Modeling and Simulation Workflow Using Natural Knee Data

Model Calibration Specifications

Cleveland Clinic Approach

Document created on November 26, 2018; last updated on December 7, 2019.

Document prepared by

Ariel Schwartz, BSc – schwara2@ccf.org, +1 (216) 445 0373

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

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

***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

> Schwartz A, Chokhandre, SK, Erdemir, A, *Modeling and simulation workflow using Natural Knee data: model calibration specifications – Cleveland Clinic approach*, December 7, 2019, Cleveland Clinic, Cleveland, Ohio, USA.

Contributions to this document by Ariel Schwartz, Snehal Chokhandre, 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 calibration specifications that are aimed for generating a calibrated model of the knee joint based on an initial working model[1–3] and an existing, additional joint testing data set from the Natural Knee Data[4,5], which was acquired for the same specimen. The proposed modeling activities are in response to the *Model Calibration* phase[6] 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)[7]. 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)[8]. These choices are primarily aimed for pragmatic, yet comprehensive, calibration of an anatomically and mechanically detailed and extensible knee joint model incorporating its major tissue structures.

Initial Working Model

Described model calibration workflow utilizes an initial working model and derivative modeling and simulation outputs generated through the *Model Development* phase[9] 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)[7]. This model was based on an existing data set from the Natural Knee Data[4,5], specifically those from specimen DU02. Development of this model was described as part of the model development specifications[1] and protocol deviations[2]. In addition to the initial working model[3], the model calibration workflow will leverage raw tissue geometries and meshes[3], and model templating and customization[1,2] to support mesh convergence studies.

Data Utilized

Described model calibration workflow utilizes Natural Knee Data[4,5], specifically those from specimen DU02. 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)[7].

Data set was part of the *Model Calibration* phase of the project[6] and can be accessed as the package Data for MC – DU02[10]. Donor specifics of specimen DU02 are:

- Right knee
- Age: 44 years
- Gender: Male
- Height: 1.83 m
- Weight: 70.31 kg
- BMI: 21.02

Described model calibration specifications utilize data extracted from the following specimen-specific mechanical testing data sets, for calibration and for post-calibration simulations:

1. DU02_INTACT_KE_AP.csv - Anterior-posterior laxity at flexion up to 120 degrees (plain text in CSV format[11]) (anterior-posterior laxity data at 0° flexion for calibration of in situ ligament strains; for simulations of anterior-posterior laxity at all flexion angles)
2. DU02_INTACT_KE_IE.csv - Internal-external rotation laxity at flexion up to 120 degrees (plain text in CSV format[11]) (for simulations of internal-external rotation laxity at all flexion angles)
3. DU02_INTACT_KE_VV.csv - Varus-valgus laxity at flexion up to 120 degrees (plain text in CSV format[11]) (varus-valgus laxity data at 0° flexion for calibration of in situ ligament strains; for simulations of varus-valgus laxity at all flexion angles)
4. DU02_INTACT_KE_Passive.csv - Passive flexion up to 120 degrees (plain text in CSV format[11]) (for passive flexion simulations)

Described model calibration specifications also utilize probed points (DU02_raw_probed_points), which were previously disseminated as part of the *Model Development* phase[9]. Details of joint mechanics testing specifics can be found at Natural Knee Data site[4].

Overview of Modeling and Simulation Processes

A previously developed three-dimensional computational model of the knee[3], specifically finite element representation of the tibiofemoral and patellofemoral joints, will be calibrated. Calibration procedures will include mesh convergence, confirmation of material properties, and calibration of in situ ligament strains relying on specimen-specific joint kinematics-kinetics data[4,5,10]. In following, 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.

A mesh convergence analysis will be performed for all deformable tissue structures of the model, specifically femoral, tibial, and patellar cartilage; menisci; anterior and posterior cruciate ligaments; medial and lateral collateral ligaments; patellar ligament; and quadriceps tendon. Raw surface geometries for each tissue[3] will be smoothed and resampled using MeshLab[12] to generate surface meshes at four different densities. SALOME[13] will be used to generate tetrahedral volume meshes[1,2]. Compartmental models for each tissue, with each mesh density, will be generated using the material properties noted in the model development specifications[1,2]. Simulations of compression will be performed on cartilage and menisci models and tension on ligament and tendon models. For ligaments, in situ strains will not be part of these simulations as we suspect that they would not influence the trajectory of mesh convergence. Structural response of the tissue, in this case reaction force, will be used as the primary metric to establish mesh convergence, i.e., 5% difference between consecutive mesh densities will indicate converged mesh. Other metrics will also be reported; contact pressure for cartilage and menisci and fiber stretch for ligaments and tendons. Meshes of tissue structures in the full knee model will be replaced with their converged counterparts. This procedure will ensure that predicted structural response of the tissue will not be influenced by mesh density, therefore minimizing uncertainties that may emerge in the contribution of tissue loads on overall joint response. For material level metrics, e.g., contact pressure and fiber stretch, convergence may not be achieved. This is not sought after at the moment as it will likely require high mesh densities that may dramatically increase computational burden. If deemed necessary, further increase of mesh density can be performed.

Material properties of all deformable tissues will be confirmed by leveraging additional resources from literature that were not used during model development[1,2]. In general, compartmental models of tissues will be built and used to predict structural response of the tissue, replicating in situ tensile testing (for ligaments and tendons) or indentation (for cartilage). Tissue material properties used in the models will be the same as those reported in model development specifications[1,2]. Gross structural response (linear stiffness) of anterior and posterior cruciate ligaments and medial and lateral collateral ligaments will be evaluated against relevant data from literature[14–16]. For patellar ligament and quadriceps tendon, gross material response (linear modulus) will be compared to additional data[17]. Indentation stiffness of lateral tibial cartilage will be confirmed against literature[18]. Fiber modulus of menisci used in model development will be compared to reported circumferential tensile modulus of medial menisci[19]. In case of discrepancies, material properties will be modified to ensure predicted tissue response remains within two standard deviations of reported values. This step will ascertain the quality of tissue material properties using a second source of information, i.e., ensuring the tissue response to be aligned with those measured in different sample populations of human knees. The literature resources were chosen based on their testing of tissues from human knees, comprehensive level of study detail and quantitative information provided, and the experience and familiarity of the modelers to tissue characterization literature. Predicted tissue responses can also be compared to additional resources, if desired.

