Modeling and Simulation Workflow Using Open Knee(s) Data

Model Benchmarking Specifications

Cleveland Clinic Approach

Document created on February 07,2021; last updated on March 22, 2021.

Document prepared by

Snehal K. Chokhandre, MSc – chokhas@ccf.org, +1 (216) 445 3555

Department of Biomedical Engineering and Computational Biomodeling (CoBi) Core, Lerner Research Institute, Cleveland Clinic, 9500 Euclid Avenue (ND20), Cleveland, OH 44195, USA

Ellen Klonowski, BSc – klonowe@ccf.org , +1 (216) 445 3555

Department of Biomedical Engineering and Computational Biomodeling (CoBi) Core, Lerner Research Institute, Cleveland Clinic, 9500 Euclid Avenue (ND20), Cleveland, OH 44195, USA

***Ahmet Erdemir, PhD –** erdemira@ccf.org, +1 (216) 445 9523

Department of Biomedical Engineering and Computational Biomodeling (CoBi) Core, Lerner Research Institute, Cleveland Clinic, 9500 Euclid Avenue (ND20), Cleveland, OH 44195, USA

**For correspondence.*

Table of Contents

Cite this document as

> Chokhandre, SK, Klonowski E, Erdemir, A, *Modeling and simulation workflow using Open Knee(s) data: model benchmarking specifications – Cleveland Clinic approach*, March 22, 2021, Cleveland Clinic, Cleveland, Ohio, USA.

Contributions to this document by Snehal Chokhandre, Ellen Klonowski and Ahmet Erdemir were supported by the National Institute of Biomedical Imaging and Bioengineering, National Institutes of Health (R01EB024573; *Reproducibility in simulation-based prediction of natural knee mechanics*; Principal Investigator: Erdemir) and in part by the National Institute of General Medical Sciences, National Institutes of Health (R01GM104139; *Open Knee(s): virtual biomechanical representations of the knee joint*, Principal Investigator: Erdemir).

Synopsis

This document describes planned model re-calibration and benchmarking specifications that are aimed for generating a newly calibrated model of the knee joint based on an initial working and calibrated model[1–5], and an existing joint testing data set from the Open knee(s) project[6], which was acquired on the same specimen and reprocessed for re-calibration process with data descriptions clarified for comprehensibility. Additional data from the same specimen originally available, will be used for benchmarking purposes along with the newly calibrated model. The proposed modeling activities are in response to the *Model Benchmarking* phase[7] of the project *Reproducibility in simulation-based prediction of natural knee mechanics*, a study funded by the National Institute of Biomedical Imaging and Bioengineering, National Institutes of Health (Grant No. R01EB024573)[8]. The outlined choices for modeling and simulation processes represent those of the Cleveland Clinic team, who launched and has been maintaining the Open Knee(s)[6]. These choices are primarily aimed for pragmatic, yet comprehensive, re-calibration and benchmarking of an anatomically and mechanically detailed and extensible knee joint model incorporating its major tissue structures.

Initial Calibrated Model

Described model re-calibration and benchmarking workflow utilizes partial outputs of the previously calibrated model and derivative modeling and simulation outputs generated through the *Model Calibration* phase[4] of the project *Reproducibility in simulation-based prediction of natural knee mechanics*, a study funded by the National Institute of Biomedical Imaging and Bioengineering, National Institutes of Health (Grant No. R01EB024573)[8]. This model was based on an existing data set from the Open Knee(s) project, specifically those from specimen oks003[6]. Development of this model was described as part of the model development specifications[1], model calibration specifications[4] and related protocol deviations[5]. The model re-calibration workflow will utilize the model calibrated for optimal mesh densities obtained via mesh convergence studies and, material properties in the *Model Calibration* phase[9]. The re-calibration process will involve in situ strain optimization as the data was re-processed in a manner that differed from that employed in the *Model Calibration* phase[9].

Data Utilized

Described model re-calibration and benchmarking workflows utilize Open Knee(s) data, specifically those from specimen oks003[6]. This specific data set was disseminated at the project site of *Reproducibility in simulation-based prediction of natural knee mechanics*, a study funded by the National Institute of Biomedical Imaging and Bioengineering, National Institutes of Health (Grant No. R01EB024573)[8].

This data set was also part of the *Model Calibration* phase[9] of the project in an unprocessed form. The data for re-calibration and benchmarking was processed and provided in a form amenable for using with the models and can be accessed as the package Data for MB – oks003[10]. Donor specifics of specimen oks003 are:

- Left knee

- Age: 25 years
- Gender: Female
- Height: 1.73 m
- Weight: 68 kg
- BMI: 22.8

Described model re-calibration specifications utilize the following specimen-specific mechanical testing data sets for re-calibration:

1. Probed points on registration markers and anatomical landmarks; primarily for coordinate system registration. (Plain text in .xyz format[11] and binary image .png[12])

File Name	**Description**	**Coordinate system**	**Format**
Fem_RM_lateral.xyz	x,y,z coordinates of 10 probed points on lateral femur registration marker [mm]	Femur optotrack sensor	.xyz
Fem_RM_medial.xyz	x,y,z coordinates of 10 probed points on medial femur registration marker [mm]	Femur optotrack sensor	.xyz
Fem_RM_posterior.xyz	x,y,z coordinates of 10 probed points on posterior femur registration marker [mm]	Femur optotrack sensor	.xyz
Tib_RM_lateral.xyz	x,y,z coordinates of 10 probed points on lateral tibia registration marker [mm]	Tibia optotrack sensor	.xyz
Tib_RM_medial.xyz	x,y,z coordinates of 10 probed points on medial tibia registration marker [mm]	Tibia optotrack sensor	.xyz
Tib_RM_posterior.xyz	x,y,z coordinates of 6 probed points anatomical landmarks on femur bone [mm]	Tibia optotrack sensor	.xyz
Fem_AL.xyz	x,y,z coordinates of 6 probed points anatomical landmarks on femur bone [mm]	Femur optotrack sensor	.xyz
Tib_AL.xyz	x,y,z coordinates of 6 probed points anatomical landmarks on tibia bone [mm]	Tibia optotrack sensor	.xyz

2. Laxity data at 0° flexion. Primarily anterior-posterior and varus-valgus laxity data for calibration of in situ ligament strains. (plain text in csv format[13] and binary image .png[12])

File name	**Description**	**Formats**
Laxity_0deg_AP1_kinematics_in_JCS_experiment, Laxity_0deg_AP1_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Anterior loading	csv, png
Laxity_0deg_AP2_kinematics_in_JCS_experiment, Laxity_0deg_AP2_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Posterior loading	csv, png
Laxity_0deg_VV1_kinematics_in_JCS_experiment, Laxity_0deg_VV1_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Varus loading	csv, png
Laxity_0deg_VV2_kinematics_in_JCS_experiment, Laxity_0deg_VV2_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Valgus loading	csv, png

