Nonlinear surrogate modeling of tibio-femoral joint interactions

Meenakshi Mishra\textsuperscript{a}, Reza Derakhshani\textsuperscript{b,*,} Gavin C. Paiva\textsuperscript{c}, Trent M. Guess\textsuperscript{c}

\textsuperscript{a} Electrical Engineering, University of Missouri-Kansas City, 454 Robert H. Flarsheim Hall, 5110 Rockhill Road, Kansas City, MO 64110-2499, USA
\textsuperscript{b} Mechanical Engineering, University of Missouri-Kansas City, 350K Robert H. Flarsheim Hall, 5110 Rockhill Road, Kansas City, MO 64110-2499, USA
\textsuperscript{c} Electrical Engineering, 570D Robert H. Flarsheim Hall, 5110 Rockhill Road, Kansas City, MO 64110-2499, USA

\textbf{Abstract}

Musculoskeletal modeling can predict muscle forces and the resulting motion and loading during human ambulatory activities. A better understanding of the loading environment on hard and soft tissues can enhance our understanding of ligament injury and prevention, tissue engineering, prosthetic design, osteoporosis, and osteoarthritis. The current state-of-the-art in movement simulation is to use simplified representation of the joints, such as representing the knee as a simple hinge joint. The aim of this study is to produce data-driven surrogate models which effectively capture the complex three-dimensional behavior of tibio-femoral joint interactions and that have the ease of use and computational efficiency required for incorporation in existing neuromusculoskeletal simulations. In order to meet our objective, we explored and compared the performance and sensitivity of nonlinear Hammerstein–Wiener, nonlinear autoregressive, and time delay neural network models under different configurations, individually and in ensembles. These models learned from solutions calculated by a validated multibody model of the knee. Inputs to the surrogate models were positions and orientations of the tibia relative to the femur, and the outputs were resulting forces and torques at the tibia with respect to the femur. Models were mixed using mean (sum) rule, weighted mean, and stacked generalization ensemble methods. It was observed that individually, time delay neural network models performed better than other models with normalized mean square errors between 0.0509 and 0.0889 on test data. Among the ensembles, stacked generalization provided the best results reducing test errors by 13–40%.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

The ability to predict loading on musculoskeletal system tissues during dynamic activity is essential to our understanding of ligament injury and prevention, tissue engineering, prosthetic design, osteoporosis, and osteoarthritis \cite{1,3,5}. With few exceptions, the forces acting on musculoskeletal tissues cannot be directly measured in vivo. Musculoskeletal modeling and movement simulation can estimate individual muscle forces and provide insight to motor control and joint loading. For example, the forward dynamics method is commonly used in musculoskeletal modeling where the neural command signal provides the model inputs \cite{1,4}. The neural command is sent to muscle models that predict muscle forces that are then applied to the bone. The neural command can come from measured EMG (electromyography), but is often estimated by optimization methods that predict the neural command through iteration. As such, decreasing the computation time per simulation can have a significant effect on overall computation time \cite{5}. Body level musculoskeletal models typically involve simplifications of the joints, muscles, and motor control strategies \cite{6} and the knee is commonly represented as a hinge joint. But, real knees experience translation in the sagittal plane and have significant internal–external and varus–valgus rotation. Prediction of lower limb behavior would be greatly enhanced by a model that incorporates the physiological force–displacement response of the knee. In addition, the artificial constraints of an ideal hinge joint can alter the muscle activation patterns predicted by the forward dynamics and neural command optimization method. The goal of this work is to identify and develop data-driven (black-box) surrogate models that are capable of describing the complex three-dimensional behavior of tibio-femoral joint interactions with a computational efficiency and ease of use necessary for incorporation in existing movement simulation models. Specifically, this methodology will be demonstrated by producing a subject specific nonlinear six-axis displacement–force relationship in a compact format.

To meet the project objective, several nonlinear data-driven models, including mixtures of models (multi-model ensembles), were explored and compared. The solution set for black-box model training consisted of relative motion between the tibia and femur and the forces and torques required to produce this motion for a multibody model of the tibio-femoral joint. This model included representation of the ligaments crossing the joint and deformable...
contacts representing cartilage–cartilage interaction. The menisci were not represented. After training, the inputs to black-box models were the positions and orientations of the proximal tibia relative to the distal femur. The outputs were the resulting reaction forces and torques experienced by the tibia from the motion. The developed tibio-femoral surrogate is analogous to a nonlinear six-axis spring and is meant to be used within a musculoskeletal model of the lower limb that also includes representation of muscles, ground reaction forces, and a hip and ankle.

2. Methods

The datasets required to train the data-driven models were generated using a multibody knee model. First, a computational multibody model was created and validated against experimental measurements obtained from a cadaver knee (Section 2.1). Force–displacement datasets were then generated by multibody model simulations and a series of nonlinear, data-driven models were trained, validated, and tested using this data in order to study different aspects of their performance (Section 2.2). Finally, we compared these computationally efficient surrogate models in order to identify architectures and configurations that are suitable for capturing tibio-femoral dynamics (Section 3).

2.1. Data

A validated six degrees of freedom multibody model of a cadaver knee provided the datasets for surrogate training, validation, and testing. The multibody knee model was created in ADAMS (MSC Software Corporation, Santa Ana, CA) using magnetic resonance imaging to create the geometries of the femur, tibia, patella, articular cartilage, and ligaments of a cadaver knee (68-year-old left female knee). A deformable contact law was defined between the cartilage surfaces of the tibia and femur based on Hertzian contact theory and functional cartilage properties [7]. The ligament bundles were represented as nonlinear springs with insertions and zero-strain lengths determined from experimental measurements [8].

