Journal of Computational Science 79 (2024) 102291

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

OpenDiHu: An efficient and scalable framework for biophysical simulations of the neuromuscular system

Benjamin Maier [a], Dominik Göddeke [b,1], Felix Huber [a,b,1,*], Thomas Klotz [c,1], Oliver Röhrle [c,1], Miriam Schulte [a,1]

[a] Institute for Parallel and Distributed Systems, Universitätsstraße 38, 70569 Stuttgart, Germany
[b] Institute of Applied Analysis and Numerical Simulation, Allmandring 5b, 70569 Stuttgart, Germany
[c] Institute for Modelling and Simulation of Biomechanical Systems, Pfaffenwaldring 5a, 70569 Stuttgart, Germany

ARTICLE INFO

Dataset link: https://github.com/opendihu/opendihu, https://zenodo.org/record/4705982

MSC:
65Y05
65Y20
65Z05
92C05

Keywords:
Neuromuscular system
Multi-scale multi-physics simulations
High-performance computing

ABSTRACT

The versatile neuromuscular system, consisting of skeletal muscles and the nervous system, enables human to perform crucial everyday tasks. To investigate its functioning and dysfunctioning with computer simulations, highly resolved, multi-scale models are favorable, whose numerical solutions demand for high performance computing. We present *OpenDiHu*, a versatile, high-performance computing, open source software framework for detailed, systemic simulations of skeletal muscles and their recruitment mechanisms. *OpenDiHu* allows to solve a variety of multi-scale models, including 3D muscle mechanics, measurable electromyographic signals, action potential propagation in the muscle tissue, subcellular bio-chemo-electrical processes, and the neural drive to the muscle. All these components can be combined with a wide range of numerical solution schemes into comprehensive simulation setups for the entire system. Experiments on up to almost 27 000 cores demonstrate the efficiency and parallel scalability of *OpenDiHu*. This enables in silico experiments at very high spatial and temporal resolutions.

1. Introduction

In this paper, we present the computational framework *OpenDiHu* that allows to solve a variety of multi-scale models describing the neuromuscular system and which can predict in vivo measurable data such as electromyographic signals.

1.1. Motivation

The neuromuscular system must satisfy versatile requirements, encompassing fast, enduring, finely controlled or forceful actions, thus enabling humans to perform a variety of vital tasks in everyday life. Neuromuscular physiology relies on the interplay of complex processes on several scales, ranging from the macroscopic electrical and mechanical properties on the tissue scale, down to the cellular behavior of the muscle fibers and the motor neurons in the spinal cord. Skeletal muscles are different from other muscles such as the heart or the stomach as they have a distinct functional architecture based on muscle fibers, which is optimized for voluntary motions. This must be taken into account on all scales in specialized models, e.g., for electrophysical processes as well as for the generation of forces and the mechanical muscle deformation.

Detailed, systemic simulations of the neuromuscular system are thus a highly beneficial yet computationally challenging task: They can foster the physiological understanding by providing an in silico laboratory to complement limited in vivo studies. Ultimately, they can help to diagnose and treat neuromuscular disorders, possibly in a patient-specific way. Simulations for state-of-the-art multi-scale models of the musculoskeletal system imply high temporal and spatial resolution, translating to high computational requirements. Consequently, efficient parallelization and tailored numerical schemes are mandatory to exploit the computational power of medium-sized compute clusters and supercomputers.

OpenDiHu (short for "digital human") implements a broad set of model components available in the literature, which can be combined into a comprehensive simulation setup comprising the neural drive to the muscle, action potential propagation in the muscle tissue, muscle contraction and mechanical coupling to tendons as well as other tissues such as fat. Furthermore, (non-invasively) measurable electromyographic (EMG) signals in the muscle and the body

* Corresponding author at: Institute of Applied Analysis and Numerical Simulation, Allmandring 5b, 70569 Stuttgart, Germany.
E-mail address: felix.huber@mathematik.uni-stuttgart.de (F. Huber).
[1] Authors in alphabetical order.

https://doi.org/10.1016/j.jocs.2024.102291
Received 2 December 2022; Received in revised form 23 March 2024; Accepted 9 April 2024
Available online 16 April 2024

fat can be simulated. Considering the electro-physiological behavior of skeletal muscles, currently two main model variants are available: The *fiber-based modeling approach* represents muscle fibers explicitly by one-dimensional (1D) structures and the respective reaction diffusion equations for the action potentials; the *multi-domain approach* represents muscle fibers implicitly through anisotropies in the three-dimensional (3D) homogenized model equations and the possibility to assign proportional 'contributions' of multiple activated motor units.

A detailed comparison of the model variants from the perspective of possible applications is beyond the scope of this paper. We show, however, that *OpenDiHu* constitutes a capable tool for such an endeavor. Instead, our main focus is to provide an efficient, scalable, versatile and open source software framework that offers sufficient modularity and flexibility in the choice of model components and solvers. Accordingly, we present representative simulation scenarios as well as more detailed performance and scalability results. The main design goals of *OpenDiHu* are:

Usability. Domain scientists can easily configure models and model parameters, as well as parameters controlling the numerical simulation scheme, to enable parameter studies. In addition to a command-line mode, *OpenDiHu* supports 'sand-box' scripting, which enables users to define their application and the respective parametrization of models and solvers in an intuitive and compact way. Various input and output data formats allow for a flexible integration into the pre- and postprocessing pipeline.

Performance. The framework and all individual solvers can be run in parallel on distributed memory systems, ranging from laptops and workstations over medium-sized compute clusters to supercomputers. Emphasis is also put on efficient single-core performance by use of cache-efficient data handling, avoidance of unnecessary copy operations, and utilization of instruction-level parallelism.

Extensibility. An *OpenDiHu* scenario consists of modular solvers for sub-models and model components that are orchestrated in a tree structure. The framework provides the infrastructure of data and control flow between the solvers. In this context, a solver module can be any operation on given data, e.g., a time stepping or operator splitting scheme to solve ordinary differential equations (ODEs), a linear or nonlinear solver, or an adapter to a *CellML* model. *CellML* is a standard used in the bioengineering community to define ODE-based models on the (sub)cellular level [1]. It defines a machine-readable format to represent differential–algebraic systems of equations. It is possible to create new solvers for additional model components and combine them with the existing ones.

1.2. Related work

Several other simulation environments for the simulation of muscle stimulation and contraction have been developed throughout recent years. We here mention only those that have particular similarities to *OpenDiHu*, without postulating completeness. *OpenCMISS* [2] is a general purpose open source solver framework with focus on biomechanics, with model flexibility as its main feature. It includes the computational core *OpenCMISS Iron* and related visualization software *OpenCMISS Zinc*. However, this framework does not aim at or achieve optimal computational efficiency and suitability for high-resolution simulations. It is one of very few codes that have implemented the coupled 1D–3D approach for fiber-based skeletal muscle models similar to *OpenDiHu*. *Chaste* [3] and *openCARP* [4] are electrophysiology(-mechanics) codes for muscle simulations. They have, however, been designed with cardiac simulations in mind: The heart muscle has a different structure than skeletal muscles, such that the particular fiber-representing features, that make the skeletal muscle simulation particularly challenging, are not required. *Chaste* has also been run successfully on bigger clusters [4,5]. *LiveV* and its successor *lifex*[2] are finite element software libraries that can be run on supercomputers and are used to simulate cardiac electro-mechanics in [6]. *Alya* is a general purpose high-performance computing multi-physics code that has been used to simulate cardiac electro-mechanics on supercomputers in [7]. *FEBio* is a more general multiphysics software tool specializing in nonlinear finite element analysis for biomechanics and biophysics. It handles large deformation problems, allows for rigid-body modeling and includes a user-friendly graphical interface.

[2] https://lifex.gitlab.io

1.3. Contribution and paper outline

In this paper, we demonstrate that *OpenDiHu* is an efficient and capable tool that enables highly resolved simulations of skeletal muscles including their stimulation, force generation, mechanical deformation, and the propagation of electrical potentials (EMG). Our results show that flexible simulations of various model variants are possible without compromising computational efficiency and parallel scalability. Such simulations can capture effects on smaller spatial scales compared to low resolution simulations, e.g., effects of individual motor units on muscle activation and contraction. This clearly shows the necessity of large-scale simulations and, thus, optimized parallel performance. In representative scenarios, we can reduce runtimes by two orders of magnitude compared to previous implementations. Consequently, *OpenDiHu* enables high resolution simulations that were not feasible previously.

The paper is organized as follows: Section 2 summarizes the comprehensive multi-scale model framework describing the musculoskeletal system, which can be fully parameterized with *OpenDiHu*. Section 3 discusses the used discretization and solution schemes. Section 4 discusses aspects of the usage and the implementation of the software. Section 5 presents simulation results and evaluates performance and scalability for various scenarios.

2. The neuromuscular modeling framework

Our software package *OpenDiHu* is designed to specifically address the requirements of systemic neuromuscular in silico experiments that aim to relate the neural drive to muscle activity and the experimentally measurable motor output. In the following, we outline the corresponding coupled multi-scale and multi-physics problem. For further details we refer to [8].

2.1. Overview of the multi-scale and multi-physics model

Motion is achieved through a specialized physiological system. In summary, motor neurons in the spinal cord integrate signals from supraspinal centers and sensory organs. If a motor neuron depolarizes, an action potential travels along its axon and innervates a set of muscle fibers at the neuromuscular junctions. A motor neuron and all the muscle fibers it innervates form functional units, which are known as motor units (MU). Once a muscle fiber is stimulated, the action potential travels along the muscle fiber and triggers an intracellular signaling cascade, which ultimately activates the skeletal muscle's molecular motor. The generated microscopic active stresses are transmitted through the muscle and connective tissues to finally yield muscle motion.

The outlined physiological system can be closely replicated by a coupled multi-scale and multi-physics model which we define in the following sections. Fig. 1 shows its structure where single-headed arrows indicate a one-directional coupling between two sub-models and left–right arrows indicate a bi-directional coupling between two sub-models.

The neural drive from the brain acts as an input to the neural stimulation model (a), which causes action potential propagation (b) in the muscle. One of the two model alternatives (b1) and (b2) has to be chosen, either the multi-domain description (see Section 2.3) or a muscle fiber-based approach (see Section 2.4), respectively. Both

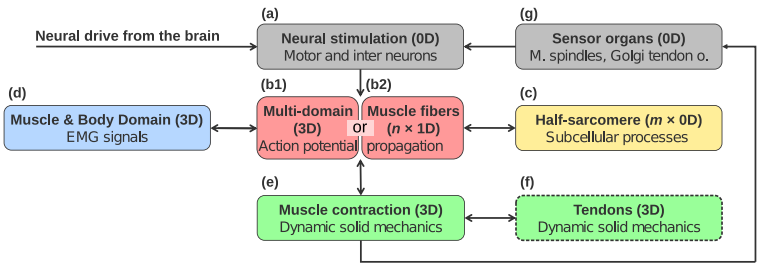

Fig. 1. Models (a) — (f) in the framework that can be linked together to form a multi-scale model: (a) is formulated in (6); (b1) is described by (3), (b2) by (4) and (5); (c) is modeled as in (6); (d) is defined by (7) and (8); (e) is given in (1) and (2); (f) models the deformation of tendons, the coupling to this model component is, however, not considered in this paper.

approaches are strongly coupled to instances of a subcellular model (c) described in Section 2.5. To simulate EMG signals on the skin surface, electric conduction in the body domain (d) can be considered (see Section 2.7). The models (b) to (d) are bidirectionally coupled to the description of muscle contraction (e) via the muscle activation and the deforming geometry as discussed in Section 2.2. In addition to the muscle contraction model, the mechanics of the tendons (f) can also be computed and numerically coupled using *OpenDiHu*. However, this part of the multi-scale model is beyond the scope of this paper. Finally, sensor organs (g) located inside the muscle sense properties of the contraction process and influence the motor neurons, forming the neural system (see Section 2.6).

2.2. Continuum mechanics model (3D)

To describe muscle contraction and deformations in the body, we use a continuum-mechanical model. Thereby, each material point $\mathcal{P}$ is parameterized by a coordinate $\mathbf{X} \in \Omega_0 \subset \mathbb{R}^3$ in the reference configuration and the coordinate $\mathbf{x} \in \Omega_\mathrm{M} \subset \mathbb{R}^3$ in the actual configuration. The deformation gradient tensor $\mathbf{F} = \partial\mathbf{x}/\partial\mathbf{X}$ represents a mapping relating referential coordinates and actual coordinates. The fundamental governing equations of the continuum mechanical problem are summarized in material description:

$$\mathrm{div}_\mathbf{X}(\mathbf{F}\,\mathbf{S}) + \mathbf{b} = \rho_0\,\dot{\mathbf{v}} \qquad \text{(balance of linear momentum)}, \tag{1a}$$

$$\mathbf{S} = \mathbf{S}^\top \qquad \text{(balance of angular momentum)}, \tag{1b}$$

$$\mathrm{div}_\mathbf{X}(\mathbf{v}) = 0 \qquad \text{(incompressibility)}, \tag{1c}$$

where the primary variables are given by the velocity vector $\mathbf{v} = \dot{\mathbf{u}}$, where $\mathbf{u}$ is the displacement vector $\mathbf{u} = \mathbf{x} - \mathbf{X}$. Further, $\mathbf{S}$ is the second Piola–Kirchhoff stress tensor, $\mathbf{b}$ is a vector of body forces and ρ_0 is the mass density. Finally, the incompressibility constraint (1c) follows from conservation of mass and assuming a constant mass density ρ_0 for all material points $\mathbf{X} \in \Omega_0$.

To solve the given equations, the second Piola–Kirchhoff stress tensor $\mathbf{S}$ needs to be defined and, within this work, is adopted from Heidlauf et al. [9]. It is given as the linear superposition of a hyperelastic, transversely-isotropic material model describing the muscle's passive material behavior and an active stress tensor reflecting the muscle's active behavior:

$$\mathbf{S} = \frac{2\partial\Psi_\mathrm{passive}(I_1, I_2, I_4)}{\partial\mathbf{C}} + \frac{1}{\lambda_\mathrm{f}} S_\mathrm{max}^\mathrm{active}\, f_{\ell}(\lambda_\mathrm{f})\,\bar{\gamma}\,\mathbf{a}_0 \otimes \mathbf{a}_0. \tag{2}$$

Here, Ψ_passive is a strain–energy function as given in [9], I_1 and I_2 are invariants of the right Cauchy–Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ and $I_4 = \mathbf{F}\mathbf{a}_0 \cdot \mathbf{F}\mathbf{a}_0$ is a mixed invariant with $\mathbf{a}_0$ denoting a referential unit vector aligned with the muscle fiber direction. Further, $S_\mathrm{max}^\mathrm{active}$ is a parameter indicating the maximum isometric muscle stress, f_{ℓ} is a function describing the muscle's isometric force-length relation, $\lambda_\mathrm{f} = \sqrt{I_4}$ is the stretch in the muscle fiber direction and $\bar{\gamma}$ is a homogenized activation value, which can be calculated from the microscopic active stresses (Section 2.5). For more details on the formulation, refer to [10,11].

2.3. Multi-domain electrophysiology model (3D)

The active stresses generated by the muscle fibers are controlled by an excitation–contraction coupling cascade. Before presenting the respective coupling in Section 2.5, we introduce two different modeling approaches for the electric muscle activation. Muscle contraction is controlled by the nervous system and the neural stimuli are propagated through the tissue via electrical signals. This can be simulated with the multi-domain model [12,13], representing a multi-scale continuum model resolving the electrophysiological properties of skeletal muscles. When assuming that all muscle fibers are electrically independent, it is possible use an alternative fiber-based approach that is presented in Section 2.4 (e.g. see [14]).