Described model re-calibration specifications utilize the following specimen-specific mechanical testing data sets for post-calibration simulations:

1. Optimized passive flexion, up to 90° for passive flexion simulation. (plain text in csv format[13] and binary image .png[12])

File name	**Description**	**Formats**
Passive_Flexion_Kinematics_in_JCS_experiment Passive_Flexion_TibiaKinetics_in_TibiaCS_experiment	Passive flexion processed data	csv, png

2. All laxity data (anterior-posterior, varus-vagus, internal-external rotation) at 0°, 30°, 60°, 90° flexion for laxity simulations. (plain text in csv format[13] and binary image .png[12])

File name	**Description**	**Formats**
Laxity_0deg_AP1_kinematics_in_JCS_experiment, Laxity_0deg_AP1_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Anterior loading	csv, png
Laxity_0deg_AP2_kinematics_in_JCS_experiment, Laxity_0deg_AP2_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Posterior loading	csv, png
Laxity_0deg_EI1_kinematics_in_JCS_experiment, Laxity_0deg_EI1_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, External rotation loading	csv, png
Laxity_0deg_EI2_kinematics_in_JCS_experiment, Laxity_0deg_EI2_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Internal rotation loading	csv, png
Laxity_0deg_VV1_kinematics_in_JCS_experiment, Laxity_0deg_VV1_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Varus loading	csv, png
Laxity_0deg_VV2_kinematics_in_JCS_experiment, Laxity_0deg_VV2_TibiaKinetics_in_TibiaCS_experiment	0 degree laxity, Valgus loading	csv, png
Laxity_30deg_AP1_kinematics_in_JCS_experiment, Laxity_30deg_AP1_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, Anterior loading	csv, png
Laxity_30deg_AP2_kinematics_in_JCS_experiment, Laxity_30deg_AP2_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, Posterior loading	csv, png
Laxity_30deg_EI1_kinematics_in_JCS_experiment, Laxity_30deg_EI1_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, External rotation loading	csv, png
Laxity_30deg_EI2_kinematics_in_JCS_experiment, Laxity_30deg_EI2_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, Internal rotation loading	csv, png
Laxity_30deg_VV1_kinematics_in_JCS_experiment, Laxity_30deg_VV1_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, Varus loading	csv, png
Laxity_30deg_VV2_kinematics_in_JCS_experiment, Laxity_30deg_VV2_TibiaKinetics_in_TibiaCS_experiment	30 degree laxity, Valgus loading	csv, png
Laxity_60deg_AP1_kinematics_in_JCS_experiment, Laxity_60deg_AP1_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, Anterior loading	csv, png
Laxity_60deg_AP2_kinematics_in_JCS_experiment, Laxity_60deg_AP2_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, Posterior loading	csv, png
Laxity_60deg_EI1_kinematics_in_JCS_experiment, Laxity_60deg_EI1_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, External rotation loading	csv, png

Laxity_60deg_EI2_kinematics_in_JCS_experiment, Laxity_60deg_EI2_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, Internal rotation loading	csv, png
Laxity_60deg_VV1_kinematics_in_JCS_experiment, Laxity_60deg_VV1_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, Varus loading	csv, png
Laxity_60deg_VV2_kinematics_in_JCS_experiment, Laxity_60deg_VV2_TibiaKinetics_in_TibiaCS_experiment	60 degree laxity, Valgus loading	csv, png
Laxity_90deg_AP1_kinematics_in_JCS_experiment, Laxity_90deg_AP1_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, Anterior loading	csv, png
Laxity_90deg_AP2_kinematics_in_JCS_experiment, Laxity_90deg_AP2_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, Posterior loading	csv, png
Laxity_90deg_EI1_kinematics_in_JCS_experiment, Laxity_90deg_EI1_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, External rotation loading	csv, png
Laxity_90deg_EI2_kinematics_in_JCS_experiment, Laxity_90deg_EI2_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, Internal rotation loading	csv, png
Laxity_90deg_VV1_kinematics_in_JCS_experiment, Laxity_90deg_VV1_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, Varus loading	csv, png
Laxity_90deg_VV2_kinematics_in_JCS_experiment, Laxity_90deg_VV2_TibiaKinetics_in_TibiaCS_experiment	90 degree laxity, Valgus loading	csv, png

Described model benchmarking specifications utilize the following specimen-specific mechanical testing data sets for re-calibration:

1. Combined loading data (for benchmarking). Combination of valgus moment, internal rotation moment at different flexion angles. (plain text in csv format[13] and binary image .png[12])

File name	**Description**	**Formats**
Combined_Kinematics Combined_Kinetics	Combined loading	csv, png

kinematics_offsets.csv file is also provided with the re-calibration and benchmarking data which contains the kinematics offsets (provided in the same units as the kinematics data). Details of joint mechanics testing specifics can be found at Open Knee(s) project site[6].

Overview of Modeling and Simulation Processes

A previously developed and calibrated three-dimensional computational model of the knee, specifically finite element representation of the tibiofemoral and patellofemoral joints, will be utilized to complete the re-calibration and benchmarking. Re-calibration procedures will include calibration of in situ ligament strains relying on specimen-specific joint kinematics-kinetics data. The probed points information required for registration and the kinetics-kinematics data have been provided in a processed form for re-calibration and benchmarking. Also, the joint data processing differs from the manner in which the data was processed in the *Model Calibration* phase[9] hence necessitating the in-situ strain re-calibration. This model has been originally calibrated for mesh convergence and confirmation of material properties. For re-calibration assessment, two sets of simulations will be performed to understand the influence of model modifications on predicted passive flexion response and to document the correspondence of predicted joint kinematics-kinetics response with specimen-specific passive flexion and joint laxity data. For model benchmarking, a simulation of joint motion under combined loads will be performed and the correspondence of predicted joint kinematics-kinetics response with specimen-specific combined loading will be obtained to assess the efficacy of the calibration process in development of a general purpose model.

Specimen-specific joint mechanics data that was provided will be used for in situ ligament strain re-calibration and for simulation of experimental loading scenarios. Registration markers were attached on the specimen (three for each bone) as part of testing protocols. Raw surface geometries of these markers were already generated from imaging data, as part of model development[1]. Joint testing data also have points probed on each registration marker[14,15]. A sphere will be fit to registration marker surface points (both to those from imaging and from joint testing). Centers of these spheres will be used to calculate the coordinate system transformation matrix between local bone coordinate systems of femur and tibia during joint testing and image (therefore, model) coordinate system. Anatomical landmarks collected during joint testing[14,15] will be transformed to the model to establish an anatomical joint coordinate system registered to experiments[16]. Joint kinematics-kinetics collected during passive flexion and laxity testing (anterior-posterior translation, internal-external rotation, varus-valgus at 0°, 30°, 60°, 90° flexion) and combined loading[14,17] will be used. It should be noted that experimental joint kinematics-kinetics were provided relative to a reference state, for which joint coordinate system offsets were also provided. Joint kinematics-kinetics of simulations will be relative to the reference state of the model (as build the images), where offsets of the joint coordinate system after its reconstruction can be calculated.

