CusToM : a Matlab toolbox for musculoskeletal simulation

Introduction to Inverse dynamics

Charles Pontonnier, Pierre Puchaud

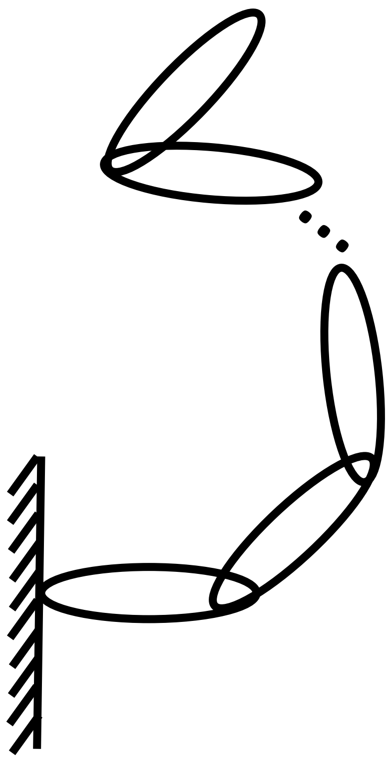
Mostly comes from

[1] Featherstone, R. (2014). *Rigid body dynamics algorithms*. Springer.

[2] Kajita, S., Hirukawa, H., Harada, K., & Yokoi, K. (2014). *Introduction to humanoid robotics* (Vol. 101). Springer Berlin Heidelberg.

Model description

Connectivity



n_b solids
n_q joints

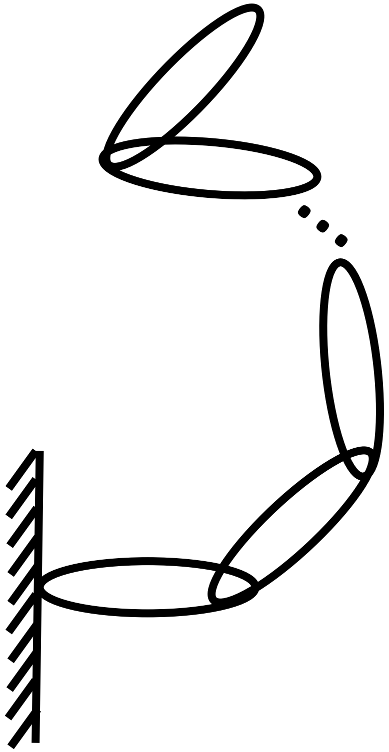
A polyarticulated system of rigid bodies needs to be represented by a connectivity graph, consistuted from nodes and arcs

The first node is the system reference

Each following node is a solid

Each arc between nodes is a joint

Connectivity

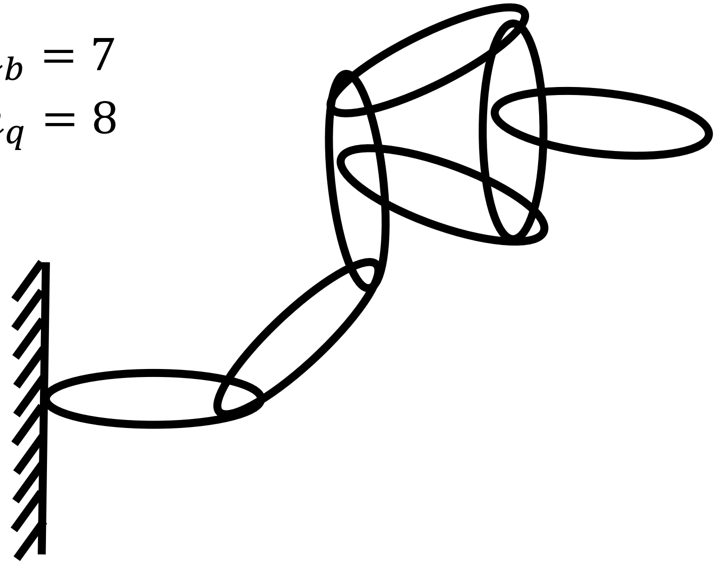


n_b solids
n_q joints

1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

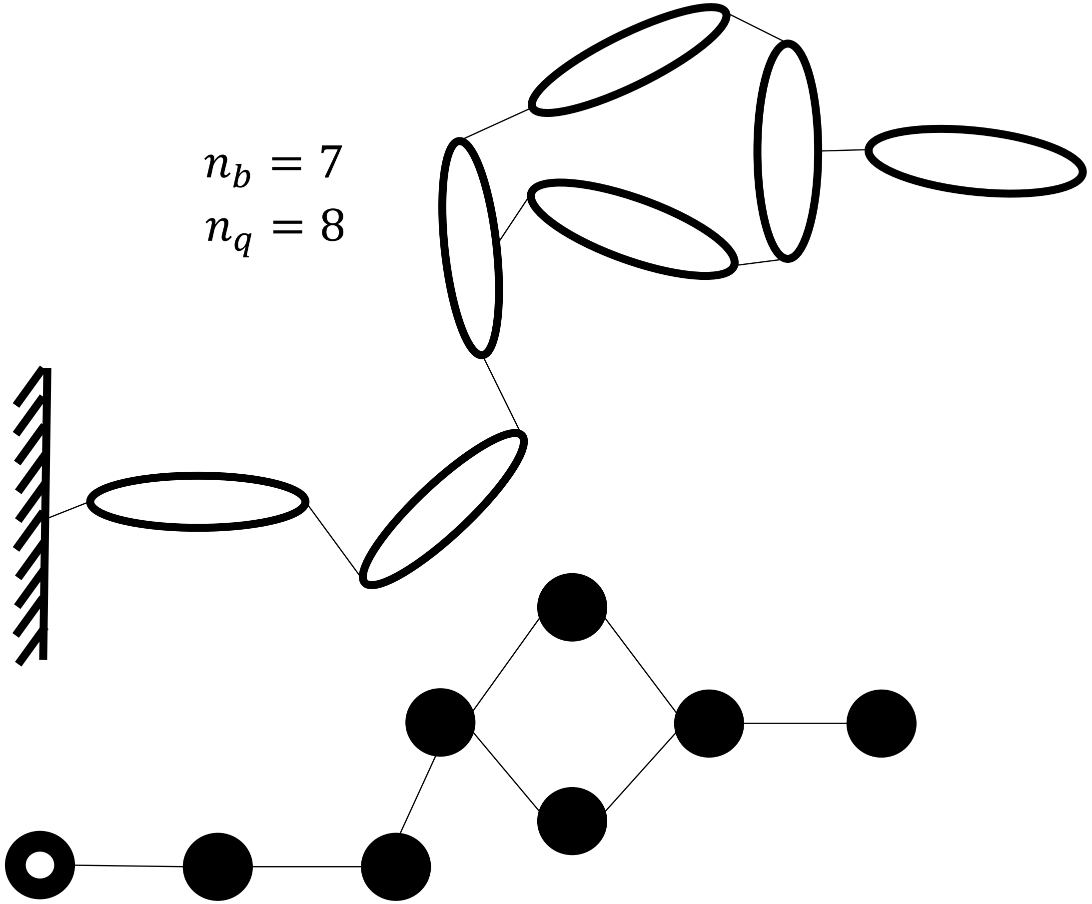
Connectivity

$n_b = 7$
$n_q = 8$



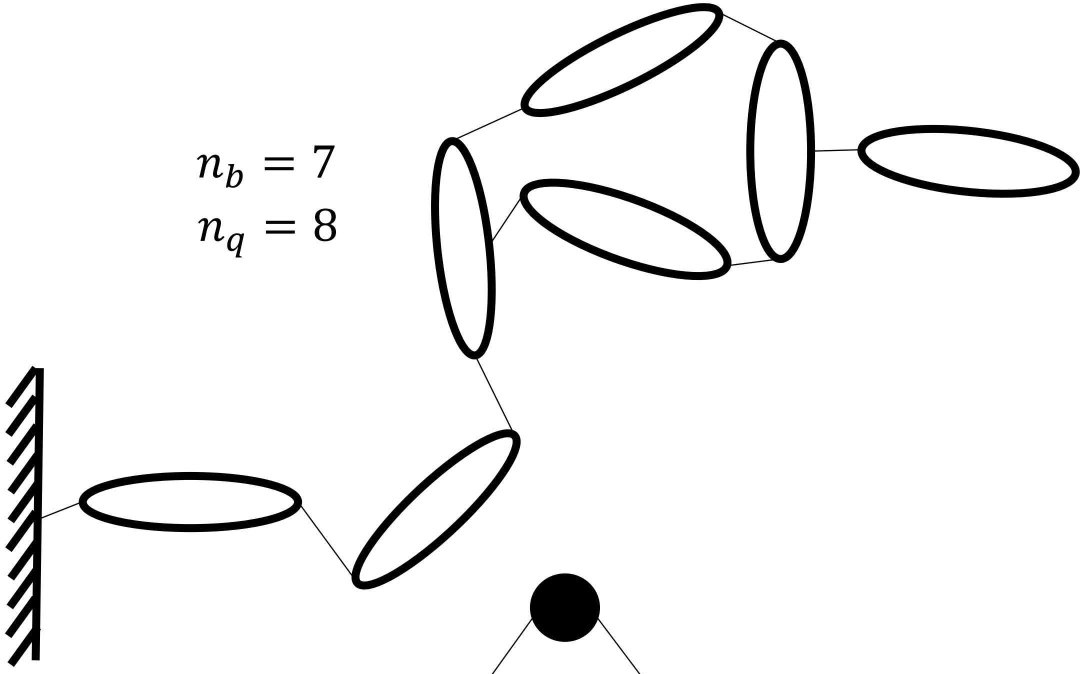
1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

