Universidad del Valle de Guatemala
Departamento de Ingeniería Mecatrónica
MT3005 - Robótica 1
MSc. Miguel Zea

Laboratorio 7

Cinemática inversa de manipuladores seriales

Objetivos

- Implementar un algoritmo para resolver numéricamente el problema de cinemática inversa para el manipulador myCobot 280.
- Evaluar la diferencia en rendimiento entre los métodos iterativos para cinemática inversa basados en la transpuesta del jacobiano, la pseudoinversa y el algoritmo de Levenberg–Marquardt.

Procedimiento

En la práctica de esta semana usted tendrá como tarea desarrollar e implementar un algoritmo en MATLAB que permita encontrar la solución al problema de cinemática inversa mediante diversos métodos numéricos iterativos. A modo de ayuda, a continuación se le presenta una derivación alternativa del algoritmo de cinemática inversa, la cual puede emplear junto con la de las notas de clase durante la implementación.

Los métodos de cinemática inversa se basan en el hecho que el jacobiano relaciona un cambio en las coordenadas del efector final con un cambio en la configuración, de la forma

$$\begin{bmatrix} {}^B\mathbf{v}_E \\ {}^B\boldsymbol{\omega}_E \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}.$$

Es posible descomponer esta relación en una parte traslacional y una rotacional como

$$\begin{bmatrix} {}^B\mathbf{v}_E \\ {}^B\boldsymbol{\omega}_E \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{J}_v(\mathbf{q}) \\ \mathbf{J}_\omega(\mathbf{q}) \end{bmatrix}}_{\mathbf{J}(\mathbf{q})} \dot{\mathbf{q}} \qquad \Rightarrow \begin{cases} {}^B\mathbf{v}_E = \mathbf{J}_v(\mathbf{q})\dot{\mathbf{q}} \\ {}^B\boldsymbol{\omega}_E = \mathbf{J}_\omega(\mathbf{q})\dot{\mathbf{q}} \end{cases}$$

Como ejemplo, se hará la derivación de la cinemática inversa de posición, por lo que sólo será de interés la posición del efector final, simplificando la cinemática diferencial a la forma

$${}^B\mathbf{v}_E = \mathbf{J}_v(\mathbf{q})\dot{\mathbf{q}} \qquad \Rightarrow \frac{d^B\mathbf{o}_E(\mathbf{q})}{dt} = \mathbf{J}_v(\mathbf{q})\frac{d\mathbf{q}}{dt},$$

donde el jacobiano $\mathbf{J}_v(\mathbf{q})$ está conformado por las primeras tres filas del jacobiano completo $\mathbf{J}(\mathbf{q})$ (**NOTA:** de aquí en adelante se referirá a ${}^B\mathbf{o}_E(\mathbf{q})$ y a ${}^B\mathbf{v}_E$ como $\mathbf{o}$ y $\mathbf{v}$ para simplificar la notación). De forma diferencial, la relación descrita por el jacobiano implica que

$$\lim_{\Delta t\to 0}\frac{\Delta\mathbf{o}}{\Delta t} = \mathbf{J}_v(\mathbf{q})\lim_{\Delta t\to 0}\frac{\Delta\mathbf{q}}{\Delta t} \qquad \Rightarrow \Delta\mathbf{o} = \mathbf{J}_v(\mathbf{q})\Delta\mathbf{q},$$

es decir, relaciona un cambio en la configuración con un cambio de posición en el efector final. Suponiendo que el jacobiano es invertible, puede obtenerse la relación inversa

$$\Delta\mathbf{q} = \mathbf{J}_v^{-1}(\mathbf{q})\Delta\mathbf{o},$$

la cual corresponde al *método de Newton* y puede implementarse mediante un método iterativo para resolver el problema de la cinemática inversa.

Suponga que el manipulador se encuentra en una configuración inicial $\mathbf{q}_0$ tal que el efector final se encuentra en la posición $\mathbf{o}_0$. En el problema de la cinemática inversa, se busca determinar la configuración $\mathbf{q}$ que lleva al efector final a una posición deseada $\mathbf{o}_d$. Nótese entonces que $\Delta\mathbf{o} = \mathbf{o}_d - \mathbf{o}_0$ efectivamente representa al *error* entre la posición actual y la deseada en el efector final, del cual podríamos encontrar el cambio requerido en $\mathbf{q}$ mediante $\Delta\mathbf{q} = \mathbf{J}_v^{-1}(\mathbf{q})\Delta\mathbf{o}$ y sabiendo que $\Delta\mathbf{q} = \mathbf{q} - \mathbf{q}_0$. Sin embargo, la relación descrita por el jacobiano se cumple sólo para el caso diferencial, por lo que tanto $\Delta\mathbf{o}$ como $\Delta\mathbf{q}$ deben ser pequeños. La idea del método iterativo es entonces no hacer un único salto de $\mathbf{o}_0$ a $\mathbf{o}_d$ sino más bien acercarse con pasos pequeños hasta llegar lo suficientemente cerca (en términos de error) a la posición deseada. Se presenta el pseudo-código del algoritmo a continuación:

- **Inicialización:** Dada una posición del efector final deseada $\mathbf{o}_d \in \mathbb{R}^3$ y una configuración inicial $\mathbf{q}_0 \in \mathbb{R}^n$, se establece $k = 0$.
- **Mientras** que $\|\mathbf{e}_k\| > \epsilon$ (para ϵ pequeño), se repite:
  - Se calcula el error actual $\mathbf{e}_k = \mathbf{o}_d - \mathbf{o}(\mathbf{q}_k)$, donde $\mathbf{o}(\mathbf{q}_k)$ corresponde a la posición actual del efector final.
  - Se actualiza el valor actual de la configuración en la dirección del jacobiano mediante $\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{J}_v^{-1}(\mathbf{q}_k)\mathbf{e}_k$.
  - Se incrementa k.

