|  |  |  |  |
| --- | --- | --- | --- |
| Imagen relacionada | **UNIVERSIDAD AUTÓNOMA CHAPINGO** | | |
|  | **DEPARTAMENTO DE INGENIERÍA MECÁNICA AGRÍCOLA**  **(DIMA)** | |  |
|  | | |
| **Dinámica y control de robots** | |  |
| INFORME 1:  CINEMÁTICA DIRECTA E INVERSA | |  |
|  | |  |
| INGENIERÍA MECATRÓNICA AGRÍCOLA | |  |
| ELABORADO POR:  **SANCHEZ RODRIGUEZ LUIS ANGEL** | |  |
| 7°7  PROFESOR:  Luis Arturo Soriano Avendaño | |  |
| Fecha de entrega: 13 de octubre de 2021 |  | |

Índice

[Introducción 2](#_Toc85191432)

[Desarrollo 3](#_Toc85191433)

[Descripción de todas las ecuaciones para la obtención de la cinemática directa 3](#_Toc85191434)

[Ecuaciones 17](#_Toc85191435)

[Obtención de todas las ecuaciones de la cinemática inversa 17](#_Toc85191436)

[Programa para la cinemática directa e inversa de cada uno de los robots 17](#_Toc85191437)

[Elegir algún robot y aplicar otro método diferente a los vistos en clase 17](#_Toc85191438)

[Conclusión 17](#_Toc85191439)

[Fuentes bibliográficas 17](#_Toc85191440)

# Introducción

La cinemática está definida como el estudio del movimiento sin considerar las fuerzas que intervienen en este (Norton, 2016).

En el problema cinemático directo se conocen los valores de sus coordenadas articulares y busca determinar la posición del efector final. En este caso la solución del problema es única. En general, un robot de n grados de libertad está formado por n eslabones unidos a n articulaciones, a cada eslabón se le puede asociar un sistema de referencia solidario a él y utilizando las transformaciones homogéneas es posible representar las rotaciones y traslaciones relativas entre los distintos eslabones que componen el robot.

Para su solución, se utiliza el algoritmo de Denavit-Hartenberg a través del cual se obtendrán matrices de transformación homogénea para cada grado de libertad. Cada matriz de transformación tendrá la información relativa a la posición y orientación de sus ejes coordenados correspondientes, respecto a ejes anteriores o de referencia.

En el problema cinemático inverso se conoce la posición y orientación del efector final, pero se desconocen los ángulos de las articulaciones. Para ello, se requiere resolver un conjunto de ecuaciones para llegar a conocer estos valores. La obtención de la cinemática inversa depende fuertemente de la configuración del robot.

Los métodos usados para la solución del problema de la cinemática inversa son:

* Métodos geométricos: se basa en encontrar suficientes relaciones geométricas en las que intervendrán las coordenadas del efector final del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos.
* Resolución a partir de las matrices de transformación homogénea: se obtiene la cinemática inversa a partir de la modelación de su cinemática directa.
* Desacoplamiento cinemático: consiste en la separación de orientación y posición.
* Reducción polinómica: transforma las ecuaciones obtenidas de manera algebraica o geométrica para que adopten una forma polinómica (Granja Oramas, 2014).

Este documento presenta la resolución de los ejercicios de la tarea 1 de la materia de dinámica y control de robots.

# Desarrollo

Se comenzará realizando la cinemática directa, posteriormente la cinemática inversa. Además, se añaden los programas correspondientes a cada ejercicio.

## Descripción de todas las ecuaciones para la obtención de la cinemática directa

Para la obtención de la cinemática directa, se hizo uso del método de Denavit Hartenberg, el cual consiste en los siguientes pasos:

