l_3 * sin(\theta)$. Podemos obtener q_1 y q_2 con $q_2 = arctg(\frac{\pm\sqrt{1-cos^2(q_2)}}{cos(q_2)})$ donde $cos(q_2) = \frac{x_3^2+y_3^2-l_1^2-l_2^2}{2l_1l_2}$ y $q_1 = arctg(\frac{y_3}{\pm x_3}) - arctg(\frac{l_2 sin(q_2)}{l_1+l_2cos(q_2)})$ (Barrientos, Fundamentos de Robótica, 2007).

IV.1 Código en python

```
In [2]: from numpy import sqrt, arcsin
         def cin_inv(x,y,theta):
             theta = theta / fact
             x_3 = x - l_3 * cos(theta)
             y_3 = y - l_3 * sin(theta)
             cos_q_2 = (x_3**2 + y_3**2 - l_1**2 - l_2**2) / (2 * l_1 * l_2)
             q_2 = arctan2(sqrt(1 - (cos_q_2)**2),cos_q_2)
             q_1 = arctan2(y_3,x_3) - arctan2((l_2 * sin(q_2)),(l_1 + l_2 * cos_q_2))
             l = sqrt(l_1**2 + l_2**2 - 2 * l_1 * l_2 * cos_q_2)
             alpha = arcsin(l_1 * sin(q_2) / l)
             beta = arctan2(y_3,x_3)
             k = pi - theta
```

```
    q_3 = (alpha + k + beta)
    plt.plot(x, y, 'r+')
    plt.ylim(-9.5, 9.5)
    plt.xlim(-9.5, 9.5)
    plt.plot([-1,1],[0,0],'k-')
    x_2 = l_1 * cos(q_1)
    y_2 = l_1 * sin(q_1)
    x_1 = 0
    y_1 = 0
    plt.plot([x_1,x_2],[y_1,y_2],'r-')
    plt.plot([x_2,x_3],[y_2,y_3],'b-')
    plt.plot([x_3,x],[y_3,y],'k-')
    plt.grid(True)
    print "q1 [deg] = ", q_1 * fact
    print "q2 [deg] = ", q_2 * fact
    print "q3 [deg] = ", q_3 * fact
    print "l1 [m] = ", sqrt((x_2**2)+(y_2**2)), l_1
    print "l2 [m] = ", sqrt(((x_2-x_3)**2)+((y_2-y_3)**2)), l_2
    print "l3 [m] = ", sqrt(((x_3-x)**2)+((y_3-y)**2)), l_3
x_slider = FloatSlider(min=-9, max=9, step=0.1, value=-1, description='X')
y_slider = FloatSlider(min=-9, max=9, step=0.1, value=4, description='Y')
theta_slider = FloatSlider(min=0, max=360, step=0.1, value=200, description='Theta')
w=interactive(cin_inv,x=x_slider,y=y_slider,theta=theta_slider)
display(w)
```

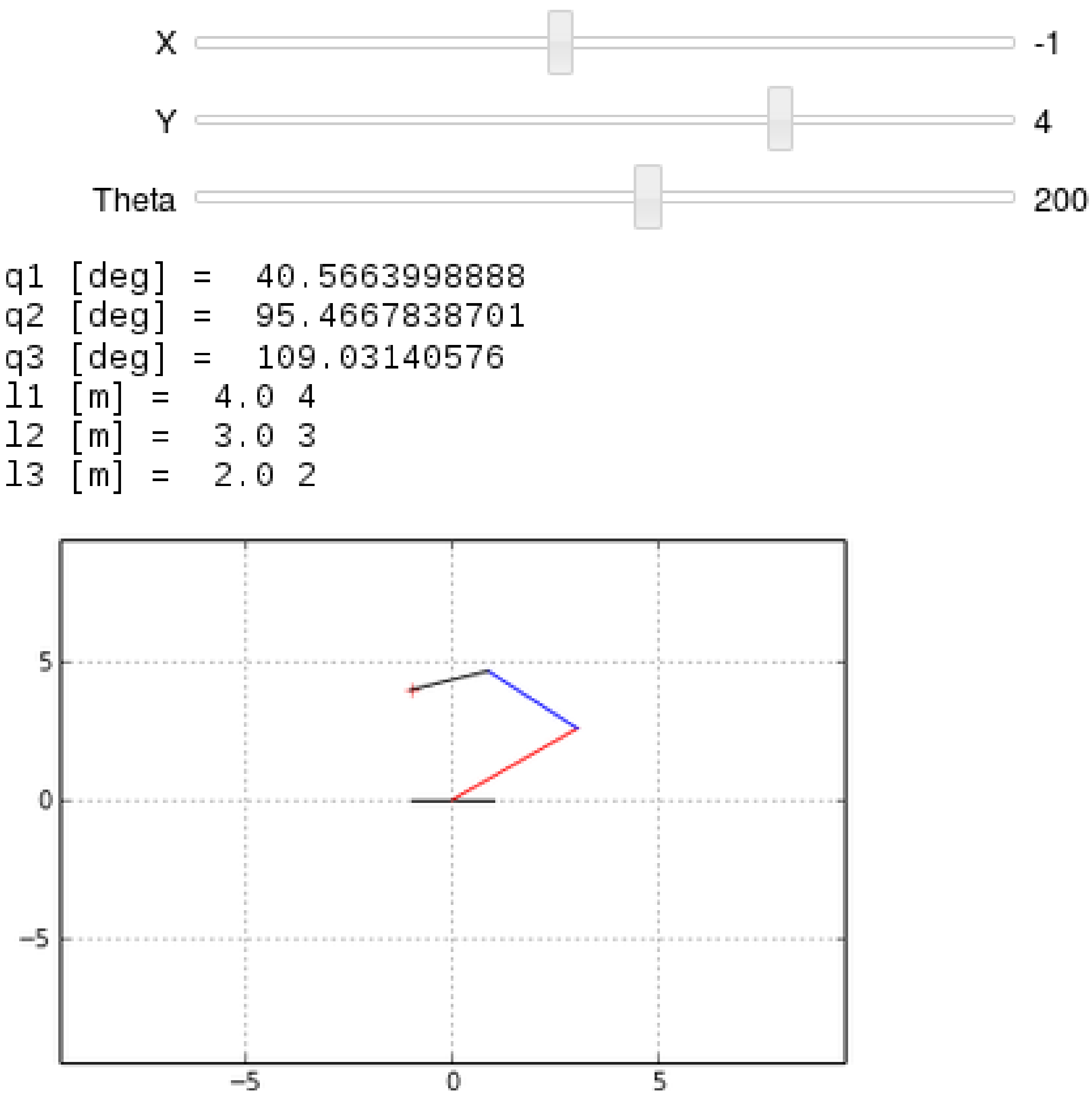

Figura 4: Captura de la ejecución del código en un Notebook de IPython