An Open-Source Toolbox for Surrogate Modeling of Joint Contact Mechanics

Ilan Eskinazi and Benjamin J. Fregly*

Abstract*—*Goal: **Incorporation of elastic joint contact models into simulations of human movement could facilitate studying the interactions between muscles, ligaments, and bones. Unfortunately, elastic joint contact models are often too expensive computationally to be used within iterative simulation frameworks. This limitation can be overcome by using fast and accurate surrogate contact models that fit or interpolate input–output data sampled from existing elastic contact models. However, construction of surrogate contact models remains an arduous task. The aim of this paper is to introduce an open-source program called Surrogate Contact Modeling Toolbox (SCMT) that facilitates surrogate contact model creation, evaluation, and use.** ***Methods:*** **SCMT interacts with the third-party software FEBio to perform elastic contact analyses of finite-element models and uses MATLAB to train neural networks that fit the input–output contact data. SCMT features sample point generation for multiple domains, automated sampling, sample point filtering, and surrogate model training and testing.** ***Results:*** **An overview of the software is presented along with two example applications. The first example demonstrates creation of surrogate contact models of artificial tibiofemoral and patellofemoral joints and evaluates their computational speed and accuracy, while the second demonstrates the use of surrogate contact models in a forward dynamic simulation of an open-chain leg extension–flexion motion.** ***Conclusion:*** **SCMT facilitates the creation of computationally fast and accurate surrogate contact models. Additionally, it serves as a bridge between FEBio and OpenSim musculoskeletal modeling software.** ***Significance:*** **Researchers may now create and deploy surrogate models of elastic joint contact with minimal effort.**

***Index Terms*—Biomechanics, joint contact, musculoskeletal modeling, surrogate modeling, toolbox.**

I. Introduction

BIOMECHANICS researchers model and simulate joint mechanics to gain insight into a variety of clinical problems such as the onset and progression of degenerative joint disease [1], [2], tissue loading and overloading [3], [4], muscle force estimation [5], [6], and implant design performance [7], [8]. Musculoskeletal simulations on the human- and limb-scale tend to use idealized joint models to approximate the net effect of contact and ligaments on the musculoskeletal model's kinematics. Examples include: revolute joints that model the knee [9]–[12], hinge-matrix formulations that allow rolling and sliding [13], and spherical joints that model the hip [9], [11], [12]. A parallel mechanism is a more complex type of joint model that can accommodate ligaments and multiple contact regions. Parallel mechanisms have been used to model the knee [14] and ankle [15] and to calculate ligament and knee contact forces [16]. Moreover, these mechanisms may also model ligament deformations by means of constraints [17]. While idealized joint models and parallel mechanisms are computationally fast and easy to implement, both suffer from two important limitations. The first is that the joint model may be unable to capture some variables of interest. For example, a knee modeled as a pin joint could not be used to explore the effect of transecting a ligament. The second limitation is that simplifications could result in nonphysiological simulation results. For example, modeling the knee as a planar mechanism with isometric ligaments may imply that the ligaments can exert compressive as well as tensile forces.

The disadvantages of simplified joint models necessitate consideration of more realistic joint models that include ligaments and surface–surface interactions (i.e., between cartilage surfaces or implant components). These interactions can be simulated using deformable contact models that output a set of contact loads given the relative position and orientation of the contacting bodies. Use of explicit contact models in simulations presents several advantages: 1) ligament and contact forces can be calculated; 2) no assumptions are made regarding a joint's axis of rotation; 3) more inverse dynamics loads can be balanced during muscle force optimizations, resulting in a tighter solution space; 4) consequences of injury, surgery, or rehabilitation on ligaments and contact surfaces can be predicted; and 5) the influence of articular geometry on joint kinematics can be taken into account. Despite these advantages, deformable contact models are rarely incorporated into iterative simulation frameworks due to their high computational cost [18].

Recent research efforts have overcome the computational cost issue by mapping computationally "slow" deformable contact models into computationally "fast" metamodels or surrogate models [6], [19], [20]. Although these models can replace original "slow" contact models within simulations, researchers often lack the resources and expertise to develop their own surrogate contact models. To our knowledge, there is no software framework currently available that can streamline the entire surrogate contact model creation, evaluation, and deployment process. This situation poses a major barrier to the use of realistic subject-specific joint contact models in multibody simulations of human movement. Due to the lack of a software framework for surrogate contact modeling, researchers have applied general

Manuscript received March 18, 2015; revised May 21, 2015; accepted June 30, 2015. Date of publication July 13, 2015; date of current version January 16, 2016. This work was supported by the National Institutes of Health under Grant R01EB009351 and the University of Florida. *Asterisk indicates corresponding author.*

I. Eskinazi is with the Department of Mechanical and Aerospace Engineering, University of Florida.

*B. J. Fregly is with the Department of Mechanical and Aerospace Engineering, Department of Biomedical Engineering, and Department of Orthopaedics and Rehabilitation, University of Florida, Gainesville, FL 32611-6250 USA (e-mail: fregly@ufl.edu).

This paper contains supplementary material available online at http://ieeexplore.ieee.org (File size: 1 MB).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TBME.2015.2455510

surrogate modeling packages to the contact problem. Halloran *et al.* [20] used a lazy learning toolbox for MATLAB [21], [22] to build and use surrogate contact models, while Lin *et al.* used the DACE Kriging toolbox for MATLAB [23] to develop a surrogate contact modeling approach [19]. An assortment of other tools such as the SUMO Toolbox [24] and MATLAB's Neural Network Toolbox can also be used for surrogate modeling. However, none of these programs facilitate the workflow needed to create surrogate contact models from beginning to end.

