Universidad AUTÓNOMA de Occidente

Manipulator Inverse kinematics I

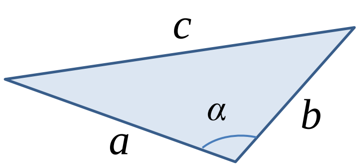
Agenda

1. **- Introducción**
2. - Modelo geométrico
3. - Matrices de transformación homogénea (MTH)
4. - Método algebraico
5. - Desacoplo cinemático
6. Cálculo Numérico del Jacobiano

Soluciones Geométricas

- Tratan de encontrar los valores articulares $q_1, q_2, \ldots, q_n$ analizando la geometría (en el plano o espacio) del robot
  - No es una solución genérica → depende del robot

- Pasos útiles (depende del caso):
  - Proyectar en el plano (y analizar en el plano)
  - Realizar "trazos" para encontrar triángulos rectángulos (y obtener tangentes)
  - Utilizar la regla del coseno:



$$c^2 = a^2 + b^2 - 2ab\cos\alpha$$

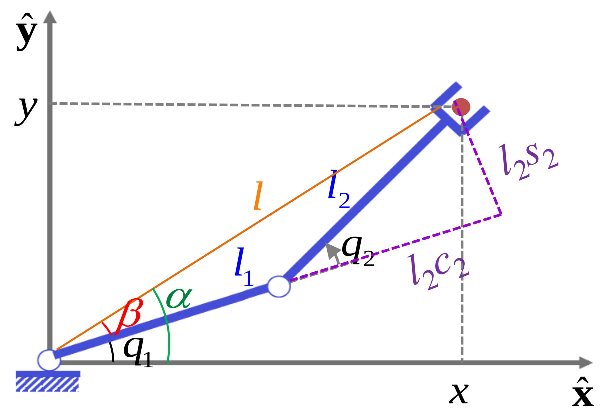
- Aplicable a robots con pocos *gdl* (3 o menos) … ¿por qué?
- En robots de más *gdl* puede ser aplicable a los 3 primeros *gdl* (si los demás gdl solo se usan para orientación)

Soluciones Geométricas

- **Ejemplo:**

Encontrar la cinemática inversa para la posición del robot RR usando el enfoque geométrico

Solución



Para q_2:

- Usando la ley de cosenos:

$$l^2 = l_1^2 + l_2^2 - 2l_1 l_2 \cos(180 - q_2)$$

$$x^2 + y^2 = l_1^2 + l_2^2 + 2l_1 l_2 \cos(q_2)$$

$$c_2 = \frac{x^2 + y^2 - (l_1^2 + l_2^2)}{2l_1 l_2}$$

- Usando una identidad trigonométrica:

$$s_2^2 + c_2^2 = 1$$

$$s_2 = \pm\sqrt{1 - c_2^2}$$

$$q_2 = \text{atan2}(s_2, c_2)$$

Para q_1 (usando la geometría de la figura)

$$q_1 = \alpha - \beta$$

$$\alpha = \text{atan2}(y, x)$$

$$\beta = \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$

Cinemática inversa:

$$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2 s_2, l_1 + l_2 c_2)$$

$$q_2 = \text{atan2}(s_2, c_2)$$

Soluciones Algebraicas

- Tratan de encontrar los valores articulares $q_1, q_2, \ldots, q_n$ manipulando algebraicamente las ecuaciones de la cinemática directa
  - Aplicable a robots con pocos grados de libertad

- Elementos útiles (depende del caso):
  - Elevar al cuadrado para encontrar la identidad trigonométrica (permite eliminar algún valor articular):
$$\sin^2\theta + \cos^2\theta = 1$$
  - Expandir expresiones del ángulo doble (a veces se pueden simplificar con otros sumandos)
$$\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \sin\beta\cos\alpha$$
$$\cos(\alpha \pm \beta) = \cos\alpha\cos\beta \mp \sin\alpha\sin\beta$$
  - Encontrar el seno y coseno de forma matricial y reemplazarlo en atan2

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} \cos q_i \\ \sin q_i \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix} \quad \Rightarrow \quad q_i = \operatorname{atan2}(\sin q_i, \cos q_i)$$

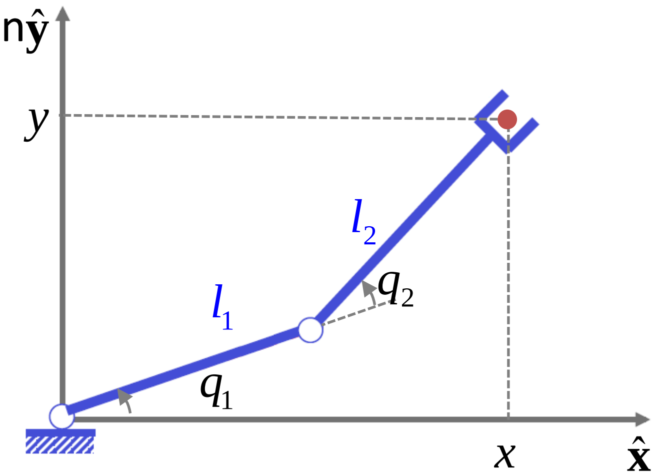
Soluciones Algebraicas

- **Ejemplo 1**

Solución

Encontrar la cinemática inversa para la posición de un

algebraico



- Cinemática Directa:

$$x = l_1\,c_1 + l_2\,c_{12} \qquad\qquad y = l_1\,s_1 + l_2\,s_{12}$$

- Para q_2 :

De C.D.
$$\begin{cases} x^2 = l_1^{\,2}\,c_1^{\,2} + l_2^{\,2}\,c_{12}^{\,2} + 2l_1c_1l_2c_{12} \\ y^2 = l_1^{\,2}\,s_1^{\,2} + l_2^{\,2}\,s_{12}^{\,2} + 2l_1s_1l_2s_{12} \end{cases}$$

$$x^2 + y^2 = l_1^{\,2} + l_2^{\,2} + 2l_1l_2(c_1c_{12} + s_1s_{12})$$

$$x^2 + y^2 = l_1^{\,2} + l_2^{\,2} + 2l_1l_2c_2$$

$$c_2 = \frac{x^2 + y^2 - (l_1^{\,2} + l_2^{\,2})}{2l_1l_2}$$

$$s_2^2 + c_2^2 = 1$$

$$s_2 = \pm\sqrt{1 - c_2^2}$$

$$q_2 = \operatorname{atan2}(s_2, c_2)$$

Soluciones Algebraicas

- Ejemplo 1

Solución

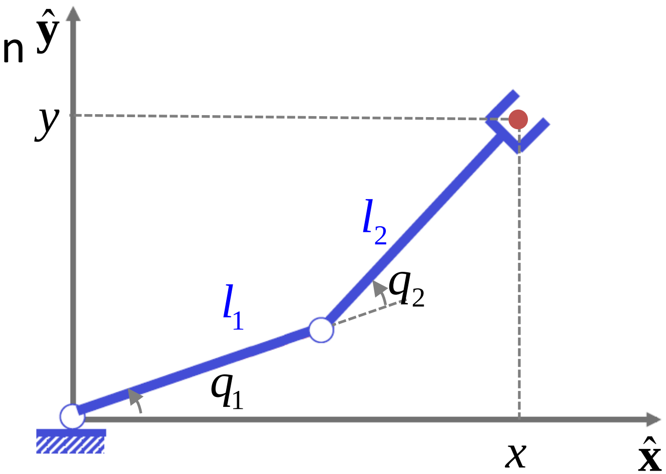
Encontrar la cinemática inversa para la posición de un algebraico

- Para q_1 (expandiendo términos de cinemática directa):

$$x = l_1 c_1 + l_2(c_1 c_2 - s_1 s_2)$$
$$y = l_1 s_1 + l_2(s_1 c_2 + c_1 s_2)$$

Ecuaciones son lineales en c_1, s_1: resolver

$$x = c_1(l_1 + l_2 c_2) - s_1(l_2 s_2)$$
$$y = s_1(l_1 + l_2\, \mathrm{c}_1) + c_1(l_2 s_2)$$



En forma matricial:

$$\begin{bmatrix} l_1 + l_2 c_2 & -l_2 s_2 \\ l_2 s_2 & l_1 + l_2 c_2 \end{bmatrix} \begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{\text{resolviendo}} \left\{ \begin{aligned} s_1 &= \frac{y(l_1 + l_2 c_2) - x l_2 s_2}{\det} \\ c_1 &= \frac{x(l_1 + l_2 c_2) + y l_2 s_2}{\det} \\ \det &= l_1^2 + l_2^2 + 2 l_1 l_2 c_2 \end{aligned} \right.$$

$$q_1 = \operatorname{atan2}(s_1, c_1)$$

Soluciones Algebraicas

- ## Ejemplo 2

El efector de un robot de 3 *gdl RPR*, con respecto a su base está dado por

$$
{}^{0}\boldsymbol{T}_{3} = \begin{bmatrix} \cos(\theta_1+\theta_3) & 0 & \sin(\theta_1+\theta_3) & a_3\cos(\theta_1+\theta_3)+q_2\sin\theta_1 \\ \sin(\theta_1+\theta_3) & 0 & -\cos(\theta_1+\theta_3) & a_3\sin(\theta_1+\theta_3)-q_2\cos(\theta_1) \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

donde a_3 y d_1 son constantes. La configuración deseada del efector final es:

$$
T_{des} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Solución

Encontrar la cinemática inversa $(\theta_1, q_2, \theta_3)$ de este robot como una función de los elementos de T_{des}.

El procedimiento consiste en obtener relaciones entre ambas matrices, y aplicar álgebra, para que el valor de cada articulación pueda despejarse.

