CusToM Workshop

Muscle forces tutorial

Charles Pontonnier, Pierre Puchaud

Pre-work

1. Go in Examples / 2_SideStep_Muscle
2. Copy/paste SideStep_Geometric_Calibration folder in 2_SideStep_Muscle and rename it SideStep_Muscle_Opti
3. Supress the BiomechanicalModel.mat file inside the folder SideStep_Muscle_Opti

Should look like this:

Examples folder

1_SideStep_Kinematic	26/11/2018 13:21	Dossier de fichiers	
2_SideStep_Muscle	28/11/2018 21:48	Dossier de fichiers	
3_SideStep_Force_Prediction	28/11/2018 21:48	Dossier de fichiers	
4_XSENS_VICON	28/11/2018 11:54	Dossier de fichiers	

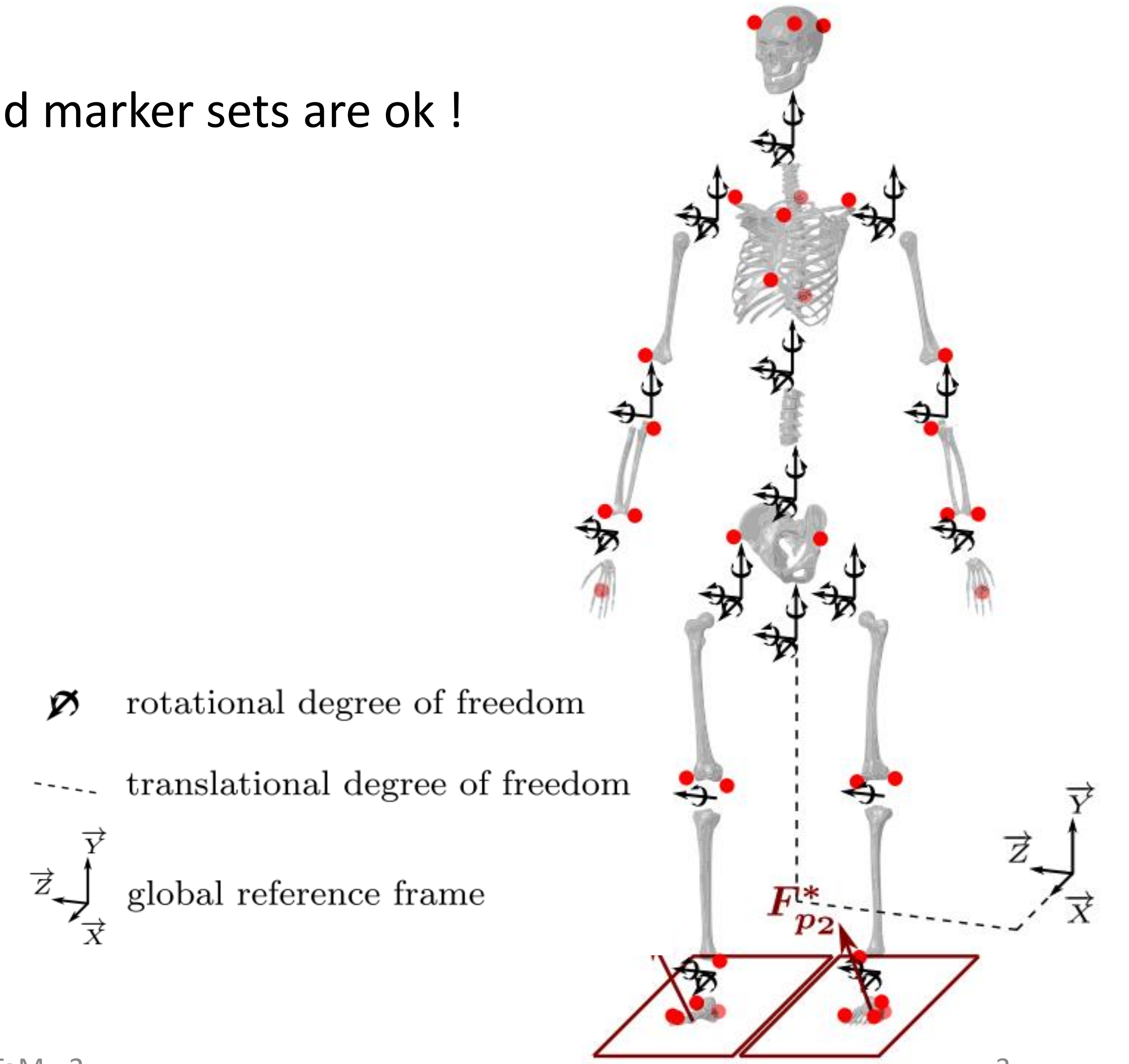
SideStep_Muscle_Opti	28/11/2018 21:51	Dossier de fichiers	
PostProcessingMuscles1.m	28/11/2018 21:38	MATLAB Code	3 Ko
PostProcessingMuscles2.m	28/11/2018 21:37	MATLAB Code	4 Ko

ChgtDirection04	28/11/2018 12:04	Dossier de fichiers	
Symbolic_function	28/11/2018 12:04	Dossier de fichiers	
AnalysisParameters.mat	28/11/2018 11:33	Fichier MAT	2 Ko
ChgtDirection04.c3d	26/11/2018 13:21	Fichier C3D	347 Ko
ModelParameters.mat	28/11/2018 11:33	Fichier MAT	1 Ko
ROM.c3d	28/11/2018 11:33	Fichier C3D	32 858 Ko

Generate Parameters of the Model

```
>> GenerateParameters
```

- First load parameters: size, mass, osteoarticular and marker sets are ok !
- Add muscles to right and left legs
- Classical Leg model **[Daamsgard2006]**
- **Generate Parameters**

