

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	



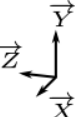
	SideStep_Muscle_Opti	28/11/2018 21:51	Dossier de fichiers	
	PostProcessingMuscles1.m	28/11/2018 21:38	MATLAB Code	3 Ko
	PostProcessingMuscles2.m	28/11/2018 21:37	MATLAB Code	4 Ko

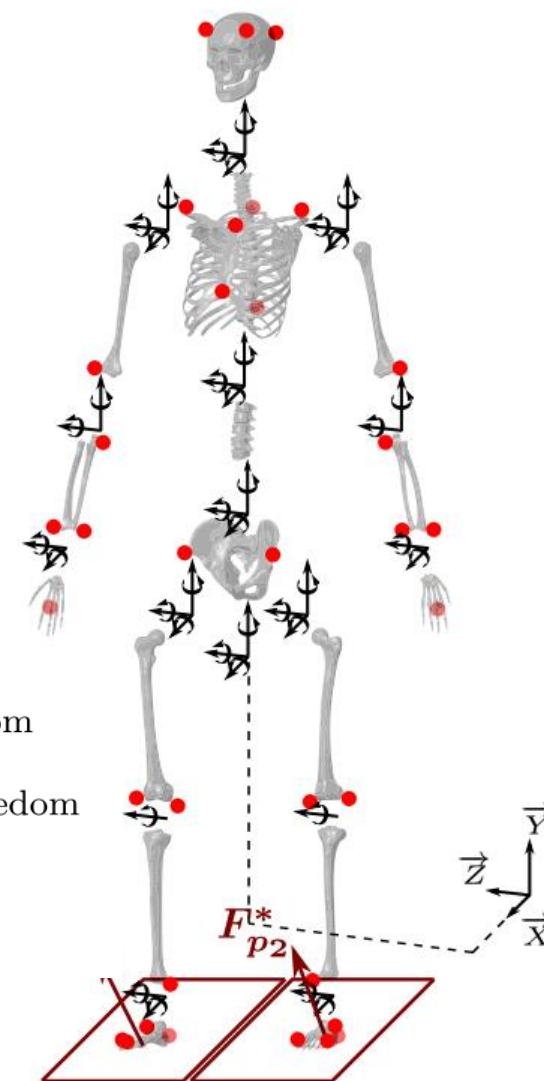
	ChgtDirection04	28/11/2018 12:04	Dossier de fichiers	
	Symbolic_function	28/11/2018 12:04	Dossier de fichiers	
	AnalysisParameters.mat	28/11/2018 11:33	Fichier MAT	2 Ko
	ChgtDirection04.c3d	26/11/2018 13:21	Fichier C3D	347 Ko
	ModelParameters.mat	28/11/2018 11:33	Fichier MAT	1 Ko
	ROM.c3d	28/11/2018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


 rotational degree of freedom
 translational degree of freedom
 global reference frame

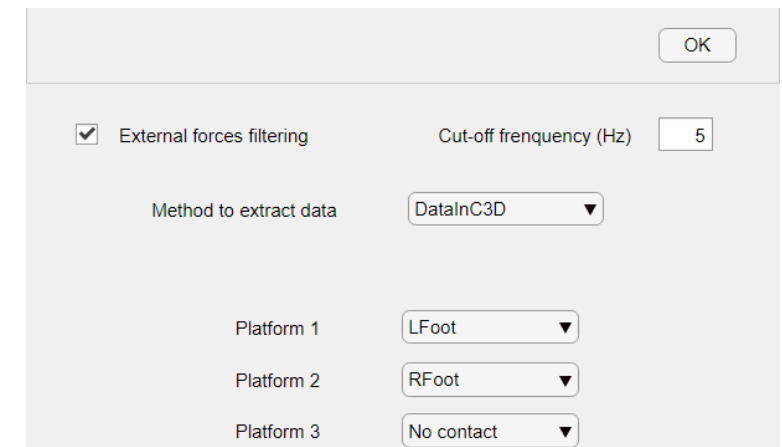


Generate AnalysisParameters

- 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

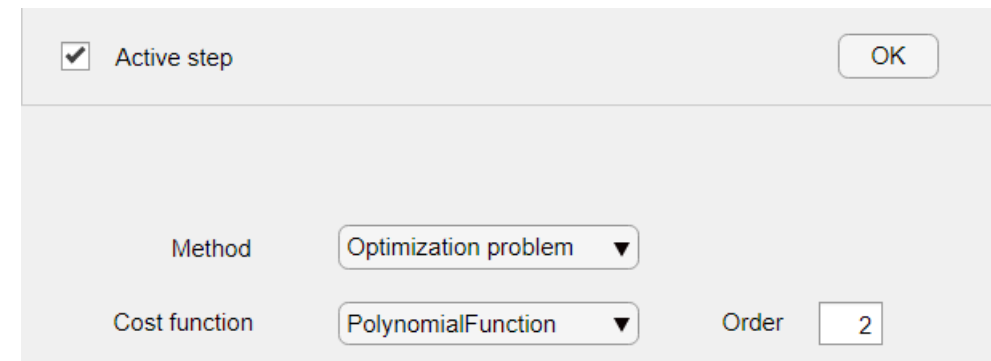

This is a priori knowledge that you should know from your own experiments !!!!!



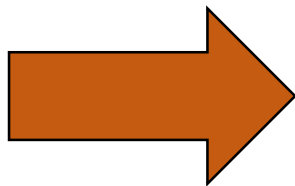
<input checked="" type="checkbox"/> External forces filtering		Cut-off frequency (Hz)	5
Method to extract data		DataInC3D ▼	
Platform 1	LFoot ▼		
Platform 2	RFoot ▼		
Platform 3	No contact ▼		

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)



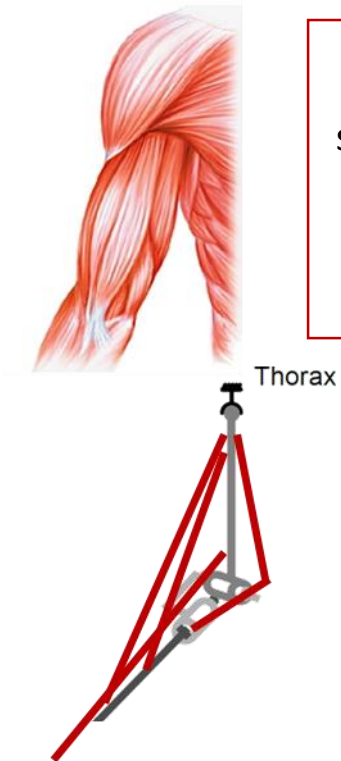
The screenshot shows a software interface for configuring an 'Active step'. At the top, there is a checkbox labeled 'Active step' which is checked, and an 'OK' button to its right. Below this, the 'Method' is set to 'Optimization problem' via a dropdown menu. The 'Cost function' is set to 'PolynomialFunction' via another dropdown menu. To the right of the cost function dropdown, the 'Order' is specified as '2' in a text input field.