Soluciones Algebraicas

- Ejemplo 2

\- Para

$$\begin{cases} p_x = a_3 c_{13} + s_1 q_2 \longrightarrow p_x - a_3 c_{13} = s_1 q_2 \\ p_y = a_3 s_{13} - c_1 q_2 \longrightarrow a_3 s_{13} - p_y = c_1 q_2 \end{cases}$$

$$\tan(\theta_1) = \frac{p_x - a_3 c_{13}}{a_3 s_{13} - p_y} = \frac{p_x - a_3 n_x}{a_3 n_y - p_y} \Rightarrow \theta_1 = \mathrm{atan2}(p_x - a_3 n_x, a_3 n_y - p_y)$$

\- Para θ_3:

$$\frac{a_x}{n_x} = \tan(\theta_1 + \theta_3) \longrightarrow \theta_1 + \theta_3 = \mathrm{atan2}(a_x, n_x)$$

$$\theta_3 = \mathrm{atan2}(a_x, n_x) - \theta_1$$

\- Para q_2:

$$\begin{cases} s_1 p_x = a_3 s_1 c_{13} + s_1^2 q_2 \\ c_1 p_y = a_3 s_{13} c_1 - c_1^2 q_2 \end{cases}$$

$$s_1 p_x - c_1 p_y = a_3(s_1 n_x - n_y c_1) + q_2$$

$$q_2 = s_1 p_x - c_1 p_y - a_3(s_1 n_x - n_y c_1)$$

Solución por Transformaciones Inversas

- Realiza multiplicaciones (a la derecha o izquierda) por las inversas de transformaciones homogéneas
  - Busca obtener expresiones "más simples" (menos variables)

Valores numéricos

- Se desea la transformación homogénea T_{des} para el efector final:

$$T_{des} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{0}T_n(q_1, q_2, \cdots, q_n) = {}^{0}T_1(q_1)\,{}^{1}T_2(q_2)\cdots{}^{n-1}T_n(q_n) = T_{des}$$

$$^{0}T_1(q_1)^{-1}T_{des} = {}^{1}T_2(q_2)\cdots{}^{n-1}T_n(q_n)$$

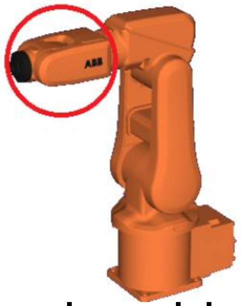
$$^{0}T_1(q_1)^{-1}T_{des}\,{}^{n-1}T_n(q_n)^{-1} = {}^{1}T_2(q_2)\cdots{}^{n-2}T_{n-1}(q_{n-1})$$

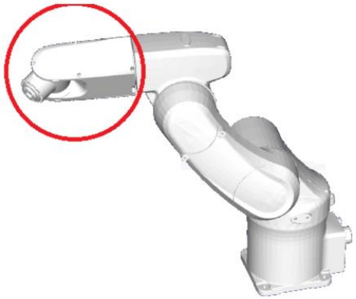
- Expresión analítica (posición/orientación) del efector

Desacoplo Cinemático

- Condición suficiente (no necesaria) para la existencia de solución analítica para un robot de 6 grados de libertad:
  - El robot tiene 3 articulaciones de revolución consecutivas que se intersecan en un punto común (muñeca esférica), o
  - El robot tiene 3 articulaciones de revolución consecutivas que son paralelas

- Se le conoce como el método de Pieper: se aplica a muñecas esféricas



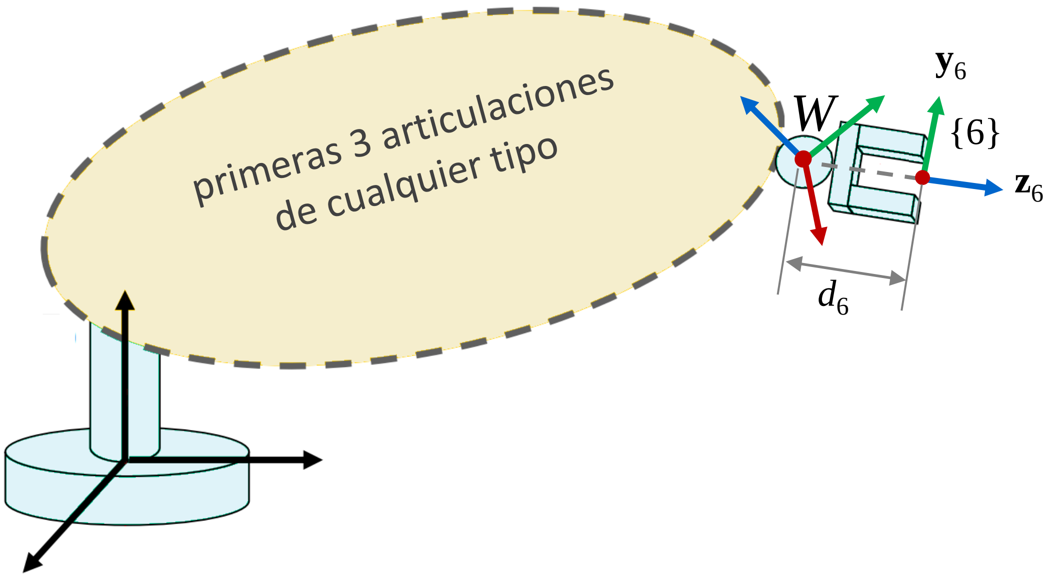


Bastantes robots industriales tienen muñecas esféricas

- Idea: separar el problema en 2 subproblemas
  - Cinemática Inversa para la posición (de la muñeca ["wrist"])
  - Cinemática inversa para la orientación (del efector final con respecto a la muñeca)

Desacoplo Cinemático

- Robot con 6 grados de libertad:



$$T_{des} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Posición y orientación deseadas para el efector final {6} con respecto a la base

- Posición de la muñeca: $p_w = \begin{bmatrix} p_x - d_6 a_x \\ p_y - d_6 a_y \\ p_z - d_6 a_z \end{bmatrix}$ Valor numérico

- Orientación del efector final con respecto a la muñeca: ${}^3R_6$

Desacoplo Cinemático

- Procedimiento

1. Se obtiene la posición deseada de la muñeca (a partir de T_{des})

$$p_w = \begin{bmatrix} p_x - d_6 a_x \\ p_y - d_6 a_y \\ p_z - d_6 a_z \end{bmatrix}$$

y su expresión analítica a partir de ${}^0T_3(q_1, q_2, q_3)$. Relacionando ambos, se obtiene los ángulos q_1, q_2, q_3

2. Con los valores de q_1, q_2, q_3 se calcula la rotación ${}^0R_3$ y la transformación homogénea ${}^0T_3$

3. Se obtiene q_4, q_5, q_6 despejando que lleva a:

$$T_{des} = {}^0T_3\, {}^3T_6(q_4, q_5, q_6)$$

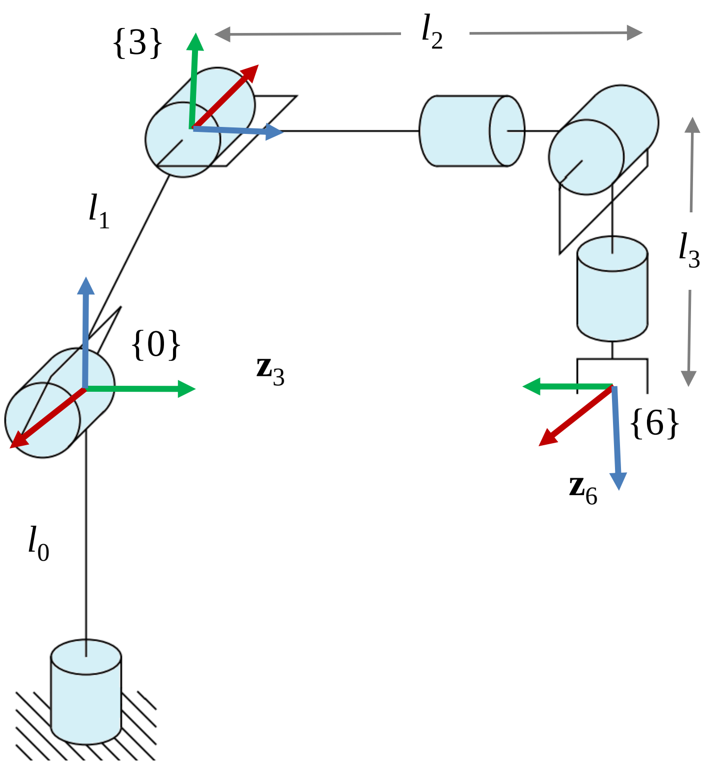
$$\underbrace{{}^3T_6(q_4, q_5, q_6)}_{\text{Tiene la estructura de ángulos de Euler ZYZ o ZXZ}} = {}^0T_3^{-1} T_{des}$$

- *Nota*: en los pasos 1 y 3 se requiere análisis algebraico y/o geométrico

Desacoplo Cinemático

- Ejercicio

  Resolver el problema de cinemática inversa para el siguiente robot de 6 *gdl,* si se sabe que sus parámetros de Denavit-Hartenberg son los mostrados en la tabla