In the multi-domain model presented in this section, it is assumed that, at each skeletal muscle material point $\mathcal{P} \in \Omega_\mathrm{M}$, an extracellular space and one intracellular space for each MU coexist. Accordingly, an electrical potential is defined for each compartment, i.e., ϕ_e for the extracellular space and ϕ_i^k ($\forall k \in \{1, \dots, N_\mathrm{MU}\}$) for the intracellular spaces, where k refers to a specific MU and N_MU is the number of MUs. Further, a transmembrane potential $V_\mathrm{m}^k = \phi_\mathrm{i}^k - \phi_\mathrm{e}$ can be introduced for each MU. The domains are coupled by taking into account key features of skeletal muscle tissue's microscopic composition, the shape of the muscle fibers and the behavior of the fiber membranes. Those assumptions yield a coupled system of PDEs for the potentials V_m^k on the deforming muscle domain Ω_M, which is summarized by

$$\mathrm{div}\left[\left(\boldsymbol{\sigma}_\mathrm{e} + \sum_{k=1}^{N_\mathrm{MU}} f_\mathrm{r}^k \boldsymbol{\sigma}_\mathrm{i}^k\right)\mathrm{grad}\,\phi_\mathrm{e}\right] + \sum_{k=1}^{N_\mathrm{MU}} f_\mathrm{r}^k\,\mathrm{div}\left[\boldsymbol{\sigma}_\mathrm{i}^k\,\mathrm{grad}\,V_\mathrm{m}^k\right] = 0\,, \tag{3a}$$

$$\frac{\partial V_\mathrm{m}^k}{\partial t} = \frac{1}{C_\mathrm{m}^k A_\mathrm{m}^k}\left(\mathrm{div}\left[\boldsymbol{\sigma}_\mathrm{i}^k\,\mathrm{grad}(V_\mathrm{m}^k + \phi_\mathrm{e})\right] - A_\mathrm{m}^k I_\mathrm{ion}^k(\mathbf{y}^k, V_\mathrm{m}^k)\right), \quad k = 1, \dots, N_\mathrm{MU}\,. \tag{3b}$$

Here, $\boldsymbol{\sigma}_\mathrm{i}^k$ and $\boldsymbol{\sigma}_\mathrm{e}$ are the conductivity tensors in the intracellular domains and the extracellular domain, respectively. Further, $f_\mathrm{r}^k(\mathbf{x}) \in [0,1]$ with $\sum_k f_\mathrm{r}^k(\mathbf{x}) = 1$ is a microstructural field considering the (spatial) MU distribution, A_m^k is the surface-to-volume ratio for a muscle fiber associated with MU k and C_m^k is the specific capacitance of the fiber membranes associated with MU k. The current I_ion^k over the membrane is predicted from an electrical circuit model and is a function of the transmembrane voltage V_m^k and a vector of (membrane) state variables $\mathbf{y}^k$ on the subcellular scale. The vector $\mathbf{y}^k$ resolves the entire excitation–contraction coupling pathway addressed separately in Section 2.5.

2.4. Electrophysiology model with muscle fibers (1D)

An alternative to the multi-domain approach is to model the propagation of the action potentials on n 1D muscle fiber domains Ω_f^j that are

embedded in the deforming muscle domain Ω_M using the monodomain equation [11,14]

$$\frac{\partial V_\mathrm{m}}{\partial t} = \frac{1}{A_\mathrm{m}^j C_\mathrm{m}^j}\left(\sigma_\mathrm{eff}\frac{\partial^2 V_\mathrm{m}}{\partial s^2} - A_\mathrm{m}^j I_\mathrm{ion}(V_\mathrm{m}, \mathbf{y}^j)\right), \quad \forall j = 1,\dots,n, \quad \text{on} \quad \Omega_\mathrm{f}^j \subset \Omega_\mathrm{M}. \tag{4}$$

Analogous to (3b), A_m^j and C_m^j are the surface-to-volume ratio and the electrical capacitance of the membrane of fiber j, respectively. Further, V_m is the transmembrane voltage, σ_eff is the effective conductivity, s is the spatial coordinate along a fiber path, and I_ion is the current over the membrane, which is formulated in Section 2.5.

Following the key assumption of the bidomain model, that an intracellular space and the extracellular space coexist at each material point, the extracellular potential ϕ_e in the deforming 3D muscle domain Ω_M can be obtained from

$$\mathrm{div}\big((\boldsymbol{\sigma}_\mathrm{e} + \boldsymbol{\sigma}_\mathrm{i})\ \mathrm{grad}(\phi_\mathrm{e})\big) + \mathrm{div}\big(\boldsymbol{\sigma}_\mathrm{i}\ \mathrm{grad}(V_\mathrm{m})\big) = 0, \quad \text{on} \quad \Omega_\mathrm{M}. \tag{5}$$

As before, $\boldsymbol{\sigma}_\mathrm{i}$ and $\boldsymbol{\sigma}_\mathrm{e}$ are the intra and extracellular conductivity tensors and ϕ_e is the extracellular potential. (5) is unidirectionally coupled to (4) by the transmembrane voltage V_m. Note that this requires a mapping of the transmembrane potential V_m from the 1D muscle fiber domains Ω_f^j to the 3D muscle domain Ω_M.

2.5. Subcellular model (0D)

The mechanical behavior and the electrical behavior of the muscle are linked by a subcellular model of the excitation–contraction coupling. This means that the electrical stimulus given by the transmembrane potential V_m is translated into a mechanical stress response represented by the mechanical activation parameter $\gamma(\mathbf{y}) \in [0,1]$. In addition, the subcellular models describe the generation of the current I_ion^k over the membrane. In summary, such models include (i) the dynamics of the muscle fiber membrane, (ii) intracellular calcium dynamics, which serves as a second messenger system, and finally (iii) force production through cross-bridge cycling. While there exist various models that vary in the level of detail, all models yield systems of differential–algebraic equations. Accordingly, in the multi-scale modeling framework, each material point is associated with a "0D" subcellular model, which is mathematically described by

$$\frac{\partial \mathbf{y}}{\partial t} = G(V_\mathrm{m}, \mathbf{y}), \quad I_\mathrm{ion} = I_\mathrm{ion}(V_\mathrm{m}, \mathbf{y}) \quad \text{on } \Omega_\mathrm{M}. \tag{6}$$

The vector $\mathbf{y}$ summarizes all subcellular state variables that are associated with excitation–contraction coupling such as the opening probability of ion channels, the intracellular calcium concentration or the mechanical state of a sarcomere. The subcellular state vector $\mathbf{y}$ is coupled to (3b) or (4), respectively, via the transmembrane voltage V_m. Such cellular models can be conveniently stored in *CellML* format [1], an XML-based description language, and can directly be used in *OpenDiHu*. An open source model repository established by the Physiome project [15] hosts a multitude of curated *CellML* models. Further, various tools for solving and editing such models are available. Considering skeletal muscle fibers, the work proposed by Shorten et al. [16] is a rare example for a model, which considers all aspects of excitation–contraction coupling. It consists of $\mathbf{y} \in \mathbb{R}^{56}$ state variables. Note that, if a single physics problem should be solved, a subcellular model focusing on a single aspect is sufficient. For example, only the membrane dynamics are required when simulating EMG signals with the multi-domain model. Fig. 2 shows an overview on how *OpenDiHu* uses external *CellML* models which are automatically compiled and optimized for different backends during the simulation setup. For more details on the *CellML* adapter we refer to [17].

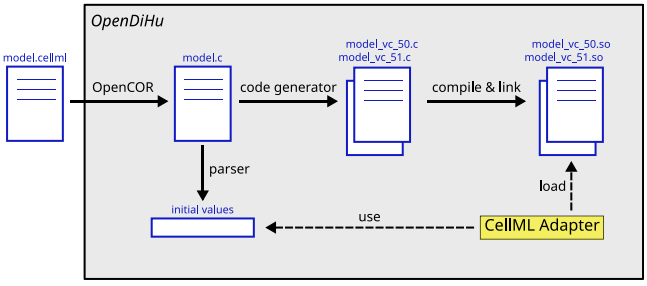

Fig. 2. Overview on using *CellML* models in *OpenDiHu*. *CellML* model files are automatically converted to C source files using *OpenCOR*. The resulting code is parsed and optimized for efficient execution (e.g., for vectorized execution of 50 and 51 cells as shown in the figure) and compiled into a shared library that is loaded by the *CellML* adapter.

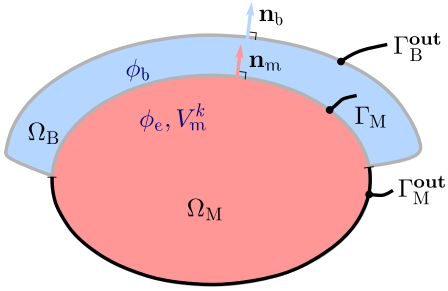

Fig. 3. Computational domains of the multi-domain model. The muscle domain, Ω_M, is adjacent to the body domain Ω_B, the shared border is Γ_M with normal vector $\mathbf{n}_\mathrm{m}$. The outer boundary is composed of $\Gamma_\mathrm{B}^\mathrm{out}$ and $\Gamma_\mathrm{M}^\mathrm{out}$ and has the outward normal vector $\mathbf{n}_\mathrm{b}$. The extracellular electric potential ϕ_e and the membrane voltage V_m^k for every MU $k \in \{1,\dots,N_\mathrm{MU}\}$ are defined in Ω_M, the electric potential ϕ_b of the body domain is defined in Ω_B.

2.6. Neural system models (0D)

Motion is controlled by the neural system in a feedback loop between brain activity and the muscle's mechanical state. Within the presented modeling framework, each muscle is innervated by a corresponding motor neuron pool model. The motor neurons are modeled through electrical circuit models [8,18,19]. Those models yield systems of ordinary differential equations generating action potentials V_m at the connection points of motor units in the muscle. They can integrate different sources of input, in particular an externally prescribed neural drive reflecting the control signals from the brain, but also afferent feedback, which contains information on the mechanical state of the muscle modeled in Section 2.2.

We exemplarily showcase such an approach by simulating afferent feedback of the muscle spindles with a model of Mileusnic et al. [20]: The muscle spindle models sense the local stretch, velocity and acceleration at a set of muscle material points, which are specified as the spindle's locations. We use respective *CellML* descriptions to load the evolution equations for both the motor neuron models and the muscle spindle models into the proposed simulation framework.

2.7. Electric conduction in electrically inactive tissue (3D)

The electrical activity of muscles can be observed non-invasively by means of surface EMG and, thus, is a frequently used window into neuromuscular physiology. Predicting surface EMG signals requires that the electrophysiological muscle models described in Sections 2.3 and 2.4 are coupled to surrounding electrically inactive tissue, e.g., a layer of fat and skin tissue on top of the muscle, which is illustrated in Fig. 3. To do so, we assume a harmonic electric potential ϕ_b in the

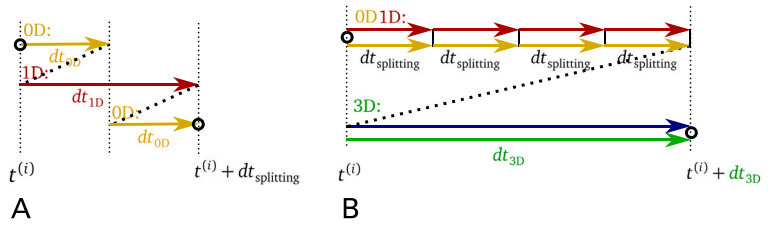

Fig. 4. Staggered approach to solve the components of the multi-scale model with the colors matching the schematic in Fig. 1, assuming the description with muscle fibers ((b2) in Fig. 1) is used. A: Operator splitting approach to solve the subcellular model (6) with time step width dt_{0D} (0D, yellow arrows) and the monodomain equation (4) with time step width dt_{1D} (1D, red arrows). B: Subcycling scheme to solve all model components. The pairs of red and yellow arrows each correspond to one instance of the operator splitting in A. After several time steps, the electric conduction models (blue arrow, (5) and (7)) and the solid mechanics model (green arrow, (1)) get solved before the scheme repeats.

body domain Ω_B (cf. e.g., [12,14]), yielding

$$\operatorname{div}\left(\boldsymbol{\sigma}_b \operatorname{grad}(\phi_b)\right) = 0 \quad \text{on } \Omega_B \,. \tag{7}$$

Herein, $\boldsymbol{\sigma}_b$ denotes the conductivity tensor in the body region. The muscle domain Ω_M and the body domain Ω_B are coupled at the common border Γ_M via interface conditions. That is, the extracellular potential ϕ_e and the electrical potential in the body region ϕ_e are continuous at the boundary and the current leaving the muscle through the extracellular space must enter the body region, yielding

$$\phi_e = \phi_b \qquad \text{on } \Gamma_M, \tag{8a}$$

$$(\boldsymbol{\sigma}_e \nabla \phi_e) \cdot \mathbf{n}_m = -(\boldsymbol{\sigma}_b \nabla \phi_b) \cdot \mathbf{n}_m \quad \text{on } \Gamma_M. \tag{8b}$$

3. Discretization and solvers

In the following, we describe the available numerical discretization and solution techniques for the presented models. Due to the complexity of the overall model and our design goal to offer a modular simulation framework, we implement a staggered solution approach. Fig. 4 schematically shows the interplay of the model components in this staggered scheme: As shown in Fig. 4A, the combination of subcellular models as given in (6) and (i) the monodomain and bidomain equations, (4) and (5), or (ii) the multi-domain equations in (3) is based on a second order accurate time splitting scheme with time step width $dt_{splitting}$. The splitting scheme separates the diffusion term from the I_{ion} term given by the subcellular model and allows to use different time integrators for the diffusion and the subcellular term with time step widths dt_{1D} (fiber-based approach) or $dt_{multi\text{-}domain}$ (multi-domain approach) and dt_{0D}, respectively.

Fig. 4B shows the encompassing subcycling scheme that is used to solve all parts of the multi-scale model. After several splitting time steps, we solve one step with time step width dt_{3D} of the electric conduction models in the muscle and body domains, given by (5) and (7), respectively, and one step of the solid mechanics model, given by (1). While Fig. 4B correctly visualizes the scheme only for the fiber-based approach, the difference for the multi-domain approach lies in the time when the model for electric volume conduction (blue arrow) is solved. In the latter approach, the electric conduction and EMG model are computed together with the multi-domain equations. Instead of the blue arrow in Fig. 4B, the respective model is part of the red arrows in every splitting scheme.

The mechanical deformation of the muscle as given in (1) is solved with the respective active stress as input in a second step. This can be done in two ways: The first variant is a frozen state approach neglecting the time dependency of the deformation, thus neglecting inertia. The second variant executes a time step for the transient equations.

In terms of spatial representation, Fig. 5 shows the interplay between the 1D muscle fibers carrying material points with the respective ODEs for the subcellular state vector and the three-dimensional muscle domain for a simplified cuboid muscle: Fig. 5A visualizes the corresponding computational domains, consisting of a three-dimensional (3D) domain of muscle tissue (green color) and embedded one-dimensional (1D) muscle fibers (red color). Models (b1), (d), (e) and (f) in Fig. 1 are formulated on the 3D domain, model (b2) on the 1D domains, and separate instances of model (c) are given for spatial points on the 1D fibers. The contraction of the 3D muscle domain also deforms the 1D domains, as shown in Fig. 5B. In the following, we describe the discretization of the single model components in more detail.

3.1. Discretization and solution of the solid mechanics model

For the solid mechanics model (1) to (2), we use a mixed $(\mathbf{u}, \mathbf{v}, p)$-formulation, where the hydrostatic pressure p acts as the Lagrange multiplier that enforces the incompressibility constraint. We discretize the model in space by Taylor-Hood finite elements, i.e., quadratic ansatz functions for $\mathbf{u}$ and $\mathbf{v}$ and linear ansatz functions for p on 3D hexahedral elements in a deformed regular mesh in Ω_M. We use the implicit Euler method for the time discretization and solve the resulting non-linear system of equations in every time step by a Newton scheme.

To speed up the computation, the initial guess of the vector of unknowns in every time step is linearly extrapolated from the two previous time steps. There is no guarantee that this always saves iterations, but in cases where it does, the benefit of saving only a single Newton iteration including many inner linear solver iterations is large, whereas the cost of the extrapolation is negligible.

