Model Benchmarking Phase Specifications ABI - OKS003

1. Metadata

Authors:
Thor Besier
Nynke Rooks
Marco Schneider

Date: 26/02/2021

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
~ 60 degrees knee flexion: Internal - external rotation moment - rotation data
~ 60 degrees knee flexion: Anterior - posterior translation force - displacement data

Selected OKS003 robot data

Load - displacement data

Table 1. Selected Load - displacement data for model recalibration and benchmarking.

File	Variables
Laxity_0deg_VV1_kinematics_in_JCS_experiment.csv	Applied Load (Nm) Knee JCS Valgus [deg]
Laxity_0deg_VV2_kinematics_in_JCS_experiment.csv	Applied Load (Nm) Knee JCS Valgus [deg]
Laxity_60deg_EI1_kinematics_in_JCS_experiment.csv	Applied Load (Nm) Knee JCS Internal Rotation [deg]
Laxity_60deg_EI2_kinematics_in_JCS_experiment.csv	Applied Load (Nm) Knee JCS Internal Rotation [deg]
Laxity_60deg_AP1_kinematics_in_JCS_experiment.csv	Applied Load (N) Knee JCS Posterior [mm]
Laxity_60deg_AP2_kinematics_in_JCS_experiment.csv	Applied Load (N) Knee JCS Posterior [mm]
kinematic_offsets.csv	Knee JCS Medial [mm] Knee JCS Posterior [mm] Knee JCS Superior [mm] Knee JCS Flexion [deg] Knee JCS Valgus [deg] Knee JCS Internal Rotation [deg]

Combined_Kinematics.csv	Knee JCS Medial [mm] Knee JCS Posterior [mm] Knee JCS Superior [mm] Knee JCS Flexion [deg] Knee JCS Valgus [deg] Knee JCS Internal Rotation [deg]
Combined_Kinetics.csv	JCS Load Lateral Drawer [N] JCS Load Anterior Drawer [N] JCS Load Distraction [N] JCS Load Extension Torque [Nm] JCS Load Varus Torque [Nm] JCS Load External Rotation Torque [Nm]

Anatomical landmarks data

Files:

- Fem_AL.xyz
- Tib_AL.xyz

Registration marker data

Files:

- Fem_RM_lateral.xyz
- Fem_RM_medial.xyz
- Fem_RM_posterior.xyz
- Tib_RM_lateral.xyz
- Tib_RM_medial.xyz
- Tib_RM_posterior.xyz

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

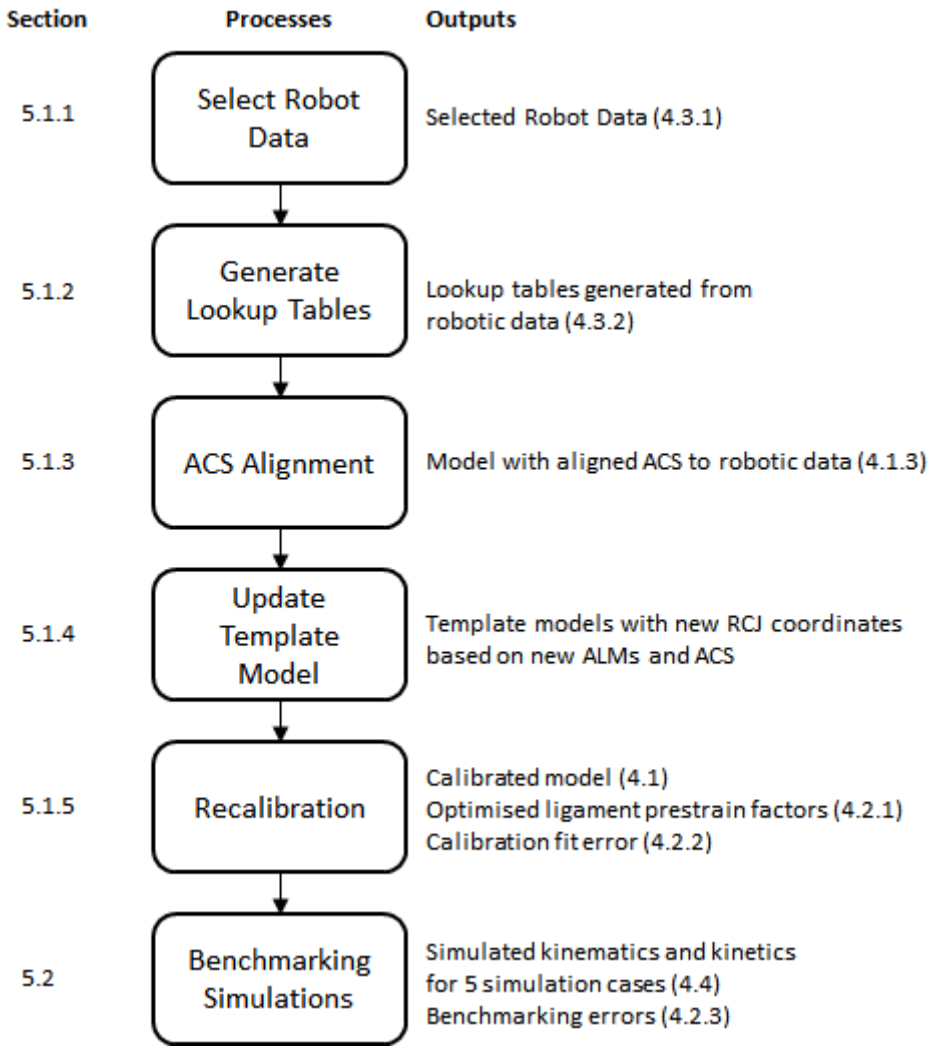

Figure 1. Overview of the model benchmarking processes and the target outputs. In summary, the robot data for recalibration and benchmarking will be selected and processed into lookup tables. Anatomical coordinate systems (ACS) will be calculated and implemented in the knee model after which the model will be calibrated against the robot data. Once calibrated, benchmarking simulations will be performed.

