Modelado Dinámico

CONTROL Y PROGRAMACIÓN DE ROBOTS

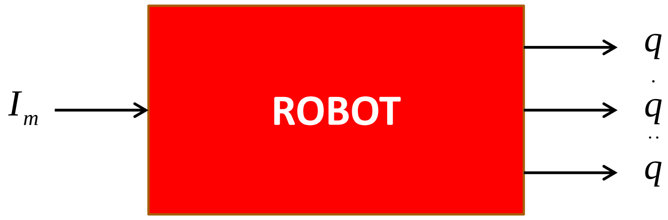
Grado en Electrónica, Robótica y Mecatrónica

Índice

1. Objetivo para el trabajo de curso
2. Simulación del robot
3. Modelo basado en objeto *robot* (con *Robotics Toolbox*)
4. Ecuaciones para el modelado
5. Implementación del modelo
6. Linealización del modelo

Objetivo para trabajo de curso

Simulador para control de un robot de N g.d.l.:



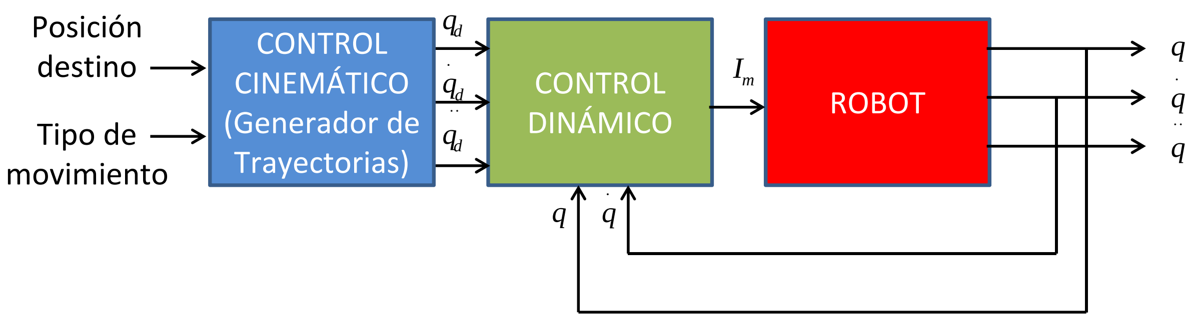
$$\boxed{K_t R I_m = \left(M(q) + J_m R^2\right)\ddot{q} + \left(C(q,\dot{q}) + B_m R^2\right)\dot{q} + G(q) + F(\dot{q})}$$

$$\boxed{K_t I_m / R = \left(M(q)/R^2 + J_m\right)\ddot{q} + \left(C(q,\dot{q})/R^2 + B_m\right)\dot{q} + G(q)/R^2 + F(\dot{q})/R^2}$$

Objetivo para trabajo de curso

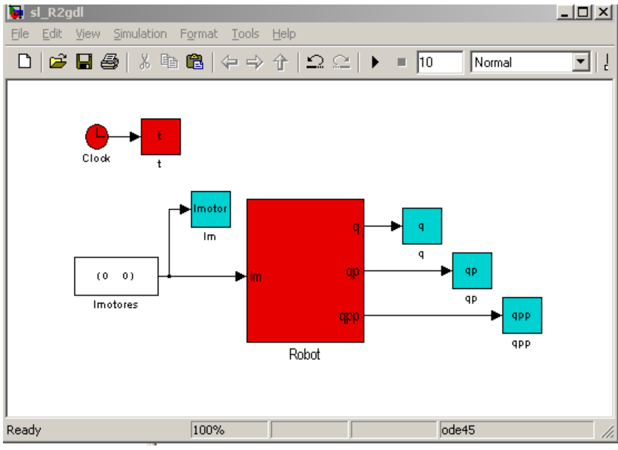
Esquema general de control:

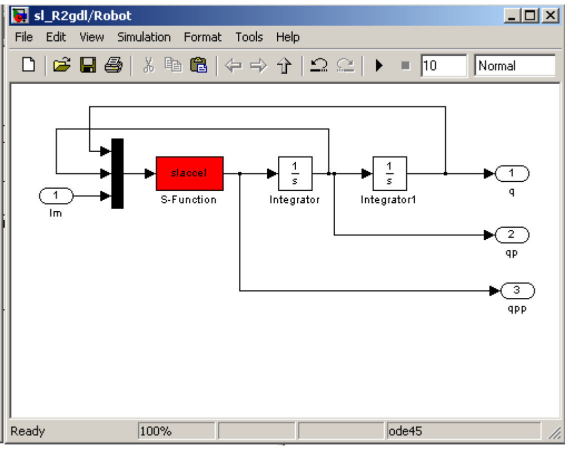
- Simulación de comandos como MOVE, MOVEL, MOVED, …
- Entorno de simulación en ***Matlab*** y ***Simulink***.