3.2. Discretization of the electrophysiology models

For the three-dimensional parts of the electrophysiology models, we discretize the muscle domain Ω_M and the body domain Ω_B, again with the finite element method and hexahedral elements. Usually, we use a higher resolution than for the mechanical model, i.e., the nodes of the mechanical mesh are a subset of the nodes of the electrophysiology mesh. Nodes on the internal boundary Γ_M are shared between the meshes of Ω_M and Ω_B. For the unknowns ϕ_i^k, ϕ_e, ϕ_b and V_m^k in (3a), (3b) and (5), we use tri-linear ansatz functions.

The subcellular points Ω_s^i in the multi-domain model (3b) are located at the nodes of the muscle domain mesh in Ω_M.

In the fiber-based model (4), the 1D meshes for the fiber domains Ω_f^j are embedded in the muscle domain Ω_M and possibly deformed by the mechanics model together with Ω_M, as shown in Fig. 5A and B. In the discretization, the nodes of the meshes for Ω_f^j and Ω_M do not need to coincide. In our experiments, the fibers typically use a finer spatial resolution than the surrounding muscle domain. Along each fiber Ω_f^j, the transmembrane voltage V_m is discretized with 1D linear ansatz functions. The nodes of these 1D meshes are also the locations of the internal states $\mathbf{y}$ of the subcellular model. To solve the bidomain equation (5) for ϕ_e, we map the values of V_m at the nodes of the fibers to the nodes of the three-dimensional mesh of the body domain Ω_M using tri-linear interpolation.

The diffusion steps of (4), (5) or (3) in the time splitting scheme are integrated by an implicit Euler scheme or a Crank–Nicolson scheme with a time step width dt_{1D}.

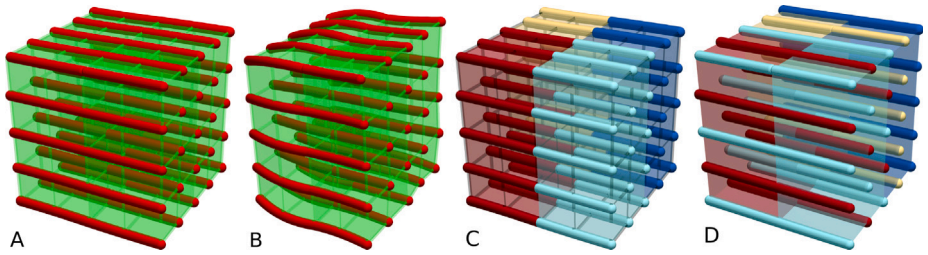

Fig. 5. The computational domain used with our multi-scale models as explained in Section 3.2. A: Exemplary cube consisting of a 3D domain (green color) and embedded 1D muscle fibers (red color). B: Deformed cube. C: Parallel domain decomposition of the 3D domain into four subdomains (different colors). D: Domain decomposition for the 1D fibers. Note, a cube is shown here, while in our simulations, we use the geometry of a biceps brachii muscle obtained from medical imaging.

For the multi-domain model (3), we get a linear system of equations that we solve in every time step using a GMRES solver with the parallel incomplete LU factorization preconditioner Euclid [21] from the HYPRE package [22]. As the formulation of this linear system following from a finite element discretization has not yet been published, we provide details in Appendix A.

In the fiber-based model, we solve the linear systems with Thomas' algorithm, because the one-dimensional monodomain equation (4) leads to tri-diagonal matrices for the discretized diffusion terms. In our case, Thomas' algorithm can be used in its original sequential form even for a parallel simulation run as we can parallelize between fibers instead of within fibers given that we typically have thousands of fibers in a single muscle. The system of ordinary differential equations arising from the subcellular term is solved with Heun's method with time step width dt_{0D}. Because of the nature of this coupled non-linear system, we use an explicit time stepping scheme instead of an implicit one, which turned out to be more efficient.

3.3. Parallelization

We parallelize the computation by a domain decomposition approach where the spaces of the 1D and 3D domains are partitioned identically along axis-aligned planar cuts. Fig. 5C shows a parallel partitioning with four subdomains for four processes.

To exploit the tri-diagonal structure of the 1D diffusion equations in the fiber-based approach, i.e., to be able to apply the sequential Thomas algorithm in each fiber, we temporarily redistribute all values for V_m, such that, for each fiber, all values are available on a single compute node. This partitioning is shown in Fig. 5D. The linear systems are then computed in parallel without any additional communication, before the solution is sent back to their original MPI rank as shown in Fig. 5C. This proved to be more scalable and efficient than resorting to parallel tri-diagonal solvers such as block-Gauss–Seidel solvers with Thomas' algorithm in each block. For the latter, we would need to iterate at least a few times with communication between blocks after each iteration and an imbalanced number of iterations between different fibers, which would lead to load imbalance.

3.4. Assignment of motor unit territories

As described in Section 2.1, the activation of muscle fibers occurs in groups within MUs such that all fibers innervated by the same motor neuron are always stimulated simultaneously. In the following, we present our algorithm to associate the simulated fibers in the muscle domain with a given number of MUs. Histochemical methods reveal that the fibers of each particular MU are located in proximity within subregions of the muscular cross-section [23]. The number of fibers per MU is distributed exponentially [24].

Similar to, e.g., [25,26], we model each MU territory by defining a center point in the muscle cross-section area from which the occupancy of the particular MU decreases in radial direction. The center points for all MUs are quasi-randomly chosen by a 2D Weyl low-discrepancy sequence, which produces equidistributed coordinate values in the interval $[0, 1)$ [27].

If the multi-domain model from (3) is used, the spatially varying occupancy factors f_r^k of compartment k are chosen as a radial basis function (RBF) p_k around the MU center point $\mathbf{x}_k$:

$$f_r^k(\mathbf{x}) = p_k(\mathbf{x}) = \frac{c}{1 + a\,\|\mathbf{x}_k - \mathbf{x}\|_2^2}, \quad \text{with } a = \frac{\pi^2}{4\sigma^4}$$

By choosing varying values for the standard deviation σ, we account for the different MU territory sizes, which are exponentially distributed over the MUs. The constant factor c is chosen such that the sum over all f_r^k is nowhere greater than one, i.e.,

$$\max_{\mathbf{x}\in\Omega_M} \sum_{k=1}^{N_{MU}} f_r^k \leq 1.$$

If the fiber-based description from (4) is used, the RBFs can be interpreted as probability density functions and the assignment of MUs to fibers is the realization of a stochastic experiment. However, simply drawing MUs with the given probability for every fiber does not result in exponentially distributed numbers of fibers per MU. Following the model proposed by Enoka et al. [24], the desired portion of fibers that should be in MU k is given as $q(k) := b^k / \sum_\ell b^\ell$ with basis $b = R^{1/N_{MU}}$ and constant ratio R between the number of fibers of the largest and smallest MUs. To enforce this condition, we include scaling factors $\lambda_k > 0$ in the RBF,

$$\tilde{p}_k(\mathbf{x}; \{\lambda_k\}_{1\ldots N_{MU}}) = p_k(\mathbf{x}) \cdot \frac{\lambda_k}{\sum_{\ell=1}^{N_{MU}} p_\ell(\mathbf{x}) \cdot \lambda_\ell},$$

and determine the factors as minimizer of the following objective function F, which penalizes the deviation from the desired exponential progression $q(k)$:

$$F(\{\lambda_k\}_{1\ldots N_{MU}}) = \sum_{k=1}^{N_{MU}} \left(q(k) - \sum_{j=1}^{n} \tilde{p}_k(\mathbf{x}_j; \{\lambda_k\}_{1\ldots N_{MU}})/n \right)^2.$$

Because F contains two nested sums of the number of MUs and fibers, the optimization problem can become computationally expensive. Therefore, our implementation decomposes the problem into multiple smaller optimization problems, each determining a subset of the sought factors $\{\lambda_k\}_{1\ldots N_{MU}}$ and, thus, incrementally increasing the number of known factors. For details, we refer to the dissertation [17].

Fig. 6 visualizes results for associating $523^2 = 273\,529$ fibers with 100 MUs, which are realistic numbers for a human biceps brachii muscle. Fig. 6A visualizes the resulting spatial arrangements of fibers for the three exemplary MUs 4, 40 and 100. It can be seen how the algorithm places the fibers close to the prescribed MU territory center points $\mathbf{x}_k$. Fig. 6B shows the constructed exponential progression $q(k)$ in the number of fibers per MU. In this case, we get a distribution with the parameter values $b = 1.02$ and $R \approx 7.2$.

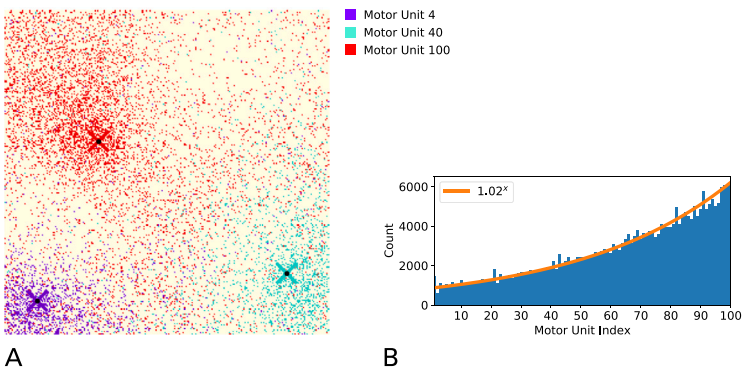

Fig. 6. Association of 273 529 muscle fibers with 100 MUs, constructed by the algorithm described in Section 3.4. A: The square corresponds to a transversal cross-section of the muscle. Each pixel represents one muscle fiber where the purple, blue and red colors denote MUs 4, 40, 100 and yellow color denotes other MUs. The colored crosses show the MU territory center points. B: Prescribed exponential progression $y = 1.02^x$ and resulting number of fibers per MU constructed by the algorithm.

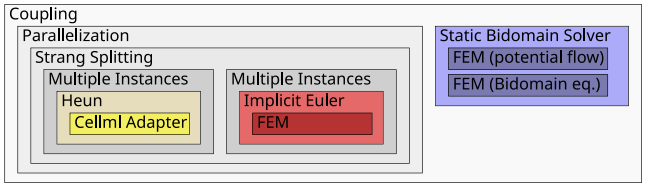

Fig. 7. Structure of an *OpenDiHu* simulation of a surface EMG signal. The colors correspond to the colors in the model overview in Fig. 1. The 0D subcellular model (6) loaded by the *CellML* adapter is integrated using Heun's method (yellow). The monodomain equation (4) along the muscle fibers is discretized in space with finite elements and integrated by an implicit Euler method (red). For each fiber, the models are coupled using a Strang splitting. The bidomain equation (5) is solved in the three-dimensional muscle domain (blue) and is coupled to the fibers.

4. Usage and implementation of *OpenDiHu*

In this section, we briefly sketch the most important features of *OpenDiHu* from a user's perspective. Readers interested in using *OpenDiHu* are referred to the exhaustive documentation[3] which also contains a range of examples. A more detailed introduction including code snippets can also be found in Sec. 6.2.3ff of [17].

4.1. Usage

New simulation scenario are set up by writing a small C++ source file that instantiates the required models, time stepping schemes and solvers. These components are instantiated in a tree structure of class templates that represent their interplay and coupling. Fig. 7 shows how such components are combined for the simulation setup in Section 5.3. The corresponding C++ source file can be found in the *OpenDiHu* repository.[4] The functionality of *OpenDiHu* is incorporated by inclusion of a header file and linkage to the core library.

In addition to the C++ source, a Python settings script has to be created where all numerical and material parameters are specified. Both the C++ and the Python source files are organized based on the same tree structure. The compiled C++ program will invoke the Python interpreter to parse the Python settings script at runtime. This allows users to calculate parameters, parse input files in custom formats, define callback functions to alter boundary conditions during the simulation, and to directly postprocess the computed values, e.g., to derive total muscle forces.

Every solver class defines *connector slots*, which expose some of the solver's scalar and vector fields representing, e.g., the computed degrees of freedom or derived values. Unidirectional and bidirectional links between the slots of multiple coupled solvers can be defined in the Python settings script and, accordingly, data are transferred between the solver schemes during the simulation. Following efficiency considerations, the computed data is copied only where necessary, otherwise the same memory locations are reused by multiple coupled solvers in sequence.

External dependencies. *OpenDiHu* uses the *Message Passing Interface (MPI)* [28] for distributed-memory parallelism. The *Portable, Extensible Toolkit for Scientific Computation (PETSc)* [29–31] is used for linear algebra, linear and non-linear system solvers and also interfaces to the *Multifrontal Massively Parallel sparse direct Solver (MUMPS)* [32] and the high performance preconditioners and solvers package *HYPRE* [22]. The explicitly data-parallel programming packages *Vc* [33,34] and *std-simd* [35] are used for instruction-level parallelism. They provide vector data structures, corresponding arithmetic operations and mathematical functions which employ the processor's SIMD instructions. The library *SEMT* [36] is used for computing symbolic derivatives used in the non-linear mechanics solver at compile time. The coupling library *preCICE* [37] can be used to couple external physics solvers, e.g., for the mechanics problem.

4.2. Implemented solvers

OpenDiHu provides a finite element solver class that allows to discretize a generalized Laplace operator in 1D, 2D and 3D using linear or quadratic Lagrange ansatz functions depending on the specific requirements of the simulated scenario. Both Dirichlet and Neumann type boundary conditions can be specified. In terms of mesh types, deformed regular meshes are used that can be easily partitioned for parallel execution [38,39] as mentioned in Section 3.2.

Specialized solvers for the multi-domain equations (3), the monodomain equation (4), and the static bidomain equation (5) exist, which reuse the basic finite element solver class to build the respective system matrices.

Multi-physics solvers can be created by combining multiple solver classes, including time stepping schemes, e.g., explicit and implicit Euler solvers, Heun's method, and the Crank–Nicolson scheme, and operator splitting schemes, e.g., Godunov splitting and Strang splitting.

A dedicated *CellML* model adapter provides easy access to existing models from the bioengineering community. The *CellML* adapter parses the model descriptions and generates efficient source code, either for the CPU employing single instruction multiple data (SIMD) instructions or for the GPU using *OpenMP* [40]. The imported *CellML* models can

[3] https://opendihu.readthedocs.io/en/latest/

[4] https://github.com/opendihu/opendihu/blob/develop/examples/electrophysiology/fibers/fibers_emg/src/fast_fibers_emg.cpp

be solved in *OpenDiHu* by combining the *CellML* adapter with a time stepping scheme.

Elasticity problems can also be computed in *OpenDiHu*, using the solver classes for quasi-static linear elasticity, quasi-static hyperelasticity and dynamic hyperelasticity. The hyperelasticity solvers support compressible and incompressible materials as well as isotropic and anisotropic materials. A tailored muscle contraction solver adds an active stress term to the hyperelastic formulations and allows us to solve the models described in Section 2.2. The material model in the hyperelasticity solvers can be customized at compile time by specifying the strain energy function in coupled or decoupled form in terms of the five (reduced) strain invariants or directly in terms of the right Cauchy–Green tensor. The Jacobian matrix used by the non-linear Newton solver is automatically derived from the custom material model at compile time [41].

In addition, it is possible to prescribe values of scalar and vector fields over time and to pick certain degrees of freedom from a mesh and copy their values to other fields. The latter allows implementing sensory organs that sense, e.g., the fiber stretch and the contraction velocity at specified locations in the muscle domain.

For most of the solver and time stepping classes, output writers can be configured that write simulation results in the parallel file formats of *VTK* to be viewed with *ParaView* [42], of *ADIOS2* [43] to be viewed with *MegaMol* [44,45], in the *CMGUI EX file* format [46] to be viewed with *CMGUI* and *OpenCMISS Zinc* [2], and in an either human-readable or binary Python-based format for custom post-processing.