The multibody knee model was validated by comparing kinematics to an identically loaded cadaver knee. The knee model was placed in a validated model of a dynamic knee loading machine (Knee Knee Simulator (KKS), University of Kansas, Lawrence, KS) [9]. The KKS reproduces the net loading and motion of physiological activities, such as walking, using five axes controlled through servo-hydraulic actuators (quadriceps force, vertical force applied at the hip, medial-lateral ankle force, ankle vertical torque, and ankle flexion force) [10]. Experimental measurements collected during testing of the cadaver knee included the forces produced by the servo-hydraulic actuators of the machine and the resulting motion of the femur, tibia, and patella. During simulation, the measured forces were applied to the model of the knee in the KKS and the resulting predicted bone motion was compared to measured bone motion. Once validated, the multibody model of the tibio-femoral joint could be extracted from the knee and KKS model to generate the force–displacement data required for the surrogate training, validation, and testing. This step of creating and validating the multibody model was done for two reasons: (1) reaction forces, and (2) the multibody model can easily generate the large amounts of force–displacement data needed for surrogate training. The multibody model of the knee in the KKS was used to generate the motion components of the datasets for surrogate modeling. The simulated motion data extended well beyond the limited motion data measured during experimental testing.

After validation, the model of the cadaver knee in the dynamic knee simulator was used to generate the relative motion between an anatomical coordinate system based on Grood and Suntay [11] which was placed in the tibia (TC) and femur (FC) (Fig. 1). Experimental testing of the cadaver knee in the KKS consisted of a simulated 10 s walk cycle based on ISO specification 1243-1 [12]. The measured forces produced by the actuators of the dynamic knee simulator during experimental testing provided the simulation inputs to the knee and dynamic knee simulator model. The relative motion between the TC and FC was recorded during the simulated walk. Additional motion data was created by repeating the 10 s walk profile and subjecting it to perturbations by applying forces during simulation. The resulting motion data was not meant to encompass the entire envelope of possible tibio-femoral motion, but to provide a continuous and physiologically relevant envelop of motion centered on the ISO walk profile.

Three different datasets were generated for the purpose of training, validating, and testing our data-driven surrogate models. Five force profiles, \( f_i(t) \), were given as the input to the multibody model; namely 1: hip angle, 2: vertical force, 3: lateral force, 4: vertical torque, and 5: ankle force. They were perturbed to include multiple walking paces. The period of the walk cycle, \( T(t) \), was varied by

\[
T(t) = 10 + 2.5q \sin \left( \frac{2\pi t}{10p} \right)
\]

where \( p \) and \( q \) were set to 190 and 1.75, 175 and 2.0, and 210 and 2.25 for training, validation and testing datasets, respectively.

Forces \( f_i(t) \) provided to the model were generated by an nth order Fourier series with \( n_{max} \) being 3 for hip angle, 5 for vertical force, and 4 for remaining force profiles, using a randomized sinusoidal phase angle. The histogram of induced residuals by this method demonstrated an approximately normal distribution.

\[
f_i(t) = \sum_{n=1}^{n_{max}} \left[ C_{n,i} \cos \left( \frac{2\pi n t}{T(t)} \right) - \varphi_{n,i} - \frac{1}{J_i} \cos \left( \frac{2\pi n t}{T(t)} \right) \right]
\]

Here, \( r \) is a uniformly distributed random number. The modified phase term produced pseudo-random combinations of disturbances over the interval of the datasets, while keeping the inputs to the model near the bounds of the ISO-described force profiles. \( C_n \) and \( \varphi_n \) were cosine series’ amplitude and phase calculated from the original ISO profile. \( J_i \) was equal to 1 for all but \( i = 2 \), i.e. vertical force. The original profile for this force exhibited sudden fluctuations in time that were poorly approximated using the same \( J_i \) and \( n_{max} \) as other profiles. To improve this approximation, the order of the Fourier series \( n_{max} \) was increased to 5 and \( J_2 \) was set equal to \( n \) to reduce the disturbance factor in each successive term. Additionally, as the knee joint remains in compression during a normal
The datasets (c) that were used for surrogate training, validation, and testing were recorded. The reaction forces and relative motion from model (b) comprised the datasets (c) that were used for surrogate training, validation, and testing.

The relative motion was applied to the tibio-femoral model (b) and the reaction forces were recorded. The reaction forces and relative motion from model (b) comprised the datasets (c) that were used for surrogate training, validation, and testing.

The tibio-femoral joint with ligaments was then isolated from the machine (b). The model was used to generate relative tibio-femoral motion by systematically modifying the forces representing the hydraulic actuators of the knee loading machine. This model was used to validate the knee model by comparing predicted and actual values for the hip angle (\( \theta \)), the knee angle (\( \phi \)), and the tibio-femoral angle (\( \psi \)).

Using the process mentioned above, three sets of data were obtained: two of them were 301 s long, and were assigned to training and testing sets; and one was 273 s long, which was used for validation.

### 2.2. Models

We considered six Multiple-Input, Single-Output (MISO) models to represent the Multiple-Input, Multiple-Output (MIMO) relative motion–reaction force relationship of the tibia-femoral interactions. From the aforementioned datasets, all models receive their six shared positions and displacement input signals, and each generates their target reaction forces and torques (Fig. 3).