Simulación del robot

Sin ecuaciones analíticas:



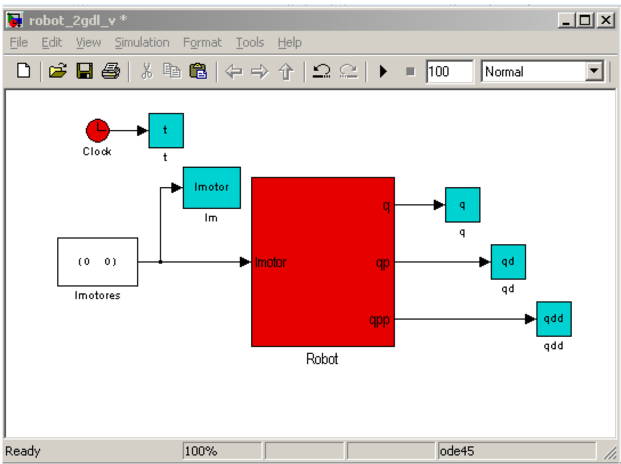


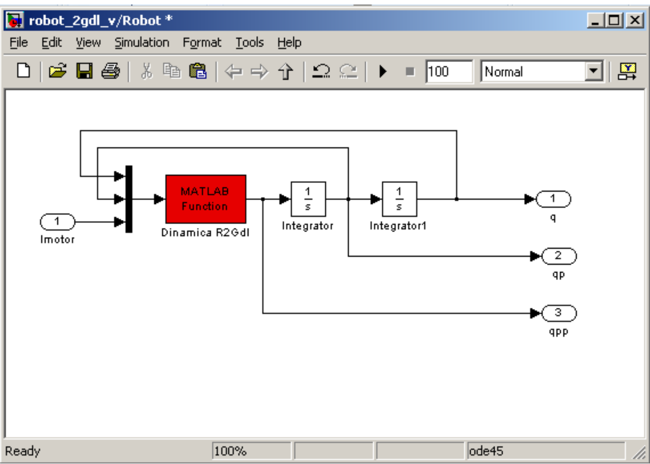
Basada en la *S-function* **slaccel** de *Robotics Toolbox*
(ejecución lenta)

Condiciones iniciales en los integradores

Simulación del robot

Con ecuaciones analíticas:





Basada en una ***M-function*** creada por el alumno
(ejecución más rápida)

Condiciones iniciales en los integradores

Otra posibilidad es crear una S-function, compilada en C

Modelo basado en objeto *robot*

- Uso del *Robotis Toolbox* para crear un objeto robot
- Información cinemática de eslabones y articulaciones:
  - Basada en parámetros de *Denavit-Hartengerb*

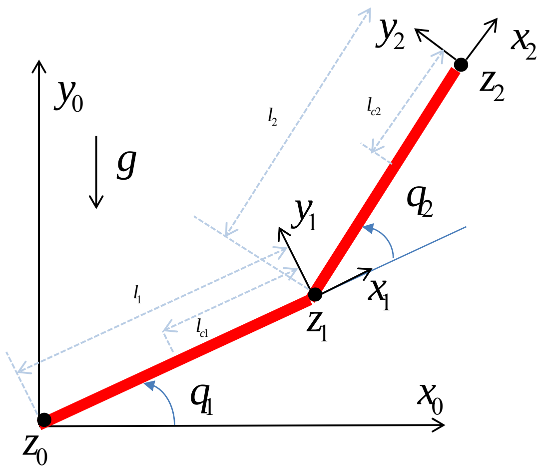
$$X_i = Z_i \times Z_{i-1}$$

- Información dinámica:
  - Eslabones: masas, centro de masas, tensores de inercia, …
  - Actuadores: inercias, fricciones, factores de reducción, …
  - Simulador recibe “pares en las articulaciones” (no en motores)

Cuidado con el **modelo dinámico** y la **orientación de la base** cuando el eje **Z0 no está en vertical**

Modelo basado en objeto *robot*

- Ejemplo: Robot plano RR en vertical



$${}^{i-1}\mathbf{A}_i = \mathbf{Rotz}(\theta_i)\,\mathbf{T}(0,0,d_i)\,\mathbf{T}(a_i,0,0)\,\mathbf{Rotx}(\alpha_i)$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \qquad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

Eslabones

$m_1 = m_2 = 3\,Kg \qquad l_1 = l_2 = 1\,m$

$I_1 = I_2 = 0.2536\ Kg.m^2 \qquad l_{c1} = l_{c2} = 0.5\ m$

Motores

$J_{m1} = J_{m2} = 0.025\ Kg.m^2$

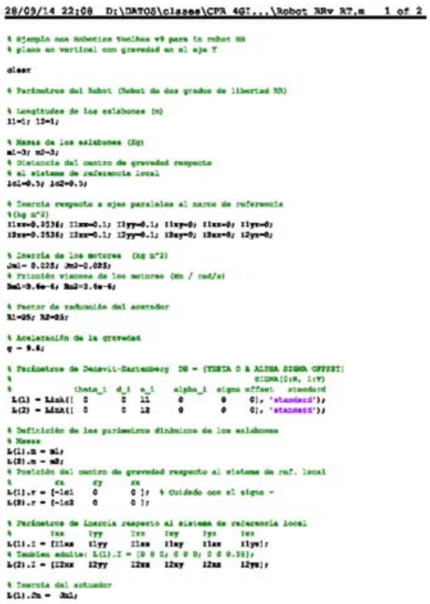
$B_{m1} = B_{m2} = 3.6 \times 10^{-6}\ Nm/(rad/s)$

