Huxley Muscle Model Surrogates for High-speed Multi-scale Simulations of Cardiac Contraction

Bogdan Milićević[1,2], Miloš Ivanović[2,3], Boban Stojanović[2,3], Miljan Milošević[2,4,5], Miloš Kojić[2,6,7] and Nenad Filipović[1,2,*]

[1] Faculty of Engineering, University of Kragujevac, Kragujevac 34000, Serbia.
[2] Bioengineering Research and Development Center (BioIRC), Kragujevac 34000, Serbia.
[3] Faculty of Science, University of Kragujevac, Kragujevac 34000, Serbia.
[4] Institute for Information Technologies, University of Kragujevac, Kragujevac 34000, Serbia.
[5] Belgrade Metropolitan University, Belgrade 11000, Serbia.
[6] Serbian Academy of Sciences and Arts, Belgrade 11000, Serbia.
[7] Houston Methodist Research Institute, Houston TX 77030, USA.

*Correspondence should be addressed to Nenad Filipović; fica@kg.ac.rs

Abstract

The computational requirements of the Huxley-type muscle models are substantially higher than those of Hill-type models, making large-scale simulations impractical or even impossible to use. We constructed a data-driven surrogate model that operates similarly to the original Huxley muscle model but consumes less computational time and memory to enable efficient usage in multiscale simulations of the cardiac cycle. The data was collected from numerical simulations to train deep neural networks so that the neural networks' behavior resembles that of the Huxley model. Since the Huxley muscle model is history-dependent, time series analysis is required to take the previous states of the muscle model into account. Recurrent and temporal convolutional neural networks are typically used for time series analysis. These networks were trained to produce stress and instantaneous stiffness. Once the networks have been trained, we compared the similarity of the produced stresses and achieved speed-up to the original Huxley model, which indicates the potential of the surrogate model to replace the model efficiently. We presented the creation procedure of the surrogate model and integration of the surrogate model into the finite element solver. Based on similarities between the surrogate model and the original model in several types of numerical experiments, and also achieved speed-up of an order of magnitude, it can be concluded that the surrogate model has the potential to replace the original model within multiscale simulations. Finally, we used our surrogate model to simulate a full cardiac cycle in order to demonstrate the application of the surrogate model in larger-scale problems.

Keywords: surrogate modeling, recurrent neural networks, Huxley's muscle model, finite element analysis, multi-scale modeling

Introduction

Clinicians can use simulations of cardiac muscle behavior to evaluate various real and fictional events. Present understanding of the molecular processes behind muscle contraction inspired biophysical muscle models. Biophysical muscle models, often called the Huxley-type models, are based on the underlying physiology of muscles, making them suitable for modeling non-uniform and unsteady contractions. Furthermore, biophysical muscle models

can be used to investigate the influence of the genetic properties on the behavior of muscles. Genetic mutations can lead to modulated cross-bridge kinetics and contractile characteristics of muscle cells. These mutations account for a significant percentage of cardiomyopathies.

Multiscale finite element simulation with the Huxley muscle model at a microlevel was presented by Stojanovic et al. [1,2]. To analyze muscle behavior via in silico analysis, they modeled biophysical processes on multiple spatial and temporal scales. They performed a multi-scale simulation in which continuum muscle mechanics was modeled using the finite element method, while the material characteristics of muscle at the microscopic scale were defined by Huxley's muscle contraction model [1,2]. During transient finite element simulation, they employed Huxley's model to calculate stress and instantaneous stiffness, given the muscle activation, stretch, and other material parameters and properties [1]. These finite element simulations can be quite computationally intensive. The microscale calculations are the most time-consuming component of these simulations. Single-time step simulations with hundreds of finite elements take up to several thousand seconds to execute. To speed up the simulations, they employed hybrid MPI-GPU parallelization to simulate a 2D model of the tongue [3,4]. The general flaw of the biophysical muscle models is a large computational requirements in terms of memory and time consumption. The speed-up achieved by parallelization can be significant, but this requires acces to high performance computing clusters. In their research, it is obvious that even with a computing cluster, a multi-scale finite simulation took a long time to execute. To lower the computational requirements of such simulations, we created a more computationally efficient surrogate model to replace the real Huxley muscle model. The advantage of the surrogate model is that high speed-up can be achieved using just a single processor. The drawback can be the loss of accuracy.

Machine learning is becoming increasingly popular in scientific research. Some of the advantages in this area are shown by S. Aydın [5,6] using support vector machines and long short-term memory (LSTM) units to classify discrete emotional states. High classification accuracy is achieved indicating that these methods are very suitable for prediction tasks. A surrogate model based on artificial neural networks for composite materials considering progressive damage was firstly presented by Yan et al. [7]. A multi-layer perceptron was employed to construct the surrogate model by conducting regression for the constitutive law and classification for the damage information [7]. A surrogate model for strain-softening Perzyna visco-plasticity as the nonlinear material model at the micro-level was presented by Ghavamain and Simone [8]. They used modified long short-term memory (LSTM) units to predict stresses based on provided stretches. In both papers [7,8], a good resemblance between original and surrogate models was achieved. The multiscale simulation with generalized continua was presented by Feyel et al. [9]. These models are different from muscle models. For muscle models, more input features are needed and additional learning mechanisms had to be employed to create the muscle surrogate models. In research presented by Ghavamain and Simone [8], only strains are required for stress prediction, which isn't sufficient in the case of muscles, since their behavior is dependent also on activation and previous stresses. They predict stress directly, but in the case of muscle modeling, this would result in low numerical precision and poor generalization. In our research, the stress increment is predicted instead. Also, in research presented by Yan et el. [7] and Ghavamain and Simone [8], the execution time is not explicitly measured, while we take it as a major motivation and first class performance metrics in our approach. As far as the authors know, the surrogate model of the Huxley muscle model, such that the model is operating dynamically within the multi-scale simulation has not been created before.

Our goal was to create a surrogate model of the Huxley muscle model for a predefined set of parameters, varying only the muscle activation and external loads. During finite element analysis, we provide stretch to the material model and expect the stress and instantaneous stiffness, which depends on the state and parameters of the material. Therefore, stretch and muscle activation at the current time step, for each of the integration points, are necessary to calculate stress and instantaneous stiffness, but they are not sufficient since muscle behavior is history-dependent. To take this history dependence of muscle behavior into account, we used time series consisting of activation at current and previous time steps, stretch at current and previous time steps, stress at previous time steps, and instantaneous stiffness at previous time steps to predict stress instantaneous stiffness for the current time step. We also present the complete procedure for surrogate model creation, using recurrent and convolutional neural networks, together with the integration of the surrogate model into the finite element analysis framework. The results obtained from different numerical experiments using the original and the surrogate Huxley model show the similarities between these two models. The proposed surrogate model is tens of times faster than the original model, making it significantly more convenient in simulating the left ventricle mechanical behavior.

Materials and Methods

In this section, we briefly present the finite element procedure used at the macro level together with the Huxley muscle model used at the microlevel of the multi-scale simulation. Further, we specify neural networks utilized for the creation of the surrogate model. In more detail, we demonstrate the surrogate model creation procedure and integration of the surrogate model into the finite element simulation, since, as far as we know, this approach has not been carried out before for surrogate muscle models.

Finite element method

From a mechanical point of view, a muscle can be considered a mechanical system. The most common method for solving complex materially and geometrically nonlinear structural problems is the finite element method. In an incremental-iterative scheme, the equilibrium configuration of a muscle can be calculated, considering the muscle as a structure composed of active fiber elements, able to contract under activation within the deformable connective tissue continuum as demonstrated by Kojic and Bathe [10,11]. The governing equilibrium equation of an FE structure in the deformed configuration at a time step (t) and iteration (i) is formulated as:

$$({}^{t+\Delta t}K_{pass} + {}^{t+\Delta t}K_{act})\,{}^{(i-1)}\delta U^{(i)} = {}^{t+\Delta t}F_{ext} + {}^{t+\Delta t}F_{pass}^{(i-1)} + {}^{t+\Delta t}F_{act}^{(i-1)} \quad (1)$$

where ${}^{t+\Delta t}F_{ext}$, ${}^{t+\Delta t}F_{pass}^{(i-1)}$, ${}^{t+\Delta t}F_{act}^{(i-1)}$ are vectors of external loads, passive internal nodal forces, and active molecular forces wrapped into finite element nodal forces, respectively; ${}^{t+\Delta t}K_{pass}^{(i-1)}$ is the stiffness matrix of passive muscle components and ${}^{t+\Delta t}K_{act}^{(i-1)}$ is the cumulative stiffness of actomyosin bonds; $\delta U^{(i)}$ are increments of nodal displacements at iteration (i). Active force generation ${}^{t+\Delta t}F_{act}^{(i-1)}$ and stiffness ${}^{t+\Delta t}K_{act}^{(i-1)}$ are directly dependent on the rate of muscle deformation in the principal direction of muscle fibers. The total stress $\bar{\sigma}_m$ is expressed as the contribution of active muscle forces and the contribution of passive elasticity of connective tissue and noncontractile parts of tissue:

$$\bar{\sigma} = \phi\bar{\sigma}_m + (1-\phi)\bar{\sigma}^E \quad (2)$$

where ϕ is the fraction of muscle fibers in total muscle volume, $\bar{\sigma}_m$ is the active stress generated in muscles and $\bar{\sigma}^E$ is the stress in the passive part of the muscle. A similar procedure was applied with the phenomenological Hill model as demonstrated by Stojanovic et al. [12]. More details about multiscale finite element analysis with the Huxley muscle model at a micro-level as demonstrated by Mijailovich et al. [15]. Figure 1a shows the finite element model. At each integration point of the finite element, muscle fiber is observed (Figure 1b) and the stress and stress derivative (instantaneous stiffness) are calculated by the Huxley micromodel (Figure 1c) or surrogate model (Figure 1d). The calculations to obtain active stresses are shown in further detail in the next section.