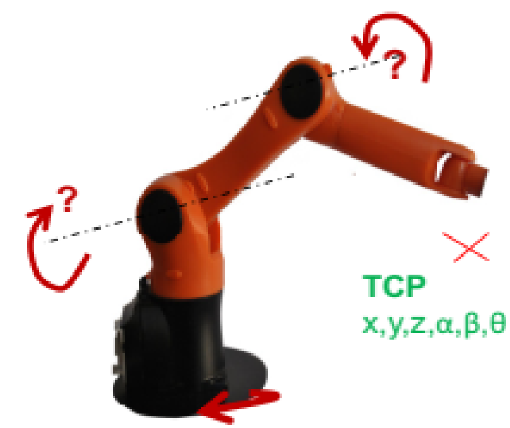


θ	d	A	α
q_1	0	0	90°
q_2	0	l_1	0°
q_3	0	0	90°
q_4	l_2	0	-90°
q_5	0	0	90°
q_6	l_3	0	0°

CINEMÁTICA DE POSICIÓN INVERSA

PROBLEMA CINEMATICO INVERSO

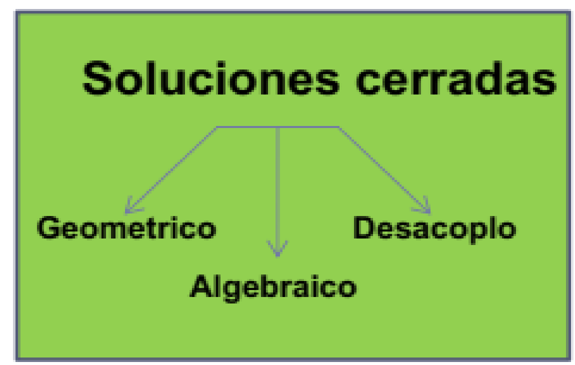
Conocida la localización del robot, determina cual debe ser la configuración del robot (articulaciones y parámetros geométricos).



1) La resolución del problema no es sistemático.
2) Depende fuertemente de la configuración del robot
3) Es encontrar la siguiente relación explicita (solución cerrada):

$$q_k=f(x,y,z,\alpha,\beta,\Upsilon) \quad k=1\ldots n \text{ (GDL)}$$

4) No existe siempre solución cerrada.
   Condiciones para que exista solución cerrada:
   a) 3 ejes de articulación adyacentes interseccionan en un punto (robot PUMA y stanford).
   b) 3 ejes de articulación adyacentes son paralelos entre si (ASEA, etc.).

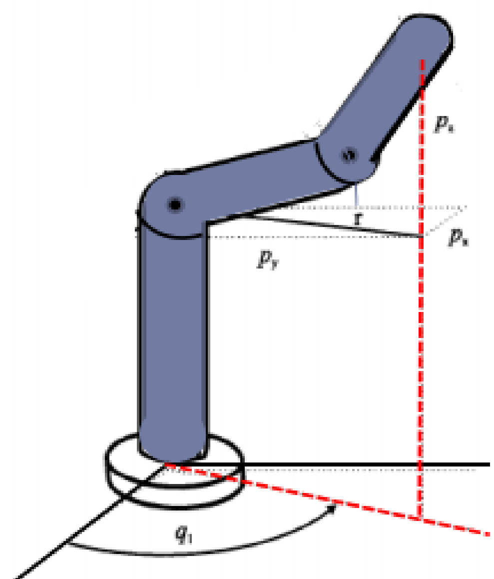


CINEMÁTICA INVERSA: MÉTODO GEOMÉTRICO

Se basan en descomponer la cadena cinemática en distintos planos geométricos y resolviendo por trigonometría cada plano. Se trata de encontrar el numero suficiente de relaciones geométricas para posicionar el extremo del robot. Se utiliza para las primeras articulaciones.

Ejemplo: **Robot esférico 3 GDL**

Datos: Px, Py, Pz donde se quiere situar el extremo del robot.



$$\mathrm{q}_1 = \arctan\left(\frac{p_y}{p_x}\right)$$

$$\cos q_3 = \frac{p_x^2 + p_y^2 + p_z^2 - l_2^2 - l_3^2}{2 l_2 l_3}$$

$$\sin q_3 = \pm\sqrt{1-\cos^2 q_3}$$

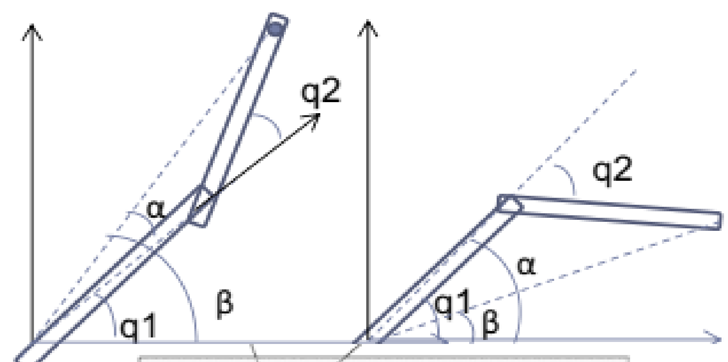
$$\mathrm{q}_3 = \arctan\left(\frac{\pm\sqrt{1-\cos^2 q_3}}{\cos q_3}\right)$$

a articulación q_2 tiene dos soluciones: (codo arriba y codo abajo):

$$q_2 = \beta - \alpha$$

$$\beta = \arctan\left(\frac{p_z}{r}\right) = \arctan\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right)$$

$$\alpha = \arctan\left(\frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right)$$



$$\mathrm{q}_2 = \arctan\left(\frac{p_z}{\pm\sqrt{p_x^2 + p_y^2}}\right) - \arctan\left(\frac{l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right)$$

CINEMÁTICA INVERSA: MATRICES HOMOGÉNEAS

Se basan en manipular las ecuaciones resultantes obtenidas a partir del modelo Cinematico Directo:

Esto es, despejar las n variables $\mathbf{q_i}$ en función de los vectores **n, o, a, p:**

Utilizando las **Matrices de Transformación Homogénea**, operamos de la siguiente forma:

$$
\begin{aligned}
&{}^{0}_{n}\boldsymbol{T} = {}^{0}_{1}\boldsymbol{A}\,{}^{1}_{2}\boldsymbol{A}\,{}^{2}_{3}\boldsymbol{A}\cdots{}^{n-1}_{n}\boldsymbol{A} \\
&\left({}^{0}_{1}\boldsymbol{A}\right)^{-1}\,{}^{0}_{n}\boldsymbol{T} = {}^{1}_{2}\boldsymbol{A}\,{}^{2}_{3}\boldsymbol{A}\cdots{}^{n-1}_{n}\boldsymbol{A} \Rightarrow \boldsymbol{depejamos} \quad \boldsymbol{q}_1 \\
&\left({}^{1}_{2}\boldsymbol{A}\right)^{-1}\left({}^{0}_{1}\boldsymbol{A}\right)^{-1}\,{}^{0}_{n}\boldsymbol{T} = {}^{2}_{3}\boldsymbol{A}\cdots{}^{n-1}_{n}\boldsymbol{A} \Rightarrow \boldsymbol{depejamos} \quad \boldsymbol{q}_2 \\
&\vdots \\
&\left({}^{2}_{3}\boldsymbol{A}\right)^{-1}\left({}^{0}_{1}\boldsymbol{A}\right)^{-1}\,{}^{0}_{n}\boldsymbol{T} = \cdots{}^{n-1}_{n}\boldsymbol{A} \Rightarrow \boldsymbol{depejamos} \quad \boldsymbol{q}_{n-1}\ \boldsymbol{y}\ \boldsymbol{q}_n
\end{aligned}
$$

$$
{}^{0}_{n}\boldsymbol{T}(\boldsymbol{q}_1,..,\boldsymbol{q}_n) = \begin{bmatrix} \vec{\boldsymbol{n}} & \vec{\boldsymbol{o}} & \vec{\boldsymbol{a}} & \vec{\boldsymbol{p}} \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Consideraciones:

Se trata de igualar elementos de ambos lados de cada ecuación, tomando los casos en los que solo aparezca una variable de articulación, empleando identidades trigonométricas y buscando divisiones en función de arco tangentes.

CINEMÁTICA INVERSA: MATRICES HOMOGÉNEAS

Ejemplo: **Robot esférico 3 GDL**

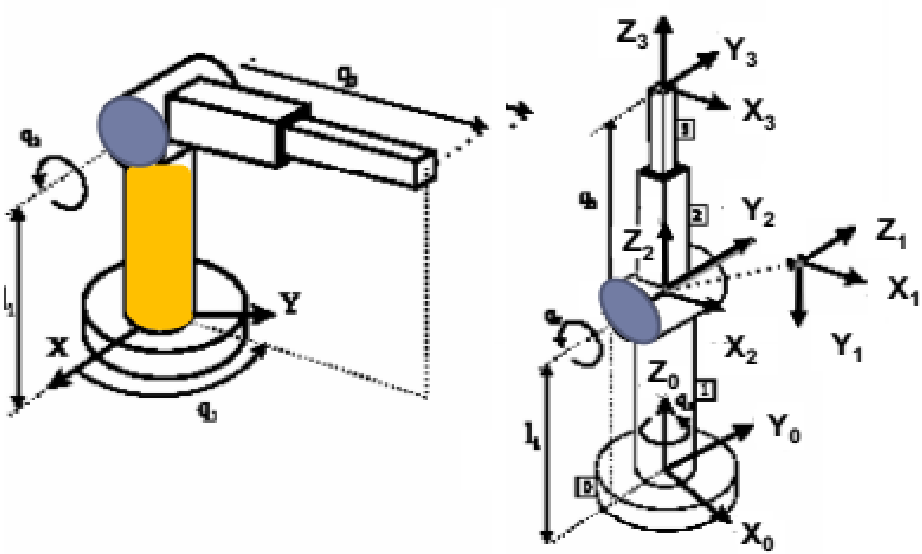


Tabla D-H

	θi	di	ai	αi
1	q1	L_1	0	-90º
2	q2	0	0	90º
3	0	q3	0	0º

MTH:

$$ {}^{0}_{1}A = \begin{bmatrix} Cq1 & 0 & Sq1 & 0 \\ Sq1 & 0 & -Cq1 & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}A = \begin{bmatrix} Cq2 & 0 & -Sq2 & 0 \\ Sq2 & 0 & Cq2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