A simplified calibration approach will be implemented to manage the high cost of simulations. The full knee model with converged meshes, confirmed material properties, and experiment coordinate systems[4,5] will be simulated to replicate a subset of laxity tests performed on the specimen at 0° flexion. Given an initial set of in situ strains[1,2], a sequence of simulations will calculate anterior cruciate ligament, posterior cruciate ligament, medial collateral ligament, and lateral collateral ligament in situ strains to minimize differences between predicted and measured anterior laxity, posterior laxity, valgus laxity, and varus laxity responses, respectively. This sequence of simulations and optimizations will be repeated until in situ strains of these ligaments stabilize within a predefined threshold (absolute change of 0.001). Performing simulations only at 0° flexion will prevent potentially costly flexion simulations. This analysis assumes that the contribution of certain ligaments dominate laxity characteristics and by doing so, simplifies a multivariate optimization problem to a sequence of scalar optimization problems. Repeated optimizations will likely accommodate for potential contributions from other ligaments.

Multiple customized full knee models will be generated for simulations of passive flexion (test case used in model development[1,2]) and laxity testing. Passive flexion simulations in this attempt will document the role of in situ strain calibration on predictions of knee joint response. Two cases will include models with 1) converged meshes, confirmed material properties, and re-calibrated in situ ligament strains; and, 2) with all modifications including experiment coordinate systems. Laxity testing simulations and benchmarking simulation will aim for in silico reproduction of laxity and combined loading experiments. Full knee models with all modifications, driven by registered experiment loading at a given flexion, will be used. Simulation outputs will be reported along with measured joint kinematics-kinetics data to understand predictive capacity of the model for specimen-specific laxity response. A total of 24 laxity cases will be simulated: a combination of flexion angles at which laxity tests were performed (0°, 30°, 60°, 90°), axis of loading (tested degrees of freedom - anterior-posterior translation, internal-external rotation, varus-valgus), direction of loading (positive, negative). The benchmarking case will be a combined loading scenario with a combination of valgus moment and internal rotation moment at different flexion angles.

FEBio[18], along with FEBio PreStrain Plugin[19], will be used to conduct finite element analysis (solid mechanics, based on implicit static solver). Simulation results will be visualized using PostView[20]. Specimen-specific joint mechanics data and predicted kinematics-kinetics of the joints will be processed to report joint movements in all loading cases. Python[21] and SciPy[22], along with auxiliary Python packages[23], will be used to automate data analysis, model customization, and post processing. All modeling and simulation outputs, intermediate and final, will be publicly disseminated through an online repository[24].

Detailed Modeling and Simulation Outputs

Model Benchmarking specifications will result in the following intermediate and final outputs. The Workflow section below provides detailed instructions on how to obtain these.

File	**Description**	**File Format**
Model Properties	XML based text file which specifies the material properties of all tissues in the model, and the coordinates of the manually chosen anatomical landmarks (used as input for the customization script); including templates for in situ ligament strain calibration, and post-calibration simulations of experiment conditions.	.xml[25]
Customized Models (FEBio Input File)	XML based text file (for finite element analysis with FEBio) customized to include mesh definitions, tissue interactions, tissue-specific constitutive models, in situ ligament strains, representation of additional stabilizing structures, anatomical knee joint coordinate systems, specialized loading and boundary conditions to represent passive flexion, output requests relevant to knee mechanics; including numerical analysis settings; including customizations for in situ ligament strain calibration, and post-calibration simulations of experiment conditions.	.feb[26]
Raw Simulation Results	Binary (.xplt) and text files (.log) obtained by simulation of passive flexion using calibrated customized model with FEBio; in addition, results of in situ ligament strain calibration, post-calibration simulations of experiment conditions.	.xplt[26] .log[26]
Processed Simulation Results	CSV based text files storing extracted knee kinematics and kinetics during passive flexion simulations using calibrated model; processed using raw simulation results and supported by graphs as binary image files; in addition, processed results of in situ ligament strain calibration, post-calibration simulations of experiment conditions.	.csv[13] .png[12]

File	Description	File Format
Registration Results	XML based text files which includes coordinate system transformation matrices between joint testing and imaging coordinate systems of bones, experimental anatomical landmarks transformed to model coordinate systems, and registration error estimates.	.xml[25]
Model Prediction Errors	CSV based text files storing experimental and model predicted knee kinematics and kinetics and errors describing correspondence between model predictions against experimental data; supported by XML based text file providing prediction errors in summary form.	.csv[13] .xml[25]
Calibration Results	XML based text files summarizing target parameters and fit error before and after calibration.	.xml[25]

Workflow

Registration for Specimen-Specific Calibration

Target Outcome

Coordinate system transformation matrices between joint testing and imaging coordinate systems of bones and experimental anatomical landmarks transformed to model coordinate system in XML based text files. Full knee model with joint coordinate system defined to align with the experimental coordinate system, in FEBio format (.feb, XML[25] based text file)[26].

Burden

Software requirements:

Python. Python is a high-level multi-platform programming language (free and open source GPL compatible Python Software Foundation license, see https://www.python.org)[21]. Any contemporary version available for the computing platform can be used; 3.8.0 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, version 2.7 may be used.

SciPy. SciPy is a Python based open source software platform for mathematics, science and engineering (free and open source BSD-new license, see https://www.scipy.org)[22]. Any contemporary version available for the computing platform can be used; 1.3.1 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, a version compatible with Python 2.7 may be used.

Python Scripts. Python[21] script register_probed_points.py developed and used in the *Model Calibration* phase[4,5] will be used. The Python[21] script was written in house to fit spheres to the digitized registration markers, and the segmented registration markers, and find the transformation from experiment to image coordinate system using singular value decomposition between sphere centers. Digitized anatomical landmarks are then transformed from the experiment coordinate system to the image coordinate system. Source code is available at https://simtk.org/svn/openknee/app/KneeHub/src . The script will be modified such that instead of reading and interpreting information from state file, provided .xyz[11] files will be used.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux.

Anticipated Man Hours and Expertise Level:

1 day of full-time effort from a research engineer with bachelor's degree, mechanical/biomedical background, <3 years of research experience, and familiarity to rigid body dynamics, data processing, and scripting.

Computational Cost:

Minimal compared to required interactions with computer scripts.

Protocols

Input

Experimental probed points on registration markers and anatomical landmarks; raw registration marker geometries from imaging[3]; FEBio model file of the full knee with converged meshes and confirmed material properties.