Specimen-specific joint mechanics data will be processed to prepare for in situ ligament strain calibration and for simulation of experimental loading scenarios. Registration will be based on alignment of articular joint surface geometries measured during joint testing by probing[4,5], and those extracted from raw cartilage geometries[3]. Coherent point drift algorithm[20] will be used to obtain the transformation matrices between probed and image based articular surfaces, which will essentially provide coordinate system transformations between local bone coordinate systems of femur, tibia, and patella during joint testing and image (therefore, model) coordinate system. An initial alignment, using at least three anatomical landmarks from the probed points, and the same anatomical landmarks in the image coordinate system[21], will provide a rough registration as a starting point for the coherent point drift algorithm[20]. Anatomical landmarks collected during joint testing[4,5] will be transformed to the model to establish an anatomical joint coordinate system registered to experiments[5]. Joint kinematics-kinetics collected during passive flexion and laxity testing (anterior-posterior translation, internal-external rotation, varus-valgus)[4,5] will be processed to prepare for modeling and simulation. First, kinematics data and kinetics data will be time-synchronized by aligning the peaks of kinematics with those of kinetics. This will be done by adjusting indexing of data with the average index difference of kinematics vs kinetics peaks and will accommodate a known data limitation[6]. In following, data cropping, sorting, and resampling will be performed. For passive flexion, data points with off-axis loading (except that of flexion and compression) that are within 1 N or 0.1 Nm will be accepted as response of the unloaded knee joint. Data will be sorted in an increasing flexion angle and resampled at every 5° of flexion (up to maximum experimental value) with an averaging threshold of 0.1°. Laxity data will be split based on a combination of flexion (approximately 0°, 15°, 30°, 45°, 60°, 75°, 90°, 105°, 120°), dominant loading axis (anterior-posterior translation, internal-external rotation, varus-valgus), and loading direction (positive, negative). Data will be cropped to ensure flexion is within 1° of experimental target value and off-axis loading (except that of flexion, compression and dominant loading) is within 1 N or 0.1 Nm to extract pure laxity response (rather than combined loading). Laxity data will be resampled at experimental loading intervals of the dominant loading with an averaging threshold of 1 N or 0.1 Nm. It should be noted that experimental joint kinematics data were assumed to be provided in an absolute fashion including joint coordinate system offsets[5]. 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. Experimental joint kinematics-kinetics will be reported in its native convention[5,22] (in an absolute fashion accommodating experiment offsets) and in a convention amenable to modeling (kinematics described as cylindrical joint movements, kinetics described as loads applied on femur) accommodating both coordinate system offsets of experiment and model reference states.

A simplified calibration approach will be implemented to manage the high cost of simulations[2]. The full knee model with converged meshes, confirmed material properties, and experiment coordinate systems will be simulated to replicate a subset of laxity tests performed on the specimen at approximately 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 will document the role of model modifications on predictions of knee joint response. Cases will include models with converged meshes; with converged meshes and confirmed material properties; with converged meshes, confirmed material properties, and calibrated in situ ligament strains; and with all modifications including experiment coordinate systems. Laxity testing simulations will aim for in silico reproduction of laxity 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 54 cases will be simulated: a combination of flexion angles at which laxity tests were performed (approximately 0°, 15°, 30°, 45°, 60°, 75°, 90°, 105°, 120°), axis of loading (tested degrees of freedom - anterior-posterior translation, internal-external rotation, varus-valgus), direction of loading (positive, negative).

FEBio[23], along with FEBio PreStrain Plugin[24], will be used to conduct finite element analysis (solid mechanics, based on implicit static solver). Simulation results will be visualized using PostView[25]. Specimen-specific joint mechanics data and predicted kinematics-kinetics of the joints will be processed to report joint movements in all loading cases. Python[26] and SciPy[27], along with auxiliary Python packages[28], 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[29].

Detailed Modeling and Simulation Outputs

Model Calibration 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**
Smooth Geometries	Geometric reconstruction of tissues of interest as triangulated watertight surface representations, with several mesh densities for each tissue. Created from raw geometry (output of *Model Development* phase[3]) after applying volume-preserving smoothing procedures.	.stl[30]
Connectivity	XML based text file which specifies which geometry files will be used in the model, and the ties and contacts between tissues to be included in the model (used as input for the mesh generation, and template model generation scripts).	.xml[31]
Meshes	Finite element meshes of tissues of interest as triangulated surface (bones) or tetrahedral volume meshes (cartilage, menisci, ligaments, tendon) in binary format, with several mesh densities for each tissue including converged mesh densities. Created from smooth geometry with node, face, element sets to facilitate model assembly, material property definitions, and assignment of tissue interactions.	.med[32]
Template Models (FEBio Input File)	XML based text file (for finite element analysis with FEBio[23]) including mesh definitions, template constitutive models (rigid – bones; deformable – other tissue), template interactions between tissue, and template loading and boundary conditions (for rigid objects); including templates for mesh convergence, material property confirmation, and in situ ligament strain calibration, and post-calibration simulations of experiment conditions.	.feb[33]
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 mesh convergence, material property confirmation, and in situ ligament strain calibration, and post-calibration simulations of experiment conditions.	.xml[31]
Customized Models (FEBio Input File)	XML based text file (for finite element analysis with FEBio[23]) 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 mesh convergence, material property confirmation, and in situ ligament strain calibration, and post-calibration simulations of experiment conditions.	.feb[33]

File	Description	File Format
Raw Simulation Results	Binary (.xplt) and text files (.log) obtained by simulation of passive flexion using calibrated customized model with FEBio[23]; in addition, results of mesh convergence, material property confirmation, and in situ ligament strain calibration, post-calibration simulations of experiment conditions.	.xplt[33] .log[33]
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 mesh convergence, material property confirmation, and in situ ligament strain calibration, post-calibration simulations of experiment conditions.	.csv[11] .png[34]
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[31]
Processed Kinematics-Kinetics Data	CSV based text files storing experimental knee kinematics and kinetics processed for use in calibrated model, as loading and boundary conditions and to assess predictive capacity; supported by graphs as binary image files, including all experiment conditions.	.csv[11] .png[34]
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[11] .xml[31]
Calibration Results	XML based text files summarizing target parameters and fit error before and after calibration.	.xml[31]

