# Model calibration specifications ABI - OKS003

## 1. Metadata

Authors:
Thor Besier
Marco Schneider
Nynke Rooks

Date: 2/10/2019

Contact: t.besier@auckland.ac.nz

## 2. Summary of data utilized with the M&S processes

Calibration data needed:
~ 0 degrees knee flexion: Varus - valgus rotation moment - rotation data
~ 90 degrees knee flexion: Internal - external rotation  moment - rotation data
~ 90 degrees knee flexion: Anterior - posterior translation force - displacement data

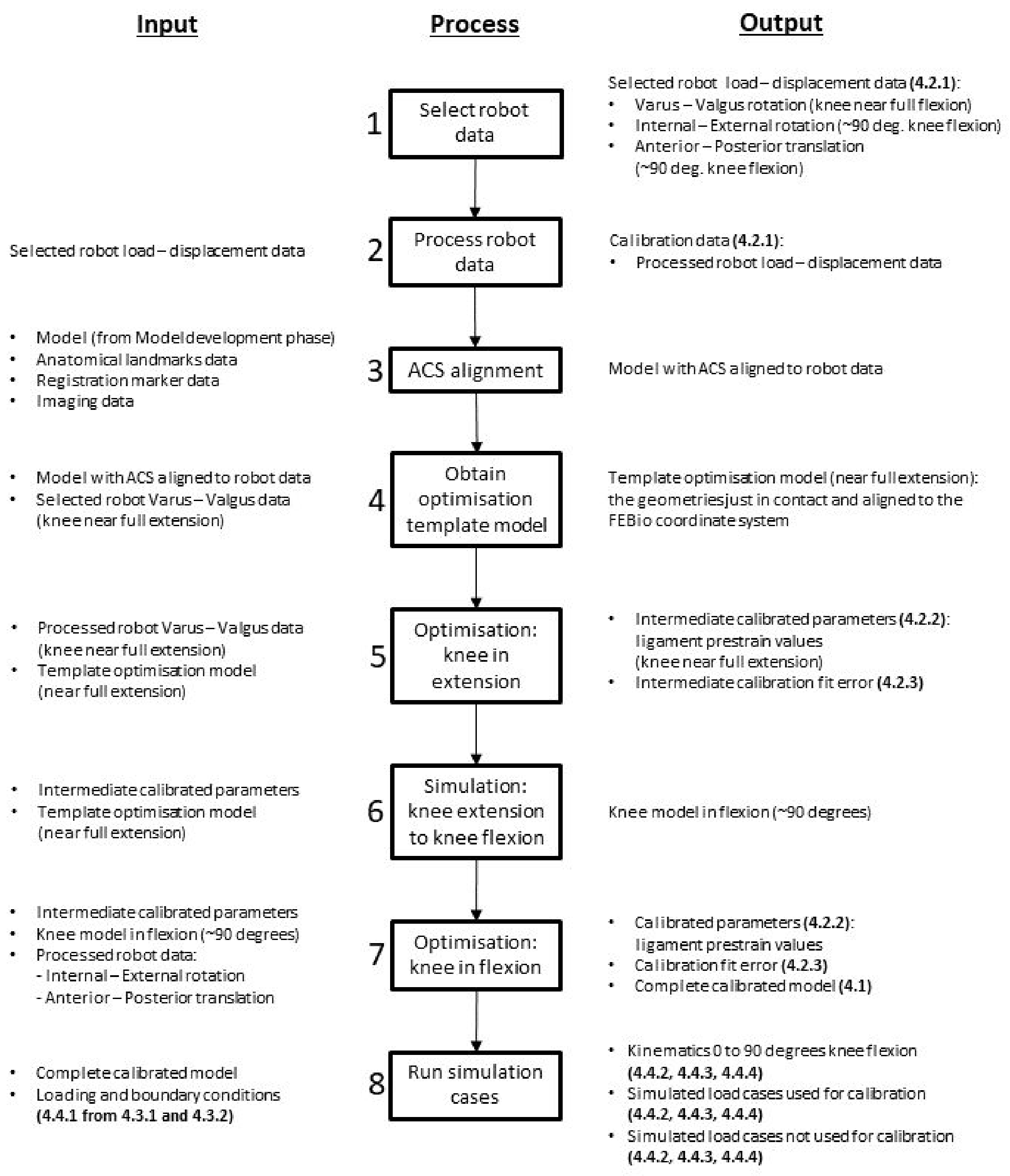
*Selected OKS003 robot data*
Load - displacement data:
File: 006_All Laxity 0deg_main_processed.tdms
    State.Knee JCS: Knee JCS Valgus
    State.JCS Load: JCS Load Varus Torque
File: 014_AllLaxity 90deg_main_processed.tdms
    State.Knee JCS: Knee JCS Internal Rotation
    State.JCS Load: JCS Load External Rotation Torque
    State.Knee JCS: Knee JCS Posterior
    State.JCS Load: JCS Load Anterior Drawer
File: state.cfg
    Knee JCS: Position offset