Connectivity



1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

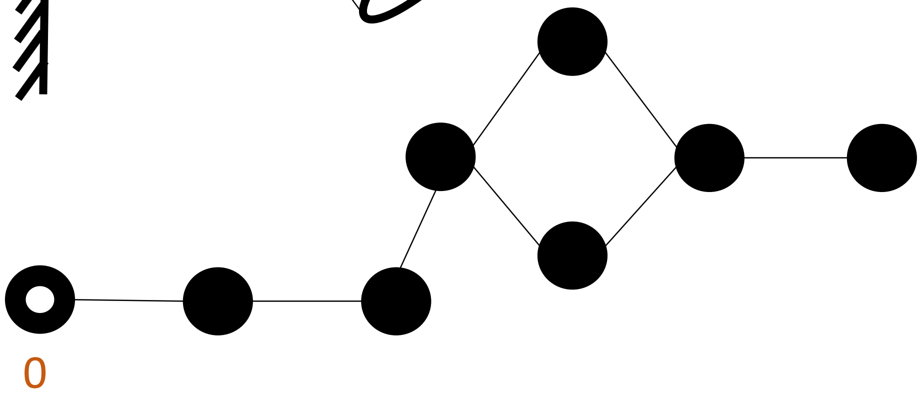
Connectivity



$n_b = 7$

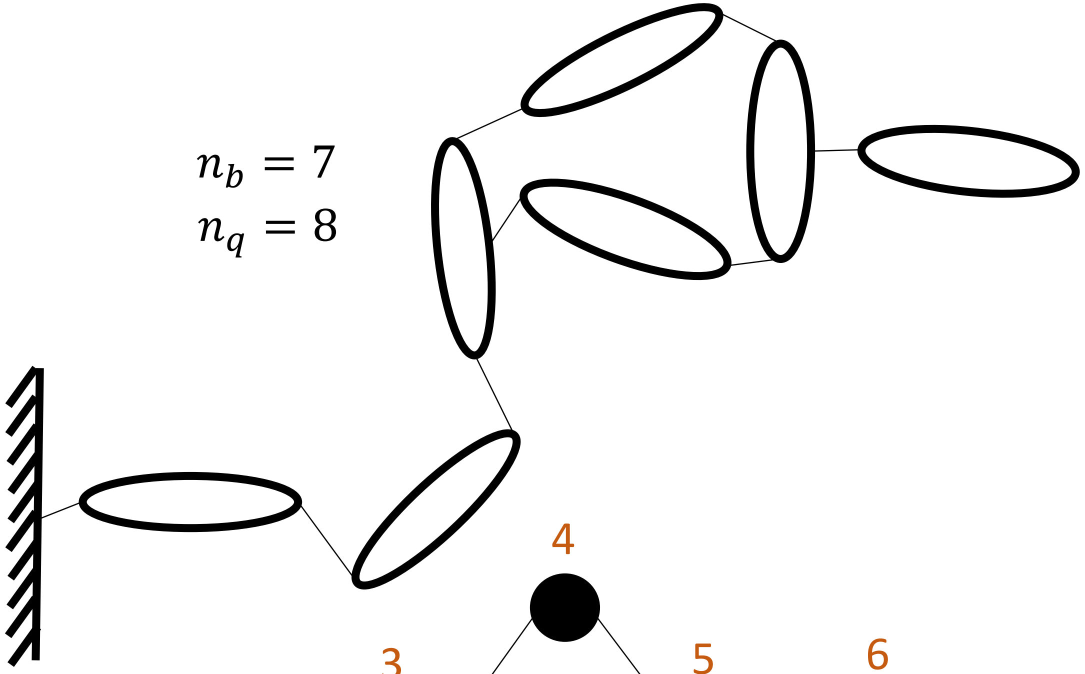
$n_q = 8$

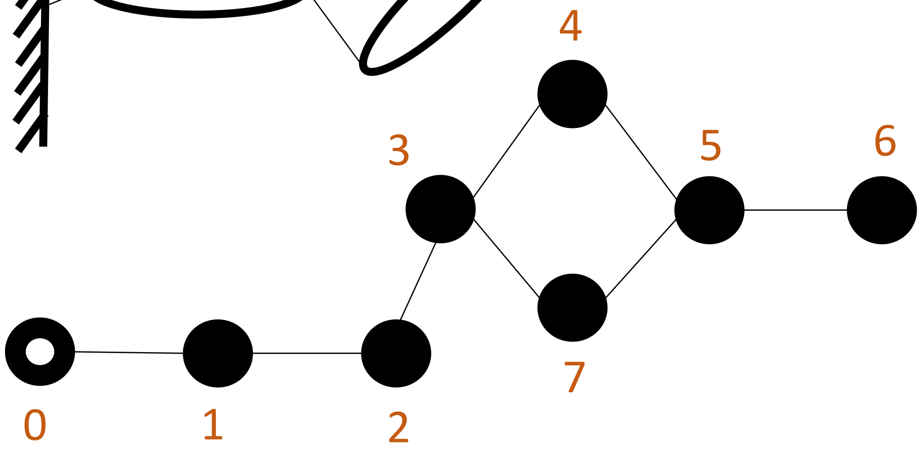
1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc



Connectivity

$n_b = 7$
$n_q = 8$

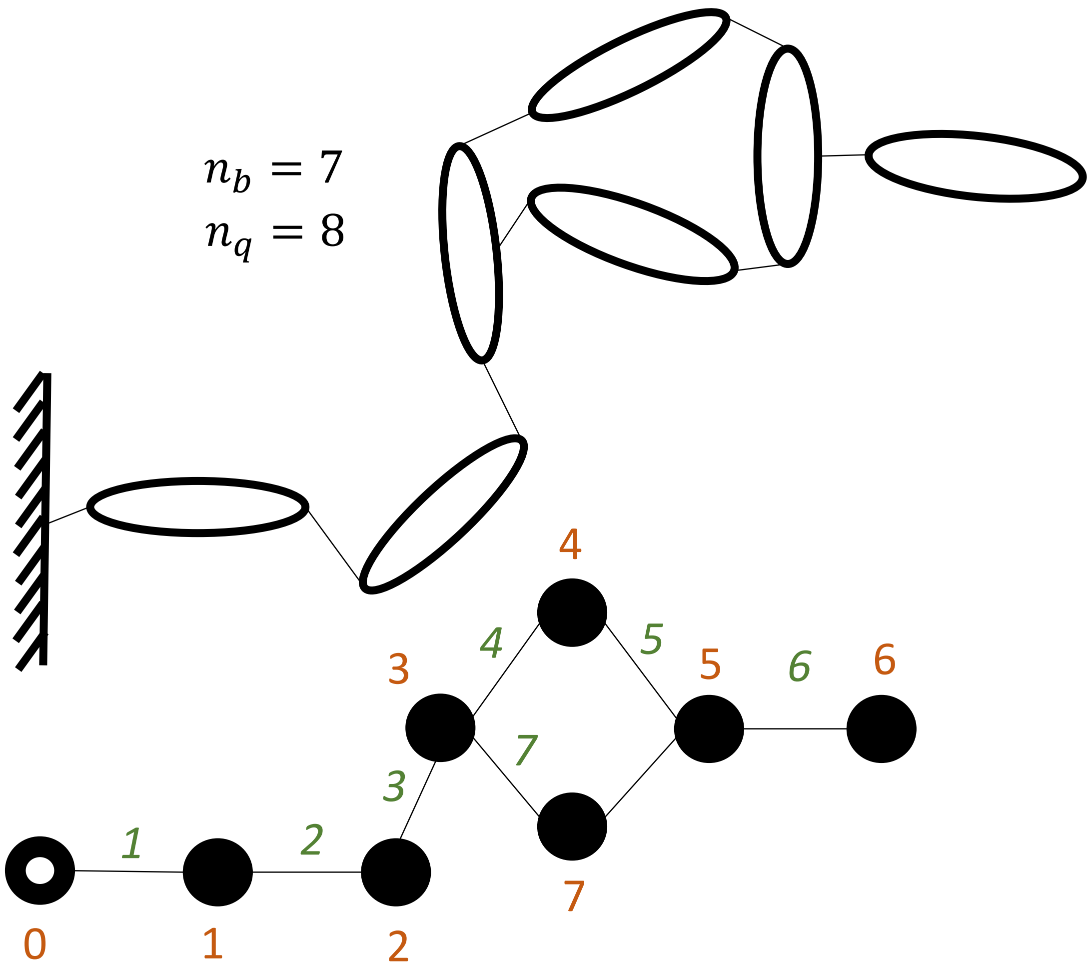




1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

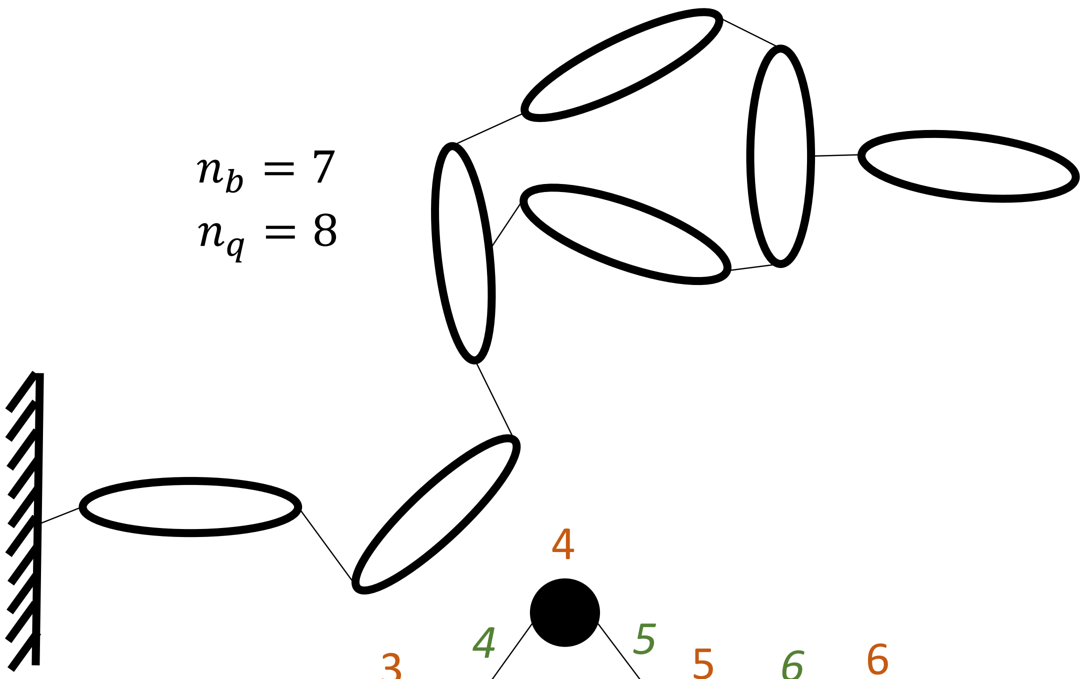
Connectivity

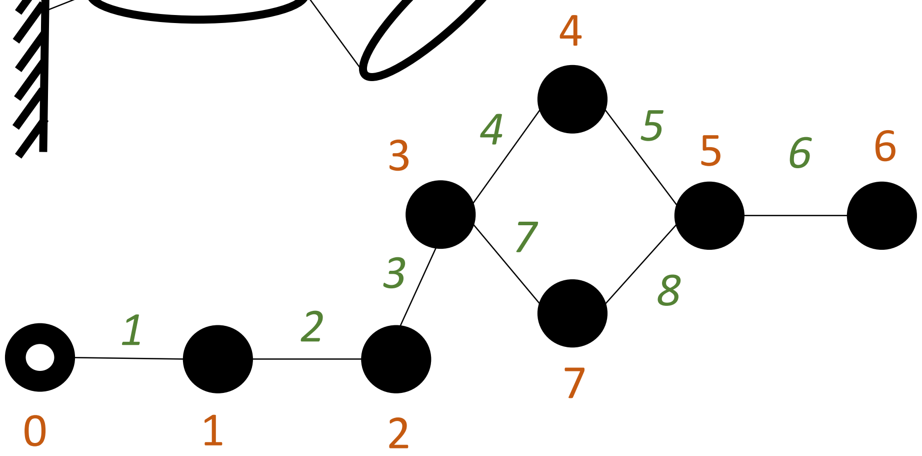
$n_b = 7$
$n_q = 8$



1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

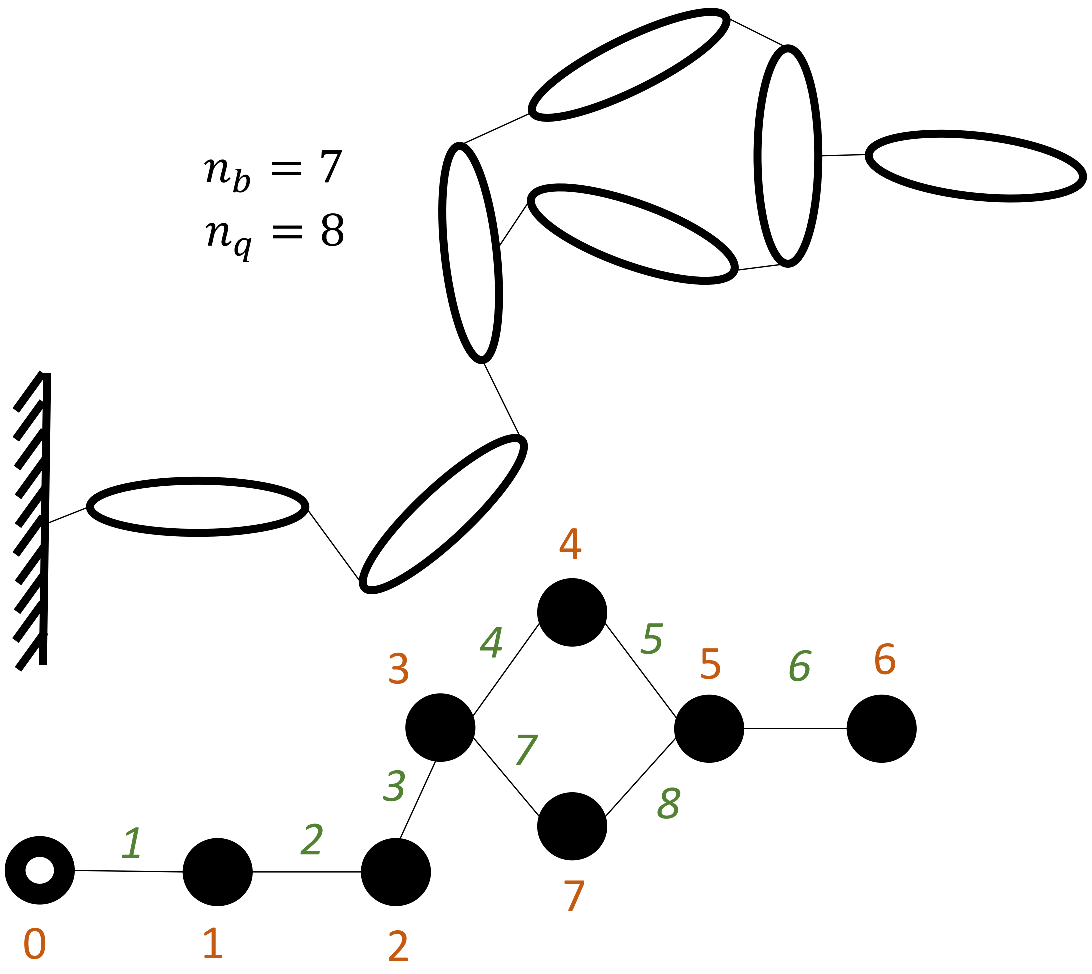
Connectivity





1. Define an arborescence (nodes and arcs)
2. Define the fixed or mobile base as node 0 (reference)
3. Number nodes from 1 to n_b with the rule that each node as an higher number than its predecessor (parent)
4. Number arcs from 1 à n_b with i connecting node i to its parent node
5. Number the reamining arcs $n_b + 1$ à n_q in any order
6. Each solid is numbered as its node, each joint as its arc

Connectivity



Properties

- Predecessors table ($p(i)$ solid before joint i)

  $$\boldsymbol{p = \{0, 1, 2, 3, 4, 5, 3, 7\}}$$

- Successors table ($s(i)$ solid after joint i)

  $$\boldsymbol{s = \{1, 2, 3, 4, 5, 6, 7, 5\}}$$

- Parents table ($\lambda(i)$ solid parent of solid i)

  Properties: $\lambda(i) = \min\big(p(i), s(i)\big)$ $\quad i \in [1, n_b]$

  $$\boldsymbol{\lambda = \{0, 1, 2, 3, 4, 5, 3\}}$$

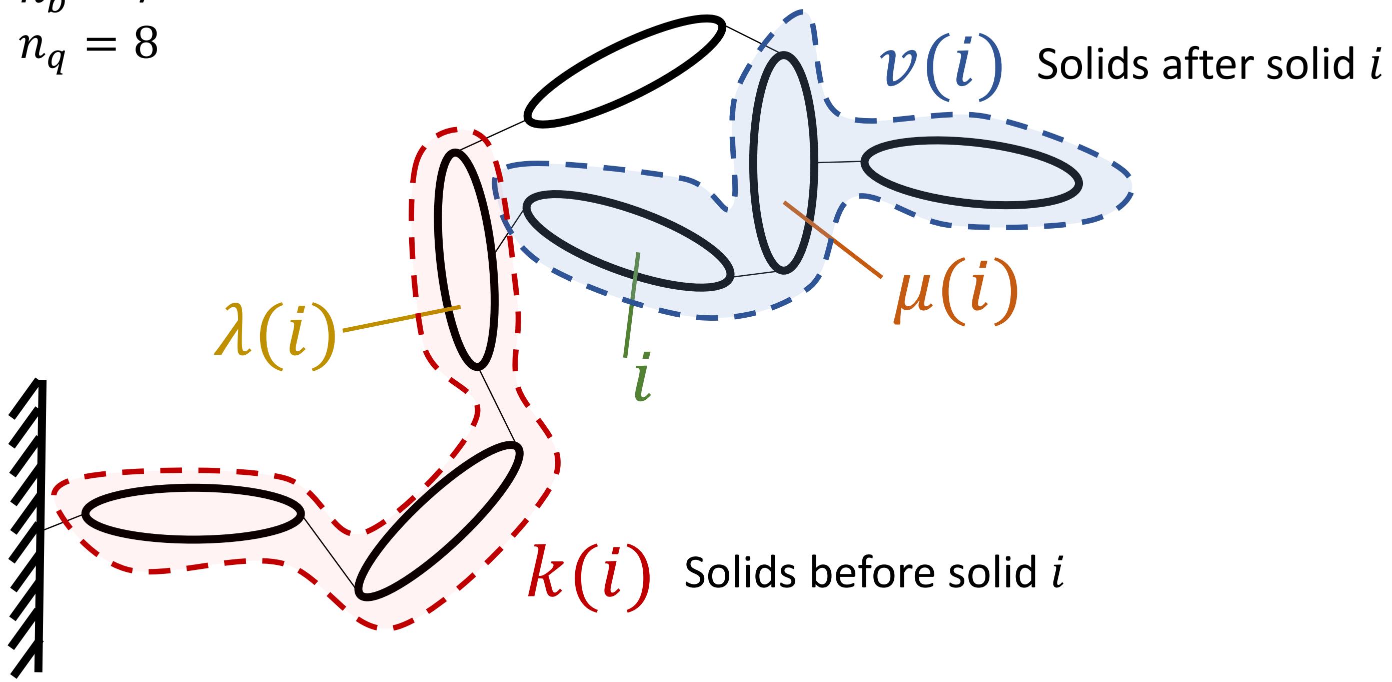
- Children table ($\mu(i)$ solids children of solid i)

  $$\boldsymbol{\mu(0) = 1}$$
  $$\boldsymbol{\mu(1) = 2}$$
  $$\vdots$$
  $$\boldsymbol{\mu(3) = \{4, 7\}}$$
  $$\vdots$$

λ and μ are fundamental to establish recursivity between the state of the solids and the forces applying on them