4. Detailed descriptions of target M&S outputs

4.1 Complete re-calibrated model and benchmarked model

To ensure consistency of Model Benchmarking (MB) with the Model Calibration (MC) phase, we will use the same geometries as described in section 4 of the ABI MC Protocol Deviations document. These geometries were updated during MC. For more information on how this model was built, please refer to the ABI MC phase protocol deviations document.

The complete re-calibrated model and benchmarking models will be provided as *.feb files together with their corresponding *.log and *.xplt files.

4.1.1 Anatomy

The model will consist of the tibiofemoral joint. The patella will also be present in the model as a rigid body, but will be fixed in all degrees of freedom and will not have any interactions with the other structures in the model.

The tibiofemoral joint structures present in the re-calibrated model will be the femur, tibia, fibula, femoral cartilage, medial tibial cartilage, lateral tibial cartilage, mMen, lMen, ACL, PCL, MCL, and LCL.

In the model, the bones (femur, tibia, fibula, patella) consist of triangulated surface meshes and the soft-tissues, including the femoral cartilage, tibial cartilages, patellar cartilage, the medial meniscus (mMen), lateral meniscus (lMen), medial collateral ligament (MCL), lateral collateral ligament (LCL), posterior cruciate ligament (PCL), and anterior cruciate ligament (ACL) consist of tetrahedral meshes. The menisci will be attached to the tibia with 36 linear springs each that have a stiffness of 1 N/mm.

The anatomy of the individual components of the re-calibrated knee model will be provided as meshes in *.feb, *.ele, *.node and *.fsprj format.

4.1.2 Mechanical properties

As in the MC phase, the bones and cartilages will be modelled as rigid bodies.

The ligaments will be modelled using a prestrain elastic Neo-Hookean material model.
Each ligament will be prescribed it's own Young's modulus value ({LCL, MCL, ACL, PCL}: [280, 224, 123, 168] Nmm^{-1}) based on the literature values (Orozco et al., 2018) and a prestrain stretch factor (Maas et al., 2016) which will be optimised individually. The menisci will be modelled with a Fung orthotropic material model.

The mechanical properties including the material models used, specific material model parameters, and the calibrated prestrain values, will be provided in the complete re-calibrated model and benchmarking model *.feb files. They will also be described separately in *.txt files.

4.1.3 Coordinate systems

The anatomical coordinate system (ACS) will be based on the Grood and Suntay coordinate system (Grood and Suntay, 1983) and will be implemented via the rigid cylindrical joint (RCJ) facility in FEBio:

- For the tibial ACS, the x-axis will point medially, the y-axis will point anteriorly, and the z-axis will point superiorly.
- For the femoral ACS, the x-axis will point medially, the y-axis will point anteriorly, and the z-axis will point superiorly.
- A separate RCJ will be used to implement the floating axis of the Grood & Suntay Knee Joint Coordinate System (JCS). FE points medially, VV points anteriorly, IE points superiorly

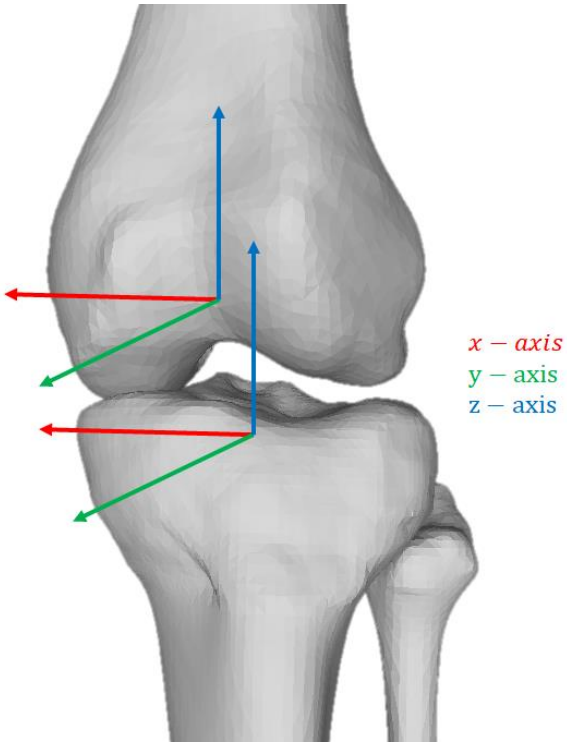

Figure 2. Local ACS directions for the femur and tibia.

All intermediate outputs, including the scripts used to find the anatomical landmarks (ALMs), *.txt files containing the ALM node numbers for the femur and tibia, and scripts for calculating and defining the ACS of the model for the RCJ will be provided.

Two ALM node number files will be provided, one for the femur and one for the tibia/fibula. The contents of the files will be provided in the following order.
Femur ALM node numbers:

- Femoral head point 1
- Femoral head point 2
- Femoral head point 3
- Femoral head point 4
- Lateral femoral epicondyle
- Medial femoral epicondyle

Tibia ALM node numbers:

- Lateral tibial plateau
- Medial tibial plateau
- Medial distal tibia
- Lateral distal fibula

4.1.4 Loading and boundary conditions (for the simulation cases)

The loading and boundary conditions for the recalibration and benchmarking simulations will be provided.