Evidentemente, el algoritmo presentado funciona únicamente si el jacobiano es invertible lo cual en general es raro. Esta limitación puede superarse al sustituir la inversa del jacobiano por expresiones *similares* que sí puedan computarse sin depender de las dimensiones del jacobiano. En esta práctica se considerarán 3 distintos métodos que resuelven el problema de optimización que no dependen si el jacobiano es invertible o no. Estos métodos emplean (aplica tanto para el jacobiano de velocidad lineal $\mathbf{J}_v$ como para el de velocidad angular $\mathbf{J}_\omega$ y, por ende, el completo $\mathbf{J}$ también):

- **La pseudo-inversa del jacobiano:** Se reemplaza $\mathbf{J}^{-1}$ por la pseudo-inversa de Moore-Penrose $\mathbf{J}^\dagger$. La pseudo-inversa corresponde a $\mathbf{J}^\dagger = \mathbf{J}^\top \left(\mathbf{J}\mathbf{J}^\top\right)^{-1}$ si el jacobiano es "gordo" (tiene más columnas que filas), y a $\mathbf{J}^\dagger = \left(\mathbf{J}^\top\mathbf{J}\right)^{-1} J^\top$ si el jacobiano es "delgado" (tiene más filas que columnas). Esta inversa es computable siempre y cuando el jacobiano no presente una *singularidad*.
- **Damped Least-Squares (o el algoritmo de Levenberg–Marquardt):** Se reemplaza $\mathbf{J}^{-1}$ por $\mathbf{J}^\top \left(\mathbf{J}\mathbf{J}^\top + \lambda^2\mathbf{I}\right)^{-1}$, con $\lambda > 0$ y pequeño (por ejemplo $\lambda^2 = 0.1$).
- **La traspuesta del jacobiano:** Se reemplaza $\mathbf{J}^{-1}$ por $\alpha\mathbf{J}^\top$, donde una selección popular para α es $\alpha = \dfrac{\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e}}{\mathbf{e}^\top\mathbf{J}\mathbf{J}^\top\mathbf{J}\mathbf{J}^\top\mathbf{e}}$.

θ_j	d_j	a_j	α_j
q_1	131.22	0	$\pi/2$
$q_2 - \pi/2$	0	−110.4	0
q_3	0	−96	0
$q_4 - \pi/2$	63.4	0	$\pi/2$
$q_5 + \pi/2$	75.05	0	$-\pi/2$
q_6	45.6	0	0

Figura 1: Manipulador serial myCobot 280 de Elephant Robotics.

Puede repetirse el mismo proceso para la cinemática inversa de orientación, sólo cambiando el jacobiano al de velocidad angular (para la cinemática completa se cambia al jacobiano completo) y definiendo el error de orientación como $\mathbf{e}_o = \boldsymbol{\epsilon}_e$, en donde $\mathcal{Q}_e = \mathcal{Q}_d * \mathcal{Q}^{-1}(\mathbf{q}_k) = \{\eta_e[k], \boldsymbol{\epsilon}_e[k]\}$, $\mathcal{Q}_d \sim \mathbf{R}_d$ y $\mathcal{Q}(\mathbf{q}_k) \sim {}^B\mathbf{R}_E(\mathbf{q}_k)$.

Como caso particular, usted desarrollará un algoritmo para resolver la cinemática inversa del manipulador myCobot 280 con el que se comenzó a trabajar en la práctica pasada, y que se describe nuevamente en la Figura 1. Para esto, realice lo siguiente:

1. Descargue de Canvas el archivo `mt3005lab7.zip` y extraiga sus contenidos dentro de una carpeta en una ubicación de su preferencia. Cambie el folder actual de MATLAB para que coincida con esta carpeta. Dentro de este archivo se encuentran las funciones auxiliares:

   - `robot_def`, `robot_fkine` y `robot_jacobian` resultado del Laboratorio 6 y encargadas de calcular la matriz de parámetros DH, la cinemática directa y la cinemática diferencial del robot dado respectivamente.
   - `rot2cuat`, `cuat2rot`, `multcuat` e `invcuat`, las cuales permiten pasar de matriz de rotación a cuaternión unitario (y viceversa), multiplicar cuaterniones e invertir cuaterniones respectivamente (estas serán de utilidad para la cinemática inversa de orientación).

2. Implemente **todas** las variantes de cinemática inversa en la función `robot_ikine`. Note que esta función ya tiene implementada la validación de argumentos y la definición de todas las posibles combinaciones de los mismos. Tome nota de los comentarios en la función misma para evitar confusión en su uso.

3. Experimente con los métodos desarrollados empleando diversas configuraciones de prueba, poses de efector final deseadas y visualizando el histórico de la solución. Responda a las

siguientes preguntas (dentro del código del script `laboratorio7.m` se le detalla cómo hacerlo):

- Sin tomar en cuenta configuraciones singulares, en general, ¿Cuál de los 3 métodos numéricos converge a la solución de forma más rápida?
- ¿Cuál método presenta la convergencia más suave hacia la solución?
- De todas las posibles combinaciones de tipo de cinemática inversa y método numérico, hay una que no funciona según lo esperado, ¿Cuál es esta combinación?

Para verificar si sus soluciones están correctas, corra la sentencia `calificar('laboratorio7')` en la línea de comando. Cuando esté satisfecho con los resultados, preséntenselos al profesor del laboratorio o al auxiliar. Recuerde que entregas tardías representan una penalización del 25 % por semana.