Connectivity

$n_b = 7$
$n_q = 8$

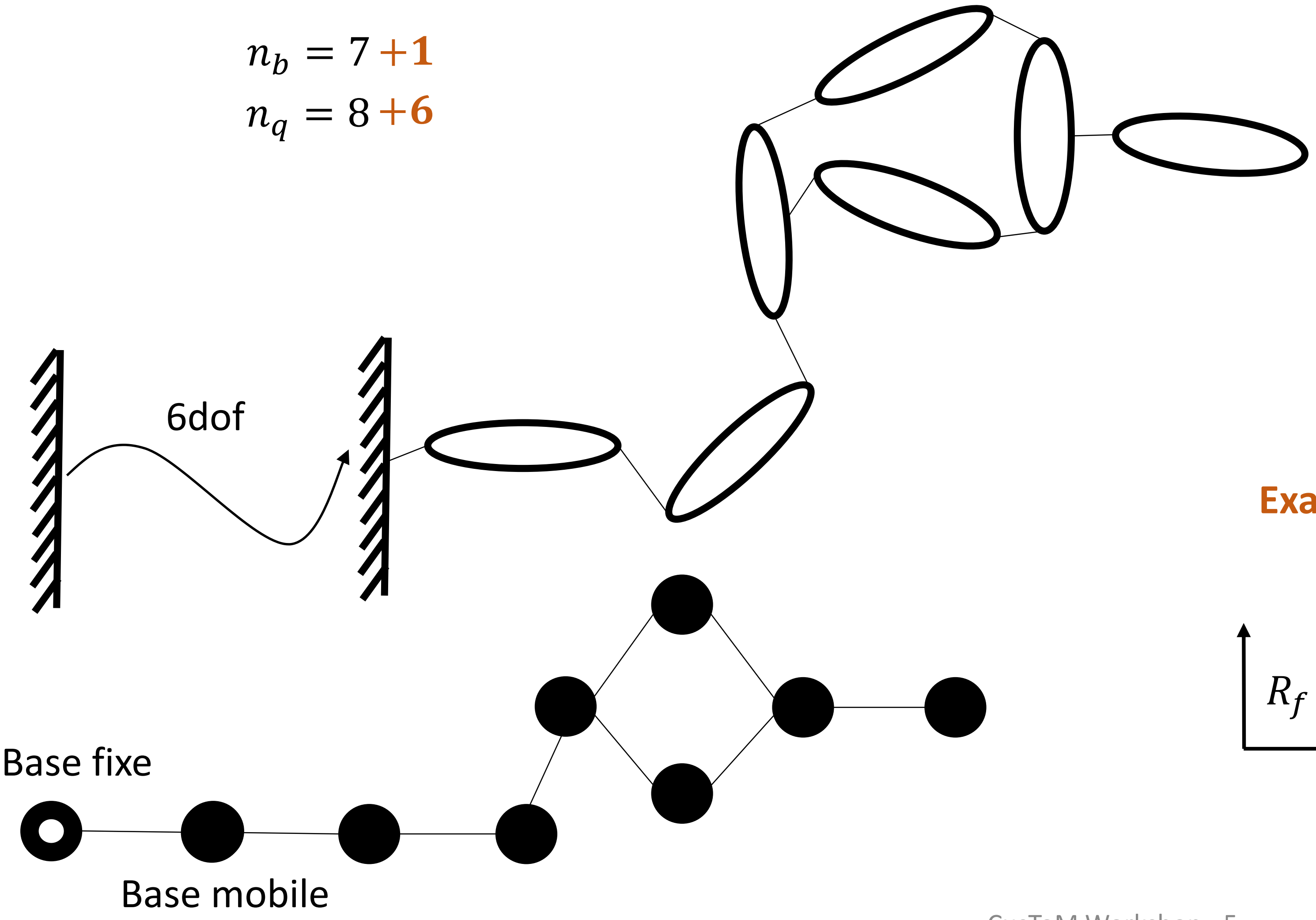


Practivcally, $\lambda(i)$ is sufficient to establish all of the recursivity rules to be applied to the system

Mobile base

$n_b = 7 + \mathbf{1}$

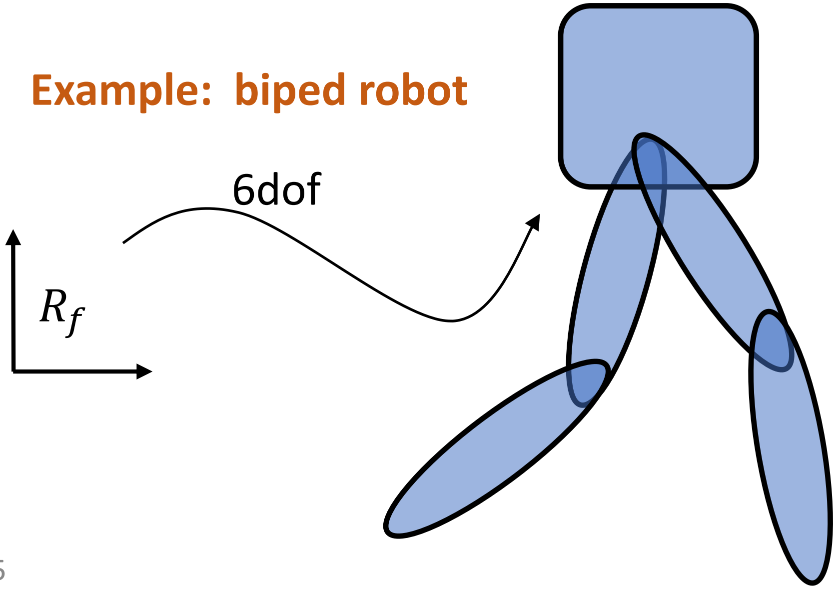
$n_q = 8 + \mathbf{6}$



A systemof rigid bodies is linked with the reference frame through **6 degrees of freedom (6dof)** adding no constraint to the system.

The first solid linked to te reference is called mobile base

Example: biped robot



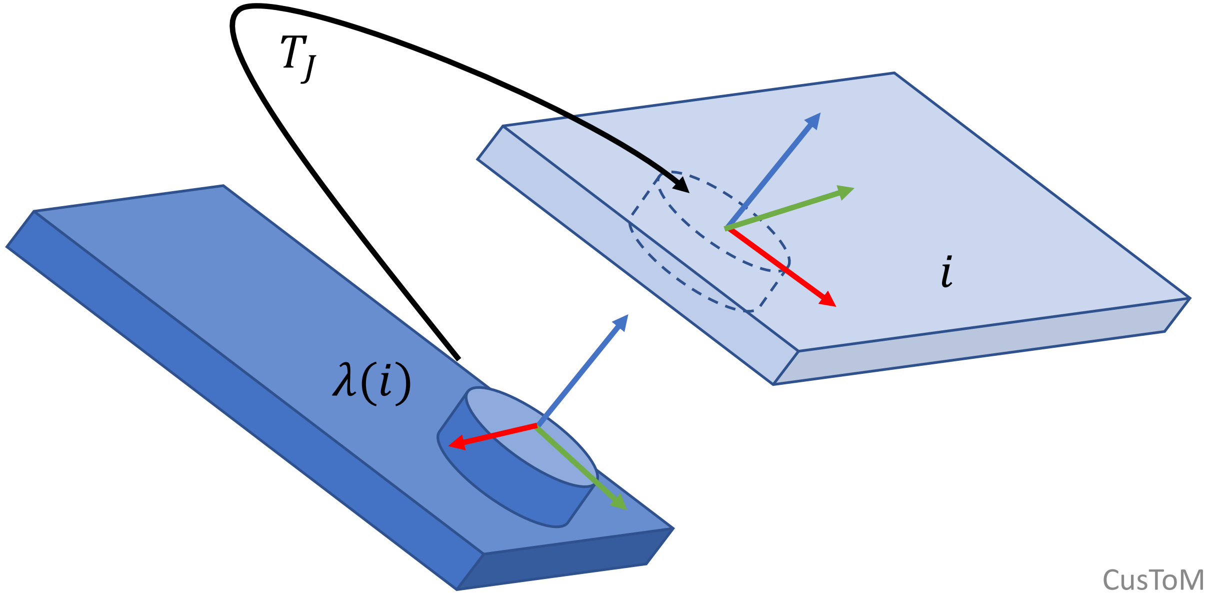
Joints

First identify joint type (analyzing dofs)

Define associated **motion parameters** if necessary (**Kinematic coupling if necessary**)

Define **homogeneous transfrom** between both solidsT_J

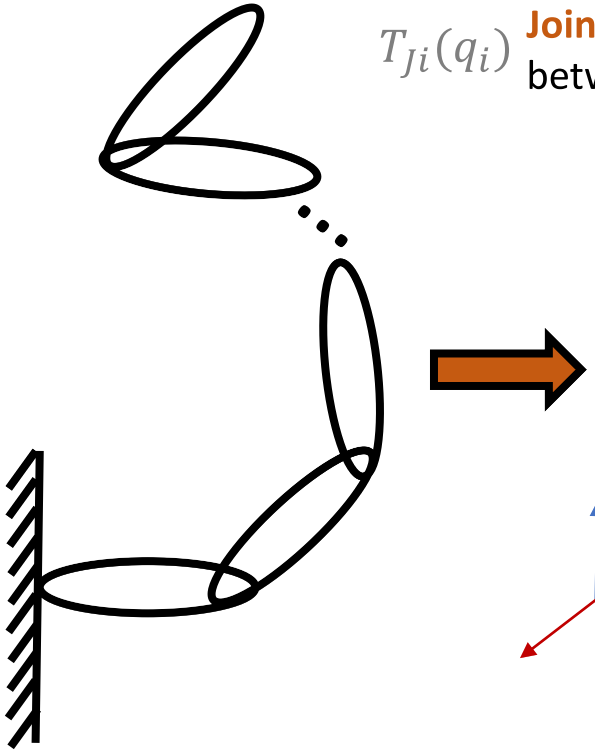
Define motion joint subspace by defining the **articular space** in the follower frame