We first go straight

What does this mean ?

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



$$\begin{aligned} \min f(F) \\ \text{s.t. } \vec{\tau} &= \sum_i \vec{F}_{m_i} \times \vec{R}_{m_i} \\ F_{\min_i} &< F_{m_i} < F_{\max_i} \\ i &= 1 \dots n_m \end{aligned}$$

☒ Active step OK

Method: Optimization problem

Cost function: PolynomialFunction

Order: 2

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

Alternatives

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

Change Polynomial Function to MinMax \longrightarrow $f(\mathbf{F}) = \max_{m \in \llbracket 1:n_m \rrbracket} \left(\frac{F_m}{F_{max,m}} \right)$ \longrightarrow $p \rightarrow \infty$ 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

$$\vec{\tau} = \sum_i \vec{F}_{m_i} \times \vec{R}_{m_i}$$

Analytical solution computed and gathered as a matlab function



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

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

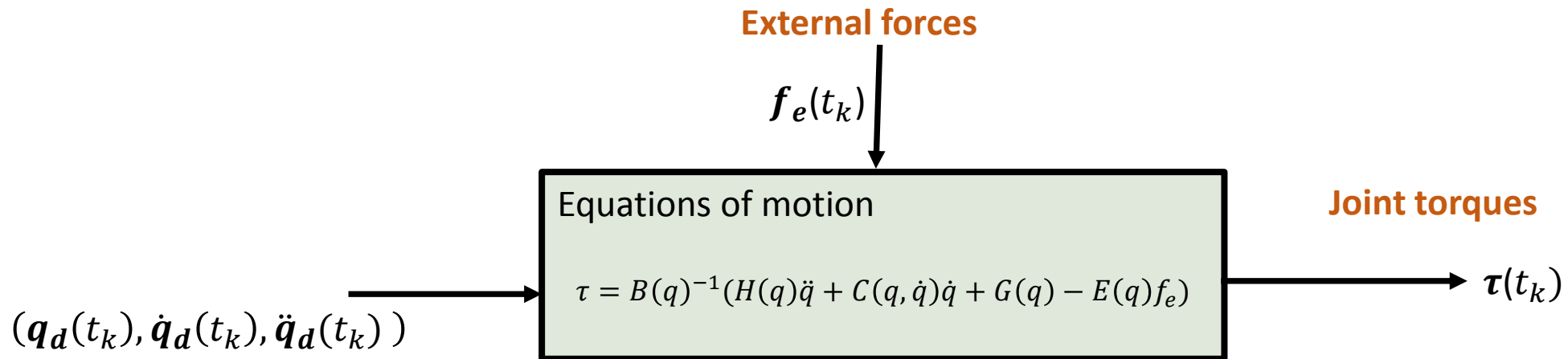
Newton Euler-Algorithm

For more details, see [5_INVERSE-DYNAMICS.pdf](#)

Charles Pontonnier, Pierre Puchaud

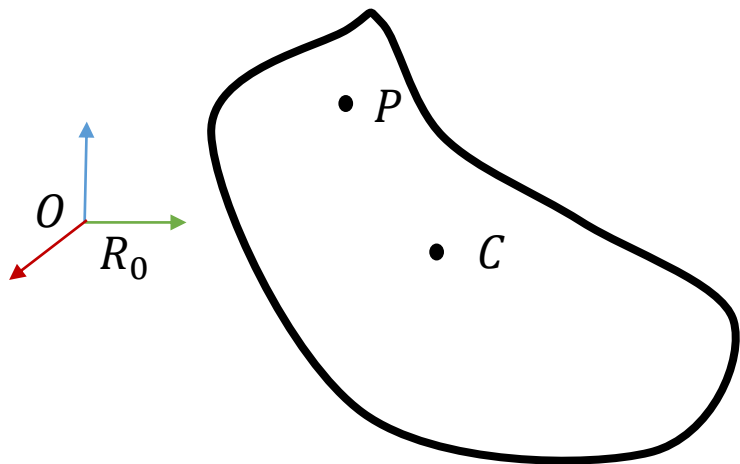
Main issue

At time t_k



Angles, angular velocities and accelerations

Newton Euler equations for a solid S



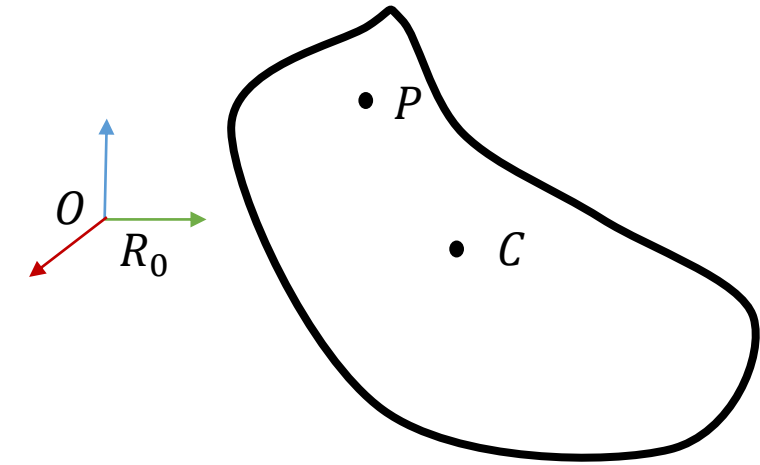
At center of mass

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

Acceleration momentum

\mathbf{f}	external forces
m	solid mass
\mathbf{c}	center of mass of the solid in R_0 (world) frame
$\boldsymbol{\omega}$	angular velocity of the solid in R_0
\mathbf{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 :

$$\mathbf{v}_O = \dot{\mathbf{c}} + \mathbf{c} \times \boldsymbol{\omega}$$

Acceleration of S in O :