$K_{t1} = K_{t2} = 10\ Nm/A$

$R_1 = R_2 = 25 \quad vs \quad R_1 = R_2 = 1$

Modelo basado en objeto *robot*

◦ Ejemplo: Robot plano RR en vertical



```
28/09/14 22:08  D:\DATOS\clases\CPR 4GI...\Robot RRv RT.m   1 of 2

% Ejemplo con Robotics Toolbox v9 para in robot RR
% plano en vertical con gravedad en el eje Y

clear

% Parámetros del Robot (Robot de dos grados de libertad RR)

% Longitudes de los eslabones (m)
l1=1; l2=1;

% Masas de los eslabones (Kg)
m1=3; m2=3;
% Distancia del centro de gravedad respecto
% al sistema de referencia local
lc1=0.5; lc2=0.5;

% Inercia respecto a ejes paralelos al marco de referencia
%(kg m^2)
I1zz=0.2536; I1xx=0.1; I1yy=0.1; I1xy=0; I1xz=0; I1yz=0;
I2zz=0.2536; I2xx=0.1; I2yy=0.1; I2xy=0; I2xz=0; I2yz=0;

% Inercia de los motores  (kg m^2)
Jm1= 0.025; Jm2=0.025;
% Fricción viscosa de los motores (Nm / rad/s)
Bm1=3.6e-4; Bm2=3.6e-4;

% Factor de reducción del acoplador
R1=25; R2=25;

% Aceleración de la gravedad
g = 9.8;

% Parámetros de Denavit-Hartenberg  DH = [THETA D A ALPHA SIGMA OFFSET]
%                                                   SIGMA(0:R, 1:P)
%              theta_i  d_i  a_i    alpha_i  sigma offset   standard
 L(1) = Link([  0        0   l1      0        0     0], 'standard');
 L(2) = Link([  0        0   l2      0        0     0], 'standard');

% Definición de los parámetros dinámicos de los eslabones
% Masas
L(1).m = m1;
L(2).m = m2;
% Posición del centro de gravedad respecto al sistema de ref. local
%          cx      cy      cz
L(1).r = [-lc1    0       0 ];  % Cuidado con el signo -
L(2).r = [-lc2    0       0 ];

% Parámetros de inercia respecto al sistema de referencia local
%         Ixx     Iyy     Izz    Ixy     Iyz     Ixz
L(1).I = [I1xx   I1yy    I1zz   I1xy    I1xz    I1yz];
% Tambien admite: L(1).I = [0 0 0; 0 0 0; 0 0 0.35];
L(2).I = [I2xx   I2yy    I2zz   I2xy    I2xz    I2yz];

% Inercia del actuador
L(1).Jm =  Jm1;
```

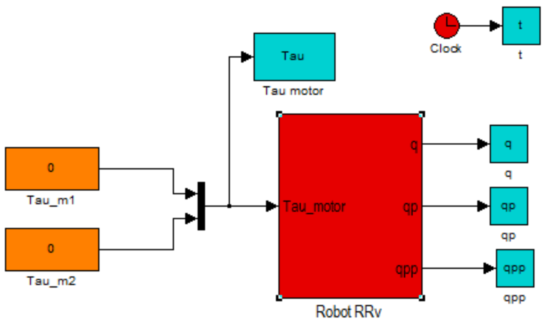
El modelo se puede hacer con variables simbólicas (utilizando *Symbolic Toolbox* de *Matlab*), pero la ejecución en Simulink es extremadamente lenta.

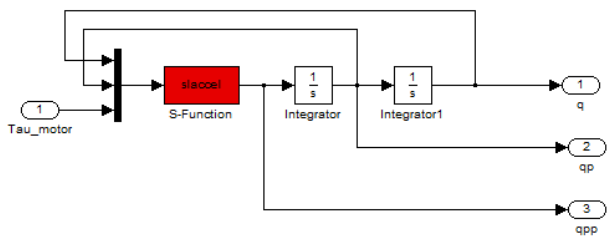
Si se recomienda el uso de variables simbólicas con *Robotics Toolbox* para el cálculo de la *cinemática directa*.

```
syms x y z q1 q2 real
...
Robot.fkine([q1 q2])
```


Modelo basado en objeto *robot*

◦ Ejemplo: Robot plano RR en vertical





Basada en la *S-function* **slaccel** de *Robotics Toolbox*
(ejecución lenta)