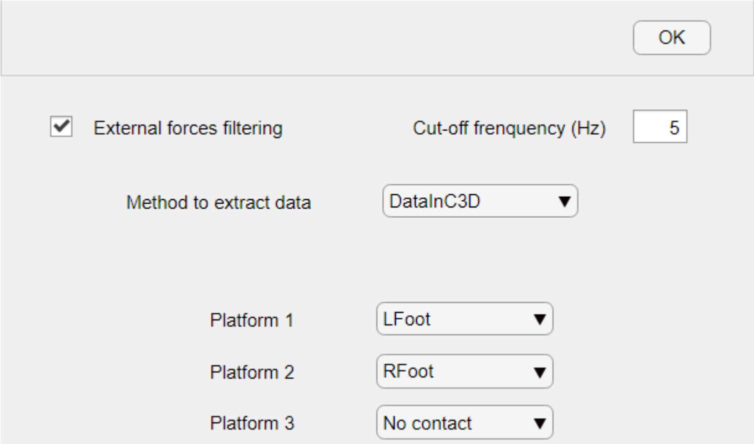


Generate AnalysisParameters

- First load parameters: kinematics are ok!
- Open Inverse Dynamics options: enable
- Select « From Experiments » for External forces
- Open Options, select « DataInC3D »
- Select « Lfoot » for Platform 1 and « Rfoot » for Platform 2

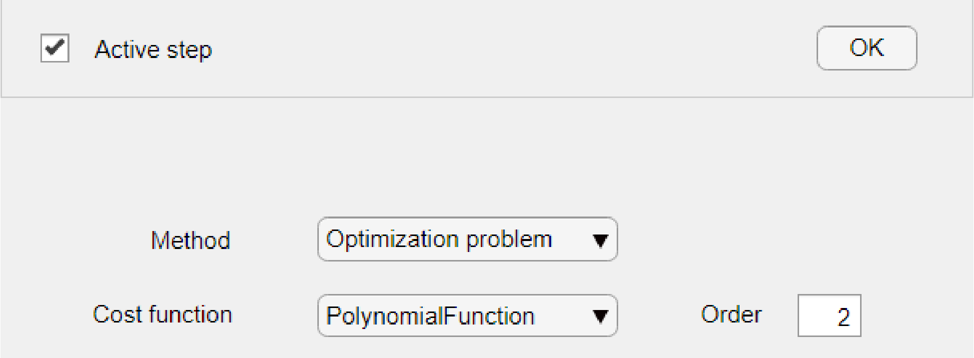
This is the source where external forces applied to the model will be read

This is a priori knowledge that you should know from your own experiments !!!!!



Generate AnalysisParameters

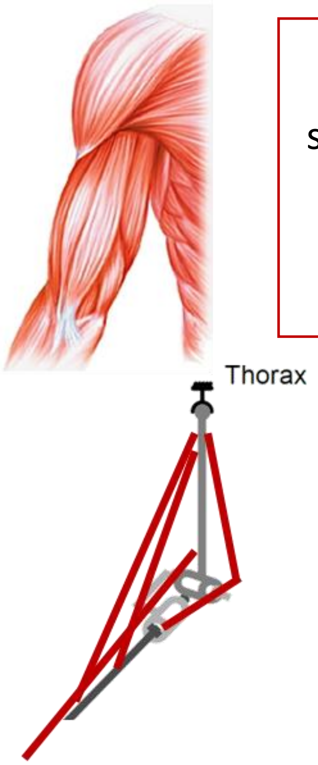
- First load parameters: kinematics are ok!
- Open « Inverse Dynamics » options: enable
- Select « From Experiments above » for External forces
- Open Options, select « DataInC3D »
- Select « Lfoot » for Platform 1 and « Rfoot » for Platform 2
- Open « Muscle forces estimation » Options
- Select **Optimization problem**, and Polynomial function (order 2)



We first go straight

What does this mean ?

The resulting problem to solve for MS forces estimation will be:

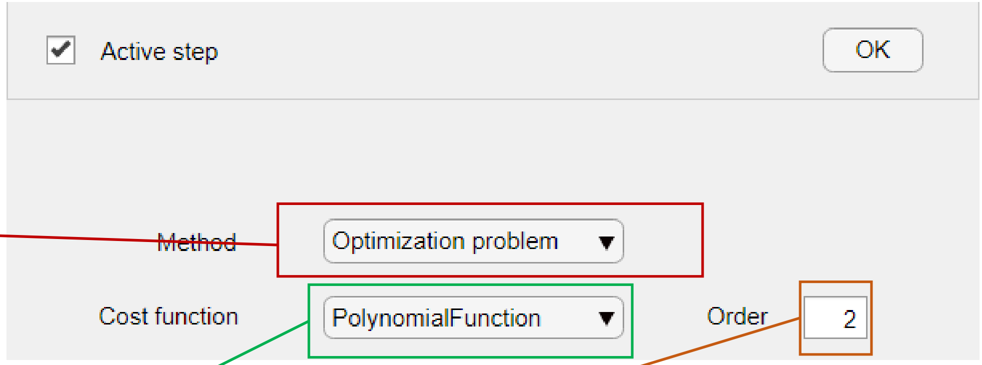


$$\min f(F)$$

$$\text{s.t.} \quad \vec{\tau} = \sum_i \overrightarrow{F_{m_i}} \times \overrightarrow{R_{m_i}}$$

$$F_{min_i} < F_{m_i} < F_{max_i}$$

$$i = 1 \dots n_m$$



$$f(\boldsymbol{F}) = \sum_{i=1}^{n_m} \left(\frac{F_{m_i}}{F_{max,i}} \right)^2$$

Alternatives

Change order to p ⟶ $f(\boldsymbol{F}) = \sum_{i=1}^{n_m} \left(\frac{F_{m_i}}{F_{max,i}}\right)^p$ ↗ p ↗ Muscle synergy

Change Polynomial Function to MinMax ⟶ $f(\boldsymbol{F}) = \max_{m \in [\![1:n_m]\!]} \left(\frac{F_m}{F_{max,m}}\right)$ ⟶ $p \to \infty$ Maximal Synergy

See Rasmussen, J., Damsgaard, M., & Voigt, M. (2001). Muscle recruitment by the min/max criterion—a comparative numerical study. *Journal of biomechanics*, *34*(3), 409-415.

