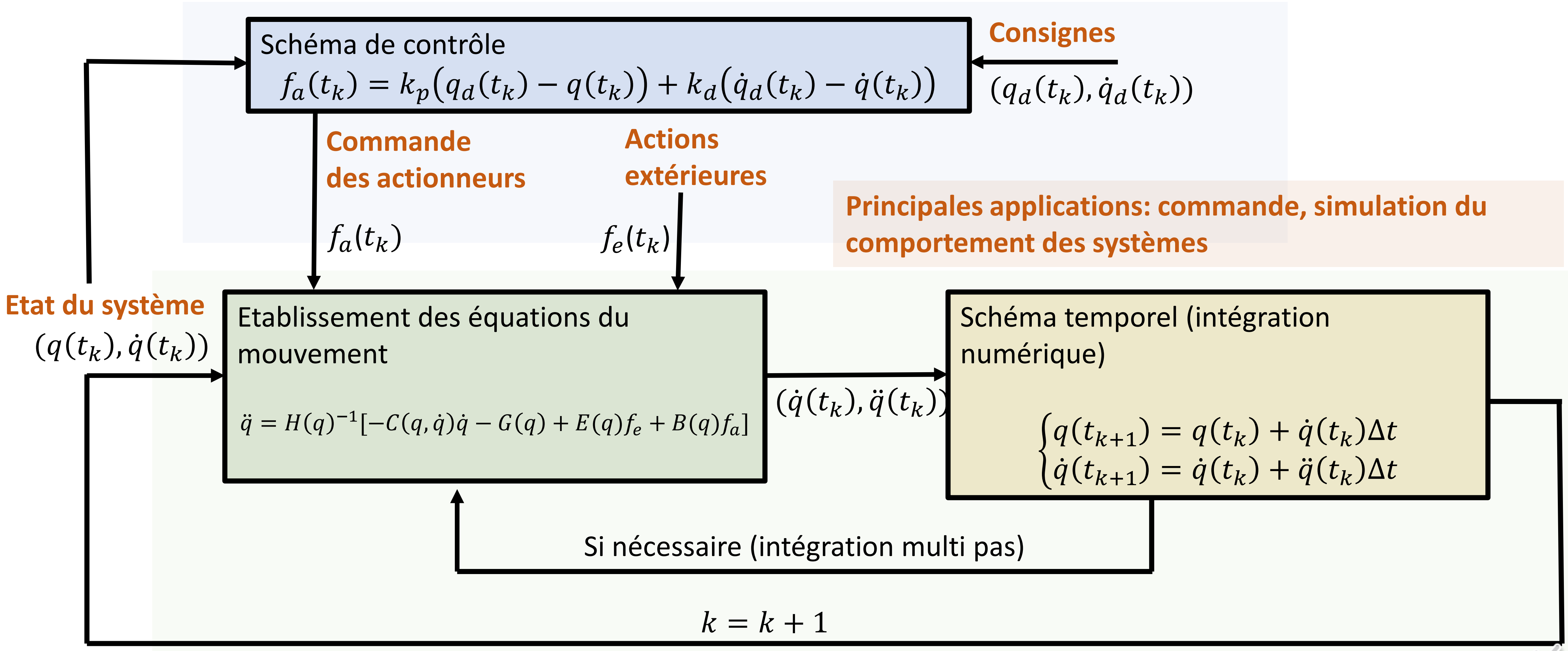
Simulation des systèmes de solides rigides polyarticulés

Dynamique directe

Charles Pontonnier

Rappel du problème



Le problème pour un système arborescent

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + \underbrace{C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + G(\boldsymbol{q}) = E(\boldsymbol{q})\boldsymbol{f_e}} + \underbrace{B(\boldsymbol{q})\boldsymbol{f_a}}$$

Ces termes peuvent être regroupés sous un terme $\boldsymbol{C(q,\dot{q})}$

$B(q)$ est composé uniquement de 1 dans l'immense majorité des cas (liaisons rotoïdes ou prismatiques)

$$\boxed{H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\tau}}$$

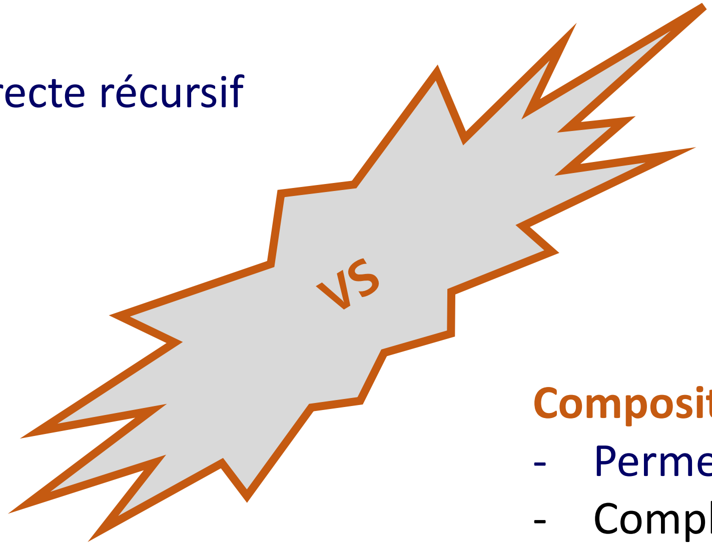
Avec

$H(q)$	matrice de masse
$\ddot{\boldsymbol{q}}$	vecteur des accélérations articulaires
$\boldsymbol{\tau}$	vecteur des actions articulaires (couples moteurs, forces des vérins)
$C(q,\dot{q})$	matrice regroupant les actions centrifuges, de Coriolis et de gravité

2 solutions efficaces pour résoudre ce problème

Articulated–Body Algorithm

- Permet le calcul de dynamique directe récursif
- Complexité en $O(n)$
- Utile pour les systèmes ouverts



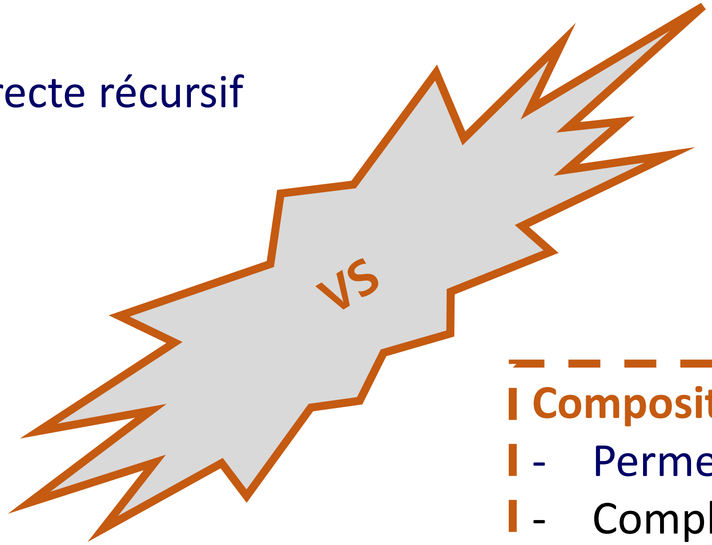
Composite–Rigid–Body Algorithm

- Permet le calcul de $H(q)$
- Complexité en $O(nd)$
- Utile pour les systèmes contraints

2 solutions efficaces pour résoudre ce problème

Articulated–Body Algorithm

- Permet le calcul de dynamique directe récursif
- Complexité en $O(n)$
- Utile pour les systèmes ouverts



Composite–Rigid–Body Algorithm

- Permet le calcul de $H(q)$
- Complexité en $O(nd)$
- Utile pour les systèmes contraints

Trouver $\ddot{q}$ tel que

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}$$

Composite-rigid-body algorithm

Trouver $\ddot{q}$ tel que

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}$$

1. Calculer $\boldsymbol{C}$

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\tau}$$

$\Rightarrow$ C ne dépend que de l'état du système $(q, \dot{q})$

$\Rightarrow$ Si l'on sait calculer la dynamique inverse du système, alors $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{ID}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{0})$

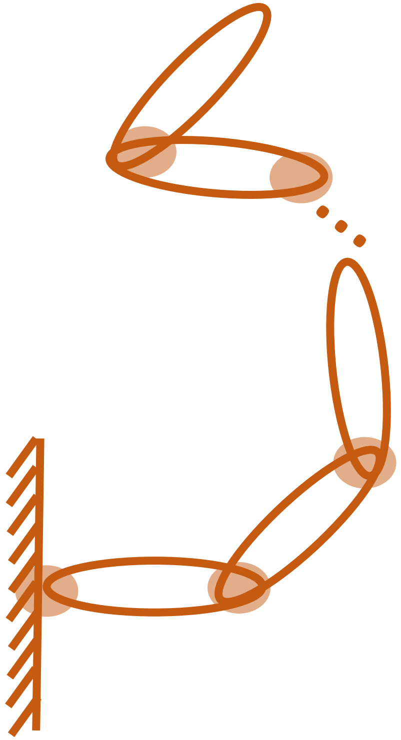
Car $H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{ID}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(q)\ddot{q} + C(q,\dot{q}) = \tau$$

2. Calculer $\boldsymbol{H}$



Pour l'ensemble du système Σ

$$\boldsymbol{T(\Sigma/R_0) = \frac{1}{2}\dot{q}^t H(q)\dot{q} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} H_{ij}(q)\dot{q}_i\dot{q}_j}$$

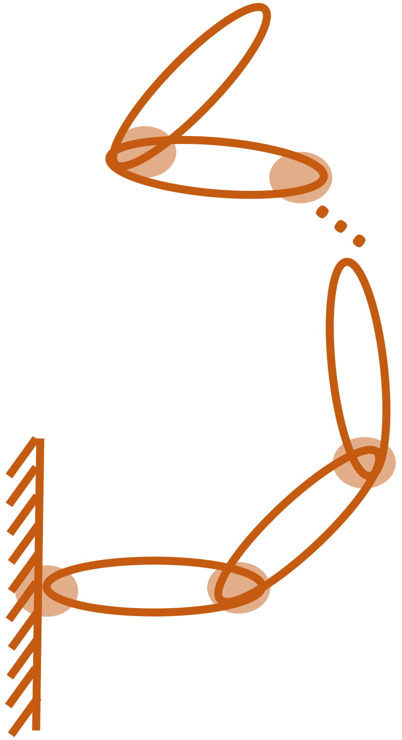
$H(q)$ est alors la matrice de masse généralisée du système Σ

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(q)\ddot{q} + C(q, \dot{q}) = \tau$$

2. Calculer $\boldsymbol{H}$



On peut aussi l'exprimer sous la forme d'un produit des inerties et vitesses spatiales

$$T(\Sigma/R_0) = \sum_{k=1}^{n} \frac{1}{2} \xi_k^t \, I_k^{(S)} \xi_k$$

$I_k^{(S)}$ l'inertie spatiale du solide k qui sera simplement notée I_k ensuite

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\tau}$$

$$\boldsymbol{\xi_k} = \boldsymbol{\xi_{\lambda(k)}} + \begin{bmatrix} \boldsymbol{u_k} \\ \boldsymbol{p_k} \times \boldsymbol{u_k} \end{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{k}} = \sum_{i=1}^{k} \begin{bmatrix} \boldsymbol{u_i} \\ \boldsymbol{p_i} \times \boldsymbol{u_i} \end{bmatrix} \dot{\boldsymbol{q}}_{\boldsymbol{i}} = \sum_{i=1}^{k} \boldsymbol{s_i} \dot{\boldsymbol{q}}_{\boldsymbol{i}}$$

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(q)\ddot{q} + C(q,\dot{q}) = \tau$$

$$\xi_k = \xi_{\lambda(k)} + \begin{bmatrix} u_k \\ p_k \times u_k \end{bmatrix} \dot{q}_k = \sum_{i=1}^{k} \begin{bmatrix} u_i \\ p_i \times u_i \end{bmatrix} \dot{q}_i = \sum_{i=1}^{k} s_i \dot{q}_i$$

Ce qui donne pour l'énergie cinétique

$$T(\Sigma/R_0) = \sum_{k=1}^{n} \frac{1}{2} \left(\sum_{i=1}^{k} s_i \dot{q}_i \right)^t I_k \left(\sum_{j=1}^{k} s_j \dot{q}_j \right)$$

$$T(\Sigma/R_0) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{k} \sum_{j=1}^{k} s_i I_k s_j \dot{q}_i \dot{q}_j$$

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(q)\ddot{q} + C(q,\dot{q}) = \tau$$

$$\xi_k = \xi_{\lambda(k)} + \begin{bmatrix} u_k \\ p_k \times u_k \end{bmatrix} \dot{q}_k = \sum_{i=1}^{k} \begin{bmatrix} u_i \\ p_i \times u_i \end{bmatrix} \dot{q}_i = \sum_{i=1}^{k} s_i \dot{q}_i$$

Ce qui donne pour l'énergie cinétique

$$T(\Sigma/R_0) = \sum_{k=1}^{n} \frac{1}{2} \left(\sum_{i=1}^{k} s_i \dot{q}_i \right)^t I_k \left(\sum_{j=1}^{k} s_j \dot{q}_j \right)$$

$$T(\Sigma/R_0) = \frac{1}{2} \sum_{k=1}^{n} \sum_{i=1}^{k} \sum_{j=1}^{k} s_i I_k s_j \dot{q}_i \dot{q}_j$$

Ce qui peut encore s'écrire

$$T(\Sigma/R_0) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=r}^{n} s_i I_k s_j \dot{q}_i \dot{q}_j$$

Avec $r = \max(i,j)$

Composite rigid-body-algorithm

Trouver $\ddot{q}$ tel que

$$H(q)\ddot{q} + C(q,\dot{q}) = \tau$$

Par identification $$H_{ij} = \sum_{k=r}^{n} s_i I_k s_j$$ Avec $r = \max(i,j)$

Si l'on définit $$I_r^C = \sum_{k=r}^{n} I_k$$ La matrice d'inertie composite des solides r à n

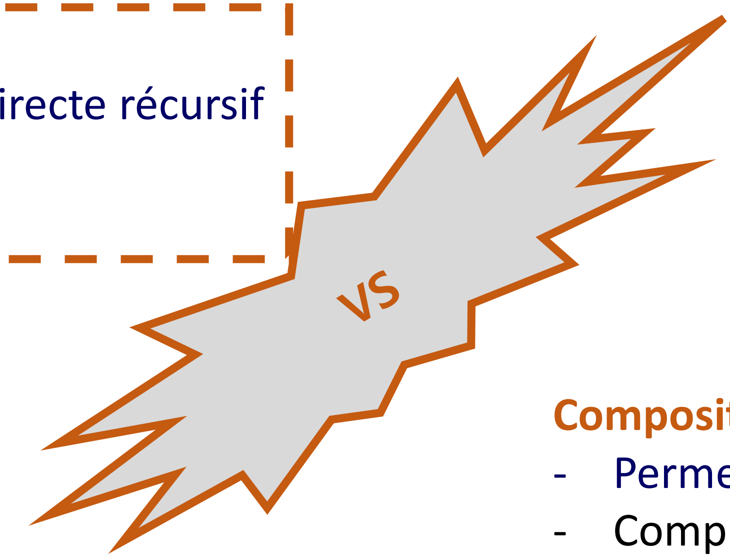
On peut alors calculer H_{ij} en deux étapes:

$I_i^C = I_{i+1}^C + I_i$	(On calcule toutes les inerties composites de n_b à 1)
$H_{ij} = s_i I_r^C s_j$	(On calcule tous les termes de la matrice globale de 1 à n_b)

Et après ? Inverser H…

2 solutions efficaces pour résoudre ce problème

Articulated–Body Algorithm
- Permet le calcul de dynamique directe récursif
- Complexité en $O(n)$
- Utile pour les systèmes ouverts



Composite–Rigid–Body Algorithm
- Permet le calcul de $H(q)$
- Complexité en $O(nd)$
- Utile pour les systèmes contraints

Articulated-body algorithm

Les équations de la dynamique d'un solide S donnent une relation linéaire entre les forces appliquées sur et l'accélération de ce solide.

Ainsi, considérant que l'on applique un effort $\boldsymbol{f}$ sur un solide S faisant partie d'un système de solides rigides Σ, on doit pouvoir exprimer son **équilibre dynamique** sous la forme:

$$\boldsymbol{f} = I^A \dot{\boldsymbol{\xi}} + \boldsymbol{p}^{\boldsymbol{A}}$$

Où $\dot{\boldsymbol{\xi}}$ est **l'accélération spatiale** du solide, I^A la **matrice d'inertie spatiale** que le solide semble avoir, en tenant compte des effets inertiels liés aux autres solides auxquels il est lié, et $\boldsymbol{p}^{\boldsymbol{A}}$ la **force de rappel** (la valeur de $\boldsymbol{f}$ quand le solide n'est pas accéléré).

I^A est appelée l'inertie du solide articulé (**articulated-body inertia**)

Articulated-body algorithm

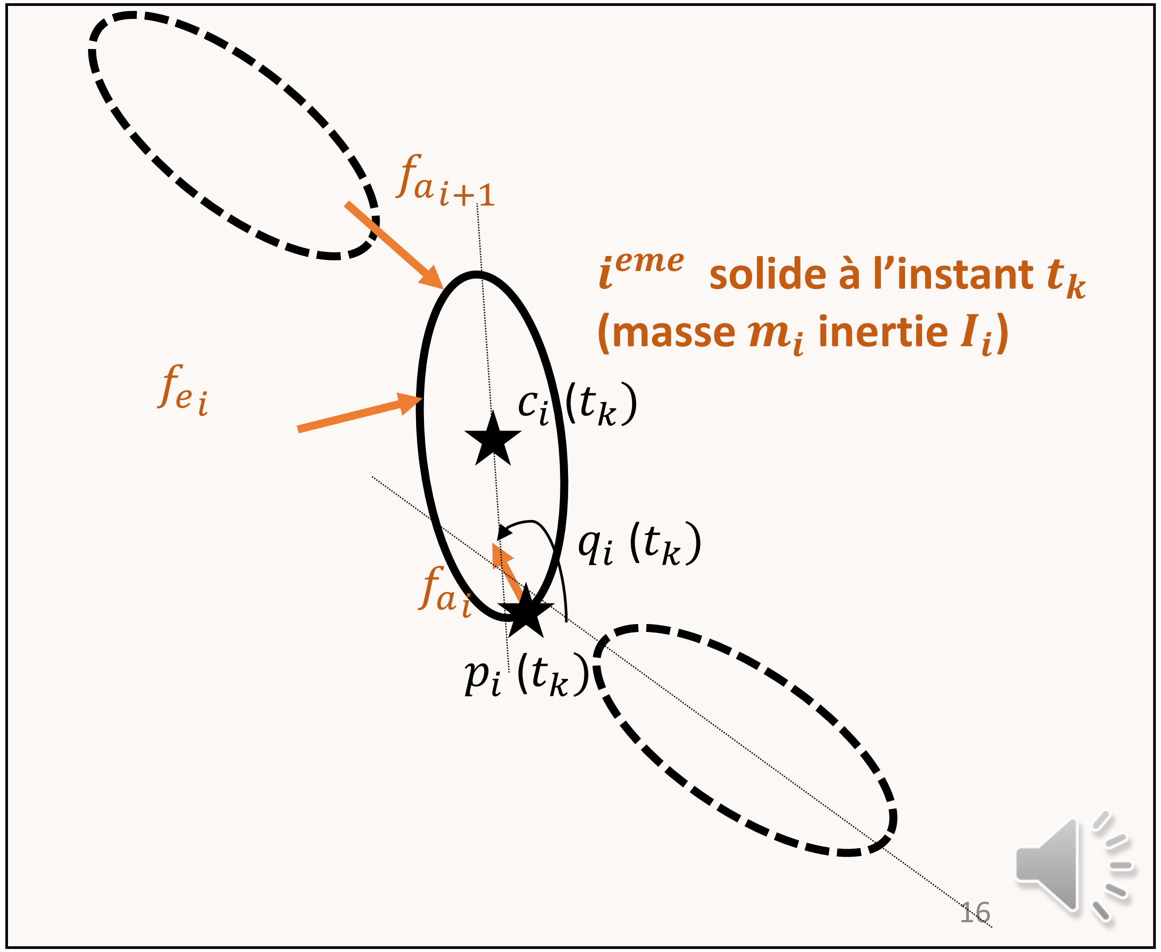
Considérons la chaine constituée d'un solide i et de ses successeurs. Soit $\boldsymbol{f}_{\boldsymbol{a}_i}$ les efforts transmis par la liaison i et $\dot{\xi}_i$ l'accélération du solide i. L'équation du mouvement du solide i s'écrit tout simplement sous la forme (voir Newton-Euler):

$$\boldsymbol{f}_{\boldsymbol{a}_i} = \boldsymbol{I}_i^s\, \dot{\boldsymbol{\xi}}_i + \boldsymbol{\xi}_i \times \boldsymbol{I}_i^s \boldsymbol{\xi}_i - \boldsymbol{f}_{\boldsymbol{e}_i} + \boldsymbol{f}_{\boldsymbol{a}_{i+1}}$$

$$\boldsymbol{f}_{\boldsymbol{a}_i} - \boldsymbol{f}_{\boldsymbol{a}_{i+1}} = \boldsymbol{I}_i^s\, \dot{\boldsymbol{\xi}}_i + \boldsymbol{p}_i$$

Que l'on doit transformer en

$$\boldsymbol{f}_{\boldsymbol{a}_i} = \boldsymbol{I}_i^A\, \dot{\boldsymbol{\xi}}_i + \boldsymbol{p}_i^A$$

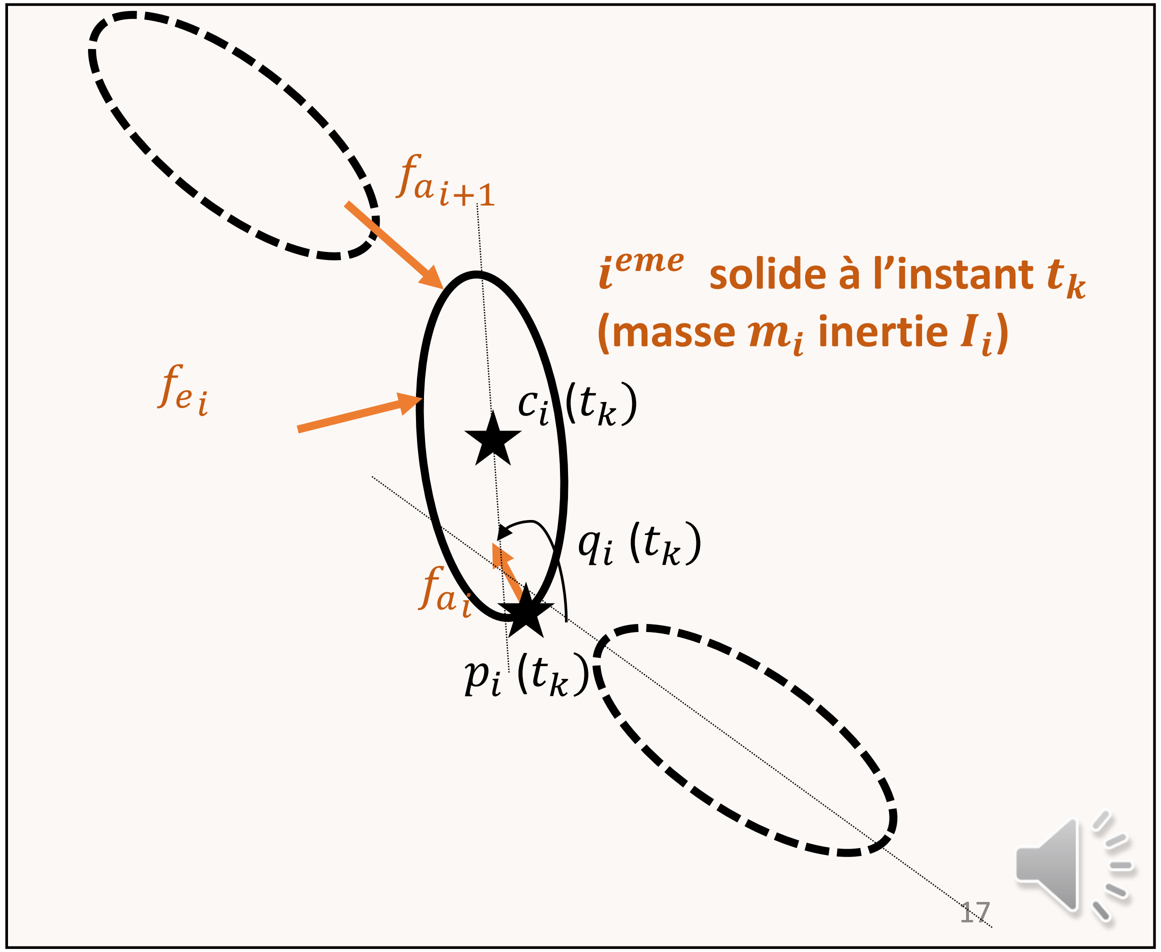


Articulated-body algorithm

L'articulated body algorithm fonctionne en deux étapes:

1. Calculer $\boldsymbol{I}_i^A$ et $\boldsymbol{p}_i^A$ à partir de $\boldsymbol{I}_{i+1}^A$ et $\boldsymbol{p}_{i+1}^A$ en commençant avec $\boldsymbol{I}_n^A = \boldsymbol{I}_n^S$ et $\boldsymbol{p}_n^A = \boldsymbol{p}_n^S$
   (On parcourt l'arbre de n_b à 1)

2. Calculer $\ddot{\boldsymbol{q}}_i$ et $\dot{\boldsymbol{\xi}}_i$ à partir de $\boldsymbol{I}_i^A$ et $\boldsymbol{p}_i^A$ et $\dot{\boldsymbol{\xi}}_{i-1}$
   (On parcourt l'arbre de 1 à n_b)



Articulated-body algorithm

Etape 1

En exploitant les équations suivantes

$$\boldsymbol{f}_{\boldsymbol{a}_i} - \boldsymbol{f}_{\boldsymbol{a}_{i+1}} = \boldsymbol{I}_i^s\ \dot{\boldsymbol{\xi}}_i + \boldsymbol{p}_i$$

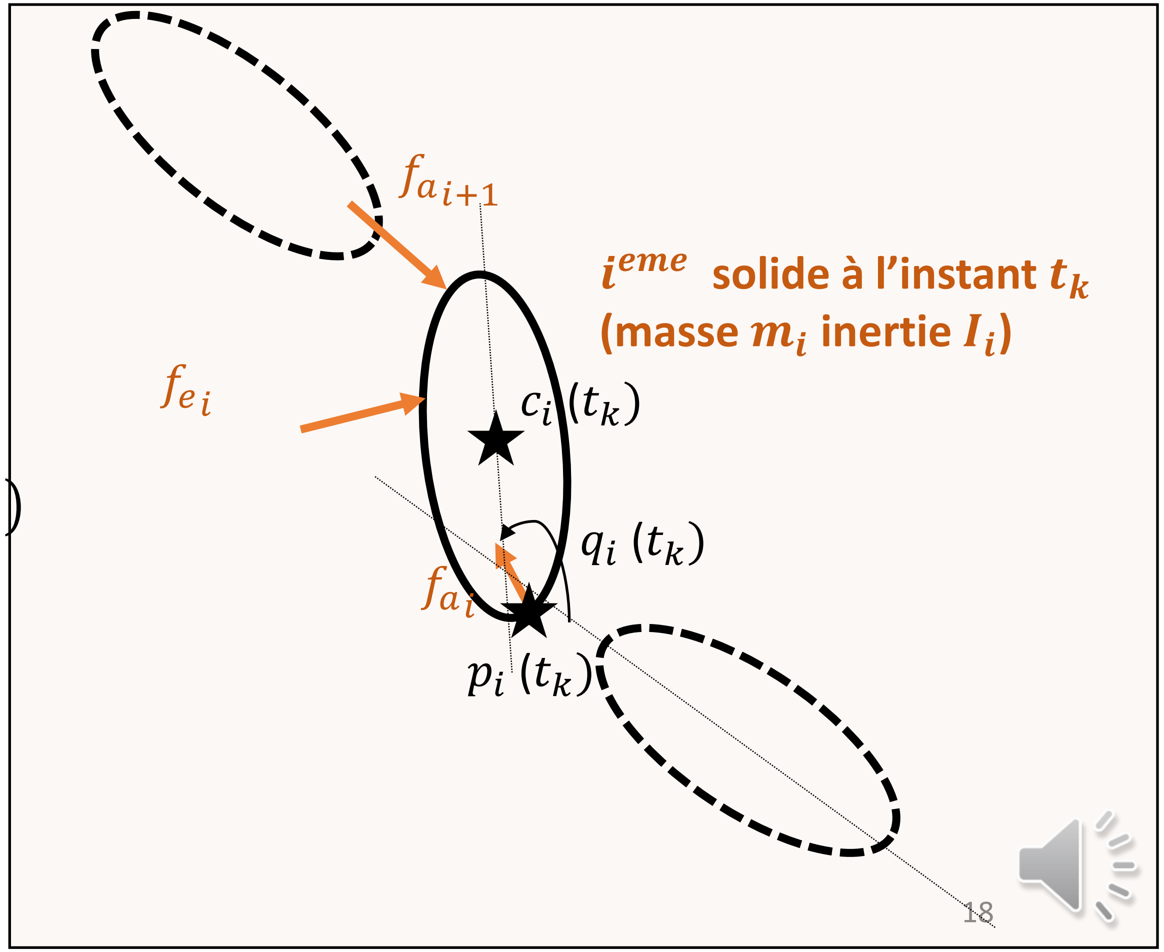
$$\boldsymbol{f}_{\boldsymbol{a}_{i+1}} = \boldsymbol{I}_{i+1}^A\ \dot{\boldsymbol{\xi}}_{i+1} + \boldsymbol{p}_{i+1}^A$$

$$\dot{\boldsymbol{\xi}}_{i+1} = \dot{\boldsymbol{\xi}}_i + \boldsymbol{s}_{i+1}\ddot{\boldsymbol{q}}_{i+1} + \dot{\boldsymbol{s}}_{i+1}\dot{\boldsymbol{q}}_{i+1}$$

Et enfin

$$\boldsymbol{\tau}_{i+1} = \boldsymbol{s}_{i+1}^t \boldsymbol{f}_{\boldsymbol{a}_{i+1}} = \boldsymbol{s}_{i+1}^t\left(\boldsymbol{I}_{i+1}^A(\ \dot{\boldsymbol{\xi}}_i + \boldsymbol{s}_{i+1}\ddot{q}_{i+1} + \dot{\boldsymbol{s}}_{i+1}\dot{q}_{i+1}) + \boldsymbol{p}_{i+1}^A\right)$$

Pour se débarrasser de $\ddot{q}_{i+1}$



Articulated-body algorithm

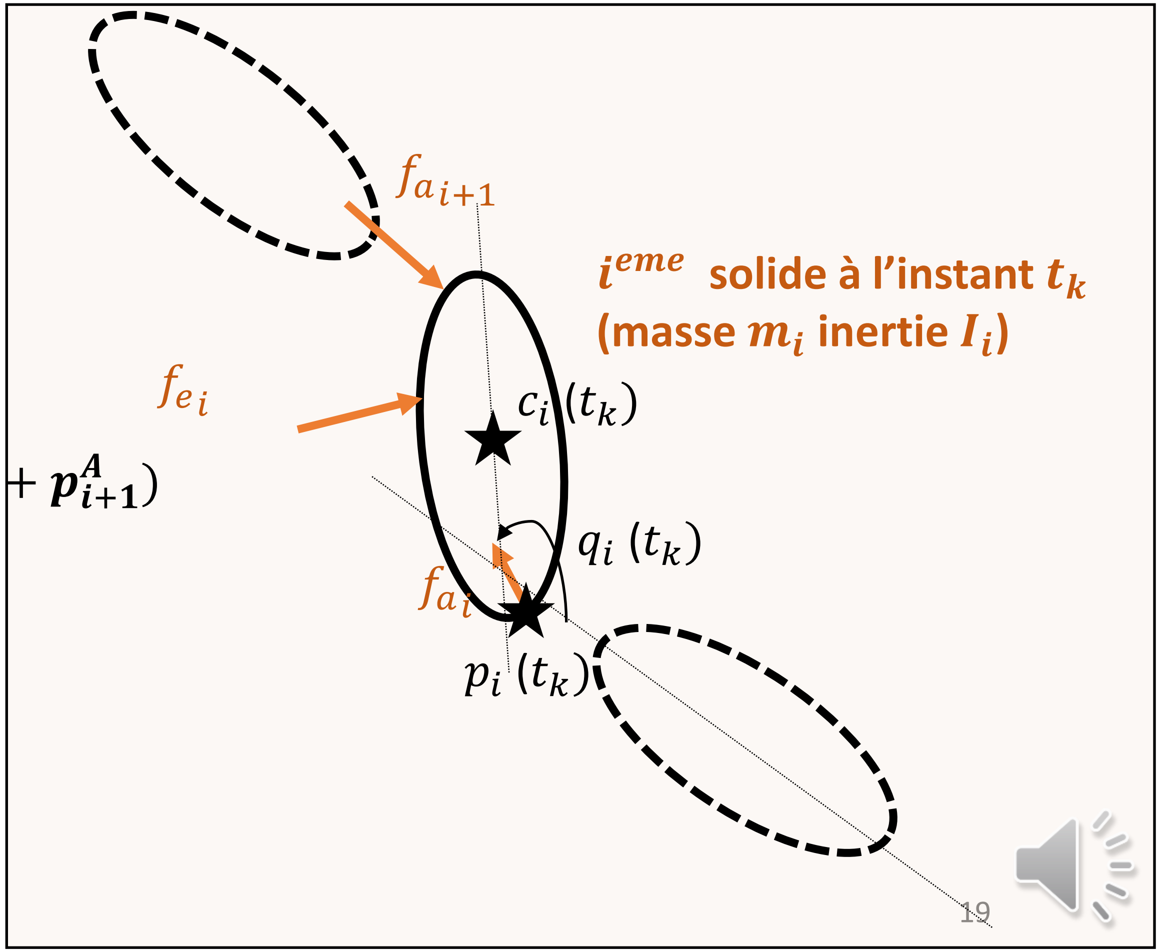
Etape 1

On obtient les expressions suivantes pour $\boldsymbol{I_i^A}$ et $\boldsymbol{p_i^A}$

$$\boldsymbol{I_i^A}=\boldsymbol{I_i^s}+\boldsymbol{I_{i+1}^A}-\left(\boldsymbol{s_{i+1}^t}\boldsymbol{I_{i+1}^A}\boldsymbol{s_{i+1}}\right)^{-1}\boldsymbol{I_{i+1}^A}\boldsymbol{s_{i+1}}\boldsymbol{s_{i+1}^t}\boldsymbol{I_{i+1}^A}$$

$$\begin{aligned}\boldsymbol{p_i^A} &= \boldsymbol{p_i}+\boldsymbol{p_{i+1}^A}+\boldsymbol{I_{i+1}^A}\dot{\boldsymbol{s}}_{i+1}\dot{\boldsymbol{q}}_{i+1}\\ &\quad -\left(\boldsymbol{s_{i+1}^t}\boldsymbol{I_{i+1}^A}\boldsymbol{s_{i+1}}\right)^{-1}\boldsymbol{I_{i+1}^A}\boldsymbol{s_{i+1}}\left(\tau_{i+1}-\boldsymbol{s_{i+1}^t}\left(\boldsymbol{I_{i+1}^A}\dot{\boldsymbol{s}}_{i+1}\dot{\boldsymbol{q}}_{i+1}+\boldsymbol{p_{i+1}^A}\right)\right)\end{aligned}$$

Expressions somme toute sympathiques !



Articulated-body algorithm

Etape 2

$$\tau_i = \boldsymbol{s}_i^t \boldsymbol{f}_{\boldsymbol{a}_i}$$

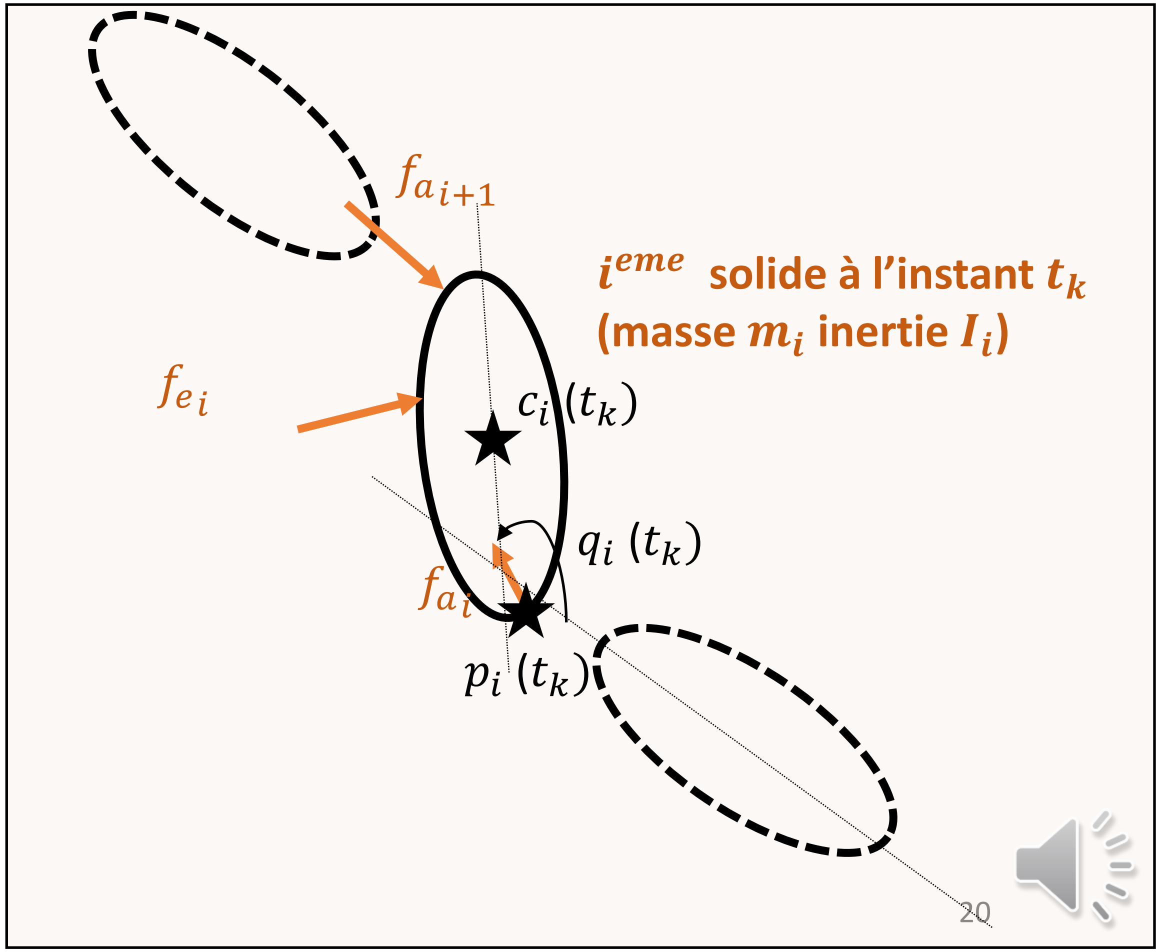
$$\tau_i = \boldsymbol{s}_i^t(\boldsymbol{I}_i^A\ \dot{\xi}_i + \boldsymbol{p}_i^A)$$

$$\tau_i = \boldsymbol{s}_i^t(\boldsymbol{I}_i^A(\dot{\xi}_{i-1} + \boldsymbol{s}_i\ddot{q}_i + \dot{\boldsymbol{s}}_i\dot{q}_i) + \boldsymbol{p}_i^A)$$

Ce qui donne

$$\ddot{\boldsymbol{q}}_i = (\boldsymbol{s}_i^t\boldsymbol{I}_i^A\boldsymbol{s}_i)^{-1}(\tau_i - \boldsymbol{s}_i^t(\boldsymbol{I}_i^A(\dot{\xi}_{i-1} + \dot{\boldsymbol{s}}_i\dot{\boldsymbol{q}}_i) + \boldsymbol{p}_i^A)$$

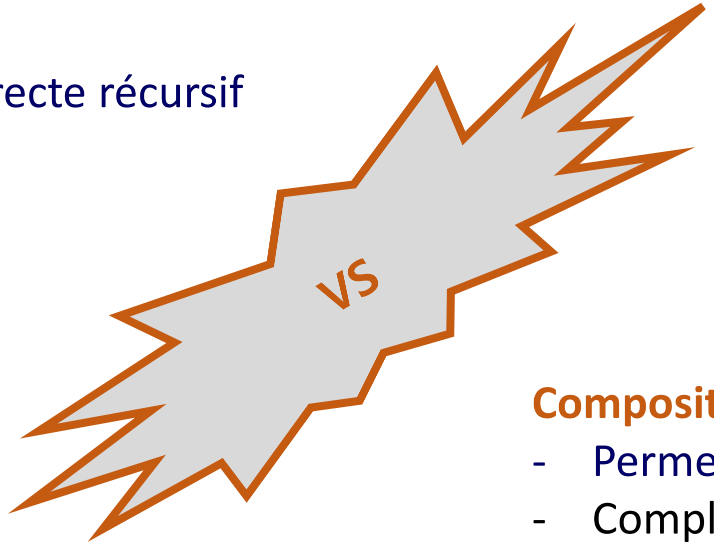
$$\dot{\xi}_i = \dot{\xi}_{i-1} + \boldsymbol{s}_i\ddot{q}_i + \dot{\boldsymbol{s}}_i\dot{q}_i$$



Résumé

Articulated–Body Algorithm

- Permet le calcul de dynamique directe récursif
- Complexité en $O(n)$
- Utile pour les systèmes ouverts



Composite–Rigid–Body Algorithm

- Permet le calcul de $H(q)$
- Complexité en $O(nd)$
- Utile pour les systèmes contraints