Journal of the Serbian Society for Computational Mechanics / Vol. 19 / No. 1, 2025 / pp 327-346 / review paper
(10.24874/jsscm.2025.19.01.20)

OVERVIEW OF SURROGATE MODELING FOR HUXLEY-TYPE MUSCLE SIMULATIONS IN CARDIAC BIOMECHANICS

Bogdan Milićević[1,2*] [0000-0002-0315-8263], **Miloš Ivanović**[2,3] [0000-0002-8974-2267], **Boban Stojanović**[2,3] [0000-0003-2901-0061], **Miljan Milošević**[1,2,4] [0000-0003-3789-2404], **Miloš Kojić**[2,5] [0000-0003-2199-5847] **and Nenad Filipović**[2,6] [0000-0001-9964-5615]

[1] Institute for Information Technologies, 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] Belgrade Metropolitan University, Belgrade 11000, Serbia
[5] Serbian Academy of Sciences and Arts, Belgrade 11000, Serbia
[6] Faculty of Engineering, University of Kragujevac, Kragujevac 34000, Serbia
**corresponding author*

Abstract

Huxley-type muscle models offer a physiologically grounded description of cardiac contraction but remain computationally prohibitive for large-scale, multi-scale simulations. This article reviews surrogate modeling strategies that alleviate these costs for ventricular biomechanics, with emphasis on data-driven (RNN/TCN/GRU) and physics-informed (PINN) formulations and their coupling to finite-element solvers. The data-driven approach utilizes deep neural networks trained on numerical simulation data to replicate the behavior of the Huxley model while significantly reducing processing costs. The physics-informed approach approximates solutions to Huxley's muscle contraction equation, which governs cross-bridge dynamics and force generation. By predicting the probability of myosin-actin interactions, this method enables direct calculation of stress and stiffness for finite element simulations. The coupling of these surrogate models with finite element computational frameworks allows for faster and more scalable simulations. Our goal is to provide a consolidated reference and actionable guidance for selecting and implementing surrogate approaches for Huxley-type muscle simulations.

Keywords: surrogate modeling, recurrent neural networks, physics-informed neural networks, finite element method, Huxley's muscle model

1. Introduction

Cardiac muscle simulations serve as a powerful tool for clinicians, enabling the assessment of both real-world and hypothetical physiological scenarios. The foundation of these computations lies in current understanding of the molecular mechanisms that regulate muscle contractions, leading to the formulation of biophysical muscle models. Among these, Huxley-type models are particularly valuable due to their strong physiological basis, making them well-suited for capturing the complex and non-uniform nature of muscle contractions. Additionally, these models facilitate the study of genetic influences on muscle function, as mutations can alter cross-bridge kinetics and contractile properties, contributing to cardiomyopathies.

A multiscale finite element simulation approach that employs the Huxley muscle model was introduced to analyze muscle behavior across different spatial and temporal scales as shown in Stojanović (2020), Stojanović (2015). Their methodology integrated continuum mechanics through the finite element method while defining material properties at the microscopic scale using the Huxley model. By incorporating this model into transient finite element simulations, they could compute stress and instantaneous stiffness based on factors such as muscle activation, deformation, and other material properties. However, the microscale computations required for these simulations proved computationally demanding, with single-time step calculations taking thousands of seconds. To address these bottlenecks, the researchers implemented hybrid MPI-GPU parallelization techniques for simulating a two-dimensional tongue model as shown in Ivanović (2015), Ivanović (2019). Despite these optimizations, even with high-performance computing clusters, executing multiscale finite element simulations remained time-intensive.

To mitigate these computational burdens, the development of a surrogate model that could replace the original Huxley muscle model became essential. The concept of surrogate modeling has been successfully applied in fields such as progressive damage modeling in composite materials as shown in Yan (2020) and micro-level nonlinear material modeling as shown in Ghavamian (2019), achieving strong agreement with the original models. However, muscle modeling presents additional challenges due to its dependence on multiple input features and complex learning mechanisms. Prior to our work Milićević (2022), no surrogate model had been specifically designed to replicate the Huxley muscle model within a multiscale simulation framework.

Physics-informed neural networks (PINNs) present a cutting-edge approach to supervised learning tasks while enforcing fundamental physical laws, often expressed through nonlinear partial differential equations as presented by Raissi (2017a), Raissi (2017b) and Raissi (2019). Unlike conventional neural networks, PINNs inherently embed these governing equations as prior knowledge, functioning as universal function approximators with built-in physics constraints as presented by Raissi (2017a), Raissi (2017b) and Raissi (2019). A crucial innovation in PINNs is the inclusion of a residual network, which represents the underlying physics equations and computes residual errors as described by Markidis (2021). Training a PINN involves minimizing these residuals, ensuring that the neural network's output satisfies the governing differential equations. A distinctive advantage of PINNs is their ability to learn without reliance on labeled data, precomputed simulations, or experimental results, instead, they primarily depend on evaluating residual functions. However, in certain cases, such as inverse problems, supplementary simulation or experimental data can be incorporated for supervised training. This is particularly useful when critical information, such as boundary conditions or an equation of state, is missing from a system of equations. Once trained, a PINN can serve as a highly efficient replacement for conventional numerical solvers in scientific computing as stated in Markidis (2021). One of the standout features of PINNs is their grid-free nature, allowing them to accept any point within the domain as input without requiring a predefined computational mesh. Additionally, a trained PINN can predict values across different grid resolutions without the need for retraining as stated in Markidis (2021). Their ability to treat time as a standard input variable makes them especially effective for time-dependent problems. Unlike traditional computational approaches, which require solving previous time steps sequentially, PINNs can directly predict outputs for any desired time point, avoiding the computational costs associated with time-stepping. These attributes make PINNs particularly advantageous for solving a wide range of equations, including Burgers' equation, the Navier–Stokes equations, and the Schrödinger equation as shown in Raissi (2017a), Raissi (2017b), Raissi (2019) and Markidis (2021).

Previous work (Milićević et al., 2022) introduced a surrogate representation of the Huxley muscle model based on a fixed subset of parameters, while allowing variations in muscle stretch, activation, and external loading. Within the finite element framework, stretch was supplied as input to the material model, with stress and instantaneous stiffness obtained as outputs. Because muscle behavior is history-dependent, it was shown that time-series data incorporating not only the current values of activation and stretch but also their previous states, together with past stress and stiffness values, were required for accurate prediction. Recurrent and convolutional neural networks were therefore employed and integrated into the finite element analysis environment. Numerical experiments demonstrated that these surrogates closely replicated the original Huxley model while providing a substantial computational speed-up, thereby enhancing the feasibility of large-scale simulations of left ventricular mechanics. In the present article, we revisit and contextualize these results as part of a broader overview of surrogate modeling strategies for Huxley-type simulations. Previous studies, including our own implementations demonstrated that PINNs can approximate solutions of Huxley's cross-bridge dynamics for both isometric and isotonic contractions with good agreement to conventional numerical solvers. These results highlighted the ability of PINNs to embed physics constraints without relying on large labeled datasets.