CINEMÁTICA INVERSA: MATRICES HOMOGÉNEAS

Ejemplo: **Robot esférico 3 GDL**

$${}^{0}_{3}T = {}^{0}_{1}A\,{}^{1}_{2}A\,{}^{2}_{3}A$$

px py pz DATOS (Posición del terminal)

$$\left({}^{0}_{1}A\right)^{-1}\,{}^{0}_{n}T = {}^{1}_{2}A\,{}^{2}_{3}A \Rightarrow despejamos \quad q_1$$

$$\left({}^{0}_{1}A\right)^{-1}\,{}^{0}_{n}T = \begin{bmatrix} Cq1 & 0 & -Sq1 & 0 \\ Sq1 & 0 & Cq1 & 0 \\ 0 & -1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Cq2 & 0 & Sq2 & 0 \\ Sq2 & 0 & -Cq2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left({}^{0}_{1}A\right)^{-1}\,{}^{0}_{n}T = \begin{bmatrix} Cq1 & Sq1 & 0 & 0 \\ 0 & 0 & -1 & L1 \\ -Sq1 & Cq1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Cq2 & 0 & Sq2 & \boxed{Sq2\,q3} \\ Sq2 & 0 & -Cq2 & -Cq2\,q3 \\ 0 & 1 & 0 & \boxed{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$-p_x\,Sq1 + p_y\,Cq1 = 0 \Rightarrow \frac{Sq1}{Cq1} = \frac{p_y}{p_x} \Rightarrow \boxed{q1 = \arctan\left(\frac{p_y}{p_x}\right)}$$

CINEMÁTICA INVERSA: MATRICES HOMOGÉNEAS

Ejemplo: **Robot esférico 3 GDL**

$$\left({}^{1}_{2}A\right)^{-1}\left({}^{0}_{1}A\right)^{-1}\,{}^{0}_{3}T = {}^{2}_{3}A \Rightarrow despejamos \quad q_2 \quad y \quad q_3$$

$$\left({}^{1}_{2}A\right)^{-1}\left({}^{0}_{1}A\right)^{-1}\,{}^{0}_{n}T = \begin{bmatrix} Cq2 & Sq2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -Sq2 & Cq2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cq1 & Sq1 & 0 & 0 \\ 0 & 0 & -1 & -L1 \\ -Sq1 & Cq1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\left({}^{1}_{2}\boldsymbol{A}\right)^{-1}\left({}^{0}_{1}\boldsymbol{A}\right)^{-1}\,{}^{0}_{n}\boldsymbol{T} = \begin{bmatrix} \boldsymbol{Cq2Cq1} & \boldsymbol{Cq2Sq1} & \boldsymbol{Sq2} & \boldsymbol{-L1Sq2} \\ \boldsymbol{-Sq1} & \boldsymbol{Cq1} & 1 & 0 \\ \boldsymbol{-Sq2Cq1} & \boldsymbol{-Sq2Sq1} & \boldsymbol{Cq2} & \boldsymbol{-L1Cq2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{n_x} & \boldsymbol{o_x} & \boldsymbol{a_x} & \boldsymbol{p_x} \\ \boldsymbol{n_y} & \boldsymbol{o_y} & \boldsymbol{a_y} & \boldsymbol{p_y} \\ \boldsymbol{n_z} & \boldsymbol{o_z} & \boldsymbol{a_z} & \boldsymbol{p_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \boldsymbol{q3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Px\,Cq2\,Cq1 + PySq1\,Cq2 + PzSq2 - L1Sq2 = 0 \Rightarrow -\frac{Sq2}{Cq2} = \frac{PxCq1 + PySq1}{Pz - L1} \Rightarrow \boxed{q2 = \arctan\left(-\frac{PxCq1 + PySq1}{Pz - L1}\right)}$$

$$\boldsymbol{-Sq2\,Cq1Pz - Sq2\,Sq1Py + PzCq2 - L1Cq2 = q3} \Rightarrow \boxed{\boldsymbol{q3 = Cq2(Pz - L1) - Sq2(Cq1Pz + Sq1Py)}}$$

CINEMÁTICA INVERSA: DESACOPLO CINEMÁTICO

Se basan en al resolución independiente de los grados de libertad que posicionan (3) y de los que orientan la muñeca (3).
Por lo que el problema cinemático inverso se divide en dos subproblemas:

1. Resolver las tres primeras articulaciones de posición.
2. Resolver las tres ultimas articulaciones que corresponden a la muñeca.

El método de resolución:

1) A partir de la posición y orientación que se busca [n,o,a,p], se obtiene el punto de corte de los 3 últimos grados de libertad (punto de muñeca Pm).
2) Se resuelve el problema cinemático inverso para el brazo de 3 GDL (q_1,q_2,q_3) que llega hasta la **Pm** (desde la base).
3) Se resuelve el problema cinemático inverso que va desde **Pm** hasta el punto final pf (calculando q_4,q_5,q_6).

Soluciones Numéricas

- Se utilizan cuando:
  - No existe solución analítica
  - Hay infinitas soluciones (ejm. robots redundantes)
  - Es “muy complicado” hallarla
- Idea:

$$\underbrace{\mathbf{x} = f(\mathbf{q})}_{\text{Cinemática directa}} \Rightarrow \underbrace{\mathbf{x} - f(\mathbf{q}) = 0}_{\text{Encontrar } \mathbf{q} \text{ (por iteración) tal que la diferencia sea cero}}$$

- Métodos de solución:
  - Método de Newton: encontrar las raíces del problema
  - Método de máximo decrecimiento (*Gradient descent*): minimiza la

Soluciones Numéricas

a) Método de Newton

- Problema: dado $\boldsymbol{x}_d$ (constante), encontrar $\boldsymbol{q}$ tal que

$$\mathbf{x}_d - f(\mathbf{q}) = 0$$

$\boldsymbol{x}_{\mathrm{d}}$: valor deseado para $\boldsymbol{x}$
$\boldsymbol{f}$: función de cinemática directa

- Procedimiento para la solución
  - Aproximación de primer orden de Taylor para $f(\mathbf{q})$

$$f(\mathbf{q}) \approx f(\mathbf{q}_k) + J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k)$$

Matriz Jacobiana

$$J(\mathbf{q}_k) = \frac{\partial f(\mathbf{q}_k)}{\partial \mathbf{q}} \in \mathbb{R}^{n \times m}$$

n gdl
m: tamaño de x

  - Reemplazando: $\mathbf{x}_d - \left(f(\mathbf{q}_k) + J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k) \right) = 0$

$$\mathbf{x}_d - f(\mathbf{q}_k) = J(\mathbf{q}_k)(\mathbf{q} - \mathbf{q}_k)$$

  - Asumiendo J cuadrado e invertible ($n = m$):

$$J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right) = (\mathbf{q} - \mathbf{q}_k)$$

- Solución final ($\mathbf{q} = \mathbf{q}_{k+1}$):

$$\boxed{\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right)}$$

Soluciones Numéricas

a) Método de Newton

- Algoritmo:
  - Se comienza con un $\mathbf{q}_0$ inicial (suele ser la configuración actual)
  - Se actualiza de manera iterativa usando:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right) \qquad k = 0,1,2,3,\ldots$$

  - Se detiene cuando:

$$\underbrace{\| \mathbf{x}_d - f(\mathbf{q}_k) \| < \varepsilon}_{\text{error Cartesiano pequeño}} \quad \text{o} \quad \underbrace{\| \mathbf{q}_{k+1} - \mathbf{q}_k \| < \varepsilon}_{\text{Incremento articular pequeño}} \qquad \varepsilon \text{ : valor pequeño}$$

- Comentarios:
  - Convergencia si se comienza con $\mathbf{q}_0$ (valor inicial) cercano a la solución
  - Cuando hay redundancia ($m < n$):

    ¡J no es cuadrada (y no existe inversa)! → se usa la pseudo-inversa
  - Desventajas:
    - Tiempo de cálculo de la inversa (o pseudo-inversa)
    - Problemas cerca a las singularidades de J (matriz Jacobiana)
  - Ventaja: es rápido (convergencia cuadrática)

Soluciones Numéricas

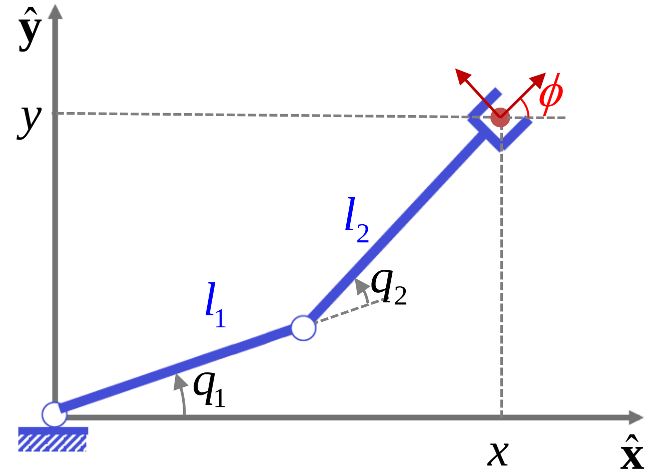
a) Método de Newton

- Ejemplo: robot RR

Calcular los valores articulares para que el efector final llegue al punto $x = 1.2$, $y = 1.2$ usando el método de Newton. Asumir que $l_1 = l_2 = 1$.