5. Simulation results and performance analysis

This section showcases the capabilities of the *OpenDiHu* framework in terms of combinations of model components and associated performance and runtimes. Although we show qualitatively plausible simulation results, we explicitly do not aim to achieve new scientific insights in muscle functioning nor do we evaluate the various model options in terms of their range of applicability, disadvantages and advantages.

Section 5.1 addresses the validation of the solvers. In Section 5.2 to Section 5.5, we present four exemplary simulation scenarios for muscle contraction and surface EMG signals, which use different subsets of the full multi-scale framework summarized in Fig. 1. Then, we analyze various computational performance aspects of our *OpenDiHu* framework, by considering specific simulation scenarios: Node level performance and scalability to HPC systems are covered in Sections 5.6 and 5.7 respectively, different solvers for the multi-domain model are compared in Section 5.8, and the spatial discretization is evaluated in Section 5.9.

The subsequent scenarios use the muscle geometry of a biceps brachii. Fig. 8A shows the anatomical setting of the biceps brachii, augmented with a rendering of tendons and bones. To improve readability, all simulation parameters for all experiments are summarized in Appendix B.

5.1. Validation of the implemented solvers

We conduct several validation studies where we compare numerical and analytic solutions and find good agreement. Thus, we verify the correctness of the basic spatial Finite Element solvers and time stepping schemes as well as *OpenDiHu*'s solid mechanics solver for incompressible nonlinear hyperelasticity. The details can be found on the validation page in *OpenDiHu*'s documentation.[5]

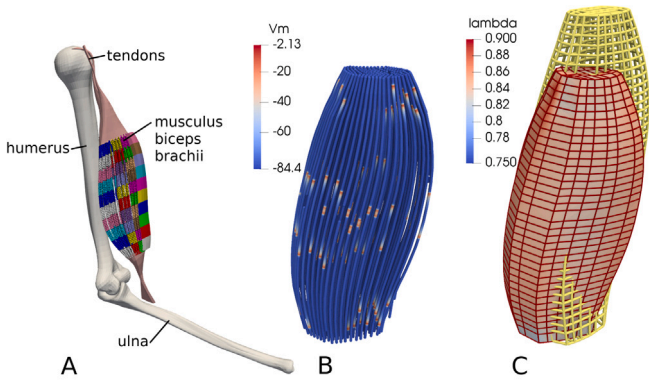

Fig. 8. A: The biceps brachii muscle with body fat layer rendered with corresponding tendons and bones, colored by an exemplary parallel partitioning. B, C: Simulation results of the fiber-based muscle contraction model, B shows the transmembrane potential V_m on 169 muscle fibers. C shows the 3D mechanics mesh in the reference configuration (yellow color) and the contracted muscle (red color), which is further colored according to the material stretch factor λ.

5.2. Simulation of muscle contraction using the fiber-based model

In the first presented scenario, we simulate muscle contraction ((1) to (2)) using the fiber-based electrophysiology model ((4) to (6)). This corresponds to models (b2), (c), and (e) in the schematic in Fig. 1.

Fig. 8B visualizes the simulated transmembrane potential V_m on the muscle fibers at simulation time $t = 2084\,\text{ms}$. Almost all fibers exhibit action potentials, which propagate from the neuromuscular junctions around the center towards the ends of each fiber. Fig. 8C shows the reference and contracted configurations of the 3D muscle domain Ω_M as yellow and red meshes, respectively. The muscle in this scenario exhibits a shortening factor λ between 75 % and 90 %.

5.3. Simulations of surface EMG using the fiber-based model

Next, we simulate EMG signals on the surface of a fat and skin layer on top of the muscle domain. We use the 1D fiber-based electrophysiology model with 3D electric conduction in the body domain ((4) to (7)) corresponding to the models (b2), (c) and (d) in the schematic in Fig. 1.

We use 961 muscle fiber domains consisting of 1D meshes with 1481 nodes each. The 3D meshes of the muscle belly and body fat contain 18 944 and 6882 nodes, respectively, and are partitioned into 128 subdomains. The fibers are organized in 20 MUs and a time span of 1 s is simulated. For reference, we state that the computation takes 8 h 38 min, using all cores of a two-socket shared-memory node comprising two AMD EPYC 7742 64-core processors with 2.25 GHz base frequency and 2×1 TiB RAM.

Fig. 9 visualizes the action potentials on the fibers at simulation time $t = 600\,\text{ms}$, the resulting electric potential ϕ_b on the skin surface at the top, and an indication of the 3D mesh on the left-hand side of the figure.

In a next experiment, we study the effect on the resulting EMG signals when using different muscle fiber counts and mesh resolutions. We solve the same model as before, but without the body fat domain. As listed in Table 1, we increase the number of muscle fibers from 1369 to 273 529 and the 3D mesh resolution from 67×10^3 to 10^8 degrees of freedom while keeping the 1D mesh resolution constant at 1481 nodes per fiber. To tackle the high computational load, we increase the number of processes to 26 912. We run these simulations with 64 processes per compute node of the supercomputer Hawk[6] at the High Performance Computing Center Stuttgart, where each compute node is

[5] https://opendihu.readthedocs.io/en/latest/user/validation.html

[6] https://www.hlrs.de/solutions/systems/hpe-apollo-hawk

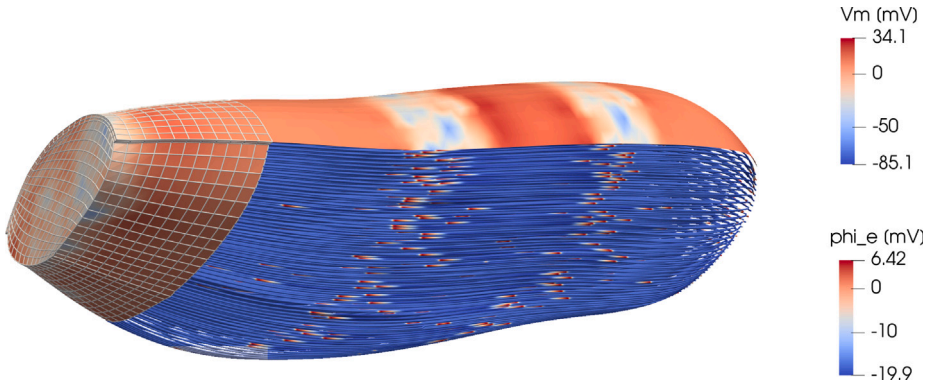

Fig. 9. Electrophysiology and surface EMG simulation with activated fibers, a body fat layer and the skin surface. The color of the fibers represents the value of the transmembrane voltage V_m, the color on the skin surface represents the electric potential ϕ_e of the extracellular space. On the left-hand side, a part of the 3D mesh that discretizes the electric volume conduction model is shown.

Table 1
Parameters of the surface EMG simulation study with different resolutions: Number of muscle fibers, degrees of freedom (DOF) in the 3D mesh of the volume conduction model, its restriction to the 2D top surface used for file output, number of DOF in all 0D subcellular models, number of employed processes, corresponding number of required compute nodes on the supercomputer Hawk, and average number of DOFs per process.

	# muscle fibers	# DOFs 3D mesh	# DOFs 2D surface mesh	# DOFs 0D models	# processes; # compute nodes	av. # DOFs per process
A	$37^2 = 1369$	67 k	19 × 186	8109 k	144; 3	57 k
B	$67^2 = 4489$	428 k	34 × 371	26 592 k	448; 7	60 k
C	$187^2 = 34\,969$	6547 k	94 × 741	207 156 k	3600; 57	59 k
D	$523^2 = 273\,529$	101 661 k	262 × 1481	1620 M	26 912; 421	64 k

identical to the system used in the previous experiment, except that we use only every second core per node because this gives the best execution time on this system.[7] Details on the parallel scaling are given in Section 5.7.

The muscle fibers are associated with 100 MUs and the association is determined using the algorithm described in Section 3.4. The association for the largest scenario is shown in Fig. 6. Fig. 10 depicts the computed surface EMG signals given by the extracellular potential ϕ_e at simulation times $t = 179.5$ ms and $t = 399.5$ ms, as indicated in the figures. It can be seen that the number of action potentials increases with a higher number of simulated fibers and the spatial extent in the signal becomes smaller for higher mesh resolutions. The simulation scenario in Fig. 10D–F contains a realistic number of muscle fibers for the biceps brachii. Fig. 10E shows that the simulated action potentials exhibit thin 'widths', which decrease with the mesh resolutions from A to D.

5.4. Simulation of surface EMG using the multi-domain model

In this section, we demonstrate the behavior of the multi-domain description of electrophysiology as an alternative to the fiber-based approach. We simulate surface EMG signals governed by (3a), (3b), (6) and (7) or models (b1), (c) and (d) in Fig. 1. The scenario simulates 25 MUs and the occupancy factors f_r^k are chosen by radial functions in a cross-section of the muscle, as visualized in Fig. 11A. Whenever a MU fires, a specific set of nodes in the muscle mesh is activated. The locations of these nodes vary in longitudinal direction of the muscle by 10 % of muscle length. Consequently, the activation of a single MU yields a characteristic wavefront of the transmembrane voltage V_m^1 propagating to the ends of the muscle, as shown in Fig. 11B. The backside of the front is smoothed as a result of electric conduction in transverse direction. The superposition of these MU action potentials weighted by the occupancy factors contributes to the EMG signal on the surface.

Fig. 11C shows the extracellular potential ϕ_e on the muscle surface and also indicates the 3D mesh width. Fig. 11D visualizes the electric potential ϕ_b on the surface of the body fat domain, corresponding to surface EMG. The comparison with Fig. 11C shows the smoothing effect of the body fat layer.

5.5. Simulation of muscle contraction with afferent feedback

In the next scenario, we simulate the reflex response of a muscle when externally applying a stretch (with ramp and hold phase as shown in Fig. 13 top row). The scenario uses a motor neuron pool model, the full chemo-electro-mechanical muscle model and a model for the afferent feedback induced by muscle spindles, i.e., models (a) to (e) in the schematic in Fig. 1. In detail, for the muscle simulation we employ the fiber-based electrophysiology model coupled with muscle contraction. Thereby, the dynamic incompressible hyperelasticity model is solved and a regularizing damping force of $\mathbf{b} = -2\mathbf{v}$ is added to reduce mechanical oscillations. Note that in vivo damping is expected due to the interaction with surrounding inactive tissues and the herein neglected viscous material properties.

The motor neuron pool consists of 10 motor neurons that stimulate the respective muscle fibers in the muscle model at their neuromuscular junctions and which are described by the model of Negro and Farina [18,19]. To replicate the biophysical behavior of the motor neuron pool, we follow the approach of [19] and vary certain model parameters between the individual motor neurons (see Appendix B.5 for details). We use an injected current as input to motor neuron models, which is computed by (cf. [19])

$$I_{MN}^k(t) = I_{common}(t) + I_{independent}^k(t) + I_{MN,spindles}(t)\ ,\ \forall\ k \in \{1, \dots, N_{MU}\}. \tag{9}$$

[7] Note that this is a highly hardware and case dependent property that is difficult to explain a priori as it depends not only on the computation versus memory access granularity of our code, but also on the details of the hardware, in particular the memory hierarchy. Thus, this is not a general recommendation for running *OpenDiHu* on high performance computing hardware.

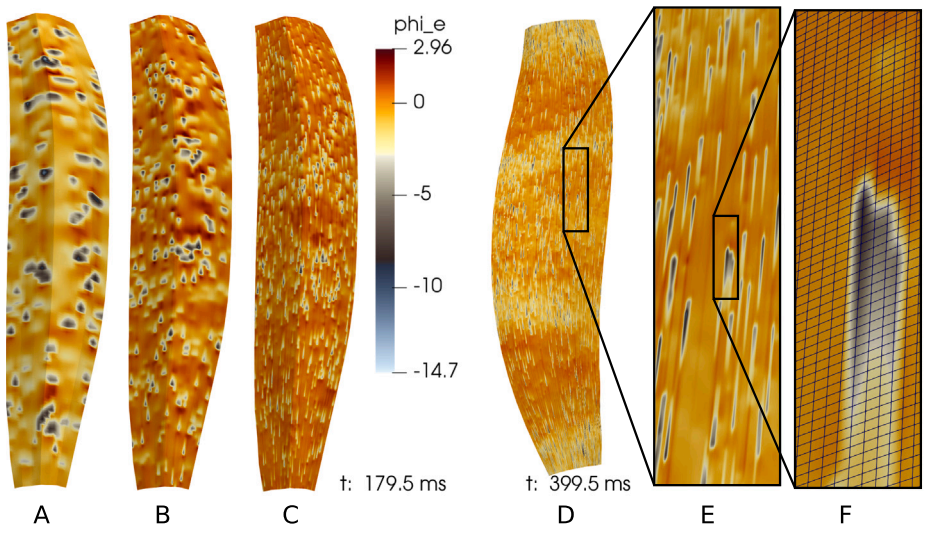

Fig. 10. A–D: Simulations of surface EMG with the different spatial resolutions given in Table 1. The color coding specifies the extracellular potential ϕ_e in millivolts on the surface of the muscle domain. E, F: Detail view of the result in D, zooming in on a single 'spike' in the signal. F also shows the mesh resolution.

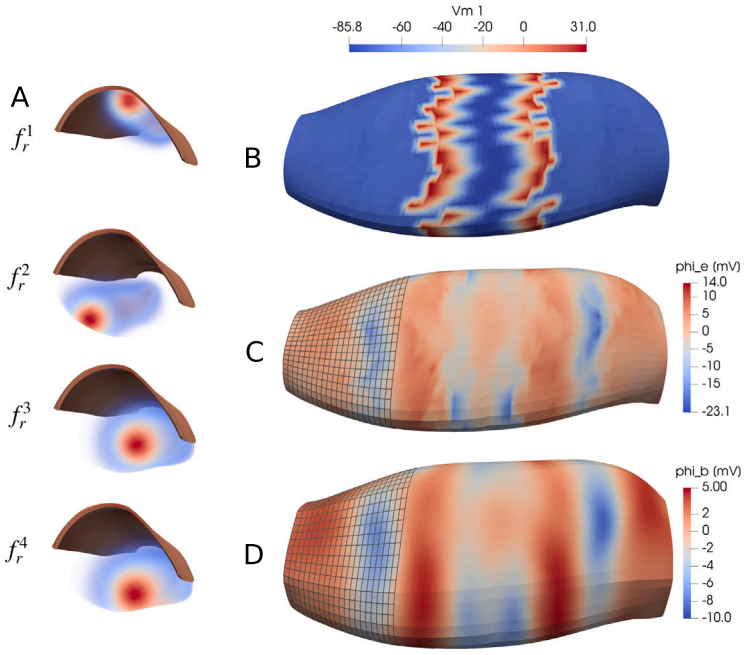

Fig. 11. Simulation of surface EMG using the multi-domain model. A: Exemplary occupancy factors f_r^k for four MUs as seen from one end of the muscle, together with the body fat domain for reference. The color transition from red to blue and transparent corresponds to the decreasing value of f_r^k according to the radial basis function. B–D: Simulation results for $t = 20\,\text{ms}$. B: Transmembrane voltage V_m^1 of the first MU. C: Extracellular potential ϕ_e on the muscle mesh. D: EMG signal ϕ_b on top of the body domain. The used mesh widths is indicated on the left sides of the figures C and D.