Registration

A sphere will be fit to probed points on each registration marker to obtain its center in the local bone motion tracking system coordinate system. Similarly, a sphere will be fit to raw registration marker geometries obtained by segmentation of imaging data to obtain their centers in image coordinate system. For each cluster of registration markers on the bone, the transformation matrix will be calculated between the local bone motion tracking system and image coordinate system using centroids of registration markers[27]. In following, anatomical landmarks on each bone, which are probed during mechanical testing, will be transformed to image coordinate system to serve as the foundation to redefine model joint coordinate systems and cylindrical joint axes. The coordinate systems of tibia and femur will be updated based on descriptions provided in the experiment documentation[16] (also see below). Patella coordinate system will remain the same due to incomplete data on patella registration marker assembly.

Tibia

T_1= Most lateral point on the tibial plateau

T_2= Most medial point on the tibial plateau

T_3= Distal tibia point (medial malleolus of the tibia: most medial point)

T_4= Distal tibia point (medial malleolus of the tibia: most medial point)

T_5= Distal tibia point (lateral malleolus of the tibia: most lateral point)

T_6= Distal tibia point (lateral malleolus of the tibia: most lateral point)

tibial origin: $O_T = \frac{T_1 + T_2}{2}$

Tibia z-axis : $T_z = \frac{O_T - T_{distal}}{|O_T - T_{distal}|}$, where $T_{distal} = \frac{T_3 + T_4 + T_5 + T_6}{4}$

Tibia y-axis: $T_y = T_z \times T_{x,temp}$, normalized; where $T_{x,temp} = \frac{T_2 - T_1}{|T_2 - T_1|}$

Tibia x-axis: $T_x = T_y \times T_z$, normalized

Femur

F_1= Lateral femoral epicondyle

F_2= Medial femoral epicondyle

F_3= Proximal femur point

F_4= Proximal femur point

F_5= Proximal femur point

F_6= Proximal femur point

Femur origin: $O_F = \frac{F_1 + F_2}{2}$

Femur x_axis: $F_x = \frac{F_2 - F_1}{|F_2 - F_1|}$

Femur y_axis: $F_y = F_{z,temp} \times F_x$ where $F_{z,temp} = \frac{F_{proximal} - O_F}{|F_{proximal} - O_F|}$, $F_{proximal} = \frac{F_3 + F_4 + F_5 + F_6}{4}$

Femur z_axis: $F_z = F_x \times F_y$

Customized Full Knee Model with Experiment Coordinate Systems

After transformation of anatomical landmarks, which were collected and used for coordinate system definitions in experimentation, a full knee model will be created by following the procedures from the model development specifications[1,2]. This time experimental landmarks will be used to define anatomical joint coordinate system in the model. The tibiofemoral floating axis (FTF-axis) will be defined as the cross product between the T_z-axis and the F_x-axis at any given joint position. The patellofemoral floating axis (FPF-axis) will be defined as the cross product between the P_z-axis and the F_x-axis. The in house script FebCustomization_p3.py needs to be run for this purpose. The model will therefore be aligned with the experiment such that the axes as defined in the experiment are the same as the ones defined in the model.

Specimen-Specific Kinematics-Kinetics Data Processing

Target Outcome

Experimental kinematics-kinetics data transformed, reduced, and presented in a form amenable for simulations with the full knee model, as text files (.csv)[13] and graphs as binary images (.png)[12]. Representation of experimental kinematics-kinetics will be separated for passive flexion and for laxity data (as a function of target flexion angle, dominant degree of freedom, loading direction in the dominant degree of freedom).

Burden

None.

Protocols

Input

The original joint kinetics-kinematics files were provided in TDMS[28] format for *Model Calibration* phase[9] and needed to be processed. The files provided for re-calibration and benchmarking are already provided in the form amenable for simulations as text files(.csv)[13] and binary images (.png)[12].

Processing of Passive Flexion Data

For the *Model Calibration* phase[9], the data was provided in the original TDMS[28] format and cropped (off-axis loading in kinetics except compression-distraction is within 3N or 0.3Nm), sorted (based on increased flexion) and re-sampled (5° flexion increments by averaging each kinematics and kinetics channel where flexion angle was within 0.1°) to get in the text format (.csv)[13]. For the re-calibration process, the data was already extracted such that the data from 0° to maximum flexion was re-sampled at 5° increments, averaging data on each channel where flexion angle was within +/- 0.5 ° and provided as text files (.csv[13]).

Processing of Laxity Data

For the *Model Calibration* phase[9], the data was provided in the original TDMS[28] format and cropped (kinematics channel for flexion angle was within 1° desired flexion, relevant kinetics channel for dominant loading axis was positive and off-axis loading in any other kinetics channel was within 3 N or 0.3 Nm), sorted (based on increased loading along dominant loading axis) and re-sampled (at experimental loading intervals of dominant loading axis by averaging each kinematics and kinetics channel where dominant load was within 1 N or 0.1 Nm) to get in the text format (.csv)[13]. For the re-calibration process, the data was already extracted by finding the indices of the data points (from Kinetics.JCS.Desired) where the force was held constant for all loading cases at all degrees of flexion where laxity data was collected (0, 30, 60, 90), taking the index of the final data point for each flat section of the data and extracting the actual kinetics and kinematics data from ‘State.JCS Load’ and ‘State.Knee JCS’ using the indices. and provided as text files (.csv[13])

Processing of Benchmarking Data

For benchmarking process the data was processed such that using the desired kinetics, the data point where valgus loading was 10Nm, internal moment is 5 Nm and anterior loading is 0 were found for each tdms file (ie at each degree of flexion). The data point at each degree of flexion was saved, and then all data merged into one file (Combined_Kinetics.csv, Combined_Kinematics.csv) and provided as text files (.csv[13]).

For re-calibration and benchmarking the kinematics data was provided in experimental joint coordinate system and kinetics data was provided in experimental tibia coordinate system. Further details of data processing for *Model Calibration* phase[9] and *Model Benchmarking* phase[29] are provided in the Model Calibration Specifications and deviations[4,5] and the DESCRIPTION-DataRepresentation_OpenKnees.docx. file available with the download package at https://simtk.org/frs/?group_id=1061 and the *Model Benchmarking* site[29] respectively.

Re-calibration of In Situ Ligament Strains

Target Outcome

Full knee models with converged meshes, confirmed material properties, joint coordinate system defined to align with the experimental coordinate system, and loading and boundary conditions of experiments selected for calibration in FEBio[18] format (.feb, XML[25] based text file)[26]. Simulation results as binary and text output files (.xplt and .log, respectively)[26] and as summary of calibration process including target metric, model predictions as a function of in situ ligament strains and fit error (XML[25] based text file). Full knee model with calibrated in situ ligament strains in FEBio[18] format (.feb, XML[25] based text file)[26]. Tissues for which in situ ligament strains will be calibrated include ligaments – anterior/posterior cruciate, medial/lateral collateral.