The three multibody model-generated datasets were allocated to training (301 s walk cycle), validation (273 s walk cycle), and testing (301 s walk cycle), with a 50 Hz sampling rate. The validation set was used for early stopping to avoid over-training as well as evaluating and tuning different model parameters, as described later. All signals were normalized linearly between +1 and -1 to facilitate model training.

In this study, we turned our focus to nonlinear dynamic models given the nonlinear nature of most musculoskeletal systems in general (14–16) and the characteristics of our detailed knee joint interactions in particular (see Section 2.1). Being simpler and easier to analyze, linear models have also been considered for biological system identification (17,18). However, as shown by earlier studies (19,20), nonlinear dynamic models, such as those introduced here, outperform linear models for the problem at hand.

The following nonlinear dynamic black-box models were explored for this study: (1) cascaded nonlinear–linear–nonlinear Hammerstein–Wiener (NLHWW) models, (2) nonlinear autoregressive models with exogenous inputs (NLARX), and (3) time delay feed-forward neural network models with input tapped delay line (TDNN).

In general, one can describe the transfer function of such dynamic models as

\[
y_j(t) = f_j(y(t-\Delta t), y(t-2\Delta t), \ldots, y(t-n_j\Delta t), \ddot{u}(t-n_\delta\Delta t), \ldots, \dddot{u}(t-n_\delta\Delta t) - (n_\delta-1)\Delta t) \tag{3}
\]

where \( y_j(t), j = 1, 2, \ldots, 6 \) indicates the output of MISO model for \( F_x, F_y, F_z, T_x, T_y, \) and \( T_z, \) \( \ddot{u}(t) \) is the shared six-dimensional position and orientation input vector (Fig. 3), \( \Delta t = 0.02 \) s is the sampling period here, \( n_\delta \) is input to output delay, and \( n_\delta \) and \( n_\delta \) are the length (depth) of input and output tapped delay lines (TDLs), respectively, in terms of number of samples. Note that \( n_\delta \) is not a transmission delay, but rather the time shift for the starting point of the sliding input window.

Learning capacity of data-driven models depends on their size and structure (21–23). Thus, we explored the structure space of our NLHWW, NLARX and TDNN models by changing their architectural parameters and measuring the resulting performance changes. More specifically, we trained a variety of configurations from the aforementioned family of models and compared their performance using validation dataset. The chosen performance measure was normalized mean squared error, NMSE, calculated as the average of the squared differences between model-predicted vs. desired outputs, then normalized over the desired output variance:

\[
\text{NMSE} = \frac{\sum(y - \hat{y})^2}{\sum(y - \bar{y})^2} \tag{4}
\]

where \( y \) is the discrete-time desired output signal, \( \hat{y} \) is its mean, and \( \bar{y} \) is the corresponding model-generated output. The choice of MSE for fitness metric was further corroborated by our observation on Gaussianity of the walk cycle signal perturbations, as with Bayesian statistics it can be shown that maximum likelihood solution for regression leads to minimizing the sum of squared errors if perturbations are Gaussian-distributed (21). By taking signal variance and length into account, NMSE allows for a more equitable comparison between model outputs of different lengths and swings.

All the models discussed hereafter were simulated using 64 bit MATLAB®, version R2008b (Mathworks, MA) and its System Iden-
The Hammerstein–Wiener model is composed of a static (memoryless) nonlinear block followed by a dynamic linear block, whereas the Wiener model places its dynamic linear block in front of the static nonlinear block [24–26]. Both Hammerstein and Weiner models have been used to model different nonlinear dynamic systems, including biological systems [27,28]. Hammerstein models could be easier to estimate [29], but Hammerstein (or Wiener) models individually might not be adequate for identification of the systems of interest [24]. Therefore, a cascade of Hammerstein–Wiener models have been proposed, in which the dynamic linear block, usually an autoregressive, moving average (ARMA) system, is embedded in between the two static nonlinear blocks [24,26]. The nonlinear block can be realized by polynomial, piecewise-linear, or radial basis function networks with sigmoid or Gaussian nodes. Given their added complexity, nonlinear–linear–nonlinear Hammerstein–Wiener models (NLNHW) are mathematically more capable than their individual building blocks [28]. NLNHW are composed of tapped delay lines that keep a history of input and output signals, and feed them to a linear section in parallel with a nonlinear section (Fig. 5). Thus the linear section creates an ARMA model, while the nonlinear section realizes a nonlinear ARMA (NARMA) model in parallel. The NARMA model by itself, under certain conditions such as observability, is equivalent to a dynamic state-space model, which in turn can emulate a wide range of dynamic models [33].

The linear portion of NLARX provides an additional linear approximation in case of partially nonlinear systems, such as the problem at hand, since our earlier studies indicate that linear ARMA models can capture such dynamics to some extent [20]. NLARX parameter estimation algorithm details, performance criteria, and structural variables including the associated greedy search algorithm are similar to those described for NLNHW. However, for choice of nonlinearity, we examined piecewise-linear functions with tree partitioning, sigmoid ensembles, and wavelet ensembles.

2.2.3. Time delay neural network models with input tapped delay lines (TDNN)

Feed-forward neural networks with input tapped delay lines (TDNN) use TDLS as simple memory structures to keep the last \( n_b \) samples of the input signal \( u(t) \) and present them to their feed-forward multi-layer perceptron section to generate model output (Fig. 6):

\[
y_j(t) = f_j(\tilde{u}(t), \tilde{u}(t - \Delta t), \ldots, \tilde{u}(t - (n_b - 1)\Delta t))
\]

where parameter refers to a solution variable.