Workflow

Mesh Convergence

Target Outcome

Geometric reconstruction of tissues of interest as smooth and watertight triangulated surface representations (.stl)[30] and finite element meshes (.med)[32] with several mesh densities including converged mesh densities. Compartmental models of tissues of interest in FEBio[23] format (.feb, XML[31] based text file) for mesh convergence simulations. Simulation results as binary and text output files (.xplt and .log, respectively)[33] and as summary of mesh convergence analysis including target convergence metric as a function of mesh density (XML[31] based text file). Full knee model with converged meshes in FEBio[23] format (.feb, XML[31] based text file)[33]. Tissues for which mesh convergence will be conducted include cartilage – femoral, tibial (medial & lateral), patellar; menisci – medial & lateral, ligaments – anterior/posterior cruciate, medial/lateral collateral, patellar; tendons – quadriceps.

Burden

Software requirements:

MeshLab. MeshLab is an open source, portable, and extensible system for processing and editing of unstructured 3D triangular meshes (free and open source GPL license, see http://www.meshlab.net/)[12]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (MeshLab 2016.12 at the time of preparation of this document), will be used.

SALOME. SALOME is an open-source software that provides a generic platform for pre- (cad, meshing) and post-processing for numerical simulation (free and open source LGPL license, see http://www.salome-platform.org/)[13]. SALOME includes built-in scripting functionality using Python[26], which will be required to utilize Python scripts for mesh generation and annotation, and model assembly. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (SALOME 9.3 at the time of preparation of this document), will be used.

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)[23]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (FEBio 2.9.1 at the time of preparation of this document), will be used.

PostView. PostView is a post-processor to visualize and analyze results from FEBio, 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/)[25]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (PostView 2.4.4 at the time of preparation of this document), 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)[26]. 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)[27]. 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[26] scripts used and/or developed through the *Model Development* phase[1,2]. Scripts for *Template Model Generation*, specifically those for *Mesh Generation and Annotation* and *Model Assembly*, will be reused, along with any related *Customization* scripts (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 Development* phase[1,2])[35]. Additional Python scripts, yet to be developed, will assist transfer of mesh region definitions, customization for compartmental model assembly and reduction, and prescription of loading and boundary conditions and extraction of metrics relevant to mesh convergence analysis.

Hardware requirements:

Any contemporary computer; desktop, workstation, or laptop. All aforementioned software, with the exception of SALOME, are supported on multiple platforms including Windows, Mac OS X, and Linux. While Windows version of SALOME exists, Linux is preferable.

Anticipated Man Hours and Expertise Level:

2-4 weeks of full-time effort from a research engineer with bachelor’s degree, mechanical/biomedical background, <3 years of research experience, and familiarity to geometry generation, meshing, finite element analysis, and scripting.

Computational Cost:

A few minutes of simulation time (wall clock) for each compartmental model simulation; a total of ~80 simulations.

Protocols

Input

Raw triangulated surface representations of tissues of interest (without filtering and smoothing) in .stl format in image coordinate system.

Geometry Generation Procedures

For each tissue of interest, several geometries will be created at different mesh densities. Smoothing procedures will be performed as described in model development specifications[1], taking into account any protocol deviations[2]. The Iso Parameterization Remeshing phase will be repeated several times, to obtain the mesh densities specified below.

	Iso Parameterization Sampling Rates
Femoral Cartilage	8,10,15,20
Tibial Cartilage	6,8,10,15
Patellar Cartilage	6,8,10,15
Menisci	4,6,8,10
ACL	4,6,8,10
PCL	3,4,6,8
Patellar Ligament	4,6,8,10
Quadriceps Tendon	4,6,8,10
MCL	5,7,9,11
LCL	3,4,6,8

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

Mesh Generation Procedures

For each tissue geometry that was generated above, a mesh will be created in Salome. A new MED file will be generated where all node/element/face groups, which were already created for original tissue mesh in the *Model*

Development phase[1,2], will be transferred to the new MED file. This will be done in a scripted fashion, by treating the groups as a binary field and applying that binary field to the new mesh to find the correspondence of nodes and elements in the new mesh. This correspondence will be used to select nodes and elements to generate groups with the new node and element sets. A quality assurance check should be completed here to ensure that all node/element/face groups are as expected. For example, a check should be done for all ligament insertion origins that the insertion area covers the entire width of the ligament where it connects to the bone.

Model Generation Procedures

Each of the tissues of interest will be tested by creating a model with the boundary conditions outlined in the table below. The models will be created for each mesh density, such that 4 models exist for each tissue, with the only difference being the density of the mesh of the tissue of interest. For all other parts of the model, the MED file used be the one from the *Model Development* phase outputs package[3]. Model generation will be performed according to the model development specifications[1,2] (using in home scripts MedToFebio.py, FebCustomization.py), and boundary conditions will be updated manually in the FEBio model input files for each model.

Tissue of Interest	**Included Parts**	**Boundary Conditions***	**Simulation Outputs**
Femoral Cartilage	FMC, FMB, TBC-L, TBC-M	Compression of the FMC between the FMB (rigid) and the TBC-L, TBC-M modeled as rigid bodies. TBC fixed, FMB displaced 2 mm in -z direction.	Z-reaction force in FMB (primary) Contact pressure (secondary)
Tibial Cartilage	1. FMB, TBC-L, TBB 2. FMB, TBC-M, TBB	Compression of the Tibial Cartilage between the TBB (rigid) and the FMC modeled as a rigid body. TBB fixed, FMC displaced 2 mm in the -z direction. Test will be applied to TBC-M, TBC-L separately.	Z-reaction force in FMC (primary) Contact pressure (secondary)
Patellar Cartilage	PTC, PTB, FMC	Compression of the PTC between the PTB (rigid) and the FMC modeled as a rigid body. FMC fixed, PTB displaced 2mm in the y direction.	Y-reaction force in PTB (primary) Contact pressure (secondary)
Menisci	1. MNS-M, FMC, TBC-M 2. MNS-L, FMC, TBC-L	Compression of the MNS between the FMC and TBC, both modeled as rigid bodies. TBC fixed, FMC displaced 2 mm in the -z direction. Test will be applied to MNS-M, MNS-L separately.	Z-reaction force in FMC (primary) Contact pressure (secondary) Fiber stretch (secondary)
ACL	ACL, FMB, TBB	Tension test. TBB fixed, FMB displaced 3 mm in the z direction.	FMB reaction force (primary) Fiber stretch (secondary)
PCL	PCL, FMB, TBB	Tension test. TBB fixed, FMB displaced 3 mm in the z direction.	FMB reaction force (primary)