Burden

Software requirements:

FEBio. FEBio is a nonlinear implicit finite element analysis framework designed specifically for analysis in biomechanics and biophysics (binaries custom open source license; free for academic research use, licensing for commercial use is available, see http://www.febio.org)[18].FEBio 2.9.1 will be used.

FEBio PreStrain Plugin. PreStrain Plugin provides a general framework for representing prestrain in a finite element model using a prestrain gradient method[19]. Version 1.0 will be used.

Python. Python is a high-level multi-platform programming language (free and open source GPL compatible Python Software Foundation license, see https://www.python.org)[21]. Any contemporary version available for the computing platform can be used; 3.8.0 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, version 2.7 may be used.

SciPy. SciPy is a Python based open source software platform for mathematics, science and engineering (free and open source BSD-new license, see https://www.scipy.org)[22]. Any contemporary version available for the computing platform can be used; 1.3.1 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, a version compatible with Python 2.7 may be used.

Python Scripts. There are existing Python[21] scripts used and/or developed through the *Model Development* phase[30] and *Model Calibration* phase[9]. Scripts for *Customization*, specifically for in situ ligament strains, will be reused (latest editions can be found at the source code repository at https://simtk.org/svn/openknee/app/KneeHub/src/, including revisions used for the *Model Calibration* phase[9]). These scripts will be used for additional customization for compartmental model assembly and reduction, prescription of loading and boundary conditions, extraction of metrics relevant to calibration, and automation of the iterative process to perform calibration. Python[21] scripts *experiment_to_model.py* will be used to to update a full knee model in FEBio[18] to replicate experimental conditions. Experimental kinetics are applied as external femur loads, and experiment flexion angle is prescribed to the extension-flexion joint. Python[21] script *InSituOptimization.py* will be used to update the in situ strain of the target ligament in the FEBio[18] model file, to read simulation results (displacement and load in dominant degree of freedom), to implement a scalar (one-dimensional) optimization that will minimize the sum of squared differences between model predicted and experimental loading response in the dominant degree of freedom, and to write optimization results in a text.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux.

Anticipated Man Hours and Expertise Level:

1-2 days of full-time effort from a research engineer with bachelor's degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis and scripting.

Computational Cost:

A few hours of simulation time (wall clock) for each calibration simulation; a total of 10-30 simulations.

Protocols

Input

Template FEBio model file of the full knee (.feb) and model properties (.xml) files for with converged meshes, confirmed material properties, and experiment coordinate systems; processed specimen-specific kinematics-kinetics data (.csv).

Models

Customization scripts developed for *Model Development* phase[30] and modified for *Model Calibration* phase[9], FebCustomization_p3.py, and experiment_to_model.py will be used to generate models representative of the loading and boundary conditions of selected laxity tests to calibrate in situ strains. Only joint laxity data at 0° flexion will be used to modify in situ strains for anterior cruciate ligament (ACL), posterior cruciate ligament (PCL), medial collateral ligament (MCL), and lateral collateral ligament (LCL). The decision to use only these data was made to save on computational cost and time. All other loading scenarios include flexion of the joint prior to performing laxity testing, which can be costly, and often, convergence issues may arise. This way, the calibration can be performed quickly, and the models are unlikely to have any convergence issues.

Template model (.feb)[26] and model properties (.xml)[25] reflective of converged meshes, confirmed material properties, and experiment coordinate systems will be the basis for customization of models for calibration. Customization script FebCustomization_p3.py was modified to create registered model and experiment_to_model.py was developed to incorporate experimental load cases into the model. Overall, application of loading and boundary conditions and output requests will be similar to those described in the model development specifications[1] with exceptions noted in here. Tibia will be fixed; femur and patella will be free to move and all loads and boundary conditions will be applied in one step. From time 0 to 1, in situ strain will be applied while keeping flexion at 0°. From time 1 to 2, the loads and boundary conditions at the start of experiment will be prescribed, i.e., the flexion angle will be set and the loads in the remaining degrees of freedom will be applied on femur to reflect the loading state of the joint at the start of testing. This step will account for any offsets in bone configuration between imaging and the experiment and it should be done after the prestrain step as we do not want the in situ strain calibration to be dependent on the orientation of the knee in different experiment trials. From time 2 to 3, the loads and boundary conditions of the experiment will be prescribed, i.e., the flexion angle will be constant and the loads in the remaining degrees of freedom will be applied on femur. Load curves for each degrees of freedom (particularly the dominant loading) will be defined based on experiment data points and simulation output will be requested at each experiment point. A total of 4 models will be generated:

Model Name	**Flexion (°)**	**Loading from Experiment**	**To Calibrate**
F00_AT_C	0	Anterior laxity	ACL
F00_PT_C	0	Posterior laxity	PCL
F00_VL_C	0	Valgus laxity	MCL
F00_VR_C	0	Varus laxity	LCL

ACL: anterior cruciate ligament. PCL: posterior cruciate ligament. MCL: medial collateral ligament. LCL: lateral collateral ligament.

Calibration Procedure

An iterative procedure will be used to identify optimal in situ ligament strains in anterior cruciate ligament (ACL), posterior cruciate ligament (PCL), medial collateral ligament (MCL), and lateral collateral ligament (LCL):

1. Use F00_AT_C (with previously modified ACL, PCL, MCL, LCL in situ strains, if any) to find ACL in situ strain by minimizing the difference between model predicted and experimental anterior translation and force.
2. Use F00_PT_C (with previously modified ACL, PCL, MCL, LCL in situ strains, if any) to find PCL in situ strain by minimizing the difference between model predicted and experimental posterior translation and force.
3. Use F00_VL_C (with previously modified ACL, PCL, MCL, LCL in situ strains, if any) to find MCL in situ strain by minimizing the difference between model predicted and experimental valgus rotation and moment..
4. Use F00_VR_C (with previously modified ACL, PCL, MCL, LCL in situ strains, if any) to find LCL in situ strain by minimizing the difference between model predicted and experimental varus rotation and moment.
5. Repeat steps 1-4 until convergence of in situ strains, i.e., stop when absolute change in calculated in situ strain is less than 0.001.

Customized Full Models for Post Re-calibration Simulations

Target Outcome

Customized full knee models in FEBio[18] format (.feb, XML[25] based text file)[26] prepared for all simulation cases, including passive flexion with experimental joint coordinate system and all experimental loading conditions. Models will include converged meshes, confirmed material properties and calibrated in situ ligament strains, and for reproduction of experiments, loading and boundary conditions of joint testing registered and transformed to model coordinate system.

Burden

Software requirements:

Python. Python is a high-level multi-platform programming language (free and open source GPL compatible Python Software Foundation license, see https://www.python.org)[21]. Any contemporary version available for the computing platform can be used; 3.8.0 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, version 2.7 may be used.