2. Methodology

This section provides a concise overview of the finite element method applied at the macroscopic scale, alongside the Huxley muscle model, which governs microscale muscle behavior within the simulation. Additionally, we outline the neural network architectures employed in constructing the data-driven surrogate model. A more detailed discussion follows, covering the step-by-step process of developing the surrogate model and its seamless integration into the finite element framework. Furthermore, we overview physics-informed neural networks (PINNs) and explain the methodology used to incorporate them into the finite element solver.

2.1 Finite element method

From a mechanical perspective, muscle can be interpreted as a dynamic mechanical system. The finite element method (FEM) is the most widely used technique for solving complex structural problems that involve both material and geometric nonlinearities. Within an incremental-iterative framework, a muscle's equilibrium state can be established by modeling it as a structure composed of fiber components. Activation triggers fiber contraction within the compliant connective tissue, as described by Kojic (2005) and Bathe (1996). The equilibrium equation governing a finite element structure in its deformed state at a given time step (t) and iteration (i) is expressed as:

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

where ${}^{t+\Delta t}F_{ext}$, ${}^{t+\Delta t}F_{pass}^{(i-1)}$, ${}^{t+\Delta t}F_{act}^{(i-1)}$ represent external loads, passive internal nodal forces, and active molecular forces incorporated 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; $\delta U^{(i)}$ represents increments of nodal displacements at iteration (i). The development of active force ${}^{t+\Delta t}F_{act}^{(i-1)}$ and stiffness ${}^{t+\Delta t}K_{act}^{(i-1)}$ is governed by the deformation rate along the primary muscle fiber direction. The total stress $\bar{\sigma}_m$ arises from both the contractile forces generated by the muscle and the passive elastic behavior of the surrounding connective tissue and non-contractile structures:

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

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.

2.2 Huxley's muscle model

Huxley's research focused on understanding filament interactions within muscle tissue, particularly the probability of forming temporary molecular connections, known as cross-bridges, between myosin heads and actin filaments inside sarcomeres as stated in Huxley (1957), Gordon (1966), Stojanović (2007), Mijailovich (2010). The function $n(x,t)$ represents the fraction of cross-bridges formed at a given time, defined by the relative displacement x of the nearest actin-binding site with respect to the myosin head's equilibrium position. This relationship is governed by the following equation:

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

where *f(x,a)* and *g(x)* represent the cross-bridge attachment and detachment rates of cross-bridges, respectively, while *v* denotes the filament sliding velocity, positive in the direction of contraction, and *a* corresponds to a muscle activation expressed as a time-dependent function as stated in Mijailovich (1996). The partial differential Equation (3) is typically solved using the method of characteristics with initial condition $n(x,0)=0$. After obtaining the $n(x,t)$ values, the generated force *F* within the muscle fiber and stiffness *K*, can be computed using the following relations:

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

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

where *k* denotes the stiffness of the cross-bridges. The stress and instantaneous stiffness are 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 λ is stretch, σ_{iso} is maximal stress achieved during isometric conditions, L_0 is the initial length of sarcomere, and F_{iso} is a maximal force achieved during isometric conditions.

These stresses and stiffness values are subsequently incorporated into the finite element framework to yield ${}^{t+\Delta t}F_{act}^{(i-1)}$ and ${}^{t+\Delta t}K_{act}^{(i-1)}$.

Muscle activation is derived from intracellular calcium concentration. The calcium profile is modeled using the parametric function proposed by Hunter (1998):

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

where $Ca_i(t)$ is intracellular Ca^{2+} concentration, starting from a resting value Ca_0 and reaches its maximum Ca_{max} at time $t = \tau Ca$. The calcium concentration is transformed into an activation α according to:

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

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

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

where λ is stretch, pC_{50ref}, n_{ref}, β_1 and β_2 are constant coefficients. Values $n_{ref} = 5.2$, $pC_{50ref} = 6.18$, $\beta_1 = 1.95$ and $\beta_2 = 0.31$ were taken from Hunter (1998).

2.3 Recurrent and convolutional neural networks

Recurrent neural networks (RNNs) are widely utilized for analyzing time-dependent data, as they allow previous outputs to influence future inputs while preserving hidden state information. However, training RNNs presents significant challenges due to their tendency to suffer from exploding or vanishing gradients, which can hinder effective learning as stated in Jain (1999) and Pascanu (2012). Several advancements have been introduced to mitigate these issues in conventional RNN architectures.

One of the most well-known improvements is the long short-term memory (LSTM) network, which combats the vanishing gradient problem by incorporating an internal memory mechanism that facilitates long-term retention of information as stated in Yu (2019) and Hochreiter (1997). Another widely used alternative is the gated recurrent unit (GRU), which simplifies the architecture by omitting internal memory, resulting in faster computations while maintaining performance comparable to LSTMs as stated in Dey (2017). A more recent variant, the nested long short-term memory (nested LSTM), extends the LSTM model by incorporating additional memory layers, aiming to enhance long-term information retention as stated in Moniz (2018) . The underlying premise is that increased memory capacity enables more effective learning of complex temporal dependencies. Given their potential for improving muscle surrogate modeling, we provide an overview of the GRU and nested LSTM architectures, emphasizing their advantages in capturing time-dependent muscle dynamics.

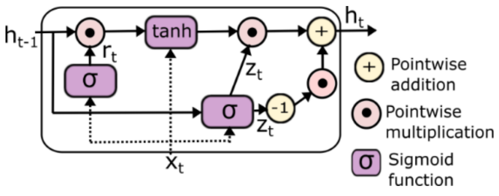

Fig. 1. Gated recurrent unit (GRU) scheme (according to Milićević (2022))

The GRU cell is defined by two gating mechanisms: the update and reset gates. As presented in Fig. 1, the dynamic of the GRU cell is described by:

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

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

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

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 components, b_r, b_z denote 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:

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

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

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

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

$$h_t = \sigma_i \odot tanh\left(c_t\right) \tag{18}$$

where the sigmoidal functions $\sigma_i, \sigma_f, \sigma_o$ represent the activation functions for input (*i*), forget (*f*), and output gate (*o*) respectively; $tanh$ is a tangential hyperbolic function; c denotes the cell value, *h* denotes the hidden state of the cell, W denotes weights, *b* denotes bias, and x is an input vector. The nested LSTM modifies the standard update rule for c_t by replacing the additive operation with a learned stateful function: $c_t = m_t\left(f_t \odot c_{t-1}, i_t \odot g_t\right)$. The function m_t is parameterized by an additional LSTM cell, which introduces an inner memory state, producing the nested LSTM architecture. The input and the hidden states of the memory function in a Nested LSTM are given by:

$$\tilde{h}_{t-1} = f_t \odot c_{t-1} \tag{19}$$

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

Beyond recurrent networks, convolutional neural networks can also model sequential data as demonstrated by Bai (2018) and Van den Oord (2016). 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:

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

where d is the dilation factor, k is the filter size, and $s-di$ accounts for the direction of the past. The scheme of TCN is shown in Fig. 2.