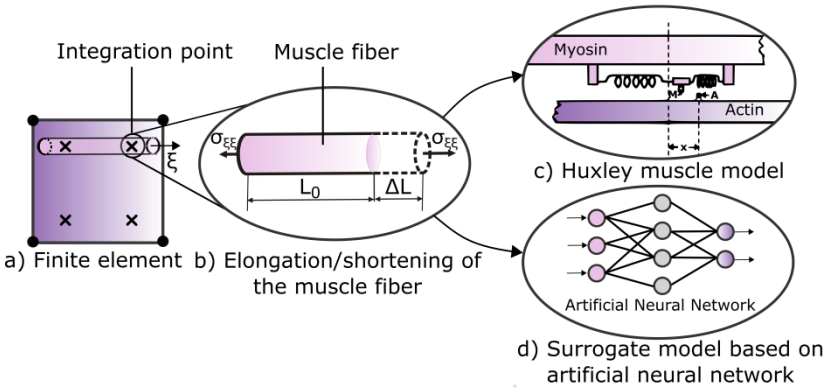

Figure 1: Finite element with c) original Huxley muscle model c) and d) surrogate model at the microlevel.

Huxley muscle model

Huxley considered the dynamics of the filaments within muscle and the probability of establishing connections (cross-bridges) of myosin heads to actin filaments inside sarcomeres [15, 16]. The *n(x,t)* function describes the rate of connections between myosin heads and actin filaments, as a function of the position of the nearest available actin-binding site relative to the equilibrium position of the myosin head *x*:

$$\frac{\partial n(x,t)}{\partial t} - v\frac{\partial n(x,t)}{\partial x} = [1 - n(x,t)]f(x,a) - n(x,t)g(x), \forall x \in \Omega \tag{3}$$

where *f(x,a)* and *g(x)* represent the attachment and detachment rates of cross-bridges respectively, *v* is the velocity of filaments sliding, positive in the direction of contraction, and *a* is a muscle activation given as a function of time [14]. The partial differential Equation (3) can be solved using the method of characteristics with initial condition $n(x,0) = 0$. Once the $n(x,t)$ values are acquired, we can calculate the generated force *F* within the muscle fiber and also stiffness *K* using the equations:

$$F(t) = k\sum_{-\infty}^{\infty} n(x,t)x \; dx \tag{4}$$

$$K(t) = k\sum_{-\infty}^{\infty} n(x,t) \; dx \tag{5}$$

where *k* is the stiffness of the cross-bridges. The stress and instantaneous stiffness can be calculated as:

$$\sigma_m = F\frac{\sigma_{iso}}{F_{iso}} \tag{6}$$

$$\frac{\partial \sigma_m}{\partial e} = \lambda L_0 K \frac{\sigma_{iso}}{F_{iso}} \tag{7}$$

where F_{iso} is a maximal force achieved during isometric conditions, σ_{iso} maximal stress achieved during isometric conditions, L_0 the initial length of sarcomere and λ is stretch. Calculated stresses and instantaneous stiffness are directly utilized at the upper, macro-level during finite element analysis to form ${}^{t+\Delta t}F_{act}^{(i-1)}$ and ${}^{t+\Delta t}K_{act}^{(i-1)}$. The muscles are activated under the influence of calcium. The muscle activation can be derived from the calcium

concentration function, which is prescribed in our numerical experiments. To generate calcium concentration function, we employ the parametric equation [17]:

$$Ca_i(t) = Ca_0 + (Ca_{max} - Ca_0)\frac{t}{\tau Ca}e^{1-\frac{t}{\tau Ca}} \quad (8)$$

where $Ca_i(t)$ is a time-dependent intracellular concentration of Ca^{2+}, which has a resting level Ca_0 and achieves its maximum value Ca_{max} at time $t = \tau Ca$. The calcium concentration is converted to an activation α using the formula:

$$\alpha = \frac{(Ca)^n}{(Ca)^n + (C_{50})^n} \quad (9)$$

where C_{50} is the value required to achieve 50% availability of calcium, calculated using $pC_{50} = pC_{50ref}(1 + \beta_2(\lambda - 1))$; $C_{50} = 10^{6-pC_{50}}$ [μM], and n is defined as

$$n = n_{ref}(1 + \beta_1(\lambda - 1)) \quad (10)$$

where λ is stretch, pC_{50ref}, n_{ref}, β_1 and β_2 are constant coefficients. We used values $n_{ref} = 5.2$, $pC_{50ref} = 6.18$, $\beta_1 = 1.95$ and $\beta_2 = 0.31$, taken from [17].

Recurrent and convolutional neural networks

Recurrent neural networks are often used for time series analysis [18]. These networks allow previous outputs to be used as inputs while having hidden states. Recurrent neural networks are hard to train because they are especially susceptible to the exploding and vanishing gradient problem [19]. There are several improvements to the basic recurrent neural networks, trying to resolve vanishing gradient problems. While the standard long short-term memory unit (LSTM) solves the vanishing gradient problem by adding internal memory [20,21] and the gated recurrent unit (GRU) attempts to be a faster solution than the long short-term memory unit by using no internal memory [22], the nested long short-term memory (nested LSTM) goes in the opposite direction of the gated recurrent unit by adding additional memory to the unit. The idea here is that adding additional memory to the unit allows for more long-term memorization. We briefly describe GRU and nested LSTM approaches, since they exhibit a clear potential for muscle surrogate modeling.

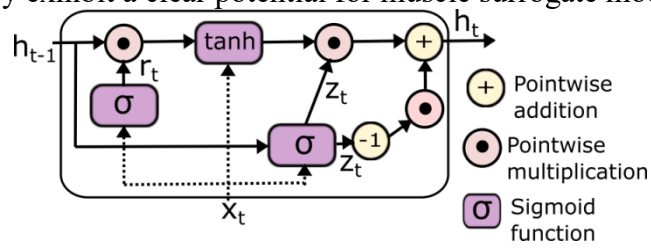

Figure 2: Gated recurrent unit (GRU) scheme

The GRU cell has only an update and a reset gate. Based on the information shown in Figure 2, the mathematical expressions of the GRU cell are as follows:

$$r_t = \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r) \quad (11)$$

$$z_t = \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z) \quad (12)$$

$$h_t = (1 - z_t)h_{t-1} + z_t tanh(W_{hh}(r_t h_{t-1}) + W_{hx}x_t + b_z) \quad (13)$$

where *t* denotes a time step, *r* denotes reset gate output, *z* denotes update gate output and *h* is the hidden state, σ is sigmoid activation function, $tanh$ is a tangential hyperbolic function, W represents weights of appropriate GRU cell parts, b_r, b_z represent reset and update biases and *x* is an input vector.

In an LSTM, the equations of updating the cell state and the gates are given by:

$$i_t = \sigma_i(x_t W_{xi} + h_{t-1}W_{hi}\, b_i) \quad (14)$$

$$f_t = \sigma_f(x_t W_{xf} + h_{t-1}W_{hf} + b_i) \quad (15)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(x_t W_{xc} + h_{t-1} W_{hc} + b_c) \quad (16)$$

$$\sigma_t = \sigma_o(x_t W_{xo} + h_{t-1} W_{ho} + b_o) \quad (17)$$

$$h_t = \sigma_i \odot tanh(c_t) \quad (18)$$

where the input (i), forget (f), and output gate (o) activation functions $\sigma_i, \sigma_f, \sigma_o$ are sigmoid activation functions; $tanh$ is a tangential hyperbolic function; c represents the cell value, h represents the hidden state of the cell, W represents weights, b denotes bias, and x is an input vector. Nested LSTM replaces the addition operation used to compute c_t in LSTM with a learned stateful function $c_t = m_t(f_t \odot c_{t-1}, i_t \odot g_t)$. The state of the function m at time t is inner memory, which is computed by another LSTM memory cell, producing a nested LSTM. The memory function can be another Nested LSTM cell, enabling deep nesting. The input and the hidden states of the memory function in a Nested LSTM become:

$$\tilde{h}_{t-1} = f_t \odot c_{t-1} \quad (19)$$

$$\tilde{x}_t = i_t \odot \sigma_c(x_t W_{xc} + h_{t-1} W_{hc} + b_c) \quad (20)$$

Besides recurrent neural networks, convolutional neural networks can also be used for modeling sequential data. For a 1-D sequence input $x \in \mathbb{R}^n$ and a filter $f: \{0, \dots, k-1\} \to \mathbb{R}$ the dilated convolution operation F on element s of the sequence is defined as:

$$F(s) = (x *_d f)(s) = \sum_{i=0}^{k-1} f(i)\, x_{s-di} \quad (21)$$

where d is the dilation factor, k is the filter size, and $s - di$ accounts for the direction of the past. Choosing larger filter sizes and increasing the dilation factor increases the receptive field of the TCN. The scheme of TCN is shown in Figure 3.