1. Numerar los eslabones comenzando con 1 (primer eslabón móvil de la 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 sobre el eje de la articulación i+1.
5. Situar el origen del sistema de la base {} en cualquier punto del eje . Los ejes y se situarán de modo que formen un sistema dextrógiro con .
6. Para i de 1 a n-1, situar el origen del sistema {} (solidario al eslabón i) en la intersección del eje con la línea normal común a . Si ambos ejes se cortasen, se situarían en el punto de corte. Si fuesen paralelos {} se situaría en la articulación i+1.
7. Situar en la línea normal común a .
8. Situar de modo que forme un sistema dextrógiro con .
9. Situar el sistema {} en el extremo del robot de modo que en coincida con la dirección de sea normal a .
10. Obtener como el ángulo que hay que girar en torno a para que queden paralelos.
11. Obtener como la distancia, medida a lo largo de , que habría que desplazar {} para que quedasen alineados.
12. Obtener como la distancia medida a lo largo de (que ahora coincidiría con ) que habría que desplazar el nuevo {} para que su origen coincidiese con {}.
13. Obtener como el ángulo que habría que girar en torno a , para que el nuevo {} coincidiese totalmente con {}.
14. Obtener las matrices de transformación .
15. Obtener la matriz de transformación que relaciona el sistema de la base con el del extremo del robot .
16. La matriz define la orientación (submatriz de rotación) y posición (submatriz de traslación) del extremo referido a la base, en función de las n coordenadas articulares.

* Robot cilíndrico con dos articulaciones primaticas y dos rotacionales

Diagrama

Descripción generada automáticamente

Imagen que contiene Diagrama

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  | 0 | 0 |
| 2 | 0 |  |  | 0 |
| 3 | 0 |  | 0 | 0 |
| 4 |  |  | 0 | 0 |

* Robot cartesiano con tres articulaciones prismáticas y una rotacional

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  |  | 0 |
| 2 | 0 |  | 0 | 0 |
| 3 | 0 |  |  | 0 |
| 4 |  |  | 0 | 0 |

* Robot con dos articulaciones primaticas y 2 rotacionales

Imagen que contiene antena, objeto, tabla, aire

Descripción generada automáticamente

Imagen que contiene lego, juguete

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  |  | 90 |
| 2 | 0 |  |  | 0 |
| 3 |  |  | 0 | 0 |
| 4 |  |  | 0 | 0 |

* Robot esférico con tres articulaciones rotacionales

Imagen que contiene lego

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  |  | 90 |
| 2 |  |  |  |  |
| 3 |  |  | 0 | 0 |

* Robot articulado con tres juntas rotacionales

Imagen que contiene Diagrama

Descripción generada automáticamente

Imagen que contiene avión

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  |  | 0 |
| 2 |  |  |  |  |
| 3 |  |  |  | 0 |

* Robot con cinco articulaciones rotacionales

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Junta () |  |  |  |  |
| 1 |  |  |  | 0 |
| 2 |  |  |  |  |
| 3 |  |  |  | 0 |
| 4 |  |  |  | 0 |
| 5 |  | 0 |  | 0 |

## Ecuaciones

* Robot 1

Las transformaciones se han de realizar en el orden que sigue:

Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.

Tabla

Descripción generada automáticamente

Donde:

* Robot 2

Las transformaciones se han de realizar en el orden que sigue:

Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.

Imagen que contiene Tabla

Descripción generada automáticamente

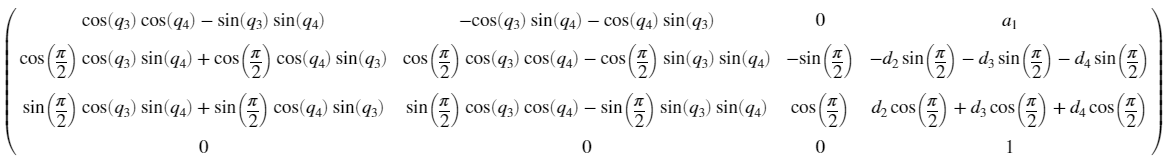
Donde:

* Robot 3

Las transformaciones se han de realizar en el orden que sigue:

Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.

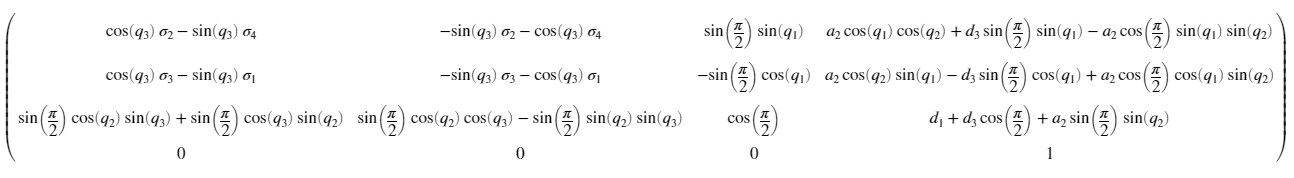


* Robot 4

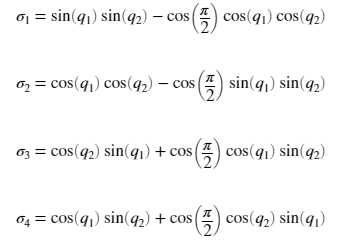
Las transformaciones se han de realizar en el orden que sigue:

Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.



