CEINMS User Guide Documentation

Release 0.9.0

Elena Ceseracciu, Monica Reggiani

May 27, 2015

Contents

Part I

Introduction

CHAPTER

ONE

CEINMS

1.1 Overview

CEINMS (Calibrated EMG-Informed Neuromusculoskeletal modelling toolbox) is the result of an interdisciplinary collaboration among the biomechanics and the computer science worlds. Inspired by early electromyography-driven (EMG-driven) methods [McG92] used in the biomechanical community, Lloyd and colleagues developed algorithms and software to calibrate EMG-driven, or now called EMG-informed, neuromusculoskeletal models to match each individual's characteristics [BLMB04][GSB+13][LB03][LB96][LB01][LBWB08][SFL13][SRFL12][WGKL13][WLBK09]

Fundamental to these calibrated methods was the ability to validate the outputs against other data not used for calibration. All these algorithms and software were collected and integrated together to create CEINMS.

CEINMS was designed and written to be flexible and generic software, that is, given the appropriate anatomical and physiological data, it can operate with any number of musculotendon units (MTU) and any number of degree of freedoms (DOF). Moreover, the modular design allows the independent selection of different operation modes:

1. *Full-predictive open-loop mode.* The experimentally recorded EMG signals and 3D joint angles are used as input to a neuromusculoskeletal model to directly drive the computations of the musculotendon forces [GSB+13][LB03][LB96][LB01][LBWB08][SRFL12][WGKL13][WLBK09] (Fig. 1.1).
2. *Hybrid mode.* The excitation patterns of muscles from which it is not practical or possible to routinely collect EMG signals (e.g. deep muscles) are constructed using optimization algorithms. Then, the constructed excitations, experimental EMGs, and 3D joint angles are used as input for the neuromusculoskeletal model [SFL13].
3. *EMG-assisted mode.* This mode is a more generalizable form of the Hybrid mode. It uses optimization to adjust both the excitations determined from experimentally recorded EMG signals and to determine the excitations of muscles without experimental EMG. Then the muscle excitations, coupled with 3D joint angles, are used as input to the neuromusculoskeletal model.
4. *Static optimisation mode.* In this mode, without the aid of experimental EMG data, an optimization algorithm is used to construct all the muscle excitations to drive the neuromusculoskeletal model [EMHvdB07][TBB97].

Importantly, the different operation modes can be executed on the same neuromusculoskeletal model, allowing a consistent comparison among the different neural solutions. Fundamental to the current EMG-informed methods and the above modes of operation, CEINMS can be calibrated to the individual subject (Fig. 1.2) [LB03]. This is an optional operation procedure in CEINMS, which can therefore run in un-calibrated or calibrated state.

1.1.1 Calibration

The aim of calibration is to determine the values for a set of parameters for each musculotendon unit. The first parameter set defines the musculotendon unit's activation dynamics (see *Activation dynamics* and *Neural activation to muscle activation*), which characterise the transformation of muscle excitation to muscle activation. The second parameter set defines the musculotendon contraction dynam-

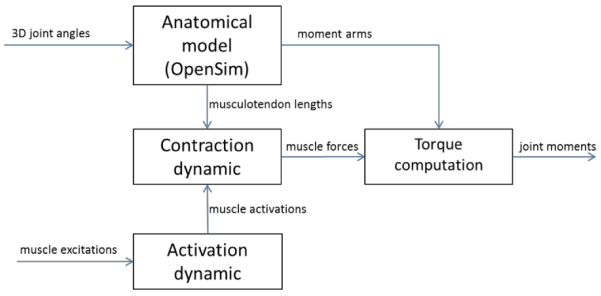


Figure 1.1: Figure 1: The schematic structure of CEINMS using the open-loop full predictive mode.

ics (*Contraction dynamics*), which transform the muscle activation and musculotendon kinematics into force [BLMB04][GSB+13][LB03][LB96][LB01][LBWB08][SFL13][SRFL12][WGKL13]. These parameters may change non-linearly across individuals, therefore an optimization algorithm, such as *simulated annealing* [GFR94a], is employed to alter the values of the parameters to enable close tracking of the experimental joint moments and/or excitations derived from EMG signals, which are acquired during the execution of different motor tasks [BLMB04][GSB+13][LB03][LB96][LB01][LBWB08][SFL13][SRFL12][WGKL13][WLBK09]. Various calibration control functions (e.g. minimize maximum activation, minimize maximum joint contact forces etc.) can also be implemented to direct the final set of model parameters [GSB+13][SFL13]. Finally, during calibration, the parameters are also constrained to vary within predefined boundaries to ensure that the muscles operate in their physiological range.

The result of the calibration is a subject-specific neuromusculoskeletal (NMS) model, which reflects the musculotendon physiology, activation and contraction dynamics for an individual. Finally, CEINMS can be validated with a novel set of input data, which has not been used for the calibration process, and run with any of the four execution modes.

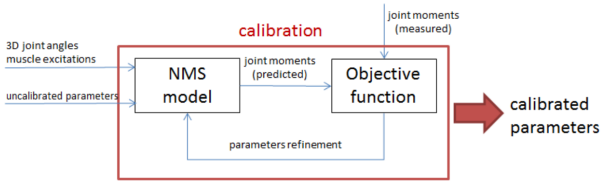


Figure 1.2: Figure 2: Schematic representation of the calibration procedure implemented in CEINMS. The neuromusculoskeletal (NMS) model is started with an initial set of un-calibrated parameters taken from literature. The parameters are refined using an optimization algorithm to minimize the error between the estimated and the measured joint moments.

1.1.2 Hybrid mode

Except for *full-predictive open-loop* mode, the operation modes require the solution of an optimization problem. For all problems, the objective function is defined as:

$$F_{obj} = \alpha * \sum_{k \in DOFs} (\tau_k - \tilde{\tau_k})^2 + \beta * \sum_{j \in MTUs} (e_j - \tilde{e_j})^2 + \gamma * \sum_{j \in MTUs} (e_j^2)$$

where τ_k is the moment at joint k as estimated by CEINMS, $\tilde{\tau_k}$ is the experimental moment at joint k, e_j is the estimated excitation for MTU j, and $\tilde{e_j}$ is the experimental excitation for MTU j.

Changing the weight ratios α, β, γ of the objective function will result in different behaviours, seamlessy shifting from static optimization to EMG-assisted mode (see *Using CEINMS*).

1.2 Neuromusculoskeletal models used in CEINMS

What follows is a description of all the various EMG-informed models that have been included in CEINMS. CEINMS implements two different models of the activation dynamics, which convert the neural drive to the muscle activation [Zaj88], and three models of the muscle contraction dynamics, which represent the transformation between the muscle activation and muscle kinematics into force (Fig. 1.3).

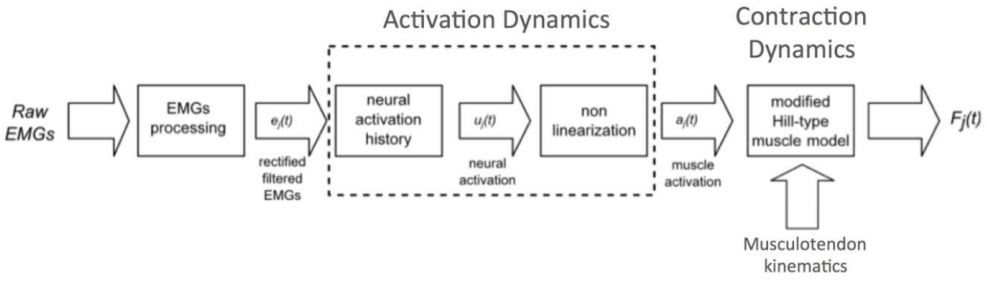


Figure 1.3: Figure 3: General data processing flow showing activation dynamics and contraction dynamics.

1.2.1 Activation dynamics

Muscle excitation signals $e(t)$ represent the neural drive to the muscles and are commonly extracted from experimental EMG signals. Typically, raw EMG signals are first high-pass filtered using a zero-lag fourth-order recursive Butterworth filter (30 Hz), then full wave rectified, and finally filtered using a Butterworth low-pass filter with a 6 Hz cutoff frequency. CEINMS software uses muscle excitation as input signals (see *Input data*).

Neural activation is derived from muscle excitation by modelling the muscle's twitch response in the activation dynamic model, which has been shown to improve muscle force predictions [BLMB04][LB03][LBWB08]. This is represented by a critically damped linear second-order differential system [MBSY73], expressed in a discrete form by using backward differences [BLMB04][LB03][LBWB08].

$$u_j(t) = \alpha e_j(t-d) - \beta_1 u_j(t-1) - \beta_2 u_j(t-2) \tag{1.1}$$

where $e_j(t)$ is the j-th muscle excitation at time t, $u_j(t)$ is the neural activation, α is the muscle gain coefficient, β_1 and β_2 are the recursive coefficients, and d is the electromechanical delay. A stable solution for this is obtained including the following constraints [BLMB04][LB03]

$$\beta_1 = C_1 + C_2$$
$$\beta_2 = C_1 \cdot C_2$$

where:

$$|C_1| < 1$$
$$|C_2| < 1$$

and

$$\alpha - \beta_1 - \beta_2 = 1$$

Neural activation to muscle activation

The relation between neural activation and the muscle activation is non-linear, and CEINMS has two different solutions [BLMB04][LB03][MB03]. The first was introduced by [LB03],

$$a_j(t) = \frac{e^{A_j u_j(t)} - 1}{e^{A_j} - 1} \tag{1.2}$$

where $a_j(t)$ is the activation of the j-th muscle, and A_j is the non-linear shape factor, constrained in the interval $(-3, 0)$.

The second model was introduced and described by [MB03]. The $u_j \rightarrow a_j$ transformation is defined as a piecewise parametric function.

$$\begin{aligned} a_j(t) &= \alpha_j^{act} ln(\beta_j^{act} u_j(t) + 1), 0 \leq u_j(t) \leq u_0 \\ a_j(t) &= m_j u_j(t) + c_j, u_0 \leq u_j(t) \leq 1 \end{aligned} \tag{1.3}$$

For each muscle j, the parameters α_j^{act}, β_j^{act}, m_j, c_j depend only on the shape factor A_j, constrained in the interval $(0, 0.12]$.