Here, k is the index of the motor neuron and N_{MU} is the total number of motor neurons. Further, I_{common} is the cortical drive, which is shared across all motor neurons, with a mean value of 8 nA which is superimposed, bandpass filtered white noise (15 to 35 Hz) and an amplitude corresponding to 16 % covariance with respect to the mean value. The term $I^k_{\text{independent}}(t)$ is independent for each motor neuron and is modeled as low pass filtered white noise, with a standard deviation of 25 % relative to the common noise and a low pass threshold of 100 Hz. To compute the afferent drive, which describes the feedback signal from the muscle spindles to the motor neuron, three representative muscle spindles are located inside the muscle belly at randomly chosen locations and sense the stretch along the muscle fiber direction in the tissue for the presented example. We solve an instance of the model proposed by Mileusnic et al. [20] for each muscle spindle. The primary afferents of the muscle spindles affect the firing rates of the motor neurons. We compute the post-synaptic current induced by the muscle spindles as

$$I_{\text{MN,spindles}}(t) = \frac{1}{3}\sum_{\ell=1}^{3}(s_{\text{spindle},\ell} * g_\sigma)(t - t_{\text{delay}}). \tag{10}$$

The firing frequency $s_{\text{spindle},\ell}(t)$ of each muscle spindle $\ell \in \{1,2,3\}$ gets convolved with a Gaussian kernel g_σ with standard deviation $\sigma = 2\,\text{ms}$ and which mimics the smoothing effect of the excitatory synapses. To consider the conduction velocity of the afferent nerve, the resulting signal gets delayed by $t_{\text{delay}} = 30\,\text{ms}$ and the average from all three muscle spindles contributes to the input current of all motor neuron model instances.

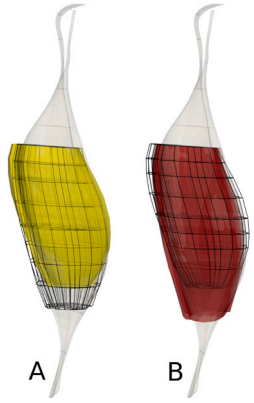

Fig. 12. Scheme to impose a prestress initial condition for the solid mechanics problem. A: The first step is to contract the muscle from the reference configuration (black wireframe) to the yellow volume. B: The second step is to use the result of the first step as new reference configuration (black wireframe) and extend the muscle by a downwards pointing muscle force to obtain the initial geometry with prestress (red volume).

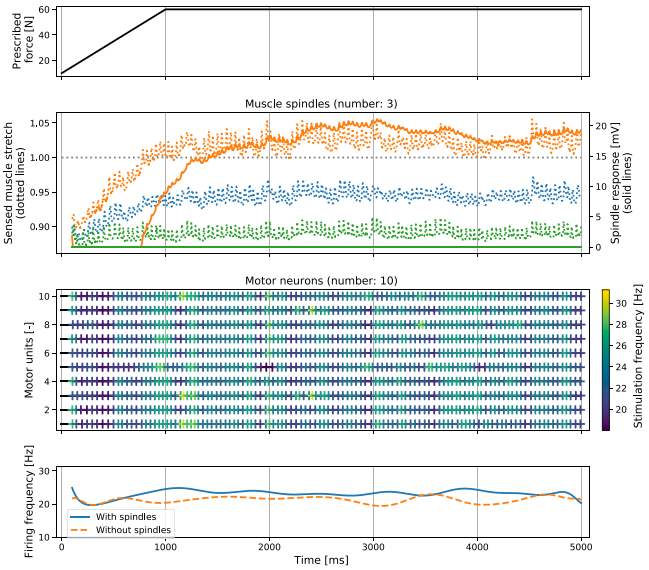

Fig. 13. Neuronal feedback in simulated eccentric muscle contraction in a 'ramp and hold' scenario with three muscle spindles, for the simulation time span 0.1 to 5 s. From top to bottom: (i) prescribed external force. (ii) Input and output of the model for the three muscle spindles (different colors), the input to the spindle model is the sensed muscle stretch λ (dotted lines, left axis), the spindle response is the primary afferent voltage level (solid lines, right axis). (iii) Firing times of the 10 MUs, colored by their frequency. (iv) Mean firing frequency of this scenario (blue solid line) and, for comparison, the same curve for an equivalent scenario without muscle spindles (dashed orange line).

Prior to solving the presented models for the simulation time span, we perform two steps to add prestress to the reference configuration of the mechanics model: The muscle is fixed at its top tendons as shown in Fig. 12. In the first step, we prescribe a constant activation level $\bar{\gamma} = 10\,\%$ in the muscle and a constant body force density of $\rho_0\, 20g$ pointing upwards, i.e., 20 times the normal body weight, but in upward direction. We solve the static contraction problem using the described incompressible material. Fig. 12A shows the resulting shortened muscle. In the second step, we set the new reference configuration of the mechanics problem to the previously shortened muscle geometry and re-extend the muscle by applying a downwards pulling force $F_{\text{prestress}}$, shown in Fig. 12B. The result of these two steps is approximately the same muscle geometry as before (due to the incompressible material), but with a constant stress throughout the domain. In this scenario, we only add a small prestress force of $F_{\text{prestress}} = 10\,\text{N}$ to obtain an initial muscle configuration that is shortened by approximately 10 %.

After the prestress has been computed, the main simulation loop iterates over solving the 0D/1D and 3D electrophysiology, the 3D solid mechanics, the muscle spindle models and the motor neuron models, potentially subcycling the solvers of each model part until the time step widths of the main simulation loop are reached.

Dirichlet boundary conditions fix the vertical displacement of the muscle at the top end. At the bottom end, the horizontal displacements are constrained to constant zero to simulate the horizontal guidance of the tendons and the tissue surrounding the muscle. Dirichlet boundary conditions at the center of the bottom plane prescribe a constant stretch that increases linearly to 1 cm in the first second. All other nodes of the bottom plane have Neumann boundary conditions imposed. A downwards pulling force is specified and affinely increases from the initial prestress value of 10 N to 60 N in the first second. After one second, the Dirichlet and Neumann boundary conditions are kept constant. This setup is also known as 'ramp and hold' scenario, e.g., [47,48].

Fig. 13 visualizes simulation results of the presented scenario for a simulation time of 5 s. The initial time span $[0\,\text{s}, 0.1\,\text{s}]$ has been trimmed from the plots, as the dynamic system exhibits non-equilibrium behavior resulting from the initial and boundary conditions. The top plot shows the prescribed external force that stretches the muscle. The second plot from the top shows the material stretch as sensed by the three muscle spindles. The values oscillate and increase during the first second following the external 'ramp' of increasing force. After one second, the curves level off at different values for the three muscle spindles at different locations in the muscle. The response $s_{\text{spindle},\ell}(t)$ of spindle ℓ is only positive for positive stretch values. In the presented scenario, only the muscle spindle corresponding to the orange curve senses an extension of $\lambda > 1$ and shows a non-zero response.

The third plot from the top in Fig. 13 shows the firing times of the 10 MUs with their frequencies visualized by the color coding. The irregular firing pattern is a result of the physiological motor neuron drive formulated in (9) and (10).

The bottom plot visualizes the mean firing frequency of all MUs, smoothly approximated by Gaussian process regression with a squared exponential kernel with length scale 800 ms. In addition to the mean firing frequency of the described scenario, the firing frequency for the same scenario without muscle spindles is included. The comparison shows that the muscle spindles increase the average firing frequency of the motor neurons.

5.6. Node level performance

In this and the following sections, we evaluate the computational performance of different parts of the software framework *OpenDiHu* by considering specific simulation scenarios. Again, all detailed simulation parameters can be found in Appendix B.

We begin with an evaluation of the performance on the compute node level, more specifically, we consider parallel execution with up to 18 processes. The computed scenario is the fiber-based electrophysiology model given by (4) to (6), i.e., models (b2) and (c) in Fig. 1 with the subcellular model of Shorten et al. [16]. The monodomain model from (4) and (6) is solved using a Strang operator splitting with the Crank–Nicolson method for the diffusion term and Heun's methods for the subcellular model. A conjugate gradient solver with no preconditioning is used for the linear system stemming from the bidomain equation (5).

We simulate a time span of $t_{\text{end}} = 2\,\text{ms}$. During this time, the resulting values are written to output files 20 times after every 0.1 ms. The fibers are assigned to 10 MUs, which are activated by a ramp signal every 0.2 ms from $t = 0\,\text{ms}$ to $t = 1.8\,\text{ms}$.

At first, we carry out a strong scaling study of this scenario, i.e., the same computation is performed with different numbers of processes from one to 18. To assess the performance of *OpenDiHu*, we exemplarily choose *OpenCMISS Iron*, and simulate the same scenario in both software packages by removing the non-linear mechanics loop from *OpenCMISS* to enable comparability. We measure the total user time of the simulation program, including time for initialization, assembly of system matrices and file I/O. This experiment is run on an Intel Core i9-10980XE machine with 3.00 GHz base frequency, 18 physical cores, L3, L2 and L1 cache sizes of 24.8 MiB, 18 MiB and 576 KiB respectively, and 31 GiB main memory.

Within *OpenDiHu*, we evaluate the following two variants: The first variant performs exactly the same computations as in *OpenCMISS Iron*. The second variant only solves the equations on muscle fibers that have been stimulated. Additionally, it only computes the subcellular model instances that are not in equilibrium. The subcellular model instances that are not in equilibrium are determined by checking whether the relative change of all subcellular states is below the threshold $\epsilon = 10^{-5}$. Thus, the second *OpenDiHu* variant arrives at (approximately) the same numerical solution as the first variant, while carrying out less calculations. In the considered scenario, only half of the subcellular models has to be computed due to the ramp activation pattern.

Fig. 14A shows the resulting runtimes of this study. It can be seen that the runtime decreases monotonically with the number of processes for all three tested codes/variants. The reduction in runtime between *OpenCMISS Iron* and the first *OpenDiHu* variant reaches an impressive factor of about 100 with a maximum value of 186 for 4 processes. In addition, the second *OpenDiHu* variant approximately halves the runtime further, as expected. The main reasons for this speedup are that *OpenDiHu* fully exploits vectorization by using AVX-512 SIMD instructions (note the inclusion of the *Vc* and *std-simd* packages as discussed in Section 4.1), that *OpenDiHu* avoids data copy as much as possible, and uses the optimal, linear-complexity Thomas algorithm for solving the diffusion part of (4).

To further investigate the computational behavior, we present measurements of the solvers in a roofline model [49] in Fig. 14B. The memory bandwidths of the caches and main memory and the peak performance were measured using the Empirical Roofline Tool [50]. Hardware counters were used to count floating point instructions and memory operations. In this study, we focus on performance during the computations and exclude the initialization phase of the simulation, which involves large portions of I/O and, thus, low FLOPs numbers. Accordingly, for the runs with *OpenCMISS Iron* and *OpenDiHu*, counters were started 90 s, respectively 15 s after the beginning of the simulation to account for the typical runtimes of initialization in the two software packages.

The resulting curves of the three tested variants are all to the right of the 'memory wall' bound, which means that the simulation is compute-bound. The highest computational intensity is achieved by the *OpenDiHu* variant that computes the model on all fibers. The performance in terms of GFlop/s increases with higher parallelism because more of the computational resources of the processor are used. The highest performance value is 180.2 GFlop/s for the first *OpenDiHu* variant and all 18 processes. This corresponds to 25.2 % of the peak performance, which we consider impressive since the run still contains I/O phases and is, in contrast to other roofline examinations, not focusing on single compute kernels.

5.7. Large scale parallel weak scaling

Next, we study the parallel weak scaling on the supercomputer Hawk at the High Performance Computing Center Stuttgart, see Section 5.3 for its specifications. We simulate the same fiber-based electrophysiology model as in Section 5.6 with the subcellular model of Hodgkin and Huxley [51], and again use 64 processes per compute node. The number of fibers and the number of processes is varied while their ratio is kept approximately constant. For this study, we simulate a time span of $t_{\text{end}} = 2$ ms, corresponding to two invocations of the 3D bidomain model solver.

Fig. 15 presents the resulting runtimes of the initialization phase and the solvers of the 0D, 1D and 3D parts of the model. To proportion the initialization runtime in a realistic scenario, the solver runtimes are multiplied by the factor 500 to match an end time of 1 s and the initialization time remains unchanged. This is done because full simulations at scale are wasteful for performance studies rather than production runs and physiologically meaningful scenarios typically consider time spans in the range of seconds.

It can be seen that the runtimes for the initialization, which includes loading input files and computing system matrices, are relatively negligible. The solvers for the 0D and 1D models show perfect weak scaling behavior, as both the 0D computations and the 1D computations on different fibers are independent of each other, the latter due to our 'trick' to reduce on-node communication, which is described in Section 3.2. More details on this algorithm can be found in [52].

The 3D problem is solved with an unpreconditioned conjugate gradient solver using the implementation of PETSc and a relative tolerance on the residual norm of 10^{-5}. The number of iterations increases from 72 for 18 ranks to approximately 3×10^3 for 26 912 ranks, corresponding to an increase in 3D DOF from approximately 2×10^3 to approximately 2×10^6 (see Appendix B.7 for details). As a result, also the runtime of the 3D solver increases and dominates the computation for the scenarios with more than 7000 fibers. However, the total runtime for the largest scenario with a realistic number of 273 529 fibers in this muscle is still feasible. Larger scenarios are currently not required from an anatomical point of view. In spite of this, further optimizing the numerical and parallel scaling of the three-dimensional solver is work in progress.

5.8. Solvers for the multi-domain model

In the next study, we investigate the parallel weak scaling behavior for the multi-domain model with fat domain, i.e., (3a), (3b), (6) and (7) or models (b1), (c) and (d) in Fig. 1. The goal is to evaluate different preconditioners for the solution of the resulting linear system of equations when solving the multi-domain equation with an implicit Euler scheme.

We simulate five different scenarios with between 4 and 12 MUs. Correspondingly, we partition the 3D meshes of muscle and body fat domains with between 24 and 72 processes, yielding an approximately constant problem size per utilized process.

We solve the linear system of the multi-domain equations in every scenario by the same GMRES solver, as the system matrix is non-symmetric, but we vary the choice of preconditioners. We use the GMRES implementation of *PETSc*. The stopping thresholds on the relative and absolute residual norms are set to 10^{-15}, the specified maximum number of GMRES iterations is 10^4, and the restart parameter is set to 30 iterations.

Fig. 16 shows the number of GMRES iterations in the upper plot and the total runtime for the preconditioner and solver in the lower plot. Apart from varying the preconditioner, we also investigate two variants for every preconditioner: The preconditioning is either computed using the non-symmetric system matrix (solid lines in Fig. 16) or using a symmetric matrix that is obtained by only considering the diagonal blocks of the system matrix (dashed lines in Fig. 16). The specific blocks of the matrix used in the preconditioner are described in Appendix A.2.

The reference measurement is given by using no preconditioner. It can be seen that the maximum number of 10^4 iterations is reached, which means that the specified tolerance is not achieved by the GMRES solver alone.

We evaluate three different preconditioners. The first and second preconditioners are block Jacobi schemes, which iteratively approximate the solution by solving smaller linear systems given by the block

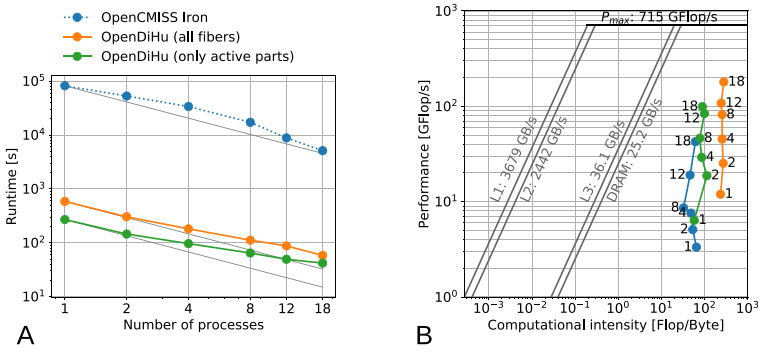

Fig. 14. Parallel strong scaling study of the fiber-based electrophysiology model on a single node. A: Runtime comparison of the reference software *OpenCMISS Iron* and two variants of our implementation in *OpenDiHu*, theoretical perfect scaling is indicated by the inclined thin gray lines. B: Roofline model to evaluate performance characteristics, numbers in the plot indicate used cores.