Donde:

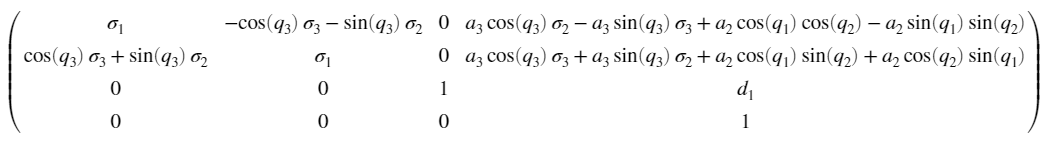


* Robot 5

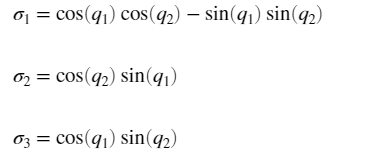
Las transformaciones se han de realizar en el orden que sigue:

Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.



Donde:

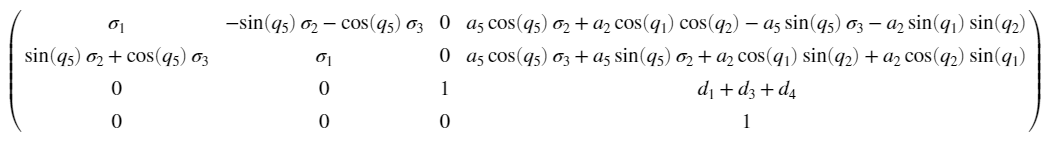


* Robot 6

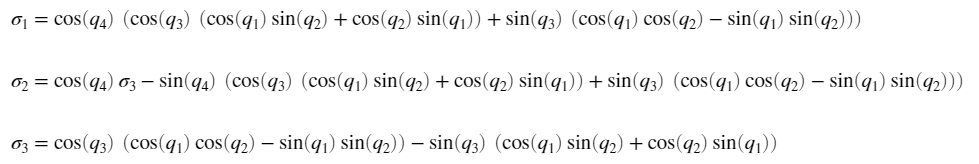
Las transformaciones se han de realizar en el orden que sigue:

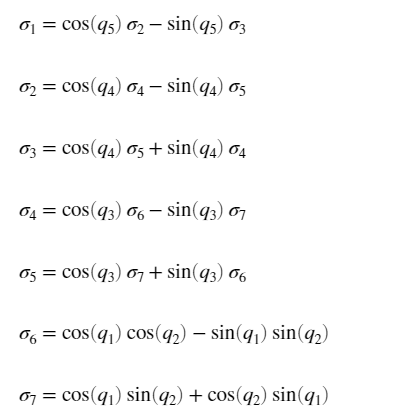
Al realizar la multiplicación de las matrices anteriores, se obtiene lo siguiente:

Al sustituir los valores y resolver la matriz obtenemos lo siguiente.



Donde:





## Obtención de todas las ecuaciones de la cinemática inversa

* Robot 1

Para

Para

Para

* Robot 2
* Robot 3

Incógnitas

Para el eslabón 3

* Robot 4

Eslabón 1

Eslabón 2

* Robot 5

## Programa para la cinemática directa de cada uno de los robots

* Robot 1

1 # UNIVERSIDAD AUTÓNOMA CHAPINGO

2 # DEPARTAMENTO DE INGENIERÍA MECÁNICA AGRÍCOLA

3 # INGENIERÍA MECATRÓNICA AGRÍCOLA

4 # DINÁMICA Y CONTROL DE ROBOTS

5 # TAREA 1 ROBOT 1

6 # LUIS ANGEL SANCHEZ RODRIGUEZ

7 # 7°7

8

9

10 # Importamos las librerías para el programa

11 import numpy as np #Importamos la librería numpy como np para vectores y matrices