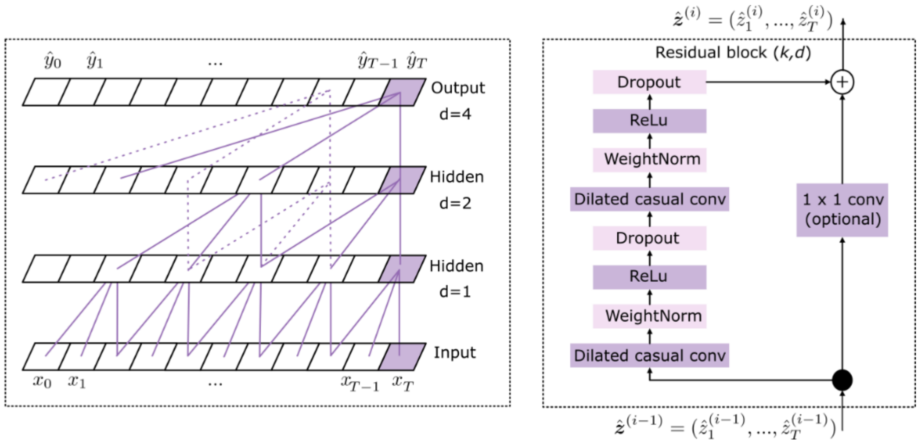
$\hat{y}_0$ $\hat{y}_1$ ... $\hat{y}_{T-1}$ $\hat{y}_T$
Output d=4
Hidden d=2
Hidden d=1
Input
x_0 x_1 ... x_{T-1} x_T
$\hat{z}^{(i)} = (\hat{z}_1^{(i)}, \dots, \hat{z}_T^{(i)})$
Residual block (k,d)
Dropout
ReLu
WeightNorm
Dilated casual conv
1 x 1 conv (optional)
$\hat{z}^{(i-1)} = (\hat{z}_1^{(i-1)}, \dots, \hat{z}_T^{(i-1)})$

Fig. 2. Temporal convolutional network scheme (according to Milićević (2022))

2.4 Data-driven surrogate model

To build the data-driven surrogate model, numerical experiments were generated to provide input data for finite element simulations. These experiments used a single 2D finite element model with varying boundary conditions depending on the experiment type. Four distinct numerical experiment types can be distinguished:

1. **Isotonic Contraction:** Motion constraints were applied to specific nodes while activation or calcium concentration was varied. Upon deactivation, the muscle returned to its initial position.
2. **Quick Release:** The muscle was fully activated with all nodes constrained until a set time step, at which point some constraints were lifted, leading to a force drop below 25% of the maximum.
3. **Prescribed Force:** External forces were applied at selected nodes, with constraints mirroring those in isotonic contraction experiments.

4. **Prescribed Displacement:** Specific displacements were assigned to nodes under similar constraints as isotonic contractions.

Each experiment type introduced variations in activation functions, prescribed forces, or displacements, ensuring a diverse dataset for training. Once experiment data was generated, finite element simulations were conducted using the Huxley muscle model and recorded key parameters such as activation, stretch, stress, and stress derivatives. The raw data was then processed into a time-series format, forming a structured input tensor for the neural network. After model training, the same numerical experiments were re-run using the surrogate model to compare its performance against the original Huxley model. If discrepancies arose, model is iteratively refined and the dataset was expanded. Once the model demonstrated strong agreement, additional randomly generated experiments were introduced for further validation. The dataset used for training and validation comprised of **160 numerical experiments**, divided into: 45 isotonic contraction experiments, 20 quick release experiments, 40 prescribed force experiments, 55 prescribed displacement experiments. Every **fourth experiment** was allocated for validation, while the remaining were used for training. An additional **eight experiments** (two of each type) were used to assess model performance.

The data-driven surrogate model was embedded into the finite element framework through a structured pipeline involving initialization, input updating, prediction, and result retrieval. During initialization the neural network architecture, trained weights, and preprocessing parameters were loaded. The input tensor was initialized, with stretch values set to one and all other values to zero. At each time step, activation and stretch values were updated at each integration point, the neural network predicted stress and stiffness increments and input tensor was shifted back in time to prepare for the next step.

2.5 Physics-informed neural networks

An alternative approach to surrogate modeling of muscle behavior is based on physics-informed neural networks (PINNs). These networks integrate deep learning with fundamental physical principles, particularly for solving partial differential equations (PDEs). Unlike purely data-driven models, PINNs enforce compliance with governing physical laws by incorporating a physics-based loss function, which quantifies the deviation of the predicted solution from the PDE constraints. By minimizing this loss, the model inherently adheres to the fundamental physics of the system.

A key innovation in PINNs is the use of residual networks, inspired by deep learning techniques in which residual connections facilitate training (Fig. 3). These networks encode the governing equations by calculating the residual error, which is the difference between the left-hand and right-hand sides of the PDE. The objective is to drive this residual toward zero, ensuring the model's predictions are physically consistent. Another crucial component of PINNs is automatic differentiation, which enables precise computation of gradients needed to optimize network parameters during training. Standard optimization techniques, such as stochastic gradient descent (SGD) or the Adam optimizer, are commonly used to refine the model's predictions. This automated approach ensures the network learns solutions that align with the underlying physics.

