SOLVING THE HUXLEY EQUATION FOR ISOMETRIC MUSCLE CONTRACTION USING PHYSICS-INFORMED NEURAL NETWORK

Bogdan Milićević[1,2], Miloš Ivanović[2,3], Boban Stojanović[2,3], Nenad Filipović[1,2]

[1] Faculty of Engineering, University of Kragujevac, Sestre Janjić 6, 34000 Kragujevac, Serbia
e-mail: bogdan.milicevic@uni.kg.ac.rs, fica@kg.ac.rs
[2] Bioengineering Research and Development Center (BioIRC), Prvoslava Stojanovića 6, 34000 Kragujevac, Serbia
[3] Faculty of Science, University of Kragujevac, Radoja Domanovića 12, 34000 Kragujevac, Serbia
e-mail: mivanovic@kg.ac.rs, boban.stojanovic@pmf.kg.ac.rs

Abstract:

The Huxley muscle model is a biophysical muscle model based on the sliding filament and cross-bridge theory. The myosin filaments of muscle fibers slide past the actin filaments during muscle contraction. The actin and myosin form a protein complex when the myosin head is attached to the actin filament. Each of the attached heads contributes to the total force generated within the muscle fiber. Huxley's muscle equation, used to describe the distribution of attached myosin heads to the actin-binding sites, is usually solved by using the method of characteristics. Once this equation is solved we can calculate the generated force as well as the stiffness in muscle fibers which can be further used in finite element analysis to perform simulations on macro-level. In our paper, we present the alternative method for solving this partial differential equation and the goal is to accelerate multi-scale simulations. We simplified the equation so that only the isometric contraction is modeled in order to prove that the presented concept has the potential to replace the numerical methods in the solving the partial differential equation. We achieved similar distributions of attached myosin heads with the method of characteristics and physics-informed neural networks, and consequently, we also got similar stresses with the two methods.

Keywords: Huxley muscle model, physics-informed neural networks, numerical solving of partial differential equations, multi-scale modeling

1. Introduction

Physics-informed neural networks (PINNs) are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations [1]. These neural networks form a new class of data-efficient universal function approximators that naturally encode any underlying physical laws as prior information [1]. The major innovation with PINN is the introduction of a residual network that encodes the governing physics equations.It then takes the output of a deep-learning network, which is called the surrogate, and calculates a residual value [2]. The residual of the differential equation is minimized by training the neural network. PINNs calculate differential operators on graphs by using automatic differentiation.

The basic formulation of the PINN training does not require labeled data, results from other simulations or experimental data, and furthermore, it is unsupervised. PINNs only require the evaluation of the residual function. Providing simulation data or experimental data which is later used to train the network in a supervised manner is also possible and necessary in some

cases, especially with inverse problems. The supervised approach is often used for solving ill-defined problems when, for instance, we lack boundary conditions or an Equation of State to close a system of equations. Once a PINN is trained, the inference from the trained PINN can be used to replace traditional numerical solvers in scientific computing [2]. PINNs are a gridless method because any point in the domain can be taken as input without requiring the definition of a mesh. Moreover, the trained PINN network can be used for predicting the values on simulation grids of different resolutions without being retrained [2]. PINNs can also be used for time-dependent problems. Since time is represented as any other variable, it's possible to predict the outputat the specified time without solving the previous time steps. For these reasons, the computational cost does not scale with the number of grid points like many of the traditional computational methods. PINN has been used for predicting the solutions for the Burgers' equation, the Navier–Stokes equations, and the Schrodinger equation [3]. In this study, we focused on the basic PINNs and on solving the PDE without relying on other simulations that assist the training process. We solved the modified Huxley equationusing PINN, so as to acquire the distribution of attached myosin heads to the actin binding sites.

2. Methods

Huxley took into account the dynamics of the filaments within muscle and the probability of establishing connections (cross-bridges) of myosin heads to actin filaments inside sarcomeres [4]. The $n(x,t)$ function describes the rate of connections between myosin heads and actin filaments, as thefunction of position of the nearest available actin binding site relative to equilibrium position of myosin head x:

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

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 muscle activation given as a function of time. The attachment and detachment rate functions are shown in Figure1 along with the schematic of the actin and myosin filaments. The partial differential equation (1) can be solved by using the method of characteristics with the initial condition *n(x,0)=0*. We only took into account the isometric contraction, so we set the velocity to zero to test the possibilities of the physics informed neural network and thusly solve the equation (1).