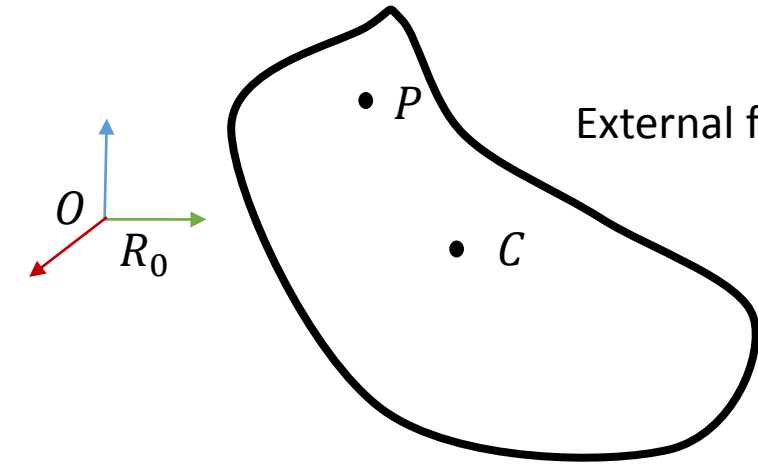
$$\dot{\mathbf{v}}_O = \ddot{\mathbf{c}} + \dot{\mathbf{c}} \times \boldsymbol{\omega} + \mathbf{c} \times \dot{\boldsymbol{\omega}}$$

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

Replacing in (1), it comes

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

Spatial equations of motion [Featherstone2007]



External forces torque expressed at O :

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

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

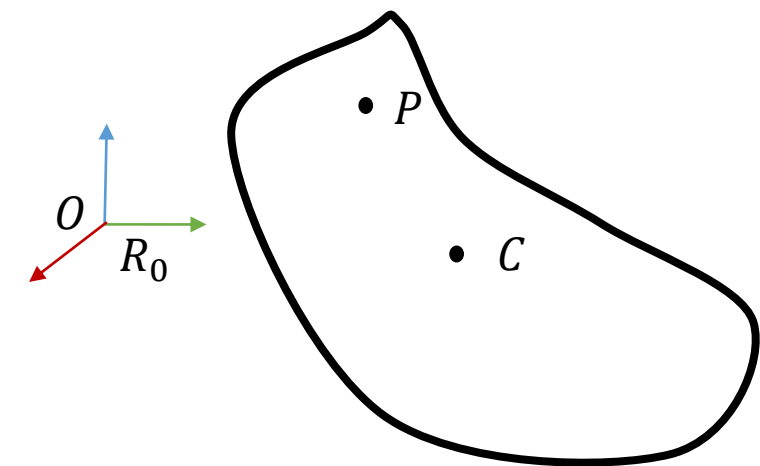
And

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

Finally:

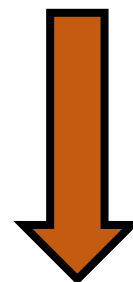
$$\boldsymbol{\tau}^{(O)} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} + \boldsymbol{c} \times m(\dot{\boldsymbol{v}}_O - \boldsymbol{c} \times \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{v}_O + \boldsymbol{\omega} \times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

Spatial equations of motion [Featherstone2007]



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

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



That can be expressed
with a matrix expression

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

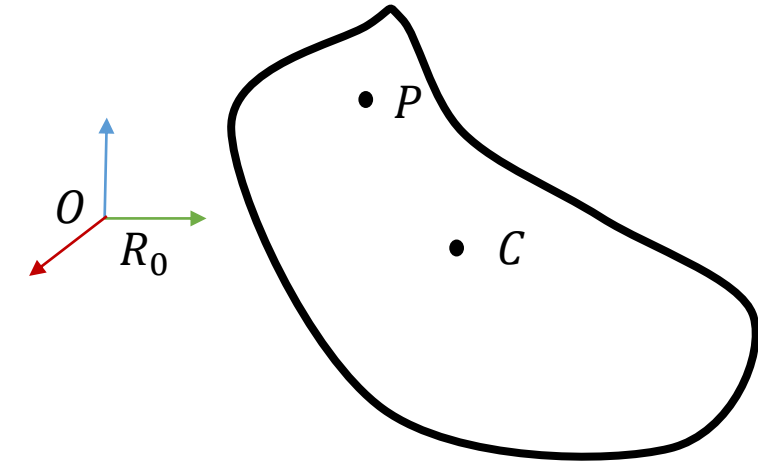


(application produit vectoriel)

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

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

Spatial inertia and spatial acceleration



\mathbb{I} identity matrix

Spatial inertia matrix: 6 x 6 Symmetrical matrix:

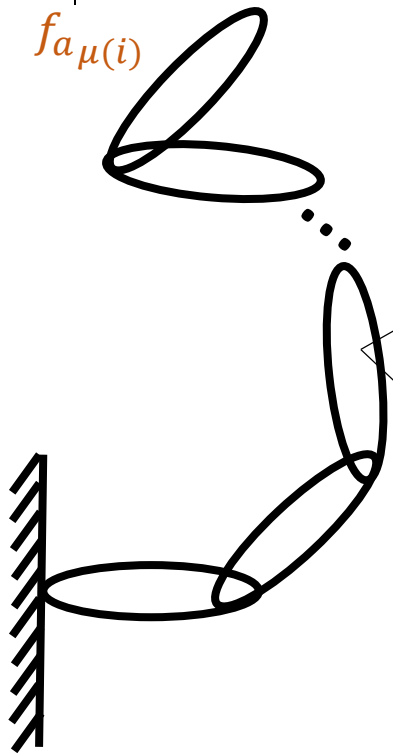
$$\mathbf{I}^s = \begin{bmatrix} m\mathbb{I} & m\hat{\mathbf{c}}^t \\ m\hat{\mathbf{c}} & m\hat{\mathbf{c}}\hat{\mathbf{c}}^t + \mathbf{I} \end{bmatrix}$$