The loading and boundary conditions will be provided for the following benchmarking simulation cases:

- Passive flexion from 0 to 90 degrees
- Combined 10Nm valgus moment and 5 Nm internal rotation at 0 degrees flexion
- Combined 10Nm valgus moment and 5 Nm internal rotation at 30 degrees flexion
- Combined 10Nm valgus moment and 5 Nm internal rotation at 60 degrees flexion
- Combined 10Nm valgus moment and 5 Nm internal rotation at 90 degrees flexion

For all simulation cases, 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.

4.1.5 Interactions between model components (contacts, ties)

Cartilages will be modelled with the same rigid body material as their corresponding bones and therefore no contact implementation is required to maintain contact.

A sliding-elastic contact will be used for the contact between the tibial cartilage and femur cartilage. It will also be used for contact between any cartilage and the menisci.

Ligaments will be tied to their corresponding bones with a nodal rigid-tied interface.

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).

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

4.2 Intermediate and final calibration outcomes

4.2.1 Re-calibrated parameters

The prestrain values of the ACL, PCL, MCL and LCL that will be optimised during recalibration with the ~0 degrees and ~60 degrees robot data will be provided in a *.xlsx file.

4.2.2 Calibration fit error

The calibration error (a.k.a. Total error) to be minimized will be an error that depends on a number of factors including if the model ran to completion, if the AP position after prestretch was in the specified range, if the model ran to completion, and how far it converged.

The process for obtaining the Total error is detailed in section 5.1.5.

The optimised Total errors and Normalised errors (used to calculate the total error) obtained through each optimiser will be provided in a *.xlsx file.

4.2.3 Benchmarking error

Benchmarking errors will be reported as the RMSE between the simulated kinematics (position and angle) and robot kinematics, and the simulated kinetics (forces and moments) and robot kinetics, in all converged time steps for all 6 degrees of freedom (medial translation, posterior translation, superior translation, flexion rotation, valgus rotation, internal rotation). FIgures will be provided as *.png files and data provided in a *.xlsx file.

4.2.4 Changed model components

During the recalibration and benchmarking phase, only the prestrain stretch values of the ligaments will be changed. These will be provided as described in 4.2.2.

4.3 Intermediate and final outcomes of analysis of experimental load cases

4.3.1 Source data (as extracted from earmarked data set)

The robot data that will be used for the recalibration and benchmarking simulations will be provided in their original form (*.csv file).

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

Lookup tables consisting of displacements at any applied load will be used for recalibration. The lookup tables will be obtained from the earmarked robot data through polynomial fitting and interpolation of the robot data.

The fitted polynomials used to interpolate the robot data will be provided as *.npy files:

- AP_polynomial_60deg.npy
- AP_polynomial_coeffs_60deg.npy
- IE_polynomial_60deg.npy
- IE_polynomial_coeffs_60deg.npy
- VV_polynomial_0deg.npy
- VV_polynomial_coeffs_0deg.npy

The lookup tables generated from the polynomials will be provided as *.npy files:

- AP_forces_60.npy
- AP_position_60.npy
- IE_torques_60.npy
- IE_angles_60.npy
- VV_torques_0.npy
- VV_angles_0.npy

The load curves used for the prescribed displacement/rotation for the simulation cases will be provided as *.txt files, with images provided as *.png files.

4.4 Simulation cases

4.4.1 Loading and boundary conditions

The loading and boundary conditions for all simulations will be provided as plots in *.png files and next to that their corresponding *.feb, *.log and *.xplt files will be provided per simulation.

The simulations will be divided into five groups:

1. Passive flexion from 0 to 90 degrees
2. Combined 10Nm valgus moment and 5 Nm internal rotation at 0 degrees flexion
3. Combined 10Nm valgus moment and 5 Nm internal rotation at 30 degrees flexion
4. Combined 10Nm valgus moment and 5 Nm internal rotation at 60 degrees flexion
5. Combined 10Nm valgus moment and 5 Nm internal rotation at 90 degrees flexion

4.4.2 Target metrics for predictions

The RMSE between the kinematics and kinetics in the robot the model simulation results will be plotted and provided in *.png, with data 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.

4.4.4 Anticipated results

The tibiofemoral kinematics (6 DoF) during the simulation cases will be plotted and provided as a *.png file. These will include the model's anterior position (mm), joint distraction (mm), external rotation (rad), flexion rotation (rad), medial position (mm), and valgus rotation (rad) plotted against simulation time-step. These will be overlaid with the robot data.

The RMSE between the robot force and the simulated force in the model will be provided 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

We will recalibrate the knee against the robot data in two positions:

1. knee in extension at ~0 degrees flexion.
2. knee at ~60 degrees knee flexion.

We will use the VV robot data for the recalibration in extension, and the AP and IE robot data for recalibration at ~60 degrees (Table 1)

5.1.2 Generate lookup tables

We will add the kinematic offsets (from kinematic_offsets.csv) to the kinematic data in the files listed in Table 1.

This offset data will be fitted with a polynomial in python (numpy package: polyfit and poly1D) to smooth the calibration data and allow for interpolation between the data points.

The script (*OKS003_Robot_data_lookup_table_generation.py*) will be used to generate lookup tables to allow for rapid error calculations during calibration.