We have developed a freely available open-source program called Surrogate Contact Modeling Toolbox (SCMT) to facilitate the development, deployment, and sharing of surrogate contact models by the research community. SCMT interfaces with the third-party finite-element analysis software FEBio [25] which performs "slow" quasi-static analyses of elastic contact models. SCMT also interfaces with MATLAB's Neural Network Toolbox and MATLAB Coder to train and deploy neural networks. FEBio was chosen because it is well suited for solving biomechanical contact problems, is open source, and is free for noncommercial use. MATLAB's toolboxes were chosen mainly for their neural network training algorithms and ability to export trained neural networks as dynamic-link libraries (DLLs). We have also developed a plugin for OpenSim [26] musculoskeletal modeling software that allows users to incorporate surrogate contact models created with SCMT into musculoskeletal models without having to write any code.

This paper summarizes SCMT's design, features, and workflow. To demonstrate the full spectrum of SCMT's functionality, we provide two examples applications. The first demonstrates the creation of multidomain surrogate contact models for artificial tibiofemoral (TF) and patellofemoral (PF) joints and evaluates their accuracy and computational speed. The second example demonstrates a forward dynamic simulation of an open-chain knee extension–flexion motion where the knee is modeled as a 12 DOF joint possessing deformable TF and PF surrogate contact models, seven ligaments, and four muscles. We hope that making SCMT available to the research community will lead to the widespread use of surrogate contact models in simulations of human motion, ultimately advancing the field of musculoskeletal simulation.

II. Overview

SCMT is a software framework that allows users to generate neural network-based surrogate contact models from finite-element models. SCMT includes an application programming interface (API) and a stand-alone graphical user interface (GUI). The main features of SCMT are a Sample Point Generator, a Model Sampler, a Sample Point Filter, an Out-of-Contact sampler, a Surrogate Model Designer, and a Surrogate Model Tester (see Fig. 1). The GUI allows users to interact with SCMT's core functionality with the aid of tooltips and without having to write code.

SCMT is written in C++ with the GUI designed in Qt Creator (Digia Plc, Valimotie 21, 00380 Helsinki). The source code and compiled executable are freely available (https://simtk.org/home/scmt). The software is designed for use

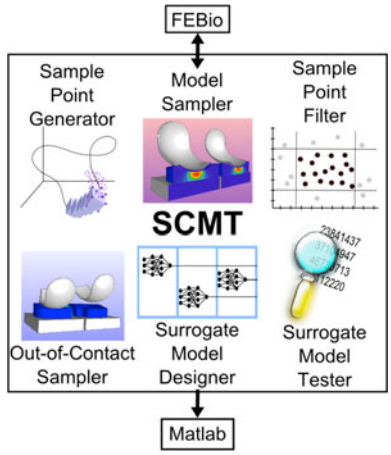

Fig. 1. SCMT is composed of several tools or modules. The Sample Point Generator creates sample points which may cluster around reference trajectories. The Model Sampler performs multiple static analyses by calling FEBio. The Sample Point Filter removes unwanted sample points. The Out-of-Contact Sampler creates sample points corresponding to either fully OOC or partly OOC configurations. The Surrogate Model Designer defines the structure of the surrogate model and the neural network training criteria. The Surrogate Model Tester calculates the surrogate model errors.

in Microsoft Windows operating systems. Third-party tools are used extensively throughout the code. Among these tools are the Boost serialization library to read and write XML files and binary files [27], Hammersley libraries for creating low-discrepancy sequences of sample points [28], and the OpenMP API for multithreading [29]. SCMT interfaces with FEBio and MATLAB via command line execution using the Windows Shell API. A user guide is available online and the code is heavily commented to allow users with C++ knowledge to explore the algorithms and to make changes to the code if desired.

The GUI provides a variety of tools for surrogate model creation that may be used either independently or as part of a structured workflow. The first tool is a Sample Point Generator that allows users to specify multiple domains of input space and the number of sample points desired in each domain. The second tool is a Model Sampler that interfaces with FEBio by parsing files and executing command line calls in a multithreaded fashion. The third tool is a Sample Point Filter that eliminates sample points containing values that exceed user-defined limits. The fourth tool is an Out-of-Contact Sampler which creates sample points that are in either fully out-of-contact (OOC) or partly OOC configurations. The fifth tool is a Surrogate Model Designer that specifies the inputs to and outputs of each artificial neural network (ANN) and the overall surrogate model. The Surrogate Model Designer also writes the training and testing data to disk and can launch a MATLAB routine that generates and compiles the ANNs. The sixth tool is a Surrogate Model Tester that calculates root-mean-squared (RMS) errors and maximum absolute errors for the surrogate model outputs in multiple domains of input space. The seventh tool is a set of Utilities that simplify tasks regarding joining sets of sample points and converting sample point data into different formats. Together, these tools allow users to generate surrogate contact models easily and efficiently.

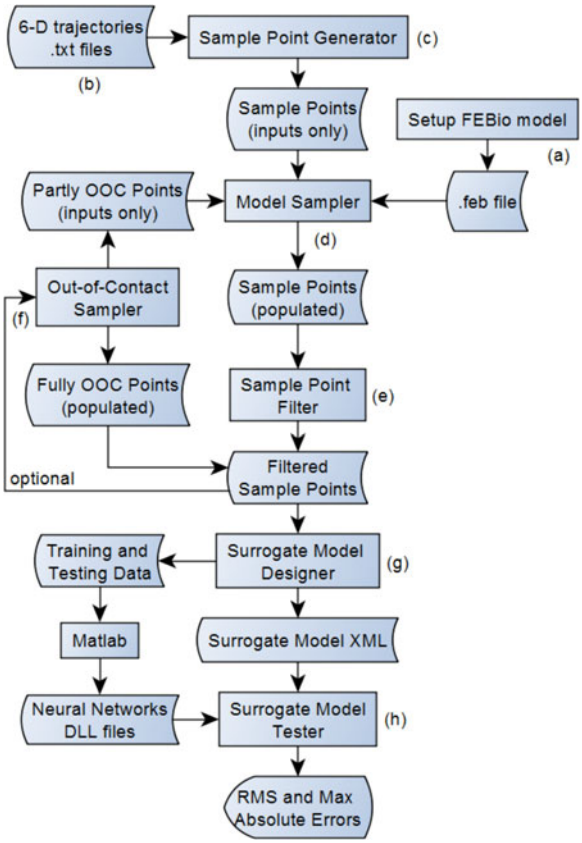

Fig. 2. Diagram describing the steps required to generate and test surrogate contact models using SCMT. The letters in parenthesis correspond to the steps described in the workflow section.