SciPy. SciPy is a Python based open source software platform for mathematics, science and engineering (free and open source BSD-new license, see https://www.scipy.org)[22]. Any contemporary version available for the computing platform can be used; 1.3.1 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, a version compatible with Python 2.7 may be used.

Python Scripts. There are existing Python scripts developed in *Model Calibration* phase[9]. Latest editions can be found at the source code repository at https://simtk.org/svn/openknee/app/KneeHub/src/. Experimental loading cases will be generated using in house Python script experiment_to_model.py developed in the *Model Calibration* phase[9]. This script allows updating the registered models (obtained using FebCustomization_p3.py) with the appropriate experimental loading. The registered models are customized with converged meshes and confirmed material properties and calibrated in situ ligament strains and experimental joint coordinate system.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux.

Anticipated Man Hours and Expertise Level:

2-3 days of full-time effort from a research engineer with bachelor’s degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis and scripting.

Computational Cost:

Minimal compared to required interactions with computer scripts.

Protocols

Input

Template FEBio[18] model file of the full knee (.feb)[26] and model properties (.xml)[25] files for with converged meshes, confirmed material properties, experiment coordinate systems, and calibrated in situ ligament strains; processed specimen-specific kinematics-kinetics data (.csv)[13].

Customization for Test Simulation Case

Customization scripts developed for *Model Development* phase[30], and modified in *Model Calibration* phase[9] in particular FebCustomization_p3.py will be used to generate models representative of the test simulation case (passive flexion). Loading and boundary conditions and output requests will be the same, as described in model development specifications and are briefly summarized in here. Tibia will be fixed; femur and patella will be free to move. In one step, in situ strain will be applied from time 0 to 1 while keeping flexion at 0° and flexion will be prescribed from time 1 to 2 up to 90°. Models will be generated to reflect model parameters that are modified in the *Model Calibration* (converged meshes, confirmed material properties) and *Model Benchmarking* phase (in situ strain calibration):

- with converged meshes, confirmed material properties, and calibrated in situ strains
- with converged meshes, confirmed material properties, calibrated in situ strains, and experiment coordinate systems (to be compared with experimental kinematics-kinetics)

Customization for Experiment Loading Cases

Customization scripts developed for *Model Development* phase[30], and *Model Calibration* phase[9] in particular FebCustomization_p3.py, and experiment_to_model.py will be used to generate models representative of the loading and boundary conditions of experiment laxity tests. Template model (.feb[26]) and model properties (.xml[25]) reflective of converged meshes, confirmed material properties, calibrated in situ strains, and experiment coordinate systems will be the basis. Overall, application of loading and boundary conditions and output requests will be similar to those described in the model development specifications[1,2] and updated in Model Calibration phase[9]. Tibia will be fixed; femur and patella will be free to move and all loads and boundary conditions will be applied in one step. From time 0 to 1, in situ strain will be applied while keeping flexion at 0°. From time 1 to 2, the loads and boundary conditions at the start of experiment will be prescribed, i.e., the flexion angle will be set and the loads in the remaining degrees of freedom will be applied on femur. From time 2 to 3, the loads and boundary conditions of the experimental trial will be applied until the end point of the experiment. Load curves for each degrees of freedom (particularly the dominant loading) will be defined based on experiment data points and simulation output will be requested at each experiment point. The kinematics-kinetics trajectories of experiment will be split to facilitate prescription of loading scenarios in simulations. A total of 25 models will be generated:

Model Name	**Flexion (°)**	**Loading from Experiment**
F00_AT	0	Anterior laxity
F30_AT	30	Anterior laxity
F60_AT	60	Anterior laxity
F90_AT	90	Anterior laxity
F00_PT	0	Posterior laxity
F30_PT	30	Posterior laxity
F60_PT	60	Posterior laxity
F90_PT	90	Posterior laxity
F00_IR	0	Internal rotation laxity
F30_IR	30	Internal rotation laxity
F60_IR	60	Internal rotation laxity
F90_IR	90	Internal rotation laxity
F00_ER	0	External rotation laxity
F30_ER	30	External rotation laxity

Model Name	Flexion (°)	Loading from Experiment
F60_ER	60	External rotation laxity
F90_ER	90	External rotation laxity
F00_VR	0	Varus laxity
F30_VR	30	Varus laxity
F60_VR	60	Varus laxity
F90_VR	90	Varus laxity
F00_VL	0	Valgus laxity
F30_VL	30	Valgus laxity
F60_VL	60	Valgus laxity
F90_VL	90	Valgus laxity
F90	90	Experiment passive flexion

Customized Full Models for Benchmarking

Target Outcome

Customized full knee model in FEBio[18] format (.feb, XML[25] based text file)[26] prepared for given simulation case of combined loading. Model will include converged meshes, confirmed material properties and calibrated in situ ligament strains, and for reproduction of experiment, loading and boundary conditions of joint testing registered and transformed to model coordinate system.

Burden

Software requirements:

Python. Python[21] is a high-level multi-platform programming language (free and open source GPL compatible Python Software Foundation license, see https://www.python.org)[21]. Any contemporary version available for the computing platform can be used; 3.8.0 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, version 2.7 may be used.

SciPy. SciPy is a Python based open source software platform for mathematics, science and engineering (free and open source BSD-new license, see https://www.scipy.org)[22]. Any contemporary version available for the computing platform can be used; 1.3.1 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, a version compatible with Python 2.7 may be used.

Python Scripts. There are existing Python scripts developed in *Model Calibration* phase[9]. Latest editions can be found at the source code repository at https://simtk.org/svn/openknee/app/KneeHub/src/. Experimental loading cases will be generated using in house Python script experiment_to_model.py developed in the *Model Calibration* phase[9]. This script allows updating the registered models (obtained using FebCustomization_p3.py) with the appropriate experimental loading. The registered models are customized with converged meshes and confirmed material properties and calibrated in situ ligament strains and experimental joint coordinate system.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux.

Anticipated Man Hours and Expertise Level:

2-3 hrs of full-time effort from a research engineer with bachelor's degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis and scripting.

Computational Cost:

Minimal compared to required interactions with computer scripts.

Protocols

Input

Template FEBio[18] model file of the full knee (.feb[26]) and model properties (.xml[25]) files for with converged meshes, confirmed material properties, experiment coordinate systems, and calibrated in situ ligament strains; processed specimen-specific kinematics-kinetics data (.csv[13]).