Tissue of Interest	**Included Parts**	**Boundary Conditions***	**Simulation Outputs**
			Fiber stretch (secondary)
Patellar Ligament	P TL, PTB, TBB	Tension test. TBB fixed, PTB displaced 4 mm in the z direction.	Z-reaction force in PTB (primary) Fiber stretch (secondary)
Quadriceps Tendon	QAT, PTB, QSO (rigid body connected to QAT)	Tension test. QSO fixed, PTB displaced 4 mm in the z direction.	Z-reaction force in PTB (primary) Fiber stretch (secondary)
MCL	MCL, TBB, FMB	Tension test. TBB fixed, FMB displaced 4 mm in the z direction	Z-reaction force in FMB (primary) Fiber stretch (secondary)
LCL	LCL, TBB, FBB	Tension test. FBB fixed, FMB displaced 4 mm in the z direction	Z-reaction force in FMB (primary) Fiber stretch (secondary)

FMB: femur bone. TBB: tibia bone. FBB: fibula bone. PTB: patella bone. FMC: femoral cartilage. TBC-L: tibial lateral cartilage. TBC-M: tibial medial cartilage. PTC: patellar cartilage. MNS-L: lateral meniscus. MNS-M: medical meniscus. ACL: anterior cruciate ligament. PCL: posterior cruciate ligament. MCL: medial collateral ligament. LCL: lateral collateral ligament. PTL: patellar ligament. QAT: quadriceps tendon. QSO: quadriceps origin.

*Displacement levels are chosen to induce approximately 10% nominal strain on target tissue.

Mesh Convergence Procedures

Each of the models created above will be run in FEBio, beginning with the coarsest mesh density. The simulation will be repeated, each time with a finer mesh. The primary measured outputs will be compared with those of the previous simulation, and when there is less than 5% difference, convergence will be assumed. The mesh density at which the output converged will then be used in the final model. If no convergence is found within the specified mesh densities, finer/coarser mesh densities can be created to continue testing until convergence is found. Dependency of secondary outputs on mesh density will also be reported for completeness.

Template Full Knee Model with Converged Meshes

After determining the optimum mesh density for each of the tissues, a full knee model will be created by following the procedures from the model development specifications[1,2]. In summary, In summary, the in house scripts MedToFebio.py, and FebCustomization.py need to be run including all of the selected knee tissues with converged mesh densities.

Confirmation of Material Properties

Target Outcome

Compartmental models of tissues of interest in FEBio[23] format (.feb, XML[31] based text file)[33] for simulations to confirm and modify material properties using converged meshes. Simulation results as binary and text output files (.xplt and .log, respectively)[33] and as summary of calibration process including target metric, model predictions as a function of material property and fit error (XML[31] based text file). Full knee model with confirmed and modified material properties in FEBio[23] format (.feb, XML[31] based text file)[33]. Tissues for which material properties will be confirmed include cartilage – femoral, tibial (medial & lateral), patellar; menisci – medial & lateral, ligaments – anterior/posterior cruciate, medial/lateral collateral, patellar; tendons – quadriceps.

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)[23]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (FEBio 2.9.1 at the time of preparation of this document), 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)[26]. 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)[27]. 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[26] scripts used and/or developed through the *Model Development* phase[1,2]. Scripts for *Customization*, specifically for material properties, 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 Development* phase[1,2])[35]. Additional Python scripts, yet to be developed, will assist any 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 material property modification.

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:

2-5 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 minutes of simulation time (wall clock) for each compartmental model simulation; a total of 25-50 simulations.

Protocols

Input

FEBio model file of the full knee with converged meshes, including all tie/contact surfaces.

Compartmental Modeling of Structural Tissue Behavior

In order to confirm and modify material properties, structural tissue response or gross material behavior will be compared with literature. If, the tissue behavior falls outside the range of expected behavior, the material properties will be adjusted until the mechanical response of the tissue is considered within the reported normal range. This process will essentially confirm of each tissue material property used for model development[1,2] with a second information resource. Details of the simulations (and/or analyses) to be performed are provided below. An in house script ModelReduction.py, will be used to reduce the full knee model to include only the desired components for each of the test simulation cases.

Ligaments and Tendons

Tissue to Calibrate	**Included Parts**	**Boundary Conditions***	**Simulation Outputs**	**Expected Behavior**
ACL	ACL, FMB, TBB	Tension: TBB fixed, FMB displaced 3 mm in the fiber direction of the ACL.	FMB reaction force-displacement curve	Linear stiffness = 242±28 N/mm[14]
PCL	PCL, FMB, TBB	Tension: TBB fixed, FMB displaced 3 mm in the fiber direction of the PCL.	FMB reaction force-displacement curve	Linear stiffness = 258±62 N/mm[15]
Patellar Ligament	PTL, PTB, TBB	Tension: TBB fixed, PTB displaced 5 mm in the fiber direction of the PTL.	PTB reaction force-displacement curve	Linear modulus = 337.8.±67.7 MPa[17]
Quadriceps Tendon	QAT, PTB, QSO	Tension: QSO fixed, PTB displaced 5 mm in the fiber direction of the QAT.	PTB reaction force-displacement curve	Linear modulus = 255.3±64.1 MPa[17]
MCL	TBB, MCL, FMB	Tension: TBB fixed, FMB displaced 5 mm in the fiber direction of the MCL.	FMB reaction force-displacement curve	Linear stiffness = 63±14 N/mm[16]
LCL	FBB, FMB, LCL	Tension: FBB fixed, FMB displaced 5 mm in the fiber direction of the LCL	FMB reaction force-displacement curve	Linear stiffness = 59±12 N/mm[16]

FMB: femur bone. TBB: tibia bone. FBB: fibula bone. PTB: patella bone. ACL: anterior cruciate ligament. PCL: posterior cruciate ligament. MCL: medial collateral ligament. LCL: lateral collateral ligament. PTL: patellar ligament. QAT: quadriceps tendon. QSO: quadriceps origin.

*Displacement levels are chosen to induce approximately 10% nominal strain on target tissue. No prestrain will be applied.

For the tension test simulations in the fiber direction, a load curve will be defined from 0 to X, where X is the desired total displacement. Then, in the boundary section the prescribed displacement will be specified in the fiber direction. For example, given a fiber direction of [-0.101,-0.511,0.854] a unit displacement (X=1) will be:

```
<rigid_body mat="4">
        <prescribed bc="x" lc="3">-0.101</prescribed>
        <prescribed bc="y" lc="3">-0.511</prescribed>
        <prescribed bc="z" lc="3">0.854</prescribed>
        <fixed bc="Rx"/>
        <fixed bc="Ry"/>
        <fixed bc="Rz"/>
</rigid_body>
```

A Python script will be developed to extract rigid body reaction force – displacement data from simulation output (.log). In following, force-displacement response of the tissue along its fiber direction will be calculated. Linear stiffness of the tissue (k) will be calculated by fitting a line to the linear portion (upper third) of the tissue's simulated structural response. Linear modulus (E) will be calculated as (k x L_o / A_o), where L_o and A_o are the reference ligament length and cross sectional area, respectively. The reference length will be calculated as the distance between the centers of the insertion and origin. The cross sectional area will be calculated as the average cross sectional area between the insertion and origin. If the simulated tissue properties (k and/or E) fall outside two standard deviations of the expected behavior, the fiber modulus (C5) will be scaled to bring the tissue response within the expected range.

Cartilage

Since we assume cartilage properties to be consistent for all cartilage tissues, we will perform one indentation test to confirm if cartilage structural response is within what is reported in literature. For this purpose, we will use indentation stiffness of lateral tibial cartilage, which was reported as 20.38±5.32 N/mm[18]. A model will be generated to reproduce the experiment conditions[18]. The model will include the tibia bone (TBB), lateral tibial cartilage (TBC-L), and a 1mm diameter indenter as a rigid body, which will be in frictionless contact with the cartilage. The indenter will be placed above the cartilage near the meniscus, as described in the study[18], and a load of 0.5 N will be applied to force the indenter against the tibial cartilage. A Python script will extract indenter force and displacement data from simulation results (.log) and the linear stiffness will be calculated using the linear region (upper third) of the force-displacement curve. Material properties of cartilage (C1) will be scaled as needed to get the range within two standard deviations of the reported stiffness value. Bulk modulus parameter (K) will likely need to be scaled accordingly.

Meniscus

Depending on the location of the sample (anterior, central, posterior) and the thickness of it, circumferential tensile modulus of medial meniscus was reported as 43.4±26.8 MPa to 141.2±56.7 MPa[19]. Fiber modulus (C5) of meniscus will be scaled to match within two standard deviations of the reported modulus.

Customized Full Knee Model with Confirmed Material Properties

After confirming and modifying material properties of tissues, a full knee model will be created by following the procedures from the model development specifications[1,2]. In summary, the in house script FebCustomization.py need to be run including all of the selected knee tissues with updated material properties.

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[31] based text files. Full knee model with joint coordinate system defined to align with the experimental coordinate system, in FEBio[23] format (.feb, XML[31] based text file)[33].

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)[26]. 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)[27]. 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.

PyCPD. PyCPD is an implementation of coherent point drift algorithm[20] for registration of point clouds including i) scale and rigid registration, ii) affine registration, and iii) Gaussian regularized non-rigid registration (free and open source MIT license, see https://pypi.org/project/pycpd/)[28]. Any contemporary version available for the computing platform can be used; 1.0.5 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.

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 weeks 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 articular cartilage surfaces and anatomical landmarks[9] and raw cartilage geometries from imaging[3]; FEBio model file of the full knee with converged meshes and confirmed material properties.

Registration

Probed points on articular surfaces were provided in the *Model Development* phase[9]. These probed points will be used to find a transformation matrix to transform from each bone coordinate system to the image coordinate system. Coherent point drift algorithm (CPD)[20] will be used to get a rigid transformation to align two point clouds using Python[28]. For each bone, a set of probed points on the articular surfaces are chosen that are similar to a tissue anatomy that was segmented, in this case cartilage. The probed points, and vertices of raw surface geometries will be used as source and target point clouds, respectively, for registration. First, an initial "rough" registration will be done by choosing at least three anatomical landmarks that were probed during experiments, and the same anatomical landmarks in the image coordinate system. These landmarks do not need to perfectly align, they should just be chosen at roughly the same area on the bone. Singular value decomposition will be performed on the point sets to get an initial "rough" transformation matrix (T_i)[21]. This rough transformation matrix will be applied to the source point cloud. Next, the CPD algorithm will be applied to the transformed source, and target. This provides a final registration of the point clouds to minimize error while aligning the two point clouds (T_f). The overall transformation from the bone coordinate system to image coordinate system will be calculated as $T = T_\mathrm{f}\ T_\mathrm{i}$. These will all be performed in an in house Python script, register_probed_points.py, where the required inputs are the source probed points, target probed points, source anatomical landmarks, target anatomical landmarks. The following data will be used for the calculations for each bone:

Bone to Register	**Source Probed Points**[a]	**Target Geometry File**[b]	**Source Anatomical Landmarks**[c]	**Target Anatomical Landmarks**[d]
Femur	DU02_fem_artgeo.txt	natural_knee_FMC_SKC_01_RAW.stl	points 5, 4, 3 from DU02_fem_GPS.txt DU02_HipBall.txt[e]	LFC, MFC, DFP from ModelProperties.xml
Tibia	DU02_tib_artgeo.txt	natural_knee_TBC-L_SKC_01_RAW.stl, natural_knee_TBC-M_SKC_01_RAW.stl	points 4, 2, 3, 1 from DU02_tib_GSPts.txt	LTS, LTP, MTS, MTP from ModelProperties.xml
Patella	DU02_Pat_artgeo.txt	natural_knee_PTC_SKC_01_RAW.stl	Points 4,3[f] from DU02_Pat_GSPts.txt	LPR, MPR from ModelProperties.xml

[a] Probed points on articular cartilage surfaces were provided as part of *Model Development* phase[9].

