Documentation for CusToM: Customizable Toolbox for Musculoskeletal simulation

Antoine Muller, Charles Pontonnier, Pierre Puchaud, Georges Dumont

Contents

1 Introduction

Customizable Toolbox for Musculoskeletal simulation (CusToM) is a MATLAB® toolbox aimed at performing inverse dynamics based musculoskeletal analyzes. CusToM exhibits several features. It can generate a personalized musculoskeletal model, and can solve from motion capture data inverse kinematics, external forces estimation, inverse dynamics and muscle forces estimation problems.

According to user choices, the musculoskeletal model generation is achieved thanks to libraries containing pre-registered models. These models consist of body parts osteoarticular models, set of markers or set of muscles to be combined together. From an anthropometric based model, the geometric, inertial and muscular parameters are calibrated to fit the size and mass of the subject to be analyzed. Then, from motion capture data, the inverse kinematics step computes joint coordinates trajectories against time. Then, joint torques are computed thanks to an inverse dynamics step. To this end, external forces applied to the subject have to be known. They may be directly extracted from experimental data – as platform forces – or be estimated from motion data by using the equations of motion in an optimization scheme. Last, muscle forces are estimated. It consists in finding a repartition of muscle forces respecting the joint torques and representing the central nervous system strategy.

For a large set of musculoskeletal models and motion data, CusToM can easily performs all of the analyzes described above. CusToM has been created as a modular tool to let the user being as free and autonomous as possible. The osteoarticular models, set of markers and set of muscles are defined as bricks customizable and adaptable with each other. The design or the modification of a musculoskeletal model is simplified thanks to this modularity. Following the same idea, some methods are defined as adaptable bricks. Testing new cost functions in the optimization schemes, changing performance criteria or creating alternative motion analysis methods can be done in a relatively easy way.

A user interface has been developed to facilitate the data management and the model definition during a given study.

2 Installation instructions

CusToM was implemented and tested with the MATLAB® 2018a version and requires the Symbolic Math Toolbox™, the Optimization Toolbox™ and the Parallel Computing Toolbox™ to be fully efficient.

After downloading the main folder named `CusToM` and placing it in a relevant location on the computer, the installation only consists in adding the folder `...\CusToM\Functions` and its subfolders to the MATLAB path.

3 Application programming interface

To perform a new study, the user has to create a new folder containing all the experimental data to be analyzed. This folder has to be the current folder of MATLAB during the study.

3.1 Musculoskeletal model generation

The first step consists in defining the musculoskeletal model parameters. To this end, the first user interface (Figure 1-4) is opened by executing `GenerateParameters`. It is composed of two parts. The left one allows the user to choose all the musculoskeletal model parameters on different tabs (1.1). The right one allows the visualization of the model resulting from the parameters tuning. This interface generates a parameters file named `ModelParameters` by clicking on the **Generate parameters** button (1.3).

The first tab (Figure 1) contains the general parameters. The size and the mass of the subject are defined here. It also contains the button **Load parameters** (1.4) allowing to load a `ModelParameters` file already computed for another study.

The second tab (Figure 2) allows the user to define the osteoarticular model. The whole body is divided into 5 parts: trunk (2.1), right leg (2.2), left leg (2.3), right arm (2.4) and left arm (2.5). The

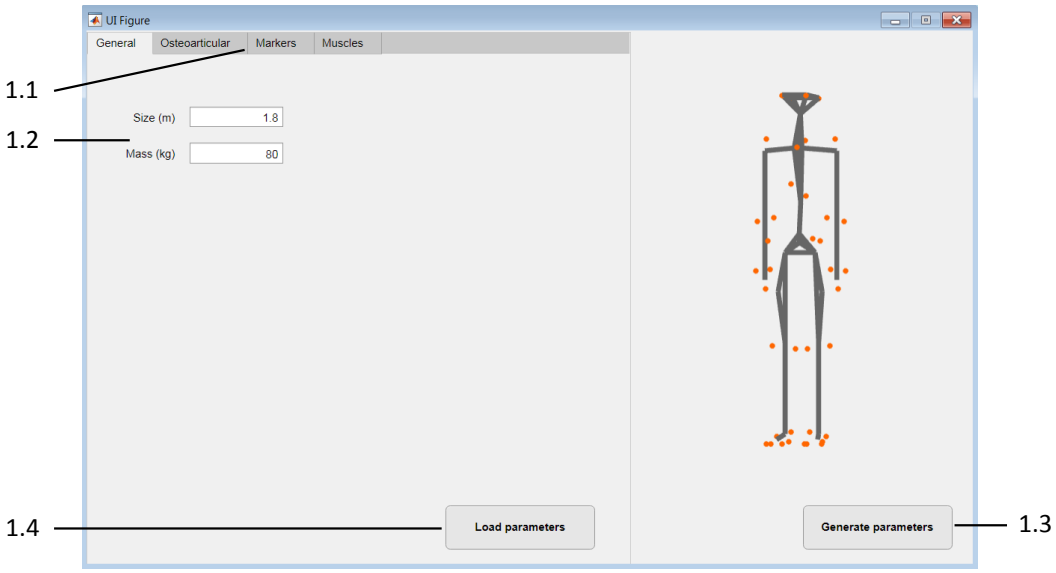

Figure 1: First tab of the application programming interface for the definition of the musculoskeletal model parameters: general parameters

trunk is itself divided into the pelvis and lower trunk, the upper trunk and the head parts. For each of these parts, the body part model is chosen in a list of pre-defined models. A field **NoModel** can be used to define a partial model or a particular morphology. A user interested in defining new models can do it by defining a new model file in the corresponding folder and following the syntax of the pre-defined ones. The created model will then appear in the interface as a new choice for a given body part.