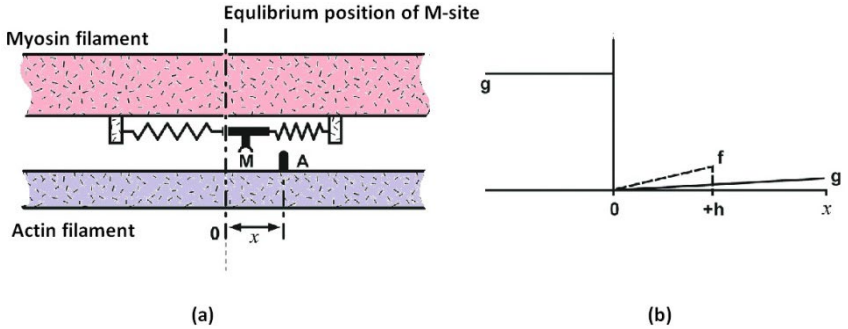

Fig. 1. Actin and myosin filaments (a), attachment f and detachment g rate functions (b).

Once the $n(x,t)$ values are acquired, we can calculate generated force F within the muscle fiber and thestiffness K using the equations:

$$F(t) = k\sum_{-\infty}^{\infty} n(x,t)x\ dx \quad \text{and} \quad K(t) = k\sum_{-\infty}^{\infty} n(x,t)\ dx \qquad (2)$$

where k is the stiffness of cross-bridges. Stress and stress derivative can be calculated as:

$$\sigma_m = F\frac{\sigma_{iso}}{F_{iso}} \quad \text{and} \quad \frac{\partial \sigma_m}{\partial e} = \lambda L_0 K\frac{\sigma_{iso}}{F_{iso}}, \qquad (3)$$

where F_{iso} is the maximal force achieved during isometric conditions, σ_{iso} shows the maximal stress achieved during isometric conditions, L_0 *the* initial length of sarcomere and λ

is called stretch. The Calculated stresses and stress derivatives can be further used on a macro-level during finite element analysis. To implement PINN and incorporate the equation (1), we used SciANN [6], a high-level artificial neural networks API, written in Python using the Keras and TensorFlow backends. SciANN is designed to abstract neural network construction and for scientific computations and solutions as well as the discovery of partial differential equations (PDE) using the physics-informed neural networks [5].

3. Results and discussion

By Using the SciANN framework, we constructed a neural network with 8 layers, each layer containing 20 neurons with a hyperbolic tangent activation function. Since we are interested in the force generated during isometric contraction, our network takes only two input values x and t, and predicts the n value. The network is trained by minimizing the residual derived from equation (1) and by providing initial conditions to solve the equation. We used the Adam optimizer with the learning rate of 10^{-4} and batch size of 512, with a total number of 15000 epochs. We also used the neural tangent kernel (NTK) method to get the adaptive weights, while balancing between the number of collocation points, used to minimize the residual of PDE, and the number of points used to minimize the residual of the initial condition. We generated a data grid consisting of 17 values for x in the range of $-20.8\,nm \leq x \leq 62.4\,nm$ and 200 values for t in the range $0s \leq t \leq 2.0s$. The generated points were used to train the network. Once the network training was done, we compared the distributions which we acquired from the PINN and from the method of characteristics (Figure2) Here, we used the same division along the *x*, which was used during the training process.

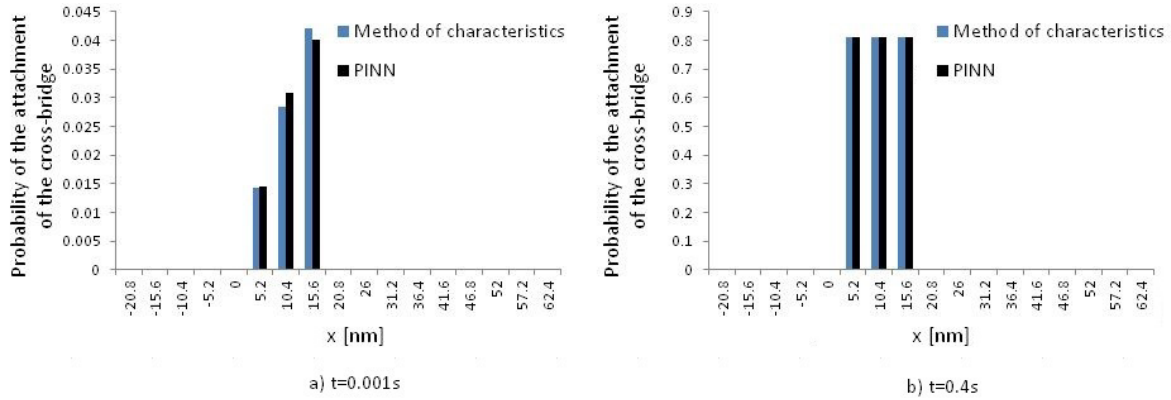

Fig. 2. Distributions of attached cross-bridges at t=0.001s (a) and t=0.4s(b).

The distributions provided by PINN and the method of characteristics are very similar. We also built the trained PINN into a finite element solver in order to test the accuracy of acquired stresses and stress derivatives. Acquired stresses and stress derivatives are shown in Fig. 3. Once again, we see that the results provided by PINN and the numerical method are very similar. It's important to note that this time, the range and the division along the *x*is different from the one we used during training, which indicates that the trained PINN generalizes properly.

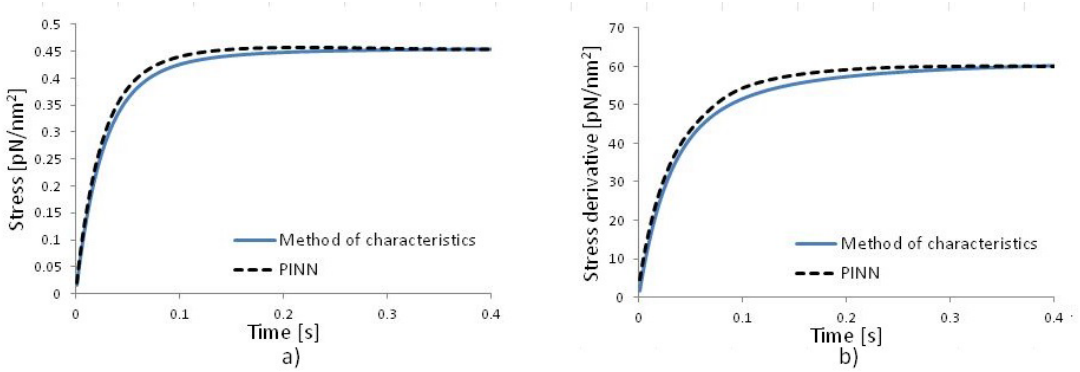

Fig. 3. Stress a) and stress derivative b) calculated during isometric contraction by the method of characteristics and by PINN

4. Conclusions

Based on the similarities between the results achieved with PINN and the method of characteristics, we can conclude that PINNs can be used to solve the Huxley equation for the distribution of attached myosin heads to actin binding sites in an isometric case. Our future research will include isotonic contractions with non-zero velocities of sliding filaments, and also, the correction of probabilities of attachment will be modified according to Gordon's stress-stretch function. In the future, we all also be integrating PINN into the finite element analysis where finite elements are used at a macro-level, while at a micro-level, PINN will be used to solve Huxley's muscle equation,and thusprovide the distribution of attached myosin heads and consequently measurethe stresses and stress derivatives that can be used on a macro-level.

Acknowledgement: 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' view. The European Commission is not responsible for any use that may be made of the information the article contains. This work was supported by the Serbian Ministry of Education, Science and Technological Development (Agreement No. 451-03-68/2022-14/200107 and 451-03-9/2022-14/200122).

References

[1] Raissi M., Perdikaris P., and Karniadakis G. E., Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations. New York City, NY: arXiv preprint arXiv:1711.10561. 2017a .

[2] Markidis Stefano, The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers, Frontiers in Big Data, Vol. 4, ISSN 2624-909X 2021. https://doi.org/10.3389/fdata.2021.669097

[3] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 686–707. 2019.

[4] Williams, W.O., Huxley's Model of Muscle Contraction with Compliance. Journal of Elasticity, 105, 365-380. 2011.

[5] Haghighat E. and Juanes R., Sciann: A Keras Wrapper for Scientific Computations and Physics-Informed Deep Learning Using Artificial Neural Networks. New York City, NY: arXiv preprint arXiv:2005.08803. 2020.