1.2.2 Contraction dynamics

Musculotendon kinematics and muscle activation (Fig. 1.3) are used as input for a modified Hill-type muscle model, which consists of an active force generating component, the muscle fibres, in series with a passive one, the tendon. The muscle fibre force depends on three main factors: $f_a(\widetilde{l}_m)$ is the active force-length relation that expresses the ability of muscle fibres to produce force at different lengths; $f_p(\widetilde{l}_m)$ is the passive force-length relation that represents the force response of the fibres to strain; and finally, $f_v(\widetilde{v}_m)$ accounts for the force contribution of the fibres contraction velocity. These curves, represented in Fig. 1.4 -a,b, are normalised to maximum isometric muscle force (F^{max}), to optimal fibre length (L_m^0) for a), and maximum muscle contraction velocity (v^{max}) for b). The optimal fibre length decreases as the activation increases, and this interaction is considered in the computation of the active force [Hui95]: curves for different levels of activation are shown, with 1.0 being 100% activation. The force (F^{mt}) produced by the musculotendon unit (MTU) is a function of muscle activation and muscle kinematics:

$$F^{mt} = F^t = F^{max}[f_a(\widetilde{l}_m) \cdot f_v(\widetilde{v}_m) \cdot a + f_p(\widetilde{l}_m) + d_m \cdot \widetilde{v}_m] \cdot \cos\varphi \tag{1.4}$$

where F^t is the tendon force, a is the muscle activation, d_m is the muscle damping element, and φ is the pennation angle of the fibres, which is function of the instantaneous fibre length l_m

$$\varphi = \sin^{-1}(\frac{L_m^0 \sin\varphi_0}{l_m}) \tag{1.5}$$

where φ_0 is the pennation angle of the fibres at their optimal length. The tendon element influences the estimation of the fibre length, i.e.

$$l_m = \frac{l_{mt} - l_t}{\cos\varphi} \tag{1.6}$$

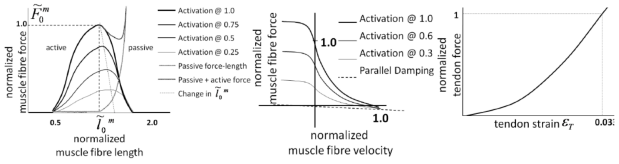


Figure 1.4: Figure 4: Active and passive force length curves. Values are normalised by F^{max} and L_m^0 with 1.0 being 100% activation. Optimal muscle fibre length was scaled with activation by a relationship experimentally determined in [Hui95] (b) Normalised force-velocity relationship. Note the parallel damping element added to prevent singularities in the inverted force-velocity relationship [Sch93] when activation or isometric force equals 0.0. (c) Exponential tendon force-strain relationship

Tendon models

CEINMS includes three different tendon models to estimate MTU forces. In the first, the equations for the musculotendon force dynamics are solved by numerically integrating a set of ordinary differential equations. This *integration elastic tendon (IET)* model calculates the muscle fibre length by forward integration of the muscle fibre velocity. The starting value of fibre velocity v^m is first estimated through an optimization routine, which distributes the total MTU velocity between fibres and tendon. v^m is then integrated using a Runge-Kutta-Fehlberg algorithm to calculate first l_m and l_t then from equation (1.6). The strain of the tendon is then calculated as

$$\epsilon = \frac{l_t - l_{ts}}{l_t} \tag{1.7}$$

where l_{ts} is the slack length of the tendon. Then, F^t is calculated using the force strain relation of the tendon (Fig. 1.4-c) and used with the activation to calculate the active and passive components of the force. Finally, v^m is calculated inverting the force velocity function, and used as new value for the next integration step. Unfortunately F^{mt} relies on the numerical integration of the stiff MTU equations and robust solutions are not always found.

A more robust implementation of the elastic tendon MTU model, which does not rely on forward integration, is the *equilibrium elastic tendon* (EET) model. The model uses a Van Wijngaarden-Dekker-Brent optimization routine to find the root of the equation

$$F^{mt}(\widetilde{l}_m) = F^t(\widetilde{l}_m) \tag{1.8}$$

where $F^t(\widetilde{l}_m)$ is obtained from the tendon force-strain relation expressing the tendon strain ϵ as a function of $\widetilde{l}_m$ by combination of equation (1.5), equation (1.6), and equation (1.7). $F^{mt}(\widetilde{l}_m)$ is obtained from equation (1.4), calculating $\widetilde{v}_m$ as numerical derivative of $\widetilde{l}_m$. This implementation provides a robust method for the solution of the fibre length, which, unlike the IET, always guarantees the equilibrium of the musculotendon unit.

The last tendon implementation is the *stiff tendon* (ST) model [SLRP10][SRPL12], which simplifies the tendon model in order to reduce the computation time. The tendon is considered as an element of infinite stiffness, with length equal to the slack length.

Although a complete comparison of the all proposed implementations of the elastic tendon is not yet available, the results of the two elastic tendon implementations have overlapping fibre length estimates (Fig. 1.5 -a). However, when the length of the tendon is short compared to the total length of the musculotendon unit, the integration model may produce wrong results. For example when examining the fibre length of the gracilis muscle during the stance phase of a walking trial (Fig. 1.5 -b), the fibre length calculated by the stiff tendon model (which presents a fixed tendon length) must always be greater than the one calculated using an elastic tendon. This is not the case for the integration elastic tendon model, which provides a wrong estimation of the fibre length. However, the equilibrium model has behaviour very close to the stiff model, which is compatible with the small ratio for l_{ts}/l_{mt}.

1.3 Appendices

1.3.1 Appendix A: Simulated Annealing

In this section we present the simulated annealing algorithm proposed by [CMMR87] and that we use for the calibration process. Part of this section is an extract from [GFR94b].

```
X = X_0
CALCULATE f(X)
X_opt = X
f_opt = f(X)

DO UNTIL convergence
    DO NT times
```

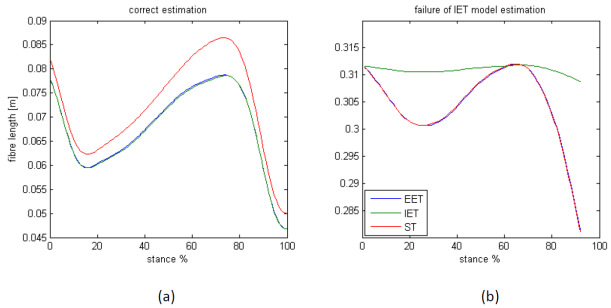


Figure 1.5: Figure 5: Estimation of muscle fibre length using three different tendon models. (a) The integration elastic tendon model (IET) and the equilibrium elastic tendon model (EET) produce the same estimation for the fibre length of the gastrocnemius medialis muscle. (b) The estimation of gracilis muscle fibre length given by IET and EET is different because of problems in the integration of the fibre velocity in the IET model. The fibre length estimated by IIT is greater than the one estimated by the stiff tendon model (ST), while the EET model has behaviour very close to the stiff model, which is compatible with the small ratio l_{ts}/l_{mt}.

```
8           DO NS times
9               DO k = 1, ... , n
10              x'_k = x_k + r * v_k, r in [-1,1)
11              CALCULATE f(X')
12              IF f(X') < f(X) THEN
13                  X = X'
14              END IF
15              IF f(X') >= f(X) THEN
16                  apply Metropolis criteria
17                  IF accepted: X = X'
18              END IF
19              IF f(X') > f_opt THEN
20                  X_opt = X, f_opt = f(X_opt)
21              END IF
22          END DO
23      END DO
24      ADJUST V such half of all trials are accepted
25  END DO
26  X*_h =X
27  increase h index
28  IF |f(X*_l) - f_opt| < eps, for each l=h, h-1, ... , h-N_eps        THEN
29      REPORT X_opt, f_opt, V
30      STOP
31  ELSE
32      T=r_T ... T, reduce T
33      X = X_opt, start at the current best optimum
34  END IF
35  CONTINUE
```

If `f(X')` is greater then or equal to `f(X)`, the Metropolis criterion decides on acceptance (lines 15 - 16). The value

$$p = e^{(f'-f)/T} \tag{1.9}$$

is computed and compared to `p'`, a uniformly distributed random number from `[0,1)`. If `p` is greater then `p'`, the new point is accepted. `X` is updated with `X` and the algorithm moves uphill (line 17). Otherwise, `X'` is rejected. Two factors decrease the probability of an uphill move: lower temperature and larger differences in the function values.

Every `NS` steps through all the elements of `X`, the step length vector `V` is adjusted so that half of all moves are accepted (line 24). The goal is to sample the function widely. If a greater percentage of points are accepted for `x_k`, then the

relevant element of V is enlarged. For a given temperature, this increases the number of rejections and decreases the percentage of acceptances. Every NT times through the above loops, the temperature T is reduced (line 32). The new temperature is given by

$$T' = r_T \cdot T \tag{1.10}$$

where rT ranges in [0,1). A lower temperature makes a given uphill move less likely, so the number of rejections increases and the step lengths decline. After a change in the temperature, the X vector is reset to the current X_opt (line 33). This selection of the starting point together with a smaller step focuses search efforts on the most promising area.

After the temperature reduction, we define X*_h=X where X is the vector used in the last function evaluation and h is increased every N_T times (lines 26 - 27). The algorithm ends by comparing f(X*_l) with f_opt, where l=h, h-1, ... , h-N_eps. If all the N_eps differences are less then epsilon, the algorithm terminates (lines 28 - 30). This criterion helps to ensure that global minimum is reached.

Part II

Reference Manuals

CHAPTER

TWO

CEINMS INSTALLATION INSTRUCTIONS

Please use the provided installation packages for Linux and Windows, according to your computer's architecture.

2.1 Windows

Execute the 32-bit or 64-bit version of the installer according to your computer's architecture. After you accept the licence agreement, you are shown the *Install Options* page. You should mark either `Add CEINMS to the system PATH for all users` or `Add CEINMS to the system PATH for current user`, otherwise it will be cumbersome to execute CEINMS from the command prompt. The `Create CEINMS Desktop Icon` instead is currently useless: no icons will be created anyway, since there are no graphical interfaces. You can proceed with the installation, selecting your preferred options for installation directory and *Start Menu* options.

2.1.1 Check your installation

Open a new command prompt and type:

```
CEINMS
```

You should read a brief help message. If this is not the case, please contact the developers.

CHAPTER

THREE

CEINMS CONFIGURATION FILES

CEINMS is entirely configured through XML files. Each file type is described by an XML schema (XSD)[1]. This allows to check the *formal* correctness of an XML file without even launching CEINMS, using any XML-validating software (for example, XML Copy Editor[2]). XSD files are provided together with the software.