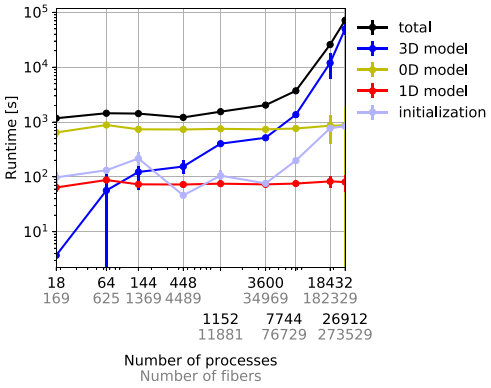

Fig. 15. Parallel weak scaling of the fiber-based electrophysiology model on the supercomputer Hawk, simulating up to more than 270 000 muscle fibers. The total runtime and the runtimes of the initialization phase and the 0D, 1D and 3D parts of the model are shown. The vertical lines represent the standard deviation of the measurements across all processes.

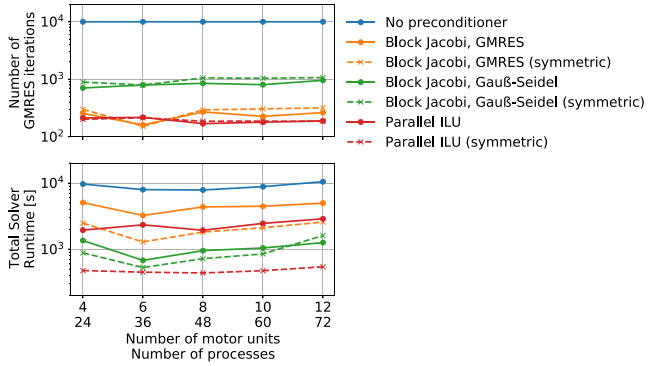

Fig. 16. Evaluation of different preconditioners and weak parallel scaling for the multi-domain electrophysiology model. The different line colors correspond to different choices for the preconditioner used together with a GMRES solver in PETSc with restart after 30 iterations. The upper plot shows the required (outer) iterations in the GMRES solver for the preconditioned system. The lower plot shows the total runtimes of preconditioner and solver combinations. Dashed lines correspond to the usage of a block-diagonal system matrix for preconditioning instead of the original matrix. The measurements are carried out for different problem sizes and numbers of processes in a weak scaling setting.

diagonals of the system matrix and the respective parts of the right-hand sides. To solve the smaller linear systems, we investigate another GMRES solver with restart after 30 iterations and alternatively a Gauß-Seidel scheme. For both iterative solvers, a relative tolerance of 10^{-5} is set as the stopping criterion. A third preconditioner is applied to the full system, and employs the *'Euclid'* algorithm from the *HYPRE* package [21], a parallel implementation of an incomplete LU factorization (ILU(k)) with factorization level $k = 1$.

A comparison of the two block Jacobi schemes shows that the preconditioner variants using the GMRES solver requires less (outer) iterations in the solver than the variant with the Gauß-Seidel scheme. However, the total combined runtime is larger for the GMRES variant, i.e., the smaller number of iterations in the outer GMRES solver does not outweigh the increased compute time in the preconditioner. Furthermore, the total runtime for the block Jacobi schemes is lower with a symmetric preconditioner matrix than if the original system matrix is used.

The number of GMRES iterations for the parallel ILU preconditioner are low, comparable to the block Jacobi with GMRES scheme. However, we expect these solvers to lose their advantage with higher core counts or number of DOF [53]. For the symmetric matrix, the total runtime is the lowest of all evaluated solver-preconditioner combinations. As a result, we use the Euclid preconditioner with a symmetric preconditioning matrix in our computations of the multi-domain model.

The lower plot of Fig. 16 also shows the parallel weak scaling of the preconditioners and the GMRES solver. Because the number of unknowns is not exactly proportional to the number of processes, the runtime slightly decreases from 24 to 36 processes. For more processes and MUs, the runtime using the parallel ILU preconditioner stays approximately constant and only slightly increases for the last data point with 72 processes. Thus, we conclude a good weak scaling behavior of the considered multi-domain computations.

5.9. Evaluation of spatial discretization in the fiber-based model

Next, we evaluate the effect of the number of muscle fibers and the 3D EMG mesh resolution on the overall simulation result. We use the fiber-based electrophysiology models (b2), (c) and (d) in Fig. 1 with the subcellular model of Hodgkin and Huxley [51]. In our study, we vary the total number of fibers between 7^2 and 97^2. We hierarchically refine the scenarios with finer fiber distributions from the baseline scenario with 7^2 fibers to ensure that the different grids correspond to the same geometry. The 3D structured meshes are derived from the 1D fiber meshes and consist of a subset of the 1D nodes. To quantify the effect of the discretization on the solution for the electric potential ϕ_e, we compute the differences between each solution and the reference solution obtained from the finest fiber distribution with 97^2 fibers.

We perform two different studies. In the first study, shown in Fig. 17A, only the subset of fibers corresponding to the coarse grid of 7^2 fibers gets activated in line with the respective MUs, while additional fibers on finer grids never get activated. Fig. 17A visualizes this setup. Fig. 17B and C plot the difference between the computed extracellular potential ϕ_e and a reference solution obtained with the

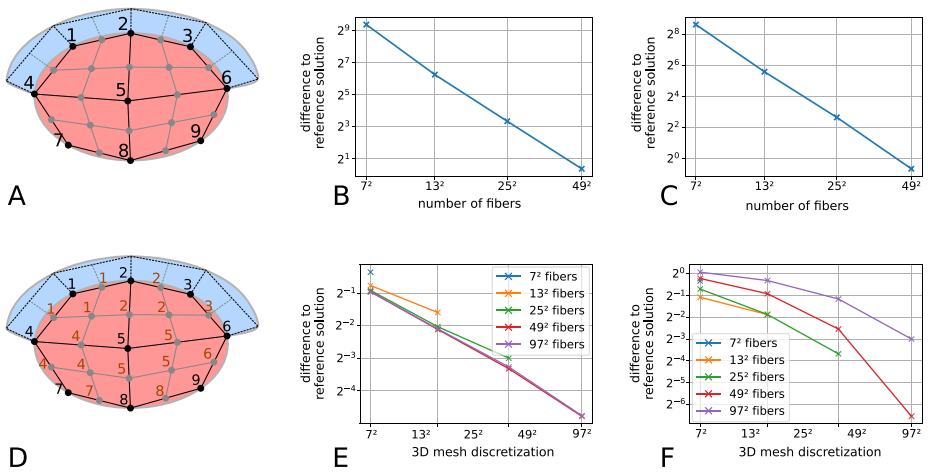

Fig. 17. Relative L_2 difference of the computed extracellular potential ϕ_e for $t = 80\,\text{ms}$ in the muscle domain Ω_M (B, E) and on the skin surface Γ_B^{out} (C, F) for different mesh widths. A–C: Fibers that are not present on the coarse 7^2 fiber grid are not activated. A: Discretization and subset of activated fibers (black numbers). The black colored mesh corresponds to the coarsest discretization, the gray lines specify the refined grid. B: Difference of ϕ_e in the muscle domain Ω_M. C: Difference of ϕ_e on the skin surface Γ_B^{out}. D–F: Fibers on finer grids (red numbers) are activated according to the MUs of the neighboring fibers on the coarse grid (black numbers). The numbers specify the MUs. E: Difference of ϕ_e in the muscle domain Ω_M. F: Difference of ϕ_e on skin surface Γ_B^{out}.

finest discretization of 97^2 fibers. The progression of the difference is shown for increasingly fine discretizations and given as the L_2 norms over the muscle domain Ω_M in Fig. 17B and over the body domain Ω_B in Fig. 17C.

We observe that the solutions on coarse grids differ significantly from solutions computed on finer grids. The difference to the reference solution reduces for finer grids, but is still on a high level even for the finest grid. A reason for this is that, on a coarser mesh, the activation of each fiber influences a larger region of the 3D mesh, which leads to a significant over-estimation of the resulting electrical signal.

In the second study, we instead activate all fibers. The additional fibers on finer grids are associated with the same MUs as the neighboring fibers on the coarse 7^2 grid, as illustrated in Fig. 17D. Fig. 17E and F show the difference of ϕ_e to the reference solution in the muscle domain Ω_M and body domain Ω_B, respectively. Here, the results show much smaller differences because the activated region in the 3D mesh remains similar for every MU. The combination of coarser 3D discretizations with finer fiber distributions shows that the overall difference mostly depends on the discretization width of the 3D mesh, as a finer fiber distribution only adds few new data.

These results show the importance of the 3D discretization and how simulation results, such as the surface EMG signal, are influenced by the 3D discretization. Furthermore, different assumptions on the activation of individual muscle fibers affect the discretization quality: Small individual activations are not well represented on coarser grids and lead to large differences, while larger scaled activations permit the use of a coarser 3D mesh to reduce computational resources.

6. Conclusion

We have presented the open-source framework *OpenDiHu*, which enables detailed and comprehensive skeletal muscle mechanics simulations. Computational efficiency and parallel scalability allow us to use compute-intense, detailed physical models with a high spatial and temporal resolution.

The paper showed *OpenDiHu*'s versatility in terms of models and solvers. The latter refers to both individual components of the entire multi-scale neuromuscular system for (single) skeletal muscles, and to their combination into comprehensive systemic simulation setups. Combined with the ability to predict measurable bio-signals like (non-invasive) surface electromyography, this paves the way towards an in silico laboratory that allows to explore physiological hypotheses as well as enables the use of digital twins for personalized healthcare [8].

For example, simulations can be used to generate benchmark datasets for decoding algorithms [54], optimize the design of experimental measurement systems [55], optimize the design of prosthetic devices [56] or to reduce training time of myoelectrically controlled prostheses [57]. Moreover, inverse modeling and uncertainty quantification [58] allow to augment experimentally measured data and hence enable applications like functional imaging [59,60] or the estimation of the neural drive to the muscle [61,62]. For instance, we plan to integrate the results from [58] into this framework.

The *OpenDiHu* framework itself deserves some more work and improvements. In particular, it is worth investigating if the 3D mechanics and EMG conjugate gradient or the multi-domain GMRES solvers can benefit from more capable multigrid or domain decomposition preconditioners, to counteract the worsening scalability implied by the increasing number of iterations of the currently employed schemes.

CRediT authorship contribution statement

Benjamin Maier: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Dominik Göddeke:** Funding acquisition, Project administration, Resources, Supervision, Writing – original draft, Writing – review & editing. **Felix Huber:** Writing – original draft, Data curation, Investigation, Methodology, Project administration, Software, Visualization, Writing – review & editing. **Thomas Klotz:** Formal analysis, Methodology, Project administration, Writing – original draft. **Oliver Röhrle:** Funding acquisition, Project administration, Supervision. **Miriam Schulte:** Resources, Supervision, Writing – original draft, Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Software and required data is freely available at https://github.com/opendihu/opendihu and https://zenodo.org/record/4705982.

Acknowledgments

Funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC2075–390740016 and under the International Research Training Group 'Soft Tissue Robotics' GRK 2198/1–277536708. Furthermore, we acknowledge the support by the Stuttgart Center for Simulation Science (SimTech) and the High Performance Computing Center Stuttgart (HLRS) providing compute resources and support (project 'DiHu', reference 'Acid 4410').

Appendix A. Finite element formulation of the multi-domain model with muscle and body domain

A.1. Governing system of equations

The finite element formulation of the multi-domain model including electric conduction in the body domain from (3) and (7) yields the following linear system of equations, which has to be solved in every time step of the Strang splitting, in alternation to the subcellular model from (6):

$$\begin{bmatrix} \mathbf{A}^1_{V_\mathrm{m},V_\mathrm{m}} & & & \mathbf{B}^1_{V_\mathrm{m},\phi_\mathrm{e}} & \\ & \ddots & & \vdots & \\ & & \mathbf{A}^{N_\mathrm{MU}}_{V_\mathrm{m},V_\mathrm{m}} & \mathbf{B}^{N_\mathrm{MU}}_{V_\mathrm{m},\phi_\mathrm{e}} & \\ \mathbf{B}^1_{\phi_\mathrm{e},V_\mathrm{m}} & \cdots & \mathbf{B}^{N_\mathrm{MU}}_{\phi_\mathrm{e},V_\mathrm{m}} & \mathbf{B}_{\phi_\mathrm{e},\phi_\mathrm{e}} & \mathbf{D} \\ \hline & & & \mathbf{E} & \mathbf{C}_{\phi_\mathrm{b},\phi_\mathrm{b}} \end{bmatrix} \begin{bmatrix} V_m^{1,(i+1)} \\ \vdots \\ V_m^{N_\mathrm{MU},(i+1)} \\ \hline \boldsymbol{\varphi}_e^{(i+1)} \\ \hline \hat{\boldsymbol{\varphi}}_b^{(i+1)} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{1,(i+1)}_{V_\mathrm{m}} \\ \vdots \\ \mathbf{b}^{N_\mathrm{MU},(i+1)}_{V_\mathrm{m}} \\ \mathbf{0} \\ \hline \mathbf{0} \end{bmatrix}. \tag{A.1}$$

The vector of unknowns to be solved for contains the transmembrane voltages $V_\mathrm{m}^{k,(i+1)}$ for MU $k \in \{1,\dots,N_\mathrm{MU}\}$, the extracellular potential $\boldsymbol{\varphi}_\mathrm{e}^{(i+1)}$, and the electric potential $\hat{\boldsymbol{\varphi}}_\mathrm{b}^{(i+1)}$ in the body domain, which represents the EMG signal on the skin surface. The quantities with super-script (i) refer to the previous time step, the quantities with super-script $(i+1)$ are the solution for the next time step after time step width dt.

The system matrix is sparse and contains blocks of non-zero entries (Note that the all-zero blocks are not explicitly shown in (A.1)). The non-zero blocks are given by the following matrices and vectors:

$$\mathbf{A}^k_{V_\mathrm{m},V_\mathrm{m}} = \frac{1}{A^k_\mathrm{m}\, C^k_\mathrm{m}}\theta\, \mathbf{K}_{\sigma_\mathrm{i}^k} - \frac{1}{dt}\mathbf{M},$$

$$\mathbf{B}^k_{V_\mathrm{m},\phi_\mathrm{e}} = \frac{1}{A^k_\mathrm{m}\, C^k_\mathrm{m}}\theta\, \mathbf{K}_{\sigma_\mathrm{i}^k}, \quad \mathbf{B}^k_{\phi_\mathrm{e},V_\mathrm{m}} = f^k_\mathrm{r}\, \mathbf{K}_{\sigma_\mathrm{i}^k}, \quad \mathbf{B}_{\phi_\mathrm{e},\phi_\mathrm{e}} = \mathbf{K}_{\sigma_\mathrm{e}+\sigma_\mathrm{i}},$$

$$[\mathbf{B}^k_{\Gamma_\mathrm{B}}]_{ab} = \int_{\Gamma_\mathrm{M}} \varphi_b\, \varphi_a \,\mathrm{d}\mathbf{x}, \quad \mathbf{C}_{\phi_\mathrm{b},\phi_\mathrm{b}} = \mathbf{K}_{\sigma_\mathrm{b}}.$$

Herein, $\mathbf{K}$ and $\mathbf{M}$ designate the canonical finite element stiffness and mass matrices using ansatz functions φ_a,

$$[\mathbf{K}_\sigma]_{ab} = -\int_\Omega \sigma\, \nabla\varphi_b \cdot \nabla\varphi_a \,\mathrm{d}\mathbf{x}, \quad [\mathbf{M}]_{ab} = \int_\Omega \varphi_b\, \varphi_a \,\mathrm{d}\mathbf{x}.$$

The variable A^k_m denotes the fibers' surface-to-volume ratio in MU k, the variable C^k_m is the specific capacitance of the fiber membranes associated with MU k. The factor f^k_r describes the local volume fraction that is occupied by MU k.