We also produced an OpenSim plugin to support surrogate contact models created with SCMT. The plugin allows OpenSim users to incorporate their surrogate contact models into OpenSim models as "Force" components. With this plugin, SCMT effectively provides a bridge between FEBio and OpenSim, two simulation tools widely used in the biomechanics community.

III. SCMT Workflow

The following section describes the recommended workflow to create surrogate contact models with SCMT (see Fig. 2). All of these steps are illustrated in detail in two example applications presented in the next section. A more detailed description is available in the SCMT user guide (https://simtk.org/home/scmt).

A. Setting up the FEBio Model

The FEBio model must contain one or two fixed rigid bodies and a single moving rigid body. The fixed bodies should have fixed constraints on all degrees of freedom while the moving body should have a combination of prescribed load constraints and prescribed kinematic constraints. Meshes with deformable material models should be attached to the rigid bodies. SCMT captures the reaction forces and torques on the fixed rigid bodies and the pose of the moving rigid body at the end of each static analysis.

Each static analysis requires two FEBio analysis steps. The first step is configured to apply large loads while the prescribed kinematic constraints are being met. The second step keeps the kinematics from the end of the first analysis step while trying to meet prescribed load constraints on the moving body. Step by step instructions on how to configure the FEBio model for use with SCMT are provided in the user guide.

B. Setting up Reference Trajectories

SCMT generates sample point inputs that cluster around user-defined reference trajectories. These trajectories are 6-D representations of pose and/or load measures in the six spatial directions. For example, a point in a trajectory could be described as $(force_x, force_y, translation_z, rotation_x, torque_y, torque_z)$. The simplest trajectory is composed of two points that define a 6-D "bounding box" encompassing the desired input space. The bounding box approach yields a domain of large span and low sample point density. To improve sample point density, the user may specify a more detailed trajectory and SCMT will create multiple small subdomains around it. This approach mitigates the curse of dimensionality and allows for regions of high accuracy in the surrogate models.

C. Defining Domains and Generating Sample Point Inputs

Before generating sample points, the user defines the domains to be sampled using the Sample Point Generator. To create a domain, the user first loads a reference trajectory. Then, the user gives a name to the domain and defines several settings (number of sample points per domain, Hammersely sequence details, etc.). The user should keep in mind that surrogate model accuracy decreases with domain size and increases with number of sample points. Finally, once the Sample Point Generator settings are defined, sample points populated with inputs may be created.

D. Sampling FEBio Model to Obtain Outputs

Once the FEBio model is configured and the sample point inputs are generated, the user performs multiple contact simulations in FEBio with the Model Sampler. The user loads the sample points previously generated and specifies the name of the FEBio file, the FEBio executable to be used, the number of concurrent threads, and the settings required for parsing the FEBio file. The FEBio model is then sampled via repeated static analyses and the progress is monitored in the console window. Alternatively, the user could utilize a contact simulation program other than FEBio by manipulating the sample point inputs and outputs directly with the SCMT API.

E. Filtering Sample Points

Once sampling has been completed, the user inspects how many sample points converged with the Sample Point Filter. The user then removes unwanted sample points from the dataset. This step is necessary because the static analyses sometimes result in sample points with unrealistic poses or loads.

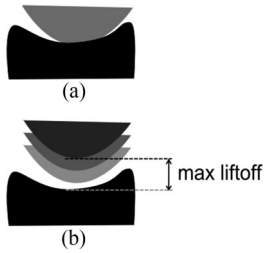

Fig. 3. Description of fully OOC sample point generation. (a) Preexisting sample point configuration. (b) Moving body is translated along a selected direction by specifying a maximum liftoff value and all contact loads corresponding to the translated configuration are set to zero. Intermediate translations yield intermediate poses.

F. Generating OOC Points

SCMT includes a tool for creating sample points in OOC configurations. There are two types of OOC situations. The first occurs when there is no contact happening between the fixed and moving bodies. In this case, the moving body can simply be translated in a specified direction t_x, t_y, or t_z relative to a previously sampled configuration to achieve a specific amount of liftoff (see Fig. 3). Sample points corresponding to intermediate liftoff values are also created. The sample points generated are called "fully OOC points" and require no simulations since all contact loads are known to be zero.

The second situation occurs when one pair of potentially interacting contact surfaces (or contact pair) is OOC yet a second contact pair is still in contact. These sample points are called "partly OOC points." Because these points share similar translations and rotations to sample points that are fully in contact, it was convenient to borrow some inputs from these sample points to be used as initial configurations. SCMT implements the following method to sample a partly OOC point: A partly OOC sample point S_{ooc} is created using values from an existing sample point S where both contact interfaces were in contact. Three rotations corresponding to S are prescribed to S_{ooc} as inputs. Two translations from S are prescribed to S_{ooc} as well as one contact force in the remaining spatial direction. Next, a user-specified rotation r_x, r_y, or r_z of S_{ooc} is offset to tilt the moving body. When S_{ooc} is processed with the Model Sampler, the resulting configuration will be partly OOC (see Fig. 4) and could be added to the database of sample points to be fitted with a surrogate model.

G. Designing and Creating Surrogate Model

Once all sample points have been collected, the user proceeds to define the structure of the surrogate model with the Surrogate Model Designer. SCMT creates surrogate models composed of sequential blocks or computational stages. Each stage corresponds to an ANN with multiple inputs and a single output, where the outputs of earlier blocks can be used as inputs to the ones that follow. Using the Surrogate Model Designer, the user specifies the inputs and outputs of the surrogate model and its stages. The Surrogate Model Designer is also used to write sample points to file in the proper format for MATLAB to read.