Anatomical landmarks data:
File: state.cfg
    Collected points rigid body 1
    Collected points rigid body 2

Registration marker data:
File: state.cfg
    MRI Fiducial Sphere positions

Imaging data:
General purpose imaging (MRI) - sagittal

## 3. Overview of target M&S outputs and M&S processes


Input
Process
Output
1 Select robot data
Selected robot load – displacement data (4.2.1):
• Varus – Valgus rotation (knee near full flexion)
• Internal – External rotation (~90 deg. knee flexion)
• Anterior – Posterior translation (~90 deg. knee flexion)
Selected robot load – displacement data
2 Process robot data
Calibration data (4.2.1):
• Processed robot load – displacement data
• Model (from Model development phase)
• Anatomical landmarks data
• Registration marker data
• Imaging data
3 ACS alignment
Model with ACS aligned to robot data
• Model with ACS aligned to robot data
• Selected robot Varus – Valgus data (knee near full extension)
4 Obtain optimisation template model
Template optimisation model (near full extension): the geometries just in contact and aligned to the FEBio coordinate system
• Processed robot Varus – Valgus data (knee near full extension)
• Template optimisation model (near full extension)
5 Optimisation: knee in extension
• Intermediate calibrated parameters (4.2.2): ligament prestrain values (knee near full extension)
• Intermediate calibration fit error (4.2.3)
• Intermediate calibrated parameters
• Template optimisation model (near full extension)
6 Simulation: knee extension to knee flexion
Knee model in flexion (~90 degrees)
• Intermediate calibrated parameters
• Knee model in flexion (~90 degrees)
• Processed robot data:
- Internal – External rotation
- Anterior – Posterior translation
7 Optimisation: knee in flexion
• Calibrated parameters (4.2.2): ligament prestrain values
• Calibration fit error (4.2.3)
• Complete calibrated model (4.1)
• Complete calibrated model
• Loading and boundary conditions (4.4.1 from 4.3.1 and 4.3.2)
8 Run simulation cases
• Kinematics 0 to 90 degrees knee flexion (4.4.2, 4.4.3, 4.4.4)
• Simulated load cases used for calibration (4.4.2, 4.4.3, 4.4.4)
• Simulated load cases not used for calibration (4.4.2, 4.4.3, 4.4.4)

## 4. Detailed description target M&S outputs

### *4.1 Complete calibrated model*

The model will be based on the model built in the Model Development phase, for more information on how this model was built, we refer to the ABI Model Development specifications and protocol deviations documents. The complete calibrated model will be provided as *.feb file together with the corresponding *.log and *.xplt file.

#### 4.1.1 Anatomy

The model will consist of the tibiofemoral joint. The patella will also be present in the model, but is not taken into account in the calibration. It will be displaced 10mm in the anterior direction to make sure it does not interact with the other structures in the model. The tibiofemoral joint structures present in the complete calibrated model will be the femur and tibia/fibula bones and cartilages, the medial and lateral meniscus (mMen & lMen), the anterior cruciate ligament (ACL), the posterior cruciate ligament (PCL), the medial collateral ligament (MCL) and the lateral collateral ligament (LCL).

In the model, the femur and tibia/fibula bones consist of a triangulated mesh and the cartilages are represented using hexahedral elements. Both the ligaments and the menisci are hexahedral meshes, where the menisci are attached to the tibia using linear springs.

The anatomy of the individual components of the calibrated knee model will be provided as meshes in *.feb, *.prv and *.vtk format.

#### 4.1.2 Mechanical properties

The mechanical properties of each structure in the model are dependent on their material properties. The bones will be represented as rigid bodies. The cartilages will also be represented as rigid bodies, since this decreases the computational time of the model and will be sufficient if the kinematics of the joint are of primary interest. In future models where cartilage stresses and strains will need to be found, the cartilage model will be changed to an isotropic elastic material model.

The ligaments will be modelled using a prestrain elastic transverse isotropic Mooney-Rivlin material model. Using the prestrain stretch factor (Maas et al., 2016), the initial stretch in the ligament can be prescribed. All ligaments have the same material properties, but the prestrain stretch factor will be different for each ligament. The menisci will be modelled using a Fung orthotropic material model. Please refer to the ABI Model development M&S outputs and documents for more detailed information.