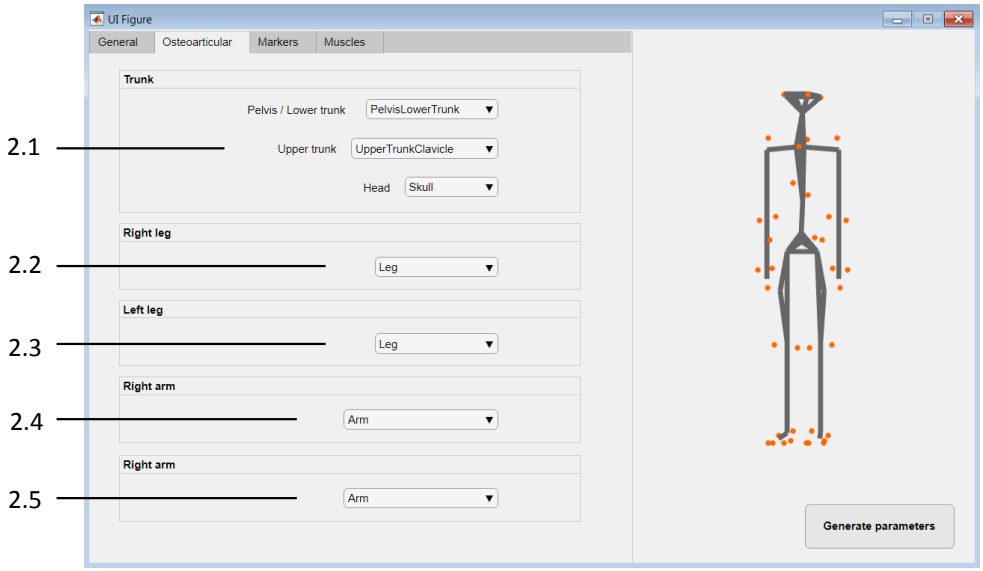

Figure 2: Second tab of the application programming interface for the definition of the musculoskeletal model parameters: body parts models.

The third tab (Figure 3) contains parameters about the markers set. The choice of the markers set into a pre-defined list is done by using a drop down menu (3.1). According to the chosen set, additional options may be required (3.2). This could be a variable parameters on a given markers set, for example, the number of markers used on hands. Each marker is thus represented by a button (3.3). Each of them allows to remove or to add the considered marker from the set. A user interested in defining new markers sets can do it by defining a new markers set file in the corresponding folder and following the syntax of the pre-defined ones. The created markers set will then appear in the interface as a new choice in the drop down menu.

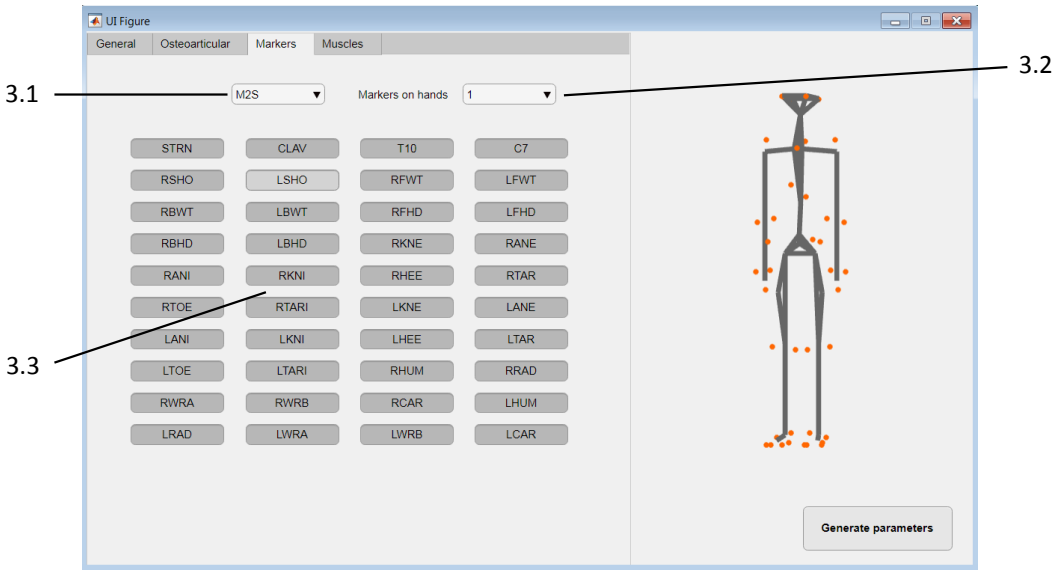

Figure 3: Third tab of the application programming interface for the definition of the musculoskeletal model parameters: marker set.

The fourth tab (Figure 4) allows to add muscles on the model. The toolbox contains several pre-defined sets of muscles. The addition or the deletion of a muscle set is done by using the **Delete** (4.3) and **Add** (4.4) buttons. For each added muscles set (4.1), additional informations (4.2) as the desired side of the muscles may be required. A user interested in defining new muscles sets can do it by defining a new muscles set file in the corresponding folder and following the syntax of the pre-defined ones. The created muscles set will then appear in the interface as a new choice in the drop down menu.