RUN

What CusToM is doing ?

```
... Anthropometric Model Generation done
Geometrical Calibration ...
... Geometrical Calibration done
Preliminary Computations ...
... Preliminary Computations done
Moment Arms Computation ...
Starting parallel pool (parpool) using the 'local' profile ...
connected to 2 workers.
... Moment Arms Computation done
```

Not new

What CusToM is doing ?

```
... Anthropometric Model Generation done
Geometrical Calibration ...
... Geometrical Calibration done
Preliminary Computations ...
... Preliminary Computations done
```

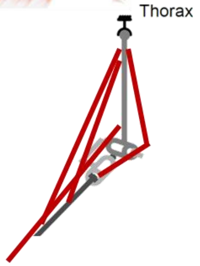
Not new

```
Moment Arms Computation ...
Starting parallel pool (parpool) using the 'local' profile ...
connected to 2 workers.
... Moment Arms Computation done
```

New

$$\vec{\tau} = \sum_i \overrightarrow{F_{m_i}} \times \overrightarrow{R_{m_i}}$$

Analytical solution computed and gathered as a matlab function



What CusToM is doing ?

Not new

Inverse kinematics (ChgtDirection04) ...
... Inverse kinematics (ChgtDirection04) done
Inverse dynamics (ChgtDirection04) ...
... Inverse dynamics (ChgtDirection04) done
Forces Computation (ChgtDirection04) ...
... Forces Computation (ChgtDirection04) done

What CusToM is doing ?

Inverse kinematics (ChgtDirection04) ...
... Inverse kinematics (ChgtDirection04) done

Inverse dynamics (ChgtDirection04) ...
... Inverse dynamics (ChgtDirection04) done

Forces Computation (ChgtDirection04) ...
... Forces Computation (ChgtDirection04) done

Not new

Newton-Euler Algorithm

Newton Euler-Algorithm

For more details, see 5_INVERSE-DYNAMICS.pdf

Charles Pontonnier, Pierre Puchaud

Main issue

At time t_k

External forces

$\boldsymbol{f_e}(t_k)$

Equations of motion

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

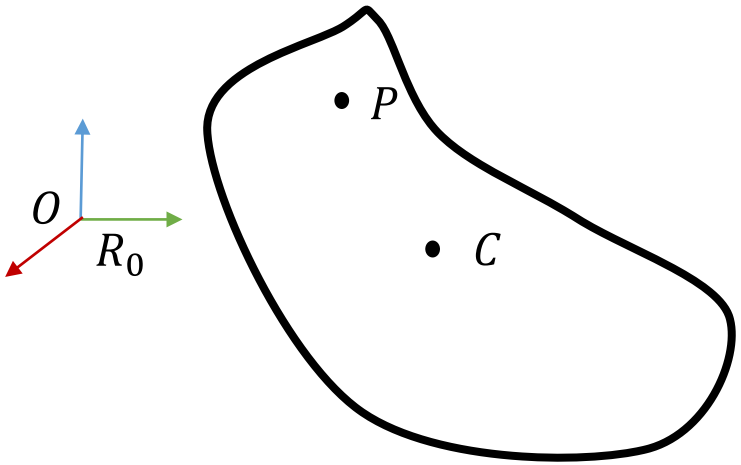
Joint torques

$\boldsymbol{\tau}(t_k)$

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

Angles, angular velocities and accelerations

Newton Euler equations for a solid S



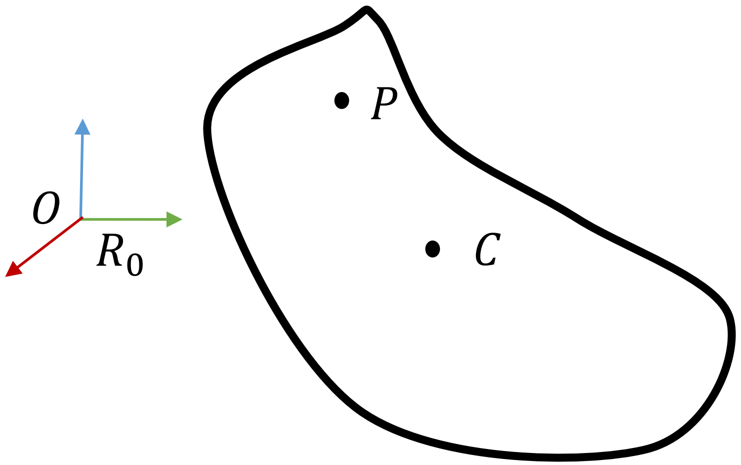
At center of mass

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

Acceleration momentum

$\boldsymbol{f}$	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 : $\boldsymbol{v_O} = \dot{\boldsymbol{c}} + \boldsymbol{c} \times \boldsymbol{\omega}$

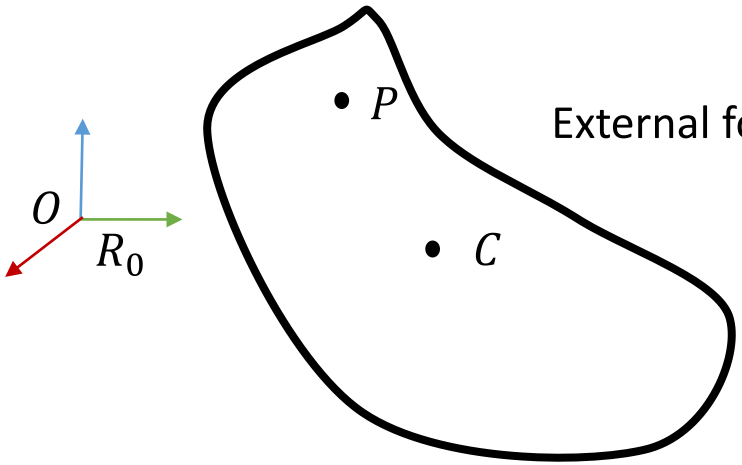
Acceleration of S in O: $\dot{\boldsymbol{v}}_{\boldsymbol{O}} = \ddot{\boldsymbol{c}} + \dot{\boldsymbol{c}} \times \boldsymbol{\omega} + \boldsymbol{c} \times \dot{\boldsymbol{\omega}}$

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

Replacing in (1), it comes

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

Spatial equations of motion [Featherstone2007]



External forces torque expressed at O: $\boldsymbol{\tau}^{(O)} = \boldsymbol{\tau}^{(c)} + \boldsymbol{c} \times \boldsymbol{f}$