The mechanical properties and the material models used, and the calibrated prestrain values, will be provided in the complete calibrated model *.feb file and will also be described separately in *.txt files.

#### 4.1.3 Coordinate systems

The anatomical coordinate system will be based on the Grood and Suntay coordinate system (Grood and Suntay, 1983). The anatomical coordinate system of the model will be matched to the anatomical coordinate system of the robot data. The mesh nodes representing anatomical landmarks for the local coordinate systems per bone will be based on the anatomical landmarks used in the robot data. The FEBio coordinate system (Figure 1) will be matched to the femur local anatomical coordinate system defined by the Grood and Suntay coordinate system.

The calculations of the anatomical coordinate system and the tibiofemoral kinematics, executed in Python 3.6, will be provided in a *.txt file. All intermediate outputs, used to match the anatomical coordinate system of the model to that of the robot data, will also be provided.

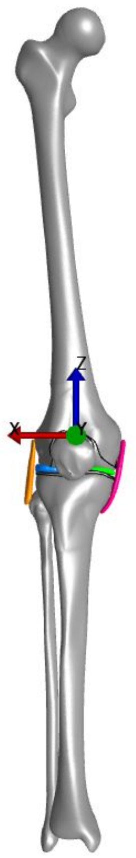


*Figure 1: FEBio coordinate system.*

### 4.1.4 Interactions

Since the cartilages will be  modelled as the same rigid body material as their corresponding bone, no separate contact implementation between the bones and cartilages is needed. In future models where the cartilage is modelled as an isotropic elastic material, a rigid tied interface between the bone and cartilage contacting surfaces will be required.

A sliding elastic contact will be used for the tibia cartilage - femur cartilage contact and also for all cartilage - menisci contacts. A facet-on-facet sliding contact between the MCL and the parts of the cartilage and bone which could be in contact with the middle part of the MCL will be applied, to ensure that the MCL does not penetrate the bones and cartilages. To attach the ligaments to the bones, a rigid tied interface between the attachment nodes of the ligaments and the bone rigid bodies will be applied.

A rigid cylindrical joint between the femur and the tibia, using two imaginary rigid bodies will be used to prescribe the flexion - extension rotation in the knee joint (Erdemir and Sibole, 2010). This rigid cylindrical joint will be used to be able to prescribe one rotational degree of freedom (DoF) without prescribing or fixing the other 2 rotational DoFs (a current constraint of FEBio).

The interactions will be provided in the complete calibrated model *.feb file and will also be described separately in *.txt files.

#### 4.1.5 Loading and boundary conditions

To simulate 0 to 90 degrees of knee flexion, a rotation will be prescribed in the rigid cylindrical joint. The femur will be free in all DoFs where the tibia will be fixed in all DoFs. An axial load of -20N will be applied to the femur to ensure cartilage - cartilage contact. For all other simulations, the rotations and displacements will be prescribed to the rigid bodies instead of in the rigid cylindrical joint. This is because FEBio is currently unable to resolve the rigid body forces when motion is applied via the rigid cylindrical joint.

The DoFs that are fixed and prescribed vary depending on the simulation case.

For all simulations, the *.feb file and the corresponding *.log and *.xplt file will be provided. Also an overview (*.txt file) of the settings per simulation will be provided.

### *4.2 Intermediate and final calibration outcomes*

#### 4.2.1 Calibration data

The selected raw data, the time synchronized raw data (if time synchronization is needed) and the polynomial fitted data will be provided in *.txt files.

Three data sets will be used for calibration:
~ 0 degrees of knee flexion: Varus - valgus rotation moment - rotation data
~ 90 degrees of knee flexion: Internal - external rotation moment - rotation data
~ 90 degrees of knee flexion: Anterior - posterior translation force - displacement data

#### 4.2.2 Calibrated parameters

The pre-strain values of the ACL, PCL, MCL and LCL will be optimised during the calibration phase. The optimised pre-strain values after each optimisation round (~0 and ~90 degrees of knee flexion) will be provided in a *.txt file.

#### 4.2.3 Calibration fit error

The root mean square error (RMSE) for each timestep (4 timesteps in total) between the force/moment in the robot data and the force/moment found in the model for a displacement/rotation simulated in the model will be calculated. This RMSE will then be normalized to the maximum force/moment in the corresponding processed robot data. This normalized RMSE error is the value to be minimized during the optimisation of the ligament prestrain stretch values.

The final RMSE and the normalized RMSE will be provided for each simulation in each optimisation round (~0 and ~90 degrees of knee flexion) in a *.txt file.