The matrices $\mathbf{D}$ and $\mathbf{E}$ contain the coupling terms on Γ_M in (8a) and (8b). The right-hand side contains the entries

$$\mathbf{b}^{k,(i+1)}_{V_\mathrm{m}} = \left((\theta-1)\,\frac{1}{A^k_\mathrm{m}\, C^k_\mathrm{m}}\mathbf{K}_{\sigma_\mathrm{i}^k} - \frac{1}{dt}\mathbf{M}\right)\mathbf{V}^{k,(i)}_\mathrm{m} + (\theta-1)\,\frac{1}{A^k_\mathrm{m}\, C^k_\mathrm{m}}\,\mathbf{K}_{\sigma_\mathrm{i}^k}\,\boldsymbol{\varphi}^{(i)}_\mathrm{e}$$

for $k \in \{1,\dots,N_\mathrm{MU}\}$. The Crank–Nicolson parameter is chosen as $\theta = 1$ to get the first-order consistent implicit Euler scheme or as $\theta = \frac{1}{2}$ to get the second-order consistent canonical Crank–Nicolson scheme.

The size of the system matrix in (A.1) depends on the number N_MU of compartments. The first N_MU rows originate from the second multi-domain equation, (3b), the next row originates from the first multi-domain equation, (3a) and the last row from the equation in the body domain, (7). The vector $\hat{\boldsymbol{\varphi}}_\mathrm{b}^{(i+1)}$ for which we solve in (A.1) only contains the degrees of freedom in the body domain, Ω_B, without the border Γ_M, as those are already contained in $\boldsymbol{\varphi}_\mathrm{e}^{(i+1)}$.

A.2. Details on the block-Jacobi preconditioner

For the "symmetric" variant of the block-Jacobi preconditioner investigated in Section 5.8, we consider the symmetric system matrix blocks $\mathbf{A}^1_{V_m,V_m},\dots,\mathbf{A}^{N_\mathrm{MU}}_{V_m,V_m}$, $\mathbf{B}_{\phi_e,\phi_e}$, $\mathbf{C}_{\phi_b,\phi_b}$, $\mathbf{D}$, and $\mathbf{E}$. For the non-symmetric variant, we consider all non-zero blocks (additionally $\mathbf{B}^i_{V_m,\phi_e}$ and $\mathbf{B}^i_{\phi_e,V_m}$ for $i = 1,\dots,N_\mathrm{MU}$). Out of these system matrix entries, the block Jacobi preconditioner only uses the matrix entries where the row and column degrees of freedom both reside on the same processor's subdomain, such that no communication is required in the block-Jacobi preconditioner step.

Appendix B. Detailed simulation parameters

In this section, we provide all physical and numerical parameters that are used in the presented simulations in Section 5. Appendix B.1 lists parameters that are common to all simulations, Appendices B.2 to B.9 describe the specific simulation scenarios of Sections 5.2 to 5.9.

B.1. Parameters common to all simulations

All simulations use geometric data of the biceps brachii muscle, which we obtain from the Visible Human Project®[8] [63]. In all computations, we used the following standard physical properties, unless otherwise noted. Note that we express length and time dependent values using the physical units of [cm] and [ms] to keep the numbers in a value range that can be properly represented by the data types in the program.

C^j_m [μF cm^{-2}]	σ_eff [mS cm^{-1}]	$\sigma_\mathrm{i}^{\parallel}$ [mS cm^{-1}]	$\sigma_\mathrm{i}^{\perp}$ [mS cm^{-1}]	σ_e [mS cm^{-1}]	σ_b [mS cm^{-1}]
0.58 (slow-twitch), 1 (fast-twitch)	3.828	8.93	0.893	6.7	0.4

The value for the electric membrane capacitance C^j_m is set to 0.58 μF cm^{-2} or 1 μF cm^{-2}, for slow-twitch and fast-twitch fibers, respectively [16]. The conductivity tensors $\boldsymbol{\sigma}_\mathrm{e}$ and $\boldsymbol{\sigma}_\mathrm{b}$ in the extracellular and body regions, respectively, are assumed to be isotropic. The conductivity tensor $\boldsymbol{\sigma}_\mathrm{i}$ in the intracellular region is assumed to be anisotropic with a larger conductivity $\sigma_\mathrm{i}^{\parallel}$ parallel to the muscle fiber and a smaller conductivity $\sigma_\mathrm{i}^{\perp}$ perpendicular to the muscle fibers, see [14].

B.2. Simulation parameters in Section 5.2

The muscle contraction scenario is described by the following parameters:

	Fibers	MUs	Nodes per fiber	3D mesh nodes Ω_M	dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	dt_{3D} [ms]	ρ_0 [kg cm^{-3}]	s^{active}_{max} [N cm^{-2}]
Fig. 8	$13^2 = 169$	15	1481	1295	5×10^{-5}	10^{-4}	10^{-4}	1	10^{-3}	7.3

Subcellular model. The simulation uses the subcellular model of Shorten et al. [16].

[8] Courtesy of the U.S. National Library of Medicine: https://www.nlm.nih.gov/research/visible/visible_human.html.

Fiber parameters. The surface-to-volume-ratio A_m of the fibers is chosen differently for different MUs, analogously to the work of [12]. We compute $A_m = 2/r$ from an idealized fiber radius r and vary r between 40 µm and 55 µm with an exponential progression from the smallest MU to the largest MU. We choose the smallest 7 MUs to contain slow-twitch fibers with $C_m = 0.58\,\mu F\,cm^{-2}$ and the largest 8 MUs to contain fast-twitch fibers with $C_m = 1\,\mu F\,cm^{-2}$

Activation. The 15 MUs are activated in a ramp pattern in the first 1.4 s starting with the smallest MU at $t = 0$ ms and adding the next larger MU every 100 ms. Once activated, each MU fires with a particular base frequency, which is exponentially progression from 24 Hz for the smallest MU down to 7 Hz for the largest MU. The actual firing times vary by a 10% uniformly random jitter value from the base frequencies, see [12].

The neuromuscular junctions of the fibers, where the fibers are stimulated, are distributed uniformly within the central 10% of the fiber length.

Boundary conditions of the solid mechanics problem. The muscle is fixed at its bottom (statically determined) by proper Dirichlet boundary conditions on the displacements. A gravitational body force of $\mathbf{b} = (0, 0, -9.81 \times 10^{-4}\,cm\,ms^{-2})^T$ acts on the muscle.

B.3. Simulation parameters in Section 5.3

Section 5.3 contains two simulation scenario of the fiber-based electrophysiology model in Figs. 9 and 10. They use the same numerical parameters:

dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	dt_{3D} [ms]
1.25×10^{-3}	6.25×10^{-4}	2.5×10^{-3}	5.0×10^{-1}

The first scenario is described by the following discretization parameters:

	Fibers	MUs	Nodes per fiber	3D mesh nodes Ω_M	3D mesh nodes Ω_B
Fig. 9	$31^2 = 961$	20	1481	18 944	6882

For the second scenario, visualized in Fig. 10, Table 1 provides information about the discretization and partitioning.

Subcellular model. Both scenarios use the subcellular model of Hodgkin and Huxley [51].

Fiber parameters. The parameters A_m and C_m are chosen as described in Appendix B.2. For the first scenario with 20 MUs shown in Fig. 9, the 15 smallest MUs consist of slow-twitch fibers and the 5 largest MUs consist of fast-twitch fibers. For the scenarios with 100 MUs shown in Fig. 10, there are 71 MUs with slow-twitch fibers and 29 MUs with fast-twitch fibers.

Activation. In all scenarios, the MUs are activated in a ramp pattern as described in Appendix B.2. For the scenario with 20 MUs, the next MU is activated every 100 ms such that all MUs are active for $t \geq 1.9$ s. For the scenarios with 100 MUs, the next MU is activated every 2 ms such that all MUs are active for $t \geq 198$ ms.

B.4. Simulation parameters in Section 5.4

The multi-domain simulation scenario is described by the following parameters:

	MUs	3D mesh nodes Ω_M	3D mesh nodes Ω_B	dt_{0D} [ms]	dt_{3D} [ms]	$dt_{splitting}$ [ms]
Fig. 11	25	12 675	9375	5×10^{-4}	10^{-3}	10^{-3}

Subcellular model. The scenario uses the subcellular model of Hodgkin and Huxley [51].

Electrophysiology parameters. The parameters A_m and C_m are chosen similarly as in the previous scenarios. The A_m parameter follows again an exponential progression between the smallest and largest MU. The 18 smallest MUs consist of slow-twitch fibers and the 7 largest MUs consist of fast-twitch fibers and use according values for the C_m parameter.

Activation. The 25 MUs are all activated from the beginning, with the first firing time being uniformly random chosen in the first 20 ms.

B.5. Simulation parameters in Section 5.5

The discretization parameters for the scenario with neuronal feedback are given below:

	Fibers	MUs	Nodes per fiber	3D mesh[1] nodes Ω_M	3D mesh[1] nodes Ω_B	3D mesh[2] nodes Ω_M	3D mesh[2] nodes Ω_B
Fig. 13	$9^2 = 81$	15	1481	2349	2465	441	351

The 3D meshes marked with [1] and [2] are used for the electrophysiology and solid mechanics models, respectively.

The following time step widths are used:

	dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	$dt_{3D,EMG}$ [ms]	$dt_{3D,elasticity}$ [ms]	$dt_{motor\text{-}neuron}$ [ms]	$dt_{spindles}$ [ms]
Fig. 13	5×10^{-4}	10^{-3}	10^{-3}	10^{-3}	10^{-1}	10^{-2}	10^{-3}

For the motor neuron model described by the authors of [18], we use the CellML model file published as complementary material to the publication [8] with the following model parameters:

	C_m [µF cm^{-2}]	R_i [Ω cm]	r_d [cm]	l_d [cm]	r_s [cm]	l_s [cm]	R_{ms} [kΩ cm^{-2}]	R_{md} [kΩ cm^{-2}]
First	1	0.07	20.75×10^{-4}	0.55	38.75×10^{-4}	77.5×10^{-4}	1.15	14.4
Last			46.25×10^{-4}	1.06	56.5×10^{-4}	113×10^{-4}	0.65	6.05

For the parameters p where "first" and "last" values are given, the value varies among the $N_{MU} = 10$ motor neurons. The parameter value for motor neuron k is given by

$$p(k) = p_{first} + 100^{\frac{k-1}{N_{MU}-1}-1} \cdot (p_{last} - p_{first}), \quad k \in \{1, \ldots, N_{MU}\},$$

i.e., the value is approximately p_{first} for the first motor neuron, p_{last} for the last motor neuron, and has an exponential progression with base 100 over the range of motor neurons.

Subcellular model. The scenario uses a combination of the subcellular models of Hodgkin and Huxley [51] and Razumova [64].

B.6. Simulation parameters in Section 5.6

The scenario to evaluate the node level performance uses the same parameters in both the *OpenDiHu* and the *OpenCMISS Iron* implementations:

	Fibers	MUs	Nodes per fiber	3D mesh nodes	dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	dt_{3D} [ms]
Fig. 14	$9^2 = 81$	10	1481	775	8.33×10^{-5}	5×10^{-4}	5×10^{-4}	10^{-1}

Subcellular model. The scenario uses the subcellular model of Shorten et al. [16].

Activation. The MUs are activated in a ramp pattern as described in Appendix B.2. The next MU is activated every 0.2 ms such that all MUs are active for $t \geq 1.8$ ms.

B.7. Simulation parameters in Section 5.7

The weak scaling study uses the following parameters and time step widths:

	MUs	Nodes per fiber	dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	dt_{3D} [ms]
Fig. 15	10	1481	10^{-3}	2×10^{-3}	2×10^{-3}	10^{-1}

The discretization parameters for the weak scaling study in Fig. 15 are chosen as follows:

Fibers	3D mesh nodes Ω_M	Total dofs	Processes	Compute nodes	Fibers per process	CG iterations
$13^2 = 169$	1984	1 003 140	18	1	9.39	72
$25^2 = 625$	6468	3 708 968	64	1	9.77	115
$37^2 = 1369$	13 200	8 123 156	144	3	9.51	176
$67^2 = 4489$	41 580	26 634 416	448	7	10.02	339
$109^2 = 11881$	118 800	70 501 844	1152	18	10.31	561
$187^2 = 34969$	316 932	207 473 288	3600	57	9.71	1056
$277^2 = 76729$	703 428	455 246 024	7744	121	9.91	1636
$523^2 = 273529$	2 282 577	1 622 668 373	26 912	421	10.16	2807

Subcellular model. The scenario uses the subcellular model of Shorten et al. [16].

B.8. Simulation parameters in Section 5.8

The material and activation parameters for the multi-domain study are the same as in Appendix B.4.

	3D mesh nodes Ω_M	3D mesh nodes Ω_B	dt_{0D} [ms]	dt_{3D} [ms]	$dt_{splitting}$ [ms]	dt_{end} [ms]
Fig. 16	50 024	37 000	2.5×10^{-4}	5×10^{-4}	5×10^{-4}	10^{-1}

The discretization parameters for the study are chosen as follows:

MUs	Processes	System matrix size
4	24	$279\,720^2$
6	36	$379\,768^2$
8	48	$479\,816^2$
10	60	$579\,864^2$
12	72	$679\,912^2$

Subcellular model. The scenario uses the subcellular model of Shorten et al. [16].

B.9. Simulation parameters in Section 5.9

The parameters for the mesh resolution studies are chosen as follows:

	Fibers	Nodes per fiber	3D mesh elements Ω_M	3D mesh elements Ω_B	dt_{0D} [ms]	dt_{1D} [ms]	$dt_{splitting}$ [ms]	dt_{3D} [ms]	A_m [cm^{-1}]
Fig. 17	7^2–97^2	1481	7^2–$97^2\times31$	7–$97\times5\times31$	3×10^{-3}	1.5×10^{-3}	3×10^{-3}	1	500

The neuromuscular junctions of the fibers are located at the exact center for every muscle fiber.

References

[1] A.A. Cuellar, C.M. Lloyd, P.F. Nielsen, D.P. Bullivant, D.P. Nickerson, P.J. Hunter, An overview of CellML 1.1, a biological model description language, Simulation 79 (12) (2003) 740–747.

[2] C. Bradley, A. Bowery, R. Britten, V. Budelmann, O. Camara, R. Christie, A. Cookson, A.F. Frangi, T.B. Gamage, T. Heidlauf, S. Krittian, D. Ladd, C. Little, K. Mithraratne, M. Nash, D. Nickerson, P. Nielsen, Ø. Nordbø, S. Omholt, A. Pashaei, D. Paterson, V. Rajagopal, A. Reeve, O. Röhrle, S. Safaei, R. Sebastián, M. Steghöfer, T. Wu, T. Yu, H. Zhang, P. Hunter, OpenCMISS: A multi-physics & multi-scale computational infrastructure for the VPH/Physiome project, Prog. Biophys. Mol. Biol. 107 (1) (2011) 32–47, Experimental and Computational Model Interactions in Bio-Research: State of the Art.

[3] G.R. Mirams, C.J. Arthurs, M.O. Bernabeu, R. Bordas, J. Cooper, A. Corrias, Y. Davit, S.-J. Dunn, A.G. Fletcher, D.G. Harvey, M.E. Marsh, J.M. Osborne, P. Pathmanathan, J. Pitt-Francis, J. Southern, N. Zemzemi, D.J. Gavaghan, Chaste: An open source C++ library for computational physiology and biology, PLOS Comput. Biol. 9 (3) (2013) 1–8.

[4] J. Sánchez, M. Nothstein, A. Neic, Y.-L. Huang, A.J. Prassl, J. Klar, R. Ulrich, F. Bach, P. Zschumme, M. Selzer, et al., openCARP: An open sustainable framework for in-silico cardiac electrophysiology research, in: 2020 Computing in Cardiology, IEEE, 2020, pp. 1–4.

[5] M.O. Bernabeu, M.J. Bishop, J. Pitt-Francis, D.J. Gavaghan, V. Grau, B. Rodriguez, High performance computer simulations for the study of biological function in 3D heart models incorporating fibre orientation and realistic geometry at para-cellular resolution, in: 2008 Computers in Cardiology, 2008, pp. 721–724.

[6] A. Gerbi, L. Dedè, A. Quarteroni, A monolithic algorithm for the simulation of cardiac electromechanics in the human left ventricle, Math. Eng. 1 (1) (2019) 1–37.