Configuration files are presented in the following sections. An example of each file is provided and explained, and the corresponding grammar scheme is reported. Files are divided into three categories:

- *Execution (configuration parameters for a CEINMS simulation)*
- *Calibration (configuration parameters for a CEINMS calibration)*
- *Data description (representation of the subject model and experimental data)*

3.1 Execution

Files in this category serve to define how to perform the simulation (see *Using CEINMS*)

3.1.1 CEINMS main setup

This file is a simple list of the configuration and data files that define a CEINMS simulation.

CEINMS setup XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ceinms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="ceinmsSetup.xsd">
  <subjectFile>subjectCalibrated.xml</subjectFile>
  <inputDataFile>../TestData/walking1/walking1.xml</inputDataFile>
  <executionFile>Execution/executionTest.xml</executionFile>
  <excitationGeneratorFile>excitationGenerator-16To34.xml</excitationGeneratorFile>
  <outputDirectory>../../OutputData/walking1/</outputDirectory>
</ceinms>
```

A CEINMS setup file consists of a root element named `ceinms` that contains the following elements:

- `subjectFile` the location of the *subject description file*
- `inputDataFile` the location of the *trial (input) data description file*
- `executionFile` the location of the *execution parameters file*

[1] http://www.w3schools.com/schema/
[2] http://xml-copy-editor.sourceforge.net/

- `excitationGeneratorFile` the location of the *Excitation mappings description file*
- `outputDirectory` the folder where output results are saved

CEINMS setup XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>
  <xs:complexType name="CeinmsType">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element maxOccurs="1" name="subjectFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="inputDataFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="executionFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="excitationGeneratorFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="outputDirectory" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ceinms" type="CeinmsType"/>
</xs:schema>
```

3.1.2 Execution configuration file

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<execution xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="execution.xsd">
  <NMSmodel>
     <type>
        <hybrid>
            <alpha> 1 </alpha>
            <beta> 2 </beta>
            <gamma> 3 </gamma>
            <synthMTUs>psoas_r illiacus_r</synthMTUs>
            <adjustMTUs>
                          recfem_r vasmed_r vaslat_r vasint_r bicfemsh_r
                          gmin1_r gmin2_r gmin3_r gmed1_r gmed2_r gmed3_r
                          tfl_r addbrev_r addlong_r addmag1_r addmag2_r
                          addmag3_r gra_r sar_r semiten_r semimem_r bicfemlh_r
                          gmax1_r gmax2_r gmax3_r gaslat_r gasmed_r sol_r perlong_r
                          tibant_r perter_r perbrev_r
            </adjustMTUs>
            <algorithm>
                <simulatedAnnealing>
                    <noEpsilon>4</noEpsilon>
                    <rt>0.3</rt>
                    <T>20</T>
                    <NS>15</NS>
                    <NT>5</NT>
```

```
                    <epsilon>1.E-3</epsilon>
                    <maxNoEval>200000</maxNoEval>
                </simulatedAnnealing>
            </algorithm>
        </hybrid>
    </type>

    <tendon>
        <stiff/>
    </tendon>

    <activation>
        <exponential/>
    </activation>
  </NMSmodel>

  <offline/>

  <elaboratedDoFs>
  </elaboratedDoFs>

</execution>
```

An execution configuration file consists of a root element named `execution` that contains the following elements:

- `NMSmodel`: the specification of muscular model to simulate (see NMSmodel)
- an `online` or `offline` empty element, that determines wheter fiber velocities are computed using a causal filter (`online`) or a non-causal filter (`offline`, recommended for most applications)
- (optional) `elaboratedDoFs` to select the degrees of freedom (see *dof*) to elaborate

> **Note:** this feature is not implemented yet.

NMSmodel

This is the specification of the type of muscular model to use among the ones described in *Neuromusculoskeletal models used in CEINMS* and the desired operation mode:

- `type`: the operation mode might contain an `openLoop` empty element, or an `hybrid` element
- `tendon`: the behaviour of the tendon: it can be (see *Tendon models*)
  - `stiff` (ST) rigid tendon
  - `integrationElastic` (IET) integration elastic tendon
  - `equilibriumElastic` (EET) equilibrium elastic tendon
- `activation`: the formula to use to model the excitation-to-activation non-linear relation (see *Neural activation to muscle activation*)
  - `exponential`
  - `piecewise`

hybrid

The hybrid execution modality (see *Hybrid mode*) configuration is provided here:

- `alpha` weight for the sum of the squared difference between experimental joint moments provided as *input data* and the joint moments estimated by CEINMS
- `beta` weight for the sum of squared muscle excitations
- `gamma` weight for the sum of the squared difference between experimental and predicted muscle excitations
- `trackedMuscles` list of muscles for which an EMG reference signal to "track" is provided
- `predictedMuscles` list of muscles for which no EMG signal is available, therefore activation is only predicted using static optimization techniques
- `algorithm` the optimization algorithm that solves the hybrid static optimization problem, see *algorithm*

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>
  <xs:complexType name="TendonElementType">
    <xs:sequence>
      <xs:element name="tolerance" minOccurs="0" maxOccurs="1" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TendonType">
    <xs:choice>
      <xs:element name="stiff" type="TendonElementType"/>
      <xs:element name="integrationElastic" type="TendonElementType"/>
      <xs:element name="equilibriumElastic" type="TendonElementType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="ActivationElementType"/>
  <xs:complexType name="ActivationType">
    <xs:choice>
      <xs:element name="exponential" type="ActivationElementType"/>
      <xs:element name="piecewise" type="ActivationElementType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="SimulatedAnnealingType">
    <xs:sequence>
      <xs:element name="noEpsilon" type="xs:int"/>
      <xs:element name="rt" type="xs:double"/>
      <xs:element name="T" type="xs:double"/>
      <xs:element name="NS" type="xs:int"/>
      <xs:element name="NT" type="xs:int"/>
      <xs:element name="epsilon" type="xs:double"/>
      <xs:element name="maxNoEval" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="HybridAlgorithmType">
    <xs:choice>
      <xs:element name="simulatedAnnealing" type="SimulatedAnnealingType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="OpenLoopType"/>
```

```
  <xs:simpleType name="MuscleListType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>
  <xs:complexType name="HybridType">
    <xs:sequence>
      <xs:element name="alpha" type="xs:double"/>
      <xs:element name="beta" type="xs:double"/>
      <xs:element name="gamma" type="xs:double"/>
      <xs:element name="synthMTUs" type="MuscleListType"/>
      <xs:element name="adjustMTUs" type="MuscleListType"/>
      <xs:element name="algorithm" type="HybridAlgorithmType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TypeType">
    <xs:choice>
      <xs:element name="openLoop" type="OpenLoopType"/>
      <xs:element name="hybrid" type="HybridType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="NMSModelType">
    <xs:sequence>
      <xs:element name="type" type="TypeType"/>
      <xs:element name="tendon" type="TendonType"/>
      <xs:element name="activation" type="ActivationType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="ElaboratedDoFsType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>
  <xs:complexType name="ExecutionElementType"/>
  <xs:complexType name="FileType"/>
  <xs:complexType name="LoggingType">
    <xs:choice>
      <xs:element name="txt" type="FileType"/>
      <xs:element name="csv" type="FileType"/>
      <xs:element name="mot" type="FileType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="ExecutionType">
    <xs:sequence>
      <xs:element name="NMSmodel" type="NMSModelType"/>
      <xs:choice>
        <xs:element name="online" type="ExecutionElementType"/>
        <xs:element name="offline" type="ExecutionElementType"/>
      </xs:choice>
      <xs:element minOccurs="0" name="elaboratedDoFs" type="ElaboratedDoFsType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="execution" type="ExecutionType"/>
</xs:schema>
```

3.2 Calibration

Files in this category serve to define how to perform the model calibration (see *Using CEINMScalibrate*)

3.2.1 CEINMScalibrate configuration file

This file is a simple list of the configuration and data files that define a CEINMS subject calibration procedure.

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ceinmsCalibration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:noNamespaceSchemaLocation="ceinmsCalibrationSetup.xsd">
  <subjectFile>subject.xml</subjectFile>
  <excitationGeneratorFile>excitationGenerator-16To34.xml</excitationGeneratorFile>
  <calibrationFile>Calibration/calibrationTest.xml</calibrationFile>
  <outputSubjectFile>subjectCalibrated.xml</outputSubjectFile>
</ceinmsCalibration>
```

A CEINMScalibrate setup file consists of a root element named `ceinmsCalibration` that contains the following elements:

- `subjectFile` the location of the *subject description file*
- `excitationGeneratorFile` the location of the *Excitation mappings description file*
- `calibrationFile` the location of the *calibration parameters file* (including which trials to calibrate on)
- `outputSubjectFile` the name of the output (calibrated) subject file.

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>
  <xs:complexType name="CeinmsCalibrationType">
    <xs:sequence>
      <xs:choice maxOccurs="unbounded">
        <xs:element maxOccurs="1" name="subjectFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="excitationGeneratorFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="calibrationFile" type="xs:string"/>
        <xs:element maxOccurs="1" name="outputSubjectFile" type="xs:string"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ceinmsCalibration" type="CeinmsCalibrationType"/>
</xs:schema>
```

3.2.2 Calibration configuration file

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<calibration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="calibration.xsd">
```

```
<algorithm>
  <simulatedAnnealing>
    <noEpsilon>4</noEpsilon>
    <rt>0.1</rt>
    <T>200000</T>
    <NS>30</NS>
    <NT>20</NT>
    <epsilon>1.E-5</epsilon>
    <maxNoEval>200000</maxNoEval>
  </simulatedAnnealing>
</algorithm>

<NMSmodel>
  <type>
    <openLoop/>
  </type>

  <tendon>
    <stiff/>
  </tendon>

  <activation>
    <exponential/>
  </activation>
</NMSmodel>

  <calibrationSteps>
      <step>
          <dofs>knee_angle_r hip_flexion_r hip_adduction_r ankle_angle_r</dofs>
          <objectiveFunction>
              <minimizeTorqueError/>
          </objectiveFunction>
          <parameterSet>
              <parameter>
                  <name>c1</name>
                  <single/>
                  <absolute>
                      <range>-0.95 -0.05</range>
                  </absolute>
              </parameter>
              <parameter>
                  <name>c2</name>
                  <single/>
                  <absolute>
                      <range>-0.95 -0.05</range>
                  </absolute>
              </parameter>
              <parameter>
                  <name>shapeFactor</name>
                  <single/>
                  <absolute>
                      <range>-2.999 -0.001</range>
                  </absolute>
              </parameter>
              <parameter>
                  <name>tendonSlackLength</name>
                  <single/>
                  <relativeToSubjectValue>
```

```
                    <range>0.85 1.15</range>
                </relativeToSubjectValue>
            </parameter>
            <parameter>
                <name>strengthCoefficient</name>
                <muscleGroups>
                    <muscles>addbrev_r addlong_r addmag1_r
                             addmag2_r addmag3_r </muscles>
                    <muscles>bicfemlh_r semimem_r semiten_r sar_r</muscles>
                    <muscles>bicfemsh_r</muscles>
                    <muscles>gaslat_r gasmed_r </muscles>
                    <muscles>gmax1_r gmax2_r gmax3_r</muscles>
                    <muscles>gmed1_r gmed2_r gmed3_r
                             gmin1_r gmin2_r gmin3_r</muscles>
                    <muscles>gra_r recfem_r tfl_r</muscles>
                    <muscles>illiacus_r psoas_r</muscles>
                    <muscles>perbrev_r perlong_r perter_r tibant_r</muscles>
                    <muscles> sol_r</muscles>
                    <muscles>vasint_r vaslat_r vasmed_r</muscles>
                </muscleGroups>
                <absolute>
                    <range>0.5 2.5</range>
                </absolute>
            </parameter>

        </parameterSet>
    </step>
  </calibrationSteps>

 <trialSet>../../subject01/walking1/walking1.xml</trialSet>

</calibration>
```

A calibration configuration file consists of a root element named `calibration` that contains the following elements:

- `algorithm`: a description of the optimization algorithm to use (see algorithm)
- `NMSmodel`: a collection of options for the simulation of the model (see the corresponding section *NMSmodel* in the execution description file)
- `calibrationSteps`: a list of `step` elements (at least one) that describe an optimization procedure (see step)
- `trialSet`: a list of XML files, each describing a trial to be used for calibration (see *Trial (input) data description file*)

algorithm

This is the optimization algorithm that is used to minimize the objective function associated to each calibration step (`objectiveFunction` element). At the moment, only the *simulated annealing* algorithm is available, that is described in *Appendix A: Simulated Annealing*.

The parameters for the simulated annealing algorithm are:

- `noEpsilon` number of consecutive times that the cost function differs less than `epsilon` from the current optimum value
- `rt` coefficient by which the temperature is multiplied when it is reduced

- `T` initial value for the temperature
- `NS` number of evaluation steps after which the step length is re-adjusted
- `NT` number of step-length adjustment loops after which the temperature is decreased
- `epsilon` threshold value to consider negligible the difference of current cost function value from optimal one
- `maxNoEval` maximum number of evaluations of the cost function

step

This is the description of a calibration tasks. It contains the following elements:

- `dofs`: a list of the degrees of freedom to consider for model calibration
- `objectiveFunction`: the target function to minimize in order to calibrate the subject model
- `parameterSet`: a list of `parameter` elements, each describing how to calibrate one type of muscle parameter (see parameter

parameter

- `name`: parameter identifier
- a specifier for which muscles "share" the same parameter value (this varies the number of parameters to calibrate):
  - `global`: single value for all muscles, or
  - `single`: different value for each muscle, or
  - `muscleGroups`: a list of `muscles` elements, each containing a list of muscles that share the same value for the parameter
- a specifier for the calibration range. All elements of this type contain a `range` element, that consists of two numbers, representing respectively the minimum and maximum value of the range. The actual extremes depend on the type of specifier:
  - `absolute`: the values in `range` are the absolute minimum and maximum values that are allowed for the parameter
  - `relativeToSubjectValue`: the values in `range` represent the multiplying factor by which to multiply the initial value for this parameter (in the input subject description XML, see *Subject description file*) to obtain the actual minimum and maximum values.

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>
  <xs:complexType name="SimulatedAnnealingType">
    <xs:sequence>
      <xs:element name="noEpsilon" type="xs:int"/>
      <xs:element name="rt" type="xs:double"/>
```

```
      <xs:element name="T" type="xs:double"/>
      <xs:element name="NS" type="xs:int"/>
      <xs:element name="NT" type="xs:int"/>
      <xs:element name="epsilon" type="xs:double"/>
      <xs:element name="maxNoEval" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="AlgorithmType">
    <xs:choice>
      <xs:element name="simulatedAnnealing" type="SimulatedAnnealingType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="OpenLoopType"/>
  <xs:complexType name="TypeType">
    <xs:choice>
      <xs:element name="openLoop" type="OpenLoopType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="TendonElementType">
    <xs:sequence>
      <xs:element name="tolerance" minOccurs="0" maxOccurs="1" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TendonType">
    <xs:choice>
      <xs:element name="stiff" type="TendonElementType"/>
      <xs:element name="integrationElastic" type="TendonElementType"/>
      <xs:element name="equilibriumElastic" type="TendonElementType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="ActivationElementType"/>
  <xs:complexType name="ActivationType">
    <xs:choice>
      <xs:element name="exponential" type="ActivationElementType"/>
      <xs:element name="piecewise" type="ActivationElementType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="NMSModelType">
    <xs:sequence>
      <xs:element name="type" type="TypeType"/>
      <xs:element name="tendon" type="TendonType"/>
      <xs:element name="activation" type="ActivationType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="TrialSetType">
    <xs:list itemType="xs:string"/>
  </xs:simpleType>
  <xs:complexType name="ObjectiveFunctionElementType"/>
  <xs:complexType name="GlobalParameterType"/>

  <xs:simpleType name="SingleParameterType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>

  <xs:complexType name="ObjectiveFunctionType">
    <xs:choice>
      <xs:element name="minimizeTorqueError" type="ObjectiveFunctionElementType"/>
    </xs:choice>
```

```
</xs:complexType>
<xs:simpleType name="DoFsType">
  <xs:list itemType="xs:token"/>
</xs:simpleType>
<xs:simpleType name="TwoDoublesType">
  <xs:list itemType="xs:double"/>
</xs:simpleType>

<xs:complexType name="RangeType">
  <xs:sequence>
     <xs:element name="range" type="TwoDoublesType"/>
  </xs:sequence>
</xs:complexType>


<xs:simpleType name="ParameterName">
  <xs:restriction base="xs:token">
    <xs:enumeration value="c1"/>
    <xs:enumeration value="c2"/>
    <xs:enumeration value="shapeFactor"/>
    <xs:enumeration value="optimalFibreLength"/>
    <xs:enumeration value="pennationAngle"/>
    <xs:enumeration value="tendonSlackLength"/>
    <xs:enumeration value="maxContractionVelocity"/>
    <xs:enumeration value="maxIsometricForce"/>
    <xs:enumeration value="strengthCoefficient"/>
    <xs:enumeration value="emDelay"/>
    <xs:enumeration value="percentageChange"/>
    <xs:enumeration value="damping"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="MuscleListType">
  <xs:list itemType="xs:token"/>
</xs:simpleType>
<xs:complexType name="MuscleGroupsType">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="muscles" type="MuscleListType"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ParameterType">
  <xs:sequence>
    <xs:element name="name" type="ParameterName"/>
    <xs:choice>
      <xs:element name="muscleGroups" type="MuscleGroupsType"/>
      <xs:element name="global" type="GlobalParameterType"/>
      <xs:element name="single" type="SingleParameterType"/>
    </xs:choice>
    <xs:choice>
      <xs:element name="absolute" type="RangeType"/>
      <xs:element name="relativeToSubjectValue" type="RangeType"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="parameterSetType">
  <xs:sequence>
    <xs:element minOccurs="1" maxOccurs="unbounded" name="parameter" type="ParameterType"/>
  </xs:sequence>
```

```
  </xs:complexType>
  <xs:complexType name="StepType">
    <xs:sequence>
      <xs:element name="dofs" type="DoFsType"/>
      <xs:element name="objectiveFunction" type="ObjectiveFunctionType"/>
      <xs:element name="parameterSet" type="parameterSetType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CalibrationStepsType">
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="unbounded" name="step" type="StepType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CalibrationType">
    <xs:sequence>
      <xs:element name="algorithm" type="AlgorithmType"/>
      <xs:element name="NMSmodel" type="NMSModelType"/>
      <xs:element name="calibrationSteps" type="CalibrationStepsType"/>
      <xs:element name="trialSet" type="TrialSetType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="calibration" type="CalibrationType"/>
</xs:schema>
```

3.3 Data description

Files belonging to this category serve to describe the data that is required by the former categories, i.e., the model under investigation and how to find the input (experimental) data

3.3.1 Subject description file

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<subject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="subject.xsd">
    <mtuDefault>
        <emDelay>0.015</emDelay>
        <percentageChange>0.15</percentageChange>
        <damping>0.1</damping>
        <curve>
            <name>activeForceLength</name>
            <xPoints>-5 0 0.401 0.402 0.4035 0.52725 0.62875 0.71875 0.86125 1.045
                     1.2175 1.43875 1.61875 1.62 1.621 2.2 5</xPoints>
            <yPoints>0 0 0 0 0 0.226667 0.636667 0.856667 0.95 0.993333 0.77
                    0.246667 0 0 0 0 0</yPoints>
        </curve>
        <curve>
            <name>passiveForceLength</name>
            <xPoints>-5 0.998 0.999 1 1.1 1.2 1.3 1.4 1.5 1.6 1.601 1.602 5</xPoints>
            <yPoints>0 0 0 0 0.035 0.12 0.26 0.55 1.17 2 2 2 2</yPoints>
        </curve>
        <curve>
            <name>forceVelocity</name>
```

```
            <xPoints>-10 -1 -0.6 -0.3 -0.1 0 0.1 0.3 0.6 0.8 10</xPoints>
            <yPoints>0 0 0.08 0.2 0.55 1 1.4 1.6 1.7 1.75 1.75</yPoints>
        </curve>
        <curve>
            <name>tendonForceStrain</name>
            <xPoints>-10 -0.002 -0.001 0 0.00131 0.00281 0.00431 0.00581
                0.00731 0.00881 0.0103 0.0118 0.0123 9.2 9.201 9.202 20</xPoints>
            <yPoints>0 0 0 0 0.0108 0.0257 0.0435 0.0652 0.0915
                     0.123 0.161 0.208 0.227 345 345 345 345</yPoints>
        </curve>
    </mtuDefault>
    <mtuSet>
        <mtu>
            <name>add_brev_r</name>
            <c1>0.768956</c1>
            <c2>-0.990048</c2>
            <shapeFactor>0.101198</shapeFactor>
            <optimalFibreLength>0.152726160798647</optimalFibreLength>
            <pennationAngle>0</pennationAngle>
            <tendonSlackLength>0.0229663399697213</tendonSlackLength>
            <maxIsometricForce>429</maxIsometricForce>
            <strengthCoefficient>0.5</strengthCoefficient>
        </mtu>

                ...

        <mtu>
            <name>vas_med_r</name>
            <c1>0.768956</c1>
            <c2>-0.990048</c2>
            <shapeFactor>0.101198</shapeFactor>
            <optimalFibreLength>0.1021279880827</optimalFibreLength>
            <pennationAngle>0.08726646</pennationAngle>
            <tendonSlackLength>0.144585690993485</tendonSlackLength>
            <maxIsometricForce>1294</maxIsometricForce>
            <strengthCoefficient>1.01759</strengthCoefficient>
        </mtu>
    </mtuSet>
    <dofSet>
        <dof>
            <name>hip_flexion_r</name>
            <mtuNameSet>add_brev_r add_long_r add_mag1_r add_mag2_r add_mag3_r
                        bifemlh_r glut_max1_r glut_max2_r glut_max3_r glut_med1_r
                        glut_med2_r glut_med3_r glut_min1_r glut_min2_r
                        glut_min3_r grac_r iliacus_r psoas_r rect_fem_r
                        sar_r semimem_r semiten_r tfl_r</mtuNameSet>
        </dof>
        <dof>
            <name>hip_adduction_r</name>
            <mtuNameSet>add_brev_r add_long_r add_mag1_r add_mag2_r add_mag3_r
                        bifemlh_r glut_max1_r glut_max2_r glut_max3_r glut_med1_r
                        glut_med2_r glut_med3_r glut_min1_r glut_min2_r
                        glut_min3_r grac_r iliacus_r psoas_r rect_fem_r
                        sar_r semimem_r semiten_r tfl_r</mtuNameSet>
        </dof>
        <dof>
            <name>hip_rotation_r</name>
            <mtuNameSet>add_brev_r add_long_r add_mag1_r add_mag2_r add_mag3_r
```

```
                        bifemlh_r glut_max1_r glut_max2_r glut_max3_r glut_med1_r
                        glut_med2_r glut_med3_r glut_min1_r glut_min2_r
                        glut_min3_r grac_r iliacus_r psoas_r rect_fem_r
                        sar_r semimem_r semiten_r tfl_r</mtuNameSet>
        </dof>
        <dof>
            <name>knee_angle_r</name>
            <mtuNameSet>bifemlh_r bifemsh_r lat_gas_r med_gas_r grac_r
                        rect_fem_r sar_r semimem_r semiten_r tfl_r
                        vas_int_r vas_lat_r vas_med_r</mtuNameSet>
        </dof>
        <dof>
            <name>ankle_angle_r</name>
            <mtuNameSet>lat_gas_r med_gas_r per_brev_r per_long_r
                        per_tert_r soleus_r tib_ant_r</mtuNameSet>
        </dof>
    </dofSet>
    <calibrationInfo>
        <calibrated>
            <startSubjectFile>subject.xml</startSubjectFile>
            <calibrationSequence/>
        </calibrated>
    </calibrationInfo>
</subject>
```

A subject description file consists of a root element named `subject` that contains the following elements:

- `mtuDefault` that contains the parameters that describe muscle (or, to be more precise, muscle-tendon-unit, MTU) properties common to all muscles (see mtuDefault)
- `mtuSet` a list of `mtu` elements, each describing a muscle-tendon unit that actuates the model (see mtu)
- `dofSet` a list of `dof` elements, each describing a degree of freedom of the model (see dof)
- `calibrationInfo` that reports whether the subject has been calibrated (see calibrationInfo)

mtuDefault

The parameters common to all muscles are:

- `emDelay` the electromechanical delay (d in *Activation dynamics*)
- `percentageChange` the percentage change in optimal fibre length depending on activation (γ in *Contraction dynamics*)
- `damping` the muscle damping coefficient (d_m in *Contraction dynamics*)
- `maxContractionVelocity` the normalised maximum contraction velocity of the fibre
- four `curve` elements, describing the normalized force/length (active and passive), normalized force/velocity, normalized tendon force/strain curves shown in the *active and passive force curves*) figure in section *Contraction dynamics*. Each `curve` has a name, a list of y-coordinates, and the corresponding x-coordinates that can be interpolated to provide the complete curves.

mtu

Each muscle tendon unit is further described by the following, muscle-specific properties:

- `name` the identifier for the MTU

- `c1` and `c2` are the constants C_1 and C_2 that describe the recursive coefficients in *Activation dynamics*
- `shapeFactor` is the non-linear shape factor that describes the relation between neural excitation and the muscle activation (A_j in *Activation dynamics*)
- `optimalFibreLength` is the optimal fibre length at maximum activation, (L_m^0 in *Contraction dynamics*)
- `pennationAngle` is the pennation angle of the fibre at its optimal length (φ_0 in *Contraction dynamics*)
- `tendonSlackLength` is the tendon slack length (l_{ts} in *Contraction dynamics*)
- `maxContractionVelocity` the normalised maximum contraction velocity of the fibre
- `maxIsometricForce` is the maximum isometric muscle force (F^{max} in *Contraction dynamics*)
- `strengthCoefficient` is a multiplicative factor for `maxIsometricForce`; the rationale for its inclusion is that, instead of calibrating the `maxIsometricForce` parameter for each muscle, you may want to keep it fixed and calibrate a reduced number of `strength coefficient` parameters instead, that can be shared by multiple muscles (e.g., belonging to the same muscle group - this is under the assumption that muscles in the same group develop in the same way).

dof

Each degree of freedom is thus defined:

- `name` the identifier for the degree of freedom
- `mtuNameSet` a list of the muscles that act on it

calibrationInfo

If the file represents an uncalibrated subject, `calibrationInfo` will contain an `uncalibrated` element, made of:

- a `subjectID` element, containing a literal identifier for the subject
- an `additionalInfo` element, containing any additional note or information on the "origin" of the file (i.e., any pre-processing or scaling step that was performed)

If the file represents a calibrated subject (i.e, it is an output of the *CEINMScalibrate software*), `calibrationInfo` will contain a `calibrated` element, made of:

- `startSubjectFile` the original XML subject file that was used as starting point of calibration
- `calibrationSequence` a list of the calibration steps that were performed (see *CEINMScalibrate configuration file*)

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>

  <xs:complexType name="AngleInputType">
    <xs:simpleContent>
```

```
      <xs:extension base="xs:double">
        <xs:attribute name="unit" default="rad">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="deg" />
              <xs:enumeration value="rad" />
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>


  <xs:complexType name="MTUType">
    <xs:sequence>
      <xs:element name="name" type="xs:token"/>
      <xs:element minOccurs="0" name="emDelay" type="xs:double"/>
      <xs:element name="c1" type="xs:double"/>
      <xs:element name="c2" type="xs:double"/>
      <xs:element name="shapeFactor" type="xs:double"/>
      <xs:element name="optimalFibreLength" type="xs:double"/>
      <xs:element name="pennationAngle" type="AngleInputType"/>
      <xs:element name="tendonSlackLength" type="xs:double"/>
      <xs:element minOccurs="0" name="maxContractionVelocity" type="xs:double"/>
      <xs:element minOccurs="0" name="damping" type="xs:double"/>
      <xs:element name="maxIsometricForce" type="xs:double"/>
      <xs:element name="strengthCoefficient" type="xs:double"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MTUSetType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="mtu" type="MTUType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="MTUNameSetType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>
  <xs:complexType name="DoFType">
    <xs:sequence>
      <xs:element name="name" type="xs:token"/>
      <xs:element name="mtuNameSet" type="MTUNameSetType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="CalibrationSequenceType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>
  <xs:complexType name="CalibratedType">
    <xs:sequence>
      <xs:element name="startSubjectFile" type="xs:string"/>
      <xs:element name="calibrationSequence" type="CalibrationSequenceType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="UncalibratedType">
    <xs:sequence>
      <xs:element name="subjectID" type="xs:token"/>
      <xs:element name="additionalInfo" type="xs:string"/>
    </xs:sequence>
```

```
  </xs:complexType>
  <xs:complexType name="CalibrationInfoType">
    <xs:choice>
      <xs:element name="uncalibrated" type="UncalibratedType"/>
      <xs:element name="calibrated" type="CalibratedType"/>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="DoFSetType">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="1" name="dof" type="DoFType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="PointsSequenceType">
    <xs:list itemType="xs:double"/>
  </xs:simpleType>
  <xs:complexType name="CurveType">
    <xs:sequence>
      <xs:element name="name" type="xs:token"/>
      <xs:element name="xPoints" type="PointsSequenceType"/>
      <xs:element name="yPoints" type="PointsSequenceType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="MTUDefaultType">
    <xs:sequence>
      <xs:element name="emDelay" type="xs:double"/>
      <xs:element name="percentageChange" type="xs:double"/>
      <xs:element name="damping" type="xs:double"/>
      <xs:element maxOccurs="4" minOccurs="0" name="curve" type="CurveType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SubjectType">
    <xs:sequence>
      <xs:element name="mtuDefault" type="MTUDefaultType"/>
      <xs:element name="mtuSet" type="MTUSetType"/>
      <xs:element name="dofSet" type="DoFSetType"/>
      <xs:element name="calibrationInfo" type="CalibrationInfoType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="subject" type="SubjectType"/>
</xs:schema>
```

3.3.2 Trial (input) data description file

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<inputData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:noNamespaceSchemaLocation="inputData.xsd">
    <muscleTendonLengthFile>
            MuscleAnalysis/subject01_MuscleAnalysis_Length.sto
    </muscleTendonLengthFile>
    <excitationsFile>emg.mot</excitationsFile>
    <momentArmsFiles>
        <momentArmsFile dofName="ankle_angle_r">
              MuscleAnalysis/subject01_MuscleAnalysis_MomentArm_ankle_angle_r.sto
        </momentArmsFile>
```

```
        <momentArmsFile dofName="hip_adduction_r">
              MuscleAnalysis/subject01_MuscleAnalysis_MomentArm_hip_adduction_r.sto
        </momentArmsFile>
        <momentArmsFile dofName="hip_flexion_r">
              MuscleAnalysis/subject01_MuscleAnalysis_MomentArm_hip_flexion_r.sto
        </momentArmsFile>
        <momentArmsFile dofName="knee_angle_r">
              MuscleAnalysis/subject01_MuscleAnalysis_MomentArm_knee_angle_r.sto
        </momentArmsFile>
    </momentArmsFiles>
    <externalTorquesFile>ID/inversedynamics.sto</externalTorquesFile>
</inputData>
```

An input data description file consists of a root element named `inputData` that contains the following elements:

- `muscleTendonLengthFile`: the location (absolute or relative to the location of the xml file itself) of the file containing the length of the muscle-tendon units during the trial
- `momentArmsFiles`: a list of `momentArmsFile` elements, each being the location (absolute or relative to the location of the xml file itself) of the file containing the moment arms of the muscle-tendon units, relative to the degree of freedom given as `dofName` attribute, during the trial
- `excitationsFile`: the location (absolute or relative to the location of the xml file itself) of the file containing the excitations (usually, EMG signal envelopes) recorded for the trial
- (optionally) `externalTorquesFile`: the location (absolute or relative to the location of the xml file itself) of the file containing the torque that the subject exerts at each joint.

For a more detailed explanation of the format and content of all input data files, see *Preparing your experimental data*.

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:annotation>
        <xs:documentation xml:lang="en"/>
    </xs:annotation>

    <xs:complexType name="momentArmsFileType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="dofName" use="required" type="xs:string" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="momentArmsListType">
        <xs:sequence>
             <xs:element maxOccurs="unbounded"
                         name="momentArmsFile" type="momentArmsFileType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="InputDataType">
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element minOccurs="1" maxOccurs="1"
                            name="muscleTendonLengthFile" type="xs:string"/>
```

```
                <xs:element minOccurs="1" maxOccurs="1"
                            name="momentArmsFiles" type="momentArmsListType"/>
                <xs:element minOccurs="1" maxOccurs="1"
                            name="excitationsFile" type="xs:string"/>
                <xs:element maxOccurs="1"
                            name="externalTorquesFile" type="xs:string"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="inputData" type="InputDataType"/>
</xs:schema>
```

3.3.3 Excitation mappings description file

XML example

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<excitationGenerator xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:noNamespaceSchemaLocation="excitationGenerator.xsd">

<inputSignals type="EMG">addmag_r bicfemlh_r gaslat_r gasmed_r
            gmax_r gmed_r gra_r perlong_r
            recfem_r sar_r semimem_r sol_r
            tfl_r tibant_r vaslat_r vasmed_r
</inputSignals>

<mapping>
    <excitation id="add_brev_r">
        <input weigth="1"> addmag_r </input>
    </excitation>
    <excitation id="add_long_r">
        <input weigth="1"> addmag_r </input>
    </excitation>
    <excitation id="glut_min1_r">
        <input weigth="0.5"> gmed_r </input>
        <input weigth="0.5"> gmax_r </input>
    </excitation>
    <excitation id="psoas_r"/>
 </mapping>

</excitationGenerator>
```

An excitations mapping description file consists of a root element named `excitationGenerator` that contains the following elements:

- `inputSignals`: a list of identifiers for the input signals that are selected or combined to generate the excitation patterns for the model's MTUs. The identifiers need to match the labels that will be present in the excitations data file (see *Input data*). Optionally, a `type` attribute can be added to specify the nature of these input signals (e.g., *EMGenvelopes*, *synergies*, ...)
- `mapping`: a list of excitation elements

excitation

An `excitation` element is identified by an `id` literal attribute, that must correspond to an MTU name in the subject's model. It contains a set of `input` elements, each being the name of one of the abovementioned

`inputSignals`, with a corresponding `weight` numeral attribute. The excitation value for the MTU will be given by a linear combination of the listed input signals, each multiplied by its weight.

Examples of most common cases:

- one-to-one mapping is represented as a single input signal with weight 1
- the mean of two signals is represented as a list of the two signals, each with weight 0.5
- an unknown excitation is represented as an empty element.

XSD grammar

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!--Authors: Elena Ceseracciu, Claudio Pizzolato, Monica Reggiani -->

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="en"/>
  </xs:annotation>

  <xs:simpleType name="StringListType">
    <xs:list itemType="xs:token"/>
  </xs:simpleType>

  <xs:complexType name="InputSignalsType">
    <xs:simpleContent>
        <xs:extension base="StringListType">
          <xs:attribute type="xs:token" name="type"/>
        </xs:extension>
     </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="InputType">
    <xs:simpleContent>
        <xs:extension base="xs:token">
          <xs:attribute type="xs:double" name="weight" use="required"/>
        </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="InputListType">

  </xs:complexType>

  <xs:complexType name="ExcitationType">
    <xs:sequence>
      <xs:element name="input" maxOccurs="unbounded" minOccurs="0" type="InputType"/>
    </xs:sequence>
    <xs:attribute type="xs:token" name="id" use="required"/>
  </xs:complexType>

  <xs:complexType name="ExcitationListType">
    <xs:sequence>
      <xs:element name="excitation" maxOccurs="unbounded" minOccurs="1"
                  type="ExcitationType"/>
    </xs:sequence>
  </xs:complexType>
```

```
  <xs:element name="excitationGenerator">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="inputSignals" maxOccurs="1" type="InputSignalsType"/>
        <xs:element name="mapping" minOccurs="1" type="ExcitationListType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

CHAPTER

FOUR

PREPARING YOUR EXPERIMENTAL DATA

The aim of CEINMS is to simulate muscle dynamics, taking into account information on muscle excitations and the subject's kinematics during a task. Joint kinematics data collected experimentally however is not directly used as input for the CEINMS software: preprocessing steps are required to compute how the subject's motion reflects on the *geometrical* state of each muscle. The length of the muscle-tendon unit is a geometric property that affects force generation in Hill-type muscle models, while the moment arm of a muscle insertion point with respect to a joint rotation axis determines how that forces contributes to the joint total moment. This preprocessing step can be done through an anatomical modeling tool such as OpenSim[1]. In this section, an explanation of the input data files required for CEINMS execution will be provided, together with some indications on how to collect experimental data for EMG-driven muscle simulations.

4.1 Input data

The input quantities to run a CEINMS simulation from experimental data are:

- *musculo-tendon lengths* for the muscles in the model, calculated by means of an anatomical musculoskeletal model of the subject
- *moment arms* for the muscles that insist on each joint, calculated by means of an anatomical musculoskeletal model of the subject
- *muscle excitations* usually estimated from sEMG signals

To calibrate the subject's model on experimental data, for each of the involved trials one also needs

- *joint moments* at the joints of interest (usually estimated by means of inverse dynamics, see Exploiting OpenSim)

Since these quantities are usually computed with different tools and in different steps, as will be shown in the next sections, they are stored in multiple files. In order to let the software know which files correspond to each trial, a configuration file for each trial has to be prepared, according to *Trial (input) data description file*.

4.1.1 Storage file format

File formats such OpenSim motion (.mot) files[2] or OpenSim storage (.sto) files[3] are accepted for input files. Output files are written as OpenSim storage.

The specifications for CEINMS input files however are more relaxed than OpenSim ones:

- there is an header section, that ends with a `endheader` line, followed by a data section

[1] http://simtk.org/home/opensim
[2] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Motion+(.mot)+Files
[3] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Storage+(.sto)+Files

- within the header section, a line must indicate the number of total columns which is the first one) that will be found in the data section, for example 15, in one of these two formats:
  - nColumns=15
  - dataColumns 15
- within the header section, a line must indicate the number of data rows (samples) that will be found in the data section, for example 1264, in one of these two formats:
  - nRows=1264
  - dataRows 1264
- the data section begins with with a line containing tab-delimited or space-delimited labels for each column. The first column is *time*, followed by the list of muscle names or degrees of freedom that identify the data (note that these labels must match the names written in the XML configuration files, i.e., degrees of freedom and muscle names must match the *Subject description file*, and muscle activations must match the *inputSignals* in *Excitation mappings description file*). The rows below this line of column labels must be the corresponding values of each of these quantities at the time represented by the first number in each row.

4.2 Additional acquisitions

4.2.1 Static acquisition

When acquiring exponential data, we suggest that, **for each subject**, you acquire a static pose of the subject with your motion capture system. This allows to create a more subject-specific model, starting from generic models, as explained in Scaling.

4.2.2 Maximum voluntary contractions (MVC) acquisitions

Since your subject's model should be independent from the electrode configuration you use during an acquisition session, you need to normalize each muscle's EMG data to a repeatable reference value. Currently, the most common way among CEINMS users to acquire these reference values is by having the subject perform, **for each acquisition session**, a set of tasks in which one or more muscles are maximally contracted isometrically (no motion occurring), according to a pre-defined protocol.

4.3 Exporting and converting data

In order to elaborate the experimental data acquired with the motion analysis equipment available in your laboratory, we suggest that first of all you export your acquisitions into a standard C3D format[4], which is supported by almost all manufacturers and vendors.

Secondly, we suggest that you use a conversion tool such as MOtoNMS[5] to extract and convert the data relevant for NMS modeling into appropriate file formats.

[4] http://www.c3d.org/
[5] https://simtk.org/home/motonms

4.3.1 MOtoNMS

MOtoNMS is an open source MATLAB[6] toolbox that allows to pre-process data in a transparent and repeatable way. Moreover, it can be configured via XML files (or user-friendly GUIs) to adapt to different laboratory setups, without requiring any custom coding.

Key outputs of MOtoNMS are:

- from *static elaboration*:
  - marker traces to use to scale the subject (see Scaling). Might include additional traces of calculated points, such as joint centers (see *Static Elaboration: process your static trials*[7])
- from *dynamic elaboration*:
  - marker traces for the trial (see Inverse Kinematics)
  - external loads for the trial (see Inverse Dynamics)
  - normalized EMG envelopes (see MOtoNMS manual's *Data Processing: elaborate your dynamic trials*[8] page), that can be used as *muscle excitations* (see Input Data)

4.3.2 Data organization

MOtoNMS requires that input C3D files are located inside an `InputData` directory, or any of its subdirectory; processed data will be saved into an `ElaboratedData` directory, keeping the same folder organization, as explained in the MOtoNMS manual *Data Organization*[9] page.

We suggest to enforce the same data organization for dealing with OpenSim and CEINMS processing: inside the `ElaboratedData`, along with `sessionData`, `staticElaborations`, and `dynamicElaborations`, we suggest you create a folder for each of the processing steps involved (e.g., Inverse Kinematics, CEINMS). This allows to automatize the most time-consuming tasks (i.e., preparing configuration files and pre-processing data) as will be discussed in Batch processing scripts.

4.4 Exploiting OpenSim

In this section, a brief explanation is given of the preprocessing steps through which it is possible to obtain the input files listed in the Input Data section. The "meaning" of each step is discussed, so that it should be easier to obtain the same quantities with any other musculoskeletal modeling tool you may want to use.

4.4.1 Scaling

In the absence of subject-specific neuromusculoskeletal models obtained, e.g., from imaging techiques (see [BAGD07] for an overview on the subject), it is common practice to generate subject-specific muscluloskeletal models starting from generic ones. This means that the anthropometry of a generic model is altered so that it matches a particular subject as closely as possible.

For additional information on scaling procedures, please refer to the OpenSim manual Scaling page[10].

[6] http://www.mathworks.com/products/matlab/
[7] http://rehabenggroup.github.io/MOtoNMS/manual/staticElaboration.html#staticelaboration
[8] http://rehabenggroup.github.io/MOtoNMS/manual/dataProcessing.html#dataprocessing
[9] http://rehabenggroup.github.io/MOtoNMS/manual/folders.html#dataorganization
[10] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Scaling

4.4.2 Inverse Kinematics

Starting from markers' positions, the values of the generalized coordinates (generally corresponding to joint angles) that describe the motion of body segments are estimated. This is needed both to determine muscle kinematics during the task, through Muscle Analysis, and to determine how the external loads are applied on the subject, which is needed for Inverse Dynamics.

For additional information on inverse kinematics in OpenSim, please refer to the OpenSim manual Inverse Kinematics page[11].

4.4.3 Inverse Dynamics

Inverse dynamics is aimed at estimating the net joint moments that are responsible for the given movement (i.e., the kinematics estimated through Inverse Kinematics), given the kinetics information recorded on the environment (i.e., ground reaction forces measured through force platforms, the *external loads* mentioned in MOtoNMS[12]). This step is actually optional: it provides the optional *joint moments* file (see Input Data) that can be used for calibration or for validation of the model.

For additional information on inverse dynamics in OpenSim, please refer to the OpenSim manual Inverse Dynamics page[13].

4.4.4 Muscle Analysis

Muscle analysis reports the state of the muscles during the execution of a given movement. We use it to obtain the lengths of muscle-tendon units, and the moment arms of each muscle with respect to the degrees of freedom it acts upon.

Muscle analysis can be performed using OpenSim's Analyze Tool[14]. The muscle-tendon unit lengths are written to a `<prefix>_Length.sto` file, while moment arms file names follow this convention: `<prefix>_MomentArm_<dof_name>.sto`. Since OpenSim documentation on this particular analysis is scarce, we include a template setup file in the following. Fields that must be filled in are marked by text within ** symbols. Most important fields within the `MuscleAnalysis` block are:

- `compute_moments` that must be set to true (default is false)
- `muscle_list` list of muscles for which perform the analysis: you can specify a subset of muscles instead of `all` to reduce computation time and the number of output files, but be sure to include all muscles that belong to your CEINMS model
- `moment_arm_coordinate_list` list of degrees of freedom for which to compute the moment arms: as with the muscle list, you can specify a subset of degrees of freedom instead of all to reduce computation time and the number of ouput files, but be sure to include all the degrees of freedom of your CEINMS model.

```
<?xml version="1.0" encoding="UTF-8" ?>
<OpenSimDocument Version="30000">
    <AnalyzeTool name="MuscleAnalysisTool">
        <!--Name of the .osim file used to construct a model.-->
        <model_file> ** MODEL FILE **</model_file>
        <!--Replace the model's force set with sets specified in <force_set_files>?
            If false, the force set is appended to.-->
        <replace_force_set>false</replace_force_set>
        <!--List of xml files used to construct an force set for the model.-->
```

[11] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Inverse+Kinematics
[12] https://simtk.org/home/motonms
[13] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Inverse+Dynamics
[14] http://simtk-confluence.stanford.edu:8080/display/OpenSim/Analyses

```
<force_set_files />
<!--Directory used for writing results.-->
<results_directory>.</results_directory>
<!--Output precision.  It is 8 by default.-->
<output_precision>8</output_precision>
<!--Initial time for the simulation.-->
<initial_time> ** INITIAL TIME ** </initial_time>
<!--Final time for the simulation.-->
<final_time> ** FINAL TIME ** </final_time>
<!--Flag indicating whether or not to compute equilibrium values for states
    other than the coordinates or speeds.  For example, equilibrium muscle
    fiber lengths or muscle forces.-->
<solve_for_equilibrium_for_auxiliary_states>
    false
</solve_for_equilibrium_for_auxiliary_states>
<!--Maximum number of integrator steps.-->
<maximum_number_of_integrator_steps>20000</maximum_number_of_integrator_steps>
<!--Maximum integration step size.-->
<maximum_integrator_step_size>1</maximum_integrator_step_size>
<!--Minimum integration step size.-->
<minimum_integrator_step_size>1e-008</minimum_integrator_step_size>
<!--Integrator error tolerance. When the error is greater,
    the integrator step size is decreased.-->
<integrator_error_tolerance>1e-005</integrator_error_tolerance>
<!--Set of analyses to be run during the investigation.-->
<AnalysisSet name="Analyses">
    <objects>
        <MuscleAnalysis name="MuscleAnalysis">
            <!--Flag (true or false) specifying whether whether on.
                True by default.-->
            <on>true</on>
            <!--Start time.-->
            <start_time> ** INITIAL TIME, AS ABOVE ** </start_time>
            <!--End time.-->
            <end_time> ** FINAL TIME, AS ABOVE ** </end_time>
            <!--Specifies how often to store results during a simulation.
                More specifically, the interval (a positive integer) specifies
                how many successful integration steps should be taken before
                results are recorded again.-->
            <step_interval>1</step_interval>
            <!--Flag (true or false) indicating whether the results are
                in degrees or not.-->
            <in_degrees>true</in_degrees>
            <!--List of muscles for which to perform the analysis. Use 'all' to
                perform the analysis for all muscles.-->
            <muscle_list> all</muscle_list>
            <!--List of generalized coordinates for which to compute moment arms.
                Use 'all' to compute for all coordinates.-->
            <moment_arm_coordinate_list> all</moment_arm_coordinate_list>
            <!--Flag indicating whether moments should be computed.-->
            <compute_moments>true</compute_moments>
        </MuscleAnalysis>
    </objects>
    <groups />
</AnalysisSet>
<!--Controller objects in the model.-->
<ControllerSet name="Controllers">
    <objects />
```

```
            <groups />
        </ControllerSet>
        <!--XML file (.xml) containing the forces applied to the model as ExternalLoads.-->
        <external_loads_file> ** EXTERNAL LOADS FILE (optional) ** </external_loads_file>
        <!--Storage file (.sto) containing the time history of states for the model.-->
        <states_file />
        <!--Motion file (.mot) or storage file (.sto) containing the time history of the
            generalized coordinates for the model. These can be specified in place of
            the states file.-->
        <coordinates_file> ** OUTPUT FILE FROM INVERSE KINEMATICS** </coordinates_file>
        <!--Storage file (.sto) containing the time history of the generalized speeds
            for the model. If coordinates_file is used in place of states_file, these can be
            optionally set as well to give the speeds. If not specified, speeds will be
            computed from coordinates by differentiation.-->
        <speeds_file />
        <!--Low-pass cut-off frequency for filtering the coordinates_file data
            (currently does not apply to states_file or speeds_file). A negative value
            results in no filtering. The default value is -1.0, so no filtering.-->
        <lowpass_cutoff_frequency_for_coordinates>
            -1
        </lowpass_cutoff_frequency_for_coordinates>
    </AnalyzeTool>
</OpenSimDocument>
```

4.4.5 Batch processing scripts

If you have many experimental trials to process, we warmly suggest that you automatize this pre-processing pipeline (from inverse kinematics onward). A simple way is to use scripts that customize template setup files with the actual paths of data files to use. This is most easy when you enforce a clear data organization scheme, as suggested in section Data organization.

Examples of such scripts for inverse kinematics and inverse dynamics are available at https://github.com/RehabEngGroup/OpenSimProcessingScripts.

Important: Performing these steps automatically does not ensure that the results are correct. You should **always** check the results and, if they do not look plausible, check that that the template setup files are correct for your type of data/experiments, and that the setup files are generated correctly.

You may also want to automatize the creation of *trial (input) data description files*.

CHAPTER

FIVE

USING CEINMSCALIBRATE

The CEINMScalibrate executable implements the subject calibration procedure presented in *Calibration*. It is aimed at refining muscle parameters, that determine how muscles generate force, using an optimization algorithm that minimizes a cost function. Currently, the cost function that is implemented is the error between the estimated and the measured joint moments during a set of tasks.

To summarize, before running the CEINMScalibrate software, you should:

- prepare your experimental data, as explained in *Preparing your experimental data*. Remember that you need the *joint moments* files for the trials that you want to use for calibration.
- prepare your setup files: the *main CEINMScalibrate setup file* and the files it includes.

Most of the setup files are related to the model definition and the interpretation of input data; please refer to the related sections in *CEINMS configuration files* for complete information. The file that is most critical and needs further explanations is the *Calibration configuration file*. Indeed, modifying this file might yield to very different results in terms of calibrated parameters: extra care must be taken when defining calibration properties, as they depend both on the subject description and the application you are interested in.

Parameters of the optimization algorithm can be modified to reduce computation time (e.g., for *simulated annealing*, increasing the value for `epsilon` or `rT`, or reducing `noEpsilon`, `NT` or `NS`), but convergence to the global minimum becomes less likely.

As for parameter ranges specification, one must always keep in mind that increasing the number of variables in an optimization problem yields to increased computational costs and less reliable solutions (the so-called *curse of dimensionality*). Therefore, while it is technically possible to calibrate all muscular parameters that describe the subject, one should give some consideration to which parameters can be estimated (and how reliably) with other techniques (i.e., scaling generic models appropriately, with imaging techniques, and so on) and which are to be adjusted through the calibration process.

5.1 Executing the software

Once you have installed the software (see *CEINMS installation instructions*) and prepared the above-mentioned setup and data files, you can run the software by typing in your terminal (or command prompt in Windows)

```
CEINMScalibrate -S <path to main XML setup file>
```

If setup and data files can be correctly found, calibration will start after the number of total parameters is printed to screen. Calibration time may vary depending on the number of parameters you are calibrating and the computing power of your computer, so it may take several minutes before any information is printed to screen.

Important: Please make sure that the output subject file specified in the *main setup XML file* is to be written into a folder that already exists.

5.2 Output

When software execution finishes, an XML description of the calibrated subject will be written to the file that was specified as `outputSubjectFile` in the *main setup XML file*. We encourage you to check that calibrated parameters are still consistent with any information you have on the subject's physiology. In particular, we suggest that you check if any parameter hit the boundary of the calibration range, which may suggest that either the boundaries are incorrect, or that the experimental data you provided was not consistent with the model.

CHAPTER

SIX

USING CEINMS

The CEINMS executable allows to run neuromusculoskeletal simulations in different operation modes, as was discussed in the *introduction*, depending on the content of the `NMSmodel` element in *Execution configuration file*.

Within this element, the `openLoop` type means that the execution will run in *full-predictive open-loop*, therefore no joint moments file (see *Input data*) is required for the trial that is being processed. The `hybrid` execution modality is more complex, as it allows to select the objective function weightings alpha, beta and gamma, the muscles considered in the optimization, and to select the optimization algorithm. Depending on the choice of `adjustMTUs`, `synthMTUs` and weightings, we label the algorithm differently (see also *Hybrid mode*):

Hybrid mode

alpha = 1, beta > 0, gamma = 0

synthMTUs = every muscle without experimental excitations

adjustMTUs = none

EMG-assisted mode

alpha = 1, beta > 1, gamma > 1

synthMTUs = every muscle without experimental excitations

adjustMTUs = every muscle with experimental excitations

Static optimisation mode

alpha = 1, beta > 0, gamma = 0

synthMTUs = every muscle

adjustMTUs = none

As for the `tendon` element, the `equilibriumElastic` option usually provides more accurate results, while using a `stiff` tendon reduces computation time.

6.1 Executing the software

Once you have installed the software (see *CEINMS installation instructions*) and prepared the above-mentioned setup and data files, you can run the software by typing in your terminal (or command prompt in Windows)

```
CEINMS -S <path to main XML setup file>
```

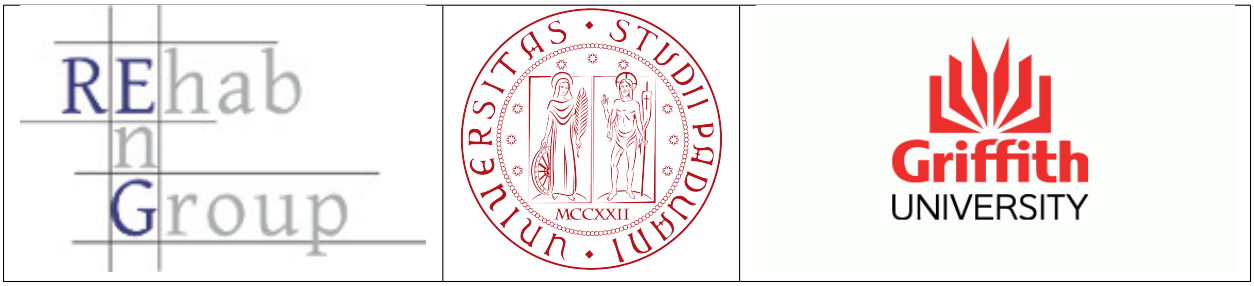
If setup and data files can be correctly found, the execution will run and some information on current results will be reported to screen. Upon completion, you will find, inside the output folder you specified in the setup file, a set of storage files containing all the quantities that are calculated during the simulation of each muscle's behaviour, such as activation, lenght and contraction velocity of the fibres, and ultimately muscle forces. Furthermore, joint moments

are computed, that can be compared to experimental ones. While it is generally difficult to compare single muscles' quantities against experimental data, these files are useful to get a better insight at how each muscle is behaving, and to verify that there are no errors or artifacts in the input data or in the model parameter.

Part III

Acknowledgements

CEINMS is being developed by the Rehabilitation Engineering Group[1], Dipartimento di Tecnica e Gestione dei Sistemi Industriali[2], Universita' degli Studi di Padova[3], and the Centre For Musculoskeletal Research[4], Griffith University[5].



Monica Reggiani and Elena Ceseracciu acknowledge the financial support of the ICT programme within the Seventh Framework Programme for Research of the European Commission (BioMot[6] project, grant number: 611695).

[1] http://reg.gest.unipd.it
[2] http://www.gest.unipd.it
[3] http://www.unipd.it
[4] http://www.griffith.edu.au/health/musculoskeletal-research
[5] http://www.griffith.edu.au/
[6] http://www.biomotproject.eu/

[BLMB04] Thomas S Buchanan, David G Lloyd, Kurt Manal, and Thor F Besier. Neuromusculoskeletal modeling: estimation of muscle forces and joint moments and movements from measurements of neural command. *Journal of applied biomechanics*, 20(4):367, 2004.

[CMMR87] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "simulated annealing algorithm". *ACM Transactions on Mathematical Software*, pages 262–280, 1987.

[EMHvdB07] Ahmet Erdemir, Scott McLean, Walter Herzog, and Antonie J van den Bogert. Model-based estimation of muscle forces exerted during movements. *Clinical Biomechanics*, 22(2):131–154, 2007.

[GSB+13] Pauline Gerus, Massimo Sartori, Thor F Besier, Benjamin J Fregly, Scott L Delp, Scott A Banks, Marcus G Pandy, Darryl D D'Lima, and David G Lloyd. Subject-specific knee joint geometry improves predictions of medial tibiofemoral contact forces. *Journal of biomechanics*, 46(16):2778–2786, 2013.

[GFR94a] William L Goffe, Gary D Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, 60(1):65–99, 1994.

[GFR94b] William L. Goffe, Gary D. Ferrier, and John Rogers. Global optimization of statistical functions with simulated annealing. *Journal of Econometrics*, pages 65–99, 1994.

[Hui95] PA Huijing. Important experimental factors for skeletal muscle modelling: non-linear changes of muscle length force characteristics as a function of degree of activity. *European journal of morphology*, 34(1):47–54, 1995.

[LB03] David G Lloyd and Thor F Besier. An emg-driven musculoskeletal model to estimate muscle forces and knee joint moments in vivo. *Journal of biomechanics*, 36(6):765–776, 2003.

[LBWB08] David G Lloyd, Thor F Besier, Christopher R Winby, and Thomas S Buchanan. Neuromusculoskeletal modelling and simulation of tissue load in the lower extremities. *Handbook of Biomechanics and Human Movement Science. New York: Routledge*, pages 3–17, 2008.

[LB01] David G Lloyd and Thomas S Buchanan. Strategies of muscular support of varus and valgus isometric loads at the human knee. *Journal of biomechanics*, 34(10):1257–1267, 2001.

[LB96] DG Lloyd and TS Buchanan. A model of load sharing between muscles and soft tissues at the human knee during static tasks. *Journal of biomechanical engineering*, 118(3):367–376, 1996.

[MB03] Kurt Manal and Thomas S Buchanan. A one-parameter neural activation to muscle activation model: estimating isometric joint moments from electromyograms. *Journal of biomechanics*, 36(8):1197–1202, 2003.

[McG92] Stuart M. McGill. A myoelectrically based dynamic three-dimensional model to predict loads on lumbar spine tissues during lateral bending. *Journal of Biomechanics*, 25(4):395 – 414, 1992.

[MBSY73] HS Milner-Brown, Richard B Stein, and R Yemm. Changes in firing rate of human motor units during linearly changing voluntary contractions. *The Journal of physiology*, 230(2):371, 1973.

[SFL13] Massimo Sartori, Dario Farina, and David G Lloyd. Hybrid neuromusculoskeletal modeling. In *Converging Clinical and Engineering Research on Neurorehabilitation*, pages 427–430. Springer, 2013.

[SLRP10] Massimo Sartori, David G Lloyd, Monica Reggiani, and Enrico Pagello. Fast operation of anatomical and stiff tendon neuromuscular models in emg-driven modeling. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2228–2234. IEEE, 2010.

[SRFL12] Massimo Sartori, Monica Reggiani, Dario Farina, and David G Lloyd. Emg-driven forward-dynamic estimation of muscle force and joint moment about multiple degrees of freedom in the human lower extremity. *PloS one*, 7(12):e52618, 2012.

[SRPL12] Massimo Sartori, Monica Reggiani, Enrico Pagello, and David G Lloyd. Modeling the human knee for assistive technologies. *Biomedical Engineering, IEEE Transactions on*, 59(9):2642–2649, 2012.

[Sch93] Lisa Margaret Schutte. *Using musculoskeletal models to explore strategies for improving performance in electrical stimulation-induced leg cycle ergometry*. PhD thesis, Stanford University, 1993.

[TBB97] Dimitrios Tsirakos, Vasilios Baltzopoulos, and Roger Bartlett. Inverse optimization: functional and physiological considerations related to the force-sharing problem. *Critical Reviews\u 2122 in Biomedical Engineering*, 1997.

[WLBK09] Christopher R Winby, David Gavin Lloyd, Thor F Besier, and T Brett Kirk. Muscle and external load contribution to knee joint contact loads during normal gait. *Journal of Biomechanics*, 42(14):2294–2300, 2009.

[WGKL13] CR Winby, Pauline Gerus, TB Kirk, and David Gavin Lloyd. Correlation between emg-based co-activation measures and medial and lateral compartment loads of the knee during gait. *Clinical Biomechanics*, 28(9):1014–1019, 2013.

[Zaj88] Felix E Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, 17(4):359–411, 1988.

[BAGD07] Silvia S. Blemker, Deanna S. Asakawa, Garry E. Gold, and Scott L. Delp. Image-based musculoskeletal modeling: applications, advances, and future opportunities. *Journal of Magnetic Resonance Imaging*, 25(2):441–451, 2007.