Spatial acceleration: not a physical acceleration

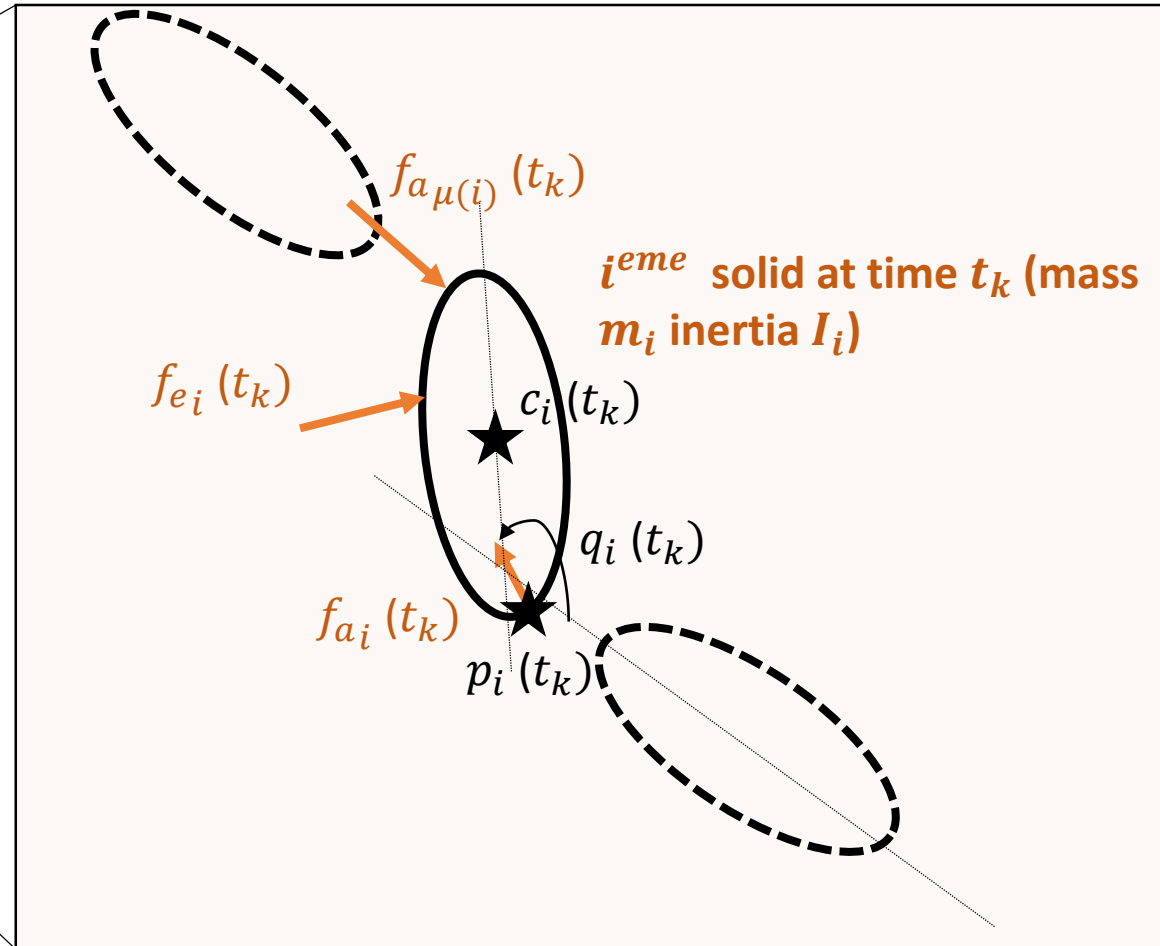
$$\begin{bmatrix} \dot{\mathbf{v}}_O \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \ddot{\boldsymbol{\xi}}$$

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

$$\underbrace{\begin{bmatrix} f_i \\ \tau_i \end{bmatrix}}_{f_{ai}} - \underbrace{\begin{bmatrix} f_{i+1} \\ \tau_{i+1} \end{bmatrix}}_{f_{a\mu(i)}} + f_{ei} = I_i^S \begin{bmatrix} \dot{v}_{0i} \\ \dot{\omega}_i \end{bmatrix} + \begin{bmatrix} v_{0i} \\ \omega_i \end{bmatrix} \times I_i^S \begin{bmatrix} v_{0i} \\ \omega_i \end{bmatrix}$$



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  
     $\dot{\xi}_i = f(q, \dot{q}, \ddot{q}, f_{e_i}, \dot{\xi}_{\lambda(i)})$   
End  
For  $i = n_B$  to 1 do  
     $f_{a_i} = f(\dot{\xi}_i, f_{e_i}, f_{a_{\mu(i)}})$   
End
```

Newton-Euler algorithm

For $i = 1$ to n_B do

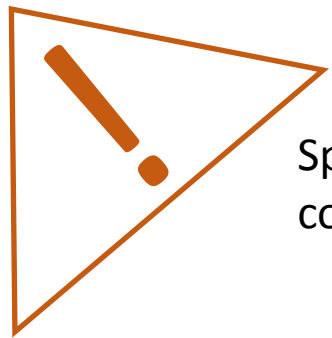
$$\dot{\xi}_i = f(q, \dot{q}, \ddot{q}, f_{e_i}, \dot{\xi}_{\lambda(i)})$$

End

For $i = n_B$ to 1 do

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

End



Spatial \rightarrow all is
computed in O



$$\left. \begin{aligned} {}^0R_i &= {}^0R_{\lambda(i)} {}^{\lambda(i)}R_i(q_i) \\ p_i &= p_{\lambda(i)} + {}^0R_{\lambda(i)} b_i \end{aligned} \right\}$$

**Solid position and
orientation update**

$${}^{(0)}u_i = {}^0R_{\lambda(i)} {}^{\lambda(i)}u_i$$

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

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

Spatial velocity update

$$\dot{\xi}_i = \dot{\xi}_{\lambda(i)} + \begin{bmatrix} \hat{\omega}_i & \hat{v}_{0_i} \\ 0 & \hat{\omega}_i \end{bmatrix} \begin{bmatrix} p_i \times u_i \\ u_i \end{bmatrix} \dot{q}_i + \begin{bmatrix} p_i \times u_i \\ u_i \end{bmatrix} \ddot{q}_i$$

Spatial acceleration update

Newton-Euler algorithm

For $i = 1$ to n_B do

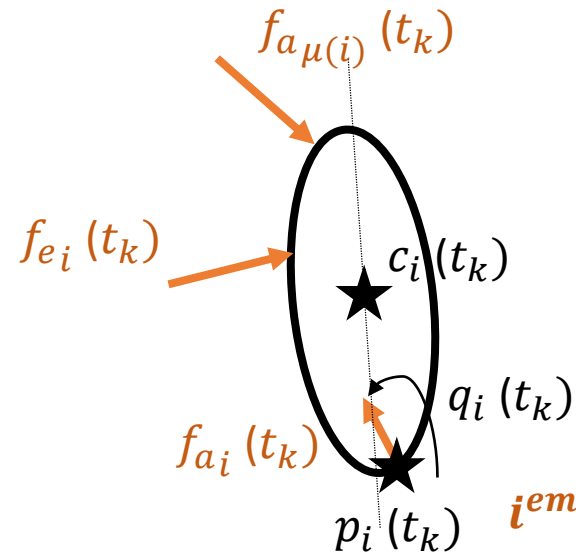
$$\dot{\xi}_i = f(q, \dot{q}, \ddot{q}, f_{e_i}, \dot{\xi}_{\lambda(i)})$$

End

For $i = n_B$ to 1 do

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

End



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

$${}^{(0)}c_i = p_i + {}^0R_i{}^{i-1}c_i$$

Center of mass position update for solid i

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

Spatial inertia matrix computation for solid i

$$f_i^{acc} = I_i^s \dot{\xi}_i + \xi_i \times I_i^s \xi_i$$

Spatial acceleration quantities computation for solid i

$$f_{a_i} = \begin{bmatrix} f_i \\ \tau_i \end{bmatrix} = f_i^{acc} - f_{e_i} - \sum_{\mu(i)} f_{a_j}$$

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

$$(\text{Joint torque extraction } \tau_i = \begin{bmatrix} p_i \times u_i \\ u_i \end{bmatrix}^t f_{a_i})$$

$$\tau_i^{(p_i)} \cdot u_i = (\tau_i^{(0)} + f_i \times p_i) \cdot u_i$$

Summary

With synthetic notations:

$$\mathbf{s}_i = \begin{bmatrix} \mathbf{p}_i \times \mathbf{u}_i \\ \mathbf{u}_i \end{bmatrix} \quad \dot{\mathbf{s}}_i = \begin{bmatrix} \hat{\boldsymbol{\omega}}_i & \hat{\mathbf{v}}_{0_i} \\ \mathbf{0} & \hat{\boldsymbol{\omega}}_i \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \times \mathbf{u}_i \\ \mathbf{u}_i \end{bmatrix}$$

4 steps Newton-Euler algorithm

$\xi_i = \xi_{i-1} + \mathbf{s}_i \dot{\mathbf{q}}_i$ (spatial velocity update)

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

$\mathbf{f}_{a_i} = \mathbf{I}_i^s \dot{\xi}_i + \xi_i \times \mathbf{I}_i^s \xi_i - \mathbf{f}_{e_i} + \sum_{\mu(i)} \mathbf{f}_{a_j}$ (computing actions of $\lambda(i)$ on i)

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

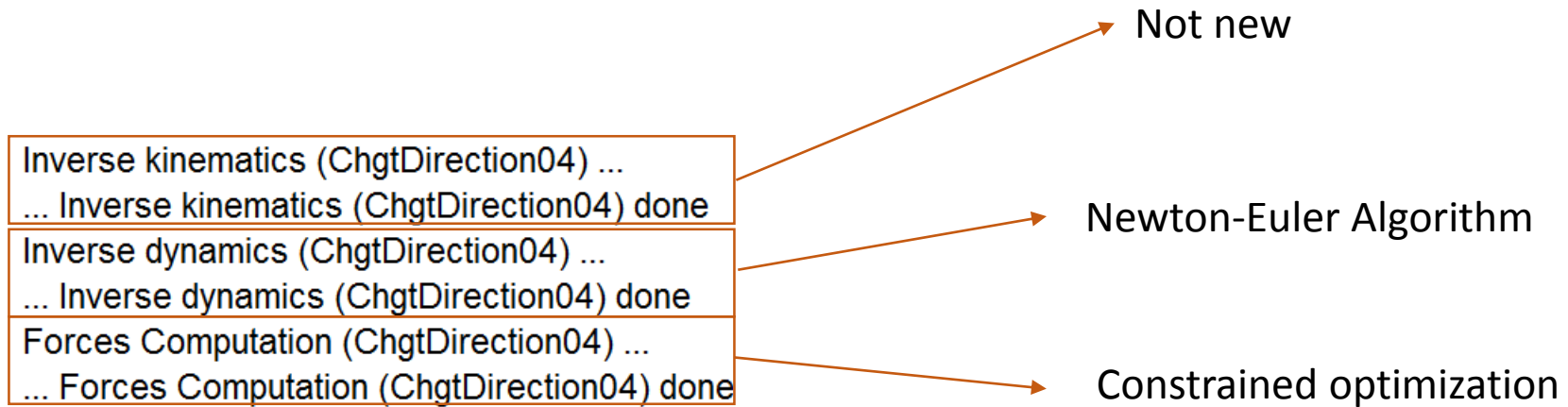
Pour i from 1 to n_B

Pour i from n_B to 1



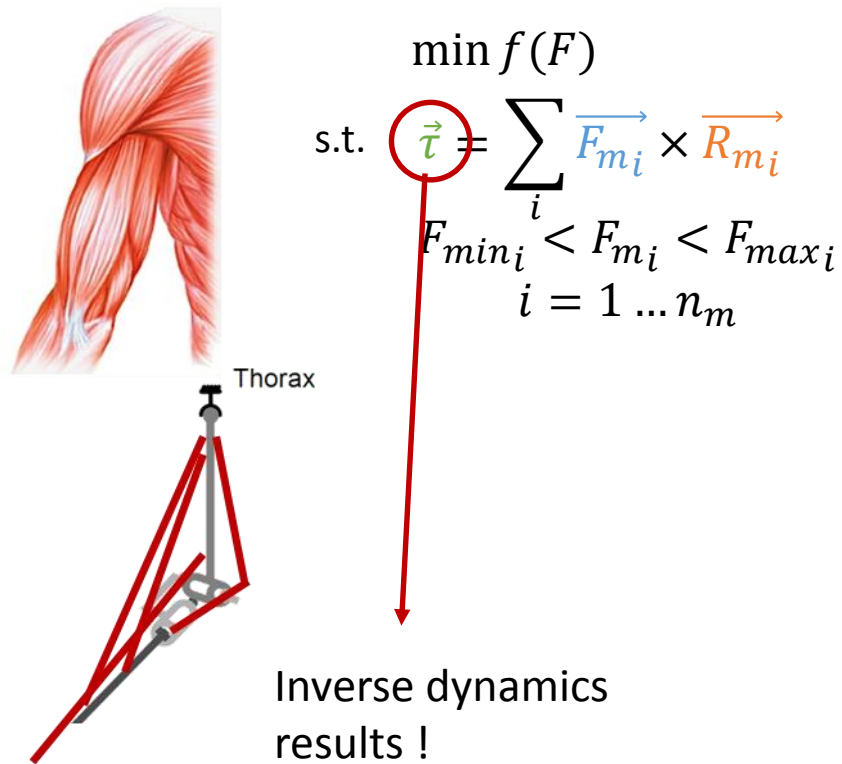
This implementation ask for a knowledge of all quantities at point O

What CusToM is doing ?



Constrained optimization

At each frame, solve



Sequential Quadratic Programming Method

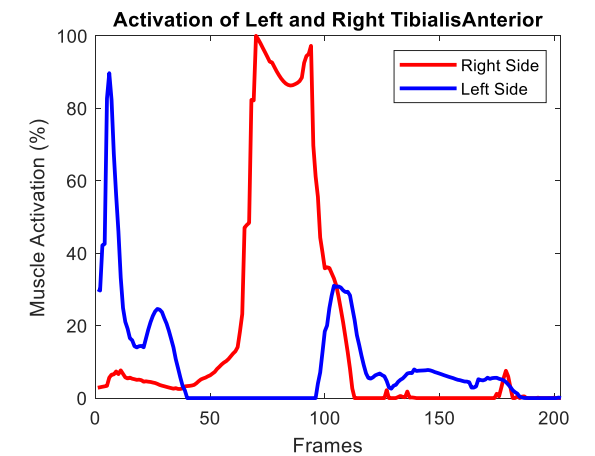
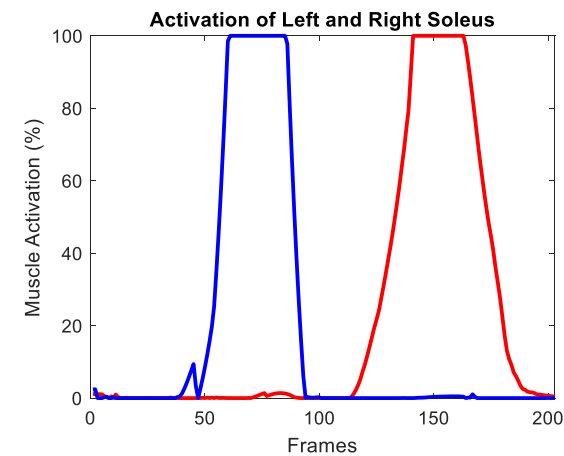
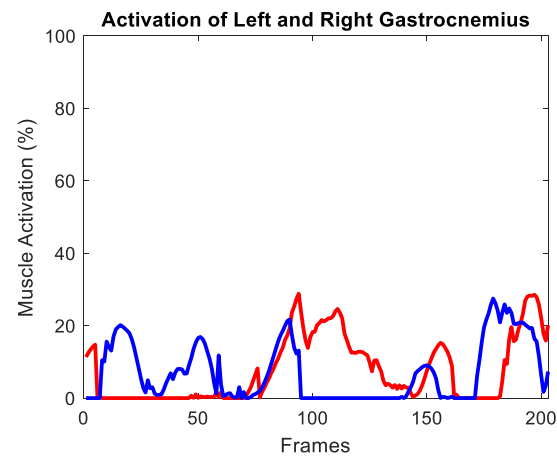
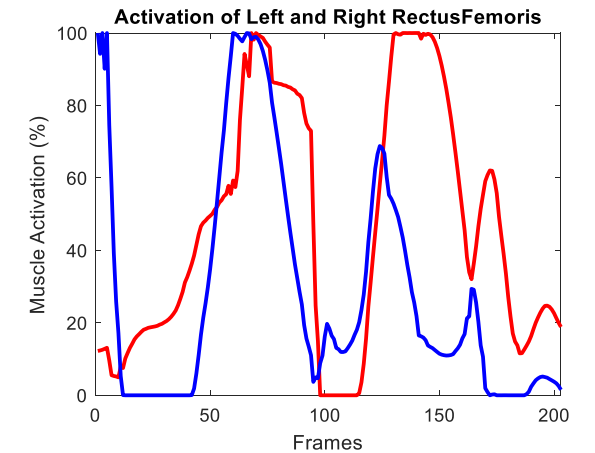
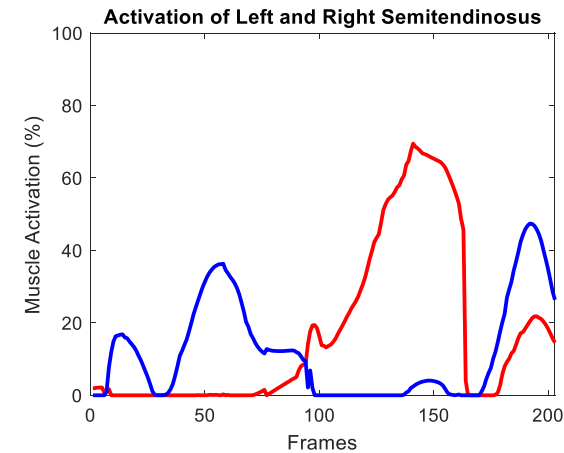
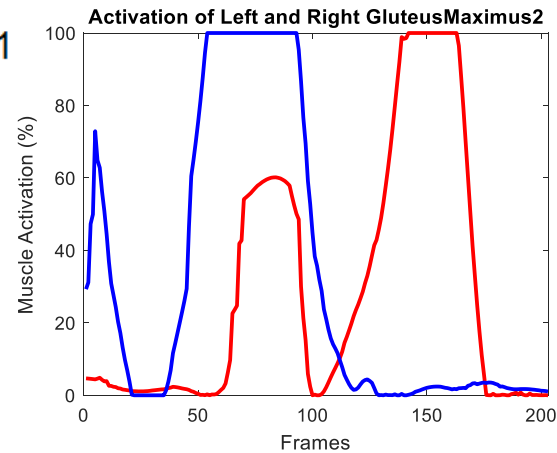
Replaces the cost function by a quadratic approximation and the constraints by linear approximations (and then active-set successive solutions until next step)

Results

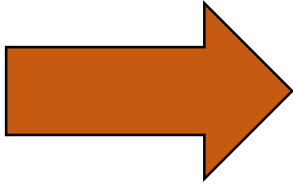
>> PostProcessingMuscles1

Pierre will soon provide you
EMG measures to compare !

#SupervisionBullyJoke



Muscle forces estimation



Alternative solution: the MuslC 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_MuslC	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

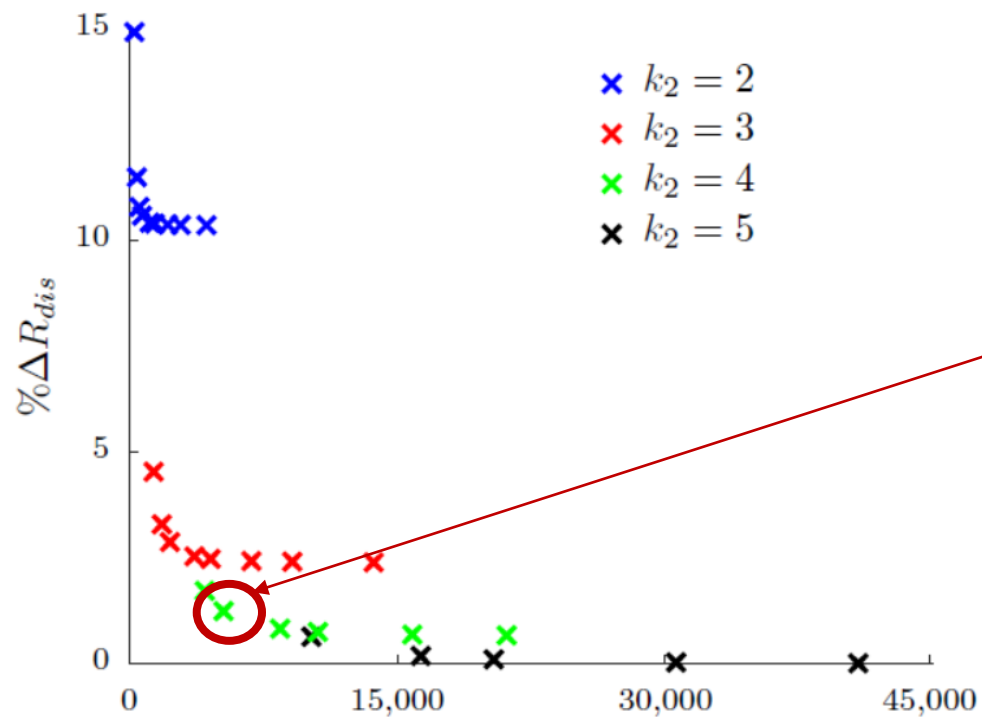
Generate model parameters

```
>> 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



The image shows a software interface for generating analysis parameters. It includes a checkbox for 'Active step' which is checked. Below this, there are two dropdown menus: 'Method' set to 'MuslC method' and 'Cost function' set to 'PolynomialFunction'. To the right of the 'Cost function' dropdown is a text box for 'Order' set to '2'. At the bottom, there is a section labeled 'Database density' with two input boxes for 'k1' and 'k2', both set to '4'. A red box highlights these two input boxes, and a red line connects this box to the highlighted data point in the scatter plot above.

Guaranteeing small error, small computation time

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

RUN

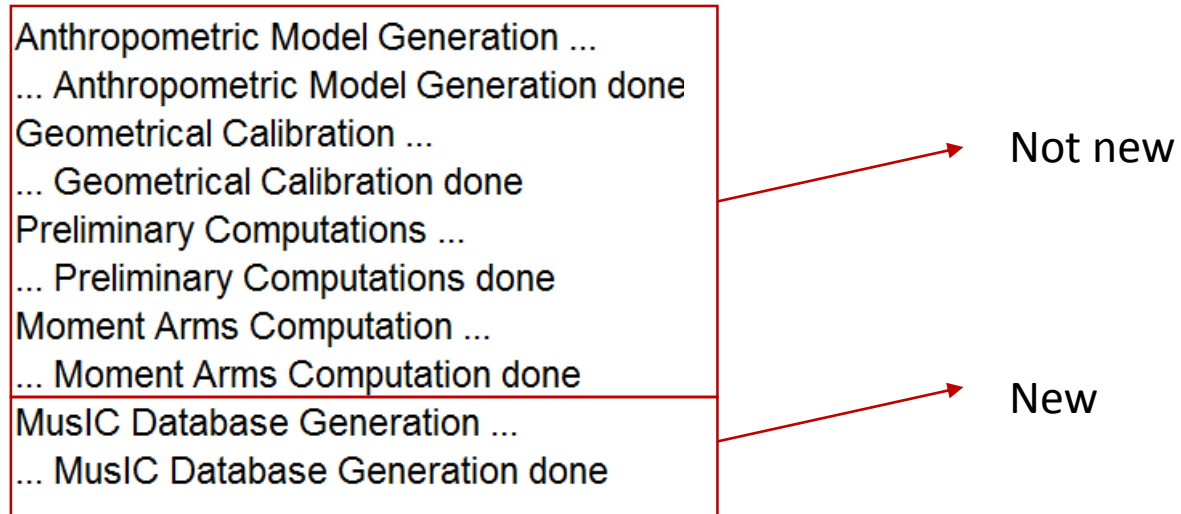
What CusToM is doing ?

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

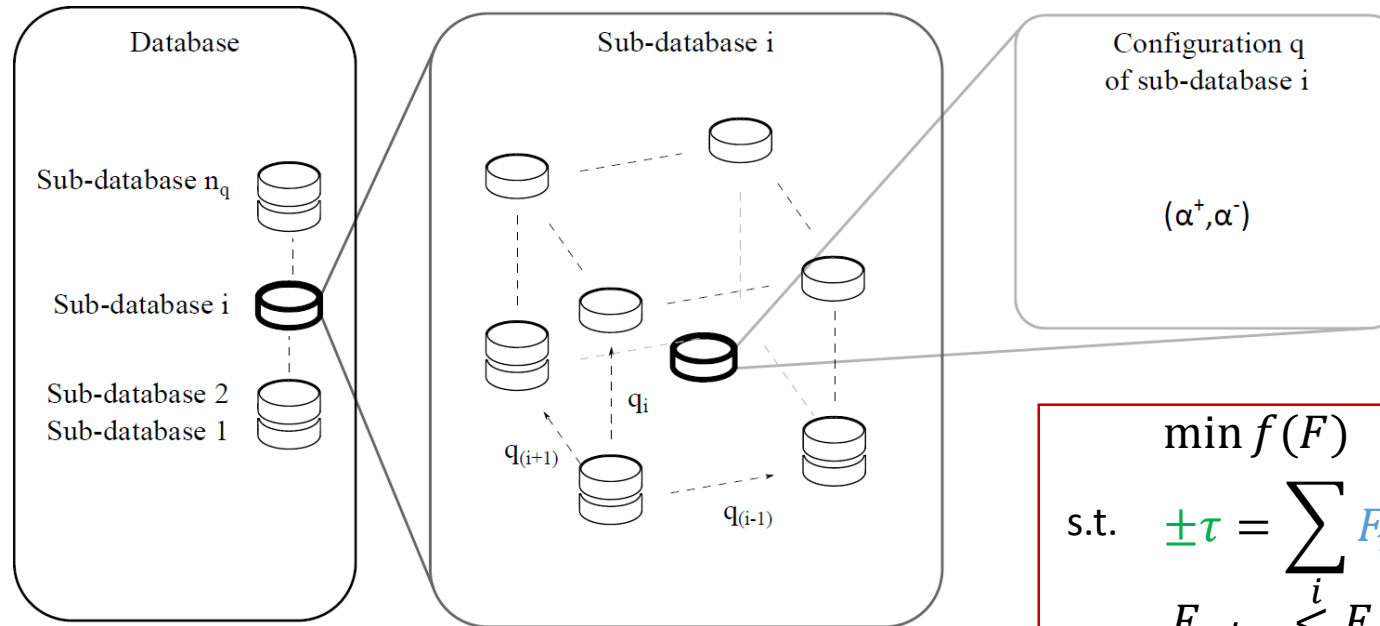


Not new

What CusToM is doing ?