5.1.3 Anatomical coordinate system alignment

The ACS of the model will be registered to match the ACS of the robot data, which is based on the Grood and Suntay coordinate system (Grood and Suntay, 1983). The local ACS for each bone is defined by ALMs which will be obtained using the process below:

1. Finding the centroids of the segmented registration markers.
2. Fitting a hypersphere to the probed points of the registration markers.
3. Calculating the centroids of the hyperspheres.
4. Calculating the transform for registering the centroids from the segmented markers to the centroids of the probed markers.
5. Applying the registration transform to the segmented bone.
6. Fitting bone meshes to the transformed segmented bone.
7. Finding the closest bone mesh node to the probed ALMs.

This process will be done by modifying the script *OKS003_ACS_alignment_01092020.py*.
The following files (from previous phases) will be used in the process above.
Bone segmentations:

- OK_GP_NR1_Femur_Bone.pts
- TIBFIB_segm_resampled25000.xyz

Bone meshes:

- OKS003_fem_mesh_feb_nrs.txt
- SSM_fitted_5modes_newsegmentation.ply

Segmented registration marker spheres:

- fem_segmented_spheres_middle_lat.txt
- fem_segmented_spheres_middle_med.txt
- fem_segmented_spheres_middle_post.txt
- OKS003_tib_lat.ply
- OKS003_tib_med.ply
- OKS003_tib_post.ply

Probed registration markers:

- Fem_RM_lateral.xyz
- Fem_RM_medial.xyz
- Fem_RM_posterior.xyz
- Tib_RM_lateral.xyz
- Tib_RM_medial.xyz
- Tib_RM_posterior.xyz

5.1.4 Updating the template optimisation model

We will use the template models from the MC phase.

The template files that will be used for recalibration include:

- Base template model (defines control settings, geometry, initial orientation, initial loads, and boundary conditions, and applies prestrain):
  - OKS003_KM2_05112020.feb
- Template for calibration at ~0 degrees:
  - OKS003_VV.feb
- Template for performing flexion to ~60 degrees:
  - OKS003_Extension_to_Flexion_67.feb
- Templates for calibration at ~60 degrees:
  - OKS003_IE.feb
  - OKS003_AP.feb

The RCJ in the base template file will be updated with the new ALMs and ACS obtained from 5.1.3. This will be done using the python script, OKS003_write_ACS_in_feb.py, to write the new RCJ to the template OKS003_KM2_05112020.feb files.

The control settings will remain as in the MC phase with the following parameters:

- max_refs = 100 instead of 200
- rtol was deleted
- lstol changed from 0.5 to 0.9
- Timestepper:
  - min step size (dtmin) = 0.02 instead of 0.01
- Max retries = 3 instead of 5

The loads and boundary conditions will remain as in the MC phase with -20N of axial force applied during the recalibration simulations at ~0 (VV) and ~60 degrees (AP and IE).

The RCJ parameters will remain as in the MC phase with the following parameters:

- Tolerance = 0
- gaptol = 0.01
- Angtol =0.0001
- Force_penalty =10000
- Moment_penalty = 3000000

The sliding-elastic contacts will remain as in the MC phase with the following parameters:

- contact type = "sliding-elastic"
- laugon = 0
- tolerance = 0
- gaptol =0
- penalty = 1 OR 10
- auto_penalty = 0
- two_pass = 1
- search_tol = 0.01
- symmetric_stiffness = 0
- search_radius = 0.005
- seg_up = 0
- tension = 0
- minaug = 0
- maxaug = 40
- fric_coeff = 0
- smooth_aug = 0
- node_reloc = 0
- knmult = 0

For specific details on how the template model was obtained, please refer to the MC phase protocol deviations document.

5.1.5 Recalibration Optimisation

We will recalibrate using the same objective function and optimisation methods as in the MC phase except we will only use three optimisation algorithms: TNC, L-BFGS-B, and Powell. We will exclude using the Nelder-Mead optimisation algorithm as it performed poorly during MC.

This will be done using the scripts prepared in the MC phase:

- OKS003_Optimisation_Powell.py
- OKS003_Optimisation_TNC.py
- OKS003_Optimisation_LBFGSB.py

Five different sets of initial prestrain factor values will be used (Table 5), along with two different cartilage contact penalty factors (set to either 1 or 10), and three different optimisation algorithms (TNC, L-BFGS-B, and Powell) resulting in 30 separate optimisations (5 x 2 x 3). These will be run on high performance computers (HPCs).

Table 5. Five sets of initial prestrain values for recalibration.

Ligament / Set	1	2	3	4	5
ACL	0.85	0.75	0.85	0.85	0.85
PCL	1.15	1.15	1.25	1.15	1.15
MCL	0.98	0.98	0.98	0.90	0.98
LCL	0.95	0.95	0.95	0.95	0.95