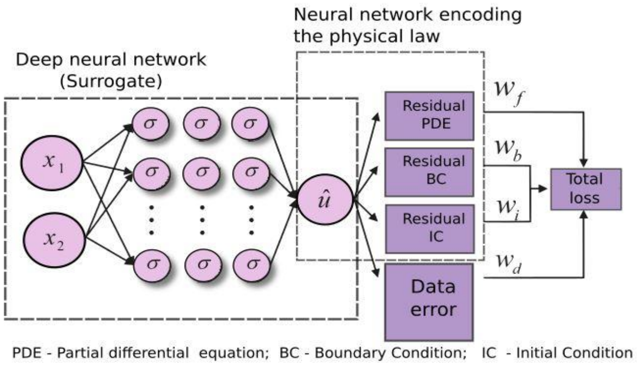
Neural network encoding the physical law
Deep neural network (Surrogate)
x_1
x_2
σ
$\hat{u}$
Residual PDE
Residual BC
Residual IC
Data error
w_f
w_b
w_i
w_d
Total loss
PDE - Partial differential equation; BC - Boundary Condition; IC - Initial Condition

Fig. 3. Schematic of the Physics-informed neural network (PINN)

2.6 Integration of PINN-Based Surrogate Model into Finite Element Analysis

The process of integrating a PINN-based surrogate model into finite element analysis follows a structured sequence of steps, ensuring computational efficiency while maintaining the accuracy of muscle behavior simulations (Table 1). At the beginning of the simulation, the neural network is loaded from a pre-trained file. Several key parameters are defined, including the spatial region where myosin-actin interactions will be observed. This involves specifying the start and end positions of actin-binding sites, the number of spatial subdivisions, and essential model properties such as initial sarcomere length, cross-sectional area, and stiffness. Additionally, an input tensor is allocated, with its size determined by the number of spatial divisions, the number of input features, and the total number of integration points within the model. To ensure consistency, the tensor is populated with static spatial position values that remain unchanged throughout the simulation. As the finite element simulation progresses, certain attributes within the input tensor need to be updated dynamically. This is handled by a function that assigns new values for muscle activation, the current time step, and stretch values at both the current and previous time steps. Since each integration point in the finite element mesh requires multiple spatial subdivisions, these updated values are replicated across the corresponding segments of the input tensor to ensure proper data alignment. Once the input tensor has been populated with the latest values, it is fed into the neural network, which predicts the probability of myosin heads attaching to actin-binding sites. These probabilities serve as the foundation for stress and stiffness calculations in the muscle model. A separate function is then used to extract the predicted probabilities for each integration point, enabling the computation of stress and its derivatives. By embedding the PINN-based surrogate model into the finite element framework, the simulation gains a substantial computational advantage, reducing processing time while preserving the fidelity of muscle contraction mechanics. This integration allows for more efficient large-scale simulations, addressing the limitations associated with traditional muscle models.

Input and output tensor description. The input tensor is defined at every finite element integration point and contains the spatial coordinate of actin-binding site subdivisions (x), the current simulation time (t), the muscle activation level (α_t), the current stretch (λ_t), and the previous stretch (λ_{t-1}). These values are replicated across all spatial subdivisions, resulting in a tensor of size $N_{ip} \times N_x \times F_{in}$, where N_{ip} is the number of integration points, N_x the number of subdivisions, and F_{in} the number of input features. At each time step, only the time-dependent entries are updated, while the spatial coordinate remains fixed. The output tensor consists of the predicted probabilities of cross-bridge attachment (n) for each subdivision, which are subsequently reduced to compute stress and stress derivatives at every integration point. These quantities are then supplied back to the finite element solver to update the global system of equations.

Pseudo code for finite element analysis with physics-informed neural network at microlevel as a replacement for Huxley's muscle model

```
1:  huxpinn_init(NQP, K, xstart, xend, xdiv, L0, A, modelfile)
2:  loop over time steps:
3:      while convergence criteria not met:
4:          for ip in 0..NQP-1:
5:              huxpinn_set_values(ip, time, activation, stretch, stretch_prev)
6:          huxpinn_predict()
7:          for ip in 0..NQP-1:
8:              huxpinn_get_values(ip, stress, dstress)
9:  huxpinn_destroy()
```

Table 4 Pseudo code for PINN-based surrogate model integration into FEM framework

3. Results

This section presents the results obtained from the proposed data-driven surrogate models across various numerical simulations. The performance of these models is assessed in terms of both accuracy and computational efficiency, highlighting the achieved acceleration compared to the original Huxley model. Additionally, the outcomes of physics-informed surrogate models are demonstrated in cases of isometric and isotonic contractions. We also introduce multi-scale left ventricle model. In these simulations, the most effective surrogate model from our study is seamlessly integrated as a replacement for the Huxley model, functioning as the muscle material model within the finite element framework.

3.1 Evaluation of Neural Network Architectures for Surrogate Modeling

During the development of the surrogate model, multiple neural network architectures were explored. he selected neural network architectures for surrogate modeling varied in structure and complexity, each incorporating different configurations of hidden layers and parameter sizes. The Temporal Convolutional Network (TCN) consisted of 11 convolutional layers, each equipped with 192 filters. The dilation rates followed an exponential progression of 1, 2, 4, 8, and 16, with a kernel size of 4. This architecture was designed to capture temporal dependencies efficiently while maintaining a relatively compact model with 928,706 trainable parameters.

The Nested Long Short-Term Memory (Nested LSTM) model featured a single nested layer with a depth of 8, where each depth level contained 128 neurons. This hierarchical structure allowed the model to process long-range dependencies in the data while maintaining a total of 992,002 trainable parameters.

The Gated Recurrent Unit (GRU) network followed a multi-layered design, incorporating five stacked GRU layers. The first, fourth, and fifth layers contained 128 neurons, while the second and third layers were expanded to 256 neurons each, increasing the network's capacity for learning temporal patterns. This architecture resulted in a total of 992,770 trainable parameters, making it one of the more complex models in the study.

A hybrid approach combining Nested LSTM and TCN was also explored, integrating a nested LSTM layer with a depth of 6, each containing 128 neurons, alongside a TCN structure comprising seven convolutional layers with 128 filters. The convolutional layers used dilation rates of 1, 2, and 4, with a kernel size of 4. This hybrid architecture slightly reduced the parameter count to 991,874 while leveraging the strengths of both recurrent and convolutional networks.

Similarly, the GRU-TCN model integrated recurrent and convolutional layers, featuring a GRU architecture where the first and fourth layers contained 64 neurons, while the second and third layers expanded to 256 neurons. The TCN component consisted of seven convolutional layers, each with 128 filters, maintaining the same dilation structure as the Nested LSTM-TCN model. This hybrid design slightly reduced the overall parameter count to 974,978 while aiming to balance computational efficiency and predictive accuracy.

Across all architectures, the Rectified Adam optimizer was employed with an initial learning rate 10^{-3}, momentum parameters set to $\boldsymbol{\beta}_1 = 0.99$, $\boldsymbol{\beta}_2 = 0.9999$, and a gradient clipping norm of 10^{-4}. The models were trained using the Huber loss function with a smoothing parameter of $\boldsymbol{\delta} = 58 \times 10^{-6}$. Training was conducted over a maximum of 50,000 epochs, with a batch size of 16,384. To prevent overfitting and optimize computational efficiency, an early stopping mechanism was implemented, halting training if no improvement was observed over 500 consecutive epochs.

