MANUAL DE ESTUDIO · SISTEMAS ROBÓTICOS

Cinemática de manipuladores: Denavit–Hartenberg y jacobiano

Capítulo 04 · En el video: **minuto 45:56** · Duración 15:09

Objetivo. Calcular la pose del TCP a partir de los ángulos articulares, resolver el problema inverso y reconocer una singularidad.

Ideas clave

Pose y rotaciones

La pose del TCP son seis números: posición (x, y, z) y orientación. La orientación se guarda en una matriz de rotación R de 3×3: sus columnas son vectores unitarios perpendiculares, $R^{T} \cdot R = I$ y det R = 1, así que la inversa es la transpuesta.

Transformaciones homogéneas

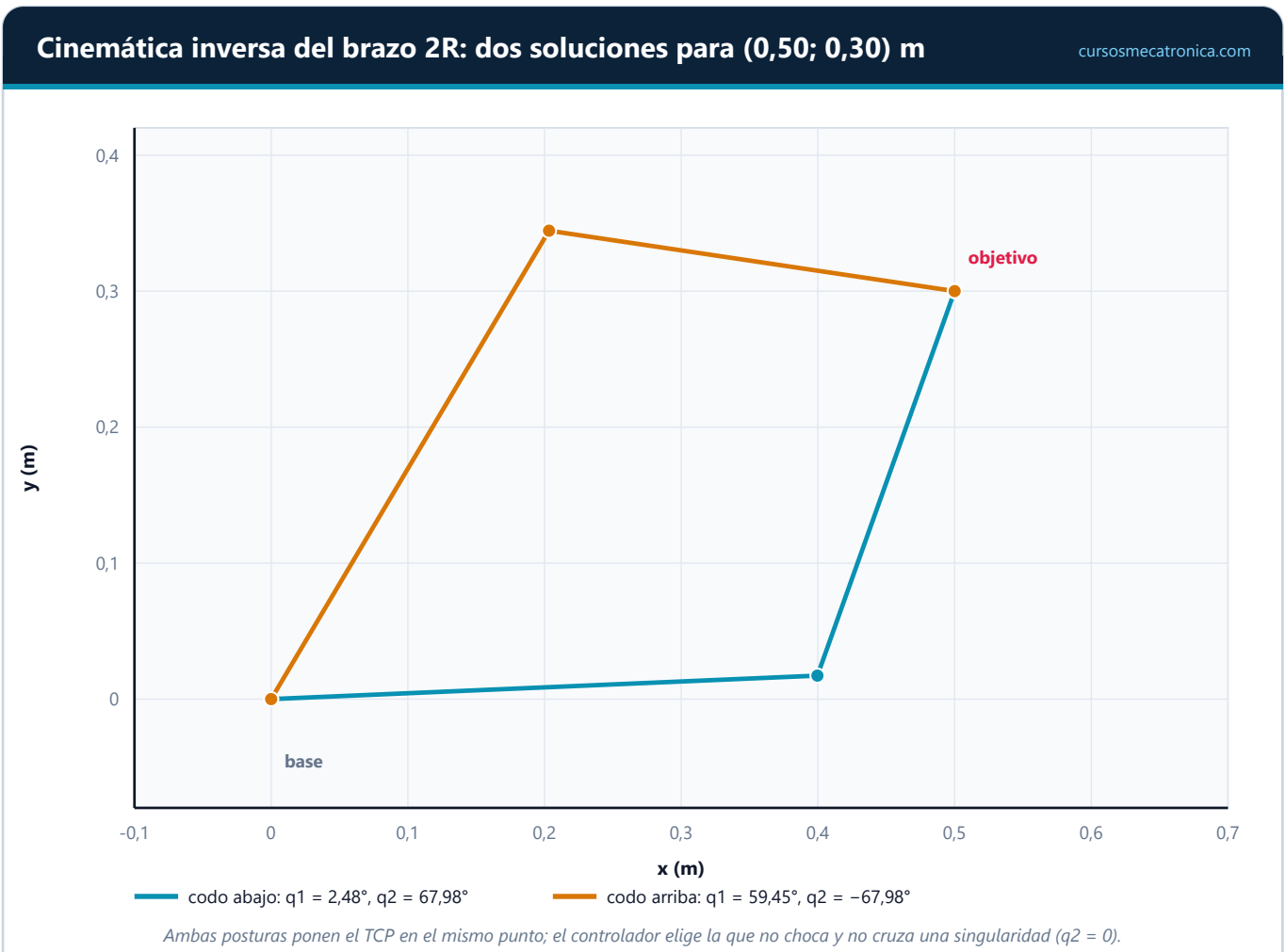
Una matriz 4×4 junta rotación y traslación: $T = [R\ p;\ 0\ 1]$. Encadenar marcos es multiplicar matrices: ${}^{0}T_{n} = {}^{0}T_{1} \cdot {}^{1}T_{2} \cdots$.