Solución

- Cinemática directa: $f(\mathbf{q}) = \begin{bmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{bmatrix}$



- Matriz Jacobiana (Jacobiano):

$$J(\mathbf{q}) = \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \frac{\partial f_x}{\partial q_2} \\ \frac{\partial f_y}{\partial q_1} & \frac{\partial f_y}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -(s_1 + s_{12}) & -s_{12} \\ c_1 + c_{12} & c_{12} \end{bmatrix}$$

- Posición deseada:

$$\mathbf{x}_d = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}$$

- Inversa del Jacobiano:

$$J^{-1}(\mathbf{q}) = \frac{1}{s_2} \begin{bmatrix} c_{12} & s_{12} \\ -(c_1 + c_{12}) & -(s_1 + s_{12}) \end{bmatrix}$$

Soluciones Numéricas

a) Método de Newton

- **Ejemplo**: robot RR

Calcular los valores articulares para que el efector final llegue al punto $x = 1.2$, $y = 1.2$ usando el método de Newton. Asumir que $l_1 = l_2 = 1$.

Solución

- Ecuación genérica del método de Newton:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + J^{-1}(\mathbf{q}_k)(\mathbf{x}_d - f(\mathbf{q}))$$

- Reemplazando los valores anteriores:

$$\mathbf{q}_{k+1} = q_k + \left(\frac{1}{s_2} \begin{bmatrix} c_{12} & s_{12} \\ -(c_1 + c_{12}) & -(s_1 + s_{12}) \end{bmatrix} \right) \underbrace{\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{bmatrix} \right)}_{\text{Error actual}}$$

- Se aplica iterativamente.
  Por ejemplo, tomar $q_1 = 0.5$ y $q_2 = 0.5$ como valores iniciales (nota: no se puede iniciar con $q_2 = 0$ pues J^{-1} sería indeterminado)

Soluciones Numéricas

a) Método de Newton

- Ejemplo: robot RR (con Python)

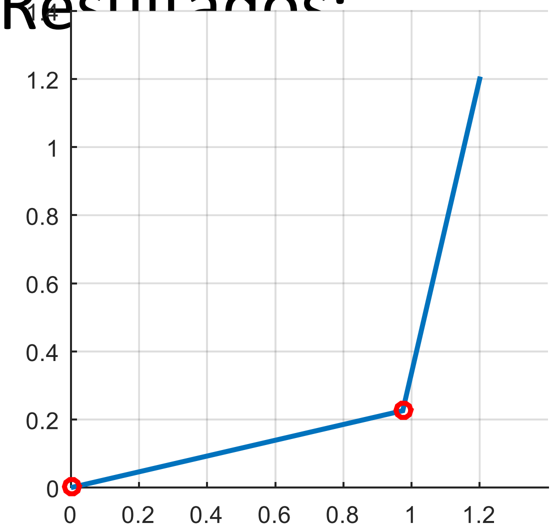
```
import numpy as np
cos=np.cos; sin=np.sin

xd = np.array([1.2, 1.2]) # Valor deseado en el espacio cartesiano
q  = np.array([0.5, 0.5]) # Valor inicial en el espacio articular
epsilon = 1e-3
max_iter = 100  # Maximo numero de iteraciones

# Iteraciones: Metodo de Newton
for i in range(max_iter):
    q1 = q[0]; q2 = q[1];
    J = np.array([
            [-sin(q1)-sin(q1+q2), -sin(q1+q2)],
            [ cos(q1)+cos(q1+q2),  cos(q1+q2)]])
    f = np.array([cos(q1)+cos(q1+q2), sin(q1)+sin(q1+q2)])
    e = xd-f
    q = q + np.dot(np.linalg.inv(J), e)
    # Condicion de termino
    if (np.linalg.norm(e) < epsilon):
        break
print(q)
```

Soluciones Numéricas

a) Método de Newton

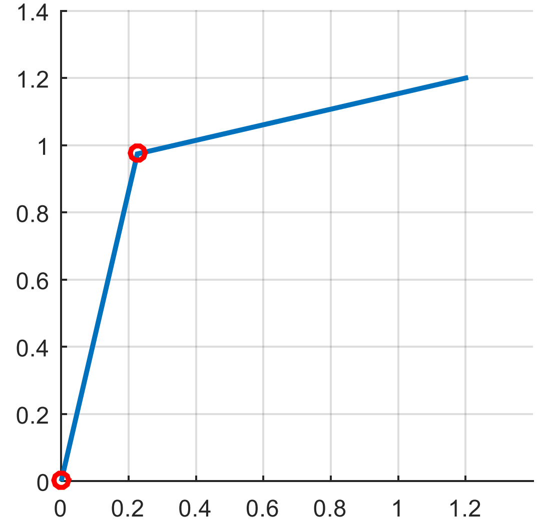
- Ejemplo: robot RR
  - Notar que el resultado depende del valor inicial utilizado.
  - Resultados:

$$\mathbf{q} = \begin{bmatrix} 0.2278 \\ 1.1157 \end{bmatrix}$$



Configuración obtenida usando q=[0.5; 0.5] como valor inicial

$$\mathbf{q} = \begin{bmatrix} 1.3430 \\ -1.1152 \end{bmatrix}$$



Configuración obtenida usando q = [1; -1] como valor inicial

(Método del Gradiente)

- Objetivo:
  - Minimizar la función genérica $g(\mathbf{q})$

$$\min_{\mathbf{q}} g(\mathbf{q})$$

- Idea:
  - Empezar con un valor inicial $\mathbf{q}_0$
  - "Moverse" en la dirección **negativa** del gradiente (∇g) iterativamente

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \nabla g(\mathbf{q}_k)$$

$\alpha \in \mathbb{R}^+$: tamaño de paso

  - El tamaño de paso (*step size*) $\alpha > 0$ debe garantizar un máximo decrecimiento de $g(\mathbf{q})$ en cada iteración:
    - α muy alto: puede ocasionar divergencia (no se encuentra el mínimo)
    - α muy pequeños: genera convergencia muy lenta

Recordar: el gradiente indica la dirección de máxima variación

Soluciones Numéricas

b) Método del Gradiente

- Cinemática directa: $\mathbf{x}_d = f(\mathbf{q})$ o equivalentemente $\underbrace{\mathbf{x}_d - f(\mathbf{q})}_{\text{error}} = 0$
- Procedimiento:
  - Definir una función escalar de *error*: $g(\mathbf{q}) = \frac{1}{2}\|\mathbf{x}_d - f(\mathbf{q})\|^2 \longleftarrow \quad g: \mathbb{R}^n \to \mathbb{R}$
  - *Objetivo*: minimizar el error: $\min_{\mathbf{q}} g(\mathbf{q})$
  - Calcular el gradiente de $g(\mathbf{q})$:

$$g(\mathbf{q}) = \frac{1}{2}(\mathbf{x}_d - f(\mathbf{q}))^T(\mathbf{x}_d - f(\mathbf{q})) \quad \Rightarrow \quad \nabla g(\mathbf{q}) = -\underbrace{\left(\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}}\right)^T}_{J(\mathbf{q})}(\mathbf{x}_d - f(\mathbf{q}))$$

  - Aplicar el método del gradiente

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \alpha \nabla g(\mathbf{q}_k)$$

$$\boxed{\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha J^T(\mathbf{q}_k)\left(\mathbf{x}_d - f(\mathbf{q}_k)\right)}$$

- Ventaja: computacionalmente más simple (transpuesta en lugar de inversa)
- Desventaja: puede tener convergencia lenta (se puede usar regla de Armijo)

Soluciones Numéricas

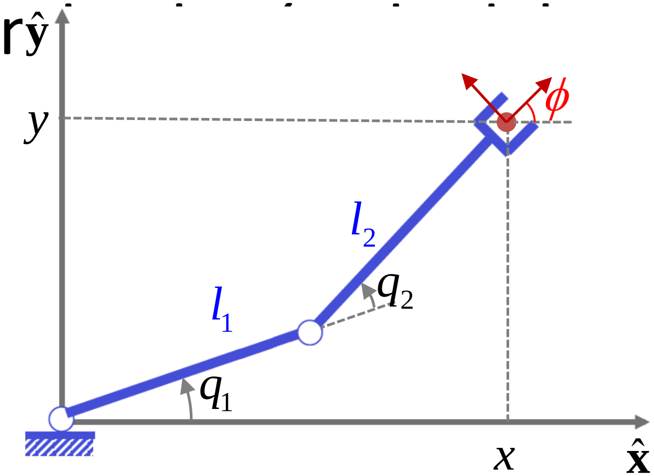
b) Método del Gradiente

- Ejemplo: robot RR

Calcular los valores articulares para que el efector final llegue al punto $x = 1.2$, $y = 1.2$ usar gradiente. Asumir que $l_1 = l_2 = 1$.

Solución

- Cinemática directa: $f(\mathbf{q}) = \begin{bmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{bmatrix}$



- Matriz Jacobiana (Jacobiano):

$$J(\mathbf{q}) = \frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial f_x}{\partial q_1} & \frac{\partial f_x}{\partial q_2} \\ \frac{\partial f_y}{\partial q_1} & \frac{\partial f_y}{\partial q_2} \end{bmatrix} = \begin{bmatrix} -(s_1 + s_{12}) & -s_{12} \\ c_1 + c_{12} & c_{12} \end{bmatrix}$$

- Posición deseada:

$$\mathbf{x}_d = \begin{bmatrix} 1.2 \\ 1.2 \end{bmatrix}$$

- Transpuesta del Jacobiano:

$$J^T(\mathbf{q}) = \begin{bmatrix} -(s_1 + s_{12}) & c_1 + c_{12} \\ -s_{12} & c_{12} \end{bmatrix}$$

Soluciones Numéricas

b) Método del Gradiente

- **Ejemplo:** robot RR

Calcular los valores articulares para que el efector final llegue al punto $x = 1.2$, $y = 1.2$ usando el método del gradiente. Asumir que $l_1 = l_2 = 1$.