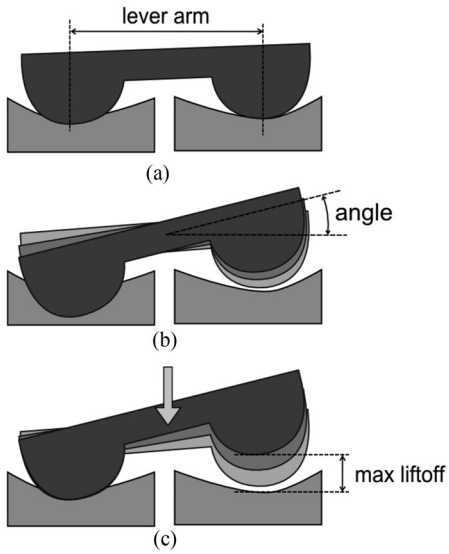

Fig. 4. Description of partly OOC sample point generation. (a) Preexisting sample point configuration. The user-specified lever arm is the estimated distance between the two contact patches. (b) Moving body is prescribed to rotate to a specified angle. (c) Force is applied in a compressive direction yielding a single contact patch and the approximate predefined maximum liftoff. Intermediate rotations yield intermediate poses.

After specifying model stages, the user defines the neural network architectures and stopping criteria. Each neural network architecture is defined in terms of number of hidden layers and number of neurons per layer. The stopping criteria options are the following: 1) exceeding maximum allotted time for training of each ANN; 2) meeting goal RMS error for training set; and 3) exceeding maximum number of consecutive epochs with increasing validation error. Criterion 1 sets a limit of maximum computation time, Criterion 2 stops the training process when an ANN is considered "accurate enough," and Criterion 3 avoids overfitting the training dataset. The training process for each ANN is terminated as soon as any of these criteria is met. Other configuration options can be selected including whether or not training should be parallelized and the percentage of training sample points to be used for cross-validation in Criterion 3. Once all options are specified, the user launches the training process from the GUI. A MATLAB console window in automation mode will show the ANN training progress. Upon finishing, all ANNs are saved to disk as MATLAB workspace files, as MATLAB functions, and as compiled DLLs. The XML file containing all surrogate model settings together with the DLLs comprise the complete surrogate model.

H. Testing Surrogate Contact Model

The Surrogate Model Tester tool allows users to load a surrogate model and test it against a separate set of sample points not used for training. The Surrogate Model Tester outputs the RMS errors and maximum absolute errors for each of the outputs. The user also has the option to evaluate errors only in specific domains. For example, the user may ask for errors only in the domain corresponding to sample points that are OOC. If the errors are unacceptably large, the user can revise the settings in

the previous steps and iterate while keeping all of the sample points previously generated.

IV. Example Applications

The two examples presented in this section illustrate: 1) How SCMT can be used to create multidomain surrogate models of deformable contact for artificial TF and PF joints; and 2) How surrogate contact models created with SCMT of artificial TF and PF joints can be used within a forward dynamic simulation of an open-chain knee extension–flexion motion. All files required to run both example applications are provided at https://simtk.org/home/scmt. The knee was selected for both example applications because it is a highly complex 12 DOF joint typically modeled as a 1 DOF joint in musculoskeletal simulations. Being able to model the knee as a 6 DOF TF joint coupled with a 6 DOF PF joint within muscle-actuated simulations could facilitate improved estimation of joint contact, ligament, and muscle forces during movement. Moreover, modeling the knee as a 12 DOF joint eliminates assumptions about the knee axis of rotation, patellar kinematics, and how contact forces contribute to the flexion–extension moment. Instrumented implant force data, fluoroscopy data, an OpenSim musculoskeletal model, and implant geometry for these examples were obtained from the First Grand Challenge Competition to Predict *in vivo* Knee Loads [30].

A. Example 1: Surrogate Model Creation

This example illustrates the creation and testing of surrogate contact models of TF and PF artificial joints to be used in gait simulations.

1) Creating TF and PF Finite-Element Models: We created FEBio finite-element models of TF and PF joints using geometric models of the implant components. The TF contact model consisted of single element meshes for the medial and lateral fixed rigid bodies representing the tibial tray, a deformable mesh representing the plastic tibial insert with its back surfaces attached to the fixed bodies, and a mesh for the metallic femoral component condyles modeled as rigid. The PF contact model consisted of a single element mesh for the fixed rigid body representing the patella bone, a deformable mesh modeling the plastic patellar button with its back surface attached to the fixed body, and a mesh of the femoral component trochlea modeled as rigid. In both cases, the rigid femoral component served as the moving body to which a combination of kinematic constraints and loads were prescribed. The plastic components made of ultrahigh molecular weight polyethylene were modeled as neo-hookean solids with their Young's modulus estimated using experimental contact force, pressure, and area data collected from a similar implant [31] (see Supplementary material). The selected Young's modulus and Poisson's ratio were 700 MPa and 0.45, respectively, which corresponded to values used in another study [18].

The implant geometries were meshed using TrueGrid (XYZ Scientific Applications, Inc., Livermore, CA) and consisted of 2784 and 2000 hexahedral elements for the tibial insert and patellar button, respectively. These meshes were tested against finer meshes (75 168 elements for tibial insert and 8000 elements for patellar button) using static analyses, and the discrepancies were below 12 N/250 N · mm in reaction forces/torques and below 15 μm/2.5 $\times$ 10^{-2} deg in poses.

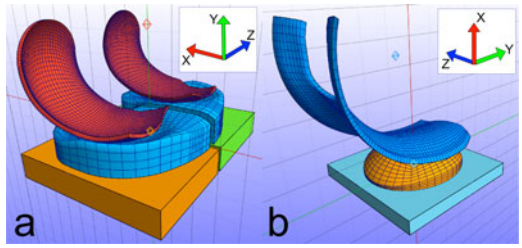

Fig. 5. (a) Finite-element model of TF joint contact. The femoral component surfaces were modeled as rigid while the tibial insert was modeled as deformable. Two "fake" rigid bodies below the tibial insert were added such that their reaction loads could be captured. (b) Finite-element model of PF joint contact. The femoral component surface was modeled as rigid while the patellar button was modeled as deformable. One "fake" rigid body was added under the patellar button to capture the reaction loads.