12 import matplotlib.pyplot as plt # Importamos matplotlib como plt para graficar

13 from mpl\_toolkits import mplot3d # Importamos mplot3d desde mplt\_toolkits subpaquete de la librería de graficación para 3D

14 from matplotlib import cm # Importamos cm desde matplotlib subpaquete para la graficación

15 from matplotlib.widgets import Slider # Subpaquete para agregar las barras de control del robot

16

17

18 # Graficación en 3d

19 fig, ax = plt.subplots() # Creamos una figura y sus ejes

20 plt.subplots\_adjust(left = 0, bottom = 0.3, right =0.74, top = 1) # Ubicamos la figura

21 ax = plt.axes(projection = "3d") # Establecemos los ejes como 3 dimensiones

22

23

24 ###################### FUNCIONES DE MATRICES DE TRASLACIÓN #########################################

25

26 # X

27 def matriz\_traslacion\_x(x): # Definimos la matriz de traslación en x

28 traslacion\_x = np.array([[1,0,0,x], # Primera fila de la matriz de traslación

29 [0,1,0,0], # Segunda fila de la matriz de traslación

30 [0,0,1,0], # Tercera fila de la matriz de traslación

31 [0,0,0,1]]) # Cuarta fila de la matriz de traslación

32 return traslacion\_x # Devolvemos la matriz de traslación en x

33

34 # Y

35 def matriz\_traslacion\_y(y): # Definimos la matriz de traslación en y

36 traslacion\_y = np.array([[1,0,0,0], # Primera fila de la matriz de traslación

37 [0,1,0,y], # Segunda fila de la matriz de traslación

38 [0,0,1,0], # Tercera fila de la matriz de traslación

39 [0,0,0,1]]) # Cuarta fila de la matriz de traslación

40 return traslacion\_y # Devolvemos la matriz de traslación en y

41

42 # Z

43 def matriz\_traslacion\_z(z): # Definimos la matriz de traslación en z

44 traslacion\_z = np.array([[1,0,0,0], # Primera fila de la matriz de traslación

45 [0,1,0,0], # Segunda fila de la matriz de traslación

46 [0,0,1,z], # Tercera fila de la matriz de traslación

47 [0,0,0,1]]) # Cuarta fila de la matriz de traslación

48 return traslacion\_z # Devolvemos la matriz de traslación en z

49

50

51

52 ###################### FUNCIONES DE MATRICES DE ROTACIÓN #########################################

53

54 # X

55 def matriz\_rotacion\_x(grados\_x): #Definimos la función de rotación en x

56 rad = grados\_x/180\*np.pi # Conversión a grados

57 matriz\_rotacion\_x = np.array([[1, 0, 0,0], #Matriz de rotación primera fila

58 [0, np.cos(rad), -np.sin(rad),0], #Matriz de rotación segunda fila

59 [0,np.sin(rad),np.cos(rad),0], #Matriz de rotación terca fila

60 [0,0,0,1]]) # Cuarta fila de la matriz

61

62 return matriz\_rotacion\_x #Devuelvo la matriz de rotación en x

63

64 # Y

65 def matriz\_rotacion\_y(grados\_y): #Definimos la función de rotación en y

66 rad = grados\_y/180\*np.pi # Conversión a grados

67 matriz\_rotacion\_y = np.array([[np.cos(rad), 0, -np.sin(rad),0], #Matriz de rotación primera fila

68 [0, 1, 0,0], #Matriz de rotación segunda fila

69 [np.sin(rad), 0, np.cos(rad),0], #Matriz de rotación terca fila

70 [0,0,0,1]]) # Cuarta fila de la matriz

71

72 return matriz\_rotacion\_y #Devuelvo la matriz de rotación en y

73

74 # Z

75 def matriz\_rotacion\_z(grados\_z): #Definimos la función de rotación en z

76 rad = grados\_z/180\*np.pi # Conversión a grados

77 rotacion\_z = np.array([[np.cos(rad),-np.sin(rad),0,0], #Matriz de rotación primera fila