5.1.5.1 Objective function simulations:

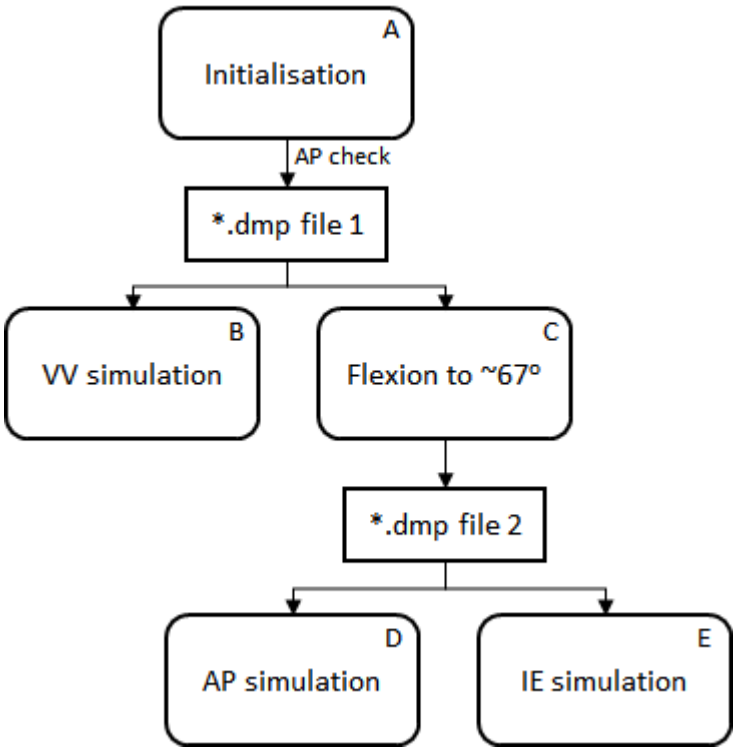

Figure 3. Overview of the 5 simulations (A, B, C, D, and E) within the objective function.

A. The objective function will begin by initialising the model in the imaging state (~5.5 degrees of knee flexion) and applying the prestrain. The model will then flex to ~7 degrees before creating a restart *.dmp file (*.dmp file 1).
    a. This initialisation will consist of three timesteps of 0.1 (i.e. 0.0 - 0.3).
    b. Prestrain will be applied using the following load curve:
    <loadcurve id="1, 2, 3 & 4" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
    </loadcurve>
    c. An axial load of -20N will be applied using the following load curve:
    <loadcurve id="8" type="linear">

```
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
    </loadcurve>
```

d. Flexion to ~7 degrees will be prescribed using the following load curve:

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.3,Flex7</point>
        <point>1.2,Flex7</point>
        <point>4,Flex7</point>
</loadcurve>
```

B. The restart facility in FEBio will be used to restart the simulation from the end of simulation A (*.dmp file 1) and perform the VV simulation.
   a. An AP check will be performed (see section 5.1.5.2).
   b. This simulation will consist of nine additional timesteps of 0.1 (i.e. 0.3 - 1.2).
   c. The VV moment will be applied using the following load curve:

```
<loadcurve id="9" type="linear">
        <point>0,0</point>
        <point>0.3,0</point>
        <point>0.6,5000</point>
        <point>1.2,-5000</point>
</loadcurve>
```

   d. The flexion angle will be fixed with the following load curve:

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.3,Flex7</point>
        <point>4,Flex7</point>
</loadcurve>
```

C. The simulation will be restarted from the end of simulation A (*.dmp file 1) to flex the knee to ~67 degrees. This will be done simultaneously with simulation B via multithreading in python. Once complete a restart *.dmp file (*.dmp file 2) will be created.
   a. This simulation will consist of nine additional timesteps of 0.1 (i.e. 0.3 - 1.2).
   b. Flexion will be prescribed using the following load curve:

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.3,Flex7</point>
        <point>1.2,Flex67</point>
        <point>4,Flex67</point>
</loadcurve>
```

D. The simulation will be restarted from the end of simulation C (*.dmp file 2) to perform the AP simulation.
    a. This simulation will consist of 9 additional timesteps of 0.1 (i.e. 1.2 - 2.1).
    b. An AP force will be applied using the following load curve:

```
<loadcurve id="10" type="linear">
        <point>0,0</point>
        <point>1.2,0</point>
        <point>1.5,100</point>
        <point>2.1,-100</point>
</loadcurve>
```

    c. The flexion angle will be fixed by the following load curve:

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.3,Flex7</point>
        <point>1.2,Flex67</point>
        <point>4,Flex67</point>
</loadcurve>
```

E. Simultaneously to simulation D, the simulation will be restarted from the end of simulation C (*.dmp file 2) to perform the IE simulation.
    a. This simulation will consist of 9 additional timesteps of 0.1 (i.e. 1.2 - 2.1).
    b. An IE moment will be applied using the following load curve:

```
<loadcurve id="7" type="linear">
        <point>0,0</point>
        <point>1.2,0</point>
        <point>1.5,4000</point>
        <point>2.1,-4000</point>
</loadcurve>
```

    c. The flexion angle will be fixed with the following load curve:

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.3,Flex7</point>
        <point>1.2,Flex67</point>
        <point>4,Flex67</point>
</loadcurve>
```

For all simulations (A, B, C, D, and E) the tibia will be fixed in all degrees of freedom, and the femur will be free in all degrees of freedom. Knee flexion will be prescribed in the RCJ for all simulations, which will consequently constrain the femur in the flexion-extension axis. The flexion angles, Flex7 and Flex67, will be the angle in radians that the knee needs to be flexed from the initial position to 7 degrees and 67 degrees respectively. This will be calculated after the ACS has been defined, but were 0.03881 and 1.0736 in the MC phase.

The input variables into the objective function will be the prestrain factors of the ACL, PCL, MCL and LCL.

5.1.5.2 Objective function minimisation error:

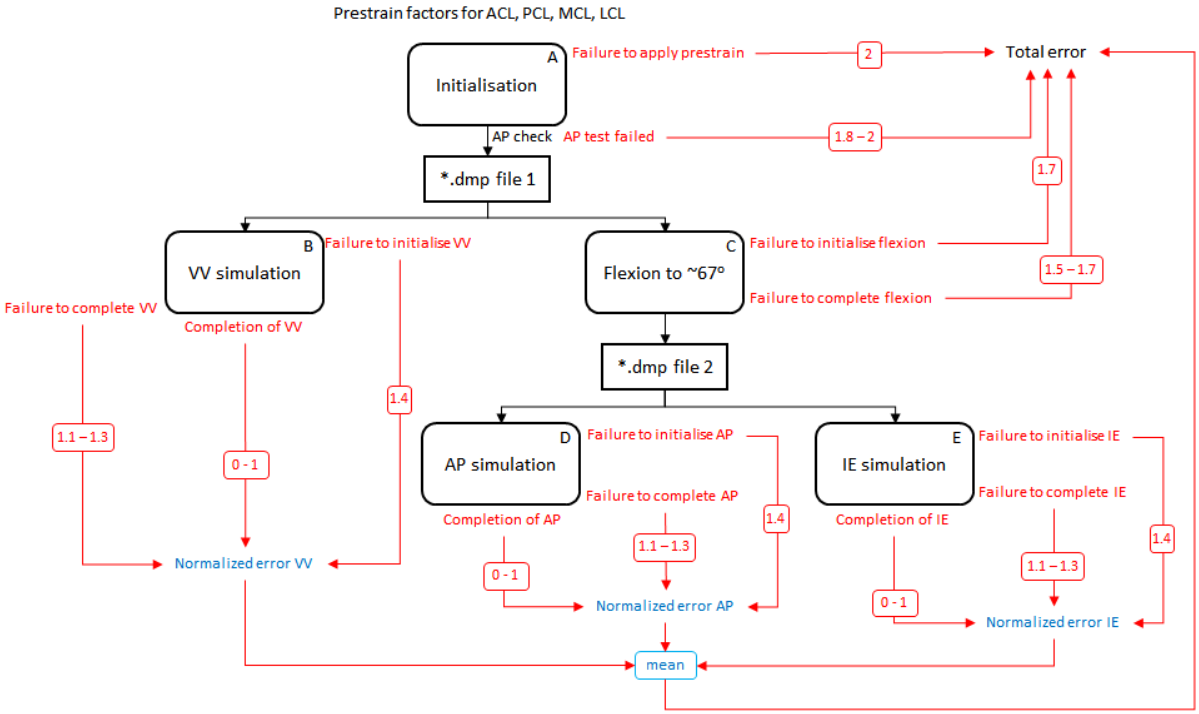

Figure 4. Schematic of the objective function including error ranges.

The conditions within the objective function and the ranges of the minimisation error that will be returned by the objective function are shown in Table 2.

Table 2. Error values and ranges during optimization for different conditions.

Condition	Error Type	Error Value/Range
Failure to apply prestrain	Total error	2
AP test failed*	Total error	1.8 - 2
Failure to initialise flexion	Total error	1.7
Failure to complete flexion	Total error	1.5 - 1.7
Failure to initialise VV	Normalised error VV	1.4
Failure to complete VV**	Normalised error VV	1.1 - 1.3
Failure to initialise AP	Normalised error AP	1.4
Failure to initialise IE	Normalised error IE	1.4
Failure to complete AP**	Normalised error AP	1.1 - 1.3
Failure to complete IE**	Normalised error IE	1.1 - 1.3
Completion of VV***	Normalised error VV	0 - 1
Completion of AP***	Normalised error AP	0 - 1
Completion of IE***	Normalised error IE	0 - 1

* If the AP position is greater than -6.196 then the “AP test failed” condition is met. The Total error will be returned according to equation 1.

** When a “Failure to complete” condition is met, the Normalised error will be set according to Table 3.

*** When a “Completion” condition is met, the Normalised error will be calculated by normalizing the RMSE between the simulated position/angle and the robot position/angle in all converged time steps to the maximum position/angle in the robot data (Table 4).

AP test failed Total error = 1.8 + (((|-6.196 - AP position|)/6.196)/5) (1)

Table 3. Normalised errors for “Failure to complete” conditions. t is the percentage completion of the simulation.

Percentage Completion (%)	Normalised error*
$0 \leq t < 25$	1.200
$25 \leq t < 50$	1.175
$50 \leq t < 75$	1.150
$75 \leq t < 100$	1.125

* If the “Failure to complete” condition is met in consecutive objective function iterations, an adjustment factor will be applied to the normalized error according to equation 2.

Normalised error = Normalised error * (1 + (0.001 * (N - 1))) (2)
where N is the number of consecutive iterations where the “Failure to complete” condition is met.

Table 4. Normalization values for MC and MB phase. Note the small difference due to only using the processed data in MB.

Simulation case	MC Normalisation value	MB Normalisation value
AP at ~60º	4.365 mm	4.303 mm
IE at ~60º	0.6561 rad	0.6530 rad
VV at ~ 0º	0.1197 rad	0.1187 rad

Unless any of the first four conditions in Table 2 are met where the “Total error” is returned immediately, the “Total error” will be calculated using equation 3. This is the error that will be minimized during re-calibration.

Total error = (Normalised error AP + Normalised error IE + Normalized error VV)/3 (3)

5.1.6 Write recalibration outcomes to file

Recalibration will result in optimised prestrain factors for the ACL, PCL, MCL, and LCL. These will be written into the template model (OKS003_KM2_05112020.feb) used in the recalibration for further benchmarking simulations. The optimised prestrain factors will also be written to a *.txt file.

5.2 Steps to implement load cases as loading and boundary conditions

New models will be generated from the optimised template created in 5.1.6, with simulation case specific loads and boundary conditions.

5.2.1 Default control settings

The benchmarking simulations will consist of 20 time steps of 0.1 with the following control settings:

```
<Control>
    <time_steps>20</time_steps>
    <step_size>0.1</step_size>
    <max_refs>100</max_refs>
    <max_ups>0</max_ups>
    <diverge_reform>1</diverge_reform>
    <reform_each_time_step>1</reform_each_time_step>
    <dtol>0.01</dtol>
    <etol>0.01</etol>
    <lstol>0.9</lstol>
    <time_stepper>
        <dtmin>0.02</dtmin>
        <dtmax>0.1</dtmax>
        <max_retries>10</max_retries>
        <opt_iter>10</opt_iter>
    </time_stepper>
    <min_residual>0.001</min_residual>
    <qnmethod>0</qnmethod>
    <analysis type="static" />
    <symmetric_stiffness>0</symmetric_stiffness>
</Control>
```

5.2.2 Template loads and boundary conditions

This section describes the default loads and boundary conditions from which the simulations will be updated.

The tibia will be fixed in all degrees of freedom using a rigid constraint:

```
<rigid_body mat="2">
    <fixed bc="x" />
    <fixed bc="y" />
    <fixed bc="z" />
    <fixed bc="Rx" />
    <fixed bc="Ry" />
    <fixed bc="Rz" />
</rigid_body>
```

There will be 11 load curves defined in this template model. Not all load curves will be used. They will correspond to:

1. Load-displacement characteristics of the ACL
2. Load-displacement characteristics of the PCL
3. Load-displacement characteristics of the MCL

4. Load-displacement characteristics of the LCL
5. Prescribed flexion rotation in the RCJ
6. Prescribed medial force in the RCJ
7. Prescribed internal rotation moment in the RCJ
8. Prescribed superior force in the RCJ
9. Prescribed valgus moment in the RCJ
10. Prescribed anterior force in the RCJ
11. Prescribed anterior translation in the RCJ

By default, their corresponding load curves will be:

```
<loadcurve id="1" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
</loadcurve>
<loadcurve id="2" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
</loadcurve>

<loadcurve id="3" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
</loadcurve>

<loadcurve id="4" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
</loadcurve>

<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>1.5,0</point>
        <point>3.5,0</point>
</loadcurve>

<loadcurve id="6" type="linear">
        <point>0,0</point>
        <point>4,0</point>
</loadcurve>
```

```
<loadcurve id="7" type="linear">
        <point>0,0</point>
        <point>1.5,0</point>
        <point>2.0,0</point>
        <point>2.5,0</point>
        <point>3.0,0</point>
        <point>3.5,0</point>
</loadcurve>

<loadcurve id="8" type="linear">
        <point>0,0</point>
        <point>0.1,0</point>
        <point>0.15,1</point>
        <point>4,1</point>
</loadcurve>

<loadcurve id="9" type="linear">
        <point>0,0</point>
        <point>1.5,0</point>
        <point>2.0,0</point>
        <point>2.5,0</point>
        <point>3.0,0</point>
        <point>3.5,0</point>
</loadcurve>

<loadcurve id="10" type="linear">
        <point>0,0</point>
        <point>1.5,0</point>
        <point>2.0,0</point>
        <point>2.5,0</point>
        <point>3.0,0</point>
        <point>3.5,0</point>
</loadcurve>

<loadcurve id="11" type="linear">
        <point>0,0</point>
        <point>4,0</point>
</loadcurve>
```

5.2.3 Passive flexion from 0 to 90 degrees

Flexion rotation will be prescribed from 0 to 90 degrees by adjusting load curve 5. The parameters Flex0 and Flex90 will need to be recalculated according to the ACS. In the MC phase, Flex0 was -0.0943379 and Flex90 was 1.4764621.

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, Flex0</point>
        <point>2, Flex90</point>
</loadcurve>
```

Axial loads will be applied by adjusting load curve 8.

```
<loadcurve id="8" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, -44</point>
        <point>2, -63</point>
</loadcurve>
```

5.2.4 Combined 10Nm valgus moment and 5 Nm internal rotation at 0 degrees flexion

This simulation will consist of setting the flexion angle while applying a compressive load, then applying a valgus moment, and finally applying an internal rotation moment. Flex7 will need to be calculated according to the new ACS but will be ~0.125425571.

Flexion rotation will be set to 7.19 degrees by adjusting load curve 5.

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, Flex7</point>
        <point>2, Flex7</point>
</loadcurve>
```

A compressive axial load of 18.84 N will be applied by adjusting load curve 8.

```
<loadcurve id="8" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, -18.84</point>
        <point>2, -18.84</point>
</loadcurve>
```

A valgus moment of 10Nm will be applied using load curve 9.

```
<loadcurve id="9" type="linear">
        <point>0,0</point>
        <point>0.5,0</point>
        <point>1.3, 9982.59</point>
        <point>2, 9982.59</point>
</loadcurve>
```

An internal rotation moment of 5Nm will be applied using load curve 7.