Customization for Experiment Loading Case

Python[21] script experiement_to_model.py developed during the *Model Calibration* phase[9] will be used to generate model representative of the loading and boundary conditions of experiment combined loading test. Template model (.feb[26]) and model properties (.xml[25]) reflective of converged meshes, confirmed material properties, calibrated in situ strains, and experiment coordinate systems will be the basis. Overall, application of loading and boundary conditions and output requests will be similar to those described in the model development specifications[30] and updated in *Model Calibration* phase[9]. Tibia will be fixed; femur and patella will be free to move and all loads and boundary conditions will be applied in one step. From time 0 to 1, in situ strain will be applied while keeping flexion at 0°. From time 1 to 2, the loads and boundary conditions at the start of experiment will be prescribed, i.e., the flexion angle will be set and the loads in the remaining degrees of freedom will be applied on femur. From time 2 to 3, the loads and boundary conditions of the experimental trial will be applied until the end point of the experiment. Load curves for each degrees of freedom (particularly the dominant loading) will be defined based on experiment data points and simulation output will be requested at each experiment point. The kinematics-kinetics trajectories of experiment will be split to facilitate prescription of loading scenarios in simulations.

Model Name	**Loading from Experiment**
F_VL_IR	Combined loading

Simulations

Target Outcome

Solutions of customized full knee models through finite element analysis using FEBio[18]; generating simulation results as binary and text output files (.xplt and .log, respectively)[26].

Burden

Software requirements:

FEBio. FEBio[18] is a nonlinear implicit finite element analysis framework designed specifically for analysis in biomechanics and biophysics (binaries custom open source license; free for academic research use, licensing for commercial use is available, see http://www.febio.org)[18]. The version used for the *Model Calibration* phase[9], will be used.

FEBio PreStrain Plugin. PreStrain[19] Plugin provides a general framework for representing prestrain in a finite element model using a prestrain gradient method. The version used for the *Model Calibration* phase[9] will be used.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux. Access to a high performance computing cluster can expedite simulations by running multiple finite element analysis cases in parallel.

Anticipated Man Hours and Expertise Level:

2 weeks of full-time effort from a research engineer with bachelor's degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis.

Computational Cost:

~6 hours of anticipated simulation time (wall clock) per simulation case; a total of 28 simulation cases.

Protocols

Input

Customized full models in FEBio format (.feb[26]).

Simulation Process

Invoke FEBio[18] with each customized model file as input. If a simulation does not convergence, convergence tolerances and utilization of alternative solution algorithms may need to be employed in a fashion similar to iterations conducted during the *Model Development phase*[30] and *Model Calibration* phase[9].

Post-Processing

Target Outcome

Extraction and summary of knee kinematics and kinetics of all simulation cases as text based files (.csv[13]); processed using raw simulation results of customized models with FEBio (.log file[26]), supported by graphs as binary image files (.png[12]); for simulations of experimental conditions, output files (.csv[12]) consolidated with processed joint kinematics-kinetics data and errors indicating correspondence between simulations and joint testing, supported by summary of predictive errors as text files (.xml[25]).

Burden

Software requirements:

Python. Python[21] is a high-level multi-platform programming language (free and open source GPL compatible Python Software Foundation license, see https://www.python.org)[21]. Any contemporary version available for the computing platform can be used; 3.8.0 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, version 2.7 may be used.

SciPy. SciPy is a Python based open source software platform for mathematics, science and engineering (free and open source BSD-new license, see https://www.scipy.org)[22]. Any contemporary version available for the computing platform can be used; 1.3.1 is the latest version at the time of preparation of this document. Depending on the requirements of legacy Python scripts, a version compatible with Python 2.7 may be used.

Python Scripts. –Post processing will be performed using scripts LogPostProcessing.py, described previously, for extracting joint kinematics and kinetics from the model outputs, and an in house script generated during *Model Calibration* phase[9] to compare kinematics and kinetics between models and experiment. The script model_prediction_errors.py is developed in house to calculate rms error between models and experiments kinematics-kinetics, generate graphs and save as png[12], and rms error are saved to xml[25]. To be used with Python[21], source code available at https://simtk.org/svn/openknee/app/KneeHub/src/

PostView. PostView[20] is a post-processor to visualize and analyze results from FEBio[18], finite element analysis package for biomechanics (binaries custom open source license; free for academic research use, licensing for commercial use is available, see https://febio.org/postview/). The version used for the *Model Calibration* phase[9] will be used.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software are supported on multiple platforms including Windows, Mac OS X, and Linux.

Anticipated Man Hours and Expertise Level:

1 week of full-time effort from a research engineer with bachelor’s degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis and scripting.

Computational Cost:

Minimal compared to required interactions with computer scripts.

Protocols

Input

Solutions (simulation results) of customized full models through finite element analysis using FEBio[18] as binary and text output files (.xplt and .log, respectively)[26]; processed experimental knee kinematics and kinetics as text files (.csv[13]).

Standalone Processing of Simulation Results

A Python script previously developed in the *Model Development* phase[30] (LogPostProcessing.py) will be used to read the log file and extract, store (as .csv[13]), and plot knee kinematics and kinetics during all simulation cases (as .png[12]) for both tibiofemoral and patellofemoral joints.

Processing of Simulation Results and Experimental Data

A Python script developed in the *Model Calibration* phase[9] will be used to consolidate tibiofemoral joint kinematics and kinetics of experimentation with that of simulation in a text file (.csv)[13]. (model_prediction_errors.py – in house script to calculate rms error between models and experiments kinematics-kinetics. Generates graphs and saves as png[12], and rms error are saved to xml[25]. To be used with Python[21], source code available at https://simtk.org/svn/openknee/app/KneeHub/src/) Simulation results will be reduced and/or re-sampled to match experimentation targets, e.g. load levels. Kinematics and kinetics data (3 rotations, 3 translations; 3 forces, 3 torques/moments) will be reported both in a connector based convention (constraint/reaction forces/torques, cylindrical joint movements) and rigid body based convention (forces/moments on femur/tibia, femur/tibia movement). All reporting will use anatomical local coordinate systems of the model, which would already be registered to the experimental local coordinate systems (see Registration for Specimen-Specific Calibration). All kinematics will be reported in a fashion to describe absolute pose and orientation of bones relative to each other, i.e., accounting for offsets of joint coordinate system in reference state of experiment (see Specimen-Specific Kinematics-Kinetics Data Processing) and that of model. Differences between predictions and measurements for each corresponding kinematics and kinetics channel will be reported in the same text file. For each degrees of freedom, root-mean-square error between experimental and model predicted kinematics and kinetics will be calculated and reported in a text file (.xml[25]) to summarize the predictive capacity of the model. Experimental and model predicted kinematics and kinetics will also be plotted (as .png[12]).

Visualization

PostView[20] will be used to take snapshots of the model at different flexion angles, as obtained through simulation of passive flexion. PostView[20] can also be used to inspect tissue stress-strain distributions, export data, images, and animations.

Dissemination

Target Outcome

Modeling and simulation outputs delivered to the public as a download package.

Burden

Infrastructure:

SimTK. SimTK is a free project-hosting platform for the biomedical computation community (see https://simtk.org/)[24]. Project sites at SimTK provide source code repositories, wikis to support development; and news, forums, downloads and documents sections to engage with user communities.