78 [np.sin(rad),np.cos(rad),0,0], #Matriz de rotación segunda fila

79 [0,0,1,0], #Matriz de rotación terca fila

80 [0,0,0,1]]) # Cuarta fila de la matriz

81 return rotacion\_z # devolvemos la matriz de rotación en z

82

83

84

85 # Configutación de la gráfica

86 def configuracion\_grafica(): # función de configuración de la gráfica

87 plt.title("Robot 4 G.D.L. RPPR", x=3, y=12) # Título de la gráfica

88 ax.set\_xlim(-20,20) # Límites en el eje x

89 ax.set\_ylim(-20,20) # Límites en el eje y

90 ax.set\_zlim(-20,10) # Límites en el eje z

91

92 ax.set\_xlabel("x(t)") # Nombre del eje x

93 ax.set\_ylabel("y(t)") # Nombre del eje y

94 ax.set\_zlabel("z(t)") # Nombre del eje z

95 ax.view\_init(elev = 25, azim = 30) # # Tipo de vista de la gráfica

96

97

98 def sistema\_coordenadas(a,b,c,a\_f,b\_f,c\_f):

99 x = [a,a\_f]

100 y = [b,b\_f]

101 z = [c,c\_f]

102

103 ax.plot3D(x,[b,b],[c,c],color="red") # X

104 ax.plot3D([a,a],y,[c,c],color="blue") # Y

105 ax.plot3D([a,a],[b,b],z,color="green") # Z

106

107

108

109 # Sistema de coordenadas móvil para la matriz de rotación

110 def sistema\_coordenadas\_movil(matriz\_rotacion): # definimos la matriz

111 r\_11 = matriz\_rotacion[0,0] # Columna 0, Fila 0

112 r\_12 = matriz\_rotacion[1,0] # Columna 1, Fila 0

113 r\_13 = matriz\_rotacion[2,0] # Columna 2, Fila 0

114 r\_21 = matriz\_rotacion[0,1] # Columna 0, Fila 1

115 r\_22 = matriz\_rotacion[1,1] # Columna 1, Fila 1

116 r\_23 = matriz\_rotacion[2,1] # Columna 2, Fila 1

117 r\_31 = matriz\_rotacion[0,2] # Columna 0, Fila 2

118 r\_32 = matriz\_rotacion[1,2] # Columna 1, Fila 2

119 r\_33 = matriz\_rotacion[2,2] # Columna 2, Fila 2

120

121 dx = matriz\_rotacion[0,3] # Columna 0, Fila 3

122 dy = matriz\_rotacion[1,3] # Columna 1, Fila 3

123 dz = matriz\_rotacion[2,3] # Columna 2, Fila 3

124

125 # Sistema que va a mover

126 ax.plot3D([dx,dx+r\_11],[dy,dy+r\_12],[dz,dz+r\_13], color = "m") # X

127 ax.plot3D([dx,dx+r\_21],[dy,dy+r\_22],[dz,dz+r\_23], color = "c") # Y

128 ax.plot3D([dx,dx+ r\_31],[dy,dy+r\_32],[dz,dz+r\_33], color = "k") # Z

129

130

131 # Función Denavit-Hartenberg

132 def denavit\_hartenberg(theta\_i,d\_i,a\_i,alpha\_i):

133 MT = matriz\_rotacion\_z(theta\_i)@matriz\_traslacion\_z(d\_i)@matriz\_traslacion\_x(a\_i)@matriz\_rotacion\_x(alpha\_i)

134 return MT

135

136

137 # Función del robot

138 def robot\_RPPR(theta\_1,d\_1,a\_1,alpha\_1,theta\_2,d\_2,a\_2,alpha\_2,theta\_3,d\_3,a\_3,alpha\_3,theta\_4,d\_4,a\_4,alpha\_4):

139

140 A0 = np.eye(4)

141 A\_0\_1 = denavit\_hartenberg(theta\_1,d\_1,a\_1,alpha\_1)

142 A\_1\_2 = denavit\_hartenberg(theta\_2,d\_2,a\_2,alpha\_2)

143 A\_2\_3 = denavit\_hartenberg(theta\_3,d\_3,a\_3,alpha\_3)

144 A\_3\_4 = denavit\_hartenberg(theta\_4,d\_4,a\_4,alpha\_4)

145

146 A\_0\_2 = A\_0\_1@A\_1\_2

147 A\_0\_3 = A\_0\_2@A\_2\_3

148 A\_0\_4 = A\_0\_3@A\_3\_4

149

150 sistema\_coordenadas\_movil(A0)