Condiciones iniciales en los integradores

Ecuaciones para el modelado

Formulación de Lagrange
- Basado en balance energético

$$\begin{cases} \mathsf{L} = E_c - E_p \\ \tau_i = \dfrac{\mathrm{d}}{\mathrm{dt}} \dfrac{\partial \mathsf{L}}{\partial \dot{q}_\mathrm{i}} - \dfrac{\partial \mathsf{L}}{\partial q_\mathrm{i}} \end{cases}$$

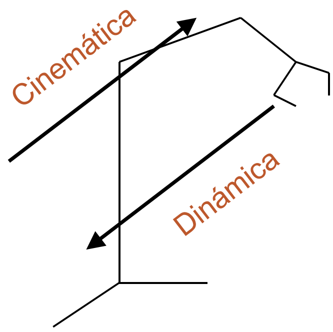
Formulación de Newton-Euler
- Basado en la segunda ley de Newton
- Además de ecuaciones de movimiento, aporta esfuerzos en las articulaciones

$$\begin{cases} \sum \mathbf{F} = \dfrac{d}{dt}(\mathrm{m}\,\mathbf{v}) \\ \sum \mathbf{T} = \dfrac{d}{dt}(\mathbf{I}\boldsymbol{\omega}) = \mathbf{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \left(\mathbf{I}\,\boldsymbol{\omega}\right) \end{cases}$$

Obtención de ecuaciones analíticas con variables simbólicas (*Symbolic Toolbox*)

Ecuaciones para el modelado

- Algoritmo de Newton-Euler:



Basado en la 2ª ley de Newton ΣF=ma

Es un procedimiento recursivo en el que para

i=1,..,n se calcula:

w_i (Velocidades angular del sistema $\{S_i\}$: qi´)

$w_i´$ (Aceleraciones angular del sistema $\{S_i\}$: qi”)

$v_i´, a_i$ (Aceleraciones lineales del sistema $\{S_i\}$ y del centro de masa de la barra i)

i=n,..1 se calcula:

F_i (Fuerza ejercida en el centro de masa de la barra i)

N_i (Par ejercido en el centro de masa de la barra i)

f_i (Fuerza ejercida sobre la articulación i-1, unión barra i-1 con barra i, expresada en {i-1})

n_i (Par ejercido sobre la articulación i-1, unión barra i-1 con barra i,expresada en {i-1})

Ecuaciones para el modelado

- Pasos del algoritmo de Newton-Euler:
  - 1.- Asignar a cada eslabón un sistema de referencia de acuerdo con las normas de D-H
  - 2.- Establecer las condiciones iniciales:

Base del robot

$^{0}\boldsymbol{\omega}_0$: velocidad angular=$[0,0,0]^T$

$^{0}\dot{\boldsymbol{\omega}}_0$: aceleracion angular=$[0,0,0]^T$

$^{0}\mathbf{v}_0$: velocidad lineal= $[0,0,0]^T$

$^{0}\dot{\mathbf{v}}_0$: aceleracion lineal= $-[g_{x0}, g_{y0}, g_{z0}]^T$

Extremo del robot

Para el extremo del robot se conocerá la fuerza y el par ejercidos externamente $^{n+1}\mathbf{f}_{n+1}$ y $^{n+1}\mathbf{n}_{n+1}$

$^{0}\boldsymbol{\omega}_0, {}^{0}\dot{\boldsymbol{\omega}}_0 \; y \; {}^{0}\mathbf{v}_0$ son típicamente nulos salvo que la base del robot esté en movimiento

$[g_{x0}, g_{y0}, g_{z0}]$ es el vector de gravedad expresado en el sistema $\{S_0\}$ (habitualmente toma el valor [0, 0, -9.8] pues z_0 se sitúa vertical hacia arriba).

Ecuaciones para el modelado

- Pasos del algoritmo de Newton-Euler:
  - 2.- Establecer las condiciones iniciales (continuación):

Configuración del robot

$\mathbf{z}_0 = [0,0,1]^T$

$^{i}\mathbf{p}_i$ = Vector que une el origen de $\{S_{i-1}\}$ con el de $\{S_i\}$ expresadas en $\{S_i\} = [a_i, d_i\,\text{sen}(\alpha_i), d_i\cos(\alpha_i)]$

$^{i}\mathbf{s}_i$ = Coordenadas del centro de masas del eslabon i respecto del sistema $\{S_i\}$

$^{i}\mathbf{I}_i$ = Matriz de inercia del eslabón i expresado en un sistema paralelo al $\{S_i\}$ y con el origen en el centro de masas del eslabón.