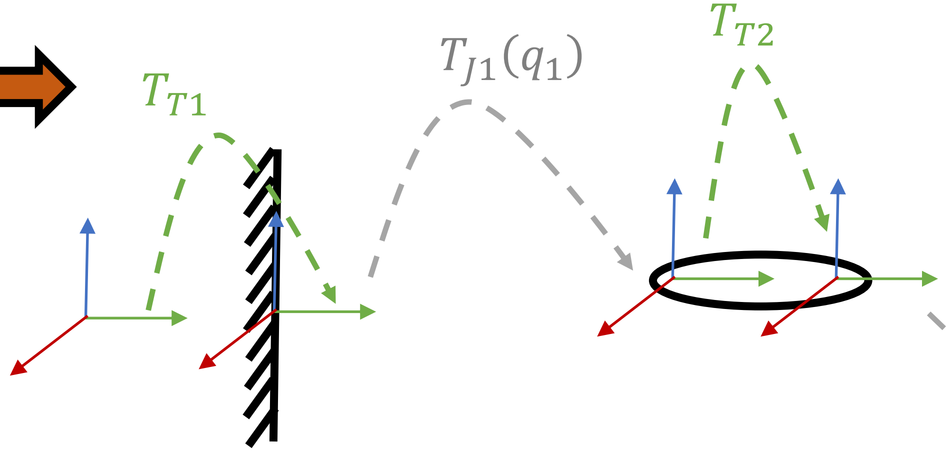


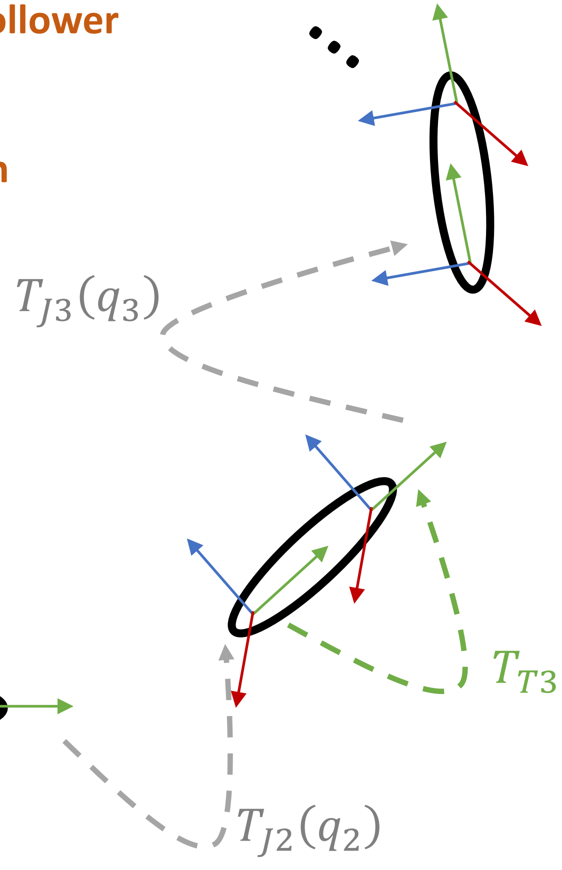
Geometry

T_{Ti} **Homogeneous transform** in a solid, placing the **follower** joint frame with regard to the **predecessor** one

$T_{Ji}(q_i)$ **Joint homogeneous transform** describing **motion** between the **predecessor** and the **follower**







Spatial algebra

Principle

Spatial algebra is a formal way to express in a concise and efficient manner **kinematics and dynamics** of rigid bodies. To do so, **6 dimension vectors and tensors** are defined.

Can be linked to the torsor definition, expressing both fixed and variable quantities associated to solids motion and forces.

Mathematical structure (short)

Spatial vectors can be defined in two mathematical spaces

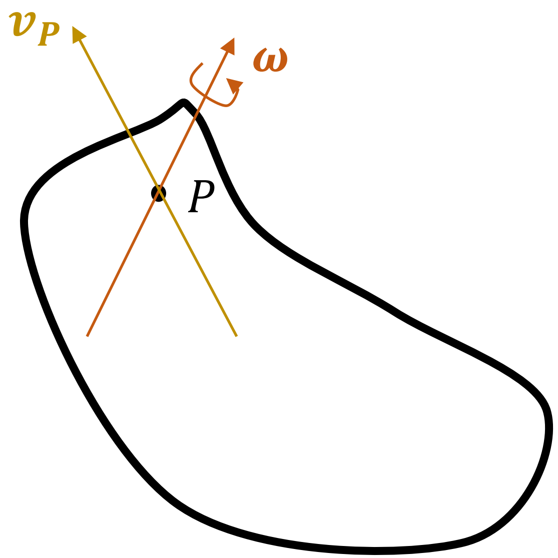
M^6 → motion vectors
F^6 → force vectors

The scalar product of these vectors from their respective spaces defines the work:

$$m \cdot f = work$$

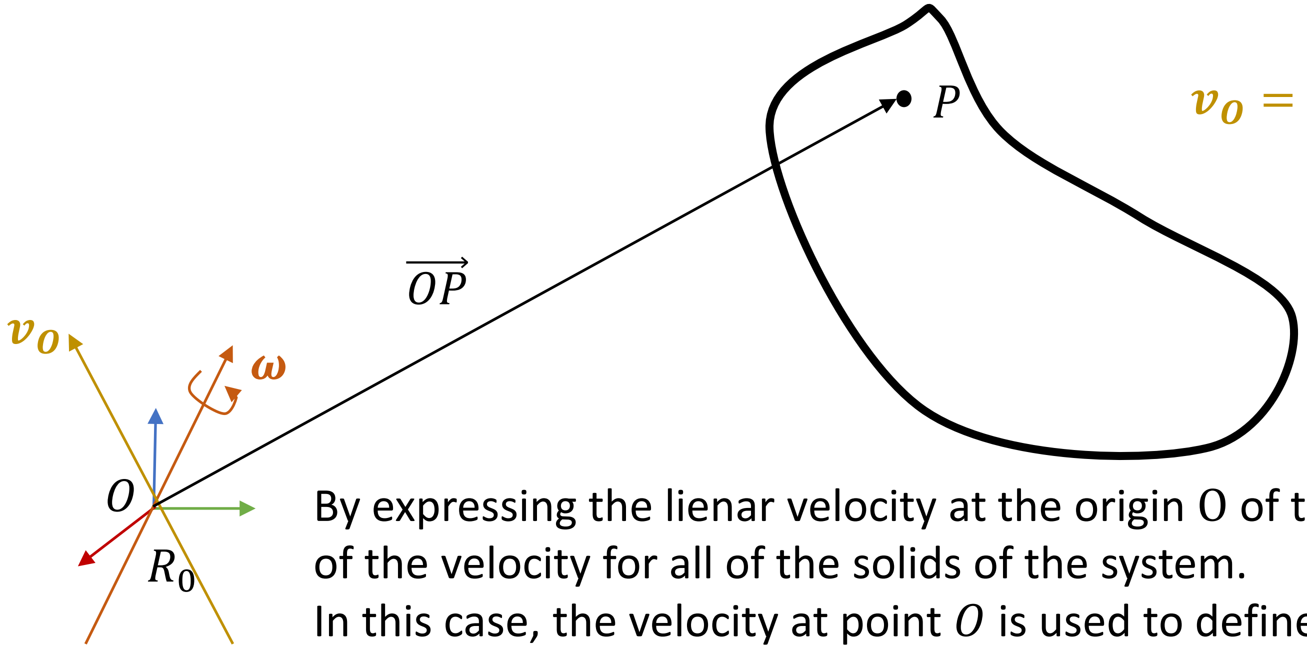
Scalar product

Spatial velocity



Classicaly, solid velocity is defined as an angular velocity $\boldsymbol{\omega}$ and a linear velocity $\boldsymbol{v_P}$ expressed at a given point P of the solid.

Spatial velocity



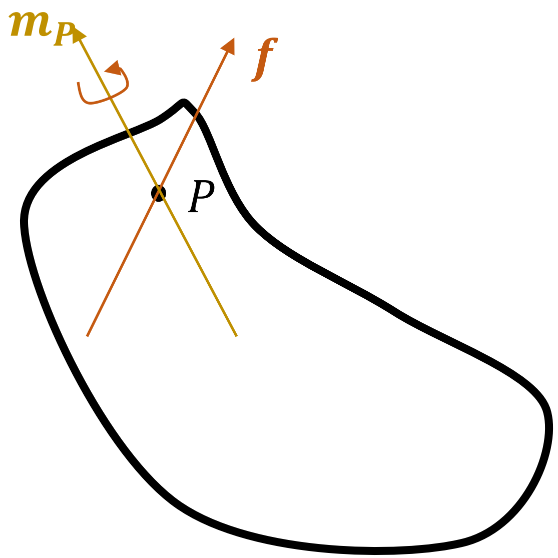
$$\boldsymbol{v_O} = \boldsymbol{v_P} + \overrightarrow{OP} \times \boldsymbol{\omega}$$

Cross product

By expressing the lienar velocity at the origin O of the reference frame, we get a unique expression of the velocity for all of the solids of the system.
In this case, the velocity at point O is used to define the **spatial velocity** of the solid :

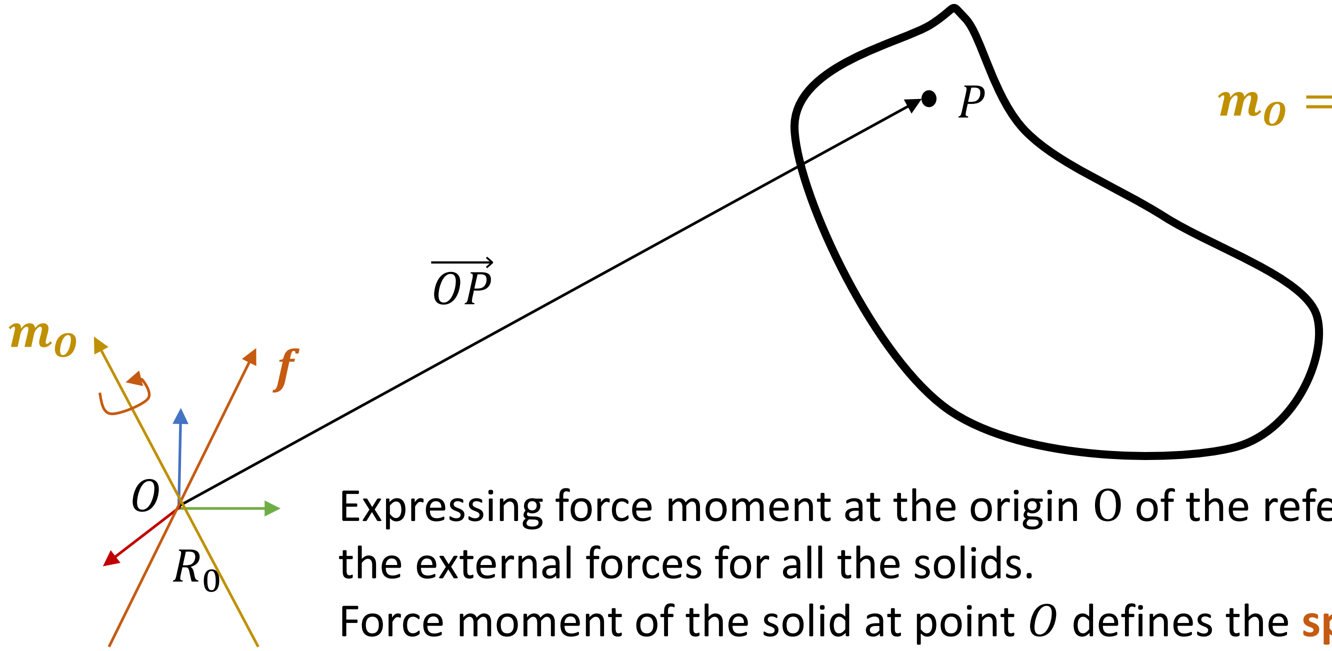
$$\xi = \begin{bmatrix} v_{O_x} \\ v_{O_y} \\ v_{O_z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Spatial force



Classicaly, external forces applied to a solid can be reduced to a linear force (resulting force) f and a force moment m_P expressed at a given point P of the solid.