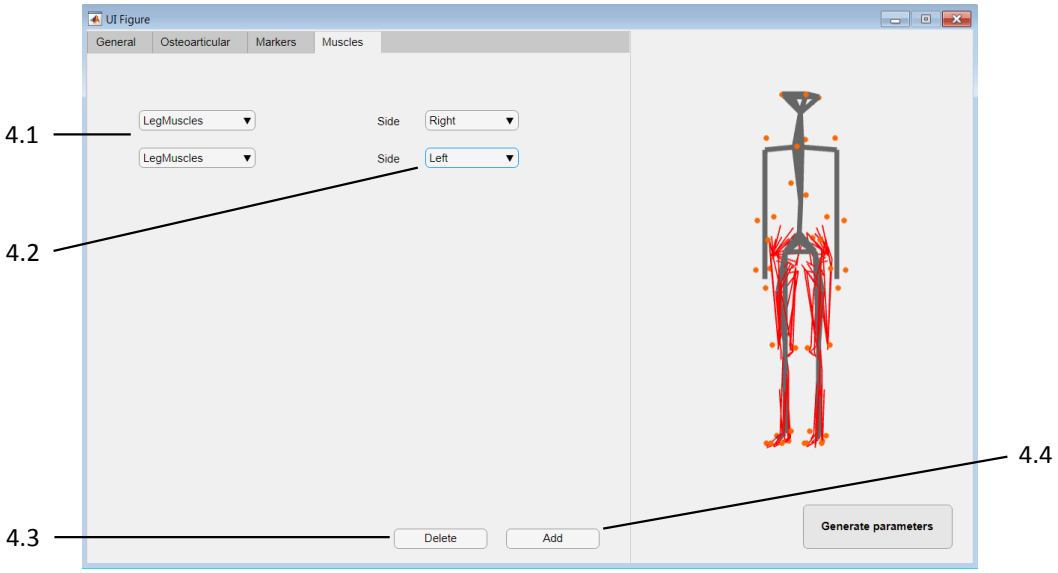

Figure 4: Fourth tab of the application programming interface for the definition of the biomechanical model parameters: muscles set.

The file `ModelParameters` contains all the musculoskeletal model parameters and is automatically generated by the interface shown above. It is a structure which contains all these fields:

- *Size*: subject size;
- *Mass*: subject mass;

- *PelvisLowerTrunk*: function handle calling the model for the pelvis and lower trunk;
- *UpperTrunk*: function handle calling the model for the upper trunk model;
- *Head*: function handle calling the model for the head;
- *RightLeg*: function handle calling the model for the right leg;
- *LeftLeg*: function handle calling the model for the left leg;
- *RightArm*: function handle calling the model for the right arm;
- *LeftArm*: function handle calling the model for the left arm;
- *Markers*: function handle calling the markers set;
- *MarkersOptions*: additional options associated to the chosen markers set;
- *MarkersRemoved*: cell array containing the names of the markers to be ignored in the analysis compared to the ones contained in the chosen markers set;
- *Muscles*: cell array containing functions handle calling the muscle models;
- *MusclesOptions*: additional options associated to the chosen muscles set. The size and the structure of *MusclesOptions* is the same as the field *Muscles*.

3.2 Motion analysis

The second user interface (Figure 5) consists in defining all the analysis parameters. It can be opened by executing `Analysis`. This interface generates a parameters file name `AnalysisParameters` by clicking on the **Run** button (5.7). All the motion analysis computations are thus automatically launched. The button `Load parameters` (5.2) allows to load a `AnalysisParameters` file already computed for another study. The interface represents the different steps available for the motion analysis. It contains 3 calibration steps: the geometrical calibration, the inertial calibration and the muscular calibration, and four analysis steps: the inverse kinematics, the external forces estimation, the inverse dynamics and the muscle forces estimation.

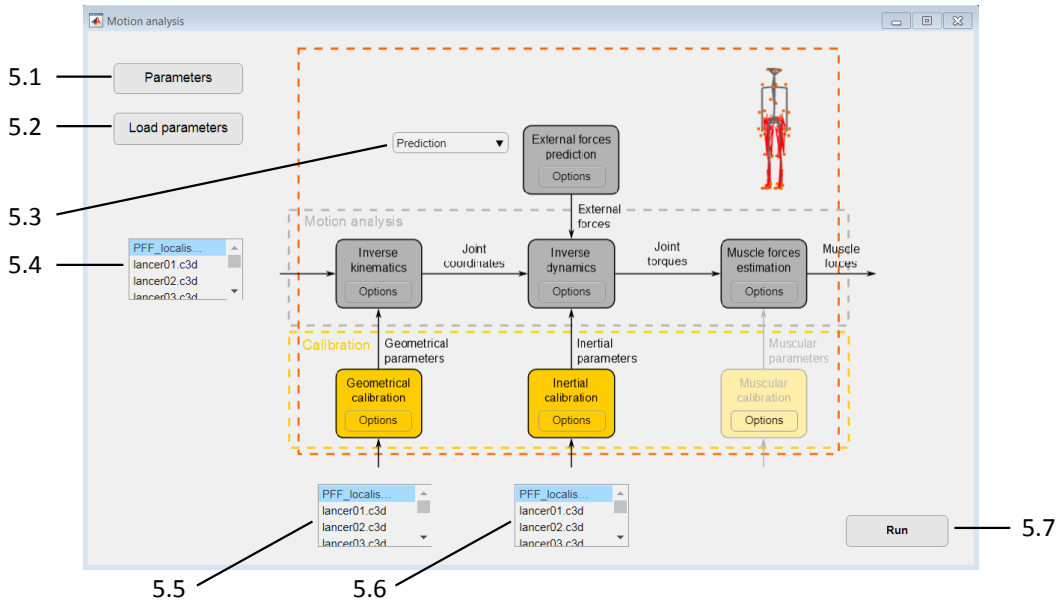

Figure 5: Application programming interface for the motion analysis parameters.

The list box (5.4) allows to select files that have to be studied in the analysis steps. It contains the name of all *c3d* files that are in the current folder. The field *filename* on the `AnalysisParameters` file

thus contains a cell array with all the selected names.

The list boxes (5.5) and (5.6) allow to select the files used respectively for the geometrical calibration and for the inertial calibration. Only one file could be selected for each list box.

General parameters of the motion analysis are available by clicking on the **Parameters** button (5.1) (Figure 6). If the check box (6.1) is active, the markers position extracted from the *c3d* file will be filtered (4-th order Butterworth low pass filter with no phase shift).The cut-off frequency of the filter can be modified (6.2). The drop down (6.3) can be used to choose a specific type of input data. Currently, only C3D files are supported. A user interested in using original input data (Kinect data, ...) can do it by defining a input handling function in the corresponding folder and following the syntax of the predefined ones. The created function and data type will then appear in the interface as a new choice in the drop down menu.