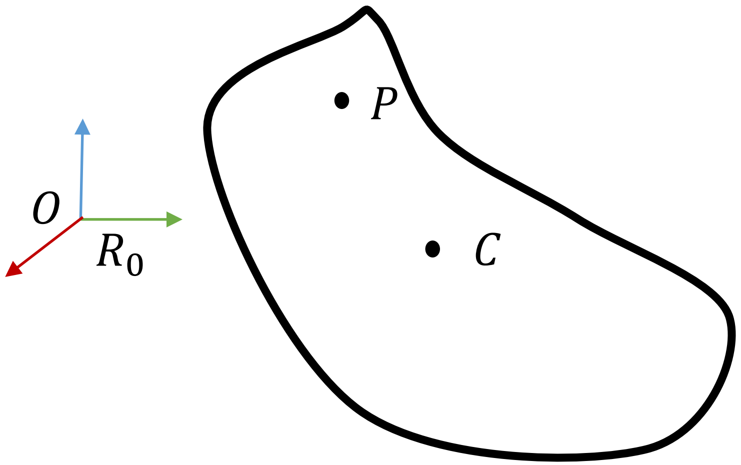
With $\boldsymbol{f} = m(\dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times \boldsymbol{c}))$

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

Finally:

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

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

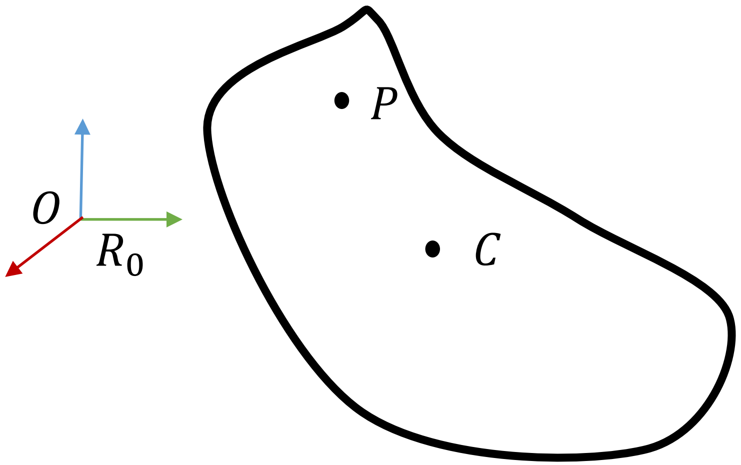
$$\boxed{\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} = \boldsymbol{I}^S \begin{bmatrix} \dot{\boldsymbol{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \widehat{\boldsymbol{\omega}} & \boldsymbol{0} \\ \widehat{\boldsymbol{v}_O} & \widehat{\boldsymbol{\omega}}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

(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:

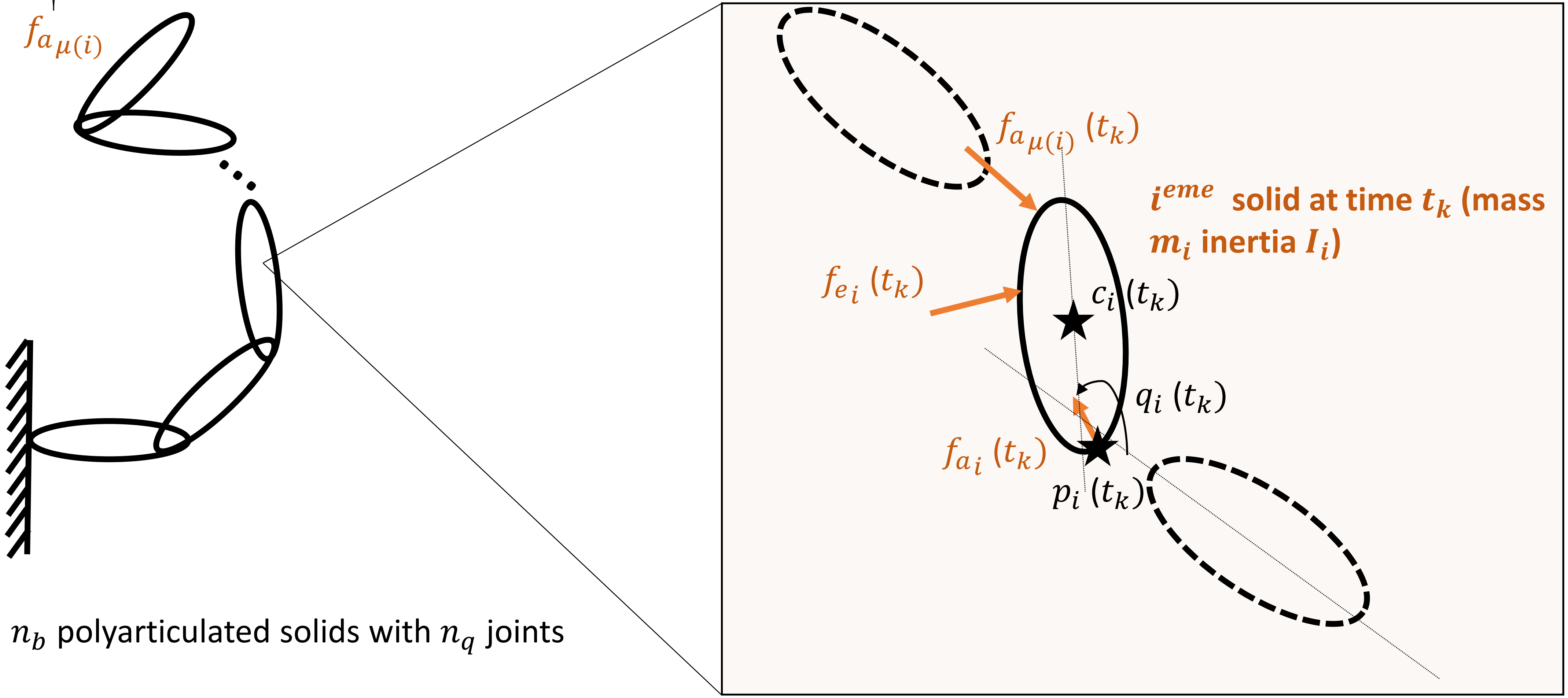
$$\boldsymbol{I}^s = \begin{bmatrix} m\mathbb{I} & m\hat{\boldsymbol{c}}^t \\ m\hat{\boldsymbol{c}} & m\hat{\boldsymbol{c}}\hat{\boldsymbol{c}}^t + \boldsymbol{I} \end{bmatrix}$$