151 sistema\_coordenadas\_movil(A\_0\_1)

152 sistema\_coordenadas\_movil(A\_0\_2)

153 sistema\_coordenadas\_movil(A\_0\_3)

154 sistema\_coordenadas\_movil(A\_0\_4)

155

156 # Eslabones dibujados

157 ax.plot3D([A0[0,3],A\_0\_1[0,3]],[A0[1,3],A\_0\_1[1,3]],[A0[2,3],A\_0\_1[2,3]], color="red")

158 ax.plot3D([A\_0\_1[0,3],A\_0\_2[0,3]],[A\_0\_1[1,3],A\_0\_2[1,3]],[A\_0\_1[2,3],A\_0\_2[2,3]], color="blue")

159 ax.plot3D([A\_0\_2[0,3],A\_0\_3[0,3]],[A\_0\_2[1,3],A\_0\_3[1,3]],[A\_0\_2[2,3],A\_0\_3[2,3]], color="green")

160 ax.plot3D([A\_0\_3[0,3],A\_0\_4[0,3]],[A\_0\_3[1,3],A\_0\_4[1,3]],[A\_0\_3[2,3],A\_0\_4[2,3]], color="red")

161 return A\_0\_4

162

163

164

165 # Función de ajuste de juntas con sliders

166 def actualizacion\_juntas(val):

167 ax.cla() # Limpiamos la gráfica

168 configuracion\_grafica()

169

170 # Los valores de las sliders serán los valores de los ángulos

171 theta\_1 = sld\_ang\_1.val

172 d\_2 = sld\_d\_2.val

173 a\_3 = sld\_d\_3.val

174 theta\_4 = sld\_ang\_4.val

175

176 #Parámetros del robot

177 robot\_RPPR(theta\_1,1,0,0, # theta\_1, d\_1, a\_1, alpha\_1

178 0,d\_2,0,0, # theta\_2, d\_2, a\_2, alpha\_2

179 0,0,a\_3,90, # theta\_3, d\_3, a\_3, alpha\_3

180 theta\_4,4,0,0) # theta\_4, d\_4, a\_4, alpha\_4

181 plt.draw()

182 plt.pause(1e-3)

183

184

185 # Definimos los sliders

186 ax1 = plt.axes([0.2,0.2,0.65,0.03])

187 ax2 = plt.axes([0.2,0.15,0.65,0.03])

188 ax3 = plt.axes([0.2,0.10,0.65,0.03])

189 ax4 = plt.axes([0.2,0.05,0.65,0.03])

190

191

192 # Inicio de las sliders

193 sld\_ang\_1 = Slider(ax1, r"$\theta\_1$", -180, 180, valinit = 90)

194 sld\_d\_2 = Slider(ax2, "d\_2", 1, 10, valinit = 10)

195 sld\_d\_3 = Slider(ax3, "a\_3", 1, 10, valinit = 8)

196 sld\_ang\_4 = Slider(ax4, r"$\theta\_4$", -180, 180, valinit = 45)

197

198

199 # Así aparece el robot primero

200 robot\_RPPR(90,1,0,0, # theta\_1, d\_1, a\_1, alpha\_1

201 0,10,0,0, # theta\_2, d\_2, a\_2, alpha\_2

202 0,0,8,90, # theta\_3, d\_3, a\_3, alpha\_3

203 45,4,0,0) # theta\_4, d\_4, a\_4, alpha\_4

204

205 configuracion\_grafica()

206

207 #Se actualizan las sliders

208 sld\_ang\_1.on\_changed(actualizacion\_juntas)

209 sld\_d\_2.on\_changed(actualizacion\_juntas)

210 sld\_d\_3.on\_changed(actualizacion\_juntas)

211 sld\_ang\_4.on\_changed(actualizacion\_juntas)

212

213 plt.show() # Mostramos la gráfica

214

* Robot 2
* Robot 3
* Robot 4
* Robot 5
* Robot 6

# Conclusión

Después de haber realizado la investigación de esta primera unidad

# Fuentes bibliográficas

1. Granja Oramas, M. V. (2014). *Modelación y análisis de la cinemática directa e inversa del manipulador Stanford de seis grados de libertad* (Bachelor's thesis, Quito, 2014.).
2. Norton, R. L. (2016). *Diseño de maquinaria*.