#### 4.2.4 Changed model components

During the calibration phase, only the prestrain stretch values of the ligaments will be changed and provided as described in 4.2.2. No geometries are expected to be changed.

### *4.3 Intermediate and final outcomes of analysis of experimental load cases*

#### 4.3.1 Source data (as extracted from earmarked data set)

The raw data that will be used for the simulation of the experimental load cases will be provided in a *.txt file per simulation.

#### 4.3.2 Processed data (as analyzed to make ready for use in simulations)

The time synchronised (if time synchronisation is needed) and polynomial fitted data will be provided in a *.txt file per simulation. The load curves used for the prescribed displacement/rotation will be provided in a *.txt file as well.

### *4.4 Simulation cases*

#### 4.4.1 Loading and boundary conditions

The loading and boundary conditions for all simulations will be provided in *.txt files and next to that the *.feb and the corresponding *.log and *.xplt files will be provided per simulation.

The simulations will be divided into three groups:
1. 0 to 90 degrees of knee flexion simulation
2. Load cases used for calibration
3. Load cases not used for calibration

#### 4.4.2 Target metrics for predictions

The RMSE and normalized RMSE between the forces in the robot and in the model simulation results will be the target metrics for the predictions.

For all simulations, the RMSE and normalized RMSE (average over the simulation and per time point) will be provided in *.txt files.

#### 4.4.3 Numerical analysis settings

Static analyses will be used for the simulations. The settings can be found in the *.feb files per simulation and will be given per simulation in *.txt files.

#### 4.4.4 Anticipated results

The tibiofemoral kinematics (6 DoF) during the 0 to 90 degrees knee flexion simulation will be obtained and provided in *.png format.

For all other simulations the tibiofemoral kinematics can also be calculated and plotted in *.png. The RMSE and the normalized RMSE between the robot force and the simulated force in the model will also be reported. These results will be given in *.txt files. The simulation results will be available in the *.log and *.xplt files per simulation.

## 5. Detailed descriptions of M&S processes

### *5.1 Steps to calibrate the models*

#### 5.1.1 Select robot data

For our modelling workflow, we aim to keep the amount of data needed to a minimum with the goal of clinical application in mind. Therefore, we choose to only optimise the knee model in extension (~0 degrees knee flexion angle) and in about 90 degrees of flexion.

Looking at what data might be clinically obtainable, we choose to use varus-valgus data with the knee in extension, and internal - external rotation and anterior - posterior translation data with the knee in about 90 degrees of flexion.

The data that matches the 0 and 90 degrees knee flexion angles closest will be selected for use in calibration. The model will be put into this knee flexion angle to match the robot knee flexion angle during optimisation.

The data representing the tibiofemoral joint kinematics based on the motion capture sensors will be used (State.Knee JCS). This data is chosen over the data representing the tibiofemoral joint kinematics based on the robot position (State.JCS), since in the simVitro User Manual (SimVitro user manual) it is stated that this is a better representation of the actual joint position, however more noise could be present. The kinematics in the data are the relative kinematics, where the offset between the relative and the absolute kinematics can be found in the state.cfg document (Knee JCS: Position offset).

For the kinetic data, the State.JCS Load data will be used, representing the tibial loads taking into account the load cell output and the relative position between the tibia and the load cell (SimVitro user manual). The State.JCS Load data includes:
- Lateral drawer (N)
- Anterior drawer (N)
- Distraction (N)
- Extension torque (Nm)
- Varus torque (Nm)
- External rotation torque (Nm)

#### 5.1.2 Process robot data

The time synchronisation of the robot data will be checked, and will be corrected if necessary by using clear start and end moments of related rotations/translations and torques/forces. The robot data points where the actual simulations take place will be selected. The processed and selected data will be polynomial fitted in python (numpy package: polyfit and poly1D) to smooth the calibration data and to be able to interpolate between the data points.

#### 5.1.3 Anatomical coordinate system alignment

The anatomical coordinate system of the model will be adjusted to match the anatomical coordinate system of the robot data, which is based on the Grood and Suntay coordinate system (Grood and Suntay, 1983). The local coordinate system per bone is defined by anatomical landmarks. The anatomical landmarks used in the model will be adjusted to match the anatomical landmarks used in the robot data.