Anticipated Man Hours and Expertise Level:

Less than a day of full-time effort (to prepare, organize, and disseminate final package) from a research engineer with bachelor’s degree, mechanical/biomedical background, <3 years of research experience, and familiarity to public dissemination.

Protocols

All modeling and simulation outputs of the *Model Benchmarking* phase[29] will be collated as a package and uploaded to the project site of *Reproducibility in simulation-based prediction of natural knee mechanics* located at SimTK[24] (https://simtk.org/projects/kneehub/). This download package will be accessible by the public licensed under Creative Commons Attribution 4.0 International License.

Protocol Deviations

Target Outcome

Protocol deviations to model benchmarking specifications documented and delivered to the public.

Burden

Infrastructure:

SimTK. SimTK[24] is a free project-hosting platform for the biomedical computation community (see https://simtk.org/). Project sites at SimTK provide source code repositories, wikis to support development; and news, forums, downloads and documents sections to engage with user communities.

Anticipated Man Hours and Expertise Level:

For each protocol deviation, on the order of minutes of full-time effort, and for final report, less than a day of full-time effort, from a research engineer with bachelor’s degree, mechanical/biomedical background, <3 years of research experience, and familiarity to finite element analysis.

Protocols

It is anticipated that some deviations to modeling and simulation workflow, described in here as part of *Model Benchmarking* phase[29], will happen. There is also the possibility that some information on model re-calibration and benchmarking specifications may be missing. All these will be documented on an ongoing basis during the execution of the planned workflow. Final document will be submitted to the project site of *Reproducibility in simulation-based prediction of natural knee mechanics* located at SimTK (https://simtk.org/projects/kneehub/) as a publicly accessible document under Creative Commons Attribution 4.0 International License.

Overall Burden

Overall burden of the modeling and simulation workflow described in here is determined by the requirements for data, labor, software and hardware, and other infrastructure. Use of existing, publicly available data, in this case Open Knee(s)[6] data set, negates the burden for data acquisition. Software and hardware costs are associated with preparation of joint mechanics data and pre-/post-processing of simulations in a coherent manner. It is anticipated that the model re-calibration, post-calibration simulations, benchmarking and analysis of simulation results in light of experimental joint mechanics data can be performed in any contemporary computer, minimizing hardware costs. All software packages used in the modeling and simulation workflow are freely available: Python[21] and SciPy[22] – to utilize Python scripts (existing and some to be developed) for reassembly of meshes, processing of experimental data, model re-calibration, benchmarking and pre- and post-processing of models; FEBio[18] and FEBio PreStrain Plugin[19] – for finite element analysis; and PostView[20] – for visualization of simulation results. The activity will leverage SimTK[24] for public dissemination. SimTK[24] is a freely available project hosting site for biomedical computing. Labor effort will be at a minimum of 4 weeks of full time effort from a research engineer with bachelor's degree, mechanical/biomedical background, less than 3 years of research experience, and familiarity to finite element analysis. This effort level includes all data processing and modeling activities, record keeping, and dissemination. It should be noted that this estimate relies on the assumption that modeling and simulation processes complete as planned, without any significant deviations and iterations. Based on our recent experience in the *Model Calibration* phase[9], convergence problems may require iterative troubleshooting of simulations. High simulation cost (~6 hours for passive flexion) may also be a confounding factor. As a result, this timeline may extend in an agile fashion. The overall burden of the model re-calibration and benchmarking specifications should be evaluated in concert with their desired final outcome – a comprehensive and extensible knee joint model incorporating anatomical and mechanical detail of its major structures, which is capable of reproducing measured specimen-specific response.

References

1. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Open Knee(s) data: model development specifications – Cleveland Clinic approach*. (2018).
2. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Open Knee(s) data: model development protocol deviations – Cleveland Clinic approach*. (2019).
3. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Open Knee(s) data: model development outputs descriptors – Cleveland Clinic approach*. (2019).
4. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Open Knee(s) data: model calibration specifications - Cleveland Clinic approach*. (2019).
5. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Open Knee(s) Data: model calibration protocol deviations – Cleveland Clinic approach*.
6. SimTK: Open Knee(s): Virtual Biomechanical Representations of the Knee Joint: Project Home. https://simtk.org/projects/openknee.
7. ModelBenchmarking - kneehub.
8. SimTK: Reproducibility in simulation-based prediction of natural knee mechanics: Project Home. https://simtk.org/projects/kneehub.
9. ModelCalibration - kneehub. https://simtk.org/plugins/moinmoin/kneehub/ModelCalibration.
10. kneehub - Revision 50: /ModelBenchmarking/Data. https://simtk.org/svn/kneehub/ModelBenchmarking/Data/.
11. XYZ File Extension - What is it? How to open an XYZ file? https://filext.com/file-extension/XYZ.
12. Roelofs, G. PNG: The Definitive Guide. (1999).
13. Shafranovich, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files. (2005).
14. Specifications/ExperimentationJointMechanics - openknee. https://simtk.org/plugins/moinmoin/openknee/Specifications/ExperimentationJointMechanics.
15. *bioRobotics, simVitro - State Configuration File Explanations, User Manual, 2013CB-031-001.A.* https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=get&target=2013CB-031-001.A_State+Configuration+File+Explanations.pdf.
16. *Knee Joint Coordinate System v2.1*. https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=get&target=Knee+Coordinate+Systems.pdf.

17. *bioRobotics, simVitro - simVITRO Data File Contents for Open Knee, User Manual, 2013CB-031-002.B.* https://simtk.org/plugins/moinmoin/openknee/Infrastructure/ExperimentationMechanics?action=AttachFile&do=get&target=2013CB-031-002.B+simVITRO+Data+File+Contents_Open+Knee.pdf.

18. FEBio Software Suite. https://febio.org/.

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

20. PostView – FEBio Software Suite. https://febio.org/postview/.

21. Welcome to Python.org. *Python.org* https://www.python.org/.

22. SciPy.org — SciPy.org. https://www.scipy.org/.

23. npTDMS · PyPI. https://pypi.org/project/npTDMS/.

24. SimTK: Welcome. https://simtk.org/.

25. Office Open XML - What is OOXML? http://officeopenxml.com/.

26. FEBio User Manual Version 2.9. https://help.febio.org/FEBio/FEBio_um_2_9/index.html.

27. Söderkvist, I. & Wedin, P. A. Determining the movements of the skeleton using well-configured markers. *J Biomech* **26**, 1473–1477 (1993).

28. The NI TDMS File Format - National Instruments. http://www.ni.com/product-documentation/3727/en/.

29. ModelBenchmarking - kneehub. https://simtk.org/plugins/moinmoin/kneehub/ModelBenchmarking.

30. ModelDevelopment - kneehub. https://simtk.org/plugins/moinmoin/kneehub/ModelDevelopment.