The structure of an NLNHW model can be fully described by the configuration of its input and output nonlinearities (in terms of type and the number of participating nonlinear nodes), and the structure of the intervening ARMA block as specified by its number of zeros, poles, and input window delay \( (n_b - 1, n_a, \text{ and } n_k) \). The aforementioned descriptors will be referred to as structural variables henceforth. As for nonlinearity types, we considered the most commonly used functions, namely sigmoid ensemble, piecewise-linear, and polynomial functions [30,31]. As a further clarification, hereafter the term nonlinear node in sigmoid or wavelet ensemble refers to a sigmoid or wavelet basis function within that ensemble’s linear mixture, or a linear section in a piecewise-linear function, or a term in polynomial function. For this study, NLNHW structure was parsed by visiting different combinations of the above-mentioned parameters in order to find stable, low-error configurations.

Since an exhaustive search across all possible combinations of structural variables is not practical, we focused on selected spans of interest using a greedy search [32], finding the best local solution for each variable while having the rest assigned to nominal initial values determined by trial and error.
TDNN was chosen because it can be shown that, given adequate TDL depth, it can approximate any nonlinear dynamic shift-invariant myopic mapping between its input and output domains [34]. The structural parameters of TDNN consist of the number of nonlinear nodes, \( m \), and the length of input tapped delay lines, \( n_b - 1 \). To keep the model as simple as possible, we used one hidden layer with hyperbolic tangent nonlinear nodes. A linear function was used at the output node since saturating nonlinearities such as hyperbolic tangent have limited dynamic range and thus is not suitable for TDNN regression. TDNNs were trained using scaled conjugate gradient descent algorithm [35] in conjunction with MSE cost function and validation-based early stopping. Reported figures are the average training, validation, and test results from 10 training runs per TDNN model, as randomly initialized neural networks may converge to a different local solution during each gradient descent. Thus, the expected merit of each configuration was estimated by averaging the outputs of the network from multiple training runs [22].

2.2.4. Ensemble models

To achieve higher model accuracy, especially for unseen data (generalization), one may combine regression models so that their uncorrelated errors cancel out in the process. Such model-aggregating methods are known as ensemble, mixture, fusion, or committee methods [36]. For instance, it can be shown if individual model error and training errors are statistically independent, for a simple output-averaging method using an ensemble of \( M \) regressors (mean or sum rule), the expected mean squared error could be up to \( M \) times smaller [21]. Ensembles may also create more powerful models that are nonexistent in the individual models’ architecture space. For instance, mixtures of sinusoids, in a Fourier sense, can approximate signals whose morphology is very different from the constituent sinusoids. Ensemble methods have also been suggested for effective modeling of time-varying signals and systems [37].

One of the main challenges in data-driven modeling and system identification is due to model variance and thus poor generalization, e.g. as a result of insufficient training data. Ensemble methods may also lower this variance. For instance, instead of creating one complex high-variance model, one can create and train multiple simpler models with lower independent variances, each capturing a different aspect of the dynamic system within a multi-model ensemble (divide and conquer).

For effective ensemble building, participating models need to have de-correlated errors. One way to do so is by structural diversity, where the ensemble members are chosen from different families and architectures (e.g. NLNHW, NLARX, and TDNN here). It is also possible to have functionally different models from similar structures, e.g. by randomizing weights prior to gradient descent learning for an artificial neural network; where during each run the neural net converges to a different local error minimum in its weight space, so even though they have the same structure, their input–output mapping functions are different [22,38].

Linear mixing is one popular ensemble building method related to Gating networks [21,38–41]. If \( y(t) = f(u(t)) \) is the desired model output, and given outputs \( y_i(t) \) of models \( i = 1, 2, \ldots, M \) in the mixture, the expected ensemble output can be viewed as

\[
E[y(t)] = \sum_{i=1}^{M} y_i(t)p(y_i(t))
\]

where \( p(y_i(t)) \) is the linear mixing weight (probability) of model \( i \) in the regression ensemble. Here the challenge is finding \( p(y_i(t)) \). In this study, we implemented and compare three methods:

\( (1) \) mean (sum) rule, assuming all the models are equally probable, i.e. \( p(y_i) = 1/M \) [42,43],

\( (2) \) setting \( p(y_i) \) according to the inverse of the corresponding model’s validation NMSE, i.e.

\[
p(y_i) = \frac{\text{NMSE}_{\text{val},i}^{-1}}{\sum_{i=1}^{M} \text{NMSE}_{\text{val},i}^{-1}}
\]

also known as weighted sum rule [44]. NMSE, is the validation NMSE of the \( i \)th participating model in the ensemble, and

\( (3) \) stacked generalization [45,46], by using supervised learning to train a combining function that fuses the output of participating models into the desired (ensemble) output, usually using a training set different from those used to training individual models. Here we used a finite impulse response (FIR) [47] linear fusion, essentially a linear filter with multiple TDNN input structures, trained to minimize output NMSE on validation dataset.

3. Results and discussions

Performance and sensitivity results for NLNHW, NLARX, and TDNN surrogates as function of model structural parameters, as well as the results for different ensemble (mixture) models, are given below. Accordingly, the best individual and ensemble model configurations, along with their results are presented. For brevity, only the results for force in x direction (\( \text{Fx} \)) are depicted, unless other force or torque models established significantly different characteristics that necessitated further deliberations.

3.1. NLNHW

To find a stable configuration with lowest validation error, we changed the type and number of nonlinear functions, number of poles, zeros and input to output delay, one at a time using greedy search algorithm. The following is a short discussion of the results.