The moving bodies for the TF and PF models were translated to suitable initial configurations where only small penetrations occurred between the contacting surfaces. The centers of rotation of the moving bodies were translated accordingly. The deformable meshes were rigidly attached to the fixed bodies using rigid contact constraints. Sliding contact was defined between the implant contact surfaces. After we configured the constraints, load curves, and analysis settings, the models were ready to be sampled (see Fig. 5). This entire configuration procedure was performed within the software package Preview [25] which exported the FEBio models.

2) Setting up Reference Trajectories and Defining Domains: The surrogate contact models created for this example were intended for use in gait simulations. Therefore, we clustered sample points along reference envelopes representing walking kinematics and loads. For the TF model, we created kinematic and kinetic envelopes for a variety of walking motions using a single cycle of fluoroscopic knee motion data (anterior–posterior translation, medial–lateral translation, internal–external rotation, and flexion–extension rotation) and multiple cycles of instrumented implant load data (inferior-superior force and adduction moment) [30]. Real variability was used to define envelope ranges for the load data and assumed variability was used for the motion data. Domain T1 was built along normal walking trajectories, and domain T2 was built for a series of different walking motions including normal gait, medial thrust gait, walking pole gait, and trunk sway gait. Domain T3 encompassed all trajectories in a bounding box without clustering sample points. Domain T4 consisted of sample points at the contact boundary for both medial and lateral sides. Domain T5 covered the configurations where both condyles were OOC. Domain T6 represented configurations where only one condyle was OOC (see Supplementary material).

We had no reliable experimental data to help define pose/load trajectories for the PF model. Therefore, we defined a large bounding box domain named P1 which encompassed estimated kinematics from cadaver studies [32], [33] and estimated loads

from simulation studies [6], [34]. We defined another domain named P2 corresponding to points at the contact boundary. Finally, we defined domain P3 as OOC configurations.

3) Sampling and Filtering: Sample points were obtained automatically using the Model Sampler. With 13 parallel threads, our 3.4 GHz PC workstation was able to perform roughly one static analysis per second necessitating about two days of computation per joint. The net number of static analyses performed was 91 263 for the TF joint and 82 126 for the PF joint.

Sample points in domains T1–T4 were filtered such that the medial and lateral inferior-superior forces were compressive ($F_y^{\mathrm{med}}, F_y^{\mathrm{lat}} < -1\,\mathrm{N}$). Sample points in domain P2 were also filtered such that the compressive force was in the expected range ($-20\,\mathrm{N} < F_x < 0\,\mathrm{N}$). The effects of filtering on the number of sample points available for training and testing is shown in the Supplementary material.

4) Designing and Training Surrogate Models: The surrogate models took pose parameters as inputs and loads as outputs. The inputs and outputs to the stages of the TF model are shown in (1)–(7) while the inputs and outputs to the stages of the PF model are shown in (8)–(13). Translations t_x, t_y, and t_z were along the global *x*, *y*, and *z* axes while rotations r_x, r_y, and r_z followed a body-fixed *x*–*y*–*z* Euler sequence. For the TF model, the global *x*-axis pointed posteriorly, the *y*-axis superiorly, and the *z*-axis medially. For the PF model, the global *x*-axis pointed posteriorly, the *y*-axis superiorly, and the *z*-axis medially. Forces acting on the fixed bodies are denoted as F and torques as T. The TF model outputs medial and lateral superior-inferior forces (F_y^{med} and F_y^{lat}) to describe the medial–lateral load split. The inputs and outputs to each stage were chosen using a previously defined method [35]. For both models, the contact loads that were highly sensitive to pose parameter variations were fit as functions of the pose parameters while the insensitive loads were fit as functions of the pose parameters and the sensitive loads calculated in the earlier stages.

For the TF model, each stage consisted of an ANN with four hidden layers of 30 neurons each. For the PF model, each stage consisted of an ANN with two hidden layers of 30 neurons each. The models were trained using 12 MATLAB workers with 20% of the training data selected randomly for cross-validation. The stopping criteria for each ANN were set to 6000 s, a training RMS value of 1 N or 1 N · mm, and 50 consecutive validation error increments.

a) Surrogate model stages for TF contact:

$$F_y^{\mathrm{med}} = f\left(t_x, t_y, t_z, r_x, r_y, r_z\right) \quad (1)$$

$$F_y^{\mathrm{lat}} = f\left(t_x, t_y, t_z, r_x, r_y, r_z\right) \quad (2)$$

$$F_x = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_y^{\mathrm{med}}, F_y^{\mathrm{lat}}\right) \quad (3)$$

$$F_z = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_y^{\mathrm{med}}, F_y^{\mathrm{lat}}\right) \quad (4)$$

$$T_x = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_y^{\mathrm{med}}, F_y^{\mathrm{lat}}\right) \quad (5)$$

$$T_y = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_y^{\mathrm{med}}, F_y^{\mathrm{lat}}\right) \quad (6)$$

$$T_z = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_y^{\mathrm{med}}, F_y^{\mathrm{lat}}\right). \quad (7)$$

b) Surrogate model stages for PF contact:

$$F_x = f\left(t_x, t_y, t_z, r_x, r_y, r_z\right) \quad (8)$$

$$F_z = f\left(t_x, t_y, t_z, r_x, r_y, r_z\right) \quad (9)$$

$$F_y = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_x, F_z\right) \quad (10)$$

$$T_x = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_x, F_z\right) \quad (11)$$

$$T_y = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_x, F_z\right) \quad (12)$$

$$T_z = f\left(t_x, t_y, t_z, r_x, r_y, r_z, F_x, F_z\right). \quad (13)$$

5) Evaluating Errors: The models were tested in all domains of inputs space (see Supplementary material for details). All RMS force/torque errors were below 24 N/ 621 N · mm and 21 N/173 N · mm for the TF and PF surrogate models, respectively. The RMS and maximum absolute errors for the TF model were found to be lowest for domain T5 (fully OOC) and highest for domains T2 (all gait) and T6 (partly OOC). The errors for the PF surrogate model were also calculated and were similarly the lowest for domain P3 (fully OOC).