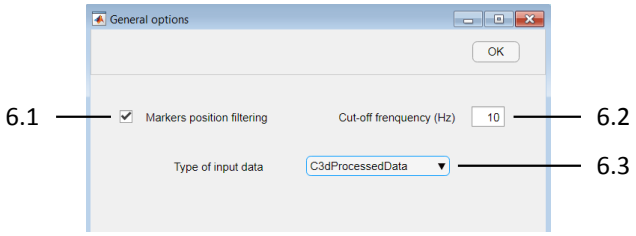

Figure 6: Application programming interface for the general parameters of the motion analysis.

In the `AnalysisParameters` file, the general parameters are stored in the field *General* which itself contains these fields:

- *FilterActive*: logical value to select if the markers position have to be filtered or not;
- *FilterCutOff*: numerical value of the filter cut-off frequency;
- *InputData*: function handle calling the function to extract and process c3d data.

Moreover, several additional options for each step of the motion analysis are available. Each of them is accessible by clicking the **Options** button associated to the step. For each step, the different options are detailed in the following sections.

Geometrical calibration

In the user interface (Figure 7-10), a check box (7.1) allows to activate or not the geometrical calibration step. If this step is activated, several options are available on four tabs.

The first tab (Figure 7) contains parameters about the frames used. The function used to select the frames throughout the motion is chosen thanks to the drop down menu (7.2). The number of frames to select is defined in the box (7.3).

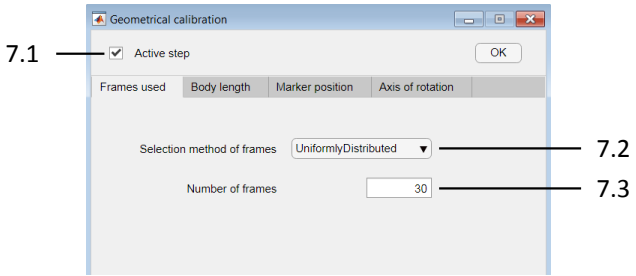

Figure 7: First tab of the application programming interface for the definition of the geometrical calibration parameters.

The second tab (Figure 8) contains parameters about the calibration of body part lengths. It allows to link two segments lengths, applying the same homothety coefficient for these two segments as a constraint in the optimization scheme. Thanks to the **Delete** (8.3) and **Add** (8.4) buttons, users can delete and add constraints. For each of them, the two solids are selected by using the drop downs menus (8.1) and (8.2).

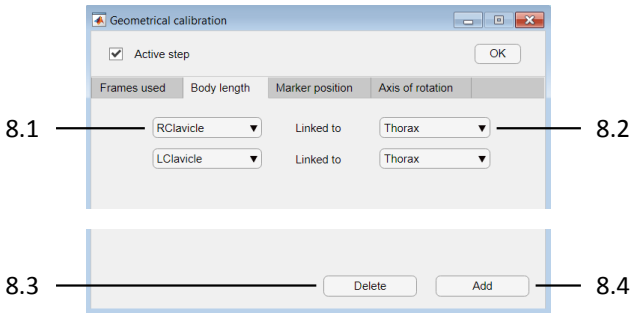

Figure 8: Second tab of the application programming interface for the definition of the geometrical calibration parameters.

The third tab (Figure 9) contains parameters about the calibration of the local positions of the model markers. The calibration of the local position of a marker could be blocked in one or more directions. So, for each marker, three check boxes (9.1) allow to fix or liberate the calibration of its local position along the x-axis, y-axis or z-axis.

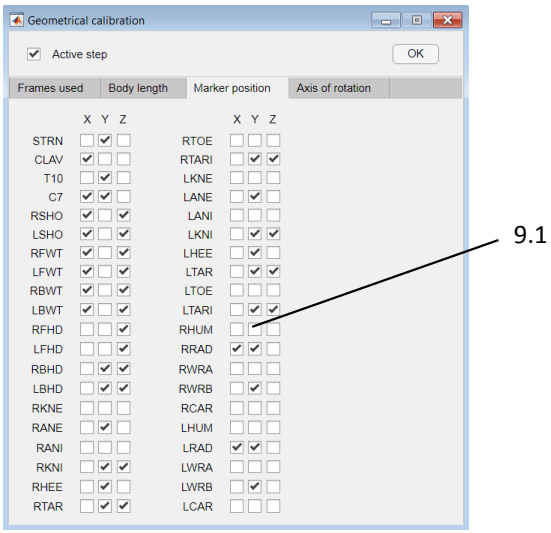

Figure 9: Third tab of the application programming interface for the definition of the geometrical calibration parameters.

The fourth tab (Figure 10) contains parameters about the calibration of the axis of rotation orientation. Thanks to the **Delete** (10.4) and **Add** (10.5) buttons, the user can delete and add constraints. For each of them, the axis is selected thanks to its associated solid by using the drop down menu (10.1). The orientation of this axis is indicated in (10.3). The three check boxes (10.2) allow to modify the orientation of the axis along the x-axis, the y-axis or the z-axis.

In the `AnalysisParameters` file, the geometrical calibration parameters are stored in the field *CalibIK* which itself contains all of these fields:

- *Active*: logical value to activate or not the geometrical calibration step;
- *filename*: characters string containing the name of the file used;

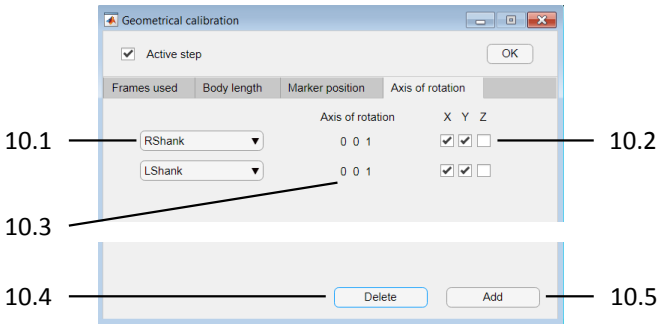

Figure 10: Fourth tab of the application programming interface for the definition of the geometrical calibration parameters.