The evaluation of the neural networks' predictive performance was based on the correlation between true and predicted stress values across different datasets and numerical simulations. The results indicated that all tested networks exhibited strong agreement with actual values when evaluated on training and validation data. However, their performance declined in numerical simulations due to the accumulation of small prediction errors over multiple time steps. Among the networks, GRU achieved the highest correlation coefficients, with values of 0.99999972, 0.999999640, and 0.99999518 for training, validation, and test datasets, respectively. It also exhibited the most accurate behavior in numerical simulations, with correlation values of 0.99977, 0.99965, and 0.989 across training, validation, and test simulations. Furthermore, GRU had the lowest standard deviation in correlation coefficients, indicating stable performance across different experiments.

The Nested LSTM-TCN hybrid demonstrated improved accuracy over its individual components, achieving correlation coefficients of 0.9999951, 0.9999956, and 0.999971 for training, validation, and test datasets. However, its simulation accuracy was lower than GRU, with correlation values of 0.981, 0.981, and 0.812 in training, validation, and test simulations. The GRU-TCN hybrid, on the other hand, failed to outperform the standalone GRU model, achieving a training data correlation of 0.99999972, but slightly lower scores of 0.9999984 and 0.999959 in validation and test datasets. While it performed well in training simulations (0.9324) and validation simulations (0.998), its test simulation correlation was slightly lower at 0.965.

A notable trend was the decline in correlation values for all networks during test simulations, reflecting challenges in generalization. The GRU model exhibited the smallest drop in correlation coefficients, making it the most reliable option for muscle surrogate modeling. The GRU-TCN hybrid, despite strong performance in some areas, did not surpass the standalone GRU model in overall effectiveness.

This section provides an analysis of the GRU-based surrogate model's performance against the original Huxley model across different numerical experiments. Stress-time diagrams generated from simulations with both models demonstrate their alignment, with instantaneous stiffness omitted for simplicity due to similar trends. The surrogate model was evaluated using training, validation, and test datasets. Fig. 4 highlights the test cases, assessing the model's ability to generalize beyond the training data. Across all test experiments, the surrogate model closely matched the original Huxley model, demonstrating its reliability in accurately replicating muscle contraction behavior.

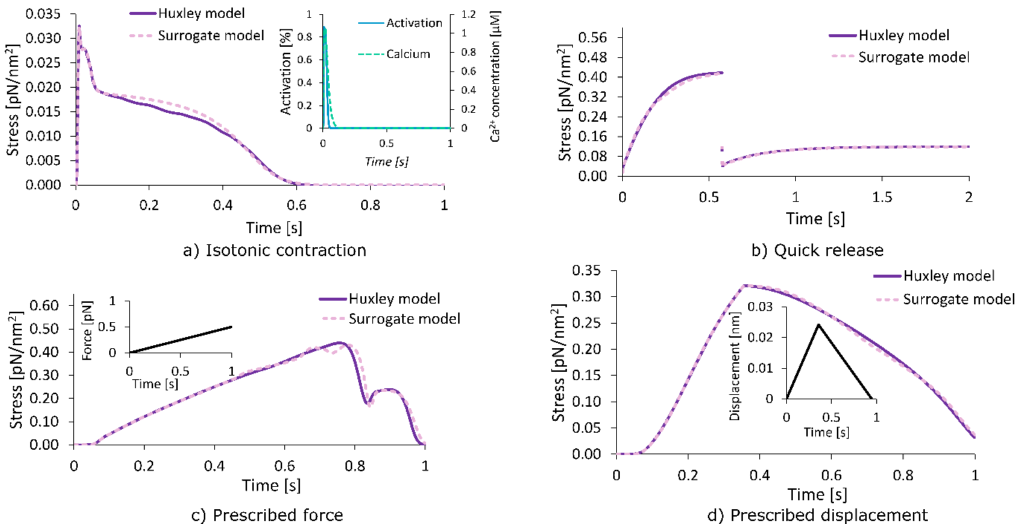

Fig. 4. Stresses obtained in numerical experiments used to test the neural network. a) Isotonic contraction), b) quick release, c) prescribed force, d) prescribed displacement (according to Milićević (2022))

3.2 Enhancing Computational Efficiency Through Data-driven Surrogate Modeling

The primary goal of integrating a surrogate model is to significantly accelerate numerical simulations while maintaining accuracy. This section evaluates the computational speed-up achieved by replacing the original Huxley muscle model with the GRU-based surrogate model. A comparison of execution times between the traditional multiscale finite element simulation and its surrogate counterpart highlights the efficiency gains. The analysis includes both sequential and parallel implementations, with parallelization applied at the integration point level, utilizing four MPI processes corresponding to the four integration points present in all numerical experiments.

The surrogate model demonstrated a dramatic reduction in computational time, achieving a speed-up of approximately 50 times for quick release, prescribed force, and prescribed displacement experiments. For isotonic contraction experiments, the speed-up was slightly lower but still substantial at around 25 times. These improvements translate to an order-of-magnitude acceleration compared to the sequential execution of the original Huxley model. The ability to perform simulations at such high speeds makes the surrogate model a practical and scalable alternative, particularly when dealing with more complex simulations involving a larger number of finite elements.

3.3 Physics-informed neural network results

This section reports the outcomes achieved through a physics-informed neural network (PINN) framework. To simulate isometric muscle contraction, we constructed a neural network comprising eight hidden layers with 20 neurons per layer, each activated by a hyperbolic tangent function. Training was performed by minimizing the residual of Huxley's muscle contraction equation, while enforcing initial conditions to guarantee solution uniqueness. Accuracy was further improved by employing the neural tangent kernel (NTK) approach, which adaptively updated the weights to balance collocation points assigned to the PDE residual with those used for initial conditions. The training dataset consisted of a grid spanning displacement values (x) from –20.8 nm to 62.4 nm and time values (t) from 0 to 0.5 s. Following training, stress was computed from the obtained n values and benchmarked against results derived using the method of characteristics. As illustrated in Fig. 5, the PINN solutions closely match those from the method of characteristics, confirming the robustness and effectiveness of the physics-informed approach.

To investigate isotonic muscle contraction, we carried out a series of numerical simulations driven by randomly generated activation functions, producing contraction followed by relaxation to the initial state. These simulations were implemented using a single finite element, as shown in Fig. 6. The boundary conditions imposed were: full constraint at point A, vertical sliding permitted at point C, and free movement at points D and B.