Solución

- Ecuación genérica del método del gradiente:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \alpha J^T(\mathbf{q}_k)(\mathbf{x}_d - f(\mathbf{q}))$$

- Reemplazando los valores anteriores:

$$\mathbf{q}_{k+1} = q_k + \alpha \begin{bmatrix} -(s_1 + s_{12}) & (c_1 + c_{12}) \\ -s_{12} & c_{12} \end{bmatrix} \underbrace{\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{bmatrix} \right)}_{\text{Error actual}}$$

- Se aplica iterativamente.
  Por ejemplo, tomar $q_1 = 0.5$ y $q_2 = 0.5$ como valores iniciales (en este caso sí se puede iniciar con $q_2 = 0$)

Soluciones Numéricas

b) Método del Gradiente

- Ejemplo: robot RR (en Python)

```
import numpy as np
cos=np.cos; sin=np.sin

xd = np.array([1.2, 1.2]) # Valor deseado en el espacio cartesiano
q  = np.array([0.5, 0.5]) # Valor inicial en el espacio articular
epsilon = 1e-3
max_iter = 1000           # Maximo numero de iteraciones
alpha = 0.5

# Iteraciones: descenso del gradiente
for i in range(max_iter):
    q1 = q[0]; q2 = q[1];
    J = np.array([[-sin(q1)-sin(q1+q2), -sin(q1+q2)],
                  [ cos(q1)+cos(q1+q2),  cos(q1+q2)]])
    f = np.array([cos(q1)+cos(q1+q2), sin(q1)+sin(q1+q2)])
    e = xd-f
    q = q + alpha*np.dot(J.T, e)
    # Condicion de finalizacion
    if (np.linalg.norm(e) < epsilon):
        break
print(q)
```

Soluciones Numéricas

Comparación

- Método de Newton
  - Rapidez de convergencia cuadrática (rápido)
  - Problemas cerca a las singularidades (las singularidades se pueden evaluar con los valores singulares de J)

- Método de gradiente
  - Rapidez de convergencia lineal (lento)
  - No presenta problemas con singularidades
  - Se debe escoger el incremento (α): se puede usar métodos adaptativos como búsqueda lineal (line search)

CINEMÁTICA DE MOVIMIENTO: MODELO DIFERENCIAL

Hasta ahora se ha considerado únicamente las relaciones de las articulaciones de una manera **estática** en ausencia de movimiento del robot (problemas Cinemático Directo e Inverso).

Cuando el robot se desplaza, los elementos de la cadena cinemática **propagan** de una articulación a al siguiente tanto **velocidades lineales como angulares**.
La velocidad del elemento i+1 será la del elemento **i** mas las componentes que añade la articulación **i+1.**

Métodos de resolución
Modelo Diferencial

Propagación de velocidades

Ecuaciones de revolución

Ecuaciones de Translación

Cálculo Matriz Jacobiana

Velocidades de las articulaciones

$$\begin{bmatrix} \dot{q}1 \\ \dot{q}2 \\ \vdots \\ \dot{q}n \end{bmatrix}$$

Jacobiano DIRECTO

Jacobiano INVERSO

Velocidades del extremo del robot

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}$$

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

La jacobiana es una matriz en derivadas. En general, para un conjunto de m funciones que dependen de n variables independientes:

$$\left.\begin{array}{l} y_1 = f_1(x_1, x_2, \ldots, x_n) \\ y_2 = f_1(x_1, x_2, \ldots, x_n) \\ \vdots \\ y_m = f_m(x_1, x_2, \ldots, x_n) \end{array}\right\} \Leftrightarrow \mathbf{y} = \mathbf{F}(\mathbf{x})$$

Si se considera que las n variables son dependientes del tiempo, se puede expresar de la siguiente manera:

Matriz Jacobiana

$$\left.\begin{array}{l} \dot{y}_1 = \dfrac{\partial f_1}{\partial x_1}\cdot \dot{x}_1 + \dfrac{\partial f_1}{\partial x_2}\cdot \dot{x}_2 + \ldots + \dfrac{\partial f_n}{\partial x_n}\cdot \dot{x}_n \\ \dot{y}_2 = \dfrac{\partial f_2}{\partial x_1}\cdot \dot{x}_1 + \dfrac{\partial f_2}{\partial x_2}\cdot \dot{x}_2 + \ldots + \dfrac{\partial f_2}{\partial x_n}\cdot \dot{x}_n \\ \vdots \\ \dot{y}_m = \dfrac{\partial f_m}{\partial x_1}\cdot \dot{x}_1 + \dfrac{\partial f_m}{\partial x_2}\cdot \dot{x}_2 + \ldots + \dfrac{\partial f_m}{\partial x_n}\cdot \dot{x}_n \end{array}\right\} \Leftrightarrow \dot{\mathbf{y}} = \boxed{\mathbf{J}(x)}\cdot \dot{x}$$

Se puede observar que al ser ***x*** función del tiempo, para cada instante de tiempo la Matriz Jacobiana es distinta.

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

En robótica, la jacobiana se utiliza tomando como funciones las correspondientes a las posiciones del extremo del robot siendo las variables independientes de dichas funciones las articulaciones:

$$
\begin{aligned}
x &= f_x(q_1, q_2, \ldots, q_n)\\
y &= f_y(q_1, q_2, \ldots, q_n)\\
z &= f_z(q_1, q_2, \ldots, q_n)\\
\alpha &= f_\alpha(q_1, q_2, \ldots, q_n)\\
\beta &= f_\beta(q_1, q_2, \ldots, q_n)\\
\gamma &= f_\gamma(q_1, q_2, \ldots, q_n)
\end{aligned}
$$

Jacobiana

Velocidad lineal del extremo

$$
\begin{aligned}
\dot{x} &= \frac{\partial f_x}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_x}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_x}{\partial q_n}\cdot\dot{q}_n\\
\dot{y} &= \frac{\partial f_y}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_y}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_y}{\partial q_n}\cdot\dot{q}_n\\
\dot{z} &= \frac{\partial f_z}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_z}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_z}{\partial q_n}\cdot\dot{q}_n
\end{aligned}
$$

$$
\begin{aligned}
\dot{\alpha} &= \frac{\partial f_\alpha}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_\alpha}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_\alpha}{\partial q_n}\cdot\dot{q}_n\\
\dot{\beta} &= \frac{\partial f_\beta}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_\beta}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_\beta}{\partial q_n}\cdot\dot{q}_n\\
\dot{\gamma} &= \frac{\partial f_\gamma}{\partial q_1}\cdot\dot{q}_1 + \frac{\partial f_\gamma}{\partial q_2}\cdot\dot{q}_2 \cdots + \frac{\partial f_\gamma}{\partial q_n}\cdot\dot{q}_n
\end{aligned}
$$

Velocidad angular extremo

Representación matricial

Velocidades del extremo del robot

Velocidades de las articulaciones

$$
\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\\ \dot{\alpha}\\ \dot{\beta}\\ \dot{\gamma}\end{bmatrix} = \mathbf{J}(q)\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \vdots\\ \dot{q}_n\end{bmatrix}
$$

$$
\begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}J_v(q)\\ J_\omega(q)\end{bmatrix}\cdot\dot{q}
$$

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

Ejemplo: **Robot esférico 3 GDL**

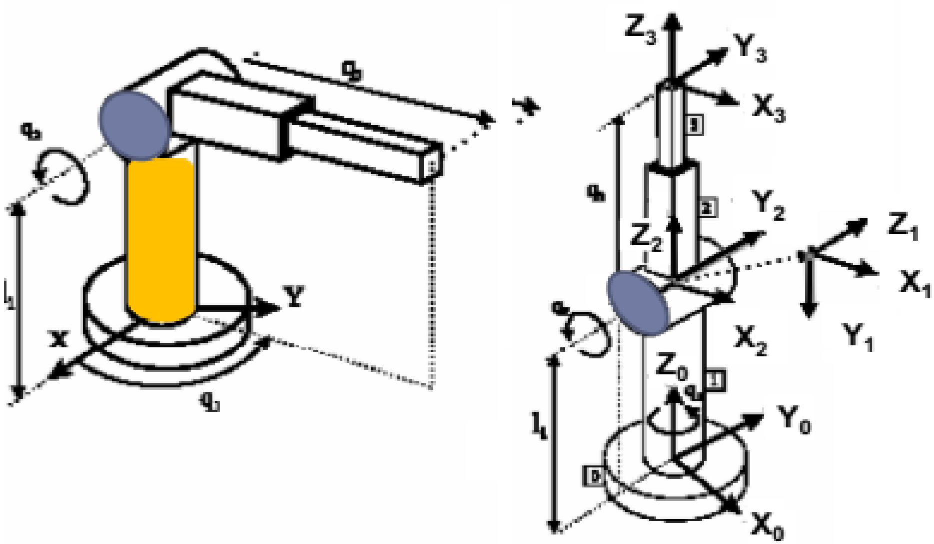


Tabla D-H

	θi	di	ai	αi
1	**q1**	**L_1**	**0**	**-90º**
2	**q2**	**0**	**0**	**90º**
3	**0**	**q3**	**0**	**0º**