- *Frames.Method*: function handle calling the function to select the frames throughout the motion;
- *Frames.NbFrames*: number of frames to be used in the geometrical calibration;
- *LengthAdd*: cell array containing the names of the solids involved in the additional constraints – compared to constraints initially defined in the model;
- *LengthDelete*: cell array containing the name of the solids of the deleted constraints – compared to constraints initially defined in the model;
- *MarkersCalibModif*: cell array containing the local calibrated directions of markers where modifications were applied compared to informations initially defined in the model;
- *AxisAdd*: cell array containing the orientation axes of the calibrated axes of rotation added to the ones being already defined in the model;
- *AxisDelete*: cell array containing the orientation of the calibrated axis of rotation deleted in the model.

Inertial calibration

In the user interface (Figure 11), a check box (11.1) allows to activate or not the inertial calibration step. If this step is activated, several options are available.

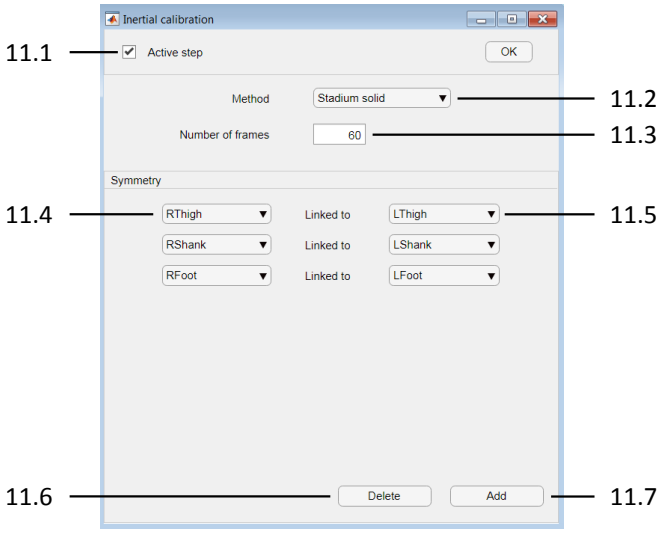

Figure 11: Application programming interface for the definition of the inertial calibration parameters.

The inertial calibration method is selected thanks to the drop down menu (11.2). The number of frames to select is defined in the editable field (11.3). Some options could be added to link the inertial parameters of two symmetric solids through the definition of additional constraints. Thanks to the

Delete (11.6) and **Add** (11.7) buttons, users can delete and add these constraints. For each of them, the two solids are selected by using the drop downs menus (11.4) and (11.5).

In the `AnalysisParameters` file, the inertial calibration parameters are stored in the field *CalibID* which itself contains all of these fields:

- *Active*: logical value to activate or not the inertial calibration step;
- *filename*: characters string containing the name of the file used;
- *Frames.NbFrames*: number of frames used;
- *Method*: function handle calling the inertial calibration method;
- *Symmetry*: array containing the position of solids in the osteoarticular model where a constraint of symmetry has been added.

Muscular calibration

At this moment, no validated muscular calibration is available. This step is thus deactivated. Muscular parameters of the models available in the current version are based on anthropometric data.

Inverse kinematics

In the user interface (Figure 12), a check box (12.1) allows to activate or not the inverse kinematics step. If this step is activated, several options are available.

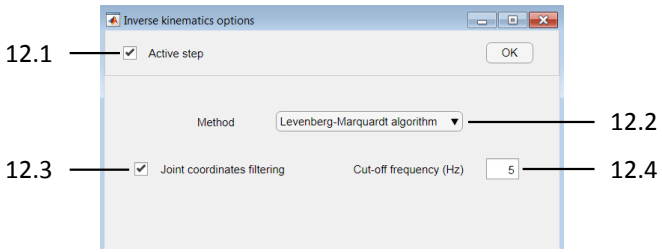

Figure 12: Application programming interface for the definition of the inverse kinematics parameters.

The inverse kinematics method is selected thanks to the drop down menu (12.2). The check box (12.3) allows to choose if the joint coordinates coming from this step will be filtered (4-th order Butterworth low pass filter with no phase shift) or not. If this box is selected, the cut-off frequency can be modified (12.4).

In the `AnalysisParameters` file, the inverse kinematics parameters are stored in the field *IK* which itself contains all of these fields:

- *Active*: logical value to activate or not the inverse kinematics step;
- *Method*: numerical value calling the inverse kinematics function (1 for the optimization method; 2 for the Levenberg-Marquardt algorithm);
- *FilterActive*: logical value to select if the joint coordinates will be filtered or not;
- *FilterCutOff*: numerical value of the filter cut-off frequency.

External forces computation

The external forces can be computed by three different methods (chosen thanks to (5.3)): either there is no external force, either external forces are based on experimental data or external forces are predicted from the motion. The field *ID.InputData* of the `AnalysisParameters` file contains this information (numerical value 0 for no external external force, 1 for the experimental data and 2 for the prediction).

If there is no external force, no additional option is available.

If external forces are based on experimental data, additional options are defined in the user interface (Figure 13). The check box (13.1) allows to choose if the external forces data will be filtered (4-th order Butterworth low pass filter with non phase shift) pr not. If this box is selected, the cut-off frequency can be modified (13.2). The drop down menu (13.3) allows to choose which function is used to extract and process c3d data. For each platform detected on the c3d, the solid with which it is in contact is selected thanks to the drop downs menus (13.4). A field **NoContact** can be used to ignore platforms data.