During the simulations, input variables—including displacement (x), time, activation, and stretch—were recorded together with the corresponding n values as outputs. This dataset was then used to train a neural network that embedded the Huxley equation with initial conditions. The network architecture comprised eight hidden layers with 200 neurons per layer, each employing a hyperbolic tangent activation.

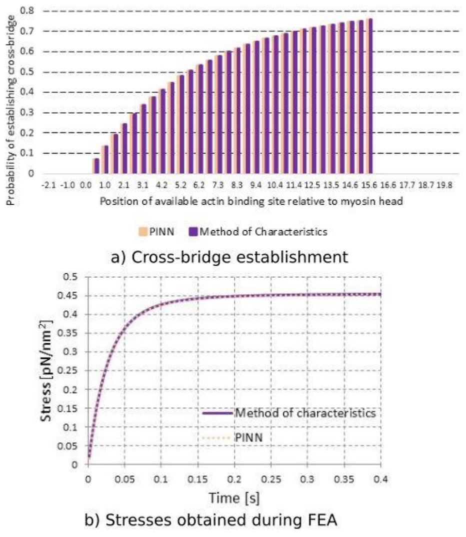

Fig. 5. Comparison of method of characteristics and PINN in the simulation of isometric contraction

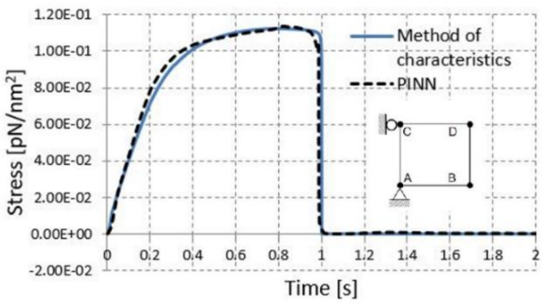

Fig. 6. Comparison of method of characteristics and PINN in the simulation of isotonic contraction case

Once trained, the neural network was incorporated as a constitutive material model within the finite element framework. At the macroscopic level, the finite element routine supplied the network with activation, time, and current stretch values. Using these inputs along with the x coordinates, the network predicted n values, which were subsequently employed to compute stress and its derivatives. As shown in Fig. 6, the stresses predicted by the PINN model closely matched those obtained through the method of characteristics.The isotonic case, characterized by variable muscle activation and nonzero contraction velocity, introduced additional complexity, leading to slightly reduced accuracy compared to the isometric scenario. However, despite these challenges, the stress values computed via the method of characteristics and PINN remained closely aligned, demonstrating the robustness of the physics-informed approach.

Finally, we assessed the computational efficiency of multi-scale finite element simulations by comparing the performance of PINN with the method of characteristics at the microscopic scale. The results indicate that PINN achieves a fourfold improvement in computational speed relative to the method of characteristics.

It should be emphasized, however, that the physics-informed and data-driven strategies are not strictly comparable, as they employ different input features and outputs. The key strength of the physics-informed framework lies in its explicit formulation for solving partial differential equations. Nonetheless, this approach requires larger input tensors, since it depends on the discretization of binding site positions. By contrast, the data-driven model is both more memory-efficient and computationally faster, making it the preferable option for large-scale simulations.

In light of these findings, a GRU (Gated Recurrent Unit)–based data-driven surrogate model was adopted for simulating left ventricular mechanics. Looking ahead, further advances in physics-informed approaches may enhance their applicability, particularly in leveraging their inherent ability to resolve complex differential equations.

The memory footprint of a PINN-augmented FE simulation scales approximately with input tensor size. By construction, PINNs require larger input tensors than purely data-driven surrogates because the binding-site coordinate x must be discretized and threaded through the network. In our experiments, this led to larger per-step tensors compared with RNN/GRU surrogates. At the whole-simulation level, prior measurements in a comparable multiscale set-up show how memory scales with the microscopic solver choice: for ~4,000 integration points, the original Huxley micro-solver required ~23.96 GB, whereas a neural surrogate required ~812 MB, illustrating the substantial savings available when replacing the classical micro-solver.

While these numbers come from a data-driven surrogate, they contextualize the expected order-of-magnitude savings versus the full Huxley model; PINN tensors sit between these two extremes.

Runtime scaling shows similar trends: replacing the method of characteristics with a PINN at the micro-scale yielded ≈4× speedup in multiscale FE tests (isometric/isotonic cases), with accuracy close to the reference solutions.

3.4. Cardiac cycle with the left ventricle model

Among the proposed surrogate models, the GRU-based network demonstrated the highest potential, making it the optimal choice for simulating the cardiac cycle using both a parameterized and an echocardiography-derived left ventricle model.

The parametric left ventricle model integrates both fluid and solid wall components, with its structure designed to capture the biomechanical interactions within the heart. The simulation focused on half of the left ventricle, incorporating mitral and aortic branches at the top, which are connected to the base through a geometrically defined connective structure. The geometry of the fluid domain is determined by specifying the lengths and diameters of each component, along with parameters that control mesh density to ensure computational accuracy.

The solid domain is seamlessly coupled with the fluid model, forming a cohesive representation of ventricular mechanics. The solid structure is defined by setting wall thickness and the number of layers composing the myocardial wall. Within this domain, muscle fiber orientation is represented by arrows, illustrating a helical pattern that transitions smoothly across the wall thickness. At the endocardium, fibers are aligned at approximately 60° (indicated by yellow arrows), while at the epicardium, they shift to -60° (represented by red arrows). The intermediate layers display a gradual linear shift in fiber direction, with fibers in the mid-wall appearing in pink. To ensure realistic mechanical behavior, the solid portion of the model includes around 4000 integration points, allowing for accurate stress and strain computations throughout the simulation.

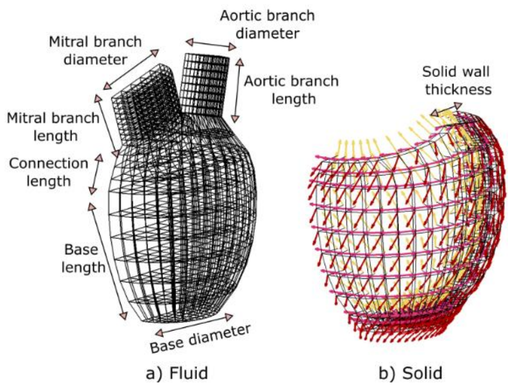

Fig. 7. Parametric left ventricle model consisting of fluid and solid domain with muscle fibers (according to Milićević (2022))

The transient simulation encompasses the full cardiac cycle, beginning with the initial geometry depicted in Fig. 7, which represents the onset of diastole. During this phase, blood enters the left ventricle at a steady velocity of 100 mm/s through the mitral valve, while the aortic valve remains closed. As diastole progresses, the velocity at the mitral valve gradually decreases until it reaches zero, at which point both valves shut simultaneously. This marks the isovolumetric contraction phase, where the volume of the ventricle remains constant while muscle activation begins, preparing for systole.