Database generation

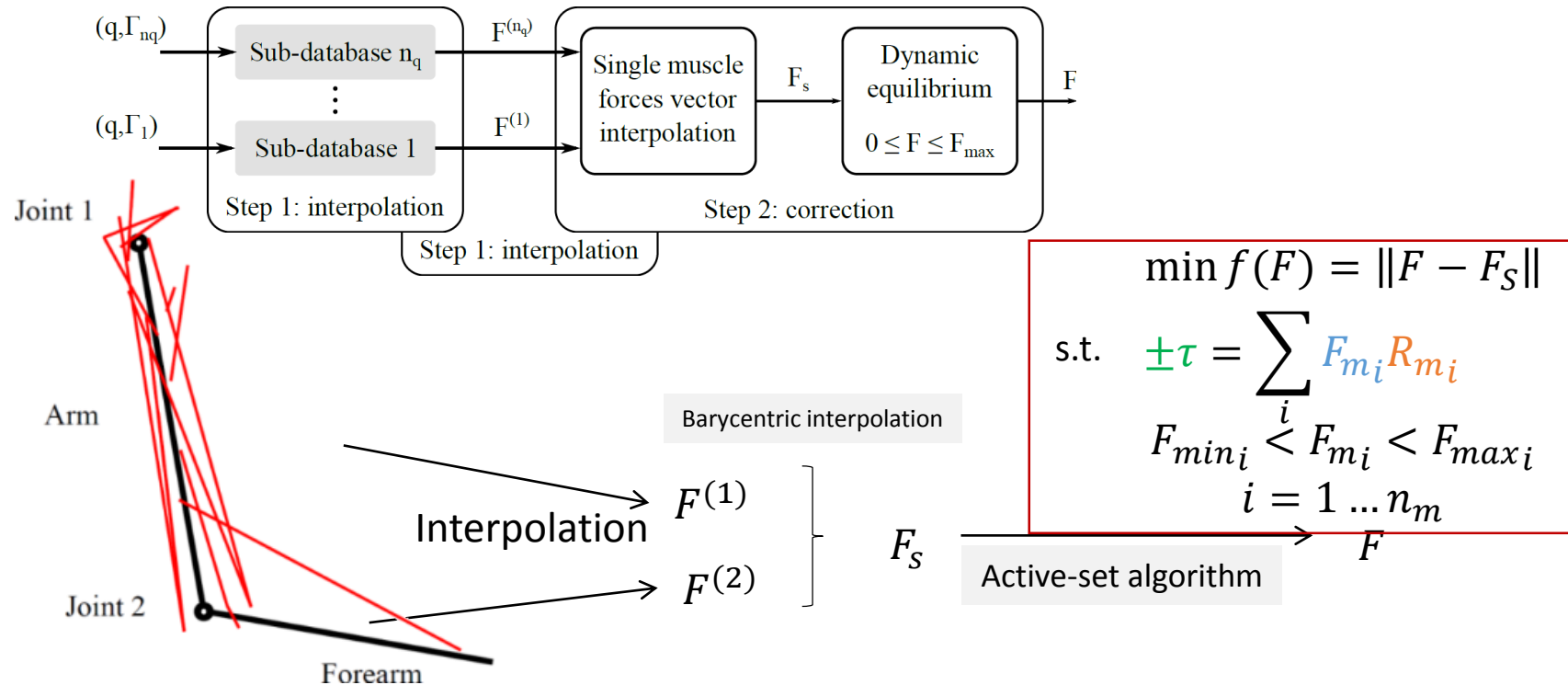


$$\begin{aligned} \min f(F) \\ \text{s.t. } \pm \tau &= \sum_i F_{m_i} R_{m_i} \\ F_{\min_i} &< F_{m_i} < F_{\max_i} \\ i &= 1 \dots n_m \end{aligned}$$

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

MuslC method

Compute forces from torques and joint configuration



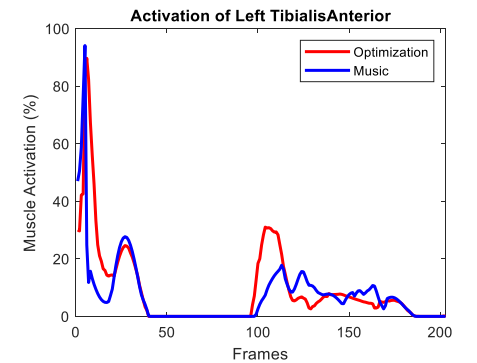
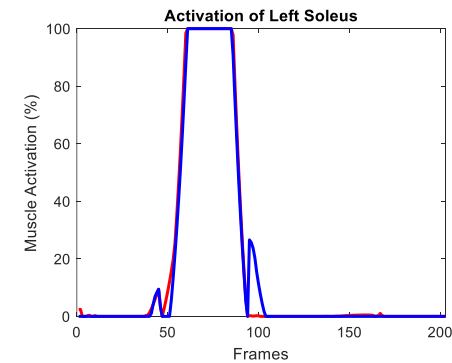
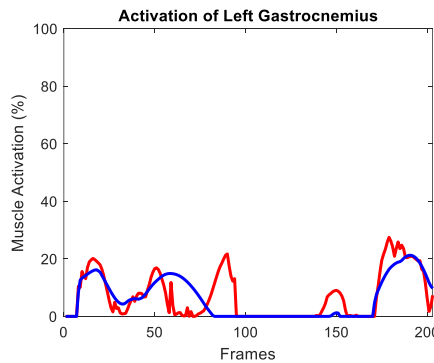
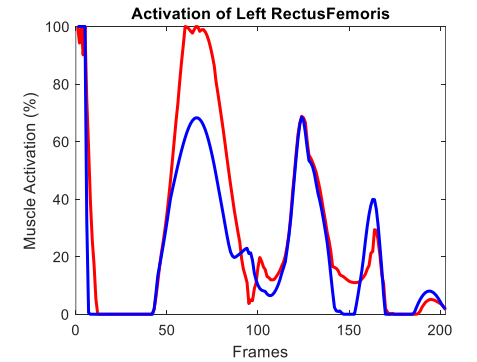
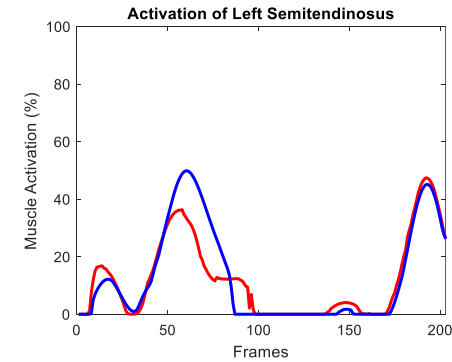
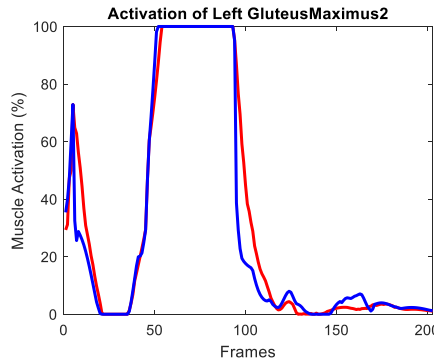
Results

>> 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 MuslC 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**