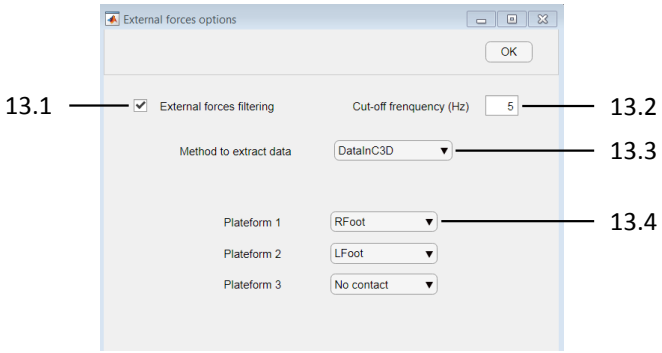

Figure 13: Application programming interface for the definition of the external forces computation parameters when they are based on experimental data.

In the `AnalysisParameters` file, the external forces computation parameters when they are based on experimental data are stored in the field *ExternalForces* which itself contains of these fields:

- *FilterActive*: logical value to select if the joint coordinates are filtered or not;
- *FilterCutOff*: filter cut-off frequency;
- *Method*: function handle calling the function to extract and process c3d data;
- *Options*: cell array containing the names of solids which are in contact with the different platforms.

If external forces are computed thanks to a prediction method, additional options are defined on the user interface (Figure 14). The check box (14.1) allows to choose if the external forces data will be filtered (4-th order Butterworth low pass filter with non phase shift) or not. If this box is selected, the cut-off frequency can be modified (14.2). The position and velocity thresholds and the friction coefficient are respectively defined by using the editable fields (14.3) and (14.4). Each contact point is chosen by using anatomical points available in the model. The drop down menu (14.5) allows to select a solid and the list boxes (14.6) allows to select a set of points into the associated solid. The deletion or the addition of a solid is made thanks to **Delete** (14.7) and **Add** (14.8) buttons.

In the `AnalysisParameters` file, the external forces computation parameters when they are based on a prediction method are stored in the field *Prediction* which itself contains all of these fields:

- *FilterActive*: logical value to select if the joint coordinates are filtered or not;
- *FilterCutOff*: filter cut-off frequency;
- *PositionThreshold*: position threshold;
- *VelocityThreshold*: velocity threshold;
- *FrictionCoef*: friction coefficient;
- *ContactPoint*: cell array containing the names of the anatomical points defined as the contact points.

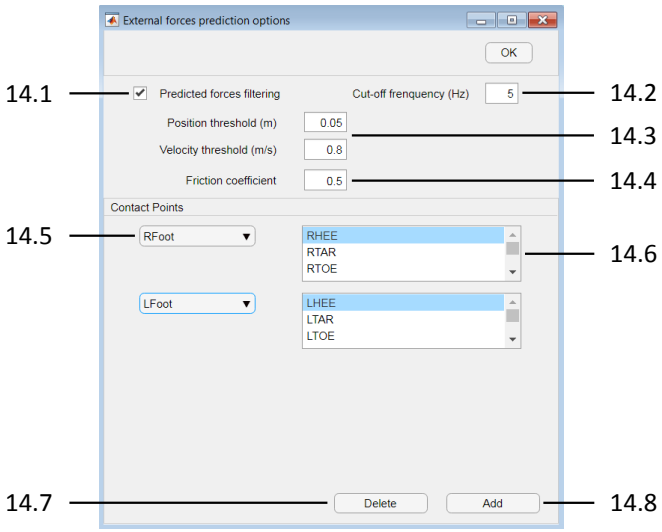

Figure 14: Application programming interface for the definition of the external forces computation parameters when there are based on a prediction method.

Inverse Dynamics

In the user interface (Figure 15), a check box (15.1) allows to activate or not the inverse dynamics step. No additional options are available for this step.

Figure 15: Application programming interface for the definition of the inverse dynamics parameters.

The field *ID.Active* of the `AnalysisParameters` file contains the logicial value to activate or not the inverse dynamics step.

Muscle forces estimation

In the user interface (Figure 16), a check box (16.1) allows to activate or not the muscle forces estimation step. If this step is activated, several options are available.

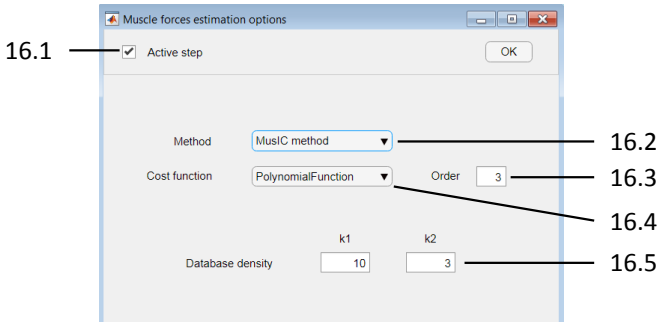

Figure 16: Application programming interface for the definition of the muscle forces estimation parameters.

The muscle forces estimation method is selected thanks to the drop down menu (16.2). If the MusIC method is selected, the densities of the database can be modified (16.5). The cost function used to solve

the force sharing problem is chosen by using the drop down menu (16.4). According to the cost function used, additional options (16.3) – as the order of the polynomial function – could be required. A user interested in using original cost functions can do it by defining a new cost function in the corresponding folder and following the syntax of the predefined ones. The created function and data type will then appear in the interface as a new choice in the drop down menu.