  - 3.- Obtener las matrices de rotación $^{i-1}R_i$ y sus inversas

$$^{i}\mathbf{R}_{i-1} = \left({}^{i-1}\mathbf{R}_i\right)^{-1} = \left({}^{i-1}\mathbf{R}_i\right)^{T} \qquad {}^{i-1}\mathbf{R}_i = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i \\ 0 & S\alpha_i & C\alpha_i \end{bmatrix}$$

Ecuaciones para el modelado

- Pasos del algoritmo de Newton-Euler:

Desde i=1 hasta n (iteración cinamética)

- 4.- Obtener la velocidad angular del sistema $\{S_i\}$:

$$
{}^{i}\omega_i = \begin{cases} {}^{i}\mathbf{R}_{i-1}\left({}^{i-1}\omega_{i-1} + \mathbf{z}_0\,\dot{q}_i\right) & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\omega_{i-1} & \text{si el eslabón i es de traslación} \end{cases}
$$

- 5.- Obtener la aceleración angular del sistema $\{S_i\}$

$$
{}^{i}\dot{\omega}_i = \begin{cases} {}^{i}\mathbf{R}_{i-1}\left({}^{i-1}\dot{\omega}_{i-1} + \mathbf{z}_0\,\ddot{q}_i + {}^{i-1}\omega_{i-1}\times\mathbf{z}_0\,\dot{q}_i\right) & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\dot{\omega}_{i-1} & \text{si el eslabón i es de traslación} \end{cases}
$$

- 6.- Obtener la aceleración lineal del sistema $\{S_i\}$

$$
{}^{i}\dot{\mathbf{v}}_i = \begin{cases} {}^{i}\dot{\omega}_i\times{}^{i}\mathbf{p}_i + {}^{i}\omega_i\times\left({}^{i}\omega_i\times{}^{i}\mathbf{p}_i\right) + {}^{i}\mathbf{R}_{i-1}\,{}^{i-1}\dot{\mathbf{v}}_{i-1} & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{R}_{i-1}\left(\mathbf{z}_0\,\ddot{q}_i + {}^{i-1}\dot{\mathbf{v}}_{i-1}\right) + {}^{i}\dot{\omega}_i\times{}^{i}\mathbf{p}_i + 2\,{}^{i}\omega_i\times{}^{i}\mathbf{R}_{i-1}\mathbf{z}_0\,\dot{q}_i + & \\ \qquad + {}^{i}\omega_i\times\left({}^{i}\omega_i\times{}^{i}\mathbf{p}_i\right) & \text{si el eslabón i es de traslación} \end{cases}
$$

Ecuaciones para el modelado

- Pasos del algoritmo de Newton-Euler:
  - 7.- Obtener la aceleración lineal del centro de gravedad del eslabón i:

$$
{}^{i}\mathbf{a}_i = {}^{i}\dot{\omega}_i\times{}^{i}\mathbf{s}_i + {}^{i}\omega_i\times\left({}^{i}\omega_i\times{}^{i}\mathbf{s}_i\right) + {}^{i}\dot{\mathbf{v}}_i
$$

Fin iteración cinemática

Desde i=n hasta 1 (iteración dinámica)

- 8.- Obtener la fuerza ejercida sobre el eslabón i

$$
{}^{i}\mathbf{f}_i = {}^{i}\mathbf{R}_{i+1}\,{}^{i+1}\mathbf{f}_{i+1} + m_i\,{}^{i}\mathbf{a}_i
$$

- 9.- Obtener el par ejercido sobre el eslabón i

$$
{}^{i}\mathbf{n}_i = {}^{i}\mathbf{R}_{i+1}\left[{}^{i+1}\mathbf{n}_{i+1} + \left({}^{i+1}\mathbf{R}_i\,{}^{i}\mathbf{p}_i\right)\times{}^{i+1}\mathbf{f}_{i+1}\right] + \left({}^{i}\mathbf{p}_i + {}^{i}\mathbf{s}_i\right)\times m_i\,{}^{i}\mathbf{a}_i + {}^{i}\mathbf{I}_i\,{}^{i}\dot{\omega}_i + {}^{i}\omega_i\times\left({}^{i}\mathbf{I}_i\,{}^{i}\omega_i\right)
$$

- 10.- Obtener la fuerza o par aplicado sobre la articulación

$$
\tau_i = \begin{cases} {}^{i}\mathbf{n}_i^T\,{}^{i}\mathbf{R}_{i-1}\mathbf{z}_0 & \text{si el eslabón i es de rotación} \\ {}^{i}\mathbf{f}_i^T\,{}^{i}\mathbf{R}_{i-1}\mathbf{z}_0 & \text{si el eslabón i es de traslación} \end{cases}
$$