6) Evaluating Computational Speed: Both surrogate models were subjected to a computational speed test consisting of 10 000 consecutive surrogate model evaluations on a 3.4 GHz computer without parallelization. The average computation time for each TF and PF model evaluation was 88.3 and 57.6 μs, respectively.

B. Example 2: Surrogate Model Utilization

We incorporated TF and PF surrogate contact models into the preexisting patient-specific pelvis and leg OpenSim model to perform a forward dynamic simulation of a seated open chain knee extension–flexion motion. The surrogate TF contact model encompassed a larger domain than the one generated for gait in Example 1, allowing for higher flexion. The surrogate PF contact model was the same one described in Example 1.

1) Creating Realistic Knee Poses: We created realistic static poses where we applied the estimated contact loads that ligaments would impose on the tibial tray and patellar button. We started by removing all muscles from the model, locking the pelvis to ground, locking the ankle DOFs, and locking the hip joint at 90° of flexion to place the model in a "sitting" position. We removed gravity from the model and added two coordinate actuators that applied generalized forces of 100 N on the patellar button and tibial insert to push them into the femoral component as ligaments would. All TF joint DOFs except for the flexion angle were free, while for the PF joint the medial–lateral and anterior–posterior translations were free and all other DOFs were locked. We then performed static analyses at 0 and 80° of knee flexion and recorded the coordinates corresponding to the static poses. We visually compared the resulting static poses to fluoroscopic images to verify that the patellar location and tilt were realistic.

2) Adding and Calibrating Ligaments at Knee Poses: Once we obtained the two static poses, we removed all coordinate actuators from the model and replaced them with ligament models. We modeled each ligament as three nearly parallel nonlinear path springs. The following ligaments were added: the

patellar ligament, the medial collateral ligament (MCL), the lateral collateral ligament, the medial patellofemoral ligament, and the lateral patellofemoral ligament. The MCL included the deep MCL, the proximal superficial MCL, and the distal superficial MCL, each with three bundles. We omitted the anterior cruciate ligament since the subject had it removed during knee replacement surgery. We also left out the posterior cruciate ligament since it applies loads only in high flexion [36] at angles beyond our simulation. We estimated the stiffness corresponding to the linear portion of the force–length curve of each ligament based on literature values [37], [38], and we visually identified the origin and insertion points of each ligament using anatomy references [39]–[42].

We performed an optimization to select the resting lengths of all ligament bundles in the model. The optimization changed the resting lengths such that the net load in each ligament (i.e., the sum of the three bundle loads) was close to 50 N for both static configurations. The optimization also selected the resting lengths of the patellar ligament bundles such that each one applied 100 N of force for the 80° flexion pose.

3) Finding Static Initial Conditions for Simulation: After selecting ligament resting lengths, we performed a series of static analyses to prepare the model for simulation. We added ideal path actuators to the model to represent the four quadriceps muscles, reintroduced gravity, and altered wrapping surfaces to increase the knee extensor moment arms. For the first static analysis, we applied constant and equal forces to the four quadriceps muscles with the knee flexion angle (for the TF joint) locked at 80° and the other 11 knee DOFs freed. This analysis balanced the ligament, contact, and muscle forces in all DOFs except for TF flexion. The second static analysis started from the previous static pose but with the knee flexion angle freed. The resulting pose yielded a static configuration where muscle, ligament, and contact forces balanced all 12 DOF in the knee. We repeated this static analysis multiple times, iterating on quadriceps forces until the static flexion angle was about 80°.

4) Performing Forward Dynamic Simulation: Once the initial static pose and muscle forces for simulation were determined, we added a PD controller to the model to track a desired sinusoidal knee flexion curve and added damper forces to the TF and PF joints to reduce the vibrations introduced by the contact and ligament forces. The controller was modified such that the force in each muscle could never go below 15 N, the value in the initial configuration. Using the OpenSim API in C++, we performed a forward dynamic simulation of a two second open-chain extension–flexion motion using the CPODES implicit integrator with the order limited to 2 and the accuracy set to 5×10^{-3}. The 12 DOF simulation (see Fig. 6) finished in 4.7 s of CPU time and calculated the TF and PF contact forces experienced during the motion (see Fig. 7).

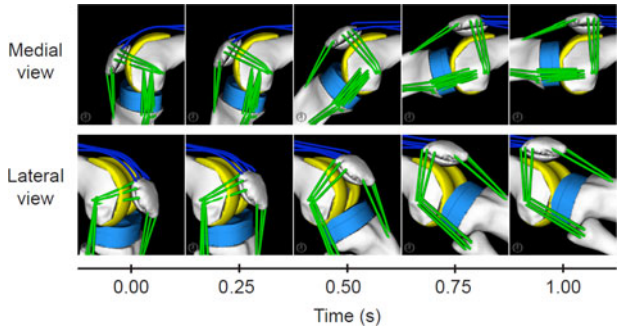

Fig. 6. Snapshots for the first second of the 2-s seated knee extension–flexion simulation for an OpenSim model with 12 DOF. The model had six DOFs for the TF joint and six for the PF joint. Ligament bundles were added and their resting lengths calibrated. A feedback controller was used to apply the path actuator forces that drove the motion.

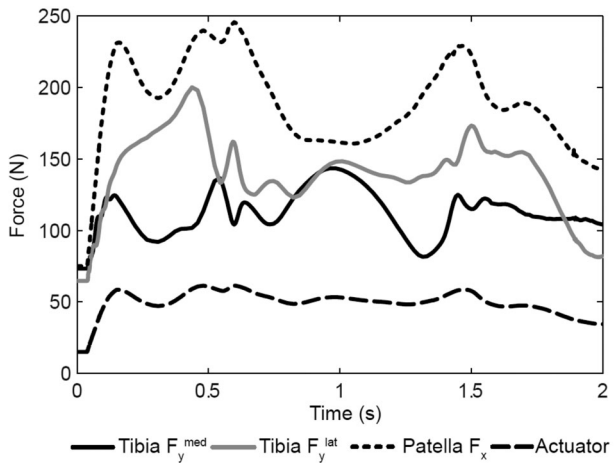

Fig. 7. Plot of the forces acting during the OpenSim simulation of knee extension and flexion. "Tibia Fymed" and "Tibia Fylat" are the superior-inferior compressive forces acting on the medial and lateral compartment of the tibial insert. "Patella Fx" is the compressive force acting on the patellar button in the direction normal to its back surface. "Actuator" stands for the force in each one of the four path actuators representing the quadriceps femoris muscles. Force oscillations at 0.6 and 1.5 s are caused by compliant knee ligaments in the leg model.