MTH:

$$ {}^{0}_{1}A = \begin{bmatrix} Cq1 & 0 & Sq1 & 0 \\ Sq1 & 0 & -Cq1 & 0 \\ 0 & 1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{1}_{2}A = \begin{bmatrix} Cq2 & 0 & -Sq2 & 0 \\ Sq2 & 0 & Cq2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{2}_{3}A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

Ejemplo: **Robot esférico 3 GDL**

$$ {}^0_3A = \begin{bmatrix} Cq1 & 0 & -Sq1 & 0 \\ Sq1 & 0 & Cq1 & 0 \\ 0 & -1 & 0 & L1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Cq2 & 0 & Sq2 & 0 \\ Sq2 & 0 & -Cq2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = $$

$$ {}^0_3A = \left[\begin{array}{ccc|c} Cq1Cq2 & -Sq1 & Cq1Sq2 & \boxed{q3Cq1Sq2} \\ Sq1Cq2 & Cq1 & Sq1Sq2 & \boxed{q3Sq1Sq2} \\ Sq2 & 0 & Cq2 & \boxed{q3Cq2+L1} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] $$

$$ \boldsymbol{x = f_x(q_1, q_2, \ldots, q_n)} $$

$$ \boldsymbol{y = f_y(q_1, q_2, \ldots, q_n)} $$

$$ \boldsymbol{z = f_z(q_1, q_2, \ldots, q_n)} $$

$$ x = q3\,Cq1\,Sq2 $$

$$ y = q3\,Sq1\,Sq2 $$

$$ z = q3Cq2 + L1 $$

Ahora se calcula el Jacobiano, a partir de las ecuaciones de posición del extremo

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

Ejemplo: **Robot esférico 3 GDL**

$$x = q3\,Cq1\,Sq2$$
$$y = q3\,Sq1\,Sq2$$
$$z = q3Cq2 + L1$$

Jacobiano

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{bmatrix} = \begin{bmatrix} -q3Sq1Sq2 & q3Cq1Cq2 & Cq1Sq2 \\ q3Cq1Sq2 & q3Sq1Cq2 & Sq1Sq2 \\ 0 & -q3Sq2 & Cq2 \end{bmatrix}$$

Las velocidades del extremo del robot, en función de las aceleraciones de las articulaciones y las posiciones de las mismas.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -q3Sq1Sq2 & q3Cq1Cq2 & Cq1Sq2 \\ q3Cq1Sq2 & q3Sq1Cq2 & Sq1Sq2 \\ 0 & -q3Sq2 & Cq2 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

CINEMÁTICA DE MOVIMIENTO: MATRIZ JACOBIANA

Ejemplo: **Robot esférico 3 GDL**

Supongamos que la posición del robot viene definida por las articulaciones: q_1=0, q_2=90, q_3=1m. El robot tiene una longitud L1=1 m y esta sometido a unas velocidades articulares de q_1'=90º/s, q_2'=0º/s y q_3'=0.5 m/s. Calcular la velocidad del extremo del robot utilizando el Jacobiano.

$$J = \begin{bmatrix} q3Sq1Sq2 & -q3Cq1Cq2 & -cq1Sq2 \\ -q3Cq1Sq2 & -q3Sq1Cq2 & -sq1Sq2 \\ 0 & -q3Sq2 & Cq2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Velocidad del extremo del robot será:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \pi/2 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ \pi/2 \\ 0 \end{bmatrix} m/s$$

Cálculo Numérico del Jacobiano

- Para robots complejos:
  - Tedioso calcular manualmente el Jacobiano
- Alternativa:
  - Cálculo numérico: diferenciación numérica
- El Jacobiano (e un robot de n gdl es:

Cada columna es la variación con respecto a una misma articulación (ángulo)

$$J(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_n} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_n} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_n} \end{bmatrix}$$

posición

$$\mathbf{x}_p = (x, y, z)$$

$$\mathbf{x}_p = \mathbf{f}(q_1, \cdots, q_n)$$

$$J(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial \boldsymbol{x}_p}{\partial q_1} & \frac{\partial \boldsymbol{x}_p}{\partial q_2} & \cdots & \frac{\partial \boldsymbol{x}_p}{\partial q_n} \end{bmatrix}$$

$$\frac{\partial \mathbf{x}_p}{\partial q_i} \approx \frac{\Delta \mathbf{x}_p}{\Delta q_i} = \frac{\mathbf{f}(q_1, \cdots, q_i + \Delta q_i, \cdots, q_n) - \mathbf{f}(q_1, \cdots, q_n)}{\Delta q_i}$$

Incremento solo para la articulación q_i

Cálculo Numérico del Jacobiano

- **Ejemplo**

Considerar la siguiente cinemática directa (robot R-R con $l_1 = l_2 = 1$):

$$f(\mathbf{q}) = \begin{bmatrix} f_x(\mathbf{q}) \\ f_y(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \cos(q_1) + \cos(q_1 + q_2) \\ \sin(q_1) + \sin(q_1 + q_2) \end{bmatrix}$$

Calcular el Jacobiano numéricamente cuando $q_1 = 0.5$, $q_2 = 1.0$

Solución

- El Jacobiano es:

$$J(\mathbf{q}) = \begin{bmatrix} \dfrac{\partial f_x}{\partial q_1} & \dfrac{\partial f_x}{\partial q_2} \\ \dfrac{\partial f_y}{\partial q_1} & \dfrac{\partial f_y}{\partial q_2} \end{bmatrix}$$

- Derivadas con respecto a q_1 (primera columna):

$$\frac{\partial f_x}{\partial q_1} = \frac{f_x(q_1 + \Delta q_1, q_2) - f_x(q_1, q_2)}{\Delta q_1} = \frac{\left[\cos(q_1 + \Delta q_1) + \cos((q_1 + \Delta q_1) + q_2)\right] - \left(\cos(q_1) + \cos(q_1 + q_2)\right)}{\Delta q_1}$$

$$\frac{\partial f_y}{\partial q_1} = \frac{f_y(q_1 + \Delta q_1, q_2) - f_y(q_1, q_2)}{\Delta q_1} = \frac{\left[\mathrm{sen}(q_1 + \Delta q_1) + \mathrm{sen}((q_1 + \Delta q_1) + q_2)\right] - \left(\mathrm{sen}(q_1) + \mathrm{sen}(q_1 + q_2)\right)}{\Delta q_1}$$

Cálculo Numérico del Jacobiano

- Ejemplo

Considerar la siguiente cinemática directa (robot R-R con $l_1 = l_2 = 1$):

$$f(\mathbf{q}) = \begin{bmatrix} f_x(\mathbf{q}) \\ f_y(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} \cos(q_1) + \cos(q_1 + q_2) \\ \sin(q_1) + \sin(q_1 + q_2) \end{bmatrix}$$

Calcular el Jacobiano numéricamente cuando $q_1 = 0.5$, $q_2 = 1.0$

Solución

- El Jacobiano es:

$$J(\mathbf{q}) = \begin{bmatrix} \dfrac{\partial f_x}{\partial q_1} & \dfrac{\partial f_x}{\partial q_2} \\ \dfrac{\partial f_y}{\partial q_1} & \dfrac{\partial f_y}{\partial q_2} \end{bmatrix}$$

- Derivadas con respecto a q_2 (segunda columna):

$$\frac{\partial f_x}{\partial q_2} = \frac{f_x(q_1, q_2 + \Delta q_2) - f_x(q_1, q_2)}{\Delta q_2} = \frac{\left[\cos(q_1) + \cos(q_1 + (q_2 + \Delta q_2))\right] - \left(\cos(q_1) + \cos(q_1 + q_2)\right)}{\Delta q_2}$$

$$\frac{\partial f_y}{\partial q_2} = \frac{f_y(q_1, q_2 + \Delta q_2) - f_y(q_1, q_2)}{\Delta q_2} = \frac{\left[\operatorname{sen}(q_1) + \operatorname{sen}(q_1 + (q_2 + \Delta q_2))\right] - \left(\operatorname{sen}(q_1) + \operatorname{sen}(q_1 + q_2)\right)}{\Delta q_1}$$

Cálculo Numérico del Jacobiano

- Ejemplo

- Usando $\Delta q_1 = \Delta q_2 = 0.001$:

$$\frac{\partial f_x}{\partial q_1} = \frac{\left(\cos(0.5+0.001)+\cos(0.5+0.001+1.0)\right)-\left(\cos(0.5)+\cos(0.5+1.0)\right)}{0.001} = -1.4774$$

$$\frac{\partial f_y}{\partial q_1} = \frac{\left(\operatorname{sen}(0.5+0.001)+\operatorname{sen}(0.5+0.001+1.0)\right)-\left(\operatorname{sen}(0.5)+\operatorname{sen}(0.5+1.0)\right)}{0.001} = 0.9476$$

$$\frac{\partial f_x}{\partial q_2} = \frac{\left(\cos(0.5)+\cos(0.5+1.0+0.001)\right)-\left(\cos(0.5)+\cos(0.5+1.0)\right)}{0.001} = -0.9975$$

$$\frac{\partial f_y}{\partial q_2} = \frac{\left(\operatorname{sen}(0.5)+\operatorname{sen}(0.5+1.0+0.001)\right)-\left(\operatorname{sen}(0.5)+\operatorname{sen}(0.5+1.0)\right)}{0.001} = 0.0702$$

- Jacobiano usando cálculo numérico:

$$J(\mathbf{q}) = \begin{bmatrix} -1.4774 & -0.9975 \\ 0.9476 & 0.0702 \end{bmatrix}$$