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

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

End

For $i = n_B$ to 1 do

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

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

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

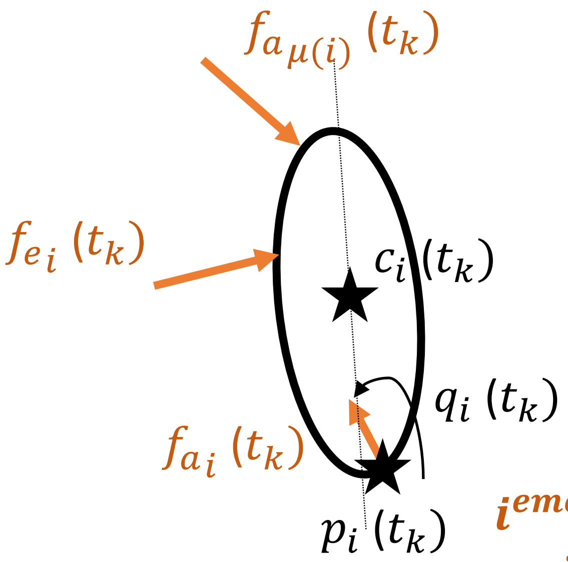
Spatial acceleration quantities computation for solid i

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

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

$\boldsymbol{\xi}_i = \boldsymbol{\xi}_{i-1} + \boldsymbol{s}_i \dot{\boldsymbol{q}}_i$ (spatial velocity update)

$\dot{\boldsymbol{\xi}}_i = \dot{\boldsymbol{\xi}}_{i-1} + \boldsymbol{s}_i \ddot{\boldsymbol{q}}_i + \dot{\boldsymbol{s}}_i \dot{\boldsymbol{q}}_i$ (spatial acceleration update)

Pour i from1 to n_B

$\boldsymbol{f}_{\boldsymbol{a}_i} = \boldsymbol{I}_i^s \, \dot{\boldsymbol{\xi}}_i + \boldsymbol{\xi}_i \times \boldsymbol{I}_i^s \boldsymbol{\xi}_i - \boldsymbol{f}_{\boldsymbol{e}_i} + \sum_{\mu(i)} \boldsymbol{f}_{\boldsymbol{a}_j}$ (computing actions of $\lambda(i)$ on i)

$\tau_i = \boldsymbol{s}_i^T \boldsymbol{f}_{\boldsymbol{a}_i}$ (extracting joint torques)

Pour i from n_B to 1

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

What CusToM is doing ?

Inverse kinematics (ChgtDirection04) ... ... Inverse kinematics (ChgtDirection04) done
Inverse dynamics (ChgtDirection04) ... ... Inverse dynamics (ChgtDirection04) done
Forces Computation (ChgtDirection04) ... ... Forces Computation (ChgtDirection04) done

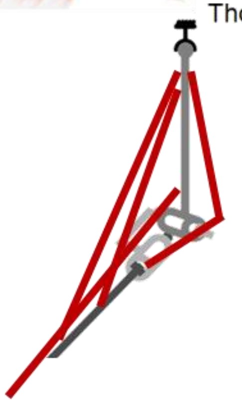
Not new

Newton-Euler Algorithm

Constrained optimization

Constrained optimization

At each frame, solve



$$\min f(F)$$

$$\text{s.t. } \vec{\tau} = \sum_i \overrightarrow{F_{m_i}} \times \overrightarrow{R_{m_i}}$$

$$F_{min_i} < F_{m_i} < F_{max_i}$$

$$i = 1 \ldots n_m$$

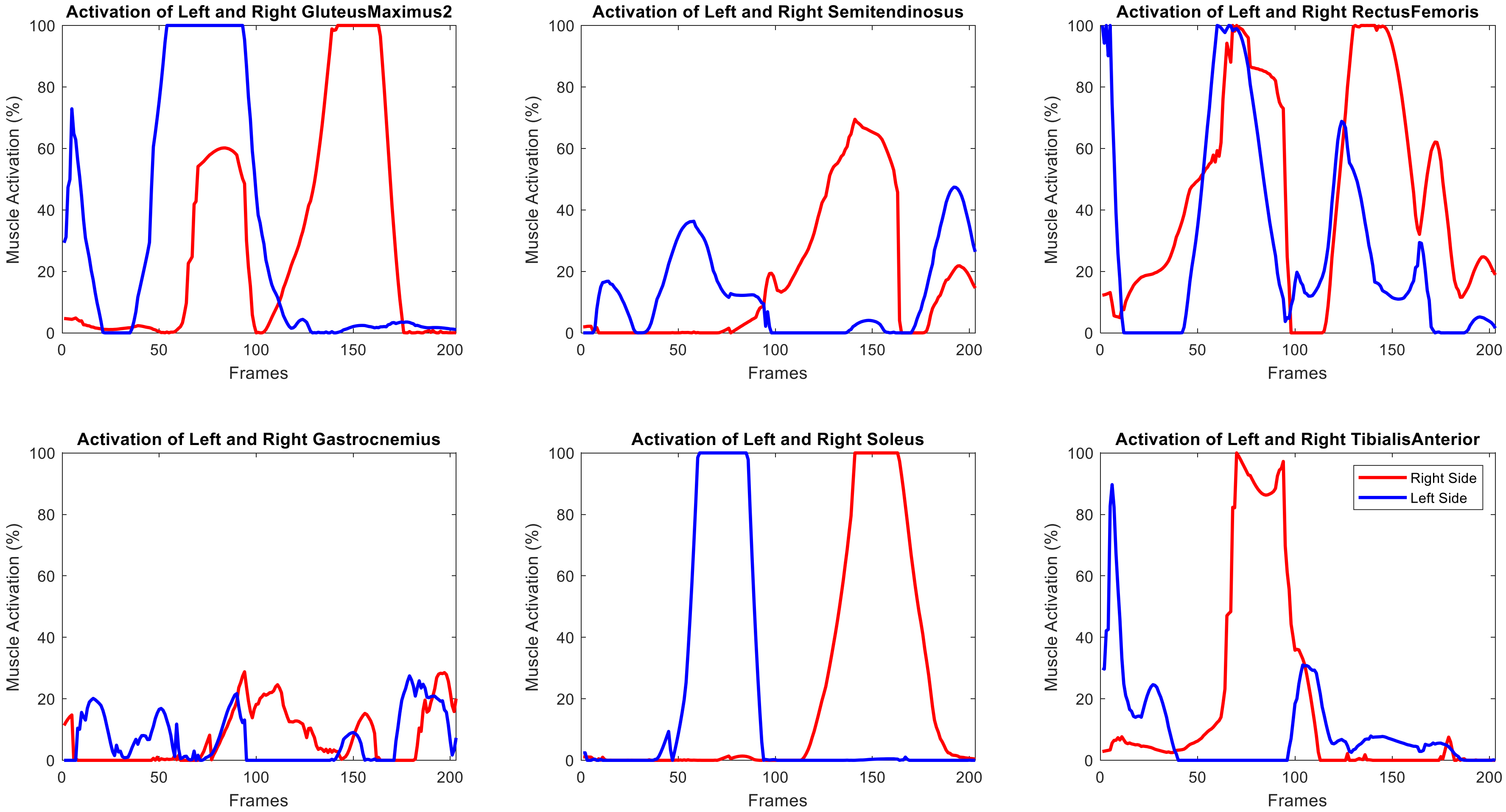
Inverse dynamics results !

Sequential Quadratic Programming Method

Replaces the cost function by a quadratic approximation and the constraints by linear approximations (and then active-set sucessive solutions until next step)

Results

>> PostProcessingMuscles1



Pierre will soon provide you EMG measures to compare !
#SupervisionBullyJoke

Muscle forces estimation

Alternative solution: the MusIC method

Pre-work

1. Copy/paste `SideStep_Muscle_Opti` folder in `2_SideStep_Muscle` and rename it `SideStep_Muscle_Music`
2. Supress the `BiomechanicalModel.mat` file inside the folder `SideStep_Muscle_Music`

Should look like this:

`Examples folder`

1_SideStep_Kinematic	26/11/2018 13:21	Dossier de fichiers	
2_SideStep_Muscle	28/11/2018 21:48	Dossier de fichiers	
3_SideStep_Force_Prediction	28/11/2018 21:48	Dossier de fichiers	
4_XSENS_VICON	28/11/2018 11:54	Dossier de fichiers	

SideStep_Muscle_MusIC	28/11/2018 12:28	Dossier de fichiers	
SideStep_Muscle_Opti	28/11/2018 11:58	Dossier de fichiers	
PostProcessingMuscles1.m	28/11/2018 21:38	MATLAB Code	3 Ko
PostProcessingMuscles2.m	28/11/2018 21:37	MATLAB Code	4 Ko

ChgtDirection04	28/11/2018 12:04	Dossier de fichiers	
Symbolic_function	28/11/2018 12:04	Dossier de fichiers	
AnalysisParameters.mat	28/11/2018 11:33	Fichier MAT	2 Ko
ChgtDirection04.c3d	26/11/2018 13:21	Fichier C3D	347 Ko
ModelParameters.mat	28/11/2018 11:33	Fichier MAT	1 Ko
ROM.c3d	28/11/2018 11:33	Fichier C3D	32 858 Ko