[b] Raw geometry files are available as part of *Model Development* phase outputs[3].

[c] Data for anatomical landmarks were provided as part of *Model Development* phase[9]. The order of the points is

important as the source and target points need to align.

[d] These anatomical landmarks are used for customization of the model to establish coordinate systems[1,2]. LFC: lateral femoral condyle. MFC: medial femoral condyle. DFP: distal femoral posterior. LTS: lateral tibial spine. LTP: lateral tibial plateau. MTS: medial tibial spine. MTP: medial tibial plateau. LPR: lateral patellar ridge. MPR: medial patellar ridge. ModelProperties.xml is an input file used for customization of the knee model[2,3].

[e] This data set may need to be used to obtain hip center by fitting a sphere.

[f] Only two points were chosen here because there were no other target anatomical landmarks to compare with. However, the algorithm was tested and it still worked using only the two points.

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[5]: probed bony landmarks on the tibia (medial and lateral dwell points of the plateau, proximal tip of the medial spine of the tibial eminence, center of the distal intramedullary canal) and femur (posterior of medial and lateral condyles, distal point between condyles, hip ball center) will used to establish the same local coordinate systems for the femur and tibia as in joint mechanical testing. Patella coordinate system will be updated based on the experimental landmarks as well.

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.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)[11] and graphs as binary images (.png)[34]. 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

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)[26]. 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)[27]. 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.

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 weeks 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 joint kinematics-kinetics data (.csv); coordinate system transformation matrices between joint testing and imaging, and experimental anatomical landmarks transformed to model coordinate system (.xml).

Processing of Passive Flexion Data

Experimental passive flexion data file (DU02_INTACT_KE_Passive.csv) will be processed. Kinematics data will be extracted from tibiofemoral joint, "TF", labeled columns in the data files, which provide joint kinematics in an anatomical joint coordinate system (defined in the experiment based on cylindrical joints)[5] presumably including offsets of joint coordinate system. Kinetics data will be extracted from tibiofemoral joint, "TF", labeled columns in the data files, which presumably provides joint kinetics in an anatomical tibia coordinate system as applied as loads on tibia[5]. A Python script will be developed for extraction, processing, and storage of passive data:

1. Resynchronize kinematics-kinetics data to accommodate for a known limitation of the data set[6]; align the peaks of kinematics (flexion) and kinetics (superior-inferior force) by adjusting indexing of data with the average index difference of kinematics vs kinetics peaks.
2. Crop kinematics and kinetics data such that off-axis loading in kinetics channels other than flexion and joint compression-distraction (where forces are applied for stabilization) is within 1 N or 0.1 Nm – an indication of pure passive flexion, i.e. unloaded knee joint response.
3. Sort kinematics and kinetics data based on increased flexion.
4. Resample kinematics and kinetics data at 5° flexion increments by averaging each kinematics and kinetics channel where flexion angle is within 0.1°.

5. Report bone pose and orientation in an absolute fashion.
6. Transform kinematics data to the convention used in the model, i.e., cylindrical joint translations and rotations, accommodating offsets at model reference state when reconstructing experiment coordinate systems in the model.
7. Transform kinetics data to the convention used in the model, i.e., joint loading applied to femur in model coordinate system, which is registered and aligned to experiment coordinate system.
8. Write kinematics and kinetics to a text based file (.csv) both in experiment and model conventions; plot and store as graphics files (.png).

Thresholds for cropping and resampling of data may change depending on data quality, e.g., noise and errors due to manual application of loads may become apparent during analysis.

Processing of Laxity Data

Kinematics-kinetics data for laxity will be split into files based on flexion angle (approximately 0°, 15°, 30°, 45°, 60°, 75°, 90°, 105°, 120°), dominant loading axis (anterior-posterior translation, internal-external rotation, varus-valgus), and direction of loading axis (positive, negative). Kinematics data will be extracted from tibiofemoral joint, “TF”, labeled columns in the data files, which provide joint kinematics in an anatomical joint coordinate system (defined in the experiment based on cylindrical joints)[5]. Kinetics data will be extracted from from tibiofemoral joint, “TF”, labeled columns in the data files, which presumably provides joint kinetics in an anatomical tibia coordinate system as applied as loads on tibia[5]. A Python script will be developed for extraction, processing, and storage of laxity data:

1. Identify target flexion angle, dominant loading axis (of laxity testing), direction of loading of interest.
2. Use the data file (.csv) relevant to the desired dominant loading axis.
3. Resynchronize kinematics-kinetics data to accommodate for a known limitation of the data set[6]; align the peaks of kinematics (of dominant axis) and kinetics (of dominant axis) by adjusting indexing of data with the average index difference of kinematics vs kinetics peaks.
4. Crop kinematics and kinetics data such that
    1. relevant kinematics channel for flexion angle is within 1° flexion (for quality assurance), and
    2. relevant kinetics channel for dominant loading axis is positive (or negative) based on the desired direction of loading, and
    3. off-axis loading in any other kinetics channel (other than flexion, dominant loading axis, and joint compression-distraction, where forces are applied for stabilization) is within 1 N or 0.1 Nm – an indication of isolated loading of degree of freedom tested for laxity.
5. Sort kinematics and kinetics data based on increased loading along dominant loading axis.
6. Resample kinematics and kinetics data at experimental loading intervals of dominant loading axis by averaging each kinematics and kinetics channel where dominant load is within 1 N or 0.1 Nm.
7. Report bone pose and orientation in an absolute fashion.

8. Transform kinematics data to the convention used in the model, i.e., cylindrical joint translations and rotations, accommodating offsets at model reference state when reconstructing experiment coordinate systems in the model.
9. Transform kinetics data to the convention used in the model, i.e., joint loading applied to femur in model coordinate system, which is registered and aligned to experiment coordinate system.
10. Write kinematics and kinetics to a text based file (.csv) both in experiment and model conventions; plot and store as graphics files (.png).

Thresholds for cropping and resampling of data may change depending on data quality, e.g., noise and errors due to manual application of loads may become apparent during analysis. Target flexion angles may deviate from desired values depending on the levels achieved in experiments.