In the `AnalysisParameters` file, the muscle forces estimation parameters are stored in the field *Muscles* which itself contains all these fields:

- *Active*: logical value to activate or not the muscle forces estimation step;
- *Method*: muscle forces estimation method (1 for the use of an optimization method; 2 for the use of the MusIC method);
- *DatabaseDensity*: array containing the densities of the database used for the MusIC method;
- *Costfunction*: function handle calling the cost function used to solve the force sharing problem;
- *CostfunctionOptions*: numerical value containing possibly options of the cost function used.

4 Results

The results of the different steps are automatically saved. The following sections detail the structure of the results. All of the numerical data is stored in SI units.

4.1 BiomechanicalModel

`BiomechanicalModel` is directly saved in the current folder. It contains all the parameters of the musculoskeletal model, independently from the data to analyze. This structure is composed of 7 fields.

OsteoArticularModel is a structure where each data represents a solid associated to a joint and contains all of these fields:

- *name*: characters string containing the solid name;
- *sister / child / mother*: numerical values representing the model structure tree – each value corresponds to the number of the solid in the structure.
- $\boldsymbol{a}$: (3x1) array containing the joint axis – defined in the mothers reference system;
- *joint*: numerical value representing the type of joint (1 for a revolute joint; 2 for a prismatic joint);
- *limit_inf / limit_sup*: numerical values containing the joint limits;
- $\boldsymbol{u}$ / *theta*: (3x1) array and numerical value representing a fixed rotation after the joint of *theta* angle according to u axis;
- $\boldsymbol{b}$: (3x1) array containing the joint position – defined in the mothers reference system;
- $\boldsymbol{c}$: (3x1) array containing the position of the center of mass – defined in the current solid reference system;
- m: numerical value containing the solid mass;
- $\boldsymbol{I}$: (3x3) array containing the solid inertia matrix – defined in the current solid reference system;
- *anat_position*: cell array containing the set of anatomical positions, each defined by its name (characters string) and its local position on regards to the center of mass ((3x1) array);
- *Visual*: logical value to identify the real solids;
- *calib_k_constraint*: numerical value representing the number of the solid which had the same homothety coefficient for the geometrical calibration, empty array otherwise;
- L: cell array containing the names of two anatomical positions which represent the centers of the stadium solid used for the inertial calibration;

- *KinematicsCut*: numerical value containing the number of the geometrical cut achieved on this solid, empty array otherwise;
- *ClosedLoop*: cell array containg the name of an anatomical position (characters string) and its position for the closed loop – defined in the current solid reference system –, empty array otherwise;
- *linear_constraint*: (2x1) array containing the number of a solid and the linear coefficient which link the joint coordinates of the two solids, empty array otherwise;
- *limit_alpha*: array containing the limits of axis orientation variation during the geometrical calibration, empty array if it is not possible to calibrate it;
- *v*: array representing axis which have to be calibrated during the geometrical calibration.

Markers is a structure where each data represents a marker and contains all of these fields:

- *name*: characters string containing the marker name;
- *anat_position*: characters string containing an anatomical landmark name where the marker is located;
- *calib_dir*: cell array indicating the local direction in which the marker position has to be calibrated during the geometrical calibration – 'On' for a calibrated direction, 'Off' otherwise.
- *exist*: logical value indicating if the marker is defined on the osteoarticular model;
- *num_solid*: numerical value indicating the number of the solid containing the marker;
- *num_markers*: numerical value indicating the number of the anatomical position on its associated solid where the marker is located.

Muscles is a structure where each data represents a muscle and contains all of these fields:

- *name*: characters string containing the muscle name;
- *f0 / l0 / Kt / ls / alpha0*: numerical values containing the muscular properties (maximum isometric force / optimal fiber length / tendon stress-strain constant / tendon slack length / pennation angle at muscle optimal fiber length);
- *path*: cell array containing the anatomical names (characters string) of the muscle path;
- *exist*: logical value indicating if the muscle is defined on the osteoarticular model;
- *num_solid*: array indicating the numbers of the solids containing the muscle;
- *num_markers*: array indicating the numbers of the anatomical positions on their associated solids where the muscle is located.

MomentArms is a cell array containing the moment arms of each muscle for each joint. If the value is not zero, the moment arm is expressed as an anonymous function according to the joint coordinates.

MuscularCoupling is an array containing the muscular coupling matrix.

MusICDatabase is a structure containing the database used for the MusIC method.

GeometricalCalibration is a structure containing the different results of the geometrical calibration. It contains all of these fields:

- *frame_calib*: array containing the number of frames used for the calibration;
- *crit*: array containing the stop criteria value for each iteration;
- *errorm*: cell array containing, for each iteration, the differences between the experimental markers positions and the reconstructed model markers positions;
- *k_calib*: array containing the homothety coefficients;
- *p_calib*: array containing the local variation of each model marker;
- *alpha_calib*: array containing the axis orientation variation.

4.2 ExperimentalData

One `ExperimentalData` is saved per studied motion in the associated folder. This file is generated during the inverse kinematics step. It contains all variables of experimental motion data and is organized into these fields:

- *FirstFrame*: numerical value of the number of the first frame;
- *LastFrame*: numerical value of the number of the last frame;
- *MarkerPositions*: structure containing, for each marker (identified thanks to their names), their experimental positions;
- *Time*: array containing the time vector.

4.3 InverseKinematicsResults

One `InverseKinematicsResults` is saved per studied motion on the associated folder. This file is generated during the inverse kinematics step and contains all the results of this step. It is organized into these fields:

- *JointCoordinates*: array containing the joint coordinates at each time;
- *FreeJointCoordinates*: array containing the joint coordinates of the 6-dof joint at each time;
- *ReconstructionError*: array containing the differences between the experimental markers positions and the reconstructed model markers positions at each time.

4.4 ExternalForcesComputation

One `ExternalForcesComputation` is saved per studied motion in the associated folder. This file is generated during the external forces computation step and contains all the results of this step. It could contain three different fields according to the chosen step: *NoExternalForce*, *Experiments* or *ExternalForcesPrediction*. Each of these fields contains the same structure where each data represents external forces at one time. Considering one of them, for each solid, a (3x2) array (field *fext*) represents the forces and torques applied on this solid – expressed on the global frame. The associated field *Visual* contains the same information adapted for the visualizaton of the results.

4.5 InverseDynamicsResults

One `InverseDynamicsResults` is saved per studied motion on the associated folder. This file is generated during the inverse dynamics step and contains all the results of this step. It is organized into these fields:

- *JointTorques*: array containing the joint torques at each time;
- *DynamicResiduals*: array containing the dynamic residuals at each time, that is the joint torques of the 6-dof joint.

4.6 MuscleForcesEstimationResults

One `MuscleForcesEstimationResults` is saved per studied motion on the associated folder. This file is generated during the muscle forces estimation step and contains all the results of this step. It is organized into these fields:

- *MuscleForces*: array containing the muscle forces at each time;
- *MuscleActivations*: array containing the muscle activations at each time.

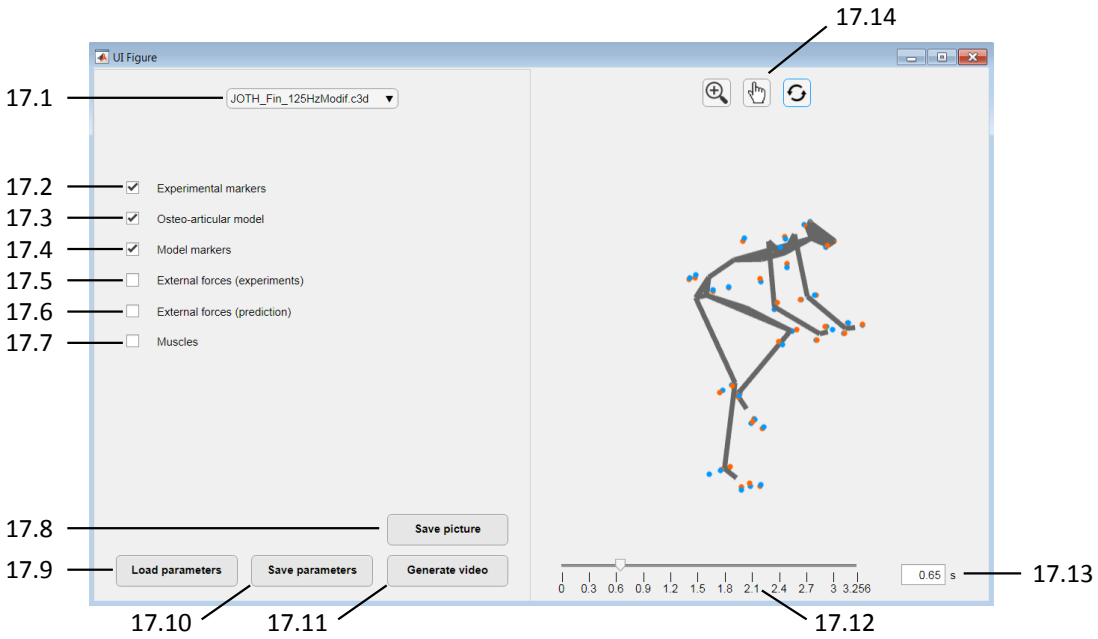

Figure 17: Application programming interface for the animation.

4.7 Visualization

By executing `GenerateAnimate`, the user interface (Figure 17) allows to visualize different results.

The motion is selected by using the drop down menu (17.1). All elements to visualize could be selected thanks to the check box (17.2-7). Navigation buttons (17.14) allow to adjust the view. The time is set thanks to the slider (17.12) or thanks to the editable field (17.13). A copy of the picture on a *png* format could be done thanks to the button (17.8). A video of all the motion on a *avi* format could be done thanks to the button (17.11). The button (17.10) allows to save the different parameters of the animation (view, elements to visualize) into the `AnimateParameters` file. The button (17.9) allows to load the parameters of a previous study.

5 Data processing examples

Two examples extracted from research works are available in the current release. The first one consists in estimating the lower limbs muscle forces during a cycling motion. The second one consists in predicting the ground reaction forces on a range of motion.

5.1 Lower limbs muscle forces estimation on a cycling motion

The first example is extracted from [2]. This research work consisted in linking the symmetry, the performance and the health during a cycling motion. The muscular symmetry is analyzed by the motion analysis containing an inverse kinematics step, an inverse dynamics step and a muscle forces estimation step. The markers set used is that called `M2S` in CusToM. External forces applied on each foot were measured with two mobile platforms located on pedals. The subject size is 184 cm and the subject weight is 70 kg. The processing and results of this example can be seen and modified by making the example folder as current in MATLAB, and launch the `GenerateParameters` and `Analysis` functions.

5.2 Ground reaction forces prediction on a range of motion

The second example is extracted from [1]. This research aims at testing a ground reaction forces prediction method based on motion data. The subject performed a range of motion, activating all of his degrees of freedom sequentially during the motion. The external forces computation is done by using **Prediction** on (5.3). 14 contact points under each foot are used – defined on **Options** (Figure 14). The subject size is 180 cm and the subject weight us 77.5 kg. The processing and results of this example can be seen and modified by making the example folder as current in MATLAB, and launch the `GenerateParameters` and `Analysis` functions.

References

[1] A. Muller. *Contributions méthodologiques à l'analyse musculo-squelettique de l'humain dans l'objectif d'un compromis précision performance.* PhD thesis, École normale supérieure de Rennes, 2017.

[2] C. Pouliquen. *Evaluation de l'asymétrie articulaire et musculaire au cours d'exercices exhaustif en cyclisme: apports de l'approche expérimentale et de la modélisation musculosquelettique.* PhD thesis, Université Rennes 2, 2017.