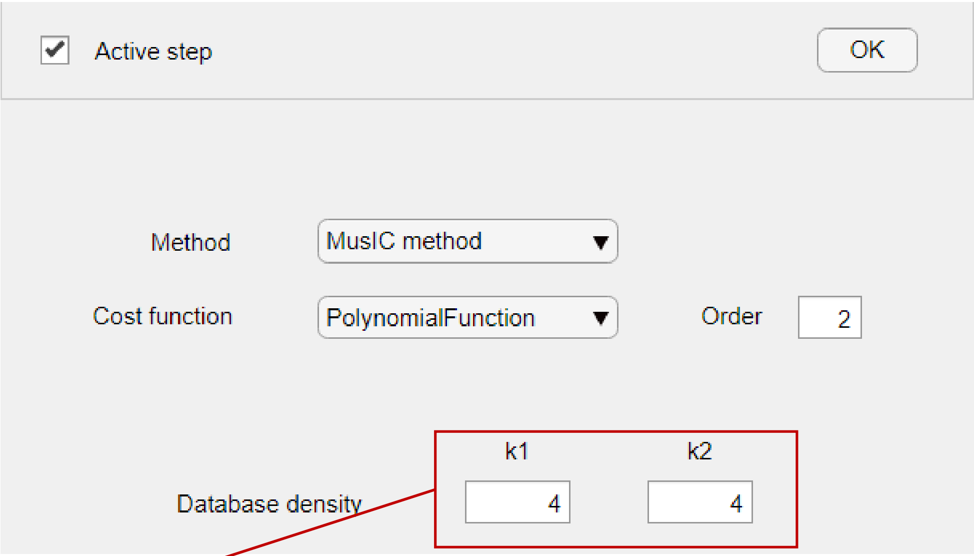
Generate model parameters

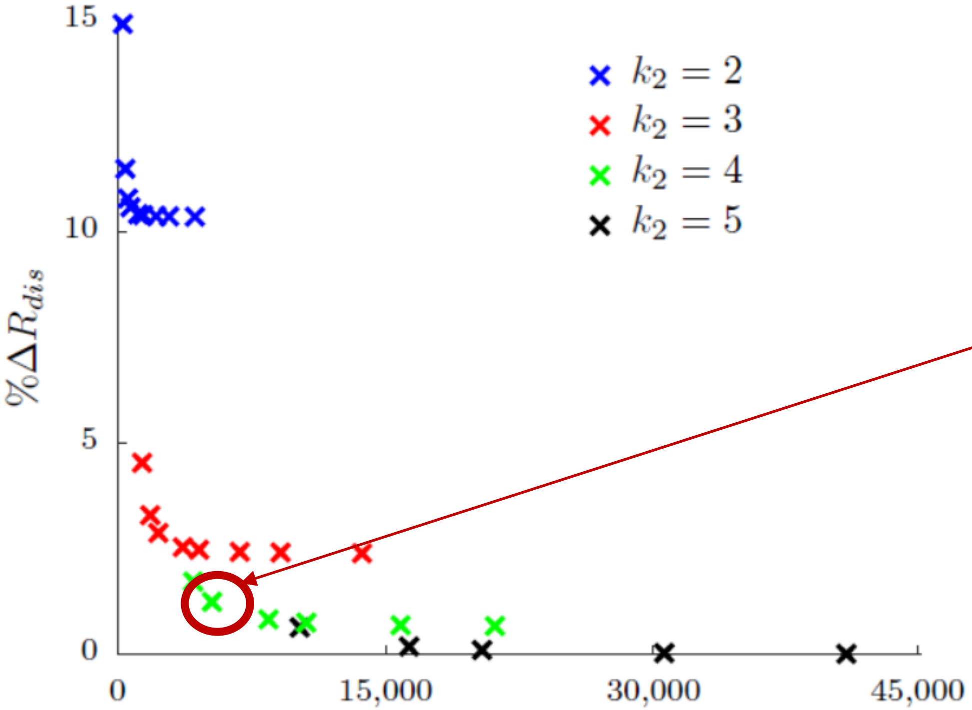
```
>> GenerateParameters
```

- Load the ModelParameters: everything is already fine

Generate AnalysisParameters

- First load parameters: kinematics and dynamics are ok!
- Open « Muscle forces estimation » Options
- Select **MusiC method**, and Polynomial function (order 2)
- In database density, select k1,k2 as 4,4





Guaranteeing small error, small computation time

Muller, A., Pontonnier, C., & Dumont, G. (2018). MusIC method enhancement by a sensitivity study of its performance: application to a lower limbs musculoskeletal model. *Computer Methods in Biomechanics and Biomedical Engineering*.

RUN

What CusToM is doing ?

Anthropometric Model Generation ...
... Anthropometric Model Generation done
Geometrical Calibration ...
... Geometrical Calibration done
Preliminary Computations ...
... Preliminary Computations done
Moment Arms Computation ...
... Moment Arms Computation done
MusIC Database Generation ...
... MusIC Database Generation done

Not new

What CusToM is doing ?

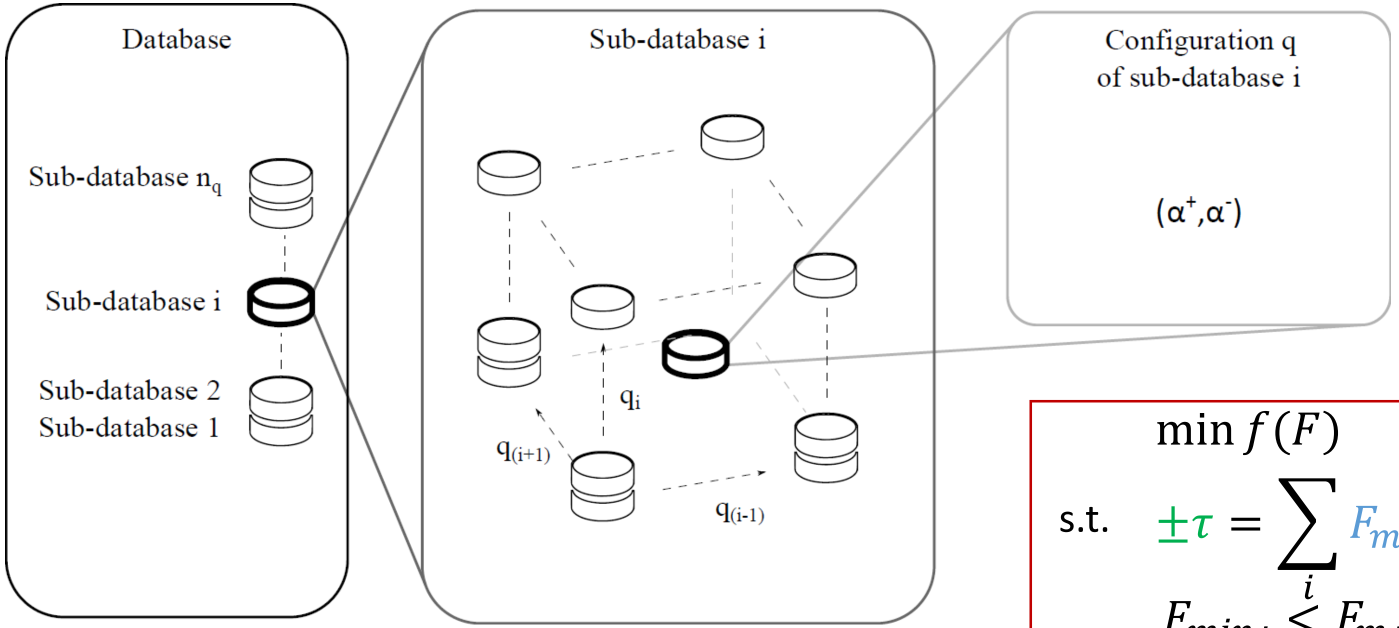
Anthropometric Model Generation ...
... Anthropometric Model Generation done
Geometrical Calibration ...
... Geometrical Calibration done
Preliminary Computations ...
... Preliminary Computations done
Moment Arms Computation ...
... Moment Arms Computation done

Not new

MusIC Database Generation ...
... MusIC Database Generation done

New

Database generation



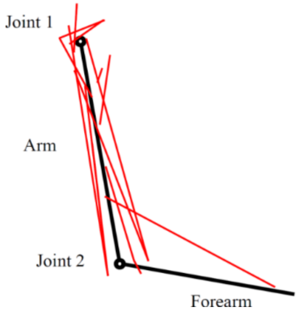
$$\min f(F)$$

$$\text{s.t.} \quad \pm\tau = \sum_i F_{m_i} R_{m_i}$$

$$F_{min_i} < F_{m_i} < F_{max_i}$$

$$i = 1 \dots n_m$$

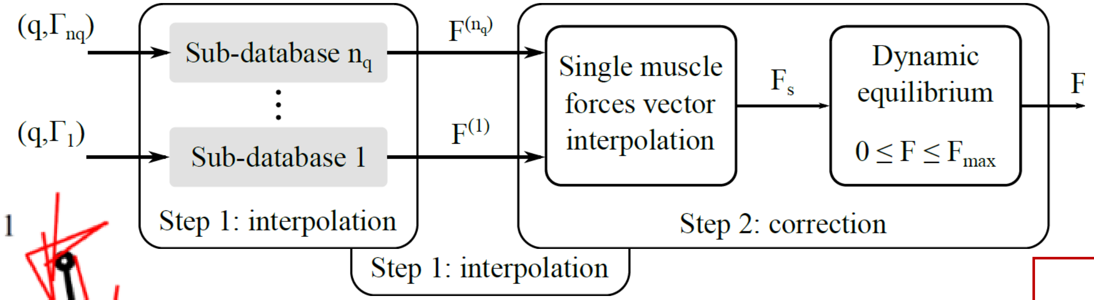
$$f(\boldsymbol{F}) = \sum_{i=1}^{n_m} \left(\frac{F_{m_i}}{F_{max,i}} \right)^2$$