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[23] format (.feb, XML[31] based text file)[33]. Simulation results as binary and text output files (.xplt and .log, respectively)[33] and as summary of calibration process including target metric, model predictions as a function of in situ ligament strains and fit error (XML[31] based text file). Full knee model with calibrated in situ ligament strains in FEBio[23] format (.feb, XML[31] based text file)[33]. Tissues for which in situ ligament strains will be calibrated include ligaments – anterior/posterior cruciate, medial/lateral collateral.

Burden

<u>Software requirements:</u>

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)[23]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (FEBio 2.9.1 at the time of preparation of this document), 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[24]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (version 1.0 at the time of preparation of this document), 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)[26]. 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)[27]. 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[26] scripts used and/or developed through the *Model Development* phase[1,2]. 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 Development* phase[1,2])[35]. Additional Python scripts, yet to be developed, will assist any 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.

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 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 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[1,2], in particular FebCustomization.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[2]. This way, the calibration can be performed quickly, and the models are unlikely to have any convergence issues.

Template model (.feb) and model properties (.xml) reflective of converged meshes, confirmed material properties, and experiment coordinate systems will be the basis for customization of models for calibration. Modifications to the customization script and/or additional scripts will likely be necessary to implement the loading scenarios. Overall, application of loading and boundary conditions and output requests will be similar to those described in the model development specifications[1,2] 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.

A script will be developed to update the in situ strain of the target ligament in the model, 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 file (.xml). It should be noted that these calculations will be performed with readily aligned kinematics-kinetics conventions of the model and experiment, i.e., following registration, kinematics-kinetics data processing, and accounting for experimental local coordinate systems offsets.

Customized Full Models

Target Outcome

Customized full knee models in FEBio[23] format (.feb, XML[31] based text file)[33] prepared for all simulation cases, including passive flexion with joint coordinate system of the *Model Development* phase[1,2] 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)[26]. 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)[27]. 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[26] scripts used and/or developed through the *Model Development* phase[1,2]. Scripts for *Customization*, specifically for full model customization with converged meshes and confirmed material properties and calibrated 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 Development* phase[1,2])[35]. Additional Python scripts, yet to be developed, will assist any additional customization for prescription of desired loading and boundary conditions and updated model joint coordinate system aligned with those of the experiments.

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

Customization for Test Simulation Case

Customization scripts developed for *Model Development* phase[1,2], in particular FebCustomization.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[2] 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* phase:

- with converged meshes
- with converged meshes and confirmed material properties
- 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[1,2], in particular FebCustomization.py, will be used to generate models representative of the loading and boundary conditions of experiment laxity tests. Template model (.feb) and model properties (.xml) reflective of converged meshes, confirmed material properties, calibrated in situ strains, and experiment coordinate systems will be the basis. Modifications to the customization script and/or additional scripts will likely be necessary to implement the loading scenarios. Overall, application of loading and boundary conditions and output requests will be similar to those described in the model development specifications[2] 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. 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 54 models will be generated based on the following model naming convention:

F###_$$

where

###: 000, 015, 030, 045, 060, 075, 090, 105, and 120 corresponding to flexion angles of 0°, 15°, 30°, 45°, 60°, 75°, 90°, 105°, and 120°, respectively.

$$: AT, PT, IR, ER, VR, and VL corresponding to anterior laxity, posterior laxity, internal rotation laxity, external rotation laxity, varus laxity, and valgus laxity, respectively.

It should be noted that cost of simulations can be decreased by storing in situ strain and passive flexion simulation results as restart files. However, restart attempts using FEBio with the prestrain plug-in have not been successful at the moment.

Simulations

Target Outcome

Solutions of customized full knee models through finite element analysis using FEBio[23]; generating simulation results as binary and text output files (.xplt and .log, respectively)[33].

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)[23]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (FEBio 2.9.1 at the time of preparation of this document), 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[24]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (version 1.0 at the time of preparation of this document), 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:

2-4 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:

~2 hours of anticipated simulation time (wall clock) per simulation case[2]; a total of 58 simulation cases.

Protocols

Input

Customized full models in FEBio format (.feb).

Simulation Process

Invoke FEBio 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[2].

Post-Processing

Target Outcome

Extraction and summary of knee kinematics and kinetics of all simulation cases as text based files (.csv)[11]; processed using raw simulation results of customized models with FEBio (.log file)[33], supported by graphs as binary image files (.png)[34]; for simulations of experimental conditions, output files (.csv)[11] 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)[31].

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)[26]. 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)[27]. 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[26] scripts used and/or developed through the *Model Development* phase[1,2]. Scripts for *Post-Processing* 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 Development* phase[1,2])[35]. Additional Python scripts, yet to be developed, will assist consolidation of simulation results with experimental kinematics-kinetics and calculation of prediction errors.

PostView. PostView is a post-processor to visualize and analyze results from FEBio, 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/)[25]. The version used for the *Model Development* phase[1,2], or if necessary, the latest version (PostView 2.4.4 at the time of preparation of this document), 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 as binary and text output files (.xplt and .log, respectively); processed experimental knee kinematics and kinetics as text files (.csv).

Standalone Processing of Simulation Results

A Python script previously developed in the *Model Development* phase[1,2] (LogPostProcessing.py) will be used to read the log file and extract, store (as .csv), and plot knee kinematics and kinetics during all simulation cases (as .png) for both tibiofemoral and patellofemoral joints.

Processing of Simulation Results and Experimental Data

A Python script will be developed to consolidate tibiofemoral joint kinematics and kinetics of experimentation with that of simulation in a text file (.csv). Simulation results will be reduced and/or resampled 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) to summarize the predictive capacity of the model. Experimental and model predicted kinematics and kinetics will also be plotted (as .png).

Visualization

As done for the *Model Development* phase[1], PostView will be used to take snapshots of the model at different flexion angles, as obtained through simulation of passive flexion. PostView 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/)[29]. 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 Calibration* phase 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[29] (https://simtk.org/projects/kneehub/)[7]. This download package will be accessible by the public licensed under Creative Commons Attribution 4.0 International License[36].