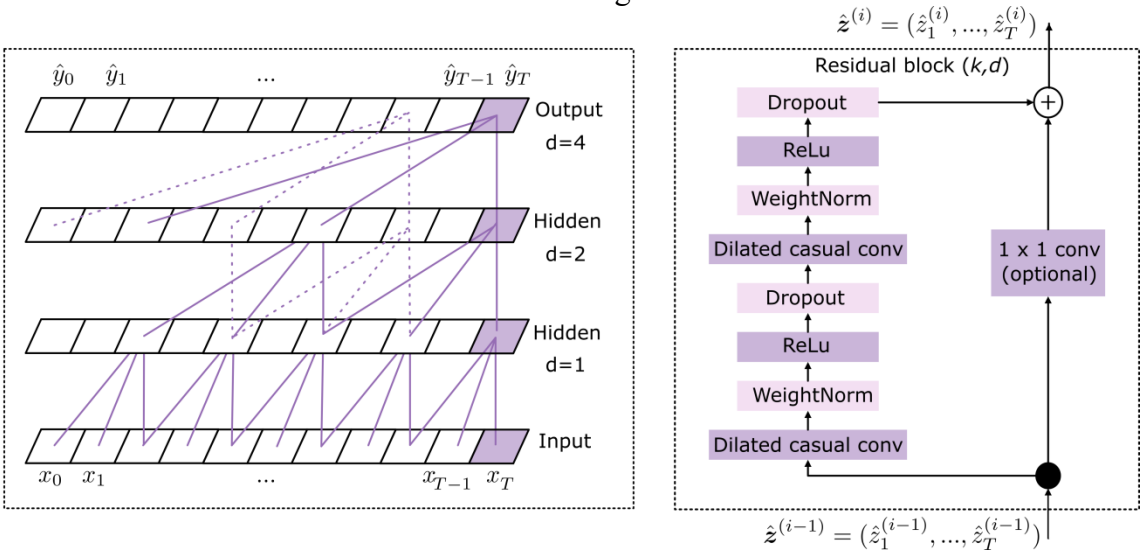

Figure 3: Temporal convolutional network scheme

The surrogate model creation procedure

In this section, we demonstrate the procedure for creating the surrogate model. We start from a numerical experiment generator, which is used to generate inputs for finite element analysis. The generated models contain a single 2-D finite element, shown next to the generator in Figure 4, with boundary conditions varying based on the type of the experiment. Using our generator, we created four types of numerical experiments: (1) isotonic contraction, (2) quick release, (3) prescribed force, and (4) prescribed displacement.

For isotonic contraction experiments, the translations were constrained at node A in x and y directions, and at point C translations are constrained in x-direction. The generator varied the activation function or calcium concentration function for each generated isotonic contraction experiment. During isotonic contraction, the muscles were activated directly by the

activation function or by calcium function, which is converted to the activation function. Under the influence of activation, muscles contracted and, once deactivated, muscles slowly returned to the initial position. During quick release experiments muscles were fully activated and translations in all directions were constrained at points A, B, and D, until a specified time step, at which translation constraints were removed at points D and B, releasing the muscle and the force was dropped down to a value less than 25% of maximum force generated. The generator varied the time at which muscle was released and the value of the prescribed force. For prescribed forces experiments, forces were prescribed at nodes D and B, while translation constraints were the same as the ones in the isotonic contraction experiment. The generator varied the prescribed force function. For prescribed displacement experiments, displacements were prescribed at nodes D and B, while translation constraints were the same as the ones in the isotonic contraction experiment. The generator varied the prescribed displacement function.

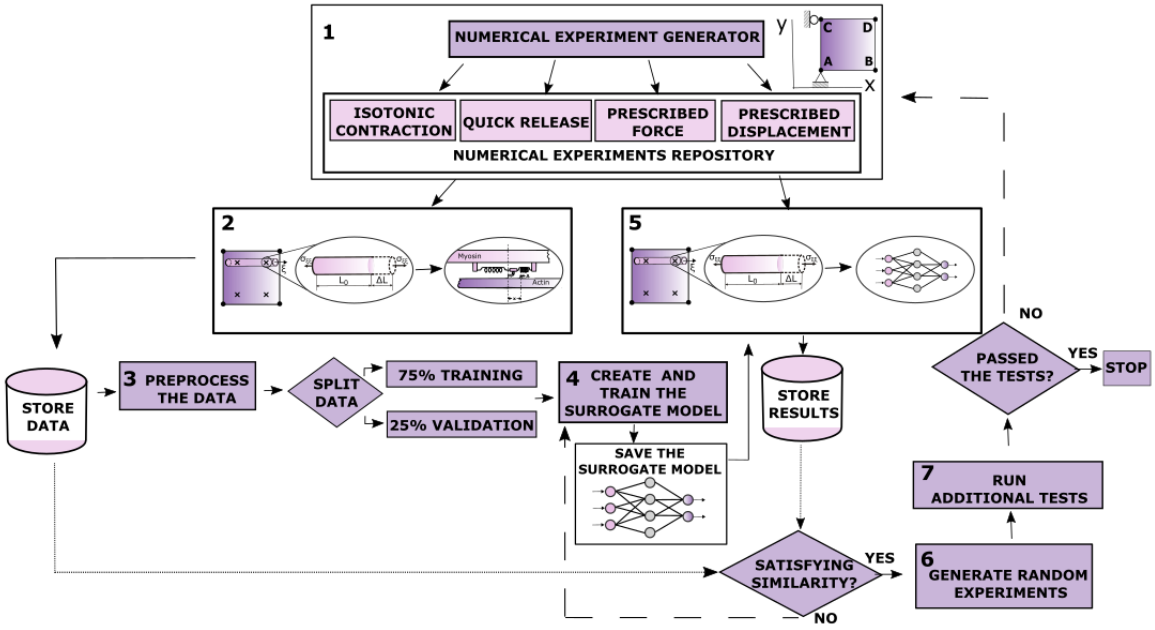

Figure 4: Surrogate creation scheme.

Once we generated the inputs, we ran finite element analysis with the original Huxley muscle model and saved the data collected from these simulations. We stored the muscle activation, stretch, stress, and stress derivatives acting as the input features for all our neural networks. We preprocessed the data so that it was converted into time series. Input tensor passing through the neural network was three-dimensional, where the dimensions were equal to (1) the number of data points at which we wanted to predict output values, (2) the number of features, and (3) time-series length. For a single data point we had an input of the following form:

$$\begin{bmatrix} {}^{t-(tl-1)\Delta t}\alpha & {}^{t-(tl-1)\Delta t}\lambda & {}^{t-tl\Delta t}\sigma_m & \frac{\partial\, {}^{t-tl\Delta t}\sigma_m}{\partial\, {}^{t-tl\Delta t}e} \\ {}^{t-(tl-2)\Delta t}\alpha & {}^{t-(tl-2)\Delta t}\lambda & {}^{t-(tl-1)\Delta t}\sigma_m & \frac{\partial\, {}^{t-(tl-1)\Delta t}\sigma_m}{\partial\, {}^{t-(tl-1)\Delta t}e} \\ \vdots & \vdots & \vdots & \vdots \\ {}^{t}\alpha & {}^{t}\lambda & {}^{t-\Delta t}\sigma_m & \frac{\partial\, {}^{t-\Delta t}\sigma_m}{\partial\, {}^{t-\Delta t}e} \\ {}^{t+\Delta t}\alpha & {}^{t+\Delta t}\lambda & {}^{t}\sigma_m & \frac{\partial\, {}^{t}\sigma_m}{\partial\, {}^{t}e} \end{bmatrix}_{(tl+1)\times 4} \tag{22}$$

where α is muscle activation, λ is stretch, σ_m is stress, $\frac{\partial \sigma_m}{\partial e}$ is stress derivative, (tl+1) is the total length of the time series and t+Δt is the end of the current time step, within finite element simulation. Based on this input tensor we predicted stress increment ${}^{t+\Delta t}\hat{\Delta\sigma}_m$ and increment of stress derivative $\partial\, {}^{t+\Delta t}\hat{\Delta\sigma}_m / \partial\, {}^{t+\Delta t}e$. With the "^" symbol, we denoted predicted values.

To train the model, we scaled input values to the range between 0 and 1, thus enabling equal influence of all input features to the prediction of the neural network. We also scaled the output stress and instantaneous stiffness to the range between 0 and 1. To get the increments, we subtracted corresponding scaled values, and we further scaled these increments, producing increments that are more separated from each other and enabling the network to achieve higher precision predictions. Once the network was trained we used simple formulas:

$$ {}^{t+\Delta t}\hat{\sigma}_m = {}^{t}\sigma_m + {}^{t+\Delta t}\hat{\Delta\sigma}_m \tag{23}$$

$$\partial\, {}^{t+\Delta t}\hat{\sigma}_m / \partial\, {}^{t+\Delta t}e = \partial\, {}^{t}\sigma_m / \partial\, {}^{t}e + \partial\, {}^{t+\Delta t}\hat{\Delta\sigma}_m / \partial\, {}^{t+\Delta t}e \tag{24}$$

to calculate the stress and instantaneous stiffness.

After preprocessing the data, creation, and training of the model, we run identical numerical experiments, which we previously ran with the original Huxley model, with our surrogate model and stored the results. If a satisfying similarity between the original and surrogate model is achieved, we proceed to generate new random experiments. Contrary, in case our model fails to achieve satisfying similarity, we go back to the creation and training of the model. Once we generate new random experiments, we run additional tests and if we achieve satisfying similarity with random numerical experiments, we stop the process. On the other hand, if our surrogate model fails to pass the tests, we start the process from the beginning and enriched the data used for the creation of the surrogate model. For the creation of the surrogate model, we generated a total of 160 numerical experiments for training and validation of the model: 45 isotonic contraction experiments, 20 quick release experiments, 40 prescribed force experiments, and 55 prescribed displacement experiments. Every fourth experiment was used to validate the model, while the rest were used for model training. We generated 8 additional experiments, 2 of each kind, to test the surrogate model. The optimal hyperparameters of the neural networks were obtained by trial-and-error, following the process shown in Figure 4. The hyperparameters for each type of network are summarized in Table 1 in the results section. The main performance criterion was the correlation coefficient between stresses produced by the original Huxley model on one side, and the surrogate model during finite element analysis, on the other side. Secondary performance criteria include achieved speed-up, together with memory consumption.

Integration of the surrogate model into the finite element analysis framework

The procedure of integrating the surrogate model into the finite element analysis framework consists of initialization, setting the input values for the neural network, getting the values, and updating the input at the end of each finite element time step. The full procedure is depicted in Figure 5.

During initialization (*surro_init*), we initialize the input tensor of the neural network, load the neural network architecture and weights, and load time series and scaling/descaling parameters from the configuration file. The input tensor size depends on the time series length, the number of input features, and the number of integration points in the model. The initial stretch values were set to 1.0, and the other values of the input tensor to zero.

During each time step of the numerical simulation and each iteration of the simulation, we provide the activation and stretch at each integration point (ip) to the surro_set_values function. These values are scaled and stored in the appropriate location in the input tensor as shown in Figure 5. Once all the values are set, by calling the surro_predict function the neural network predicts the stress increments and instantaneous stiffness increments. To utilize these values in finite element analysis, we employ the procedure surro_get_values, which descales the predicted increments and calculates stress (stress) and instantaneous stiffness (dstress), and returns the values for a specified integration point (ip). At the end of the finite element step, the data in the input tensor shifts back in time, leaving the last row for the subsequent FE time step.

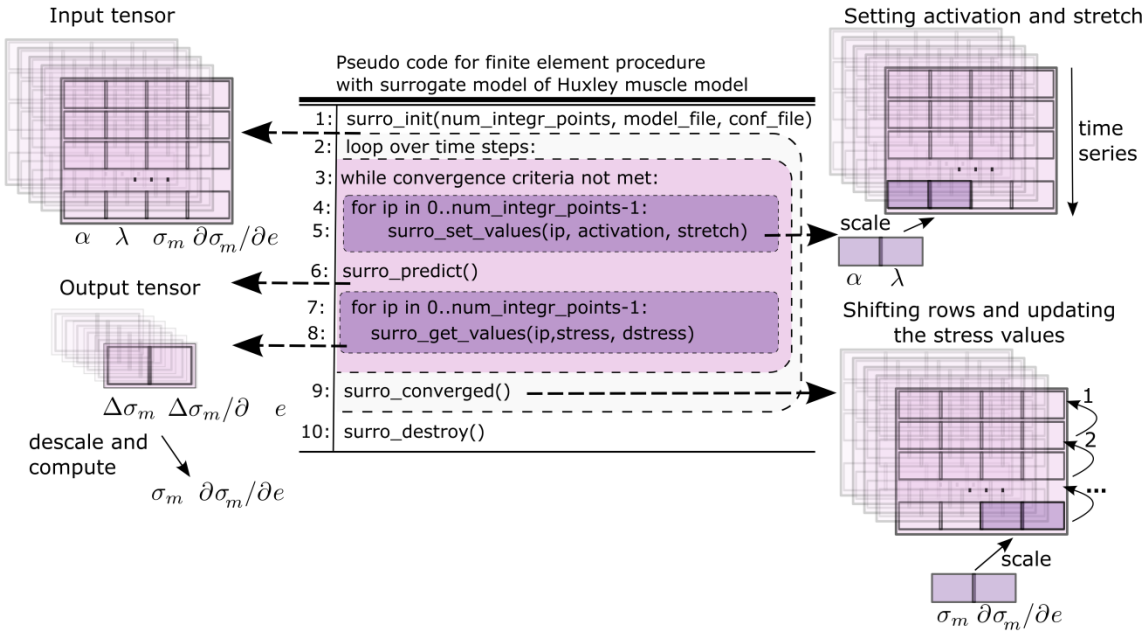

Figure 5: Surrogate integration into finite element software scheme.

Results and Discussion

In this section, we specify the nature of the collected data, present the results obtained by the proposed surrogate model on various types of numerical experiments, and analyzeachieved speed-up. Moreover, we demonstrate the left ventricle multi-scale model, in which the proposed surrogate model is employed as a drop-in replacement for the Huxley model, acting as a muscle material model.

The collected data and training mechanisms

To demonstrate the specifics of the collected data and explain why we predict increments instead of stress values directly, it is best to show the stress autocorrelation. In Figure 6, the stress at time t+Δt and previous stress values at time *t* are shown. We can separate three distinct cases in Figure 6: (1) cases with only isotonic contractions, (2) cases with isotonic contractions and quick release experiments, and (3) cases with all numerical experiments used during training. It can be noted that the correlation between stresses ${}^{t}\sigma_m$ and ${}^{t+\Delta t}\sigma_m$ is generally strong, but once the quick release experiments are added into the training set, the obvious outliers appear (circled in Figure 6).

In general, stress increments or decrements by a small value. The large difference between the current and previous muscle stress occurs at the quick release experiments at the moment where the muscle is suddenly released and the force is dropped. This is where the outliers come from. If we used neural networks to directly predict stress $^{t+\Delta t}\sigma_m$ the neural network would use this strong general correlation, and with isotonic contraction, this approach could work. However, once the quick release experiments are added to the training set, this approach fails. Adding the quick release experiments makes the training harder since the range of values is a lot larger. A larger range of stress values makes it difficult to predict small and large stress values precisely enough for the surrogate model to work inside a finite element simulation. Since the number of outliers is not high compared to the full data set size, even if, during training, the predicted values for these data points are wrong, the total loss can be small. If we tested the network only on data, the prediction for the point of release would be wrong, but other points before and after the release moment would be similar to the original model leading to overall satisfactory performance. On the other hand, if we run the finite element simulation with the surrogate model, quick release experiments would fail, from the moment of muscle release to the end of the simulation.

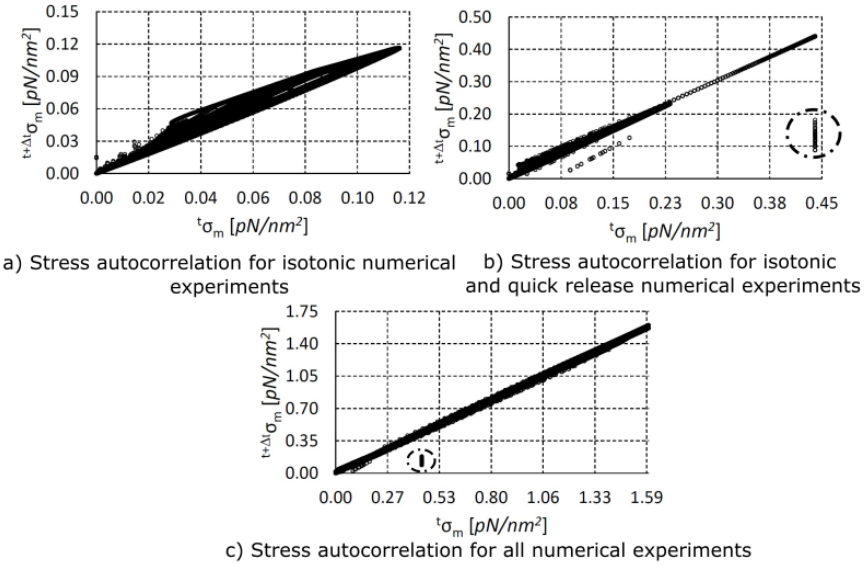

Figure 6: Stress autocorrelation.

To resolve this issue, we propose predicting the increments of stress and stress derivatives, instead of values directly. By predicting the increments, we augment the importance of time moments at which the stress changes significantly. We also do not allow the neural network from abusing a strong correlation between input and output stress, enabling it to truly grasp the change of stress and stress derivatives (instantaneous stiffness) with respect to the input features (activation, stretch, stress, and stress derivative). To enable the network to predict small and large increments precisely enough, we introduced a scaling factor for stress increments as a hyperparameter.

A few mechanisms were key to successfully training the neural networks: (1) predicting stress and stress derivative increments, (2) scaling the prediction values, (3) Huber loss function, and (4) gradient normalizing. The early stopping mechanism prevents overfitting of the neural networks. Gradient clipping is a technique to prevent exploding gradients in very deep networks, usually in recurrent neural networks. The Huber loss function clips the gradient values since the derivative of the Huber loss function for errors larger than its delta parameter is constant. To normalize gradients, we set the gradient threshold. The gradient norms that exceed this threshold are scaled down to match the norm, making the training

more stable and less susceptible to exploding gradients. Normalizing gradients also helps the optimization algorithm to behave reasonably even if the loss landscape of the model is irregular. Without normalizing and clipping the gradients, the parameters can take a huge descent step and leave the region with a potentially optimal solution. With clipping, the descent step size is restricted and the parameter values stay in the good parameter space region.

Comparison of the trained neural networks

During the creation of the surrogate model, we constructed a lot of different neural networks. In this subsection, we show a few representatives, exhibiting the greatest potential for muscle surrogate modeling.

Table 1: The hyperparameters of the selected neural networks

Type of neural network	*Number of weights*	*Hidden layers*
TCN	928,706	11 convolutional layers ×192 filters dilations = [1,2,4,8,16], kernel_size=4
Nested LSTM	992,002	1 nested layer, depth = 8, 128 neuros per depth
GRU	992,770	[1st,4th and 5th] GRU layer ×128 neurons, [2nd, 3rd] GRU layer ×256 neurons
Nested LSTM-TCN	991,874	1 nested layer, depth 6, 128 neurons per depth, 7 convolutional layers ×128 filters, dilations = [1,2,4], kernel_size=4
GRU-TCN	974,978	[1st, 4th] GRU layer ×64 neurons, [2nd, 3rd] GRU layer ×256 neurons, 7 convolutional layers ×128 filters, dilations = [1,2,4], kernel_size=4

Common hyperparameters: Rectified Adam teaches all networks with initial learning rate = 10^{-3}, $\boldsymbol{\beta_1 = 0.99}$, $\boldsymbol{\beta_2 = 0.9999}$, clip norm = 10^{-4}. Huber loss function was minimized with $\boldsymbol{\delta = 58 \times 10^{-6}}$. The total number of epochs was set to a maximum of 50000 with the batch size 16384, together with Early stopping having the patience of 500 epochs.

In Table 1, we show a summary of the hyperparameters for five different types of neural networks: (1) TCN, (2) Nested LSTM, (3) GRU, (4) Nested LSTM-TCN, and (5) GRU-TCN. TCN, GRU, and LSTM neural networks were chosen for comparison since they are typically used in time series analysis. Their hyperparameters were mainly optimized by trial-and-error, since it was not clear in advance which of these neural networks will perform the best within in the multi-scale finite element simulation. Rectified Adam turned out to be the best optimizer for our problem, so we used it to train all the networks. The rest common hyperparameters for all the networks are shown in Table 1. The total number of weights is similar in all constructed neural networks, so we can compare the computational performance of all these networks fairly enough. The mean correlation coefficients between true and predicted stress values for 5 types of neural networks are shown in Table 2. The table presents the correlation coefficients acquired on the train, validation, and test data, together with the correlation coefficients acquired on the train, validation, and test simulations. The quality of the proposed surrogate model is apparent in the high average correlation between real and predicted values and low deviation, showing that the surrogate performs similarly in all conducted numerical experiments. However, the obtained correlation coefficients are always lower in numerical simulations. For predictions made on data, we always have the correct input data, while for predictions made during simulations, inputs differ from the original under the influence of the cumulative effect of predicted stress and instantaneous stiffness. In general, high correlation acquired on data will produce high correlations with

simulations, if the generalization is good enough. The generalization is best shown in test experiments. If the network performs well on the test set, we can conclude that the generalization is sufficiently good. The low standard deviation between correlation coefficients acquired from the different numerical experiments also indicates a good generalization. The poorest results on the test set were achieved by TCN, indicating that convolutional neural networks generalize worse than recurrent neural networks in time series analysis tasks. The highest correlation coefficients on train, validation, and test data were achieved by GRU neural network. Note that the GRU neural network has the update and reset mechanisms that are mostly similar to the ones used during finite element iterations, which could explain their good performance in multi-scale simulations. With GRU approach, we also obtained the most accurate behavior in the numerical simulation. Moreover, GRU network exhibits the lowest deviations of correlation coefficients in most cases. It's interesting to see that combining the Nested LSTM network and TCN network produces better results than Nested LSTM and TCN individually. The Nested LSTM performed better on the test set than TCN. On the other hand, TCN performed better on the training set. Combining these networks helped balance out the individual flaws of the Nested LSTM and TCN. However, such an effect wasn't achieved when combining GRU and TCN, because stand-alone GRU performed better. On the other hand, combined GRU-TCN performed better than combined Nested LSTM-TCN. In total, the GRU network showed the greatest potential for muscle surrogate modeling. It can be seen that the correlation coefficients drop in test experiments for all the networks, in Tables 2 and 3, indicating that the generalization of our networks could be further improved. The drop rate of the correlation coefficients in our test experiments is the lowest considering GRU, confirming its generalization potential. The mean correlation coefficients between real and predicted instantaneous stiffness values for all 5 types of neural networks are shown in Table 3. They are very similar to the ones shown in Table 2.

Table 2: Mean and standard deviation of correlation coefficients between true and predicted stress values

Type of neural network	*Mean correlation coefficient between true and predicted stress*					
	on data			*within numerical simulation*		
	train	*validation*	*test*	*train*	*validation*	*test*
TCN	$0.\bar{9}^{5}45$	$0.\bar{9}^{4}89$	$0.\bar{9}^{3}87$	0.961	0.967	0.634
Nested LSTM	$0.\bar{9}^{3}79$	$0.\bar{9}^{3}81$	$0.\bar{9}^{4}30$	0.543	0.632	0.216
GRU	$\mathbf{0.\bar{9}^{6}72}$	$\mathbf{0.\bar{9}^{6}40}$	$\mathbf{0.\bar{9}^{5}18}$	$\mathbf{0.\bar{9}^{3}77}$	$\mathbf{0.\bar{9}^{3}65}$	**0.989**
Nested LSTM-TCN	$0.\bar{9}^{5}51$	$0.\bar{9}^{5}56$	$0.\bar{9}^{4}71$	0.981	0.981	0.812
GRU-TCN	$\mathbf{0.\bar{9}^{6}72}$	$0.\bar{9}^{5}84$	$0.\bar{9}^{4}59$	$0.\bar{9}^{3}24$	0.998	0.965

***Note**: The notation $\bar{9}^{x}$ means the 9 is repeated x times.*

Type of neural network	*The standard deviation of the correlation coefficient between true and predicted stress*					
	on data			*within numerical simulation*		
	train	*validation*	*test*	*train*	*validation*	*test*
TCN	3.31×10^{-5}	3.78×10^{-5}	2.95×10^{-4}	1.20×10^{-1}	9.29×10^{-2}	5.19×10^{-1}
Nested LSTM	5.18×10^{-4}	4.08×10^{-4}	3.89×10^{-5}	4.03×10^{-1}	3.47×10^{-1}	4.04×10^{-1}
GRU	9.60×10^{-7}	$\mathbf{1.38\times10^{-6}}$	$\mathbf{1.80\times10^{-5}}$	$\mathbf{1.03\times10^{-3}}$	$\mathbf{6.83\times10^{-4}}$	$\mathbf{2.35\times10^{-2}}$
Nested LSTM-TCN	3.03×10^{-5}	1.25×10^{-5}	5.01×10^{-5}	4.97×10^{-2}	5.28×10^{-2}	3.57×10^{-1}
GRU-TCN	$\mathbf{8.70\times10^{-7}}$	4.05×10^{-6}	9.09×10^{-5}	3.04×10^{-3}	4.52×10^{-3}	6.92×10^{-2}

Table 3: Mean and standard deviation of correlation coefficients between true and predicted instantaneous stiffness values

Type of neural network	*Mean correlation coefficient between true and predicted instantaneous stiffness*					
	on data			*within numerical simulation*		
	train	*validation*	*test*	*train*	*validation*	*test*
TCN	$0.\bar{9}^{7}50$	$0.\bar{9}^{6}20$	$0.\bar{9}^{6}73$	0.976	0.976	0.691
Nested LSTM	$0.\bar{9}^{5}78$	$0.\bar{9}^{5}51$	$0.\bar{9}^{5}87$	0.762	0.823	0.533
GRU	$\mathbf{0.\bar{9}^{7}80}$	$\mathbf{0.\bar{9}^{7}80}$	$\mathbf{0.\bar{9}^{6}84}$	$\mathbf{0.\bar{9}^{3}84}$	$\mathbf{0.\bar{9}^{3}70}$	$\mathbf{0.\bar{9}^{3}75}$
Nested LSTM-TCN	$0.\bar{9}^{7}40$	$0.\bar{9}^{6}80$	$0.\bar{9}^{4}70$	0.987	0.984	0.845
GRU-TCN	$\mathbf{0.\bar{9}^{7}80}$	$0.\bar{9}^{6}60$	$0.\bar{9}^{6}25$	$0.\bar{9}^{3}46$	0.998	0.986

***Note**: The notation $\bar{9}^{x}$ means the 9 is repeated x times.*

Type of neural network	*The standard deviation of the correlation coefficient between true and predicted instantaneous stiffness*					
	on data			*within numerical simulation*		
	train	*validation*	*test*	*train*	*validation*	*test*
TCN	1.60×10^{-7}	4.20×10^{-6}	3.90×10^{-7}	1.10×10^{-1}	7.00×10^{-2}	4.08×10^{-1}
Nested LSTM	6.35×10^{-6}	2.38×10^{-5}	1.10×10^{-6}	3.57×10^{-1}	2.41×10^{-1}	4.63×10^{-1}
GRU	6.00×10^{-8}	$\mathbf{3.10 \times 10^{-7}}$	$\mathbf{2.40 \times 10^{-7}}$	$\mathbf{4.61 \times 10^{-4}}$	$\mathbf{6.70 \times 10^{-4}}$	$\mathbf{4.64 \times 10^{-4}}$
Nested LSTM-TCN	1.60×10^{-7}	7.70×10^{-7}	4.80×10^{-7}	3.60×10^{-2}	3.91×10^{-2}	1.75×10^{-1}
GRU-TCN	$\mathbf{4.00 \times 10^{-8}}$	1.47×10^{-6}	1.43×10^{-6}	2.84×10^{-3}	6.83×10^{-3}	2.62×10^{-2}

Stress-time diagrams with GRU network and original Huxley model

In this subsection, we compare the results obtained using the proposed surrogate model based on GRU neural network, and the original model in different types of numerical experiments. Figures 7-9 show stress-time diagrams obtained from the numerical simulation carried out using the original Huxley model and the surrogate Huxley model. For the sake of simplicity, we don't show the instantaneous stiffness, since the results are analogous. Figure 7 shows some cases of numerical experiments used to train the neural network. Figure 7a shows an example of isotonic contraction with calcium concentration function prescribed and corresponding activation function. Figure 7b depicts an example of quick release experiment, where the muscle is released and the force is dropped. Figure 7c shows an example of prescribed force experiments, along with the prescribed force function. Figure 7d depicts stress acquired from a prescribed displacement experiment. Figure 8 shows some of the cases used for validation during training of the neural network, while Figure 9 demonstrates some of the cases used to test the surrogate model. In all these experiments, the surrogate model performed very similarly to the original Huxley model.

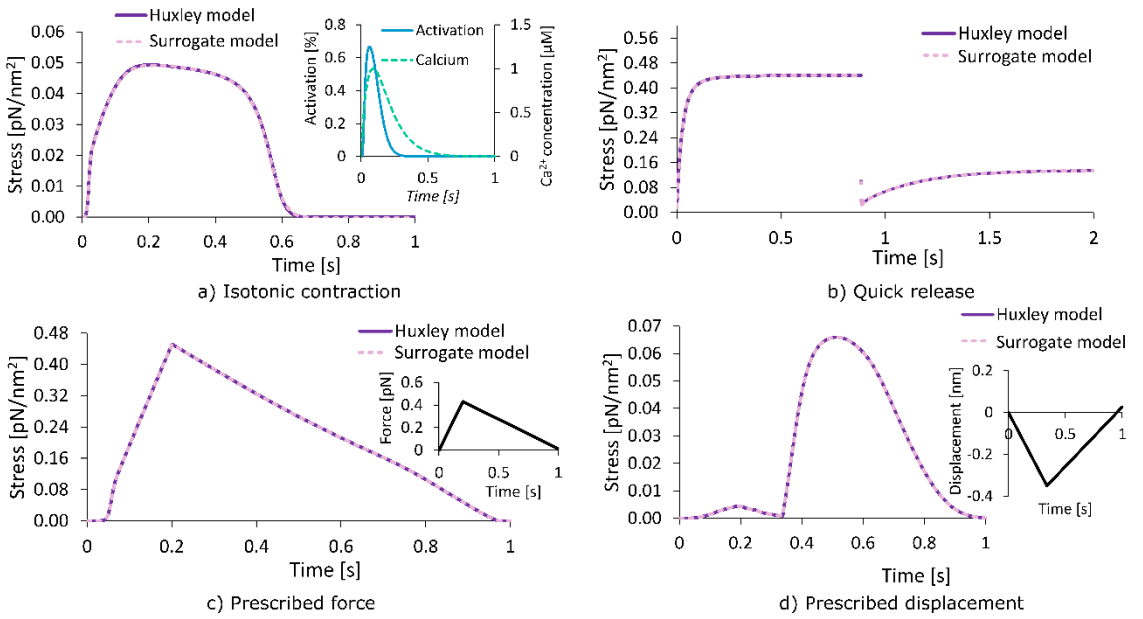

Figure 7: Stress obtained from the numerical experiments used to train the neural network. Isotonic contraction a), quick release b), prescribed force c), prescribed displacement d).

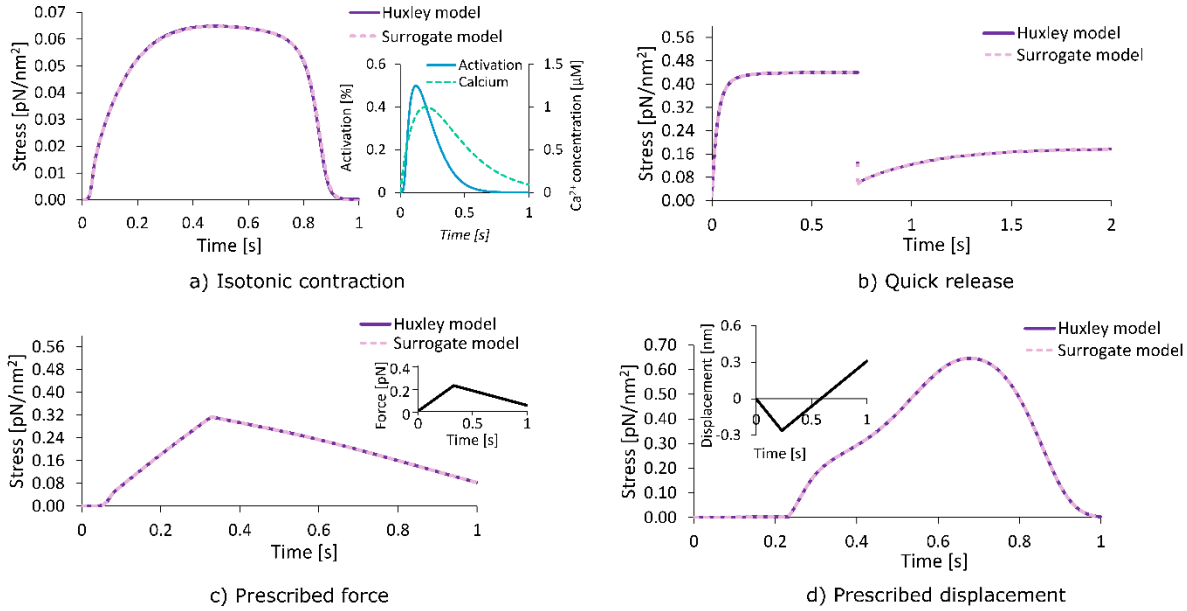

Figure 8: Stresses obtained from the numerical experiments used to validate the neural network. Isotonic contraction a), quick release b), prescribed force c), prescribed displacement d).

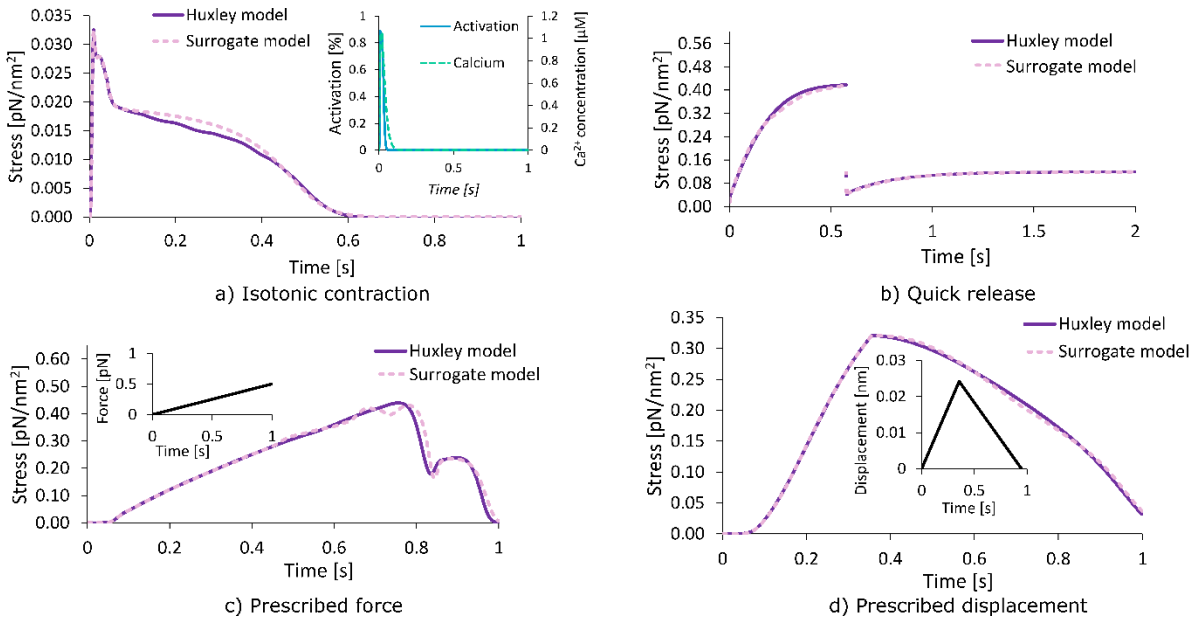

Figure 9: Stresses obtained in numerical experiments used to test the neural network. Isotonic contraction a), quick release b), prescribed force c), prescribed displacement d).

Speeding up the numerical simulations by using surrogate modeling

To recall, the main goal of the proposed surrogate is to create a model that is computationally more efficient than the original. In this section, we analyze obtained speed-up. The speed-ups achieved by the surrogate modeling are depicted in Figure 10. We compare the execution times of the sequential and parallel versions of the multiscale finite element simulation using the original Huxley model with the simulation carried out using the surrogate model. The parallelization is performed at the level of the integration points. Since all numerical experiments contain 4 integration points, we spawn 4 MPI processes. The surrogate model is faster than the original Huxley muscle model by an order of magnitude. More precisely, compared to the sequential version of the original Huxley muscle model, the surrogate model was around 50 times faster for the quick release, prescribed force, and prescribed displacements experiments, and around 25 times faster for the isotonic contraction experiments. This makes the surrogate model more usable in models with a larger number of finite elements. The memory consumption of the surrogate model depends on the size of the neural network and also the size of input and output tensors, while memory consumption of the original Huxley model mostly depends on the sizes of arrays for storing positions and probabilities of the cross-bridge attachments at the micro-level. The input tensors of the surrogate model are fairly small, since the length of the time series is 11, and the number of features is 4. In contrast, arrays in the original Huxley model implementation are large, requiring thousands of double precision numbers. In our use case with 4000 integration points, the original Huxley model consumed 23960 MB, while the simulation which incorporated the proposed surrogate required only 812 MB.

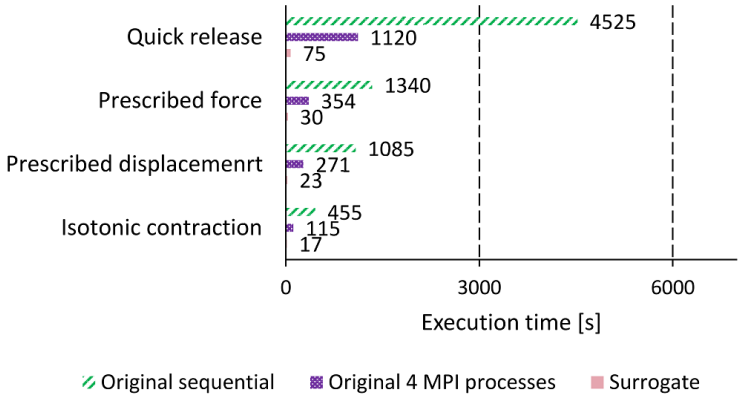

Figure 10: Execution time of the surrogate and the original model.

Cardiac cycle with the left ventricle model

To demonstrate the surrogate's potential on a larger model, we used the left ventricle model with parameterized geometry. The left ventricle model, consisting of a fluid and a solid wall is shown in Figure 11. We modeled the half of the left ventricle with a mitral and aortic branch at the top of the model. These branches are connected to the base by a connective geometrical component (Figure 11a). The fluid model geometry is generated by prescribing the lengths and diameters of all of the components. We also introduced additional parameters to control the mesh density. The solid domain is appended to the fluid domain (Figure 11b). To generate the solid domain, we prescribed wall thickness and the number of solid wall layers. The arrows in Figure 11b represent the helical muscle fibers with angles varying from 60° at the endocardium (yellow arrows) to -60° at the epicardium (red arrows). Fibers in the middle of the wall thickness are shown in pink color. A linear change of the fiber directions is assumed over the wall thickness. The solid component of the model contains around 4000 integration points.

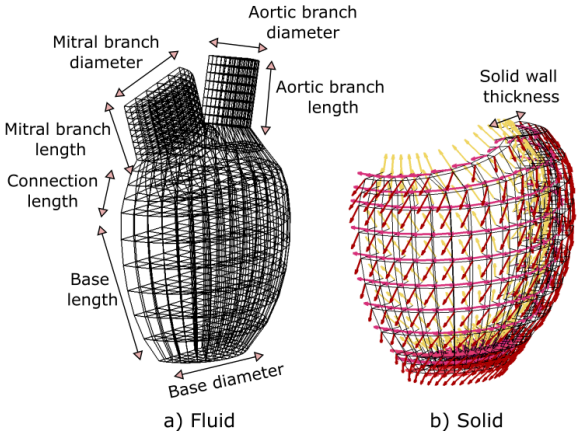

Figure 11: Left ventricle model a) fluid domain b) solid domain with muscle fibers

The transient model takes into account the entire heart cycle, starting with the initial geometry, shown in Figure 11, which corresponds to the very start of diastole. During diastole, a constant fluid velocity of 100 mm/s is prescribed at the entrance to the mitral valve, while the aortic valve is closed. At the end of the diastolic period, the velocity at the mitral valve reduces to zero, and both valves are closed. This time interval when both valves are closed corresponds to the isovolumetric compression, and the muscles are activated

during this period. The aortic valve opens at the very start of the systole and the muscles are slowly being deactivated. Under the influence of forces generated by muscles, the blood flows out of the ventricle, through the aortic valve. We used the Holzapfel material model to calculate the passive stresses generated within left ventricle tissue [23-29]. To determine the active stress, we employ the surrogate model based on GRU neural network, which showed the best similarity to the original Huxley muscle model (Figure 9).

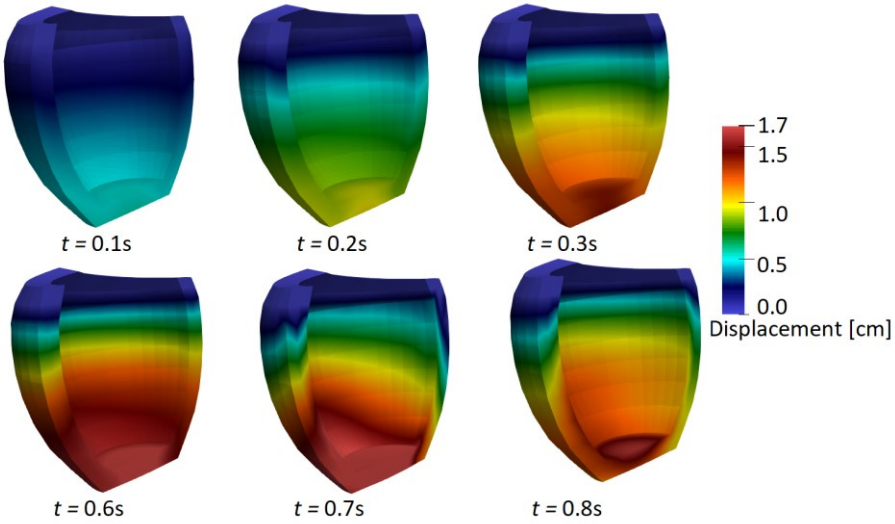

Figure 12: Left ventricle displacement field at start of the diastole (t=0.1s, t=0.2s, t=0.3s), and at the start and the middle of the systole (t=0.6s, t=0.7s, t=0.8s)

In Figure 12 we showed displacement fields, in the solid wall of the model, at the start of the diastole (t=0.1s, t=0.2s, t=0.3s), and the start and middle of the systole (t=0.6s, t=0.7s, t=0.8s). Figure 13 shows the velocity field in fluid, at the start and middle of the diastole and the middle of the systole. It is visible that the muscles are generating the force at the start of the systole, contracting the ventricle, and towards the end of the cardiac cycle, the muscles are slowly being deactivated returning the ventricle to the initial configuration. These results confirm that the proposed surrogate model can be used in larger models.

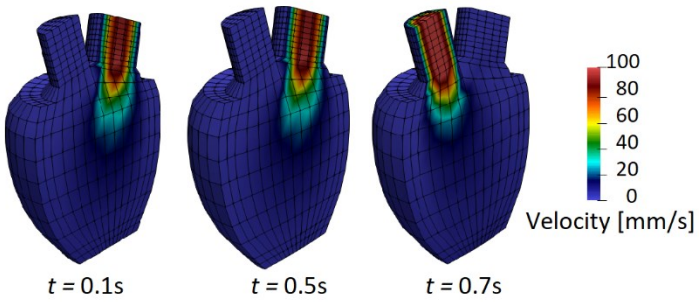

Figure 13: Left ventricle fluid velocity field at the start and the middle of the diastole, and the middle of the systole

Conclusions

In this paper, we presented the surrogate model of the Huxley muscle model, which is a biophysical muscle model based on the underlying physiology of the muscles. The surrogate

model was created via a data-driven approach, in which the data is collected from the multi-scale finite element simulation with the original Huxley muscle model built in. Since muscles are a history-dependent type of material, i.e. their behavior is influenced by previous states of the model, we converted the collected data into time series and trained recurrent and convolutional neural networks. During the multiscale finite element simulation, we provide the material model with stretch and muscle activation, and the material model returns the stress and instantaneous stiffness based on these inputs and internal state. To produce stress and instantaneous stiffness for the current time step, the input of the neural network consists of the activation at current and previous time steps, stretch at current and previous time steps, stress at previous time steps, and instantaneous stiffness at previous time steps.

The numerical experiment generator generates different types of experiments such as (1) isotonic contraction, (2) quick release, (3) prescribed forces, and (4) prescribed displacement experiment. The procedure to integrate the surrogate model into the finite element simulation is also presented in detail.

Training the neural network to achieve similarity to the original model within finite element simulation was quite challenging. The precision of the predicted stress had to be high for the finite element simulation to work properly. The generalization of the neural network also had to be sufficiently good. If the generalization was insufficient, the network would fail in the numerical experiments that it wasn't trained on, and it could even fail in the numerical experiment, which was used during training, since the neural network itself would produce the stress a bit different from the one used during training.

We trained different types of recurrent and convolutional neural networks such as (1) TCN, (2) Nested LSTM, (3) GRU, (4) Nested LSTM-TCN, and (5) GRU-TCN. To train the neural networks well enough, we employed several machine-learning mechanisms, such as gradient clipping and normalizing. One of the key mechanisms that resolved the issues of generalization and precision, was preventing the neural network to abuse stress autocorrelation, by predicting the stress and instantaneous stiffness increment instead of stress and instantaneous stiffness values directly. We also introduced specific scaling factors for stress and instantaneous stiffness increments during training, that enabled the network to make higher precision predictions. The best neural network for this specific problem turned out to be the GRU neural network. We achieved high precision and a sufficient generalization with this network in four different types of numerical experiments. We also achieved a high speed-up which enabled us to use the model in significantly larger, realistic, and computationally intensive simulations. We demonstrate the potential of the proposed surrogate model to be used in modeling the behavior of the left ventricle, which would be much harder to accomplish using the original Huxley model due to the enormous computational requirements.

The proposed surrogate model is created for a single set of parameters of the Huxley muscle model, so our future research could also include creating different surrogate models for a different set of Huxley model parameters that are related to the muscle protein mutations and disorders. Future research will include the application of newly discovered physics-informed neural networks [30] for solving Huxley muscle partial differential equation. The proposed surrogate model is precise enough, but generalization could be even better with a physics-informed neural network, since they are specialized for solving partial differential equations.

Data Availability

The data used to support the findings of this study have been deposited in the GitHub repository https://github.com/BogdanM1/surro-muscle/data/. The code used to preprocess the

data, create surrogate models, train and integrate the model into the finite element solver is also available in the repository https://github.com/BogdanM1/surro-muscle.

Declaration of competing interest

The authors declare that there is no competing interest regarding the publication of this paper.

Acknowledgments

Research supported by the SILICOFCM project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777204. This article reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information the article contains. The research was also funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia, contract numbers [451-03-68/2022-14/200107 (Faculty of Engineering, University of Kragujevac), 451-03-68/2022-14/200122 (Faculty of Science, University of Kragujevac) and 451-03-68/2022-14/200378 (Institute for Information Technologies Kragujevac, University of Kragujevac)]. We thank our colleague Lazar Vasović for his help and support. We also thank Neda Vidanović Miletić for English language editing.

References

[1] B. Stojanovic, M. Svicevic, A. Kaplarevic-Malisic, R. J. Gilbert, and S. M. Mijailovich, "Multi-scale striated muscle contraction model linking sarcomere length-dependent cross-bridge kinetics to macroscopic deformation," Journal of Computational Science, vol. 39. Elsevier BV, p. 101062, Jan. 2020. doi: 10.1016/j.jocs.2019.101062.

[2] B. S. Stojanovic, M. R. Svicevic, A. M. Kaplarevic-Malisic et al., "Coupling finite element and huxley models in multiscale muscle modeling," 2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE). IEEE, Nov. 2015. doi: 10.1109/bibe.2015.7367674.

[3] M. Ivanović, B. Stojanović, A. Kaplarević-Mališić, R. Gilbert, and S. Mijailovich, "Distributed multi-scale muscle simulation in a hybrid MPI–CUDA computational environment," SIMULATION, vol. 92, no. 1. SAGE Publications, pp. 19–31, Dec. 11, 2015. doi: 10.1177/0037549715620299.

[4] M. Ivanović, A. Kaplarević-Mališić, B. Stojanović, M. Svičević, and S. M. Mijailovich, "Machine learned domain decomposition scheme applied to parallel multi-scale muscle simulation," The International Journal of High Performance Computing Applications, vol. 33, no. 5. SAGE Publications, pp. 885–896, Mar. 12, 2019. doi: 10.1177/1094342019833151.

[5] S. Aydin, "Deep Learning Classification of Neuro-Emotional Phase Domain Complexity Levels Induced by Affective Video Film Clips," IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 6. Institute of Electrical and Electronics Engineers (IEEE), pp. 1695–1702, Jun. 2020. doi: 10.1109/jbhi.2019.2959843.

[6] B. Kılıç and S. Aydın, "Classification of Contrasting Discrete Emotional States Indicated by EEG Based Graph Theoretical Network Measures," Neuroinformatics. Springer Science and Business Media LLC, Mar. 14, 2022. doi: 10.1007/s12021-022-09579-2.

[7] S. Yan, X. Zou, M. Ilkhani, and A. Jones, "An efficient multiscale surrogate modelling framework for composite materials considering progressive damage based on artificial neural networks," Composites Part B: Engineering, vol. 194. Elsevier BV, p. 108014, Aug. 2020. doi:10.1016/j.compositesb.2020.108014.

[8] F. Ghavamian and A. Simone, “Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network,” Computer Methods in Applied Mechanics and Engineering, vol. 357. Elsevier BV, p. 112594, Dec. 2019. doi: 10.1016/j.cma.2019.112594.

[9] F. Feyel, A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua, Comput. Methods Appl. Mech. Engrg. 28–30 (2003) 3233–3244.

[10] M. Kojic and K.-J. Bathe, Inelastic analysis of solids and structures. Springer, 2005.

[11] K. J. Bathe, “Finite element procedures. Englewood Cliffs, NJ: Prentice-Hall; 1996.

[12] B. Stojanovic, M. Kojic, M. Rosic, C. P. Tsui, and C. Y. Tang, “An extension of Hill’s three-component model to include different fibre types in finite element modelling of muscle,” Int. J. Numer. Methods Eng., vol. 71, no. 7, pp. 801–817, 2007.

[13] S. M. Mijailovich, B. Stojanovic, M. Kojic, A. Liang, V. J. Wedeen, and R. J. Gilbert, “Derivation of a finite-element model of lingual deformation during swallowing from the mechanics of mesoscale myofiber tracts obtained by MRI,” Journal of Applied Physiology, vol. 109, no. 5. American Physiological Society, pp. 1500–1514, Nov. 2010. doi: 10.1152/japplphysiol.00493.2010.

[14] A. F. Huxley, “Muscle structure and theories of contraction,” Prog. Biophys. Biophys. Chem, vol. 7, pp. 255–318, 1957.

[15] A. M. Gordon, A. F. Huxley, and F. J. Julian, “The variation in isometric tension with sarcomere length in vertebrate muscle fibres,” J. Physiol., vol. 184, no. 1, pp. 170–192, 1966.

[16] S. M. Mijailovich, J. J. Fredberg, and J. P. Butler, “On the theory of muscle contraction: filament extensibility and the development of isometric force and stiffness.,” Biophys. J., vol. 71, no. 3, pp. 1475– 84, Sep. 1996.

[17] P. J. Hunter, A. D. McCulloch, and H. E. D. J. ter Keurs, “Modelling the mechanical properties of cardiac muscle,” Progress in Biophysics and Molecular Biology, vol. 69, no. 2–3. Elsevier BV, pp. 289–331, Mar. 1998. doi: 10.1016/s0079-6107(98)00013-3.

[18] L.C. Jain, L.R. Medsker, Recurrent Neural Networks: Design and Applications, first ed., CRC Press, Inc., Boca Raton, FL, USA, 1999.

[19] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks.” arXiv, 2012. doi: 10.48550/ARXIV.1211.5063.

[20] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” Neural Computation, vol. 31, no. 7. MIT Press - Journals, pp. 1235–1270, Jul. 2019. doi: 10.1162/neco_a_01199.

[21] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8. MIT Press - Journals, pp. 1735–1780, Nov. 01, 1997. doi: 10.1162/neco.1997.9.8.1735.

[22] R. Dey and F. M. Salem, “Gate-variants of Gated Recurrent Unit (GRU) neural networks,” 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, Aug. 2017. doi: 10.1109/mwscas.2017.8053243.

[23] J. M. Guccione, A. D. McCulloch, and L. K. Waldman, “Passive Material Properties of Intact Ventricular Myocardium Determined From a Cylindrical Model,” Journal of Biomechanical Engineering, vol. 113, no. 1. ASME International, pp. 42–55, Feb. 01, 1991. doi: 10.1115/1.2894084.

[24] E. McEvoy, G. A. Holzapfel, and P. McGarry, “Compressibility and Anisotropy of the Ventricular Myocardium: Experimental Analysis and Microstructural Modeling,” Journal of Biomechanical Engineering, vol. 140, no. 8. ASME International, May 24, 2018. doi: 10.1115/1.4039947.

[25] G. A. Holzapfel and R. W. Ogden, “Constitutive modelling of passive myocardium: a structurally based framework for material characterization,” Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 367, no. 1902. The Royal Society, pp. 3445–3475, Sep. 13, 2009. doi: 10.1098/rsta.2009.0091.

[26] M. Anić, S. Savić, A. Milovanović. M. Milošević, B. Milićević, V. Simić and N. Filipović, “Solution of Fluid Flow Through the Left Heart Ventricle,” Applied Engineering Letters : Journal of Engineering and Applied Sciences, vol. 5, no. 4. Faculty of Philology, University of Belgrade, pp. 120–125, 2020. doi: 10.18485/aeletters.2020.5.4.2.

[27] M. Kojic, M. Milosevic, V. Simic, B. Milicevic, V. Geroski, S. Nizzero, A. Ziemys and N. Filipovic, “Smeared Multiscale Finite Element Models for Mass Transport and Electrophysiology Coupled to Muscle Mechanics,” Frontiers in Bioengineering and Biotechnology, vol. 7. Frontiers Media SA, Dec. 10, 2019. doi: 10.3389/fbioe.2019.00381.

[28] M. Kojic, M. Milosevic, B. Milicevic, V. Geroski, V. Simic, D. Trifunovic, G. Stankovic and N. Filipovic, “Computational model for heart tissue with direct use of experimental constitutive relationships,” Journal of the Serbian Society for Computational Mechanics, vol. 15., no. 1, pp 1-23, 2021. doi:10.24874/jsscm.2021.15.01.01.

[29] M. Kojic, M. Milosevic, A. Ziemys and N. Filipovic, (in press.) “Computational Models in Biomedical Engineering - Finite Element Models Based on Smeared Physical Fields: Theory, Solutions, and Software,” Elsevier, 2022.

[30] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” Journal of Computational Physics, vol. 378. Elsevier BV, pp. 686–707, Feb. 2019. doi: 10.1016/j.jcp.2018.10.045.