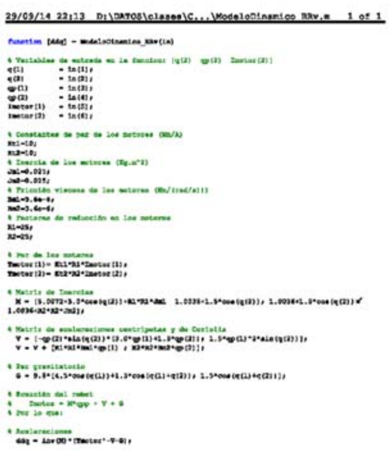
Fin iteración dinámica

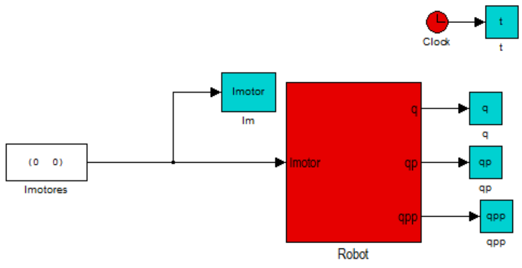
Ecuaciones para el modelado

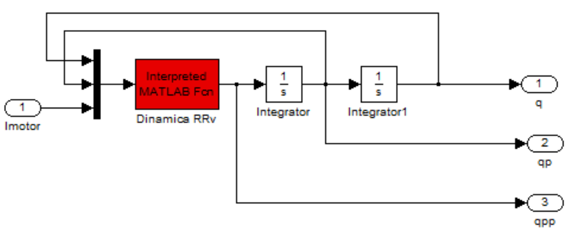
- Ejemplo: Robot plano RR en vertical

Implementación del modelo

- Utilizar en Simulink una función .m (*MATLAB function*) para simular modelo dinámico directo (similar a *slaccel*):

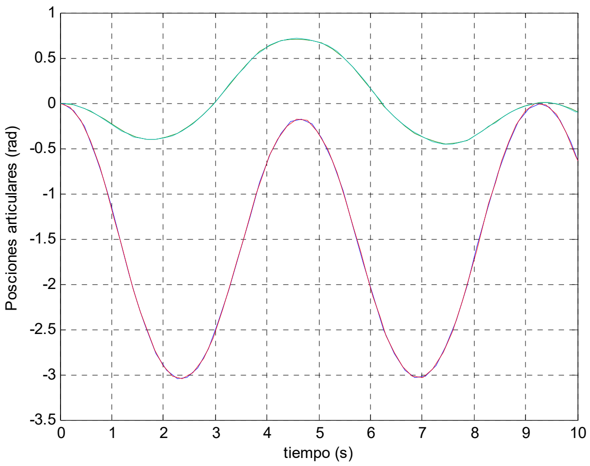


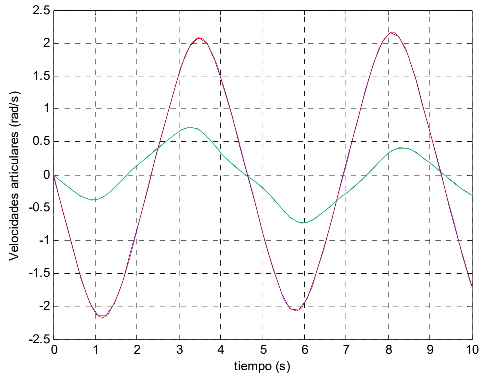


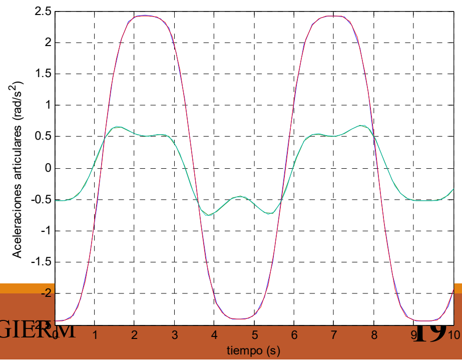


Implementación del modelo

- Comparar resultados entre los proporcionados por el *Robotics Toolbox* y los implementados en *MATLAB Function*







Cuidado con las comparativas:
- Señales de control distintas (pares vs. Intensidades)
- Condiciones iniciales en los integradores

Linealización de modelos

- Ecuación dinámica del robot:

$$\tau = RK_t I_m = \left(M(q) + R^2 J_m\right)\ddot{q} + \left(C(q,\dot{q}) + R^2 B_m\right)\dot{q} + G(q) + F(q,\dot{q})$$

$$\boxed{I_m = \underbrace{(RK_t)^{-1}\left(M(q) + R^2 J_m\right)}_{M_A(q)}\ddot{q} + \underbrace{(RK_t)^{-1}\left(C(q,\dot{q}) + R^2 B_m\right)}_{V_A(q,\dot{q})}\dot{q} + \underbrace{(RK_t)^{-1} G(q)}_{G_A(q)} + \underbrace{(RK_t)^{-1} F(q,\dot{q})}_{F_A(q,\dot{q})}}$$