Robot data anatomical landmarks of the tibia:
1. Lateral tibial plateau
2. Medial tibial plateau
3. Distal tibia point 1
4. Distal tibia point 2
5. Distal tibia point 3
6. Distal tibia point 4

Robot data anatomical landmarks of the femur:
1. Lateral femoral epicondyle
2. Medial femoral epicondyle
3. Femoral head point 1
4. Femoral head point 2
5. Femoral head point 3
6. Femoral head point 4

To find the location of the anatomical landmarks of the robot data on the model meshes, the model meshes will be registered to the robot data. This will be done using the registration markers (state.cfg: MRI Fiducial Sphere positions). The registration markers will be segmented from the MRI data (General purpose imaging - sagittal) and registered to the digitized locations of the registration markers in the robot data. The model mesh will be registered to the initial segmentation, after which the found transformation matrix will be applied to the mesh to register the mesh to the robot data. After this the robot data anatomical landmarks (state.cfg: Collected points rigid body 1 & 2) can be plotted on the model mesh and the mesh nodes corresponding to the anatomical landmarks will be selected.

The local bone coordinate systems and the joint coordinate system will be calculated as defined in the Knee Joint Coordinate System document (v 2.1) obtained from the Open Knee Simtk website (Knee joint Coordinate system document), which is based on the Grood and Suntay anatomical coordinate system (Grood and Suntay, 1983).

### 5.1.4 Obtaining the optimisation template model

The optimisation of the model will be performed in two parts, first, at approximately 0 degrees of knee flexion (knee in extension) and second, at approximately 90 degrees of knee flexion.

In the first optimisation the knee is near full extension (~0 degrees of knee flexion). The robot data to be used for optimisation is not obtained in exactly 0 degrees of knee flexion. The model will be put in the near full extension knee flexion angle in which the varus - valgus robot data is obtained (average knee flexion angle in which data used for optimisation was obtained), to mimic the robot data. The full model is then rotated to make sure the FEBio coordinate system matches the femur local coordinate system (according to the Grood and

Suntay coordinate system (Grood and Suntay, 1983)). This is done using a custom Python script.

After rotating the knee joint into the desired flexion angle, contact between the femur, tibia and menisci is ensured. In this procedure the femur is fixed in all DoFs and the tibia is free in the Z direction only. If the geometries are out of contact, a small force on the tibia will be applied pushing the tibia against the femur. This force will be an increasing force from 0 to 50N over 0 to 1 seconds with 0.1 second increments. All node coordinates at the moment when the geometries are just in contact will be saved. If there initially exists an overlap between the geometries, the model will be run with its contact definitions to remove overlap between the geometries. After running one timestep all node coordinates will be saved. The saved node coordinates will be written into a new *.feb file as the initial node coordinates using a custom Python script. The center of masses of both bones in the *.feb file will be set to the new calculated center of masses using the Grood and Suntay coordinate system (Grood and Suntay, 1983). This will be done using a custom Python script. The obtained *.feb file will be used as a template for the optimisation simulations.

### 5.1.5 Optimisation: knee in extension

The parameters to optimise are the prestrain stretch values of the four ligaments (ACL, PCL, MCL and LCL). Their initial values are 1.0. The optimisation is performed in Python using the basinhopping algorithm with the “Nelder-Mead” minimization method. In the objective function the prestrain stretch values will be optimised to minimize the difference between the forces or moments simulated and the robot forces or moments with a certain displacement or rotation.

With the knee near full extension, the modelled ligaments will be optimised to varus - valgus moment - rotation robot data. The displacement driven simulation will be performed using four time steps:
0.1: Prestrain in the ligaments activated
0.2: Varus rotation (max. varus rotation in the robot data)
0.3: Valgus rotation to return the knee back to a neutral position (the amount of rotation in the varus rotation in the previous time step)
0.4: Valgus rotation (max. valgus rotation in the robot data)

The varus and valgus rotation will be prescribed to the tibia as a prescribed rigid body motion. The femur will be fixed in flexion - extension and varus - valgus rotation (Rx and Ry) DoFs and the tibia will be fixed in all DoFs except for varus - valgus rotation, since this will be prescribed. An axial force (z direction) of -20N will be applied to the femur to ensure contact between the geometries.

*Note: In FEBio, you cannot mix prescribed and free rigid body rotational DoFs. For example, to prescribe a varus-valgus rotation in one rigid body the other rotational DoFs of this rigid body must be fixed or prescribed. To overcome this, we fix all other DoFs except for varus-valgus rotation (which is prescribed in this rigid body) and fix the varus-valgus DoF of the interacting rigid body.*

The *.feb file of the simulation will be called from the Python script in each iteration of the objective function. When FEBio is done running the model, the node coordinates needed to calculate the ACS and the forces and moments on the bones will be exported to the .log file and read by the Python script. The simulated varus - valgus angle is calculated for each timestep and the corresponding rigid body moment on the tibia is obtained. The fitted robot data is then used to calculate the difference in moment in the model and the moment in the robot for the corresponding rotation in the model. This is calculated for each timestep.