3.1.1. Sensitivity to structural parameters

3.1.1.1. Input nonlinear stage. Performance of NLNHW models were highly sensitive to the changes to input nonlinear function if it was piecewise linear. As the number of piecewise-linear sections was changed from 1 to 10, the average \( \text{Fx} \) validation NMSE came up to 3.1143 \times 10^8, while its median and variance were 0.1764 and 2.8778 \times 10^8, respectively. This difference between mean and median, as well as high variance, illustrates the earlier mentioned high sensitivity with respect to input piecewise-linear function structure. However, using sigmoid ensembles and changing the number of nodes from 1 to 10, the resulting \( \text{Fx} \) validation NMSEs had
<table>
<thead>
<tr>
<th>Model target</th>
<th>Nonlinear functions (input/output)</th>
<th>Number of nonlinear nodes (input/output)</th>
<th>Input TDL length</th>
<th>Output TDL length</th>
<th>Input to output delay</th>
<th>Training NMSE</th>
<th>Validation NMSE</th>
<th>Test NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fx</td>
<td>Sigmoid ensemble/piecewise linear</td>
<td>2/2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.1630</td>
<td>0.1665</td>
<td>0.1859</td>
</tr>
<tr>
<td>Fy</td>
<td>Sigmoid ensemble/piecewise linear</td>
<td>3/5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0.0656</td>
<td>0.0865</td>
<td>0.1298</td>
</tr>
<tr>
<td>Fz</td>
<td>Sigmoid ensemble/piecewise linear</td>
<td>8/3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.0975</td>
<td>0.1293</td>
<td>0.1618</td>
</tr>
<tr>
<td>Tx</td>
<td>Piecewise linear/piecewise linear</td>
<td>9/5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.1634</td>
<td>0.2089</td>
<td>0.277</td>
</tr>
<tr>
<td>Ty</td>
<td>Piecewise linear/piecewise linear</td>
<td>10/3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.0302</td>
<td>0.0355</td>
<td>0.0507</td>
</tr>
<tr>
<td>Tz</td>
<td>Sigmoid ensemble/sigmoid ensemble</td>
<td>2/3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.0541</td>
<td>0.0584</td>
<td>0.0886</td>
</tr>
</tbody>
</table>

a mean, median, and variance of 0.263, 0.2, and 0.0474, respectively. Similar experiments with polynomials of degrees 1–10 resulted in NMSE mean, median, and variance of 0.495, 0.54, and 0.0116, respectively. Thus, sigmoid ensembles were deemed to be the best overall choice for the input nonlinearity given their lower validation NMSE and sensitivity (i.e. lower error mean and variance). The only exception to this rule was the Tx model, where polynomial functions performed better, yielding validation NMSE of 0.5997 and 0.4653, and sigmoid ensemble’s 0.3913 and 0.0673. The only exception was Ty model, where sigmoid ensemble’s mean and variance were 0.3813 and 0.072, while polynomial provided 0.4147 and 0.0872, and piecewise linear yielded 0.4095 and 0.1696.

In summary, in terms of robust performance vs. nonlinearity types, sigmoid ensembles were overall better choices as input nonlinearities except for Tx and Ty models, while piecewise-linear functions did better at the output stages of all models except Tz (Table 1).

3.1.1.3. Delays, poles, and zeros. Finding a definite pattern of model sensitivities with respect to the delays, poles and zeros was challenging. Furthermore, not all delay values provided valid, stable results. For Fx, the resulting NMSE mean, median and variance with the change of delay between input and output \( n_k \) (1–10) were 1.1915 × 10^{-15}, 0.5916 and 8.288210^{-31}. The mean, median and variance for changing number of zeros \( n_b \), 1–10) for Fx were 0.5957, 0.3715 and 1.3427, while changing poles \( n_a \) (1–10) yielded 0.5018, 0.4653 and 0.0740. All models were the least sensitive to the change \( n_k \) with an exception of Fx. For Ty, the mean, median and variance of validation MSE as a result of changing \( n_k \) from 1 to 10 were 0.4148, 0.3923 and 0.0602, respectively. Similar numbers from changing number of poles, \( n_a \), 1–10) were 0.4962, 0.4596 and 0.0918, and 1.1473 × 10^{-14}, 0.3781 and 1.3897 × 10^{-10} for changing zeros.

![Fig. 7.](a) Performance of the best NLNHW Fx model on training data (model prediction vs. target, 100–150 s segment). (b) Corresponding regression plot depicting correlation between target and model outputs. Ideally, correlation coefficient, \( R \), should be 1 (model output = target values). (c) and (d) Same as the above, but for unseen test data. Smaller difference between training and test data performance is an indication of model generalization power.

Please cite this article in press as: M. Mishra et al., Nonlinear surrogate modeling of tibio-femoral joint interactions, Biomed. Signal Process. Control (2010), doi:10.1016/j.bspc.2010.08.005
Table 2
Best results across different NLARX configurations.

<table>
<thead>
<tr>
<th>Model target</th>
<th>Nonlinear function</th>
<th>Number of nonlinear nodes</th>
<th>Input TDL length</th>
<th>Output TDL length</th>
<th>Input to output delay</th>
<th>Training NMSE</th>
<th>Validation NMSE</th>
<th>Test NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fx</td>
<td>Tree partition piecewise linear</td>
<td>3</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>0.0197</td>
<td>0.0361</td>
<td>0.0462</td>
</tr>
<tr>
<td>Fy</td>
<td>Tree partition piecewise linear</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>0</td>
<td>0.1850</td>
<td>0.2038</td>
<td>0.2254</td>
</tr>
<tr>
<td>Fz</td>
<td>Tree partition piecewise linear</td>
<td>15</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0.1338</td>
<td>0.1623</td>
<td>0.1941</td>
</tr>
<tr>
<td>Tx</td>
<td>Tree partition piecewise linear</td>
<td>16</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0.1940</td>
<td>0.2652</td>
<td>0.2790</td>
</tr>
<tr>
<td>Ty</td>
<td>Tree partition piecewise linear</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0.0990</td>
<td>0.0732</td>
<td>0.1182</td>
</tr>
<tr>
<td>Tz</td>
<td>Tree partition piecewise linear</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0.1344</td>
<td>0.1203</td>
<td>0.1802</td>
</tr>
</tbody>
</table>

3.1.2. Validation performance vs. structural parameters
3.1.2.1. Input nonlinear stage. For Fx, the lowest validation NMSE (0.0443) was achieved with a piecewise-linear function, while the worst NMSE (0.289) occurred with a polynomial function. An exception to this rule was the Fy model, where the polynomial function yielded an NMSE of 0.0329, compared to 0.0355 of piecewise linear and 0.0459 of sigmoid ensemble. However, the piecewise-linear function was not deemed to be a better choice for the input stage of NLNHW, given its earlier mentioned higher sensitivity. These results were comparable for other models, Fy through Tz (see Table 1).

3.1.2.2. Output nonlinear stage. As shown in Table 1, piecewise-linear function was the best overall choice except for Tz, where sigmoid ensemble performed better.

3.1.2.3. Delays, poles, and zeros. The optimum number of poles and zeros was largely dependent on the models’ target (Fx through Tz), as well as the input and output nonlinearities. However, the optimum delay between input and output, $n_k$, was mostly 2 samples (0.04 s).

Based on the above, the best overall model structures are summarized in Table 1. Fig. 7 portrays the performance of a selected model.

3.2. NLARX

In order to find the best NLARX configuration, we changed the type nonlinear function, the number of its nodes, input TDL depth, output (feedback) TDL depth, and input to output delay. Parameters were changed one at a time using greedy search.

3.2.1. Sensitivity to structural parameters
3.2.1.1. Nonlinear stage. NLARX models were generally much more sensitive to the structural variables such as the type and size of their nonlinear stage, as reflected in the resulting high variances and the large differences between mean and median values. For instance, for Fx, the mean, median, and variance of validation error using sigmoid ensembles of different sizes (1–10 nodes) were $1.6977 \times 10^{143}$, $2.4657$, and $2.5939 \times 10^{287}$, respectively. The same numbers using wavelet ensembles were $0.4198$, $0.4318$, and $0.0012$. Tree partition piecewise-linear nonlinearity yielded $0.2436$, $0.2727$ and $0.0092$ for validation NMSE mean, median and variance as partition tree level was changed from 1 to 10 (best Fx result). However, the low validation error mean and variance for the latter nonlinearity was not consistent across other models and overall, wavelet nonlinear functions worked the best in terms of validation mean and variance. The exception was Fy, where sigmoid

![Fig. 8](image-url) (a) Performance of the best NLARX Fx model on training data (model prediction vs. target, 100–150 s segment). (b) Corresponding regression plot depicting correlation between target and model outputs. Ideally, correlation coefficient, $R$, should be 1 (model output = target values). (c) and (d) Same as the above, but for unseen test data. Smaller difference between training and test data performance is an indication of model generalization power.
ensemble performed better yielding 81.0409 and $3.4396 \times 10^4$ for mean and variance of validation NMSE, compared to wavelet's $2.3944 \times 10^7$ and $5.6352 \times 10^{151}$.

### 3.2.1.2. Input and output delay line depths, and input to output delay
NLARX models were sensitive to all three delay parameters in general, and became unstable for quite a few combinations. For $F_x$, the mean and variance of validation NMSE were $4.7110 \times 10^{147}$ and $5.5485 \times 10^{296}$ as output TDL depth ($n_a$) was changed from 1 to 10. Error mean and variance vs. input TDL ($n_b$, 1–10) changes were $0.033 \times 10^{25}$ and $2.6399 \times 10^{48}$, and $9.1859 \times 10^{297}$ and infinity for input to output delay ($n_k$, 1–10) changes.

### 3.2.2. Validation performance vs. structural parameters
For the following results, the 1–10 parameter search span used in sensitivity study was extended to find better parameter values during the greedy search, if the trend of sensitivity search results warranted such an expansion.

#### 3.2.2.1. Nonlinear stage
The optimum NLARX configurations, in terms of validation error, are shown in Table 2. The lowest NMSE was achieved by tree partition piecewise-linear function for all force and torque models.

#### 3.2.2.2. Input and output delay line depths, and input to output delay
The optimum values for these parameters largely depended on the output target ($F_x$ through $T_z$), and the choice of nonlinear function. Using the best-performing nonlinearity (tree partition piecewise-linear function), best validation numbers were obtained with no delay between input and output ($n_k = 0$). The training and test performance of a selected model is shown in Fig. 8.

### 3.3. TDNN

TDNN's structural variables of interest were input TDL depth, and the number of nodes in the following nonlinear section, also known as hidden layer size. When compared to NLNHW and NLARX families, TDNN models provided the overall best results in terms of stability, validation performance, and sensitivity to the structural variables. Following the common practices in design of TDNN for regression, hidden layer nonlinearity functions were set to hyperbolic tangent, and a linear function was used for the output node [22,38].

#### 3.3.1. Sensitivity to structural parameters

<table>
<thead>
<tr>
<th>Model targets</th>
<th>Nonlinear functions</th>
<th>Number of nonlinear nodes</th>
<th>Input TDL length</th>
<th>Input to output delay</th>
<th>Training NMSE</th>
<th>Validation NMSE</th>
<th>Test NMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_x$</td>
<td>Hyperbolic tangent</td>
<td>12</td>
<td>25</td>
<td>0</td>
<td>0.0174</td>
<td>0.0260</td>
<td>0.0509</td>
</tr>
<tr>
<td>$F_y$</td>
<td>Hyperbolic tangent</td>
<td>13</td>
<td>30</td>
<td>0</td>
<td>0.0349</td>
<td>0.0480</td>
<td>0.0889</td>
</tr>
<tr>
<td>$F_z$</td>
<td>Hyperbolic tangent</td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>0.0274</td>
<td>0.0344</td>
<td>0.0688</td>
</tr>
<tr>
<td>$T_x$</td>
<td>Hyperbolic tangent</td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>0.0433</td>
<td>0.0537</td>
<td>0.0889</td>
</tr>
<tr>
<td>$T_y$</td>
<td>Hyperbolic tangent</td>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0.02218</td>
<td>0.0230</td>
<td>0.0540</td>
</tr>
<tr>
<td>$T_z$</td>
<td>Hyperbolic tangent</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

alone improved the test results from 10% to 32% across sum rule, and stacked generalization using an FIR filter. Sum rule following output-level fusion methods: mean (sum) rule, weighted and \( \frac{1}{n} \sum_{i=1}^{n} x_i \) which is given in Table 4. A sample of stacked generalization ensemble output for \( Fx \) is given in Fig. 9.

15. This search span was extended given the improving TDNN performance past the original 1–10 hidden node search range. Other force and torque TDNN models displayed a similar pattern.

3.3.1.2. Input delay line depth. The mean, median and variance of \( Fx \) validation error, while varying the input TDL depth (\( n_x \)) from 1 to 100 in steps of 10 were 0.0688, 0.0672 and 1.6663 \( \times 10^{-04} \). \( n_x \) was set to 0. Note that due to low sensitivity of TDNN results with respect to original 1–10 TDL depths, the TDL search range was extended to 100 samples.

3.3.2. Validation performance vs. structural parameters

For the following, parameter search spans are the same expanded ranges used in sensitivity study (Fig. 9).

3.3.2.1. Nonlinear section. Validation NMSE changed but a little when varying the size of hidden layer. The best configurations are given in Table 3.

3.3.2.2. Input delay line depth. Validation errors did not vary much by changing the TDL depths. Results are shown in Table 3.

3.4. Ensembles

To create ensemble-based surrogate model of each force or torque signal, we pooled the top-performing instances from all architecture families to ensure a diversified, low-bias mixture of models. It is worth mentioning that choosing ensemble models only based on their mutual uncorrelated errors did not lead to the best results, as this inclusion criterion would also pick models with significantly higher NMSE in the mixture (mainly from NLARX and NLNHW families), causing unwanted mixture bias. Thus, we chose the top 2 models based on their validation NMSE from each of the three architectures, NLNHW, NLARX, and TDNN, to ensure diversity and also low bias (i.e. low NMSE). From this initial pool, those models with significantly higher validation errors were removed. For instance, for \( Fx \) ensemble, we picked two models from NLNHW (NLNHW 1, input nonlinearity: 4-node piecewise-linear input. Linear section: 2 zeros, 2 poles and 2 delay units between input and output, output nonlinearity: 3-node sigmoidal, NLNHW 2, input nonlinearity: 3-node piecewise linear. Linear section: 2 zeros, 2 poles, and 2 delay units between input and output, output nonlinearity: 10 nodes, piecewise linear), one from NLARX (nonlinear section: 3 tree partition piecewise-linear nodes, output TDL, input TDL and input to output delay: 1, 10, and 1 units, respectively), and two from TDNN (TDNN 1, hidden layer: 13 hyperbolic tangent nodes, input TDL: 25 units. TDNN 2, hidden layer: 13 hyperbolic tangent nodes, input TDL: 25 units). Note that TDNN 1 and 2 are different as their weights coming different training sessions. Similarly, for \( Fy, Ty \), and \( Tz \), two NLNHW and two TDNN models were used. For \( Fz \) and \( Tx \), only the two TDNN models were picked. We examined the following output-level fusion methods: mean (sum) rule, weighted sum rule, and stacked generalization using an FIR filter. Sum rule alone improved the test results from 10% to 32% across \( Fx–Tz \), compared to the best individual participating model. The weighted averaging ensemble only performed slightly better than the simple sum (mean) rule ensemble. The test performance of fusion for FIR stacked generalization method was considerably better, yielding 13–40% improvement in test results across \( Fx \) to \( Tz \) when compared to best individual participating model. However, the higher initial startup delay introduced by FIR’s input TDL was the trade-off to consider: for \( Fx \), a 25–30% validation NMSE improvement was gained at the expense of an extra 0.1–1.5 s in model startup times. Other stacked generalization ensembles for \( Fy–Tz \) demonstrated a similar performance-delay pattern.

The ensemble performances are summarized in Table 4. A sample of stacked generalization ensemble output for \( Fx \) is given in Fig. 10a and b.

3.5. Computational considerations

Unlike a hinge joint, the presented surrogate models allow for the complete six degrees of freedom of the knee while maintaining many of the computational advantages of an overly simplified joint representation. The surrogate models currently run inside the MATLAB® environment, but could easily be compiled into compact universal binary executables. In terms of memory footprint, a typical set of six MATLAB models take up 5 MB. By comparison, the MSC.ADAMS 6 DOF multibody model requires 128 MB. More importantly, the execution speed of the surrogates is orders of magnitude better than the multibody models. Using a Xeon X5482-based computer with 16 GB of RAM and 64 bit Windows XP (engaging one CPU core), processing a 301 s dataset with an earlier mentioned TDNN surrogate model took an average of 8.08 s to calculate all six degrees of freedom serially. In contrast, the MSC.ADAMS multibody model took 2807 s to finish the same task. Additionally, being subject specific, the multibody model requires a lengthy construction process for each individual, while further study in this domain would lead to a fully developed surrogate which would be trained on multiple knees so that the final composite would be applicable to a range of knees similar to the source knees.

Overall, these improvements provide significant portability that allows the accuracy of data derived from the multibody model to be propagated into different modeling methods and scales.

4. Conclusion and future work

In this work, we introduced data-driven surrogate models that could successfully capture realistic and complex three-dimensional behavior of tibio-femoral joint interactions, while maintaining a small computational footprint. Training, validation, and test datasets were obtained from an experimentally validated multibody model of a cadaver knee. Three different nonlinear dynamic families of models were identified, and variations of their structures were ranked using normalized mean squared error on validation and test data. Overall, TDNNs provided better results and more robust designs given their lower errors and structural sensitivities. NLARX models, on the other hand, were highly sensitive to struc-
Mixture of Experts, and Dempster–Shafer methods may also be underperforming models could be avoided the ensemble’s gate function. Thus, unduly general and therefore can be trained and activated on a different motion profile by the ensemble (divide and conquer). For instance, each model gained by using gating networks and competitive learning to specify methods were considered. Further improvements could be errors by increasing model pool diversity.

Though some of the aforementioned models could be harder to train, this shall not be an issue during implementation of trained and validated models. Post-training, these surrogate models, including the ensembles, have very low computational complexity and can run in real time even on low-end and embedded computing platforms. Paired with their fidelity to the complex dynamics of a real knee, this is a major advantage of our approach compared to traditional musculoskeletal modeling methods, enabling iterative optimization studies inside of the multibody framework without resorting to a grossly simplified tibio-femoral interaction models.

While establishing the feasibility of creating computationally light, data-driven models of the highly nonlinear and complex tibio-femoral force–displacement interactions, and providing specific configurations to do so, we also identified a number of directions for future improvements.

First, searching model structure spaces can be improved. Greedy search, though simple and fast, can be trapped in local optima during multivariate optimization with interdependent variables. Exhaustive searches, on the other hand, will be prohibitive in terms of computational costs. Thus, we suggest population search methods, such as evolutionary algorithms, for an improved exploration of model structure spaces. Such population-based, guided stochastic searches can also alleviate the need for determining suitable initial search conditions.

Other regression models, such as those based on support vector machines, and distributed tapped delay line neural networks (DTDNN), may also be tested. Though there is no guarantee that these models will perform better than the ones already considered, there is a possibility that these models could decrease ensemble errors by increasing model pool diversity.

In this paper, only a limited number of simple ensembles mixing methods were considered. Further improvements could be gained by using gating networks and competitive learning to specialize each model in a specific region of the input–output space of the ensemble (divide and conquer). For instance, each model can be trained and activated on a different motion profile by the ensemble’s gate function. Thus, unduly general and therefore underperforming models could be avoided. Other ensemble and model fusion techniques such as AdaBoost, Hierarchical Mixture of Experts, and Dempster–Shafer methods may also be explored.

Though derived from a validated subject specific model of one cadaver knee, our motion dataset was extended by the variation methods detailed in Section 2.1. In the future, we would like to include more walk cycles based on models of other cadaver knees to study the extent of subject-dependence in data-driven surrogate models.

Finally, for a comprehensive closed loop evaluation, we wish to place different surrogate models of the tibio-femoral joint within the multibody model of the dynamic knee simulator. The performance of the model will then be compared to experimentally measured kinematics for the cadaver knee. In addition, the surrogate model will be placed within a forward dynamics movement simulation using measured gait lab data (motion, ground reaction forces, and electromyography) where the computational efficiency of these surrogate models will enable optimization studies to be performed in a reasonable timeframe without gross simplification of a multibody model. Future work also includes adding the prediction of medial and lateral contact forces or pressures for given tibio-femoral motions as additional surrogate outputs.

The concept of surrogate models that efficiently capture input–output relationships, such as those presented in this work, can also be extended to other areas of musculoskeletal and biological modeling. For example, multi-scale modeling of the musculoskeletal system provides the promise of connecting various spatial scales during concurrent simulation (e.g. connecting body level loading to the tissue or cellular environment) [6]. Multi-scale simulations typically involve many models at the lower spatial scales. Solutions from these lower scale models are then requested repeatedly from higher scale models. Surrogate models, such as those described here, could be used to define the input–output relations of the smaller scale models, thus allowing for multi-scale simulations within a computationally feasible footprint.

Acknowledgements

This research was supported by the National Science Foundation, grant number 506297, under the IMAG program for Multi-scale Modeling. The authors gratefully acknowledge the contributions of researchers in the Experimental Joint Biomechanics Research Lab at the University of Kansas, Lawrence KS, for their contributions to this work (Lorin Maletsky, Kevin Dodd, Amber Reeve, Chadd Clary, and Amit Mane).

References