- Consideraciones para la linealización:
  - **Linealizar en torno a velocidades nulas:**
    - Velocidades de equilibrio:

$$\dot{q}_{eq} = 0 \ \textit{unidades velocidad articular}$$

$$\dot{q} = \dot{q}_{eq} + \Delta\dot{q} = \Delta\dot{q}$$

    - Aceleraciones de equilibrio:

$$\ddot{q}_{eq} = 0 \ \textit{unidades aceleración articular}$$

$$\ddot{q} = \ddot{q}_{eq} + \Delta\ddot{q} = \Delta\ddot{q}$$

Linealización de modelos

- **Linealización de términos de la matriz de inercia junto con la aceleración:**
  - $$M_A(q)\ddot{q} = M_A(q_{eq})\underbrace{\ddot{q}_{eq}}_{=0} + \underbrace{\ddot{q}_{eq}}_{=0}\left.\frac{\partial M_A(q)}{\partial q}\right|_{q=q_{eq}} \Delta q + M_A(q_{eq})\Delta\ddot{q} + \ldots \approx M_A(q_{eq})\Delta\ddot{q}$$
  - Despreciar términos no diagonales de la matriz de inercia frente a los diagonales:
    $$M_A(q_{eq}) \approx diag\left(M_A(q_{eq})\right)$$
  - Especialmente cierto cuando los factores de reducción son altos (R>>1)
  - Elección valores de inercia máximos de los términos de la diagonal de $M_A(q)$
  - Con estas suposiciones se está desacoplando el sistema (a nivel de modelo teórico)
    $$G_{ij}(s) = \frac{\Delta q_i(s)}{\Delta \tau_j(s)} \qquad G_{ij}(s) = 0 \quad si \quad i \neq j$$

Linealización de modelos

- **Linealización de términos centrípetos y de Coriolis:**
  - Términos de Coriolis en el equilibrio:
    $$C(q, \dot{q}_{eq}) = C(q, 0) = 0$$
  - No hay términos lineales aportados por $V(q,\dot{q}) = C(q,\dot{q})\dot{q}$

en la linealización, ya que es cuadrático con la velocidad articular:

$$V(q,\dot{q}) = C(q,\dot{q})\dot{q} = \underbrace{C(q_{eq},\dot{q}_{eq})}_{=0}\underbrace{\dot{q}_{eq}}_{=0} +$$

$$+\left(\frac{\partial C(q,\dot{q})}{\partial q}\Delta q + \frac{\partial C(q,\dot{q})}{\partial \dot{q}}\Delta\dot{q}\right)\underbrace{\dot{q}_{eq}}_{=0} + \underbrace{C(q_{eq},\dot{q}_{eq})}_{=0}\Delta q + \ldots \approx 0$$

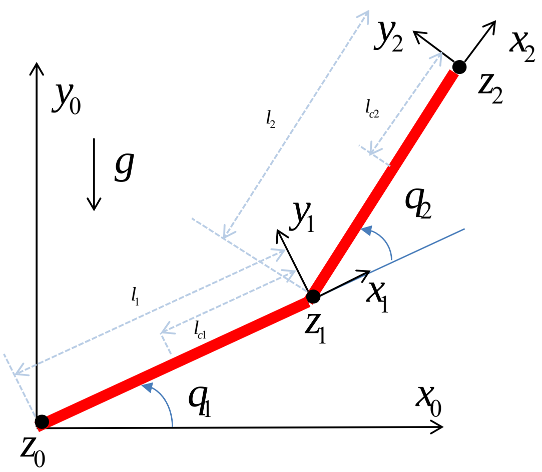
- Los términos de la fricción viscosa ($R^2 B_m \dot{q} = R^2 B_m \dot{q}_{eq} + R^2 B_m \Delta\dot{q}$) sí aporta términos en la linealización.

Linealización de modelos

- **Consideraciones sobre los términos gravitatorios:**
  - En técnicas de control básico, no se suele linealizar en torno a ninguna posición en particular.
  - Los términos gravitatorios se desprecian en los modelos incrementales, y se consideran desde el punto de vista de control como perturbaciones mantenidas a la entrada.
  - La magnitud de estas perturbaciones decrece a medida que las relaciones de las reductoras son mayores (los términos gravitatorios no se magnifican con las reductoras por las reductoras).
  - Posibilidad de compensarlos con un *Feedforward*.
- **Consideraciones sobre los términos de fricción:**
  - Se desprecian en la linealización: $F(q,\dot{q}) \approx 0$
  - Posibilidad de compensarlos con Feedforward o estructura interna con Feedback.

Linealización de modelos

- Ejemplo: Robot RR vertical:



$$^{i-1}\mathbf{A}_i = \mathbf{Rotz}(\theta_i)\,\mathbf{T}(0,0,d_i)\,\mathbf{T}(a_i,0,0)\,\mathbf{Rotx}(\alpha_i)$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \qquad q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

Eslabones

$$m_1 = m_2 = 3\,Kg \qquad l_1 = l_2 = 1\,m$$

$$I_1 = I_2 = 0.2536\ Kg.m^2 \qquad l_{c1} = l_{c2} = 0.5\ m$$

Motores

$$J_{m1} = J_{m2} = 0.025\ Kg.m^2$$

$$B_{m1} = B_{m2} = 3.6\times10^{-6}\ Nm/(rad/s)$$

$$R_1 = R_2 = 25 \quad vs \quad R_1 = R_2 = 1$$

Linealización del modelo

- Ejemplo: Robot plano RR en vertical

$$K_t R I_m = \left(M(q) + J_m R^2\right)\ddot{q} + \left(C(q,\dot{q}) + B_m R^2\right)\dot{q} + G(q) + F(\dot{q})$$