The error value to minimize is the normalized root mean square error (RMSE) value. If the simulation ran completely, the RMSE of all timesteps (4 timesteps in total) between the moment in the robot data and the moment found in the model for a rotation simulated in the model will be calculated. This RMSE will then be normalized to the maximum moment in the corresponding processed robot data.

A penalty system will be implemented to avoid error termination (equation 3). If the model terminates with an error, a higher penalty value will penalise the optimisation algorithm against the combination of prestrain values that caused the simulation to fail. In this system, the error value will be directly negatively proportional to the fractional completion of the simulation (equation 1), to penalise against values that crash the simulation early. To penalise against consecutive failed simulations, the penalty system also increments the error with each consecutive error termination (equation 2).

$$p1(t) = -50t + 20 \quad for\ 0 \leq t \leq 0.3 \tag{1}$$

Where *t* is the time-step at which the simulation has failed. Please note that normal termination occurs at *t* = 0.4.

$$p2(x) = p1(t)\,(1 + 0.1x) \tag{2}$$

Where *x* is the number of consecutive crashed iterations.

$$p(x,t) = (-50t + 20)(1 + 0.1x) \qquad for\ 0 \leq t \leq 0.3 \tag{3}$$

### 5.1.6 Simulation: knee in extension to knee in flexion

A simulation will be run to put the knee joint in the flexion angle of the ~ 90 degrees knee flexion robot data. The prestrain stretch values recovered from the knee in extension optimisation will be used in this simulation as initial values.

The FEBio (Maas et al., 2012) restart function will be used to maintain the geometry and attachments of the ligaments. In this way, the simulation from knee extension to knee flexion will only have to be run once, which decreases optimisation time for the knee in flexion. The optimisation in knee flexion can then be run from the last step of the knee extension to knee flexion simulation by restarting from where it completed in the previous step.

The knee extension to knee flexion simulation will be performed by prescribing the rotation to the femur in the rigid cylindrical joint. Where the femur is free in all DoFs but the flexion - extension DoF is prescribed. The tibia is fixed in all DoFs. An axial load of -20N will be applied to the femur to ensure cartilage - cartilage contact.

After the simulation, the whole model is rotated to match the femur local coordinate system to the FEBio coordinate system. All node coordinates are obtained and written into a new *.feb file as the initial node coordinates. The center of masses of both bones in the *.feb file will be set to the new calculated center of masses using the Grood and Suntay coordinate system. This will be done using a custom Python script. This *.feb file will be used as a template for the next optimisation simulations.

### 5.1.7 Optimisation: knee in flexion

This second optimisation with the knee in flexion will take place in a similar manner to the first optimisation where the knee is in extension. However, in this optimisation, the start prestrain stretch values will be the values found in the knee in extension optimisation. Also, during this optimisation, the change in prestrain stretch values during the optimisation will be bounded to ± 50% of the optimised prestrain value. E.g. If prestrain value is 1.06, then the bounds are ± 0.03.

With the knee in flexion two simulations will be performed to optimise the prestrain stretch values, anterior - posterior translation and internal - external rotation. Both simulations will be run using multi-threading in Python (_thread package).

The simulation of the anterior - posterior translation will have the following timesteps:
0.1: Prestrain in the ligaments activated
0.2: Anterior translation (max. anterior translation in the robot data)
0.3: Posterior translation to return the knee back to a neutral position (the amount of displacement in the anterior translation in the previous time step)
0.4: Posterior translation (max. posterior translation in the robot data)

The anterior - posterior displacement will be prescribed to the femur. The femur will only be fixed in the flexion - extension rotation DoF and prescribed in the anterior - posterior translation DoF. The tibia will be fixed in all DoFs. An axial force of -20N will be applied to the femur to ensure contact between the geometries.

The simulation of the internal - external rotation will have the following timesteps:
0.1: Prestrain in the ligaments activated
0.2: Internal rotation (max. internal rotation in the robot data)
0.3: External rotation to return the knee back to a neutral position (the amount of rotation in the internal rotation in the previous time step)
0.4: External rotation (max. external rotation in the robot data)

The internal - external rotations will be prescribed to the tibia. The femur will be fixed in the flexion - extension and internal - external rotation DoFs. The tibia will be fixed in all DoFs except for the internal - external rotation DoF, since this will be prescribed. An axial force of -20N will be applied to the femur to ensure contact between the geometries.

The normalized RMSE is calculated for both simulations as explained in 5.1.5. The mean normalized RMSE of both simulations is calculated, which is the value to be minimized in this optimisation.