Parámetros de Denavit–Hartenberg

Cada eslabón se describe con cuatro números: θ (giro alrededor de z), d (desplazamiento a lo largo de z), a (largo de la normal común) y α (torsión entre ejes). En una junta rotacional θ es la variable; en una prismática, d.

Cinemática directa e inversa

La directa multiplica las matrices A_i y siempre tiene una solución. La inversa busca los ángulos para una pose dada: puede tener varias (codo arriba y codo abajo) o ninguna si el punto está fuera del alcance. Con muñeca esférica, el criterio de Pieper permite resolver primero la posición del centro de la muñeca y luego la orientación (ángulos de Euler ZYZ).

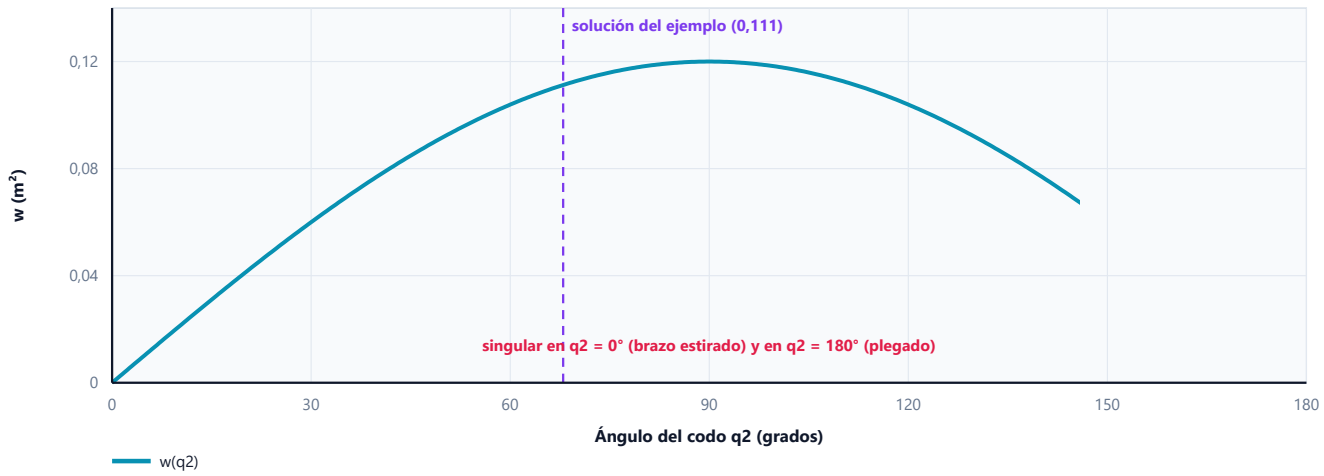


Salida de cinematica_dh.py.

Jacobiano y singularidades

El jacobiano relaciona velocidades: $v = J(q)\cdot\dot{q}$; su transpuesta lleva fuerzas del TCP a pares articulares ($\tau = J^{T}\cdot F$). Si det J = 0 el robot está en una singularidad: pierde una dirección de movimiento. El índice de Yoshikawa $w = \sqrt{\det(J\cdot J^{T})}$ mide cuán lejos está.

Manipulabilidad del brazo 2R: w = l1·l2·|sen q2| cursosmecatronica.com



Con w = 0 el jacobiano no tiene inversa: el TCP no puede moverse en alguna dirección y las velocidades articulares se disparan.

Índice de Yoshikawa del capítulo 04.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Matriz DH	$A_i = Rot_z(\theta)\cdot Tras_z(d)\cdot Tras_x(a)\cdot Rot_x(\alpha)$	Convención estándar.
Directa 2R	$x = l_1c_1 + l_2c_{12},\ y = l_1s_1 + l_2s_{12}$	$c_{12} = \cos(q_1+q_2)$.
Inversa 2R	$c_2 = (x^2 + y^2 - {l_1}^2 - {l_2}^2)/(2l_1l_2)$	Ley de cosenos; $s_2 = \pm\sqrt{(1 - {c_2}^2)}$.
Hombro	$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2s_2, l_1 + l_2c_2)$	Usar atan2, no atan.
Jacobiano 2R	$\det J = l_1\cdot l_2\cdot \text{sen}\ q_2$	Singular con el brazo estirado o plegado.

Ejemplo resuelto: inversa del brazo 2R hacia (0,50; 0,30) m

1. l_1 = 0,40 m, l_2 = 0,30 m; $x^2 + y^2$ = 0,34.
2. $c_2 = (0{,}34 - 0{,}16 - 0{,}09)/0{,}24 = 0{,}375 \rightarrow q_2 = \pm 67{,}98°$.
3. Codo abajo: $q_1 = 30{,}96° - 28{,}48° = 2{,}48°$. Codo arriba: q_1 = 59,45°.
4. Comprobación con la directa: ambas dan (0,5000; 0,3000). det J = ±0,1112 m².

Dos posturas válidas, lejos de la singularidad (w máx = 0,12 m² con q_2 = 90°). cinematica_dh.py hace el cálculo con matrices DH.

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

cinematica_dh.py — matrices DH, cinemática directa e inversa 2R, jacobiano, manipulabilidad y velocidad del TCP. *(extracto: primeras 60 de 70 líneas)*

```
"""
cinematica_dh.py — Capítulo 04: transformaciones homogéneas, Denavit-Hartenberg,
```

```
cinemática directa e inversa de un brazo plano 2R, jacobiano y manipulabilidad.
Sin NumPy: matrices como listas para ver cada operación.
"""
import math


def matmul(a, b):
    return [[sum(a[i][k] * b[k][j] for k in range(4)) for j in range(4)] for i in range(4)]


def dh(theta, d, a, alfa):
    """A_i = Rot_z(θ)·Trans_z(d)·Trans_x(a)·Rot_x(α)  (convención estándar)."""
    ct, st, ca, sa = math.cos(theta), math.sin(theta), math.cos(alfa), math.sin(alfa)
    return [[ct, -st * ca, st * sa, a * ct],
            [st, ct * ca, -ct * sa, a * st],
            [0.0, sa, ca, d],
            [0.0, 0.0, 0.0, 1.0]]


def directa(tabla, q):
    """tabla: filas (d, a, alfa) de juntas rotacionales; q: ángulos en rad."""
    t = [[1.0 if i == j else 0.0 for j in range(4)] for i in range(4)]
    for (d, a, alfa), qi in zip(tabla, q):
        t = matmul(t, dh(qi, d, a, alfa))
    return t


def inversa_2r(x, y, l1, l2, codo_arriba=False):
    c2 = (x * x + y * y - l1 * l1 - l2 * l2) / (2 * l1 * l2)
    if abs(c2) > 1:
        raise ValueError('punto fuera del alcance')
    s2 = math.sqrt(1 - c2 * c2) * (-1 if codo_arriba else 1)
    q2 = math.atan2(s2, c2)
    q1 = math.atan2(y, x) - math.atan2(l2 * s2, l1 + l2 * c2)
    return q1, q2


def jacobiano_2r(q1, q2, l1, l2):
    s1, c1 = math.sin(q1), math.cos(q1)
    s12, c12 = math.sin(q1 + q2), math.cos(q1 + q2)
    return [[-l1 * s1 - l2 * s12, -l2 * s12],
            [l1 * c1 + l2 * c12, l2 * c12]]


if __name__ == '__main__':
    l1, l2 = 0.40, 0.30
    tabla = [(0.0, l1, 0.0), (0.0, l2, 0.0)]
    x, y = 0.50, 0.30
    print(f'Objetivo ({x:.2f}, {y:.2f}) m con l1 = {l1} m y l2 = {l2} m')
    for arriba in (False, True):
        q1, q2 = inversa_2r(x, y, l1, l2, arriba)
        t = directa(tabla, (q1, q2))
        j = jacobiano_2r(q1, q2, l1, l2)
        det = j[0][0] * j[1][1] - j[0][1] * j[1][0]
        nombre = 'codo arriba' if arriba else 'codo abajo '
        print(f'  {nombre}: q1 = {math.degrees(q1):7.2f}°, q2 = {math.degrees(q2):7.2f}°'
              f' -> directa ({t[0][3]:.4f}, {t[1][3]:.4f})  det J = {det:+.4f}')
```

Errores frecuentes

- Usar atan en vez de atan2 y caer en el cuadrante equivocado.
- Mezclar grados y radianes en las funciones trigonométricas.
- Multiplicar las matrices en el orden inverso (el producto no es conmutativo).

- Pedir un punto fuera del alcance: $|c_2| > 1$.

Autoevaluación

1. ¿Cuál es la inversa de una matriz de rotación?
2. ¿Qué parámetro DH es variable en una junta rotacional?
3. ¿Cuándo es singular el brazo 2R?

Respuestas. 1. Su transpuesta. **2.** θ. **3.** Cuando sen $q_2 = 0$: estirado o plegado.