Spatial force



$$\boldsymbol{m_O} = \boldsymbol{m_p} + \overrightarrow{OP} \times \boldsymbol{f}$$

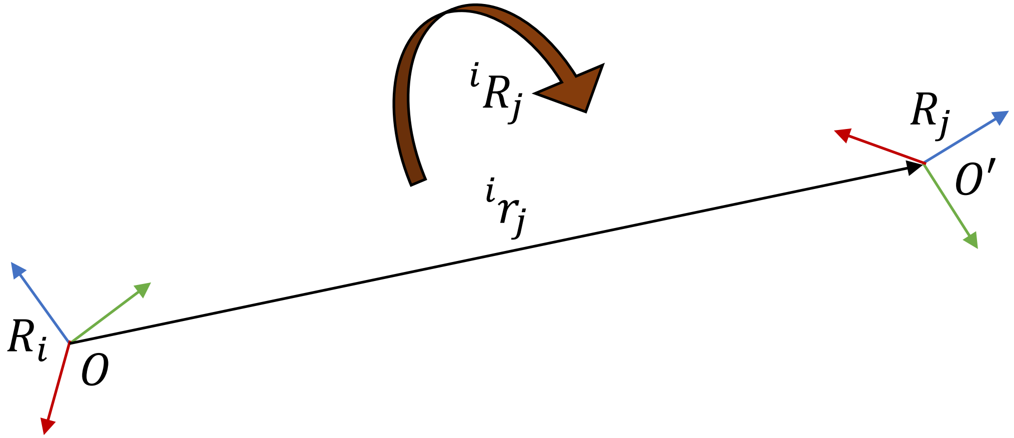
Cross product

Expressing force moment at the origin O of the reference frame, we obtain a unique expression of the external forces for all the solids.
Force moment of the solid at point O defines the **spatial force** associated to the solid:

$$\boldsymbol{\kappa} = \begin{bmatrix} f_x \\ f_y \\ f_z \\ m_{O_x} \\ m_{O_y} \\ m_{O_z} \end{bmatrix}$$

Spatial transform

We can define a transform matrix 6 x6 expressing rotation and translation between two frames to change the expression of a spatial vector from one frame to another



$$^iX_j = \begin{bmatrix} ^iR_j & ^iR_j \, {}^i\hat{r}_j{}^t \\ 0 & ^iR_j \end{bmatrix}$$

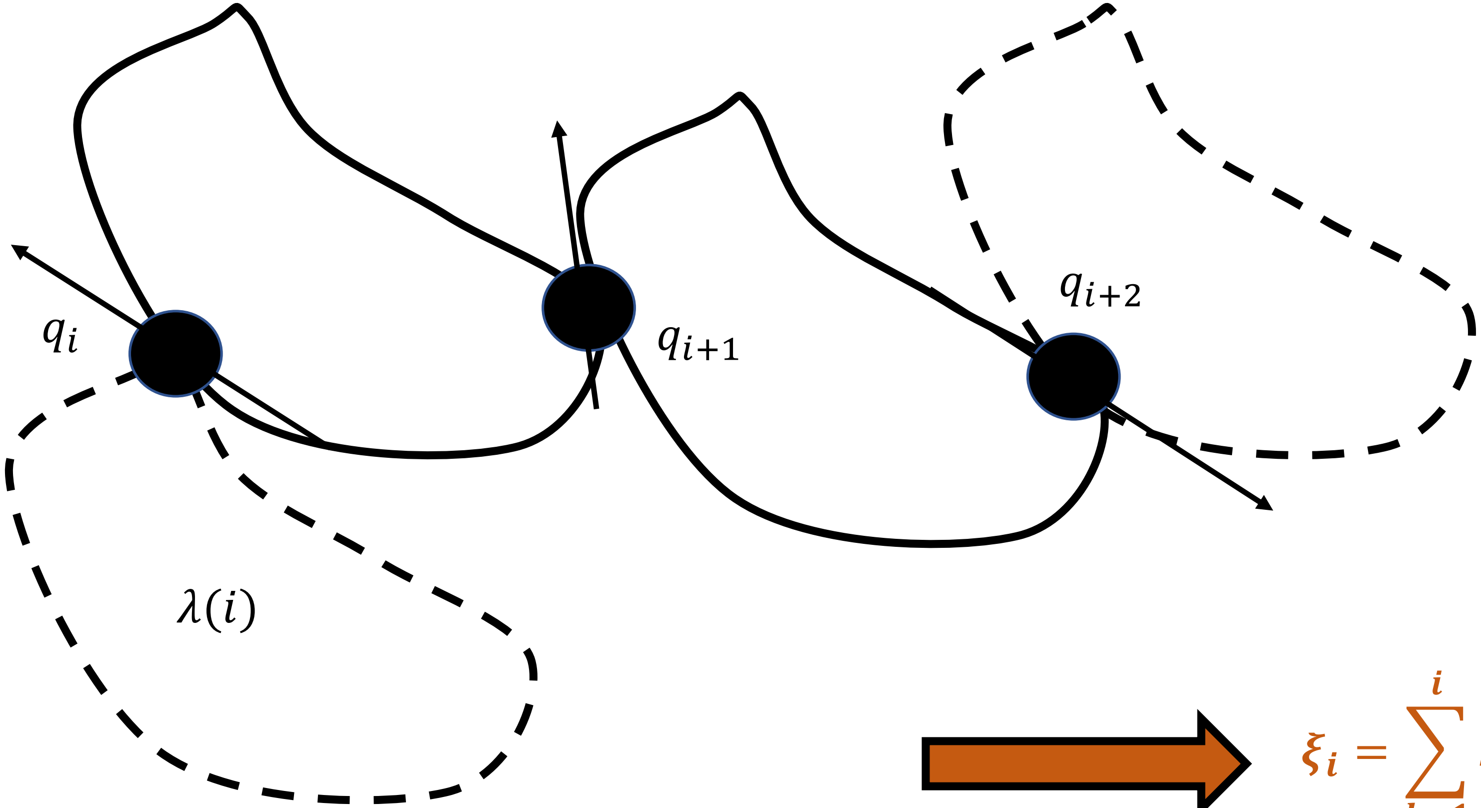
We also change the reference point of the spatial expressions (reference point becomes O instead of O')

with

$$^i\hat{r}_j = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

Cross product operator

Spatial velocity in a solid chain



Motion subspace associated to joint i (described in R_0) → $\boldsymbol{s_i}$

$$\boldsymbol{\xi_i} = \boldsymbol{\xi_{\lambda(i)}} + \overbrace{\begin{bmatrix} \boldsymbol{p_i} \times \boldsymbol{u_i} \\ \boldsymbol{u_i} \end{bmatrix}} \boldsymbol{\dot{q}_i}$$

$$\boldsymbol{\xi_{i+1}} = \boldsymbol{\xi_i} + \begin{bmatrix} \boldsymbol{p_{i+1}} \times \boldsymbol{u_{i+1}} \\ \boldsymbol{u_{i+1}} \end{bmatrix} \boldsymbol{\dot{q}_{i+1}}$$

$$\boldsymbol{\xi_i} = \sum_{k=1}^{i} \boldsymbol{s_k} \, \boldsymbol{\dot{q}_k}$$

Spatial Newton-Euler algorithm

Main issue

At time t_k

External forces

$\boldsymbol{f_e}(t_k)$

Equations of motion

$$\tau = B(q)^{-1}(H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - E(q)f_e)$$

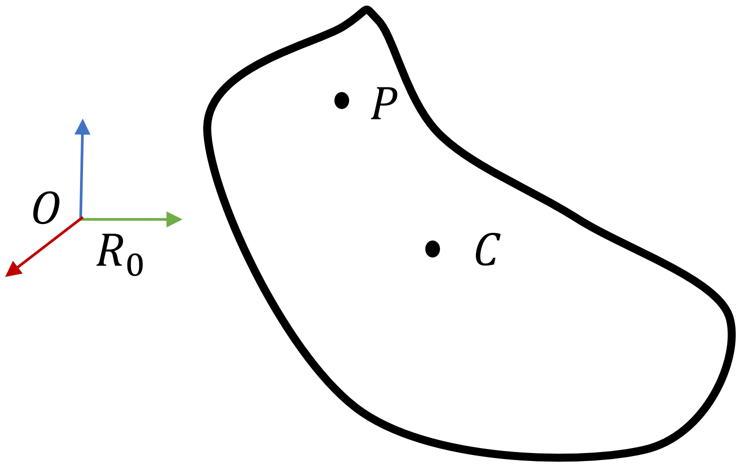
Joint torques

$\boldsymbol{\tau}(t_k)$

$(\boldsymbol{q_d}(t_k), \dot{\boldsymbol{q}}_{\boldsymbol{d}}(t_k), \ddot{\boldsymbol{q}}_{\boldsymbol{d}}(t_k))$