### 5.1.8 Run simulation cases

The final optimised prestrain stretched values will be written into the template *.feb file used in the knee in extension optimisation. This template will be saved as the simulation template *.feb file.

Using this template, the following simulations will be performed:
- 0 to 90 degrees of knee flexion
- Load cases which were used for calibration
- Load cases which were not used for calibration

The steps to implement the load cases of the earmarked data as loading and boundary conditions and the steps to perform the simulations are described in 5.2 and 5.3.

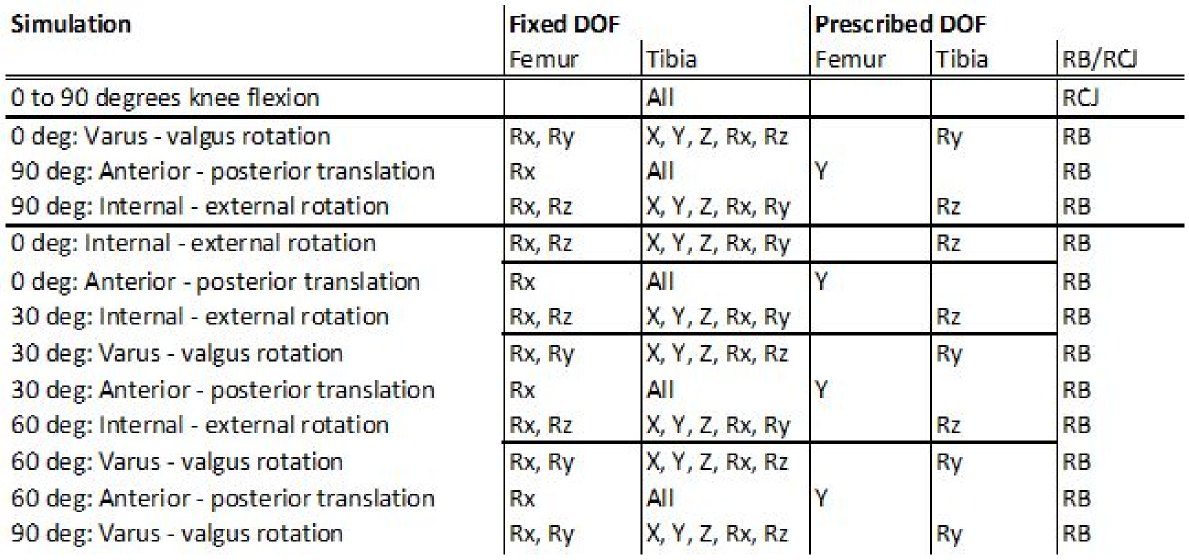
## *5.2 Steps to implement load cases of earmarked data as loading and boundary conditions & steps to perform simulations*

All simulations will be displacement driven. The displacements to be prescribed will be obtained from the robot data. An IE, VV rotation and an AP translation with the knee in about 0, 30, 60 and 90 degrees of knee flexion is executed in the robot. Each IE, VV rotation and AP translation will be simulated separately. The kinematic and kinetic data for the chosen rotation/translation will be selected (for example the IE part of the data in 0 degrees of knee flexion) and polynomial fitted to be smoothed (as done with the calibration data (5.1.2)). From this data the maximum and minimum translation/rotation will be selected in the robot data. These numbers will be put in the loadcurve of the prescribed translation/rotation as follows:
0.1: Prestrain in the ligaments activated
0.2: Minimum translation/rotation
0.3: Translation/rotation to return the knee back to a neutral position (the amount of translation/rotation in the previous time step)
0.4: Maximum translation/rotation

For the simulation of 0 to 90 degrees of passive knee flexion, the following loadcurve will be used:
0.1: Prestrain in the ligaments activated
1.0: 90 degrees of knee flexion

For all simulations the simulation template *.feb file will be used. The model will be simulated to about 0, 30, 60 and 90 degrees of knee flexion (where the same knee flexion angles will be used as in the robot data) using the restart function before running all IE, VV and AP rotation/translation simulations. The loading and boundary conditions will be manually put into this *.feb file and each *.feb file will be saved. The loading and boundary conditions per simulation are presented in table 1. Table 1 indicates if the translation/rotation is prescribed as a rigid body motion (RB) or as part of the rigid cylindrical joint (RCJ). For all simulations the auto-stepper function in FEBio will be turned on. The output of the simulations will be the kinematics and kinetics at each timepoint in the simulation, which can be compared to the robot data.