V. Discussion

This paper presented SCMT, a new program for generating surrogate contact models from elastic finite-element models. The program facilitates sampling finite-element contact models, fitting ANNs to the collected data, assembling the ANNs into surrogate models, and deploying the surrogate models in OpenSim or any other program with a C++ interface. SCMT is meant to be used for research applications involving the modeling and iterative simulation of musculoskeletal models that incorporate joint contact. The same framework could potentially be used to develop foot-ground or limb-socket surrogate contact models as well.

Two examples showed how surrogate contact models can be created with SCMT and used in forward dynamic simulations. The first example application involving surrogate modeling of the knee demonstrated the toolbox's ability to create surrogate contact models of both the TF and the PF joints with varying levels of accuracy across different domains of input space. The ability to fit OOC points was also demonstrated. In the second example, the OpenSim simulation showed how surrogate contact models created with SCMT may be incorporated into musculoskeletal models possessing ligaments and controlled by muscle forces. Simulations involving surrogate contact models

could be used to investigate how muscle, ligament, and joint contact forces interact to produce the resulting joint kinematics and loads on each of these anatomic structures.

Though SCMT facilitates surrogate contact model creation and use, it still possesses several limitations. First, The ANN-based surrogate models do not provide estimates of prediction variance, so the user has no knowledge of the error in the surrogate model outputs during a simulation. Second, the computing times required for sampling finite-element models and for training the ANNs are significant and could range from days to weeks depending on the available computer hardware. Third, SCMT can collect reaction forces and torques at the origins of no more than two fixed rigid bodies. Fourth, the current version of SCMT does not collect or fit pressure or center of pressure data from the finite-element simulations, which could be useful additional outputs for some applications. Fifth, contact loads are assumed to be functions of pose with no velocity dependence or permanent deformation, and thus only elastic contact models can be represented.

VI. Conclusion

We have shown that SCMT can produce fast and accurate surrogate contact models of more computationally expensive FE contact models. The software can significantly reduce the time and effort required to create and perform computationally efficient musculoskeletal simulations incorporating deformable joint contact models. Our hope is that SCMT will lead to realistic simulations of joint kinematics, more accurate estimation of muscle and joint contact forces, and predictive simulations of rehabilitation and surgical interventions.

References

[1] T. P. Andriacchi *et al.*, "Rotational changes at the knee after ACL injury cause cartilage thinning," *Clin. Orthopaedics Relat. Res.*, vol. 442, pp. 39–44, Jan. 2006.

[2] E. Peña *et al.*, "Computer simulation of damage on distal femoral articular cartilage after meniscectomies," *Comput. Biol. Med.*, vol. 38, no. 1, pp. 69–81, Jan. 2008.

[3] E. Aksahin *et al.*, "The effects of the sagittal plane malpositioning of the patella and concomitant quadriceps hypotrophy on the patellofemoral joint: A finite element analysis," *Knee Surg. Sports Traumatol. Arthrosc.*, Nov. 2014, doi: 10.1007/s00167-014-3421-7

[4] C. Savoldelli *et al.*, "Stress distribution in the temporo-mandibular joint discs during jaw closing: A high-resolution three-dimensional finite-element model analysis," *Surg. Radiol. Anatomy*, vol. 34, no. 5, pp. 405–413, Jul. 2012.

[5] D. G. Thelen *et al.*, "Co-simulation of neuromuscular dynamics and knee mechanics during human walking," *J. Biomech. Eng.*, vol. 136, no. 2, p. 0210331, Jan. 2014.

[6] Y.-C. Lin *et al.*, "Simultaneous prediction of muscle and contact forces in the knee during gait," *J. Biomech.*, vol. 43, no. 5, pp. 945–952, Mar. 2010.

[7] L. Zach *et al.*, "Design, analysis and verification of a knee joint oncological prosthesis finite element model," *Comput. Biol. Med.*, vol. 54C, pp. 53–60, Aug. 2014.

[8] J. P. Halloran *et al.*, "Explicit finite element modeling of total knee replacement mechanics," *J. Biomech.*, vol. 38, no. 2, pp. 323–331, Feb. 2005.

[9] F. C. Anderson and M. G. Pandy, "Dynamic optimization of human walking," *J. Biomech. Eng.*, vol. 123, no. 5, pp. 381–390, Oct. 2001.

[10] M. J. Koehle and M. L. Hull, "The effect of knee model on estimates of muscle and joint forces in recumbent pedaling," *J. Biomech. Eng.*, vol. 132, no. 1, p. 011007, Jan. 2010.

[11] H. J. Kim *et al.*, "Evaluation of predicted knee-joint muscle forces during gait using an instrumented knee implant," *J. Orthopaedics Res.*, vol. 27, no. 10, pp. 1326–1331, Oct. 2009.

[12] T. Alkjaer *et al.*, "Computational modeling of a forward lunge: Towards a better understanding of the function of the cruciate ligaments," *J. Anatomy*, vol. 221, no. 6, pp. 590–597, Dec. 2012.

[13] A. Seth *et al.*, "Minimal formulation of joint motion for biomechanisms," *Nonlinear Dyn.*, vol. 62, no. 1, pp. 291–303, Oct. 2010.

[14] R. D. Gregorio and V. Parenti-Castelli, "A spatial mechanism with higher pairs for modelling the human knee joint," *J. Biomech. Eng.*, vol. 125, no. 2, pp. 232–237, Apr. 2003.