```
<loadcurve id="7" type="linear">
        <point>0,0</point>
        <point>1.3,0</point>
        <point>2, 4995.86</point>
</loadcurve>
```

5.2.5 Combined 10Nm valgus moment and 5 Nm internal rotation at 30 degrees flexion

This simulation will consist of setting the flexion angle while applying a compressive load, then applying a valgus moment, and finally applying an internal rotation moment. Flex36 will need to be calculated according to the new ACS but will be ~0.638695923.

Flexion rotation will be set to 36.59 degrees by adjusting load curve 5.

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, Flex36 </point>
        <point>2, Flex36 </point>
</loadcurve>
```

A compressive axial load of 18.68 N will be applied by adjusting load curve 8.

```
<loadcurve id="8" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, -18.68</point>
        <point>2, -18.68</point>
</loadcurve>
```

Valgus moment of 10Nm will be applied using load curve 9.

```
<loadcurve id="9" type="linear">
        <point>0,0</point>
        <point>0.5,0</point>
        <point>1.3, 9957.86</point>
        <point>2, 9957.86</point>
</loadcurve>
```

Internal rotation moment of 5Nm will be applied using load curve 7.

```
<loadcurve id="7" type="linear">
        <point>0,0</point>
        <point>1.3,0</point>
        <point>2, 5008.15</point>
</loadcurve>
```

5.2.5 Combined 10Nm valgus moment and 5 Nm internal rotation at 60 degrees flexion

Flexion rotation will be set to 66.85 degrees by adjusting load curve 5. Flex66 will need to be calculated according to the new ACS but will be ~1.16680027.

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, Flex66</point>
        <point>2, Flex66</point>
</loadcurve>
```

A compressive axial load of 22.94 N will be applied by adjusting load curve 8.

```
<loadcurve id="8" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, -22.94</point>
        <point>2, -22.94</point>
</loadcurve>
```

Valgus moment of 10Nm will be applied using load curve 9.

```
<loadcurve id="9" type="linear">
        <point>0,0</point>
        <point>0.5,0</point>
        <point>1.3, 10036.58</point>
        <point>2, 10036.58</point>
</loadcurve>
```

Internal rotation moment of 5Nm will be applied using load curve 7.

```
<loadcurve id="7" type="linear">
        <point>0,0</point>
        <point>1.2,0</point>
        <point>2, 4960.82</point>
</loadcurve>
```

5.2.6 Combined 10Nm valgus moment and 5 Nm internal rotation at 90 degrees flexion

Flexion rotation will be set to 97.70 degrees by adjusting load curve 5. Flex97 will need to be calculated according to the new ACS but will be ~1.705248855.

```
<loadcurve id="5" type="linear">
        <point>0,0</point>
        <point>0.15,0</point>
        <point>0.5, Flex97</point>
        <point>2, Flex97</point>
</loadcurve>
```

A compressive axial load of 19.36 N will be applied by adjusting load curve 8.

```
<loadcurve id="8" type="linear">
    <point>0,0</point>
    <point>0.15,0</point>
    <point>0.5, -19.36</point>
    <point>2, -19.36</point>
</loadcurve>
```

Valgus moment of 10Nm will be applied using load curve 9.

```
<loadcurve id="9" type="linear">
    <point>0,0</point>
    <point>0.5,0</point>
    <point>1.3, 9954.79</point>
    <point>2, 9954.79</point>
</loadcurve>
```

Internal rotation moment of 5Nm will be applied using load curve 7.

```
<loadcurve id="7" type="linear">
    <point>0,0</point>
    <point>1.3,0</point>
    <point>2.5, 4990.42</point>
</loadcurve>
```

5.3 Steps to perform benchmark simulations

Once the calibrated prestrain values and new load curves have been written to the simulation case *.feb files, the benchmarking simulations will be run on a computer or HPC. The loads and moments will be outputted to the *.log file for plotting and analysis.

```
<logfile>
    <node_data data="x;y;z">9018, 9087, 15026, 15417, 105234, 21871, 3840, 39</node_data>
    <rigid_body_data data="x;y;z" delim=",">1,2,3</rigid_body_data>
    <rigid_body_data data="Fx;Fy;Fz;Mx;My;Mz" delim=",">1,2,3</rigid_body_data>
    <rigid_connector_data data="RCFx;RCFy;RCFz;RCMx;RCMy;RCMz"
delim=",">1,2,3</rigid_connector_data>
</logfile>
```

5.4 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 requirements: Microsoft office Excel
Hardware requirements: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 10 minutes
Expertise level needed: Low
Computational cost: Low

2. Generate lookup tables
Software requirements: Python 3.6
Hardware requirements: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 1 day
Expertise level needed: Medium
Computational cost: Low

3. ACS alignment
Software requirements: Python 3.6 & Matlab R2017a Academic Licence
Hardware requirements: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 1 week
Expertise level needed: Medium
Computational cost: Low

4. Update template model
Software requirements: Python 3.6 & FEBio version 3.0.0 (with prestrain plug-in)
Hardware requirements: Intel(R) Core(™) i7-7700HQ CPU @ 2.8GHz, 32.0 GB RAM.
Anticipated man hours: ~ 1 week
Expertise level needed: Medium
Computational cost: Low

5. Recalibration
Software requirements: Python 3.6 & FEBio version 3.0.0 (with restart function & prestrain plug-in)
Hardware requirements: 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 month
Expertise level needed: Medium/High
Computational cost: High

6. Benchmarking simulations
Software requirements: Python 3.6 & FEBio version 3.0.0 (with prestrain plug-in)
Hardware requirements: 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 calibration 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.

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.

Orozco, G. A., Tanska, P., Mononen, M. E., Halonen, K. S., & Korhonen, R. K. (2018). The effect of constitutive representations and structural constituents of ligaments on knee joint mechanics. Scientific reports, 8(1), 1-15.