*Table 1: Loading and boundary conditions during the simulations (in FEBio coordinate system (Figure 1)).*



| Simulation | Fixed DOF | | Prescribed DOF | | |
|---|---|---|---|---|---|
| | Femur | Tibia | Femur | Tibia | RB/RCJ |
| 0 to 90 degrees knee flexion | | All | | | RCJ |
| 0 deg: Varus - valgus rotation | Rx, Ry | X, Y, Z, Rx, Rz | | Ry | RB |
| 90 deg: Anterior - posterior translation | Rx | All | Y | | RB |
| 90 deg: Internal - external rotation | Rx, Rz | X, Y, Z, Rx, Ry | | Rz | RB |
| 0 deg: Internal - external rotation | Rx, Rz | X, Y, Z, Rx, Ry | | Rz | RB |
| 0 deg: Anterior - posterior translation | Rx | All | Y | | RB |
| 30 deg: Internal - external rotation | Rx, Rz | X, Y, Z, Rx, Ry | | Rz | RB |
| 30 deg: Varus - valgus rotation | Rx, Ry | X, Y, Z, Rx, Rz | | Ry | RB |
| 30 deg: Anterior - posterior translation | Rx | All | Y | | RB |
| 60 deg: Internal - external rotation | Rx, Rz | X, Y, Z, Rx, Ry | | Rz | RB |
| 60 deg: Varus - valgus rotation | Rx, Ry | X, Y, Z, Rx, Rz | | Ry | RB |
| 60 deg: Anterior - posterior translation | Rx | All | Y | | RB |
| 90 deg: Varus - valgus rotation | Rx, Ry | X, Y, Z, Rx, Rz | | Ry | RB |

***5.3 Burden***

The software and hardware requirements, anticipated man hours and expertise level and computational cost per process are described below.

1. Select robot data
Software used: Microsoft office Excel
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 2 hours
Expertise level needed: Low/Medium
Computational cost: Low

2. Process robot data
Software used: Python 3.6
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 2 weeks
Expertise level needed: Medium/High
Computational cost: Low

3. Anatomical coordinate system alignment
Software used: Python 3.6 & Matlab R2017a Academic Licence
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 1 weeks

Expertise level needed: Medium
Computational cost: Low

4. Obtain optimisation template model
Software used: Python 3.6 & FEBio version 2.9.1 (with prestrain plug-in)
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 0.5 weeks
Expertise level needed: Medium
Computational cost: Low

5. Optimisation: knee in extension
Software used: Python 3.6 & FEBio version 2.9.1 (with prestrain plug-in)
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM & High performance computer (Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz 1024GB Ram)
Anticipated man hours: ~ 2 days (or less)
Expertise level needed: Low/Medium
Computational cost: High

6. Simulation: knee extension to knee flexion
Software used: Python 3.6 & FEBio version 2.9.1 (with restart function & prestrain plug-in)
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM & High performance computer (Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz 1024GB Ram)
Anticipated man hours: ~ 2-3 hours
Expertise level needed: Medium
Computational cost: High

7. Optimisation: knee in flexion
Software used: Python 3.6 & FEBio version 2.9.1 (with restart function & prestrain plug-in)
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM & High performance computer (Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz 1024GB Ram)
Anticipated man hours: ~ 2 days
Expertise level needed: Medium
Computational cost: High

8. Run simulation cases
Software used: Python 3.6 & FEBio version 2.9.1 (with restart function & prestrain plug-in)
Hardware used: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM & High performance computer (Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz 1024GB Ram)
Anticipated man hours: ~ 1 week
Expertise level needed: Medium
Computational cost: High

## 6. References

ABI Model development documents and M&S outputs

Erdemir, A., & Sibole, S. (2010). Open knee: a three-dimensional finite element representation of the knee joint. User's guide, version, 1(0).

Grood, E. S., & Suntay, W. J. (1983). A joint coordinate system for the clinical description of three-dimensional motions: application to the knee. Journal of biomechanical engineering, 105(2), 136-144.

Knee joint coordinate system.pdf V2.1. Retrieved from https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=view&target=Knee+Coordinate+Systems.pdf. Retrieved on 12 Sep 2019.

Maas, S.A., Ellis, B.J., Ateshian, G.A., Weiss, J.A. (2012). FEBio: Finite Elements for Biomechanics. Journal of Biomechanical Engineering, 134(1), 011005.

Maas, S.A., Erdemir, A., Halloran, J.P., Weiss, J.A. (2016). A general framework for application of prestrain to computational models of biological materials. Journal of the Mechanical Behavior of Biomedical Materials, 61, 499-510.

simVitro Data File Contents for Open Knee User Manual, Cleveland Clinic BioRobotics, Cleveland, OH. Retrieved from https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=view&target=2013CB-031-002.B+simVITRO+Data+File+Contents_Open+Knee.pdf. Retrieved on 12 Sep 2019.