Angles, angular velocities and accelerations

Newton Euler equations for a solid S

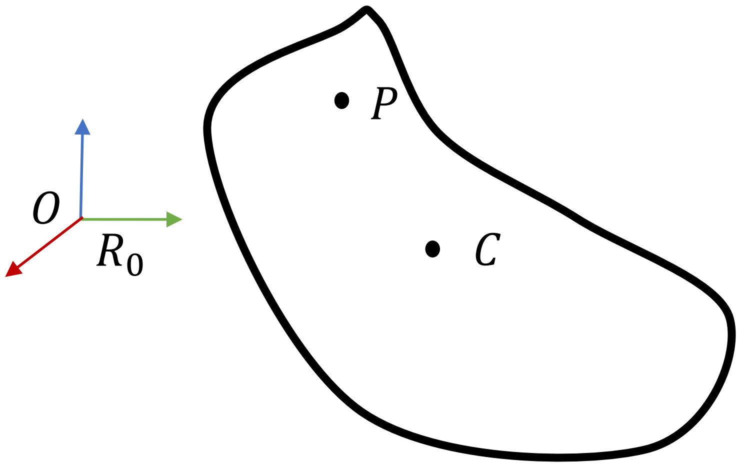


At center of mass

$$\begin{cases} \boldsymbol{f} = m\ddot{\boldsymbol{c}} & (1) \\ \boldsymbol{\tau}^{(c)} = \underbrace{\boxed{\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}}}_{\text{Acceleration momentum}} & (2) \end{cases}$$

$\boldsymbol{f}$	external forces
m	solid mass
$\boldsymbol{c}$	center of mass of the solid in R_0 (world) frame
$\boldsymbol{\omega}$	angular velocity of the solid in R_0
$\boldsymbol{I}$	inertia matrix of the solid in R_0
$\boldsymbol{\tau}^{(c)}$	torque associated to external forces, expressed in R_0 at the center of mass

Spatial equations of motion [Featherstone2007]



Velocity of S in O : $\boldsymbol{v}_O = \dot{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{\omega}$

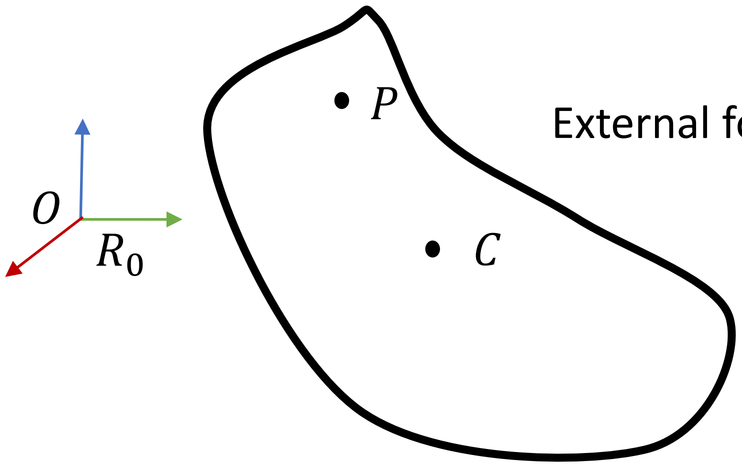
Acceleration of S in O: $\dot{\boldsymbol{v}}_O = \ddot{\boldsymbol{c}} + \dot{\boldsymbol{c}} \times \boldsymbol{\omega} + \boldsymbol{c} \times \dot{\boldsymbol{\omega}}$

The center of mass acceleration becomes: $\ddot{\boldsymbol{c}} = \dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times \boldsymbol{c})$

Replacing in (1), it comes

$$\boldsymbol{f} = m(\dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times \boldsymbol{c}))$$

Spatial equations of motion [Featherstone2007]



External forces torque expressed at O:

$$\boldsymbol{\tau}^{(O)} = \boldsymbol{\tau}^{(c)} + \boldsymbol{c} \times \boldsymbol{f}$$

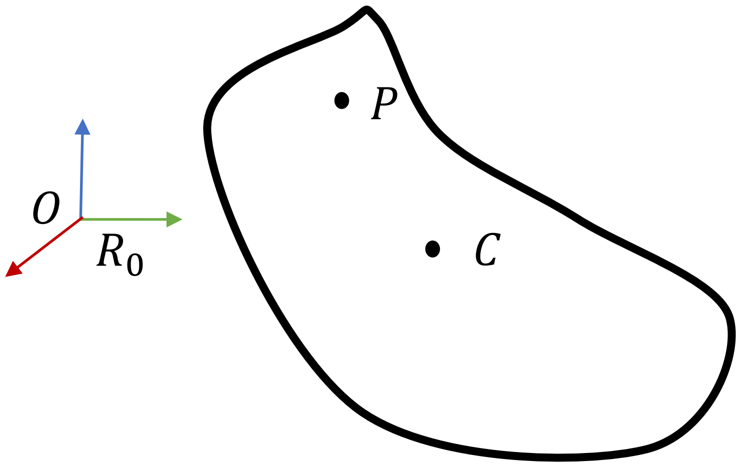
With $\boldsymbol{f} = m(\dot{\boldsymbol{v}}_{\boldsymbol{O}} - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_{\boldsymbol{O}} + \boldsymbol{\omega} \times \boldsymbol{c}))$

And $\boldsymbol{\tau}^{(c)} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}$

Finally:

$$\boldsymbol{\tau}^{(O)} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} + \boldsymbol{c} \times m(\dot{\boldsymbol{v}}_{\boldsymbol{O}} - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_{\boldsymbol{O}} + \boldsymbol{\omega} \times \boldsymbol{c}))$$

Spatial equations of motion [Featherstone2007]



$$\boldsymbol{f} = m(\dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times \boldsymbol{c}))$$

$$\boldsymbol{\tau}^{(O)} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} + \boldsymbol{c} \times m(\dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times \boldsymbol{c}))$$

That can be expressed with a matrix expression

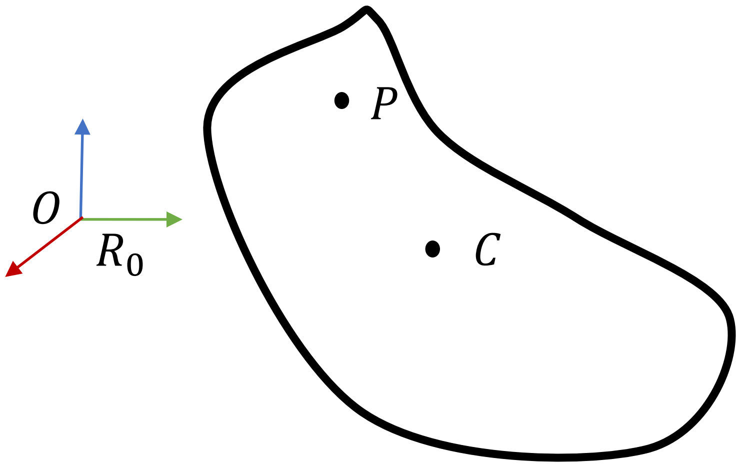
$$\boxed{\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{I}^S \begin{bmatrix} \dot{\boldsymbol{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{0} \\ \widehat{\boldsymbol{v}_O} & \widehat{\boldsymbol{\omega}} \end{bmatrix} \boldsymbol{I}^S \begin{bmatrix} \boldsymbol{v}_O \\ \boldsymbol{\omega} \end{bmatrix} = \boldsymbol{I}^S \begin{bmatrix} \dot{\boldsymbol{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_O \\ \boldsymbol{\omega} \end{bmatrix} \times \boldsymbol{I}^S \begin{bmatrix} \boldsymbol{v}_O \\ \boldsymbol{\omega} \end{bmatrix}}$$

(application produit vectoriel)

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

This compact expression involves two fundamental spatial algebra features for rigid body dynamics that are: $\boldsymbol{I}^S$ spatial inertia matrix and $\begin{bmatrix} \dot{\boldsymbol{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \dot{\boldsymbol{\xi}}$ spatial acceleration of S

Spatial inertia and spatial acceleration



$\mathbb{I}$ identity matrix

Spatial inertia matrix: 6 x 6 Symmetrical matrix:

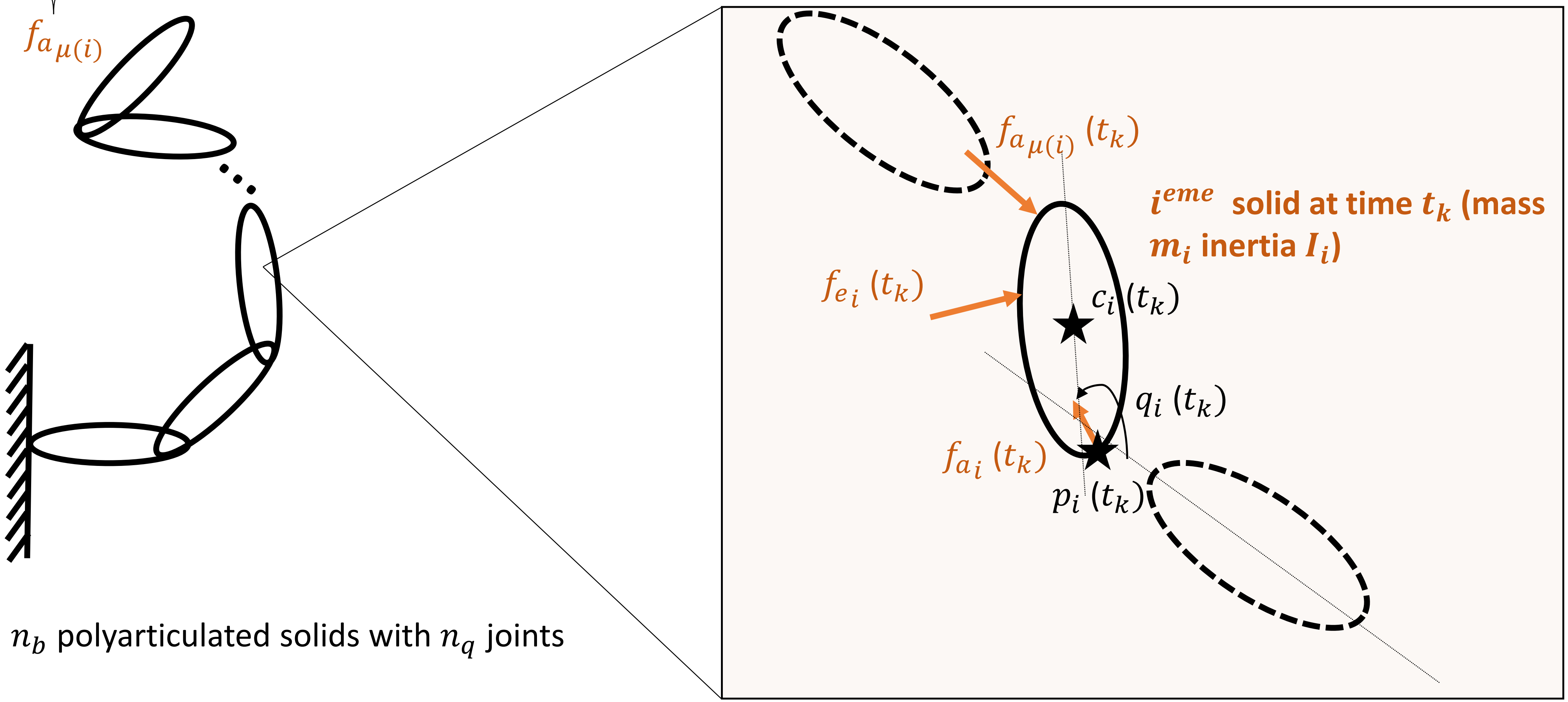
$$\boldsymbol{I}^s = \begin{bmatrix} m\mathbb{I} & m\hat{\boldsymbol{c}}^t \\ m\hat{\boldsymbol{c}} & m\hat{\boldsymbol{c}}\hat{\boldsymbol{c}}^t + \boldsymbol{I} \end{bmatrix}$$

Spatial acceleration: not a physical acceleration

$$\begin{bmatrix} \dot{\boldsymbol{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \dot{\boldsymbol{\xi}}$$

Equations of motion of one solid in a polyarticulated chain of solids

$$\underbrace{\begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{\tau}_i \end{bmatrix}}_{f_{a_i}} - \underbrace{\begin{bmatrix} \boldsymbol{f}_{i+1} \\ \boldsymbol{\tau}_{i+1} \end{bmatrix}}_{f_{a_{\mu(i)}}} + \boldsymbol{f}_{e_i} = \boldsymbol{I}_i^S \begin{bmatrix} \dot{\boldsymbol{v}}_{O_i} \\ \dot{\boldsymbol{\omega}}_i \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_{O_i} \\ \boldsymbol{\omega}_i \end{bmatrix} \times \boldsymbol{I}_i^S \begin{bmatrix} \boldsymbol{v}_{O_i} \\ \boldsymbol{\omega}_i \end{bmatrix}$$



n_b polyarticulated solids with n_q joints

Newton-Euler algorithm

Recursive Newton-Euler algorithm

Knowing joint angles, joint velocities and joint accelerations at time t_k

1. Computing cartesian and angular velocities of all bodies from the base to extremities
2. Computing joint reaction forces of all solids from extremities to the base

À un instant t_k

For $i = 1$ to n_B do

$$\dot{\boldsymbol{\xi}}_{\boldsymbol{i}} = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{f}_{\boldsymbol{e}_i}, \dot{\boldsymbol{\xi}}_{\lambda(\boldsymbol{i})})$$

End

For $i = n_B$ to 1 do

$$\boldsymbol{f}_{\boldsymbol{a}_{\boldsymbol{i}}} = f(\dot{\boldsymbol{\xi}}_i, \boldsymbol{f}_{\boldsymbol{e}_i}, \boldsymbol{f}_{\boldsymbol{a}_{\boldsymbol{\mu}(\boldsymbol{i})}})$$

End

Newton-Euler algorithm

For $i = 1$ to n_B do

$$\dot{\boldsymbol{\xi}}_i = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{f}_{e_i}, \dot{\boldsymbol{\xi}}_{\lambda(i)})$$

End

For $i = n_B$ to 1 do

$$\boldsymbol{f}_{a_i} = f(\dot{\boldsymbol{\xi}}_i, \boldsymbol{f}_{e_i}, \boldsymbol{f}_{a_{\mu(i)}})$$

End

Spatial → all is computed in O

$$\left.\begin{aligned} {}^{0}\boldsymbol{R}_i &= {}^{0}\boldsymbol{R}_{\lambda(i)} {}^{\lambda(i)}\boldsymbol{R}_i(\boldsymbol{q}_i) \\ \boldsymbol{p}_i &= \boldsymbol{p}_{\lambda(i)} + {}^{0}\boldsymbol{R}_{\lambda(i)} \boldsymbol{b}_i \end{aligned}\right\}$$

Solid position and orientation update

$${}^{(0)}\boldsymbol{u}_i = {}^{0}\boldsymbol{R}_{\lambda(i)} {}^{\lambda(i)}\boldsymbol{u}_i$$

Joint axis between $i-1$ and i orientation update

$$\boldsymbol{\xi}_i = \boldsymbol{\xi}_{\lambda(i)} + \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix} \dot{\boldsymbol{q}}_i$$

Spatial velocity update

$$\dot{\boldsymbol{\xi}}_i = \dot{\boldsymbol{\xi}}_{\lambda(i)} + \begin{bmatrix} \widehat{\boldsymbol{\omega}}_i & \widehat{\boldsymbol{v}}_{0_i} \\ \boldsymbol{0} & \widehat{\boldsymbol{\omega}}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix} \dot{\boldsymbol{q}}_i + \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix} \ddot{\boldsymbol{q}}_i$$

Spatial acceleration update

Newton-Euler algorithm

For $i = 1$ to n_B do

$$\dot{\xi}_i = f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{f}_{\boldsymbol{e}_i}, \dot{\xi}_{\lambda(\boldsymbol{i})})$$

End

For $i = n_B$ to 1 do

$$\boldsymbol{f}_{\boldsymbol{a}_i} = f(\dot{\xi}_i, \boldsymbol{f}_{\boldsymbol{e}_i}, \boldsymbol{f}_{\boldsymbol{a}_{\mu(i)}})$$

End

$$^{(0)}\boldsymbol{c}_i = \boldsymbol{p}_i + {}^{0}\boldsymbol{R}_i{}^{i-1}\boldsymbol{c}_i$$

Center of mass position update for solid i

$$\boldsymbol{I}_i^s = \begin{bmatrix} \boldsymbol{m}_i\mathbb{I} & \boldsymbol{m}_i\hat{\boldsymbol{c}}_i{}^t \\ \boldsymbol{m}_i\hat{\boldsymbol{c}}_i & \boldsymbol{m}_i\hat{\boldsymbol{c}}_i\hat{\boldsymbol{c}}_i^t + \boldsymbol{I}_i \end{bmatrix}$$

Spatial inertia matrix computation for solid i

$$\boldsymbol{f}_i^{acc} = \boldsymbol{I}_i^s\dot{\xi}_i + \xi_i \times \boldsymbol{I}_i^s\xi_i$$

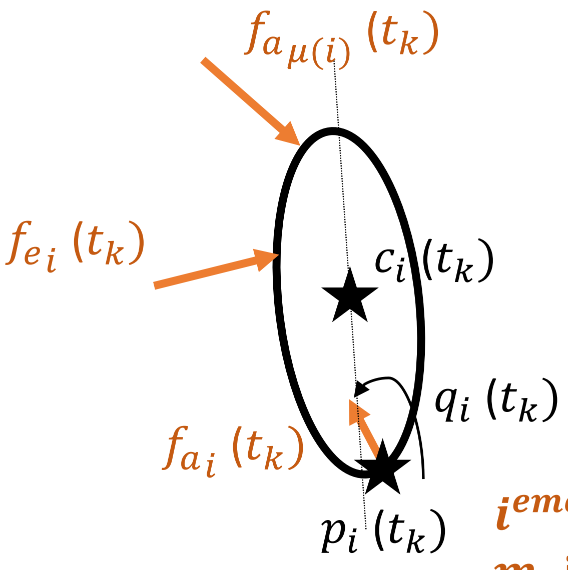
Spatial acceleration quantities computation for solid i

$$\boldsymbol{f}_{\boldsymbol{a}_i} = \begin{bmatrix} \boldsymbol{f}_i \\ \boldsymbol{\tau}_i \end{bmatrix} = \boldsymbol{f}_i^{acc} - \boldsymbol{f}_{\boldsymbol{e}_i} - \sum_{\mu(i)} \boldsymbol{f}_{\boldsymbol{a}_j}$$

Joint reaction forces between solid i and solid $i - 1$ computation

(Joint torque extraction $\boldsymbol{\tau}_i = \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix}^t \boldsymbol{f}_{\boldsymbol{a}_i}$)

$$\boldsymbol{\tau}_i^{(p_i)}.\boldsymbol{u}_i = \left(\boldsymbol{\tau}_i^{(0)} + \boldsymbol{f}_i \times \boldsymbol{p}_i\right).\boldsymbol{u}_i$$



i^{eme} solid at time t_k (mass m_i inertia I_i)

Summary

With synthetic notations: $\boldsymbol{s}_i = \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix}$ $\dot{\boldsymbol{s}}_i = \begin{bmatrix} \widehat{\boldsymbol{\omega}}_i & \widehat{\boldsymbol{v}}_{\boldsymbol{0}_i} \\ \boldsymbol{0} & \widehat{\boldsymbol{\omega}}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_i \times \boldsymbol{u}_i \\ \boldsymbol{u}_i \end{bmatrix}$

4 steps Newton-Euler algorithm

$$\left.\begin{array}{l} \boldsymbol{\xi}_i = \boldsymbol{\xi}_{i-1} + \boldsymbol{s}_i \dot{\boldsymbol{q}}_i \text{ (spatial velocity update)} \\ \dot{\boldsymbol{\xi}}_i = \dot{\boldsymbol{\xi}}_{i-1} + \boldsymbol{s}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{s}}_i \dot{\boldsymbol{q}}_i \text{ (spatial acceleration update)} \end{array}\right\} \text{Pour } i \text{ from1 to } n_B$$

$$\left.\begin{array}{l} \boldsymbol{f}_{\boldsymbol{a}_i} = \boldsymbol{I}_i^s \dot{\boldsymbol{\xi}}_i + \boldsymbol{\xi}_i \times \boldsymbol{I}_i^s \boldsymbol{\xi}_i - \boldsymbol{f}_{\boldsymbol{e}_i} + \sum_{\mu(i)} \boldsymbol{f}_{\boldsymbol{a}_j} \text{ (computing actions of } \lambda(i) \text{ on } i\text{)} \\ \tau_i = \boldsymbol{s}_i^T \boldsymbol{f}_{\boldsymbol{a}_i} \text{ (extracting joint torques)} \end{array}\right\} \text{Pour } i \text{ from } n_B \text{ to } 1$$

! This implementation ask for a knowledge of all quantities at point O