At the onset of systole, the aortic valve opens, initiating blood ejection from the ventricle as the muscle fibers gradually deactivate. The contraction-driven forces propel blood through the aortic valve, completing the cardiac cycle. The Holzapfel material model is employed to simulate passive mechanical properties of the myocardial tissue, ensuring an accurate representation of the heart's structural response. To model active stress generation, the GRU-based surrogate model replaces the computationally demanding Huxley model, significantly improving simulation efficiency while maintaining physiological accuracy.

The displacement fields within the solid ventricular wall at different phases of the cardiac cycle are illustrated at the onset of diastole (t = 0.1s, t = 0.2s, t = 0.3s) and during systole, including its initiation and midpoint (t = 0.6s, t = 0.7s, t = 0.8s) are shown in Fig. 8. The velocity field within the fluid domain is also analyzed at key moments, capturing flow dynamics at the beginning and midpoint of diastole, as well as at the midpoint of systole, is shown in Fig. 9. These visualizations highlight the contraction of the ventricle, driven by muscle forces at the onset of systole, which facilitates blood ejection through the aortic valve. As the cardiac cycle progresses, muscle deactivation leads to a gradual return of the ventricle to its original geometry, preparing for the next cycle. The observed agreement between the surrogate model's predictions and expected physiological behavior reinforces its viability for use in large-scale cardiac simulations, demonstrating its effectiveness in capturing the mechanics of left ventricular function.

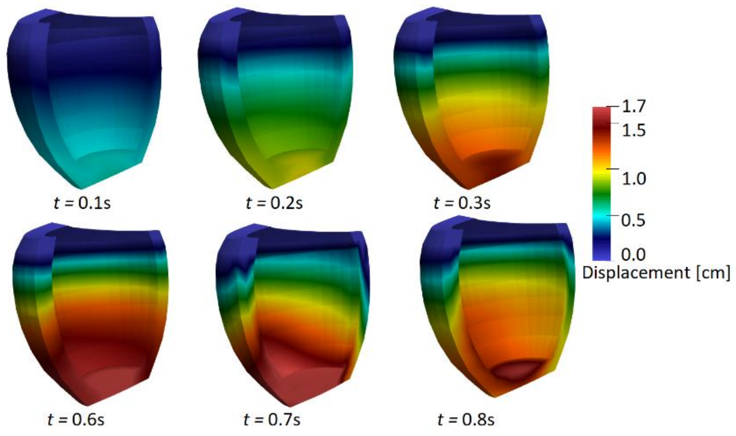

Fig. 8. Displacement field at start of the diastole (t=0.1s, t=0.2s, t=0.3s), and at the start and the middle of the systole (t=0.6s, t=0.7s, t=0.8s) within parametric LV model (according to Milićević (2022))

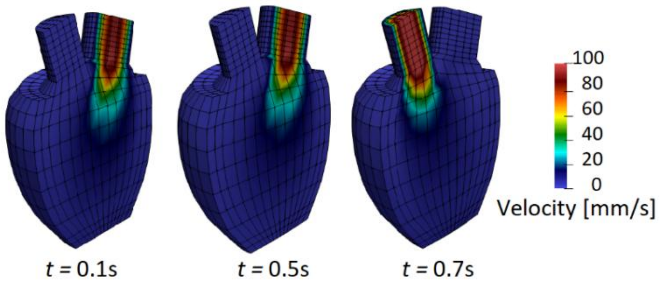

Fig. 9. Left ventricle fluid velocity field at the start and the middle of the diastole, and the middle of the systole (according to Milićević (2022))

4. Conclusions

The Huxley muscle model, grounded in fundamental muscle physiology, provides a highly accurate representation of muscle contraction dynamics but comes with significant computational demands. This complexity makes it impractical for large-scale multiscale simulations. To address this issue, surrogate modeling offers an alternative by approximating the original model with a computationally efficient representation. This paper explores the development of surrogate models for Huxley's muscle behavior using artificial neural networks, enabling faster simulations while maintaining accuracy.

Surrogate modeling can be approached in two ways: data-driven neural networks and physics-informed neural networks (PINNs). The first method involves generating numerical experiments, collecting data from finite element simulations, and training neural networks to replicate the behavior of the Huxley model. The second approach leverages PINNs to approximate the solutions to partial differential equations, ensuring that predictions adhere to physical constraints.

4.1 Data-Driven Surrogate Modeling with Neural Networks

To generate training data, a numerical experiment generator was developed to produce a variety of muscle contraction scenarios. These experiments included isotonic contractions, quick-release scenarios, prescribed force applications, and prescribed displacements. The finite element simulations used the original Huxley model at the microscopic scale, providing detailed stress and stiffness responses based on muscle activation, stretch, and material properties. Since muscle behavior is history-dependent, the collected data was structured into time series, allowing the neural networks to learn from the sequence of events rather than isolated inputs.

At each time step, the neural network received inputs representing muscle activation, stretch, stress values, and stress derivatives from both the current and previous time steps. Training these networks to mimic the finite element method's accuracy posed a significant challenge, as even small deviations could compound over time and lead to unreliable results. Furthermore, ensuring that the neural network could generalize effectively to unseen scenarios was critical. Without sufficient generalization, the network could produce inconsistent stress predictions, destabilizing the overall simulation.

Various neural network architectures were explored, including temporal convolutional networks (TCN), nested long short-term memory (Nested LSTM), gated recurrent units (GRU), and hybrid models combining recurrent and convolutional layers. To improve performance, several machine learning techniques were applied, such as gradient clipping and normalization.

One key strategy was reformulating the learning task so that the network predicted changes in stress rather than absolute values, preventing it from over-relying on high autocorrelation in stress sequences. Additionally, scaling factors were introduced to enhance numerical stability and improve predictive accuracy.

After extensive testing, the GRU-based neural network emerged as the most effective surrogate model for muscle contraction, demonstrating both high accuracy and strong generalization across numerous numerical experiments. This model proved to be significantly more computationally efficient than the original Huxley model while preserving the key biomechanical properties necessary for realistic simulations.

4.2 Physics-Informed Neural Networks for Surrogate Modeling

A second approach to surrogate modeling involved the use of physics-informed neural networks (PINNs), which integrate deep learning with the mathematical principles governing muscle contraction. Unlike purely data-driven models, PINNs incorporate the governing equations directly into the network architecture, ensuring that predictions align with fundamental physical laws.