Protocol Deviations

Target Outcome

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

Burden

Infrastructure:

SimTK. SimTK is a free project-hosting platform for the biomedical computation community (see https://simtk.org/)[29]. 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 Calibration* phase, will happen. There is also the possibility that some information on model calibration specifications may be missing. All these will be documented on an ongoing basis during the execution of the planned workflow[6]. Final document will be submitted to the project site of *Reproducibility in simulation-based prediction of natural knee mechanics* located at SimTK[29] (https://simtk.org/projects/kneehub/)[7] as a publicly accessible document under Creative Commons Attribution 4.0 International License[36].

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 Natural Knee Data, 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 calibration, post-calibration simulations, 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: MeshLab[12] and SALOME[13] – for generation of meshes with different densities to support mesh convergence analysis; Python[26] and SciPy[27] – to utilize Python scripts (existing and some to be developed) for reassembly of meshes, processing of experimental data, model calibration, and pre- and post-processing of models; FEBio[23] and FEBio PreStrain Plugin[24] – for finite element analysis; and PostView[25] – for visualization of simulation results. The activity will leverage SimTK for public dissemination. SimTK is a freely available project hosting site for biomedical computing[29]. Labor effort will be at a minimum of 8 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 Development* phase[1,2], convergence problems may require iterative troubleshooting of simulations. High simulation cost (~2 hours for passive flexion[2]) may also be a confounding factor. As a result, this timeline may extend in an agile fashion. The overall burden of the model calibration 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 Natural Knee Data: model development specifications – Cleveland Clinic approach*. (2018).
2. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Natural Knee Data: model development protocol deviations – Cleveland Clinic approach*. (2019).
3. Schwartz, A., Chokhandre, S. & Erdemir, A. *Modeling and simulation workflow using Natural Knee Data: model development outputs descriptors – Cleveland Clinic approach*. (2019).
4. Natural Knee Data | Center for Orthopaedic Biomechanics | University of Denver. https://digitalcommons.du.edu/natural_knee_data/.
5. Harris, M. D. *et al.* A Combined Experimental and Computational Approach to Subject-Specific Analysis of Knee Joint Laxity. *Journal of Biomechanical Engineering* **138**, 081004 (2016).
6. ModelCalibration - kneehub. https://simtk.org/plugins/moinmoin/kneehub/ModelCalibration.
7. SimTK: Reproducibility in simulation-based prediction of natural knee mechanics: Project Home. https://simtk.org/projects/kneehub.
8. SimTK: Open Knee(s): Virtual Biomechanical Representations of the Knee Joint: Project Home. https://simtk.org/projects/openknee.
9. ModelDevelopment - kneehub. https://simtk.org/plugins/moinmoin/kneehub/ModelDevelopment.
10. SimTK: Reproducibility in simulation-based prediction of natural knee mechanics: Downloads. https://simtk.org/frs/?group_id=1061.
11. Shafranovich, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files. (2005).
12. MeshLab. http://www.meshlab.net/.
13. Welcome to the www.salome-platform.org — SALOME Platform. https://www.salome-platform.org/.
14. Woo, S. L., Hollis, J. M., Adams, D. J., Lyon, R. M. & Takai, S. Tensile properties of the human femur-anterior cruciate ligament-tibia complex. The effects of specimen age and orientation. *Am J Sports Med* **19**, 217–225 (1991).
15. Momersteeg, T. J. *et al.* The effect of variable relative insertion orientation of human knee bone-ligament-bone complexes on the tensile stiffness. *J Biomech* **28**, 745–752 (1995).
16. Wilson, W. T., Deakin, A. H., Payne, A. P., Picard, F. & Wearing, S. C. Comparative analysis of the structural properties of the collateral ligaments of the human knee. *J Orthop Sports Phys Ther* **42**, 345–351 (2012).
17. Shani, R. H., Umpierez, E., Nasert, M., Hiza, E. A. & Xerogeanes, J. Biomechanical Comparison of Quadriceps and Patellar Tendon Grafts in Anterior Cruciate Ligament Reconstruction. *Arthroscopy* **32**, 71–

75 (2016).

18. Thambyah, A., Nather, A. & Goh, J. Mechanical properties of articular cartilage covered by the meniscus. *Osteoarthr. Cartil.* **14**, 580–588 (2006).
19. Lechner, K., Hull, M. L. & Howell, S. M. Is the circumferential tensile modulus within a human medial meniscus affected by the test sample location and cross-sectional area? *J. Orthop. Res.* **18**, 945–951 (2000).
20. Myronenko, A. & Song, X. Point Set Registration: Coherent Point Drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**, 2262–2275 (2010).
21. Söderkvist, I. & Wedin, P. A. Determining the movements of the skeleton using well-configured markers. *J Biomech* **26**, 1473–1477 (1993).
22. Grood, E. S. & Suntay, W. J. A joint coordinate system for the clinical description of three-dimensional motions: application to the knee. *J Biomech Eng* **105**, 136–144 (1983).
23. FEBio Software Suite. https://febio.org/.
24. Plugins | FEBio Software Suite. https://febio.org/plugins/.
25. PostView | FEBio Software Suite. https://febio.org/postview/.
26. Welcome to Python.org. *Python.org* https://www.python.org/.
27. SciPy.org — SciPy.org. https://www.scipy.org/.
28. pycpd · PyPI. https://pypi.org/project/pycpd/.
29. SimTK: Welcome. https://simtk.org/.
30. Chua, C. K., Leong, K. F. & Lim, C. S. Chapter 6. Rapid Prototyping Data Formats. in *Rapid Prototyping: Principles and Applications* 301–356 (World Scientific Pub Co Inc, 2010).
31. Extensible Markup Language (XML). https://www.w3.org/XML/.
32. Med — SALOME Platform. http://www.salome-platform.org/user-section/about/med.
33. FEBio User's Manual Version 2.8. https://help.febio.org/FEBio/FEBio_um_2_8/index.html.
34. Roelofs, G. *PNG: The Definitive Guide*. (O'Reilly Media, 1999).
35. openknee - Revision 2151: /app/KneeHub/src. https://simtk.org/svn/openknee/app/KneeHub/src/.
36. Creative Commons — Attribution 4.0 International — CC BY 4.0. https://creativecommons.org/licenses/by/4.0/.