Jacobiano usando el enfoque analítico (para comparación)

$$J(\mathbf{q}) = \begin{bmatrix} -1.4769 & -0.9975 \\ 0.9483 & 0.0707 \end{bmatrix}$$

Valores similares (errores de redondeo)

Cálculo Numérico del Jacobiano

- Ejemplo

Computacionalmente (usando Python):

- Función de cinemática directa

```
def fkine(q):
    x = np.cos(q[0]) + np.cos(q[0]+q[1]);
    y = np.sin(q[0]) + np.sin(q[0]+q[1]);
    return np.array([x,y])
```

- Cálculo del Jacobiano

```
q1 = 0.5; q2 = 1.0
delta = 0.001

JT = 1/delta*np.array([
          fkine([q1+delta, q2]) - fkine([q1,q2]),
          fkine([q1, q2+delta]) - fkine([q1,q2])])
J = JT.transpose()
print(J)
```

Conclusiones

- La cinemática inversa encuentra configuraciones articulares dada una posición/orientación deseada
- En cinemática inversa puede existir una solución, varias soluciones, infinitas soluciones o ninguna solución (existencia de soluciones definida por el espacio de trabajo)
- Las soluciones analíticas son más complejas, menos sistemáticas y aplicables a casos simples
- Las soluciones numéricas son más genéricas y son aplicables en todos los casos

Referencias

- B. Siciliano, L. Sciavicco, L. Villani, y G. Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010 (*Sección 2.2*)

- M.W. Spong, S. Hutchinson, y M. Vidyasagar. *Robot Modeling and Control.* John Wiley & Sons, 2006 (*Secciones 1.4, 3.3*)

- K. Lynch and F. Park. Modern Robotics: *Mechanics, Planning, and Control.* Cambridge University Press, 2017 (*Secciones* 6.1-6.2)

Apéndice 1: Método de Newton

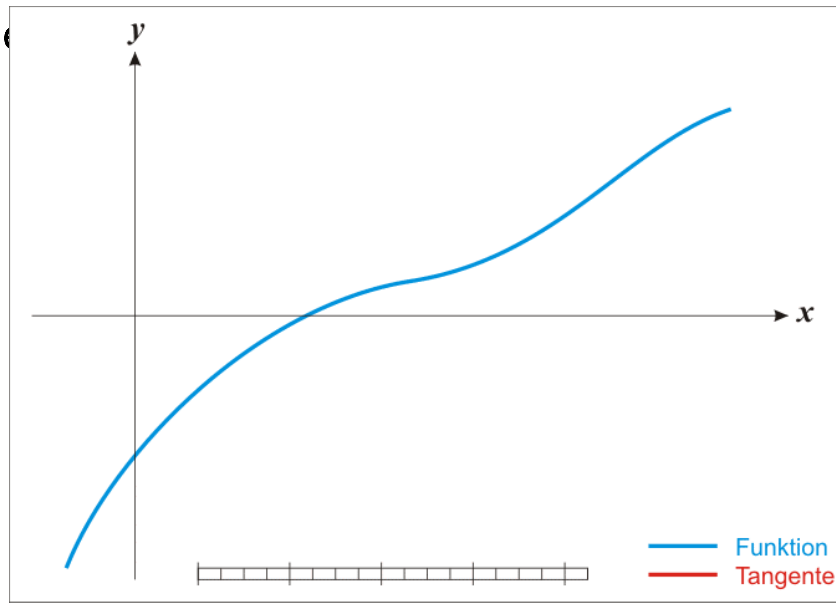
- Objetivo:
  - Encontrar el punto donde la función se hace cero
  - Equivalenteme  $y=f(x)$ con eje x, esto es: $f(x) =$

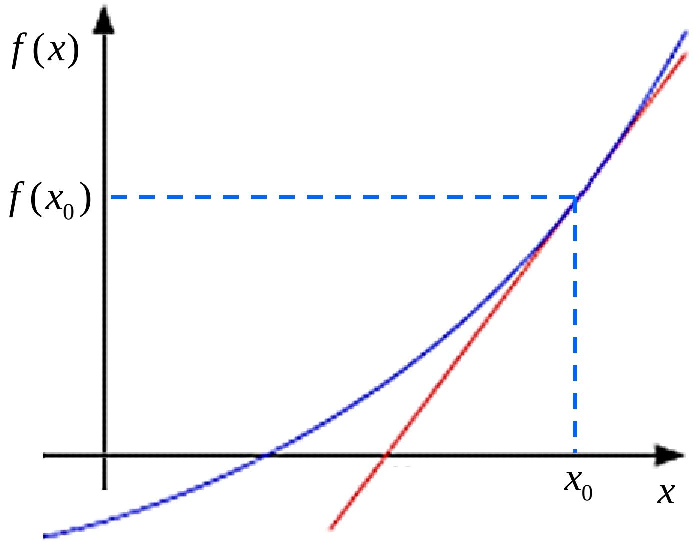
Imagen de: https://en.wikipedia.org/wiki/Newton%27s_method

Idea

Apéndice 1: Método de Newton

- En forma escalar (1-D):
  - Ecuación de la recta tangente en x_0

$$f(x) - f(x_0) = f'(x_0)(x - x_0)$$



  - Se desea la intersección con cero: $f(x) = 0$

$$-f(x_0) = f'(x_0)(x - x_0)$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

  - En general: $\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}$

Aplicar iterativamente $\Rightarrow$ $x_0, x_1, x_2, x_3, \cdots$

- Alternativamente: expansión de Taylor

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \underbrace{o\left(\| x - x_0 \|^2\right)}_{\text{Términos de orden superior}}$$

Descartando los términos de orden superior se obtiene la ecuación de la recta

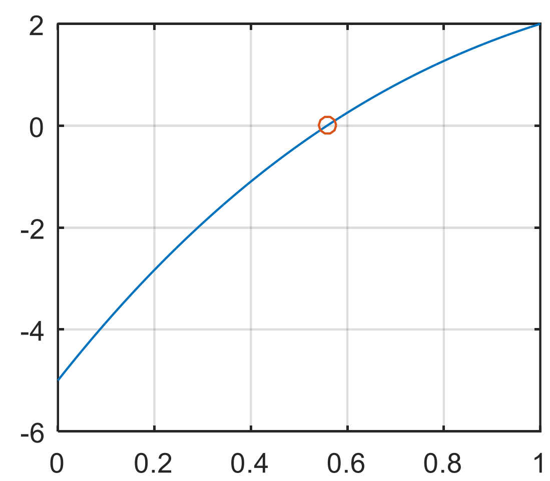
Apéndice 1: Método de Newton

- Ejemplo:

Encontrar la intersección con cero para $f(x) = 3 + (x\text{-}2)^3$ usando el método de Newton y 10 iteraciones

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \Rightarrow x_{k+1} = x_k - \frac{3+(x_k-2)^3}{3(x_k-2)^2}$$

```
% Valor inicial (cualquier valor)
x0 = 0;
% Iteraciones del método de Newton
for i=1:10
   x0 = x0 - (3+(x0-2)^3)/(3*(x0-2)^2);
end
disp(['Intersección en: ' num2str(x0)])

% Gráfico de la función
x = 0:0.001:1;
y = 3 + (x-2).^3;
plot(x,y), hold on, plot(x0,0,'o'), grid on
```



Usando el programa se encuentra la intersección en: 0.55775

Apéndice 2: Matlab

a) Método de Newton

- Ejemplo: robot RR (con matlab)

```
xd = [1.2; 1.2]; % Valor deseado en el espacio Cartesiano
q  = [0.5; 0.5]; % Valor Inicial en el espacio articular
epsilon = 1e-3;
max_iter = 100;  % Máximo número de iteraciones

% Iteraciones del método de Newton
for i=1:max_iter
    q1=q(1); q2=q(2);
    Jinv = 1/sin(q2)*[cos(q1+q2), sin(q1+q2);
                      -cos(q1)-cos(q1+q2), -sin(q1)-sin(q1+q2)];
    f = [cos(q1)+cos(q1+q2); sin(q1)+sin(q1+q2)];
    e = xd-f;
    q = q + Jinv*e;
    % Condición de término
    if (norm(e)<epsilon)
        break;
    end
end
disp(q) % Muestra el valor obtenido
```

Apéndice 2: Matlab

b) Método del Gradiente

- Ejemplo: robot RR (en Matlab)

```
xd = [1.2; 1.2];      % Valor deseado en el espacio Cartesiano
q  = [0; 0];          % Valor Inicial en el espacio articular
epsilon = 1e-3;
max_iter = 1000;      % Máximo número de iteraciones
alpha = 0.5;
% Iteraciones del método del gradiente
for i=1:max_iter
    q1=q(1); q2=q(2);
    JT = [-sin(q1)-sin(q1+q2), cos(q1)+cos(q1+q2);
                 -sin(q1+q2),          cos(q1+q2)];
    f = [cos(q1)+cos(q1+q2); sin(q1)+sin(q1+q2)];
    e = xd-f;
    q = q + alpha*JT*e;
    % Condición de término
    if (norm(e)<epsilon)
        break;
    end
end
disp(q)
```

Apéndice 2: Matlab

c) Cálculo numérico del Jacobiano

- **Ejemplo**

- Función de cinemática directa

```
function X = rr_fkine(q)

x = cos(q(1)) + cos(q(1)+q(2));
y = sin(q(1)) + sin(q(1)+q(2));

X=[x;y];
```

- Cálculo del Jacobiano

```
q1=0.5; q2=1.0;
J = 1/delta*[ rr_fkine([q1+delta,q2])-rr_fkine([q1,q2]),
rr_fkine([q1,q2+delta])-rr_fkine([q1,q2]) ]
```

Study material

Chapter 4 - Craig’s book