The key innovation in this approach was the inclusion of an auxiliary neural network that encoded the partial differential equation governing muscle contraction, specifically the Huxley equation. Instead of relying solely on data, the PINN generated an approximate solution to this equation while minimizing the residual error, which quantified how well the predicted solution satisfied the governing physics. By leveraging automatic differentiation and optimization techniques, the model iteratively adjusted its parameters to reduce this residual error, effectively learning to solve the Huxley equation with high accuracy.

This paper demonstrates the application of PINNs to approximate solutions for both isometric and isotonic contractions. The results showed a strong correlation between the solutions obtained through the traditional method of characteristics and those produced by the PINN-based approach. This similarity confirmed that PINNs could serve as a reliable alternative for modeling muscle contraction while significantly reducing computational overhead.

4.3 Computational Efficiency and Real-World Applications

The GRU-based surrogate model proved to be orders of magnitude faster than the original Huxley model, making it possible to simulate large-scale, computationally intensive scenarios that would otherwise be infeasible. This paper illustrates the application of this surrogate model in simulating the left ventricular cardiac cycle, a process that would be significantly more challenging with the full Huxley model due to the complexity of the micro-scale computations.

Left ventricle were considered, and the simulation covered the entire cardiac cycle, beginning with diastolic expansion due to blood inflow, followed by systolic contraction triggered by muscle activation, leading to blood ejection from the ventricle. The ability to efficiently model these physiological processes using surrogate models represents a major advancement in computational cardiac biomechanics.

Beyond the creation of surrogate models, this paper also discusses their integration into finite element software frameworks. The surrogate models presented here were developed for a specific set of Huxley model parameters, but future research could extend this approach to different parameter sets, accounting for genetic mutations that alter muscle protein behavior. Such developments would enable the study of how genetic variations influence the mechanical response of the left ventricle, providing valuable insights into the impact of muscle-related diseases on cardiac function.

Acknowledgements: This study has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101080905 (STRATIFYHF). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or granting authority – European Health and digital executive agency (HADEA). Neither the European Union nor the granting authority can be held responsible for them. This research was also supported by the Ministry of Science, Technological Development and Innovation of the Republic of Serbia, contract number [451-03-136/2025-03/200378 (Institute of Information Technologies, University of Kragujevac) and 451-03-136/2025-03/200107 (Faculty of Engineering, University of Kragujevac)].

References

Bai S, Kolter JZ, Koltun V (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, arXiv, 1803.0127.

Bathe KJ (1996). Finite element procedures, Prentice-Hall, Englewood Cliffs, NJ.

Dey R, Salem FM (2017). Gate-variants of Gated Recurrent Unit (GRU) neural networks, 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), IEEE.

Ghavamian F, Simone A (2019). Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network, Computer Methods in Applied Mechanics and Engineering, 357, 112594.

Gordon M, Huxley AF, Julian FJ (1966). The variation in isometric tension with sarcomere length in vertebrate muscle fibres, J. Physiol., 184(1), 170–192.

Hochreiter S, Schmidhuber J (1997). Long Short-Term Memory, Neural Computation, 9(8), 1735–1780.

Hunter PJ, McCulloch AD, ter Keurs HEDJ (1998). Modelling the mechanical properties of cardiac muscle, Progress in Biophysics and Molecular Biology, 69(2–3), 289–331.

Huxley F (1957). Muscle structure and theories of contraction, Prog. Biophys. Biophys. Chem, 7, 255–318.

Ivanović M, Kaplarević-Mališić A, Stojanović B, Svičević M, Mijailovich SM (2019). Machine learned domain decomposition scheme applied to parallel multi-scale muscle simulation, The International Journal of High Performance Computing Applications, 33(5), 885–896.

Ivanović M, Stojanović B, Kaplarević-Mališić A, Gilbert R, Mijailovich S (2015). Distributed multi-scale muscle simulation in a hybrid MPI–CUDA computational environment, SIMULATION, 92(1), 19–31.

Jain LC, Medsker LR (1999). Recurrent Neural Networks: Design and Applications, first ed., CRC Press, Boca Raton, FL, USA.

Kojic M, Bathe KJ (2005). Inelastic analysis of solids and structures, Springer.

Markidis S (2021). The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers, Frontiers in Big Data, 4.

Mijailovich SM, Fredberg JJ, Butler JP (1996). On the theory of muscle contraction: filament extensibility and the development of isometric force and stiffness, Biophys. J., 71(3), 1475–1484.

Mijailovich SM, Stojanovic B, Kojic M, Liang A, Wedeen VJ, Gilbert RJ (2010). 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, 109(5), 1500–1514.

Milićević B, Ivanović M, Stojanović B, Milošević M, Kojić M, Filipović N (2022). Huxley muscle model surrogates for high-speed multi-scale simulations of cardiac contraction, Computers in Biology and Medicine, 149, 105963.

Moniz JRA, Krueger D (2018). Nested LSTMs, arXiv, 1801.10308.

Pascanu R, Mikolov T, Bengio Y (2012). On the difficulty of training Recurrent Neural Networks, arXiv, 1211.5063.

Raissi M, Perdikaris P, Karniadakis GE (2017a). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations, arXiv, 1711.10561.

Raissi M, Perdikaris P, Karniadakis GE (2017b). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations, arXiv, 1711.10566.

Raissi M, Perdikaris P, Karniadakis GE (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378, 686–707.

Stojanovic B, Kojic M, Rosic M, Tsui CP, Tang CY (2007). An extension of Hill's three-component model to include different fibre types in finite element modelling of muscle, Int. J. Numer. Methods Eng., 71(7), 801–817.

Stojanovic B, Svicevic M, Kaplarevic-Malisic A, Gilbert RJ, Mijailovich SM (2020). Multi-scale striated muscle contraction model linking sarcomere length-dependent cross-bridge kinetics to macroscopic deformation, Journal of Computational Science, 39, 101062.

Stojanovic BS, Svicevic MR, Kaplarevic-Malisic AM et al. (2015). Coupling finite element and Huxley models in multiscale muscle modeling, 2015 IEEE 15th International Conference on Bioinformatics and Bioengineering (BIBE), IEEE.

van den Oord A, Dieleman S, Zen H et al. (2016). WaveNet: A Generative Model for Raw Audio, arXiv, 1609.03499.

Yan S, Zou X, Ilkhani M, Jones A (2020). An efficient multiscale surrogate modelling framework for composite materials considering progressive damage based on artificial neural networks, Composites Part B: Engineering, 194, 108014.

Yu Y, Si X, Hu C, Zhang J (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures