Robótica Espacial
Aula prática nº 7
Dinâmica de Manipuladores

Vitor Santos

Universidade de Aveiro

27 mar 2025

Exercício 1 - RRR planar com parâmetros cinemáticos e dinâmicos

- Definir o seguinte robô RRR planar:
- Comprimento dos elos (m):
  - `L1=1; L2=0.8; L3=0.6;`
- Massas dos elos (kg):
  - `m1=2; m2=1.5; m3=1.0;`
  - `masses = [m1,m2,m3];`
- Matrizes de inércia de 3x3 dos elos ($kg\,m^2$):
  - `I(:,:,1)=diag([0.1, 0.1, 0.2 ]);`
  - `I(:,:,2)=diag([0.05,0.05,0.1 ]);`
  - `I(:,:,3)=diag([0.02,0.02,0.05]);`
- Vetor da gravidade:
  - `g=[0 -10 0]';` (m/s^2)
- Os centros de massa são no centro geométrico dos elos:
  - `comLocal = [L1/2 0 0;L2/2 0 0;L3/2 0 0]';`

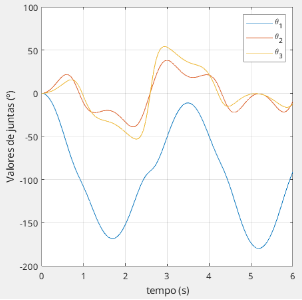
Depois de criar o robô, inicializar os parâmetros dinâmicos:

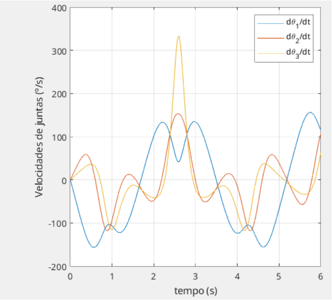
```
for i = 1:robot.n
 robot.links(i).m = masses(i);
 robot.links(i).r = comLocal(:,i);
 robot.links(i).I = I(:,:,i);
end
robot.gravity=g;
```

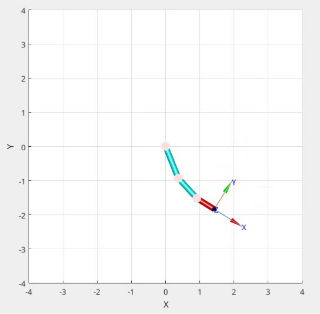
Verificar os parâmetros com o comando `robot.dyn`

Exercício 2 - Simulação de Dinâmica Direta

- No robô anterior, com a função `robot.fdyn()`, simular a resposta de ação da gravidade, durante 6 segundos (`tdur`), partindo de `q_init=[0 0 0]` e velocidade inicial nula.
- Isto é equivalente a não provocar qualquer momento nos atuadores e deixar atuar apenas a ação da gravidade!
- Sugestão de execução: `robot.fdyn(tdur,[],q_init,qd_init);`
- Obter as curvas ilustradas e a animação







External player

Equações e definições principais para a dinâmica

Formulação de Euler-Lagrange para a dinâmica inversa

$$\tau_i = \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i}$$

$$P = \sum_{i=1}^{N} m_i h_i g = -\sum_{i=1}^{N} \vec{g}^{\mathsf{T}} \vec{r}_{C_{0,i}} m_i$$

τ_i	Força ou momento (*torque*) atuante sobre o elo i
$\mathcal{L}$	Lagrangiano do sistema ($\mathcal{L} = K - P$)
q_i	Variável de junta associada ao elo (linear ou angular)
$\dot{q}_i$	Velocidade (linear ou angular) da junta.
m_i	Massa de um elo i no sistema
h_i	Altura do centro de massa do elo j face à referência
$g, \vec{g}$	Aceleração (vetor) da gravidade
$\vec{r}_{C_{0,i}}$	Vetor do centro de massa do elo i no referencial da base

Expressão genérica da energia cinética

$$K = \sum_{j}^{N} \left(K_{L_j} + K_{R_j}\right) = $$

$$= \sum_{j}^{N} \left(\frac{1}{2} m_j \vec{v}_j^{\mathsf{T}} \vec{v}_j + \frac{1}{2} \vec{\omega}_j^{\mathsf{T}} \vec{\vec{I}}_j \vec{\omega}_j\right)$$

K_{L_j}	Energia cinética linear do elo j (do seu CM)
K_{R_j}	Energia cinética rotacional do elo j
$\vec{v}_j$	Velocidade linear do elo j
$\vec{\omega}_j$	Velocidade angular do elo j
$\vec{\vec{I}}_j$	Tensor de inércia do elo j: $\vec{\vec{I}}_j = \begin{bmatrix} I_{xx} & & \\ & I_{yy} & \\ & & I_{zz} \end{bmatrix}$

Conceitos e definições envolvidos no estudo da dinâmica

Objetivo Calcular os momentos de juntas necessários para mover um robô usando a equação de Euler-Lagrange

Conceitos-chave mais importantes

- **K** Energia cinética
- **P** Energia potencial
- $\mathcal{L}$ Lagrangiano do sistema
- τ Forças generalizadas pela Equação de Euler-Lagrange

Conceitos adicionais que são parâmetros a usar

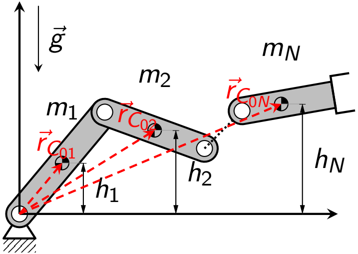
- **Coordenadas Generalizadas (q)** São os ângulos ou posições das juntas que descrevem a configuração do robô.
- **Velocidades (qd)** São as velocidades generalizadas correspondentes a cada grau de liberdade (variável de junta)
- **Vetores do Centro de Massa (CoM) (r)** Descrevem a posição do centro de massa de cada elo no referencial global.
- **Matrizes de Inércia (I)** Descrevem a inércia rotacional de cada elo no espaço 3D.
- **Massas dos elos (m)** As massas dos elos individuais.

Exercício 3 - Função para a energia potencial total de um robô

Completar o código em falta ***

```
function P = potential_energy(robot, q)
% total potential energy of robot for joints
% configuration q
n = robot.n; % Number of joints
P = 0;         % Initialize potential energy
g = robot.gravity'; % gravity vector
r = computeCoM(robot,q); % absolute CoM

% Loop over all links
for i = 1:n
    % Get the mass and CoM of the current link
    m_i = ***;
    r_i = ***;
    % Add the contribution to potential energy
    P = P + ***; % the formula on the right -->
end
```

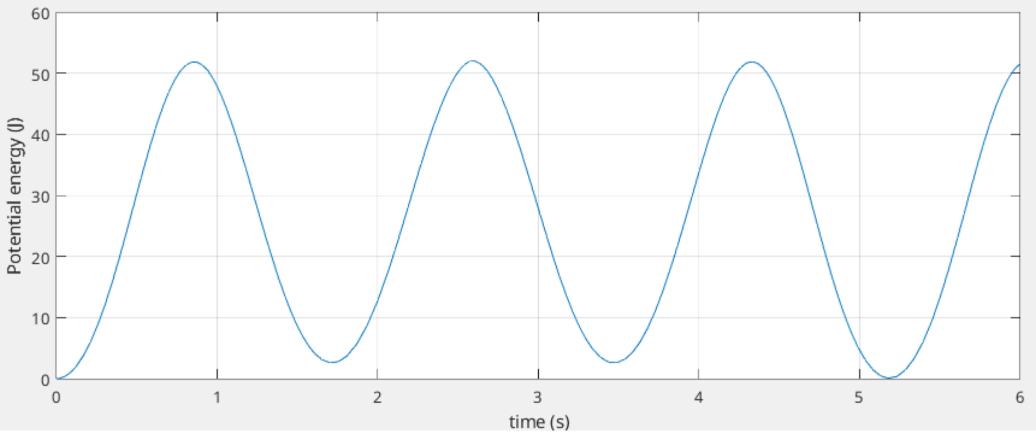


O cálculo simplificado é dado por: $P = \sum_{i=1}^{N} m_i h_i g$ mas a expressão seguinte é universal e por isso é a recomendada para implementar no código:

$$P = -\sum_{i=1}^{N} \vec{g}^{\mathsf{T}} \vec{r}_{C_{0,i}} m_i$$

Exercício 4 - Cálculo da energia potencial

- Para o robô dos exercícios anteriores e usando a função `potential_energy(robot,q)` criada anteriormente, verificar que a curva da energia potencial durante o período de simulação é como a ilustrada de seguida:



Expressão geral da Matriz de Massas e Inércias $M(q)$

Desenvolvimento da expressão genérica de K

Sabendo que, entre outros, $\vec{v}_j^\mathsf{T} = \left(J_{v_j}\dot{q}\right)^\mathsf{T} = \dot{q}^\mathsf{T} J_{v_j}^\mathsf{T}$, virá da expansão da expressão anterior:

$$K = \frac{1}{2}\dot{q}^\mathsf{T}\left[\sum_j^N \left(m_j J_{v_j}^\mathsf{T} J_{v_j} + J_{\omega_j}^\mathsf{T} R_j \vec{\vec{I}}_{L_j} R_j^\mathsf{T} J_{\omega_j}\right)\right]\dot{q} = \frac{1}{2}\dot{q}^\mathsf{T} M(q)\dot{q}$$

$$\dot{r}_j = J_j\dot{q}$$

$$J_j = \begin{bmatrix} J_{v_j} \\ J_{\omega_j} \end{bmatrix}$$

$$\dot{r}_j = \begin{bmatrix} \vec{v}_j \\ \vec{\omega}_j \end{bmatrix} = \begin{bmatrix} J_{v_j} \\ J_{\omega_j} \end{bmatrix}\dot{q}$$

$$\vec{\vec{I}}_j = R_j \vec{\vec{I}}_{L_j} R_j^\mathsf{T}$$

$M(q)$	Matriz de massas (e/ou inércias)[a]
J_j	Jacobiano parcial até à junta j
J_{v_j}	Parte de J_j para as velocidades lineares
J_{ω_j}	Parte de J_j para as velocidades angulares
$\vec{\vec{I}}_j$	Tensor de inércia do elo j (Varia com o movimento)
$\vec{\vec{I}}_{L_j}$	Tensor de inércia local do elo j (fixo)
R_j	Matriz de rotação até à junta j

[a] "Joint Space Mass Matrix" ou "Joint Space Inertia Matrix"

Jacobiano geométrico (lembrete) e Jacobianos parciais

Cálculo do Jacobiano Geométrico (operações vetoriais)

$$J = \begin{bmatrix} J_{v_1} & J_{v_2} & \cdots & J_{v_N} \\ J_{\omega_1} & J_{\omega_2} & \cdots & J_{\omega_N} \end{bmatrix}$$

$$J_{v_i} = \begin{cases} z_{i-1} \times (O_N - O_{i-1}) & \Leftarrow \mathit{rot} \\ z_{i-1} & \Leftarrow \mathit{linear} \end{cases} \qquad J_{\omega_i} = \begin{cases} z_{i-1} & \Leftarrow \mathit{rot} \\ 0 & \Leftarrow \mathit{linear} \end{cases}$$

$z_{i-1} \rightarrow$ vetor do eixo da junta i; $O_{i-1} \rightarrow$ Origem do sistema de eixos da junta i

Jacobiano parcial até à junta j

$$J_j = \begin{bmatrix} \frac{\partial x}{\partial \boldsymbol{q}_1} & \cdots & \frac{\partial x}{\partial \boldsymbol{q}_j} & 0 & \cdots & 0 \\ \frac{\partial y}{\partial \boldsymbol{q}_1} & \cdots & \frac{\partial y}{\partial \boldsymbol{q}_j} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ \frac{\partial \psi}{\partial \boldsymbol{q}_1} & \cdots & \frac{\partial \psi}{\partial \boldsymbol{q}_j} & 0 & \cdots & 0 \end{bmatrix}$$

Formalmente, o jacobiano tem sempre dimensão $6 \times N$ (N juntas).

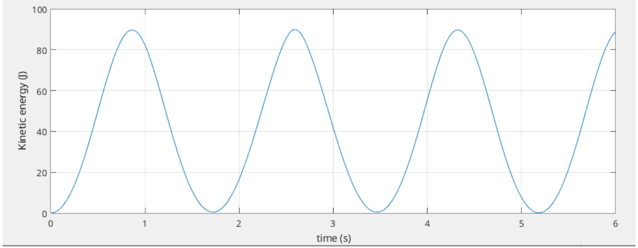
Nas variantes parciais (J_j), as colunas $j+1$ a N são nulas.

Exercício 5 - Calculo da Energia Cinética

- Com base na simulação anterior e nos seus resultados (valores de juntas e de velocidades) e usando a matriz de inércias, estimar a energia cinética usando a fórmula:

$$K = \frac{1}{2}\dot{q}^{\mathsf{T}} M(q)\dot{q}$$

- A matriz de inércias `M(q)` vem do Matlab com `M = robot.inertia(q);`
- As velocidades de juntas resultaram da simulação com `fdyn()`.



A curva segue o mesmo andamento da energia potencial como esperado (porquê?) não obstante o sinal e um fator de escala que se pode dever a algum processo numérico da simulação!

Expressão gerak

$$K = \frac{1}{2}\dot{q}^\intercal \left[\sum_{j}^{N}\left(m_j J_{v_j}^\intercal J_{v_j} + J_{\omega_j}^\intercal R_j \vec{\vec{I}}_{L_j} R_j^\intercal J_{\omega_j}\right)\right]\dot{q} = \frac{1}{2}\dot{q}^\intercal M(q)\dot{q}$$

$$\dot{r}_j = J_j \dot{q}$$

$$J_j = \begin{bmatrix} J_{v_j} \\ J_{\omega_j} \end{bmatrix}$$

$$\dot{r}_j = \begin{bmatrix} \vec{v}_j \\ \vec{\omega}_j \end{bmatrix} = \begin{bmatrix} J_{v_j} \\ J_{\omega_j} \end{bmatrix} \dot{q}$$

$$\vec{\vec{I}}_j = R_j \vec{\vec{I}}_{L_j} R_j^\intercal$$

$M(q)$	Matriz de massas (e/ou inércias)[a]
J_j	Jacobiano parcial até à junta j
J_{v_j}	Parte de J_j para as velocidades lineares
J_{ω_j}	Parte de J_j para as velocidades angulares
$\vec{\vec{I}}_j$	Tensor de inércia do elo j (Varia com o movimento)
$\vec{\vec{I}}_{L_j}$	Tensor de inércia local do elo j (fixo)
R_j	Matriz de rotação até à junta j

[a] "Joint Space Mass Matrix" ou "Joint Space Inertia Matrix"

Expressão geral da dinâmica inversa

$$\vec{\tau} = \mathbf{M}(\vec{\theta})\ddot{\vec{\theta}} + \mathbf{V}(\vec{\theta}, \dot{\vec{\theta}})\dot{\vec{\theta}} + \mathbf{G}(\vec{\theta}) + \mathbf{F}(\vec{\theta}, \dot{\vec{\theta}})$$

- $\vec{\tau}$ - Vector de momentos e forças nos atuadores
- $\mathbf{M}(\vec{\theta})$ - Matriz de massas e inércias
- $\mathbf{V}(\vec{\theta}, \dot{\vec{\theta}})$ - Matriz de Coriolis e centrífugas
- $\mathbf{G}(\vec{\theta})$ - Vetor de termos Gravíticos
- $\mathbf{F}(\vec{\theta}, \dot{\vec{\theta}}$ - Vetor de termos de atrito
- $\vec{\theta}, \dot{\vec{\theta}}, \ddot{\vec{\theta}}$ - variáveis de junta: posição, velocidade, aceleração.

Implementação da dinâmica inversa em Matlab

- $\mathbf{M}(\vec{\theta})$ - Matriz de massas e inércias — `M = robot.inertia(q)`
- $\mathbf{V}(\vec{\theta}, \dot{\vec{\theta}})$ - Matriz de Coriolis e centrífugas — `C = robot.coriolis(q, qd)`
- $\mathbf{G}(\vec{\theta})$ - Vetor de termos Gravíticos — `G = robot.gravload(q)`
- $\vec{\theta}, \dot{\vec{\theta}}, \ddot{\vec{\theta}}$ - variáveis de junta: posição, velocidade, aceleração — `q, qd, qdd`

Dados determinados valores para `q, qd, qdd` desejados pode-se calcular os momentos (forças generalizados) em cada momento com:

```
tau = M*qdd' + C*qd' + G'
```

Ou então, de forma mais direta:

```
tau = robot.rne(q,qd,qdd);
```

Exercício 6 - Uma aplicação concreta do cálculo da dinâmica

- Para o robô RRR criado anteriormente como calcular os momentos para levar as juntas de [0 0 0] até [pi/2 pi/4 pi/8] numa trajetória polinomial das juntas em 2 seguntos?
- Passos da solução:
  - Calcular as trajetórias polinomiais para `q` e os `Dt` (intervalos de tempo) associados
  - Obter numericamente as velocidades das juntas `dq`
  - Obter numericamente as acelerações das juntas `ddq`
  - Aplicar a expressão da dinâmica inversa
- Que passos adicionais haveria, ou quais seriam as diferenças, se se pretendesse descrever uma dada trajetória (linear por exemplo) da ponta do robô?