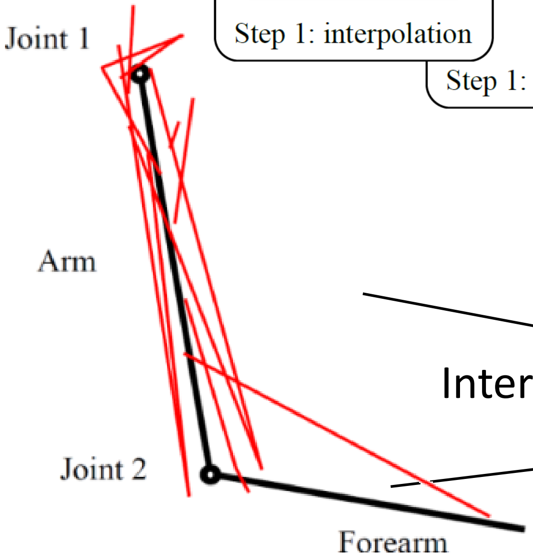


subdatabase

MusIC method

Compute forces from torques and joint configuration





Barycentric interpolation

Interpolation $F^{(1)}$, $F^{(2)}$ → F_S

$$\min f(F) = \|F - F_S\|$$

s.t. $$\pm\tau = \sum_i F_{m_i} R_{m_i}$$

$$F_{min_i} < F_{m_i} < F_{max_i}$$

$$i = 1 \ldots n_m$$

$F_S \rightarrow F$

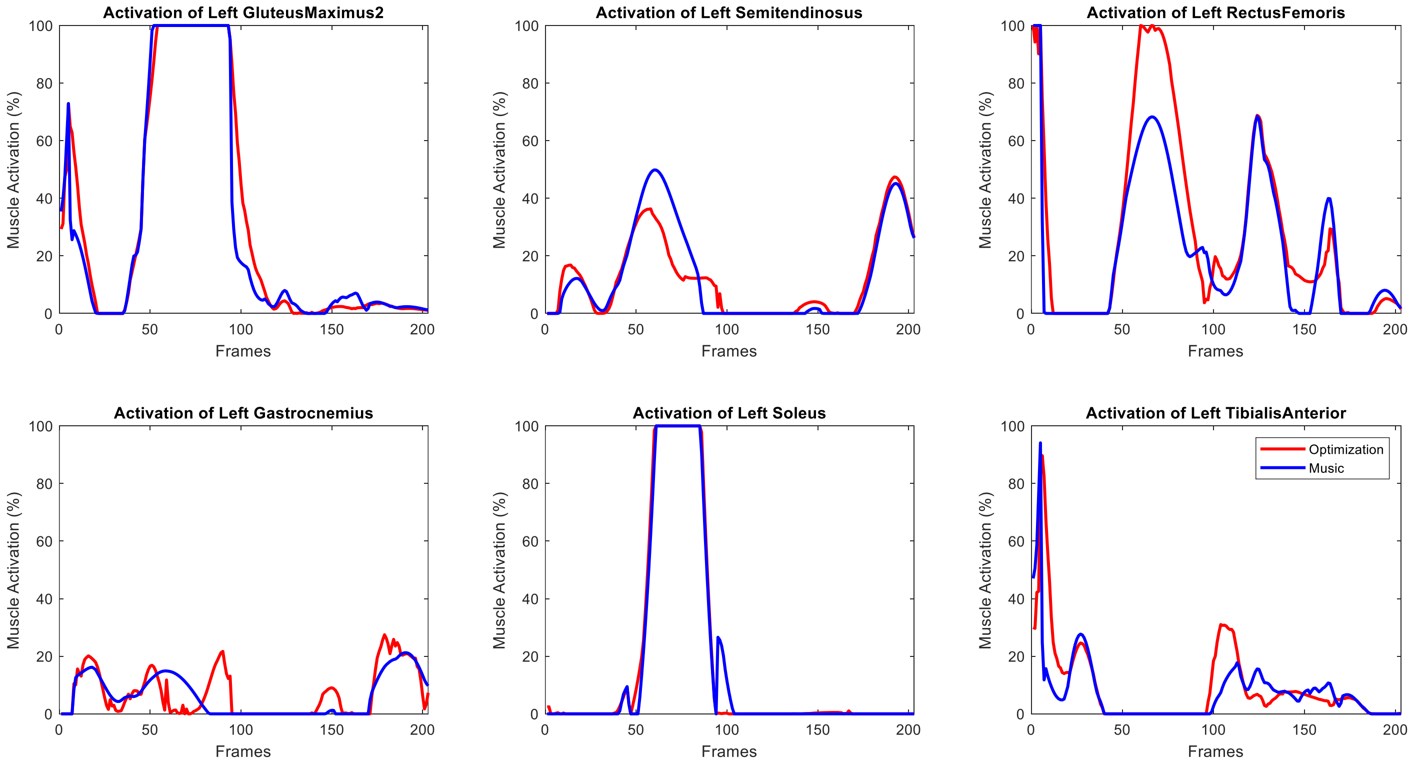
Active-set algorithm

Results

>> PostProcessingMuscles2

Maybe (4,4) is not that fine…

Pierre will soon provide you EMG measures to compare !
#SupervisionBullyJoke



Conclusion

- Be careful with the choice of the cost function to use in your analysis
- Be careful with the MusIC method
- **A wise choice between offline and online computation (if you have more than 25s of mocap to deal with, it is worth it)**
- **Choose wisely your model, thus you generate it once whatever the processing you want to run**