[7] P. Lafortune, R. Arís, M. Vázquez, G. Houzeaux, Coupled electromechanical model of the heart: Parallel finite element formulation, Int. J. Numer. Methods Biomed. Eng. 28 (1) (2012) 72–86.

[8] O. Röhrle, U.Ş. Yavuz, T. Klotz, F. Negro, T. Heidlauf, Multiscale modeling of the neuromuscular system: Coupling neurophysiology and skeletal muscle mechanics, WIREs Syst. Biol. Med. 11 (6) (2019) e1457, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/wsbm.1457.

[9] T. Heidlauf, O. Röhrle, A multiscale chemo-electro-mechanical skeletal muscle model to analyze muscle contraction and force generation for different muscle fiber arrangements, Front. Physiol. 5 (498) (2014) 1–14.

[10] T. Heidlauf, O. Röhrle, Modeling the chemoelectromechanical behavior of skeletal muscle using the parallel open-source software library opencmiss, Comput. Math. Methods Med. 2013 (2013) 1–14.

[11] T. Heidlauf, O. Röhrle, A multiscale chemo-electro-mechanical skeletal muscle model to analyze muscle contraction and force generation for different muscle fiber arrangements, Front. Physiol. 5 (498) (2014) 1–14.

[12] T. Klotz, L. Gizzi, U.Ş. Yavuz, O. Röhrle, Modelling the electrical activity of skeletal muscle tissue using a multi-domain approach, Biomech. Model. Mechanobiol. (2020).

[13] T. Klotz, L. Gizzi, O. Röhrle, Investigating the spatial resolution of EMG and MMG based on a systemic multi-scale model, Biomech. Model. Mechanobiol. 21 (3) (2022) 983–997.

[14] M. Mordhorst, T. Heidlauf, O. Röhrle, Predicting electromyographic signals under realistic conditions using a multiscale chemo-electro-mechanical finite element model, Interface Focus 5 (2) (2015) 1–11.

[15] IUPS Physiome Project, Physiome model repository, 2020, https://models.physiomeproject.org, [Online; accessed August 8, 2021].

[16] P.R. Shorten, P. O'Callaghan, J.B. Davidson, T.K. Soboleva, A mathematical model of fatigue in skeletal muscle force contraction, J. Muscle Res. Cell Motil. 28 (6) (2007).

[17] B. Maier, Scalable Biophysical Simulations of the Neuromuscular System (Ph.D. thesis), University of Stuttgart, 2021.

[18] R.R. Cisi, A.F. Kohn, Simulation system of spinal cord motor nuclei and associated nerves and muscles, in a Web-based architecture, J. Comput. Neurosci. 25 (3) (2008) 520–542.

[19] F. Negro, D. Farina, Decorrelation of cortical inputs and motoneuron output, J. Neurophysiol. 106 (5) (2011) 2688–2697.

[20] M.P. Mileusnic, I.E. Brown, N. Lan, G.E. Loeb, Mathematical models of proprioceptors. I. Control and transduction in the muscle spindle, J. Neurophysiol. 96 (4) (2006) 1772–1788.

[21] D. Hysom, A. Pothen, A scalable parallel algorithm for incomplete factor preconditioning, SIAM J. Sci. Comput. 22 (6) (2001) 2194–2215.

[22] R.D. Falgout, U.M. Yang, Hypre: A library of high performance preconditioners, in: Proceedings of the International Conference on Computational Science-Part III, ICCS '02, Springer, Berlin, Heidelberg, 2002, pp. 632–641.

[23] M. Brandstater, E. Lambert, A histochemical study of the spatial arrangement of muscle fibers in single motor units within rat tibialis anterior muscle, Bull. Am. Assoc. Electromyogr. Electrodiagnosis 82 (1969) 15–16.

[24] R.M. Enoka, A.J. Fuglevand, Motor unit physiology: Some unresolved issues, Muscle Nerve 24 (1) (2001) 4–17.

[25] O. Röhrle, J.B. Davidson, A.J. Pullan, A physiologically based, multi-scale model of skeletal muscle structure and function, Front. Physiol. 3 (2012).

[26] H. Saini, E. Altan, E. Ramasamy, T. Klotz, L. Gizzi, O. Röhrle, Predicting skeletal muscle force from motor-unit activity using a 3D FE model, Proc. Appl. Math. Mech. 18 (1) (2018) e201800035.

[27] H. Weyl, Über die Gleichverteilung von Zahlen mod. Eins, Math. Ann. 77 (3) (1916) 313–352.

[28] Message Passing Interface Forum, MPI: A message-passing interface standard, version 3.1, Specification, High Performance Computing Center Stuttgart (HLRS), 2015, URL https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf.

[29] S. Balay, W.D. Gropp, L.C. McInnes, B.F. Smith, Efficient management of parallelism in object oriented numerical software libraries, in: E. Arge, A.M. Bruaset, H.P. Langtangen (Eds.), Modern Software Tools in Scientific Computing, Birkhäuser Press, 1997, pp. 163–202.

[30] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc web page, 2018, http://www.mcs.anl.gov/petsc.
[31] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W.D. Gropp, D. Karpeyev, D. Kaushik, M.G. Knepley, D.A. May, L.C. McInnes, R.T. Mills, T. Munson, K. Rupp, P. Sanan, B.F. Smith, S. Zampini, H. Zhang, H. Zhang, PETSc users manual, Technical Report ANL-95/11 - Revision 3.14, Argonne National Laboratory, 2020, URL https://www.mcs.anl.gov/petsc.
[32] P. Amestoy, A. Buttari, J.-Y. L'Excellent, T. Mary, On the complexity of the block low-rank multifrontal factorization, SIAM J. Sci. Comput. 39 (4) (2017) 1710–1740.
[33] M. Kretz, V. Lindenstruth, Vc: A C++ library for explicit vectorization, Softw.: Pract. Exp. 42 (11) (2012) 1409–1430.
[34] M. Kretz, Extending C++ for explicit data-parallel programming via SIMD vector types (Ph.D. thesis), Univ.-Bibliothek, Frankfurt am Main, 2015, URL https://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/38415.
[35] J. Hoberock, Working draft, C++ extensions for parallelism version 2, Technical Report N4808, International Organization for Standardization (ISO), International Electrotechnical Commission (IEC), 2019, URL http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/n4808.pdf.
[36] Z. Gutterman, Symbolic Pre-Computation for Numerical Applications, Technion-Israel Institute of Technology, Faculty of Computer Science, 2004.
[37] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, B. Uekermann, preCICE – a fully parallel library for multi-physics surface coupling, Comput. & Fluids 141 (2016) 250–258, Advances in Fluid-Structure Interaction.
[38] B. Maier, N. Emamy, A. Krämer, M. Mehl, Highly parallel multi-physics simulation of muscular activation and EMG, in: Coupled Problems 2019, VIII International Conference on Coupled Problems in Science and Engineering, 2019, pp. 610–621.
[39] C.P. Bradley, N. Emamy, T. Ertl, D. Göddeke, A. Hessenthaler, T. Klotz, A. Krämer, M. Krone, B. Maier, M. Mehl, T. Rau, O. Röhrle, Enabling detailed, biophysics-based skeletal muscle models on HPC systems, Front. Physiol. 9 (2018) 816.
[40] OpenMP Architecture Review Board, OpenMP application program interface version 4.5, 2015, URL https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf.
[41] A.G. Holzapfel, Nonlinear Solid Mechanicss: A Continuum Approach for Engineering Science, John Wiley & Sons, Chichester, 2000.
[42] J. Ahrens, B. Geveci, C. Law, Paraview: An end-user tool for large data visualization, Vis. Handb. 717 (8) (2005).
[43] W.F. Godoy, N. Podhorszki, R. Wang, C. Atkins, G. Eisenhauer, J. Gu, P. Davis, J. Choi, K. Germaschewski, K. Huck, A. Huebl, M. Kim, J. Kress, T. Kurc, Q. Liu, J. Logan, K. Mehta, G. Ostrouchov, M. Parashar, F. Poeschel, D. Pugmire, E. Suchyta, K. Takahashi, N. Thompson, S. Tsutsumi, L. Wan, M. Wolf, K. Wu, S. Klasky, ADIOS 2: The adaptable input output system. a framework for high-performance data management, SoftwareX 12 (2020) 100561.
[44] T. Rau, M. Krone, G. Reina, T. Ertl, Challenges and opportunities using Software-Defined visualization in MegaMol, in: N. Ferreira, L.G. Nonato, F. Sadlo (Eds.), Workshop on Visual Analytics, Information Visualization and Scientific Visualization (WVIS) in the 30th Conference on Graphics, Patterns and Images, SIBGRAPI'17, Niterói, RJ, Brazil, 2017, URL http://sibgrapi2017.ic.uff.br/.
[45] P. Gralka, M. Becher, M. Braun, F. Frieß, C. Müller, T. Rau, K. Schatz, C. Schulz, M. Krone, G. Reina, T. Ertl, MegaMol – a comprehensive prototyping framework for visualizations, Eur. Phys. J. Spec. Top. 227 (14) (2019) 1817–1829.
[46] The cmgui EX format guide: exnode and exelem files, 2018, URL http://opencmiss.org/documentation/apidoc/zinc/docs/CMGUI_ex_fileFormatGuide.html, (Accessed: 2021-02-12).
[47] E. Stauffer, J. Stephens, Responses of golgi tendon organs to ramp-and-hold profiles of contractile force, J. Neurophysiol. 40 (3) (1977) 681–691.
[48] L. De-Doncker, F. Picquet, J. Petit, M. Falempin, Characterization of spindle afferents in rat soleus muscle using ramp-and-hold and sinusoidal stretches, J. Neurophysiol. 89 (1) (2003) 442–449, PMID: 12522192.
[49] S. Williams, A. Waterman, D. Patterson, Roofline: An insightful visual performance model for multicore architectures, Commun. ACM 52 (4) (2009) 65–76.
[50] C. Yang, Empirical roofline tool (ERT), 2020, URL https://crd.lbl.gov/departments/computer-science/par/research/roofline/software/ert/.
[51] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (4) (1952) 500–544.
[52] A. Krämer, B. Maier, T. Rau, F. Huber, T. Klotz, T. Ertl, D. Göddeke, M. Mehl, G. Reina, O. Röhrle, Multi-physics multi-scale HPC simulations of skeletal muscles, in: W.E. Nagel, D.H. Kröner, M.M. Resch (Eds.), High Performance Computing in Science and Engineering ' 20: Transactions of the High Performance Computing Center, Stuttgart (HLRS) 2020, 2021.
[53] M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm, K. Gullapalli, State-of-the-art sparse direct solvers, in: A. Grama, A.H. Sameh (Eds.), Parallel Algorithms in Computational Science and Engineering, Springer International Publishing, 2020, pp. 3–33.
[54] T. Klotz, L. Lehmann, F. Negro, O. Röhrle, High-density magnetomyography is superior to high-density surface electromyography for motor unit decomposition: A simulation study, J. Neural Eng. 20 (4) (2023) 046022.
[55] A.H. Caillet, S. Avrillon, A. Kundu, T. Yu, A.T. Phillips, L. Modenese, D. Farina, Larger and denser: an optimal design for surface grids of emg electrodes to identify greater and more representative samples of motor units, eNeuro (2023).
[56] E. Ramasamy, O. Avci, B. Dorow, S.-Y. Chong, L. Gizzi, G. Steidle, F. Schick, O. Röhrle, An efficient modelling-simulation-analysis workflow to investigate stump-socket interaction using patient-specific, three-dimensional, continuum-mechanical, finite element residual limb models, Front. Bioeng. Biotechnol. 6 (2018) 126.
[57] K. Maksymenko, A.K. Clarke, I. Mendez Guerra, S. Deslauriers-Gauthier, D. Farina, A myoelectric digital twin for fast and realistic modelling in deep learning, Nature Commun. 14 (1) (2023) 1600.
[58] A. Rörich, T.A. Werthmann, D. Göddeke, L. Grasedyck, Bayesian inversion for electromyography using low-rank tensor formats, Inverse Problems 37 (5) (2021) 055003.
[59] R. Oostenveld, P. Fries, E. Maris, J.-M. Schoffelen, FieldTrip: open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data, Comput. Intell. Neurosci. 2011 (2011).
[60] A. Gramfort, M. Luessi, E. Larson, D.A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, et al., MEG and EEG data analysis with MNE-python, Front. Neurosci. (2013) 267.
[61] A.H. Caillet, A.T. Phillips, D. Farina, L. Modenese, Estimation of the firing behaviour of a complete motoneuron pool by combining electromyography signal decomposition and realistic motoneuron modelling, PLOS Comput. Biol. 18 (9) (2022) e1010556.
[62] O.U. Khurram, G.E. Pearcey, M.K. Chardon, E.H. Kim, M. García, C. Heckman, The cellular basis for the generation of firing patterns in human motor units, in: Vertebrate Motoneurons, Springer, 2022, pp. 233–258.
[63] V. Spitzer, M.J. Ackerman, A.L. Scherzinger, D. Whitlock, The visible human male: A technical report, J. Am. Med. Inform. Assoc. 3 (2) (1996) 118–130.
[64] M.V. Razumova, A.E. Bukatina, K.B. Campbell, Stiffness-distortion sarcomere model for muscle simulation, J. Appl. Physiol. 87 (5) (1999) 1861–1876.

Benjamin Maier studied Computer Science and Simulation Technology at the University of Stuttgart. In 2022, he finished his doctoral studies in the International Research Training Group "Soft Tissue Robotics" on the topic of scalable biophysical simulations of the neuromuscular system. Since then, he works in applied research in the industry.

Dominik Göddeke is a full professor of Computational Mathematics at the University of Stuttgart, Germany, and a fellow of the Stuttgart Centre for Simulation Science (SimTech). His research interests cover hardware-oriented numerics in computational science and engineering, including parallelization, GPU Computing, large-scale computations, multigrid methods and domain decomposition; as well as the application to continuum-mechanical, geophysical and biomechanical forward and inverse problems. Prior to his position in Stuttgart, he obtained his Ph.D. from TU Dortmund (Germany) in applied mathematics, and was a Junior Professor for hardware-oriented numerics in Dortmund.

Felix Huber received his B.Sc. and M.Sc. degrees in Simulation Technology from the University of Stuttgart, Germany, in 2016 and 2019. Currently he is a Ph.D. student in the SimTech Cluster of Excellence "Data-Integrated Simulation Science" (EXC2075) at the Institute of Applied Analysis and Numerical Simulation and the Institute for Parallel and Distributed Systems at the University of Stuttgart. His research focuses on Bayesian optimization for constrained problems in a simulation context.

Thomas Klotz did his Ph.D. in biomedical engineering. Currently, he is a post-doctoral researcher at the Institute for Modelling and Simulation of Biomechanical Systems, University of Stuttgart, Germany. His research interest includes neuromuscular physiology and human motion, mathematical modeling of skeletal muscles, magnetomyography as well as bio-signal processing.

Oliver Röhrle is professor for ‘‘Continuum Biomechanics and Mechanobiology’’ and the founding director of the Institute for Modelling and Simulation of Biomechanical Systems at the University of Stuttgart, Germany. He is also a fellow of the Stuttgart Center for Simulation Technology (SimTech) and a senior research expert at Fraunhofer IPA. He received his Ph.D. in Applied Mathematics from the University of Colorado at Boulder, USA (2004) before spending 4 years as a research scientist at the Auckland Bioengineering Institute at the University of Auckland, New Zealand.

Miriam Schulte is full professor and member of the board of directors of the cluster of excellence EXC2075 Data-Integrated Simulation Science at the University of Stuttgart since 2013. Her main research interests are in the field between numerical mathematics and high performance computing with a particular focus on coupled problems involving different types of equations and other components such as inverse solvers. Miriam Schulte studied mathematics at the Technical University of Munich (Diploma 1997) and did her doctoral studies at the Department of Informatics of the Technical University of Munich (Dr. rer. nat. 2001).