Brazo mecánico:

$$M(q) = \begin{bmatrix} 5.0072 + 3.0\cos(q_2) & 1.0036 + 1.5\cos(q_2) \\ 1.0036 + 1.5\cos(q_2) & 1.0036 \end{bmatrix}$$

$$G(q) = g\begin{bmatrix} 4.5\cos(q_1) + 1.5\cos(q_1 + q_2) \\ 1.5\cos(q_1 + q_2) \end{bmatrix}$$

$$C(q,\dot{q}) = 1.5\sin(q_2)\begin{bmatrix} -2\dot{q}_2 & -\dot{q}_2 \\ \dot{q}_1 & 0 \end{bmatrix}$$

Actuadores:

$$K_t R = \begin{bmatrix} 250 & 0 \\ 0 & 250 \end{bmatrix} \qquad R = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix}$$

$$J_m R^2 = \begin{bmatrix} 15.625 & 0 \\ 0 & 15.625 \end{bmatrix} \qquad B_m R^2 = \begin{bmatrix} 0.0022 & 0 \\ 0 & 0.0022 \end{bmatrix} \qquad K_t = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

Linealización del modelo

- Ejemplo: Robot plano RR en vertical

$$K_t R I_m = \left(M(q) + J_m R^2\right)\ddot{q} + \left(C(q,\dot{q}) + B_m R^2\right)\dot{q} + G(q) + F(\dot{q})$$

$R_1 = R_2 = 25$

$$\begin{bmatrix} 250 & 0 \\ 0 & 250 \end{bmatrix}\begin{bmatrix} I_{m1} \\ I_{m2} \end{bmatrix} = \begin{bmatrix} 20.6322 + 3.0\cos(q_2) & 1.0036 + 1.5\cos(q_2) \\ 1.0036 + 1.5\cos(q_2) & 16.6286 \end{bmatrix} * \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} -1.5\sin(q_2)\left(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2\right) + 0.0022\dot{q}_1 \\ -1.5\sin(q_2)\dot{q}_1^2 + 0.0022\dot{q}_2 \end{bmatrix} + \begin{bmatrix} 44.1\cos(q_2) + 14.7\cos(q_1 + q_2) \\ 14.7\cos(q_1 + q_2) \end{bmatrix}$$

$R_1 = R_2 = 1$

$$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}\begin{bmatrix} I_{m1} \\ I_{m2} \end{bmatrix} = \begin{bmatrix} 5.0322 + 3.0\cos(q_2) & 1.0036 + 1.5\cos(q_2) \\ 1.0036 + 1.5\cos(q_2) & 1.0286 \end{bmatrix} * \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$+ \begin{bmatrix} -1.5\sin(q_2)\left(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2\right) + 3.6\times10^{-6}\dot{q}_1 \\ -1.5\sin(q_2)\dot{q}_1^2 + 3.6\times10^{-6}\dot{q}_2 \end{bmatrix} + \begin{bmatrix} 44.1\cos(q_2) + 14.7\cos(q_1 + q_2) \\ 14.7\cos(q_1 + q_2) \end{bmatrix}$$

Linealización del modelo

- Ejemplo: Robot plano RR en vertical

Modelos lineales (en torno a velocidades bajas):

$$\frac{q_1(s)}{I_{m1}(s)} \approx \frac{250}{(23.6322s + 0.0022)s}$$
$$\frac{q_2(s)}{I_{m2}(s)} \approx \frac{250}{(16.6286 + 0.0022)s}$$

$R_1 = R_2 = 25$

APROXIMACIÓN POR SISTEMAS DE SEGUNDO ORDEN Y DE TIPO 1

$$\frac{q_1(s)}{I_{m1}(s)} \approx \frac{10}{(8.0322s + 3.6 \times 10^{-6})s}$$
$$\frac{q_2(s)}{I_{m2}(s)} \approx \frac{10}{(1.0286s + 3.6 \times 10^{-6})s}$$

$R_1 = R_2 = 1$

Resumen "Symbolic Toolbox"

- syms
  - >> syms q1 q2 qn real
- PI = sym('pi');
  - >> sin(PI), sin(pi)
- pretty();
  - >> x=sin(q1)^2+cos(q1)^2
  - >> pretty(x)
- eval();
  - >> q1 = pi/4;
  - >> x
  - >> eval(x)
- simplify();
  - >> x=sin(qn)^2+cos(qn)^2
  - >> y=simplify(x)
- simple();
  - >> y=simple(x)
- vpa();
  - >> z=x/3
  - >> r=vpa(z,4)
- collect();
  - >> x=q2*cos(qn)+q2*sin(qn)
  - >> y = collect(x,q2)
- diff();
  - >> y=diff(x,q2)
  - >> z=diff(x,qn)
- **>> help symbolic**