[15] R. D. Gregorio *et al.*, "Mathematical models of passive motion at the human ankle joint by equivalent spatial parallel mechanisms," *Med. Biol. Eng. Comput.*, vol. 45, no. 3, pp. 305–313, Mar. 2007.

[16] F. Moissenet *et al.*, "A 3D lower limb musculoskeletal model for simultaneous estimation of musculo-tendon, joint contact, ligament and bone forces during gait," *J. Biomech.*, vol. 47, no. 1, pp. 50–58, Jan. 2014.

[17] X. Gasparutto *et al.*, "Validation of a multi-body optimization with knee kinematic models including ligament constraints," *J. Biomech.*, vol. 48, no. 6, pp. 1141–1146, Apr. 2015.

[18] J. P. Halloran *et al.*, "Comparison of deformable and elastic foundation finite element simulations for predicting knee replacement mechanics," *J. Biomech. Eng.*, vol. 127, no. 5, pp. 813–818, Oct. 2005.

[19] Y.-C. Lin *et al.*, "Surrogate articular contact models for computationally efficient multibody dynamic simulations," *Med. Eng. Phys.*, vol. 32, no. 6, pp. 584–594, Jul. 2010.

[20] J. P. Halloran *et al.*, "Adaptive surrogate modeling for efficient coupling of musculoskeletal control and tissue deformation models," *J. Biomech. Eng.*, vol. 131, no. 1, p. 011014, Jan. 2009.

[21] C. G. Atkeson *et al.*, "Locally weighted learning for control," *Artif. Intell. Rev.*, vol. 11, no. 1–5, pp. 75–113, 1997.

[22] M. Birattari *et al.*, "Lazy learning meets the recursive least squares algorithm," in *Proc. Conf. Adv. Neural Inf. Process. Syst. II*, 1998, pp. 375–381.

[23] S. Lophaven *et al.*, "DACE—A MATLAB Kriging Toolbox," Technical University of Denmark, Kongens Lyngby, 2002.

[24] D. Gorissen and T. Dhaene, "A surrogate modeling and adaptive sampling toolbox for computer based design," *J. Mach. Learn. Res.*, vol. 11, pp. 2051–2055, 2010.

[25] S. A. Maas *et al.*, "FEBio: Finite elements for biomechanics," *J. Biomech. Eng.*, vol. 134, no. 1, p. 011005, Jan. 2012.

[26] S. L. Delp *et al.*, "OpenSim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007.

[27] *Boost C++ Libraries*. (2015, Feb. 5). [Online]. Available: http://www.boost.org/

[28] J. Burkardt. (2015, Feb. 5). *HAMMERSLEY—The Hammersley Quasirandom Sequence*. [Online]. Available: http://people.sc.fsu.edu/~jburkardt/m_src/hammersley/hammersley.html

[29] *The OpenMP API Specification for Parallel Programming*. (2015, Feb. 5). [Online]. Available: http://openmp.org/wp/

[30] B. J. Fregly *et al.*, "Grand challenge competition to predict in vivo knee loads," *J. Orthopaedics Res.*, vol. 30, no. 4, pp. 503–513, May 2012.

[31] B. J. Fregly *et al.*, "Experimental evaluation of an elastic foundation model to predict contact pressures in knee replacements," *J. Biomech.*, vol. 36, no. 11, pp. 1659–1668, Nov. 2003.

[32] A. Keshmiri *et al.*, "Do surgical patellar interventions restore patellar kinematics in fixed-bearing, cruciate-retaining total knee arthroplasty?: An in vitro study," *J. Arthroplasty*, vol. 29, no. 11, pp. 2197–2201, Jul. 2014.

[33] K. M. Varadarajan *et al.*, "Patellar tendon orientation and patellar tracking in male and female knees," *J. Orthopaedics Res.*, vol. 28, no. 3, pp. 322–328, Mar. 2010.

[34] Z. Chen *et al.*, "Prediction of in vivo joint mechanics of an artificial knee implant using rigid multi-body dynamics with elastic contacts," *Proc. Inst. Mech. Eng.*, vol. 228, no. 6, pp. 564–575, May 2014.

[35] I. Eskinazi and B. J. Fregly, "Surrogate modeling of deformable joint contact using artificial neural networks," *Med. Eng. Phys.*, vol 37, no. 9, pp. 885–891, Sep. 2015

[36] R. Papannagari *et al.*, "Function of posterior cruciate ligament bundles during *in vivo* knee flexion," *Amer. J. Sports Med.*, vol. 35, no. 9, pp. 1507–1512, Oct. 2007.

[37] G. Li *et al.*, "A validated three-dimensional computational model of a human knee joint," *J. Biomech. Eng.*, vol. 121, no. 6, pp. 657–662, Dec. 1999.

[38] K. E. Kim *et al.*, "Tensile properties of the medial patellofemoral ligament: The effect of specimen orientation," *J. Biomech.*, vol. 47, no. 2, pp. 592–595, Jan. 2014.

[39] R. F. LaPrade *et al.*, "The posterolateral attachments of the knee: A qualitative and quantitative morphologic analysis of the fibular collateral ligament, popliteus tendon, popliteofibular ligament, and lateral gastrocnemius tendon," *Amer. J. Sports Med.*, vol. 31, no. 6, pp. 854–860, Nov. 2003.

[40] A. M. Merican and A. A. Amis, "Anatomy of the lateral retinaculum of the knee," *J. Bone Joint Surg.*, vol. 90, no. 4, pp. 527–534, Apr. 2008.

[41] E. Nomura *et al.*, "Anatomical analysis of the medial patellofemoral ligament of the knee, especially the femoral attachment," *Knee Surg. Sports Traumatol. Arthrosc.*, vol. 13, no. 7, pp. 510–515, Oct. 2005.

[42] G. C. Terry and R. F. LaPrade, "The posterolateral aspect of the knee: Anatomy and surgical approach," *Amer. J. Sports Med.*, vol. 24, no. 6, pp. 732–739, Dec. 1996.

Authors' photographs and biographies not available at the time of publication.